Robust Acquisition of 3D Informations from Short Image Sequences

Sylvain Paris François Sillion

*i*MAGIS - GRAVIR / IMAG - INRIA †

Abstract

This paper addresses the problem of 3D reconstruction from a set of viewpoints on a short baseline. Its main contribution is the development of a robust algorithm which can extract 3D informations from a set of images taken with a very small baseline, even in the presence of significant occlusion. To achieve this goal, we use a multi-pass process that works layer by layer. In each pass we begin with a "space carving" step which is made robust through some morphological operations. Then we introduce an original technique to solve the ambiguity inherent to "space carving" in the case of short baseline. Once we have a voxel representation of the objects, we propose a method based on differential geometry to build a smooth mesh. Results are presented, demonstrating the efficiency and usefulness of the method.

Keywords : structure from motion, 3D reconstruction, voxel space, dynamic programming, mathematical morphology, partial differential equation filter, occlusion.

1. Introduction

We aim at acquiring 3D information from image sequences such as a short video segment. Our goal is to use this information to add realistic details in virtual environments. For instance, we want to populate a street with passer-bys, post-boxes and other urban elements. These are not the focus of interest of the scene, but rather the small add-on that makes the whole street "alive". Therefore it is not important if the objects are not "complete" for we do not need to look at them very closely nor to see what is behind a post-box. The important point is the realistic look of the result. Moreover we do not want to be restricted to a very specific kind of image sequences - for instance, there may be an interesting object in a movie and then we have no other choice than to deal with that given video segment. Therefore we must cope:

- With viewpoints that are spatially close (we are in the case of *short baseline*) and almost aligned because we cannot assume the image sequence is long nor that it goes all around the object,
- With ordinary images for they can be noisy and we cannot assume there is a "blue screen" in the background,
- And with occlusions because there can be another object that hides the one we are interested in.

We assume that we know all the camera parameters because acquiring them from the images is a full separate problem [20], and that there is no movement in the sequence (acquiring 3D information and movement is a much more complex problem). So we can work from two different types of sequence: either the scene is static and the camera is moving (e.g. for acquiring a statue or post-box), or we have a set of synchronized cameras which shoot simultaneous images [29]. Finally we consider that the only user input is a bounding volume around the objects of interest.

Having defined our goals and the input, we can have an overview of the algorithm we propose. The main structuring point is a multi-pass strategy : we reconstruct the object information in a *front to back order* [23, 31]. During a pass, only the closest object still not reconstructed is taken into account. This guarantees all the occluders are known before we try to reconstruct an hidden object.

We need also to choose the 3D structure we use to represent the objects during the process. One way to do this is to use their contours as in [5, 18, 27, 32], but these methods are very dependent on the contour extraction algorithm that does not give satisfying results with noisy images and real background. We can also optimize a surface so it fits to the image contents using shading information as Fua [10]. This approach deals well with untextured objects but cannot cope with occlusions in the general case. Faugeras and Keriven [9] also use a surface-based approach driven by the level set technique, it works well even with occlusions and topologically complex objects but it requires textured surfaces. Since the working volume is bounded, we prefer a *voxel-based* approach as described in [3, 5, 13, 22, 23, 24, 25, 28] : we discretize the volume into

† i*MAGIS* is a joint research project of CNRS, INPG, INRIA, UJF. {Sylvain.Paris|Francois.Sillion}@imag.fr

small cubes (the *voxels*) and the goal then is to select the voxels that belong to the object volume. Thus we are not bound to a contour detection algorithm and we show in section 4 that we can work without texture. We explain quickly how this voxel space is built in section 2.

Using this voxel space, we can split each pass of the algorithm into three steps. We begin in section 3 with the selection of the *consistent* voxels as defined by Seitz [23]: we keep the only voxels that have the "same" color in all the images. This first step is quite similar to the *space carving* algorithm [13] during which we take special care to be robust to noise. Kutulakos [13] explains that there may be too many voxels left. In our specific configuration of short sequences, we show that this remark is crucial and that there is a very high number of useless consistent voxels and then, as a second step, we introduce in section 4 an original approach to solve this ambiguity. The third and last step of a pass consists in blocking the lines of sight intersecting the object we have just reconstructed. Section 5 explains this step and how to deal with occlusions.

When all passes are done, we have a voxel representation of the objects that we call *draft surface*. This discretized form is not suitable for further use because real objects are not composed of small cubes. Therefore we propose in section 6 a new technique based on morphological treatments and differential geometry to transform this representation into a smooth surface.

Figure 1 gives an overview of the whole algorithm.

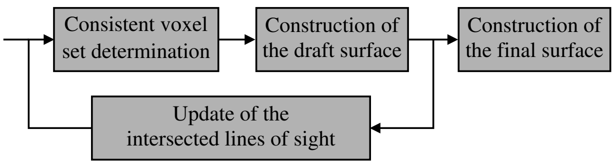

Figure 1. Overview of the whole algorithm.

We illustrate in section 7 the applicabilty of our algorithm with some results and propose some extensions.

2. Voxel space geometry

Since the volume of interest is bounded, we can discretize it into voxels. We choose to let their size vary so that they represent a fixed amount of information in image space. More precisely, we impose that all the voxels project to the same area in the camera image planes. This is called by Slabaugh [26] the *constant footprint property*.

In our specific configuration of aligned viewpoints, we build such a voxel space by introducing in the scene the "distance from the cameras". Geometrically, this distance $dist_{scene}(X)$ of a point X in the scene is defined as $d(\Delta_{cam}, X)$, where $d(x, y)$ is the euclidian distance between x and y and Δ_{cam} is the line on which the cameras lie. Then, as the perspective projection gives a resulting area inversely proportional to the depth, it is sufficient to make the size of a voxel v proportional to $dist_{scene}(v)$.

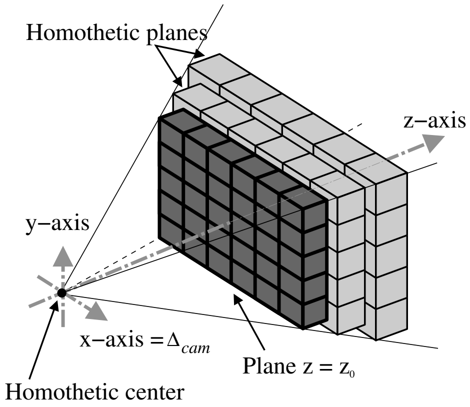

Figure 2. Construction of the voxel space using scaling.

In practice, we build the first voxel plane $z = z_0$ and then we build the other planes using a scaling whose center lies on the camera line (see figure 2). This assures the constant footprint property. Note that the global shape of the voxel space is characterised only by the the first plane and the position of the homothetic center. This construction is similar to the use of a *virtual camera* [28] with its center aligned with the input viewpoints. Saito [22] proposes another approach based only on the fundamental matrix of two views that reaches the same property but we will see in section 4 that our technique helps to define a very useful geometric property.

3. Identification of the consistent voxels

In the first step of the algorithm, we aim at reducing the space in which we are working. Actually the whole bounded space given as input is too large, typically involving tens of millions of voxels. The problem has to be alleviated, fewer voxels will allow more complex and more powerful treatments in the next steps of the algorithm.

The criterion we use here is the *voxel consistency* described in [13, 23]: a voxel is said *consistent* if its color is the same for all cameras that can see it. It is important to understand that it is a necessary but not sufficient condition. A voxel in the final scene must be consistent but a consistent voxel may not be in the final scene. For instance, if we place all the cameras in front of a red wall, each voxel is seen red by all the cameras, then all voxels are consistent although only the few that represent the wall should be in the final scene.

For now, as suggested in [13, 21, 23] we use the standard deviation of the colors seen by the cameras to evaluate the consistency. It restricts the algorithm to almost lambertian objects because it implies that the components of the voxel color are the same in all images and hence do not depend of the viewpoint. But we will show that we can still achieve satisfying results with real images even with this hypotheses. Nevertheless it is clear that there is room for improvement using better statistics.

In practice, for each voxel, we proceed as follow :

- We suppose it is the only voxel visible in the scene,
- We compute a color for each camera corresponding to the color of the pixel through which the camera sees the voxel,
- We compute the standard deviation s of the color set we have computed. In practice, we use the *hue, saturation, value* space but this choice is not critical (we only need an estimation of the distance between two colors to measure the color similarity).

Then we assume a voxel is consistent if $s < s_t$ for some threshold s_t.

At this point, we have built the subset of consistent voxels. All other voxels are inconsistent, for different cameras see them of different colors. In the following steps of our pass, we no longer need to take these voxels into account anymore.

4. Construction of a draft surface

As said before, the voxel consistency is not a sufficient condition to be part of the final reconstruction, it is only necessary. Hence, inside the subset we have just built, there are still too many voxels. The key situation in which unwanted but consistent voxels appear is around a surface of uniform color. If we look at a voxel just in front of such a surface, the lines of sight of all the cameras hit the surface,

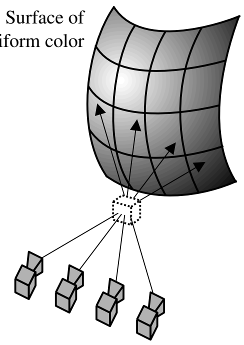

Figure 3. A voxel in front of an uniform color surface.

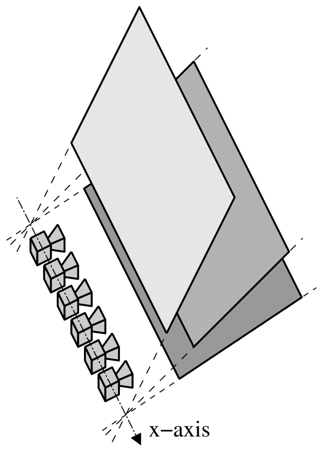

Figure 4. Epipolar planes.

then all the cameras see the same color and therefore the voxel is consistent, although it is not part of the surface (see figure 3).

To study this configuration, we observe that we can consider *epipolar planes*. In classical stereo-vision, epipolar planes are the planes which contain the two optical centers. In our case, as the cameras are aligned, we can straightforwardly extend the notion of epipolar planes to the planes which contain all the optical centers (figure 4).

These planes are of great help to describe the geometric configuration for they correspond to voxel planes in the voxel space if we choose the x-axis equal to the axis on which the cameras lie (figures 2 and 4). This is a consequence of the scaling-based construction.

Then, in these planes, the consistent voxels generated by a surface of uniform color form a quadrilateral. To show this, we only need to consider the two extreme cameras as in figure 5.

This quadrilateral has the following *vertex coordinate property*. In the plane homothetic coordinate system, each vertex has one and only one extremal coordinate. This property is satisfied if and only if the homothetic center lies between the two extremal cameras. This results from the signs of the oriented angles between the lines of sight and the boundaries of the voxel space: if we consider the two angles on the left side, l_+ and l_-, they are of opposite signs. This is the same configuration on the right side with $r_+$and r_- (figure 6).

We introduce some notations to refer to these points. We call the most distant vertex the *up* vertex (U), the most left vertex the *left* one (L), and so forth with the *down* one (D) and the *right* one (R). See figure 5.

Thanks to this property, we can now recognize the quadrilaterals formed by surfaces of uniform color. We begin to search the most left consistent voxel in a plane. It is a L vertex. From that voxel, we find the U and D vertices by following the edges of the quadrilateral and then we find R the same way. We mark all the voxels in the quadrilateral as *processed* and we iterate to find another L.

However, because of noise, the above process cannot be applied directly on the subset of consistent voxels. In fact, there are many unwanted voxels in the subset : some voxels can be isolated or there may be some small holes inside the quadrilaterals. To make more robust the quadrilateral recognition, we must perform a pre-treatment on the subset. We chose a morphological filter : we first apply a morphological closure on the subset with a cube as structuring element, followed by an opening. The closure ensures that there is not any small hole and that we do not lose small quadrilaterals between large ones. The opening removes isolated voxels. And to finish, we eliminate small groups of voxels in the subset (using a simple threshold on the number of voxels in a connected group).

Once we have achieved this denoising process, we can run properly our quadrilateral recognition algorithm.

Computation of the intersection of the surface and the epipolar plane

Now we have characterized the quadrilaterals inside which we have to find the intersection of the surface of uniform color with the epipolar plane. Let's only consider a single quadrilateral with vertices U, R, D and L.

If we observe figure 5, vertices L and R are the extreme points of the intersection we are looking for. This is the starting point of our method : we search for a line whose extreme points are L and R.

Another constraint is quite natural : the voxels of the line should maximize the consistency or equivalently minimize the standard deviation s.

The last constraint comes from the fact that the color of the surface is uniform. Since we are able to see the scene,

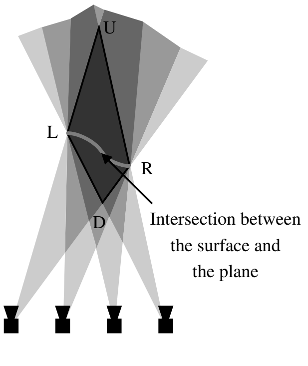

Figure 5. Quadrilateral formed in an epipolar plane by a surface of uniform color. Edges are only due to the two extreme cameras.

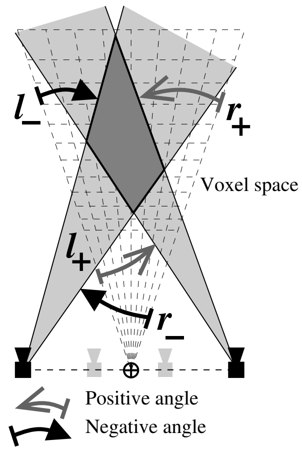

Figure 6. Angles between the lines of sight and the boundaries of the voxel space.

there is some light source somewhere in the scene and therefore there can be some shading variations. The relatively uniform color of the surface therefore implies that the surface is quite smooth.

The classical way to deal with this set of constraints is a snake-like optimization [33]. We represent the line by a function $z = f(x)$ and we want to minimize the following energy :

$$\int_L^R \left[\alpha . s^2 \left(x, f(x) \right) + \beta . \left(\frac{df}{dx}(x) \right)^2 \right] dx \qquad (1)$$

where α and β control the smoothness of f.

We can solve this optimization problem with the classical technique of dynamic programming [2]. Similar approaches have been often used in computer vision. Our contribution is to work directly in the 3D space and not in the disparity space as [11, 12, 16] so we have a more robust localization against ambiguity as shown by Okutomi and Kanade [17]. We also use dynamic programming only in the regions known to represent a smooth surface, unlike other techniques [6, 7, 11, 12, 16, 17] that work with the whole scan-line and then need some special treatments to cope with occlusions or discontinuities. In addition we introduce the vertex coordinate property to characterize rigorously the search area.

Figure 7 shows an epipolar plane : on the left, the standard deviation is represented with grey levels (white for low values and black for high ones) and on the right, the result of the thresholding after the morphological treatment (in grey) and the line (in white) obtained by the snake optimization.

It is important to notice that this step where the surface of the objects is found inside the consistent voxels is crucial. If we only use the space carving algorithm [13] (or

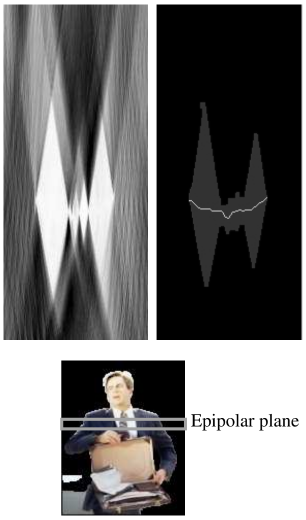

Figure 7. Views of an epipolar plane.

a variant [23, 24]) in the configuration of a short image sequence, satisfying results cannot be reached because of those quadrilaterals. Since consistency is the only criterion used to carve the voxels, none of the voxels inside the quadrilaterals are eliminated, resulting in large peaks in the uniform color areas.

We now have a collection of lines included in epipolar planes. The lines are the intersection of the model surface with the successive epipolar planes. The voxels in these lines are what we call the *draft surface* of the model.

5. Dealing with occlusions

The main point to deal with occlusions is to know at least approximately the objects that hide, the *occluders*, before reconstructing the objects that are hidden, the *occludees*. If we know this, we can evaluate the consistency of a given voxel by using only the cameras for which the given voxel is not occluded.

To achieve this goal, each pass reconstructs only the surface which is the closest to the cameras and still not reconstructed. This corresponds to the *front to back* order proposed in [23, 31]; it guarantees that in the first pass, we are reconstructing only surfaces for which there is no occlusion. And in the following pass, if there is an occluder, we are assured to have already its draft surface for it is closer to the cameras. Note this implies that the user has to include all the occluders in the volume of interest given as input.

In practice, we proceed as follow:

- We evaluate the voxel consistency with only the cameras of which line of sight is not intersected.

- In each epipolar plane we only reconstruct the nearest line corresponding to a surface-plane intersection.

- At the end of each pass, we mark for each camera which area in the image plane corresponds to the lines of sight that intersect the newly built voxel.

The last point to consider carefully is the position of the homothetic center. Since we do not use all the cameras to compute the voxel consistency, each voxel is "seen" by a subset of the cameras, and we want the homothetic center to always fall between the left most and right most cameras of this subset. However that may not be possible simultaneously for all voxels. The homothetic center may not lie between the two extreme cameras for all the voxels in a quadrilateral. This can result in one of the two following situations.

- The ideal case in which there exists a position for which the homothetic center lies between both extreme cameras for all voxels. Then we simply choose such a position as in figure 8.

- The degenerated case in which there is no such position. Then we compute a satisfying position for each voxel and we choose the mean of all these positions. Although we are not in the configuration that assures the vertex coordinate property to be true as demonstrated in section 4, the property is still valid in practice.

In both cases, we have moved the homothetic center to achieve the vertex coordinate property and therefore the quadrilaterals are still recognized properly.

We stop the iterations when a pass does not find anything to reconstruct. Then each pass has produced a piece of the draft surface (figure 9-a,b,c) and we merge all these pieces (figure 9-d).

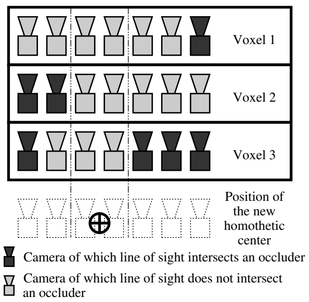

Figure 8. Choice of the new position of the homothetie center in the ideal case.

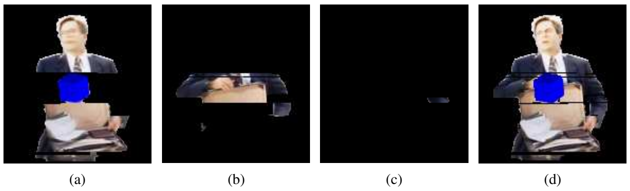

Figure 9. A sample reconstruction in three steps. We use a sequence of images in which a synthetic cube has been added to create a large occlusion. (a-b-c) The draft surfaces resulting from the three steps. (d) The union of all these surfaces.

6. Construction of the final surface

The draft surface is not a true surface : the voxels are only grouped in lines lying in epipolar planes, and do not form a continuous surface. There is no link between two successive lines. Therefore there can be holes between lines, strong discontinuities, etc... We need to include in our reconstruction algorithm some vertical connection between voxels. But we have to be careful : the final surface must be based on the information included in the draft one, we must preserve this information by using only soft filters.

Morphological operations

Firstly, we apply a morphological dilation along the z-axis. The dilation is purely “in depth” : we do not add any voxel in front of the voxels of the draft surface since we consider the draft surface to be visible from the cameras. We fix a threshold d_{hole} beyond which we consider that there is a hole between two successive lines. So the depth of the dilation corresponds exactly to d_{hole} (in figure 10, $d_{hole} = 3$).

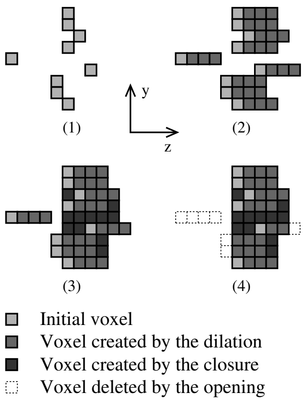

Figure 10. Morphological treatments on the draft surface.

Unfortunately, the process may have failed in some epipolar planes because of noise, resulting in holes, poorly located lines and so forth. We need to have an appropriate treatment to reinforce the process. Again we use an morphological closure and an opening. If we use a cubic structuring element as before, the action will be isotropic and we risk modifying the information inside the epipolar planes, although a lot of work has already been done to acquire this information. This action inside the plane would be an unwanted side effect. What we actually need is an anisotropic action only along the y-axis. Hence we use a purely vertical structuring element. Thus only the action between planes remains and the action internal to a plane disappears: the closure fills the holes due to missing lines and the opening deletes isolated lines. Figure 10 summarizes the morphological treatment used.

Creation of a mesh

We have now deep blocks of voxels which front represents the surface of the model. To create a mesh of the surface, we isolate blocks of voxels : once we have found a first voxel of in a block, we straightforwardly find the other voxels of the block by connexity. For a given block, we adapt a mesh using the positions of the front voxels as vertices.

Now the reconstruction is almost finished : the different objects are isolated, their surface is known. This allows to make a last and object-specific treatment.

Smoothing filter

The last treatment we perform on our model is a global filter on the whole surface against noise. The snake optimization coupled with the aliasing resulting from the alignment of the vertices on the voxel grid may have left some granularity on the surface. As we have already explained, what we are reconstructing should be quite smooth. Nevertheless, we do not want our final model to look like a gaussian filtered surface : that is to say perfectly smooth with no edge. The goal is then to smooth the surface without damaging the edges.

We choose an approach similar to that of Perona and Malik [19]: we apply an anisotropic filter whose strength is controlled by some criterion we want to preserve. Perona and Malik proposed a denoising filter for images and therefore preserved discontinuities in the image. They used the gradient norm to control their filter. We want to preserve edges on our surface, so we use the curvature to control our filter.

As Perona and Malik, we use a partial derivative equation (PDE) to define the filter. We need to model the surface by a function that can evolve in time t. A natural way to do this is to represent the surface by a height field

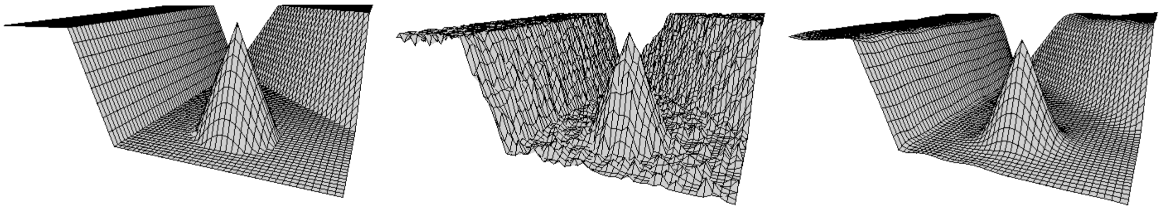

Figure 11. Illustration of the PDE filter.

$z = h(x, y, t)$. Then we exploit the ideas developed by Deriche and Faugeras [8]:

- We define the filter by two orthogonal directions : fortunately, the principal curvatures are orthogonal so we can easily use them.
- The control function expressed in these orthogonal axes has to stay between 0 and 1 as in [1] so as it tends toward 0 for high values of the parameter and then makes the filter have no effects. In our case this aims at preserving sharp edges.
- We compute the controlling parameter on a gaussian filtered surface such that only large features remain to control the filter as in [4].

Hence the equation we propose is :

$$\frac{\partial h}{\partial t} = f(\kappa_1^\sigma)\kappa_1 + f(\kappa_2^\sigma)\kappa_2 \qquad (2)$$

where :

- κ_1 and κ_2 are the principal curvatures of the surface at (x, y) (see [14] for details),
- κ_1^σ and κ_2^σ are the principal curvatures at (x, y) of the surface convolved by a gaussian function of standard deviation σ,
- f is a decreasing controlling function from $f(0) = 1$ to $f(+\infty) = 0$. For instance, we can use $x \mapsto \frac{1}{1+\alpha x^2}$.

We iterate this PDE filter so as to get a smooth surface but without affecting the sharp edges. Figure 11 shows a sample surface : we have a perfect noiseless surface (on the left) to which we have added some white noise up to 10% of the maximal height (in the middle) and on the right we have the filtered surface. The sharp features of the original surface have been pretty well preserved and especially, the action of the filter is independent of the grid orientation : all edges are sharp and not only the ones parallel to the grid principal directions.

It is important to remark that our goal is to affect the geometric properties of the surface seen as a 3D geometric entity. Our approach is different from [15, 30, 34] in which the surface is only an mathematical modelization of an image. In our case, it is in the same time easier and harder to cope with because on one hand, the amount of noise in the surface we get is far less important than the one that can be encountered in noisy images but on the other hand, noise is far more sensitive on a surface, a soft granularity on a surface is immediately caught whereas the equivalent noise in an image is almost not visible. In figure 12, we compare our filter to the one proposed by Alvarez [1] which is very efficient for images. We can see that in the case of a surface with a relatively limited noise, our filter achieves better results especially on edges, on the peak and the slopes.

After this operation, we have the surface of the object in the scene. It is smooth almost everywhere but it still conserves sharp edges.

7. Results and conclusions

To solve the 3D reconstruction problem, the algorithm described in this paper makes use of a number of classical techniques : the discretization of the space into voxels, the notion of voxel consistency, the use of epipolar planes or the dynamic programming approach. It also makes several contributions such as :

- The rigorous characterization of the situation in which the voxel consistency is not sufficient to find the surface of the model,
- The dynamic programming approach driven by this characterization and directly used in the 3D space to deal with untextured surface,
- The use of morphological treatments on the voxel space to cope with noisy data and obtain satisfying results even with real images,
- A specific PDE surface filter to achieve robustness and global consistency,

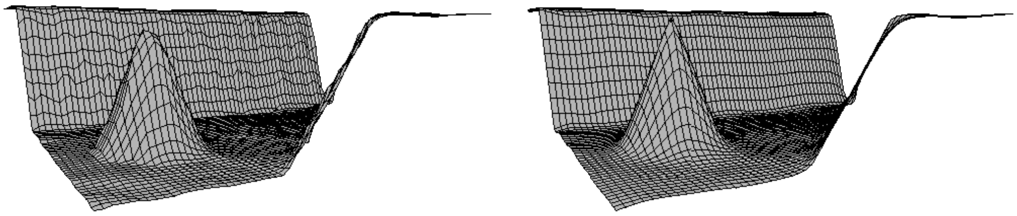

Figure 12. Comparison between the filter proposed by Alvarez [1] (left) and our surface filter (right).

- The adaptation of the "front to back" ordering to this algorithm to solve the occlusion problem.

These techniques make our algorithm able to acquire 3D informations from a set of real photographs even with some occlusions. Nevertheless the method is still quite slow, for instance the man with briefcase (figure 13-b,c , requires two passes, $140 \times 150 \times 330 \approx 7.10^6$ voxels) needs about 280 minutes to complete on a MIPS R12000 400MHz processor but the code is not fully optimized, could be parallelized and above all, during this time there is no need of the user, the only user input is a bounding volume. Then the process is robust enough and uses efficiently the redundancy in the images to achieve similar results with or without occlusion (see figures 13 and 14, 40 images with a 692×461 resolution on 1.5 meter wide line). And a very satisfying point is that it is able to achieve precise results even if the cameras have a very small view angle on the scene (see the top view on figure 14-e). This illustrates that we can use redundancy to compensate for small angle and occlusion and then achieve a robust and precise depth estimation. Note that the left shoulder of the running woman (figure 13-d,e) is not seen by any camera because the flying papers always hide it, so it is not reconstructed since we can deal with occlusions as long as they do not exist for some viewpoints. If something is never seen, the algorithm has no chance to "guess it". But for instance the head of the woman is hidden in half of the images and the algorithm is able to reach a satisfying results although the head resolution is only about 20×30 pixels.

On figure 15 (30 images with 400×300 resoltuion), we work from synthetic images so as to exactly know the objects in the scene : there are a sphere, a cone and a rectangular parallelepiped. We can see that the reconstruction gives the good geometric properties : we actually have a sphere, a cone and a parallelepiped as results although in the original sequence, the sphere is partially hidden by the cone and the parallelepiped. And if we add noise, the algorithm is robust enough to still achieve a valid reconstruction even if the noise intensity is high (figure 15-c,d). We have estimated the error on the sphere : we have measured the distance between the reconstructed points and the surface. The mean of these distances divided by the radius of the sphere is : 2.4% with no noise (figure 15-a), 2.9% with a noise with intensity 0.1 (figure 15-b) and 4.4% with a noise with intensity 0.2 (figure 15-c,d).

It is important to note that this process is fairly driven from a very limited a priori knowledge of the observed scene. Moreover it is able to cope with textured and untextured objects.

Future work

This algorithm gives satisfying results but we believe it can be still improved. We will continue to work on the voxel consistency estimation to get some more precise and more powerful expression on which we can base the process. Then we want to find a more efficient way to find the surface inside the set of the consistent voxels.

From a more global point of view, we aim at characterizing the arbitrary thresholds that appear during the process. Obviously almost all thresholds depend more or less directly on the resolution and on the scale of the reconstruction. And the most challenging work is to extend the algorithm to deal with specular surfaces and any set of cameras.

References

[1] L. Alvarez, P.-L. Lions, and J.-M. Morel. Image selective smoothing and edge detection by nonlinear diffusion. II. *SIAM Journal on Numerical Analysis*, 29(3):845–866, June 1992.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] A. C. Brik, J. Williams, and J. J. Connor. Multiresolution, incremental generation of 3D computer models from video data. In *SMA '93: Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 95–102. ACM, May 1993. held May 19-21, 1993 in Montreal, Quebec, Canada.

[4] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 29(1):182–193, Feb. 1992.

[5] R. Collins. A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition Conf.*, pages 358–363, 1996.

[6] I. J. Cox. A maximum likelihood n-camera stereo algorithm. In I. C. Society, editor, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 733–739, June 1994.

[7] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding: CVIU*, 63(3):542–567, 1996.

[8] R. Deriche and O. Faugeras. Les EDP en traitement des images et vision par ordinateur. Technical Report RR-2697, Inria, Institut National de Recherche en Informatique et en Automatique.

[9] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 1998.

[10] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, September 1995.

[11] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. In J.-O. Eklundh, editor, *European Conference on Computer Vision*, pages 179–186, Vol B, Stockholm, Sweden, May 1994. Springer-Verlag.

[12] R. Koch, M. Pollefeys, and L. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. *Lecture Notes in Computer Science*, 1406:55–??, 1998.

[13] K. Kutulakos and S. Seitz. A theory of shape by space carving. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 307–314, Los Alamitos, CA, Sept. 20–27 1999. IEEE.

[14] M. Lipshutz. Differential geometry, 1969.

[15] P. S. A. H. B. Mathieu Desbrun, Mark Meyer. Anisotropic feature-preserving denoising of bivariate data. In *Graphics Interface '00 Proceedings*, 2000.

[16] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:139–154, 1985.

[17] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–63, 1993.

[18] K.-Y. K. W. Paulo R. S. Mendonça and R. Cipolla. Camera pose estimation and reconstruction from imag profiles under circular motion. In *ECCV 2000 Proceedings*, volume 2, pages 864–878, 2000.

[19] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(7):629–639, July 1990.

[20] A. Z. Richard Hartley. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[21] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *IEEE International Conference on Computer Vision*, pages 492–499, 1998.

[22] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 49–54, Los Alamitos, June 23–25 1999. IEEE.

[23] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. 35(2):151–173, 1999.

[24] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. Improved voxel coloring via volumetric optimization. Technical report, Center for Signal and Image Processing, Georgia Institute of Technology, 2000.

[25] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In K. Mueller and A. Kaufmann, editors, *Proceedings of the Joint IEEE TCVG and Eurographics Workshop (VolumeGraphics-01)*, pages 81–100, Wien, June 21–22 2001. Springer-Verlag.

[26] G. Slabaugh, T. Malzbender, and W. B. Culbertson. Volumetric warping for voxel coloring on an infinite domain. In M. Pollefeys, L. V. Gool, A. Zisserman, and A. Fitzgibbon, editors, *3D Structure from Images - SMILE 2000*, number 2018 in LNCS, pages 109–123, Dublin, Ireland, July 2000. Springer-Verlag.

[27] Szeliski and Weiss. Robust shape recovery from occluding contours using a linear smoother. Technical Report DEC-CRL-93-7, Digital Equipment Corporation, Cambridge Research Lab, 93.

[28] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In S. Chandran and U. Desai, editors, *Procedings of the Sixth International Conference on Computer Vision (ICCV-98)*, pages 517–526, New Delhi, Jan. 4–7 1998. Narosa Publishing House.

[29] D. Taylor. http://www.virtualcamera.com.

[30] J. Tumblin and G. Turk. LCIS: A boundary hierarchy for detail-preserving contrast reduction. In A. Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, Annual Conference Series, pages 83–90, Los Angeles, 1999. ACM Siggraph, Addison Wesley Longman.

[31] H. G.-G. H. Ulvklo, Morgan; Knutsson. Depth segmentation and occluded scene reconstruction using ego-motion. *Proc. SPIE Vol. 3387, p. 112-123, Visual Information Processing VII, Stephen K. Park; Richard D. Juday; Eds*, 1998.

[32] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3-D object modeling. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):157–173, Feb. 1992.

[33] G. S. Vicent Caselles, Ron Kimmel. Geodesic active contours. *International Journal of Computer Vision*, 1997.

[34] M. K. Yu-Li You. Fourth order partial differential equations for noise removal. *IEEE Transactions on Image Processing*, 9(10), october 2000.

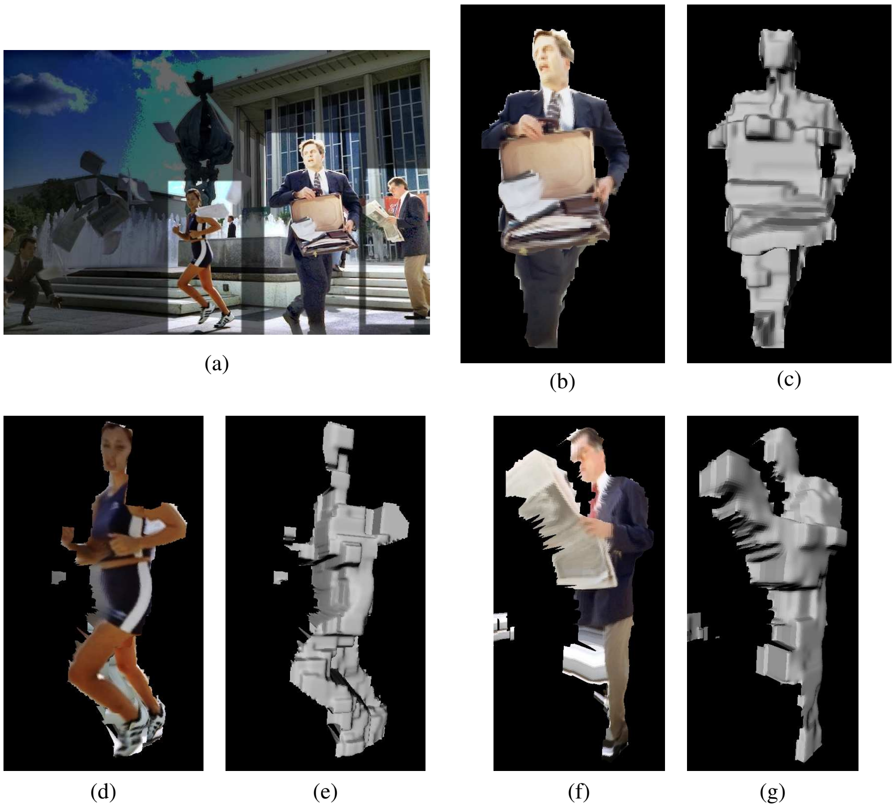

Figure 13. Results from images taken simultaneously [29]. (a) an image with the acquired areas (b-c) the man on the left ($\approx$ 5m from the cameras) (d-e) the woman ($\approx$ 6.5m) (f-g) the man on the right.

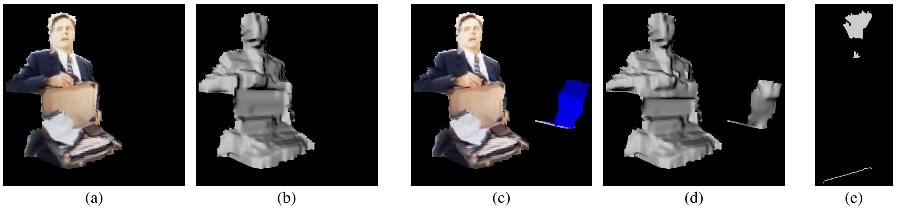

Figure 14. Illustration of the robustness to occlusion. (a-b) is obtained from the original sequence (c-d) is obtained fr om the same sequence in which a synthetic cube has been added to create a large occlusion (e) top view of the scene, from the bottom to the top : the cameras, the cube and the man.

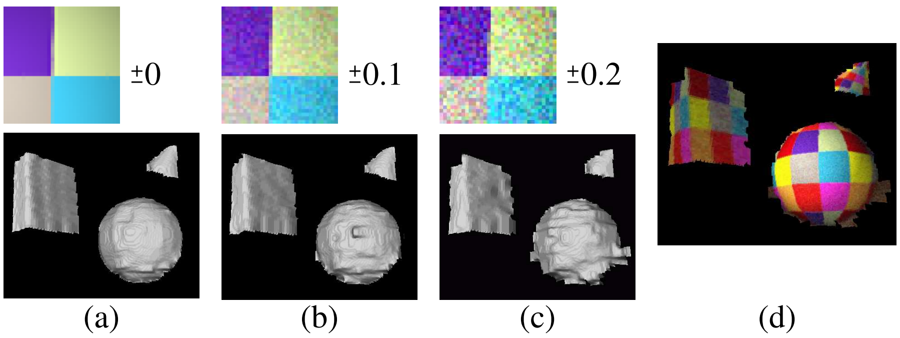

Figure 15. Reconstruction from a set of synthetic images where white noise has been added to the input images. The top raw shows a sample of th input images : noise has been added on each RGB component on a scale between 0 and 1, its maximum intensity is (a) 0 (b) ± 0.1 (c-d) ± 0.2.