

# Voxel Carving and Coloring - Constructing a 3D Model of an Object from 2D Images

A. O. Bălan  
Computer Science Department  
Brown University  
Providence RI, 02912  
Email: alb@cs.brown.edu

## Abstract

*We present a simplistic approach to obtaining a 3D representation of an object moving in a scene. The object is monitored by four different cameras with known characteristics. At any given time frame, the silhouettes of the object are extracted from each of the four images using a background subtraction method. Some strategy is afterwards used to perform a computationally efficient 3D model construction by projecting voxels from the 3D scene onto the camera's viewing planes. A 3D binary grid is obtained and visualized using a simple voxel coloring strategy that we developed. Depth maps are constructed as well for each camera image to help with the coloring.*

## 1. Introduction

We consider the problem of 3D modeling of human actions in real time. In general, we would like to extract the movements of different parts of the human body and used them in other contexts such as human computer interaction, imitation, scene rendering from new viewpoints etc. Ideally, this should be done in an as natural way as possible, without using any markers, joysticks or other devices attached to the body. The area of research that deals with this idea is called *markerless motion capture* [3].

One approach to achieve this goal is to use multi-camera images of a person moving in the scene from which to construct a 3D model of the person. Any extension of the problem, such as motion interpretation, posture estimation, kinematic model extraction or recognition, uses the volumetric model generated.

In this paper we present our approach to constructing a 3D model of an object from 2D images. Our aim is that the approach we take need not be computationally intensive so that it can be applicable to on-line processing, which however can restrict the accuracy of the results. We also develop a voxel coloring method as an application of voxel carving.

We present the general method in section 2, together with

existing methods. Algorithmic considerations of our approach follow in section 3. We show experimental results in section 4, a discussion of computer memory requirements in section 5, ways to extend this paper in section 6 and a summary and conclusion in section 7.

## 2. General Method

The main goal is to extract a volumetric model over time of an object using multiple synchronized cameras. We use the *shape from silhouette* technique [2]. There are three steps involved in this method (1-3), to which we add two more steps (4-5) to perform voxel coloring:

### 1. Initialization - camera calibration

The intrinsic and extrinsic parameters of each camera are estimated.

### 2. Silhouette segmentation

The silhouette of the object is segmented from each image using a variant of background subtraction method.

### 3. Voxel carving

A voxel grid is constructed, and voxel projections are intersected with silhouette pixels.

### 4. Depth maps construction

The depth of the closest voxel in the volume shape to each pixel in every camera image is determined.

### 5. Voxel coloring

Each voxel on the volume surface gets assigned a color value based on depth maps and camera images.

## 2.1. Previous Work

### 2.1.1 Camera Calibration

For the final result to be meaningful, the characteristics of the cameras have to be precisely determined. For example, the position of the camera influences the transformation of a point from world coordinates to camera coordinates. The behavior of a camera is modeled by its intrinsic (focal length, principal point, skew and distortions coefficients) and extrinsic (translation and rotation) parameters. Camera calibration techniques and parameter definitions are described in [1, 10].

### 2.1.2 Silhouette Segmentation

Segmenting the silhouettes from the images is typically done using image differencing. First, a background image is acquired, usually directly by taking a picture of the scene without the foreground object in it. Second, by subtracting the background image from the current image, the silhouette of the object will stand out. Whenever the difference between a background pixel and the run-time pixel colors is large, the pixel belongs to the silhouette.

One implicit assumption is brightness constancy, which says that the intensity of the background pixels is the same in all images. In practice this assumption might not necessarily hold. For example, it is known that the lighting used to illuminate the scene pulsates. Reference [9] proposes taking a few dozen background images and estimate the pixel statistics. A threshold can be set on the variation around the mean of a pixel value to segment foreground objects from background.

Another difficulty encountered in this process is dealing with shadows in an efficient manner. Shadows are the byproduct of a foreground object being present in the scene and therefore tends to be detected as part of the silhouette. A fast and robust solution to eliminating shadows is given by Cheung et. al. [2]. They propose two techniques for this. First, they use color information to identify shadow pixels. If the angle between the RGB vectors of the pixel in the two images is large, the pixel is not a shadow pixel. Second, they pre-process the background image to identify regions with different shadow statistics and assign region-dependent shadow thresholds.

Sometimes a background image cannot be acquired directly or it is slowly changing over time. In this case it is possible to indirectly estimate it and update it from a sequence of images in which the object is moving more distinctively than the background. François et. al. [5] model the background in real-time by incrementally learning and adjusting the statistical color of each pixel in the background. They also suggest using the HSV color space instead of the RGB one, as it provides stronger cues for background classification.

### 2.1.3 Voxel Carving

We can identify two major classes of methods used to construct 3D volumetric models: methods that are concerned with accuracy and methods that are concerned with speed. Typically, the methods used for accurate results [] are computationally intensive and time consuming, and hence the volumetric segmentation has to be performed off-line. In contrast, there exist many applications which need to complete the segmentation process in real-time before the next frame becomes available [2, 6, 9]. The results in this case however are not as accurate. Therefore a tradeoff needs to be made between spatial resolution and temporal resolution.

There are a number of ways to partition the space into voxels. As stated by [2], when the silhouette is projected perspectively into the world space, a conic surface is obtained which should enclose the entire object. Here the voxels have a conic shape and the size varies with depth. Intersecting the conic volumes from each of the images can be computationally expensive and therefore impractical for real-time processing.

In contrast, a rectangular bounding box can be used to enclose the relevant world space. The box can be partitioned either using octrees or voxels of equal size. Reference [9] uses an octree representation in which the bounding box is recursively subdivided into smaller voxels until a voxel is fully occupied or empty. This procedure generates voxels of different sizes, but is very concise. The voxel resolution matches the object resolution in this case.

Many methods [2, 6] use voxels of equal size because it has its own benefits. In particular, having the voxel position and size pre-determined, we can build look-up tables with depth and projection information off-line, and thus speed up the on-line processing time. We take this approach in the present paper.

### 2.1.4 Depth Maps

We have not considered any previous work regarding building depth maps.

### 2.1.5 Voxel Coloring

One application of voxel carving is synthesizing object images from viewpoints other than the camera views. It turns out that the work done towards this end did not use the volumetric model. At the same time, they aimed to the highest quality results. Reference [7] performs voxel coloring by traversing the voxel space in depth-order with respect to the new viewpoint and finding the closest voxels that are *color consistent* in all of the reference images. Recent work [4] in rendering images from new viewpoints without explicitly reconstructing the volume of the object

looks along the epipolar lines to achieve color consistency and uses image-based priors for increased realism (by preserving texture statistics).

We have developed our own method which is only intended to give a coarse approximation of how the object would look like from new viewpoints.

### 3. Our Approach - Algorithmic Considerations

Our contributions in this paper are mainly related to voxel carving and voxel coloring. Every time frame when data is captured creates a new instance of the same problem which is solved independently from the other time frames. We consider a voxel grid, which is an equal-size rectangular-shaped voxel-space representation divided into  $N_X \times N_Y \times N_Z$  voxels. Images from  $k$  cameras are acquired and silhouettes are segmented from them.

#### 3.1 Point Projections

Geometrically speaking, for an ideal camera which introduces no distortions in the image, projecting a point from world coordinates to camera coordinates can be done using the equation  $X_C = RX_W + T$ , where  $X_W$  is the world coordinates of a point,  $X_C$  is the camera coordinates, and  $R$  and  $T$  represent the orientation and the position of the camera respectively. This equation however does not take into account the intrinsic parameters that model the behavior of the camera. Nonetheless, it illustrates the main correlation between world and camera coordinates.

#### 3.2 Voxel Projections

The problem of reconstructing the volume of an object from multiple views can be ill-posed. Simply put, if only one image is available, it is clearly impossible to obtain a unique solution. Even with more than one view this might still be the case. The goal is to construct a volume that is consistent in all images, even though it might be bigger than the real one. Nonetheless this approximation is very good in most cases.

A voxel is **volume consistent** if its projections into the camera images are inside the silhouettes.

The reconstructed volume comprises all the *volume consistent voxels*. To formalize this idea, we denote the projected region of voxel  $v$  into the image plane of camera  $k$  by  $Proj^k(v)$ , and the silhouette by  $S^k$ . Let  $V$  be the volumetric model that we want to construct. Then  $v \in V$  if and only if  $\forall k Proj^k(v) \subseteq S^k$ . Whenever  $Proj^k(v) \not\subseteq S^k$  for some image  $k$ , it is certain that  $v \notin V$  and the remaining

images need not be tested, thus reducing the computational time.

Since the size of a voxel is not infinitely small, its projection into a silhouette image can occupy several pixels. The projected region is delimited by the convex hull obtained from the projections of the eight vertices of the voxel. We consider the projection of a voxel as inside the silhouette if a certain proportion of the pixels from the projected region belong to the silhouette:  $|Proj^k(v) \cap S^k| \geq \epsilon |Proj^k(v)|$ ,  $\epsilon < 1$ .

Reference [2] proposes testing only a sample of the pixels from the projected region. The smaller the sample, the smaller the computation time, but the error rate of misclassification is higher. However, as the voxel grid resolution increases, mis-classification decreases. Reference [6] takes this idea to extreme and considers only one point for each voxel which is projected and tested against the silhouette. This corresponds to a sample size of 1. We take the latter approach when performing voxel carving.

Since the position of the cameras and the voxel grid are stationary, a look-up table can be constructed before-hand containing projection information for each voxel and camera. Memory requirements are high in this case, however it drastically reduces the necessary on-line processing time by avoiding expensive matrix multiplication needed for point projections.

We made an implicit assumption that the object is entirely visible in all camera images. If this were not the case, reference [6] suggests imposing a second requirement on voxel consistency: any voxel belonging to the volume has to be visible from at least some pre-determined number of cameras.

#### 3.3 Voxel Coloring and Depth Maps

There has been previous work done involving voxel coloring, however it avoided the problem of reconstructing the volume of the object. We therefore developed our own approach.

We wish to determine the color of each voxel in the volumetric model which is consistent in all camera images. Due to occlusion and self-occlusion, voxels are not visible in all images. In an extreme case, voxels that are completely inside the volume are not visible in any image. Much computational time can be saved by removing the non-surface voxels. The set of all surface voxel,  $Surf(V)$ , is the set of voxels  $v$  from volume  $V$  for which at least one of the six neighbors does not belong to  $V$ .

Multiple voxels can have the same projected region in the image plane, but only the closest one contributes to the pixel color. We use a *depth map* for each image to indicate the depth of the voxel that induced a pixel color. Denoting the depth of a voxel  $v$  from image  $k$  by  $Depth^k(v)$ , we can

define the depth map of an image  $k$  as  $DepthMap^k(p) = \min_{v \in Surf(V), p \in Proj^k(v)} (Depth^k(v))$ , where  $p$  is a pixel in the image  $k$ .

For each camera image we want to determine the depth of the voxels that projected into any given pixel. It is computationally more efficient to iterate first over the voxels rather than over the pixels to construct the depth maps. In this way, each voxel is traversed only once. In this respect, we need to consider all the pixels that a given voxel projects into and update the  $DepthMap^k(p)$  value whenever necessary.

The pseudo-code for constructing the depth maps is included for clarity.

```

For each image  $k$  and each pixel  $p$  in image  $k$ 
  Set  $DepthMap^k(p) = +\infty$ 
For each voxel  $v \in Surf(V)$ 
  For each image  $k$  and each pixel  $p \in Proj^k(v)$ 
    If  $DepthMap^k(p) < Depth^k(v)$ 
      Then Set  $DepthMap^k(p) = Depth^k(v)$ 

```

The depth information at each pixel can be used to determine efficiently whether a voxel is visible in a given image. For a pixel  $p \in Proj^k(v)$  the inequality  $DepthMap^k(p) \leq Depth^k(v)$  holds through construction. Moreover, equality arises only when voxel  $v$  is the closest voxel that projects into  $p$ .

Ideally, the pixels from which a voxel  $v$  is visible should be color consistent. In practice, factors such as the view angle, surface material, camera bias etc. introduce small variations in pixel color intensity. Given the multi-set  $ColorSet(v)$  of all pixel colors that a voxel  $v$  induces in the camera images, we choose by convention to assign the median value in the multi-set as the color of voxel  $v$ . Formally, we assign the color by  $Color(v) = median_{k, p \in Proj^k(v), DepthMap^k(p) = Depth^k(v)} (Color^k(p))$  where  $v$  is a voxel on  $Surf(V)$ ,  $Color(v)$  is the color we assign to voxel  $v$  and  $Color^k(p)$  is the color of pixel  $p$  in image  $k$ .

We provide the pseudo-code for voxel coloring below.

```

For each voxel  $v \in Surf(V)$ 
  Set  $ColorSet(v) = \emptyset$ 
For each image  $k$  and each pixel  $p \in Proj^k(v)$ 
  If  $DepthMap^k(p) == Depth^k(v)$ 
    Then Set
       $ColorSet(v) = ColorSet(v) \cup \{Color^k(p)\}$ 
  Set  $Color(v) = median(ColorSet(v))$ 

```

It is well known that the median value is more robust to error than the mean value and thus expected to be closer to the true value; nonetheless, the mean value in the multi-set might produce smoother results.

## 4. Experimental Results

### 4.1. Data Set

The data set that we used comes from a sequence of images from four different synchronized cameras [8]. Figure 1 shows the images for one time frame. The image resolution is  $644 \times 484$ .

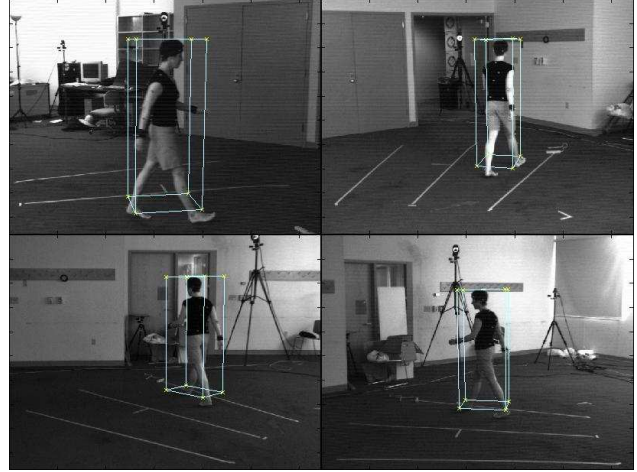


Figure 1: Original images

We have used the silhouettes generated and provided by [8]. They are shown in figure 2. As opposed to [5, 9] in which the background pixel statistics is assumed to have a Gaussian distribution, reference [8] assumes a weighted sum of three Gaussian distributions.

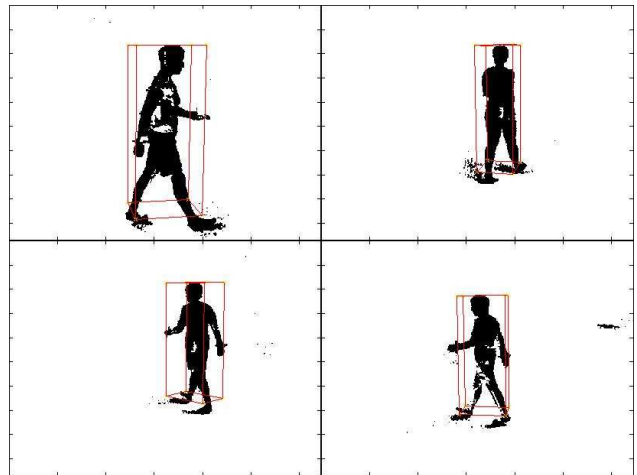


Figure 2: Silhouette images

As an exercise, we have discovered that we can obtain similar results by using the naïve background subtraction method and filtering the resulting image a few times with a median filter. The only downside of this approach is the

computation time, which is high due to the fact that the median filter is not a linear filter.

### 4.2. Generating volume model

We used a built-in function defined in the calibration toolbox in [1] to transform points from world to camera coordinates.

The size and coordinates of the world space bounding box were provided by [8]. This box was just big enough to enclose the person in her current position. Since we assume that the person is moving over time, a larger box would have had to be used which should enclose all possible localization of the person. We have used a smaller box just because we tested our method at only one time frame. This has reduced the computational time for voxel carving since less voxels had to be classified as part of or not of the volume. The size of the bounding box has no impact on any other aspects of the algorithm, which only iterate over the voxels that are part of the surface of the carved volume.

The bounding box resolution used in our experiment was  $23 \times 31 \times 80$  and the voxels were cubes of size approximately 19. About a fifth of the voxels belonged to the volume, and half of those to its surface.

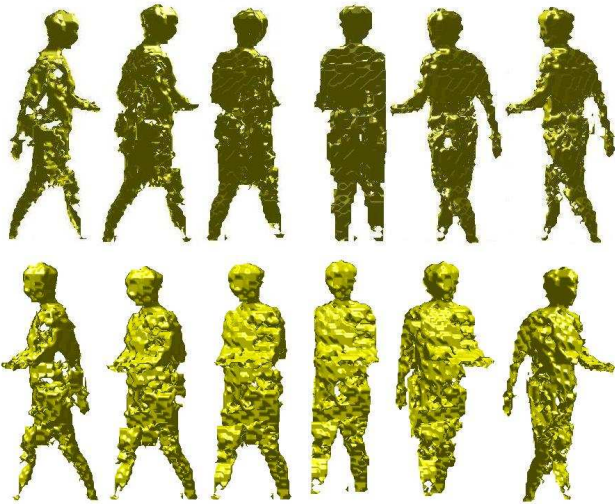


Figure 3: Volume Reconstruction

The volume that resulted from the voxel carving procedure is displayed from 12 different viewpoints in figure 3. It is clear that holes in the silhouettes translated directly into holes in the reconstructed volume. Also, the left hand of the person is constructed incorrectly, covering the entire space in front of the body from the left side to the right side. The reason for this is that, given the images in figure 1, the problem is ill-posed. This is obvious once noticing that no image shows the front-left side of the person.

### 4.3. Recovering color information

Before assigning color to surface voxels, depth maps were constructed. They are displayed in figure 4. They are very good at conveying 3D orientation and position of different body parts.

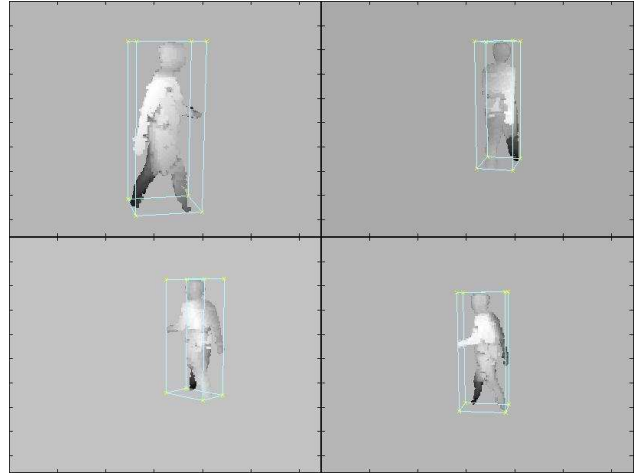


Figure 4: Depth map images - Lighter color means closer; darker means farther

Color reconstruction was performed as described in the previous section. The results are shown in figure 5. Again, the surface voxels on the front-left side of the person are not visible in any of the four camera images, and therefore there the reconstruction failed. Voxels where color reconstruction was not possible were displayed with pure-black color.

The color reconstruction is somewhat coarse and color transition is not smooth.

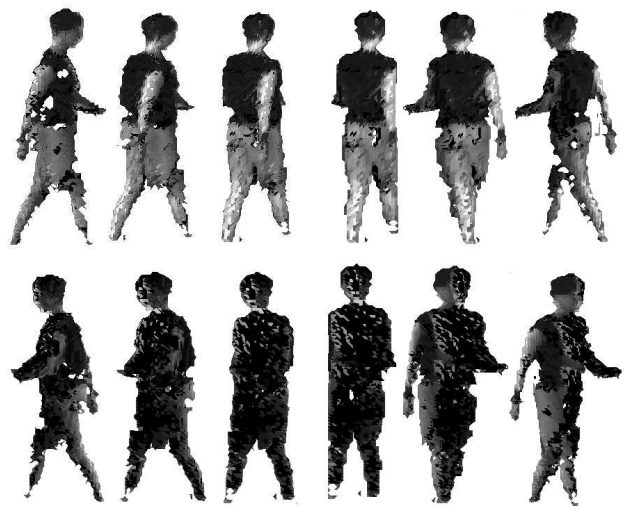


Figure 5: Color Restoration - Completely black color means color was not restored

## 5. Computer Memory Requirements

Previously in the present paper we stated that a look-up table can be pre-computed for easy access of projection information  $Proj^k(v)$ . Similarly we could use a look-up table for voxel depth information  $Depth^k(v)$ .

Our implementation did not use look-up tables for voxel depth, and as a consequence the execution time is higher during the *building depth maps* phase. First, depth information was necessary only for surface voxels. We saw in section 4.2 that in the data set example surface voxels constituted about 10% of all voxels. We did not want to use a lot of memory space to store the depth map look-up table, of which only about 10% get used. Second, the gain from using a look-up table for voxel depth information (scalar multiplication and addition) is not as significant as for the projection information table (matrix multiplication and addition). Nonetheless, if the computational time needs to be reduced further, depth look-up tables should be used.

The voxel carving procedure that we use needs to project just one point belonging to a voxel to each image plane. This is because we reduced  $Proj^k(v)$  to have only one pixel. The space requirement complexity for the projections look-up table necessary for voxel carving is therefore  $\Theta(K \cdot N^3)$ , where  $K$  is the number of cameras, and  $N$  is voxel resolution.

The voxel coloring procedure needs to construct depth maps for each pixel in the camera images, and therefore the entire projection region is needed. The pixels that belong to the projection region of a voxel are found by projecting the eight vertices of the voxel into the image plane; the pixels that are inside their convex hull are selected as part of  $Proj^k(v)$ . If we denote by  $A$  as the average number of pixels in the projected region of a voxel in one image, then the space complexity for the projections look-up table is  $\Theta(A \cdot K \cdot N^3)$ . Keep in mind that  $N$  and  $A$  are inversely related. This look-up table can be used for both voxel carving and voxel coloring. In the example we considered in this paper,  $A = 24.32$ ,  $K = 4$  and  $N^3 = 23 \times 31 \times 80$ , and the table used 36 MB.

As far as time complexity is concerned, it follows directly from the discussion in section 3 that

- **voxel carving:**  $\Theta(K \cdot N^3)$
- **depth maps:**  $\Theta(A \cdot K \cdot \xi(N^3))$
- **voxel coloring:**  $\Theta(A \cdot K \cdot \xi(N^3))$

where  $\xi(\cdot)$  takes into account surface voxels only.

## 6. Future Work

In this paper, each set of synchronized images from a given time created a new instance of the same problem. We

can envision using information acquired or computed at earlier times for processing the current data set. Probabilistic priors could be used to refine the volumetric model. Since the goal is to perform on-line voxel carving, we could recover the volumetric model incrementally, by starting with a low resolution model, and gradually increasing the voxel resolution.

We have seen that the recovered color was not smooth, which might be due to the median choice. A mean value might produce smoother coloring, but it is subject to outliers. We think that results of this quality can be obtained with less computational time by using the first pixel color that is found for which  $DepthMap^k(p) = Depth^k(v)$ , and ignoring the other pixels that satisfy this condition.

The current implementation runs somewhat slow for an on-line process. We need to investigate which part of the procedure is the bottle-neck, and if it is an algorithm complexity problem or an implementation problem.

## 7. Summary and Conclusion

We have presented a method for reconstructing the volumetric model of an object and we have recovered color information for realistically rendering the 3D shape. We did so by segmenting the silhouettes from the images, by constructing the volume as the set of voxels that project into all silhouettes, and by assigning color to each surface voxel taking into account self-occlusion.

This application can be used for monitoring over time a scene and the objects moving in it from arbitrary viewpoints with a limited immobile number of cameras. One such example is broadcasting a football game. It is impractical to have a physical camera roaming around the field, but nothing can stop a virtual camera from doing so. Images captured from around the field with real cameras can be used in synthesizing new images from the viewpoint of a virtual camera. Even though this example might be beyond the current technological capabilities due to the gigantic size of a football field, it might still be applicable in environments with a smaller scale.

Another application that could benefit from voxel carving and coloring ideas is broadcasting over Internet the 3D information of a virtual scene changing over time. Rather than transmitting the entire 3D scene, a number of 2D images are sent over the Internet and the destination could reconstruct the scene and display it from any viewpoint desired. This strategy could decrease the necessary bandwidth of the transmission.

This method is supposed to be used in real-time, however the current implementation takes about a minute to run. The applications that this method could be used to, such as the ones mentioned above and many others, makes it worthwhile and should be refined for better performance.

## Acknowledgments

I would like to thank Leonid Sigal for providing me with the data and for pointing out how to use the calibration toolbox in [1] to project points from world coordinates into camera coordinates.

## References

- [1] J.-Y. Bouguet. Camera calibration toolbox for matlab. [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/).
- [2] G. K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 00)*, volume 2,, pages 714–720, 2000.
- [3] C.-W. Chu, O. C. Jenkins, and M. J. Matarić. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, 2003.
- [4] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.
- [5] A. R. J. François and G. G. Medioni. Adaptive color background modeling for real-time segmentation of video streams. In *Proceedings of the International on Imaging Science, Systems, and Technology*, pages 227–232, 1999.
- [6] S. Penny, J. Smith, and A. Bernhardt. Traces: Wireless full body tracking in the cave. In *ICAT 99 Conference Proceedings*, 1999.
- [7] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings Computer Vision and Pattern Recognition Conference*, pages 1067–1073, 1997.
- [8] L. Sigal. Data set. 4 complete images, 4 background images, 4 filtered silhouette images.
- [9] R. Szeliski. Real-time octree generation from rotating objects. Technical Report CRL 90/12, Digital Equipment Corporation, Cambridge Research Lab, Dec 1990.
- [10] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision methodology using off-the-shelf tv cameras and lenses”. *Journal of Robotics and Automation*, Vol. RA-3(No.4):323–344, 1987.