Université Joseph Fourier de Grenoble (UJF)

Extraction of Three-dimensional Information from Images
Application to Computer Graphics

Extraction d'informations tridimensionnelles à partir d'images
Application à l'informatique graphique

Sylvain PARIS

Thèse présentée pour l'obtention du titre de
Docteur de l'Université Joseph Fourier
Spécialité Informatique
Arrêté Ministériel du 5 juillet 1984 et du 30 mars 1992
Préparée au sein du laboratoire
ARTIS-GRAVIR/IMAG-INRIA. UMR CNRS C5527.

Composition du jury :

François	SILLION	Directeur de thèse
Georges-Pierre	BONNEAU	Président du Jury
Wolfgang	HEIDRICH	Rapporteur
Bernard	PÉROCHE	Rapporteur
Long	QUAN	Examinateur

Contents

Que soient ici remerciées toutes les personnes qui ont contribué à cette thèse.

Je n'en serais pas là sans l'éducation que ma famille m'a offerte depuis le début. Bien évidemment, il n'y aurait pas eu de thèse non plus sans François dont les qualités font largement oublier le fait qu'il n'ait pas de stylo. C'est une vraie chance d'avoir pour directeur de thèse quelqu'un qui m'a encadré sans m'enfermer, qui m'a laissé libre de mes choix sans m'abandonner. Merci à tous mes collègues du labo pour la richesse de leur environnement et pour leurs innombrables coups de main. Je tiens à remercier plus particulièrement Samuel, Stéphane et Sylvain: ce fut un réel plaisir de traverser ces trois ans de thèse en même temps qu'eux; et aussi Gilles, Joëlle et Xavier qui m'ont vu plus que régulièrement débouler dans leur bureau et qui ont toujours pris le temps de me dépanner et surtout de m'apprendre les us et coutumes du “milieu”. Une pensée aussi à tous ceux qui m'ont accueilli chaleureusement à Hong Kong et plus particulièrement Long. Merci évidemment à la DGA pour avoir financé mes travaux, et spécialement aux membres de la commission de suivi pour leurs conseils avisés. Je suis très fier d'avoir eu Wolfgang, Long, Bernard, Georges-Pierre et François dans mon jury pour conclure cette thèse. Je suis tout particulièrement reconnaissant envers Wolfgang et Bernard pour leur travail sur mon manuscrit, et envers Wolfgang et Long pour s'être déplacés depuis Vancouver et Hong Kong pour ma soutenance. Enfin, merci à mes amis, ceux qui m'ont rendu la vie à Grenoble agréable et ceux qui ont gardé le contact même si j'étais loin.

J'ai choisi de ne pas faire une liste de noms interminable car j'en aurais forcément oubliés. Mais si vous m'avez aidé à être là, si nous avons partagé une discussion, un problème, un bug, une deadline, un thé, une bière, une rando, une soirée, une galère, un fou-rire... ces remerciements vous sont destinés.

1
Introduction

Ce manuscrit est en anglais, un résumé français est proposé en annexe C, page 181.

This dissertation is in English, a French summary can be found in Appendix C on page 181

This dissertation focuses on the creation of the information used in Computer Graphics. We especially concentrate on the data needed to render images. These data can be roughly categorized into three types: the shape of the objects, their appearance and their lighting environment. We are mainly interested in the creation of the first two data types (the object shape and appearance), even if we regularly deal with the light. This thesis proposes several methods to generate these data using real images: We do not ask the user to work directly on the object modeling but we rely on her to provide one or several images of the targeted object. These pictures are then automatically analyzed by the computer that extracts the sought data.

We expect from this approach data more faithful to the original object and a shorter creation time. Let's go in deeper details with a practical example.

Consider a simple Computer Graphics task, let's say producing the data needed to render an image of a teapot, the basic approach is to rely on human skills: The teapot shape is designed in a software dedicated to 3D modeling and a porcelain appearance is defined through dialog boxes that let the user choose the characteristics of the porcelain (color, shininess, patterns, etc). This modeling process is especially time-consuming since the user is in charge of creating everything. The "quality" of the result also depends on the user proficiency. A faster approach is to select these data from a library of previously created shapes and materials. It eases the current task but it does not address the issue of the creation the original data within the library.

Once these data have been created, there are several well-known methods to render the teapot image. This rendering step is out of the scope of this dissertation.

Modeling an existing object

But now, consider a trickier case. One does not want *a* teapot anymore but *the* teapot she uses every day. A direct approach is to first create a standard teapot and then to work from it to match the original teapot. Producing an approximate match is feasible. But imagine that the teapot is decorated with some carved patterns and that there are some golden paintings over the base material. In that case, an accurate match is almost intractable for the user. One has to use techniques more sophisticated than user-driven modeling.

To capture the shape of an object, one can use a 3D scanner: It is an apparatus with a laser or a beam of modulated light that measures the 3D shape of an object. The acquisition of the material can also be done using a gonioreflectometer [221]: The behavior of the material with the light is measured for all possible light positions and view directions. Both techniques are accurate but require specific equipments and configuration. This often hinders the use of these techniques. In addition, several cases foil down these measures. For instance, 3D scanners suffer from translucent and dispersive materials (*e.g.* glass, fur, hair).

Our proposal

In this dissertation, we present an alternative approach based on images. For the teapot problem, we propose to use one or several photographs of the original teapot to extract the data needed to reproduce it. Then, an analysis is performed to produce the useful data from these images. Compared to a user-driven approach, we expect several improvements:

Shorter user time: Obviously, an image-based algorithm requires less user time. It may last longer because of numerical computation. However, during this step, the user can work on other tasks.

Objective characterization of the data: If one asks several persons to determine the shininess of the same teapot, the answers are likely to be different. The different "values" result from different subjective evaluations and it would be hard to choose the right one. On the other side, an algorithm has a deterministic criterion whose accuracy can be studied. In addition, the measure is reproducible.

More details: When it comes to reproduce an existing object, the user may miss some details and features whereas an automatic reconstruction performs an exhaustive scan of the images.

However, it does not mean that we aim at replacing the user-driven creation. We propose a complementary way to produce 3D data. Our approach is compliant with the user-driven and library-based creation process: The shape stemming from the images can be further edited by the user and/or stored in a shape library for later reuse.

Robustness

Unfortunately, we believe that perturbation is inherent in the capture process. The images are inevitably corrupted by some noise, blur, distortion, etc. The question is how we handle that fact. There are two extreme solutions: On the one hand, we can allow the user to shoot images without any constraints. This implies that the effort must be done during the analysis step. The underlying algorithm must be robust to be able to extract satisfying data. The counterpart is an easy acquisition process accessible to a non-specialist user. On the other hand, the capture setup can be totally constrained (*e.g.* dark room, quality and calibrated optical lens, professional camera, robotic gantry, etc). This requires a specialist to drive the process. But the input data can be considered "perfect". It allows the algorithm to almost ignore the robustness issue and to focus on the precision. Some researchers have described intermediate solutions that impose a limited set of constraints to the user in order to extract more accurate data. This defines a continuous trade-off between the ease of acquisition and accuracy.

In this dissertation, we have deliberately chosen an approach that is more oriented toward the ease of acquisition. This does not mean that accuracy is neglected. It simply implies that we strive to design robust algorithms: We expect them to cope with potentially perturbed input in order to alleviate the requirements on the user side. This also implies that the accuracy may not be always comparable to what can be extracted from a highly controlled environment. Nonetheless, we expect the following advantages:

A less cumbersome acquisition process: The input data come from a classical digital camera or from a movie camera. Nowadays, quality cameras of small size are commonly available. In addition, if we need a dark room, we strive to be robust enough to work with a common room with only the lights turned off. We will not require a room with black matte walls and a blue screen.

A more flexible acquisition system: From the same images, we can use different analyses depending on the observed content whereas dedicated apparatuses have to be entirely changed when they do not fit with the targeted object.

A better correspondence with input images: In some cases, we may be able to work with an arbitrary background. This makes possible to work with images of an object within its context. If the task is to modify the original pictures (*e.g.* modify the appearance of an object, insert an additional object), the data obtained directly from such images are likely to be more consistent than the ones stemming from a technique that separates the object and its environment (*e.g.* they may suffer from misalignment).

A better balance of the extracted details: Acquisition from images is more likely to capture the visually important features whereas other techniques (*e.g.* laser scanners) may recover unnoticeable details and miss some others that may have a small size but a large visual impact.

Our approach: case study

Then comes a crucial question: What do we aim for? What kind of information do we extract? We have a personal conviction that the general case is not tractable: Robustly acquiring the geometry and the appearance of an object without any a priori knowledge cannot be done. We believe that a reasonable approach is to focus on a typical scenario for which we can rely on some known characteristics. One can then imagine to develop various scenarii and let the user select the appropriate algorithm. The caveat would be to multiply the number of scenarii but as long as we study scenarii of broad interest, we are convinced that this approach is valid and efficient.

Therefore, we have chosen this approach in this dissertation. We identify a few useful cases leading to interesting applications and concentrate on them. We address three main issues. We first present in Chapter 2 a method to recover the surface of a matte object from a short image sequence whose viewpoint is moving. This approach is extended to a more general set of images in Chapter 3. This technique is designed to be generic *i.e.* we avoid specific assumptions to handle as general objects as possible.

We then show in Chapter 4 how the appearance of a human face can be recovered from a single image and how the extracted data can be used to render the original face under a new lighting environment. We end this document with Chapter 5 that exposes a technique to capture the hair geometry using multiple images from a fixed viewpoint and a moving light. These last two cases (face and hair) target specific entities that are really important characteristic features of a person. High accuracy is mandatory to provide a visual match that makes the original person recognizable. We therefore develop dedicated tools to reach a precision higher than user-based and generic techniques. General conclusions are given in the last chapter.

2
Surface reconstruction

The work presented in this chapter has been done in collaboration with Long Quan from Hong Kong University of Science and Technology. He mainly participates in stating the problem and in positioning it relatively to existing approaches. In addition, his numerous comments and pieces of advice have been a great contribution to the work presented in the following sections.

2.1 Introduction

In this chapter, we focus on perhaps the most intuitive use of several images: geometry reconstruction. The starting remark is that, with our two eyes, we have a sharp and efficient perception of the 3D world. This is an intuitive and practical proof that two points of view on the same scene are enough to "reconstruct" its geometry since our brain does it permanently. This leads to our task: Reproducing the geometry-from-images reconstruction on a computer would provide a useful tool to create virtual 3D objects.

Our initial motivation to this study was to get an efficient system to populate virtual environments. Currently, state-of-the-art systems (such as MMR [5], iWalk [43] or GigaWalk [97]) are able to display very large 3D model such as a whole city. Then comes the issue of generating this huge data set within a reasonable amount of time. A pitfall would be to use the standard modeling tools to generate everything, from the building to the public benches. Clearly the modeling power of the user cannot scale up similarly to the display power over these last years. "Legions of modelers" would be required to generate a geometric data set such as a city with all the details (buildings, trees, post-boxes,...) in an acceptable period. One needs dedicated automated tools to leverage such a task. For instance, Wonka *et al.* [230] have designed an algorithm to procedurally create city buildings. From a coarse description of the building style, this method is able to create several models that respect this style. So, from a limited user input, it is possible to generate a large number of houses, towers, etc to "fill" a city. Our study follows the same idea of producing 3D models from a limited user interaction but the technique and actual targets are totally different.

We target a smaller scale; we aim at benches, post-boxes, passers-by, etc that populate the avenues. Modeling every single object would last days, if not months, even with a simplified geometry. Using a library of standard shapes makes numerous objects available. But if one to integrate a given real object in the scene, such a library does not help. Our idea to overcome those caveats is to exploit a sequence of images to capture the geometry of an object. Ideally, we want a system that captures the geometry of an object that is seen in a short video sequence. So the user would just have to "point" objects in the sequence and the computer is then in charge of providing suitable 3D models.

If we formalize a little bit more our goal, we would like to enjoy the following aspects:

- We want to be able to deal with real images of ordinary quality. We may not have access to a professional high-quality camera. Or we may want to exploit a set of images that we do not have taken by ourself. This implies that we are able to cope with non-perfect images *i.e.* that does not perfectly meet our theoretical assumptions.
- The object geometry has to convey convincing visual effects (parallax, occlusions, etc). This requires a high level of precision but we can tolerate a limited lack of accuracy as long as it does not impair the visual cues.
- We may not have control over the object environment so the process must be able to deal with unknown background (*e.g.* we cannot bring a blue screen behind the object) and with partially hidden objects (*e.g.* a street light may be in front of the object in some images).

In this chapter, we limit our study to matte surfaces and still scenes. Handling moving objects (*e.g.* a walking passer-by) and reconstructing arbitrary materials (*e.g.* reflective surfaces) would be interesting but we will consider these tasks in future work. There is however a "trick" for the movements: With a system that shoots several pictures simultaneously, we get a sequence of "frozen" images in which the moving objects are static.

Considering again the brain example, these goals may appear fairly easy. Before examining the details, we give two arguments to show it is not.

A priori knowledge: First, the brain does more than just analyzing two images to build the 3D representation of the scene. Consider for instance Figure 2.1 (on the right): Everyone sees a 3D cube although there is nothing in 3D in that picture. It is only three quadrilaterals side by side, there is not even a skew that would indicate some perspective distortion. Nonetheless, everyone does see a 3D entity. There is some part in the reconstruction process that is related to "recognition", "learning", "interpretation", etc. This explains why we recognize a 3D cube in a single 2D picture. But this psycho-cognitive part of reconstruction is far beyond the scope of our work. In this document, we will only consider a geometric approach to the reconstruction process which is based physical properties of the object or on approximations of them. This already opens a very broad range of research possibilities and leads to efficient computational solutions.

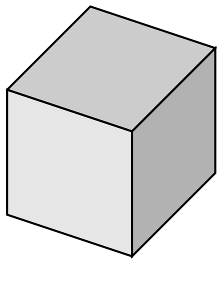


Fig. 2.1: This picture is not 3D.

Consistency is not enough: The second point to observe is purely geometric. The reconstruction problem is not well defined: Several different scenes can generate the same set of images. Let's give an extreme example to demonstrate this.

> ***Definition* [Consistency]*:*** *A reconstructed scene is said* consistent *if, using the same setup (cameras, lighting, etc), it produces the same pictures as the input images.*

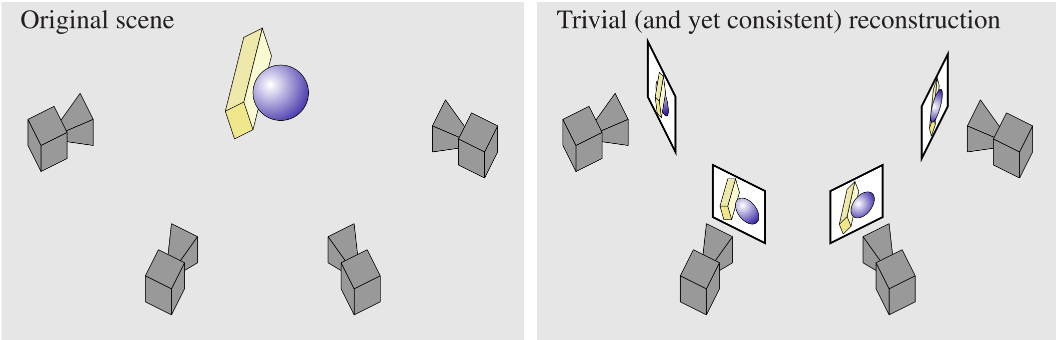


Fig. 2.2: Trivial reconstruction

For any scene, it is always possible to propose a reconstruction composed of a set of billboards facing each camera with the corresponding picture on it.

We here define an algorithm that always reconstructs a perfectly consistent geometry for any scene without any assumption. To achieve this miracle, it is sufficient to return a geometry composed of a set of billboards facing each camera. For each camera, its original image is placed on the corresponding billboard (see Figure 2.2). Now it is straightforward to show that this reconstruction made of the original pictures hanging in front of the cameras is exactly consistent. But obviously, this trivial algorithm does not fulfill our goal although its result is “perfect”. We expect a result as close as possible from the original geometry, we expect a result “more than consistent”. This point makes our study especially difficult because we have to formulate this a priori knowledge that characterizes the satisfactory reconstructions from the others.

This discussion shows that reaching our objectives might not be as easy as one may first think. In this chapter we present a method that brings several contributions to this research topic, mainly robustness and precision. Before describing our approach, we review the most relevant papers related to our work. And, since our technique involves the graph-cut tool, we expose its main properties in a general context. The end of the chapter is dedicated to the explanation of our method.

2.2 Previous work

With our two eyes, we are able to "see the 3D". This remark has originally inspired the research in stereoscopic vision: How can we recover some 3D shape from a pair of images? In this thesis, we focus on the natural extension to several images. Therefore, we do not review here all the methods stemming from the two-image problem. We only sketch the main linked ideas. This give an insight on some interesting issues before focusing on the methods dealing with more images.

The two-image case

From two images with different viewpoints, it is possible to compute the 3D position of a given point if its projections in each image are known. As seen in Figure 2.3-left, each projection defines a ray in space; the intersection of both rays is the 3D point. This is a constructive approach since two image points are matched and then, the 3D point is built from these two points. This raises the non-trivial question of the matching between two image points. This issue is discussed later in details.

This also requires that the cameras are *calibrated i.e.* that, for both images, sufficient information is known to determine the ray that contains all the 3D points which project onto a given 2D point. How to calibrate a camera is a research topic on its own. Our work assume this information to be given as input. We refer the reader to general books [61, 63, 86] for details about this point.

More about* [Calibration]*: *By default, everyone uses a classical perspective camera. In this case, the calibration consists in a* 3×4 *matrix that describes the perspective projection onto the image plane.*

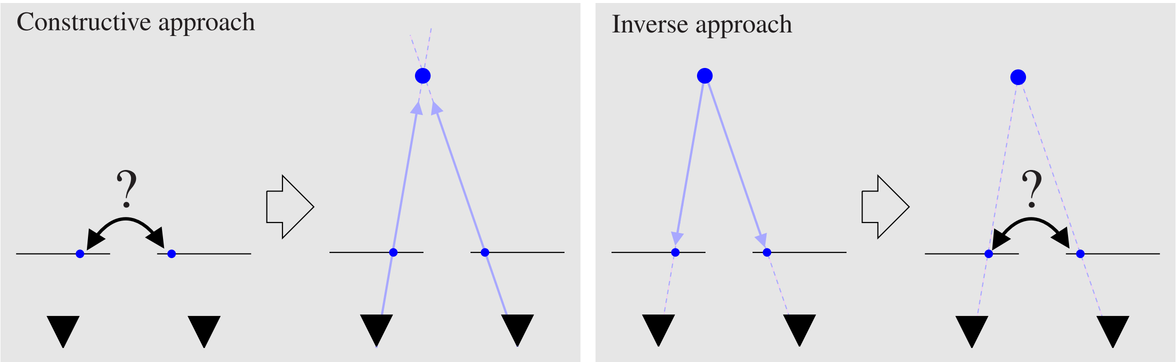


Fig. 2.3: How to recover a 3D point from two images

Left: *Constructive approach. A pair of points is matched in the images, the corresponding rays intersect and define a 3D position.* ***Right:*** *Inverse approach. A 3D point is projected in both images. The 3D point is valid if the image points match. This approach is more robust because the considered image points are guaranteed to be geometrically consistent with the 3D point.*

Contrary to the 2D case, two non-parallel 3D lines do not cross in most cases. And, because the projections are likely not to be perfectly located, the two 3D rays do not intersect in general. One may try to overcome this with the point that minimizes the distance to both rays or some equivalent criterion. But this approach is not robust: A small error on the 2D locations can produce a large error in the location of the 3D points. Therefore, the inverse approach is commonly preferred: A 3D point is first picked and the image points are built as its projections. If these image points match, the 3D point is kept; else it is discarded. This guarantees that the 2D points are perfectly consistent with the 3D one (see Figure 2.3).

Ill-posed problem: A major point to remark is that different scenes can produce the same images (Figure on the right). The problem is under-constrained: Several solutions exist and the only image consistency is not enough to differentiate them. Therefore finding a consistent surface is an ill-posed problem. In the introduction of this chapter, we have described a trivial reconstruction algorithm (see Figure 2.2 on page 7) that exploits this caveat.

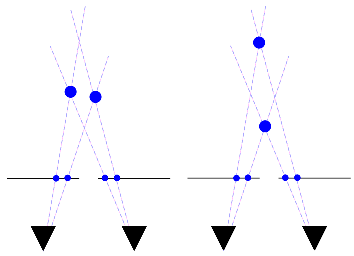


Fig. 2.4: An ill-posed case.

Conclusions of the two-image case: This overview outlines the following questions:

- How do we solve this ill-posed problem?
- How do we define and check that two points in different images “match”?
- What happens when some parts of the scene are *occluded i.e.* when an object hides another? Especially what do we do with the back parts of the objects which are not visible in the images?

With only two cameras, the last point cannot be solved: No information is available for the hidden parts. With several cameras, we expect some improvements depending on the camera positions. For instance, a region may be occluded for some viewpoints but not all. In that case, it seems possible to use the remaining viewpoints to recover the surface. And if the points of view are all around the object, it should be possible to build the whole surface (not only the front part).

Regularization: As previously shown, the problem is under-constrained, the solution is to set additional constraints. First of all, with multiple cameras, each additional viewpoint brings its own constraint: The result has to be consistent with one more image. But this may not be enough; consider a red wall for instance, all the images are red and none of them adds a significant constraint.

Then, the classical choice is to add some *a priori* knowledge on the scene. Several such constraints have been proposed [61]. But it seems that all the recent methods converge and agree on the same constraint: Common objects are composed of continuous surfaces. There may be some discontinuities but most of the surface has to be smooth. This is evaluated with different tools depending on the methods: curvature, depth variation, slope, area, etc. Nonetheless, they all rely on the following formalism: A mathematical expression – named *functional* or *energy* – is defined to represent both the consistency and the regularity of the surface. This functional assigns a value to every potential surface; the lower this value is, the better the constraints are satisfied. The aim is then to design a functional that represents our goal and to find a surface that minimizes this functional *i.e.* that satisfies as good as possible the trade-off between consistency and regularity. We give more details about functionals in Section 2.2.1.

Point matching: The question of matching two image points needs attention. The goal is to recognize in different images the projections of the same 3D point. Intuitively, we want to pair 2D points that “show” the same 3D point. We here describe the two main approaches used in surface reconstruction.

First, if the surface appearance is assumed to be *view-independent* (the aspect of a point does not dependent of the view point *e.g.* matte material), we can straightforwardly compare the colors. This is the *photo-consistency* criterion defined by Seitz and Dyer [190]. Since real images are unlikely to perfectly fulfill this criterion (the pictures may be noisy, the objects not exactly Lambertian, etc), a quantitative evaluation is defined to measure this criterion. The projections of a 3D point $\mathbf{p}$ produce a set of colors $\{C_i\}$. The photo-consistency $P(\mathbf{p})$ is evaluated by the variance of this set (with $\mathcal{V}_\mathbf{p}$ the indices of the cameras in which $\mathbf{p}$ is visible):

$$P(\mathbf{p}) = \frac{1}{|\mathcal{V}_\mathbf{p}|} \sum_{i \in \mathcal{V}_\mathbf{p}} C_i^2 - \bar{C}^2 \qquad \text{with} \qquad \bar{C} = \frac{1}{|\mathcal{V}_\mathbf{p}|} \sum_{i \in \mathcal{V}_\mathbf{p}} C_i \quad (2.1)$$

***More about* [View-independent]:** *Several authors use the word* Lambertian *as a synonym of* view-independent. *Rigorously speaking,* Lambertian *also implies that the intensity of the light reflected by the surface varies as the cosine of the angle between the incoming light and the surface normal.*

Ideally, if $\mathbf{p}$ is photo-consistent, $P(\mathbf{p})$ is null but, as discussed previously this is never the case for real images. In practice, we are looking for low values.

This criterion is necessary but not sufficient *e.g.* two images of a red wall are entirely red but very few pairs of red points correspond to points on the wall. This makes this criterion weak to reconstructing surfaces of uniform color. On the other side, it intrinsically takes into account and exploits the information coming from all the available images.

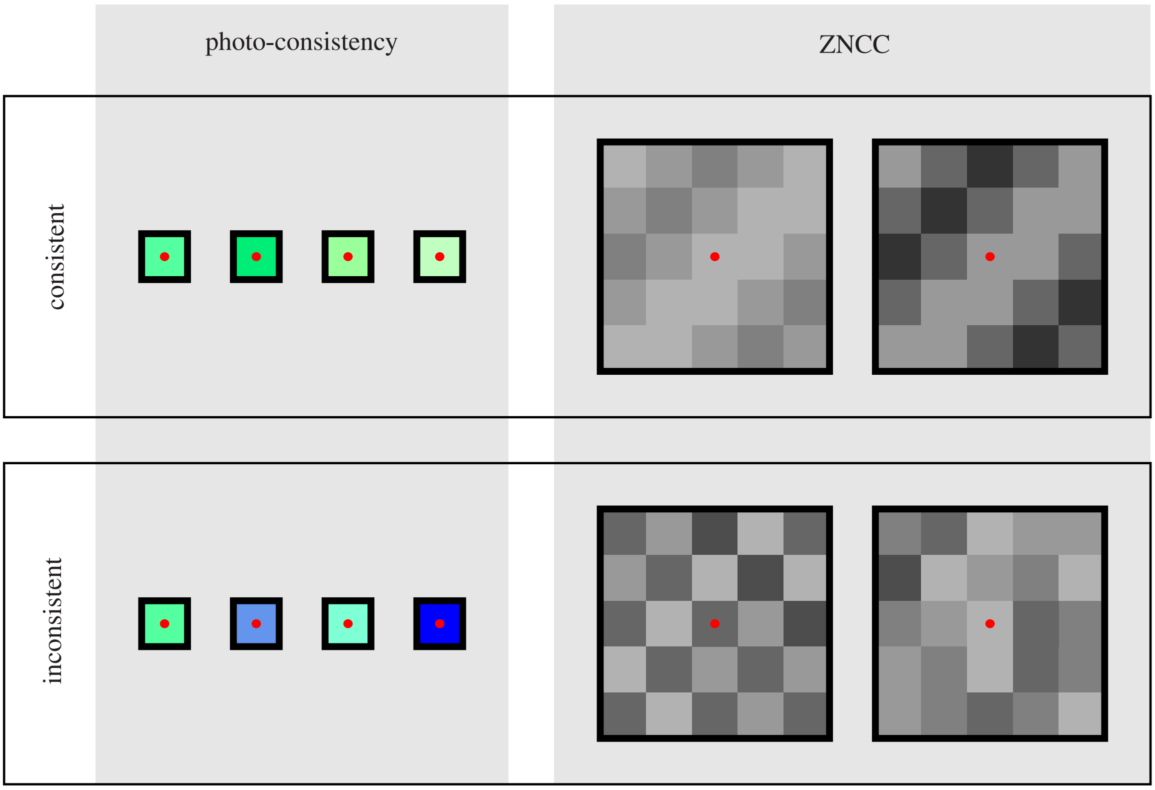


Fig. 2.5: The two main matching criteria: photo-consistency and ZNCC

Photo-consistency compares several points (four in the figure) by considering the color of the pixel under each point. ZNCC compares two points by analyzing the intensity levels in their neighborhoods (5×5 windows in this example). Photo-consistency easily handles several points whereas ZNCC is limited to two. It also exploits a richer information by using the color instead of the intensity. On the other side, ZNCC is robust against intensity scale and shift because it is normalized considering the neighborhood of the points. (The red dots indicate the points to be compared; the bold squares show the analyzed areas.)

Second, for surfaces with view-dependent variations, texture matching is often chosen. The motivation is that local patterns are likely to be robustly matched in presence of lighting changes. This approach compares the local variations of the images to find points with similar patterns *e.g.* strips or dots. This similarity is generally evaluated with the *zero-mean normalized cross-correlation* criterion (ZNCC in short). It is normalized both relatively to the mean intensity and to the standard deviation which makes it insensitive to an affine transformation of the intensities.

To define ZNCC for $\mathbf{p}_1$ and $\mathbf{p}_2$ in the first and second images, we need to introduce intensity function i, the neighborhood $\mathcal{N}_\mathbf{p}$ of $\mathbf{p}$. Since this computation considers is a small region (most often a square), the change of viewpoint introduces a perspective distortion depending on the local orientation of the surface. For instance, a square appears as square if it is seen from an orthogonal view but looks like a trapeze from a side view. Therefore, we use an homography π to account for this distortion: π is the geometric function that maps the projection of the plane tangent to the surface in one image plane onto its projection in the other image (see Figure 2.6). It induces the following correspondence $\pi(\mathbf{p}_1) = \mathbf{p}_2$ and $\pi(\mathcal{N}_{\mathbf{p}_1}) = \mathcal{N}_{\mathbf{p}_2}$.

Using $_1$ and $_2$ indices, we note $\bar{i}$ and σ the mean and standard deviation of the intensity in $\mathcal{N}_\mathbf{p}$ and $|\mathcal{N}_\mathbf{p}|$ the number of pixels in $\mathcal{N}_\mathbf{p}$, we define:

$$ZNCC(\mathbf{p}_1, \mathbf{p}_2) = \frac{1}{|\mathcal{N}_{\mathbf{p}_1}|^2 \sigma_1 \sigma_2} \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}_1}} (i(\mathbf{q}) - \bar{i}_1)(i(\pi(\mathbf{q})) - \bar{i}_2) \tag{2.2}$$

Thanks to the normalization, it is more robust to view-dependent effects with low frequency. This includes a broad range of effects. High frequency effects are similar to patterns and therefore perturbs the ZNCC matching evaluation. The counterpart to deal with local texture is that it requires the surface orientation to handle the perspective distortion *i.e.* to determine the homography π and it may be less accurate if a depth discontinuity crosses the point neighborhood.

Figure 2.5 illustrates both ZNCC and photo-consistency criteria.

Stevens *et al.* [204] and Szeliski and Scharstein [206] describe more sophisticated matching criteria. The former bases the comparison on the color histogram of the region covered by the image

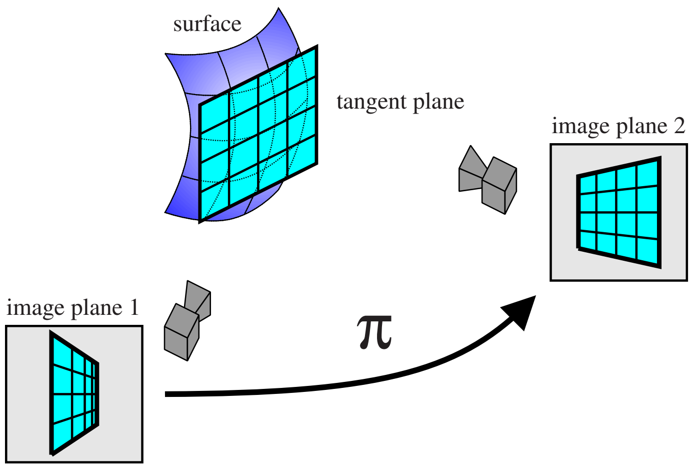


Fig. 2.6: The homography used in the ZNCC computation

To correctly compares the local texture of the surface, ZNCC involves an homography π that accounts for the perspective distortion. π transforms the projection of the tangent plane in the image plane of the first camera into its projection in the image plane of the second camera.

projection of a voxel. This makes possible to cope with a coarser discretization. The latter studies in depth the sampling issue *i.e.* how to deal with a discrete image representation. Both approaches seem interesting and would deserve further studies to examine their influence on existing reconstruction techniques.

Summary: The reconstruction problem relies on the ability to decide whether multiple image points are the projections of the same 3D point. The main techniques are color and texture comparison *i.e.* all the projections of a 3D point must have the same color or the same texture. A 3D point that projects onto matching image points is said *consistent*.

Unfortunately, this problem is ill-posed because different 3D scenes can produce the same images. Therefore additional constraints are set to overcome this point. Usually, the reconstructed objects are assumed to be composed of smooth surfaces.

Expected improvements from multiple points of view

As previously discussed multiple viewpoints should first make possible to better handle occlusions. Furthermore, since several cameras provide information on a given point, the redundancy should make the reconstruction process more robust against noise and more precise.

We now examine the most relevant approaches to our work. Note that techniques exist to extend the stereoscopic approach to three (see the work of Hartley [85] for instance); these methods do not really handle multiple views and should be more considered as “stereo from three images”. We will therefore not examine them. The other kinds of papers that we do not review are those relying on a parametric model of the scene content. One can read for instance the work of Debevec *et al.* [53] or Wilczkowiak [228] dedicated to buildings or the technique of Shan *et al.* [193] for faces. These papers, though interesting, are too specialized relatively to our goal.

Before reviewing the papers, we give more details about a few concepts needed to understand the existing work.

2.2.1 Fundamental concepts

Functional

As we have discussed previously, our goal cannot restrict to only seeking a high consistency reconstruction *i.e.* to creating a surface able to reproduce the input images. Such a problem is underconstrained and accepts several solutions. Therefore, additional constraints are imposed. The aim is then to find the “best” trade-off between the consistency and these constraints.

The computational solution to handle this trade-off is a numerical evaluation. A candidate surface is “rated”: A high score indicates a poor trade-off and a low score a satisfying one. The problem is then reduced to finding a surface minimizing this score and is therefore related to the mathematical field of numerical optimization.

Definition: The formula that gives a numerical evaluation of a surface relatively to a given goal is named *functional* (or *energy*).

The aim is then to find a surface that minimizes this functional. The problem is then twofolds:

- How to design a functional?
- How to minimize it?

These two issues will be discussed in the review of the previous work. We give in the following paragraph a brief description of a few mathematical tools useful to define a functional on surfaces.

Measuring a surface

Assigning a score to a surface is tightly related to measuring it. Intuitively, a functional is a "ruler" that "measures" the surface relatively to some dedicated distance. In the following explanation, we provide some formal definitions useful in the remaining parts of this document.

Formally speaking, measuring a surface $\mathcal{S}$ means computing the following integral:

$$\iint_{\mathcal{S}} d\mu \tag{2.3}$$

$d\mu$ indicates how the surface is measured. This integral sums all the areas of the infinitesimal elements of $\mathcal{S}$. For instance, considering the classical Euclidean metric ds gives the classical area of $\mathcal{S}$. Therefore, several functionals are defined relatively to this measure, using the following formulation:

$$\iint_{\mathbf{p}\in\mathcal{S}} w(\mathbf{p})ds \tag{2.4}$$

This modulates the surface measure with a weighting function $w(\cdot)$ depending of the consistency of $\mathbf{p}$. For instance, a surface piece that measures 1 cm^2 with the Euclidean metric can be counted as 0.1 cm^2 or 10 cm^2 for this functional, depending on the "quality" of this piece of surface relatively to the input images. The evaluation of this "quality" through the w function is a major point to be defined by a reconstruction method. Then, the problem is to find a surface of minimal area except that the metric is no more the Euclidean one.

In some approaches, the surface is parameterized as $(u,v) \in \mathcal{D} \mapsto \mathbf{x}(u,v) \in \mathcal{S}$. In this case, the measure can also be defined by:

$$\iint_{\mathcal{D}} w'(u,v,\mathbf{x})\, du\, dv \tag{2.5}$$

Contrary to formula (2.4), this area is measured in the parameter space *i.e.* u and v are not geometric entities. Even if the link with the previous formulation exists with the relation:

$$ds = \left|\left|\frac{\partial\mathbf{x}}{\partial u} \times \frac{\partial\mathbf{x}}{\partial v}\right|\right|\, du\, dv$$

some authors prefer to avoid manipulating non-geometric measures to solve a geometric problem. To overcome this, the surface is classically parameterized as a depth field using a function f:

$$(u,v) \mapsto \mathbf{x}(u,v) = \begin{pmatrix} u \\ v \\ f(u,v) \end{pmatrix} \text{ or equivalently: } z_{\mathbf{x}} = f(x_{\mathbf{x}}, y_{\mathbf{x}}) \tag{2.6}$$

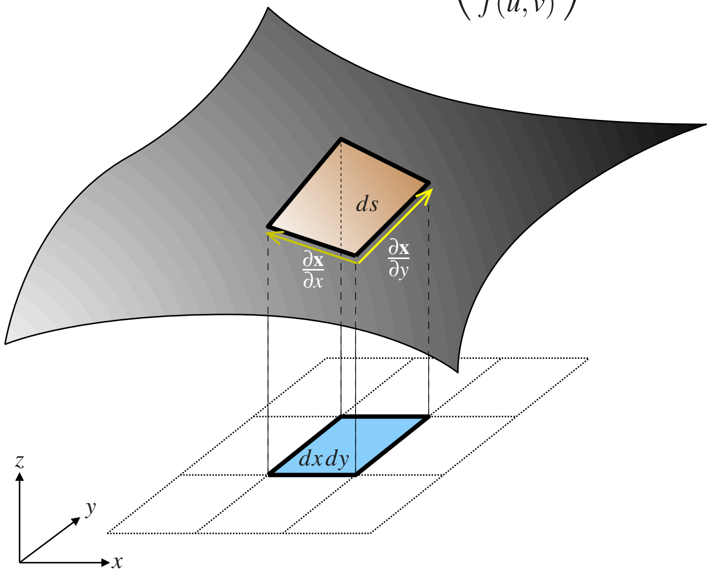


Fig. 2.7: Comparison of the elements ds and $dx\,dy$.

This straightforwardly associates u and v to the geometric lengths x and y. It leads to the special form of the functional (2.5):

$$\iint_{\mathcal{D}} w''(x,y,f)\, dx\, dy \tag{2.7}$$

The value of the integral (2.7) may depend on the coordinate system xyz *e.g.* turning the axes may change the result. In that case the functional is not *intrinsic* to the surface *i.e.* the functional value depends on the surface parameterization. It is important to remark that the functionals based on ds (eq. (2.4)) are intrinsic since their value is independent of the surface representation (*i.e.* it has the same value for all representations). Figure 2.7 relates both infinitesimal elements ds and $dx\,dy$ on a parameterized surface.

Summary: A functional is a 2D integral over a surface. It sums the weighted contributions of all the surface points. The weight function is to be defined by the reconstruction method. The objective is then to find a surface that minimizes the functional value. The minimization process is also to be specified by the method.

Surface properties

Convexity Since we aim at minimizing the functional, its analytic properties are a major point to our analysis. One may dream of a strictly convex functional $\mathcal{F}_{\text{convex}}$ that fulfills the following property for two surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$:

$$\mathcal{F}_{\text{convex}}\left(\frac{1}{2}(\mathcal{S}_1 + \mathcal{S}_2)\right) \quad < \quad \frac{1}{2}\left(\mathcal{F}_{\text{convex}}(\mathcal{S}_1) + \mathcal{F}_{\text{convex}}(\mathcal{S}_2)\right)$$

The definition of $\mathcal{S}_1 + \mathcal{S}_2$ is dependent on the surface representation. Such a strictly convex function would make the problem easier to solve because it guarantees that there is no local minimum. It would allow to use the vast literature on convex optimization [24]. Unfortunately, the problem is complex and in practice, the proposed functionals are not proven to be convex.

Continuity Since the functional is not convex, we have to rely on other techniques to minimize the functional. These methods lead to hypotheses on the continuity of the surface. But, let's first express a trivial remark about the surface continuity: If the formula of the functional contains a derivative of order n, then the surface is implicitly assumed to be at least n-times differentiable.

More about* [Discontinuity]*: It is possible to allow discontinuities even if the functional contains derivatives: It is sufficient to multiply the derivative terms by another term whose value is 0 where a discontinuity may appear, and 1 otherwise. Hence, on these 0 points, the derivative is canceled disregarding its value and the C^n continuity is not enforced. One has to take special care before applying this technique with the Euler-Lagrange formula described below because of the additional derivation that may make the 0 term "disappear".

In addition to this remark, more constraints on the continuity may come from the optimization technique. A common solution is to use the Euler-Lagrange formula to derive an evolution scheme, in spirit similar to a gradient descent. Consider a parameterization $z(x,y)$ that evolves according a pseudo-time variable t. For this paragraph only, we use the following notations: $z^{(x)} = \frac{\partial z}{\partial x}$ and $z^{(y)} = \frac{\partial z}{\partial y}$. We define a functional:

$$\mathcal{F} = \iint F\left(x, y, z, z^{(x)}, z^{(y)}\right) dx\, dy$$

where F is a scalar function of x, y, z and of the first derivatives of z. F has to be designed according to a given objective – this point is addressed later. The Euler-Lagrange formula is in this case:

$$\frac{\partial z}{\partial t} = \frac{\partial F}{\partial z} - \frac{d}{dx}\left(\frac{\partial F}{\partial z^{(x)}}\right) - \frac{d}{dy}\left(\frac{\partial F}{\partial z^{(y)}}\right)$$

This equation provides a rule for evolving the z function in order to minimize the $\mathcal{F}$ functional. There are two limitations:

- As a gradient descent, this evolution is not guaranteed to reach a global minimum. It may be stuck in a local minimum.
- It involves derivatives of z one order higher than the order used to define $\mathcal{F}$ since it contains derivatives of F which is based on derivatives of z.

Summary: In the ideal case, the functional would be convex to make the use of the convex optimization theory possible. Unfortunately, this is never the case in practice.

Therefore one has to rely on other properties of the surface, especially its continuity. If one wants to apply a descent technique (that classically involves the Euler-Lagrange formula), the surface must be differentiable one order higher than the highest derivatives that is used to define it.

Disparity map

To introduce the concept of *disparity*, let's first consider only two images of the same object from two different viewpoints. A point $\mathbf{p}_{3D}$ is projected in both images to form $\mathbf{p}'_{2D}$ and $\mathbf{p}''_{2D}$. Using the same coordinate system for both images, we define the 2D disparity vector $\mathbf{d} = \mathbf{p}'_{2D} - \mathbf{p}''_{2D}$. It can be shown that $\mathbf{d}$ is parallel to the line joining the two camera centers. Therefore only the norm $d = ||\mathbf{d}||$ is important. And, with the focal length z_f, the depth $z_\mathbf{p}$ of $\mathbf{p}$ and the baseline b (the distance between the camera centers, see the figure on the right), we get the following relation (from Thales theorem, by using the same image coordinate system *i.e.* by superposing both images):

$$b\, z_f = d\, z_\mathbf{p} \tag{2.8}$$

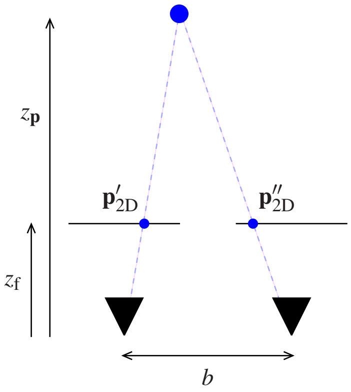
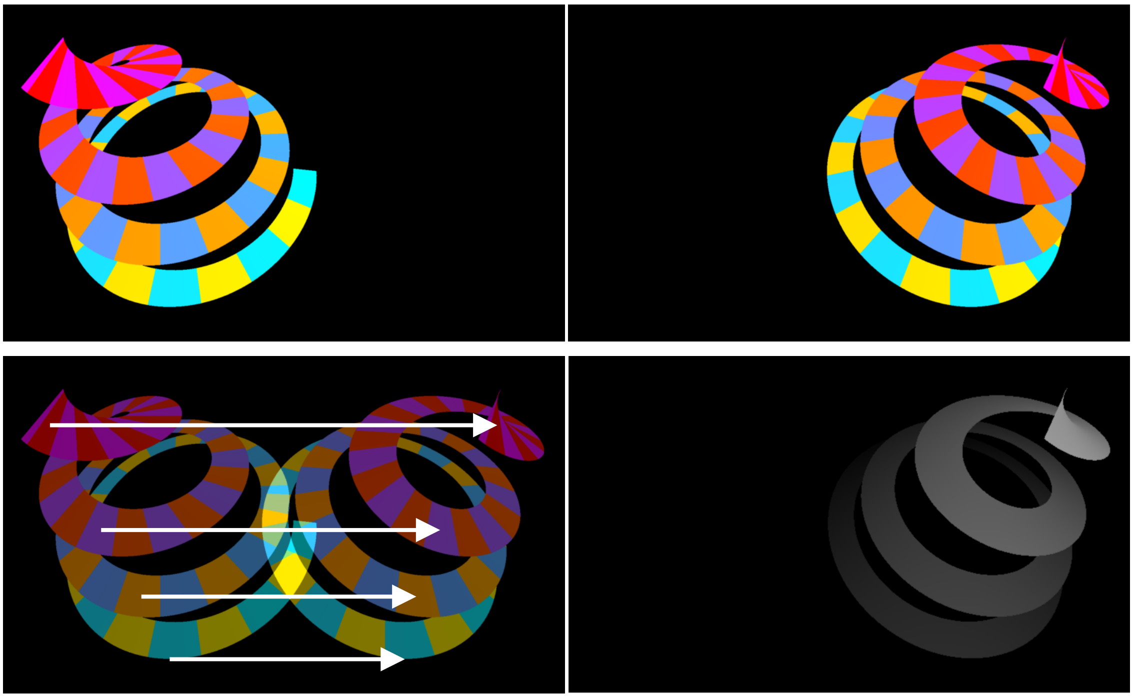


Fig. 2.8: Depth and disparity.

Stereographic pair of images



Superposition of both images

Disparity map associated with the above image

Fig. 2.9: Disparity map definition

Top row: *Two images of the same objects from two different cameras.* ***Bottom row, left:*** *The two images are superposed. The projections of the 3D points have moved between both images. Because of the perspective projection, the amplitude of this displacement varies with the depth of the considered point.* ***Bottom row, right:*** *The* disparity map *associates this amplitude to each pixel of an image (here, the top-right image). It is usually presented as a gray-scale image.*

In practice, a d value is computed for each pixel of one of the image: For each pixel, the corresponding pixel in the other image is searched (using the matching tools described in Section 2.2 on page 9), then the length of the corresponding translation is computed. This builds a *disparity map* for the image *i.e.* a function

More about **[Disparity]*****:*** *Since disparity is an image quantity, it is measured in pixels.*

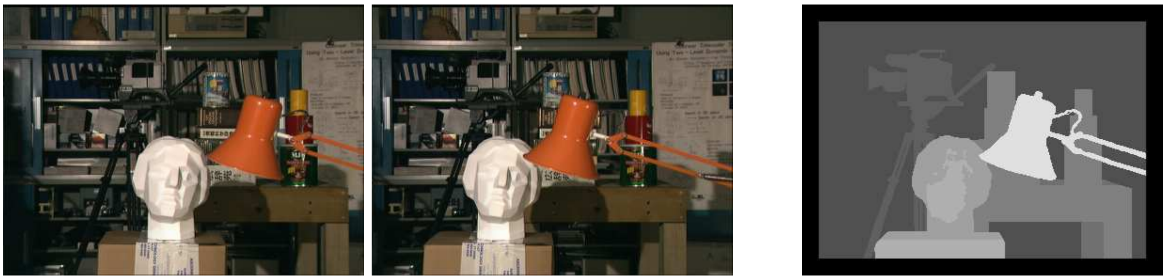


Fig. 2.10: Sample disparity map on a reference case

Left: *Two of the original images.* ***Right:*** *Ground truth disparity map (the gray levels represent the distance in pixels of the translation induced by the camera move; see Fig. 2.9 for details). Remark how the map is quantized because it considers only integer values for the disparity since it is measured in pixels.* [Data from `www.middlebury.edu/stereo`]

that associates a disparity value to each pixel. Figure 2.10 shows a sample disparity map and Figure 2.9 on the preceding page illustrates the construction of such a disparity map.

Since disparity is linked to the depth of a 3D point, many papers use it to define a 3D position instead of the classical (x_{3D}, y_{3D}, z_{3D}) system: They rely on x_{2D}, y_{2D} and d. (x_{2D}, y_{2D}) indicates a pixel in one image of a pair and d is a disparity value. From these values and the knowledge of the image pair, using equation (2.8), the location of the 3D point is known. This parameterization of the 3D space is sometimes named the *disparity space*. From this, we can extend the notion of disparity to more than two images by choosing two reference images. Then a 3D point is characterized by its projection in one of these two images and the corresponding disparity (*i.e.* the translation of the points between the two reference images).

Voxel space

Numerous methods work on a discretized 3D space. They rely on a 3D grid to represent the 3D space. This grid is comparable to the pixel array of an image. That is why, the basic 3D element is named a *voxel* in reference to the pixel. This entity has similar properties to the pixel: Various values can be assigned to it such as color, transparency, etc. This gives to this tool a great flexibility. Furthermore, as illustrated by the figure on the left, arbitrary topology can be represented. Shape precision directly depends on the discretization step, finer voxels give better approximations.

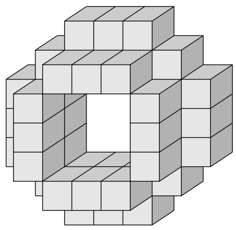


Fig. 2.11: Voxel ring.

Review of the existing techniques

In the following sections, we present the surface reconstruction approaches the most relevant to our work. To organize the discussion, we group the papers according to their methods to define and solve the functional that corresponds to the problem. Additionally to the traditional description and discussion of the methods, we also examine the usability *i.e.* the tasks that can be efficiently performed by a technique according to our experience.

2.2.2 No functional: Visual hull

This approach has been made popular by Laurentini [130] after some previous work such as the description of Baumgart [13] or the technique of Brik *et al.* [29]. This is purely geometric: assuming the

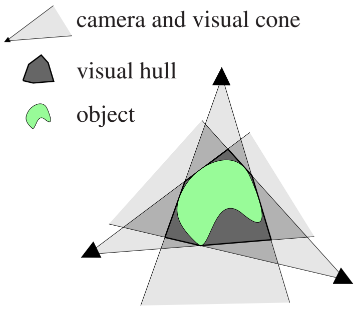


Fig. 2.12: Visual hull in 2D.

object silhouettes are known in each view, a camera defines a visual cone from its center through the corresponding silhouette. The object is guaranteed to lie inside this cone and the intersection of all these cones results in a bounding volume of the object (see Fig. 2.12).

This has the advantage of being independent of the surface aspect; only the silhouettes are important. It does not suffer from highlights, translucency or any other effects. The counterpart is that it requires a known background to extract the object contours. Zeng and Quan [237] propose a first step to alleviate this issue but it may lack accuracy compared to a chroma-key process. Furthermore, the quality of the result depends on the available views: tight results are obtained if the cameras are all around the objects but the precision significantly deteriorates with less viewpoints.

However, the computation are limited and the visual hull can be computed in real time, either as a discretized volume as used by Hasenfratz *et al.* [87] or as an exact geometric model (Matusik *et al.* [149, 150] and Franco and Boyer [25, 70]). Therefore, it is a natural choice for real-time application. It can be also used as a first estimate of the object shape as in the method of Isidoro and Sclaroff [102].

Some approaches (such as Vaillant and Faugeras [218] and Szeliski and Weiss [207]) refine the model by computing local curvatures of the observed object. Unfortunately these techniques induce unstable numerical computations and therefore have a limited robustness.

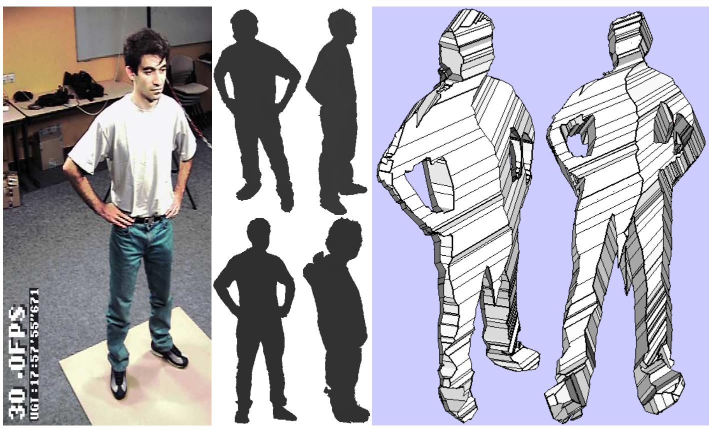


Fig. 2.13: Visual hull from 4 silhouettes

Left: *A sample input image.* ***Middle:*** *The four extracted silhouettes.* ***Right:*** *Views of the reconstructed hull. The overall shape is well captured (note that the topology is correct) but the surface suffers from too sharp edges.* [By courtesy of Jean-Sébastien Franco [70]]

Usability

These techniques are either useful "as is" when fast computation is mandatory, or as a first estimate of the object shape before applying some more precise refinements. But it should be avoided if geometric accuracy and details are targeted.

> ***Another view on* [Visual hull]*:*** *It can be seen as a reconstruction from binary images (foreground/background) whereas the other techniques handle 256 gray levels or the whole color space. One can think to a reconstruction from "quantized" information. This explains both the speed of the method (simpler data) and the lack of details (poorer information).*

Summary: The visual-hull methods work only from the object contours. They rely therefore on a side technique to extract these contours. From this information, they robustly compute the largest shape consistent with these silhouettes.

Relatively to our goal, the main drawback of this approach is its limited precision because it ignores the color information of the images. However, we may use it at a first estimate of the final shape.

2.2.3 No functional: Carving

This approach has been made popular by Seitz and Dyer [190] (Fig. 2.15) and Kutulakos and Seitz [127]. The base idea is to carve a large volume to shape a surface consistent with the input images. The core of this process is the consistency criterion that decide whether a 3D point should be carved away or not. Originally, under the Lambertian assumption (*i.e.* the objects are matte), Seitz and Dyer introduce the *photo-consistency* that we have previously described: A 3D point is consistent if all its projections in the visible cameras have the same color. This similarity is evaluated with the function P defined by formula (2.1 on page 10).

The original techniques use a threshold P_0 and carve out the points with $P > P_0$. The difference between the original *Voxel Coloring* algorithm [190] and the *Space Carving* process [127] is the visibility computation and the order in which the points are considered. The former approach restricts the camera positions (for instance, a plane must exist between the viewpoints and the scene) whereas the latter extends the method to any configuration. Figure 2.14 illustrates the first steps of the Space Carving algorithm. Figure 2.15 on page 21 shows a sample voxel reconstruction achieved with this algorithm.

The carving approach is proven to produce the largest shape photo-consistent with the input images *i.e.* that produces the same color images as the input data. There is no guarantee that it can produce new acceptable view because the process is constrained only by the input images without any additional a priori. That is why there is no need of a functional because there is no trade-off to satisfy. The counterpart is that the reconstructed surface may have noticeable artifacts such as bumps where information is poor (few visible cameras for instance). Therefore, these techniques are mostly applied with a high number of viewpoints.

Previously to this color approach, Collins [41] has proposed a related method based on the 2D features of the images: A voxel is kept if it projects on features (lines, corners, etc) in each images. This method is less precise because it uses sparse features instead of the dense information brought by the color of the pixels.

Culbertson *et al.* [45] refine the carving criterion with several technical improvements *e.g.* an exact visibility computation. Then Slabaugh *et al.* [196] add a simulated-annealing process which allow consistent points ($P \leq P_0$) to be carved although they should be kept according to the original algorithm. This allows to uncover points with better consistency. They also expose a technique [195] to warp the voxel grid to cope with the background.

***Link with* [Visual hull]*:* *The result of Space Carving is sometimes call the* photo-hull *to emphasize the similarity between both approaches: The visual hull is the largest shape consistent with the contours and the photo-hull is the largest shape consistent with the colors. One can also remark that tunning a carving process from black and white images representing the foreground/background (see Fig. 2.13 on the facing page) produces the visual hull.*

The voxel space is deformed in order to produce voxels of size increasing with the distance; the outer voxels are warped to infinity to represent the background. Kutulakos [126] also improves the carving criterion to make it more robust to image noise by analyzing the point projection and its neighborhood to find a correspondence. All these improvements make the carving approach more usable but do not significantly improve the accuracy of the final results nor extend the class of the reconstructible surfaces.

Szeliski and Golland [205] and de Bonnet and Viola [49] propose methods that allow partial carving of a point to handle transparency and antialiasing. They introduce several useful ideas but their results are limited in size and complexity. Following the same goal, Broadhurst *et al.* [30] applies a statistical framework. Their technique is more robust and more usable since it handles large real scenes and favorably compares with the classical carving techniques. It removes numerous artifacts and significantly enhances the visual quality of the results. When a surface is needed for further processing, the partial carving can be cast into the classical binary framework (a point is carved or not). Unfortunately the resulting surface is quite poor. Therefore, this method should be mainly thought to render new images from existing pictures.

All these methods are based on a discrete representation of the 3D space, the voxel space (cf. the description in Section 2.2.1 on page 17). These allow a simple and efficient exploration of the space. As the pixels, this representation suffers from the aliasing problem but this can be overcome with techniques such as *Marching Cubes* [140] or *Radial Basis Functions* as proposed by Dinh *et al.* [56].

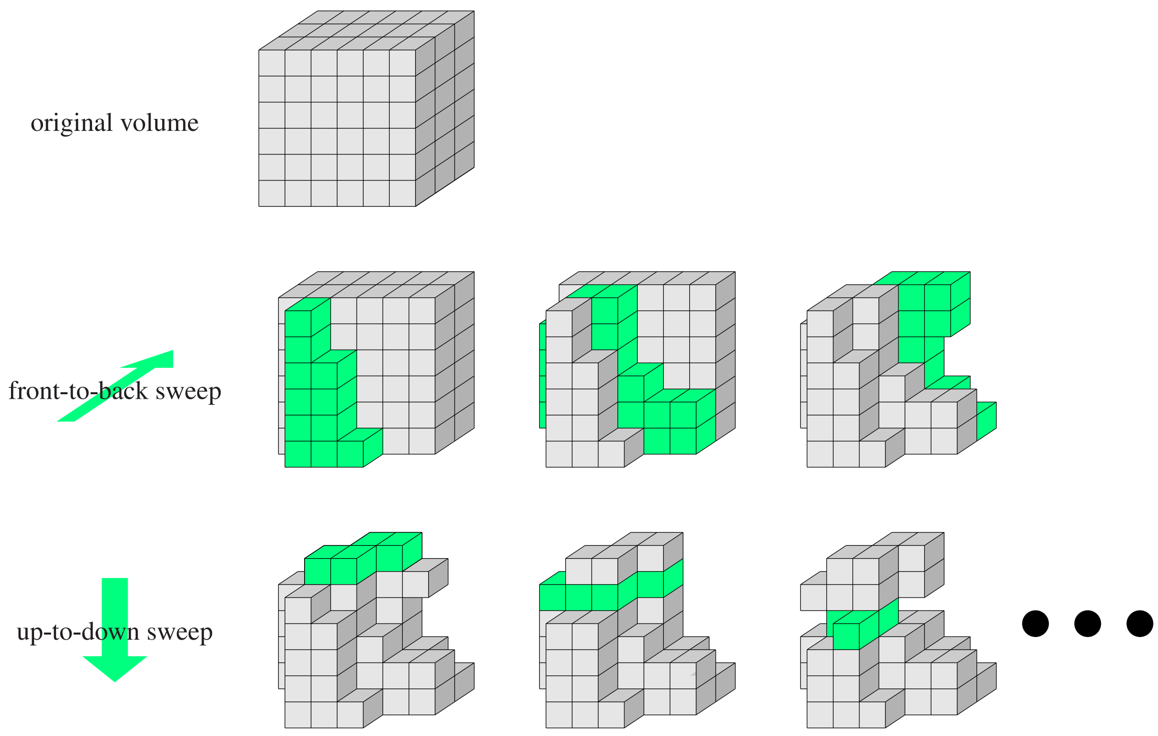


Fig. 2.14: Sample Space Carving process

The process is split into sweeps. *Each sweep analyzes all the voxels one by one. The voxels are ordered by planes in a given direction e.g. front to back as in the middle row. A voxel is kept if its projections in all the visible cameras have the same color, else it is carved away. The process loops over the 6 possible sweeps (x, y, z; both ways) and stops when no more voxel can be removed. The above illustration shows the initial volume with the first front-to-back sweep and the beginning of the following one in the up-to-down direction.*

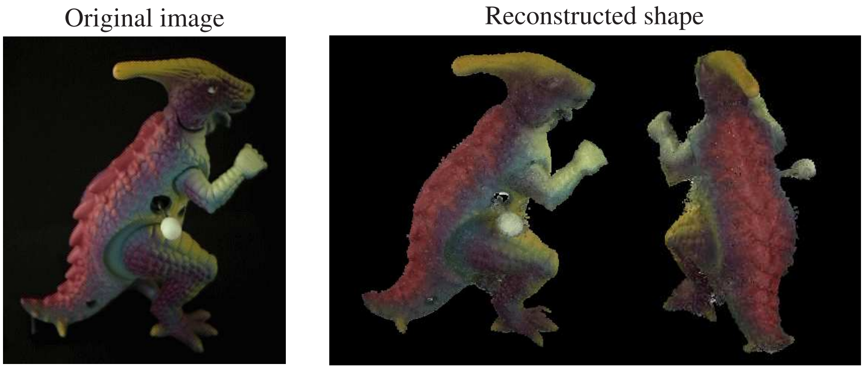


Fig. 2.15: Sample reconstruction from the Voxel Coloring technique

On textured objects, the Voxel Coloring algorithm performs well. Note that the reconstructed shape is correct although the arm hides the chest from some viewpoints. [By courtesy of Steve Seitz [190]]

However, carving seems very often linked with voxel whereas there is no fundamental reason to do so. The theoretical framework does not assume any special representation of the 3D space; voxels are only an implementation choice. For instance, Boyer and Franco [25] and Ziegler *et al.* [240] describe a carving process based on geometric polyhedra.

Discussion

The carving approach has several advantages. It is quite easy to implement, it handles any camera setup and produce surfaces with complex topology without any special adaptation. But it has also severe limitations described below.

Definition* [Topology]*:* *In the reconstruction context,* topology *is used for* genus*. It is related to the ability to recover objects with holes.

No regularization: As previously discussed, the problem is ill-posed and the carving methods do not propose any solution to that point. Thus, these techniques solve an under-constrained problem and only construct one of the many possible solutions. Snow *et al.* [201] propose to improve the resulting voxel volume but their method is a post-process totally separated from the reconstruction step. This actually does not regularize the problem but only smooth the result.

Space Carving is proven [127] to reconstruct the largest photo-consistent volume. It is also shown that the more textured the object is, the better this volume approximates the real surface. This approximation deteriorates if few viewpoints are available and if the object is textureless. In these conditions, most the carving techniques fail to recover an acceptable shape.

Low robustness: The photo-consistency handles only Lambertian surfaces. In practice, setting a higher threshold P_0 makes it tolerate low specularities but the precision loss is important with such a higher threshold.

Moreover, if a single voxel is erroneously carved *e.g.* because of an highlight, the error is likely to spread and the whole model may be carved out. This is predictable from the theory because the result of Space Carving is the largest photo-consistent volume; and if a region (even small) does not satisfy the Lambertian assumptions, then there is no photo-consistent volume and the result must be empty. However, this behavior is not satisfactory because a local error impairs the global result.

Usability

Considering these pros and cons, carving methods are useful if one needs a simple system, easy to set up and to use and if the fact that numerous failures may occur is acceptable. But it should be avoided if one targets robustness and high precision. However, this may be a good base to build upon a more sophisticated algorithm.

Summary: Carving techniques consider the 3D points individually. For each of them, they judge their consistency only regarding about the input images without accounting for the rest of the surface. This characteristic leads to an easy implement but strongly impairs its robustness because the decision is based on limited information. This last point does not suit our goal.

2.2.4 Functionals on lines: Dynamic programming

Most of these approaches aim at creating a disparity map as described in Section 2.2.1 on page 16. The map is built line by line resulting in a shape representation illustrated by Figure 2.16. This representation is only motivated by the technical solution proposed by these techniques.

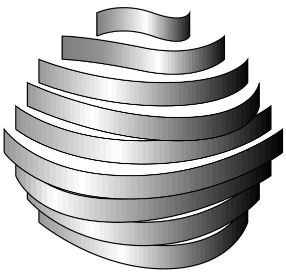


Fig. 2.16: Shape represented line by line.

In this framework, they propose to impose some constraints on neighboring points on the same y line of the image. A point (x,y) must both minimize its consistency with the input images and a smoothing criterion based on its position relatively to its neighbors $(x \pm 1, y)$. This defines a functional that associates a value to each y line. Low values correspond to smooth and consistent profiles. The advantage of such a formulation is that the problem is purely 1D (it only handles a line) and involves only first-order quantities. Therefore, a exact solution can be reached in polynomial time by a classical dynamic-programming technique [14].

The problem of this approach is that different y lines even adjacent are independent. This only enforces continuity along the x axis, the resulting surface can still be discontinuous in the y direction. Some methods try to be as precise as possible and simply stack the lines together (Okutomi and Kanade [167], Cox *et al.* [44], Pollefeys *et al.* [173], etc). However several techniques exist to enforce the continuity between adjacent lines: dynamic programming between y lines as Ohta and Kanade [166] and Bobick and Intille [21], statistical coherence as Ulvklo *et al.* [217], etc. Unfortunately, none of them propose a satisfactory approach since the x and y axes do not have the same regularization properties. Moreover in practice, it seems that a good trade-off between the y continuity and the precision is hardly reached: most of the results have either significant spurious discontinuities or poor details.

Since recent techniques such as Space Carving, level-sets and graph cuts have been exposed, these dynamic-programming approaches have appeared less interesting since they hardly compare with these new methods. The latest results of Pollefeys *et al.* [173] show convincing results thanks to a robust depth estimator based on a Kalman filter to

> ***More about* [Dynamic programming]*: It can be interpreted as an exhaustive search of the solution performed recursively while storing intermediate results to avoid redundant computation.***

aggregate the information from the multiple views. However it is not clear how this technique would perform on general objects since all the presented objects are densely textured.

Summary: These techniques introduce the notion of line regularization: The surface is built line by line and each line is imposed to be smooth. The smoothness in the orthogonal direction remains problematic and no solution seems to be acceptable compared to the latest graph-cut and level-set approaches.

2.2.5 Intrinsic functionals: Level sets

This approach has been first introduced by Faugeras and Keriven [62]. They characterize the reconstruction goal by a surface with minimal area. To ensure consistency with the input images, the area of a surface element is weighted by a term accounting for all the visible cameras. This formulation has the advantage of being intrinsic to the surface *i.e.* it does not depend on any parameterization. This permits to chose among all the possible representations. Especially, this allows the use of the *level sets* introduced by Osher and Sethian [169].

The level-set technique is based on an implicit representation of the surface: it is described by a 3D function $F(\mathbf{x})$ such that $\mathbf{x}$ is part of the surface if and only if $F(\mathbf{x}) = 0$. Osher and Sethian show that it is possible to translate an evolution rule defined on the surface to an evolution rule defined on F. Thanks to this property, implicit surfaces are efficiently manipulated. This first removes the parametrization issue and makes the description easier for complex shapes. Furthermore, topology is naturally handled whereas it is a difficult point to cope with for parametrized surfaces.

Faugeras and Keriven formulate their goal as a minimal surface according to a weight w based on the ZNCC scores of all the pairs of visible cameras:

$$\iint w(\mathbf{x})ds \tag{2.9}$$

For each small surface piece, it is projected in the input images and the covered textures are compared: If their correlation is high, the surface piece has a low weight w (*i.e.* it is assigned a small area) whereas if the correlation is low, the piece has a high weight (*i.e.* it is assigned a large area). The problem is then to compute the surface with a minimum weighted area. Section 2.2.1 on page 13 gives more details about weighted areas and Section 2.2 on page 9 about texture correlation.

They cast this problem into the variational calculus framework with the Euler-Lagrange formula which gives an elementary evolution of the surface that leads toward a minimum of the functional (2.9). Then, thanks to the level-set technique, this surface evolution is transposed into a volumetric evolution for the function F that implicitly describes the surface. So, starting from an initial shape (a sphere for instance), the surface evolves step by step and warps until it matches the object surface. This process is illustrated in Figure 2.19 on the

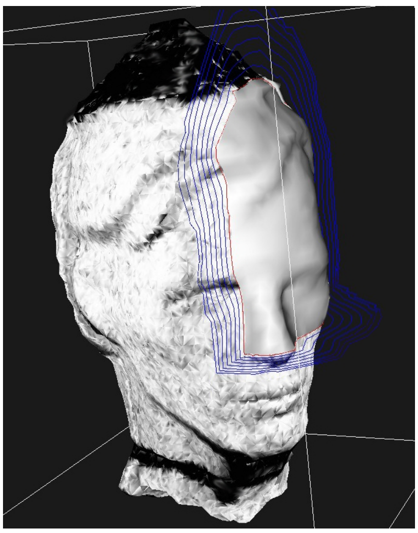


Fig. 2.17: Sample reconstruction using level sets. [By courtesy of Renaud Keriven [62]]

next page. At each step, the visibility and the orientation at every surface point is evaluated according to the current state. From these data, a new elementary change is computed and applied until a minimum is reached. This transformation may potentially include topology changes without requiring any special treatment: For instance, a sphere can naturally warp into two tori (Fig. 2.18).

Yezzi *et al.* [234] use the technique in a statistical framework that seems to produce similar results. Slabaugh *et al.* [197] propose a simplified evolution scheme based on photo-consistency. The surface can only shrink at each step. They use a multi-resolution strategy to speed up the process by first considering a coarse discretization of the space. But the convergence of this process seems uncertain in the general case since the authors mention difficulties when refining the resolution. Lhuillier and Quan [138] extend the method to account for various data types: ZNCC, 3D points reconstructed from feature points and silhouettes. Jin *et al.* adapt this framework to objects with a smoothly varying radiance [107] and with a piecewise constant radiance [106].

Discussion

These level-set approaches have the advantage to be purely geometric. Contrary to the disparity maps that characterize a 3D problem in term of image values (the disparity), the proposed formulation is based on 3D measures (surface orientation, curvature, etc). We believe that this is more coherent since we aim at reconstructing a geometric entity (a surface). A direct 3D formulation is more suitable than manipulating 2D data, even if they refer to the 3D world. For instance, Okutomi and Kanade [167] show that this may lead to an ill-posed optimization scheme.

Furthermore, the formulation is independent of the discretization needed for the resolution. In practice, the function F is sampled on a 3D grid but the functional ignores this technical issue and states a general problem. This allows multi-resolution approach like the method of Slabaugh *et al.* [197]. If the functional depends on the discretization (as all the disparity-map techniques are since they have a pixel-based functional), such an approach is hardly possible: How to guarantee that the same problem is solved at different resolution if the functional changes?

However, the level-set approach also suffers from some drawbacks discussed below.

Differentiability The weighting function w in the functional (2.9) differentiates twice values computed from the images. This computation may be hardly valid on real images that are affected by noise. Therefore, to guarantee that the images are regular enough, the derivatives are computed after

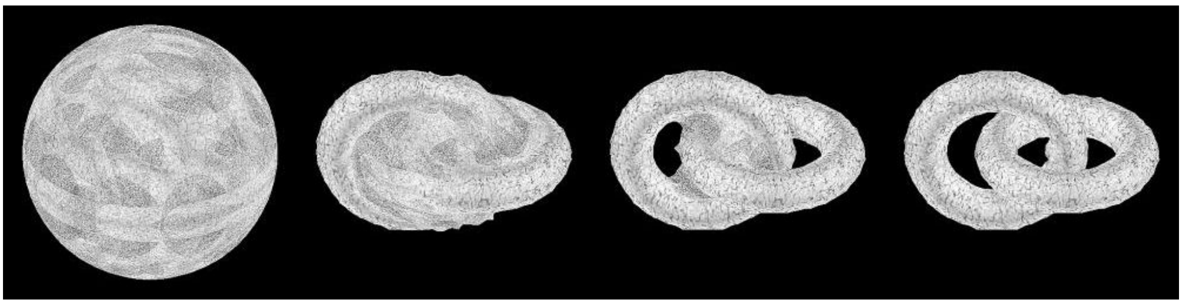


Fig. 2.18: Level sets: Sample evolution

The level-set technique is able to make a surface evolve to match two tori. The process seamlessly handles the topology change needed to represent the final shape. [By courtesy of Renaud Keriven[62]]

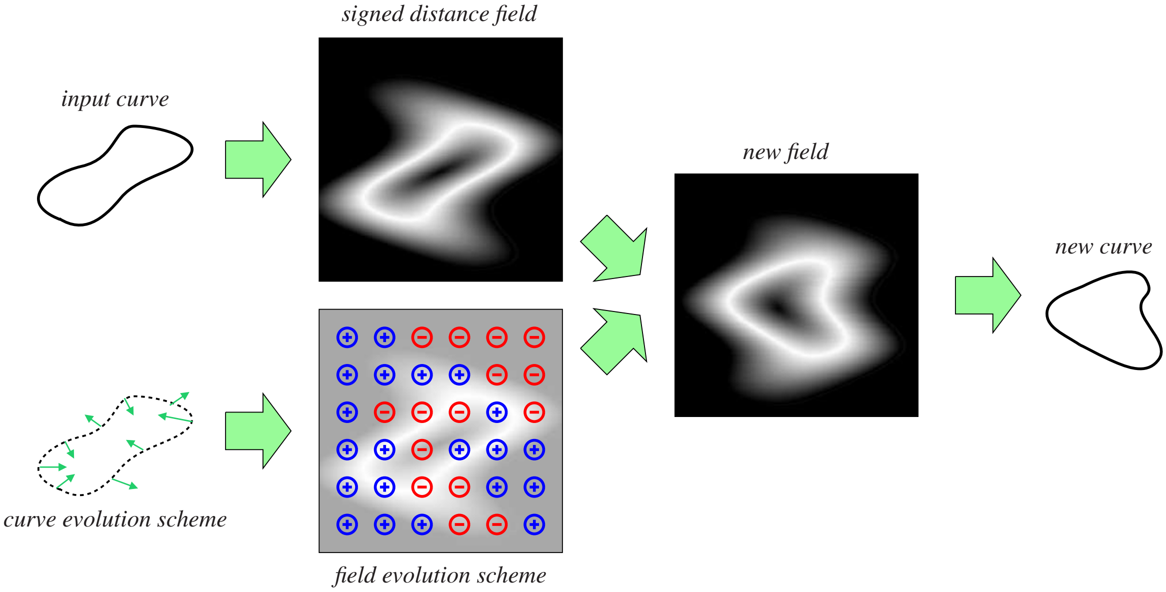


Fig. 2.19: Level sets: Overview of the process in 2D

Given a curve and an associated evolution scheme that tells how it has to be deformed, the level-set technique embeds the curve into its signed distance field and converts the curve scheme into a field scheme. Then the distance evolves according to this field. At the end, the result curve is extracted as the zero level set of the field. The interest of this method is that the field evolution is guaranteed to match the initial curve scheme while seamlessly handling topology changes and avoiding parameterization caveats.

a Gaussian convolution. It is unclear how this impacts the matching criterion (ZNCC from Faugeras' method [62]) according to the Gaussian width. To our knowledge, this crucial point is undocumented and would require further studies.

Convergence The evolution of the surface toward a steady state is a steepest descent process. The Euler-Lagrange formula gives an information that is comparable to the gradient: It "points" toward the low values of the functional. This is a purely local computation over the functional therefore the final result is only a local minimum. As for the gradient descent, the process may be stuck by a local minimum. Unfortunately there is no characterization of the distance between the value reached and an actual global minimum. Even worst, there is also no property about the shape. The consequence is twofolds:

- The produced shape may be arbitrary inconsistent and bad-looking.
- The result depends on the starting point: two runs of the algorithm on the same data may give two different results.

These points lack studies. In practice, according to the results shown in the papers, it seems that the convergence is correct if the starting surface is roughly in the neighborhood of the real object. The final shape is always acceptable. But we believe that the process would not converge toward the real shape if the initial surface is significantly distant from the object: The consistency function would be too irregular to drive any coherent evolution and the volume is unlikely to move closer to the real object where the function profile is "good enough" to make the process converge.

Therefore, level-set techniques should be associated with an initialization step using a first reliable estimate such as a visual hull or a carved volume. This may be a practical (but not theoretical) answer to this convergence issue. Lhuillier and Quan [138] point toward this way by incorporating the silhouettes into the functional.

Over-smoothed aspect All the results obtained from a level-set optimization look too smooth. A first reason to that is the convergence issue discussed above. The process requires a strong regularization term to ensure a satisfying convergence. And, since the regularization term involves some curvature flow, this yields results with no sharp curve. Furthermore, if we observe the functional (2.9), the c function contains second derivatives of the surface. Hence, the surface is sought among the surfaces twice differentiable everywhere. Therefore, the set of the potential results do not contain any shape with edges, peaks or corners. This limitation clearly impairs the produced surface because common objects have such features which are not captured by these methods.

Usability

From this discussion, we believe that the level-set methods introduce a great theoretical advantage (a geometric formulation independent of the technical issues: surface representation and discretization scheme). But the practical use needs special care: It requires a rather precise *a priori* knowledge of the object shape. Moreover, it is more appropriate to smooth shapes.

Summary: The methods based on level sets use a powerful framework that manipulate the surface using an implicit representation. This seamlessly handles topology and parameterization. Moreover, the optimized functional is formulated in the geometric world independently of the image resolution and of the optimization technique.

The counterpart is that these techniques are inherently limited to smooth surfaces without sharp feature. And, from a theoretical point of view, the convergence of the process is unknown.

However, we believe that a geometric formulation is an appropriate approach to describe a surface reconstruction because of its 3D nature. The independence relatively to the images and optimization engine is also a useful characteristic since, for instance, it allows to choose an arbitrary result resolution. Nonetheless, we will focus on providing a method that accounts for edges and corners and strive for a guaranteed convergence.

2.2.6 Parametrized functionals: Graph cuts

Graph cut is a classical problem in algorithmics [4]. It is traditionally presented as the graph flow problem. We here give a brief summary needed to discuss the following research papers.

> ***More about*** **[Graph flow]*:*** *Since graph flow is major topic of this document, Section 2.4 is dedicated to an extended presentation of this tool. This paragraph provides only a quick explanation to better understand the previous work.*

Overview of the graph flow problem Given a pipe network with a water source and a sink, the aim is to compute the maximum water flow that can go through. The problem is formalized with a graph: the pipe are the edges, the pipe junctions are the nodes, the source and the sink are special nodes of the graph. Then standard values and rules are defined such as “the flow through a pipe must be less or equal to its capacity”, “no water appears or disappears except in the source and the sink”, etc.

The graph-flow problem is: For a given graph with given edge capacities, determine a flow function on the edges that respects the rules and maximizes the total flow from the source to the sink. The two important results for the following discussion are:

- The maximum flow that can go through the pipe network can be exactly computed in polynomial time from the graph.
- A *minimal cut* (*i.e.* a set of pipes/edges that form a bottleneck limiting the flow) can be exactly computed in linear time from the graph and the maximum flow.

These two points are crucial since finding a bottleneck is intrinsically a minimization problem: Among all the sets of pipes separating the source from the sink, which one has the lowest capacity? Therefore the minimal cut problem is a minimization problem that can be exactly solved in polynomial time. All the following techniques establish a correspondence between their own minimization and a graph-cut problem. We will focus on this correspondence and the deriving properties.

Summary: The graph-flow problem is a traditional algorithmic problem which intuitively corresponds to finding how much water can go through a given pipe network. This network is represented by a graph whose edges have a fixed capacity (*i.e.* the maximum water that can go through them). A flow function is also defined on the edges to represent the actual water flow through the pipes. The problem is then to find a flow function that maximizes the total flow through the network.

If an optimization problem can be represented by a graph flow, then algorithms exist to find a global minimum of this functional.

Roy and Cox [180] first introduce the graph-cut technique to compute a disparity map $d(x,y)$ (cf. Section 2.2.1 on page 16 for details on disparity map). They do not clearly state a functional but their goal can be seen as a trade-off between the consistency of the surface points and a penalty depending on the disparity difference of adjacent points. Their graph associates a 3D point expressed in the disparity space (x,y,d) with a node. This makes the correspondence between a surface (a point set) and a cut which is an edge set difficult. However, they show that this approach extends to 2D surfaces the dynamic-programming technique (see Sec. 2.2.4 on page 22) which is intrinsically limited to 1D lines. The direct consequence is that the x and y axes are equivalent: The line-to-line coherence is no more a problem. This yields significantly better results, continuous in both directions. One year later, Roy[179] corrects the node/edge issue by associating 3D points to edges. The proposed functional involves the disparity map $d(\cdot)$, the image I, a consistency function c and a penalty function p which is applied to the neighbor points whose set is denoted $\mathcal{N}$:

$$\underbrace{\sum_{\mathbf{p}\in I} c(\mathbf{p}, d(\mathbf{p}))}_{\text{consistency term}} + \underbrace{\sum_{(\mathbf{p},\mathbf{q})\in\mathcal{N}} p(d(\mathbf{p}), d(\mathbf{q}))}_{\text{smoothing term}} \tag{2.10}$$

This functional is twofolds: The first term accounts for the consistency of the produced map relatively to the input images whereas the second term imposes a smoothing constraint over the result through the function p that penalizes the disparity variation. In Roy's formulation, $p(d(\mathbf{p}), d(\mathbf{q}))$ is proportional to $|d(\mathbf{p}) - d(\mathbf{q})|$. Hence, the penalty grows linearly with the disparity difference between two neighbors. It can be arbitrary large because of a depth discontinuity *e.g.* between an object and the background. This restricts this technique to a scene with limited depth variations.

Ishikawa and Geiger [99, 100] refine this approach. They explicitly formulate the functional corresponding to the minimization and cast it into the statistical framework by showing that it solves a Markov random field problem. They design a graph that handles any convex functions (*e.g.* a square function $(d(\mathbf{p}) - d(\mathbf{q}))^2$) for the penalty between neighbor points. This broadens the possibilities of the graph-cut method but does not overcome the limitations due to the depth variations. Buehler *et al.* [32] also propose an interpretation based on the minimal surface concept. Their objective is a minimal surface for a metric that takes the consistency into account. This leads to a geometric functional. But their graph layout is limited to two and three views and also over-penalizes large discontinuities.

The methods previously presented establish a direct correspondence between their functional and a graph-cut problem. Hence, they exactly solve the problem they have stated through their functional.

To solve issue of the large discontinuity, Zabih with Veksler [219], Boykov [28] and Kolmogorov [112, 119, 120, 121, 123] introduce another type of functionals that threshold the penalty linked to the depth difference: The penalty cannot exceed a given value. This makes arbitrary large discontinuities recoverable since their influence is bounded. The counterpart is that Boykov *et al.* [28] show that the corresponding problem is NP hard. This is a strong hint that this functional cannot be solved directly by a graph-flow problem which has a polynomial complexity.

Because of this difficulty, the functional is minimized with an evolution scheme that is not guaranteed to reach a global minimum. It nevertheless relies on graph cuts. Veksler [219] defines several *spaces* (sets) of *moves* that can be applied in sequence to a disparity map in order to reach a lower functional value. A move space defines a set of basic evolutions of the disparity map. Each move is simple enough to be optimally determined by a graph cut. In practice the moves that yield the best results are the *expansion* moves or α-*expansions*: A set of pixels $\{\mathbf{p}_i\}$ changes its disparity values to a given disparity α. Veksler [219] shows

Another view on* [Thresholded penalty]*: *The penalty can be seen as a regularizing potential that generates a force pulling the surface toward a steady state. If this penalty is thresholded, points over the threshold have a constant penalty i.e. a constant potential, hence no force is applied to them. Therefore they are controlled only by the consistency term which is ill-posed as we know. From a variational point of view, the Euler-Lagrange formula applied to the penalty term results in a constant null value over the threshold since it implies derivatives. The evolution is then equivalent to the ill-posed problem without regularization.*

that a graph cut can determine an α-expansion that reaches the minimum functional value among all the α-expansions. In other words, a graph cut can determine which pixels to change to α in order to minimize the functional under the constraint that the only allowed change is "assigning α to some pixels". Then by considering several values for α, the map evolves until no expansion can further minimize the functional. It is important to note that the solution may not be a global minimum since only the moves are optimal by the graph-cut property. There is no straightforward property about the sequence of moves *e.g.* it is not proven that a sequence of optimal expansions stops on a global minimum. Move spaces restrict the available transformations. Therefore, the allowed moves may not be able to describe a change that leads to a global minimum. Nonetheless, Boykov *et al.* [28] bound the final error of this process: The functional value reached cannot be arbitrary large compared to exact minimal value.

Using this framework, the best algorithm currently available is described by Kolmogorov and Zabih [121, 123]. They compute disparity maps for all the images of the sequence at the same time and introduce an additional visibility term that ensures that the maps are coherent between them *i.e.* that the map of an image does not occlude the map of another one. So, the process accounts for the occlusions from all the points of view during the minimization and is more precise. It is also possible to assign a disparity to pixel that represents an object occluded in another view *i.e.* to recover partially hidden objects.

Discussion

On the convergence issue, graph cuts have a clear advantage over the other existing approaches with their controlled result: Some methods guarantee to compute a global minimum and the others can evaluate the committed error. But in the latter case (the move-space techniques), the property is weak: It characterizes the convergence but, since it concerns the error on the functional, not on the surface, the result may be arbitrarily distant from the real one. On the other hand, these expansion techniques handle discontinuities through their thresholded penalty. This is clear improvement over other technique that suffer from large depth variations. However, this is not fully satisfying because the discontinuities appear as a result of the optimization engine. They are not directly linked to the input images. In practice, it produces undesirable discontinuities. Veksler [219] gives a first insight to overcome this point by modulating the penalty by the intensity difference in the reference image (the one supporting the disparity map).

> ***More about* [Binary functionals]*:*** *The set of functionals that can be minimized under the move-space framework is known as long as it involves only binary choices (e.g. an α-expansion decides whether the pixel disparity is set to α or not). We refer to the work of Kolmogorov and Zabih [122] for the details.*

Parameterized surfaces The first limitation of these graph-cut approaches is their parameterization as a disparity map depending on x and y. It makes difficult to recover objects that are occluded in some images but visible the others. It implies that a given pixel (x,y) has two disparity values for two 3D points project onto the same pixel. The method of Kolmogorov and Boykov [121, 123] indirectly overcome this problem with their multiple maps. This double-value situation is solved from another view in which the two 3D points project onto two different pixels. But it then requires further work to merge the multiple maps into a single description of the scene.

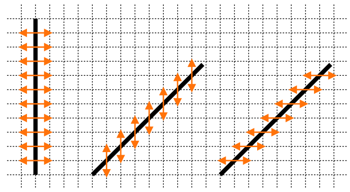


Fig. 2.20: Parametrized penalty is rotation-dependent. *A 45° discontinuity is more penalized (14 arrows: 7 on each axis) than an axis-aligned one (10 arrows).*

The *xy* parameterization also makes the functional depend on rotation. Turning the images (or the coordinate system) changes the stated problem because most of them rely on a 4-connected neighborhood system. For instance, consider a unit-length discontinuity in these two configurations:

- Parallel to the y axis: The penalty for the x axis counts for the full length and the y penalty is null.
- $45°$: Both x and y penalties count for a length of $\sqrt{2}/2$ (*i.e.* the length of the projections on the axes).

A $45°$ angle is more penalized by a factor $\sqrt{2}$ than an equivalent axis-aligned discontinuity (Fig. 2.20). This behavior is not desirable. An ideal system should be rotation-invariant as the level sets are.

Disparity maps The main drawback of these graph-cut techniques is their non-geometric formulation based on disparity. Even if disparity is linked to depth, it introduces a bias in the formulation. It implicitly defines the images as the working space whereas we seek 3D entities: Characterizing a surface by image displacements is not suitable for a geometric problem. Okutomi and Kanade [167] show that it can introduce ambiguities (several global minima) whereas directly dealing with depth yields unambiguous results. Furthermore, disparity binds the functional to the image resolution. As a consequence, the same problem can hardly be solved at two different resolutions because changing the resolution also changes the problem. Therefore, a multi-resolution approach would need at least a reformulation of the functional *i.e.* change almost everything.

Usability

If the shape precision is important, these graph-cut techniques would not be appropriate because of their limited depth precision (the results are piecewise depth-constant and therefore look blocky, see Figure 2.10 on page 17). Moreover, they only describes the front side of the objects.

But the strength of these methods lies in their precise image segmentation and robust convergence. They are therefore a good choice to decompose an image into depth layers *e.g.* for image editing or flat impostor creation.

Summary: These methods rely on functionals that can be solved with the graph flow algorithm. There are two main classes of methods. On the one hand, some techniques are directly equivalent to a single graph cut. They minimize exactly their functional but have difficulties to properly handle depth discontinuities. On the other hand, the other methods overcome this problem but their convergence is weak in theory and they yield blocky results in practice.

Considering our goal, the main drawback of all these methods is their formulation as a disparity map problem. We believe that an image-based representation does not fit an 3D geometric problem. However, the controlled convergence of some of these techniques is a positive point that we would like to enjoy whereas correctly dealing with depth variations without producing blocky surfaces is an appealing challenge.

2.2.7 Intrinsic functionals: Graph cuts

Boykov and Kolmogorov [26] describe a method that produces a surface to segment data into two sets *e.g.* foreground/background for images or organ/body for volumetric medical data. The functional is similar to the one used in the level-set approach (eq. (2.9)). They are seeking for a surface with a minimal area using a measure that accounts for the data. For instance, to segment an image along its edges, the measure varies with the gradient: Large gradients induce a smaller area measure. They rely on a Cauchy-Crofton formula to convert this minimal surface formulation into a line-crossing problem. They show that the area of a given surface can be estimated from a set of lines crossed by this surface. Intuitively, considering a set of parallel lines, a large surface would cross more lines than a small one. Therefore using several such sets of parallel lines and counting the number of lines intersected by the surface is related to measuring its area. From this relation, the problem is cast as a graph-cut problem which can be seen as an edge-counting problem.

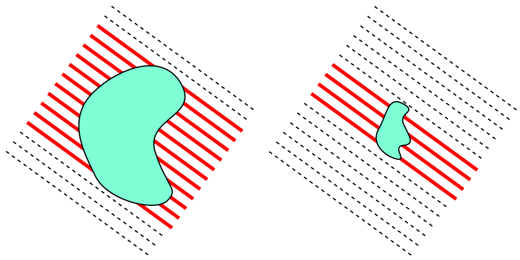


Fig. 2.21: Intuition on the Cauchy-Crofton formula in 2D. *Big objects cross more lines than small ones.*

Since level sets and this approach share the same functional, they are interesting to compare. Both have a geometric formulation independent on the space discretization used. However they differ on the topology management and on the convergence. On the one hand, level sets reach only a local minimum but handle any topology. On the other hand, this method reaches a global minimum for the whole problem is formulate as a single graph cut. But topology is user-assisted. The user is required to indicate some points (named *seeds*) in both regions, at least one for each connected component. Conversely, every seed generates a region. That is why this step is especially sensitive: A wrongly placed seed would generate an entire region or a missing seed can remove a whole region. Therefore, topology management is more awkward for this technique.

Furthermore, this graph-cut technique for data segmentation is not straightforward to adapt to surface-from-images purpose. First, the seed placement needs to be robustly automated. But, over all, visibility has to be incorporated into the process. It may be possible but clearly requires further study. The main obstacle is that the process is based on a single graph cut: Before the algorithm is run, visibility is unknown since the geometry is yet to discover and after, the result is already computed, it is too late to account for visibility. Contrary to the level sets, it is not a step-by-step process that allows an update of the visibility in parallel with the shape estimation. With a single graph cut, visibility cannot be handled separately. It has to be included inside the optimization engine and thus raises non-trivial issues. However, adapting this data segmentation to surface reconstruction would be an interesting challenge.

2.2.8 Hybrid methods

None of the previously presented methods is perfect and, naturally, some researchers propose hybrid techniques combining several approaches. Ideally, the overall hybrid algorithm would overcome the limitations of each involved technique while preserving their advantages. By nature, these methods are hard to classify. So we consider them in a subjective order of "hybridity".

Poulin *et al.* [174] propose a reconstruction made of a dense cloud of points. The originality of their approach is to let the user drive a classical point algorithm. The user indicates the regions where the algorithm has to focus on, which part should be refined, etc. Since the user "helps" the algorithm, this makes possible to recover fine details on the objects that another technique would have ignored. The counterpart is a time-consuming process. Moreover, the final result would require some post-treatment to yield a data structure more usable than a point cloud.

Snow *et al.* [201] improve the result of a carving process with a graph cut. The effect is quite similar to the morphological treatments proposed by Museth *et al.* [161]. The drawback is that the graph-cut step totally ignores the input data and may yields inconsistent results.

Isidoro and Sclaroff [102] present a surface optimization scheme that is guaranteed to respect the visual hull *i.e.* the visual hull of the result is exactly the same as the original object. The evolution scheme involves free-form deformation tools that are controlled by probabilistic techniques in order to maximize the visual match between the original images and the surface estimate. This approach performs well on objects that are almost similar to their visual hull but strongly relies on texture to refine concave regions. And as level sets, it produces over-smoothed results.

Lhuillier and Quan [138] extend the level-set technique to handle multiple sources of information. The surface evolution can be driven from the standard texture correlation and also from some 3D points reconstructed from some stereoscopic triangulation and from silhouettes extracted using chroma-key methods *e.g.* with a blue background. In practice this technique improves the quality of the results but they still lack sharp details on concave regions (observe the "hair" in Figure 2.22).

Fig. 2.22: Sample reconstruction using various information sources

Left: *Three of the 26 input images.* ***Right:*** *Results using feature points to control the surface creation. The produced can have sharper edge than those produced by the classical level-set technique (Fig. 2.17) but still lacks details in concave areas (observe the hair).* [By courtesy of Maxime Lhuillier [138]]

Some methods also mix multi-view geometrical methods as presented in this section with *shape-from-shading* techniques [95] (*i.e.* the 3D is recovered by analyzing the object appearance under known lighting conditions). These methods seem promising since they exploit more information. But, in many cases, the lighting conditions are unknown, at least partly; making more difficult such an approach. In this configuration (unknown light position), for instance, Zhang *et al.* [238] only show results with a single point light source on relatively simple objects. In addition, many of these methods rely on a model of specular materials (*e.g.* Yang *et al.* [233] assume that the highlight color is dependent only on the light) or an example (*e.g.* Treuille *et al.* [214] use objects of known geometry with similar materials as references). The major drawback to these methods is that it is difficult to a priori predict if a model or example of material will be appropriate to describe a given scene.

Summary: Hybrid methods are promising since a careful design should benefit from the best properties of the mixed methods while mutually compensating their shortcomings. However none of them still fulfills all our goals in a single method.

2.2.9 Discussion

From this review, we believe that some points are important to consider before designing a new algorithm. Since our goal is to capture the geometry of 3D objects appearing in multiple images with a limited user input, a method should ideally combine the following aspects:

- A geometric formulation that considers the object surface directly as a 3D entity and not through its image projections.
- An intrinsic functional that is independent of the parameterization which describes the surface.
- An optimization process that is guaranteed to converge to a global minimum of the functional – or, at least, whose convergence is known and characterized.

Geometric formulation

The first point is our major concern. We do not believe that disparity maps are suitable to our task. These maps are formulated as an image problem whereas we target a 3D problem. Adding a "translation step" to go from the disparity to the depth is likely to make the problem more complex.

Nonetheless, disparity maps are useful for some tasks such as image layering or impostor creation since they segment the input images into large regions with almost constant depth. Agarwala *et al.* [3] describe several applications which use this kind of image data. Moreover, since ground-truth data are available [188] and precise comparison can be made [187], this "challenge" has been highly successful and great progress has been done. But, now, the best methods yield almost perfect results: Error is at most a few percents and comparison is no more significant when the accuracy differences are in the order of 0.1% (from the website `www.middlebury.edu/stereo`). This means that the produced maps differ by at most a few pixels. Moreover, even if we observe the ground-truth data (Fig. 2.10 on page 17), a disparity map is a poor approximation of the real objects for it is a collection of flat regions. We target a more challenging result: Accurately approximating the real surface of the objects.

To build a geometric surface, we believe it is important to work directly in the 3D space. Therefore we will follow an approach similar to Faugeras and Keriven [62] and describe our goal using a geometric coordinate system (x,y,z) and geometric entities such as slopes, Euclidean distances, etc. Furthermore, this separates the functional from the input data and makes it independent of the space discretization chosen to optimize the functional.

Intrinsic functional and exact optimization

Formulating a functional independent of the surface parameterization and reaching a global minimum are two important issues because they may be the origin of a failure to reconstruct an object. In such case, the user would be required to change the setup of the algorithm and to run it again. This would clearly impair the usability of the method because it would require more user and computation time. Moreover this change of setup is likely to be unintuitive.

However, these two points seem to be less crucial issues. These are clearly important theoretical problems but they can be, at least partly, overcome in practice. The impact of coordinate system is hardly distinguishable on the graph-cut results (see [123] for instance). Level sets [62] or Space Carving [127] also yield satisfactory results on numerous practical scenarii although their convergence is weak in theory. We will nevertheless consider carefully these issues and criticize our techniques against these criteria.

2.2.10 Validation

After reading all the papers mentioned in this review, we have the feeling that the validation is a crucial but complex issue.

As we have seen, the reconstruction problem leads to theoretical studies. And it may be tempting to consider this theory as a proof of correctness. We believe that this is not sufficient, a method have to be tested on real cases. A theoretical study relies on theoretical hypotheses and we are convinced that no such hypothesis perfectly holds in practice. For instance, noise is never perfectly Gaussian or a small surface piece is not exactly matched by its tangent plane. Theoretical studies are important to design an algorithm, to characterize its fundamental properties and limitations, etc. But a practical proof is however mandatory to confirm the validity of the overall process, to evaluate how robust it is to the minor but multiple deviations from the theoretical model.

We can now wonder how to validate a reconstruction work. But, let's first ask another question: "What do we want to validate?" It may seem trivial but we have to know what we are interested in before validating it. To us, there are two main answers to this question:

Accuracy: In this case, we want to evaluate how close is the reconstructed geometry from the original one.

Property: This task is not as straightforward as accuracy. We aim at characterizing some behaviors such as topology or occlusion management.

Accuracy

This validation is the most intuitive one: Do we have reconstructed a precise shape? But it is also the most difficult to perform. A straightforward validation requires to know the exact geometry of the observed scene. And, in most of the real cases, it is unavailable. Range scanner can provide reference data for small objects and are cumbersome to use in numerous situations. Some ground-truth data exist for disparity maps (Fig. 2.10 on page 17) but these data are not the exact geometry since they include the hypotheses of the disparity maps and are piecewise flat, which is obviously not the actual scene geometry.

Another option is to use synthetic images whose geometry is known. But they would be of limited utility as practical validation since they perfectly fit all the assumptions made to design the algorithm. Therefore, we believe that such a test is almost useless to validate the accuracy because it provides no guarantee about the efficiency on real data. Among other phenomena, the noise inherent in real capture and in calibrating real images degrades the performance of any algorithm. A synthetic test does not evaluate this crucial aspect.

It seems that the solution lies in using real images with a geometry complex enough to make possible a subjective evaluation by the reader. For instance, one can show the resulting shape without texture and flat-shaded. This conveys intuitive visual cues about the geometry. Another approach is to show the reconstructed surface under the same pose as an image of the real object. A side-by-side comparison lets the user judge the fidelity of the result. We nonetheless believe that it is less compelling than the previous flat-shaded image since it mainly compares the contour accuracy because the texture "hides" most of the in-between geometry. An improvement is to use an image that is not part of the input data. So the texture mapping may be distorted, revealing underlying errors. The last option is to compare the result with existing methods on the same input data. This is interesting as long as both experiments are given the same care. To be fair, one has to ideally work with the author of the compared method to guarantee that the process is correctly run.

Summary: To evaluate the accuracy of a method, we are convinced that one has to prefer real images over synthetic pictures for computer-generated data are too perfect.

But using real data, the actual geometry is almost never available for objective comparison. Hence, only subjective evaluation is possible. To ease this task, several possibilities exist: flat-shaded rendering without texture, rendering from the same view as a real image, comparison with other existing methods.

Property

In addition to the accuracy, another important experiment is the validation of the properties such as topology handling, robustness to occlusion, sensitivity to the number of images, etc. Figure 2.18 on page 24 illustrates such a validation for the topology with level sets.

We believe that the difference between real images and synthetic ones is less crucial in many cases. For instance, topology and occlusions can be demonstrated on synthetic images that still conveys convincing demonstration. These images also provide "toy" examples *i.e.* exaggerated scenarii such as Figure 2.18 on page 24 that outlines specific capacities even on artificially complex situations. However, we still believe that real images should be preferred when it is possible since all the considered algorithms target real data. For some properties such as the influence of the number of images, real pictures are even mandatory because that kind of validation is linked to the accuracy previously discussed: It aims at characterizing the accuracy as a function of the number of images. In general the quantitative properties (most of the time accuracy as a function of the image resolution, of the number of images, etc) should follow the same logic (using only real data) whereas the qualitative properties (ability to handle topology, occlusions, etc) can be also tested with computer-generated data.

Summary: It is also preferable to validate the properties of a technique on real data. But it may also interesting to design dedicated artificial tests for the purely qualitative properties (topology, occlusion, etc). However, when it comes to test the impact of a parameters (noise, resolution, etc) on the final precision, we still believe that real pictures are mandatory.

Closure

We believe that validation is a hard part because objective evaluation is often unavailable. It therefore mostly relies on the subjective opinion of the reader. Furthermore, the aspects to validate are numerous; among the possible ones, we may cite:

Precision: How accurate is the recovered shape compared to the original one?

Occlusions: Is the technique able to recover a partially occluded objects?

Texture: How much is the method dependent on the object texture to capture its geometry?

Number of images: How many images are needed to achieve satisfactory results? How does the quality degrade with less images?

Resolution: How does the results vary with the image resolution? With the space resolution?

Regularity: How does the regularization of the functional alter the results?

Information source: For hybrid technique, how the different methods interact to produce the final result?

It is almost impossible to validate extensively everything. One has to focus on the main aspects on a technique.

Nonetheless, we have been several times disappointed by the validation provided is some papers. It might be incomplete (*e.g.* only one example shown) or not very relevant (*e.g.* too simple objects). We think that it impacts the work achieved by the authors. We are convinced that all these papers deserve attention and expose methods that are interesting to better understand the problem, even if only very few are effectively production-ready. But, from our experience, these papers make non-specialists think that "Computer Vision is theory without validation" and it is hard to argue against. We actually think that too many papers assume the reader to be a specialist and hence, do not details the real performance of their technique and discuss too shortly both the limitations and strong points of the exposed algorithm. This situation is regrettable since many valuable techniques exist and could widely spread outside the Computer Vision field. We believe that the lack of validation is mostly responsible for that although there are high quality methods.

2.3 Problem statement and design of the functional

Our goal is to create an object surface in 3D space. We first consider a general formulation that outlines the relations between our work and existing geometric methods.

Let $(u,v) \mapsto \mathbf{x}(u,v) \equiv (x_{\mathbf{x}}(u,v), y_{\mathbf{x}}(u,v), z_{\mathbf{x}}(u,v))$ be a regular parameterized surface representing the underlying object. Three-dimensional reconstruction can be cast as an optimization problem of finding a suitable function or surface $\mathbf{x}$ that minimizes the functional:

$$\iint c(\mathbf{x})\, du\, dv$$

where $c(\cdot)$ is a positive function measuring the consistency of the surface relatively to the input images. The methods to evaluate this consistency are detailed in Section 2.2 on page 9. Since reconstruction is ill-posed, this simple functional needs to be regularized to achieve reliable results.

Traditionally, the regularization terms are directly introduced for the parametric surface patch. Then, the regularized problem is formulated by Terzopoulos *et al.* [210] as a deformable surface model minimizing an energy involving terms of the form:

$$\iint \left(a \left\| \frac{\partial \mathbf{x}}{\partial u} \right\|^2 + b \left\| \frac{\partial \mathbf{x}}{\partial v} \right\|^2 + c \left\| \frac{\partial^2 \mathbf{x}}{\partial u^2} \right\|^2 + d \left\| \frac{\partial^2 \mathbf{x}}{\partial v^2} \right\|^2 + e \left\| \frac{\partial^2 \mathbf{x}}{\partial u \partial v} \right\|^2 \right) du\, dv \tag{2.11}$$

Where (a,b,c,d,e) are real weights to control the relative influence of each term. Several similar terms are then combined to define the surface evolution toward the solution. This allows a fine control over the evolution behavior since numerous control are available. The counterpart is that it jeopardizes the stability of the process because the resolution involves first and second derivatives of these terms (eq. (2.11)) that already contains second order derivatives. The consequences are twofolds; such a process is highly sensitive to noise and the produced surface is restricted to be at least twice differentiable.

The minimization is solved by local methods, a set of PDEs provided by the Euler-Lagrange equation. One common way is to use an iterative and steepest-descent method by considering a one-parameter family of smooth surfaces $\mathbf{x}(t) : (u,v,t) \mapsto (x(u,v,t), y(u,v,t), z(u,v,t))$ as a time-evolving surface $\mathbf{x}$ parameterized by the time t. The surface moves in the direction of the gradient of the functional $\mathcal{F}$ at point $\mathbf{p}$ according to the flow $\frac{\partial \mathbf{x}(u,v,t)}{\partial t} = -\nabla \mathcal{F}(\mathbf{p})$. This describes how each point on the dynamic surface moves in order to decrease the weighted surface. The final surface is then given by the steady state solution $\frac{\partial \mathbf{x}(u,v,t)}{\partial t} = 0$. The problem with this approach is well-known [191] in that it does not handle the changes in the topology because it would break the parameterization of the surface. Therefore, the optimization process requires an initial guess with the correct topology. It may also be sensitive to the coordinate system and to the parameterization of the surface (see Figure 2.20 on page 30) *i.e.* the functional is not intrinsic to the surface.

In the approach developed in [62], the regularization is introduced by considering a weighted minimal surface similarly to the approach exposed by Caselles *et al.* [36]. The weighted minimal surface is defined to be a minimizer of the functional (more details in Section 2.2.1 on page 13):

$$\iint w(\mathbf{x}) ds = \iint w(\mathbf{x}) \left\| \frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v} \right\| du\, dv$$

Again the minimization is solved by an iterative steepest-descent method. However, further development shows that the formulation is intrinsic, *i.e.*, independent of any chosen parameterization. It makes the use of level-set formulation possible. The level-set method [169, 191] regards the surface as the zero-level set of a higher dimensional function. The flow velocity is intrinsic (dependent only on the surface curvature). Topology, accuracy and stability of the evolution are handled by using the proper numerical schemes developed by Osher and Sethian [169, 191]. The main limitations of this optimization scheme is that it is not guaranteed to reach a global minimum (it may be stuck in a local trough) and it only deals with smooth surfaces *i.e.* it requires the surfaces to be at least twice differentiable.

Recently Boykov and Kolmogorov [26] shows that $\iint w(\mathbf{x})ds$ can also be minimized by a graph cut when $w(\cdot)ds$ is a Riemannian metric. Compared to the level-set method, thanks to its one-step design, this technique can cope with local minima. But on the other hand it is impossible to adapt the functional according to the current estimate, which is a common way to handle occlusions.

The connection between these two different formulations, one with a multiplicative regularization term and another with an additive one, has been studied by many researchers [36]. In the case of 2D curves, these two formulations correspond respectively to geodesic active contours and classical snakes. These two formulations are equivalent [36]. It seems that their application to 3D surfaces is still an open question. Nonetheless we show in Appendix A.1 on page 167 that they are equivalent for discrete scalar fields.

To define our functional, we consider a surface that is described by a depth field f such that $(u,v) \mapsto \mathbf{x}(u,v,f(u,v))$. In other words, the surface is defined by $z_{\mathbf{x}} = f(x_{\mathbf{x}}, y_{\mathbf{x}})$. If needed, several functions $f_1, f_2, \ldots$ can be used to parameterize the whole surface (this will be discussed later).

$$\iint \left(c(\mathbf{x}) + \alpha_u(u,v) \left| \frac{\partial z_{\mathbf{x}}}{\partial u} \right| + \alpha_v(u,v) \left| \frac{\partial z_{\mathbf{x}}}{\partial v} \right| \right) du\, dv \tag{2.12}$$

Only first derivative terms are used for smoothing. Dropping the second derivative smoothing terms is primarily due to the optimization method we introduce in the next section to solve the minimization problem. The second derivative terms also lead to a complicated solution (for instance, the Euler-Lagrange solution of deformable models would contain fourth derivative terms) and might produce over-smoothed surfaces. Therefore, we use only the first derivatives in our regularizing term. The issue is further discussed in the conclusions.

The L_1 norm is used to fulfill the smoothing objective and leads to an efficient computation. However, our work can be extended to the L_2 norm using the work of Ishikawa [99] with higher computation time. Our formulation is also not intrinsic, and is therefore dependent on parameterization. Note that α_u and α_v are not restricted to constants and thus make discontinuities recoverable because they allow local control of the smoothing term. Formally, it is possible to rigorously recover surfaces that are only piecewise differentiable *i.e.* piecewise C^1. To make a discontinuity possible at a given point $\mathbf{p}$, it is sufficient to set $\alpha_u(\mathbf{p}) = \alpha_v(\mathbf{p}) = 0$. So the influence of the derivatives is null at $\mathbf{p}$ and a discontinuity can appear.

Our approach lies between the geometric methods based on level sets [62, 138] or on deformable surface [210] that only reach a local minimum of the functional, and the graph-cut techniques [28, 32, 99, 101, 112, 120, 121, 179, 180, 219] whose convergence is controlled but whose goal is expressed with image-based quantities. We keep a continuous geometric formulation as [62, 210] but guarantee to result in a global minimum as [26, 32, 99, 101]. This allows to refine the precision while still solving the same problem.

Summary: We propose to solve the functional (C.2.3) based on the parameterization of the surface as a depth field. It regularizes the consistency goal by adding a regularizing term that involves only first derivatives. This functional is not intrinsic (*i.e.* depends on the chosen parameterization) but makes less assumptions about the reconstructed surface. Therefore it can recover surfaces with discontinuities. Furthermore, the proposed functional is purely geometric and does rely on any information about the image or space resolution. This last point offers the possibility to produce a surface solving the same problem at different resolutions which paves the way toward multi-resolution techniques.

2.4 General presentation of graph cuts

The core of the algorithm presented in this document is based on a graph cut. We give here a general presentation of this tool. This section ends with an explanation of the basic mechanism that we use to power our optimization process.

2.4.1 The graph-flow problem

It is a classical algorithmic problem [4, 69]. It states the following general problem: A water source is linked to a sink through a network of pipes that do not let more water go through than their capacity. What is the maximum water flow that can reach the sink or, equivalently, where is the network bottleneck that limits the flow? This correspondence between the maximum flow and the bottleneck is important to understand because it is the base property of our method: Finding the maximum flow is exactly equivalent to finding the set of the limiting pipes. With other words, the water flowing through the network can at most equal the bottleneck capacity and cannot exceed it.

History of* [Graph flow]*: *Schrijver [189] gives an interesting explanation of the history of this problem. It has been first motivated to study the railway transportation in the Soviet Union. In the 1930's, the Soviet side wanted to maximize its transportation plan. And in the 1950's, the US side wanted to optimize a potential attack of the Air Force to minimize the railway capacity. Both sides went to a similar graph problem. These two analyzes are an original interpretation to the max-flow/min-cut theorem that appears later in this section.*

Formally speaking, we model the problem with an oriented graph $\mathcal{G}$ (the pipe network) composed of a set $\mathcal{V}$ of vertices and a set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ of oriented edges. $\mathcal{V}$ contains two special vertices: the source s and the sink t. For each vertex $v \in \mathcal{V}$, we define the sets of entering and exiting edges:

$$\begin{aligned} \mathcal{E}_{\text{in}}(v) &= \big\{(u,v)/u \in \mathcal{V}\big\} \bigcap \mathcal{E} \\ \mathcal{E}_{\text{out}}(v) &= \big\{(v,w)/w \in \mathcal{V}\big\} \bigcap \mathcal{E} \end{aligned}$$

Two non-negative values are associated to each edge e: the capacity $Cap(e)$ (the maximum possible flow through a pipe) and the flow $Flow(e)$ (the actual flow in a pipe). From this, we impose two constraints:

$$\forall e \in \mathcal{E}, \qquad Flow(e) \leqslant Cap(e) \tag{2.13a}$$

$$\forall v \in \mathcal{V} \setminus \{s,t\}, \qquad \sum_{e_{\text{in}} \in \mathcal{E}_{\text{in}}(v)} Flow(e_{\text{in}}) = \sum_{e_{\text{out}} \in \mathcal{E}_{\text{out}}(v)} Flow(e_{\text{out}}) \tag{2.13b}$$

These rules express that the flow cannot exceed the pipe capacity (eq. (2.13a)) and that no water appears or disappears except in the source and in the sink (eq. (2.13b)). A flow satisfying these two rules is said *valid*. For now on, we only consider valid flows. To clarify the following explanations, we also assume that the source and the sink satisfy $\mathcal{E}_{\text{in}}(s) = \mathcal{E}_{\text{out}}(t) = \varnothing$.

Graph-flow problem: We define a graph flow as: $$Flow(\mathcal{G}) = \sum_{e \in \mathcal{E}_{\text{out}}(s)} Flow(e)$$

Given a graph $\mathcal{G}$ and a capacity function $Cap(\cdot)$, the graph-flow problem consists in finding a flow function $Flow(\cdot)$ that maximizes $Flow(\mathcal{G})$ while respecting the rules 2.13.

A cut $\mathcal{C}$ is a partition into two non-overlapping subsets: $\mathcal{V}_s$ that contains s and $\mathcal{V}_t$ that contains t. This defines the set of the cut edges that go from $\mathcal{V}_s$ to $\mathcal{V}_t$:

$$\mathcal{E}_{\text{cut}} = \left(\mathcal{V}_s \times \mathcal{V}_t\right) \bigcap \mathcal{E}$$

A value $Cap(\mathcal{C})$ is then associated to $\mathcal{C}$:

$$Cap(\mathcal{C}) = \sum_{(u,v)\in\mathcal{E}_{\text{cut}}} Cap(u,v)$$

More about* [Oriented edges]*: *Only source-to-sink edges are counted in the capacity of a cut. Therefore the orientation of an edge is crucial since it decides whether this edge belongs to a cut or not. It is possible to create an "unoriented" edge by associating two edges oriented both ways. So as long as the cut goes in between the two nodes of this "double edge", its capacity is added to the cut.*

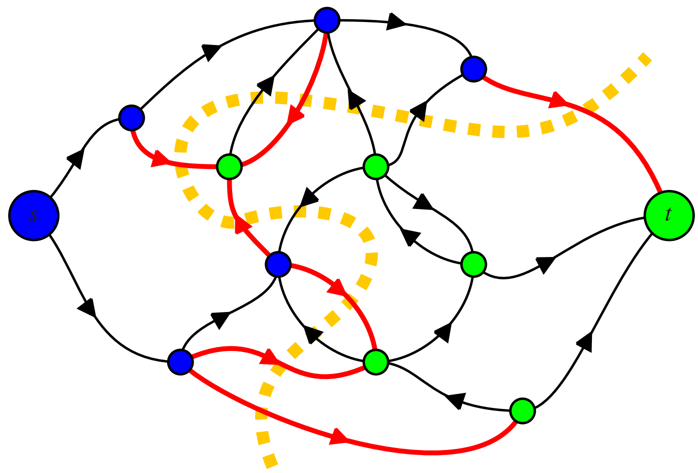


Fig. 2.23: Example of a cut on a graph

The dotted line shows a cut of the graph. The nodes in the $\mathcal{V}_s$ set are in blue and the ones in $\mathcal{V}_t$ in green. The cut edges are indicated in red (note that only the source-to-sink edges are cut).

Intuitively, a cut is a potential bottleneck that separates the pipe network into two parts and its capacity is the maximum flow that can go through. An actual bottleneck is a cut that has a minimal capacity among all the cuts. This leads to the main theorem.

Max-flow/min-cut theorem: For given graph as previously described, the maximum flow equals the minimum cut.

$$\max\ Flow(\mathcal{G}) = \min_{\mathcal{C}}\ Cap(\mathcal{C})$$

Using the water analogy, $\max\ Flow(\mathcal{G})$ is the flow through the network pipe and $\min_{\mathcal{C}} Cap(\mathcal{C})$ is the capacity of the bottleneck. Then one can get convinced that these two quantities are equal.

A complete formal proof is not very complex but out of the scope of our study. We give only the main steps to reach the theorem. First, it can be shown that the flow through all the cuts is the same (use rule (2.13b)). From this, using the rule (2.13a) gives that the max flow is less or equal to the min cut. The converse relation is trickier and introduces additional entities (the residual graph and the augmenting paths). One can read [4, 69] for details. □

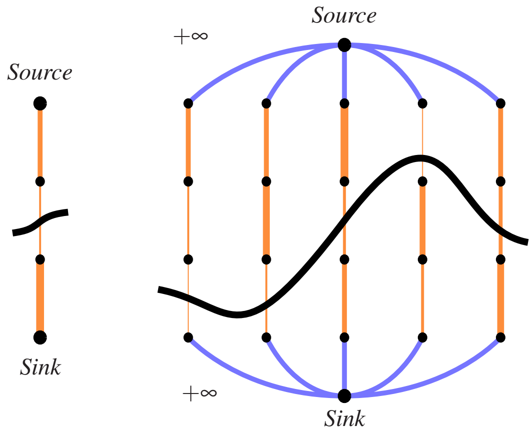


Fig. 2.24: Two simple examples of graph-cut optimization

Left: *A graph to find the minimum value among three.* ***Right:*** *A graph to find the minimum values of five sets of three.*

From a minimum cut and a maximum flow, it can also be shown that:

$$\forall e_{\text{cut}} \in \mathcal{E}_{\text{cut}}, \ \ Flow(e_{\text{cut}}) = Cap(e_{\text{cut}}) \tag{2.14}$$

In terms of pipes, it means that the bottleneck pipes are full *i.e.* no extra water can flow trough them. Thus, if a maximal flow is known, $\mathcal{V}_s$ is computed as the set of vertices that can be reached from s with only the edges such that $Flow(e) < Cap(e)$. Then, $\mathcal{V}_t = \mathcal{V} \backslash \mathcal{V}_s$. Property (2.14) guarantees that this defines a minimal cut. This may not be the only minimum cut, in that case, this corresponds to the closest one to s according to the graph distance.

The practical and main point of this graph problem is that polynomial algorithms exist [27, 39] to exactly compute a maximal flow. Then, from the theorem, we know that this flow corresponds to a minimal cut and finding such a cut is a standard graph search whose complexity is linear in the number of edges.

Hence, an exact minimal cut can be computed in polynomial time from a graph.

This provides an optimization engine that reaches an exact and global minimum of any problem that can be formulated as a graph flow.

2.4.2 Link with surfaces

At first sight, the link between this "water flow problem" and a functional defined over a surface may not be obvious. To help the reader to be familiar with this concept, two simple examples are described. None of these examples has a practical use, there are only provided for illustration purpose.

Consider a set of n non-negative values. Let's use a graph cut to find the minimum of this set. Take n edges with capacities corresponding to the n values. Link these edges to form a *string*. The source and sink are the extreme vertices. The configuration is shown on Figure 2.24-left for $n = 3$. It is straightforward to see that the minimal cut (the bottleneck) is the edge with the smallest associated value.

Consider now m sets of n non-negative values. Let's use a graph cut to find the minimum of each set in a single run *i.e.* without using m times the previous algorithm. For each set, build a string as previously described. Then create two additional vertices: the source and the sink. For each string, use an edge with infinite capacity to link the source to one extremity and another infinite edge to link the other extremity to the sink. Figure 2.24-right shows the graph for $n = 3$ and $m = 5$. The minimal cut cannot cross an infinite edge, it therefore crosses an edge of each string. Then it is straightforward again to see that the minimal cut crosses the edge with the minimum value of each string. Notice that this cut "looks like" a line.

More about* [Infinite edges]*: Any cut containing an infinite edge cannot be minimal. Therefore, infinite edges can be used as constraints to ensure that a minimal cut does lie in a given place. For instance, they can be used to create several sources and sinks as in Fig. 2.24-right.

Our strategy in the following section is to generalize this idea. Instead of considering a one-dimensional set of values that leads to a line, we use a two-dimensional set that results in a surface. Adding edges between adjacent strings will add constraints between neighbor points.

2.5 Global discrete solution

We expose in this section a technique to minimize the functional (C.2.3) defined in Section 2.3. The functional is first discretized in a classical way. We then describe a first graph that is sufficient to reach a first solution. From this graph, we lay down the main property and show the main limitation of the technique. We end with a second graph design that overcome this limitation.

2.5.1 Discretization

Without loss of generality, let's assume that the 3D object space is described by (x, y, z) and the object surface is locally parameterized by : $(u, v) \mapsto \mathbf{x}(u, v, f(u, v))$, where f is a function that can be seen as a depth field $z = f(x, y)$. Also let the domain on which f is defined be $\mathcal{D}$. This parameterization restricts the object being constructed to a single-valued depth field. If multiple depth values are needed, multiple functions $f_1, f_2, \dots$ could be used. Since x and y often play equivalent roles, the $x|y$ notation is used when a statement is applied to both x and y.

Our proposed functional consists of a consistency term $\mathcal{C}$ and a smoothing term $\mathcal{S}$:

$$\mathcal{C}(f) = \iint_{\mathcal{D}} c(x, y, f(x, y))\, dx\, dy \tag{2.15a}$$

$$\mathcal{S}(f) = \iint_{\mathcal{D}} \left(\alpha_x(x, y) \left| \frac{\partial f}{\partial x}(x, y) \right| + \alpha_y(x, y) \left| \frac{\partial f}{\partial y}(x, y) \right| \right) dx\, dy \tag{2.15b}$$

Our solution strategy relies on an approximation of equations (2.15) by a discrete formulation. Let's consider that the surface domain $\mathcal{D}$ is discretized as a regular rectangular grid. Then, x, y, z have values $\{x_1, \dots, x_{n_x}\}$, $\{y_1, \dots, y_{n_y}\}$, $\{z_1, \dots, z_{n_z}\}$ separated by Δx, Δy, Δz. The extension to a general domain $\mathcal{D}$ with varying discretization steps is however straightforward.

Discrete terms $\mathcal{C}^{\mathrm{d}}$ and $\mathcal{S}^{\mathrm{d}}$ are obtained as:

$$\mathcal{C}^{\mathrm{d}}(f) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} c(x_i, y_j, f(x_i, y_j)) \Delta x \Delta y \tag{2.16a}$$

$$\begin{aligned} \mathcal{S}^{\mathrm{d}}(f) = & \sum_{i=1}^{n_x - 1} \sum_{j=1}^{n_y} \alpha_x(x_i, y_j) \left| f(x_{i+1}, y_j) - f(x_i, y_j) \right| \Delta y \\ & + \sum_{i=1}^{n_x} \sum_{j=1}^{n_y - 1} \alpha_y(x_i, y_j) \left| f(x_i, y_{j+1}) - f(x_i, y_j) \right| \Delta x \end{aligned} \tag{2.16b}$$

2.5.2 Building a first embedded graph

Our approach is based on a topologically embedded graph in the 3D geometric space; that is to say, it can be seen as a 3D geometric entity. Nodes and edges correspond to 3D points and 3D segments. Therefore, a cut is a real surface that crosses these segments. Moreover, edge capacities are fully expressed with geometric measurements. The main idea is to build the graph such that the cut capacity is equal to the surface functional. Then, a minimal cut will be a solution to the discrete problem. The graph is a 3D grid superimposed on the voxels with correspondence shown in Figure 2.25. All edges are bidirectional (*i.e.* made of two directional edges): for x-edges, the capacity is $\alpha_x(x_i, y_j)\Delta y \Delta z$; for y-edges, $\alpha_y(x_i, y_j)\Delta x \Delta z$; and for z-edges, $c(x_i, y_j, z_k)\Delta x \Delta y$. We add source and sink nodes out of this grid and for each (x_i, y_j) voxel column, the voxel with the minimum depth is linked to the source and the one with the maximum depth is linked to the sink. All source and sink edges have an infinite capacity. From a complexity point of view, there are three edges and one 6-connected node per regular voxel (*i.e.* not on a border). Thus, the graph complexity is proportional to the number of voxels.

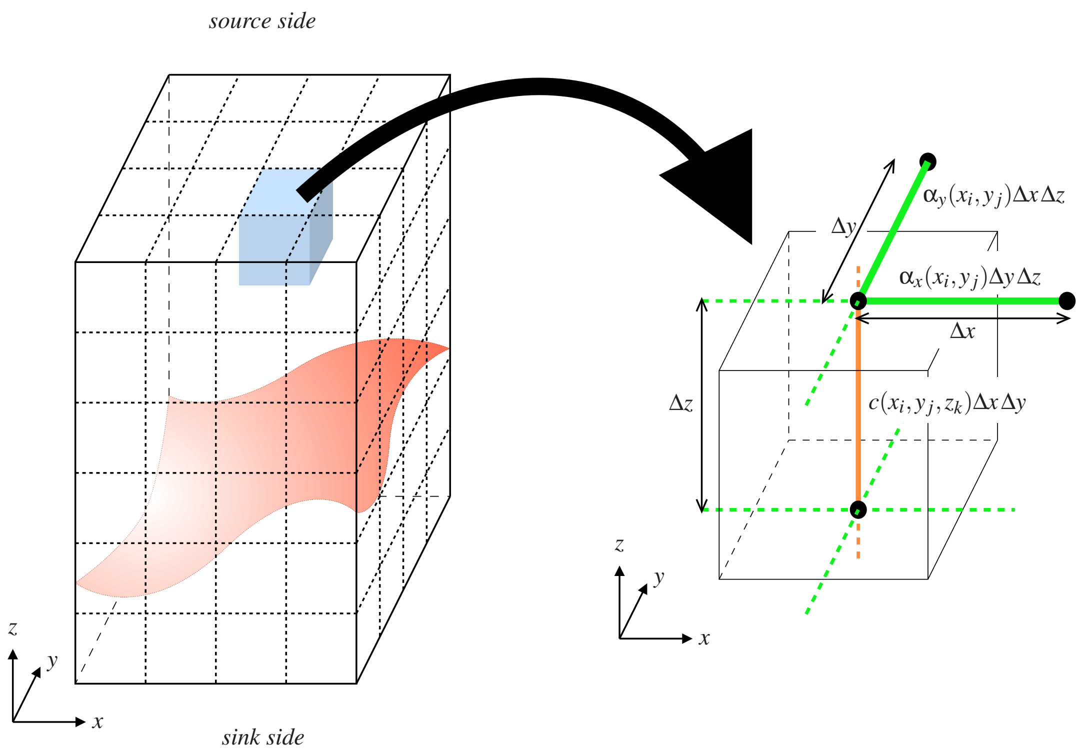


Fig. 2.25: Graph with a linear smoothing term

Left: *voxel grid with a surface inside.* ***Right:*** *Correspondence between the voxel (x_i, y_j, z_k) and the graph.*

Note that Roy [179] and Ishikawa [99, 100] have also described embedded graphs with a design rather similar to this first graph. Roy embeds his graph in the disparity space which almost equivalent to the geometric space. However disparity is inherently linked to images. This makes the relation between the produced shape and the functional unclear. Ishikawa embeds the graph in a generic Euclidean space but the links with the geometric space are unclear. In contrast to these approaches, our embedding is stronger: Our graph is a 3D entity characterized by a 3D quantities (*e.g.* geometric lengths) similarly to the method of Boykov and Kolmogorov [26].

2.5.3 Establishing a correspondence between a minimal cut and a surface

We here demonstrate the following property which proves that the previously built graph exactly computes a global minimum of the functional.

Correspondence property: There is an one-to-one correspondence between a subset of cuts called the *potential minimal cuts* and the surfaces defined by a function f. Moreover, the cut capacity is equal to the functional value of the corresponding surface.

The proof is based on necessary criteria for a cut to be minimal and a careful count of cut edges.

There are three necessary conditions for a cut to be minimal:

1. A minimal cut cannot cross an infinite edge;
2. It has to cross each (x_i, y_j) column at least once to separate the source from the sink;
3. It cannot cross a column more than once for the capacity would be higher than the one-crossing case.

A cut satisfying these three conditions is called a *potential minimal cut*. A direct one-to-one correspondence between such a cut and a surface exists: The cut is limited to the consistent voxel space (condition 1) and the single crossing point on the (x_i, y_j) voxel column (conditions 2 and 3) unambiguously determines $f(x_i, y_j)$.

The capacity of a potential minimal cut represented by f_{cut} can be computed as follows. The capacity of the crossed z-edges is exactly the consistency term $\mathcal{C}^{\text{d}}(f_{\text{cut}})$ because each (x_i, y_j) voxel column contributes with $\alpha_x(x_i, y_j)\Delta y \Delta z$. Then, if we consider the two adjacent columns (x_i, y_j) and (x_{i+1}, y_j), the cut depths are $f_{\text{cut}}(x_i, y_j)$ and $f_{\text{cut}}(x_{i+1}, y_j)$. This implies that the number of crossed x-edges is:

$$\frac{1}{\Delta z}\left|f_{\text{cut}}(x_{i+1}, y_j) - f_{\text{cut}}(x_i, y_j)\right|$$

Thus, the total capacity of crossed x-edges is:

$$\alpha_x(x_i, y_j)\left|f_{\text{cut}}(x_{i+1}, y_j) - f_{\text{cut}}(x_i, y_j)\right|\Delta y$$

There is a similar result for the y-edges. The total capacity of all the x-edges and y-edges crossed by the whole cut is exactly $\mathcal{S}^{\text{d}}(f_{\text{cut}})$.

By adding these results together, we draw the expected conclusion: there is an exact correspondence between a surface with the discrete functional $\mathcal{C}^{\text{d}} + \mathcal{S}^{\text{d}}$ and a potential minimal cut with its capacity. □

Therefore, any graph-flow algorithm is able to reach in polynomial time a global minimum of this functional by using this graph.

2.5.4 Analyzing the smoothing term

Graph-cut techniques [28, 32, 99, 101, 112, 121, 219] often yield flat and blocky results. This may not be fatal if we are only building a disparity map with limited precision (*e.g.*, 16 or 32 disparity values), but it becomes crucial if we target 3D shape reconstruction including small details and smooth slopes. We first elucidate the origin of this artifact, and then propose a solution by introducing a new smoothing term.

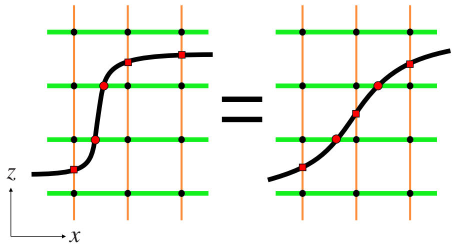


Fig. 2.26: Equivalence between a discontinuous variation and a continuous one

A discontinuous variation and a continuous one are equivalent in a region with uniform consistency term. They cross the same number of x-edges (2 circles) and z-edges (3 squares). For clarity purpose, we present a 2D xz plane of the 3D grid.

This artifact appears in regions with depth variation and rather uniform smoothing and consistency terms. Consider the following simple example (restricted to 2D for clarity) where f is monotonic between x_A and x_B and $\alpha(x)$ is constant and equal to $\bar{\alpha}$:

$$\mathcal{S}_{2\mathrm{D}}(f) = \int_{x_A}^{x_B} \alpha(x) \left| \frac{\partial f}{\partial x}(x) \right| dx = \bar{\alpha}\, |f(x_B) - f(x_A)|$$

If $\alpha(x)$ is uniform then the smoothing term only depends on the extreme values of f ignoring its actual shape (Fig. 2.26). Therefore, if the consistency term is also uniform in that region, a continuous depth change and a discontinuous one yield the same functional value. This directly stems from the linear dependency of the smoothing term on the derivative. A convex term makes a continuous variation have a lower functional and a concave one makes it have a higher functional. Now the situation boils down to three cases:

1. A convex smoothing term: It favors continuous variations but it may impede real surface discontinuities.

2. A concave smoothing term: It creates spurious discontinuities. Smoothing the surface would not be a solution because it would also remove the real discontinuities.

3. A linear smoothing term: It causes ambiguities because several surfaces have the same functional value. Unfortunately, it can be shown that, between several equivalent cuts, a graph-cut algorithm always “chooses” the most discontinuous one because it is the closest to the source or to sink. This leads to same artifacts than the concave case. (See Figure 2.27.)

This is summarized by the following property.

Smoothing term property: Concave and linear smoothing terms introduce spurious discontinuities on the surface. To avoid this artifact, the smoothing term must be strictly convex.

From this analysis, we do not use a concave smoothing term like [18, 121] neither a linear one like [32] because both introduce spurious discontinuities. We use a convex smoothing term. But as discussed in the Section 2.2 this may limit our approach to small depth variations because large ones would be heavily penalized. To overcome this point, we make use of the α_x and α_y functions to control the smoothing term and allow real discontinuities to occur (this point is discussed later).

Fig. 2.27: Comparison between a linear and a convex smoothing term (influence on 3D results)

A linear smoothing term (left) yields blocky results whereas a convex one (right) produces smooth shapes.

2.5.5 Building a second graph with a convex smoothing term

Applying the previous analysis to our first graph design (Fig. 2.25), the ambiguity inherent to the linear smoothing term is shown in Figure 2.26. This graph therefore needs to be adapted to incorporate a strictly convex component. We could have chosen the graph proposed in [99] but it introduces some constant in the smoothing term that can be difficult to handle in a multi-resolution context. We therefore propose a new graph based on the correspondence shown in Figure 2.28. We conserve the global 3D grid layout of the graph but there are now two kinds of $x|y$-edges: the $\alpha_{x|y}$-edges and the $\beta_{x|y}$-edges and the z-edge is split into 8 sub-edges. Nonetheless, the graph complexity is still proportional to the number of voxels since there are twelve edges, four 3-connected nodes and one 12-connected node per voxel. It adds an additional term $\mathcal{A}^{\mathrm{d}}$ in the functional:

$$\begin{aligned}\mathcal{A}^{\mathrm{d}}(f) = &\sum_{i=1}^{n_x-1}\sum_{j=1}^{n_y} \beta_x(x_i,y_j)\left[\left|f(x_{i+1},y_j)-f(x_i,y_j)\right|-\Delta z\right]^{+}\Delta y \\ &+\sum_{i=1}^{n_x}\sum_{j=1}^{n_y-1} \beta_y(x_i,y_j)\left[\left|f(x_i,y_{j+1})-f(x_i,y_j)\right|-\Delta z\right]^{+}\Delta x \end{aligned} \tag{2.17}$$

with $[\lambda]^+ = \max(0,\lambda)$.

Let's check that the correspondence property is still satisfied. The z-sub-edges are always cut by a group of four and therefore their total capacity is always equal to the consistency term $\mathcal{C}^{\mathrm{d}}(f_{\mathrm{cut}})$. Then, the $x|y$-axis, as the $\alpha_{x|y}$-edges correspond to the previous $x|y$-edges, always form $\mathcal{S}^{\mathrm{d}}(f_{\mathrm{cut}})$. Only the $\beta_{x|y}$-edges remain. Consider again two adjacent columns (x_i,y_j) and (x_{i+1},y_j). As previously shown, the number of crossed α_x-edges is:

$$\frac{1}{\Delta z}\left|f_{\mathrm{cut}}(x_{i+1},y_j)-f_{cut}(x_i,y_j)\right|$$

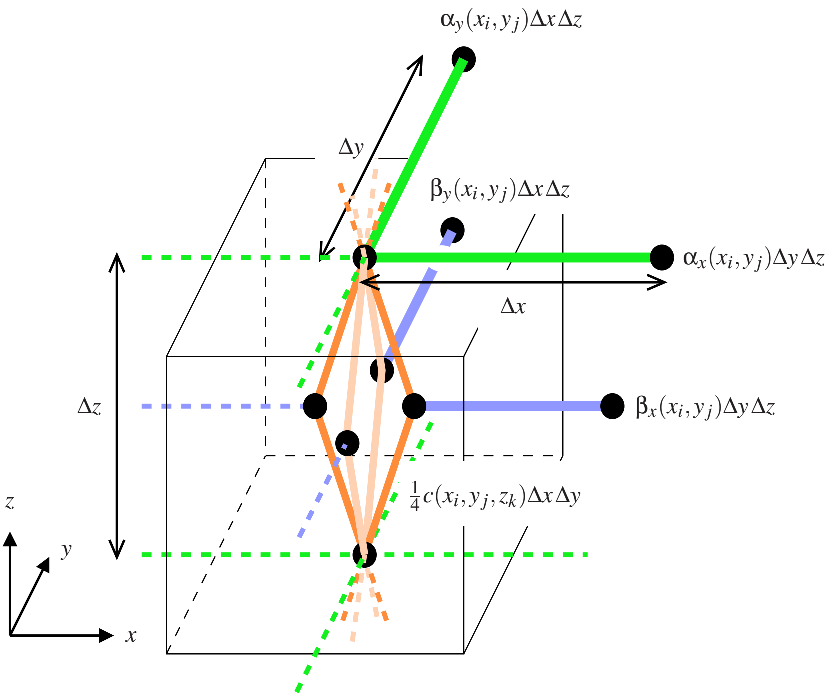


Fig. 2.28: Graph with a convex smoothing term

Correspondence between the voxel (x_i, y_j, z_k) and the graph with a convex smoothing term. The 8 z-sub-edges have the same capacity.

If this number is non-zero, because the cut can "go in the middle" of the z-sub-edges as seen in Figure 2.29, there is one less β_x-edge crossed. The capacity of the crossed β_x-edges between the columns is:

$$\beta_x(x_i, y_j)\left[\left|f(x_{i+1}, y_j) - f(x_i, y_j)\right| - \Delta z\right]^+ \Delta y$$

Hence, $\beta_{x|y}$-edges form $\mathcal{A}^{\mathrm{d}}$ which is convex because of the $[\,\cdot\,]^+$ function. □

As expected, with this convex term, the functional is lower for a continuous variation than for a discontinuous one as illustrated in Figure 2.29. This achieves our goal but the discrete $\mathcal{A}^{\mathrm{d}}$ term (2.17) has no continuous counterpart like $\mathcal{C}^{\mathrm{d}}$ with $\mathcal{C}$ and $\mathcal{S}^{\mathrm{d}}$ with $\mathcal{S}$ because of the Δz term, which is directly linked to the discretization step. However, $\beta_{x|y}$ can be made very small because we only need to penalize the step variations, no matter how important this penalty is. Therefore, $\mathcal{A}^{\mathrm{d}} \approx 0$. Then, we can compute a very close approximation of the discrete solution to the continuous functional while avoiding spurious discontinuities.

The last point to be observed is that crossing two z-sub-edges may lead to a lower capacity than crossing one β-edge. To avoid this situation, we can either add a constant on the z-sub-edge capacity (but it can be incompatible with a multi-resolution approach) or use *constraint edges* [99, 100] (*i.e.* keep the same capacity for the oriented z-edges from the source to the sink and assign an infinite capacity to their reverse oriented edges) and then have no additional constant.

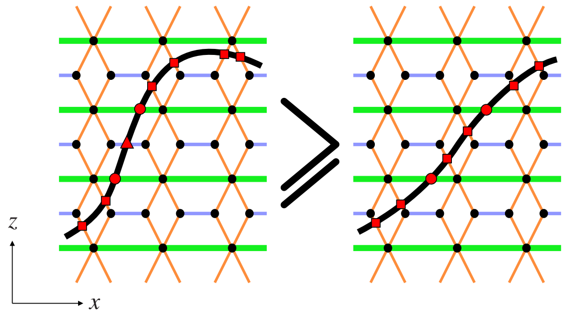


Fig. 2.29: Discontinuous variation and a continuous one are no more equivalent

The discontinuous variation crosses one more β_x-edge (the triangle) than the continuous variation. Thus, it has a higher functional value.

Summary: We have exposed a method to minimize a discretized version of the functional (2.15). The underlying algorithm runs in polynomial time and the discretization step can be arbitrarily chosen. The advantage of this technique is that it reaches a global minimum. It also avoids spurious discontinuities when several such minima exist. The counterpart is that it requires two functions α_x and α_y to be set in order to properly deal with depth discontinuities. This point is addressed in the following section.

2.6 Practical algorithm

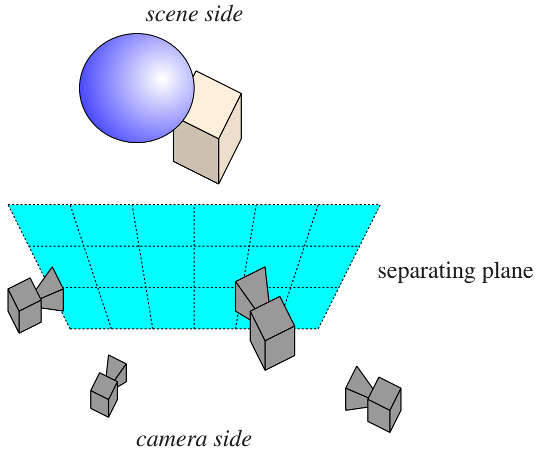


Fig. 2.30: Configuration of our scenario.

In the previous sections, we have described a general optimization tool. It has now to be adapted in a surface reconstruction framework. To demonstrate the capacity of this method, we have chosen a classical scenario which is well-known to raise precision difficulties and therefore challenges our technique. In the chosen configuration, we assume that there exists a separating plane: all the cameras are in the same half space and look toward the other half. This is a classical setup which is similar to many others [26, 32, 120, 121, 190]. This corresponds for instance to a short video sequence or to the situation in which it is impossible to go all around the scene. These two practical cases motivate this setup. We also assume that there is no moving object in the scene and since we are focusing on the reconstruction issue, the cameras are considered fully calibrated.

The proposed algorithm mixes voxels and graph cut in a consistent way since both rely on the same space discretization. The advantage of this design is to deal efficiently with the visibility. Self-occlusions and object-by-object occlusions are handled: They are detected and if an object is partially hidden but still visible from a subset of cameras, it is reconstructed.

Before examining the details, we first give an overview of the algorithm illustrated by the bloc diagram in Figure 2.31.

Input The algorithm uses a sequence of images, each image is given with its corresponding projection matrix which relates a 3D point to its projection in the image plane. The user also defines a bounding volume *e.g.* a 3D bounding box, containing the object of interest.

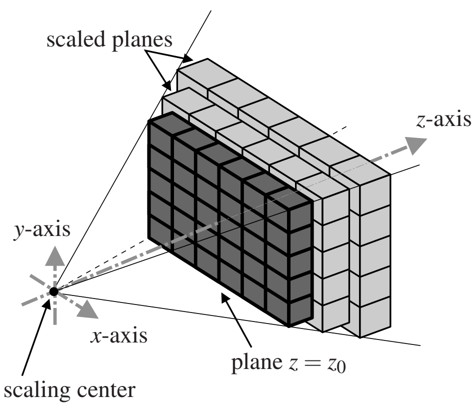


Fig. 2.32: Configuration of our study case.

Initialization Before entering the core of the process, the voxel space is built. The specific layout of the voxels (Fig. 2.32) ensure several properties including a new one which is proved. This allows a rigorous characterization of the surface boundaries.

Main loop The algorithm is based on a multi-pass process. Each object is reconstructed one by one while visibility is progressively updated. Each pass first collects consistency information and localizes the potential discontinuities. Then the graph-cut optimization is run: it will be shown how to use it while accounting for self-occlusions. Before a new pass starts, visibility is updated (occluded regions are marked in images) so the next passes handle object-by-object occlusions.

Post-process The resulting mesh is aliased because of the discrete representation of space. This artifact is removed with a specific PDE filter.

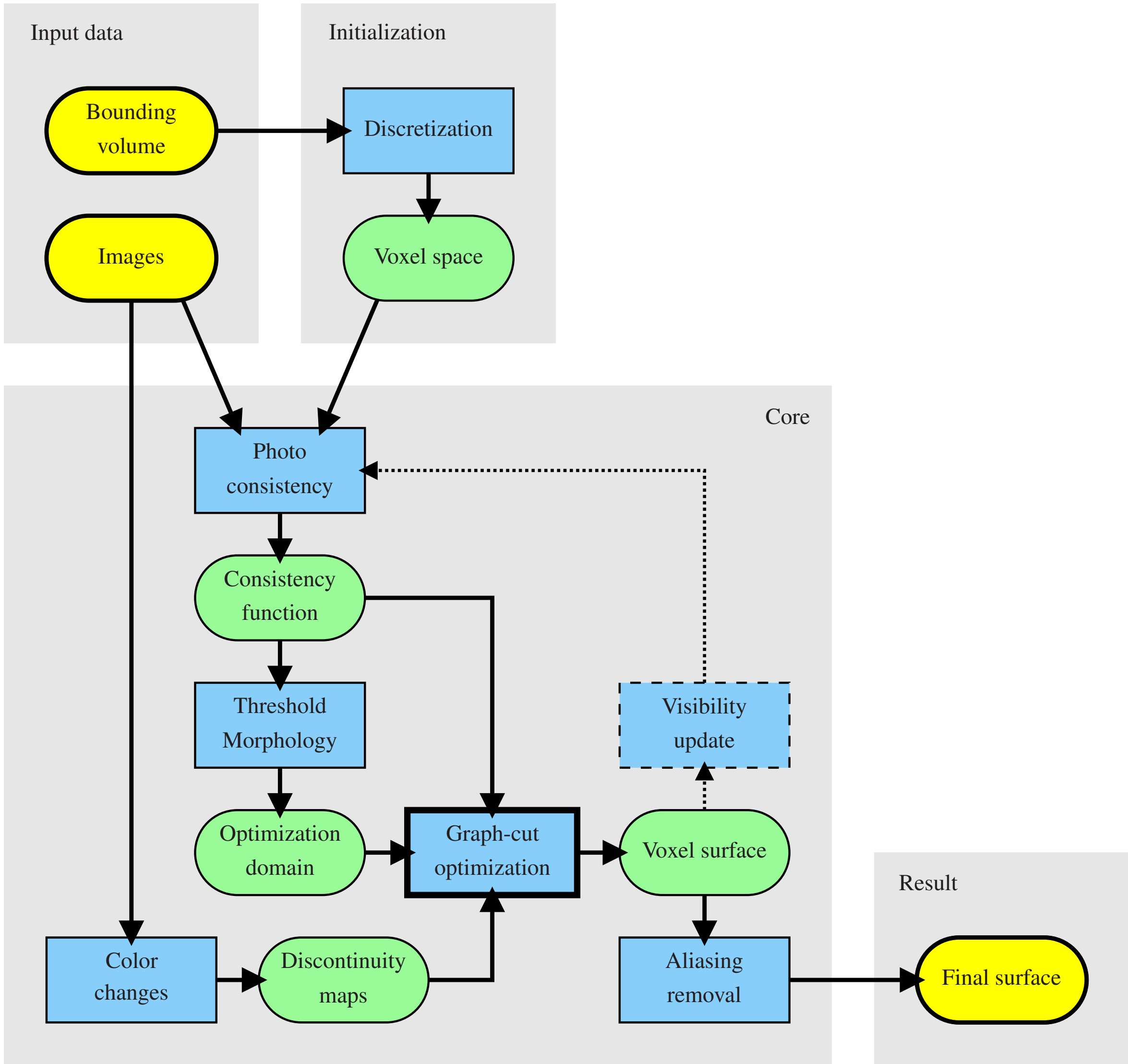


Fig. 2.31: Overview of the whole process

The core of the algorithm may be iterated as long as there are objects to reconstruct in the scene. In this case, the previously reconstructed surface is taken into account to update the visibility.

2.6.1 Initialization

The input bounding volume is discretized. This discrete space is seen in turn as a voxel space and a graph (see Figures 2.25 on page 45 and 2.28 on page 49). The space axes are determined as follows. The x-axis is defined by any two arbitrary cameras not orthogonal to the separating plane, the z-axis is orthogonal to x and going through the plane and $y = z \times x$. We then use a voxel configuration illustrated in Figure 2.32 on the page before. The voxels are organized by planes $z = cst$: each plane is a scaled version of the $z = z_0$ plane. It fulfills the *constant footprint property* [195]: Each voxel has the same projected area. There are other ways to achieve this property such as Saito and Kanade [185] and Szeliski and Golland [205] but this configuration also induces the *vertex coordinate property* which characterizes the surface boundary $\mathcal{D}$. We first demonstrate this property in a simple case and then extend it to a more general configuration.

Vertex coordinate property It characterizes the voxels near a surface of uniform color. As illustrated in Figure 2.33, a voxel near such a surface but not part of it can be consistent because all the lines of sight hit the surface and give the same color. Therefore, as shown in [127], surfaces uniform relatively to the consistency criterion used to define the c function (*e.g.* uniform color for photo-consistency) generate consistent regions which are bigger than the real surface. These regions go larger with smaller baselines. Therefore in our case, these regions are poor approximations of the real surface (Fig. 2.34-a). Nonetheless, these regions lead to useful properties.

In the following discussions, we assume that the surface color changes with its orientation relatively to the light and that the lighting is non-uniform. Conversely this implies that an uniform color surface is smooth and, for instance, cannot be peaked since it would appear of different colors because of the shading variations. We believe this hypothesis is always met by real scenes and therefore does not restrict our results.

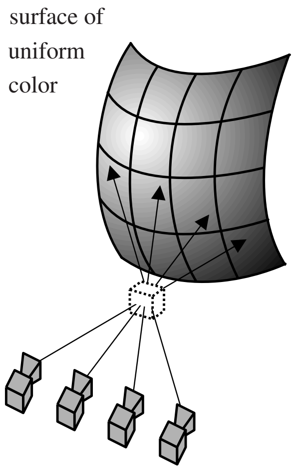


Fig. 2.33: Consistent voxel not part of the surface.

Basic vertex coordinate property: For aligned cameras, if the scaling center used to build the voxels lies between the two extreme cameras, in an epipolar plane (*i.e.* a plane containing the optical centers) a surface of uniform color forms a region with consistent voxels. This region is a quadrilateral which vertices have one and only one extremal coordinate relatively to the voxel-grid coordinate system. Moreover, the left-most and right-most vertices belong to the surface.

Sketch of proof We here give a short explanation of this result. The details would be tedious and result straightforwardly from the following main ideas. The proof relies on the relationship between the camera positions, the angles between the lines of sight and the voxel boundaries, and the slopes of these lines of sight in the voxel coordinate system. This is illustrated in Figure 2.34-a. The leading idea is that the C_1 camera generates lines of sight that have a positive slope in the voxel coordinate system. This slope is linked to the angle between the lines of sight and the borders of the voxel space; for C_1, they are always negative since C_1 lies on the left of the scaling center. Since C_2 lies on the other side, the same arguments lead to a negative slope of its lines of sight. Then examining the four vertices of the quadrilateral formed by the extremal lines of sight demonstrate the property. □

Remark Rigorously speaking, this property assumes that the surface is smooth enough so that the end points of the surface are the left and right vertices of the quadrilateral (see Figure 2.34 on the next page). One can imagine an extremely peaked surface for which this would not be true. But as previously mentioned, such a surface is unlikely to have a uniform color because of the strong shading variations induced by the peak. Practically, in such a case, the surface is made of several subregions of uniform color. The property is then valid for each subregion.

This first property can be generalized to the case of non-aligned cameras as long as a separating plane exists. This results in the following property.

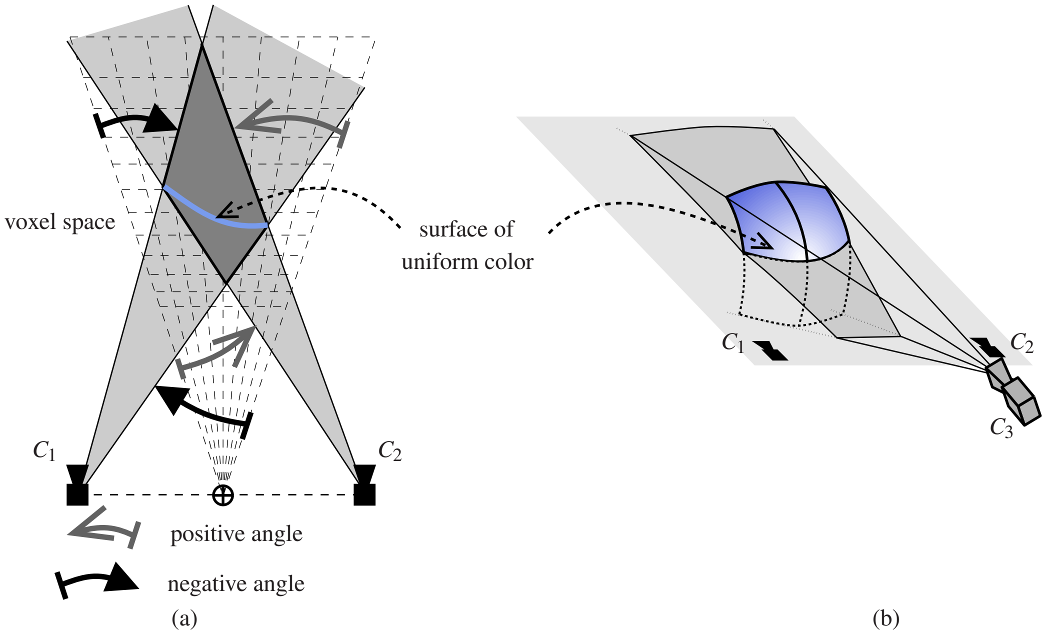


Fig. 2.34: Vertex coordinate property

(a) *An epipolar plane of C_1 and C_2. The intersection of the two visual cones defines the consistent region. Note that the left-most and right-most vertices belong to the surface.* ***(b)*** *The consistent region in the epipolar plane due to a third camera C_3. Figure 2.37 on page 56 shows such regions on a real example.*

Extended vertex coordinate property: If the scaling center used to build the voxel grid lies on a segment linking any two cameras C_i and C_j, in the epipolar planes of C_i and C_j, the consistent voxels form 2D regions which left-most and right-most points belong to the surface.

Proof Without loss of generality, consider two cameras C_1 and C_2 and name the C_1C_2 axis the x-axis, and then, as described previously, the z-axis is an orthogonal axis going through the separating plane and $y = z \times x$. The scaling center lies between C_1 and C_2. The configuration of an epipolar plane is illustrated in Figure 2.34-a.

We first restrict to the C_1 and C_2 cameras. Using the basic vertex coordinate property, we know that the left-most and right-most vertices of the corresponding consistent region belong to the surface.

Consider now one more camera C_3 lying in a general 3D position. It makes a visual 3D cone with the surface of uniform color and then forms a 2D region in the epipolar plane as depicted in Figure 2.34-b. C_3 brings an additional constraint for the consistency, therefore the region consistent with C_1, C_2 and C_3 is the intersection between the initial quadrilateral and the new C_3 region. The resulting region may not be a quadrilateral or even a polygon anymore because C_3 may create a non-polygonal shape (Fig. 2.34-b). Nevertheless, since the previously characterized left-most and right-most points are part of the generating surface, they are consistent for any camera which sees them. Hence they are part of the region consistent with C_1, C_2 and C_3 and because this region is only a restriction of the region for C_1 and C_2, they are still the left-most and right-most points. C_3 does not affect the property: the left-most and right-most points have not changed. This will therefore not change for any number of additional cameras. This demonstrates the extended property. □

Thanks to this property, the optimization domain $\mathcal{D}$ is now defined. It is sufficient to chose two cameras C_i and C_j. The voxel space is then build as previously described: The scaling origin is set between C_i and C_j, the x axis is along C_iC_j, z goes through the separating plane (it is necessary to ensure the configuration of the angles in Figure 2.34-a) and $y = z \times x$. With this setup (Fig. 2.32), $\mathcal{D}$ is characterized by its left-most and right-most points in each epipolar planes (defined by C_iC_j).

2.6.2 Main loop

The algorithm has a multi-pass layout to reconstruct objects one by one, from the closest one to the most distant. The separating plane defines unambiguously this "distance". It is a classical *front-to-back* approach [217] or it can also be seen as a *plane sweep* [127] in the z direction.

The iteration stops when no object remains unreconstructed in the scene.

Loop step 1: Consistency computation

For each voxel, a consistency value is computed (*e.g.* from ZNCC or photo-consistency). Then the voxels are thresholded to discard all the inconsistent voxels. Since we deal with real images, the resulting voxel set may contain some holes or isolated voxels. To overcome this point, we use mathematical morphology (Figure 2.35 defines the basic operators we use). The set of the consistent voxels is robustified with morphological operators

How to set* [Threshold]*: *The consistency threshold drives the detection of the objects. It is hard to give a value for this threshold since it depends both on the consistency criterion used and on image quality e.g. noisy images require a more tolerant (higher) threshold. However, if the produced objects are too big (including some spurious extensions from the background) or too small (missing some parts), this threshold is likely to be at the origin of the problem. Once a correct segmentation is achieved, this value has not to be changed anymore.*

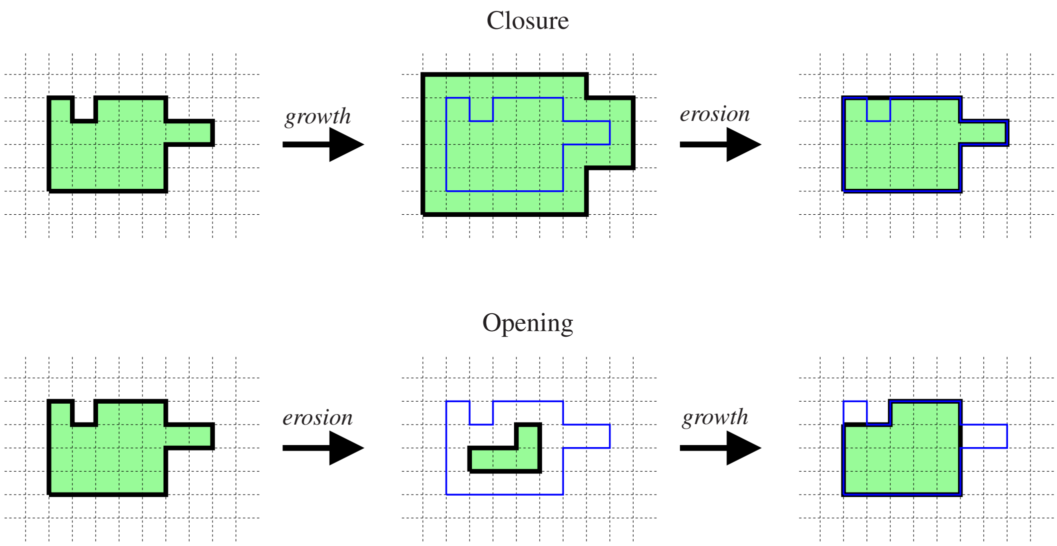


Fig. 2.35: Basic morphological operators

The operators are shown in the 2D case for clarity. For a closure (top row), the voxel set first grows (a layer of voxel is added at the boundary of the set) and then is eroded (the boundary layer is removed). This "fills" the holes of the set. For an opening, the set is first eroded and then grown. This removes the isolated voxels. More important effects can be achieved with thicker layers (adding or removing groups of voxels thicker than one voxel). (The original set is represented in blue.)

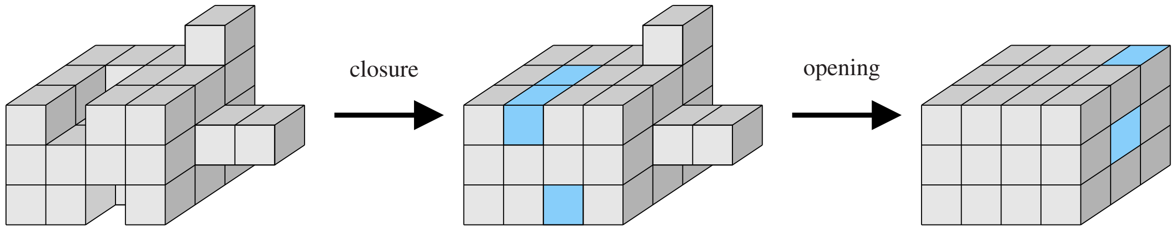


Fig. 2.36: Morphological treatment applied to the consistent voxels

The closure fills the holes. The opening removes the isolated voxels. (In blue, the changed regions.)

as described in [161]: A closure is first applied to fill in the holes and then an opening to remove isolated voxels (see Figure 2.36).

In the rest of the pass, only the set of consistent voxels that is the closest to the cameras is considered.

As illustrated by Figure 2.33 on page 53, this set of voxels is too large: There are consistent voxels that do not belong to the object surface. The problem is now to retrieve the object within this set of voxels.

This step does not only reduce the search space. It also characterizes the boundary of the surface to be built. It is important to remark that this boundary can be complex *e.g.* it can contain holes. This step drives therefore the topology of the final result.

How to set* [Morphological operators]*: If some spurious small objects appear "floating" around the main objects, it comes from the morphological treatment which is not strong enough.

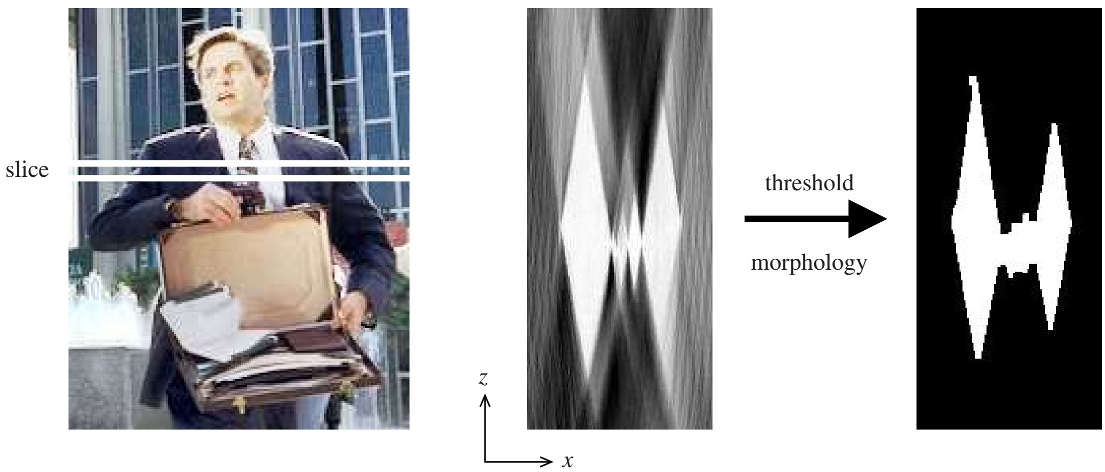


Fig. 2.37: An horizontal slice through the voxel space

***Left:** Position of the slice. **Middle:** The consistency values (high consistency is brighter) **Right:** The set of the consistent voxels determined after thresholding and morphological operators. Remark how uniform areas in the images (e.g. the jacket) creates large consistent regions in space. The size of these regions depends on the baseline (see Fig. 2.34 on page 54).*

Loop step 2: Discontinuities

Under the previously discussed and reasonable assumption that the aspect of a surface depends on its orientation, surface discontinuities result in image discontinuities. This remark is used to control the two functions α_x and α_y that appear in the smoothing term (eq. 2.15b on page 44). These functions drive the continuity of the surface: If they are null, the surface is free to be discontinuous whereas for positive values, discontinuities are penalized. Furthermore, this control over the smoothing term is mandatory to properly handle depth discontinuities because we use a convex term as motivated in Section 2.5.4 on page 46 (a non-convex one is likely to generate spurious discontinuities) but if this term is not controlled, it would hinder depth variations for they would be over-penalized.

> ***Link with* [Thresholded penalty]*:*** *Discontinuities can be handled by setting a maximum penalty applied to a depth variation (see Sec. 2.2.6 on page 26). The discontinuities are then the points where the penalty reaches this maximum. It means that the optimization engine "decides" where the discontinuities are. Our approach is different in that the discontinuities are detected a priori. The engine then "decides" to ignore some potential discontinuities. In the former case, the engine is free to make the surface discontinuous; in the latter, it is free to make it continuous.*

We propose to define these functions as $\alpha_{x|y}(x,y) = A\chi_{x|y}$ where A is a constant which corresponds to the desired smoothing strength when there is no discontinuity and the functions $\chi_{x|y}$ is a *discontinuity factor* varying between 0 (discontinuity) and 1 (no discontinuity). $\beta_{x|y}$ is defined similarly using $B\chi_{x|y}$ with constant B much smaller than A *i.e.* $B \ll A$.

$$\alpha_{x|y}(x,y) = A\,\chi_{x|y} \quad \text{and} \quad \beta_{x|y}(x,y) = B\,\chi_{x|y} \qquad \text{with} \quad B \ll A \tag{2.18}$$

We now expose how to localize potential surface discontinuities. The input images are used for this purpose: The color of each surface point is estimated relatively to the input images. But since this step occurs before the graph-cut optimization, we use the voxel approximation which a poor geometric approximation but is sufficient for this task. In practice, we compute the color of each voxel

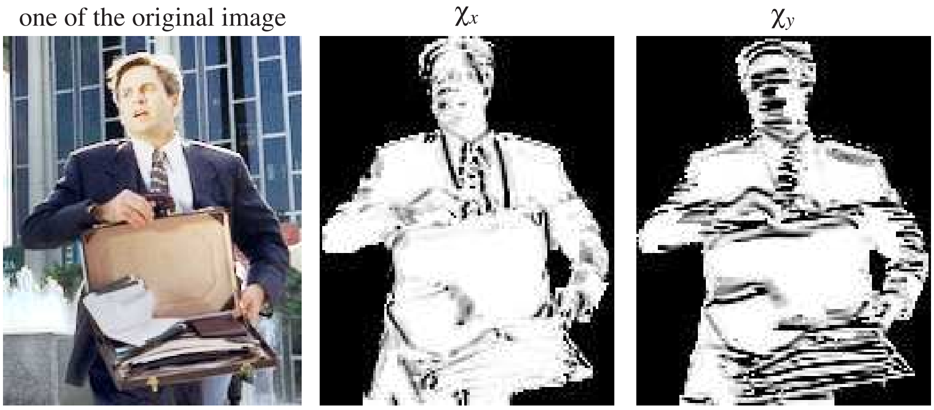


Fig. 2.38: Sample χ_x and χ_y functions

They corresponds to the briefcase man 2.43 on page 63. The gray levels go from black = 0 to white = 1. Note that all the actual discontinuities are included in the maps. There are also spurious lines but the optimization process can still produce a continuous surface at these locations (see the text for discussion of this point).

by averaging its color in each camera and the color of the surface at a point (x_i, y_j) is the average of the colors of all the voxels in the corresponding column. This estimation is reasonable because as discussed previously, depth uncertainty (*i.e.* several voxels in a (x_i, y_j) column) results from surfaces of uniform color. Therefore, only similar colors are averaged in this estimation.

Two *discontinuity maps* D_x and D_y are computed. D_x is related to the color difference between (x_i, y_j) and (x_{i+1}, y_j) (or (x_i, y_{j+1}) for D_y). As the estimation of the color discontinuities has to be independent of the local contrast (for instance, the estimation should not change between bright or dark lighting conditions), it is normalized by the standard deviation $\gamma_{\mathcal{N}_x^{ij}}$ of the color in a small neighborhood $\mathcal{N}_x^{ij}$ (*e.g.* 6×5 centered on $(x_{i+\frac{1}{2}}, y_j)$). Small values of $\gamma_{\mathcal{N}_x^{ij}}$ cause numerical instabilities and make detect spurious discontinuities. We therefore apply a threshold Γ according to the values of $\gamma_{\mathcal{N}_x^{ij}}$:

How to set* [Regularizing term]*: *It is controlled through the values A and B in equation (2.18). If the surface is too "flat", lowering these values overcome this points. On the other side, if the surface is irregular, raising the values corrects the problem.*

$$D_x(x_i, y_j) = \begin{cases} 0 & \text{if } \gamma_{\mathcal{N}_x^{ij}} < \Gamma \\ \frac{\text{distance}(\text{color}(x_i, y_j), \text{color}(x_{i+1}, y_j))}{\gamma_{\mathcal{N}_x^{ij}}} & \text{else} \end{cases}$$

We use the equivalent formula for $D_y(x_i, y_j)$ to get:

$$\chi_{x|y}(x_i, y_j) = \left[1 - D^2_{x|y}(x_i, y_j)\right]^+$$

Our approach is different from existing methods. Some of them [32, 62] consider discontinuities as normal depth variations. This is obviously a drawback for important depth changes. Or the others handle discontinuities in their optimization process but without controlling their localization [18, 28, 112, 120, 121, 219] (*i.e.* the optimization process "decides" on its own where are the discontinuities). Our method adds this important issue of discontinuity localization according to the input images.

Figure 2.42 on page 62 illustrates the practical effect of this detection. It is more limited than one may first think. The main reason is that objects are handled one by one in our algorithm. Therefore, large depth variations are not like to occur within a single object. This limits the interest of this step. However, a closer observation reveals that most of the details (in the briefcase for instance) are sharper thanks to this step.

Loop step 3: Graph cut with self-occlusions

To account for self-occlusion, the graph-cut technique is adapted according to the geometric configuration. Adapting the functional is straightforward, it is sufficient to add a *visibility term* $\mathcal{V}(f)$ which is infinite if f corresponds to a self-occluding surface and 0 in all the other cases. Obviously, no self-occluding surface can be minimal for this functional.

Note that this term is not the same as [121] but has an equivalent effect when considering one disparity map: self-occlusion cannot appear

Then, *visibility edges* are added to the graph to ensure the correspondence property. Let's consider a voxel A occluded by a voxel B in an adjacent column (Fig. 2.39-a). An oriented edge with infinite capacity is added as shown in Figure 2.39-b. This edge is oriented from the sink to the source. Therefore any surface including both A and B crosses this edge with the orientation that makes the edge count in the cut value. Note that any surface including a voxel behind A and/or a voxel in

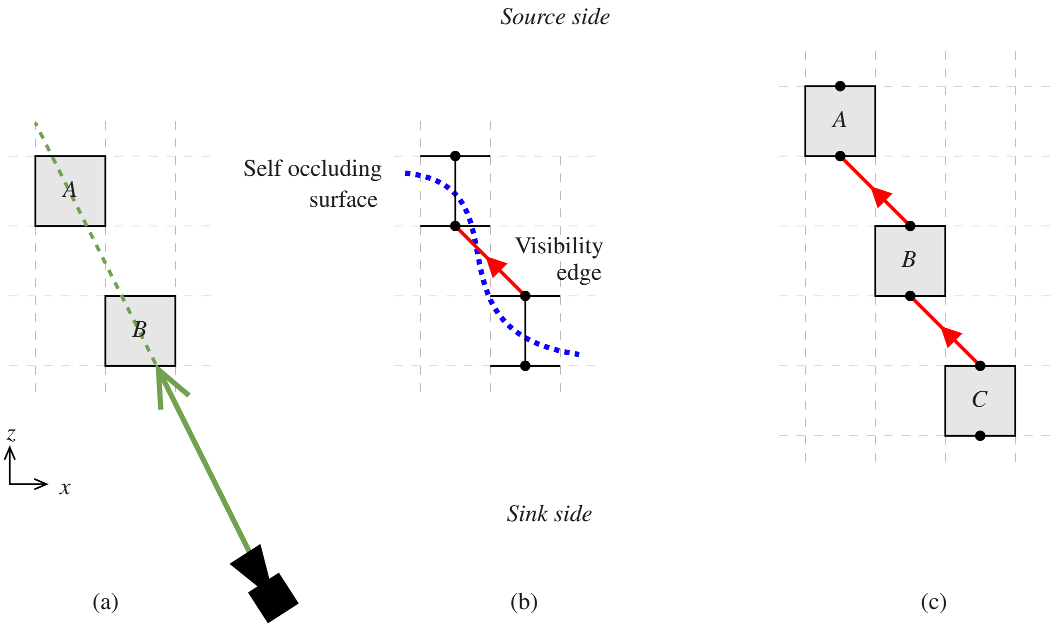


Fig. 2.39: Visibility edges

For illustration purpose, only a xz plane is presented and the graph is the simple non-convex one (Fig. 2.25 on page 45). (a) A is occluded by B. (b) The BA visibility edge and a surface containing both A and B. (c) The BA and CB edges are sufficient to handle the occlusion between C and A.

front of B is self-occluding and crosses also this infinite edge as it should be. This proves that these edges correspond to the visibility term for adjacent columns. As the relationship "is occluded by" is transitive, this is sufficient to ensure the whole visibility constraint as illustrated by the Figure 2.39-c: If A is occluded by B which is occluded by C then A is occluded by C and any surface included A and C must cross either the BA edge or the CB edge.

Hence, it is sufficient to add four visibility edges per voxel to have a graph that accounts for $\mathcal{V}(f)$. However, in practice the results we have obtained without these edges are never self-occluding. Therefore, it seems they can be omitted to alleviate the computation without any loss of quality. This point clearly deserves more study.

Loop step 4: Visibility

After each pass, the lines of sight blocked by the previously reconstructed objects are computed and ignored in the following passes. In practice, the reconstructed objects are projected in the input images and the covered pixels are flagged "occluded".

2.6.3 Post-process: Mesh smoothing

As the whole process up to this step is purely discrete, it suffers from aliasing artifacts. A smoothing filter, inspired by partial derivative equation (PDE) based image denoising [11] is adapted to a geometric surface because it can be driven by the principal curvatures κ_1 and κ_2 of the surface. Also, because the potential discontinuity lines are

> ***Another view on* [PDE smoothing]*:*** *Intuitively, the surface is made "elastic" i.e. it tends to "shrink" under some inner forces. The smoothing effect comes from the evolution of the surface driven by these forces. The feature preservation is achieved by making some regions more "rigid" i.e. they do not deform or, at least, the deformation is slower.*

already located, the PDE smoothing filter can avoid diffusing across these lines. The following explanation describes a smoothing filter designed for depth fields. This problem is simpler than the general surface smoothing problem thanks to its parameterization. One can read the discussion by Jones *et al.* [108] who give an interesting insight about the departures between both problems.

First, since $\chi_{x|y}$ evaluates the discontinuity between two adjacent columns, we define $\hat{\chi}_{x|y}$ a column-centered function:

$$\hat{\chi}_x(x_i, y_j) = \frac{1}{2}\left(\chi_x(x_i, y_j) \; + \; \chi_x(x_{i-1}, y_j)\right)$$

and equivalently for $\hat{\chi}_y$. To adapt to the directions θ_1 and θ_2 of the principal curvatures, we define a discontinuity factor $\hat{\chi}_\theta$ for the direction θ with the classical interpolation:

$$\hat{\chi}_\theta = \cos^2(\theta)\,\hat{\chi}_x \; + \; \sin^2(\theta)\,\hat{\chi}_y$$

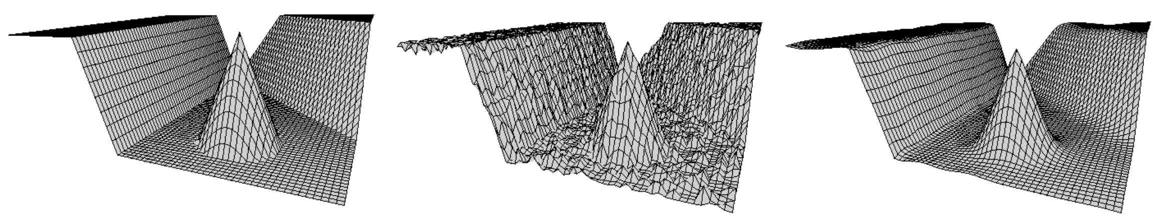
Thus, we can formulate the filter as a surface evolution driven by the following PDE with virtual time t:

$$\frac{\partial f}{\partial t} = \hat{\chi}_{\tilde{\theta}_1}\, g(\tilde{\kappa}_1)\,\kappa_1 \; + \; \hat{\chi}_{\tilde{\theta}_2}\, g(\tilde{\kappa}_2)\,\kappa_2$$

where g is a *stopping function* that controls the diffusion to preserve the curvatures (its role is discussed in detail by Black *et al.* [17]), $\tilde{\kappa}$ and $\tilde{\theta}$ are computed on a Gaussian filtered version of the surface, which leads to more robust estimations to control the filter as shown by Catté *et al.* [37]. Note that there are two controlling components: g driven by the surface geometry and $\hat{\chi}$ accounting for color discontinuity. These ensure that both curvature and discontinuities are preserved.

More about* [Stopping function]*: *It decides whether a feature should be smoothed or not. In our case, it differentiates high curvatures due to edges from those due to aliasing. Deciding on a Gaussian-convoluted version of the surface is a way to check whether a feature is large enough to survive, at least partly, a strong smoothing filter. In that case, the stopping function locally "stops" our filter to preserve the original shape of the feature.*

We demonstrate the behavior of our filter using an artificial surface in Figure 2.40: Noise is added to a known surface with a sharp peak and sharp edges. These features are correctly recovered by our



Original noiseless surface *Noisy surface* *Filtered surface*

Fig. 2.40: Effect of our PDE filter on a reference surface

Our filter is able to preserve the main feature of the surface while removing most of the added noise.

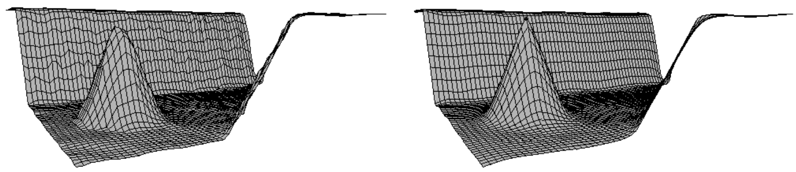


Fig. 2.41: Comparison of our filter with an image filter

Compared to the filter dedicated to image denoising proposed by Alvarez et al. [6] (left) our geometric filter (right) better preserve the peak and is insensitive to the slopes.

How to set* [Smoothing filter]*: *The only problem may appear on a surface with small details. In that case, the smoothing filter should be attenuated to preserve these details. It may let some aliasing visible. The only solution is to use a finer resolution to reconstruct the object to avoid that the details have the same scale as the aliasing.*

filter. Since we represent the surface with a depth field, a naive approach would have been to apply a denoising filter designed for gray-level images. But as discussed by Koenderink and van Doorn [118], images and surfaces have fundamental differences even if both can be represented by a scalar field (for instance, a rotation around the x axis is a standard operation for a surface whereas it is meaningless for an image). Figure 2.41 shows that, for a surface with limited noise (akin to aliasing), our geometric formulation outperforms the image filter proposed by Alvarez *et al.* [6].

Figure 2.42 shows the improvement brought by this filter. It correctly removes most of the aliasing artifacts while preserving the features of the model.

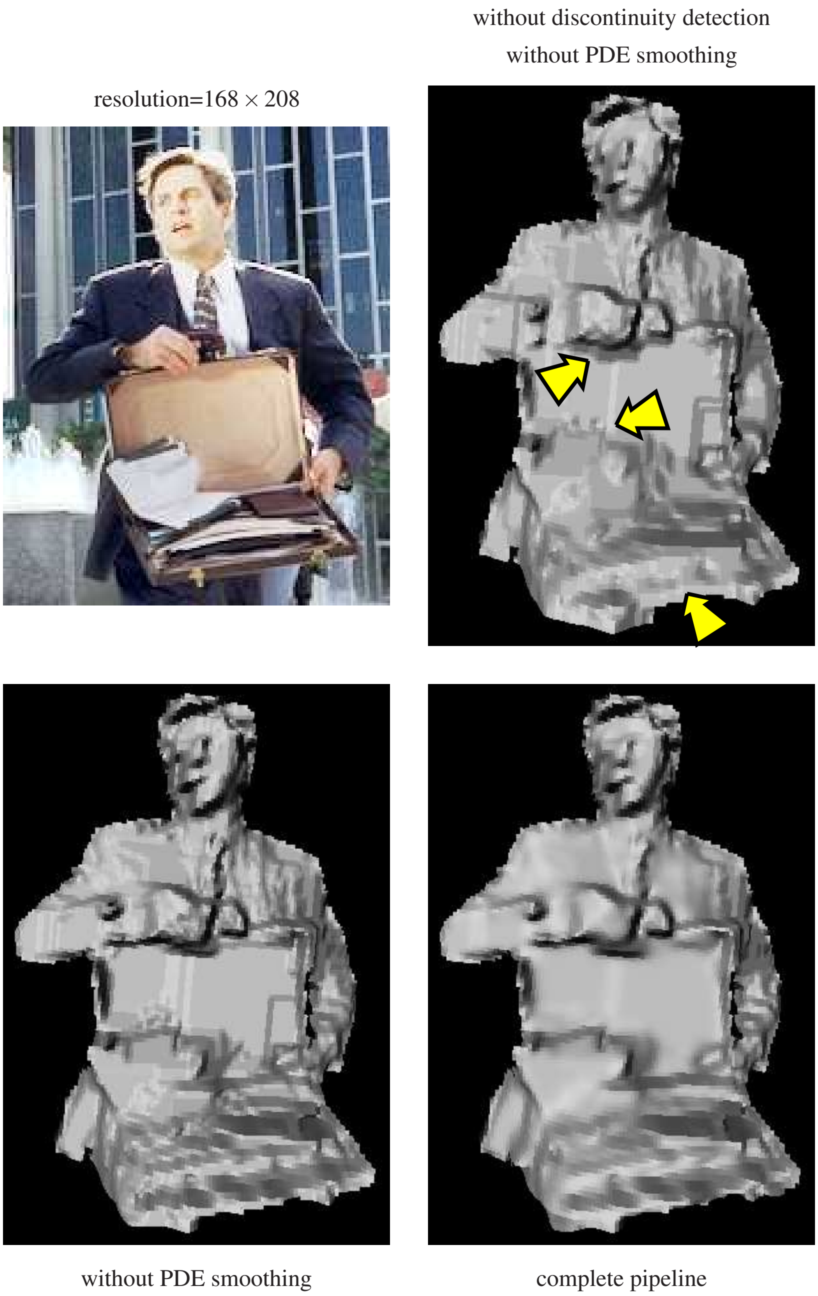


Fig. 2.42: Effects of the discontinuity detection and of PDE smoothing

Top left: *One of the original images.* ***Top right:*** *A reconstruction without the final smoothing and without detecting the discontinuities.* ***Bottom left:*** *A reconstruction without the final smoothing.* ***Bottom right:*** *A reconstruction with the complete described pipeline. The effect of the discontinuity detection is limited (the main differences are pointed by the yellow arrows); this is discussed in the text. The final smoothing removes most of the aliasing artifacts while preserving the sharp features of the model.*

2.7 Results

The implemented system is demonstrated on real examples. We show three typical sequences: the briefcase man, keyboard, and lantern sequences illustrated in Figures 2.43, 2.44 and 2.45. For the briefcase-man sequence, there are 40 frames for the street from Dayton Taylor's time-freezing setup with aligned and regularly spaced cameras. Except the time-freezing system which captures attractive "moving" scene, this setup is rather similar to short video sequence: Baseline is short (40 images spanning 1.5 meter), image resolution is limited (about 40×40 for the face of the man), the images contain a significant amount of noise and an exposition change occurs throughout the sequence due to the back light. This sequence can be considered as representative of an input provided by a non-

Fig. 2.43: Result: Man with briefcase

Reconstruction of a man with a briefcase from 40 images at 692×461, *face is about* 40×40 *pixels. Notice that the discontinuities are preserved and that the topology is correctly handled.*

specialist user. It is calibrated by a commercial system with about 70 points manually extracted from the sequence.

There are 11 frames of resolution 640×480 for the keyboard sequence and 23 frames of 800×600 for the lantern sequence captured by a hand-held digital camera. The geometry of the cameras for these sequences has been automatically computed using the system of Lhuillier and Quan [137]. To compute effective solution for results, there is a broad range for the definition of consistency. For simplicity, we use the photo-consistency (see Section 2.2 on page 9) in the *hue-saturation-value* color space for our current examples. Any other color space would yield similar results; only a color metric is needed (it may not be perceptual since we do not deal with perceptual issues). For each voxel, we use the average color of the region covered by its projection (approximated by its bounding rectangle) instead of the color of the pixel under its center. This makes the process slightly more robust to small calibration errors.

The space resolution ranges typically from 1 to 10 million voxels with 5 nodes and 12 double edges per voxel (Fig. 2.28 on page 49). The precision of the reconstruction results is very high. We notice even the geometric details on the face of the man (Fig. 2.43), which comes from only a small patch of about 40×40 pixels. Almost every key on the keyboard is reconstructed and distinguishable (Fig. 2.44). We measured the physical size of the keyboard and the keys. This gives a 1/10 pixel accuracy.

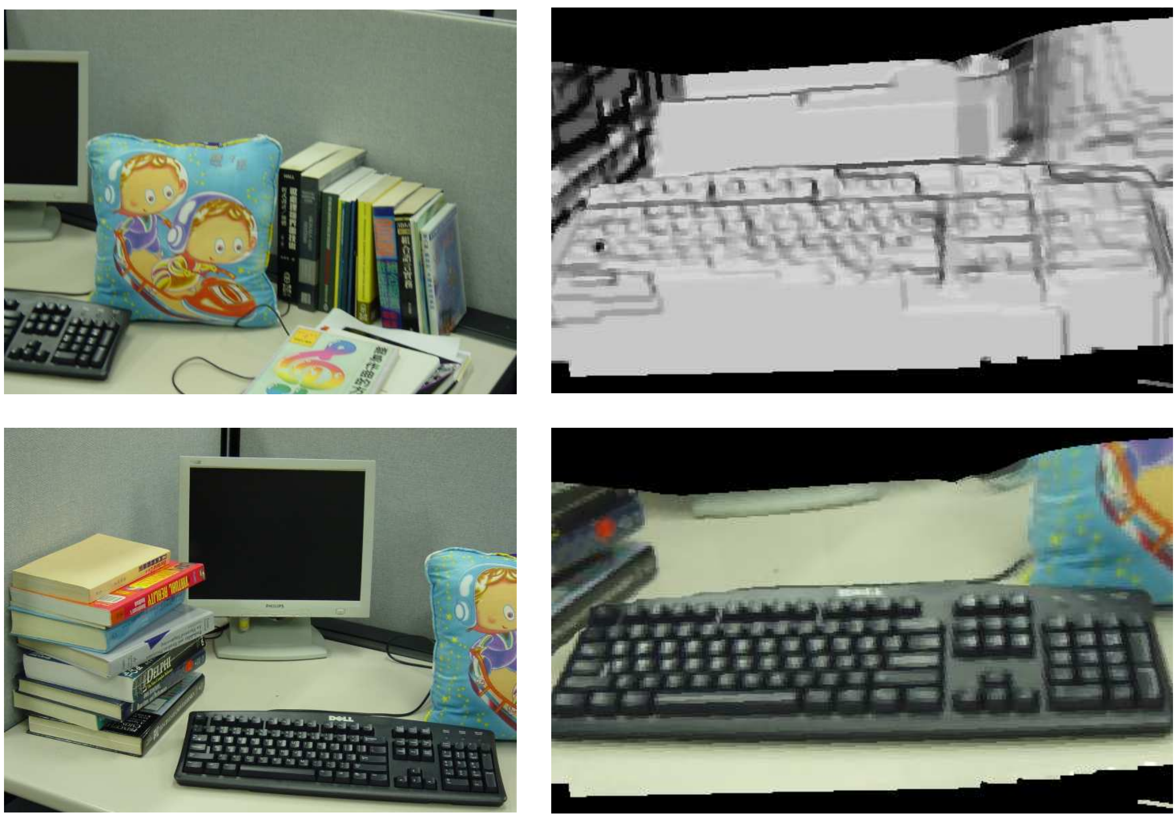


Fig. 2.44: Result: Keyboard

Reconstruction of a keyboard from 11 images at 640×480*. Notice that the keys of the keyboard are clearly distinguishable.*

Figure 2.45 shows a case with strong occlusion. The folded chess-board is hidden by the lantern in half of the images. Nevertheless, our algorithm is able to exploit the remaining unoccluded views to rebuild it.

We have tested the behavior of the technique with decreasing number of cameras. We used the sequence of the man with briefcase that is a rather difficult input as discussed previously. With 20 images among a total of 40 (Fig. 2.46-middle), some details are slightly blurred away on the face and the briefcase but the overall accuracy is almost the same. With 10 images, (Fig. 2.46-left), some spurious geometry appears on the briefcase and the contours are less precise. However, the algorithm still performs well and reaches satisfying results. We have tested with only 5 cameras, the quality loss becomes unacceptable: large spurious shapes appear due to the background and the silhouettes are too degraded. This result does not seem reasonably usable. So we have reached the limit of our technique. It shows that it works well for 10 cameras or more, even with non-perfect images. This has to be related to occlusion: the chess-board in Figure 2.45 is occluded half of the time, so it requires 20 images or more to be sure to reach a satisfying result.

2.7.1 Comparison with disparity maps

We have run the algorithm of Kolmogorov and Zabih [121] (whose code is available on the website: `http://www.cs.cornell.edu/People/vnk/software.html`) on the sequence of the man with a briefcase (Fig. 2.43). We have tried to match as close as possible our setup: we have used the same calibrated cameras with an equivalent bounding region. For the labels, we have placed 10 planes which span the whole depth of the subject, and 5 planes for the background in order to avoid spurious influence of background objects. As suggested in their paper, we have used the *robustified* L_1 *distance* to compute the smoothing term. Since the input values are limited to integers, we have tried all the relevant values for the smoothing term and selected the one leading to the best result. The obtained disparity map is shown in Figure 2.47.

We can remark that contours are very precise but depth precision is limited: only 5 labels appear for the man (among 10 possible). This let us think that this method is not able to reach a higher precision on this sequence even if we allow more labels. Moreover, these labels introduce strong

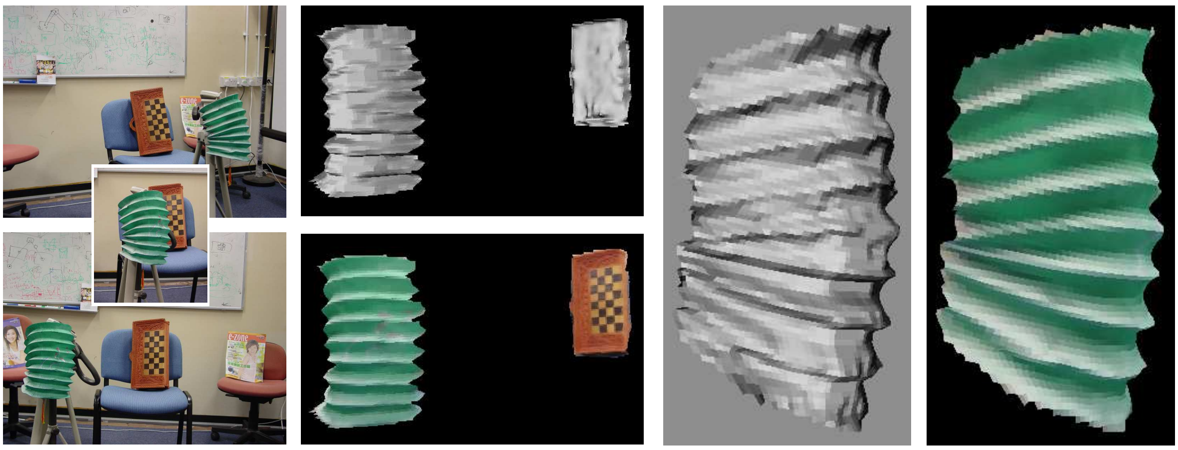
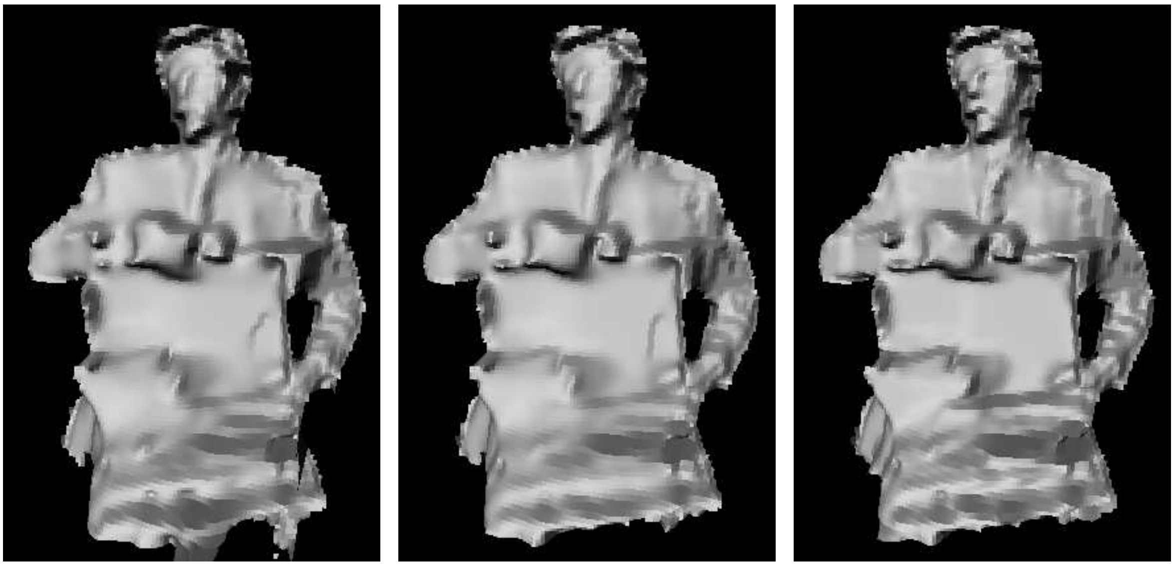


Fig. 2.45: Result: Lantern and folded chess-board

Reconstruction of a lantern and a folded chess-board from 23 images at 800×600 *(left) Although the folded chess-board is occluded from images 4 to 15, it is correctly reconstructed.*



from 10 images | from 20 images | from 40 images

Fig. 2.46: Influence of the number of images

Reconstruction from the same sequence as Figure 2.43 but with a varying number of images. The technique achieves acceptable results down to 10 images.

discontinuities in some regions where there is no clear reason to do so *e.g.* on the briefcase and on the left arm. This clearly comes from the concave shape of the *robustified* L_1 *distance*. From the presented study, this point can be overcome with a convex smoothing term associated with an image-driven detection of the discontinuities.

A side-by-side comparison (Fig. 2.47 on the next page) outlines the dramatic improvement of the depth precision brought by our technique.

2.7.2 Graph flow implementation

The graph-cut optimization process is time consuming. These examples took about 15 minutes to compute on a Intel Xeon 2.4GHz and they need between 300MB and 700MB of RAM. These values have to be evaluated considering the size of the graph ($\approx$ millions of vertices and edges). This has to be compared to the graphs used in [121] (at most 600 000 vertices and 4 millions directed edges). The huge graph-flow problem exceeds the existing graph-flow implementation. It has been made tractable through a careful implementation of the *push-relabel* algorithm and heuristics presented by Cherkassky and Goldberg [39].

Then we have added our own improvements. For memory space, the key point of our implementation is that it computes adjacency information on-the-fly instead of storing it. Then we ensure that there is no circular flow in double edges so that the flow in double edges can be stored in a single signed value instead of two. Note that the other classical implementation proposed by Boykov and Kolmogorov [27] is not suitable for our graphs because it is specifically designed for small graphs and they have shown that its higher complexity degrades performance for large graphs. This is a real drawback in our case.

Our code is available at `http://artis.imag.fr/Members/Sylvain.Paris/#code`

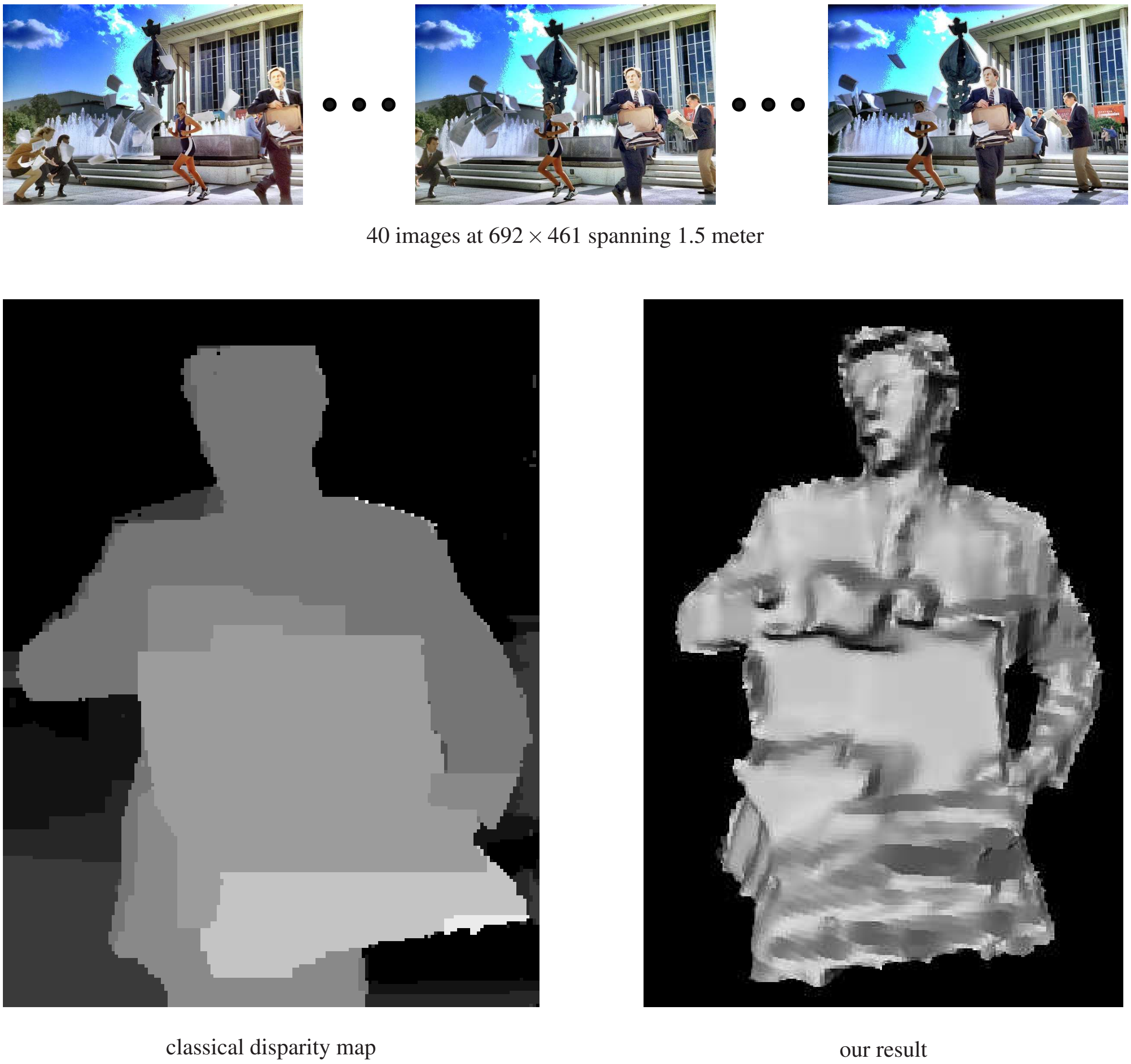


Fig. 2.47: Comparison with a disparity map

Disparity map computed with the method of Kolmogorov et al.[121] on the same sequence as Figure 2.43. [This experiment has been kindly made by Yichen Wei.]

2.8 Conclusions

We have described a method that yields accurate object reconstruction from a sequence of images. We here outlines the origins of the information that, once combined, form the final result. We then summarize the pros and cons of the presented method from a technical point of view. And, we propose a more general and more personal discussion about the presented work. We end with a few possible extensions to this work.

2.8.1 Information sources

Object topology and position: The topology of the object to be reconstructed is not limited and is determined by our algorithm (Figure 2.43 on page 63 shows a surface with a hole for instance). The genus of the surface to reconstruct comes from the thresholding of the voxel set that isolates the objects and determines optimization domain (cf. Sec. 2.6.2 on page 55). Since this step applies a threshold on voxel values, it can be linked to the carving methods which are known to recover a bounding volume of the objects and to properly estimate their topology (cf. Sec. 2.2.3 on page 19).

Object surface: The precise location of the surface comes from our graph-cut optimization. It retrieves the most consistent surface while imposing a piecewise regularity. Reaching a global minimum is a powerful feature that overcomes difficult situations (observe how large the consistent regions are in Figure 2.37 on page 56) and produces fine details as the keyboard (Fig. 2.44 on page 64).

Visibility: The visibility of given point stems from a classical front-to-back sweep of the scene. This restricts the algorithm to the scenes that can be ordered in such a way. However this corresponds to numerous practical situations. We describes in the next chapter other schemes to deal with visibility.

2.8.2 Limitations

Large resources: The described algorithm requires a lot of computation time and large resources. We believe that all existing reconstruction techniques have a limited scalability *i.e.* they cannot reconstruct arbitrary large objects or scenes. However, the results presented in the previous section are already borderline relatively to a desktop machine. This issue is treated in the following chapter.

Non-intrinsic functional: The proposed functional is not intrinsic *e.g.* turning the coordinate system changes the stated problem. This has a limited impact in practice since artifacts are hardly noticeable. One can observe the shoulders of the man in Figure 2.43 on page 63: Their slopes are roughly parallel to the axes. However, from a theoretical point of view, it would be better to have an intrinsic formulation. Appendix A.2 on page 170 proposes some ideas to alleviate this problem.

Scalar field: This point is linked to the previous one. We have presented an algorithm using a Cartesian parameterization of the surface $z = f(x,y)$. It is straightforward to adapt to other coordinate system such as the cylindrical one $r = f(h,\theta)$ or the spherical one $r = f(\theta,\varphi)$. But, our optimization engine deals only with scalar field. This implies an a priori knowledge about the surface to chose a correct parameterization. This shortcoming is addressed in the next chapter.

2.8.3 Contributions

Optimization engine: We have presented a new optimization engine that minimizes first order functionals. The solution is guaranteed to be a global minimum of the functional up to an arbitrary discretization. We believe that this is one of the explanation of the high precision of the presented results. Another benefit is that the engine does not require any "initial guess". This obviously overcomes the dependency on such a guess.

Geometric functional: We have designed a new functional that fits the proposed optimization engine. This functional is purely geometric and independent of both image and space resolution. As discussed after the study of the previous work (Sec. 2.2.9 on page 33), we believe this is a major feature of a surface reconstruction technique compared to an image-based approach involving disparity maps. Furthermore, this functional accounts for surface discontinuities; modeling explicitly edges, corners, etc.

Smooth shapes: We have formally characterized the functionals able to properly recover smooth shapes without introducing spurious discontinuities. We have coupled this theoretical criterion with a practical technique to detect potential discontinuities in order to produce correct edges and sharp features. These characterization and tools are general enough to be "plugged" into other existing methods.

Efficient algorithm: We have described an algorithm that coherently includes the previous contributions. It demonstrates that all these steps can be merged together to reach a clear gain on the resulting precision and robustness.

2.8.4 General discussion

We believe that our algorithm brings significant improvements over the existing techniques especially on the precision. Our input images do not have a very high resolution, noise is even sometimes quite high (Fig. 2.43 on page 63). So we may wonder if there is any fundamental reason beyond all the technical arguments that justifies these results. For instance, one may say that we do not have "invented" the graph-cut technique neither the geometric formulation. And we clearly agree. So why our results have more details? Do we only use a higher resolution? The comparison with an existing method on Section 2.7.1 on page 65 shows that it is not only a resolution issue since from the same input, we achieve a precision noticeably better. We here expose our point of view on the "fundamental bases" of our results.

3D space

As discussed in the Previous Work, since we target 3D shapes, working in the 3D space is an important choice. Even if from a technical point of view, our approach is close to the existing graph-cut techniques that are image-based; from a fundamental point of view, it is more similar to the 3D methods such as level sets and carving.

It would not be rigorous to argue that this 3D formulation brings any accuracy improvement by itself because disparity and depth are mathematically linked: Any 3D formula can be turned into disparity and conversely. But we are convinced that working in the space corresponding to our goal is fundamental. The problems appear more directly and more obviously. For instance, a piecewise constant disparity may look "nice" because it shows precise boundaries. But one has to imagine the represented 3D shape to discover the real geometry that may lack of depth precision or have spurious discontinuities.

Therefore, we are convinced that using a 3D representation has helped us to better characterize our difficulties and to better address them.

Summary: Working in the 3D space gives a clearer understanding of the 3D reconstruction issue than relying on an image description of the problem.

More information extracted

To us, another important point is that we extract more information from the input data. Most of the existing optimization scheme (carving, level sets, graph cuts) rely on the same information computed from the data: the consistency function. It can be evaluated with different estimators (mainly photo-consistency and ZNCC) but as discussed in the Previous Work, each one has its own pros and cons and can be "plugged" equivalently into an optimization engine. Therefore, the only major differences between these methods mainly come from the prior and/or from the optimization method:

- Carving only requires the result to be consistent. Therefore, it produces the largest consistent volume. This volume contains all the possible results since any additional prior would set more constraints on the volume and shrink it.
- Level sets implicitly add the prior that the surface is C^2 everywhere. In addition, the optimization scheme makes the process relies on the initial guess only for the spatial localization. This guess does not have to be a bounding volume of the result nor to provide its correct topology.
- Graph cuts use the prior of piecewise constant results. The technique also ensures a controlled, and sometimes exact, convergence and therefore obviate the need for an initial guess. The counterpart is the need for the topology information.

From this, we can draw our first remark: The optimization scheme provides some user facilities (fewer constraints on the input). Concerning the accuracy, we also believe that the exact convergence extracts more information from the consistency function. A local convergence may miss some information and under-exploits the available data. Hence, fewer details may be reconstructed. From this point of view, our exact convergence is one of the advantage of our method.

However, all these methods "dig the same mine" and cannot find more gold than it contains. We believe that the available information from the input images is not only contained within the consistency function. For instance, the visual-hull approach exploits different data (contours). Therefore, the content that can be extracted is different: Topology is straightforward whereas concavities are impossible to recover independently of the applied technique.

More about* [Visual hull and consistency]*: With a distant background, the contour information is "indirectly included" within the consistency function. Even with a small baseline, such a background causes significant differences in the images near the object silhouettes because of the large depth difference (see Figure 2.43 on page 63). Hence, the consistency is bad in the surrounding volume of the objects, making possible an accurate segmentation and contour reconstruction (Fig. 2.43) that is almost "equivalent" in practice to what can be done by a visual hull. Zeng and Quan propose an algorithm [237] that relies on this remark to compute the visual without knowing the background.

We believe that the presented method exploits another kind of data in addition to the consistency. The potential discontinuities are clearly another source of information that is different from the consistency and the contours. The color variations of the images are not directly included in consistency or contours. As we have seen, a desired feature is to evaluate these variations independently of the local contrast. Such an evaluation only accounts for the variations of the object color and not for the one stemming from lighting changes. This gives a hint that the underlying notion of our evaluation is the *intrinsic image* [208, 223] *i.e.* a separation of the image into albedo data (base color of the object) and lighting data (light power illuminating a given pixel). Our potential discontinuities can be interpreted as the discontinuities of the albedo layer.

To our knowledge, this information is seldom used in the existing work (only Veksler [219] proposes a first study). We believe that it also participate to recovering more details: We extract and exploit more information than the other methods.

Nonetheless, as illustrated in Figure 2.42 on page 62, it does not bring a large improvement. An explanation may be that, similarly to the contours (cf. side box on the facing page), these discontinuities are somehow “indirectly included” in the consistency function: We have shown that the consistency is ambiguous when the object has locally a uniform color. Conversely, on color changes (our potential discontinuities) the consistency gives a precise surface location (Fig. 2.37 on page 56). Since the optimization domain is composed of the most consistent point, it is very narrow near the discontinuities. The resulting constraint is almost equivalent to our discontinuity detection. However, we have shown (Fig. 2.42) that an explicit use still brings improvements, even if they are not as important as one could expect. It lets us think that a direct visual hull computation may also enhance our results.

Summary: We believe that our algorithm extracts more information from the consistency function because it converges exactly. Furthermore, even if it does not bring a dramatic improvement, the discontinuity detection introduces information that is rarely used by the other techniques.

Trade-off between regularization and consistency

We think that the regularization is another crucial point. The reconstruction problem is inherently ill-posed and regularization is mandatory in most cases. Carving techniques do not explicitly regularize their optimization. In consequence, they rely on multiple views widely spaced to constrain the problem. It is well-known that these approaches do not properly handle scenarii with few cameras or grouped viewpoints. Hence, to overcome this shortcoming, introducing a prior into the optimization scheme is a reasonable solution and, to our knowledge, it is today the only way to achieve dense reconstruction.

In the same time, a prior is by definition independent of the input data. It is an arbitrary constraint set by the user. It may not correspond to the actual content of the images. For instance, the a priori smoothness that we assume is obviously not suitable to handle the rough surface of an irregular rock. We believe that any prior suffers from such out-of-prior cases. The smoothness assumption is however reasonable and therefore largely used in the literature.

Nonetheless, even if we are convinced that this prior is a good choice, we have to handle it with care. Let’s us study the fundamental action of the prior. It modifies the results coming from the consistency *i.e.* it may somehow overrule the data to drive the reconstruction. Since the problem is ill-posed, such a behavior is desirable when several solutions are equivalent regarding the only input data. In that case, the chosen solution is likely to be erroneous because it may come from implementation choices or from the noise corrupting the input data. Hence, a prior can be seen as a way to correct the errors arising from those cases; the correction being directly under the user control instead of coming from the implementation or from the noise.

Unfortunately, the prior is likely to also impact correct points *i.e.* the data yield good results but the prior overrules them and imposes a different solution. This partly explains why most of the existing techniques (including ours) need a careful setting of the functional weights: One has to adjust the prior influence considering the actual smoothness of the observed scene versus the noise of the input data. But beyond that, we believe that a prior such as the C^2 differentiability imposed by the level sets is too strong in many cases. The images of objects with sharp edges and corners provide sufficient information to reconstruct these details. But the C^2 prior is not compliant with these features and smoothes them away, yielding to less detailed results. The piecewise-C^1 prior imposes fewer constraints while being sufficient to regularize the problem. It therefore provides an improved fidelity to the original data, producing more details.

From this analysis, we can observe that the dependency of our system relatively to orientation of the axes is undesirable. It is useless to regularize the process nor to remove the errors. It only introduces the spurious influence of an arbitrary user choice. Fortunately, this influence is light and results in mostly unnoticeable artifacts. It would however be interesting to study and address this point.

Now, the question is "Why are our results so different from the ones obtained with other graph-cut techniques [32, 179, 180] using an equivalent smoothing term?" The very first reason is our convex approximation of the linear penalty proposed in Section 2.5.4 on page 46. This point is proven and demonstrated to dramatically improve the results obtained from a linear penalty (Fig. 2.27 on page 48).

Beyond this main reason, we believe that the optimization domain (Sec. 2.6.2 on page 55) also contributes to this difference. By thresholding the consistency function, we ensure that the final surface contains only highly consistent points. The morphological operators may add only a few isolated "bad" points. Therefore, this domain acts as an additional constraint over the regularization term: It cannot change too much the surface, especially in the textured regions where the domain is narrow (see Fig. 2.37 on page 56, the stripped tie produces a thin domain). With other words, the optimization domain is almost a hard constraint on the textured points (*i.e.* the surface must go through these points). This counters the effects of the smoothing term that tends to produce flat surfaces with null derivatives. So, the depth variations are forced to respect the feature points and the convex approximation ensures smooth variations.

On the other side, disparity-map techniques do not constrain the optimization domain. Each point can have any disparity among the entire available range. As a consequence, even a clear feature (such as the tie in Fig. 2.47 on page 67) may be omitted because it suffers from a too important penalty from the smoothing term.

Summary: We believe that the regularization has an important role relatively to the accuracy of the results. It is twofold: On the one hand, it is mandatory to achieve satisfying results because the base problem is ill-posed. But, on the other hand, it may produce results that stems more from the user prior than from the input data.

Our regularizing term is lighter than the ones previously proposed (piecewise-C^1 surfaces instead of C^2) while being sufficient to make the process well-posed. Moreover, the influence of this term is further limited by the optimization domain that forces the resulting shape to respect the textured features.

Closure

To end this discussion, we would say that we are convinced that the key point to reconstruct detailed surfaces is to be as close as possible of the input data and to introduce as few as possible additional constraints that are not data-driven. The leading remark to this is that only the input data can provide the information to recover the details.

Therefore, we believe that relying on the data to drive the discontinuities and to determine the optimization is a positive aspect of our method. Assuming only a piecewise-C^1 is also an advantage. But not being intrinsic is a drawback.

Hence, we may wonder what could be an ideal algorithm, disregarding the technical feasibility. Concerning the regularization, piecewise-C^1 seems to be almost "minimal" since C^0 would not be enough (carving techniques are C^0 and irregular). The counterpart is that the resulting functional is not intrinsic. It would be interesting to study piecewise-C^2 surfaces: The constraint is one order higher but the functional would be intrinsic using the curvature and the piecewise aspect would still allow sharp features.

Then about the information sources, using as many as possible appears as a natural strategy as long as these sources are reliable. We have shown how to use the color discontinuities in addition to the traditional image content. Contours may also be used with the latest techniques [237] to extract them from unknown background. Shading has also surely a great potential [90] but it requires additional input data (reference spheres in [90]) or is still hard to use in the general case [238] because of the unknown lighting environment.

2.8.5 Extensions

The presented approach can be extended in numerous ways. The following chapter exposes a "patch technique" that addresses several shortcomings: mainly scalability and parameterization. Beyond these points, it would be interesting to study the use of more sophisticated point-matching criteria (cf. Sec. 2.2 on page 9) because the classical ones (*e.g.* photo-consistency and ZNCC) require the projected voxel size to lie in the order of a few pixels: According to our tests, the reconstruction quality degrades quickly beyond 4 to 5 pixels.

Studying matching criteria coping with view-dependent effects is also a challenging direction. Several existing approaches rely on a reflectance model *e.g.* Yang *et al.* [233] assume that only the reflection intensity is view-dependent. We believe that robust statistics [96] can also provide an interesting alternative, less dependent on a reflectance model.

A more distant future work would be to include the lighting effects *e.g.* shading within our study. Following this way, the approaches such the ones by Treuille *et al.* [214], Zhang *et al.* [238] or Yang *et al.* [233] should be inspiring. Introducing learning in the process could be also considered. In that, the technique to create new views from a set of images of Fitzgibbon *et al.* [66] is seducing since it learns only from the initial image set.

As discussed before, making the functional intrinsic or at least less dependent on the parameterization is a interesting theoretical challenge. We give a few hints about this issue in Appendix A.2 on page 170.

From a more general point of view, we have developed a system that has convincing qualities for accuracy. We believe that it is possible to push further in this direction to build a system to acquire meso-geometry *e.g.* bumps on a wall or cracks on a rock. This would capture input data to the latest rendering techniques such as the work of Wang *et al.* [220] or Sloan *et al.* [200]. This would provide a useful alternative to dedicated systems such as the one described by Han and Perlin [83].

On the other side, focusing on a larger scale such as architectural buildings (*e.g.* a cathedral) could lead to interesting issues. A possible approach would be to use new space representation to better address the specificities of this scale. For instance, the *Billboard Clouds* exposed by Décoret *et al.* [59] may be a suitable structure to study in this context.

3

Patchwork reconstruction

This chapter reads better after the previous one on surface reconstruction.

Important parts of the work presented in the following sections has been made by Gang Zeng who is a PhD student at Hong Kong University of Science and Technology with the help of his supervisor, Long Quan. We have mainly contributed in the motivation of this study whereas Gang Zeng has designed the practical algorithms and implemented them. In this dissertation, we focus on our own contributions and provide a brief summary of Gang Zeng's work. We refer the reader to his future PhD thesis and to our common publications ([236] is already published, more are coming).

3.1 Introduction

The method exposed in the previous chapter has shown great potential to recover fine details. However, it suffers from several limitations that impairs its usability:

- The amount of data to be handled is huge and requires a dedicated implementation. Even with that the running time and required resources are important and already borderline for a common desktop machine.
- The addressed scenario is not fully general: A separating plane must exist between the cameras and the scene. It inherently restricts the reconstruction to the front-facing part of the objects.
- The surface parameterization is limited to $z = f(x, y)$ that imposes the use of several functions $f_1, f_2, \ldots$ to describe complex scenes. We have shown how to deal with occluded objects. But the definition of these functions in the general case still needs to be addressed.

In this chapter, we propose a surface representation to overcome these shortcomings. The surface is (virtually) cut into small pieces that we call *patches*. In this chapter, we will show how to solve the reconstruction problem using this representation.

3.2 Motivation and concept definition

We here formalize our problem to outline the fundamental reasons that justify the use of patches. We name $\mathcal{S}$ the object surface we are seeking. Let $\mathcal{F}(\mathcal{S})$ be a functional that represents our goal. For now, we do not give more details about $\mathcal{F}$ to keep it as general as possible. We only assume that $\mathcal{F}(\mathcal{S})$ is low if and only if $\mathcal{F}$ is a correct reconstruction of the object. The design of such a functional will be discussed later. Here, we only study how to find a minimizer of $\mathcal{F}$. We first define the patch concept and then examine it from a complexity point of view and from a parameterization one.

Patch definition Intuitively, a *patch* is a small piece of the surface $\mathcal{S}$. Formally speaking, a patch $\mathcal{P}$ is a surface, subset of $\mathcal{S}$. Performing a patch reconstruction is finding a set of patches $\{\mathcal{P}_i\}$ such that $\bigcup \mathcal{P}_i = \mathcal{S}$.

3.2.1 Study of the complexity

We here observe the complexity in term of space and time for two cases, a global optimization and a patch-based one. To do so, let's consider that $\mathcal{S}$ has an area $a_\mathcal{S}$ and a volume $v_\mathcal{S}$ and that it is represented by a discrete structure with a discretization size δ. For instance, for level sets, this structure is the distance field embedding the surface and for graph cuts, it is the quantized 3D (or disparity) space that supports the surface vertices.

Global optimization: An algorithm that minimizes $\mathcal{F}$ over the whole surface $\mathcal{S}$ deals with a data structure of size at least $O\left(a_\mathcal{S}\,\delta^{-2}\right)$. This is the case for some graph-cut techniques [121] and for the narrow-band implementation of level sets [1]. Some algorithms (such as level sets, carving methods or some graph-cut techniques) use volumetric representations, hence have a space complexity in order of $O\left(v_\mathcal{S}\,\delta^{-3}\right)$.

Then, any minimizing process is at least linear respectively to the data. We consider an minimizing process with a complexity of degree $\alpha \geq 1$. Therefore the time complexity is $O\left(a_\mathcal{S}^\alpha\,\delta^{-2\alpha}\right)$ or $O\left(v_\mathcal{S}^\alpha\,\delta^{-3\alpha}\right)$ depending on the surface representation. The complexity of level sets [62, 138] is unclear because it depends on the number of iterations depending itself of the starting point and target shape. Graph-cut algorithms are typically cubic (or slightly better [39]). In practice, they behave almost linearly ($\alpha \approx 1.2$) [179]. Note that some graph-cut techniques (*e.g.* Kolmogorov and Boykov [121]) are iterative and their complexity may be higher as mentioned for level sets.

Patch optimization: Let's subdivide the surface $\mathcal{S}$ into patches $\mathcal{P}$ with area $a_\mathcal{P}$. The number of patches η is in order of $O(a_\mathcal{S}/a_\mathcal{P})$. To compare with $\mathcal{S}$, we also define a pseudo-volume $v_\mathcal{P} = v_\mathcal{S}\,\eta^{-\frac{3}{2}}$ by considering that surfaces and volumes are related by a logarithmic ratio of $\frac{3}{2}$.

Then optimizing $\mathcal{F}$ over a patch has a space complexity in order of $O\left(a_\mathcal{P}\,\delta^{-2}\right)$ (or $O\left(v_\mathcal{P}\,\delta^{-3}\right)$ for a volumetric representation). Because patches are processed one by one, the overall space complexity is the same. Only the storage of the final result requires more space but this can be done off-line (*e.g.* on the hard drive). And since we optimize η patches, the overall time complexity is in $O\left(\eta\, a_\mathcal{P}^\alpha\,\delta^{-2\alpha}\right)$ or $O\left(\eta\, v_\mathcal{P}^\alpha\,\delta^{-3\alpha}\right)$.

Comparison: Table 3.1 summarizes all these results. It appears that the patches bring significant gain in term of space and time complexity. Relatively to our goal, the gain in space is the main

	SPACE global	SPACE patches	SPACE **gain**	TIME global	TIME patches	TIME **gain**
surfacic	$a_{\mathcal{S}}\,\delta^{-2}$	$a_{\mathcal{P}}\,\delta^{-2}$	η	$a_{\mathcal{S}}^{\alpha}\,\delta^{-2\alpha}$	$\eta\, a_{\mathcal{P}}^{\alpha}\,\delta^{-2\alpha}$	$\eta^{\alpha-1}$
volumetric	$v_{\mathcal{S}}\,\delta^{-3}$	$v_{\mathcal{P}}\,\delta^{-3}$	$\eta^{\frac{3}{2}}$	$v_{\mathcal{S}}^{\alpha}\,\delta^{-3\alpha}$	$\eta\, v_{\mathcal{P}}^{\alpha}\,\delta^{-3\alpha}$	$\eta^{\frac{3}{2}\alpha-1}$

Table 3.1: Comparison of the complexity

one because we divide the memory needed by a factor in order of the number of patches used. But, we cannot decrease the size of the patches infinitely to increase their number because we would not be able to find a satisfactory result (this issue is discussed later in the paper).

More about* [Space complexity]*: *Rigorously speaking, we need to store the position of each patch relatively to the global surface. This requires a storage in order of* $O\left(\log(a_{\mathcal{S}}\,\delta^{-2})\right)$ *or* $O\left(\log(v_{\mathcal{S}}\,\delta^{-3})\right)$ *which is negligible because for all practical cases it always fits within three 32-bit values* xyz.

Scalability property: The patches allow almost unlimited scalability because the space complexity depends only on the patch size and no more on the object size.

The gain on volumetric representations is more important. This comes from the fact that the patches ignore the inner volume of the object that requires extra storage and generates extra computation. In that, they are comparable to a narrow band optimization [1].

Summary: The patch algorithm is less complex than the global approach, both in time and space. The spatial gain is the most important, since the required memory depends only on the patch size and no more on the object size.

3.2.2 Study of the parametrization

The patches also alleviates the limitation on the parametrization inherent in graph-cut methods. These methods handle scalar field: Typically, the depth is a function of the two other coordinates *i.e.* $z = f(x,y)$ for some function f. This limits the usability of these techniques. First, special care is needed to properly handle the cases that require several z values for a single (x,y). Several functions f_1, f_2,... are then manipulated. Moreover, if the object surface is tangent to the z axis, these methods fail because of $||\nabla f|| = \infty$.

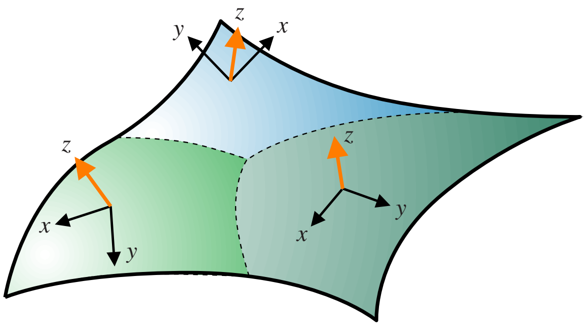


Fig. 3.1: Three patches with their local coordinate system.

The patch approach eliminates these shortcomings. By definition, the patch reconstruction deals with several surfaces and therefore intrinsically manipulate several f functions. Furthermore, the xyz coordinate system can be adapted to each patch. This means that the z axis can be chosen orthogonal to the surface to guarantee that the tangent case never occurs. Formally speaking, this is always possible if the surface is piecewise C_1: The only problem might occur with the points that are not C_1. But those points can described by a patch based on an adjacent C_1 point. Remark that topology is not a problem in the sense that patches can cope with any topology. However, topology is not determined by the patches themselves: We rely on a side technique to determine it (this point is discussed later).

Representation property: The patches can describe any surface that is piecewise C_1 independently of the topology.

Since the surface representation is defined patch by patch, it is possible to adapt its precision to focus on the most detailed parts. The discretization step can be changed to capture finer details on the "important" regions whereas the other ones are more coarsely reconstructed to save time and space.

Summary: Thanks to their local representation, the patches can handle arbitrary topology. It can also vary the resolution of the image representation from region to region.

3.2.3 Discussion

Problem specificity The complexity study is general and almost independent of the reconstruction issue. However, it is important to remark that it relies on the assumption that it is sufficient to optimize a patch once and no more. In that it is different from the classical approach in parallel computing that subdivides a large problem (*e.g.* equilibrium in Mechanics) into small subproblems and boundary problems that assure the overall coherence between the subproblems. Classically, the subproblems are iteratively solved until convergence and lead to a complexity at least equal to the original one. More details can be found in dedicated publications [133, 76]. We here assume that once we have found a patch, the following computation have no impact on it. This explains why we have a gain in time. It also implies that we do not solve the global problem (that requires the complete approach).

We are convinced that this assumption is reasonable considering the reconstruction problem: This problem does not involve values that have an overall influence (unlike forces in mechanics for instance). If a patch is correctly located in space then the problem is locally solved whatever exists in the rest of the scene.

Normals and topology As previously discussed, the surface normal has to be determined because we align the local z axis with it. To address this issue, we use a side technique that provides an initial guess. Numerous choices exist: carving, visual hull, level sets, etc. See the Previous Work section on page 8 for details. Note that we do not require this side technique to produce an accurate reconstruction, we only need an estimation of the normal. Typically, it can be run at a very coarse resolution to be able to fit within the available resources.

In addition, we might also rely on this side technique to provide the topology. In the following sections, we detail a scenario for which we use the side technique only for normals and one that requires both normals and topology.

Parameters There are several parameters that drive a patch reconstruction. We have not yet study all of them. Nevertheless, we here discuss shortly their influence:

Order: By hypothesis, a patch ignores the computation that occurs after its creation. However, its optimization can take into account the already created patches to get more information. Therefore, it is important to order the process. We first reconstruct the most reliable regions so that the weakest patches rely on them to be more accurate.

Patch size: There is a trade-off between efficiency and robustness: Smaller patches lead to faster computation and require less resources. The counterpart is to rely on less data, making the process sensitive to noise.

Patch border: To ensure a coherent optimization, the creation of patch has to deal with a larger domain than the one that supports the patch. If we restrict the optimization domain to the patch, the border points have a truncated neighborhood and may produce erroneous results. The question of the domain extension then boils down the previous trade-off on the patch size.

3.3 Implementation using graph cut and distance field

We here expose how we implement the general patch approach described in the previous section.

3.3.1 Graph-cut base

We naturally apply the graph-cut method that have motivated the patches. For each patch $\mathcal{P}$, we assume a local coordinate system $(x^{\mathcal{P}}, y^{\mathcal{P}}, z^{\mathcal{P}})$ to be known (this issue will be discussed later). We parameterize $\mathcal{P}$ by $(u,v) \mapsto \mathbf{x}_{\mathcal{P}}(u,v) \equiv (x^{\mathcal{P}}_{\mathbf{x}}(u,v), y^{\mathcal{P}}_{\mathbf{x}}(u,v), z^{\mathcal{P}}_{\mathbf{x}}(u,v))$ and we minimize the Functional C.2.3 over a domain $\mathcal{D}_{\mathcal{P}}$ as defined on page 186:

$$\iint_{\mathcal{D}_{\mathcal{P}}} \left(c(\mathbf{x}_{\mathcal{P}}) + \alpha_u(u,v) \left| \frac{\partial z^{\mathcal{P}}_{\mathbf{x}}}{\partial u} \right| + \alpha_v(u,v) \left| \frac{\partial z^{\mathcal{P}}_{\mathbf{x}}}{\partial v} \right| \right) du\,dv \tag{3.1}$$

To minimize this functional, we use the optimization engine described in the previous chapter. To evaluate the consistency $c(\cdot)$, we use the ZNCC estimator (see Section 2.2 on page 9) based on the two most front-facing cameras according to the local coordinate system. This makes our system more sensitive to noise but more robust to view-dependent effects (highlights).

Hard constraints

Some patches $\dot{Q}_i$ may have already been reconstructed in the neighborhood of $\mathcal{P}$. We have to ensure a coherent transition between the "old" patches $\dot{Q}_i$ and $\mathcal{P}$: We force $\mathcal{P}$ to contain the points of the $\dot{Q}_i$ patches that are in its domain $\mathcal{D}_{\mathcal{P}}$. Denoting these points $\{(\dot{x}^{\mathcal{P}}_j, \dot{y}^{\mathcal{P}}_j, \dot{z}^{\mathcal{P}}_j)\}$, we minimize Functional (3.1) under the hard constraints:

$$z^{\mathcal{P}}_{\mathbf{x}}\left(\dot{x}^{\mathcal{P}}_j, \dot{y}^{\mathcal{P}}_j\right) = \dot{z}^{\mathcal{P}}_j \tag{3.2}$$

These constraints are added into our optimization engine by linking with infinite edges the voxels $\{(\dot{x}^{\mathcal{P}}_j, \dot{y}^{\mathcal{P}}_j, \dot{z}^{\mathcal{P}}_j)\}$ to both the source on their top and the sink on their bottom (Fig. 3.2). Since a cut separates the source from the sink, it must go through these voxels. Hence, it satisfies the constraints (3.2). □

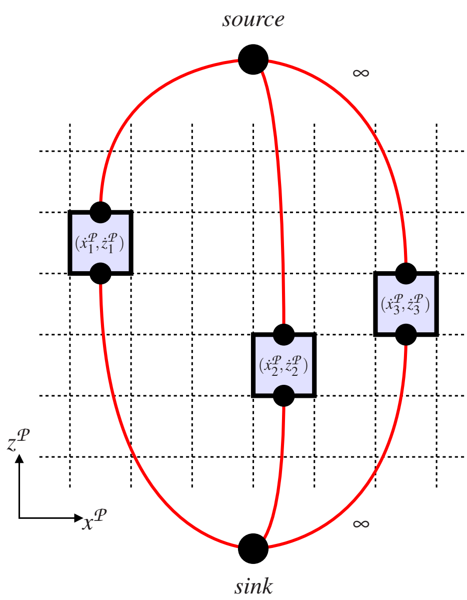


Fig. 3.2: Hard-constrained voxels in 2D.

Other data types

This extension has been proposed by Gang Zeng.

We extend Functional (3.1) to other data types beside the traditional image consistency $c(\cdot)$. Let's use a general function γ instead of c within Functional (3.1). In addition to c, we define:

- A point function p that accounts for known 3D points $\mathbf{p}_k$ coming from range scanners or from robust stereoscopic techniques for instance. This p function penalizes the surface regions far from the $\mathbf{p}_k$ points. Considering that $\mathbf{p}_k$ is weighted by β_k :

$$p(\mathbf{x}) = \beta_{k_0} \min_k ||\mathbf{x} - \mathbf{p}_k|| \quad \text{with} \quad k_0 = \operatorname*{argmin}_k ||\mathbf{x} - \mathbf{p}_k||$$

  In our current implementation, all the β_k are equal. It would be interesting to evaluate the reliability of the $\mathbf{p}_k$ to give appropriate values to the β_k.

- A visual-hull function v that ensures that the produced surface lies inside the visual hull formed by the contours $\{C_i\}$. So, we can use the information stemming from a known background or from a contour extraction technique. We define v by:

$$v(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in \textit{visual hull}\big(\{C_i\}\big) \\ \infty & \text{otherwise} \end{cases}$$

Summing any of this three functions (c, p and v) to form γ makes possible to deal with the corresponding data types (consistency, points and contours) within the same surface-reconstruction process.

Summary: We characterize the patches with the functional defined in the previous chapter. This functional is just transposed into the local coordinate system. The same graph-cut engine is used to create the patches.

The system is adapted to handle hard constraints (points which the surface must contain) and other data types (3D points and contours) in addition of the images.

3.3.2 Registration in a distance field

This part has been laid down by Gang Zeng.

We have to register all the patches into a common structure. Several choices are possible. Each one has it pros and cons. We have chosen to use a distance field: A cubic grid stores for each of its nodes the distance to the surface. The drawback of this method is that the final surface needs to be extracted with a technique such as the *Marching Cube* [140] that might lose some surface details. In counterpart, it has several advantages:

- It can be incrementally updated (*i.e.* step by step) using the technique described by Curless and Levoy [46]. A new patch is easily added in the structure.
- Its resolution can be locally adapted using a standard octree instead of a regular grid.
- The already built points $\{(\dot{x}_j^{\mathcal{P}}, \dot{y}_j^{\mathcal{P}}, \dot{z}_j^{\mathcal{P}})\}$ are efficiently found using the distance information.

3.3.3 Ordering strategy

This part has been defined in collaboration with Gang Zeng.

The order in which the patches are built is important since a patch depends only on the already existing patches and ignores those reconstructed afterward. Our strategy is to first reconstruct the most "reliable" patches so that the following patches can rely on them. For instance, if accurate 3D points are available (*e.g.* from a 3D scanner), we consider first the patches containing these points before filling the regions without such points. Defining the reliability of a patch depends on the application. We can nevertheless give a few general guidelines:

Accurate input: We may sometimes enjoy precise input data such 3D points from a range scanner. In this case, the patches containing these data should come first.

Functional value: The value of Functional (3.1) indicates how consistent and regular is a patch. Adjacent patches are likely to have similar properties. Neighbors of a regular and consistent patch (*i.e.* with a low functional value) should be considered earlier. If needed, this criterion can be split into two separated items:

- *Image consistency*: Highly consistent points (*e.g.* good ZNCC or photo-consistency score) are likely to be precisely located and should come early.
- *Singular regions*: Discontinuities and high curvatures are known to challenge surface reconstruction. Patches in those regions should be delayed.

Performance: If running time is an important issue, then patches with large overlap with the already created surface should be avoided since they would add a limited area.

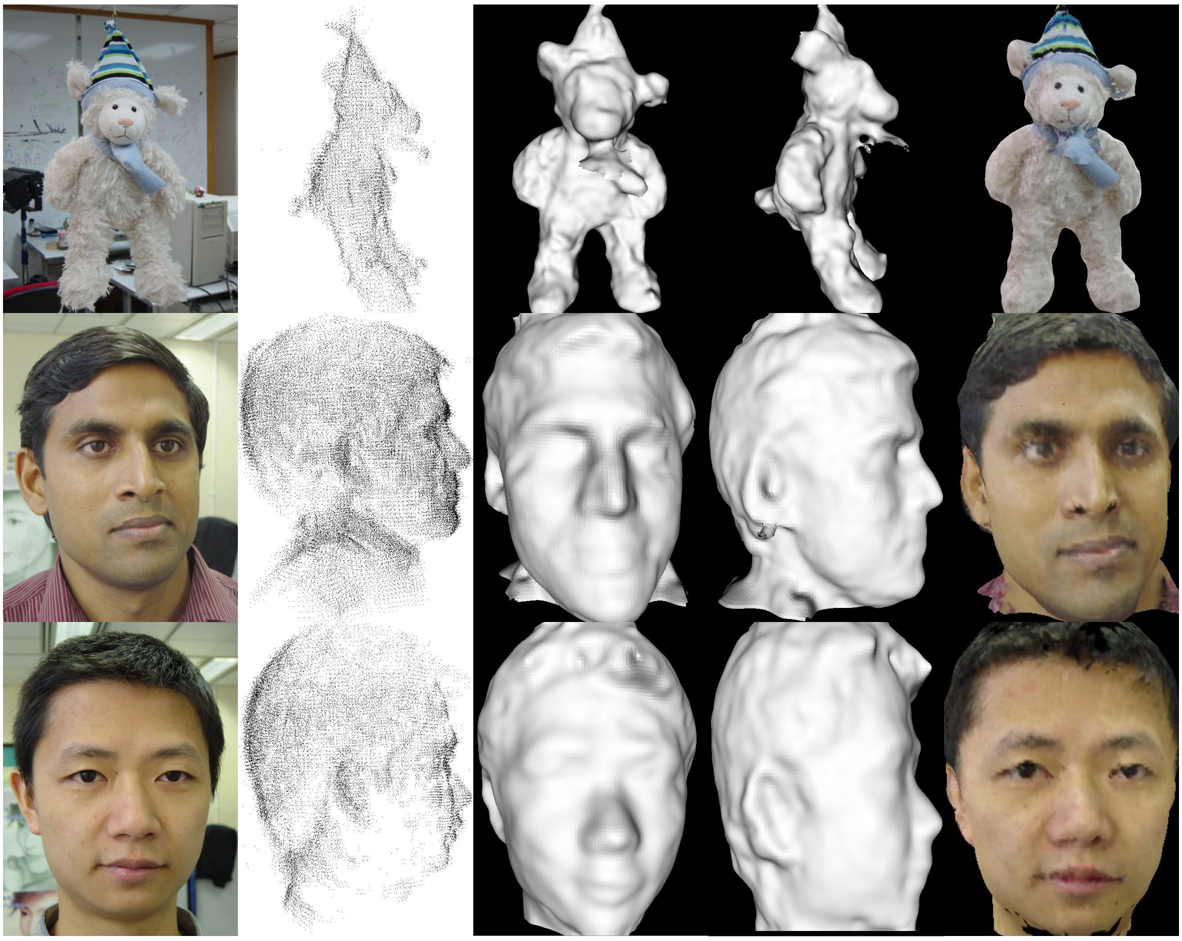
3.4 Two practical algorithms

The two presented algorithms have been designed by Gang Zeng.

We here describe shortly two algorithms that implement the patch approach following the technical choices previously presented. The first method shows the combination of images and 3D points. The second one relies on images and contours to achieve. Both illustrate the patch flexibility and ability to recover detailed shapes.

3.4.1 Propagation of 3D points

This algorithm starts from calibrated images and accurate 3D points. In our implementation, the points are extracted using the *quasi-dense* approach by Lhuillier and Quan [138]. These quasi-dense points are used to initiate a set of *seeds*. These seeds are reliable 3D points on which a patch can be based.



(a) (b) (c) (d) (e)

Fig. 3.3: Results from the propagation algorithm

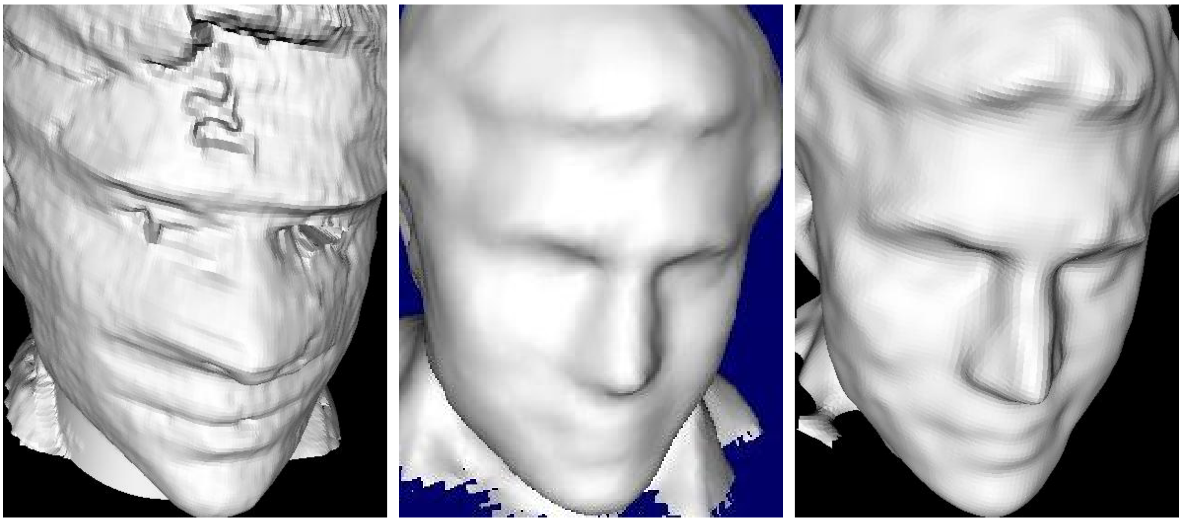
Between 20 and 30 images (600×800*) lying around the object are given as input.* ***(a)*** *Sample input image.* ***(b)*** *Initial 3D points.* ***(c-d)*** *Front and side views of the reconstructed shape.* ***(e)*** *Textured front view.* [Experiment made by Gang Zeng.]

These points are dense enough to allow an estimation of the local orientation of the surface (to align the coordinate system). Then, the algorithm the following loop until no more seeds are available:

1. The best seed among all the available seeds is picked according to the criteria defined in Section 3.3.3 on page 82.
2. If the patch domain is already covered by the previous patches the seed is discarded and a new iteration starts in 1.
3. A graph-cut optimization process is run to generate a patch containing this seed.
4. New seeds are created on the new patch according to the criteria defined in Section 3.3.3 on page 82. The normals corresponding to the selected seeds are estimated using the current patch.
5. The distance field is updated according to the new patch.

Note the difference between Step 1 which selects the best seeds among all the available seeds and Step 4 which "promotes" the best points of the newly created patch into new seeds. The behavior of the algorithm can be summarized as follows:

- First, the input 3D points are extended into patches.
- Then, the remaining holes are filled step by step starting in priority from the most reliable regions.



(a) (b) (c)

Fig. 3.4: Comparison between the propagation algorithm, space carving and level sets

Left: *Result from Space Carving. The surface is poor because of the highlights that foil the photo-consistency criterion.* ***Middle:*** *Result from level sets. The surface is more accurate but is too smooth. This caveat is inherent in level sets (Sec. 2.2.5 on page 23).* ***Right:*** *Our result. It is not yet perfect (observe the cheek) but has finer details compared to the two other techniques.* [Experiment made by Gang Zeng.]

Results

We here present our first results using the propagation algorithm. This work is still on going and we acknowledge that these results should be completed in the future with more experiments. However, these first reconstructions are encouraging and validate the patch concept.

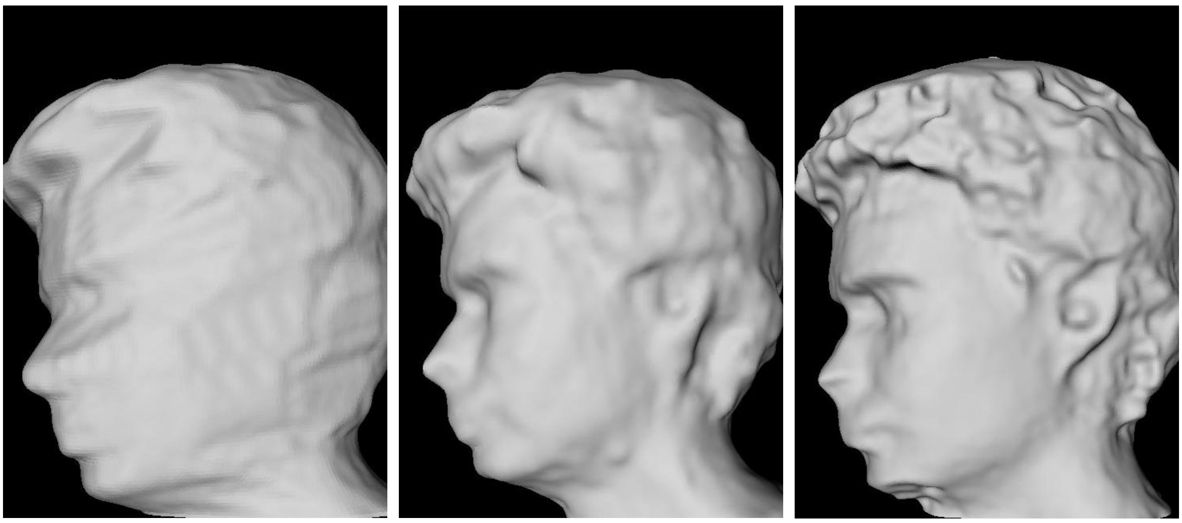
Figure 3.3 shows several reconstructed objects. The first two rows illustrate that our algorithm can behave like a surface-from-points technique similarly to the approaches of Amenta *et al.* [7, 8] or Hoppe *et al.* [91, 92, 93]. In the same time, if the point cloud becomes sparser as in the last row, it exploits the images to properly fill the holes and not only interpolating a surface from the borders.

The first row also shows that complex objects with occlusion (*e.g.* the two legs) can be captured. Visibility is not explicitly handled in our process. Nonetheless the ordering scheme ranks first the unoccluded seeds since their consistency is higher than the occluded ones. The consistency of these occluded seeds is evaluated using cameras that do not actually “see” the seeds but their occluders. It is therefore unlikely to result in a consistent match. This phenomenon delays the reconstruction of the occluded regions until their corresponding occluders have been recovered. Then, the occluded regions are built using the only visible cameras. This scheme implicitly handles the visibility issue.

Figure 3.4 on the facing page provides a comparison with the carving and level-set methods.

3.4.2 Patch-wise carving

This algorithm is built upon the classical *Space Carving* technique [127] (see the Previous Work section on page 19 for more details). The input data are a set of calibrated images with the contours of the object within these images. In our implementation, these contours are automatically extracted



(a) (b) (c)

Fig. 3.5: Comparison of the carving algorithm with Space Carving and level sets

(a) *Result from Space Carving. The surface is poor because of photo-consistency is not robust to highlights.* ***(b)*** *Result from level sets. The surface is more accurate but is too smooth. This caveat is inherent in level sets (Sec. 2.2.5 on page 23).* ***(c)*** *Our result has finer details compared to the two other techniques. It seems however more sensitive to the highlights in the hair region. This results in some over-carved points.* [Experiment made by Gang Zeng.]

from an unknown background using the method of Zeng and Quan [237] (except for the sequence used in Figure 3.6).

From these data, we compute the visual hull of the object and discretize it into voxels. Our voxels have a size typically one order larger than classical voxels: They project on several pixels. The algorithm is then very similar to Space Carving: The volume is swept in several direction. During each sweep, the voxels are examined one by one (see Figure 2.14 on page 20): The inconsistent voxels are carved out. The difference with the original process is our carving criterion. Instead of using the point-wise photo-consistency, we run a graph-cut optimization to build a patch. The current surface estimate is used to determine the local coordinate system. If the resulting value of the functional is low enough (indicating a consistent patch), the patch is added into the distance field. Otherwise, it is discarded and the voxel is carved.

This method can be seen as a *regularized carving*. The global algorithm layout is the same as Space Carving but we have regularized the problem by accounting for a patch instead of a single point. This approach makes the problem well-posed.

Results

Figure 3.6 and Figure 3.7 outline the behavior of our carving algorithm: It starts from a robust shape estimate given by the visual hull and refines it using the patches. Figure 3.5 illustrates the same

Fig. 3.6: Results of the carving algorithm on a skull

Between 20 and 30 images (600×800*) lying around the object are given as input. Background has been manually extracted.* ***First column:*** *Visual hull of the model.* ***Second column:*** *Final result.* ***Third column:*** *Textured result.* ***Fourth column:*** *Sample input images.* [Experiment made by Gang Zeng.]

remark that we have formulated for the propagation technique. The patches extract finer details than a traditional Space Carving and than level sets.

3.4.3 Discussions

Similar results A first remark is that the results produced by both techniques look very similar. They both enjoy sharp and fine details. The counterpart is that errors are noticeable because they introduce spurious sharp features (*e.g.* the hair in Figure 3.5 on page 85).

This gives an evidence that the underlying patch approach is responsible for the overall quality of the results. The practical implementation characterizes more the addressed scenario and the exploited data than the final result.

Easier implementation An important point to know is that both algorithms have been coded using the simple "linear" graph shown in Figure 2.25 on page 45 and without the discontinuity detection described in Section 2.6.2 on page 57.

The linear graph does not produce flat and blocky results as one could expect. This comes from the local coordinate system that is adapted according to the local estimate of the surface tangent. Hence, a "flat" surface is in fact a surface parallel to the surface estimate and may be curved. Therefore, the surface flatness is less noticeable. And there is almost no spurious discontinuity because the reconstruction is constrained by more data (images and points or silhouettes). However, on can observe some errors such as the teeth of the skull (the "mouth hole" is partly filled) coming from this artifact. Implementing discontinuity detection should remove these few errors.

Faster algorithm Since the patches require less resource, we do not use our code dedicated to huge graphs. This code does a lot of on-the-fly computations to avoid storing useless values. Here, storage is no more a constraint so we can use a more efficient code.

Our code, is currently based on the BOOST library [23] that implements the same *push-relabel* algorithm with stored values instead of on-the-fly computations. A further improvement would be to use the algorithm described by Boykov and Kolmogorov [27] whose running time is shorter on small graphs.

Fig. 3.7: Results of the carving algorithm on faces

Between 20 and 30 images lying around the object are given as input. ***First column:*** *Visual hull of the model.* ***Second column:*** *Final result.* ***Third column:*** *Textured result.* ***Fourth column:*** *Sample input images.* [Experiment made by Gang Zeng.]

3.5 Conclusions

This is still on-going work. Several issues have still to be explored:

- Influence of the patch size.
- Influence of the overlapping regions.
- Other ordering strategies.
- Other graph-cut algorithms.

More results are also needed to fully validate the approach. It would be especially convincing to build an object with a non-zero genus (*i.e.* with holes) and to reconstruct a larger object.

Nonetheless, the results already obtained fulfill our expectation. We enjoy the precision of our graph-cut reconstruction scheme on more general configurations. Cutting the surface into pieces preserves the essential qualities of the optimization process while tackling the scalability and parameterization issues.

We believe that the patch approach is widely applicable beyond the graph-cut algorithms. All the techniques suffer from the "limited resource" problem and several are impaired by the parameterization caveat. Patches propose a general solution to these two important shortcomings while being compliant with all the reconstruction techniques: It does not change the stated problem but only the surface domain and the coordinate system. Therefore, we are convinced that the patches have a great potential and should be considered as a general "design pattern" for reconstruction algorithms.

4
Face relighting

4.1 Introduction

In this chapter, we study how we can use a real picture to produce a new one under a different lighting condition. For instance, we aim at changing a day-light picture to make a night-time one. The main application of relighting is to pre-process images that are to be combined into a single image. If this aspect is neglected, it strongly impedes the consistency of the composited image: the different parts of the image look like they have been made separately and just pasted side by side, they never merge into a single picture. Examples of typical pitfalls that may degrade a composition are: wrongly placed highlights, shaded regions facing the light, missing shadows... All the details may be hard to notice at first sight but undoubtedly degrade the overall perception of the image and convey an uncomfortable feeling to the viewer. This is even more important for animation *e.g.* if a light is moving whereas the shadows in the scene are still, it breaks down the overall realism.

Relighting is naturally split into two steps. First the lighting cues of the original picture are removed and then the ones corresponding to a new lighting environment are added. Both steps require some knowledge about the scene content:

- The geometry of the objects influences the shape and position of the shading, shadows, highlights, etc.
- The materials drive the appearance of these cues (think of a mirror ball compared to a wooden ball for instance).
- The lighting configuration has also a clear impact. Both the color and the geometry of the light sources modulate the final image. For instance, under a cloudy sky, there is almost no shadow whereas the sun light produces sharp and contrasted shadows.

Considering the quantity of needed information, we believe that the general case in not tractable. The number of special cases would be too high.

Hence, we have chosen to focus on human face. It is obviously a key feature of a person. It is the first thing that we look at. It is perhaps the most important part of the body in the sense that a realistic rendering of a person cannot be achieved without a realistic rendering of the face. As a consequence, we believe that producing satisfying images of a human face is much harder than rendering a common object. We are so used to observe faces that our tolerance to error is very low whereas we accept comparatively large artifacts for "normal" objects such as a teapot. For instance, the acceptable range of colors for the skin is very narrow: A minor red shift can convey a "blemish" or "sun-burnt" aspect whereas a larger one makes the skin look odd. The diversity of existing faces is especially challenging. Coping with all, or at least a significant proportion of, the possible faces is not straightforward.

Furthermore, rendering skin is not trivial because of its complex interaction with light through many layers (oil, epidermis, blood,...). Many phenomena are involved in the final appearance of the skin. Among all these phenomena, we may cite:

- Its rough surface which has a typical scale inferior to one millimeter.
- Its spatially varying aspect because of scars, spots, etc.
- Its translucency that makes the light interactions with the deeper layers non negligible.
- Its layered structure that introduces inhomogeneous materials (blood, pigments, cells, etc).
- The oil that covers its surface and produces strong highlights.
- The tiny hairs that populate its surface.
- The presence of other materials in middle of the skin region (lips, eyes, teeth, etc).

Beyond these difficulties, there exist some techniques to achieve an accurate rendering of the skin. But because of the previously mentioned difficulties, none of them achieve real-time performances without consuming all the computation power of a consumer-grade graphics card. Once the skin has been rendered, the rendering time for the current frame is almost over. This makes these techniques hardly usable in games or similar applications. They require a rendering engine lightweight enough to allow other computation in the same time *e.g.* character animation. But, it is quite clear that the hardware will increase sufficiently enough in the near future to make these advanced techniques usable. However, it will take a longer time for portable devices such as cellular phones or PDAs to enjoy such a rendering power. There will still have some need for lightweight rendering techniques, even when all the desktop machines will be equipped with powerful graphics hardware.

These remarks have motivated our approach. Our rendering technique is fast (real time) even on widely available graphics cards and is light enough to allow the card to render other objects. To achieve this, we restrict ourselves to a basic rendering engine and target to produce the best images we can. Of course, we do not claim to reach the same realism than a complex model that does not run in real time. Nevertheless we believe that it is possible to produce images with a high visual quality, superior to typical results in interactive real-time applications.

Considering all these issues, we propose in this chapter a face relighting engine which runs with very low hardware requirements. We also define a robust and easily applicable method to acquire the data needed for this engine. Before going into the details of our method, we review in the following section the most significant papers relatively to our goal.

4.2 Previous work

Face rendering involves numerous aspects: modeling the geometry, acquiring the skin reflectance, building a generic model of such a reflectance, synthesizing facial expressions, etc. Most of the existing work covers several subjects simultaneously to achieve their specific goal. Since we are concentrating on relighting, we are especially concerned with reflectance acquisition and rendering issues. In the following pages, we review the techniques the most relevant to our goal.

4.2.1 Fundamental concepts

Reflectance

The lighting properties of a material is described by the notion of *reflectance*. Formally speaking, it is defined as the ratio:

$$\textit{reflectance} = \frac{\textit{exiting light power}}{\textit{incoming light power}} \tag{4.1}$$

Intuitively, it corresponds to the "quantity of light" reflected by the material. It is defined for a given wavelength and is extended to the RGB channels *i.e.* "the quantity of red (blue, green) light reflected".

Analytically, the reflectance is described by the *bidirectional reflectance distribution function* (BRDF) introduced by Nicodemus *et al.* [164]: It is computed for given incident and reflected angles. It assumes that the light-material interaction is point-wise *i.e.* that the incoming and existing rays intersect the surface at the same point. If we consider that this may not be the case, that the light penetrates inside the material and may exit at a distance from the entrance point then, the reflectance is

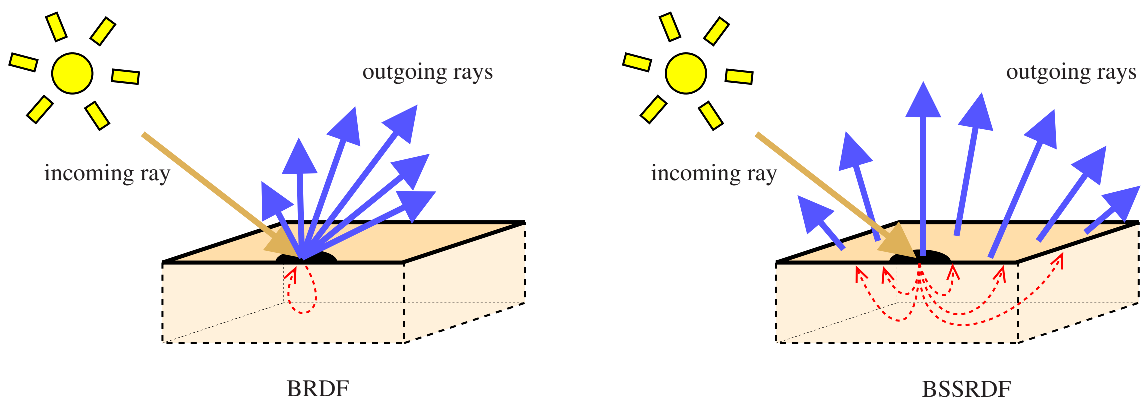


Fig. 4.1: BRDF and BSSRDF

Left: *The BRDF represents the interaction of the light with a material. It is limited to the interactions in which the incoming and outgoing points of the light are equal. For given surface point, the BRDF is a 4D function that depends on the incoming direction (two Euler angles) and on the outgoing one (two more angles).* ***Right:*** *The BSSRDF is an extension over the BRDF to model more complex interactions: An incoming ray of light diffuses inside the material and may partly go out at surface points distant from the incoming one. For a given point, it has two more parameters compared to the BRDF: It also depends on the outgoing surface point (two surface coordinates).*

described by the *bidirectional surface scattering reflectance distribution function* which also depends on the exit point in addition to the two directions. Figure 4.1 gives more details.

From a practical point of view, if the lighting environment and the reflectance of an object is known, then we can compute its appearance. That is why the reflectance notion is important for the relighting issue since it characterizes the relation between the aspect of an object and the light.

BRDF BSSRDF

Fig. 4.2: Comparison between BRDF and BSSRDF on skin

The difference between a BRDF and a BSSRDF rendering [105] is clearly visible for a finely detailed mesh as this one. The subsurface scattering smooths away the bumpy appearance of the mesh. Remark also that it produces softer shadow boundaries (near the nose). [By courtesy of Henrik Wann Jensen]

4.2.2 Reflectance acquisition: Sampling the reflectance

A natural approach is sampling the reflectance from real materials for several light and camera positions and to store these measures into a database. From these data, there are several techniques to infer the aspect of the materials from any viewpoint under any illumination condition. These mainly involve interpolation and integration of the most relevant measures in the database. We concentrate on the proposed acquisition techniques.

Numerous acquisition systems exist; among them, we may cite Ward [221], Rushmeier and Bernardini [182], Marschner *et al.* [148], Debevec *et al.* [50], McAllister [156], Matusik *et al.* [153, 154, 155]. These systems mainly acquire BRDF data (*i.e.* they assume no subsurface scattering). They differ on the kind of material they can handle and on some additional data that may be captured (*e.g.* precise contours of the object in [154]). BSSRDF is more complex to measure since it deals with a broader range of effects. Nonetheless, some techniques have been designed recently to sample the effects of subsurface scattering and evaluate a BSSRDF:

Link with* [Light field]*: *Levoy and Hanrahan [136] and Gortler et al. [75] have introduced the light field approach. As the reflectance measure, they densely sample the rays of light exiting the scene. Nevertheless, they handle a general scene instead of a material probe but restrict to fixed lighting conditions. From this information, a new image is created by collecting the rays that hit the image plane. The difficulties of these techniques lie in acquiring, representing and storing this huge amount of data. Considering that we target relighting, the fixed lighting conditions is an important drawback. We refer the interested reader to the work from Hakura et al. [82] which is a first step toward coping with varying lighting environment.*

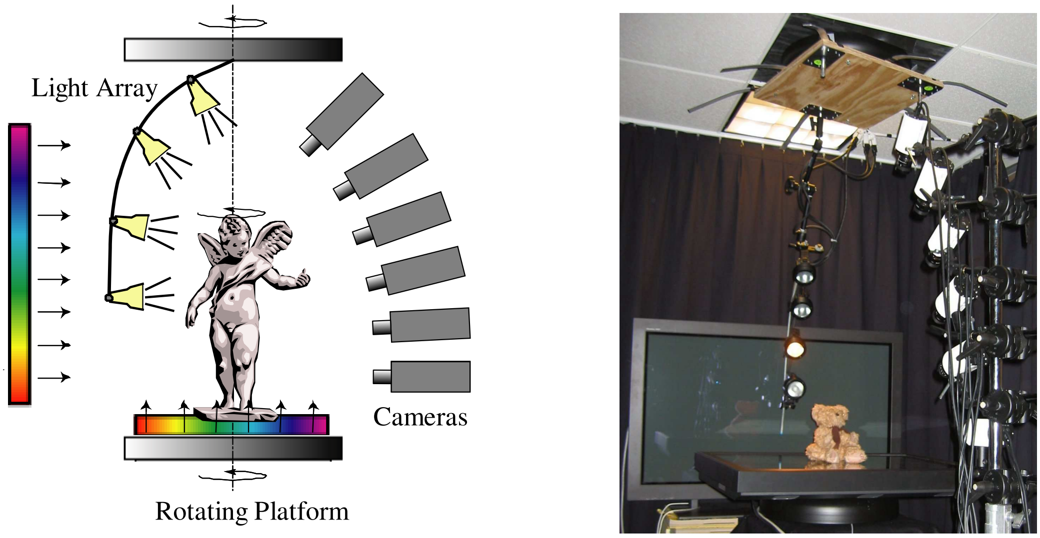


Fig. 4.3: Capture system of Matusik *et al.* [154]

The system used by Matusik et al. [154]: Dense sampling of the appearance of an object yields accurate rendering but requires a complex and cumbersome system. [By courtesy of Wojciech Matusik]

Jensen *et al.* [105] describe a method for homogeneous material and Goesele *et al.* [74] build one for more general materials. A common point to all these systems is that they need at least a dedicated apparatus. They even sometimes need a dedicated room because they use a robotic gantry or similar techniques to acquire images from controlled viewpoints and light positions. Therefore, although these techniques provide very accurate and dense measures yielding to high quality rendering, these systems are too cumbersome to fulfill our goal. We target a simpler setup acquisition setup that, ideally, does not involve dedicated apparatus. However, if one aims at precise measures, these approaches are the ones to be inspired from.

Furthermore, the amount of acquired data is generally huge (2000 photographs in [50]) because of the dense sampling of the 4D or 6D reflectance function. This induces slow rendering techniques because of complex requests in large databases (several minutes in [50]). This does not correspond to our objective of efficient rendering. But, if rendering time is not a crucial issue compared to the produced quality, these are the methods to apply.

Another drawback relatively to our goal is that only the system of Marschner *et al.* [148] is demonstrated on faces. Working on faces is hard in general because one cannot work for hours on a still material sample.

Summary: The sampling approach densely measures the aspect of a material under several light and view positions. This requires dedicated systems to control the light source and the camera. This results in cumbersome processes that are complex to use, especially for faces. Additionally dealing directly with the large amount of captured data leads to slow renderings.

Therefore, even if these methods are accurate and produce very high quality images, they do not meet our requirements on capture setup and rendering efficiency.

4.2.3 Reflectance acquisition: Parameterizing the reflectance

From the previous remarks, several approaches represent the appearance of a material by a formula which analytically computes the reflectance value from the viewpoint and the lighting condition. This formula depends on several parameters that describes the material aspect. This straightforwardly results in much more compact representations (a few parameters instead of thousands of images) and generally achieves faster computation because it avoids database search.

"Imitation" models

The very first models originally aimed at reproducing the overall appearance of an object. The physical origin of the shading is only an inspiration to design some functions that "mimic" real shadows and highlights. Among this category, the most famous model has been proposed by Phong [171]. It is the only model of this type still in use nowadays: All the cards implement it at least per-vertex *i.e.* the color is computed on the mesh vertices and then interpolated on the triangles (this techniques is also known as the *Gouraud shading* [77]). This may lead to visible artifacts for large triangles because the color of a face is computed only from its vertices, ignoring any lighting variation that occurs in between. With the new programmable hardware, it can be now computed per-pixel: the lighting computation is made for each pixel and correctly accounts for the lighting variations up to this scale (which is sufficient in all practical cases).

This model is simple but is able to model, or at least approximate, a broad range of materials. Although a more complex model could be used, our implementation is based on it to enjoy the widely available hardware implementation. We therefore give more details.

It approximates the shading by the sum of three components:

Ambient: This is a constant term that is due to the indirect light coming from the surrounding environment. This is typically a low intensity term. It describes the appearance of an object that has no direct lighting (*e.g.* in the shadows).

Diffuse: It represents the light that interacts "in depth" with the material. It is assumed to be constant in all the directions exiting the surface. It depends only on the incoming light direction.

Specular: It describes the highlights due to a surface reflection. The size of the highlight varies with the material. The reflected intensity depends on both the incoming light direction and the view direction.

Denoting $\mathbf{n}$ the surface normal, $(\mathbf{v},\mathbf{l},\mathbf{r})$ the view, light and reflection directions (see Figure 4.4); the intensity of the channel $X \in \{R,G,B\}$ is:

$$I_X(\mathbf{v}) = \underbrace{X_\mathrm{l}^\mathrm{A} X_\mathrm{m}^\mathrm{A}}_{\text{ambient}} + \underbrace{X_\mathrm{l}^\mathrm{D} X_\mathrm{m}^\mathrm{D}\, \mathbf{l}\cdot\mathbf{n}}_{\text{diffuse}} + \underbrace{X_\mathrm{l}^\mathrm{S} X_\mathrm{m}^\mathrm{S}\, (\mathbf{r}\cdot\mathbf{v})^s}_{\text{specular}} \tag{4.2}$$

where $^\mathrm{A}$, $^\mathrm{D}$, $^\mathrm{S}$ stand for the ambient, diffuse and specular components; and $_\mathrm{l}$, $_\mathrm{m}$ for the light and material values. For instance, R_l^A is the ambient light intensity in the red channel (*i.e.* the "quantity" of red light coming from the reflection on the surrounding environment),

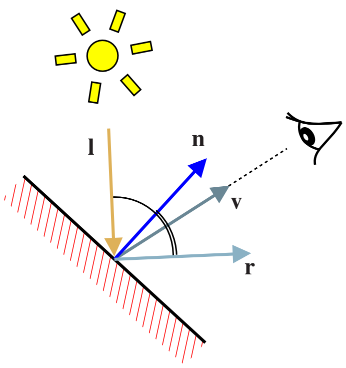


Fig. 4.4: Notations: $\mathbf{l}$, $\mathbf{n}$, $\mathbf{r}$ and $\mathbf{v}$.

and B_m^S is the specular response of the material to blue light (*i.e.* the proportion of the incoming blue light reflected as an highlight). The exponent ^{s} controls the width of the specular lobe. Typical values range from 2 (slightly glossy) to 1000 (almost reflective). Note that it makes the formula (4.2) non-linear. Figure 4.5 illustrates this model.

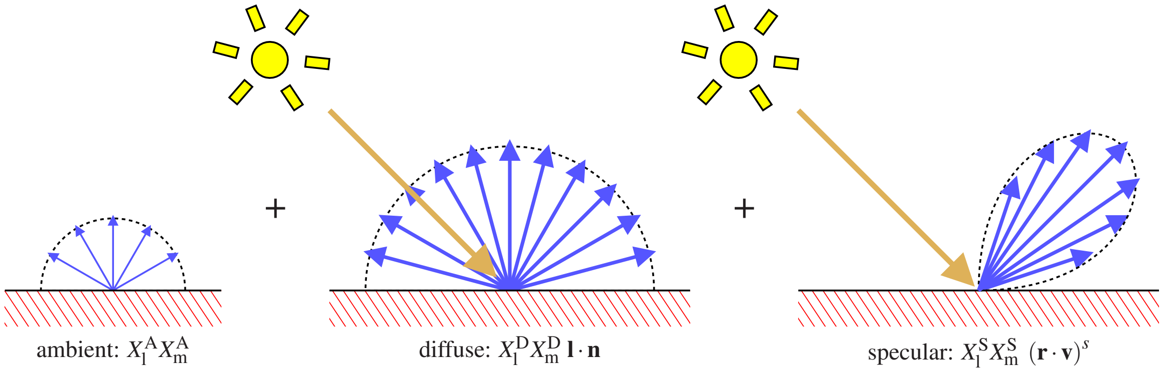


Fig. 4.5: Phong model

It decomposes into three components. The ambient *term represents the indirect lighting and is constant and of low intensity. The* diffuse *term is the omni-directional reflection due to the direct lighting. The* specular *term corresponds to the highlights and depends on both the light and view direction. The formulæ are given with the notations used in equation (4.2).*

Physically based models

Building upon the Phong model previously presented, Blinn [20] proposes a model motivated by the physical analysis of micro-faceted surfaces by Torrance and Sparrow [213]. Even if the resulting appearance may sometimes be similar with Phong's model, the main difference is that this model intends to match the real material behavior and not only mimics its appearance. Therefore, the parameters of the model are physically meaningful values *e.g.* the micro-facets distribution for [20]. Exploring the same approach, several researchers propose physically-driven models such as Cook and Torrance [42], He *et al.* [89], Oren and Nayar [168], Ashikhmin *et al.* [10], etc.

Among all these models, it is interesting to focus on the few ones that describes volumetric or layered materials such as Hanrahan and Krueger [84], Jensen *et al.* [105], Stam [202] and Premoze *et al.* [175]. All these models show that they are appropriate to render skin since it is composed of two main layers: the epidermis (outer layer) and dermis (inner layer). For these models, the typical parameters are the thickness of the layers, their absorption coefficients, the roughness of their surface, etc. So they allow a fine control over meaningful and observable characteristics of the skin (*e.g.* color and "hardness" of the shadows). Unfortunately, these models suffer from two limitations. First, they can be computationally expensive, even for the latest graphics cards. They involve complex equations to evaluate the layer interaction and the light scattering within the material. Second, to our knowledge, there is no method to accurately retrieve the parameters corresponding to the skin of a real person. And it seems to be especially hard because, as just mentioned before, these models are complex. Tsumura *et al.* [216] propose an image-based analysis of the skin that may be a first step toward characterizing a set of parameters from a real skin. They quantify the contributions of the hemoglobin (the red component of blood) and of the melanin (the brown pigment of skin). Their evaluation is proven correct by medical experiments. Then from these data, they are able to re-synthesize the same face with a paler aspect (less hemoglobin) or a sun-burnt one (more melanin). But this is still not yet a complete solution to this difficult point *e.g.* changes in the lighting condition are not handled. A complementary model is exposed by Koenderink and Pont [116] to describe the characteristic halo of the skin due to the tiny hairs that populate the human skin surface. This can be a complement to the previous model to achieve a highly photo-realistic rendering.

Approximating models

As previously discussed, direct samplings of the reflectance or physics-based models are computationally expensive because of large data sets or complex mathematical expressions. This motivates alternative methods relying on simpler representations. These techniques approximate the data which represent a given BRDF with a formula that is more suitable for rendering purpose (*e.g.* faster computation). The BRDF data can be either measurements on real materials or values computed with a parametric model among those presented before. Ward [221] uses Gaussian distributions to fit these data. Lafortune *et al.* [128] extend Phong's model with several lobes. Matusik *et al.* [152] parameterize the space of the BRDFs by a non-linear manifold of dimension 15. Lawrence *et al.* [131] decouple the BRDF into a product of three terms depending at most on two variables.

Skin approximation: From their study on real skin, Marschner *et al.* [148] show an important result concerning all these parametric models: Generic models such as Lafortune [128] and He [89] cannot capture all the specificities of the skin appearance *e.g.* the reflection at grazing angles cannot be represented. Only the dedicated models such as Hanrahan-Krueger [84] can.

Therefore, the trade-off is between the complex but accurate models and the efficient but approximate models.

The resulting parameters have a limited physical meaning since these models mainly target accurate fitting and are designed from analytical considerations and not physical ones. A typical use of these models is:

1. Capturing data from a real sample or designing the appearance with a physical model.
2. Approximating this material by one of these efficient models.
3. Rendering the scene using this model.

This has the advantage to produce almost equivalent images in a much shorter time. However, one has to consider using the original data or model whenever the numerical precision matters. This strategy is suitable only if one can accept some numerical approximation. This is typically the case when only the visual quality is important. Since it corresponds to our case, these approximating models would be a natural choice for our application. Although we use a Phong model in our implementation, we believe that using one of these models would be an interesting future work to extend the technique described in this chapter.

Summary: The parametric models are simpler representation of the reflectance than a sampling of a real material. An analytic formula embeds the interaction of the light with the material. It results in more compact data structures and more efficient computations. Therefore, we have chosen to rely on such a representation for the skin.

The choice among these models is broad. According to our goal, the trade-off lies between efficiency and accuracy. Unfortunately, the simplest models are proven to be only approximations and the accurate models involve higher computation times. In practice, our implementation is based on the simple Phong model and already shows convincing results. However, other models presented here could be used.

4.2.4 Reflectance acquisition: Retrieving the parameters

Acquiring the parameters of a model from a material sample that can be measured as densely as needed is a rather simple problem. This is a standard fitting problem that is generally addressed by the papers describing the considered model.

A more difficult problem stems from the practical case in which the available data are limited. The problem is again a fitting issue but much harder. For instance, Ikeuchi and Sato [98] recover both the reflectance parameters and the light position from a single image. But their technique seems hardly usable because of the numerous thresholds to be finely tuned. Sato *et al.* [186] assume the light position to be known and propose a more robust algorithm. Nishino *et al.* [165] propose a variant of the method of Ikeuchi and Sato [98] that is more robust but still limited to simple objects and lighting environments. Boivin and Gagalowicz [22] expose an approach in the case of a single picture of a scene with several materials of known boundaries. Lensch *et al.* [135] use several pictures but the regions for each material are unknown.

All the previously cited methods are general and we believe that they might be hard to apply in some cases on skin because of its very specific lighting behavior. Let's now observe the techniques dedicated to faces in more details. Georghiades *et al.* [72] show that – under a fixed pose, ignoring shadows and assuming that the skin aspect is purely diffuse (*i.e.* with no highlight) – the images of a face form a 3D vectorial space. From this, they parameterize the face aspect from a few pictures without shadow (*i.e.* with front-facing lighting). When rendering new images, they can correctly add the shadows using ray-tracing. Only the acquisition step works without shadow.

Marschner and Greenberg [145] propose a face relighting also based on the diffuse assumption. From an input photograph and 3D model of the face, they first determined the lighting environment. Then they can modify the photograph in order to adapt to a new environment.

An important point to notice for this method and the previous one is the Lambertian assumption which prevents them from rendering highlights. In practice, this means that the highlights are fixed on the face *i.e.* even on the relighted face, the highlights still appear on their original location. This may not be a so important drawback for still picture in real environment since it may be hard to relate the specularities on a face with the lighting conditions. But as soon as we consider animation, the specular component is fixed whereas the shadows and the "relighted" diffuse component move according the changes of the light. This breaks down the coherence and the realism of the animated sequence.

Fig. 4.6: BSSRDF rendering of a face from acquired parameters

This rendering [103] is performed from an extremely fine mesh (top left) that produces a bumpy aspect when rendered with a purely diffuse material (top middle). The base skin color (top right) is acquired from real pictures. The final image is decomposed into two main components: The highlights due to the oily layer which covers the skin (bottom left) and the deeper light interaction simulated with BSSRDF (bottom middle). The final image (bottom right) takes one week to be computed. [By courtesy of Henrik Wann Jensen]

To address the glossiness of the skin, Blanz and Vetter [19] adapt a Phong model to match the skin aspect in some real images. Hence, they are able to produce complete relighting, including the highlights. However they rely on the user to set the shininess exponent in order to reduce the problem to a linear fit (see formula (4.2)). This produces satisfying results but it would be more satisfying not to rely on the user. This would provide reproducible parameters *i.e.* independent of the user ability to evaluate a Phong exponent. And we can expect a better evaluation of the parameters if they come from an objective measure.

On the other side, Debevec *et al.* [50] show how to recover the parameters to describe the specular component of a face according to the Torrance-Sparrow model [213]. They rely on a dense sampling (2000 pictures) to render the diffuse contribution of the face. This last step is too slow to fulfill our goal. It would be however interesting to study a combination with Blanz and Vetter's technique [19] to parameterize the diffuse part.

Marschner *et al.* [146] use a Lafortune model [128] with generic parameters measured on a real person [148]. They only modulate the skin color using two pictures of the actual persons to be rendered and lower the shininess of the bottom part of the face to match practical observations. Of course, it would be possible to directly apply the complete technique described in [148] to derive the Lafortune parameters of the actual person. But this would require much more time and manipulation.

Jensen [103] describes a face rendering using his subsurface scattering engine whose parameters are tweaked from real images. This tweaking involves heuristics that are not described in this short paper. However, this is perhaps the most advanced face rendering today (Fig. 4.6 on the facing page) but it takes one week to perform.

Summary: Retrieving the parameters of a reflectance model is generally handled as a fitting problem: How to match as close as possible the aspect in the materials seen in the input pictures using the given model?

Techniques exist for common materials but it seems that the techniques dedicated to skin are more limited. Some ignore the glossiness of the skin. Other retrieve only a subset of the parameters, relying on the user to set the remaining ones. And more accurate techniques would involve a complex acquisition process.

We believe that a new balance can be found among all these dedicated techniques. We describe in this chapter a method that works from a light acquisition system while coping with the specular aspect of the skin.

4.2.5 Texture enhancement

Rendering the whole face with a skin material using only the reflectance models previously described would miss obvious features such as the mouth, the eyes, etc... One solution is to add specific 3D models for these parts. This approach is followed by Marschner *et al.* [146] and Tarini *et al.* [209]. On the one hand, this produces detailed features that tolerates close-ups. This also allows animations: The mouth can speak, the eyes blink, etc. On the other hand, these added parts are generic *i.e.* they are slightly adjusted to fit the face model but the same base shape is used for every face. This clearly impairs the fidelity of the overall model with the original faces. Since we do not target animation in particular, we do not use such a technique however the method described later in this chapter is fully compliant with it.

Another way to add these details (mouth, eyes,...) is to directly texture-map them over the skin. This is the approach we have chosen. Furthermore, this has another advantage. The mesh is likely not to be fine enough to represent all the tiny bumps at the skin surface (see Figures 4.2 on page 94 and 4.6 on the facing page). Such a fine mesh results in very high computation time (one week for Figure 4.6). Therefore most of the techniques use a coarser mesh that would give an unrealistically smooth aspect to the skin if used "as is". Texture-mapping the mesh is then an efficient mean to add this slightly rough appearance to the skin. Rushmeier et al. [184] have studied the perceptual effect of various situations: Precisely texturing a smooth surface is one of the cases that give the best improvement. This explains why texture-mapping is popular to improve the appearance of a 3D mesh: Rushmeier *et al.* [183] refine the normals and colors of their model, Loscos *et al.* [141] correct the shading of the scene after adding/removing an object,

> ***Another view on* [Texture on shading]*:*** *If the shading of the model is computed from a parametric model of reflectance, adding the fine details (such as the skin roughness) with a texture map can be seen as adding the* residual *of the fitting process which has determined the model parameters. The residual is the error that remains between the fitted model and the input data. It typically contains these local variations that convey roughness and that cannot be acquired by a global fitting. In this case, texture-mapping is a way to take this residual into account.*

Stamminger *et al.* [203] display plausible textures while the system runs a more precise but longer computation in background, Liu *et al.* [139] add facial expression on a mesh, etc.

Shading and texture: An important result from the previous work [139, 141] on shading and texturing is that the texture combination with the underlying color must be multiplicative. This preserves the underlying shading.

This comes from the multiplicative influence of the light power on an image aspect. Intuitively, doubling the power of a scene lighting produces twice brighter images. Using an additive combination would produce features glowing in the dark even without any light source since non-zero intensity values would be added at some points. We let to the interested readers the analytical details of this property since a rigorous proof from the rendering equation [109] would not be helpful to the following discussion.

The approach most similar to the method exposed in this chapter is from Wen *et al.* [227]. They use a reference sphere to capture the incoming light. The skin is modeled as a purely diffuse material. From this assumption, they derive an spherical-harmonics analysis to extract an *albedo* map of the skin (*i.e.* the base color of skin independently of the shading). This map is made from a constant color modulated by high-frequency details. It can be seen as a classical texture map that is modulated by the shading intensity to produce the final picture. From all these data, they generate convincing relighted faces (see Figure 4.7 on the next page). But, as previously discussed, because of the Lambertian assumption, the highlights never move thus impairing the consistency of the produced pictures, especially in animation.

Summary: Texture mapping is a simple and efficient way to represent both the large features of a face (nose, mouth, eyes, etc) and the more subtle rough appearance of the skin that is not caught by a reasonable 3D mesh (capturing these tiny details requires sub-millimeter precision).

Previous work shows that such a texture should be multiplicative to preserve the shading properties of the underlying material.

4.2.6 Complex material rendering

With the recent advances in hardware-accelerated rendering, several techniques have been proposed to render complex material in more or less complex lighting environments. Several aspect are covered by these methods, we may cite:

- The bumpy surfaces by Kautz and Seidel [111].
- The woven clothes by Daubert *et al.* [48].
- The spatially varying materials by McAllister *et al.* [157],

Fig. 4.7: Face relighting with Lambertian assumption

Top row: *Ground truth (real pictures).* ***Bottom row:*** *Relighted faces using the method of Wen et al. [227]. Because of the Lambertian assumption, the highlights do not move. This can be observed more clearly when the lighting is significantly different from the input one (first and last columns), especially on the nose, around the eyes and on the forehead.* [By courtesy of Zhen Wen]

- The complex materials in complex environments by Ramamoorthi and Hanrahan [176, 177], Ng *et al.* [162, 163], Latta and Kolb [129] and Sloan *et al.* [198, 199].
- The BSSRDF rendering by Jensen *et al.* [104], Lensch *et al.* [134] and Mertens *et al.* [159, 160].
- The surface details by Tong *et al.* [212], Wang *et al.* [220] and Sloan *et al.* [200].

The methods that seem the most suitable to skin rendering are the BSSRDF ones but even the fastest one is still too slow to meet our requirements: Mertens *et al.* [159] propose perhaps one of the most efficient methods available today for skin and nonetheless reach a frame rate only in order of 5Hz. In general, the main drawback of these methods considering our goal is that they only achieve real time rendering for a single object even with the latest available hardware. However, these methods will be fully in the future, the time to wait depending only the type of machine targeted (advanced hardware will take longer to appear on cell phones).

All these techniques tend to achieve efficient representations of the object appearance. Recently, complementary approaches have appeared to efficiently handle the lighting environment: Ng *et al.* [162] use wavelets to compactly represent high and low lighting frequencies, Agarwal *et al.* [2] and Ostromoukhov *et al.* [170] determine a few directional light sources to approximate the original dense lighting environment. We are especially interested in this result:

Complex environment approximation: If we are able to perform an efficient rendering for a single directional light source then complex environments can be approximated by summing the contributions of several sources determined by one of the previously mentioned techniques.

Finally, Debevec *et al.* [51] propose an original solution to obviate the face relighting problem. First, they measure a target lighting environment varying in time. Then, using an immersive lighting stage, they directly shoot a video of a real person in the captured environment. This hybrid approach is by definition photo-realistic but requires a dedicated light stage and suffers from all the limitations inherent in live shooting (need for a real actor, compositing, etc).

Summary: Techniques exist to render complex materials – and among them, some are able to deal with skin. But they still requires the entire power of the graphics card; hence impeding a combination in a full interactive system.

Related work also exposes how to decompose a complex lighting environment into several directional light sources. This shows that we can restrict our study to the simple case of one directional source and then produce complex results by combining them.

4.2.7 Discussion

From this review, we believe that numerous useful tools exist to handle the skin. It could be certainly possible to combine some of them to achieve face relighting in a satisfying way. Unfortunately, because of the skin specificities discussed in Introduction (cf. page 91), we believe that quality renderings require a dedicated solution. For instance, it seems that there exists no simple and robust data acquisition setup for faces. The existing ones are either relatively cumbersome or partial (*i.e.* they rely on some additional user input). Concerning the rendering, most of the methods target first image accuracy and yield to computationally expensive solutions. On the other side, if one looks at the face in games, they are still using simple textured model with limited light interaction. Hence, we believe that a new balance exists to achieve quality images while using the standard capabilities of the consumer-grade hardware. Of course, we cannot clearly produce as perfect images as the most advanced but slow techniques but we are convinced that satisfying approximations can be made.

4.3 Overview of the technique

The method exposed in this chapter presents both a lightweight rendering engine dedicated to face relighting and the techniques to acquire the input data needed by this engine. In this section, we give an overview of the engine and characterize the data that the acquisition process has to produce.

First, we use a 3D mesh of the face we are working with. To acquire this 3D geometric model, we have explored various acquisition methods. We have mainly used a 3D scanner which directly provides a precise mesh. We have also done some early work with models obtained from Computer Vision techniques, which use only a camera to acquire the data. In our algorithm, this 3D mesh is given off-line by the user *i.e.* we use it without any a priori about its origin and it does not change from frame to frame.

Naturally, we also need information about the scene configuration: the face and light positions and the camera setup (*i.e.* its 3×4 projection matrix). These data can change dynamically between two rendering phases to allow animations of camera, lights and face (only rigid transformations are handled in our system but there is no intrinsic limitations against facial animations).

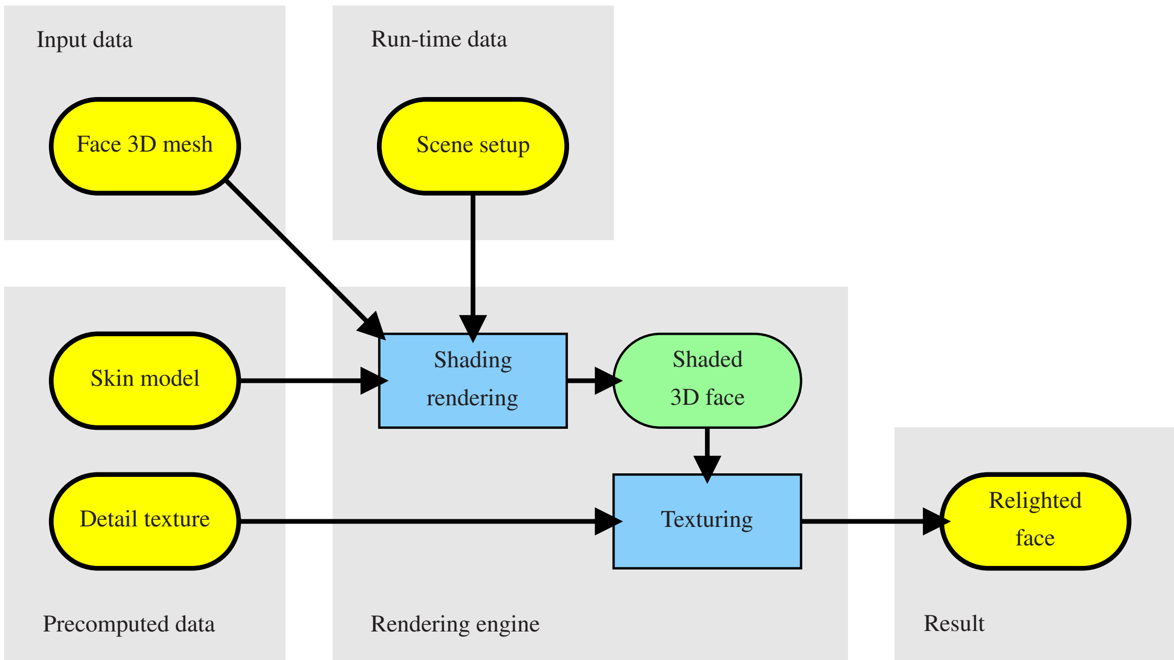


Fig. 4.8: Overview of the rendering engine

First the engine uses the 3D mesh of the face and a model of the skin appearance to render a shaded version of face with only skin and no feature (no eyes, no mouth,...). Then a texture is applied to add all the missing features to produce the final result. A challenge for the engine is to be able to adapt in real time to the scene variations: the light, face and camera may move and we want to produce efficiently the corresponding images.

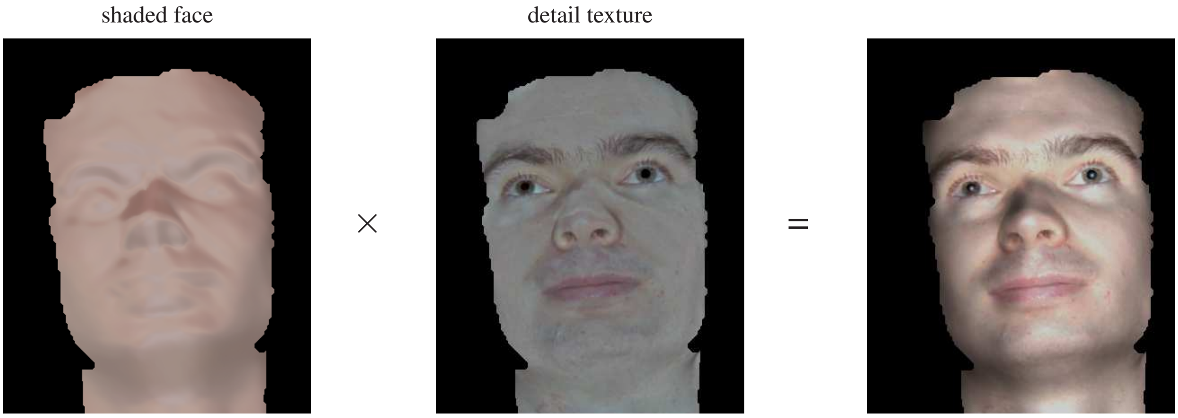


Fig. 4.9: Sample combination between a shaded 3D face and a detail texture

The shaded 3D face is rendered as if it were only composed of skin. It is composed afterward with the detail texture to add all the missing features (eyes, mouth, etc) and the skin roughness.

We have chosen to separate the rendering into two steps:

1. The 3D mesh of the face is rendered as if it were only composed of skin. This results in what we call the *shaded face*. For this rendering, we use a parametric model of the skin which is precomputed from a photograph of the real face. Intuitively, this step renders all the lighting cues: shading, shadows and highlights.

2. Then a texture (the *detail texture*) is combined with the shaded face to produce the final result. As shown in the previous work section (cf. Sec. 4.2.5 on page 101), this combination must be multiplicative. The detail texture is also precomputed from the real photograph. Intuitively, this step adds all the lighting-independent features of the face.

Figure 4.9 illustrates the final image formation and Figure 4.8 shows the global organization of the rendering engine.

How to precompute the parametric model and the detail texture are two main aspects of our method. This will be exposed in the following sections. Both use a photograph of the real face taken in a dark room using only a flash light. They also need a picture of a spherical mirror taken in the same conditions in order to analyze the lighting environment. So, to summarize the input of our method, the following data are needed:

- A 3D model of the face to relight.
- A calibrated photograph (*i.e.* with the corresponding projection matrix) of the face taken in a dark room with a flash.
- A light probe (a spherical mirror) image taken in the same conditions.

In the following sections, we describe how the detail texture is built and how a parametric model of the skin is retrieved. Then, we expose in more details the implementation of the rendering engine.

Summary: Our rendering engine uses a 3D mesh of the face as basis. It is first completely rendered using a parametric skin model precomputed from a real photograph of the skin. This is the *shaded face*; it contains all the lighting cues (shading, shadows and highlights). It is then textured to add the eyes, mouth, etc and the skin roughness. This texture is named the *detail texture*; it contains the lighting-independent features.

4.4 Detail texture

As briefly explained in the previous section, a texture map is used to add all the details that are independent of the underlying skin model: the eyes, the mouth, etc. We name it the *detail texture*. Since the skin model renders all the lighting-dependent effects, this texture is lighting independent (*i.e.* without shadow, highlight, etc.). Conceptually, to create the texture, we "subtract the shading from the input photograph".

To model the shading, we need a description of the skin appearance. We first compute a *reflectance map* [94] of the skin as it is seen in the input image (a photograph of the real face).

Definition: A *reflectance map* describes the appearance of the skin for a fixed viewpoint under a fixed lighting environment.

A reflectance map gives the color of the skin as a 2D function of the orientation of the skin surface. Since the visible orientations form an hemisphere, a reflectance map can be represented as a colored hemisphere: Each point on the hemisphere is painted with the color associated to its orientation.

The entire 3D model is then rendered with this reflectance map. This gives us the information needed for the "subtraction".

4.4.1 Skin reflectance map

We build a specific model of the skin as it is seen in the input photograph. Our goal is here to capture as precisely as possible the lighting cues of the skin. We only aim at removing the shading from the photograph, not at relighting.

Therefore, we use an approach similar to the methods presented in Section 4.2.2 on page 94, we build a skin model by densely sampling the appearance of the skin from a real photograph. But we do that only for the input viewpoint and lighting. So, we avoid the previously discussed caveat of a large amount of data. And since this step is precomputed, the rendering time is not an important issue. The advantage of this approach is to provide a skin model which is more accurate than the parametric one that we use for the final rendering. So we can afford a longer and less general computation as long as it provides reliable data for the given viewpoint and lighting.

Another view on* [Reflectance map]*: *It can be seen as a picture of a sphere composed of this material seen under a given viewpoint and a given illumination.*

A face presents almost all the 3D orientations facing toward the camera. On the sphere of the directions (*a.k.a.* the Gaussian sphere), the normals cover the whole front facing hemisphere. Hence, under the approximation that skin follows the same reflectance map on the whole face, the skin reflectance is sampled on the hemisphere with only one image. To know where skin lies in the input photograph (as opposed to background, hair, eyes, etc), we can either use a segmentation algorithm or ask the user to paint the skin region. Figure 4.10 shows a sample segmentation made by the user, which only requires a few minutes.

Link with* [BRDF]*: *A reflectance map describes a subset of the information embedded in a BRDF since it deals only with a single viewpoint and a fixed lighting. It also assumes an anisotropic material since only the surface normal is important. It is invariant to a rotation of the material around the normal. It also handles a complete lighting environment instead of a single directional light source. Formally speaking, it integrates of the BRDF over all the lighting environment. And classically, the BRDF does not include the* $\mathbf{n} \cdot \mathbf{l}$ *term that accounts for the spreading of the incoming light flux over the surface. This term is included within the reflectance map.*

input photograph mask skin region

Fig. 4.10: Sample skin segmentation

We need to segment the skin region from the input photograph (left). We rely on the user to provide this information. In practice, a binary mask (middle) is painted using a standard image editing software. This mask does not need to be very accurate because the following processes can handle outliers.

Each skin pixel is associated to its normal on the 3D model. This results in color samples on the Gaussian sphere. Because of the hair, there are fewer samples for the upward directions (and for the downward directions for bearded people). The samples are then grouped into clusters and the color of each cluster is determined through a robust mean that discards outliers: We only consider the samples whose distance to the classical mean is less than the standard deviation. The outliers may result from errors in the 3D model, inaccuracies in the skin segmentation, etc. With the robust mean, all these artifacts are handled without any user intervention.

Using a front facing flash eliminates self-shadowing. This is important because with other lighting condition, coping with shadows would have been tedious (such preprocessing can take hours [199]) and would most probably have introduced more approximations and outliers.

We then extend the cluster values to a dense and continuous reflectance function by interpolation and extrapolation. For a given point P on the sphere, we use a scheme that:

1. For each distance d, averages all the clusters at the same distance d of P to get one color value $v(d)$ per distance.

2. Associates a weight $w(d) \approx e^{-d}$ to $v(d)$ to guarantee that only the closest values to P have a significant weight.

3. Computes the mean of the values to get the value of P.

This process slightly smooths the resulting function *i.e.* it removes high frequencies. This fits the demonstration of Ramamoorthi and Hanrahan [177] who have shown that the reflected radiance of a material is band-filtered by its BRDF. As a consequence, a material that has a low-frequency BRDF (such as skin) cannot have a high frequency reflectance map.

Note that the map (Fig. 4.11-right) contains both the diffuse and specular components because no separation is done after sampling the photograph. Highlights do not suffer from the outlier removal since we sample a single picture with fixed viewpoint and lighting. The shading does not change during the sampling process and is captured as it appears in the photograph, including both the diffuse and specular components.

samples clusters final map

Fig. 4.11: Reflectance map

Left: *The samples measured on the original photograph. From each skin pixel, we compute its normal and a color point is placed on the position with the same normal on the hemisphere.* ***Middle:*** *These samples are then clustered. Each cluster is assigned a color computed as a robust mean of the samples it contains.* ***Right:*** *The clusters are then interpolated and extrapolated to form a complete and smooth reflectance map.*

Summary: We have computed the reflectance map of the skin. This gives the color of the skin as a function of the surface orientation.

This construction relies on a segmentation of the skin in the input photograph. In our current implementation, this segmentation is provided by the user. The computation is made robustly to account for outliers. Therefore we cope with a non-perfect segmentation (*e.g.* scars and spots do not need to be marked).

4.4.2 Ratio image

The whole 3D model is then rendered with the same pose as the photograph and using the acquired skin reflectance map (Fig. 4.11-right) to define the colors of the vertices. We call the resulting image the *shaded face*. As shown in [139, 141], the combination of a detail texture with a base image should be multiplicative. Therefore, the detail texture (Fig 4.12-right) is the *ratio image* between the input photograph (Fig 4.12-left) and the shaded face (Fig 4.12-middle). It can be formally defined by:

$$\textit{ratio image} = \frac{\textit{input photograph}}{\textit{shaded model}} \tag{4.3}$$

Operation (4.3) is done pixel by pixel, RGB-component by RGB-component. To avoid numerical problems, 0 is replaced by a small value in the *shaded model*.

The main caveat at this stage is to lose some information because of the photograph saturation (*i.e.* the scene is too bright and exceed the captor capacity) due to a too intense flash. It would make the skin texture disappear in saturated areas. We therefore use a flash of limited power which we have tested to guarantee that it produces images whose maximum intensity is about 80% of the captor maximum.

Summary: The shaded face is rendered using the reflectance map of the skin previously computed on the whole 3D mesh of the face. Then the detail texture is computed as the ratio of the input photograph over the shaded face.

Figure 4.13 on the next page shows together the main steps to obtain the detail texture.

4.4.3 Limitations and discussion

The detail texture results from a process that makes several approximations. We discuss here their limitations and validity.

Uniform skin First of all, we assume the skin to be an uniform material over the whole face. Clearly, the skin is not exactly the same on the nose and on the chin. Nevertheless, as it is always composed in the same way there is no large variation across the face. The small variations are partly corrected in the final rendering by the detail texture. If one looks carefully, the detail texture (Fig 4.12-right) still exhibits some highlights on the most shiny regions (*e.g.* on the nose). This comes from this approximation because these regions are shinier than the "average" reflectance. However, these remaining highlights have a limited intensity corresponding to the difference with the average.

Skin base We also consider that everything is a detail based on the skin reflectance. This is obviously wrong for the eyes, the mouth, etc. The color of these details is so different from the skin color that it induces extreme values in the texture (*i.e.* RGB ratios are far from 1) and both colors

input photograph shaded face detail texture

Fig. 4.12: Detail texture

The shaded face rendered using the reflectance map computed in Section 4.4.1 on page 108. The detail texture is then the ratio image between the input photograph and the shaded face.

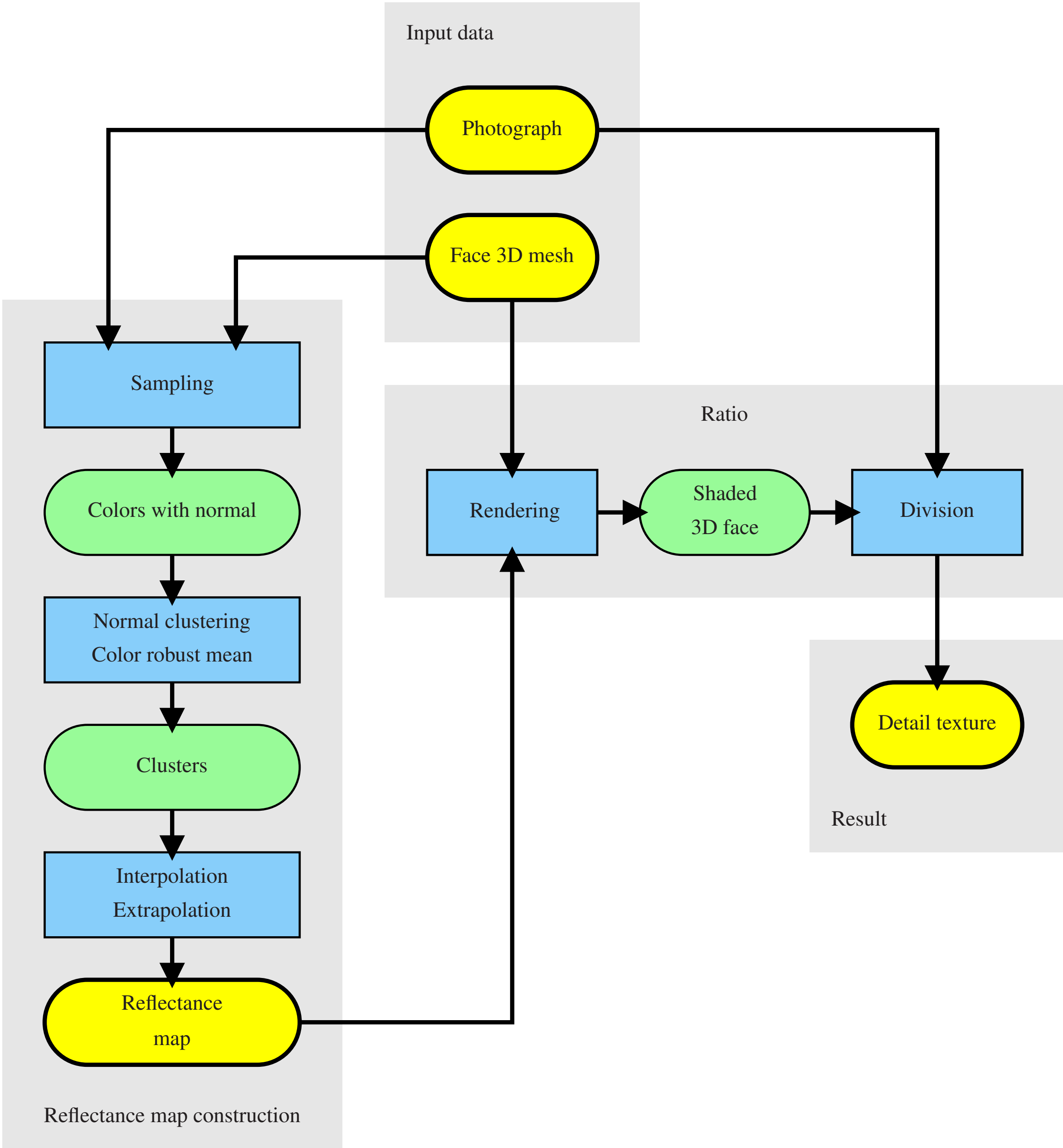


Fig. 4.13: Overview of the creation of the detail texture

First a reflectance of the skin is build from the input 3D mesh and photograph. Then it used to render the shaded 3D face. Finally, the photograph is divided pixel-by-pixel by the shaded face.

are almost decorrelated. These details contain many high frequency features (iris in the eyes, small wrinkles on the lips, etc.), which hide most of the shading effects. Therefore, although they are not skin details, this approximation does not lead to visible artifacts. The only caveat may be the eye and lip highlights: They are not removed by the ratio image because they are not rendered in the shaded face. They are then rendered by the detail texture independently of the light and the viewpoint. This can impede the overall consistency especially in close-up. This is overcome by removing them from the input photograph by inpainting either automatically [16, 209] or manually.

Lighting-independent roughness Since we cannot afford to handle a 3D model precisely up to the skin roughness, the skin textured aspect is rendered by the detail texture. This implies that it is lighting independent whereas it results from the interaction of the light with the micro-relief of the skin surface. This interaction is well described by subsurface scattering [105] but is hardly rendered in real time [159]. Fortunately, contrary to the eye highlights that give strong cues about the lighting environment, these low amplitude variations do not carry much information. Therefore, even if we do not render the aspect changes of the skin roughness due to the light changes, it does not impact strongly the coherence of the produced image.

4.5 Parameters of the skin model

In the previous section, we have built a texture which embeds all the lighting-independent parts of the face. We now explain how to manage the lighting-dependent part.

This step must be efficient and lightweight since it is used to render the final images for which we target real time. From the discussion in Section 4.2.3 on page 96, a suitable representation to fulfill this goal is a parametric reflectance model. It must also be expressive enough to match the main characteristics of the skin as seen in the input photograph. As explained in Section 4.2.4 on page 99, this requirement excludes the too simplistic Lambertian model that ignores the highlights.

We have chosen the classical Phong model [171] in the RGB space because it is implemented in all 3D cards and therefore achieves the best performance. In the following paragraphs we focus on this model but a similar study could be done on any other parametric model (cf. Sec. 4.2.3 on page 96).

From formula (4.2) on page 96, Phong's model is controlled by 10 parameters for the material: the RGB values of ambient, diffuse and specular colors and the shininess exponent ^{s} and 9 parameters for the light: the RGB values of ambient, diffuse and specular colors.

Our strategy is to match as closely as possible the input photograph with the model. We first explain how to recover the lighting parameters and then the skin parameters.

4.5.1 Lighting parameters

The input face photograph is lit by a flash that we approximate by a point light source. Its characteristics are determined with a photograph of a spherical mirror (*a.k.a.* light probe) taken under the same condition as the face.

The first information to retrieve is the light position. A solution is to do physical measures with a classical ruler. We prefer an evaluation from the probe picture that is likely to be more coherent since we exploit the same information source. In practice, we have done both and they give similar values.

The 9 Phong parameters of the light are also needed. But in our setup, we have only two light sources: the flash and the ambient light (*i.e.* the indirect light coming from all the surrounding objects). Hence, among the 9 parameters, there are only 6 meaningful degrees of freedom: the RGB values for both light sources. Hence according to Phong's model, the flash corresponds to the specular and diffuse sources (direct lighting). Consequently, the RGB values of the diffuse and specular are equal to the flash color. In addition, the 3 parameters for the ambient lighting have to be estimated.

Flash position

The light position is computed thanks to the highlight position on the probe. In the (θ, ϕ) spherical coordinate system, we have the classical relation (see the vector **l** and **r** in Figure 4.4 on page 96), with $_l$ for the light and $_h$ for the highlight:

$$(\theta_l, \phi_l) = (2\theta_h, 2\phi_h)$$

(θ_l, ϕ_l) are evaluated by selecting the pixels corresponding to the direct reflection of the flash on the light probe (in red on Figure 4.14). Each of this pixel corresponds to a point on the probe sphere. We average all these positions to estimate (θ_l, ϕ_l). In practice, the angular deviation of the flash light compared to the view direction is about 2° above *i.e.* the highlight is located on the sphere at $\theta_h \approx 0°$ and $\phi_h \approx 1°$.

Flash color

The light color cannot be directly reached in the highlight because it is always saturated because of the direct reflection in the mirror. But as the mirror BRDF is never a ideal peak, there is always a halo around the highlight (cf. Figure 4.14). This halo does not saturate the camera sensors and can be used to measure the light hue. Formally speaking, we measure the proportion between the red, green and blue channels *i.e.* the measured values are valid up to a scale factor. This method gives no information about the color intensity since the measured color has been attenuated by the mirror BRDF. For simplification, we assume this attenuation to be equal for the three RGB channels. For more accuracy, one can calibrate the light probe by comparing a white reference with its reflection on the mirror.

input photograph analysis

Fig. 4.14: Light probe analysis. ***Red:*** *Direct reflection (saturated pixel i.e. intensity over 0.85). Gives the flash position.* ***Yellow:*** *Halo (middle intensity i.e. intensity between 0.5 and 0.85). Gives the flash hue.* ***Blue:*** *Background reflection giving the ambient light (intensity under 0.5). The given threshold are those used in our implementation. Their exact values have a limited influence over the extracted parameters.*

Note that even *high dynamic range* imaging [52] could hardly capture this information about the intensity of a flash light because the flash is both short and very intense. Specialized instruments such as the filters described by Goesele *et al.* [73] are needed. We do not apply that kind of techniques because it is too complex relatively to our goals. Nonetheless, if one can afford such a system, it could be easily inserted in our method. Our approach is to arbitrarily set the intensity to an approximate value. It means that the light intensity is now expressed in relation to the input photograph. In practice, with a good setup which makes the photographs bright but not saturated, 1 is a satisfying value. However, to work with different flash lights, one must take photographs of a white reference (*e.g.* a white matte ball) to calibrate the relative intensities of the flashes.

Ambient color

The ambient lighting is measured by integrating the light samples in a ring surrounding the highlight halo (Fig. 4.14). The whole spherical probe is not used to guarantee that we have no outliers (mainly from the support of the probe). Since there is no saturation, both hue and intensity are recovered. However to be fully consistent with the flash light, their relative intensities have to be adjusted. Unfortunately, the ambient contribution is too low to be used as a precise reference to set the flash intensity. The ambient intensity is therefore optimized relatively to the flash. We use a gradient descent performed once the skin parameters are known. The resulting variation is almost unnoticeable but makes the set of parameters consistent.

Summary: Using a photograph of a spherical mirror, we can retrieve the flash position and color, and the color of the ambient light. From these data, we straightforwardly derive the lighting part of the Phong model.

Notice that the colors are dependent on the used flash. If several flashes are used, their relative intensities have to be calibrated with a reference object (a white ball for instance).

4.5.2 Skin parameters

As suggested in [192] and also [50], we consider that the specular reflection introduces no spectral distortion in the light; it is directly reflected by the upper oily layer of the skin. This means that the specular color is (i_s, i_s, i_s) *i.e.* the reflection does not change the hue of the light, it may only attenuate its intensity. Only i_s remains unknown. Diffuse and ambient colors are due to a deeper interaction between the light and the skin. Because these colors result from the same cause, they have the same value $(r_{\text{skin}}, g_{\text{skin}}, b_{\text{skin}})$. With the shininess exponent, s, there are 5 unknowns.

Now, our task is to match the input photograph as well as possible. To do so, we rely on an optimization process. Unfortunately, there is a strong ambiguity due to the specular part of the model. If the shininess exponent is low, the specular component is flat and thus almost equivalent to the ambient component. High values of the exponent make the specular component concentrated only on a tiny area and almost equivalent to no specular reflection. Optimizing the exponent at the same time as the other parameters makes the whole process non-linear and unstable due to numerous local minima. s is therefore determined separately before the other parameters.

Specular exponent

The idea is to formalize the intuitive remark that a high exponent corresponds to a small highlight and a low exponent to a wide one. We propose a method that measures the "size" of the highlight and relates it to the Phong exponent s. This computation cannot rely on a full diffuse-specular separation because it occurs before the determination of the corresponding parameters. Therefore it implies an indirect estimator.

As explained in Section 4.5.2, each color in the reflectance map is a mix between the light and skin colors. Since the flash is bright and almost white and the skin is darker and colored, color saturation is a good indicator of the mix proportions. To evaluate the highlight "size", we locate the regions with the fastest saturation changes. These fast-changing regions can be seen as the "highlight boundaries". Note that these boundaries are robust since they do not depend on the actual values of the flash and skin colors and do not require a full separation between the diffuse and specular parts. Our strategy is to relate the angular position of a boundary with the shininess exponent.

Definition* [Color saturation]*: *It intuitively measures how "colorful" is a color. It has to be clearly differentiated from the* captor saturation *that we have mentioned before (this one indicates that the light hitting a CCD captor exceeds its measure range). We use the definition of the color saturation used in the HSV color space:*

$$\frac{\max(r,g,b) - \min(r,g,b)}{\max(r,g,b)}$$

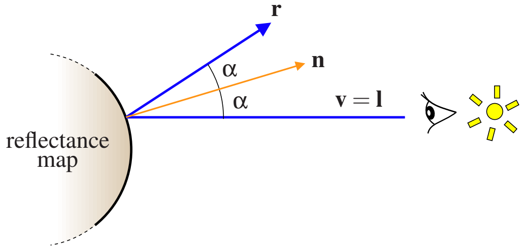


Fig. 4.15: Configuration for highlight measurement.

The highlight position on the reflectance map is known thanks to light position previously computed. Since the sampling resolution is finer near the equator region (Sec. 4.4.1 on page 108), the angular deviation of the fastest variations are measured toward the left and right directions. These two values are averaged to obtain a robust estimation of the angle corresponding to the maximum variation of the saturation. We denote this angle $\alpha_{\text{sat}}^{\max}$.

We then study the Phong model. We compute the specular intensity $I_{\text{flash}}(\alpha)$ and diffuse intensity $I_{\text{skin}}(\alpha)$ relative to the angle α (Fig. 4.15), as well as their ratio:

$$\rho(\alpha) = \frac{I_{\text{flash}}(\alpha)}{I_{\text{skin}}(\alpha)}$$

We denote $\alpha_\rho^{\max}$ the angle where the ratio has its most important change. We conclude with the argument that the largest saturation change is linked to the largest change in the ratio between the specular and diffuse components. Hence we have:

$$\alpha_{\text{sat}}^{\max} = \alpha_\rho^{\max} \tag{4.4}$$

To derive the expressions, we make some approximations: The view and light directions are constant on the face (the variations are at most 2.5° in our experiments) and equal (the difference is at most 2° in our experiments), and the ambient intensity is neglected compared to the diffuse one in the reflectance model (this is reasonable because the input photograph is made in a dark room). Using the notation of Figure 4.15 on the facing page and $\sim$ standing for "proportional to":

$$I_{\text{skin}}(\alpha) \sim \cos(\alpha) \quad \text{and} \quad I_{\text{flash}}(\alpha) \sim \cos^s(2\alpha)$$

which gives the ratio:

$$\rho(\alpha) \sim \frac{\cos^s(2\alpha)}{\cos(\alpha)} \tag{4.5}$$

And since $\alpha_I^{\max}$ corresponds to the largest variation of $\rho(\alpha)$, it implies:

$$\frac{d^2\rho}{d\alpha^2}(\alpha_I^{\max}) = 0$$

which leads with equations (4.5) and (4.4) to the following relation where $\zeta = \sin(\alpha_{\text{sat}}^{\max})$:

$$s = \frac{4\zeta^4 - 2\zeta^2 - 1 - \sqrt{-16\zeta^6 + 8\zeta^4 + 1}}{8\zeta^2(\zeta-1)(\zeta+1)} \tag{4.6}$$

Relation (4.6) fully determines the exponent s with α_{sat}^{max} measured on the reflectance map. It is plotted in Figure 4.16. As the slope may be steep, precision is crucial. We therefore use a high angular resolution of about 1°.

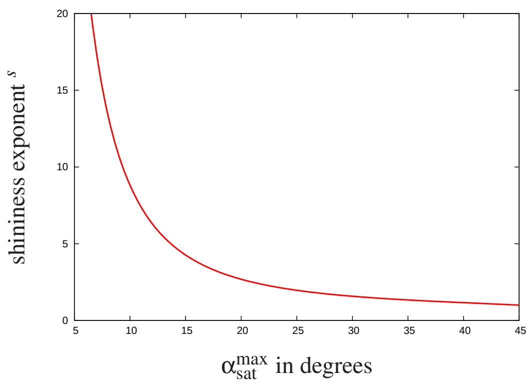
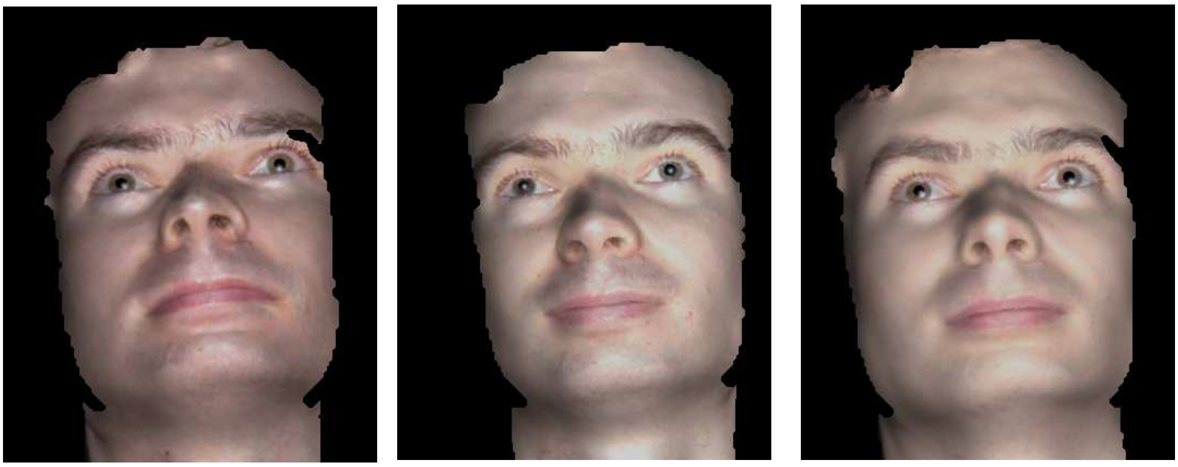


Fig. 4.16: Plot of function (4.6).

To validate the technique, a photo of the same person has been scanned with different skin conditions. As illustrated in Figure 4.17, numerical results agree with the images. The results are best seen on video.



wet ($s = 8.0$) dry ($s = 2.6$) matte ($s = 1.7$)

Fig. 4.17: Comparison of the shininess exponent under various skin conditions

We have retrieved the Phong exponent for two extreme skin conditions and compared them with a reference state (middle). The wet skin (left) is simply obtained with water. The matte skin (right) is made with foundation cream. Our measurements are precise enough to recover the shininess variations between these three faces.

Summary: Among the skin parameters, we compute the shininess apart because it introduces strong numerical instabilities if optimized in the same time as the other parameters.

To evaluate the shininess exponent, we first measure the highlight width on the previously computed reflectance map. Then, we show that the exponent is a direct function of this width. Hence, we can extract the exponent before the other parameters (see Figure 4.17 for practical validation).

Other parameters

For the four remaining parameters (the skin color $(r_{\text{skin}}, g_{\text{skin}}, b_{\text{skin}})$ and the specular intensity i_{s}), we run an optimization process which minimizes the difference between the input photograph and an image of the 3D model rendered with these parameters. This is sometimes referred as *analysis by synthesis* in the literature. We use only the skin region in this process – using the same mask as previously used to compute the reflectance map.

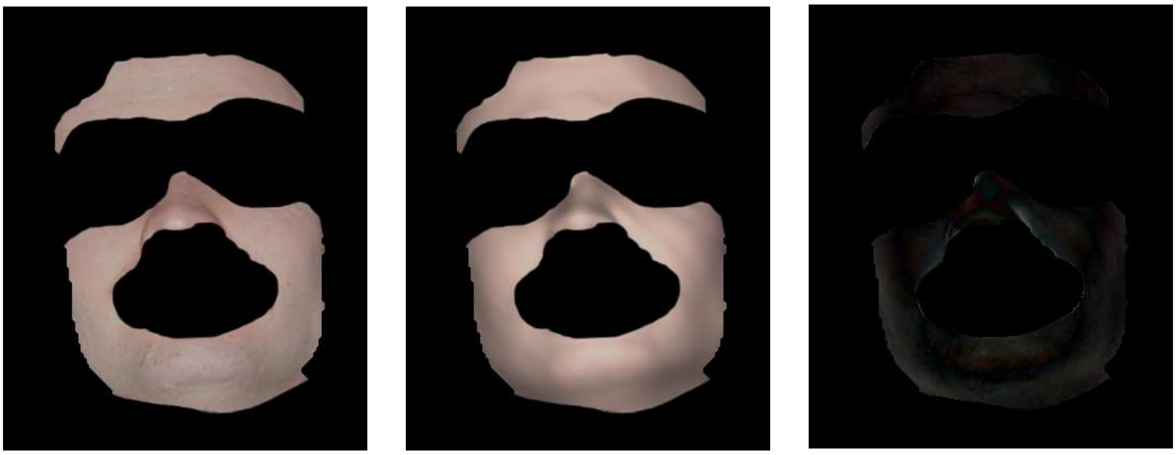
Since we target a visual match, the perceptual Luv space [67] is used to quantify the difference between both images. The comparison is only made on the skin region using the classical L_2 norm:

$$\sqrt{\sum_{\text{skin}} \Big((L_{\text{i}} - L_{\text{r}})^2 + (u_{\text{i}} - u_{\text{r}})^2 + (v_{\text{i}} - v_{\text{r}})^2 \Big)}$$

where $_{\text{i}}$ is for the input photograph and $_{\text{r}}$ is the rendered 3D model.

The Luv space is not linear relatively to the RGB space *i.e.* the Luv-RGB transformation is more complex than a matrix multiplication. We cannot apply a standard linear method such as a least-square optimization. Nevertheless, the problem is not a very hard one since we have overcome the main ambiguity source by separately computing the shininess exponent. Each one of the four parameters has now a clear influence over the aspect of the final image. The last difficulty may come from the fact that a brighter skin color may have, in some situations, similar effects as brighter highlights. Fortunately, this point is not major since the spatial influence of both effects are different.

So, the minimization is done through a varying step gradient descent which performs a coarse-to-fine refinement of the parameters. A standard gradient descent may be used but a coarse-to-fine approach simply makes it faster. Descent techniques often suffer from local minima: We have validated its robustness by running the process 200 times on the same input data with random starting points. The standard deviation on the parameters is less than 2.5%. This shows that the process is almost insensitive to the starting point, a "good" initial guess only speeds up convergence. Figure 4.18 shows a typical result of our system.



input optimized model difference

Fig. 4.18: Result of the parameter optimization

The Phong parameters are obtained with a coarse-to-fine gradient descent. It minimizes the perceptual difference between the skin in the original photograph and the image produced with the model. Note that the printed version may show a larger visual difference. The difference image (right) shows that the front-facing regions are too bright and the silhouettes too dark. It is a consequence of the approximate match provided by the Phong model. However, it is hard to notice without a visual reference.

Summary: The remaining four parameters are optimized with a gradient descent that adjusts them in order to produce an image similar to the input photograph. Since the parameters have separated influences, the process is robust and has almost no dependence on the starting values.

Figure 4.19 on the next page gives an overview of the entire process leading to each parameter needed by the rendering engine.

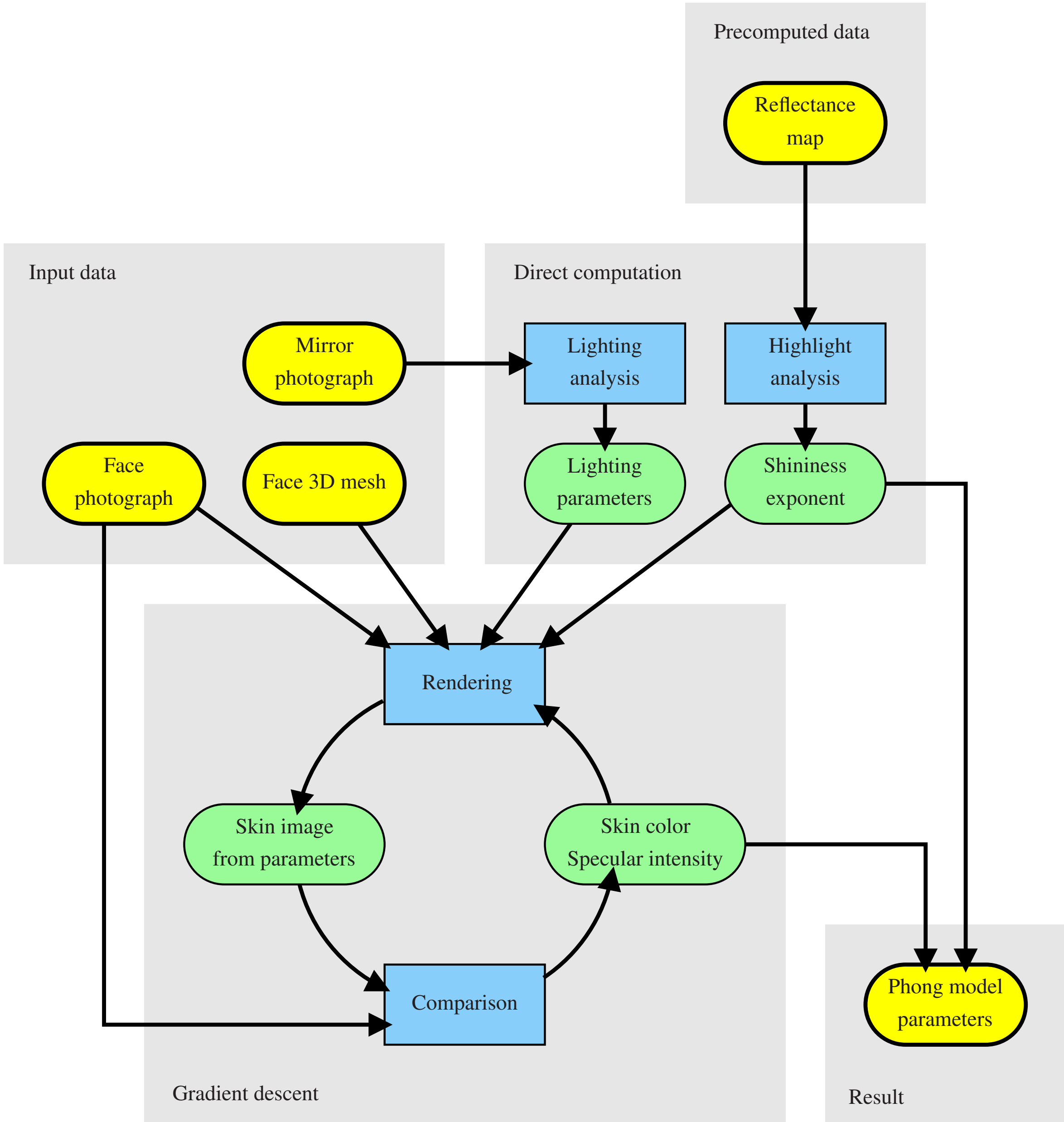


Fig. 4.19: Overview of the Phong parameter retrieval

Some parameters are directly computed from the input data (i.e. without optimization): The lighting parameters and the shininess exponent come respectively from the analysis of the light probe and of the reflectance map. The other parameters (skin color and specular intensity) are adjusted with an optimization process to match the input photograph.

4.6 Implementation of the rendering engine

Our rendering engine is based on OpenGL but it can be easily implemented on any platform that provides basic 3D rendering functions. We restrict the set of functions to the most basic ones to ensure a low hardware dependency. In particular, we have not used the recent programmable pipelines based on *vertex programs* and *pixel shaders* because the available functions still vary a lot according to the graphics card. Nevertheless, since these features are extensions of the basic ones, one can implement our engine using the latest improvements to get better performance on recent cards.

4.6.1 Basic rendering

Our engine performs first a Phong rendering (`GL_LIGHTING`) and then a multiplicative texture mapping (`GL_MODULATE`). Unfortunately, a straightforward implementation is not enough for our purposes. Details brighter than the underlying skin color (the eyes for instance) require a multiplication higher than 1 whereas a basic texture can only store values in $[0,1]$. To work around this constraint, we use two passes:

1. We render the model using a texture with halved values.
2. Then the 3D mesh is rendered without texture nor lighting but with a $(1,1,1)$ color and the blending equation `glBlendFunc(GL_DST_COLOR,GL_ONE)`. This results in doubling the pixel RGB values.

The available range is then $[0,2]$. The method can be extended to $[0,2^n]$ with a texture multiplied by 2^{-n} and n doubling passes. Note that each of these doubling passes divides the color precision by two: the multiplication by 2^{-n} rounds down the values losing n bits of precision. In practice, up to $n=2$ results are visually similar and $n=3$ produces an almost unnoticeable alteration, which makes multiplicative values up to 8 available without visible loss of quality (see Figure 4.20).

Since our rendering engine is lightweight, it supports various improvements to enhance the final visual quality while still achieving fast rendering. We present the ones we have implemented in following sections.

Fig. 4.20: Effect of the doubling passes

Each doubling pass loses one bit of precision of RGB components. It therefore results in a color quantization that increases with the number of doubling passes. However, this technique allows to handle multiplicative values higher than 1. And up to three passes, the resulting artifacts are almost unnoticeable at a reasonable scale (we here zoom on an area in the order of mm^2*). (The quantization may not be visible on the printed version. In that case, please refer to the electronic version.)*

4.6.2 Camera sensitivity

Fig. 4.21: Illustration of the overexposure effect. *Stronger contrast thanks to overexposure (exaggerated for illustration purpose). Note that the details correctly disappear in the highlight.*

When someone is photographed under the sun or with a strong flash, some regions in the picture are poorly exposed. Parts of the face that are in the shadow are often underexposed and those directly in the light are over exposed. Technically, this comes from the limited range of sensitivity of the sensors compared to the intensity range in the observed scene (see [52] for details). The effects are characterized by the disappearance of the details either in the darkness (underexposure) or in the highlights (overexposure).

Underexposure is straightforwardly obtained by lowering the light intensity. Overexposure can be simulated by clamping to 1 (thresholding the highest values to 1); it corresponds to a simple sensor model which is linear in [0,1] and saturates all the values higher than 1. However, although light intensity can be set higher than 1, direct rendering does not perform the correct computation because clamping occurs before the texture mapping whereas we need it afterward. We use the same process as previously: we divide the light intensity by 2^m so it is lower than 1 and makes m additional doubling passes. This ensures that no value is clamped before the texture mapping and thus achieves the correct result since the clamping only occurs during the pass combination. This effects is shown in Figure 4.21.

4.6.3 Eye highlights

without

with

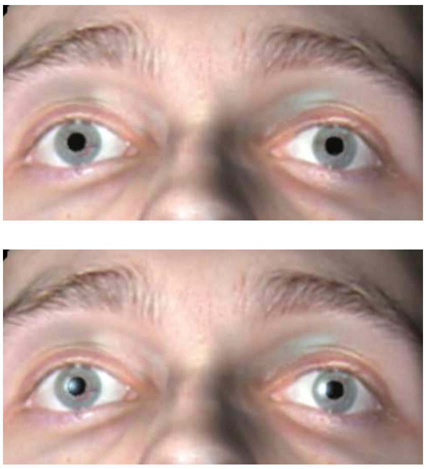


Fig. 4.22: Highlights in the eyes.

Eye highlights are a very strong cue for the lighting environment: Due to their very high specularity, eyes behave almost like mirrors that reflect the surrounding objects. However, because of the iris, the eyes contains a lot of details and, thus, only high intensity features of the environment are actually distinguishable. Therefore we can limit the surrounding environment to only the light sources and still having a convincing effect.

Practically, we place two hemispheres corresponding to both eyes. These spheres are purely specular with a very high shininess exponent (*e.g.* 75). Because of this high exponent, the hemispheres have to be finely subdivided (in practice, we use two hemispheres with 256 vertices each). We perform an additive composition: The light reflected by the cornea is superimposed to the light coming from the deeper part of the eye (iris, vitreous, etc.). This gives a hint of the radius of the spheres, it should be approximately the cornea radius *e.g.* 7.8mm [231].

Figure 4.22 illustrates the impact of the highlights in the eyes. Without it, eyes look dry and frozen whereas with it, they have the expected shiny aspect. The improvement can be better seen in the video.

4.6.4 Cast shadows

Our model already includes shading effects because of the underlying Phong model. However, under unidirectional lighting or with a point light source, cast shadows are also a significant lighting cue. We use the *shadow maps* [229] to add cast shadows to the Phong rendering. Compared to other classical methods such as *shadow volumes*, this method has the advantage that it can handle complex geometry like faces. But it involves three passes per light source: one rendering from the light view point, one for the lit part and one for the shadowed part. This can significantly reduce the frame rate if there are many sources. It should be used carefully (*e.g.* in a level of detail context). Figure 4.23 shows the improvement obtained with the shadows.

without with

Fig. 4.23: Improvement brought by the cast shadows.

Summary: The basic rendering engine is made only from very standard functions such as Phong lighting and texture mapping.

Furthermore, the engine can be simply extended to handle special situation. We here demonstrate some extensions: dynamic setting of the camera exposure, highlights in the eyes and cast shadows. However, one has to realize that each additional effect is time-consuming and may impair the efficiency. These extensions should be therefore thought in level-of-details context *e.g.* eye highlights should be activated only on face close-ups.

4.7 Results

Our scanner provides a mesh at a resolution of 200×200 with a maximum error of 300 micrometers. It also shoots a 400×400 photograph of known parameters while digitizing. This photograph has a poor quality and no flash. So, we also shoot a high quality photograph with a flash. From this one we extract a 512×512 face picture and the correspondence between this new image and the 3D model is made by matching feature points between this photograph and the scanned one with known parameters. The subject is always between 2 and 3 meters from the camera. Since the flash is about 6 centimeters above the optical center, we can estimate to $2°$ the maximum angular distance between the view direction and the light direction and to $4°$ the maximum angular deviation of the view angle on the face. Thus, the assumption made to compute the shininess exponent that these values are null is reasonable. Our flash is a rectangle of $1\text{cm} \times 2.5\text{cm}$, and has in our acquisition setup a maximum angular dispersion of $0.75°$. Thus it is accurately approximated by a point light source.

NVIDIA GeForce 3 NVIDIA GeForce FX

Fig. 4.24: Face renderings from the latest graphics cards

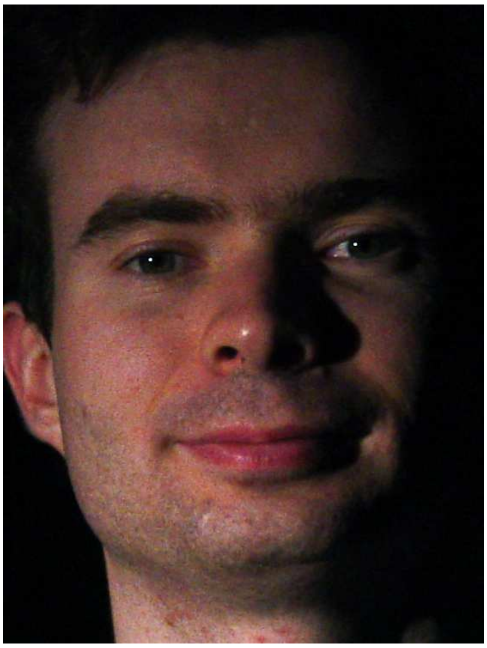
These images are extracted from the Zoltar *demo for NVIDIA GeForce 3 (left) and from the* dawn *demo for NVIDIA GeForce FX.* [By courtesy of NVIDIA]

For comparison purposes, Figure 4.24 shows sample images rendered with the latest available hardware. These images suffer either from their unrealistic texture (Fig. 4.24-left) or from a too smooth aspect (Fig. 4.24-right). The method presented in this paper can easily improve both techniques without losing their intrinsic qualities.

To validate our process, we have taken photographs with various light positions and compared them with relighting results (Fig. 4.25 on the facing page). As the reference photographs have not been taken under the same conditions as the input photograph, the pose and the facial expression are different. There are other differences to discuss:

Facial expression All our captured faces look tensed. We have asked the subjects to tilt their head backward to capture the geometry of their chin. Setting the 3D scanner on a lower position would let the subject use a more comfortable pose and should address this point.


(a) real (b) synthetic




(c) real (d) synthetic

Fig. 4.25: Comparison between real photographs and relighted images.

Shadows The contrast is lower in the relighted picture. This mainly comes from the 3D model, which does not capture all the geometric details (*e.g.* the lips and wrinkles are not deep enough). This makes some shadows too smooth. On the other hand, some shadow edges are too "hard" (near the lips in Figure 4.25-b) because subsurface scattering is not rendered.

Highlights As previously discussed, the nose highlight is not strong enough and Marschner *et al.* [148] mention that parametric models do not catch the right skin properties for grazing angles, this can be seen in Figure 4.25-b on the left cheek.

Fig. 4.26: Relighting in a complex environment.

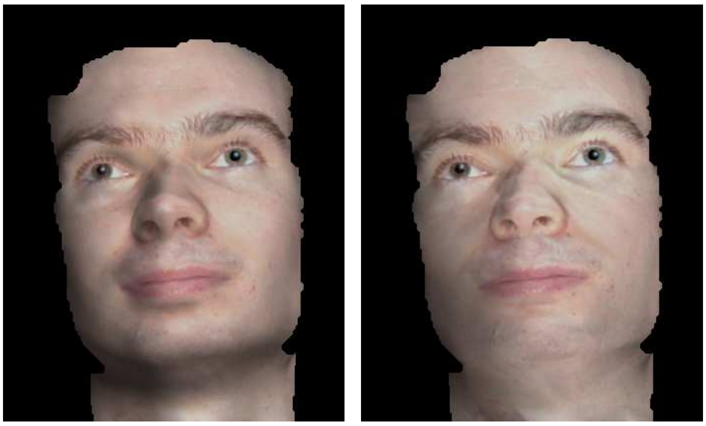
Details Some tiny details are also missing (wrinkles around the mouth in Fig 4.25-d) or appear blurry (eyebrows in Fig 4.25-b) because of the texture resolution which is lower than the photograph resolution.

Nevertheless, the overall appearance matches: shadow boundaries and shading are accurate. As the images are rendered in real time, lighting variations therefore make convincing shading variations including highlights and cast shadows (see the video). Moreover, without any additional feature, the method deals with a complex lighting environment (Fig 4.26) and a difficult case with a short beard through which skin appears (Fig 4.27).

Fig. 4.27: Relighting a bearded face.

We have measured the frame rate for two models: *scanned* coming directly from the scanner with 12,250 vertices and 24,060 triangles (Fig 4.28-a) and *simplified* which has been simplified to 787 vertices and 1,143 triangles (Fig 4.28-b). We have tested a basic lighting environment with one light source and a complex environment with 8 sources and over-exposure (two more passes). The graphics cards are from NVIDIA: a TNT 2 (low 3D capacity), a GeForce 4 MX 440 (middle 3D capacity) and a GeForce 4 Ti 4400 (high 3D capacity). The results are summarized in Table 4.1. Note that the engine is able to render the simplified mesh on all three cards at a rate high enough to be able to render other objects and still maintain real time. This point is remarkable since the TNT2 (1999) has limited capabilities compared to the newer cards. Moreover our acquisition process produces data which resists extreme degradation of the supporting mesh. Figure 4.28 shows that even with 20 times fewer triangles, the face looks almost the same. There is a limited loss of contrast on the coarser mesh because creases have been smoothed but the visual quality is comparable.

We have tried to extend our method to 3D models obtained from Computer Vision techniques. The main advantage of these techniques is that they use less equipment (only a camera) and a strong consistency between the geometry and the input photographs. Readers interested by more details about



fine mesh coarse mesh

Fig. 4.28: Comparison between a fine mesh and coarse one

Computer Vision are referred to books like [61, 63, 86]. The latest methods are very promising, including the work of Lhuillier and Quan [137, 138], Zhang *et al.* [239], and Shan *et al.* [193]. We tested our method with a head model produced with the method of Lhuillier and Quan [137, 138]. Even though the resulting mesh is less precise than that from a scanner, we achieve satisfying results on this model. It would be interesting as a future to study a combination with our surface-reconstruction methods proposed in Chapters 2 and 3. A sample image is shown in Figure 4.29. The main artifacts appear near the hair boundary. The sharp change of orientation of the surface is not correctly caught by the model and results in erroneous normals. This produces spurious lighting effects that look more or less like caustics but move inconsistently when animated. However, the overall aspect looks correct and let us think that better results can be achieved.

Fig. 4.29: First results from a Computer Vision model.

	SIMPLIFIED MESH		SCANNED MESH	
	1 light	8 lights	1 light	8 lights
TNT2	95	28	3	1
MX 440	90	90	78	58
Ti 4400	110	110	110	60

Table 4.1: Performance of the rendering engine

The frame rate (in Hz) is measured with the simple *mesh (787 vertices and 1143 triangles) and the scanned mesh (12,250 vertices and 24,060 triangles), and with 1 and 8 light sources. The 8-light environment also includes over-exposure that requires two more rendering passes. We have noticed some surprising results for the very high frame rates (e.g. the TNT2 card is faster than the MX440). We suppose that it may come from some "hidden" internal structures of the hardware that we do not take into account. However, frame rates over 60Hz are only a performance proof that is not useful since it may exceed the screen frame rate. This only shows that we do not use the full available power.*

Summary: Our approach yields convincing results. As expected, some features are missing such as the skin reflections at grazing angles. However, the overall aspect is preserved and the produced animations convey compelling lighting cues.

Furthermore, the rendering is performed in real time even on "old" graphics cards. It is suitable for interactive applications since it does not consume the whole rendering power available and let some resources available for other tasks.

4.8 Conclusions and future work

Our results suggest several conclusions. Nowadays hardware rendering uses programmable chips that push farther and farther the limits of real-time rendering. Obviously, these cards introduce an extended expressiveness associated with an enhanced rendering power. This allows us to simulate complex phenomena faster and faster. For instance, subsurface scattering is almost already available in real time with dedicated hardware functions.

However, this can sometimes hide the reflection that discerns which features are useful to simulate exactly and which ones can be approximated or even omitted. For instance, we are convinced that subsurface scattering coupled with an extremely fine mesh such as the one shown in Figure 4.6 on page 100 should be reserved to the only tasks that require very high quality images without imposing constraints on the rendering time. We believe that one of the interesting points of our work, beyond the technical aspect, is to show that a classical texture can convey a visually compelling skin aspect. In the same time, it shows what the missing features are. And it would be interesting to study in future work whether some extensions could overcome these shortcomings. The recent techniques for soft shadows [88] in hardware should be useful to approximate correct shadow boundaries. Simulating the grazing-angle reflection would also be a challenging issue.

Another approach for future work would be to concentrate the hardware power on the spots where it is actually needed whereas the other regions are rendered with a lightweight technique like ours. One could imagine using a time-consuming shading system on limited areas such as the silhouettes and the nose that are typically lacking highlights. This raises two challenging issues: locating the regions to focus on and mixing two rendering systems.

It could be also possible to examine how this rendering technique can be adapted to animated faces. We believe that there is no major obstacle to animate the base 3D mesh as long as the texture coordinates are not altered by the animation.

We also plan to look carefully at hair. On the one hand, hair is quite different from the skin but on the other, it also has a complex lighting behavior that naturally introduces numerous questions. We believe that there is a way to find an efficient trade-off for hair rendering between exact rendering and simplistic approximation. Moreover, this would nicely improve our face relighting engine which lacks a specialized model for the hair and would extend our face rendering to the entire head. This is all the more interesting that some Computer Vision algorithms are now able to acquire the whole head geometry.

We provide in the next chapter a powerful tool to analyze and capture the hair geometry. This paves toward such an efficient manipulation of hair.

5
Capture of hair geometry

The work presented in this chapter has been partly done with the participation of Hector Briceño who is post-doctoral researcher in the Artis team. He has contributed to writing the following text and to acquiring the input data. He has authored the movie that presents our method. His regular feedback was of valuable help for designing the following method.

5.1 Introduction

Hair is an important part of the general aspect of a human character. It has a great impact on the look of the head and therefore requires special attention when creating digital characters. Today, hair manipulation relies strongly on user input. The process can be split into three parts: modeling, animating, and rendering. Among these three steps, modeling is the time-consuming step since great progress has been made for animation [9, 15, 38, 81, 172] and rendering [110, 114, 147]. Unfortunately, hair modeling still requires a user-assisted process. Dedicated methods exist to drive the hair creation process and help the user: [47, 80, 115]. These clearly allow a fine control over the geometry but it becomes tedious if one wants to reproduce complex features like curls and waves of a real character.

We focus on this modeling step and propose an alternative solution to the creation of the hair geometry: We use sequences of images of a real hair to produce a set of hair strands which capture most of the features of the original hair. This hair-from-images approach is complementary to the hair-from-user techniques. The produced set of strands can be used directly “as is” or may be the starting point of a user-driven approach.

The advantage of the image-based approach is the reproduction of the hair of an existing person. The main application is virtual actors: more and more movies use digital clones of the human actors in their action scenes to shoot dangerous or technically impossible situations. Until now, the hair of these digital actors is either short or straight – often a poor match with the original hair. The presented technique is a first step toward duplicating a real hairstyle with complex features. Furthermore, one can also imagine using this approach in games (to have your own clone inside the game), styling software (to preview color changes of your hair), criminal investigation and law enforcement (to render people with different hairstyles), etc.

The goal of our method is to produce a set of hairs which can be rendered under different lighting conditions while preserving the features of the original hairstyle. To achieve this, we create dense curved lines. These lines do not exactly match each fiber (it seems unfeasible from images with common resolution) but nonetheless reproduce the main features of the input hair (curls, waves, etc). An alternative method could produce a textured surface, but this would be hard to animate and render properly. We strive for a dense set of hair such that no holes are visible. We would, in the future, also like to capture the reflectance parameters and adapt the hair for animation.

More about* [Hair]*:* *The diameter of a hair fiber ranges from to 17 to 181μm and there are from 90,000 to 140,000 fibers on a scalp [35].

Our strategy is to overcome one of the major difficulties with hair: light interaction that results in large specularities, scattering, glints, etc. These properties hinder the use of 3D scanners and foil computer vision techniques. We turn this specific behavior into a source of information: we analyze the hair under various illuminations to extract their geometry. Our approach is to shoot image sequences of a still head with a fixed viewpoint under a moving light source. The correspondence between images is straightforward but gives no stereoscopic localization, so depth and shape information come from appearance variations. In that respect, it is related to *shape-from-shading* [31] and *shape-from-specularities* [233]. Unfortunately, these techniques only recover a surface and may have trouble with complex materials. Even example-based methods [90] that overcome this last point, recover only a surface which is poor approximation of the hair geometry.

5.2 Previous work

There are many Computer Graphics methods for rendering hair from synthetic data, but few for capturing it as pointed by Rushmeier [181]. Traditional 3D capture systems, such as laser scanners, have trouble with hair due to its complex reflection properties, and yield erroneous results.

5.2.1 Generic approach

Matusik *et al.* [154, 155] propose using image-based rendering for various objects with complex lighting behaviors (see Figure 5.1). Even if not demonstrated on hair in particular, this method undoubtedly yields quality results. However, it may suffice for rendering, but without geometrical information, it would be difficult to animate or edit captured objects.

Fig. 5.1: Sample result from Matusik *et al.* [154]. [By courtesy of Wojciech Matusik]

5.2.2 User-driven approach

For better editing and rendering, we need a geometrical model of hair. There are many packages for generating hair models such as the characteristic strands from which a whole hairstyle is build by Daldegan *et al.* [47], the flow-editing system of Hadap and Magnenat-Thalmann [80] (Figure 5.3) or hierarchical approach of Kim and Neumann [115] (Figure 5.2). Interested readers can find more methods in the state-of-the-art review [144] and course notes [143] of Magnenat-Thalmann *et al.*, and Kim's web repository [113]. All these methods allow a fine control over the created hairstyle while leveraging the editing power of the user by making possible to control several hair strands at once.

Unfortunately, since they start from scratch, these methods are time-consuming when it comes to generate complex hairstyles. We even believe that it would be hard to reproduce a given hairstyle with one of this system because it would require to edit every single hair of the head. Nonetheless, these approaches will always be needed since they give a total control to the user. A hair-from-images approach as we propose in this chapter is complementary to these editing techniques. Such an approach could be used as a bootstrapping step that provides a first hair volume which the user can work from.

Fig. 5.3: Sample hairstyle created with a flow-editing system [80]. [By courtesy of Nadia Magnenat-Thalmann, MIRALab - University of Geneva]

5.2.3 Image-based approach

The work of Kong *et al.* [124, 125] considers modeling hair from pictures. It works by building a 3D hair volume from various viewpoints of the subject's hair. They point out the difficulty of extracting a complete hair strand from an input image. They propose a procedural method to fill this volume with strands. For each hairstyle, a new generating function has

Fig. 5.2: Various hairstyles from a hierarchical editing system [115]

These hairstyles are created using the system of Kim and Neumann [115]. [By courtesy of Tae-Yong Kim]

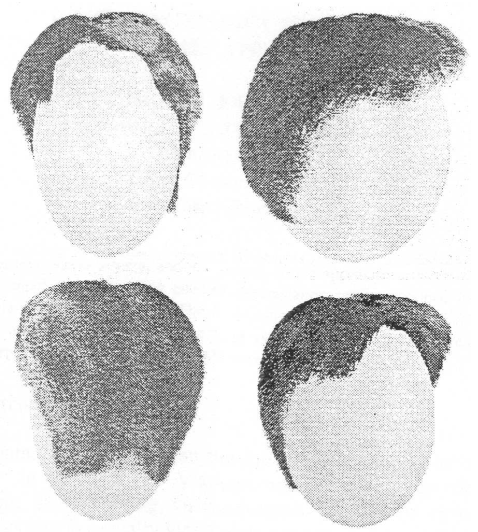


Fig. 5.4: Sample hairstyle created with a procedural and image-based system [125]. [By courtesy of Hiroki Takahashi]

to be designed. The original paper describes one for straight hair and each new style (curly, wavy, tangled, etc) requires a new filling procedure. Moreover, there are no guarantees regarding any local match; only the global style is preserved. Thus, this method seems unlikely to handle complex hairstyles well nor to reproduce an existing style. Sample result is shown in Figure 5.4.

Recent work in this area from Grabli *et al.* [78] (see Figure 5.5 on the facing page), is the most relevant to ours. This system works by studying the subject's hair under various lighting conditions. By fixing the viewpoint, they can work with almost perfectly registered images. Their approach uses a single filter to determine the orientation of hair strands, therefore, many images and sequences may be required to achieve a dense sampling. They only consider one general viewpoint and do not reconstruct all of the hair. Our technique builds upon their approach and addresses these short-comings.

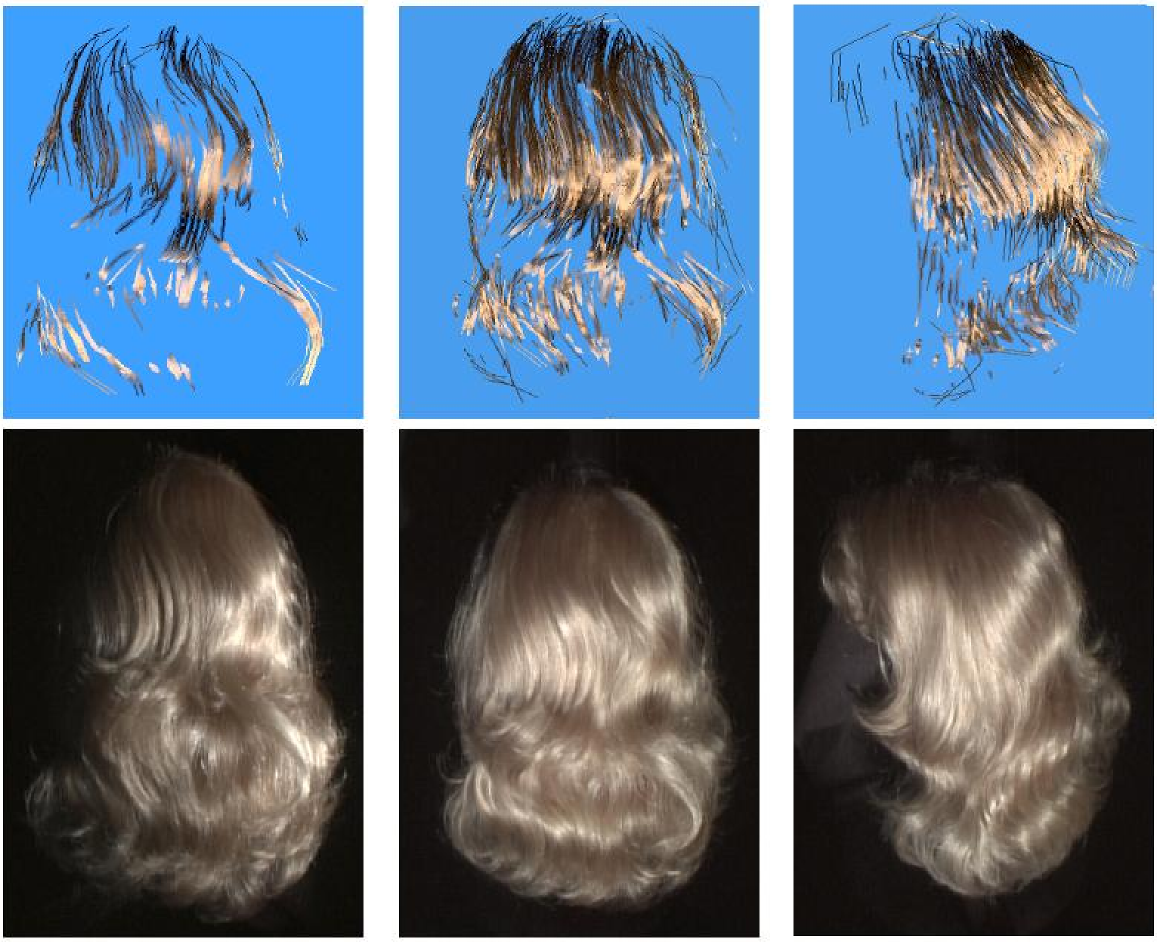


Fig. 5.5: Results from Grabli *et al.* [78]

These models (top row) are extracted from an image sequences of wig captured with robotic gantry (the bottom row shows a sample input image). [By courtesy of Stéphane Grabli]

Summary: To our knowledge, there do not exist today any method to reproduce a given real hairstyle. Systems exist to help the user create a whole hairstyle. But they seem limited when it comes to reproduce the numerous details that make a hair model look like the hairstyle of a given real person. On the other side, there are very few approaches using images and none them is able to reproduce a whole head of hair.

Hence, we believe that a complete system working from images would be useful and complementary to user-driven techniques.

5.3 Overview

Before exposing the main ideas of our method, we give a few definitions to clarify useful entities. This nomenclature is specific to the following sections.

5.3.1 Definitions

We call a *fiber* a single and entire hair. A *strand* is a small group of fibers that are tightly grouped together along their whole length. This is the visible entity in images since a fiber width is smaller than a pixel. A *segment* is a small section (≈ 1mm) of a strand, it is well approximated by a small line. These three entities are illustrated on Figure 5.6.

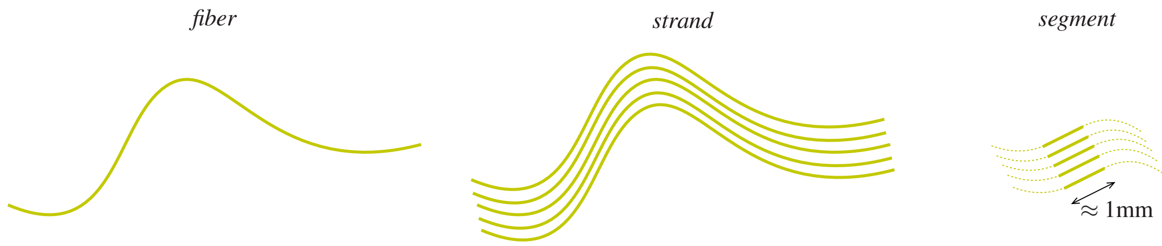


Fig. 5.6: Definitions of *fiber*, *strand* and *segment*

We call *orientation* the un-directed line containing a segment. To characterize one of the two corresponding directed lines, we need to provide a *direction*.

5.3.2 Global approach

Our strategy relies on image sequences with fixed viewpoints and moving light source (Fig. 5.10 on page 139). On the one hand, since the camera does not move, a pixel always represents the same 3D location on the hair surface. On the other hand, this gives no stereoscopic information about this 3D location. Therefore the 3D information comes from the analysis of the image variations throughout the sequence. This analysis considers the segments individually and is split into two parts:

1. A 2D analysis of the image properties recovers the 2D orientation of a segment projected in the image plane and characterizes a plane containing the segment (Fig. 5.7).

2. A 3D analysis of the illumination variations gives a normal to the segment that results in a second plane.

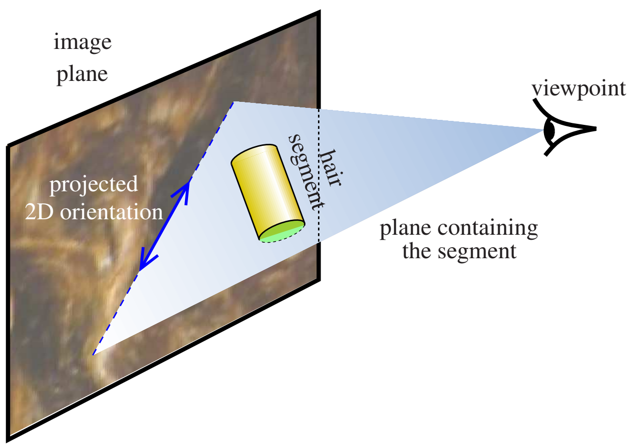


Fig. 5.7: Viewpoint and 2D orientation of a segment in image.

Fig. 5.8: The four viewpoints

We use four viewpoints (left, right, top, back) to capture a whole head of hair. For each viewpoint, we acquire about 200 images (see Figure 5.10 on page 139).

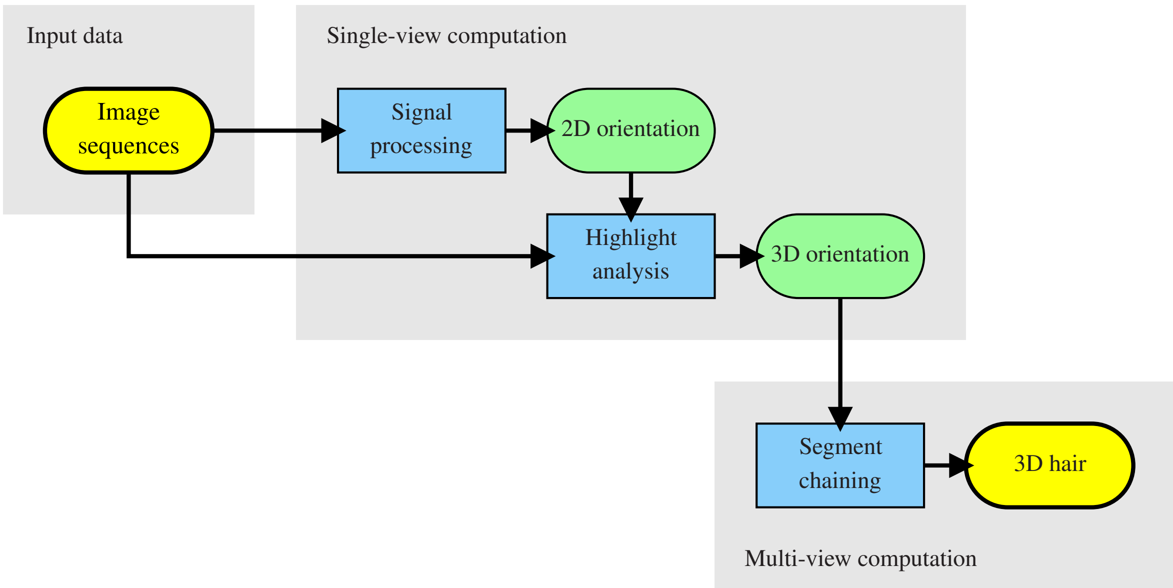


Fig. 5.9: Overview of the hair capture process

The main part of the algorithm works from a single point of view: First the 2D orientation of the segments is computed using signal processing techniques (Sec. 5.4.1 on page 140) and then, it raises into a 3D orientation by analyzing the reflection of the light on the hair (Sec. 5.4.2 on page 148). Only the last step combines several viewpoints to grow the strands one by one by chaining the segments together (Sec. 5.5 on page 151).

Intersecting both planes forms the 3D orientation of the segment. Linking these segments together builds strands.

To form a full geometric models of a person's hair, we propose in this paper a method to merge the information from several image sequences with different viewpoints (Fig. 5.8 on the preceding page). Figure 5.9 summarizes the whole algorithm.

Summary: We propose a method that exploits image sequences from a fixed viewpoint with a moving point light source. There are three main steps in our algorithm: First, we analyze the image content with signal processing techniques (Sec. 5.4.1 on page 140). Second, we recover the 3D orientation of the hair segments by tracking the highlight position (Sec. 5.4.2 on page 148). Third, we chain the segments together to form strands (Sec. 5.5 on page 151). The first two steps work from a single viewpoint at a time. The collected data are merged together during the last step.

5.3.3 Limitations

The method described in this chapter is widely usable. However, there are a number of assumptions and some cases cannot be handled. Since we work from images, hidden hair strands are not captured. For instance, curls that form toward the camera are partially reproduced: Only the visible half is captured. Furthermore, our technique relies on the assumption that hair strands are thin and that their orientation is visible in images. This implies that we cannot handle thick strands (like dreadlocks) or short hairs pointing toward the camera. Lastly, there are inputs where our system would not work well; for example, tangled hair might be unrecoverable because we work from the assumption that there is one orientation per pixel.

Fig. 5.10: Sample input sequence from a single viewpoint

The image sequences are composed of two sweeps: bottom to top and left to right. We show later that one would not be enough to analyze the light reflection and that these two are sufficient. Each sweep has between 50 and 100 images of resolution 1024×768 *(useful area is about* 550×550*) recorded at 7.5Hz.*

5.4 Orientation of the segments

The first step in reconstructing complete hair strands is to retrieve the 3D orientation of each segment. To do so, we analyze the images from a single viewpoint. We proceed in two steps:

1. Signal processing retrieves the 2D orientation of the segments projected in the image plane
2. A highlight analysis raises this 2D data into 3D orientations.

5.4.1 Capturing the 2D orientation

Our technique starts by measuring the 2D orientation of each segment projected in the images. Since a segment contains several fibers with the same orientation, it induces a local orientation in the image. As illustrated by Figure 5.7 on page 136, this information gives a first indication on the actual 3D orientation of a segment since it characterizes a 3D plane that contains the segment. Practically, this steps boils down to a classical signal processing issue: What is the local orientation of an image?

Many approaches exist to give an answer to this question. Ziou and Tabbone [241] offer an overview of edge-detection technique, and the signal-processing literature suggests many approaches such as the oriented filters studied by Freeman and Adelson [71], the structure tensor of Granlund and Knutsson [79], the curvelets of Candes and Donoho [33], the beamlets of Donoho and Huo [57], the bandelets of Le Pennec and Mallat [132], etc. In general, these methods are proven "optimal" under some theoretical hypotheses on the images (*e.g.* edge profile or differentiability of the iso-lines).

Unfortunately, we face a more complex problem. First, fibers are smaller than a pixel and introduce aliasing. Moreover, hair lighting properties (self-shadowing, light scattering, etc.) make it hopeless to predict any strong properties (*e.g.* the size of oriented segments or the shapes of edges). Therefore, we do not rely on such a property and only assume that there is one orientation in each pixel.

Our approach is not to propose a new filter but to choose for each pixel among a set of existing filters the one that gives the most reliable results. Several existing methods [65, 79, 158] evaluate their own reliability. This evaluation is filter-specific and cannot be compared across filters. Baker and Nayar [12] compare filters on reference images regarding some known properties *e.g.* parallel edges. As discussed before, hair images have few such properties.

Definition: Our evaluation is based on *oriented filters* [71]. A filter in this class is defined by its kernel K designed to detect an x-aligned orientation. To test an arbitrary orientation θ in a image I at pixel (x,y), K is rotated by θ (K_θ in short) and convoluted with I. It produces a score:

$$F(\theta) = |K_\theta * I|(x,y)$$

The result of the filter is the orientation with the highest score:

$$\tilde{\theta}_F = \operatorname{argmax}(F(\theta))$$

In practice, there is only one maximum response when a filter is applied to real images. Hence, $\tilde{\theta}_F$ is uniquely defined.

We then observe the response curve of such a filter. We are seeking a peaked curve. If the curve is flat, the result is uncertain because other results can almost fulfill the same criterion. The peakiness of the curve is evaluated with its variance. Let's define:

$$
\begin{aligned}
F(\theta) &= |K_\theta * I| && \text{filter response for orientation } \theta \in]-\frac{\pi}{2}, +\frac{\pi}{2}] \\
\hat{F} &= \frac{F}{\int F} && \text{normalized version of } F \\
d(\theta_1, \theta_2) &= \min(|\theta_1 - \theta_2|, |\theta_1 - \theta_2 \pm \pi|) && \text{angular distance between } \theta_1 \text{ and } \theta_2.
\end{aligned}
$$

The measure is then the classical variance formula:

$$
V(F) = \int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} d^2(\tilde{\theta}_F, \theta)\hat{F}(\theta)d\theta \tag{5.1}
$$

Comparison property: Since the variance only considers the normalized response curve of the oriented filter, it is independent of the image and of the oriented filter. Therefore, different filters for different pixels on different images can be rigorously compared together.

Moreover, this measure has these good characteristics:

- The normalization makes it insensitive to the scale of the filter and of the image, so an intensity scale does not change our evaluation.
- When formula (5.1) is discretized, comparison can be done for any number of θ samples arbitrarily chosen as long as each sample is weighted with the measure function $d\theta$.

Ishikawa [99] also introduces a statistical approach to chose among point-matching criteria. But since his method is based on entropy, it is sensitive to the number and positions of the samples and is invariant to a permutation of the response values, *i.e.*, it does not distinguish whether the significant scores are grouped or randomly spread. Yitzhaky and Peli [235] use a statistical analysis to aggregate edge-detector results into a single edge map. But Forbes and Draper [68] show that this evaluation is image dependent. And, it requires numerous filters to properly work whereas our technique gives meaningful results with as few as two filters.

Practical Uses: For a given 2D location, several filters are tested with several parameters on images with different lighting. $V(\cdot)$ gives an objective rating to select the "best" option. As we will see in Section 5.4.1 on page 145, we can also use these values to compare and compute the similarity between adjacent pixels to enhance our results.

Summary: We use oriented filters to detect the 2D orientation of the hair segments projected in the image plane. We introduce the variance of the response curve of these filters to evaluate their reliability. This objective measure makes it possible to chose the "best option" between several filters, several sets of parameters and several images.

Practically, we compute the orientations for each pixel of the hair region. The value for different pixels can be computed from different filters on different images. Intuitively, the variance selects the "best" filter and the "best" lighting condition to determine the orientation at a given pixel.

Filters

We now detail the oriented filters that we use and their parameters. The first parameter is the scale at which the images are analyzed. Figure 5.11 on the facing page shows that hair appearance significantly varies with the observed wavelength. Frequency selection is done with a band-pass filter (a difference-of-Gaussian filter in practice). Since we track hair strands which are very small, we only select the short wavelengths of the image with a Gaussian band-pass filter centered on $\lambda = 2$ to be as close as possible of the Nyquist sampling rate.

We test 64 orientations regularly spaced; this is a good trade-off between accuracy and computation time. We have chosen some oriented filters among the relevant existing ones (mainly edge and line detectors). Table 5.1 gives the formulæ of the kernels we use. Canny [34] shows that these filters can be decomposed into a *detector profile* that detects the signal variations and a *projection profile* orthogonal to it, that accounts for the neighborhood of the examined point. See Figure 5.14.

Detector Profile: Depending on this profile, a filter detects the orientation of different features. This has an impact on the response curve (Fig. 5.12). Therefore several profiles are used (plot in Figure 5.13). They are scaled so that their *pseudo-wavelength* (*i.e.*, the wavelength that best describes their largest variations) equals the high-pass wavelength (2 pixels in our case). For the Gabor filter, from our experiments, we have found that testing

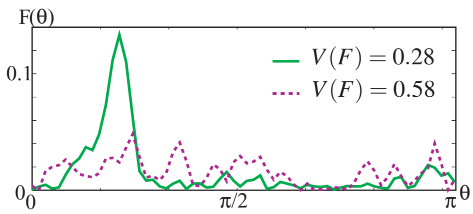


Fig. 5.12: Response curves of two different filters applied to the same pixel.

Gaussian first derivative (Canny) [34]	$-x\,e^{-\frac{1}{2}(x^2+y^2)}$
Gaussian second derivative	$(x^2-1)\,e^{-\frac{1}{2}(x^2+y^2)}$
Canny-Deriche [54]	$x\,e^{-\lvert x\rvert-\frac{1}{2}y^2}$
Shen-Castan [194]	$-\frac{x}{\lvert x\rvert}\,e^{-\lvert x\rvert-\frac{1}{2}y^2}$
Gabor [64]	$\cos(\omega x+\phi)\,e^{-\frac{1}{2}(x^2+y^2)}$

Table 5.1: Oriented filters we use

Formulæ are given to detect a signal parallel to the y axis, rotation terms and scale factors are omitted for clarity.

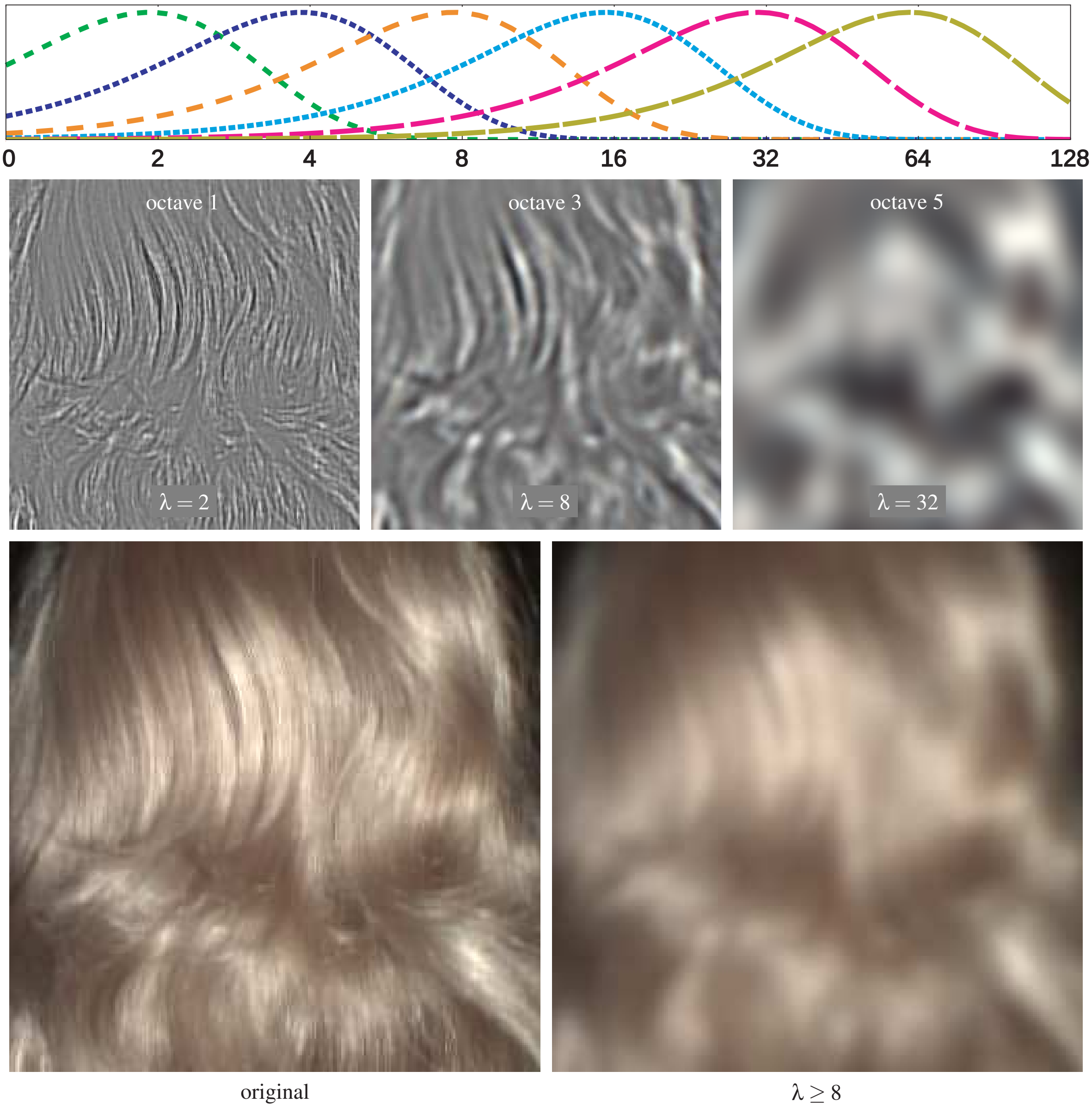


Fig. 5.11: Frequency decomposition of an image

Top: *Frequency profiles of the band-pass filters.* ***Middle row:*** *Three sample frequency bands a.k.a. octaves (intensity is scaled to [0,1]).* ***Bottom row:*** *The original image and a low-frequency version (octaves 1 and 2 are removed). The high-frequency image ($\lambda = 2$) mainly represents the micro-structure of the image due to the hair strands, while the low-frequency one ($\lambda \geq 8$) presents its macro-structure due to illumination. The second octave ($\lambda = 4$) is skipped because it still contains significant frequencies from the first one (observe the top plot).*

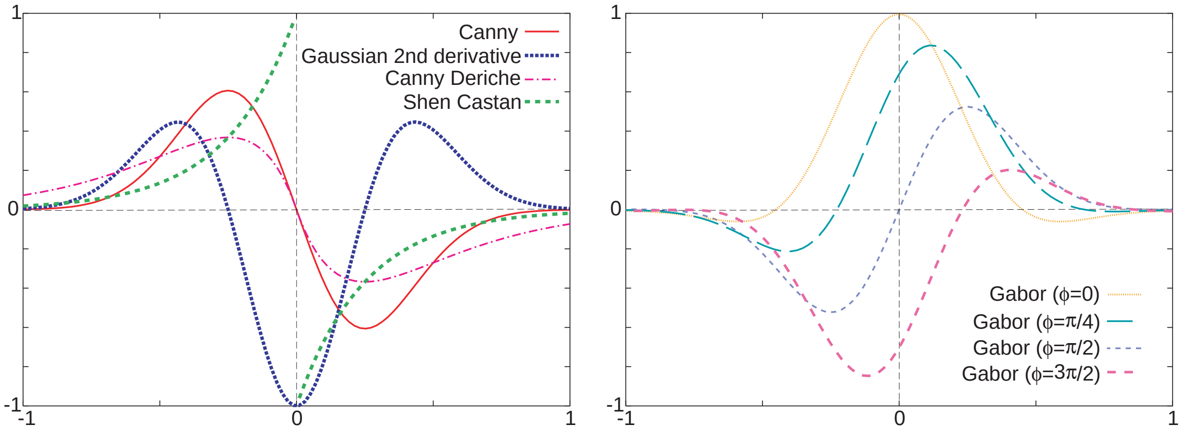


Fig. 5.13: Detector profiles

These are the shapes of the filter kernels along the axis in which we are looking for a signal variation (see the detector profile on Figure 5.14). The profiles are scaled to a unit pseudo-wavelength.

four values for $\phi \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ sufficiently covers the possible phases. We set $\omega = 1$: smaller values do not add significant improvements and larger values detect wider features (several strands side by side) that do not correspond to what we track (the very local orientation of a segment) and degrade the results.

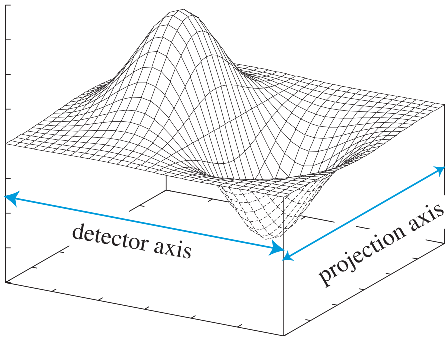


Fig. 5.14: Detector and projection profiles.

Projection Profile: This accounts for the local environment. A Gaussian shape is generally used for this profile. Canny [34] shows that its extension improves efficiency; at the same time, he points out that orientation in real images is a local phenomenon and the extension cannot be too large.

We have analytically computed the response of Canny's filter on a perfect sine signal (Fig. 5.15 on the facing page). It proves that the variance estimator subsumes Canny's remark: The more extended the profile is, the lower is the variance (for an infinitely extended signal) – the better the filter is.

In practice, the best results are reached with Gaussian profiles of standard deviations of 2, 4, and 8 pixels. Thus we test these three values.

Summary: Among the available settings, we first use a fixed frequency band within the input images to deal only with the shortest wavelength. Then we use our variance-based selection scheme to chose among several detector profiles and several stretching factors while testing several images.

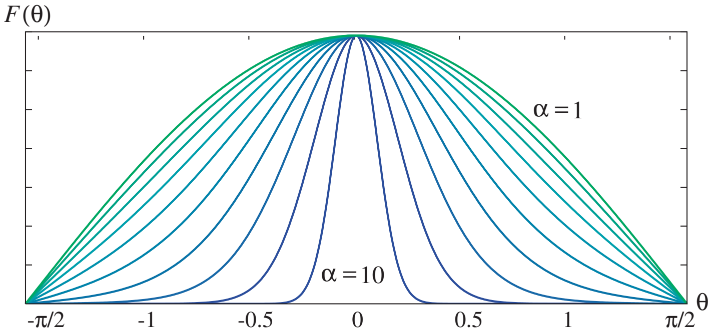


Fig. 5.15: Influence of the projection profile

The plots show the responses $F(\theta)$ of a Canny filter on a sine signal $(x,y) \mapsto \sin(x)$. The projection profile of the filter is stretched α times (from $\alpha = 1$ for no stretching to $\alpha = 10$ for a 10 times larger profile). This graph illustrates that, the more extended is the filter, the more peaked is the response curve and the lower is its variance. Details are in Appendix B.1 on page 175.

Data Enhancement with Bilateral Filtering

Hair geometry has a local coherence: Even if there are some discontinuities, in most cases, neighboring points are similar, thus interact analogously with the light. One consequence is the large extent of the highlights on the hair surface. Information can be extracted from this coherence. If a point has a poorly evaluated orientation, it can be "corrected" by the neighboring points that are more reliable. If these neighboring points have the same appearance in the images they are likely to have the same geometry.

We locally spread the information according to the reliability of the measure (*i.e.* its variance) and the appearance of the pixel in the sequence. We apply a treatment inspired by the bilateral filter introduced by Tomasi and Manduchi [211]. A diffusion process (such as the approach of Tschumperlé and Deriche [215]) could also be used; our choice is motivated by recent results such as the work of Durand and Dorsey [58] and by the theoretical properties demonstrated by Elad [60].

> ***How to* [Compute a mean with 2D orientations]*:*** *To compute the weighted mean of several 2D orientations, the orientation $\theta \in [0,\pi[$ with weight w is mapped to the complex number $c = w\exp(2i\theta)$. The reverse mapping to the orientation is then $\theta = \frac{1}{2}\arg(c)$. Our system does not suffer from degenerated cases ($c \approx 0$) for it averages only coherent values. See [222] for more details.*

The resulting orientation $\overline{\mathbf{o_p}}$ at point $\mathbf{p}$ is a weighted mean over the neighborhood $\mathcal{N}_\mathbf{p}$:

$$\overline{\mathbf{o_p}} = \sum_{\mathbf{q}\in\mathcal{N}_\mathbf{p}} w_\mathrm{s}(\mathbf{p},\mathbf{q})w_\mathrm{v}(\mathbf{p},\mathbf{q})w_\mathrm{c}(\mathbf{p},\mathbf{q})\overline{\mathbf{o_q}}$$

The weight of $\overline{\mathbf{o_q}}$ is split into:

- A spatial term w_s that considers the location of $\mathbf{p}$ and $\mathbf{q}$.
- A variance term w_v that accounts for the reliability of the measure.
- A color term w_c that accounts for the color similarity in the sequence.

We use Gaussian functions for each of them. The first one uses the Euclidean distance:

$$w_\mathrm{s}(\mathbf{p},\mathbf{q}) = \exp\left(-\frac{||\mathbf{p}-\mathbf{q}||^2}{\sigma_\mathrm{d}^2}\right)$$

For the variance term, we rely on the ratio $\rho(\mathbf{p},\mathbf{q}) = \dfrac{V_\mathbf{q}}{V_\mathbf{p}}$ to define:

$$w_\mathrm{v}(\mathbf{p},\mathbf{q}) \;=\; \exp\left(-\frac{\rho(\mathbf{p},\mathbf{q})}{\sigma_\rho^2}\right)$$

The appearance similarity is evaluated from the maximum color difference Γ between $\mathbf{p}$ and $\mathbf{q}$ in the image sequence. This comparison must only account for illumination similarity disregarding other phenomena like glints that are related to the fiber structure and not to the strand geometry [147]. Therefore, Γ is computed on the low frequencies of the images (Fig. 5.11). This leads to:

$$w_\mathrm{c}(\mathbf{p},\mathbf{q}) \;=\; \exp\left(-\frac{\Gamma^2(\mathbf{p},\mathbf{q})}{\sigma_\Gamma^2}\right)$$

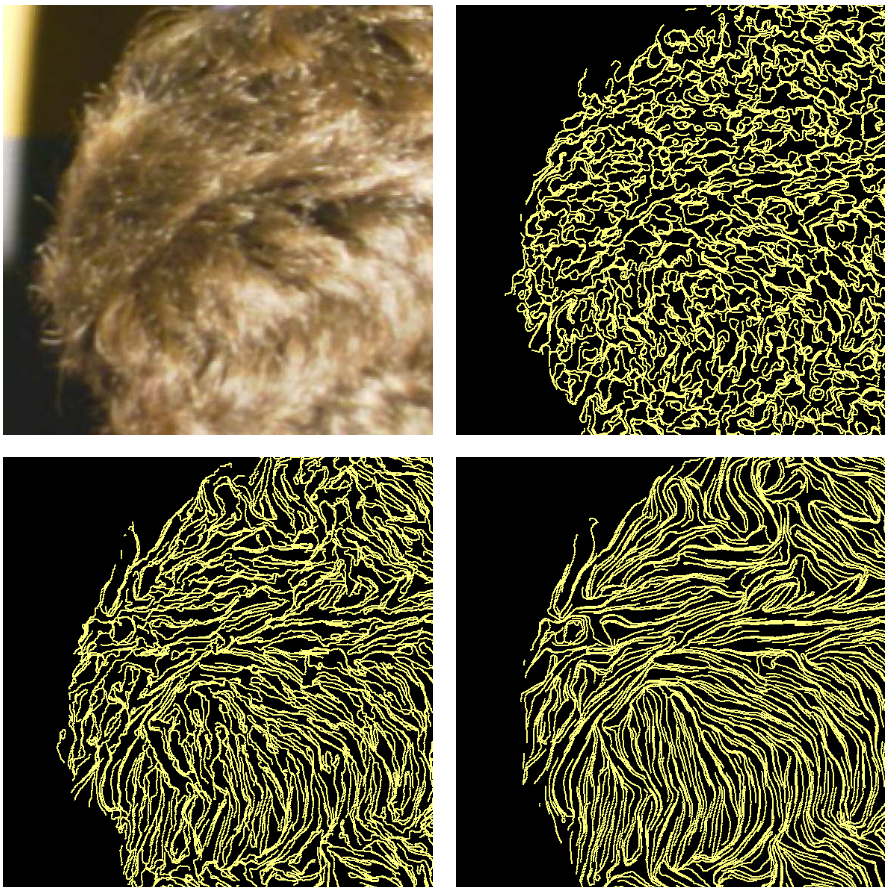


Fig. 5.16: Orientation result on a real hair

We compute an orientation for each pixel but displaying them all would clutter the picture. So we show flow lines from 2D orientation fields. ***Upper-left:*** *A closeup image from the sequence.* ***Upper-right:*** *Results of Sobel filter.* ***Lower-left:*** *Our* unenhanced *results before the bilateral filter.* ***Lower-right:*** *Same results* enhanced *by the bilateral filter.*

Fig. 5.17: Influence of the difference treatments on the final result

Upper-left: *Original Image from sequence.* ***Upper-right:*** *Result from selecting among many filters (different pixels can use different filters) but no bilateral filtering; note the noisy aspect of the image.* ***Lower-left:*** *Result from using single Gabor filter on a single image with bilateral filtering; note the large error on the right part and the wrong highlights on the top and in the middle region.* ***Lower-right:*** *Result from selecting among many filters with bilateral filtering.*

Figure 5.16 on the preceding page illustrates the significant improvement brought by this treatment. Note that it preserves the orientation discontinuities due to overlapping strands.

In our experiments, the best results are achieved with: $\sigma_d = 3$, $\sigma_\rho = 1$ and $\sigma_\Gamma = 0.1$.

How to set **[Parameter sets and weights]*:*** *We have tested several sets of parameters on the reference image shown in Figure 5.18 on the following page. From these experiments, we have selected the set with the highest precision. We have performed a similar choice for the weights of the bilateral filter.*

Comparison and Validation

Figure 5.16 on page 146 compares the orientations computed by the Sobel technique used by Grabli *et al.* [78] with our *unenhanced* results and with the same results *enhanced* by a bilateral filter. Sobel fails to provide any satisfying orientations whereas the enhanced results are convincing. Figure 5.17 on the page before show the importance of using various filters and bilateral filtering on the final result of the pipeline.

The method is further validated with a reference image of known orientations. The result of this experiment is shown in Figure 5.18. We have made a numerical evaluation of: the Canny filter in its classical use (the gradient is estimated from the x and y first derivatives of an isotropic Gaussian, the pseudo-wavelength is set to 2), the Sobel filter, our unenhanced and enhanced method; the mean errors for these filters are: 43°, 17°, 2.9°, and 2.3°, respectively. It outlines the precision of our method and shows that our bilateral filtering is a real enhancement (20% better) and not only visually pleasing. More figures are given in Appendix B.2 on page 177.

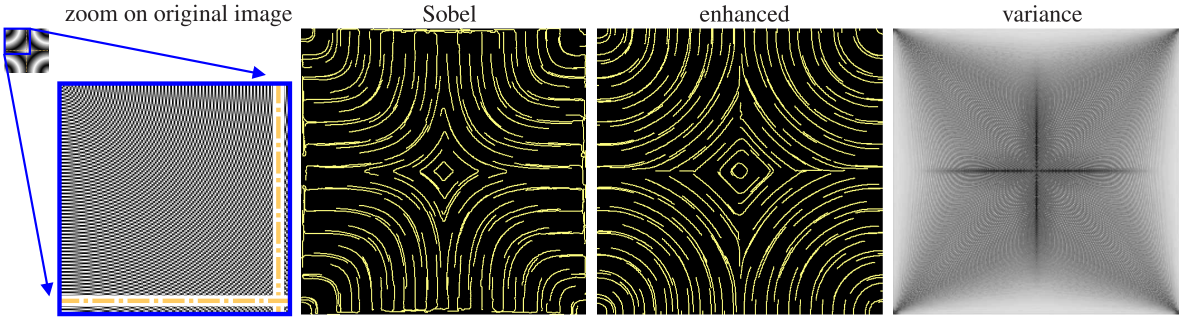


Fig. 5.18: Validation of the orientation measure on a reference image

The original image is composed of 4 radial sine signals centered on the 4 corners with wavelength $\lambda = 2$ (dotted lines are the symmetry axes of the image). The image is aliased by itself because of the short wavelength; there may be additional artifacts due to your printer or screen. Orientation measures are actually dense (1 per pixel) but only some flow lines are shown for clarity. The Sobel filter produces results too curved whereas our enhanced *(using variance-based filter selection and bilateral filtering) data retrieve correct values. The right image shows the reliability (variance: white for 0, black over 1) of the filter selected by our method. As expected, it distinguishes the discontinuities and circle centers that make the measure less reliable.*

Summary: Experiments show that our variance-based selection yields especially accurate results on both synthetic and real pictures. Both visual and numerical evaluations confirm this performance.

5.4.2 Capturing the normal vector

From our 2D study, a given segment is constrained to lie in a plane (Fig. 5.7 on page 136). We now characterize its orientation inside that plane by analyzing its appearance under varying illumination. This relies on a minimal knowledge of the fiber model from Marschner *et al.* [147]: The specular peak

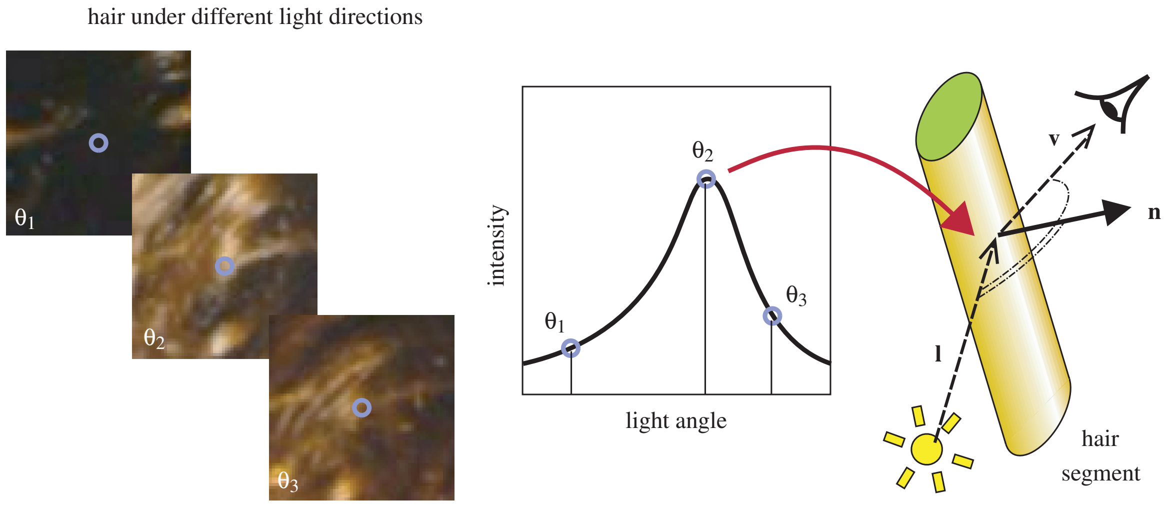


Fig. 5.20: Determination of a vector normal to a hair segment

For each pixel, the maximum intensity in the image sequence characterizes the peak reflection configuration for the underlying segment. This directly leads to a normal vector: $\mathbf{n} = \frac{\mathbf{v}}{||\mathbf{v}||} - \frac{\mathbf{l}}{||\mathbf{l}||}$.

occurs in the standard reflection direction but the surface is slightly tilted toward the root because of the *cuticles* (see Figure 5.19).

Therefore, from the intensity curves (*i.e.* intensity as a function of the light position) we can recover the light position corresponding to maximum reflection. As explained by Lu *et al.* [142], this gives a normal to the hair fiber (see Figure 5.20).

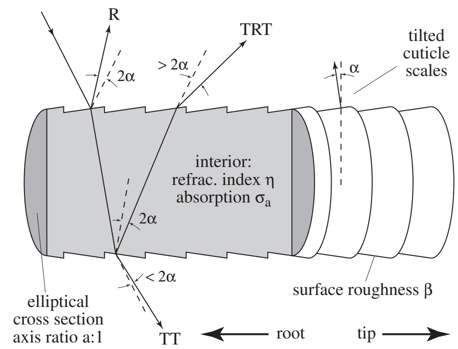


Fig. 5.19: Hair fiber model of Marschner *et al.* [147]. *Our study only relies on the existence of a direct reflection (R).* [By courtesy of Steve Marschner]

By fixing the viewpoint (and the subject) we can observe the same segment under varying lighting conditions. Consider a light describing a circular motion around a segment modeled as a cylinder. If the plane of the light motion is not perpendicular to the segment axis, we are guaranteed to have a specular highlight. If the light motion is perpendicular to it, there is no intensity peak because we are always in the specular region. To avoid this, we capture images with the light moving in two orthogonal planes (from left-to-right and bottom-to-top). Thus, we are sure to have a useful sequence. We select the one whose plane has the lowest angular deviation with the computed 2D orientation. For example, if the segment is vertically aligned, we choose the bottom-to-top path so that the trajectory is never orthogonal to the segment. Figure 5.21 on the next page shows sample intensity profiles that are obtained from these light paths.

This technique still holds for an elliptical cylinder hair model without modification, but it would need to compensate for the cuticle angle by slightly rotating the computed segment toward its root. However, we have found that the results are satisfactory without this offset and in practice, on some complex hairstyles, the root direction may be unknown.

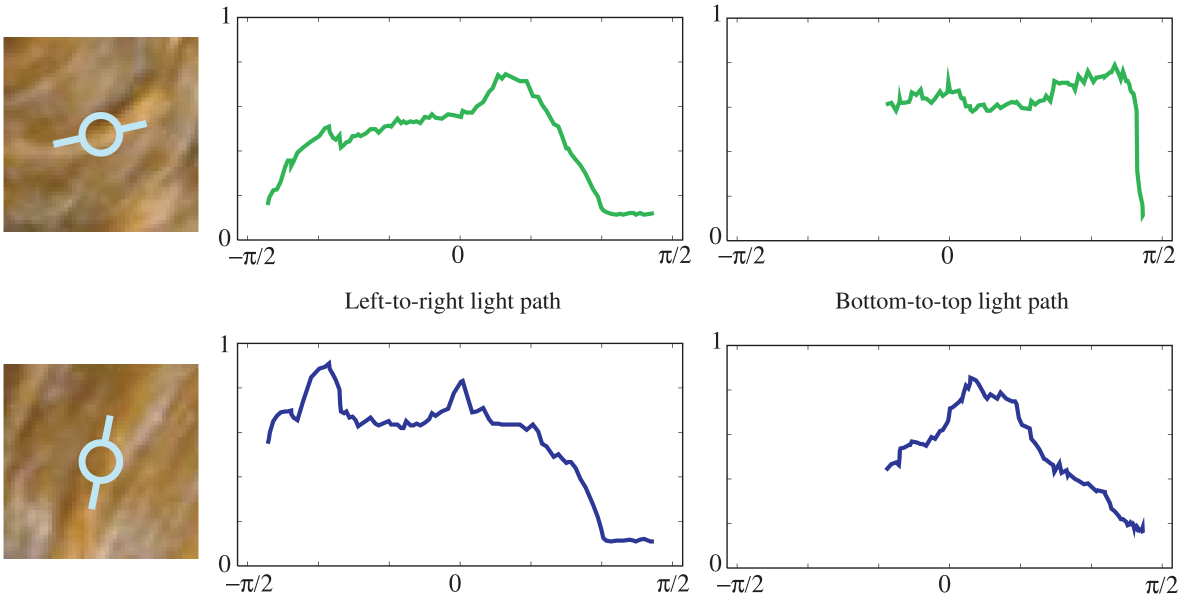


Fig. 5.21: Intensity plots for two segments with different orientations

The light paths orthogonal to the segments' axes are flat (with potentially the 2 caustic lobes predicted by Marschner et al. [147]) and give no information about the normal. The other paths have a "diffuse+specular" response that characterizes the normal. The choice between both paths is made from the 2D orientation of the segment. The bottom-to-top path is limited by the floor.

One important caveat is that the light source does not describe a full circle. Our measurements would be impaired. Forward scattering would occur if the segment were between the light and the camera. We therefore limit the light-view angle to lie between $\pm\frac{\pi}{2}$ (For the vertical light path the bottom angle is limited by the floor to $\approx -20^{o}$). This bounds the detectable normal between $\pm\frac{\pi}{4}$, and still addresses the case of a great majority of the visible segments. The segments which have normals outside this interval generally lie near the silhouettes and will be better captured from another viewpoint.

Summary: Tracking the highlights through the image sequences characterizes a light position corresponding to the mirror reflection configuration. In that case, since the camera position is known, it is straightforward to compute a vector normal to the considered segment.

For each segment, this normal defines a 3D plane that is intersected with the plane coming from 2D analysis. The resulting line is the 3D orientation of the segment.

5.5 Practical implementation

In the previous section we have shown how to build a 3D orientation field (sometimes called *line field* in the literature) from a given image sequence from a given viewpoint. To build a full model of a person's hair we need multiple viewpoints. For our initial implementation, we use four sequences: right, left, back, and top. We register these four sequences to build a 3D orientation field that covers the whole head. The final part of the algorithm is to grow strands following the 3D orientations. The produced strands are dense enough to form the visible part of the hairstyle. As previously mentioned, the hidden part of the hair is not recovered by our system.

5.5.1 Viewpoint registration

To generate hair throughout a head, we need a 3D orientation field that covers the whole head. We use a simple setup to capture image sequences using a single camera. Hence, we need to register the different viewpoints on a common coordinate system (the intrinsic parameters of the camera are known and constant). A more sophisticated setup using multiple cameras could obviate this step.

Our registration is done manually: An ellipsoid is fitted to bound the hair volume. In practice, a bounding ellipse (the projection of the ellipsoid into the viewing plane) is fitted for each viewpoint. Using the camera parameters and the ellipses, the ellipsoid is characterized and the cameras are located relatively to it (see Appendix B.3 on page 179). The camera-to-image precision is in the order of 1mm (all the previous steps rely only on this one) whereas the registration between the different viewpoints is in the order of 10mm (impacting only the thin overlapping regions, see Figure 5.22 on the next page).

More about* [Acquisition system]*: *We have used a home-made system and we consider it only as a proof of concept. An ideal setup would be composed of:*
- *A head support to avoid the head moves (e.g. dentists have such systems for tooth X-ray).*
- *Several fixed cameras (at least four but more would be better) to capture all the viewpoints at once. These cameras would finely be calibrated off-line.*
- *Finally, we see two main choices for the light: either a robotic gantry or several fixed bulbs. The former one permits almost continuous light paths but would be less precise since the light moves. The latter restricts the light positions but they are accurately known since off-line calibration is possible.*

5.5.2 Hair growth

For each view, we define a hair region (mask) which we use to compute the *visual hull* [130] of the hair volume (see Section 2.2.2 on page 17 for details). We limit the influence of a viewpoint to half of the volume because the left viewpoint would interfere with the right one – for instance, the boundaries around the ears may not match. The visual hull is only used to ensure that the synthetic hair lies inside the original hair volume.

The bounding ellipsoid we used for the viewpoint registration is then shrunk to fit inside the hull to approximate the *skull*. More precisely, this will be the surface of the starting points of the visible strands. When the hair is thick, this surface will be slightly bigger than the real skull. Alternatively, another method for acquiring or approximating a person's skull could be used (this point is discussed later in Section 5.7.1 on page 159).

To form a strand, a point is picked on this skull. From it, we iteratively chain the 3D segments (Fig. 5.23). The 3D segment at point **p** is computed by projecting **p** into the visible image planes. The 3D orientation at the projected point is computed in one of two ways:

If only one camera "sees" the point. The 3D segment is straightforwardly computed with a length that corresponds to the back-projection of the image pixel ($\approx$ 1mm); its direction is the one that is most similar to the last segment added to this strand (no sharp angles). The initial direction of the strand is chosen to match the general hair direction e.g. up-to-down for most long hairs. If no such direction exists, the direction pointing outside the skull is used.

Alternatively, if more than one camera sees the point, we limit the use of grazing lines of sight, because corresponding normals may not be accurate as previously discussed. We select the two views with the lowest grazing angles ψ (see Figure 5.22). We then average the corresponding orientations weighted by $\cos^2(\psi)$. The desired behavior for this blend function is favoring the single-view data that enjoy an accurate calibration for both the subject and the camera are fixed. It should also provide a smooth transition between the different views but on limited areas since the camera-to-camera calibration is likely to introduce additional errors (especially in our home-made system). Figure 5.22 shows the influence of each view and confirms that 4 views (top, back, left and right) correctly cover the hair volume. We can also check that the 3D information for pixels is blended between different views only in limited overlapping areas.

How to* [Compute a mean with 3D orientations]*: *In 3D, the sum of more than 2 orientations is not defined. The sum of 2 orientations is defined by considering their common plane and the sum of 2D orientations within this plane.*

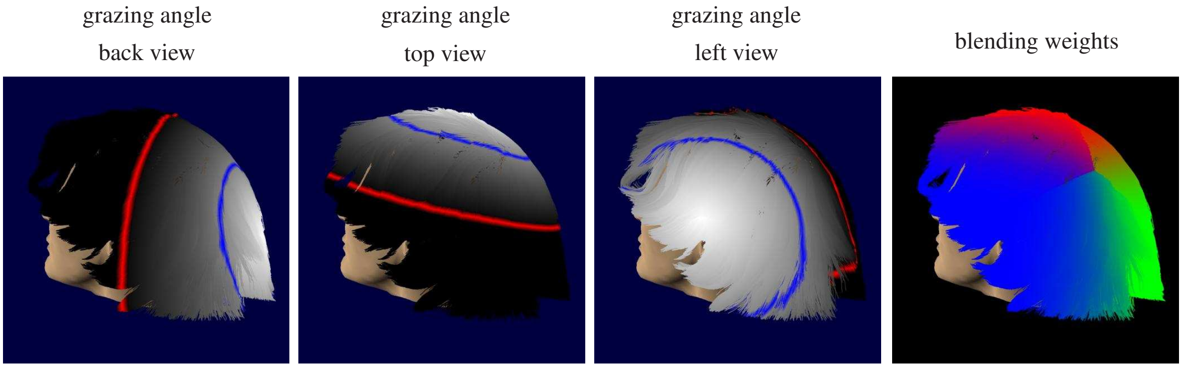


Fig. 5.22: Grazing angles relatively to three views

The color scale goes from white (0^o) to black (over 90^o). The blue and red lines correspond to 45^o and 90^o. The lower-right corner shows the blending weights deduced for the three views (blue: left, green: back, red: top).

A strand is ended if it touches the visual hull boundary or if it reaches a certain length; the latter being more common. To reduce unnecessary computation, we end the hair strand if more than a number of them pass over a pixel (10 segments/pixel in practice). Depending on the model, we generate from 30,000 to 70,000 strands.

Though it is a robust process, some isolated segments may still be wrongly oriented. Even if their number is limited, their visual effects may be noticeable. Therefore, in a post-process we reduce the highest curvatures of a strand with a diffusion process on the vertices $\mathbf{p}$ for which the curvature κ is higher than a threshold κ_0. The strand evolves according to (with s the curvilinear abscissa):

$$\frac{\partial \mathbf{p}}{\partial t} = \begin{cases} \dfrac{\partial^2 \mathbf{p}}{\partial s^2} & \text{if } \kappa > \kappa_0, \\ 0 & \text{otherwise.} \end{cases}$$

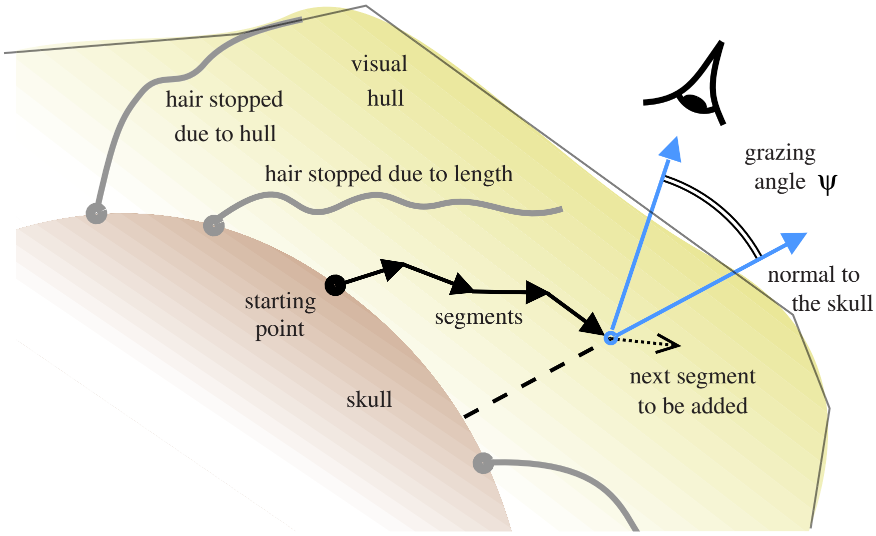


Fig. 5.23: Hair strand growth

The starting point is picked on the skull, segments are then added one by one. The influences of the cameras are weighted by their grazing angles.

until it stabilizes. In practice, we use κ_0 to bound the curvature radius over $\approx$ 10mm. We also limit the number of iterations to the number of segments within the strand to avoid modifying too much the original captured geometry.

Summary: The last step to create a whole head of hair is to build strands from the 3D segments we have previously computed. This step is the one that aggregates the data from the different viewpoints.

The strands are grown by chaining the segments together. When a segment is seen by several cameras its orientation is the average of the value coming from these views to produce smooth transitions. A strand is ended when it reaches the visual hull or a threshold length.

5.6 Captured hair results

We first present the setup we use to capture hair. We then present and discuss our results.

5.6.1 Technical details

Setup

To acquire the image sequences, we use a setup in which the subject is able to keep his or her head fixed for several seconds. A fixed video camera captures images at 7.5 frames per second with a 1024×768 resolution (in practice a region of $\approx 550 \times 550$ for the hair because the mirror balls must visible). A point light source aimed at the subject's hair is moved while its distance to the head is constrained ($\approx$1 m) and its angular position is known thanks to 2 mirror balls. Figure 5.24 shows a picture of our setup. To acquire several viewpoints, the person turns her head.

In a few seconds, hundreds of images are taken. Thus we can assume that the hair is not moving throughout the sequence. For each viewpoint we currently segment the hair region manually.

Four viewpoints are used: top, right, left, and back. This is a minimal set for reconstructing the whole model of hair. A more sophisticated setup with more cameras would do this in one pass.

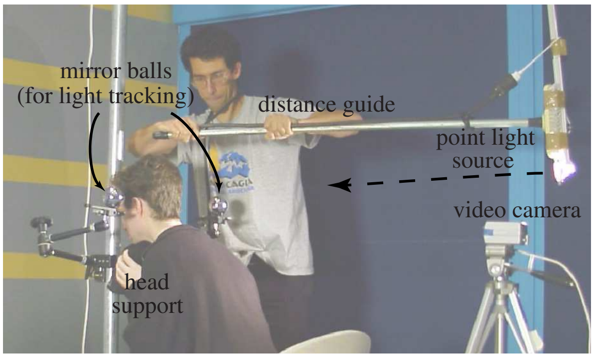


Fig. 5.24: Our simple setup to capture a subject under moving light and fixed camera.

Rendering

We use the rendering model of Marschner *et al.* [147] for our visual comparisons. The parameters are determined to roughly match the original color of the hair. This model is restricted to a single fiber and does not account for hair-to-hair shadowing, scattering, etc. A neutral head is placed under the hair to provide a consistent image. Each strand is rendered by anti-aliased GL lines; the color is computed at each vertex. This rendering step is only provided for illustration purpose: highlights convey useful visual cues about the hair geometry. We do not claim any contribution about hair rendering.

Difficulties

For some persons, the skull is poorly approximated by an ellipsoid (it would require a more sophisticated shape such as a super-quadric); for long-haired person, it may also be hard to guess it under the hair. Figure 5.25 shows an example of this problem; the ellipsoid cannot perfectly match both side and back silhouettes. We have chosen to better approximate the side profile, thus the back profile does not match; however, the hairstyle is still correct.

Acquiring data for long hair requires special care: Hair strands are likely to move when the subject turns for a new viewpoint – especially for the sequence from the top. In this view, the hair silhouette is no longer consistent with the other views; we ignore it for the hull computation. To minimize the potential perturbation, the blending weights of the top view are halved. These slight changes overcome these difficulties with a limited impact on the overall quality. A complete system with several cameras would eliminate this issue (see the box on page 151).

The hair falling from the top of the head over the top of the face are captured with a lower precision because we do not have a front view. An additional camera should ameliorate this point.

Timing

The acquisition of the 4 sequences takes about 5 minutes. Segmenting the hair areas takes 5 to 10 minutes. Running the filters and selecting the lowest variance lasts about 1 hour per view point (the convolutions involve fast Fourier transforms on large domains); we found that using nine images representative of each sequence yields good results. All the other steps (light tracking, segment chaining, etc) takes only a few minutes. (We use an Intel-Xeon 2.4 GHz.)

5.6.2 Validation

Overall accuracy

Ideally we would like to compare our captured model with the ground truth which is obviously unknown. A useful consistency check however involves verifying that a rendered image sequence using the light path from the capture setup on our hair model creates similar reflection patterns. So, for each

Fig. 5.25: Capture of a black tangled hair

Left: *comparison with original view.* ***Right:*** *a view different from the input sequences. This example demonstrates the interest to exploit several images to compute the 2D orientation. With only one picture, a large portion of the hair is too dark. Exploiting several images and hence several light position makes possible to always work on a well-illuminated image.*

pixel we find the angle in 3D space of the light direction corresponding to the specular peak in the synthesized sequence and compare it to the angle in the original sequence (Fig. 5.28; we compare an image sequence like Fig. 5.27 on page 158). The actual corresponding normal error is half this angular error (from the classical mirror reflection formula).

For example, a 5^o error at a pixel means that the highlight in the synthetic image occurs at a light position that is 5^o different from the real sequence.

We find a mean error of 13^o and median error of 6.4^o. This difference shows that there are a few large errors. Those are mainly near the silhouette because of hair-to-hair forward scattering which is not rendered. This is confirmed by only considering the front facing region (inside the blue line in Figure 5.22): the mean and median errors drop down to 7.6^o and 5.0^o, respectively.

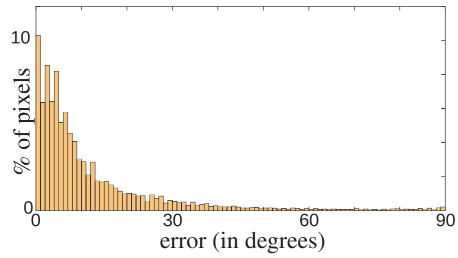


Fig. 5.28: Angular specular error distribution.

This error seems reasonable since it sums the errors from the whole process (setup calibration, blending, no-cuticle approximation, etc). Moreover a visual comparison conveys a convincing similarity. Figure 5.27 on page 158 shows one such result, notice the highlights at the back and near the top of the head are aligned.

Fig. 5.26: Capture of a long wavy hair

Left: *Comparison with an original view.* ***Right:*** *A view different of the input sequences. We encounter special difficulties for this hairstyle because of our setup. Changing the head position makes the hair strands move. However, a production-grade system would overcome this difficulty.*

Types of hair

Our technique is able to work on a wide range of hairstyles and colors as illustrated by Figures 5.25, 5.26 and 5.27,. Large and small curls and waves are accurately captured. Black hair is challenging because the strands are only visible in the highlights. Nonetheless, our method is robust enough to handle it, even with complex small features.

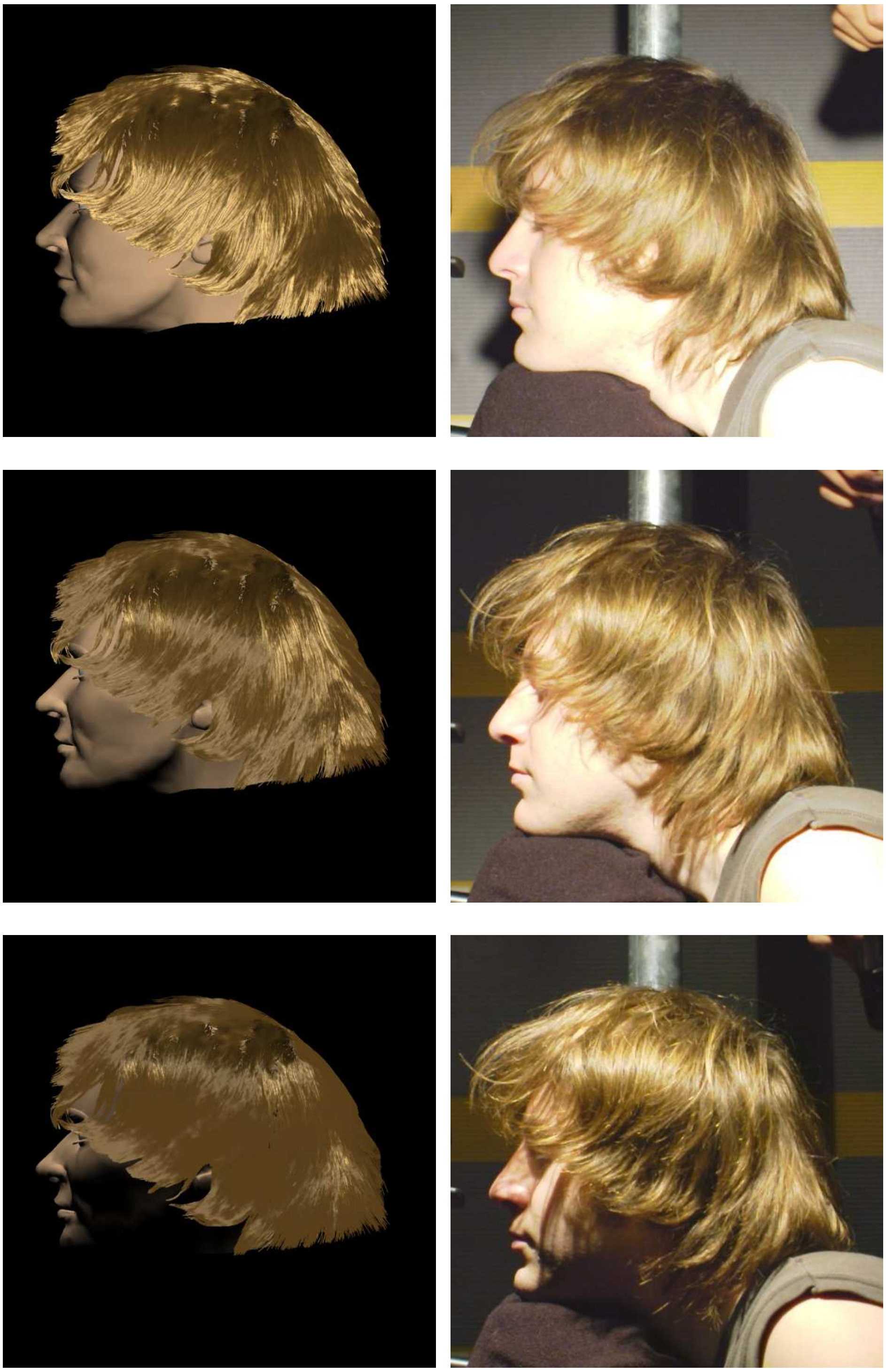


Fig. 5.27: Side by side comparison

The captured geometry is rendered under the same lighting condition as the real images on the right. Note how the reflection patterns are similar.

5.7 Discussion

5.7.1 Approximating the skull

Determining the starting point of a strand is a crucial issue. In our current implementation, we rely on the user to provide an ellipsoid which is sometimes a poor approximation of the skull shape. Therefore, this point is to be improved to construct a production-grade system.

Hairstyle transfer

A first remark is that even with a poor approximation of the skull, our technique reproduces the correct hairstyle. In other words, our technique is robust enough to handle slight deviations of the skull shape while still capturing the overall aspect of the original hair. We may exploit this feature to “transfer” an hairstyle to another skull: Currently, we first create the hair geometry (a set of curved lines) and then fit a generic head model to render complete model (head+hair). As a consequence, the head shape depends on the hair geometry. It may be useful to inverse the process: Given a head model, produce a hair geometry adapted to it. This could be done by using the head 3D model as the skull shape on which the strands are grown. This would produce a hairstyle that globally matches original one while being tightly adjusted on the head 3D model.

Better shape

Hairstyle transfer can be useful but does not address our main problem: capturing the original shape of the skull. The difficulty of this task is that what we call “skull” is not the skin-covered bone that supports the hair roots. Our “skull” is the surface formed by the starting points of the visible strands. Hence, X-rays are not adapted to retrieve its shape. Thick hairstyles are especially difficult because the hidden part of the strands is important. It creates a large difference between the bone skull and the surface we seek.

We believe that the solution is to use an image-based approximation of this “skull”. First, one should notice that retrieving an accurate shape is almost equivalent to solving the whole hair-geometry problem. It would be then sufficient to compute the 2D orientations and to project them on this surface. Unfortunately, as previously discussed in the Previous Work, producing an accurate surface is hard. Hence, we target only an approximation.

Multi-view techniques A first option is to use a technique such as the ones reviewed in Section 2.2 on page 8. But, from our experience, these techniques hardly produce precise results on hair. Their accuracy is impaired by complex phenomena that are both view-dependent and high frequency. These methods currently produce plausible results whose accuracy has still to be evaluated.

Visual hull A second option is to use the visual hull. Its main advantage in our context is its robustness to view-dependent effects. But it suffers from the concavities that are not captured. It also requires numerous view to produce a satisfying geometry (without sharp edges).

We think that if the approximation is good enough, the edges can be smoothed away. A shrunk version of the visual hull should be a satisfying approximation of the skull. The concavities are so penalizing because hair volume seldom has large ones. Numerous images should be used. A correct setup would at least use a 45° sampling of the view sphere. This guarantees at most 45° edges. Note that our four-camera setup is 90° sampling that hinders this approach.

Summary: Estimating the "skull" is a complex issue since what we seek is not the classical skull but the surface formed by the points where the strands become visible. It therefore challenges techniques such as X-rays that would capture the real skull.

The robustness of our technique can be exploited to use any skull shape as a support for the geometry. This would allow to transfer a given hairstyle on a head model different from the original one.

But to better match the original hair geometry, we need to use a more precise approximation than our simple ellipsoid. We see two main options. On one hand, Computer Vision techniques recover plausible shapes but still lack of accuracy on this complex task. On the other hand, visual-hull technique is more robust but requires a more sophisticated setup.

5.7.2 Orientation evaluation and a priori knowledge

In our approach, we introduce the a priori knowledge of the local coherence of the segment orientations. It results in the bilateral filtering. This step is separated from the orientation measure *i.e.* we measure a dense orientation field and then we apply a bilateral filter.

Remember our surface reconstruction technique (Chapter 2 on page 5): we measure a dense depth field while accounting for an a priori piecewise smoothness. The difference is that the prior is directly integrated in the measure.

The first remark is that, opposite to surface reconstruction, the orientation problem is well-posed. Hence, regularization (the prior in our case) is not mandatory to properly solve the problem. However, using an integrated approach may be interesting.

The bilateral filter has an important drawback: It does not estimate how consistent the newly assigned value is. The new value is not tested on the input images before being assigned. Because of the separation, the bilateral filter ignores the filter response curve; it "knows" only its variance. On the other side, an integrated scheme would account for the filter response of the potential new orientation. Hence, such a scheme is likely not to assign an inconsistent value.

But an integrated approach raises a hard question. Since it considers the actual value of the filter response and handles several pixels at the same time, it somehow "compares" responses from different filters. A straightforward comparison is not rigorous since it deals with different entities. A solution would be to work from a single filter and a single image but the loss of information would be too important. Therefore, studying the comparison issue would be an interesting problem for future work.

To conclude, one has to remark that our current system does not impair the final result and already provides accurate results. Hence, the expected improvement would essentially be the ability to work from a more limited input (less images).

Summary: Our orientation measures are post-processed by the bilateral filter to account for the local coherence of the hair strands. This treatment is applied afterward without evaluating the relevance of the newly assigned values and, therefore, may induce inconsistency relatively to the input images.

To address this issue, we may use an approach similarly to our surface reconstruction technique described in Chapter 2. On one hand, it would better integrate the local coherence within the process. On the other hand, preserving the advantages of the presented method (selection between several filters and several images) does not appear straightforward. It would be an interesting direction for future work.

Since the current results are already accurate, the potential gain would be the possibility to work from less images.

5.8 Conclusions

We have presented a technique to capture the real geometry of a person's hair from multiple images. This system uses the complex reflectance properties of hair to retrieve its 3D geometry. To take advantage of this light interaction, we introduce a way to compare results from different filters and parameters. The variance method links signal processing and statistics to reach precise and robust measures. The theoretical foundation of this approach has been studied and shown to be related to other classical methods. We have also exposed how to exploit the light reflection of hair to extract valuable 3D information.

As a proof-of-concept, we propose a simple practical setup that exploits different image sequences of a real person and show how to blend their results to generate a full hair geometry. This experiment reaches satisfying results that justify the conception of a more sophisticated system. Several cameras can be used at the same time with a light source on a robotic gantry. Such a setup would obtain a high level of precision that could open new research directions. For instance, it could be possible to densely measure the appearance properties of the hair to retrieve the parameters of a scattering model. We believe that it would also be possible to acquire such a model from a single sequence. Because this valuable knowledge would permit to work with less information, *e.g.*, less images and light positions, one can conceive the motion capture of hair.

It would also be interesting to adapt this technique to the other methods that manipulate hair. Our poly-lines cannot be directly used for efficient animation and editing. Creating specific data structures from images for these tasks like wisp hierarchy [15, 115] or fluid flow [80, 81] is a promising direction to explore.

From a theoretical point of view, the study of filters also requires further attention. We have presented and validated some parameters that we use in our filter selection. Future work will look at other parameters. Since it only exploits the response curves, it is possible to apply this selection scheme to other fields. The only requirement is, for a given measure, to have several "sensors" that provide a response curve.

6
Conclusions

In this dissertation, we have addressed three main tasks involving data extraction from images:

- The surface reconstruction from several points of view.
- The face relighting from a single image and a 3D model.
- The capture of hair geometry from multiple light positions.

Our first remark is that these three points illustrate the variety of the information that can be exploited in images: contour, highlight, texture, color, shading, etc. Furthermore, this information becomes even richer when the evolution between several images can be observed.

We have also shown several useful configurations: fixed or moving camera, fixed or moving light, images alone or image and 3D model. We propose a dedicated algorithm for each setup that extracts a dense information.

From these case studies, we propose a few general conclusions.

Redundancy is useful

In the work on surface reconstruction and hair capture, we use numerous images of the same object with a limited variation between two consecutive images (small camera or light move). Even if the acquired data are redundant, we are convinced that this redundancy contributes to the accuracy of the presented results.

Exploiting redundancy makes our algorithms robust *i.e.* they can work from non-perfect input data. Since the information is “reproduced” several times in the data, we are able to compensate for potential erroneous values. So we can “trust” more the input data and use less a priori knowledge to drive our processes. We are “closer to the original data” because we can detect the outliers. This robustness is all the more important that we target a dense information: We have to produce a reasonable result even for the parts that are poorly represented in the input images.

The completeness and the high level of detail of the surface and of the hair are an evidence of this situation: We produce a good match with the real objects.

Information is visible

In all the presented pieces of work, the extracted information is "visible" in the input data. If we play the sequence of images used for surface reconstruction as a short movie, everyone perceives the 3D of the scene. For face relighting, everyone evaluates at first sight the skin state between matte, dry and wet. And it is easy to follow the hair strands with a pen within our hair sequences.

This remark has encouraged us to push forward our study even in the hardest case. For instance, for the hair capture, it was not clear from the original images that we can recover a dense orientation. But once, we have seen the high-frequency image with clear lines and almost no highlights, we were convinced that "if we *see* these lines, we can get them".

This fact might not be always true. Remember the "flat cube" (Fig. 2.1 on page 6): We guess a three-dimensional shape whereas there is objectively nothing 3D in there. However, in that case, it is possible to introduce the a priori human knowledge based on conventions about parallel lines and angles to retrieve the 3D information.

Nonetheless, it might exist more complex cases that we cannot work around even we "see" the information. But we are convinced that these cases are rare and deserve to be studied: Our brain extracts the information, hence there is a chance to find a solution.

Appropriate structure is important

We have already discussed this point for the surface reconstruction issue: We are convinced that a 3D formulation should be used when a 3D shape is targeted. We also think that reconstructing lines for the hair geometry is an crucial choice that paves the way toward broader applications (reflectance acquisition, geometry edition and animation) and is more appropriate than a textured surface. And about the face relighting, understanding the influence of the parameters leads to determining a meaningful set of parameters and to designing a robust extraction scheme. Manipulating an appropriate set of parameters makes possible to robustly extract a complete reflectance model from a single image.

In this dissertation, we present methods aiming at reconstructing the original object disregarding the use of the extracted data (except the relighting part that targets efficient rendering): We want a surface that matches the one from the real objects, a set of hair that copies the original hairstyle. These data are somehow "all purposes" data; they can be used for any application, potentially after some treatment. It would be interesting to study the creation of "targeted" data that accounts for the application. We detail more this idea in the Future Work section.

6.1 Future work

Beyond the future work of each presented section, we can propose a few general insight about the future research following our work.

Volumetric data First, it would be interesting to study other cases. In that way, Reche *et al.* [178] propose a study about tree reconstruction: Trees can be seen as non-opaque material. This leads to a volumetric reconstruction which differs from our surface and line approaches. Following this direction, we may consider other volumetric entities such as translucent objects (*e.g.* glass, bottle) but also smoke and water. Applications would be inserting a digital actor within a real cloud of gas or making possible complex interaction between real water and digital entities.

Influence on the environment Another interesting direction is pointed by Chuang *et al.* [40] who capture the shadow of an object under the sun and adapt it to a new scene. This work is inspiring in that it does not capture the object but rather its effect on the environment. Goesele *et al.* [73] also propose a first step to measure a directional light. It would be interesting to extend these techniques to other "exiting" effects such as a general light source or more complex global illumination effects (caustics, soft shadows, etc). This could lead to useful applications such as improved relighting or better inclusion of the captured objects into a new scene.

"Targeted" data Beside considering other scenarii, we can also study "targeted" data as already mentioned. By that, we mean that we could account for the future use of the extracted data during the capture process. In this dissertation, only the face relighting technique follows this approach. Considering surface reconstruction, it would be for instance result in creating NURBS surfaces with control points if the geometry is to be further edited by the user. A proper approach will be to directly determine the NURBS from the images and not to fit them on previously extracted data. Such a direct link is more likely to produce consistent results. If the surface is to be rendered, then considering dedicated data structures like *billboard clouds* [59] is a suitable choice.

Applying this "targeted" approach to hair leads to consider structures dedicated to animation such as wisp hierarchy [15] or to editing such as fluid flow [80]. Note that our current process is close to a fluid structure when we are dealing with 3D orientation field. On the other side, creating an image-based wisp hierarchy appears more challenging.

Closure

As a closure, we would say that this research field is broad and challenging. Great progress is yet to expect and we feel that we are getting closer and closer of production-grade algorithms. We have shown that complete and dense information can be obtained from images. Usability is still to be improved (*e.g.* automatically setting all the process thresholds and weights). However, the communities of Computer Vision and Computer Graphics have come closer and the results are numerous and promising: Numerous researchers propose three-dimensional approaches based on images: 3D photography [154], 3D video recorder [232], 3D TV [151], etc. We are convinced that this research field will soon spawn new massively available interesting devices.

A

Technical details and perspective on surface reconstruction

A.1 First-order and second-order regularization terms

We here consider the problem of the functional design for surface reconstruction (see Section 2.3 on page 37). We demonstrate in the discrete case that using a first-order regularization term is equivalent in some sense to a second-order term.

For remaining of the discussion, we consider two discrete non-negative smoothing terms defined over a surface $\mathcal{S}$: a first-order one $\mathbb{S}_1^{\mathrm{d}}$ and a second-order one $\mathbb{S}_2^{\mathrm{d}}$. We show in this section that they are *equivalent* in the sense that there exist two positive constants c and C such that for any surface:

$$c\,\mathbb{S}_1^{\mathrm{d}} \quad \leq \quad \mathbb{S}_2^{\mathrm{d}} \quad \leq \quad C\,\mathbb{S}_1^{\mathrm{d}} \tag{A.1}$$

The right part demonstrates that bounding $\mathbb{S}_1^{\mathrm{d}}$ also bounds $\mathbb{S}_2^{\mathrm{d}}$, and that minimizing $\mathbb{S}_1^{\mathrm{d}}$ lowers the bound on $\mathbb{S}_2^{\mathrm{d}}$. The left part shows the converse result which is less interesting in our case.

Formally, we work with a surface $\mathcal{S}$ parametrized by $z = f(x,y)$ on a rectangular domain $\mathcal{D}$ (extension to more general domains is straightforward). We assume that $\mathcal{D}$ has a finite area as it is always the case in practice. The x and y axes are discretized into $\{x_1,\ldots,x_{n_x}\}$ and $\left\{y_1,\ldots,y_{n_y}\right\}$.

To define the smoothing term, we originally use α functions to deal with discontinuities (see Section 2.3 on page 37). However, in the discrete case, the maximum and minimum of these functions are defined. For the sake of clarity, we do not address the case for which the minimum is zero but it can be added by interested readers. So we have: $0 < \min\alpha(\cdot) \leq \alpha(x_i,y_j) \leq \max\alpha(\cdot)$. Hence, we can omit the α functions in the definition of $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$ for they would only affect the constants in equation (A.1).

We use the following notations:

$$\Delta_x f(x_i, y_j) = f(x_{i+1}, y_j) - f(x_i, y_j)$$
$$\Delta_y f(x_i, y_j) = f(x_i, y_{j+1}) - f(x_i, y_j)$$
$$\Delta_{xx} = \Delta_x \Delta_x \qquad \Delta_{yy} = \Delta_y \Delta_y \qquad \Delta_{xy} = \Delta_x \Delta_y$$

First study: We first prove the equivalence in a simpler case. We define $\tilde{\mathbb{S}}_1^{\mathrm{d}}$ and $\tilde{\mathbb{S}}_2^{\mathrm{d}}$ (for clarity purpose, we do not handle the index problems due to the boundaries of $\mathcal{D}$):

$$\begin{aligned}
\tilde{\mathbb{S}}_1^{\mathrm{d}} &= \sum\sum_{\mathcal{D}} \left(\left| f(x_i, y_j) \right| + \left| \Delta_x f(x_i, y_j) \right| + \left| \Delta_y f(x_i, y_j) \right| \right) \\
\tilde{\mathbb{S}}_2^{\mathrm{d}} &= \tilde{\mathbb{S}}_1^{\mathrm{d}} + \sum\sum_{\mathcal{D}} \left(\left| \Delta_{xx} f(x_i, y_j) \right| + \left| \Delta_{yy} f(x_i, y_j) \right| + \left| \Delta_{xy} f(x_i, y_j) \right| \right)
\end{aligned}$$

Proof for $\tilde{\mathbb{S}}_1^{\mathrm{d}}$ and $\tilde{\mathbb{S}}_2^{\mathrm{d}}$: First, we remark that the set of the surfaces defined on the discretized domain $\mathcal{D}$ by a function f is a vector space $\mathfrak{S}$. Then, $\tilde{\mathbb{S}}_1^{\mathrm{d}}$ and $\tilde{\mathbb{S}}_2^{\mathrm{d}}$ are two norms over this vector space (they are actually the first- and second-order Sobolev norms of degree 1). In addition, $\mathfrak{S}$ is a Banach space [224] of finite dimension for it is isomorph to $\mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$. And it is known that all norms are equivalent on such a space. Therefore, $\tilde{\mathbb{S}}_1^{\mathrm{d}}$ and $\tilde{\mathbb{S}}_2^{\mathrm{d}}$ satisfy equation (A.1) and are equivalent. $\square$

Equivalence of $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$: We then define $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$ from $\tilde{\mathbb{S}}_1^{\mathrm{d}}$ and $\tilde{\mathbb{S}}_2^{\mathrm{d}}$ by removing the zero-order term of the sum:

$$\begin{aligned}
\mathbb{S}_1^{\mathrm{d}} &= \sum\sum_{\mathcal{D}} \left(\left| \Delta_x f(x_i, y_j) \right| + \left| \Delta_y f(x_i, y_j) \right| \right) \\
\mathbb{S}_2^{\mathrm{d}} &= \mathbb{S}_1^{\mathrm{d}} + \sum\sum_{\mathcal{D}} \left(\left| \Delta_{xx} f(x_i, y_j) \right| + \left| \Delta_{yy} f(x_i, y_j) \right| + \left| \Delta_{xy} f(x_i, y_j) \right| \right)
\end{aligned}$$

Proof for $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$: $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$ are not anymore norms on $\mathfrak{S}$. However, we can build another vector space on which they are norms. Let's consider the subspace $\mathfrak{F}$ of $\mathfrak{S}$ composed of the flat surfaces *i.e.* their associated function f is constant. With this subspace, we define the quotient vector space $\mathfrak{Q} = \mathfrak{S}/\mathfrak{F}$ *i.e.* surfaces are considered up to a constant z shift. We let the reader verify that $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$ are norms on $\mathfrak{Q}$; in short:

- $\left(\mathbb{S}_*^{\mathrm{d}}(\mathcal{S}) = 0\right) \Rightarrow (f \equiv 0)$ since all the derivatives are null and therefore the surface is flat.
- $\forall\lambda \geq 0$, $\mathbb{S}_*^{\mathrm{d}}(\lambda\mathcal{S}) = \lambda\mathbb{S}_*^{\mathrm{d}}(\mathcal{S})$ is straightforward.
- $\mathbb{S}_*^{\mathrm{d}}(\mathcal{S}_1 + \mathcal{S}_2) \leq \mathbb{S}_*^{\mathrm{d}}(\mathcal{S}_1) + \mathbb{S}_*^{\mathrm{d}}(\mathcal{S}_2)$ comes from the triangular inequality on the absolute values.

$\mathfrak{Q}$ is also a Banach space of finite dimension. Therefore, $\mathbb{S}_1^{\mathrm{d}}$ and $\mathbb{S}_2^{\mathrm{d}}$ are equivalent. $\square$

Discussion

We have proven that a first-order regularization term is equivalent to a second-order one in the discrete case. We here provide a few theoretical and practical remarks.

From a theoretical point of view:

- This proof is valid only if the second-order term also contains the first-order derivatives.
- As a corollary, the property can be extended to higher order as long as the first order is present.
- More generally, a term $\mathcal{T}_{\text{low}}$ at order n is equivalent to a term $\mathcal{T}_{\text{high}}$ at order $p \geq n$ if $\mathcal{T}_{\text{high}}$ includes order n (use a subspace of order $n-1$ instead of $\mathfrak{F}$ in the proof).

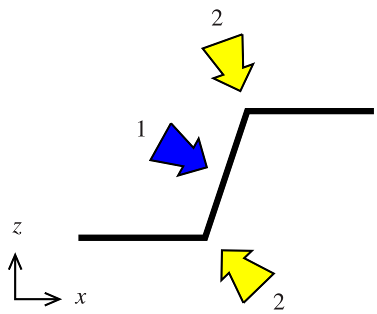


Fig. A.1: Penalty location. *A first-order term impacts the slopes (in blue) whereas a second-order term penalizes the creases (in yellow).*

From a practical point of view: In practice, it means that minimizing the first derivatives tends to also minimize the second derivatives. However, the constants in the equivalence definition (eq. (A.1)) "hide" the behavior of the regularization term. For instance, a first-order term penalizes steep slopes whereas a second-order one penalizes creases (cf. Fig A.1 for an illustration in the 2D case). Hence, even if we have shown that both terms are *mathematically equivalent*, they are not in practice.

Ideally, a regularization term would contain second-order quantities to allow the computation of values independent of the coordinate system orientation such as the surface curvature. However, we have shown that we can reach satisfying results with only first-order terms. This formal proof is one of the explanation for these results: Even if both terms do not have the same practical effects, they are sufficiently linked for our purpose.

A.2 Some ideas to extend the functional

We propose a few directions to make the functional defined on page 186 intrinsic or, at least, less dependent on the surface parameterization. We present here variations of the functional accompanied by the graph design that minimizes it. These are only early ideas that have not been implemented. They deserve further studies to achieve complete proofs and to be validated.

A.2.1 Minimal surfaces

A first solution would be to use the graph design proposed by Boykov and Kolmogorov [26]. This implies a functional of the type:

$$\iint w(\mathbf{x})\, ds$$

that corresponds to a weighted minimal surface.

Our proposal is a straightforward use of this graph within the optimization domain determined after the consistency thresholding (see Figure 2.37 on page 56). Similarly to our original scheme, linking the upper boundary to the source and the lower one to the sink ensures that the surface spans correctly according to the *vertex coordinate property* (see page 53).

Accounting for the potential discontinuities is possible by modulating the weight function $w(\cdot)$ in a way similar to the α functions of our system. This may require some attention since the α functions are 2D whereas w is 3D. But the adaptation seems possible.

A more important problem may be the lack of control over the surface shape. Without such a control, the produced surface may have some spurious folds because of noise. This would create inconsistent self-occlusions. Experiments are needed to evaluate the importance of this phenomenon. It may be that this does not happen as we have observed with our functional. Otherwise, a possible solution may be *visibility edges* (see page 58).

We lack experience with the technique of Boykov and Kolmogorov [26]. It is therefore hard to predict the quality of the potential results. Nevertheless, this approach deserves further study.

A.2.2 Functional invariant to planar rotation

We here describe some ideas to design a graph that minimizes:

$$\iint \Big(c(\mathbf{x}) + \alpha(x,y)\, ||\nabla z_{\mathbf{x}}|| \Big)\, dx\, dy \tag{A.2}$$

The advantage of this functional is that it does not depend anymore on x and y. Only the z axis matters. Therefore, the x and y axes can be arbitrarily chosen without impacting the result.

Depending on the z axis is acceptable since it ideally corresponds to the surface normal. Hence, if Functional (A.2) can be minimized, the remaining problem would be to correctly evaluate this normal.

The following derivations are mainly "informal" mathematics. It only conveys an intuition of a graph design. Many arguments clearly lack mathematical rigor and have to be better formalized in the future. Interesting directions are contained in [55, 117, 225]. Nonetheless, we are confident in the proposed approach.

Intuition on the 2D case

We first focus on the integral over the domain $\mathcal{D}_x$ of the x axis:

$$\int_{\mathcal{D}_x} \left|\frac{\partial f}{\partial x}\right| dx \tag{A.3}$$

Our strategy is to transform this integral on x axis into an integral on the z axis. The result will be more "graph-cut friendly".

Consider a small variation δx of x and the corresponding small variations δf of f and δs of the curvilinear abscissa s. Now, we rewrite Integral (A.3) and "simplify by δx":

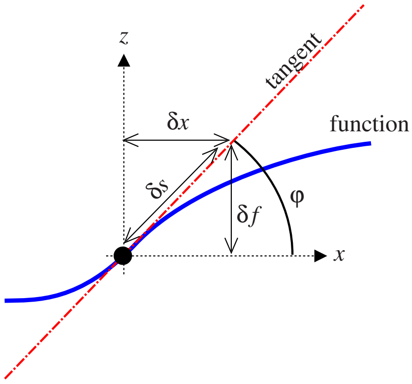


Fig. A.2: Function with its tangent.

$$\int_{\mathcal{D}_x} \frac{|\delta f|}{\delta x}\,\delta x \quad = \quad \int_{\mathcal{D}_x} |\delta f| \tag{A.4}$$

Hence, Integral (A.3) only sums the vertical variations of f. Let's introduce the angle φ of the tangent to f at point $(x, f(x))$ (Fig. A.2). We have now:

$$\int_{\mathcal{D}_x} |\delta f| \quad = \quad \int_{\mathcal{D}_x} |\sin(\varphi)|\ \delta s$$

And in fact, this relation is rigorously valid:

$$\int_{\mathcal{D}_x} \left|\frac{\partial f}{\partial x}\right| dx \quad = \quad \int_{\mathcal{D}_x} |\sin(\varphi)|\ ds \tag{A.5}$$

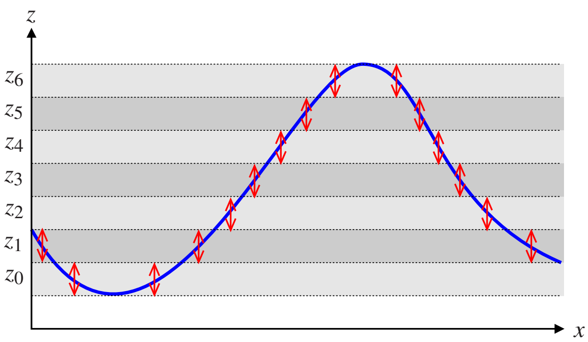


Fig. A.3: Intuition of the integration on the z domain

Let's now turn this integration on the x axis into an integration on z axis. To do so, we introduce the domain $\mathcal{D}_z$ and the set of points $\mathcal{L}(z) = \{(x, f(x)) / f(x) = z\}$.

Figure A.3 gives a first intuition for a z axis discretized into $\{z_i\}$. The total z variation is the length δz times the sum of the number of points in all the $\mathcal{L}(z_i)$:

$$\int_{\mathcal{D}_x} |\delta f| \quad = \quad \delta z \times \sum_i \underbrace{\sum_{\mathbf{p}\in\mathcal{L}(z_i)} 1}_{\#\text{points in } \mathcal{L}(z_i)}$$

Considering the continuous case, we integrate over all the z values and for each of them, we sum the contributions of the points in $\mathcal{L}(z)$. Using the tangent angle φ and the curvilinear length $\delta s = \delta z / |\sin(\varphi)|$, we obtain:

$$\int_{\mathcal{D}_x} |\delta f| \quad = \quad \int_{\mathcal{D}_z} \sum_{\mathbf{p}\in\mathcal{L}(z)} \big|\sin\big(\varphi(\mathbf{p})\big)\big|\,\delta_s \tag{A.6}$$

Formula (A.6) has a problem when $\mathcal{L}(z)$ is not countable. This happens each time f is constant on an interval. Other degenerate cases may exist but we ignore them since they do not occur in practice. In that case, the sum $\sum$ is indefinite but fortunately, all the summed terms are null because of $\sin(\varphi) = 0$. Hence, informally, it is coherent to define $\sum_{\mathcal{S}} 0\,\delta s = 0$ for any δs value and even if $\mathcal{S}$ is uncountable. So we can write:

$$\int_{\mathcal{D}_x} \left|\frac{\partial f}{\partial x}\right| dx \quad = \quad \int_{\mathcal{D}_z} \sum_{\mathbf{p}\in\mathcal{L}(z)} \left|\sin\big(\varphi(\mathbf{p})\big)\right| ds$$

And ignoring the degenerate case, we obtain:

$$\int_{\mathcal{D}_x} \left|\frac{\partial f}{\partial x}\right| dx \quad = \quad \int_{\mathcal{D}_z} \sum_{\mathcal{L}(z)} dz \tag{A.7}$$

Formula (A.7) is interesting because it shows that evaluating the integral (A.3) is equivalent to count the number of points in the $\mathcal{L}$ sets. Intuitively, it results in a simpler task from a computational point view: Sweep the $\mathcal{L}$ sets and count the number of points in each set.

Extension to the 3D case

We now consider the integral over the planar domain $\mathcal{D}_{xy}$:

$$\iint_{\mathcal{D}_{xy}} ||\nabla f||\, dx\, dy \tag{A.8}$$

We naturally extend the definition of $\mathcal{L}(z)$ to the surface points of depth z. The question is to find the equivalent of the $\left(\sum_{\mathcal{L}(z)} 1\right)$ of 2D case. In fact, one can get convinced that in the general case, $\mathcal{L}(z)$ is a planar curve. It is the classical *level curve* [226] displayed on the geographical maps. Hence, we can guess that the length of this curve may have a role.

We turn the area element $dx\,dy$ into $dg\,dl$ where dg is aligned with the gradient and dl is orthogonal to dg (a similar idea appears in [55, 117]). Hence, dl is always tangent to the curves $\mathcal{L}(z)$ and corresponds also to their curvilinear element. For dg, we have the following relation:

$$||\nabla f|| \quad = \quad \left|\frac{\partial f}{\partial g}\right| \tag{A.9}$$

Defining φ as the angle between the surface normal and the z axis and ds as the projection of dg onto the surface, we have a relation equivalent to the previous case:

$$|sin(\varphi)|\, ds \quad = \quad dz$$

We now apply relation (A.9):

$$\iint_{\mathcal{D}_{xy}} ||\nabla f||\, dx\, dy \quad = \quad \iint_{\mathcal{D}_{xy}} \left|\frac{\partial f}{\partial g}\right| dg\, dl$$

We remark that $\left|\frac{\partial f}{\partial g}\right| dg$ is informally equal to the variation $|\delta f|$ of f. Hence, we split it into several δz and integrate over the z values and over the $\mathcal{L}(z)$ curve:

$$\iint_{\mathcal{D}_{xy}} \left|\frac{\partial f}{\partial g}\right| dg\, dl \quad = \quad \int_z \int_{\mathcal{L}(z)} |sin(\varphi)|\, ds\, dl \tag{A.10}$$

The integral $\int_{\mathcal{L}(z)}$ suffers from a problem equivalent to the 2D case when f is constant on a region, making $\mathcal{L}(z)$ 2D instead of 1D. Fortunately we also have $sin(\varphi) = 0$ in that case. So, if we ignore this issue, we can write:

$$\iint_{\mathcal{D}_{xy}} ||\nabla f||\, dx\, dy \quad = \int_z \underbrace{\int_{\mathcal{L}(z)} dl}_{\text{length of } \mathcal{L}(z)} \quad dz \tag{A.11}$$

Formula (A.11) is interesting because it shows that Integral (A.8) can be computed by measuring the length of the level curves of the surface.

This derivation is not very complex should already have demonstrated in some mathematical book. Looking at Green's theorem [225] may be interesting.

Boykov and Kolmogorov [26] show how to compute the length of a planar curve with a graph cut using the Cauchy-Crofton formula. And if we observe a xy slice of our graph design, it corresponds to the length measure using a 4-neighborhood. This a poor approximation of the isotropic Cartesian distance. Boykov and Kolmogorov [26] expose how to improve this approximation using a more complex neighborhood system.

Applied to our graph, the 4-neighborhood means that we have edges only in the x and y directions. Improving our functional toward Functional (A.2) implies a more complex neighborhood *i.e.* using more edge directions in the xy plane. The capacities of the xy edges are defined from the formulæ in [26]. The discontinuity maps can be straightforwardly adapted to more directions and similarly used to modulate the edge capacities.

In the z direction, we keep the same edge scheme based on four sub-edges for each direction used in the xy plane. This still provides a convex approximation of the linear term $\alpha(x, y)\, ||\nabla z_{\mathbf{x}}||$.

To conclude, note that the degenerate case is not a problem since a function constant on a region does not cut any edges in the xy planes. Hence, this region has a null contribution as expected.

Summary: Our functional can be made invariant to a planar rotation by using more edges in the xy planes. Their capacity is are derived from the Cauchy-Crofton formula using the method exposed by Boykov and Kolmogorov [26].

B
Technical details on hair capture

B.1 Influence of the Projection Profile on Canny's Filter

We derive the formula used to plot the graph in Figure 5.15 on page 145. It demonstrates that the more extended the projection profile of an oriented filter is, the lower is the variance of its response curve – the more reliable the filter is.

We focus on the response of Canny's filter $F^{\mathrm{C}}_{(0,0)}(\theta)$ at the origin for a sinusoidal signal $s(x,y)$. For a shorter derivation and without loss of generality, we make s turn according to $-\theta$ and fix the orientation kernel filter. Since the final computation is normalized and with absolute value, we ignore the real constants and use $\sim$ to indicate proportional quantities.

Let's define:

$$G_{\sigma_x,\sigma_y}(x,y) = \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right) \qquad \text{2D Gaussian function,}$$

$$\frac{\partial G_{\sigma_x,\sigma_y}}{\partial x}(x,y) = -\frac{x}{\sigma_x^2} G_{\sigma_x,\sigma_y}(x,y) \qquad \text{Canny's kernel,}$$

$$s_\theta(x,y) = \sin\left(\omega\left(x\cos(\theta) + y\sin(\theta)\right)\right) \qquad \text{tested sinusoidal signal.}$$

Matching the signal wavelength and the filter pseudo-wavelength implies $\sigma_x = \frac{\omega}{4}$. To study the filter extension, we set

$$\sigma_y = \alpha\sigma_x$$

and focus on the influence of α. With $\mathcal{F}(\cdot)$ the Fourier transform, the formula derives from:

$$F^{\mathrm{C}}_{(0,0)}(\theta) = \left|\mathcal{F}^{-1}\left(\mathcal{F}\left(\frac{\partial G_{\sigma_x,\sigma_y}}{\partial x}\right)\mathcal{F}\left(s_{-\theta}\right)\right)(0,0)\right|$$

For (u,v) the Fourier coordinates, $\delta(\cdot)$ Dirac's delta function, $\mathbf{k_0} = (\frac{\omega}{2\pi}\cos(\theta), -\frac{\omega}{2\pi}\sin(\theta))$, $\mathbf{k} = (u,v)$ and $i^2 = -1$, the classical formulæ give:

$$\mathcal{F}\left(\frac{\partial G_{\sigma_x,\sigma_y}}{\partial x}\right)(u,v) \quad \sim \quad iu\mathcal{F}\left(G_{\sigma_x,\sigma_y}\right) \quad \sim \quad iuG_{\sigma_x^{-1},\sigma_y^{-1}}(u,v)$$

$$\mathcal{F}\left(s_{-\theta}\right) \quad \sim \quad i\delta\left(\mathbf{k}+\mathbf{k_0}\right) - i\delta\left(\mathbf{k}-\mathbf{k_0}\right)$$

Multiplying both and using $G_{\sigma_x,\sigma_y}(-\mathbf{k}) = G_{\sigma_x,\sigma_y}(\mathbf{k})$ we get a term proportional to:

$$\cos(\theta)\; G_{\sigma_x^{-1},\sigma_y^{-1}}(\mathbf{k_0})\left(\delta\big(\mathbf{k}+\mathbf{k_0}\big)+\delta(\mathbf{k}-\mathbf{k_0})\right)$$

Since $\mathcal{F}^{-1}\left(\delta(\mathbf{k}+\mathbf{k_0})+\delta(\mathbf{k}-\mathbf{k_0})\right) \sim \cos(\omega(x\cos(\theta)+y\sin(\theta)))$ that equals 1 at the origin (0,0), we have: $F^{C}_{(0,0)}(\theta) \sim |\cos(\theta)G_{\sigma_x^{-1},\sigma_y^{-1}}(\mathbf{k_0})|$. With $\beta = \omega^2\sigma_x^2/8\pi^2$ and then simplifying the cosine:

$$F^{C}_{(0,0)}(\theta) \quad \sim \quad \left|\cos(\theta)e^{\beta\left(\cos^2(\theta)+\alpha^2\sin^2(\theta)\right)}\right| \quad = \quad \left|\cos(\theta)e^{\beta(1-\alpha^2)\sin^2(\theta)}\right|$$

$\square$

Figure 5.15 on page 145 is a plot of $F^{C}_{(0,0)}$ for $\alpha \in \{1,\cdots,10\}$ and $\beta = 1$.

B.2 More figures on the orientation measure

We here present a more detailed evaluation of our orientation measure technique. We use the reference image presented in Figure 5.18 on page 148 to compare four methods:

- The Sobel filter.
- The Canny filter in its classical form (the gradient is estimated from the x and y first derivatives of an isotropic Gaussian, the pseudo-wavelength is set to the image wavelength.
- Our *unenhanced* variance-based selection.
- The selection *enhanced* with bilateral filtering (without the Γ color term that is not available).

Three values are computed relatively to the reference:

- The mean error e (in degrees).
- The proportion of *perfect* results $\%_{\text{perfect}}$: The selected θ sample is the nearest available.
- The proportion of *good* results $\%_{\text{good}}$: The selected sample is the nearest available or one of its neighbors.

We also compute their weighted counterparts (with a ^{w} superscript) that accounts for the variance according to $\exp(-V/V_{\text{mean}})$ similarly to the bilateral filter (we use the average variance V_{mean} over the image to compute the weight). These values have no practical meaning, they only give a cue about the pertinence of weights *e.g.* if the weighted error is lower than the normal one, it shows that the

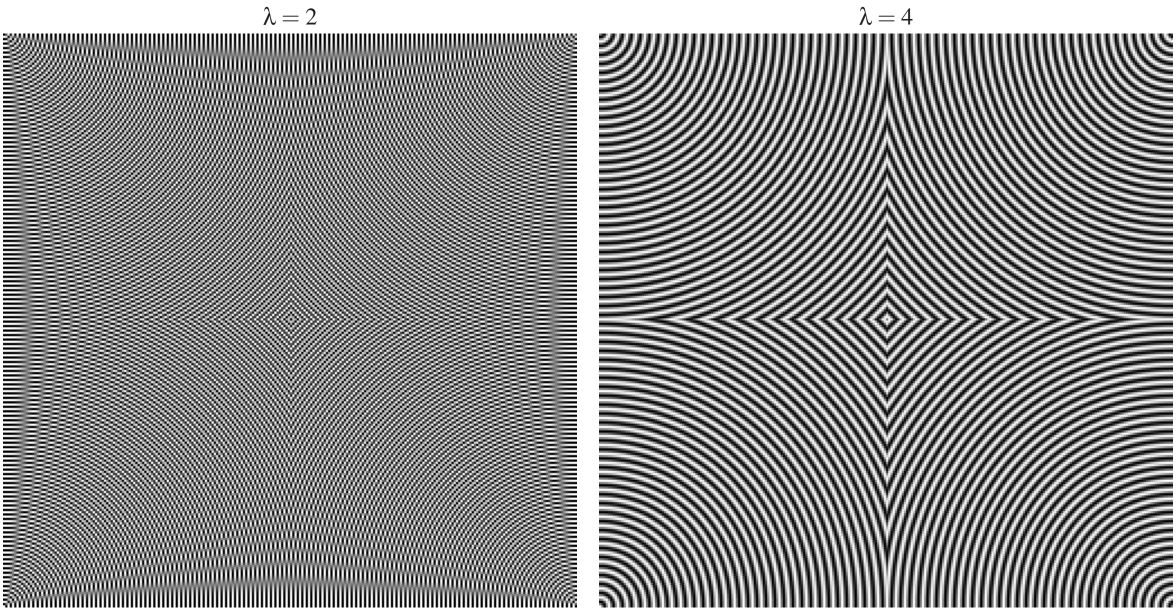


Fig. B.1: Reference images used in our experiment weights correctly distinguish the accurate measures from the others. These weighted values are not available for Sobel and Canny filters since they do not rely on a response curve.

We compute these figures for two different images: an extremely aliased version with $\lambda = 2$ (the one used in Figure 5.18 on page 148) and an aliasing-free version with $\lambda = 4$ (see Figure B.1 on the page before). For comparison, we give the expected values of random measure using a uniform distribution on $[0, \pi[$ and the values of a perfect pick among the 64 possible θ values assuming a uniform distribution of the reference orientations.

$\lambda = 2$	e	$\%_{\text{perfect}}$	$\%_{\text{good}}$	e^{w}	$\%^{\text{w}}_{\text{perfect}}$	$\%^{\text{w}}_{\text{good}}$
random	45°	1.6	4.7	-	-	-
perfect	0.70°	100	100	-	-	-
Sobel	17°	3.6	11	-	-	-
isotropic Canny	43°	2.7	5.9	-	-	-
unenhanced	2.9°	61	79	1.9°	49	91
enhanced	2.3°	49	86	1.9°	38	98

$\lambda = 4$	e	$\%_{\text{perfect}}$	$\%_{\text{good}}$	e^{w}	$\%^{\text{w}}_{\text{perfect}}$	$\%^{\text{w}}_{\text{good}}$
random	45°	1.6	4.7	-	-	-
perfect	0.70°	100	100	-	-	-
Sobel	2.7°	30	97	-	-	-
isotropic Canny	1.6°	91	97	-	-	-
unenhanced	1.9°	93	97	0.69°	≈ 100	≈ 100
enhanced	0.80°	93	96	0.053°	≈ 100	≈ 100

Table B.1: Detailed evaluation of our orientation measure

Table B.1 summarizes our results. We can observe that, on the aliased image, the Sobel filter gives erroneous orientations (too curved, see Figure 5.18 on page 148) and the isotropic Canny filter clearly fails because it is unadapted for such short wavelengths (it is foiled by the aliasing patterns). Our evaluation reaches nonetheless accurate results. A surprising result: The number of "perfect" results decreases after bilateral filtering and using the weighted figures. This indicates that our reliability estimation is also impaired by the extreme aliasing. But, compared to the other tested methods, the measure is robust enough to yield acceptable values.

On the aliasing-free image, all the tested techniques perform better. Sobel filter lacks accuracy compared to the others. An interesting point is that the isotropic Canny filter yields more precise results that our technique without bilateral filtering. This shows that it would be interesting to add an isotropic Canny filter in our filter list when the analyzed images are not aliased.

The other remarkable point is the high accuracy of the enhanced measures. This is confirmed by the weighted values that illustrate the relevance the weights. The committed error (0.80°) has to be compared to the error obtained from an exact selection among the θ samples (0.70°). This shows that our measure is almost optimal. The last point is the unexpected degradation of $\%_{\text{good}}$ after the bilateral filtering. Its limited amplitude let us think that it is due to numerical problems.

B.3 Geometric registration of the viewpoints

We here present in more details how the viewpoints can be registered with the help of the mirror balls and of the fitted ellipses.

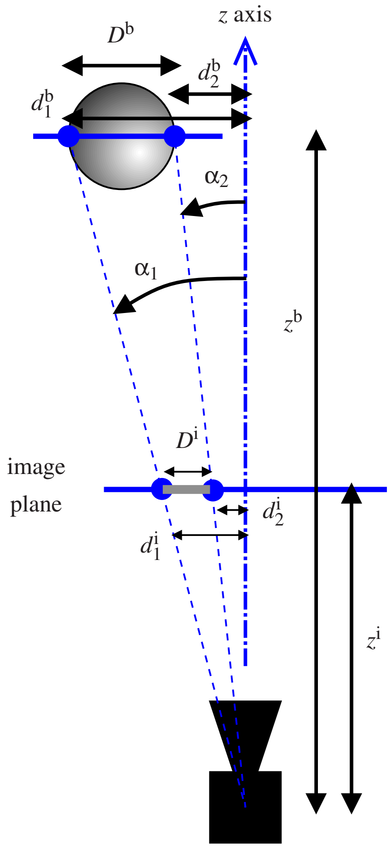


Fig. B.2: Geometric configuration of a camera and a mirror ball. *We have measured D^b. The size of a CCD cell and z^i are given by the camera manufacturer. The user indicates d_1^i and d_2^i.*

B.3.1 Single camera registration

We first expose how a camera is located relatively to the mirror balls. Figure B.2 illustrates the geometric setup. We make the approximation that the contour of the mirror ball is its intersection with the plane containing its center and parallel to the image plane. This approximation simplifies the study and is minor since the view angle is low in our configuration.

We have measured the ball diameter D^b with an electronic device up to 0.1mm. The camera manufacturer provides the focal length z^i and the dimension of a CCD cell with a precision between 1% and 0.1%. The user provides the image boundaries of the ball d_1^i and d_2^i with a pixel accuracy (these values are converted in length unit using the CCD size). We can estimate that the input data are known with an error in order of 1%.

Hence from classical geometry formulæ:

$$\tan(\alpha_1) = \frac{d_1^i}{z^i} = \frac{d_1^b}{z^b} \tag{B.1a}$$

$$\tan(\alpha_2) = \frac{d_2^i}{z^i} = \frac{d_2^b}{z^b} \tag{B.1b}$$

leading to:

$$z^b = \frac{D^b}{\tan(\alpha_1) - \tan(\alpha_2)} \tag{B.2}$$

Using Equations (B.1) and (B.2) for both the x and y axes locate the camera relatively to the mirror ball. Since we have two spherical mirrors in our setup. We average their results to get a global registration.

Then we assume that the depth of the ellipsoid skull is in between both mirrors (this is precise up to a few centimeters and is later refined by the user). This gives the z coordinate of its center. The x and y coordinates are computed with formulæ similar to Equation (B.1).

D^b	67.2mm
CCD size	6.25μm
z^i	12.5mm

Table B.2: Characteristic lengths of our setup

Table B.2 gives the characteristic values of our setup.

B.3.2 Global registration

From the previous step, each camera is located relatively to the skull center. Then, the user indicates the position of the view among {*left*, *right*, *top*, *back*}. This defines the axis of the skull in the current coordinate system.

From these data, we can directly register all the views in the skull coordinate system: the skull axes of each view are aligned and the skull centers are superposed (see Figure B.4).

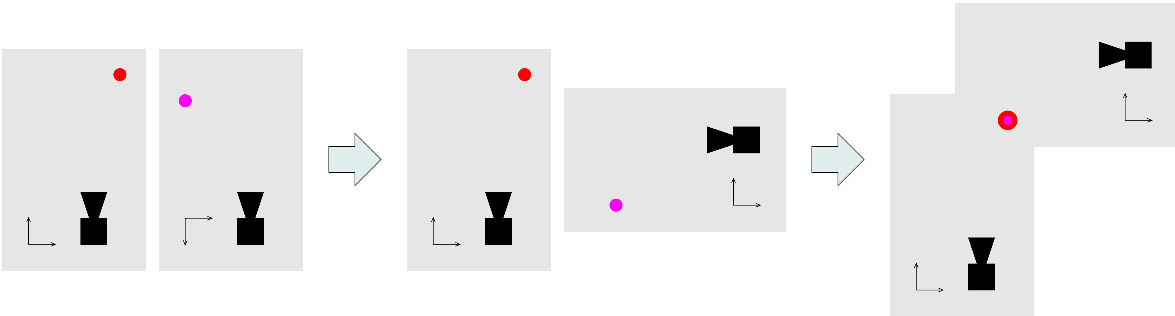


Fig. B.4: Global registration of the viewpoints

First, the axes are aligned. Then, the skull centers are superposed. This is a classical registration in a common coordinate system, considering the skull coordinate system. (The color dots represent the skull centers.)

Then, the user fits an ellipse for each viewpoint to match closely the hair volume seen in the images (Figure B.3). With a formula equivalent to Equations (B.1), this characterizes the length of two of the three axes of the 3D ellipsoid. Averaging the data from the four viewpoints gives the final geometry of the ellipsoid.

This process has a precision in the order of few centimeters which is borderline for our purpose. The geometry actually used in our experiment is manually tweaked to achieve a below-centimeter accuracy.

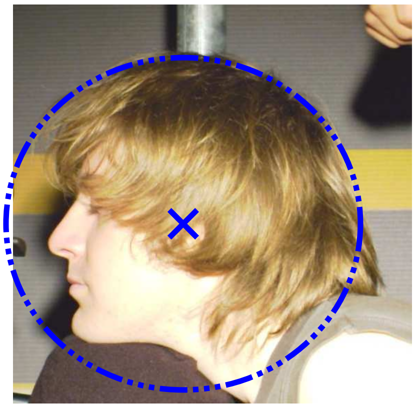


Fig. B.3: Ellipse fitted on the hair area.

C
Résumé français

Nous donnons ici un résumé français de ce manuscrit. L'essentiel des idées développées dans cette thèse est repris dans ces quelques pages. Toutefois, il ne s'agit en aucun cas d'une traduction intégrale (sauf pour l'introduction et la conclusion) et nous conseillons vivement au lecteur averti de se reporter à la version anglaise pour plus détails.

C.1 Introduction

Le sujet de cette thèse porte sur la création des données utilisées en informatique graphique et plus particulièrement pour synthétiser des images. Ces données peuvent être regroupées grossièrement en trois catégories : la forme des objets, leur apparence et leur environnement lumineux. Nous nous sommes principalement intéressés à la création de la forme et de l'apparence bien que l'on soit amené régulièrement à travailler avec la lumière. Cette thèse propose plusieurs méthodes pour générer ces données à partir d'images réelles : on ne demande pas à l'utilisateur de directement modéliser l'objet souhaité mais plutôt d'en fournir une ou plusieurs photographies. Ces images sont alors automatiquement analysées par l'ordinateur qui en extrait l'information recherchée.

On attend de cette approche des données plus fidèles à l'original et un temps de création plus court. Observons plus en détails un exemple concret.

Une tâche classique en informatique graphique est la création des données nécessaires à la synthèse d'une image (d'une théière par exemple). L'approche standard est de compter sur les capacités de l'utilisateur : la forme de la théière est modelée dans un logiciel dédié et l'aspect de porcelaine est défini dans une boîte de dialogue qui laisse l'utilisateur choisir les caractéristiques du matériau (couleur, brillance, motif, etc). Ce processus de création est particulièrement long car l'utilisateur doit tout créer. Qui plus est, la «qualité» du résultat dépend l'habileté du créateur. Une approche plus rapide est de choisir ces données dans une bibliothèque de formes et de matériaux déjà créés. Cela facilite la tâche courante mais ne résout pas le problème de la création des données de la bibliothèque.

Une fois que ces données ont été produites, il existe de nombreuses méthodes pour synthétiser l'image de la théière. Toutefois, cette synthèse dépasse le cadre de ce manuscrit.

Modéliser un objet existant Mais observons maintenant un cas plus délicat. L'utilisateur ne veut plus modéliser *une* théière mais *la* théière qu'il utilise tous les jours. Une approche directe est de commencer par créer une théière comme précédemment puis de l'adapter pour qu'elle ressemble à l'original. Atteindre une ressemblance approximative est faisable. Mais imaginons que cette théière soit décorée avec des motifs gravés et/ou peints avec de la dorure. Dans cette hypothèse, faire une reproduction précise n'est plus possible pour l'utilisateur. Il faut employer des techniques plus sophistiquées qui ne reposent pas sur le seul utilisateur.

Pour capturer la forme d'un objet, on peut utiliser un scanner 3D : un appareil avec un laser ou un faisceau de lumière modulée qui mesure la forme 3D d'un objet. L'acquisition d'un matériau peut se faire à l'aide d'un gonioréflectomètre [221] : le comportement lumineux du matériau est mesuré pour toutes les positions possibles de la lumière et du point de vue. Ces deux techniques sont précises et nécessitent des équipement spéciaux ce qui est souvent un frein à leur mise en œuvre. De plus, plusieurs situations peuvent perturber les mesures; par exemple, les scanners 3D échouent avec les matériaux translucides ou dispersifs (le verre, la fourrure, les cheveux, etc).

Notre proposition Dans ce manuscrit, nous présentons une alternative utilisant des images. Pour le problème de la théière, nous proposons d'utiliser une ou plusieurs images de la théière originale pour en extraire les données nécessaires à sa reproduction. Ensuite, une analyse appropriée est menée pour produire l'information utile. Par rapport à une approche utilisateur, on attend plusieurs améliorations :

Moins de temps utilisateur: De façon immédiate, un algorithme à base d'images demande moins de temps à l'utilisateur. Le temps total peut être plus long à cause des calculs mais ce temps machine laisse l'utilisateur libre pour d'autres tâches.

Caractérisation objective des données: Si on demande à plusieurs personnes de définir la brillance d'une même théière, les réponses seront très probablement différentes. Ces «valeurs» proviennent d'estimations subjectives et il serait difficile de choisir la bonne. À l'opposé, un algorithme a un critère déterministe dont la précision peut être étudiée. Autre avantage, cette mesure est reproductible.

Plus de détails: En reproduisant un objet réel, un utilisateur peut oublier certains détails là où une reconstruction automatique fait un parcours systématique des images.

Malgré cela, notre objectif n'est pas de remplacer la création «humaine». Nous proposons une manière complémentaire de produire des données 3D qui est compatible avec la création «à la main» ou l'exploitation de la bibliothèque : la forme obtenue à partir des images peut être éditée par la suite et/ou stocker dans une bibliothèque.

Robustesse Nous sommes convaincus que les perturbations sont inhérentes à la capture : les images sont inévitablement corrompues par du bruit, du flou, de la distortion, etc. La question est comment gérer ce fait. Il y a deux réponses extrêmes. D'un côté, on peut laisser l'utilisateur prendre des photos sans contrainte, ce qui implique un effort particulier lors de la phase d'analyse. L'algorithme sous-jacent doit être robuste pour être capable d'extraire des données acceptables. En contrepartie, le processus d'acquisition est simple et accessible à un non spécialiste. D'un autre côté, les conditions de capture peuvent être totalement contraintes (salle noire, système optique de qualité et calibré avec précision, bras robotique, etc). Il faut alors un spécialiste pour mener à bien la capture mais on peut alors considérer les données acquises comme «parfaites». Cela permet alors à l'algorithme d'ignorer les problèmes de robustesse et de se concentrer sur la précision. Des chercheurs ont décrit des solutions intermédiaires qui imposent quelques contraintes à l'utilisateur en échange de plus précision. Cela définit un continuum entre la facilité de capture et la précision.

Dans ce manuscrit, on a délibérément choisi une approche plus proche de la facilité de mise en œuvre, sans pour autant négliger la précision. Cela implique simplement que l'on s'efforce de mettre au point des algorithmes robustes : on accepte de travailler sur des données perturbées pour alléger la charge du côté utilisateur. Cela sous-entend aussi que la précision obtenue peut ne pas être toujours comparable aux techniques en environnement totalement contrôlé. On obtient néanmoins les avantages suivants :

Un système d'acquisition moins encombrant: Les données initiales proviennent typiquement d'un appareil photo numérique ou d'une caméra numérique dont on peut aujourd'hui trouver couramment des modèles de qualité et de petite taille. Si l'on a besoin d'une salle noire, on s'efforcera d'être suffisamment robuste pour pouvoir travailler dans une salle normale dont on a simplement éteint la lumière. Nous n'imposerons pas de salle au murs noirs mats ou un écran bleu.

Un système plus flexible: À partir des mêmes images, on peut utiliser différentes techniques d'analyse en fonction de leur contenu alors qu'avec un appareil dédié, il faut tout changer s'il ne correspond pas.

Une meilleure correspondance avec les images initiales: Dans certains cas, on peut travailler avec un arrière-plan quelconque. Cela rend possible de travailler avec les objets dans leur contexte. Si l'objectif est de modifier les images initiales (changer l'apparence d'un objet ou un insérer un objet supplémentaire par exemple), les données extraites directement de ces images sont plus à même d'être cohérentes que celles obtenus avec une technique qui les isole de leur contexte (ce qui peut introduire des erreurs d'alignement par exemple).

Un meilleur choix des détails extraits: L'acquisition à partir d'images capturent les détails visuellement importants là où d'autres techniques (comme les scanners lasers) risquent de récupérer des détails trop petits et d'en manquer d'autres de petites tailles mais avec un fort impact visuel.

Notre approche : étude de cas Se pose alors une question cruciale : Quel est notre but? Quel type d'information veut-on obtenir? Nous sommes convaincus que le cas général n'est pas traitable : acquérir de manière robuste la géométrie et l'apparence d'un objet sans aucun a priori est impossible. Nous pensons qu'une approche plus raisonnable est de se concentrer sur un scénario représentatif pour lequel on peut s'appuyer sur des caractéristiques connus. On peut imaginer alors développer plusieurs scénarii et laisser l'utilisateur choisir l'algorithme approprié. Le piège serait alors de multiplier le nombre de scénarii à l'infini mais tant que l'on étudie des scénarii avec un intérêt suffisant, nous sommes convaincus que cette approche est valide et efficace.

Nous avons choisi ce chemin dans ce manuscrit. Nous identifions quelques cas intéressants qui mènent vers des applications utiles and nous concentrons sur ceux-ci. Nous travaillons sur trois points principaux : tout d'abord, nous présentons dans le chapitre 2 une méthode pour reconstruire la surface d'un objet matte à partir d'une séquence d'images dont le point de vue se déplace. Cette technique est étendue à un ensemble plus général de points de vue dans le chapitre 3. Cette méthode est conçue pur être générique *i.e.* nous évitons de formuler des hypothèses trop spécifiques pour rester aussi général que possible.

Nous montrons ensuite dans le chapitre 4 comment on peut capturer l'apparence d'un visage à partir d'une seule image et comment les données ainsi récupérées peuvent être utilisées pour synthétiser le visage original sous un nouvel éclairage. Nous terminons ce document avec le chapitre 5 qui expose une technique pour la capture de la géométrie d'une chevelure à partir de plusieurs images prises avec une caméra fixe et avec une lumière qui se déplace. Une grande précision est nécessaire pour que la ressemblance soit suffisante pour reconnaître la personne originale. Nous développons donc un outils spécifique pour atteindre une fidélité plus importante qu'un utilisateur ou une méthode générique. Une conclusion générale est donné dans le dernier chapitre.

C.2 Reconstruction de surface

C.2.1 Introduction

Il est aujourd'hui possible d'afficher des modèles 3D de taille de plus en plus importante. Les progrès récents des cartes graphiques et des méthodes de gestion de la complexité rendent possible la manipulation de modèles gigantesques. Ainsi par exemple dans les jeux vidéos actuels, plus personne ne s'étonne de pouvoir se déplacer dans une ville. Néanmoins, la question de créer des modèles de cette taille se pose toujours. Même s'il est toujours possible de faire appel à un ou plusieurs «artistes» qui réaliseront cette tâche «à la main», cette approche nécessite de plus en plus de temps en temps et semblent de moins en moins raisonnable. Quelques méthodes sont apparues pour créer des bâtiments automatiquement mais peu de solutions convaincantes existent pour les objets de taille réduite (boîte aux lettres, bancs, statues, etc). Cette remarque nous a été notre motivation initiale pour étudier le problème de la reconstruction de la surface d'un objet à partir d'une séquence d'images. L'objectif est de mettre au point un processus automatique qui crée un modèle tridimensionnel d'un objet simplement à partir de quelques photographies.

Cette technique vient en complément de l'approche «à la main». De nombreux modèles pourraient être acquis avec une intervention minimum de l'utilisateur, libre à ce dernier de retoucher les modèles ainsi obtenus par la suite.

C.2.2 État de l'art

Initialement de nombreux travaux ont été menés pour imiter la vision humaine. Nos deux yeux associés à notre cerveau sont la preuve que deux images d'une même scène sont suffisantes pour en retrouver un modèle tridimensionnel.

Les problèmes à deux images (potentiellement étendus à trois) ont été très largement étudiés et nous ne nous attarderons pas sur leur description. Les principaux résultats qu'il faut néanmoins en retenir sont :

- La reconstruction consiste essentiellement à être capable de «reconnaître» le même point de l'espace dans plusieurs images. Une fois plusieurs points 2D associés comme étant les projections d'un même point 3D, reconstruire ce point n'est qu'une classique triangulation.
- Le problème de la reconstruction à partir d'images est intrinsèquement mal posé *i.e.* il existe plusieurs scènes 3D qui produisent exactement les mêmes images.

Avec plusieurs images, on espère améliorer les points suivants par rapport au cas à deux images : une plus grande robustesse face aux possibles imperfections des images (bruit, flou, etc), la possibilité de travailler malgré des occultations partielles (quand un objet est caché par un autre seulement dans quelques images) et une plus grande précision grâce à la redondance de l'information disponible.

Pour résoudre ce problème mal posé, la plupart des méthodes existantes introduisent une connaissance a priori sur la scène observée, généralement l'hypothèse que les objets sont (plus ou moins) lisses. Cette hypothèse est introduite formellement sous la forme d'une *fonctionnelle* (une expression mathématique) qui associe une composante qui, pour une reconstruction 3D donnée, évalue la cohérence ce modèle avec les images et qui évalue aussi à quel point l'hypothèse a priori est respectée. Intuitivement, la fonctionnelle «note» une reconstruction, l'objectif étant d'avoir la reconstruction avec la «meilleure note possible». D'un point de vue mathématique, il s'agit là d'un problème d'optimisation.

Les méthodes suivantes sont regroupées selon la forme de leur fonctionnelle et selon la méthode d'optimisation utilisée.

Pas de fonctionnelle, enveloppe visuelle Ces méthodes se basent la connaissance des silhouettes de l'objet qui sont obtenues à l'aide de méthode type «écran bleu» ou soustraction d'arrière-plan. Ces méthodes ont le désavantage de nécessiter cette connaissance supplémentaire et manquent aussi de détails si peu de points de vue sont disponibles. Néanmoins, elles sont très rapides et robustes aux effets comme les reflets ou la transparence.

Pas de fonctionnelle, sculpture À partir d'un volume englobant l'objet, l'algorithme supprime (sculpte) petit à petit les parties incohérentes avec les images. Ces techniques nécessitent aussi de nombreux points de vue pour être précises et sont peu robustes aux imperfections des images. Elles sont toutefois simples à implémenter et à mette en œuvre.

Fonctionnelle par ligne, programmation dynamique L'objet est reconstruit tranche par tranche. Comme dans chaque tranche, on cherche une ligne 1D et non plus une surface 2D, on peut utiliser la technique de programmation dynamique qui permet de résoudre exactement une fonctionnelle imposant une certaine régularité. Néanmoins cette régularité n'est imposée que ligne par ligne et un post-traitement est nécessaire pour obtenir une régularité entre lignes. Ces méthodes sont aujourd'hui délaissées au profit des méthodes suivantes plus récentes qui évitent cette séparation en lignes et dont les résultats sont meilleurs.

Fonctionnelle intrinsèque, niveaux Grâce à la méthode des niveaux, une surface est représentée implicitement par une fonction potentielle dont elle est un niveau. Cette technique permet de définir une fonctionnelle *intrinsèque* (qui ne dépend pas de la paramétrisation de la surface) et gérer simplement la topologie de la surface. Toutefois, cette technique impose que la surface soit deux fois différentiables partout ce qui produit des résultats trop lisses (sans coin ni arête). Par ailleurs, la convergence du processus d'optimisation n'est pas garantie en théorie même si elle est observée en pratique.

Fonctionnelle paramétrique, coupure de graphe La méthode de coupure de graphe permet de produire des résultats très précis au niveau des silhouettes des objets. Cette approche contrôle la convergence de l'optimisation, voire pour certaines méthodes, en garantit la convergence exacte. Néanmoins la fonctionnelle proposée est paramétrique (dépend de la paramétrisation de la surface) et surtout, ces techniques sont appliquées à des cartes de disparité qui sont une représentation «images» des objets 3D. Pour la plupart, ces résultats manquent aussi de précision pour la profondeur.

Fonctionnelle intrinsèque, coupure de graphe Récemment une méthode est apparue pour manipuler une fonctionnelle intrinsèque avec une coupure de graphe. Cette approche semblent prometteuse mais est principalement appliquée au problème de la segmentation de données. Son utilisation dans le cadre de la reconstruction 3D n'est pas trivial et l'étude en serait intéressante.

Méthodes hybrides Des approches mélangeant plusieurs des techniques précédentes existent et apparaissent comme une voie naturelle pour tirer profit des qualités de plusieurs techniques. Des résultats confirmant cette intuition ont été obtenus et cette voie mérite d'être explorée plus encore qu'elle ne l'est aujourd'hui.

Au vu de cet état de l'art, nous avons choisi une approche basée sur une coupure de graphe afin de contrôler la convergence de notre optimisation. Néanmoins nous efforcerons de poser notre problème dans un cadre directement 3D et de nous affranchir de la notion de carte de disparité qui est peu appropriée à notre objectif à cause de sa formulation dans l'espace image.

C.2.3 Définition du problème et formulation de la fonctionnelle

Pour étudier le problème de la reconstruction à partir d'images, on représente la surface recherchée par un champs de profondeur standard : $(u,v) \mapsto \mathbf{x}(u,v) \equiv (u,v,f(u,v))$; si plusieurs valeurs sont nécessaires pour z, on introduit plusieurs fonctions $f_1, f_2, \ldots$. Le problème que l'on se pose alors est de formuler une fonctionnelle qui représente sous une forme analytique un compromis entre la fidélité aux images initiales et l'hypothèse a priori que la surface des objets est lisse.

Pour cela, nous nous appuyons sur une fonction de cohérence $c(\cdot) \geq 0$ dont les valeurs sont d'autant plus faibles que le point 3D considéré est cohérent avec les images fournies. Plusieurs choix sont possibles pour cette fonction (voir le manuscrit détaillé, section 2.2 en page 9). Nous proposons ici la fonctionnelle suivante :

$$\iint \left(c(\mathbf{x}) + \alpha_u(u,v) \left| \frac{\partial z_{\mathbf{x}}}{\partial u} \right| + \alpha_v(u,v) \left| \frac{\partial z_{\mathbf{x}}}{\partial v} \right| \right) du\, dv$$

Cette formulation n'implique que des dérivées du premier ordre. Ce choix est motivé par deux points. Premièrement, la technique de minimisation proposée par la suite tire profit de cet aspect. Secondement, les dérivées d'ordre supérieur rendent le problème d'optimisation d'autant plus difficile.

Il est important de remarquer que cette fonctionnelle n'est pas intrinsèque : elle dépend de la paramétrisation de la surface. Cela implique par exemple que la formulation du problème dépend de l'orientation des axes xyz pourtant arbitraire.

Cette fonctionnelle combine l'avantage d'être formulée directement dans l'espace géométrique propre au problème posé et celui de pouvoir être résolue par une technique de coupure de graphe. Cette technique d'optimisation est détaillée dans les sections suivantes et permet d'obtenir un minimum global de la fonctionnelle contrairement au minimum local atteint par les approches à base de courbes de niveau ou certaines méthodes de coupures de graphe.

Par ailleurs, les fonctions α_u et α_v permettent un contrôle local de la régularisation. En leur assignant une valeur nulle, on peut totalement supprimer cette régularisation et ainsi autoriser des discontinuités dans la surface.

C.2.4 Présentation générale des coupures de graphe

Intuition Le problème de la coupure de graphe peut être vu à travers une analogie avec un réseau de tuyaux. Étant donné un réseau de tuyaux relié à une source et un puits, on se pose la question de savoir quelle est le débit d'eau maximum qui peut passer à travers le réseau, de la source vers le puits. Dans ce problème, les tuyaux sont donnés, ainsi que le débit maximum qui peut les traverser; l'inconnue est le débit réel qui traverse chaque tuyau.

Intuitivement, on peut se convaincre que trouver ce débit maximum est lié à trouver le goulot d'étranglement du réseau, c'est-à-dire l'ensemble de tuyaux qui sépare la source du puits et qui limite le plus le passage de l'eau.

Formalisation Formellement, ce problème se décrit à l'aide d'un graphe orienté. La source et le puits sont deux nœuds particuliers de ce graphe. Chaque arc du graphe est associé à une valeur de *capacité* qui représente le débit maximum qui peut le traverser. Le problème posé est alors défini par la donnée du graphe et de la fonction de capacité associée à ses arcs. L'objectif est de trouver une fonction de *flot* maximale. Une fonction de *flot* associe à chaque arc une valeur au plus égale à sa capacité. Le fait qu'elle soit maximale signifie qu'elle permet à un maximum d'eau de traverser le graphe – c'est-à-dire de quitter la source ou de façon équivalent d'atteindre le puits.

On définit ensuite une coupure comme un ensemble d'arcs qui partitionne le graphe en deux parties, l'une contenant la source et l'autre le puits. Les notions de flot et de capacité s'étendent à une coupure en sommant les valeurs des arcs coupés par cette partition. On a alors un théorème fort qui montre que le flot maximum à travers un graphe est égale au minimum de la capacité sur l'ensemble des coupures.

Intuitivement, une coupure est un goulot d'étranglement potentiel (l'eau qui va de la source au puits doit nécessairement traverser au moins un arc de la coupure). Le goulot d'étranglement effectif est la coupure de capacité minimale, celle qui permet au moins d'eau de la traverser.

Théorème flot maximum - coupure minimale Ce théorème montre que résoudre le problème du flot maximum dans un graphe revient à en trouver la coupure minimale. Ce résultat présente le problème du flot maximal comme un problème de minimisation. Ceci est d'autant plus intéressant qu'il existe des algorithmes pour résoudre de façon exacte ce problème en temps polynomial.

Tout problème d'optimisation qui peut être formulé comme une recherche de coupure minimale dans un graphe peut donc être résolu de manière exacte en temps polynomial. La version détaillée présente deux exemples simples d'application de cette méthode.

C.2.5 Solution discrète globale

Pour la suite de l'étude, on sépare la fonctionnelle en deux termes : $\mathcal{C}$ qui évalue la cohérence de la surface par rapport aux images et $\mathcal{S}$ qui en évalue la régularité.

$$\mathcal{C}(f) = \iint_{\mathcal{D}} c(x,y,f(x,y))dxdy$$

$$\mathcal{S}(f) = \iint_{\mathcal{D}} \left(\alpha_x(x,y) \left| \frac{\partial f}{\partial x}(x,y) \right| + \alpha_y(x,y) \left| \frac{\partial f}{\partial y}(x,y) \right| \right) dxdy$$

Discrétisation Le premier important pour mettre au point une technique à base de graphe pour minimiser la fonctionnelle choisie est de discrétiser cette fonctionnelle. En effet, par nature, un graphe est une structure composée d'entités discrètes (les nœuds et les arcs). On obtient ainsi les formules suivantes :

$$\mathcal{C}^{\mathrm{d}}(f) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} c(x_i, y_j, f(x_i, y_j)) \Delta x \Delta y$$

$$\mathcal{S}^{\mathrm{d}}(f) = \sum_{i=1}^{n_x-1} \sum_{j=1}^{n_y} \alpha_x(x_i, y_j) \left| f(x_{i+1}, y_j) - f(x_i, y_j) \right| \Delta y \; + \; \sum_{i=1}^{n_x} \sum_{j=1}^{n_y-1} \alpha_y(x_i, y_j) \left| f(x_i, y_{j+1}) - f(x_i, y_j) \right| \Delta x$$

Premier graphe On propose dans un premier graphe qui permet de minimiser la fonctionnelle discrète. Ce graphe a plusieurs propriétés fortes :

- La capacité de ses arcs est définie en termes géométriques. Cela permet de décrire le graphe comme une entité 3D : les nœuds ont une position dans l'espace, les arcs une longueur, etc.
- Une surface correspond directement à une coupure grâce à cette correspondance géométrique : les arcs coupés par la surface forme une coupure du graphe. On montre en plus qu'une coupure minimale du graphe correspond nécessairement à une fonction du type $z_{\mathbf{x}} = f(x_{\mathbf{x}}, y_{\mathbf{x}})$.
- Pour finir, on démontre que la capacité d'une coupure est égale à la valeur de la fonctionnelle pour la surface correspondante. Cela assure que l'on peut minimiser exactement la fonctionnelle en calculant la coupure minimale du graphe proposé. On résout donc le problème discret en temps polynomial.

Analyse du terme de régularisation Les résultats obtenus par ce premier graphe sont satisfaisants mais présentent un aspect crénelé à cause de discontinuités parasites dans la surface reconstruite. Une étude montre que cet effet indésirable est dû à la linéarité du terme de régularisation. La solution à ce problème est un terme strictement convexe.

Second graphe On propose un second graphe qui supprime les discontinuités parasites. Pour cela, le graphe résout une fonctionnelle légèrement différente dont le terme de régularisation est strictement convexe et peut approcher arbitrairement près le terme linéaire définit dans l'intégrale.

Il est important de noter que la stricte convexité de la régularisation rend nécessaire la gestion des discontinuités par les fonctions α_u et α_v. Sans celle-ci, les discontinuités de la surface seraient trop pénalisées et aboutiraient à des résultats trop lisses.

C.2.6 Mise en œuvre

On applique le moteur d'optimisation précédemment décrit dans un scénario classique. On souhaite reconstruire la surface d'un ou plusieurs objets qui reposent dans un demi espace à partir d'une séquence d'images dont les points de vue sont dans l'autre demi espace.

L'algorithme proposé prend en entrée la séquence d'images calibrées et un volume englobant la partie de la scène que l'on souhaite reconstruire.

Initialisation On travaille dans un espace discrétisé en *voxels*. La grille utilisée est taillée de façon à assurer une aire constante à la projection des voxels dans les images. On propose une construction de la grille qui respecte cette propriété et l'on démontre qu'elle permet aussi de caractériser le contour des objets à reconstruire.

Boucle principale Le cœur de l'algorithme est itératif et enchaîne les étapes suivantes jusqu'à ce que tous les objets inclus dans le volume d'intérêt soient reconstruits.

1. On calcule la valeur de la fonction de cohérence c pour chaque voxel. On sélectionne les voxels les plus cohérent en comparant cette valeur à un seuil. Afin de rendre robuste l'ensemble de voxels ainsi obtenus (boucher les trous et supprimer les voxels isolés), on lui applique un traitement morphologique. Par la suite, on ne considère plus que l'ensemble de voxels le plus proche des caméras.

2. On détecte les discontinuités potentielles de la surface. Pour cela, on examine les changements de couleurs dans les images. À l'aide des discontinuités de couleurs dans les images, on peut définir les fonctions α_u et α_v.

3. La fonctionnelle est minimisée à l'aide d'une coupure de graphe. On montre comment adapter le graphe pour éviter les auto-occultations. En pratique, cela ne s'est jamais avéré utile pour nos tests.

4. La visibilité est mise à jour. Il s'agit simplement de marquer quels sont les pixels occultés dans les images.

Post-traitement Pour finir, on lisse les surfaces obtenues pour supprimer les «marches d'escalier» dues à la discrétisation de l'espace. En prend soin toutefois de préserver les discontinuités et points caractéristiques de la surface.

C.2.7 Résultats

Plusieurs résultats sont présentés. Un homme portant une mallette illustre la robuste au bruit et au faible écartement des caméras. Un clavier montre le haut niveau de précision que la méthode présentée peut atteindre. Un échiquier caché par un lampion démontre la capacité de notre technique à lever les occultations partielles.On a étudié l'impact du nombre d'images sur la qualité des résultats : pour une séquence d'images significativement bruitée, les résultats sont acceptables jusqu'à 10 images. On finit en comparant notre résultat avec une méthode reconnue à base de carte de disparité : nos résultats sont plus précis et répondent mieux à la question de la reconstruction tridimensionnelle.

C.2.8 Conclusion

On a présenté un système de reconstruction de surface à partir d'images qui met en jeu à la fois des techniques de type «sculpture de voxels» et une optimisation à base de coupure de graphe.

Cette approche soufre des limitations suivantes :

Ressources importantes: Les temps de calcul sont longs (plusieurs dizaines de minutes) et la mémoire requise est importante (plusieurs centaines de mégaoctets). Le chapitre suivante propose d'améliorer ce point.

Fonctionnelle non intrinsèque: Le problème défini par la fonctionnelle dépend de la paramétrisation de la surface alors que, idéalement, il devrait en être indépendant. L'annexe A.2 en page 170 propose quelques pistes pour améliorer ce point.

Paramétrisation de la surface: On a montré dans ce chapitre comment travailler avec des surfaces paramétrées par $z = f(x,y)$. On peut étendre cette approche à d'autres systèmes de coordonnées mais le cas reste problématique. Le chapitre suivant s'attaque à ce point.

Les contributions principales de cette méthode sont :

Technique d'optimisation: La méthode à base de graphe décrite atteint un minimum global de la fonctionnelle pour une discrétisation qui peut être choisie arbitrairement. Cette méthode ne requiert pas d'estimation initiale de la solution, ce qui résout directement le problème de la construction d'une telle estimation.

Fonctionnelle géométrique: Nous avons proposé une fonctionnelle qui décrit le problème posé indépendemment de la résolution de l'espace 3D et de l'espace image. Qui plus est, notre formulation prend explicitement en compte l'existence de discontinuité dans les objets.

Formes lisses: La méthode proposée reconstruit des formes lisses, sans discontinuité parasite, tout en étant capable de détecter et reconstruire les discontinuités réelles de la scène.

Aglorithme efficace: Nous avons mis au point un algorithme qui intègre de manière cohérente et unifiée les points précédemment mentionnés. Les résultats obtenus montrent un gain signification en précision et en robustesse par rapport aux méthodes existantes.

C.3 Reconstruction de patchwork

C.3.1 Introduction

La méthode présentée dans le chapitre précédent est limitée par les ressources nécessaires et la paramétrisation de la surface. On propose dans ce chapitre une méthode de reconstruction morceau par morceau. On appelle chaque morceau reconstruit un *patch*.

C.3.2 Motivation

Cette approche se justifie par deux aspects principaux. Tout d'abord, si l'on considère que le problème de la reconstruction est local, c'est-à-dire que les propriétés locales d'un objet ne dépendent pas du reste de l'objet, on montre que les complexités spatiales et temporelles du problème découpé en patchs sont inférieures à celles du problème «non découpé». Notamment la complexité spatiale (la mémoire requise) ne dépend plus de la taille de la scène traitée mais seulement de la taille des patchs, ce qui résout le problème des ressources en mémoire.

Ensuite, les patchs permettent de définir localement le système de coordonnées et la paramétrisation de la surface et ainsi offrent la flexibilité nécessaire à la description d'objets complexes. Cela ouvre aussi les portes a une approche de type multi-résolution qui adapte le niveau de détails de la reconstruction aux caractéristiques locales de l'objet.

C.3.3 Mise en œuvre à l'aide de coupures de graphe et d'un champ de distance

On propose d'appliquer l'idée de la reconstruction par patchs à l'aide de la technique de coupure de graphe du chapitre précédent, d'un champ de distance et d'une stratégie d'ordonnancement.

Coupure de graphe Chaque patch est déterminé par une coupure de graphe. On garantit ainsi que chacun est optimal dans le sens où il réalise un minimum global de la fonctionnelle sur le problème réduit à son domaine local. On propose aussi plusieurs amélioration de la technique initiale.

Contraintes: On montre comment modifier le graphe de manière à imposer que certains points fassent partie de la surface résultat. Cela permet de prendre en compte les patchs déjà reconstruits et d'assurer la continuité aux bordures.

Autres informations: La méthode initiale ne prend en compte que des informations de cohérence basées sur la couleur dans les images. On propose ici d'étendre la fonctionnelle pour ajouter à ce critère des informations de contour et des points 3D fiables (reconstruits avec un scanner 3D par exemple).

Champ de distance Nous avons choisi une structure de champ de distance pour agréger les patchs dans une structure commune. Concrètement, il s'agit d'une grille 3D où chaque nœud stocke sa distance au point le plus proche de la surface. Il est aisé de reconstruire la surface à partir de cette structure en utilisant l'algorithme du *Marching Cube*. Cette technique présente aussi les avantages d'être incrémentale, de pouvoir être adapté à la multi-résolution grâce à un *octree* et de faciliter les requêtes pour localiser des points dans la structure.

Ordonnancement L'ordre dans lequel on reconstruit les patchs est crucial puisque le processus de création d'un patch peut tirer partie des patchs déjà reconstruits et uniquement de ceux-ci. La stratégie à mettre en œuvre dépend des données disponibles; on peut toutefois dresser les grandes lignes d'une telle stratégie :

- Il faut traiter en premier les patchs contenant les données les plus fiables; par exemple les points 3D issus d'un scanner.
- La valeur de la fonctionnelle sur un patch indique la «qualité» d'un patch; elle renvoie à la fois à sa cohérence par rapport aux images et à sa régularité. On préférera traiter en priorité les patchs voisins des «bons» patchs. Si besoin on peut découpler ce critère pour ne considérer que la qualité d'un patch ou sa régularité (ou non singularité).
- Si les performances (temps de calcul) sont importantes, on donnera plus de priorité aux patchs qui créent une surface nouvelle importante.

C.3.4 Deux algorithmes

Propagation de points 3D On reconstruit la surface à partir d'un ensemble de points 3D fiables obtenus à partir d'un scanner 3D ou de la triangulation des points caractéristiques des images.

La surface est d'abord reconstruite à partir des régions denses en points 3D et ensuite les trous sont comblés. L'algorithme se base sur une liste de *graines* (des points 3D fiables) qui est initialisée avec les points fiables. Les points sont considérés un par un et «étendus» en un patch. À chaque fois, de nouvelles graines sont extraites en localisant les points les plus fiables sur le nouveau patch. Ces nouvelles graines sont ajoutées dans la liste et traitées par la suite. La reconstruction est finie quand la liste des graines est vide.

Cette approche se présente à la fois comme une technique de reconstruction de surface à base de points dans les régions denses car dans ce cas la surface est guidée par les points fournis en entrée, et à la fois comme une technique à base d'images car, en l'absence de points fiables, les patchs continuent à exploiter l'information des images.

Sculpture par patchs Cette méthode s'inspire de la technique de sculpture par voxels dont un des défauts majeurs est d'être trop locale. On propose ici d'utiliser de «gros» voxels. De manière classique, ces voxels sont considérés un par un et selon leur cohérence par rapport aux images, ils sont soit conservés soit supprimés. Notre apport à la méthode est le critère pour faire ce choix.

Pour décider de la cohérence d'un voxel, on essaie de construire un patch à l'intérieur du voxel. Si on trouve un patch suffisamment bon *i.e.* avec une fonctionnelle suffisamment basse, on conserve le voxel et on agrège le patch trouvé dans le champ de distance, sinon on le supprime.

Cette approche conserve les qualités de l'approche classique par sculpture tout en apportant une cohérence locale qui transforme le problème initiale qui est mal posé, en un problème bien posé.

C.3.5 Conclusion

Ces travaux sont encore en cours et de nombreux points restent à étudier. Néanmoins, les premiers résultats sont très encourageants. Et au-delà de ces résultats, nous sommes convaincus que le fondement de l'approche par patchs peut être largement appliqué, y compris à d'autres méthodes et que la flexibilité apportée en terme de ressources matériels et de paramétrisation en fait un outils des plus utiles pour mettre au point un système de reconstruction adapté aux grandes scènes ou aux modèles très détaillés.

C.4 Ré-éclairage de visage

C.4.1 Introduction

L'objectif de ce chapitre est de changer les conditions d'éclairage d'une photographie réelle. On souhaite par exemple être capable de changer une image prise en plein jour en une image de nuit. L'intérêt premier d'une telle technique est de pouvoir combiner plusieurs images d'origines différentes en accordant leur éclairage. Si cette étape n'est pas réaliser avant d'associer les images, la qualité globale en est grandement réduite à cause d'indices visuels incohérents (ombres mal placées, etc).

Le ré-éclairage se décompose en deux parties : en premier lieu, il faut supprimer les indices visuels (ombres, reflets, etc) liés à l'éclairage initial; ensuite, il faut en générer de nouveaux correspondant aux nouvelles conditions. Ces deux étapes requièrent et utilisent les informations suivantes sur la scène observée : la géométrie des objets (responsable de la forme et de la position de l'ombrage), leur matériau (qui influe sur l'apparence de l'ombrage) et la configuration lumineuse (qui est impliqué dans tous ces points). L'information nécessaire est particulièrement importante et traiter le cas général ne semble pas faisable tant les cas rencontrés seraient divers. On s'est donc restreint au cas des visages.

Les visages sont une part importante du corps humain. Notre habitude de les observer nous rend tout particulièrement exigeant quant à la qualité d'une image synthétique. La tolérance aux erreurs est bien moindre pour un visage que pour un objet «normal». Qui plus est la peau est un matériau qui a une interaction spécifique avec la lumière à cause des multiples couches qui la compose (huile, épiderme, sang, etc).

Il existe des techniques pour reproduire cette interaction complexe mais les temps de calcul sont particulièrement longs. Elles deviennent aujourd'hui abordables sur les cartes graphiques les plus récentes mais resteront inaccessibles longtemps sur les systèmes portables (PDA, téléphone, etc).

Ces travaux sont motivés par la mise au point d'une technique qui soit efficace même avec une configuration matérielle limitée. On vise par conséquent un rendu temps réel sur du matériel grand public. On recherche aussi un rendu de qualité, sachant qu'un rendu quasiment photoréaliste comme en produisent les méthodes les plus sophistiquées n'est pas accessible dans notre cadre.

C.4.2 État de l'art

Acquisition de la réflectance : Échantillonage Pour obtenir la réflectance d'un objet (la fonction qui définit son apparence en fonction de l'éclairage et du point de vue), une solution possible est d'échantillonner son apparence sous plusieurs points de vue et plusieurs éclairages. On obtient ainsi une base de données que l'on peut interroger afin de synthétiser de nouvelles images.

Cette approche comporte deux inconvénients majeurs. Le premier est le système d'acquisition qui est toujours encombrant et est la plus part du temps construit spécifiquement pour la tâche visée. Un tel système est difficilement utilisable. Le second désavantage vient de la taille des données acquises : elle est si importante que le rendu d'une nouvelle image ne peut être fait en temps réel à cause de la complexité des requêtes dans la base de données.

Acquisition de la réflectance : Paramétrisation Un représentation plus compacte de la réflectance est d'utiliser un modèle paramétrique *i.e.* une expression analytique qui donne la valeur de la réflectance en fonction de quelques paramètres. Il existe plusieurs grandes catégories parmi ces modèles :

Modèles d'imitation: Il s'agit là de simplement produire un effet visuel convaincant. Le seul modèle encore populaire aujourd'hui est le modèle de Phong. Celui-ci est simple mais est suffisamment expressif pour reproduire reproduire ombrage et reflet. Nos travaux sont basés sur ce modèle car il est aujourd'hui disponible sur toutes les cartes graphiques, même les plus anciennes.

Modèles physiques: Ces modèles cherchent à reproduire au mieux les phénomènes physiques et optiques en jeu au niveau de la surface. Les paramètres de contrôle sont des grandeurs physiques comme l'absorbance du matériau ou sa rugosité. Certains modèles sont spécifiquement conçus pour simuler la peau. Malheureusement ceux-ci sont généralement trop complexes pour atteindre des performances temps réel.

Approximations efficaces: Cette classe de modèle vise au contraire les perfomances lors de l'étape de rendu. Les paramètres n'ont généralement pas de signification physique et sont déterminées à partir d'un jeu d'échantillons réels ou d'un modèle physique. Ces modèles sont une piste pour améliorer par le futur notre rendu.

Acquisition de la réflectance : Déterminer les paramètres Les méthodes évoquées ci-dessus fournissent pour la plupart une technique efficace pour en déterminer les paramètres. Toutefois, dans de nombreux cas pratiques les données sont insuffisantes pour effectuer correctement cette opération. Pour résoudre cette difficulté, plusieurs techniques ont été décrites dans le cas général.

À cause des spécificités de la peau, ces méthodes ne sont généralement pas applicables et plusieurs auteurs ont proposé des approches dédiées aux visages. Mais, soit ces méthodes travaillent avec un modèle trop simplifié de la peau pour notre objectif (les reflets sont ignorés); soit elles reposent sur l'utilisateur pour obtenir une partie des paramètres. Nous pensons qu'il est possible de trouver un compromis avec un modèle comme celui de Phong qui est à la fois assez expressif pour prendre en compte les reflets et à la fois suffisamment simple pour que l'ensemble des paramètres puisse être calculé.

Amélioration avec une texture L'usage d'une texture pour rajouter du détail à un modèle est largement répandu aujourd'hui. Dans le cas du visage, il s'agit de rajouter les éléments tels que les yeux et la bouche qui ne peuvent pas être représentés par le modèle de peau. Une texture permet aussi de rajouter les détails de la peau de taille plus petite comme les grains de beauté, cicatrices, etc. À une échelle plus petite encore, on utilise une texture pour donner à la peau une certaine rugosité.

Plusieurs études ont montré que, dans le cadre du ré-éclairage, la texture doit être combinée multiplicativement avec le modèle sous-jacent pour préserver l'ombrage.

Rendu de matériaux complexes Depuis l'apparition des cartes graphiques programmables, il existe de nombreuses méthodes pour représenter des matériaux complexes d'une manière efficace qui puisse tirer profit de ces cartes. Le problème avec ces techniques est qu'elle utilise une très grande partie de la puissance de calcul disponible. Cela en restreint l'usage à un objet et limite la complexité du reste de la scène. Toutefois, les cartes graphiques étant de plus en plus puissantes, ces méthodes seront rapidement utilisables; et il est envisageable de remplacer le moteur de rendu dans ce chapitre par une de ces techniques, la partie analyse de nos travaux pouvant facilement s'adapter.

D'autres chercheurs ont obtenu un résultat connexe intéressant en travaillant cette fois sur la structure de l'environnement lumineux : il est possible d'approcher un environnement complexe par un nombre fini de sources ponctuelles de lumière. Le corollaire de ce résultat est qu'il suffit d'être capable de synthétiser une image pour une seule source ponctuelle pour ensuite, en répétant le processus pour plusieurs sources, obtenir un rendu dans un environnement complexe.

C.4.3 Vue d'ensemble de la méthode

Les données initiales sont :

Maillage 3D du visage: On utilise un modèle 3D du visage qui doit être suffisamment précis. En pratique, nous avons utilisé un scanner 3D laser et nous avons aussi testé un modèle reconstruit avec une technique de vision utilisant plusieurs images.

Photographie du visage: Cette photographie est prise au flash dans une salle sans lumière. Cette image est calibrée : la correspondance avec le modèle 3D est connue.

Photographie d'un miroir sphérique: On prend une photo d'un miroir sphérique dans les mêmes conditions que le visage.

À partir de ces données, une phase d'analyse extrait les paramètres nécessaires pour approcher au mieux la peau avec un modèle de réflectance. Une texture de détails est aussi déterminée pour améliorer le rendu de la peau.

On aboutit ainsi à un moteur de rendu particulièrement simple : le maillage est intégralement rendu avec le modèle de la peau et la texture vient y ajouter tous les détails manquants (yeux, tâches de rousseur, rugosité, etc). Cette simplicité permet d'obtenir un rendu rapide, qui n'accapare pas toute la puissance graphique disponible et qui ne nécessite que des fonctions disponibles sur toutes les cartes graphiques grand public.

C.4.4 Texture de détails

Cette texture est appliquée sur le rendu de la peau et ne change pas en fonction des conditions lumineuses. Elle se doit donc de ne contenir ni la peau du visage ni les indices visuels qui dépendent du point de vue, c'est-à-dire principalement les reflets. Pour la calculer, on construit une image du visage «entièrement en peau» et éclairé dans les conditions de la photographie. Ensuite la texture est obtenue en divisant la photo initiale par cette image.

Carte de réflectance de la peau Pour synthétiser l'image du visage «entièrement en peau», on détermine un modèle de la réflectance de la peau telle qu'elle apparaît dans la photographie. Pour cela, on échantillonne la photographie et on met chaque couleur obtenue avec sa normale. On peut représenter ces données comme des points à la surface de la sphère des directions. À partir de ces points, on calcule par interpolation - extrapolation une carte complète de la réflectance pour chaque orientation possible de la surface du visage.

La synthèse de l'image du visage «en peau» est alors directe à partir de cette carte de réflectance : on assigne une couleur à chaque point du visage en fonction de sa normale.

Image ratio La texture est calculée en divisant pixel par pixel, composante par composante la photographie au flash par l'image du visage «en peau».

C.4.5 Paramètres du modèle de peau

Les travaux présentés dans ce chapitre s'appuie sur le modèle de Phong. Ce modèle décrit la couleur de l'éclairage par 9 paramètres (rouge, vert, bleu pour spéculaire, diffus, ambiant) et le matériau par 10 paramètres (la brillance en plus).

Paramètres de l'éclairage Grâce au reflet du flash dans le miroir sphérique, on peut déterminer sa position angulaire. En observant le halo qui entoure cette réflexion, on détermine la teinte du flash, son intensité étant fixée arbitrairement et toutes les intensités lumineuses impliquées dans le processus sont exprimées en rapport à celle du flash. La partie sombre du miroir sphérique permet de calculer la couleur de l'éclairage ambiant (*i.e.* qui vient après réflexion sur les murs).

Paramètres de la peau Notre stratégie est une analyse par synthèse : à partir d'un jeu de paramètres, on synthétise une image du visage que l'on compare à la photographie initiale et on adapte les paramètres pour améliorer la ressemblance. Pour cela, on utilise une descente de gradient à pas variable. Toutefois, le paramètre de brillance ne peut pas être inclus dans ce processus car il introduit plusieurs cas ambigus qui rendent l'optimisation instable. Ce paramètre est donc déterminé séparément en analysant la carte de réflectance précédemment construite.

C.4.6 Moteur de rendu

Le principe du moteur est basique : un rendu Phong avec les paramètres de la peau est combiné avec la texture de détails. Il peut donc être implémenté avec n'importe quelle bibliothèque de rendu. Nous avons utilisé OpenGL pour nos exemples.

Comme la texture contient des valeurs supérieures à 1, il faut soit bénéficier d'une carte supportant cette option soit faire appel à un rendu multi-passes (qui est décrit dans la version complète).

Ce système étant particulièrement léger, il est aisé de lui ajouter des fonctions pour améliorer un point spécifique. À titre d'exemples, nous avons ajouté la simulation de la sous- et surexposition, des reflets dans les yeux et des ombres projetées.

C.4.7 Résultats

Les différents résultats montrent que la technique présentée permet de créer des images de qualité à une cadence élevée (au delà de 30Hz) y compris pour des cartes graphiques peu puissantes. Par rapport à d'autres techniques où les données sont produites par des artistes, l'aspect des visages sont plus réalistes. En comparant avec des images réelles, on peut noter un certain nombre de différences significatives (reflet du nez absent, ombres trop dures ou trop douces, etc) mais l'ombrage dans son ensemble est cohérent. Plus particulièrement, les résultats animés mettant en scène des sources lumineuses en mouvement donnent des résultats convaincants et très satisfaisants, tout en laissant de la puissance de calcul «en réserve».

C.4.8 Conclusion

On a décrit une analyse robuste qui permet, à partir d'une photographie du visage et de son modèle 3D, d'extraire les informations nécessaires pour effectuer un rendu rapide et de qualité. Qui plus est la technique de rendu est compatible avec l'animation du visage tant que celle-ci respecte les coordonnées de texture, ce qui est le cas la plupart du temps.

Les résultats obtenus montrent que malgré la simplicité de la technique la plupart des régions du visage est synthétisée de manière convaincante. Cela invite à étudier la possibilité d'un rendu en niveau de détails où seules les régions les plus importantes se verraient synthétiser avec un modèle complexe alors que les autres régions seraient rendues avec une méthode simple et rapide comme celle que l'on vient de présenter.

C.5 Capture de la géométrie d'une chevelure

C.5.1 Introduction

La création de la géométrie d'une chevelure est la partie la plus fastidieuse pour un utilisateur qui travaille avec des cheveux virtuels. Le rendu et l'animation ont connu des progrès importants ces dernières années et sont aujourd'hui largement automatiques alors que la modélisation repose encore essentiellement sur l'utilisateur qui doit tout créer à partir de zéro. Quand en plus il s'agit de reproduire une chevelure donnée, le processus d'édition devient particulièrement fastidieux pour reproduire chaque boucle, ondulation...

On propose dans ce chapitre de créer la géométrie d'une chevelure à partir d'une série de photos d'une vraie personne. La technique proposée reproduit l'ensemble des caractéristiques géométriques visibles dans les photos. Cette approche est complémentaire des systèmes d'édition traditionnels : le résultat de la capture peut être utilisé tel quel ou utiliser comme base pour une session utilisateur.

L'application principale de la méthode proposée est la création de la chevelure d'un clone virtuel. Une telle technique peut aussi être utile dans le domaine de la coiffure et de la recherche criminelle pour établir une base de données de chevelures.

L'objectif de ces travaux est de produire des données exploitables pour le rendu de la chevelure sous un point de vue et/ou un éclairage nouveau. On cherche par conséquent à reconstruire un ensemble de lignes représentant les mèches de la chevelure.

Un des points forts de notre méthode est d'exploiter les propriétés de l'interaction lumineuse des cheveux pour en tirer de l'information alors que pour les méthodes classique de reconstruction cette interaction complexe est un problème majeur empêchant d'atteindre des résultats corrects.

C.5.2 État de l'art

Les techniques de reconstruction à partir d'images sont pour la plupart fortement perturbées par l'apparence des cheveux qui varie fortement en fonction du point de vue. Beaucoup de scanners 3D échouent aussi à cause de la diffusion de la lumière par les fibres capillaires.

Et dans le meilleur des cas, ces techniques renvoient une surface qui n'est pas une description appropriée de la chevelure.

Méthodes génériques On peut utiliser des méthodes qui capturent l'apparence d'un objet indépendemment de sa géométrie. Ces techniques fonctionnent sur les cheveux mais les données ainsi acquises sont particulièrement difficiles à éditer.

Modélisation par l'utilisateur Il existe plusieurs approches pour faciliter le travail d'un «artiste» qui doit créer une chevelure. Ces techniques proposent des outils pour éviter d'avoir à manipuler les cheveux un par un. Toutefois créer une chevelure est toujours un processus long car il faut partir de zéro. Et s'il faut reproduire toutes les boucles et ondulations d'une personne réelle, la tâche devient particulièrement fastidieuse.

Méthodes à base d'images Il existe très peu de méthodes à base d'images. Certaines proposent de capturer uniquement le volume des cheveux à l'aide des images et ensuite de remplir ce volume avec des cheveux créés indépendamment des images. Dans ce cas, seul le volume est reproduit, le style lui n'est au mieux que ressemblant.

La seule méthode travaillant 100% à partir d'images à ce jour ne donne malheureusement que des résultats partiels. Néanmoins la trame de la méthode proposée dans ce chapitre est inspirée de ces travaux.

C.5.3 Vue d'ensemble de la méthode

Les données d'entrée sont des séquences d'images prises à partir d'une caméra fixe avec une source lumineuse qui se déplace. Ainsi, on peut observer un point de la chevelure sous plusieurs éclairages différents. En contrepartie, on ne peut extraire aucune information stéréoscopique des images. Par conséquent, l'information tridimensionnelle provient de l'analyse des reflets sur la chevelure. Pour capture l'intégralité des cheveux, on utilise plusieurs séquences d'images prises autour de la tête.

Notre étude se base sur les mèches qui sont l'entité visible dans les images car les fibres sont trop fines. Ces mèches sont considérés comme un ensemble de courts segments rectilignes d'environ un millimètre. L'approche présentée se décompose en deux grands blocs : tout d'abord une phase qui analyse les séquences une par une pour calculer l'orientation tridimensionnelle des segments; puis dans un second temps, une phase qui construit la géométrie de la chevelure en combinant l'information venant de tous les points de vue.

Le calcul de l'orientation se décompose en deux temps : l'orientation de segments est calculé en 2D dans le plan image avant d'être transformée en une orientation 3D grâce à l'analyse des reflets.

Limitations La méthode proposée fonctionne dans de nombreux cas. Néanmoins, il est des situations qui ne peuvent être résolues par notre technique. Travaillant à partir d'images, le système ne capture que la partie visible de la chevelure. La méthode repose sur l'hypothèse que les mèches sont à l'échelle d'un pixel, des dreadlocks ne seraient pas correctement reconstruits par exemple. On suppose aussi que chaque pixel à une seule orientation ce qui peut poser problème avec des cheveux très emmêlés.

C.5.4 Orientation des segments

Orientation 2D On veut calculer l'orientation de la projection de chaque segment dans le plan image. Il s'agit d'un problème classique en traitement d'images : quelle est l'orientation locale d'une image?

Il existe de nombreuses techniques pour répondre à cette question. Ces techniques se fondent toutes sur une propriété théorique de l'image (le profil des contours par exemple). Malheureusement les images de cheveux sont complexes car une fibre est plus fine qu'un pixel ce qui introduit des problèmes d'*aliasing*. En plus, les cheveux réfléchissent, diffractent, diffusent,... ce qui rend sans espoir de définir une propriété forte sur les images dont l'on dispose. On suppose alors uniquement qu'il existe une orientation en chaque pixel.

On propose dans ce chapitre, non pas une nouvelle technique de mesure de l'orientation dans les images, mais une méthode pour choisir la technique de mesure la plus fiable parmi plusieurs. Pour cela, on travaille avec les filtres dits *orientés* : l'image est convoluée localement avec un noyau que l'on fait tourner pour tester toutes les orientations possibles. On obtient ainsi une courbe réponse dont la valeur la plus élevée correspond à l'orientation résultat. Pour évaluer la fiabilité de ce résultat, on examine la courbe réponse. Si elle est plate, alors le résultat n'est pas fiable car plusieurs orientations remplissent le même critère. Si la courbe est piquée, le filtre est discriminant et le résultat est fiable. Pour chiffrer cette fiabilité, on utilise donc la variance de la courbe réponse : une faible variance indique un résultat fiable.

L'intérêt majeur de ce critère est qu'il permet de comparer rigoureusement la fiabilité de plusieurs filtres appliqués potentiellement sur des images différentes et/ou à des positions différentes. Pour un pixel donné, on peut ainsi tester plusieurs filtres, chacun sur plusieurs images avec des conditions d'éclairage différentes et choisir l'orientation issue de la configuration la plus fiable.

Plusieurs paramètres rentrent en jeu pour cette série de tests :

Fréquence: Afin de n'étudier que l'orientation des mèches et non pas celle des reflets, on applique un filtre passe-bande aux images pour ne conserver que les fréquences les plus élevées.

Profile détecteur: Il s'agit du profile du noyau du filtre dans la direction de la variation à détecter. Nous avons testé 8 profiles différents.

Profile projecteur: C'est le profile orthogonal. Nous avons montré que le critère de variance a le même comportement des critères connus : pour un signal infiniment allongé, plus ce profil est allongé, plus fiable est la détection. Comme en pratique le signal n'est pas infiniment allongé, nous testons 3 allongements différents.

Pour tirer partie de la cohérence locale de l'orientation des cheveux, on finit en appliquant un filtre bilatéral aux orientations 2D calculées. Ce filtre «corrige» l'orientation d'un pixel à l'aide de ses voisins en prenant en compte : la distance qui les sépare (un voisin distant a moins d'influence), leur fiabilité (un voisin plus fiable a plus d'influence) et la similarité de leur apparence (un voisin qui a une apparence proche a plus de poids).

Cette technique pour mesurer l'orientation 2D locale des images est validée sur des images synthétiques de référence et des images réelles et comparer à des méthodes connues. À chaque fois, les résultats obtenus sont plus précis, ce qui montre l'intérêt de cette méthode pour les cas difficiles tels que les images de cheveux.

Orientation 3D L'idée ici est de trouver, pour chaque segment, une position de la lumière qui correspond à la configuration miroir (où le point de vue et la lumière forment deux angles égaux avec la normale au segment). Pour chaque pixel, on cherche dans les images celle qui donne la réflexion la plus importante. La position de la lumière correspondante est dans la configuration miroir et on peut ainsi calculer une normale au segment qui se trouve le pixel étudié.

Connaissant la position de la caméra et l'orientation 2D d'un segment, on peut calculer un premier plan dans l'espace 3D qui contient ce segment. Ensuite, la normale que l'on vient de calculer définit un second plan qui, intersecté avec le premier, caractérise l'orientation 3D du segment.

C.5.5 Mise en œuvre

Pour couvrir l'ensemble de la chevelure, nous utilisons 4 points de vue : haut, arrière, gauche et droite. L'étude précédente est répétée à chaque fois afin d'obtenir une orientation 3D pour chaque pixel.

Calibration du système L'utilisateur fournit pour chaque point de vue une ellipse qui approche la silhouette de la chevelure. En combinant ces ellipses ensemble on construit un ellipsoïde 3D que l'on considère comme une approximation de l'ensemble des points de départ des mèches. Cet ellipsoïde permet aussi de ramener dans un repère commun tous les points de vue pour calibrer le système.

Croissance des mèches Les mèches sont créées une par une. La croissance commence toujours par un point à la surface de l'ellipsoïde. Ensuite les segments sont ajoutés à la suite des un des autres. Quand plusieurs caméras «voient» la position du prochain segment à chaîner, on combine linéairement les segments que chacune propose en tenant compte de leur angle de vue : les angles rasants sont moins influents car ils correspondent aux silhouettes où l'information est de piètre qualité à cause de la perspective fuyante.

C.5.6 Résultats

Nous avons reconstruits plusieurs chevelures de plusieurs types : une blonde mi-longue à grandes ondulations, une auburn longue ondulée et une brune courte à petites boucles désordonnées. Nous avons rencontrés des difficultés : l'ellipsoïde est parfois une mauvaise approximation de la forme de cheveux, les cheveux longs ont tendance à bouger lors de la capture.

Néanmoins, les résultats sont à chaque fois convaincants. Nous avons synthétisé une séquence qui reproduit le même mouvement de lumière : les motifs des reflets sont similaires aux originaux. Nous avons aussi mesurés l'erreur angulaire sur la déviation de la lumière nécessaire en chaque pixel pour obtenir le même reflet. L'erreur ainsi obtenue est de l'ordre de quelques degrés ce qui est acceptable compte tenu du fait que notre système expérimental n'a vocation qu'à être une preuve de concept et reste améliorable sur de nombreux points.

C.5.7 Conclusion

D'un point de vue pratique, le système proposé fonctionne et capture de manière précise la géométrie d'une chevelure. D'un point de vue théorique, la sélection par la variance a été prouvée partager des propriétés communes avec d'autres méthodes.

Nous pensons que cette technique ouvre la voie vers une nouvelle façon de travailler avec les cheveux. Il serait entre autres intéressants et utile d'acquérir aussi la réflectance des cheveux. Plus ambitieusement, on pourrait étudier le problème de la capture des mouvements des cheveux. Pousser plus loin l'étude théorique serait aussi intéressant.

C.6 Conclusion générale

Dans ce manuscrit, nous avons abordé trois points principaux autour de l'extraction de données à partir d'images :

- La reconstruction de surface à partir de plusieurs points de vue.
- Le ré-éclairage de visage à l'aide d'une seule image et d'un modèle 3D.
- La capture de la géométrie d'une chevelure à l'aide d'une lumière placée à plusieurs endroits.

Notre première remarque est de souligner que ces trois points illustrent la variété de l'information qui peut être exploitée dans les images : contour, reflet, texture, couleur, ombrage, etc. Et ces informations sont d'autant plus riches quand on peut observer leur évolution sur plusieurs images.

Nous avons aussi montré plusieurs configurations utiles : caméra fixe ou mobile, lumière fixe ou mobile, images seules ou image plus modèle 3D. Pour chaque configuration, nous proposons un algorithme dédié qui extrait une information dense.

À partir de ces études de cas, nous proposons quelques conclusions générales.

La redondance est utile

Dans nos travaux sur la reconstruction de surface et la capture de chevelure, nous utilisons de nombreuses images du même objet avec une faible variation entre deux images consécutives (petit déplacement de la caméra ou de la lumière). Même si les données utilisées sont redondantes, nous sommes convaincus que cette redondance participe à la précision des résultats obtenus.

En exploitant cette redondance, nos algorithmes sont robustes – c'est-à-dire qu'ils peuvent fonctionner à partir de données non parfaites. Puisque l'information est «reproduite» plusieurs fois dans les données, il nous est possible de compenser certaines erreurs. Nous pouvons ainsi accorder plus de «confiance» aux données en entrée et introduire moins de connaissances a priori pour contrôler nos processus. Nous sommes «plus près» des données originales parce que nous pouvons détecter les données corrompues. Cette robustesse est d'autant plus importante que l'on recherche des données denses : nous devons produire un résultat raisonnable même pour les régions peu représentées dans les images de départ.

La complétude et le haut niveau de détails des surfaces et des chevelures confirment cette situation : nous reproduisons correctement les objets réels.

L'information est visible

Dans toutes les parties présentées, l'information extraite est «visible» dans les données de départ. Si l'on joue la séquence d'images utilisée pour la reconstruction de surface comme un film, on perçoit la 3D de la scène. Pour le ré-éclairage de visage, on peut évaluer facilement l'état de la peau entre mat, sec et mouillé. Et il est facile de suivre les mèches de cheveux avec un crayon sur nos séquences de chevelures.

Cette remarque nous a encouragé à poursuivre nos recherches dans les cas les plus durs. Par exemple, pour la capture de chevelure, il n'était pas évident à partir des images originales qu'il était possible de retrouver une orientation dense. Mais une fois que l'on a vu les hautes fréquences des images avec des lignes bien marquées et presque aucun reflet, nous étions convaincus que «si on *voit* les lignes, on peut les retrouver».

Cela peut ne pas toujours être vrai. Reprenons le «cube plat» (figure 2.1 en page 6) : on devine une forme tridimensionnelle tandis qu'objectivement il n'y a rien de 3D. Malgré tout, dans ce cas, il est possible de d'introduire la connaissance humaine a priori qui s'appuie sur les lignes parallèles et les angles pour retrouver l'information 3D.

Néanmoins, il existe peut-être des cas plus compliqués pour lesquels on ne peut pas résoudre le problème même si l'on «voit» l'information. Mais nous sommes néanmoins convaincus que ces cas sont rares et méritent d'être étudiés : notre cerveau est capable d'extraire l'information, il y a donc une chance de trouver une solution.

Une structure de données appropriée est importante

Nous avons déjà discuté ce point pour la reconstruction de surface : nous sommes convaincus qu'une formulation dans l'espace 3D est fondamentale pour capturer une forme 3D. Nous pensons aussi que créer des lignes pour représenter la géométrie d'une chevelure est un choix crucial qui ouvre la voie à d'autres applications (l'acquisition de la réflectance, l'édition de la géométrie et son animation) et que ce choix est plus judicieux qu'une surface texturée. À propos du ré-éclairage de visage, comprendre l'influence de chaque paramètre permet de déterminer un jeu de paramètres qui fait sens et de mettre au point un schéma d'optimisation robuste. Manipuler un bon jeu de paramètres rend possible l'extraction complète d'un modèle de réflectance à partir d'une seule image.

Dans ce manuscrit, nous présentons des méthodes qui visent à reconstruire l'objet original indépendemment de l'usage ultérieur de cette donnée (à l'exception du ré-éclairage de visage qui est adapté au rendu rapide) : nous cherchons une surface qui ressemble à l'objet de départ, un ensemble de cheveux qui copie la chevelure originale. Ces données sont en quelque sorte des données «à tout faire»; elles peuvent être utilisées pour n'importe quelle application, éventuellement après un traitement. Il serait intéressant d'étudier la création de données dédiées qui tiennent compte de l'usage visé. Cette idée est discutée plus loin dans les travaux futurs.

C.6.1 Travaux futurs

Au-delà des travaux futurs présentés dans chaque chapitre, nous proposons un aperçu de ce que pourrait être la suite de nos travaux.

Données volumétriques Tout d'abord, il serait intéressant d'étudier d'autres cas. Dans cette direction, Reche *et al.* [178] proposent une étude sur la reconstruction d'arbres en les considérant comme un matériau non opaque. Cela amène à une reconstruction volumétrique différente de notre reconstruction de surfaces et de lignes. Toujours dans cette direction, on peut aussi envisager de travailler sur des entités volumétriques comme des objets translucides (un verre, une bouteille par exemple) mais aussi la fumée et l'eau. Les applications seraient l'insertion d'un acteur digital dans un nuage de gaz réel ou rendre possible des interactions complexes entre de l'eau réelle et des éléments virtuels.

Influence sur l'environnement Une autre direction intéressante est pointé par Chuang *et al.* [40] qui capturent l'ombre d'un objet sous le soleil et l'adapte à une nouvelle scène. Ce travail est original en ce qu'il capture non pas l'objet mais son effet sur son environnement. Goesele *et al.* [73] proposent aussi un premier pas pour mesurer une lumière directionnelle. Il serait intéressant d'étendre ces techniques à d'autres effets «périphériques» tels qu'une source de lumière plus générale ou des effets d'illumination globale plus complexes (caustiques, ombres douces, etc). Cela pourrait aboutir à des applications utiles comme du ré-éclairage amélioré ou une meilleure incrustation des objets capturés dans une nouvelle scène.

Données «dédiées» À côté des autres scénarii, on peut aussi étudier des données «dédiées» comme déjà mentionné. On entend par cela que l'on pourrait prendre en compte l'usage futur des données pendant leur capture. Dans ce manuscrit, seul le ré-éclairage de visage suit cette voie. Pour la reconstruction de surface, il s'agirait par exemple de créer des surfaces NURBS avec des points de contrôle pour donner la possibilité de l'éditer. Une approche appropriée serait de déterminer les NURBS directement à partir des images et non pas de les ajuster sur la surface déjà extraite. Une telle relation direction aboutirait à une résultat plus cohérent. Si la surface doit être rendue alors étudier des représentations spécifiques comme les «nuages d'imposteurs» [59] est un choix pertinent.

Pour les cheveux, cela amènerait à considérer des structures dédiées à l'animation comme les hiérarchies de mèches [15] ou à l'édition comme les flots de fluide [80]. Il est bon de remarquer que notre technique actuelle est proche d'un flot lorsqu'on manipule le champ d'orientation 3D. De l'autre côté, créer une hiérarchie de mèches à partir d'images serait plus ambitieux et délicat.

Clôture

En conclusion, nous souhaitons dire que ce domaine de recherche est large et recèle de nombreux défis à relever. De grands progrès sont encore à attendre et nous sentons que nous sommes de plus en plus proche d'algorithmes utilisables en production. Nous avons montré qu'une information complète et dense peut être obtenue à partir d'images. La facilité d'utilisation doit encore être améliorée (en réglant automatiquement tous les seuils et poids des algorithmes par exemple). Néanmoins les communautés de informatique graphique et de vision par ordinateur se sont rapprochées et les fruits en sont nombreux et prometteurs : de nombreux chercheurs proposent des techniques à base d'images : la photographie 3D [154], le magnétoscope 3D [232], la télévision 3D [151], etc. Nous sommes convaincus que ce domaine de recherche va bientôt engendrer de nouveaux appareils qui seront accessibles au grand public.

List of Figures

List of Tables

List of Tables

Bibliography

[1] David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269–277, 1995. cited on page(s) 76, 77

[2] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[3] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David H. Salesin, and Michael F. Cohen. Interactive digital photomontage. *ACM Transactions on Graphics*, 23(3):294–302, July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 33

[4] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993. ISBN 013617549X. cited on page(s) 26, 40, 41

[5] Daniel Aliaga, Rui Bastos, Mary Whitton, Fred Brooks, Dinesh Manocha, Jon Cohen, Andrew Wilson, Eric Baker, Hansong Zhang, Carl Erikson, Kenny Hoff, Tom Hudson, and Wolfgang Stuerzlinger. MMR: an interactive massive model rendering system using geometric and image-based acceleration. In *Proceedings of the symposium on Interactive 3D graphics*, pages 199–206. ACM SIGGRAPH, 1999. ISBN:1-58113-082-1. cited on page(s) 5

[6] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion (II). *SIAM Journal of Numerical Analysis*, 29(3), June 1992. cited on page(s) 61

[7] Nina Amenta, Marsahll Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the ACM SIGGRAPH conference*, pages 415–421. ACM, 1998. cited on page(s) 85

[8] Nina Amenta, Sunghee Choi, and Ravi Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001. cited on page(s) 85

[9] Ken-ichi Anjyo, Yoshiak Usami, and Tsuneya Kurihara. A simple method for extracting the natural beauty of hair. *Computer Graphics*, 26(2):111–120, July 1992. Proceedings of the ACM SIGGRAPH conference. cited on page(s) 131

[10] Michael Ashikhmin, Simon Premoze, and Pete Shirley. A microfacet-based BRDF generator. In *Proceedings of the SIGGRAPH conference*. ACM, 2000. cited on page(s) 97

[11] Gilles Aubert and Pierre Kornprobst. *Mathematical problems in image processing: Partial Differential Equations and the Calculus of Variations*, volume 147 of *Applied Mathematical Sciences*. Springer, 2002. cited on page(s) 59

[12] Simon Baker and Shree Nayar. Global measures of coherence for edge detector evaluation. In *Proceedings of the conference on Computer Vision and Pattern Recognition*, volume 2, June 1999. cited on page(s) 140

[13] Bruce Guenther Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford University, 1974. cited on page(s) 17

[14] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957. cited on page(s) 22

[15] Florence Bertails, Tae-Yong Kim, Marie-Paule Cani, and Ulrich Neumann. Adaptive wisp tree. In *Proceedings of the Symposium on Computer Animation*. ACM, 2003. cited on page(s) 131, 162, 165, 202

[16] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the SIGGRAPH conference*, pages 417–424. ACM, 2000. cited on page(s) 113

[17] Michael J. Black, Guillermo Sapiro, David H. Marimont, and David Heeger. Robust anisotropic diffusion. *Transactions on Image Processing*, 7(3):421–432, March 1998. cited on page(s) 60

[18] Andrew Blake and Andrew Zisserman. *Visual reconstruction*. Mit Press, 1987. ISBN:0-262-02271-0. cited on page(s) 47, 58

[19] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the SIGGRAPH conference*, pages 187–194. ACM, 1999. cited on page(s) 100

[20] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the SIGGRAPH conference*, pages 192–198. ACM, 1977. cited on page(s) 97

[21] Aaron F. Bobick and Stephen S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, September 1999. cited on page(s) 22

[22] Samuel Boivin and André Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proceedings of the SIGGRAPH conference*, pages 107–116. ACM, 2001. cited on page(s) 99

[23] BOOST website. `http://www.boost.org/`. cited on page(s) 87

[24] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787. cited on page(s) 14

[25] Edmond Boyer and Jean-Sébastien Franco. A hybrid approach for computing visual hulls of complex objects. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, volume 1, pages 695–701, June 2003. cited on page(s) 18, 21

[26] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 26–33. IEEE, October 2003. cited on page(s) 31, 38, 45, 51, 170, 173

[27] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, To appear 2004. cited on page(s) 42, 66, 87

[28] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. cited on page(s) 28, 29, 38, 46, 58

[29] Allan C. Brik, John Williams, and Jerome J. Connor. Multiresolution, incremental generation of 3D computer models from video data. In *Proceedings of the symposion on Solid Modeling and Applications*, pages 95–102. ACM, 1993. cited on page(s) 17

[30] Andrian Broadhurst, Tom Drummond, and Roberto Cipolla. A probabilistic framework for space carving. In *Proceedings of the International Conference on Computer Vision*, pages 388–393. IEEE, July 2001. cited on page(s) 20

[31] Michael J. Brooks and Berthold K. P. Horn, editors. *Shape from Shading*. MIT Press, 1989. ISBN:0-262-08183-0. cited on page(s) 132

[32] Chris Buehler, Steven Gortler, Michael Cohen Leonard, and McMillan. Minimal surfaces for stereo. In *Proceedings of the European Conference on Computer Vision*, 2002. cited on page(s) 28, 38, 46, 47, 51, 58, 72

[33] Emmanuel Candès and Dave Donoho. Recovering edges in ill-posed inverse problems: Optimality of curvelet frames. Technical Report 2000-16, Department of Statistics, Stanford University, 2000. cited on page(s) 140

[34] John Canny. Finding edges and lines in images. Master's thesis, Massachusetts Institute of Technology, June 1983. cited on page(s) 142, 144

[35] CAQTI website. `http://www.caqti.com/`. cited on page(s) 132

[36] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):694–699, February/March 1997. cited on page(s) 37, 38

[37] Francine Catté, Pierre-Louis Lions, Jean-Michel Morel, and Tomeu Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal of Numerical Analysis*, 29(1):182–193, February 1992. cited on page(s) 60

[38] Johnny T. Chang, Jingyi Jin, and Yizhou Yu. A practical model for hair mutual interactions. In *Proceedings of the Symposium on Computer Animation*. ACM, 2002. cited on page(s) 131

[39] Boris V. Cherkassky and Andrew V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997. cited on page(s) 42, 66, 76

[40] Yung-Yu Chuang, Dan B Goldman, Brian Curless, David H. Salesin, and Richard Szeliski. Shadow matting and compositing. *ACM Transactions on Graphics*, 22(3):494–500, July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 165, 202

[41] Robert Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 358–363. IEEE, June 1996. cited on page(s) 19

[42] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1), 1982. cited on page(s) 97

[43] Wagner T. Corrêa, James T. Klosowski, and Cláudio T. Silva. iWalk: Interactive out-of-core rendering of large models. Technical Report TR-653-02, Princeton University, 2002. cited on page(s) 5

[44] Ingemar J. Cox, Sunita L. Hingorani, Satish B. Rao, and Bruce M. Maggs. A maximum likelihood stereo algorithm. *Journal of Computer Vision and Image Understanding*, 63(3), May 1996. cited on page(s) 22

[45] W. Bruce Culbertson, Thomas Malzbender, and Gregory G. Slabaugh. Generalized voxel coloring. In *Proceedings of the International Workshop on Vision Algorithms*, Lecture Notes on Computer Science, pages 100–115. Springer Verlag, September 1999. cited on page(s) 19

[46] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the SIGGRAPH conference*. ACM, 1996. cited on page(s) 81

[47] Agnes Daldegan, Nadia Magnenat-Thalmann, Tsuneya Kurihara, and Daniel Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum*, 12(3):211–221, 1993. cited on page(s) 131, 133

[48] Katja Daubert, Hendrik P. A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. Efficient cloth modeling and rendering. In *Proceedings of Eurographics Workshop on Rendering*, 2001. cited on page(s) 102

[49] Jeremy S. de Bonet and Paul Viola. Poxels: Probabilistic voxelized volume reconstruction. In *Proceedings of the International Conference on Computer Vision*. IEEE, 1999. cited on page(s) 20

[50] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the SIGGRAPH conference*. ACM, 2000. cited on page(s) 94, 95, 100, 116

[51] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 104

[52] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the SIGGRAPH conference*. ACM, August 1997. cited on page(s) 115, 122

[53] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs. In *Proceedings of the SIGGRAPH conference*. ACM, August 1996. cited on page(s) 12

[54] Rachid Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2), May 1987. cited on page(s) 142

[55] Rachid Deriche and Olivier Faugeras. Les EDP en traitement des images et vision par ordinateur. *journal du Traitement du Signal*, 13(6), 1996. cited on page(s) 170, 172

[56] Huong Quynh Dinh, Greg Turk, and Greg Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of the International Conference on Computer Vision*. IEEE, 2001. cited on page(s) 20

[57] Dave Donoho and Xiaoming Huo. Beamlet pyramids. In *Proceedings of SPIE conference*, volume 4119, 2000. cited on page(s) 140

[58] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 145

[59] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 74, 165, 202

[60] Michael Elad. On the bilateral filter and ways to improve it. *IEEE Transactions On Image Processing*, 11(10):1141–1151, October 2002. cited on page(s) 145

[61] Olivier Faugeras. *Three-Dimensional Computer Vision*. MIT Press, November 1993. ISBN 0-262-06158-9. cited on page(s) 8, 9, 127

[62] Olivier Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. *IEEE Transactions on Image Processing*, 7(3), 1998. cited on page(s) 23, 24, 25, 33, 34, 37, 38, 58, 76

[63] Olivier Faugeras, Quang-Tuan Luong, and Theo Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001. cited on page(s) 8, 127

[64] Hans G. Feichtinger and Thomas Strohmer, editors. *Advances in Gabor Analysis*. Birkhauser, 2003. cited on page(s) 142

[65] Michael Felsberg and Gerald Sommer. A new extension of linear signal processing for estimating local properties and detecting features. In *Proceedings of DAGM Symposium Mustererkennung*, 2000. cited on page(s) 140

[66] Andrew W. Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. Image-based rendering using image-based priors. In *Proceedings of the International Conference on Computer Vision*. IEEE, 2003. cited on page(s) 74

[67] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips. *Introduction to Computer Graphics*. Addison Wesley Professional, 1993. ISBN: 0201609215. cited on page(s) 118

[68] Lance A. Forbes and Bruce A. Draper. Inconsistencies in edge detector evaluation. In *Conference on Computer Vision and Pattern Recognition*. IEEE, June 2000. cited on page(s) 141

[69] Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, New Jersey, 1962. cited on page(s) 40, 41

[70] Jean-Sébastien Franco and Edmond Boyer. Exact polyhedral visual hulls. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 329–338, September 2003. cited on page(s) 18

[71] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. cited on page(s) 140

[72] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. Illumination-based image synthesis: Creating novel images of human faces under differing pose and lighting. In *Proceedings of the Workshop on Multi-View Modeling and Analysis of Visual Scenes*, pages 47–54. IEEE, 1999. cited on page(s) 99

[73] Michael Goesele, Xavier Granier, Wolfgang Heidrich, and Hans-Peter Seidel. Accurate light source acquisition and rendering. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 115, 165, 202

[74] Michael Goesele, Hendrik P. A. Lensch, Jochen Lang, Christian Fuchs, and Hans-Peter Seidel. DISCO - acquisition of translucent objects. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 95

[75] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the SIGGRAPH conference*. ACM, 1996. cited on page(s) 94

[76] Pierre Gosselet. *Méthodes de décomposition de domaine et méthodes d'accélération pour les problèmes multichamps en mécanique non-linéaire*. PhD thesis, Université Paris 6, 2003. cited on page(s) 78

[77] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–628, 1971. cited on page(s) 96

[78] Stéphane Grabli, François Sillion, Stephen R. Marschner, and Jerome E. Lengyel. Image-based hair capture by inverse lighting. In *Proceedings of the Graphics Interface conference*, pages 51–58, 2002. cited on page(s) 134, 135, 148, 207

[79] Gösta H. Granlund and Hans Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. cited on page(s) 140

[80] Sunil Hadap and Nadia Magnenat-Thalmann. Interactive hair styler based on fluid flow. In *Proceedings of the Workshop on Computer Animation and Simulation*. Eurographics, August 2000. cited on page(s) 131, 133, 162, 165, 202, 207

[81] Sunil Hadap and Nadia Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3), 2001. cited on page(s) 131, 162

[82] Ziyad S. Hakura, Jerome E. Lengyel, and John M. Snyder. Parameterized animation compression. In *Proceedings of the Eurographics Workshop on Rendering*, 2000. cited on page(s) 94

[83] Jefferson Y. Han and Ken Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 74

[84] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the SIGGRAPH conference*, pages 165–174. ACM, 1993. cited on page(s) 97, 98

[85] Richard Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997. cited on page(s) 12

[86] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000. cited on page(s) 8, 127

[87] Jean-Marc Hasenfratz, Marc Lapierre, Jean-Dominique Gascuel, and Edmond Boyer. Real-time capture, reconstruction and insertion into virtual world of human actors. In *Proceedings of the Vision, Video and Graphics conference*, pages 49–56. Eurographics, Elsevier, 2003. cited on page(s) 18

[88] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003. State-of-the-Art Reviews. cited on page(s) 129

[89] Xiao Dong He, Kenneth E. Torrance, François Sillion, and Donald P. Greenberg. A comprehensive physical model for light reflection. In *Proceedings of the SIGGRAPH conference*. ACM, 1991. cited on page(s) 97, 98

[90] Aaron Hertzmann and Steve Seitz. Shape and materials by example: A photometric stereo approach. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, 2003. cited on page(s) 73, 132

[91] Hugues Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Computer Science and Engineering, University of Washington, June 1994. cited on page(s) 85

[92] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the SIGGRAPH conference*, pages 295–302. ACM, 1994. cited on page(s) 85

[93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics journal*, 26(2), 1992. Proceedings of the ACM SIGGRAPH conference. cited on page(s) 85

[94] Berthold K. P. Horn. *Robot Vision*. MIT Press, 1986. ISBN 0-262-08159-8. cited on page(s) 108

[95] Berthold K. P. Horn and Michael J. Brooks, editors. *Shape From Shading*. MIT Press, July 1989. ISBN 0-262-08183-0. cited on page(s) 32

[96] Peter J. Huber. *Robust Statistics*. Probability and Statistics. Wiley-Interscience, February 1981. cited on page(s) 73

[97] William V. Baxter III, Avneesh Sud, Naga Govindaraju, and Dinesh Manocha. Gigawalk: Interactive walkthrough of complex environments. In *Proceedings of the Eurographics Rendering Workshop*, 2002. cited on page(s) 5

[98] Katsushi Ikeuchi and Kosuke Sato. Determining reflectance properties of an object using range and brightness images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1139 – 1153, November 1991. cited on page(s) 99

[99] Hiroshi Ishikawa. *Global Optimization Using Embedded Graphs*. PhD thesis, New York University, May 2000. cited on page(s) 28, 38, 45, 46, 48, 49, 141

[100] Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336, October 2003. cited on page(s) 28, 45, 49

[101] Hiroshi Ishikawa and Davi Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *Proceedings of the European Conference on Computer Vision*, pages 232–248, June 1998. cited on page(s) 38, 46

[102] John Isidoro and Stan Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Proceedings of the International Conference on Computer Vision*, pages 1335–1342. IEEE, 2003. cited on page(s) 18, 32

[103] Henrik Wann Jensen. Digital face cloning. In *Proceedings of the SIGGRAPH conference*. ACM, 2003. Technical Sketch. cited on page(s) 100

[104] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[105] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the SIGGRAPH conference*. ACM, 2001. cited on page(s) 94, 95, 97, 113

[106] Hailin Jin, Anthony J. Yezzi, and Stefano Soatto. Region-based segmentation on evolving surfaces with application to 3D reconstruction of shape and piecewise constant radiance. In *Proceedings of the European Conference on Computer Vision*, 2004. cited on page(s) 24

[107] Hailin Jin, Anthony J. Yezzi, Yen-Hsi Tsai, Li-Tien Chen, and Stefano Soatto. Estimation of 3D surface shape and smooth radiance from 2D images: a level set approach. *journal of Scientific Computing*, 19(1-3):267–292, 2003. cited on page(s) 24

[108] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 60

[109] James T. Kajiya. The rendering equation. In *Proceedings of the SIGGRAPH conference*, pages 143–150. ACM, 1986. cited on page(s) 102

[110] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In *Proceedings of the SIGGRAPH conference*. ACM, 1989. cited on page(s) 131

[111] Jan Kautz and Hans-Peter Seidel. Towards interactive bump mapping with anisotropic shift-variant BRDFs. In *Proceedings of the conference on Graphics Hardware*. ACM SIGGRAPH/Eurographics, 2000. cited on page(s) 102

[112] Junhwan Kim, Vladimir Kolmogorov, and Ramin Zabih. Visual correspondence using energy minimization and mutual information. In *Proceedings of the International Conference on Computer Vision*. IEEE, October 2003. cited on page(s) 28, 38, 46, 58

[113] Tae-Yong Kim. `http://www.rhythm.com/~tae/Links.htm`. cited on page(s) 133

[114] Tae-Yong Kim and Ulrich Neumann. Opacity shadow maps. In *Proceedings of the Eurographics Rendering Workshop*, July 2001. cited on page(s) 131

[115] Tae-Yong Kim and Ulrich Neumann. Interactive multiresolution hair modeling and editing. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 131, 133, 134, 162, 207

[116] Jan J. Koenderink and Sylvia Pont. The secret of velvety skin. *Journal on Machine Vision and Applications*, 14(4):260–268, September 2003. cited on page(s) 98

[117] Jan J. Koenderink and Ans J. van Doorn. Two-plus-one-dimensional differential geometry. *Pattern Recognition Letters*, 15:439–443, 1994. cited on page(s) 170, 172

[118] Jan J. Koenderink and Ans J. van Doorn. Image processing done right. In *Proceedings of the European Conference on Computer Vision*, pages 158–172, May 2002. cited on page(s) 61

[119] Vladimir Kolmogorov. *Graph Based Algorithms for Scene Reconstruction from Two or More Views*. PhD thesis, Cornell University, January 2004. cited on page(s) 28

[120] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the International Conference on Computer Vision*. IEEE, July 2001. cited on page(s) 28, 38, 51, 58

[121] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the European Conference on Computer Vision*, May 2002. cited on page(s) 28, 29, 38, 46, 47, 51, 58, 65, 66, 67, 76

[122] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 2004. cited on page(s) 29

[123] Vladimir Kolmogorov, Ramin Zabih, and Steven Gortler. Generalized multi-camera scene reconstruction using graph cuts. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, July 2003. cited on page(s) 28, 29, 34

[124] Wai Ming Kong, Hiroki Takahashi, and Masayuki Nakajima. Generation of 3d hair model from hair volume. In *Proceedings of the SPIE conference*, 1997. cited on page(s) 133

[125] Wai Ming Kong, Hiroki Takahashi, and Masayuki Nakajima. Generation of 3d hair model from multiple pictures. In *Proceedings of the Multimedia Modeling conference*, pages 183–196, 1997. cited on page(s) 133, 134, 207

[126] Kiriakos N. Kutulakos. Approximate N-view stereo. In *Proceedings of the European Conference on Computer Vision*, pages 67–83, 2000. cited on page(s) 20

[127] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000. cited on page(s) 19, 21, 34, 53, 55, 85

[128] Eric P.F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the SIGGRAPH conference*, pages 117–126. ACM, 1997. cited on page(s) 98, 100

[129] Lutz Latta and Andreas Kolb. Homomorphic factorization of BRDF-based lighting computation. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[130] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994. cited on page(s) 17, 151

[131] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 98

[132] Erwan Le Pennec and Stéphane Mallat. Sparse geometric image representation with bandelets. *IEEE Transactions on Image Processing*, 2003. cited on page(s) 140

[133] Patrick Le Tallec. *Computational Mechanics Advances*, volume 1, chapter Domain Decomposition Methods in Computational Mechanics, pages 123–217. North Holland, 1994. cited on page(s) 78

[134] Hendrik P. A. Lensch, Michael Goesele, Philippe Bekaert, Jan Kautz, Marcus A. Magnor, Jochen Lang, and Hans-Peter Seidel. Interactive rendering of translucent objects. In *Proceedings of the Pacific Graphics conference*, 2002. cited on page(s) 103

[135] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Proceedings of the Eurographics Workshop on Rendering*, 2001. cited on page(s) 99

[136] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the SIGGRAPH conference*. ACM, 1996. cited on page(s) 94

[137] Maxime Lhuillier and Long Quan. Match propagation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1140–1146, 2002. cited on page(s) 64, 127

[138] Maxime Lhuillier and Long Quan. Surface reconstruction by integrating 3D and 2D data of multiple views. In *Proceedings of the International Conference on Computer Vision*. IEEE, October 2003. cited on page(s) 24, 26, 32, 38, 76, 83, 127

[139] Zicheng Liu, Ying Shan, and Zhengyou Zhang. Expressive expression mapping with ratio images. In *Proceedings of the SIGGRAPH conference*, pages 271–276. ACM, 2001. cited on page(s) 102, 110

[140] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the SIGGRAPH conference*, pages 163–169. ACM, 1987. cited on page(s) 20, 81

[141] Céline Loscos, Marie-Claude Frasson, George Drettakis, Bruce Walter, Xavier Granier, and Pierre Poulin. Interactive virtual relighting and remodeling of real scenes. In *Proceedings of Eurographics Workshop on Rendering*, volume 10, pages 235–246, June 1999. cited on page(s) 101, 102, 110

[142] Rong Lu, Jan J. Koenderink, and Astrid M.L. Kappers. Specularities on surfaces with tangential hairs or grooves. In *Proceedings of the International Conference on Computer Vision*, pages 839–846. IEEE, September 1999. cited on page(s) 149

[143] Nadia Magnenat-Thalmann. Photorealistic hair modeling, animation, and rendering. In *Course notes of SIGGRAPH conference*. ACM, 2003. cited on page(s) 133

[144] Nadia Magnenat-Thalmann, Sunil Hadap, and Prem Kalra. State of the art in hair simulation. In *Proceedings of International Workshop on Human Modeling and Animation*, pages 3–9. Korea Computer Graphics Society, June 2002. cited on page(s) 133

[145] Stephen R. Marschner and Donald P. Greenberg. Inverse lighting for photography. In *Proceedings of the Color Imaging conference*, pages 262–265. IS&T/SID, 1997. cited on page(s) 99

[146] Stephen R. Marschner, Brian Guenter, and Sashi Raghupathy. Modeling and rendering for realistic facial animation. In *Proceedings of the Eurographics Workshop on Rendering*, 2000. cited on page(s) 100, 101

[147] Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 131, 146, 148, 149, 150, 154, 207

[148] Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. Image-based BRDF measurement including human skin. In *Proceedings of the Eurographics Workshop on Rendering*, pages 139–152, 1999. cited on page(s) 94, 95, 98, 100, 126

[149] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the Eurographics Workshop on Rendering*, 2001. cited on page(s) 18

[150] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steve J. Gortel, and Leonard McMillan. Image-based visual hulls. In *Proceedings of the SIGGRAPH conference*. ACM, 2001. cited on page(s) 18

[151] Wojciech Matusik and Hanspeter Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 165, 203

[152] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 98

[153] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic BRDF measurement. In *Proceedings of the Eurographics Symposium on Rendering*, 2003. cited on page(s) 94

[154] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3D photography using opacity hulls. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 94, 95, 133, 165, 203, 206, 207

[155] Wojciech Matusik, Hanspeter Pfister, Remo Ziegler, Addy Ngan, and Leonard McMillan. Acquisition and rendering of transparent and refractive objects. *journal on Rendering Techniques*, 2002. Proceedings of the Eurographics Workshop on Rendering. cited on page(s) 94, 133

[156] David K. McAllister. *A Generalized Surface Appearence Representation for Computer Graphics*. PhD thesis, University of North Carolina, 2002. cited on page(s) 94

[157] David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. In *Proceedings of the conference on Graphics Hardware*. ACM SIGGRAPH/Eurographics, 2002. cited on page(s) 102

[158] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12), 2001. cited on page(s) 140

[159] Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and Hans-Peter Seidel. Efficient rendering of local subsurface scattering. In *Proceedings of the Pacific Graphics conference*, 2003. cited on page(s) 103, 113

[160] Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and Hans-Peter Seidel. Interactive rendering of translucent deformable objects. In *Proceedings of the Eurographics Symposium on Rendering*, 2003. cited on page(s) 103

[161] Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. Level set surface editing operators. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 32, 56

[162] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[163] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[164] Fred E. Nicodemus, Joseph C. Richmond, Jack J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, National Bureau of Standards, 1977. cited on page(s) 93

[165] Ko Nishino, Zhengyou Zhang, and Katsushi Ikeuchi. Determining reflectance parameters and illumination distribution from a sparse set of images for view-dependent image synthesis. In *Proceedings of the International Conference on Computer Vision*, pages 599–606. IEEE, 2001. cited on page(s) 99

[166] Yuichi Ohta and Takeo Kanade. Stereo by intra- and interscanline using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985. cited on page(s) 22

[167] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993. cited on page(s) 22, 24, 30

[168] Michael Oren and Shree K. Nayar. Generalization of Lambert's reflectance model. In *Proceedings of the SIGGRAPH conference*, pages 239–246. ACM, 1994. cited on page(s) 97

[169] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988. cited on page(s) 23, 38

[170] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[171] Bui-Tuong Phong. Illumination for computer generated images. *Communication of the ACM*, 18(6):311–317, June 1975. cited on page(s) 96, 114

[172] Eric Plante, Marie-Paule Cani, and Pierre Poulin. A layered wisp model for simulating interactions inside long hair. In *Proceedings of the Eurographics workshop on Computer Animation and Simulation*, 2001. cited on page(s) 131

[173] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc Van Gool. Automated reconstruction of 3D scenes from sequences of images. *ISPRS Journal Of Photogrammetry And Remote Sensing*, 55(4):251–267, 2000. cited on page(s) 22

[174] Pierre Poulin, Marc Stamminger, François Duranleau, Marie-Claude Frasson, and George Drettakis. Interactive point-based modeling of complex objects from images. In *Proceedings of the Graphics Interface conference*, June 2003. cited on page(s) 31

[175] Simon Premoze. Analytic approximations for light transport in volumetric materials. In *Proceedings of the Pacific Graphics conference*, 2002. cited on page(s) 97

[176] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the SIGGRAPH conference*. ACM, 2001. cited on page(s) 103

[177] Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 103, 109

[178] Alex Reche, Ignacio Martin, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference. cited on page(s) 164, 202

[179] Sébastien Roy. Stereo without epipolar lines: A maximum-flow formulation. *International Journal of Computer Vision*, 34(2/3):147–162, August 1999. cited on page(s) 28, 38, 45, 72, 76

[180] Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proceedings of the International Conference on Computer Vision*, pages 492–499. IEEE, January 1998. cited on page(s) 27, 38, 72

[181] Holly E. Rushmeier. 3D capture for computer graphics. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, 2001. cited on page(s) 133

[182] Holly E. Rushmeier and Fausto Bernardini. Computing consistent normals and colors from photometric data. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 99–108. IEEE, 1999. cited on page(s) 94

[183] Holly E. Rushmeier, Fausto Bernardini, Joshua Mittleman, and Gabriel Taubin. Acquiring input for rendering at appropriate levels of detail: Digitizing a pietà. In *Proceedings of the Eurographics Workshop on Rendering*, pages 81–92, 1998. cited on page(s) 101

[184] Holly E. Rushmeier, Bernice E. Rogowitz, and Christine Piatko. Perceptual issues in substituing texture for geometry. In *Proceedings of the conference on Human Vision and Electronic Imaging*, volume 3959, pages 372–383. SPIE, 2000. cited on page(s) 101

[185] Hideo Saito and Takeo Kanade. Shape reconstruction in projective grid space from large number of images. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, June 1999. cited on page(s) 52

[186] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Proceedings of the SIGGRAPH conference*. ACM, 1997. cited on page(s) 99

[187] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002. cited on page(s) 33

[188] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, volume 1, pages 195–202. IEEE, June 2003. cited on page(s) 33

[189] Alexander Schrijver. On the history of the transportation and maximum flows problems. `http://homepages.cwi.nl/~lex/files/histtrpclean.ps`. cited on page(s) 40

[190] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1067–1073. IEEE, 1997. cited on page(s) 9, 19, 21, 51

[191] James A. Sethian. *Level Set Methods and Fast Marching Methods (Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science)*. Cambridge University Press, 1999. cited on page(s) 37, 38

[192] Steven A. Shafer. Using color to separate reflection components. *Color Resolution Applications*, 10(4):210–218, 1985. cited on page(s) 116

[193] Ying Shan, Zicheng Liu, and Zhengyou Zhang. Model-based bundle adjustment with application to face modeling. In *Proceedings of the International Conference on Computer Vision*, pages 644–651. IEEE, 2001. cited on page(s) 12, 127

[194] Jun Shen and Serge Castan. An optimal linear operator for edge detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, 1986. cited on page(s) 142

[195] Gregory G. Slabaugh, Thomas Malzbender, and W. Bruce Culbertson. Volumetric warping for voxel coloring on an infinite domain. In *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large Scale Environments*, Lecture Notes on Computer Science, pages 109–123. Springer Verlag, July 2000. cited on page(s) 19, 52

[196] Gregory G. Slabaugh, Thomas Malzbender, W. Bruce Culbertson, and Ron Schafer. Improved voxel coloring via volumetric optimization. Technical Report 3, Center for Signal and Image Processing, 2000. cited on page(s) 19

[197] Gregory G. Slabaugh, Ronald W. Schafer, and Mat C. Hans. Multi-resolution space carving using level sets methods. In *Proceedings of the International Conference on Image Processing*, 2002. cited on page(s) 24

[198] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[199] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 103, 109

[200] Peter-Pike Sloan, Xinguo Liu, Heung-Yeung Shum, and John Snyder. Bi-scale radiance transfer. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 74, 103

[201] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, June 2000. cited on page(s) 21, 32

[202] Jos Stam. An illumination model for a skin layer bounded by rough surfaces. In *Proceedings of the Eurographics Workshop on Rendering*, June 2001. cited on page(s) 97

[203] Marc Stamminger, Jörg Haber, Hartmut Schirmacher, and Hans-Peter Seidel. Walkthroughs with corrective texturing. In *Proceedings of the Eurographics Workshop on Rendering*, 2000. cited on page(s) 102

[204] Mark R. Stevens, W. Bruce Culbertson, and Thomas Malzbender. A histogram-based color consistency test for voxel coloring. In *Proceedings of the International Conference on Pattern Recognition*, August 2002. cited on page(s) 11

[205] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, 1999. cited on page(s) 20, 52

[206] Richard Szeliski and Daniel Scharstein. Sampling the disparity space image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):419–425, March 2004. cited on page(s) 11

[207] Richard Szeliski and Richard Weiss. Robust shape recovery from occluding contours using a linear smoother. In *Proceedings of the conference on Computer Vision and Pattern Recognition*. IEEE, June 1993. cited on page(s) 18

[208] Marshall F. Tappen, William T. Freeman, and Edward H. Adelson. Recovering intrinsic images from a single image. In *Processing of the conference on Neural Information Processing Systems*, 2002. cited on page(s) 71

[209] Marco Tarini, Hitoshi Yamauchi, Jörg Haber, and Hans-Peter Seidel. Texturing faces. In *Proceedings of the Graphics Interface conference*, 2002. cited on page(s) 101, 113

[210] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, 1988. cited on page(s) 37, 38

[211] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the International Conference on Computer Vision*, pages 839–846. IEEE, 1998. cited on page(s) 145

[212] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference. cited on page(s) 103

[213] Kenneth E. Torrance and Ephraim M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of Optical Society of America*, 1967. cited on page(s) 97, 100

[214] Adrien Treuille, Aaron Hertzmann, and Steve M. Seitz. Example-based stereo with general BRDFs. In *Proceedings of the European Conference on Computer Vision*, 2004. cited on page(s) 32, 74

[215] David Tschumperlé and Rachid Deriche. Orthonormal vector sets regularization with PDE's and applications. *International Journal on Computer Vision*, 50:237–252, 12 2002. cited on page(s) 145

[216] Norimichi Tsumura, Nobutoshi Ojima, Kayoko Sato, Mitsuhiro Shiraishi, Hideto Shimizu, Hirohide Nabeshima, Syuuichi Akazaki, Kimihiko Hori, and Yoichi Miyake. Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 97

[217] Morgan Ulvklo, Hans Knutsson, and Gösta Granlund. Depth segmentation and occluded scene reconstruction using ego-motion. In *Proceedings of the Conference on Visual Information Processing*, pages 112–123. SPIE, April 1998. cited on page(s) 22, 55

[218] Régis Vaillant and Olivier Faugeras. Using extremal boundaries for 3-D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):157–173, 1992. cited on page(s) 18

[219] Olga Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, August 1999. cited on page(s) 28, 29, 38, 46, 58, 71

[220] Lifeng Wang, Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. View-dependent displacement mapping. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 74, 103

[221] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the SIGGRAPH conference*, pages 265–272. ACM, 1992. cited on page(s) 2, 94, 98, 182

[222] Geoffrey S. Watson. *Statistics on spheres*. John Wiley and Sons, 1983. cited on page(s) 145

[223] Yair Weiss. Deriving intrinsic images from image sequences. In *Proceedings of International Conference on Computer Vision*. IEEE, 2001. cited on page(s) 71

[224] Eric W. Weisstein et al. Banach space. MathWorld – A Wolfram Web Resource . `http://mathworld.wolfram.com/BanachSpace.html`. cited on page(s) 168

[225] Eric W. Weisstein et al. Green's theorem. MathWorld – A Wolfram Web Resource . `http://mathworld.wolfram.com/GreensTheorem.html`. cited on page(s) 170, 173

[226] Eric W. Weisstein et al. Level set. MathWorld – A Wolfram Web Resource . `http://mathworld.wolfram.com/LevelSet.html`. cited on page(s) 172

[227] Zhen Wen, Zicheng Liu, and Thomas S. Huang. Face relighting with radiance environment maps. In *Proceedings of the conference on Computer Vision and Pattern Recognition*. IEEE, 2003. cited on page(s) 102, 103

[228] Marta Wilczkowiak. *3D Modelling From Images Using Geometric Constraints*. PhD thesis, Institut National Polytechnique de Grenoble, 2004. cited on page(s) 12

[229] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the SIGGRAPH conference*, pages 270–274. ACM, 1978. cited on page(s) 123

[230] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference. cited on page(s) 5

[231] Günther Wyszecki and Walter S. Stiles. *Color science: Concepts and methods, quantitative data and formulae*. John Wiley and Sons, 1982. cited on page(s) 122

[232] Stephan Würmlin, Edouard Lamboray, Oliver G. Staadt, and Markus H. Gross. 3D video recorder. In *Proceedings of Pacific Graphics conference*, 2002. cited on page(s) 165, 203

[233] Ruigang Yang, Marc Pollefeys, and Greg Welch. Dealing with textureless regions and specular highlights – a progressive space carving scheme using a novel photo-consistency measure. In *Proceedings of International Conference on Computer Vision*. IEEE, October 2003. cited on page(s) 32, 73, 74, 132

[234] Anthony Yezzi, Greg Slabaugh, Adrian Broadhurst, Roberto Cipolla, and Ron Schafer. A surface evolution approach to probabilistic space carving. In *Proceedings of the International Symposium on 3D Processing, Visualization, and Transmission*, 2002. cited on page(s) 24

[235] Yitzhak Yitzhaky and Eli Peli. A method for objective edge detection, evaluation and detector parameter selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8), 2003. cited on page(s) 141

[236] Gang Zeng, Sylvain Paris, Long Quan, and Maxime Lhuillier. Surface reconstruction by propagating 3d stereo data in multiple 2d images. In *Proceedings of the European Conference on Computer Vision*, 2004. cited on page(s) 75

[237] Gang Zeng and Long Quan. Silhouette extraction from multiple images of an unknown background. In *Proceedings of the Asian Conference of Computer Vision*, 2004. cited on page(s) 18, 70, 73, 86

[238] Li Zhang, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *Proceedings of the International Conference on Computer Vision*. IEEE, 2003. cited on page(s) 32, 73, 74

[239] Zhengyou Zhang, Zicheng Liu, Dennis Adler, Michael Cohen, Erik Hanson, and Ying Shan. Cloning your own face with a desktop camera. In *Proceedings of the International Conference on Computer Vision*. IEEE, 2001. cited on page(s) 127

[240] Remo Ziegler, Wojciech Matusik, Hanspeter Pfister, and Leonard McMillan. 3D reconstruction using labeled image regions. In *Proceedings of the Eurographics Symposium on Geometry Processing*, 2003. cited on page(s) 21

[241] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998. cited on page(s) 140

This dissertation focuses on the creation of the information used in Computer Graphics. We especially concentrate on the data needed to render images. We do not ask the user to work directly on the object modeling but we rely on her to provide one or several images of the targeted object. These pictures are then automatically analyzed to extract the sought data. From this approach, we expect data more faithful to the original object and a shorter creation time for the user.

Our work is centered on three case studies that have useful applications. We first recover the surface of a matte object from a short image sequence whose viewpoint is moving. We then show how the appearance of a human face is retrieved from a single image and how the extracted data are used to render the original face under a new lighting environment. We end with a technique to capture the hair geometry using multiple images from a fixed viewpoint and a moving light.

We introduce several theoretical and technical contributions that enhance both precision and robustness of the capture. Results are provided to illustrate these improvements.

Le sujet de cette thèse porte sur la création des données utilisées en informatique graphique pour synthétiser des images. On ne demande pas à l'utilisateur de modéliser l'objet souhaité mais plutôt d'en fournir une ou plusieurs photographies. Ces images sont automatiquement analysées pour en extraire l'information recherchée. On attend de cette approche des données plus fidèles à l'original et un temps de création plus court pour l'utilisateur.

Nos travaux sont centrés sur trois cas d'études qui mènent à des applications utiles. Tout d'abord, nous reconstruisons la surface d'un objet matte à partir d'une séquence d'images dont le point de vue se déplace. Nous capturons ensuite l'apparence d'un visage à partir d'une seule image et montrons comment les données récupérées sont utilisées pour synthétiser ce visage sous un nouvel éclairage. Nous terminons avec la capture de la géométrie d'une chevelure à partir de plusieurs images prises avec une caméra fixe et une lumière qui se déplace.

Nous introduisons plusieurs contributions théoriques et techniques qui améliorent aussi bien la précision que la robustesse de la capture. Des résultats illustrent ces améliorations.