A Real Time System for Robust 3D Voxel Reconstruction of Human Motions

German K.M. Cheung, Takeo Kanade
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{german+,tk+}@cs.cmu.edu

Jean-Yves Bouguet, Mark Holler
Microprocessor Research Laboratory
Intel Corporation
San Jose, CA 95054
{jean-yves.bouguet, mark.holler}@intel.com

Abstract

We present a multi-PC/camera system that can perform 3D reconstruction and ellipsoids fitting of moving humans in real time. The system consists of five cameras. Each camera is connected to a PC which locally extracts the silhouettes of the moving person in the image captured by the camera. The five silhouette images are then sent, via local network, to a host computer to perform 3D voxel-based reconstruction by an algorithm called **SPOT***. Ellipsoids are then used to fit the reconstructed data. By using a simple and user-friendly interface, the user can display and observe, in real time and from any view-point, the 3D models of the moving human body. With a rate of higher than 15 frames per second, the system is able to capture non-intrusively sequence of human motions.*

1. Introduction

Due to the vast applications in the fields of telepresence, virtual reality, human machine interface, the research of real time human motion modelling and tracking is gaining much attention. In the past decade, efforts have been put in developing methods for reconstructing three dimensional models of human actions or tracking human/hand movements from multiple cameras ([9], [7], [11], [6], [5], [2], [3] and [12]). These methods can be divided into two categories. The first category focuses on detailed model reconstruction [9] or segmentation [6] of humans in 3D. Despite the fact that image acquisition is real time, the reconstruction or segmentation of the methods in this category is done offline. The second category [2], [3], [12], [1] involves real-time 3D human motion tracking without explicit or detailed 3D human model reconstruction. Almost all of the methods in this category assume a generic 3D human model and the tracking/model fitting is done by comparing the camera projections of the generic model with the silhouettes [3], edges [5] or the optical flow [1] of camera images. Although these methods work well in general, none of them have tried to reconstruct and fit humans in real-time directly in the 3D domain. In view of this, we have built a system that can achieve real-time 3D reconstruction of human body and movements. Our approach combines the features of model reconstruction and fitting of the above two categories in the 3D domain.

A brief overview of our approach is given in Section 2, followed by detailed descriptions of each step in Section 3 and 4. The system architecture is discussed in Section 5. The performance of using our system to capture human motion sequences is presented in Section 6. Finally, conclusion and future work is given in Section 7.

2. Overall Algorithm

The major aim of our system is to create a robust system which can be used to capture 3D human motions in real time. The system consists of two core processes : 3D reconstruction and ellipsoids fitting. The 3D reconstruction method we used is based on the technique known as shape from silhouettes [8] [10]. This technique is chosen over other methods such as stereo [9] because it can be implemented real time by using look up table as described in Section 3. After the camera has captured an image, the silhouette of the moving humans and objects in the scene are extracted. A robust background subtraction approach for cluttered environment is used and the details are presented in Section 3.1. Figure 1(a). illustrates how 3D models can be built from the extracted silhouettes. By the principle of perspective projection, the object has to lie within the cone (bounding volume) formed by the silhouette and the camera view-point. Hence given multiple silhouette images of an object from different viewpoints, its 3-D shape can be reconstructed by intersecting the corresponding bounding volumes formed by the silhouette constraints as shown in Figure 1(b).

In Section 3.2, we will introduce an algorithm called **S**parse **P**ixel **O**ccupancy **T**est **(SPOT)** which is a digital im-

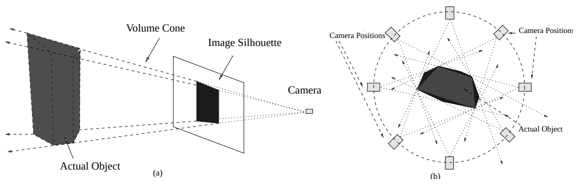

Figure 1. Shape from silhouettes : (a) bounding volume constraints, (b) intersection of bounding volumes.

plementation of the above reconstruction technique. As will be seen shortly, **SPOT** allows us to perform 3D voxel reconstruction robustly in real time.

After obtaining the 3D model of the moving human, ellipsoids are used to fit the data for subsequent motion analysis. The fitting consists of two procedures : body parts segmentation and parameters estimation. The first procedure makes use of a proximity criteria while the second uses moment analysis. The details of these procedures are discussed in Section 4.

3. Shape From Silhouette

3.1. Silhouette generation

One difficult problem of generating silhouette by background subtraction is to remove shadows quickly and effectively. In our system, two techniques are used to tackle this problem. Firstly we incorporate color information to distinguish shadow and non-shadow pixels. Secondly, we use different threshold constants based on the type of the pixels. With these two techniques, we are able to perform robust background subtraction up to thirty frames per second.

Given a background image whose $(i, j)^{th}$ pixel color is denoted by the vector $\boldsymbol{c}_b(i, j)$), the algorithm for extracting the silhouette pixels from the run-time image (with the $(i, j)^{th}$ pixel color being $\boldsymbol{c}_r(i, j)$) is stated formally as follows:

```
1. Calculate the intensity difference
         $d(i,j) = ||\boldsymbol{c}_r(i,j) - \boldsymbol{c}_b(i,j)||$.
   **If** $d(i,j) > T^U$,
       **then** the $(i,j)^{th}$ pixel is a silhouette pixel, **end**
       **else** go to Step 2.

2. **If** $d(i,j) < T^L$,
       **then** the $(i,j)^{th}$ pixel is *NOT* a silhouette
                 pixel, **end**
       **else** go to Step 3.

3. Calculate the color difference
         $\theta = \cos^{-1}[\frac{\boldsymbol{c}_r(i,j) \cdot \boldsymbol{c}_b(i,j)}{||\boldsymbol{c}_r(i,j)||||\boldsymbol{c}_b(i,j)||}]$.
   **If** $\theta > T^C$,
       **then** the $(i,j)^{th}$ pixel is a silhouette pixel, **end**
       **else** the $(i,j)^{th}$ pixel is *NOT* a silhouette pixel,
                 **end**.
```

Here, $||\boldsymbol{c}||$ represents the norm of a vector $\boldsymbol{c}$ and $\cdot$ is the dot product operator. The algorithm basically consists of three tests. Steps 1 and 2 test the intensity difference between the run-time and background pixels. If the intensity difference is very large (as compared to an up threshold constant T^U), then the pixel should be a silhouette pixel. If the intensity difference is very small (as compared to a low threshold constant T^L), then the pixel must be a background pixel. For pixels with intensity differences lie between T^L and T^U, a third test (Step 3) is performed to determine if it is a shadow pixel. The value θ is the angle between the vectors $\boldsymbol{c}_r$ and $\boldsymbol{c}_b$ in the RGB color domain and hence is a measure of the color difference between the run-time and background pixels. For a shadow pixel, the color difference between run-time and background should be small because the difference lies mainly in the intensity values. Hence by comparing θ with a color threshold constant T^C, most of the shadow pixels can be removed.

The second technique we employed in our background subtraction process is to use different threshold constants for different regions of the image. Right after the background image is acquired, it is automatically segmented into different regions by using color information. For example, Figure 2(c) shows the segmentation of the background image in Figure 2(b) into two regions: floor and non-floor. The rationale behind this is that different types of regions have different color statistics and shadow probabilities. For example, the floor region has a higher probability of having shadows than the non-floor region and therefore different thresholds must be used. This region-based threshold constant approach is more flexible than the simplest method of using only one set of thresholds for the whole image. Moreover, as compared to the method of having thresholds for *each* pixel, our approach is more practical and accurate. It is because the number of thresholds in our approach is not high and hence we can determine each of them manually. On the other hand, for the pixel-based method, the large number of thresholds are usually determined by using the pixel color variances and cannot be fine tuned individually. In our system, there are only six thresholds for each camera (two regions and each region has three thresholds) and they are predetermined manually prior to the start of the system by studying the color and the shadows of the floor and non-floor regions. Figure 2(d) is the foreground silhouette image extracted from the run-time and background images in Figure 2(a) and 2(b) respectively. It can be seen that the shadows cast by the legs are removed completely.

To study the performance of our approach, we measure the error rates of extraction. There are two types of er-

1063-6919/00 $10.00 © 2000 IEEE

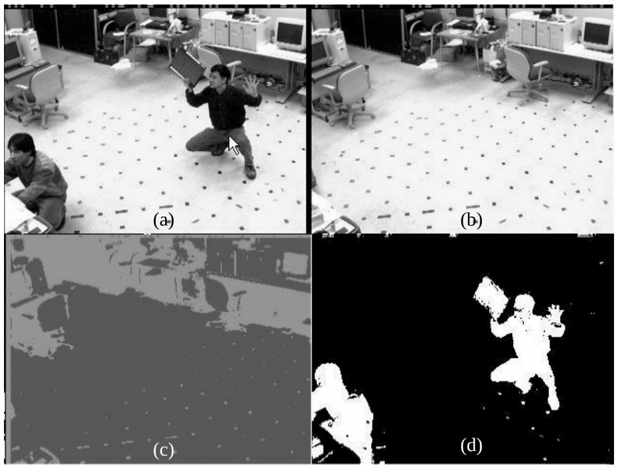

Figure 2. Images from silhouette generation : (a) run-time image, (b) background image, (c) segmented background image, (d) extracted silhouette image

rors. The first one refers to the case that a silhouette pixel is wrongly marked as a non-silhouette pixel and we denote the probability that this case happens as η. The second type of error happens when a non-silhouette pixel is wrongly marked as a silhouette pixel and we use ξ to denote its probability . As will be obvious in the next section, these two probabilities are important quantities in our 3D voxel reconstruction algorithm **SPOT**. To measure these two probabilities, 150 run-time images are taken and their silhouette images are computed by using our approach. The correct silhouette images are then generated semi-manually by using a mask editing interface program and the correct and error pixels are identified. With these experiments, the error probabilities of our background subtraction approach are found to be

$$\eta = 0.043 \tag{1}$$
$$\xi = 0.021 \tag{2}$$

It can be seen that the error probabilities are indeed very low.

3.2. 3D Voxel Reconstruction by SPOT

In Section 2, shape from silhouette is explained by using bounding ray cones. In practice, however, it is not implemented as shown in Figure 1(b) because intersecting rays in 3D is very numerically unstable. Instead we use a voxel-based implementation [10]. The whole volume of interest is divided into N x N x N equal sized 3D voxels. Each voxel v is tested if it belongs to the foreground objects by projecting itself onto all the K silhouette images. More formally, denote *Proj*$^k(v)$ as the projected region of voxel v on the k^{th} silhouette image. Also let *Is_Overlap*$^k(A)$ be the function which will return **TRUE** if region A overlaps the k^{th} silhouette and **FALSE** otherwise. Moreover, the set of all voxels which belong to the foreground objects is denoted by $INSIDE$ while $OUTSIDE$ represents the set of voxels which are not occupied by the foreground objects. With these definitions, voxel v is classified by using the following algorithm.

```
1. Set $k$, the index of silhouette image as 1.
2. If Is_Overlap$^k(Proj^k(v))$ is FALSE,
     then set $v \in$ OUTSIDE, end
   else if $k$ is equal to K
           then set $v \in$ INSIDE, end
           else set $k = k + 1$, goto Step 2.
```

Note that if the projection of voxel v is found to be not overlapping with *one* silhouette image, it is classified as outside voxel and the rest of the images are not tested. On the other hand, for a voxel v to be classified as inside, its projection has to be overlapped with all the silhouette images.

The implementations of *Proj*$^k()$ and *Is_Overlap*$^k()$ are critical for real-time and robust performance. The projected region of v can be found by first projecting (perspectively) the eight vertices of v onto the k^{th} image plane and then computing the convex hull of the eight projected points on the image. Theoretically by testing all the pixels inside the convex hull against the k^{th} silhouette image, the voxel can be classified accordingly. Suppose on the average, the number of pixels inside the convex hull is S, then if we test all the points inside the convex hull, N^3S tests are required for the whole volume. Although this version of *Is_Overlap*^{k} is the most accurate one, it is also computationally very expensive for our real-time purpose. Here instead of testing all the S pixels, Q *uniformly distributed* pixels (denoted by $q_v^k(l), l = 1, \cdots\cdots, Q$) within the convex hull are chosen and if at least ϵQ ($\epsilon < 1$) of these chosen pixels are silhouette pixels, the voxel is classified as belonging to the foreground objects. We named this implementation of *Is_Overlap*$^k()$ as **S**parse **P**ixel **O**ccupancy **T**est **(SPOT)** and stated it formally as follows:

Sparse **P**ixel **O**ccupancy **T**est **(SPOT)**

```
1. Set $incount$, the number of silhouette pixels be 0.
2. For $l = 1, \cdots\cdots, Q$,
   if $q_v^k(l)$ is a silhouette pixel
     then $incount = incout + 1$.
3. If $incout \geq \epsilon Q$,
     then return TRUE,
     else return FALSE.
```

One big advantage of **SPOT** is that it is $\frac{S}{Q}$ times faster as only N^3Q pixels are tested. Moreover, the positions of the chosen pixels for each voxel for each image are pre-calculated and stored in a lookup table to enhance the speed of testing. The whole voxel-based 3D model reconstruction process is revealed in Figure 3.

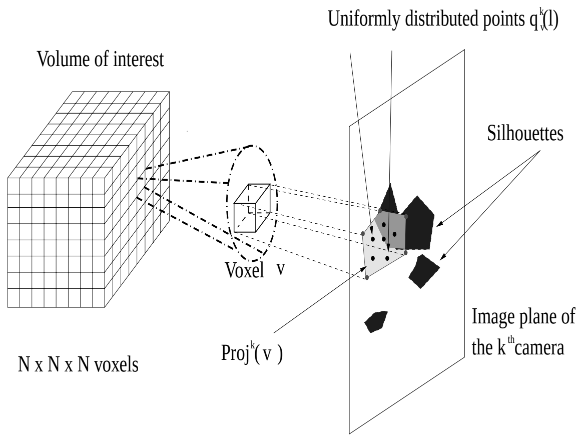

Figure 3. The voxel-based 3D model reconstruction.

The key question of using **SPOT** is how do we choose the value of Q such that the total number of testings is small (real-timeness) while maintaining a low probability of misclassification (robustness). To answer the question, we set $\epsilon = \frac{1}{Q}$ (i.e. $\epsilon Q = 1$) and consider the effect of Q on the probabilities of misclassification. There are two types of misclassification. The first one is False Acceptance (denoted by **FA**) which means that an outside voxel is falsely classified as an inside voxel. The second one is False Rejection (denoted by **FR**) which means that an inside voxel is falsely classified as an outside voxel. Now let us state the relationship of the probabilities of these two types of misclassification with Q.

I. **FA** : Recall ξ is the probability that, during silhouette extraction, a non-silhouette pixel is wrongly marked as a silhouette pixel. It can be shown that the probability of False Acceptance is given as

$$P(\mathbf{FA}) = [1 - (1 - \xi)^Q]^K \ . \tag{3}$$

The exponential of K is due to the fact that an outside voxel has to be misclassified as inside in *all* of the K images for **FA** to happen.

II. **FR** : Similarly, the probability of False Rejection is given as

$$P(\mathbf{FR}) = \eta^Q \sum_{i=0}^{K-1} (1 - \eta^Q)^i \ . \tag{4}$$

The summation comes from the fact that an inside voxel will be misclassified as outside once it is misclassified in one image and the rest of the images are not tested.

Graphs of $P(\mathbf{FA})$ and $P(\mathbf{FR})$ against Q with $K = 5$ and different values of ξ and η are plotted in Figure 4. Notice that the lower the value of Q, the faster the system but the higher the value of $P(\mathbf{FR})$. Based on the values of η and ξ, we can choose Q to obtain a desired rate of false rejection and false acceptance for **SPOT**.

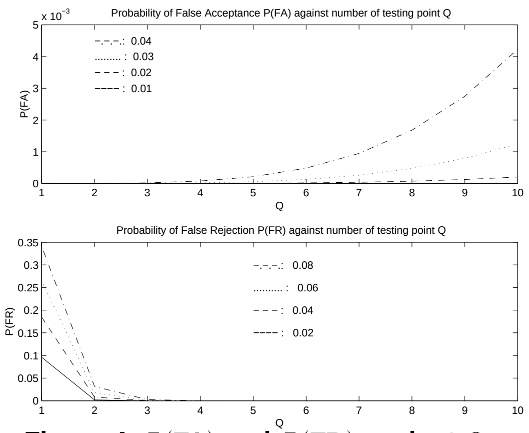

Figure 4. $P(\mathbf{FA})$ and $P(\mathbf{FR})$ against Q.

3.3. Internal Voxels Removal

With the above voxel-based 3D reconstruction algorithm, the reconstructed 3D model will be a list of inside voxels $v \in INSIDE$ which composed the whole volume of the objects being reconstructed. However, in many applications, we may not be interested in the whole volume of the reconstructed data. For example, in displaying the 3D model, since the internal voxels are not visible, much CPU time can be saved by displaying only the surface voxels. To define the notion of surface voxels, we used the concept of six-connectedness. A voxel $v \in INSIDE$ is said to be a surface voxel if one or more of its 6-connected neighbors are vacant voxels. We denote the set of all surface voxels as $\mathscr{S}$. A post-processing step is added after all the voxels are tested for occupancy to extract the surface voxels.

4. Ellipsoids Fitting

After obtaining the 3D surface voxel data (the set $\mathscr{S}$ described above) of the humans, ellipsoid models are used to fit the data in real-time. As a preliminary effort, six ellipsoidal shells $E_f, f = 1, \cdots\cdots 6$ (as analog to the head, the body, two arms and two legs of a human body) are used. Each ellipsoidal shell is represented by nine parameters: the center, the orientation and the sizes (lengths of the axes) denoted by (x_E, y_E, z_E), $\boldsymbol{R}_E$ and $(\alpha_E, \beta_E, \gamma_E)$ respectively. The whole fitting process must be fast for our real-time system. To achieve this goal, we use the following two-step Expectation-Maximization (EM) like approach [4].

4.1. Data segmentation

The first step of the fitting process is to segment the voxel data by using a proximity criterion. Let us define $Dist_{E_f}(v)$ as the closest distance from the voxel v to the ellipsoidal shell E_f. Then we assign the surface voxel v to the shell E_F (i.e. $v \in E_F$ and $F \in \{1, 2, \cdots, 6\}$) if $Dist_{E_F}(v)$ has the smallest value among the six shells. In order to calculate $Dist_{E_f}(v)$, the shell parameters estimated from the previous frame are used. Notice that this proximity criterion is fast and easy to compute.

4.2. Parameters estimation

After the segmentation process, moment analysis is used to estimate the ellipsoidal shell parameters from the centers (x_v, y_v, z_v) of the surface voxels $v \in E$. The zeroth, first and second moments are defined as

$$M_0 = \sum_{v \in E} 1 \tag{5}$$

$$M_u = \frac{1}{\boldsymbol{M}_0} \sum_{v \in E} u_v, \tag{6}$$

$$M_{uw} = \frac{1}{\boldsymbol{M}_0} \sum_{v \in E} u_v w_v \tag{7}$$

where $(u, w) \in \{x, y, z\}$. From these equations, the ellipsoidal shell parameters are given by

$$(x_E \;\; y_E \;\; z_E) = (M_x \;\; M_y \;\; M_z)\,, \tag{8}$$

$$\boldsymbol{M} = 4 \begin{bmatrix} M_{xx} & M_{xy} & M_{xz} \\ M_{yx} & M_{yy} & M_{yz} \\ M_{zx} & M_{zy} & M_{zz} \end{bmatrix} \tag{9}$$

$$= \boldsymbol{R}_E \begin{bmatrix} \alpha_E^2 & 0 & 0 \\ 0 & \beta_E^2 & 0 \\ 0 & 0 & \gamma_E^2 \end{bmatrix} \boldsymbol{R}_E^T\,, \tag{10}$$

with $\boldsymbol{M}$ being the moment matrix. $\boldsymbol{R}_E$ and $(\alpha_E, \beta_E, \gamma_E)$ can be obtained easily by performing an eigen-decomposition on the (3 by 3) matrix $\boldsymbol{M}$.

4.3. Ellipsoids initialization

As described in Section 4.1, to perform segmentation for the current frame, we need the estimated ellipsoidal shell parameters from the previous frame. To initialize the shells when the system is first started, we used an ellipsoidal shell activation procedure. When the system is first started, only ellipsoid E_1 is activated for the fitting. All the voxel data is used to estimate the parameters of E_1 and the error of fitting is calculated. If this error is larger than a threshold this means that one ellipsoid shell is not enough to model the voxel data. Under this case, E_2 is activated so that two shells are used to fit the data. As the human moves his arms and legs around, the shells are activated sequentially to fit different parts of the body until all six of them are activated. The fitting can then be proceeded in the way discussed in Section 4.1 and 4.2.

4.4. Advantages and disadvantages

One problem of our two-step fitting approach is that the joints and sizes constraints of the human body are not incorporated. Hence the fitting fails when the body parts are too close together which causes problems in segmentation by proximity. This means the current algorithm could not handle appearing and disappearing body parts well. A better 3D fitting/tracking algorithm which makes use of the knowledge of the body constraints is under development by the authors. However, the current simple approach can be used to obtain important modeling and tracking information such as joints parameters, sizes and dimension of body parts of an unknown person. Hence we viewed the current approach as an automatic initializing procedure to start the system so that a more sophisticated method can kick in for better motion tracking after enough information is collected by this initialization.

5. System Architecture

Figure 5 shows the architecture of the whole system. There are totally five cameras which are positioned spatially so that their field of views covered the volume of interest. The cameras are calibrated by planar calibration patterns and the method described in [13]. Each camera is connected to an individual PC installed with capture card. The individual computer captures run-time images at 30 frames/s and generates the silhouette images as bit masks which are then sent to a host computer through the high speed hub. The host computer, which is equipped with dual Pentium II 400

MHz processors, collects all the five bit masks from the individual computers and perform voxel reconstruction, internal voxels removal, model fitting and display at the same time by multi-threading the operations. The timing diagram for the pipeline processing can be found in Figure 6.

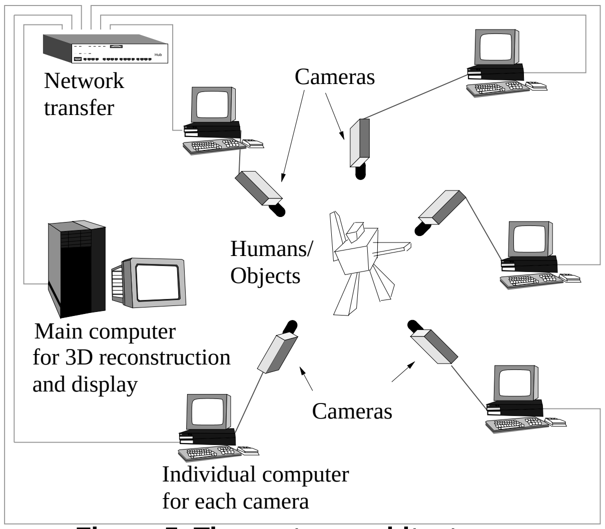
Network transfer
Cameras
Humans/ Objects
Main computer for 3D reconstruction and display
Cameras
Individual computer for each camera

Figure 5. The system architecture.

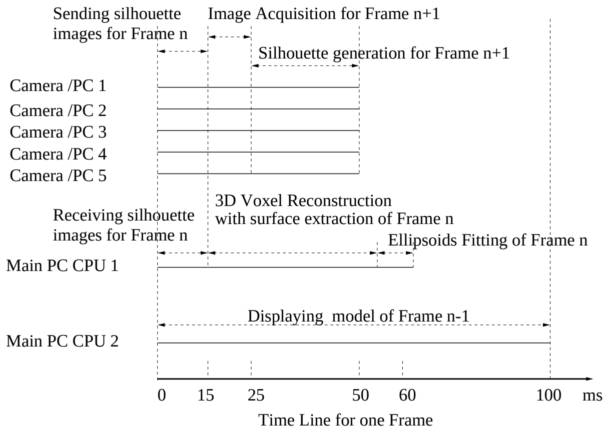
Sending silhouette images for Frame n
Image Acquisition for Frame n+1
Silhouette generation for Frame n+1
Camera /PC 1
Camera /PC 2
Camera /PC 3
Camera /PC 4
Camera /PC 5
Receiving silhouette images for Frame n
3D Voxel Reconstruction with surface extraction of Frame n
Ellipsoids Fitting of Frame n
Main PC CPU 1
Displaying model of Frame n-1
Main PC CPU 2
0
15
25
50
60
100
ms
Time Line for one Frame

Figure 6. The pipeline architecture and timing diagram.

6. Performance

The system is used to capture human motions with the following settings :

1. The volume of interest is a cube of size 2m x 2m x 2m.

2. The volume is divided into 64 x 64 x 64 (i.e. N = 64) voxels with a resolution of about 3cm for each voxel.

3. The images are at a resolution of 320x240 pixels.

4. With the above settings, the average number of pixels for a projected voxel is 10 (i.e. $S = 10$). Two points (i.e. $Q = 2$) are chosen for each voxel for **SPOT**. This corresponds to a False Rejection probability of 0.0092 for $\eta = 0.043$ (Eq.1) and a False Acceptance probability of 1.239e-7 for $\xi = 0.021$ (Eq.2).

Figure 7 is a screen shot of the user-interface program at the host computer which shows the reconstructed voxels and fitted ellipsoids of a person holding one arm horizontal. Table 1 summaries the approximate time required for each

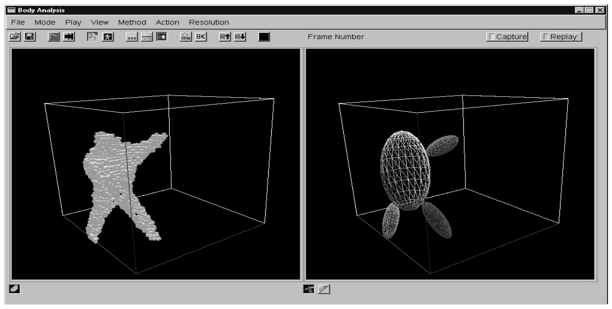

Figure 7. Screen shot of the user-interface host program which shows the reconstructed voxels and fitted ellipsoids of a person holding one arm horizontal.

step. The bottleneck for our system right now is the display part as the host computer is not equipped with a fast display card. Without displaying the reconstructed 3D voxels, the system can run about 16 frames per second.

Step	**Approximate Time Required**
Sending/Receiving Silhouette Images	15 ms
Image Acquisition	10 ms
Voxel Reconstruction without Surface extraction	35 ms
Voxel Reconstruction with Surface extraction	40 ms
Silhouette Generation	15 ms
Ellipsoids Fitting	10 ms
Model Display	100 ms

Table 1. The approximate time for each step in our system.

Two movie clips for the system can be found at the site http://cs.cmu.edu/˜german/research/CVPR2000/. The first movie clip realtime.mpg shows an user moving inside the volume of interest and the system successfully reconstructed and fitted ellipsoid models to the body parts in real-time. The actual setup of the system with the cameras' positions can also be seen in the clip. The second movie clip motionsequence.mpg shows a sequence which was captured real-time from the system. Within each frame, the upper left picture is the run-time image captured by one of the five cameras and the lower left picture depicts its silhouette generated by the method described in Section 3.1. The upper right picture shows the reconstructed voxels of the person by using **SPOT** while the lower right picture represents the fitted ellipsoidal shells. Notice that there are two frames delay between the the run-time image/silhouette and the reconstructed voxels/ellipsoids due to the pipeline processing of the system. Four representative frames are extracted from the movie clip and are shown in Figure 8.

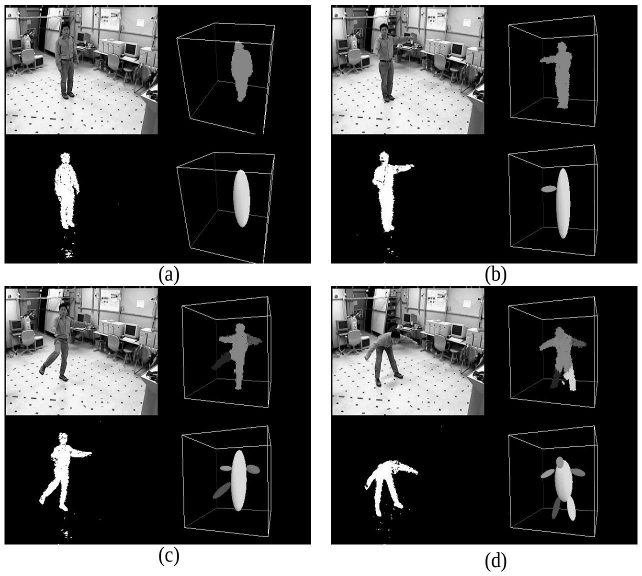

Figure 8. Four representative frames from the movie clip motionsequence.mpg. (a). frame 59, (b). frame 109, (c). frame 139, (d). frame 299.

7. Conclusion and Future Work

In this project, we have shown that by using multiple cameras and voxel-based algorithm, full 3D reconstruction and ellipsoids fitting of human actions can be done in real time. Our approach eliminates the problem of occlusion and hence simplifies subsequent analysis of the motion. As different from most of the other approaches, our system outputs 3D (either volume or surface) voxel data. Furthermore, fitting/tracking in our system is done directly in the 3D domain, instead of projecting the assumed human model back onto the 2D images. Without displaying the voxel data, the system is running at a rate of over 15 frames/s. With this capability, it is useful for applications such as real time electronic puppet, athletic performance analysis, non-intrusive motion captures, human computer interface and entertainment.

As mentioned in Section 4, a more precise 3D body tracking/motion capture algorithm is being developed to improve the fitting process of the system. Moreover, by adding more information, such as color, to the voxels, the system can be considered as a 3D color camera system. This will also improve body part segmentation results. Non model-based methods such as eigen-shape is also being considered for extracting the human movement and posture.

References

[1] C. Bregler and J. Malik. Tracking people with twists and exponential map. In *Proceedings ACCV'98*, volume 2, pages 416–23, Hong Kong, January 1998.

[2] Q. Cai and J. Aggarwal. Tracking human motion using multiple cameras. In *Proceedings of ICPR'96*, volume 3, pages 68–72, Vienna, Austria, August 1996.

[3] Q. Delamarre and O. Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *Proceedings of ICCV'99*, Corfu, Greece, September 1999.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J.R. Statist. Soc.*, B 39:1–38, 1977.

[5] G. Gavrila and L. Davis. 3-d model-based tracking of humans in action : a multi-view approach. In *Proceedings of CVPR'96*, pages 73–80, San Francisco, CA, June 1996.

[6] I. A. Kakadiaris and D. Metaxas. 3d human body model acquisition from multiple views. In *Proceedings of ICCV'95*, pages 618–623, Boston, MA, June 1995.

[7] S. Moezzi, L.-C. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Computer Society Multimedia*, 4(1), January-March 1997.

[8] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40:1–20, 1987.

[9] P. Rander, P. Narayanan, and T. Kanade. Virtualized reality : Constructing time-varying virtual worlds from real world events. In *Proceedings of IEEE Visualization '97*, pages 277–283, Phoenix, AZ, October 1997.

[10] R. Szeliski. Rapid octree construction from image sequences. *CVGIP : Image Understanding*, 58(1):23–32, July 1993.

[11] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. PAMI*, 19(7):780–785, July 1997.

[12] Y. Wu and T. S. Huang. Capturing articulated human hand motion : A divide-and-conquer approach. In *Proceedings of ICCV'99*, Corfu, Greece, September 1999.

[13] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of ICCV'99*, pages 666–673, Corfu, Greece, September 1999.