Eurographics Symposium on Virtual Environments (2004)
S. Coquillart, M. Göbel (Editors)

A Real-Time System for Full Body Interaction with Virtual Worlds

J-M. Hasenfratz and M. Lapierre and F. Sillion

ARTIS†, GRAVIR/IMAG - INRIA

Abstract

Real-time video acquisition is becoming a reality with the most recent camera technology. Three-dimensional models can be reconstructed from multiple views using visual hull carving techniques. However the combination of these approaches to obtain a moving 3D model from simultaneous video captures remains a technological challenge. In this paper we demonstrate a complete system architecture allowing the real-time ($\geq$ 30 fps) acquisition and full-body reconstruction of one or several actors, which can then be integrated in a virtual environment. A volume of approximately $2m^3$ is observed with (at least) four video cameras and the video fluxes are processed to obtain a volumetric model of the moving actors. The reconstruction process uses a mixture of pipelined and parallel processing, using N individual PCs for N cameras and a central computer for integration, reconstruction and display. A surface description is obtained using a marching cubes algorithm. We discuss the overall architecture choices, with particular emphasis on the real-time constraint and latency issues, and demonstrate that a software synchronization of the video fluxes is both sufficient and efficient. The ability to reconstruct a full-body model of the actors and any additional props or elements opens the way for very natural interaction techniques using the entire body and real elements manipulated by the user, whose avatar is immersed in a virtual world.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation – Bitmap and framebuffer operations I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Virtual reality I.4.8 [Image Processing and Computer Vision]: Scene Analysis – Tracking

1. Introduction

Inserting live-action movement in virtual worlds is a requirement for many applications. In the context of the CYBER project† we focus on the capture of live movements in real time, for the incrustation of the acquired actors and objects in a virtual world. This operation is important for instance in the television industry, for virtual sets and online presentation by a real person; for games, in order to insert a real person in the game world; for education and entertainment, to allow visits and presentation of remote places.

In this paper, we present a real-time system allowing full body interaction with the virtual world. The novelty is a fully real-time system (min. 25 frames per second) providing a 3D model of the body of an actor (and additional objects) which can be used for interaction and other calculations. Such a 3D model is useful to allow fully 3D interaction using body parts, natural gestures or additional props and objects manipulated by the user.

The paper is organized as follows: Section 2 presents the previous work in trying to capture and reconstruct an actor in real-time. In Section 3, we present the hardware and software architectures used in our system. Section 4 is devoted to the real-time reconstruction of the actor. Section 5 proposes particular applications to demonstrate possible interaction with the virtual environment. Finally, results are discussed in Section 6 before concluding and discussing possible future work.

† ARTIS is a research project in the GRAVIR/IMAG laboratory, a joint unit of CNRS, INPG, INRIA and UJF.
† http://www-artis.imag.fr/CYBER

2. Previous Work

In the last few years, several real-time systems have been presented to capture the 3D shape of a dynamic object, typically a person. We discuss these approaches by grouping them according to their main goal: reconstructing a 3D shape (as fast as possible), rendering a 3D shape, or replaying a 3D sequence. We first observe that performance can be characterized by the acquisition rate and the image display rate, possibly two very different values.

Borovikov and Davis[BD00] use a system with 16 groups of 4 cameras (3 black and white and one color) to represent a moving actor by an octree of voxels. With a cluster of 16 PCs they obtain a maximal processing rate of 10Hz. Wu and Matsuyama[WM03] propose a parallel plane-based volume intersection method to parallelize the reconstruction. With a Myrinet network, 9 PCs and 9 cameras they achieve a frame rate of 15Hz.

Different projects aim at a realistic rendering of the 3D reconstructed model. Matusik *et al.* [MBR+00] describe an image-based approach to compute and shade visual hulls from silhouette image data. 640x480 images are produced at a frequency of 8 frames per second by using 4 client computers and a quad-processor PC as server. In this approach, no explicit 3D shape is necessary because all is done in the image space. In [MBM01], Matusik *et al.* present an algorithm creating and rendering an exact polyhedral visual hull. The system runs at 15 fps with 4 cameras and 5 PCs. Li *et al.*[LMS03b, LMS03a] propose an improved hardware acceleration to render visual hulls. No explicit volume is produced, all computation and rendering is done by the GPU. This ingenious approach produces textured images at 84 fps. In [LMS03c], Li *et al.* remove artifacts on the dynamic object by using projective texture mapping. In [LSMS02], the authors use stereo to compensate for some of the inherent drawbacks of the visual hull method, such as inability to reconstruct surface details and concave regions. Matsuyama and Takai[MT02] use 9 cameras and 9 PCs on a Myrinet network to produce a "3D video" permitting a free view point visualization. Goldlücke and Magnor[GM03a, GM03b] use a voxel structure and render the actor by placing suitable textured billboards at the center of each voxel. They obtain a frame rate of 15Hz with 4 cameras, 2 client PCs and one server PC. Another interesting approach, 3D video fragments[WLG04], generates free viewpoint video by using splatting, and takes into account time and spatial coherence between frames instead of regenerating the whole scene. Real-time interaction is demonstrated by Prince[PCF+02] in a more complex setup where the reconstructed body can be seen and manipulated by another user in an Augmented Reality context.

Another step after obtaining the 3D shape of a moving actor is to reconstruct a skeleton of his body and then to fit it to a virtual body model. In [CMSS03, TMSS02], the authors fit the volumetric reconstruction to a humanoid skeleton. Cheung *et al.*[CKBH00] propose to fit ellipsoids on a volumetric reconstruction. Starck and Hilton[SH03] use a stereo approach to reconstruct the voxel shape of an actor, find its skeleton and fit his body to a virtual model. In [TSS02, TLMS03, TCM+03, TCMS03, CTMS03], the authors propose a full system to produce free view-point video of human actors. They depict the 3D shape reconstruction, the body fitting approach, and the texturing of the virtual body.

Because reconstruction of the 3D shape of the dynamic object is time critical, we have compared the different approaches in Table 1. The proposed classification is done by reconstruction algorithm: volumetric (using voxels) or polyhedral. Solutions such as [MBR+00, LMS03b, LMS03a] are not represented in this table because of their implicit reconstruction on the graphic hardware which limits the interaction with the virtual world (see discussion in Section 5).

Fitting the silhouettes to a predefined body model is not a viable approach for our application, first because it is typically restricted to a simple model and does not allow complex clothing movements; second because it places a severe limitation on what can be captured (i.e. a single human body). Instead we want to be able to capture multiple bodies or additional objects.

3. System architecture

3.1. Hardware architecture

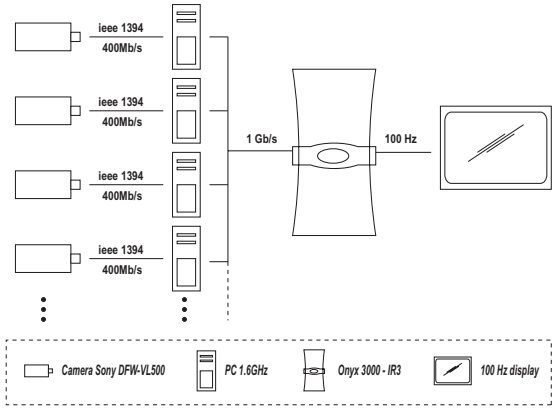

Figure 1: *Hardware configuration of the system.*

Our system is built using a number of independent components and a combination of pipeline and parallel organizations. Our current implementation uses four video cameras but this number could be scaled up as discussed later. Each video camera is connected to a PC, linked to a supercomputer for further processing. The image of the virtual world with the embedded actor is then projected on a screen (Figure 1).

The cameras are standard IEEE 1394 cameras (Sony

Ref	Goal	# PC[1]	Network	# cameras	Time[2] ms
Volumetric					
[BD00]	Reconstruction	16	Ethernet 100	14	100
[CKBH00]	Ellipsoid fitting	5+1	"high speed hub"	5	40
[MT02]	Editing	9	Myrinet 1.28GBits/s	9	64
[WM03]	Reconstruction	6 to 10	Myrinet	6	40 to 23
[TLMS03]	Skeleton + Playback	3+1	Ethernet 100	6	100 to 125
[GM03a, GM03b]	Rendering	2+1	Ethernet 100	4	66
Polyhedral					
[MBM01]	Rendering	4+1	100MBit/s	4	66
[TLMS03]	Skeleton + Playback	3+1	Ethernet 100	6	40

Table 1: *Comparison of recent systems using real-time 3D object reconstruction for different goals. (1) Notation C+S corresponds to C PC clients and S PC server. (2) "Time" corresponds only to the reconstruction time.*

DFW–VL500) running in the YUV4:2:2 mode at a resolution of 640x480 pixels. The PCs are Pentium4-class running at 1.6 GHz. The link between the PCs and the SGI Onyx 3000–IR3 supercomputer is a 1Gbit/sec Ethernet network. The Onyx is configured to use 8 R12000 processors running at 400Mhz. The output of the Onyx is a standard video projector or a 100Hz display screen.

Figure 2: *Experimental setup. Cameras and actor capture region are surrounded by red circles.*

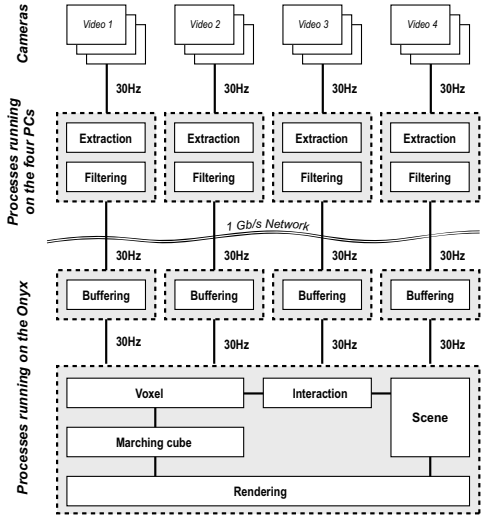

Figure 3: *The different modules involved (dashed rectangles correspond to processes.)*

The actor can move freely inside a volume of approximately 2m cubed. We decided not to use a blue background to retain maximal freedom in the setup, and allow for a transportable system, but rather we use a controlled lighting environment with a grid of fluorescent lights (see Figure 2). The cameras are not externally synchronized to allow capture at their maximum rate, a software mechanism is used to ensure the synchronization of captured images (see Section 4.2.3).

3.2. Software architecture

The PCs are running Windows 2000. An IEEE 1394 library developed at Carnegie Mellon University[UBNN] is used to drive the cameras and the OpenCV[DHF+] library is used for camera calibration.

The whole system comprises nine processes running in parallel (see Figure 3). Four identical processes run on the PCs (one on each): they repeatedly wait for a complete image sent by the camera, acquire it, perform background subtraction and filtering, and send the result to the Onyx by the network via an UDP port (see pseudocode bellow). As we shall see in Section 6 a constant sustained data flow of 30Hz is maintained.

```
// Pseudocode running on each PC
while(true) {
  AcquireImage(*img);
  BackgroundExtraction(*img, *BWBitmap);
  Filter(*BWBitmap);
```

```
  sendUDP(*BWBitmap);
}
```

Five processes run in parallel on the Onyx. Four of them are buffering silhouette images received through the network. Silhouettes are overwritten as they are received, allowing for a good synchronization as discussed in Section 4.2.3. The ninth process runs an infinite loop: it first sets a mutual exclusion to take data at the same time from each buffer (see Pseudocode bellow). The process then fills a voxel area (see Section 4.2.1), applies the marching cubes algorithm to smooth the geometry of the actor (see Section 4.2.2), tests for possible interaction and performs the integration with the virtual world (see Section 5).

```
// Pseudocode of the main process run-
ning on the Onyx
while(true) {
  Mutex.lock();
  ReadAllBuffers(# of camera, *buff);
  Mutex.unlock();
  updateVoxelArea(buff, *voxel);
  updateInteractionData(voxel);
  processMarchingCube(voxel, *polygons)
  render(polygons);
}
```

4. Reconstruction

The reconstruction of the actor model consists of finding the silhouette of the actor viewed by each camera and using these to estimate the 3D shape of the full body. We describe these steps and discuss the issue of synchronization in order to obtain a consistent reconstruction.

4.1. Silhouette Extraction

Cameras must first be calibrated. Estimating cameras parameters from coplanar points with known positions is a well known problem, see [Tsa86, Zha00] for instance. In practice, we use a large (1m) checkerboard pattern and the OpenCV library [DHF+] to calibrate the cameras.

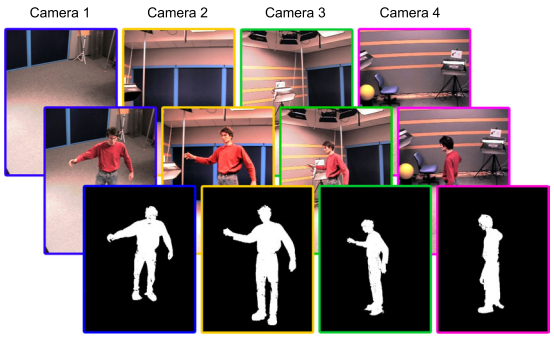

Figure 4: *Background subtraction.*

In our system, the background is static and the actor moves. To determine the silhouette viewed by each camera, we first acquire the background (without the actor) and when the actor is in the field of view of a camera we detect pixels whose value has changed. Indeed a background pixel value should be constant over time while a foreground pixel value can vary. Following this principle, several approaches exist that check the intensity functions at each pixel. Robust ones use temporal filters to detect changes as well as spatial filters to cluster pixels and eliminate false detection (see [TKBM99] for a comparative study). However, in our context the critical issue is not robustness but rather speed. Furthermore, even if some artifacts exist in an image, they are unlikely to be consistent in the whole set of images, and will thus be removed by the reconstruction step. Therefore we use a simple but fast method. Background pixels are assumed to follow Gaussian distributions, possibly correlated, in YUV space. Such distributions are learned *a priori* from a set of background images. The fact that a pixel belongs to the silhouette or the background is then checked by thresholding its Mahalanobis distance to the background mean position in the YUV space. Note that in order to take into account shadows during the subtraction, constraints on the intensity (the Y parameter) values could easily be relaxed. As a result, we obtain a flow of black and white pictures representing the silhouettes as viewed by the different cameras (see Figure 4). This operation takes an average time of 22ms per image, fully consistent with our 30 fps acquisition rate.

4.2. Shape reconstruction

The shape that can be estimated from the different silhouettes is the visual hull[Lau94] of the objects under construction. The visual hull is in fact the maximal solid shape consistent with the object silhouettes. Several approaches have been proposed to compute this visual hull, which we group into the following two categories: surface based approaches and volume based approaches.

Surface based approaches [CG99, MBM01] are not well suited to our application primarily because of the complexity of the underlying geometric calculations. Incomplete or corrupted surface models can be created, and ill-shaped polygons can produce rendering artifacts.

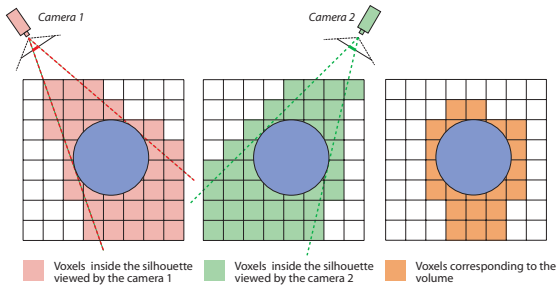

Figure 5: *Principle of voxel carving with two cameras.*

Space carving approaches[SCMS01, Dye01] operate on a discrete space (typically a voxel cube) and mark each space element according to its projection in the images from different viewpoints. Voxels that project outside the object's silhouette in one of the images cannot belong to the object (see Figure 5). These techniques are quite robust, easy to implement, and guaranteed to produce an approximation to the result, commensurate with the chosen resolution. Further as discussed below they can be accelerated using graphics hardware.

4.2.1. Hardware-assisted voxel reconstruction

We observe that the voxel carving approach is essentially a boolean operation on a number of silhouette volumes, computing their intersection. Such boolean operations can be computed on images during a texture mapping step by graphics hardware. The classical N^3 voxel cube is considered as a stack of N images (of resolution N^2), and the stacking direction is chosen to be the closest to the optical axes of the four cameras. The silhouette image for each camera view is projected on each of these slices using the proper perspective projection as texture transform [Lok01]. Note that by using a silhouette image resolution greater than the voxel cube resolution, together with texture filtering, a continuous gray-level image is obtained in each stack. The appropriate blending mode is used to compute the logical AND operation of the four camera views at each slice (Figure 6). The result of this operation is a voxel cube where each non-zero pixel in an image corresponds to a full voxel. This hardware-assisted voxel carving approach is described in more detail in [HLGB03].

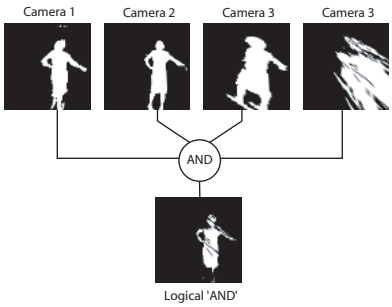

Figure 6: *Logical* AND *operation on four textures projected to the same slice.*

4.2.2. Surface generation

Viewing the reconstructed geometry as points or as cubes is not satisfactory for most applications (Figure 8, left). We apply a "marching cubes" algorithm to obtain a smooth, continuous surface while maintaining real-time.

The marching cubes process allows a good rendering but becomes costly if applied to the entire voxel space. We have tested different approaches to accelerate this process by reducing the number of parsed voxels: the scene can be limited to a bounding box, or divided into hierarchical/regular sub-regions, or into "Bounding Slices" (per-slice bounding boxes). This information can be computed at low cost while transferring graphics card results to our data structure. Best results were obtained with Bounding Slices (see Figure 7), which in most cases permit to only parse between 10% and 14% of the voxel space depending on the scene complexity. This saving accelerates the marching cubes process, but also filtering and collision detection: the smooth surface generation then can be performed in 9ms, instead of more than 30ms when operating on the whole voxel space.

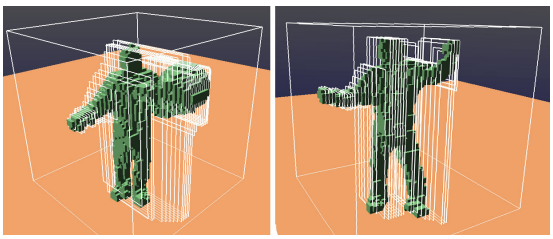

Figure 7: *Bounding box of each slice of the voxel region (the entire voxel region and the minimal bounding box are outlined in white).*

As mentioned before, we retrieve silhouettes as black and white images. If we simply transfer them as binary values in our voxel data structure, the marching cubes algorithm only produces 45 degrees oriented facets (Figure 8, center).

Applying a filter on voxel values before generating the marching cubes allows us to obtain a smooth geometry. This filtering can be done by using the "Imaging Subset" facilities of OpenGL: before reading pixels, we can activate a separable convolution filter (`glSeparableFilter2D`). However we observed that since we only work on a small part of the voxel space, better performance is obtained using software filtering (less than 1ms instead of 3ms). We have tested different filters, the best results being obtained with a simple gaussian filter ($[0.25, 0.5, 0.25]$ kernel).

Now that we have continuous values, we can select a threshold for the marching cubes, so that the isosurface geometry can be generated at a tunable position around the voxels. This threshold choice will act as a dilation (large values) or an erosion (small values) 3D filter. Its value depends on the geometry that is reconstructed. We generally use the default value of 0.5 to correctly smooth geometry without making thin objects disappear (Figure 8, right). As a result we obtain our reconstructed geometry with a smooth surface in real-time.

4.2.3. Synchronization

In theory, synchronization of the cameras should be crucial to obtain an exact reconstruction: if images are not acquired

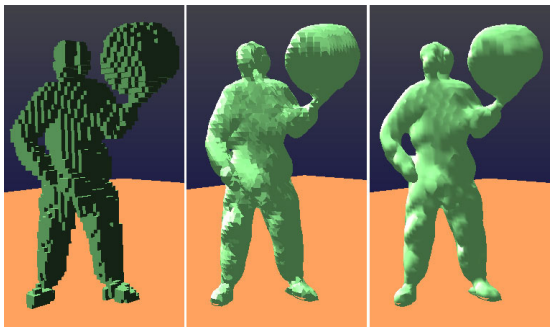

Figure 8: *From left to right: 3D object representation by cubes, pure marching cubes surface, and smoothed surface*

exactly at the same time the voxel carving approach would only reconstruct a part of the object (see Figure 9).

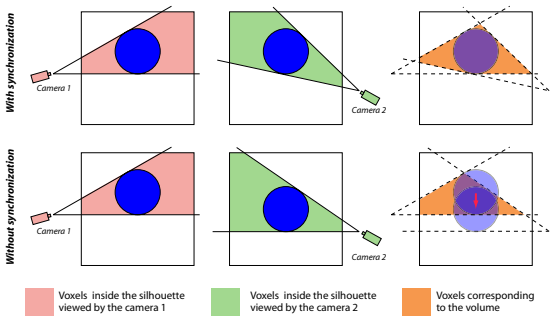

Figure 9: *In the upper row, the two cameras are synchronized, in the lower row, the blue ball had time to move between the two image captures, the final reconstruction (on the right) is only partial.*

To maintain real-time, we must have 30 silhouettes per second. Our cameras nonsustain this rate, but are limited to 15Hz in triggered mode. To bypass this limitation we implemented a software synchronization upon *reception* of the silhouettes (see Section 3.2), which lets us obtain the smoothest possible reconstruction. When shooting a scene, we can only be sure that the worst time shift between any two frames is strictly less than 33ms (see Figure 10). Should this lack of hardware synchronization lead to reconstruction inconsistencies? When the scene is almost static there is obviously no problem, but we could imagine artifacts would occur in the case of fast moving objects. In fact, we did not observe such problems, and instead always obtain realistic results: if we take into account the camera exposure time (independent of framerate), a moving object produces motion blur and thus generates a larger silhouette than its real size. If cameras were synchronized, the 3D reconstructed geometry would be much larger than the original object. Since our silhouettes are slightly time shifted, and only synchronized when used for reconstruction, we finally obtain a realistic

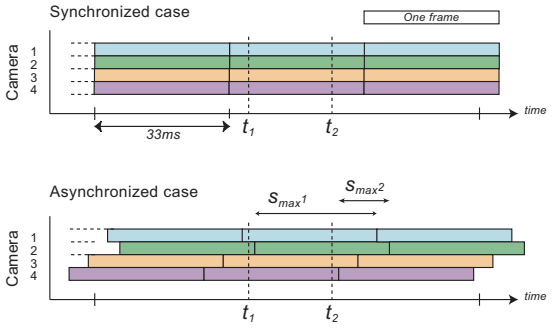

Figure 10: *Best and worst case of synchronization. Let D_i be the delay between t_k and the last frame received from camera i ($0 \leq D_i < 33$). The time shift $S_{i,j}$ between cameras i and j is $\|D_i - D_j\|$. In synchronize case all silhouettes are received at the same time: $S_{i,j} = 0ms$ for any t_k. In asynchronous case, frames are shifted. At t_1 the maximum shift is $S_{max^1} = S_{1,2} \approx 30ms$, at t_2 the maximum shift is $S_{max} = S_{2,4} \ll 33ms$: S_{max^k} is always smaller than 33ms.*

shape, which can be seen as the ”3D average” of the moving object. Finally this configuration allows us to obtain both fluid and consistent reconstructed geometry.

5. Interaction

Our interaction model is quite simple, based on “active” regions within the voxel space. Actions are triggered when a tunable percentage of these regions is filled by voxels, which is both a fast and robust test. There are no constraint on the shape of these regions, and the volume approach lets us use any body parts, or objects manipulated by the actor, to perform actions. On the other hand, of course, we cannot detect which part of the body (arm, foot, etc.) has triggered actions. Nevertheless it allows the actor to use 3D regions as buttons (which can be switched when voxels enter/exit the region as in Figure 11), or even 3D sliders whose value can be interpolated according to the filled voxels. Dynamic virtual objects can also react according to the actor’s position. For example falling balls will correctly bounce off the reconstructed body of the actor (or other objects), since we have normal vectors at each point of its geometry (Figure 12). These interactions are made possible thanks to the combination of different factors: instant feedback (low latency) and immersion sensation.
More results about experiments are described in next section.

Low latency The time between capture and rendering is kept minimal. We have measured that this delay is constant and does not exceed 4 frames: one is due to the transfer from the IEEE cameras to their PC, and the other 3 are due to network transfer, CPU (background subtraction, filtering,

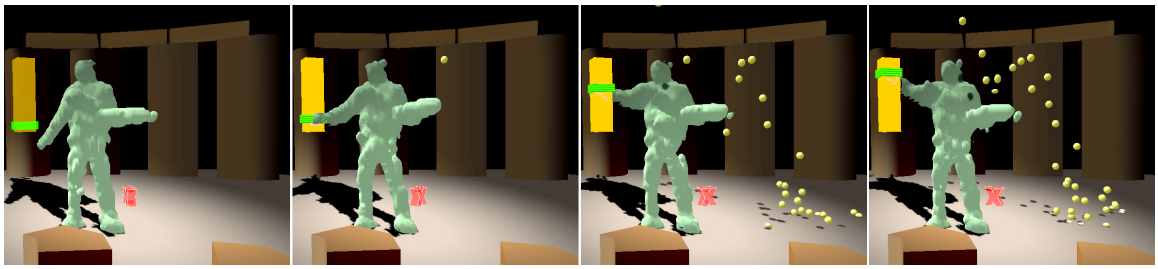

Figure 12: *Example of interaction with balls. Moving the slider on the left changes the flow of balls. Balls are bouncing according to normals of the body or objects.*

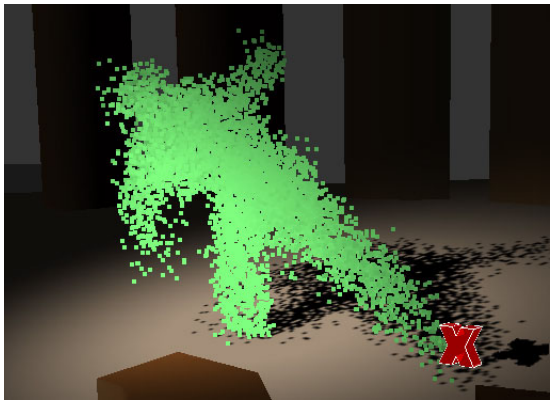

Figure 11: *Example of interaction with an on/off button. Touching the red cross starts the "disintegration" process of the actor!*

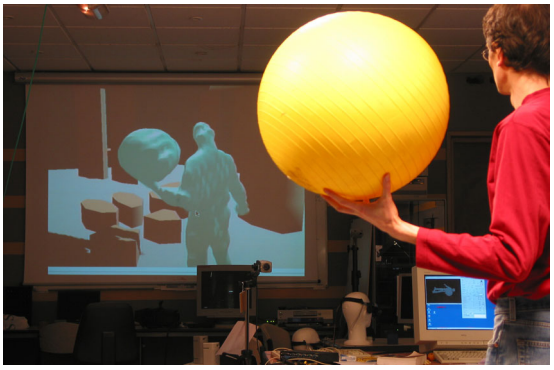

Figure 13: *The actor can visualize himself in the virtual world on a projected screen*

marching cubes) and GPU (conversion to texture, graphical reconstruction and geometry rendering) processing.

Feedback This small latency allows a very comfortable feedback: the user can see him/herself on a projected screen (see Figure 13) and react in real-time within the virtual world. To improve visual feedback we added projected shadows that give important cues for the relative position of the actor with surrounding objects in the virtual scene (see [HLHS03]). Feedback can also be auditive, as some actions generate sound, and allow for example playing on a virtual synthesizer.

6. Results and discussion

6.1. Real-time

As shown in the timeline of Figure 14 we maintain an output image at video framerate and with low latency: treating new silhouettes is done in less than 2ms, transforming them to voxels in 13ms, and conversion to smooth surface and interaction in less than 15 ms: we generate images as fast as silhouettes are coming with no loss. The performance of the last step varies according to scene complexity, for both marching cubes step and most of rendering. Note that body geometry is generated at each frame and can not be memorized on the graphics card as a display list, unlike the others elements of the virtual world.

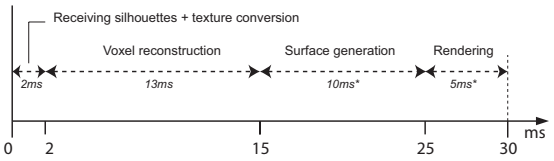

Figure 14: *Time line of the different processes. (*) time spent in surface generation and rendering depends on the complexity of the surface – shown value are typical bounds.*

6.2. Interaction

We have shown in Section 5 that these results allow real-time interaction in complex dynamic virtual scenes. Note that the use of cameras, as opposed to body trackers, allows any person (or object) to enter the scene at any time without any special apparatus, and be instantly embedded in the virtual

world.
Different experiments were performed with external users during demo sessions, on several platforms. In all cases the user can watch his avatar inserted in a virtual scene on a 2m-wide screen.
The first experiment is a virtual synthesizer where a user can play music by hitting 3D virtual coloured parallepipeds: in this case feedback is both auditive and visual. After a few seconds the user understands where he takes place within the virtual world and can play easily a desired melody, with his arms and feet. The impression is quite good, but only a lack of touch sense is felt by the users when hitting an active zone.
An other experimental application was developped around possibilities to create a virtual appartment, where the user can change furniture via a 3D menu, select its colour (by hitting corresponding colour boxes), and choose lighting and music environments. Time needed to feel immersed was also quite short and users could define their environment in less than two minutes.
The global feeling of users is that it is a funny and potentially powerful device: for the first time they could interact with their whole body in 3D (instead of moving their hand in 2D like with a mouse), with no constraints (no apparatus) and at high speed. The restricted zone of capture (limited to 2m) was not perceived as a strong constraint.
To conclude on interaction possibilities we observe that the immersion feeling is easily perceived by users, by quickly identifying them to their avatar even if not realistic (no textures, simple lighting effects). This plaform is therefore adapted to any application not requiring a very high precision but fast and pleasant feedback, and usable without any training.

6.3. Scalability

It is particularly interesting to study the scalability of our system, given the fast-moving pace of technology progress. We briefly discuss here the addition of more cameras, and resolution changes for the voxel space or the output image.

Number of cameras: Using more cameras will improve the carving process. Adding cameras would increase the network traffic, increase reconstruction time, but reduce rendering time (since each new silhouette removes voxels, less geometry should be processed).

- Network overhead: each silhouette is only 10Kb, thus it produces a flow of 300Kb/s. The network - and the Ethernet cards - which can transfer 125Mb/s are far from being saturated.

- Receiving overhead: the processes on the server that are receiving silhouettes all work in parallel and are not very time consuming: adding a few cameras would not slow down this step.

- Reconstruction overhead: as seen in Section 4.2.1, reconstruction is performed on the graphics card. The time to transfer graphics card results to memory is constant. Thus the only additional cost for using a new camera is to convert the silhouette to texture and to render n (voxel size) quads of $n.n$ pixels with this texture projected on it. This step takes under 2ms for each camera.

Thus adding new cameras is not a major bottleneck, adding about 2ms per cameras to our timeline.

Output image resolution Since we are working with standard 3D geometry, there is no specific limitation on the size of the output image. In fact its size hardly changes the rendering times: 4ms are needed to render a reconstructed scene in a 512x512 output window, and 5ms in a 1280x1024 window. Furthermore, as we only use standard graphics card facilities for reconstruction, we can take advantage of its additional features for adding effects like complex shadows to the final rendering (Figure 11 and 12).

Voxel space size The main bottleneck in our configuration is the time spent transferring the final silhouette rendering (corresponding to the results of the intersection of silhouettes) from the graphics card to main memory. This operation is quite slow (80Mpixels/sec) and would slow down the whole process if working with a higher resolution voxel space.

7. Conclusions and future work

We have presented a complete solution that allows one or several live actors to be reconstructed and embedded in a virtual scene at video framerate with reduced latency. Thanks to the availability of 3D information, virtual objects can be interactively manipulated, real-time lighting computation can be performed, and the whole body can be used to generate a new kind of performance. We plan to combine this technique with real textures as it will surely improve the immersion feeling, and body parts recognition would allow more advanced interactions. We can also now imagine sharing the same virtual world with other actors by connecting this system with distant ones, or combining it with augmented reality setups. We also plan to investigate if it could be possible to replace the Onyx server by an on-the-shelf PC, that should obtain equivalent performance, and would allow more up-to-date graphics effects.

References

[BD00] E. Borovikov and L. Davis. A distributed system for real-time volume reconstruction. In *Computer Architectures for Machine Perception*, pages 183–190, 2000.

[CG99] R. Cipolla and P.J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, 1999.

[CKBH00] Kong Man Cheung, Takeo Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition*, 2:714–720, June 2000.

[CMSS03] C.Theobalt, M. Magnor, P. Schueler, and H.P. Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. *International Journal of Image and Graphics - Special issue on Combining Images amd Graphics*, 2003.

[CTMS03] J. Carranza, C. Theobalt, M Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. on Computer Graphics*, 22(3), July 2003.

[DHF+] Bob Davies, Chris Halsall, Frank Fritze, Gen-Nan Chen, Valery Mosyagin, Dr. Neurosurgus, Stelian Persa, Victor Eruhimov, Sergey Molinov, and Vadim Pisarevsky. Open computer vision library. http://sourceforge.net/projects/opencvlibrary/.

[Dye01] Charles Dyer. Volumetric scene reconstruction from multiple views. In L. S. Davis, editor, *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[GM03a] B. Goldlücke and M. Magnor. Real-time, free-viewpoint video rendering from volumetric geometry. *Proc. SPIE Conference on Visual Communications and Image Processing*, 5150(2):1152–1158, June 2003.

[GM03b] B. Goldlücke and M. Magnor. Real-time microfacet billboarding for free-viewpoint video rendering. *Proc. IEEE International Conference on Image Processing*, 3:713–716, September 2003.

[HLGB03] Jean-Marc Hasenfratz, Marc Lapierre, Jean-Dominique Gascuel, and Edmond Boyer. Real-time capture, reconstruction and insertion into virtual world of human actors. In *Vision, Video and Graphics*, pages 49–56. Eurographics, Elsevier, 2003.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4), 2003. State-of-the-Art Report.

[Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150–162, February 1994.

[LMS03a] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Proceedings of Graphics Interface*, pages 65–71, June 2003.

[LMS03b] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Improved hardware-accelerated visual hull rendering. In *Proceedings of Vision, Modeling, and Visualization*, pages 151–158, Nov 2003.

[LMS03c] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Online accelerated rendering of visual hulls in real scenes. *WSCG 2003*, 2(11):290–297, Feb 2003.

[Lok01] Benjamin Lok. Online model reconstruction for interactive virtual environments. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 69–72. ACM Press, 2001.

[LSMS02] Ming Li, Hartmut Schirmacher, Marcus Magnor, and Hans-Peter Seidel. Combining stereo and visual hull information for on-line reconstruction and rendering of dynamic scenes. In *Proc. 5th Conf. on Multimedia Signal Processing*, 2002.

[MBM01] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Eurographics Workshop on Rendering*, pages 115–126, 2001.

[MBR+00] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Siggraph 2000, Computer Graphics Proceedings*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.

[MT02] Takashi Matsuyama and Takeshi Takai. Generation, visualization, and editing of 3d video. In *1st International Symposium of 3D Data Processing Visualization and Transmission*, pages 234–245, 2002.

[PCF+02] Simon Prince, Adrian David Cheok, Farzam Farbiz, Todd Williamson, Nik Johnson, Mark Billinghurst, and Hirokazu Kato. 3-d live: real time interaction for mixed reality. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 364–371. ACM Press, 2002.

[SCMS01] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafe. A survey of methods for volumetric scene reconstruction from photographs. In *International Workshop on Volume Graphics*, 2001.

[SH03] J. Starck and A. Hilton. Towards a 3d virtual studio for human appearance capture. In *Vision, Video and Graphics*, pages 17–24. Eurographics, Elsevier, 2003.

[TCM+03] C. Theobalt, J. Carranza, M. Magnor, J. Lang, and H.-P. Seidel. Enhancing silhouette-based human motion capture with 3d motion fields. *Proc. IEEE Pacific Graphics 2003*, pages 185–193, Oct 2003.

[TCMS03] C. Theobalt, J. Carranza, M. Magnor, and H.-P. Seidel. A parallel framework for silhouette-based human motion capture. *Proc. Vision, Modeling, and Visualization*, pages 207–214, Nov 2003.

[TKBM99] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and Practice of Background Maintenance. In *iccv99*, pages 255–261, 1999.

[TLMS03] C. Theobalt, M. Li, M Magnor, and H.P. Seidel. A flexible and versatile studio for synchronized multi-view video recording. In *Vision, Video and Graphics*, pages 9–16, 2003.

[TMSS02] C. Theobalt, M. Magnor, P. Schueler, and H.P. Seidel. Multi-layer skeleton fitting for online human motion capture. In *7th International Fall Workshop on Vision, Modeling and Visualization*, pages 471–478, 2002.

[Tsa86] Roger Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3d Machine Vision. In *Proceedings of IEEE Coonference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[TSS02] Christian Theobalt, Marcus Magnor Pascal Shüler, and Hans-Peter Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. In *Proc. IEEE Pacific Graphics 2002*, pages 96–103, 2002.

[UBNN] Iwan Ulrich, Christopher Baker, Bart Nabbe, and Illah Nourbakhsh. Ieee-1394 digital camera windows driver. http://www-2.cs.cmu.edu/∼iwan/1394/.

[WLG04] S. Würmlin, E. Lamboray, and M. Gross. 3d video fragments: Dynamic point samples for real-time free-viewpoint video. In *Computers & Graphics*, pages 3–14, 2004.

[WM03] X. Wu and T. Matsuyama. Real-time active 3d shape reconstruction for 3d video. In *Proc. of 3rd International Symposium on Image and Signal Processing and Analysis*, pages 186–191, 2003.

[Zha00] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 22(11):1330–1334, 2000.