Proceedings of Third International Workshop on Cooperative Distributed Vision

November 19–20, 1999

Kyoto, Japan

Sponsored by

Cooperative Distributed Vision Project
Japan Society for the Promotion of Science

Contents

Multi-perspective Analysis of Human Action

Larry Davis
Eugene Borovikov
Ross Cutler
David Harwood
Thanarat Horprasert
Department of Computer Sciene
University of Maryland
College Park, Maryland 20742 USA
e-mail: {lsd,yab,rgc,thanarat}@umiacs.umd.edu
http://www.umiacs.umd.edu/users/lsd/

Abstract

We describe research being conducted in the University of Maryland's Keck Laboratory for the Analysis of Visual Motion. The Keck Laboratory is a multi-perspective computer vision Laboratory containing sixty four digital, progressive scan cameras (forty eight monochromatic and sixteen single CCD color) configured into sixteen groups of four cameras. Each group of four is a quadranocular stereo rig consisting of three monochromatic and one color camera. The cameras are attached to a network of sixteen PC's used for both data collection and real time video analysis.

We first describe the architecture of the system in detail, and then present two applications:

1. Real time multi-perspective tracking of body parts for motion capture. We have developed a real time 3D motion capture system that integrates images from a large number of color cameras to both detect and track human body parts in 3D. A preliminary version of this system (developed in collaboration with ATR's Media Integration & Communications Research Laboratories and the M.I.T. Media Laboratory) was demonstrated at SIGGRAPH '98. That version, based on the W4 system for visual surveillance developed in our laboratory. We describe improved versions of the background modeling and tracking components of that system

2. Real-time volume intersection. Models of human shape can also be constructed using volume intersection methods. Here, we use the same background modeling and subtraction methods as in our motion capture system, but then utilize parallel and distributed algorithms for constructing an oct-tree representation of the volume of the person being observed. Details of this algorithm will be described.

1 Introduction

In this paper we describe ongoing research at the University of Maryland Computer Vision Laboratory on problems related to measuring human motion and activity using multi-perspective imaging. This research is being carried out in the Keck Laboratory for the Analysis of Visual Motion, a multiperspective video capture and analysis facility established with a grant from the Keck Foundation. In Section 2 of this report we describe the architecture of that Laboratory.

We can envision many applications in which a suite of cameras is employed to model or monitor an object or a small environment. Representative examples are work on multi-perspective stereo [1], space carving for volume reconstruction [2] and applications such as Georgia Tech's Smart Room [3, 4].

Our own work focuses on real time distributed algorithms for motion capture and gesture recognition. We describe two ongoing projects in recovery of articulated body models from multi-perspective video. In Section 3 we describe a feature based approach, in which each image in a multi-perspective suite of images is analyzed to identify the locations of principal body parts such as the head, hands, elbows, feet, etc. The three dimensional locations of those body parts is then determined by triangulation and trajectory smoothing. An early version of this system was demonstrated at SIGGRAPH in 1998. Finally, in Section 4 we present recent research on volumetric reconstruction using distributed volume intersection algorithm. Our current goals are to combine shape and color analysis to identify body parts and gestures in this volumetric representation.

2 Keck Laboratory Architecture

The Keck Laboratory for the Analysis of Visual Movement is a multi-perspective imaging laboratory, containing 64 digital, progressive-scan cameras organized as sixteen short baseline stereo rigs (see Figure 1). In each quadranocular rig, there are three monochromatic and one color camera. The cameras are connected to a network of PC's running Windows NT that can collect imagery from all of the cameras at speeds of up to 85 frames per second. The dimensions of the keck lab are 24' by 24' by 10'; a panoramic view of the lab is shown in Figure 2.

2.1 System design

A primary goal in the design of the Keck lab was to maximize captured video quality, while using commonly available hardware for economy. To meet this goal, uncompressed

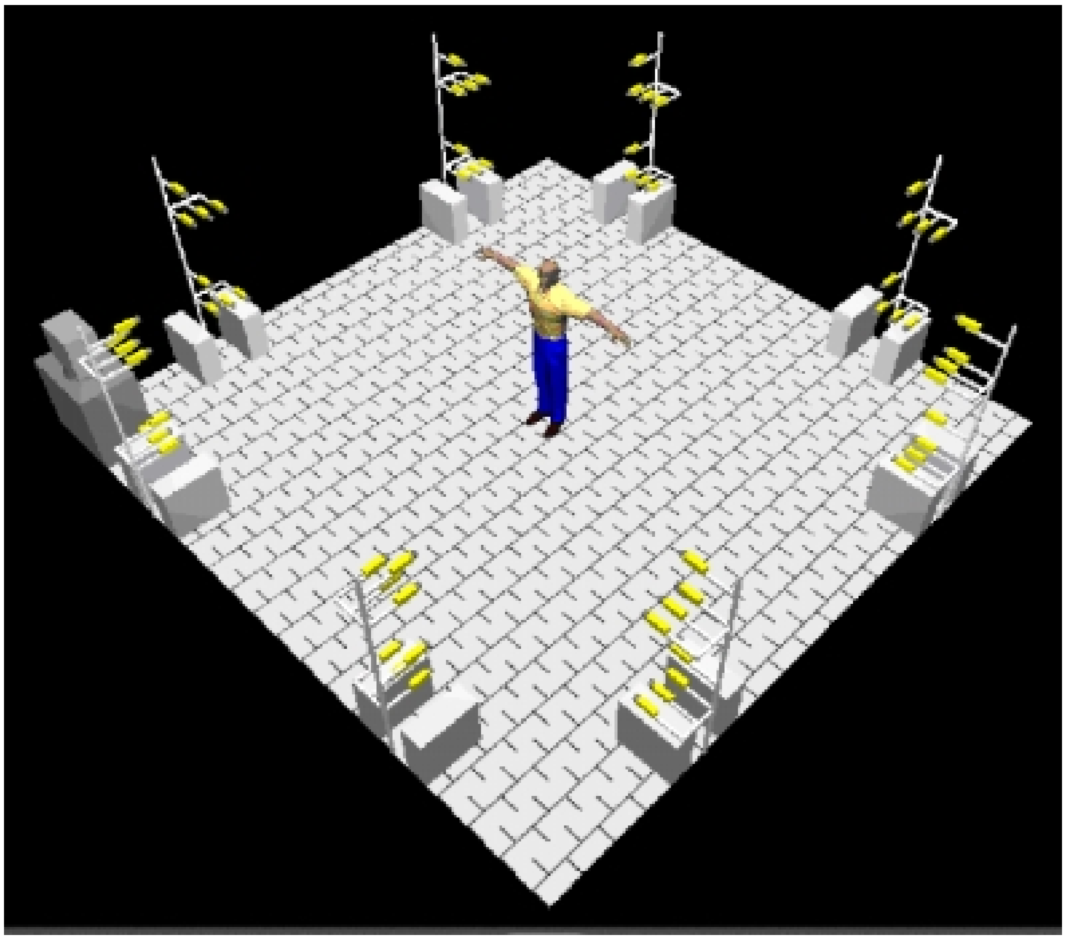

Figure 1: Keck Lab Architecture

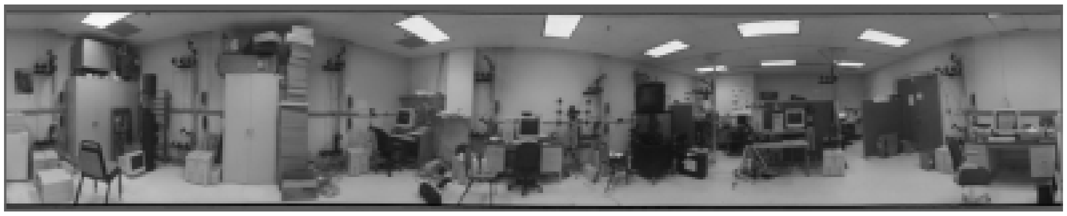

Figure 2: Keck Lab panorama

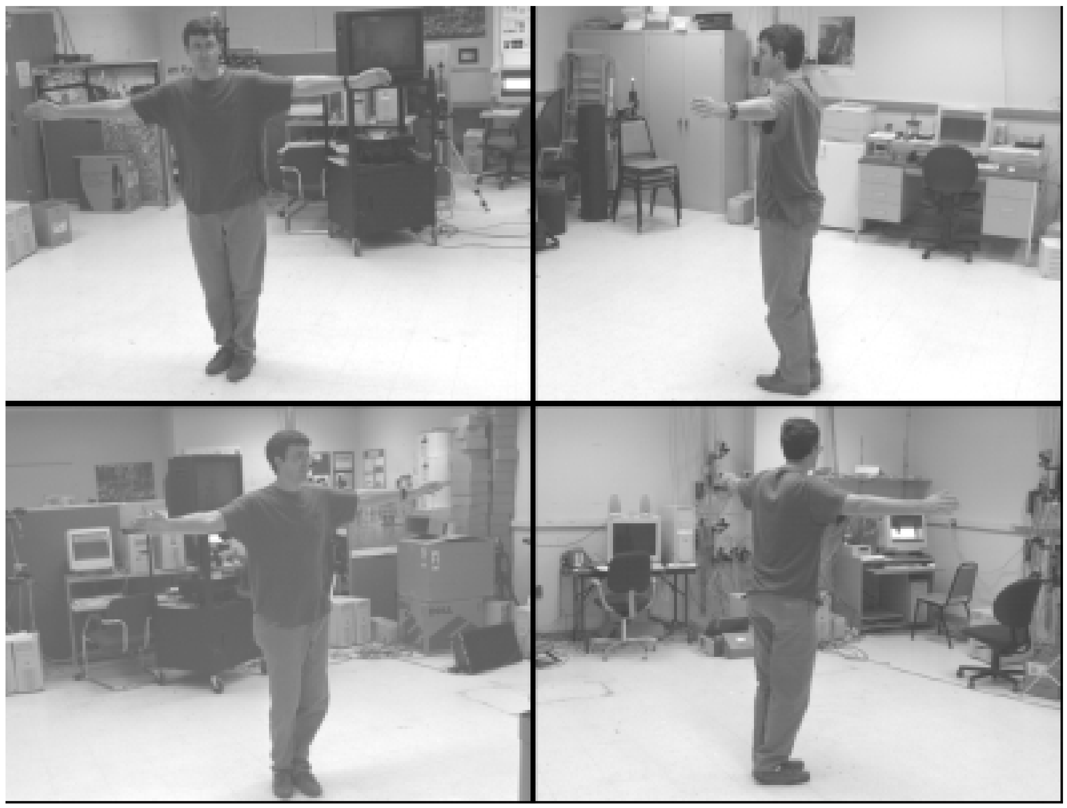

Figure 3: Keck Lab example images from four viewpoints

# cameras	FPS	Throughput (MB/s)
1	30	8.9
4	30	35.9
4	60	71.8
4	85	101.7

Table 1: Data throughput requirements

video is captured using digital, processive scan cameras directly to PCs. A schematic of the Keck lab is shown in Figure 4. The Keck lab was designed to capture uncompressed video sequences to both memory and disk. The data throughput requirements for various number of cameras and frame rates are shown in Table 1. The design of the Keck lab allows capturing uncompressed video to memory at up to 100 MB/s, and capturing to disk at up to 50 MB/s. In order to achieve the required 50 MB/s disk throughput, 3 SCSI Ultra 2 Wide disks (Seagate Cheetah) are used in a RAID configuation. Double the disk throughput could be achieved by writing a custom frame grabber device driver, which would write the images directly to the SCSI controller, instead of buffering the images to memory (which requires transmitting them over the PCI bus twice) [5].

The hardware used in the Keck lab includes the following:

- 64 digital 85 FPS progressive-scan cameras
  - 48 grayscale, Kodak ES-310
  - 16 color, Kodak ES-310C (Bayer color filter version of ES-310)
- 64 Schneider 8 mm C-mount lenses
- 64 Matrox Meteor II Digital frame grabbers
- 17 Dell 610 Precision Workstations
  - Dual Pentium II Xeon 450 MHz
  - 1 GB SDRAM, expandable to 2 GB
  - 9 GB SCSI Ultra II Wide hard drive
  - integrated 100 Mbps Ethernet interface
- Data Translation DT340 Digital IO board

- Peak Performance calibration frame
- 3 Apex Outlook monitor switches
- 21” Dell monitor
- 3COM 100 Mbps 24-port network switch
- Blackbox RS-485 interface adapter
- Quantum 35 GB Digital Linear Tape drive

The Kodak ES-310 cameras have a resolution of 648x484x8 and can operate at up 85 FPS in full frame progressive scan mode (speeds up to 140 FPS can be achieved using a smaller region of interest window). The ES-310 has a 10-bit digitizer for each pixel, in which the user can select which 8 bits are used for digital output. The ES-310 can be configured using either a RS-232 or RS-485 interface. In the Keck Lab, we have designed a RS-485 network to configure the 64 ES-310 cameras.

All 64 cameras are frame synchronized using a TTL-level signal generated by a Data Translation DT340. For video acquisition, the synchronization signal is used to simultaneously start all cameras. No timecode per frame is required.

2.2 Acquisition software

The software for video acquisition has been custom written for the Keck lab, using the following tools:

- Matrox Imaging Library 6.0
- Visual C++ 6.0
- Windows NT 4.0
- Data Translation DT340 SDK

The acquisition software uses a custom DCOM server, KeckServer, which runs on each of the 16 PCs. The controller PC makes connections with each of the camera PCs, and sends and retrieves messages and images. The ICamera interface used for the KeckServer is:

HRESULT ICamera::openCameras(char cameras, char *dcf) Opens the cameras specified by the bits in *cameras*, using the given Matrox DCF file.

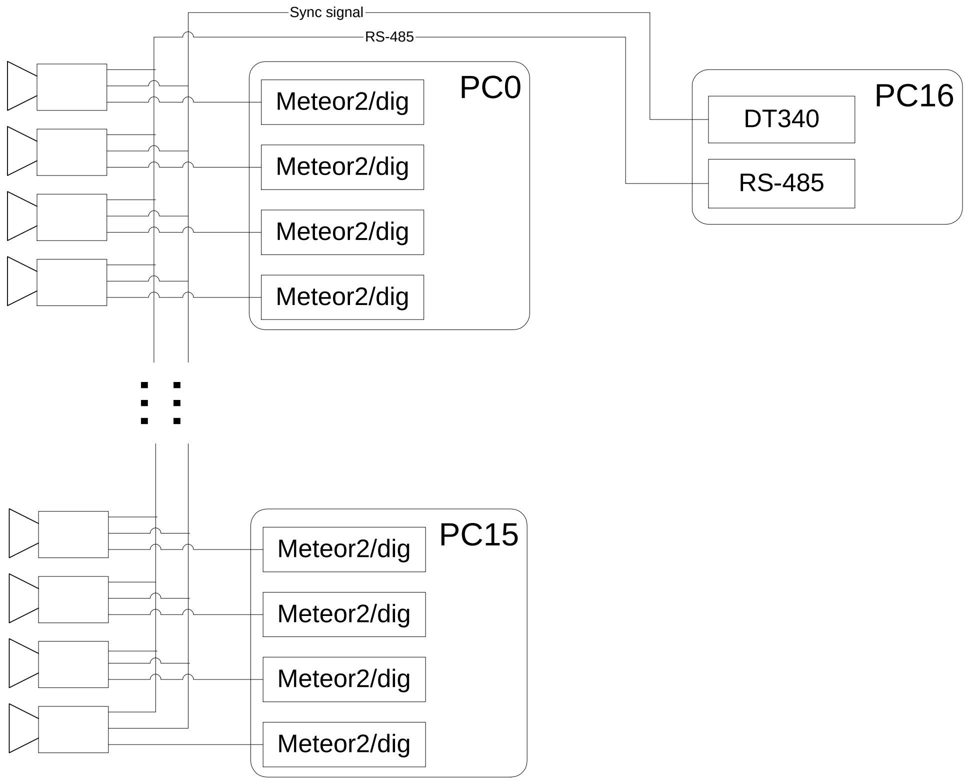

Figure 4: Keck Lab Schematic

# cameras	FPS	Max duration (sec)
1	30	99.8
4	30	24.9
4	60	12.5
4	85	8.8

Table 2: Maximum capture durations

HRESULT ICamera::closeCameras() Close any open cameras.

HRESULT ICamera::startCapture(int numFrames) Start a capture to memory for the specified number of frames.

HRESULT ICamera::saveCapturedSequenceToFile(char *fileName) Saves the captured memory sequences to an AVI file.

HRESULT ICamera::getLiveImage(int cameraNumber, int *imageSize, unsigned char **image) Returns the next live image (specified by the camera number) in the *image* buffer. The *image* buffer must be freed when it is no longer needed.

HRESULT ICamera::getCapturedImage(int cameraNumber, int imageNumber, int *imageSize, unsigned char **image) Returns the specified captured image in the *image* buffer. The *image* buffer must be freed when it is no longer needed.

2.3 Capabilities

The Keck lab is currently configured to capture up to 896 MB of video into upper memory (above the 128 MB allocated for Windows NT). This corresponds to 2995 648x484 frames. The maximum capture durations are given in Table 2. The capture durations can be increased by a factor of 2.14 by expanding the PCs from 1 GB to 2 GB.

The 450 MHz Pentium II is capable of 1800 MIPS, using the MMX operations [5]. With dual CPUs per PC, this provides significant computational power for real-time computer vision applications. The Dell 610 is capable of being upgraded to faster Pentium III processors, which would further increase the computational capabilities.

Each Dell 610 PC has a 100 Mbits/s Ethernet adapter, which is connected to a 3COM Ethernet switch. The effective throughput is such that each PC can communicate up to 10 MBytes/s to any other PC.

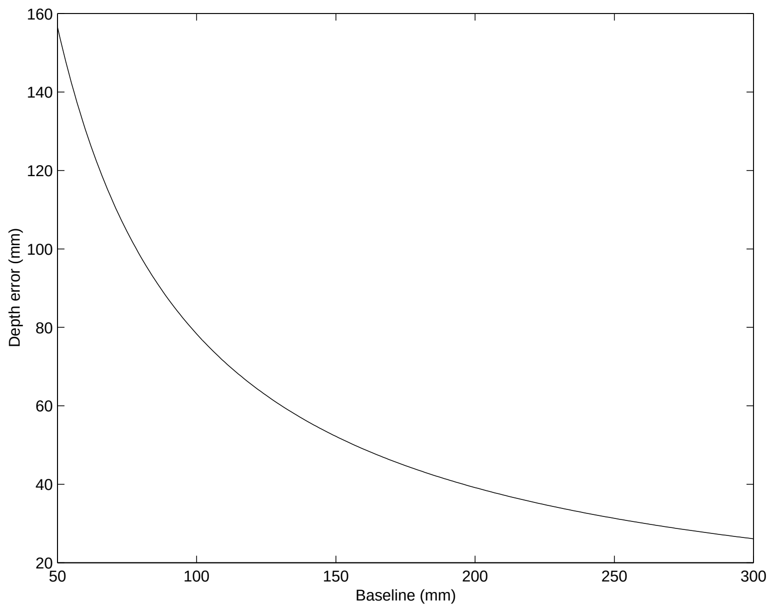

Figure 5: Stereo error analysis

2.4 Stereo error analysis

The quadranocular camera nodes of the Keck lab are designed to facilitate stereo depth computations. The trinocular baseline is adjustable from 150 to 300 mm. With a 300 mm baseline, a distance of 6' between the object and camera, and assuming single pixel correlation accuracy, then the depth precision is 26 mm. The depth precision for a range of baselines is given in Figure 5.

2.5 Lens distortion

In selecting the lenses for use with the Keck lab, we considered both the field of view and lens distortion (in general, as FOV increases, so does the lens distortion). We compared the image distortion for 3 commonly available C-mount lenses, using a line pattern commonly used for camera calibration purposes. From the images shown in Figure 6, the Schneider lens clearly had the least amount of distortion. Moreover, the Schneider lens was the only lens tested that did not have significant defocus near the perimeter of the images. Note that while certain types of lens distortion (e.g., radial) can be corrected in software, image defocus cannot be easily corrected, particularly within a real-time system.

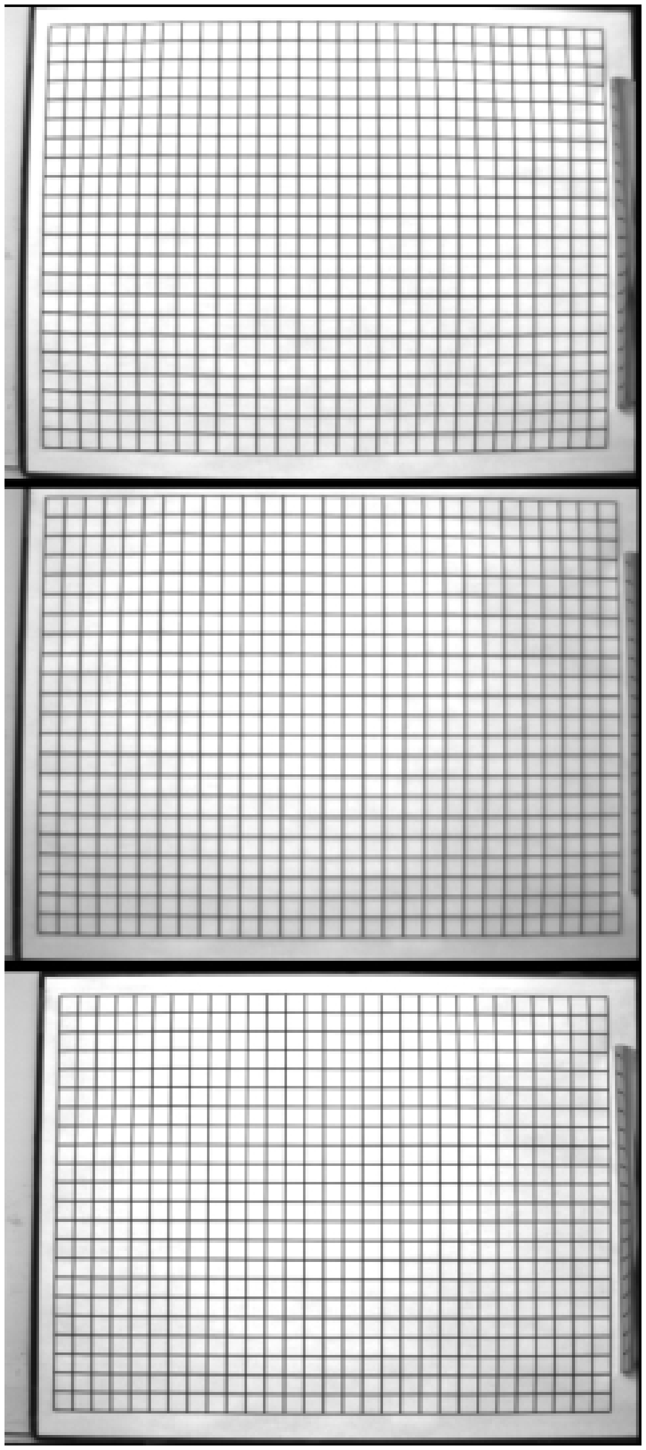

Figure 6: Lens distortion analysis images. Top: Cosmicar 6 mm, Middle: Canon 7.5 mm, Bottom: Schneider 8 mm.

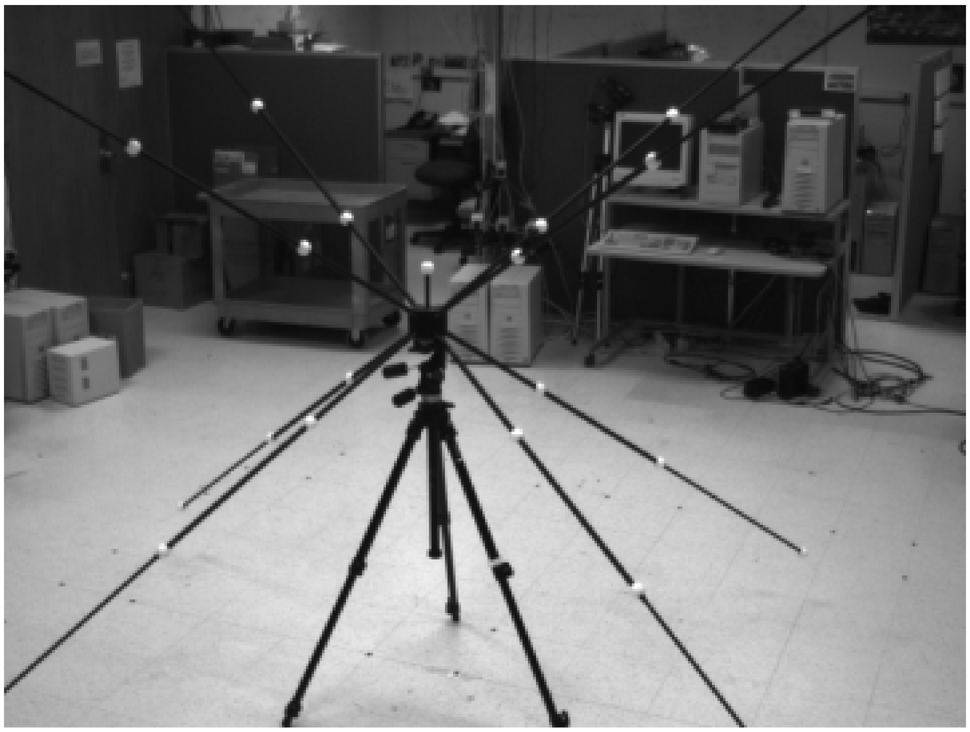

Figure 7: Peak Performance calibration frame

2.6 Calibration hardware

To facilitate strong calibration of the camera system, a Peak Performance calibration frame is utilized (see Figure 7). The calibration frame contains 25 white balls (1" in diameter), each of which has a known location accuracy of 1 mm. Additional hardware, such as a 1 m length wand with LED's at known locations, are also used for weak calibration.

3 Real-time 3-D Motion Capture System

Motion capture systems are used to detect any human movement and transfer that movement to 3-D graphical models used in animation for movies, games, commercials, etc. While motion capture is typically solved using magnetic systems and optical systems [6, 7], there exist mass market applications in which such solutions are untenable either due to cost or because it is impractical for people entering an environment to be suited up with active devices or special reflectors. Due to these restrictions of existing systems, a vision-based motion capture system which does not rely on contact devices would have significant advantages.

We have developed a real-time 3-D motion capture system that integrates images from a number of color cameras to detect and track human movement in 3D. It provides a person with control over the movement of a virtual computer graphics character. A preliminary version of this system (developed in collaboration with ATR's Media Integration & Communications Research Laboratories and the M.I.T. Media Laboratory) was demonstrated at SIGGRAPH'98 [8, 9].

3.1 System Overview

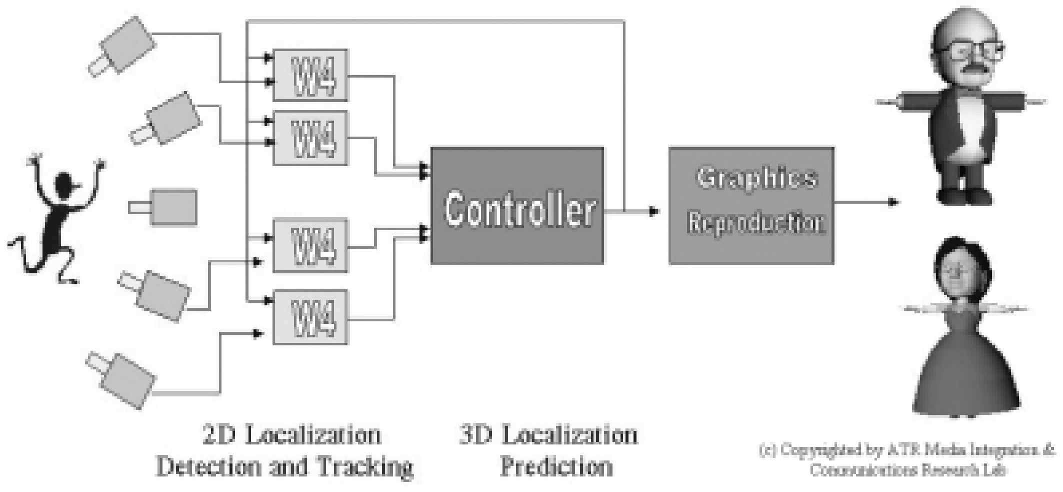

Figure 8: Block diagram of the system.

Figure 8 shows the block diagram of the system. A set of color CCD cameras observes a person. Each camera is attached to a PC running the W4 system [10]. W4 is a real-time vision system that detects people, and locates and tracks body parts. It performs background subtraction (described in detail in Section 3.2), silhouette analysis and template matching (described in Section 3.3)to locate and track the 2-D positions of salient body parts, e.g., head, torso, hands, and feet, in the image. A central controller obtains the 3-D positions of these body parts by triangulation and optimization processes. A lightweight version of the dynamical models developed by M.I.T.'s Media Laboratory [11] are used to smooth the 3D body part trajectories and predicted locations of those parts in each view. The graphic reproduction system developed by ATR's Media Integration & Communications Research Laboratories uses the body posture output to render and animate a cartoon-like character.

3.2 Background Modeling and Foreground Detection

One approach for discriminating a moving object from the background scene is background subtraction. The idea of background subtraction is to subtract the current image from a reference image, which is acquired from a static background during a training period. The subtraction leaves only non-stationary or new objects, which include the objects' entire silhouette region. The technique has been used in many vision systems as a preprocessing step for object detection and tracking, for examples, [10, 12, 9, 13, 14]. The results of the existing algorithms are fairly good; in addition, many of them run in real-time. However, many of these algorithms, including the original version of W4 (which was originally designed for outdoor visual surveillance system, and operates on monocular gray scale imagery), are susceptible to both global and local illumination changes such as shadows and highlights. These cause the consequent processes, e.g. tracking, recognition, etc., to fail. The accuracy and efficiency of the detection are clearly very crucial to those tasks. This problem is the underlying motivation of our extension to W4's background modeling described below.

3.2.1 Computational Color Model

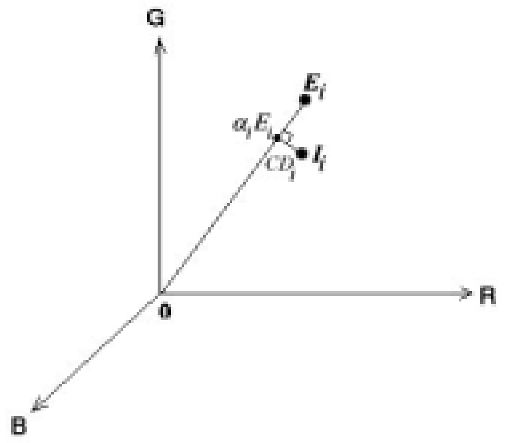

Figure 9: Our proposed color model in the three-dimensional RGB color space; the background image is statistically pixel-wise modeled. E_i represents an expected color of a given i^{th} pixel and I_i represents the color value of the pixel in a current image. The difference between I_i and E_i is decomposed into brightness (α_i) and chromaticity (CD_i) components.

Our background model is a color model that separates *brightness* from *chromaticity*. Figure 9 illustrates the proposed color model in three-dimensional RGB color space. Consider a pixel, i, in the image; let $E_i = [E_R(i), E_G(i), E_B(i)]$ represent the pixel's *expected RGB color* in the reference or background image. The line OE_i passing through the origin and the point E_i is called the *expected chromaticity line*. Next, let $I_i = [I_R(i), I_G(i), I_B(i)]$

denote the pixel's RGB color value in a current image that we want to subtract from the background. Basically, we want to measure the distortion of I_i from E_i. We do this by decomposing the distortion measurement into two components, *brightness distortion* and *chromaticity distortion*, defined below.

Brightness Distortion (α) The brightness distortion (α) is a scalar value that brings the observed color close to the expected chromaticity line. It is obtained by minimizing

$$\phi(\alpha_i) = (I_i - \alpha_i E_i)^2 \tag{1}$$

α_i represents the pixel's strength of brightness with respect to the expected value. α_i is 1 if the brightness of the given pixel in the current image is the same as in the reference image. α_i is less than 1 if it is darker, and greater than 1 if it becomes brighter than the expected brightness.

Color Distortion (CD) Color distortion is defined as the orthogonal distance between the observed color and the expected chromaticity line. The color distortion of a pixel i is given by

$$CD_i = \|I_i - \alpha_i E_i\| \tag{2}$$

3.2.2 Background Subtraction

The basic scheme of background subtraction is to subtract the image from a reference image that models the background scene. The steps of the algorithm are as follows:

- *Background modeling* constructs a reference image representing the background.
- *Threshold selection* determines appropriate threshold values used in the subtraction operation to obtain a desired detection rate.
- *Subtraction operation or pixel classification* classifies the type of a given pixel, i.e., the pixel is the part of background (including ordinary background and shaded background), or it is a moving object.

Background Modeling In the background training process, the reference background image and some parameters associated with normalization are computed over a number of static background frames. The background is modeled statistically on a pixel by pixel basis. A pixel is modeled by a 4-tuple $< E_i, s_i, a_i, b_i >$ defined below.

E_i is the expected color value of pixel i given by

$$E_i = [\mu_R(i), \mu_G(i), \mu_B(i)] \tag{3}$$

where $\mu_R(i)$, $\mu_G(i)$, and $\mu_B(i)$ are the arithmetic means of the i^{th} pixel's red, green, blue values computed over N background frames.

s_i is the standard deviation of color value defined as

$$s_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)] \tag{4}$$

where $\sigma_R(i)$, $\sigma_G(i)$, and $\sigma_B(i)$ are the standard deviation of the i^{th} pixel's red, green, blue values computed over N frame of the background frames.

We balance color bands by rescaling the color values by this pixel variation factors, s_i. Next, we consider the variation of the brightness and chromaticity distortions over space and time of the training background images. We found that different pixels yield different distributions of α and CD. These variations are embedded in the background model as a_i and b_i in the 4-tuple background model for each pixel, and are used as normalization factors.

a_i represents the variation of the brightness distortion of i^{th} pixel, which is given by

$$a_i = RMS(\alpha_i) = \sqrt{\frac{\sum_{i=0}^{N}(\alpha_i - 1)^2}{N}} \tag{5}$$

b_i represents the variation of the chromaticity distortion of the i^{th} pixel, which is given by

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{i=0}^{N}(CD_i)^2}{N}} \tag{6}$$

We then rescale or normalize the α_i and CD_i by a_i and b_i respectively. Let

$$\widehat{\alpha_i} = \frac{\alpha_i - 1}{a_i} \tag{7}$$

$$\widehat{CD}_i = \frac{CD_i}{b_i} \tag{8}$$

be the *normalized brightness distortion* and the *normalized chromaticity distortion* respectively.

Pixel Classification or Subtraction Operation In this step, the difference between the background image and the current image is evaluated. The difference is decomposed into brightness and chromaticity components. Applying the suitable thresholds on the brightness distortion (α) and the chromaticity distortion (CD) of a pixel i yields an object mask $M(i)$ which indicates the type of the pixel. Our method classifies a given pixel into four categories. A pixel in the current image is

- *Original background (B)* if it has both brightness and chromaticity similar to those of the same pixel in the background image.
- *Shaded background or shadow (S)* if it has similar chromaticity but lower brightness than those of the same pixel in the background image. This is based on the notion of the shadow as a semi-transparent region in the image, which retains a representation of the underlying surface pattern, texture or color value [15].
- *Highlighted background (H)*, if it has similar chromaticity but higher brightness than the background image.
- *Moving foreground object (F)* if the pixel has chromaticity different from the expected values in the background image.

Based on these definitions, a pixel is classified into one of the four categories B, S, H, F by the following decision procedure.

$$M(i) = \begin{cases} F & : \widehat{CD}_i > \tau_{CD} \quad or \quad \widehat{\alpha}_i < \tau_{\alpha lo}, \quad else \\ B & : \widehat{\alpha}_i < \tau_{\alpha 1} \quad and \quad \widehat{\alpha}_i > \tau_{\alpha 2}, \quad else \\ S & : \widehat{\alpha}_i < 0, \quad else \\ H & : otherwise \end{cases} \tag{9}$$

where τ_{CD}, $\tau_{\alpha 1}$, and $\tau_{\alpha 2}$ are selected threshold values used to determine the similarities of the chromaticity and brightness between the background image and the current observed image. $\tau_{\alpha lo}$ is a lower bound for the normalized brightness distortion. This is used to avoid a problem of dark pixels being misclassified as a shadow. Because the color point of a dark pixel is close to the origin in RGB space, and because all chromaticity lines in RGB space meet at the origin, a dark point is considered to be close or similar to any chromaticity line.

Automatic Threshold Selection Typically, if the distortion distribution is assumed to be a Gaussian distribution, then to achieve a desired detection rate,r, we can threshold the distortion by $K\sigma$ where K is a constant determined by r and σ is the standard deviation of the distribution. However, we found from experiments that the distribution of $\widehat{\alpha}_i$ and $\widehat{CD}_i$ are not Gaussian (see Figure 10). Thus, our method determines the appropriate thresholds by a statistical learning procedure. First, a histogram of the normalized brightness distortion, $\widehat{\alpha_i}$, and a histogram of the normalized chromaticity distortion, $\widehat{CD}_i$, are constructed as shown in Figure 10. The histograms are built from combined data through a long sequence captured during the background learning period. The total sample would be NXY values for a histogram. (The image is $X \times Y$ and the number of trained background frames is N.) After constructing the histogram, the thresholds are now automatically selected according to the desired detection rate r. A threshold for chromaticity distortion, τ_{CD}, is the normalized chromaticity distortion value at the detection rate of r. In brightness distortion, two thresholds ($\tau_{\alpha 1}$ and $\tau_{\alpha 2}$) are needed to define the brightness range. $\tau_{\alpha 1}$ is the $\widehat{\alpha_i}$ value at that detection rate r, and $\tau_{\alpha 2}$ is the $\widehat{\alpha_i}$ value at the $(1 - r)$ detection rate.

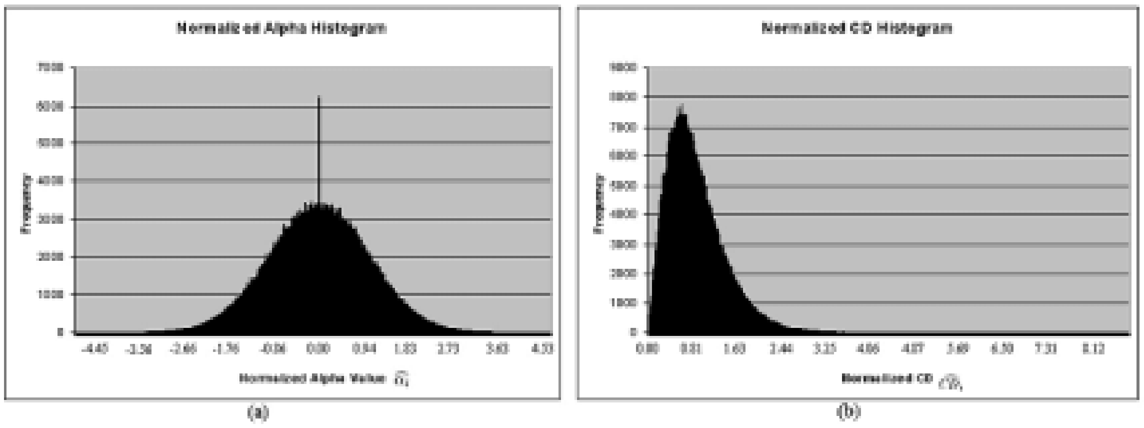

Figure 10: **(a)** is the normalized brightness distortion ($\widehat{\alpha_i}$) histogram, and **(b)** is the normalized chromaticity distortion ($\widehat{CD}_i$) histogram.

3.2.3 Background Subtraction Result

Figure 11 shows the result of applying the algorithm to several frames of an indoor scene containing a person walking around the room. As the person moves, he both obscures the background and casts shadows on the floor and wall. Red pixels depict shadows, and we can easily see how the shape of the shadow changes as the person moves. Although it is difficult to see, there are green pixels which depict the highlighted background pixels, appearing along the edge of the person's sweater.

Figure 12 illustrates our algorithm being able to cope with the problem of global illumination change. It shows another indoor sequence of a person moving in a room; at the middle of the sequence, the global illumination is changed by turning half of the fluorescence lamps off. The system is still able to detect the target successfully.

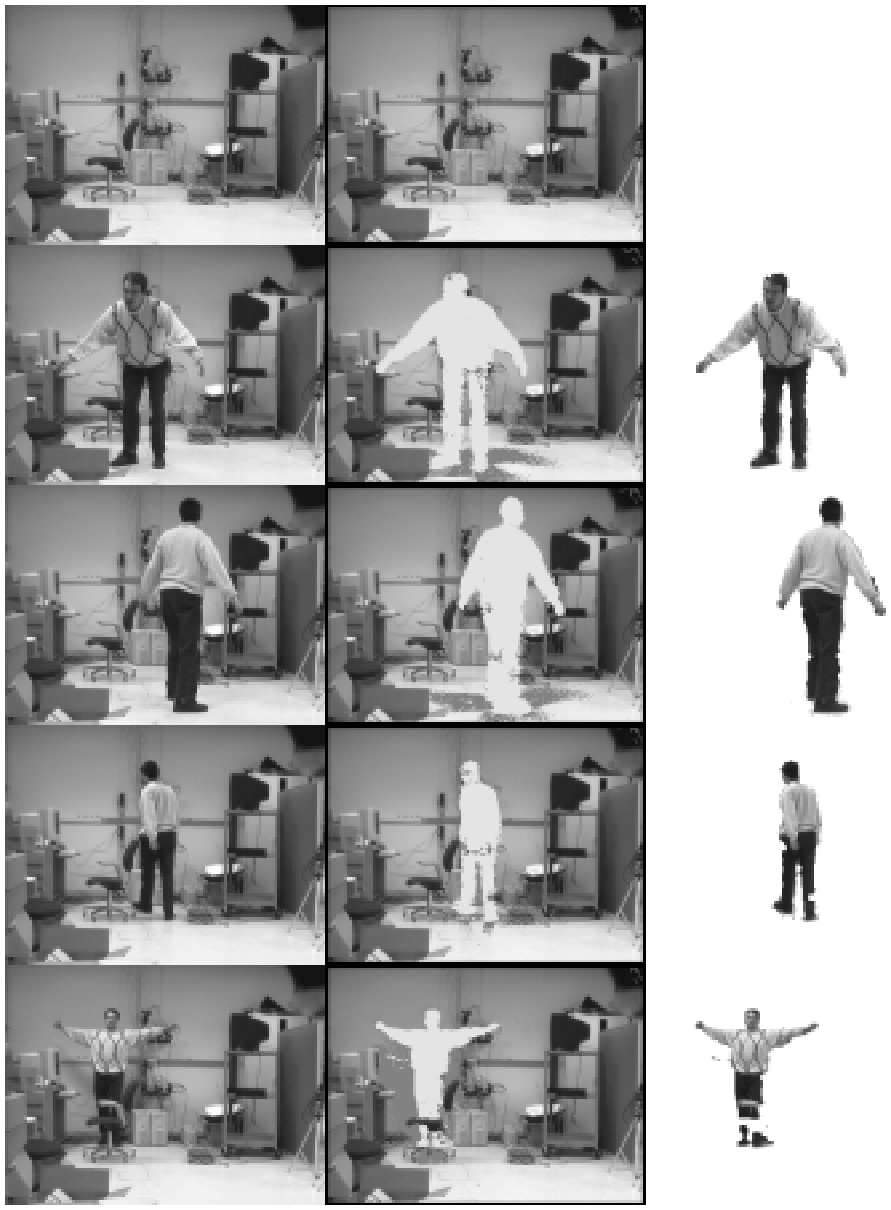

Figure 11: An example shows the result of our algorithm applying on a sequence of a person moving in an indoor scene. The upper left image is the background scene, the upper right image is the input sequence, and the lower left image shows the output from our background subtraction (the foreground pixels are overlaid by blue, the shadows are overlaid by red, the highlights are overlaid by green, and the ordinary background pixels are kept as the original color.) The lower right image shows only foreground region after noise cleaning is performed.

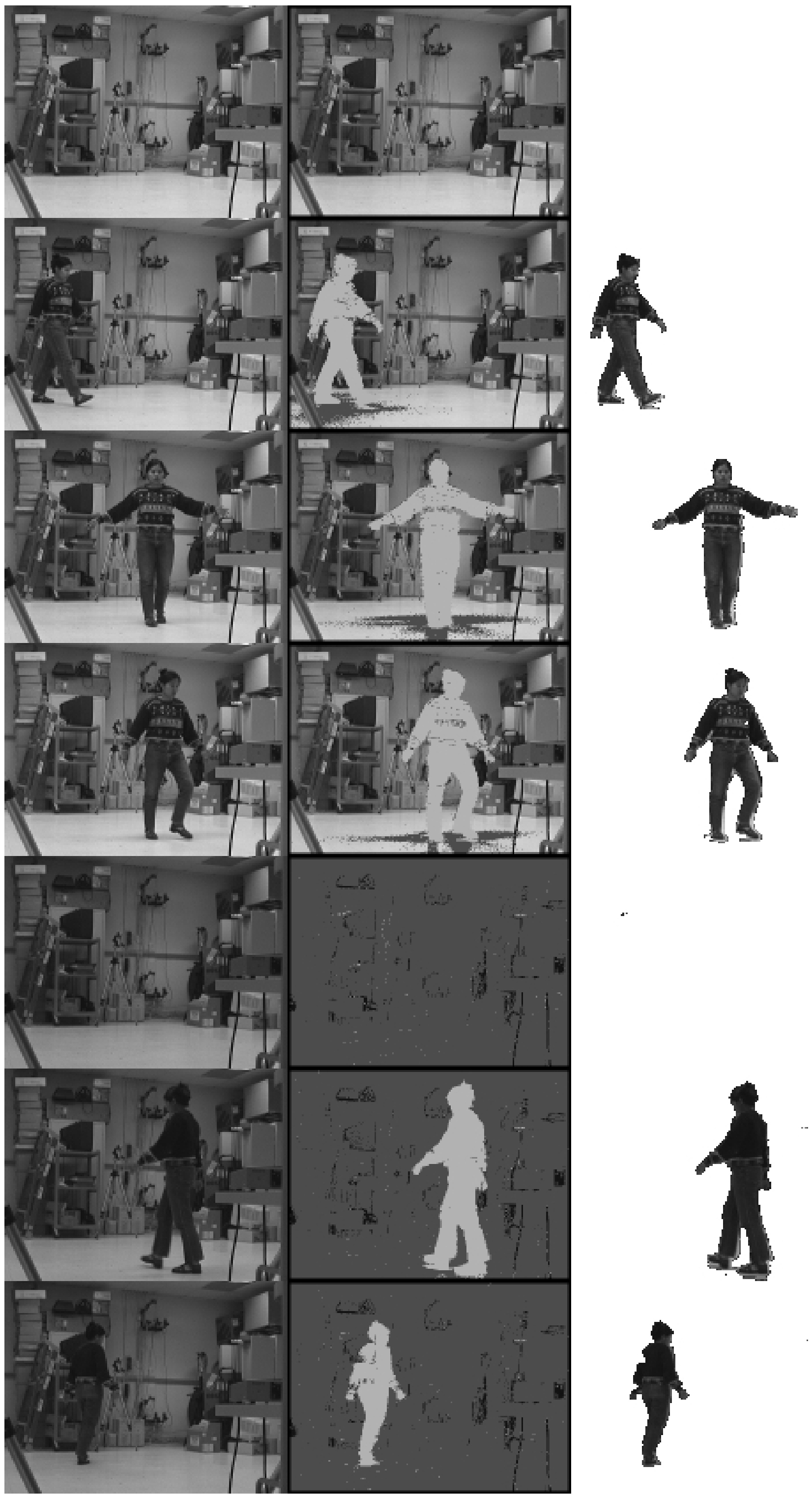

Figure 12: An illustration shows our algorithm can cope with the global illumination change. At the middle of the sequence, half of the fluorescence lamps are turned off. The result shows that the system still detects the moving object successfully.

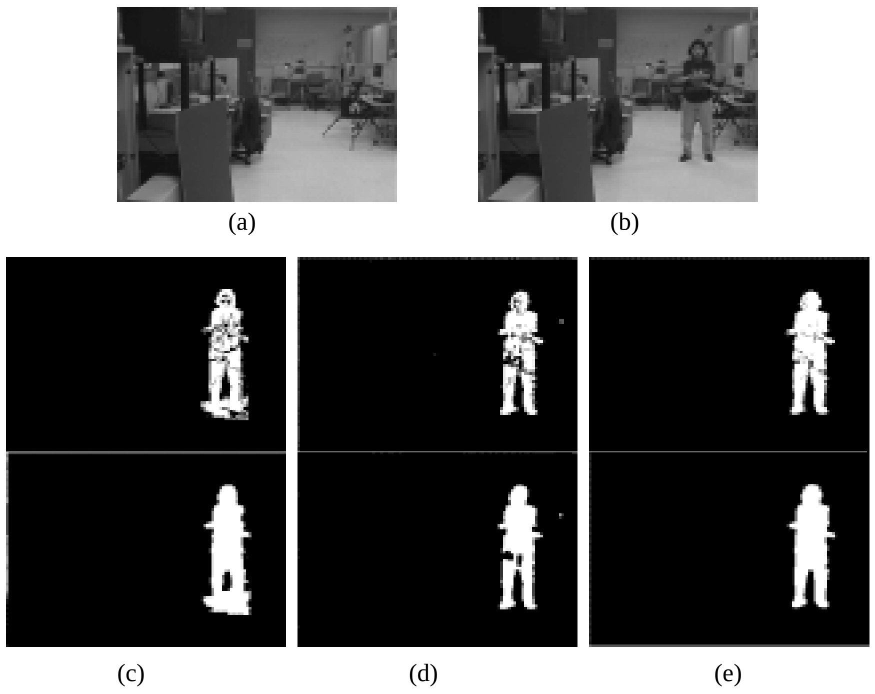

Figure 13: Comparison of the different background subtraction methods. (a) is an image of the background scene and (b) is an incoming image with a person moving in the scene. The results of the three mentioned methods -W4's gray-scale background subtraction (c), YIQ background subtraction (d), and the new method (e) are shown. The top row contains the intermediate results after thresholding, while the bottom row shows the final results after noise cleaning post-processing.

Note that sequences shown here are 320x240 images. The detection rate, r, was set at 0.9999, and the lower bound of the normalized brightness distortion ($\tau_{\alpha lo}$) is set at 0.4.

Figure 13 compares the results of our algorithm to two other methods used in W4 [10] and in [14]. W4's gray scale background subtraction model does not work well in an indoor environment with strong fluorescent light. The method of YIQ pixel classification used in [14] is too noisy. On the other hand, our new method works well, even against a complex background while it can be computed very efficiently for real-time applications.

3.3 Silhouette Analysis and 2-D Body Part Localization

W4's shape analysis and robust tracking techniques are used to detect people, and to locate and track their body parts (head, hands, feet, torso). The system consists of five computational components: background modeling, foreground object detection, motion estimation of foreground objects, object tracking and labeling, and human body parts locating and tracking. The background scene is statically modeled and the foreground region is segmented as explained in the previous section. A geometric cardboard human model [10] of a person in a standard upright pose is used to model the shape of human body and to locate the body parts (head, torso, hands, legs and feet).

3.3.1 Template Matching

After predicting the locations of the head and hands using the cardboard model and motion model (see Section 3.4.1), their positions are verified and refined using dynamic template matching. Multiple cues such as distance, color and shape which define feature appearance are used in matching. The template consists of three main regions: background border, foreground border, and foreground interior, see Figure 14. They are weighted differently in matching. Including foreground/background pixels in the matching helps to accurately locate the features. To combine shape information in matching, the color error is computed only at the pixel coordinates for which either the template or the image is a foreground pixel. Let $T_c(x)$ be the color of pixel x of the template, and $I_c(x)$ be the color of pixel x of the image. The color error of pixel x, $CE(x)$, is defined as:

$$CE(x) = \sum_{i=1}^{N} \left[w(i) \sum_{c=R,G,B} \left(\frac{T_c(i) - I_c(i)}{\sigma_c} \right)^2 \right]$$

for every pixel x such that $T(x)$ or $I(x)$ is a foreground pixel.

Next, for each pixel, the color error is normalized by subtracting a median color error (MCE) which is the median value of the error surface. This normalization allows us to compare the correlation peaks for the same feature across multiple views, and to estimate weights for the least square 3-D estimation of the features' location. Thus, the normalized color error is

$$\widetilde{CE}(x) = CE(x) - MCE$$

In addition, the distance error (DE) is combined. The distance error is the distance between the predicted location of the feature being tracked and the pixel coordinate. The final dissimilarity or total error (E) is defined as

$$E(x) = \widetilde{CE}(x) + DE(x)$$

Ideally, the matching result should yield a single sharp peak error surface. However, due to many factors such as motion blur and image blur, an error surface with multiple or shallow peaks can occur. We thus threshold the peaks to eliminate the outliers (bad peaks) and keep only the promising peaks (good peaks). The peak thresholding is defined as follows: The peak is a good peak if

$$MP - P > K \cdot MAD$$

where MP is the median error value of all peaks, P is the error value of the peak, MAD is the Median Absolute Difference of the error surface (defined below), and K is a constant.

$$MAD = median\{x : x = |MP - P_i| \quad \text{for all} \quad i \quad \text{peaks.}\}$$

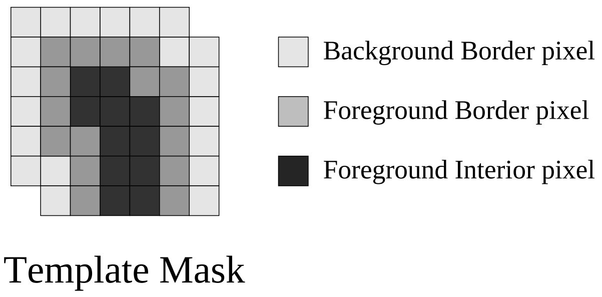

Figure 14: Representations of template and image for matching.

After finding the best match, the color templates of the body parts are then updated unless they are located within the silhouette of the torso. In this case, the pixels corresponding to the head and hand are embedded in the larger component corresponding to the torso. This makes it difficult to accurately estimate the position of the part, or, more important, to determine which pixels within the torso are actual part pixels. In these cases, the parts are tracked using correlation, but the templates are updated using only skin color information and the location prediction comes from the 3-D controller. Figure 15 illustrates the body part localization algorithm.

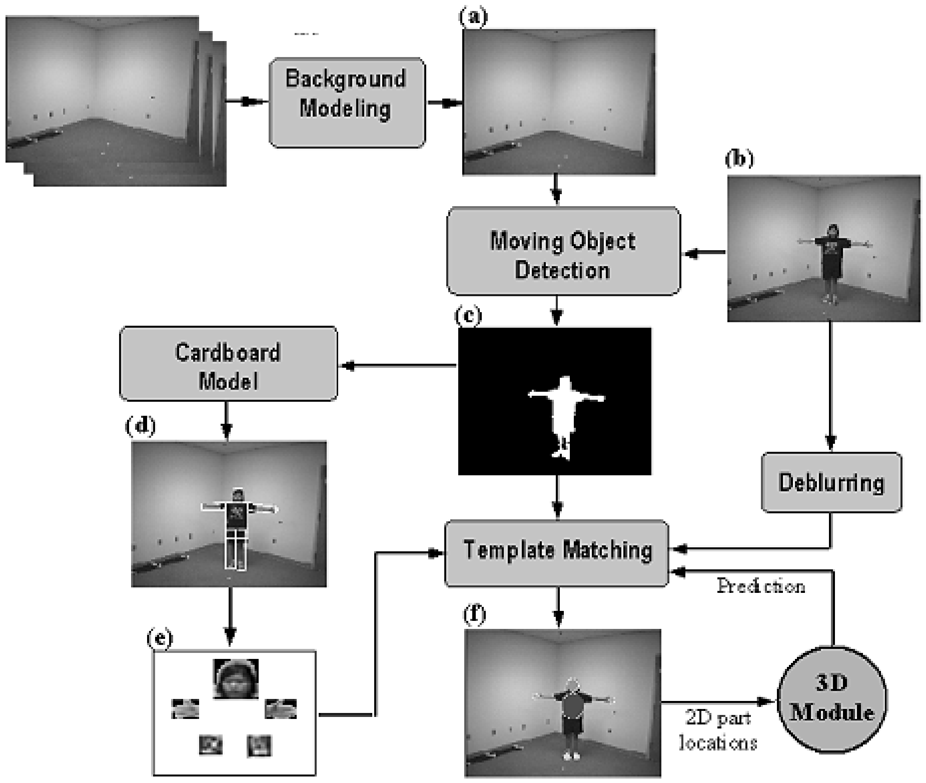

Figure 15: 2-D body part localization process diagram. First, background scene is modeled (a). For each frame in the video sequence (b), the foreground region (c) is segmented by the new method of pixel classification. Base on the extracted silhouette and the original image, the cardboard model is analyzed (d) and salient body part templates are created (e). Finally, these parts (head, torso, hands and feet) are located by a combined method of shape analysis and color template matching (f).

3.4 3-D Reconstruction and Human Motion Model

By integrating the location data from each image, the 3-D body posture can be estimated. First, the cameras are calibrated to obtain their parameters. For each frame in the sequence, each instance of W4 sends to a central controller not only the body part location data but also a corresponding confidence value that indicates the level of confidence of its 2-D localization for each particular part. The confidence value is obtained from the similarity score of the template matching step. The controller then computes the 3-D localization of the body part by performing a least square triangulation over that set of 2-D data which has confidence values higher than a threshold. We treat each body part separately; i.e. at a certain frame, the 3-D position of the right hand and the left hand may be obtained from triangulation of different subsets of the cameras. A linear optimization method for camera calibration and triangulation [16, 17] are employed here.

3.4.1 Motion Model and Prediction

The knowledge that the system will be tracking a human body provides many useful constraints because humans only move in certain ways. To constrain the body motion and smooth the motion trajectory, a model of human body dynamics [11] developed by MIT's Media Lab was first employed. However, the framework, while powerful, was computationally too expensive, especially when applied to the whole body. Thus, we experimented with a computationally light-weight version that utilizes several linear Kalman filters in tracking and predicting the locations of the individual body parts. This system required much less development time than the full dynamic model. These individual filters are then linked together by a global kinematic constraint mechanism. The linear Kalman filters approximate the low-level, dynamic constraints while the global constraint system maintains the kinematic constraints. We found that this optimization provides sufficient predictive performance while making the system computationally more accessible and easier to construct. These predictions are then fed back to the W4 systems to control their 2-D tracking.

3.5 3D Motion Capture System Result

Figure 16 demonstrates the system's performance on some key frames in a video sequence. Figure 17 shows our demonstration area at SIGGRAPH'98. The cameras were placed in a semi-circle arrangement pointing toward the dancing area. A projector was placed next to the dancing area and displayed the animated graphical character. In the demonstration, a person entered the exhibit area and momentarily assumed a fixed posture that allowed

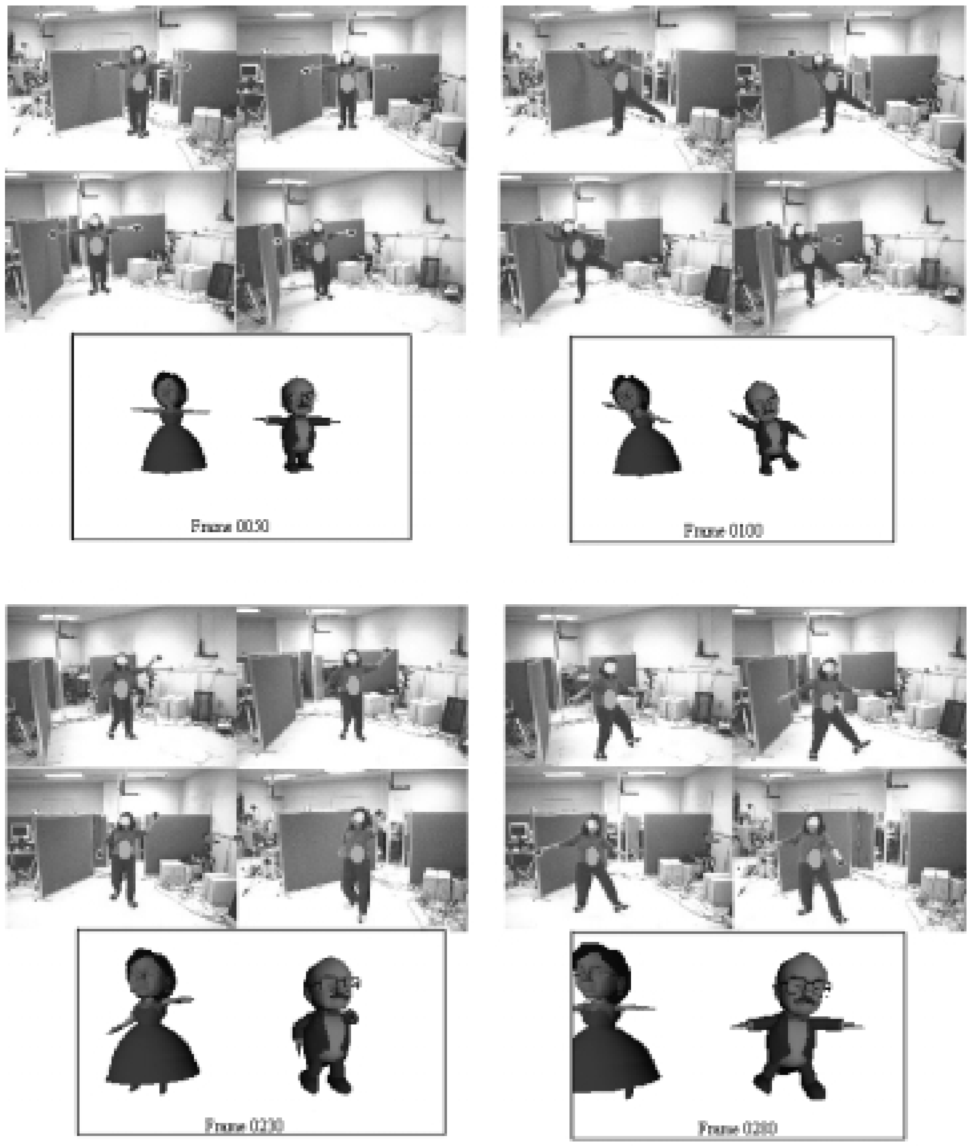

Figure 16: An illustration of our system's result on some key frames in the video.

the system to initialize (i.e. locate the person's head, torso, hands, and feet). They were then allowed to "dance" freely in the area. The trajectories of their body parts are used to control the animation of a cartoon-like character developed by ATR. Whenever the tracking fails, the person can himself reinitialize the system by assuming the fixed posture at the center of the demonstration area. The demonstration attracted many attendees. Although our original target audiences were young children and young adults, it turned out that the system also appealed to aged people as well as mass media people.

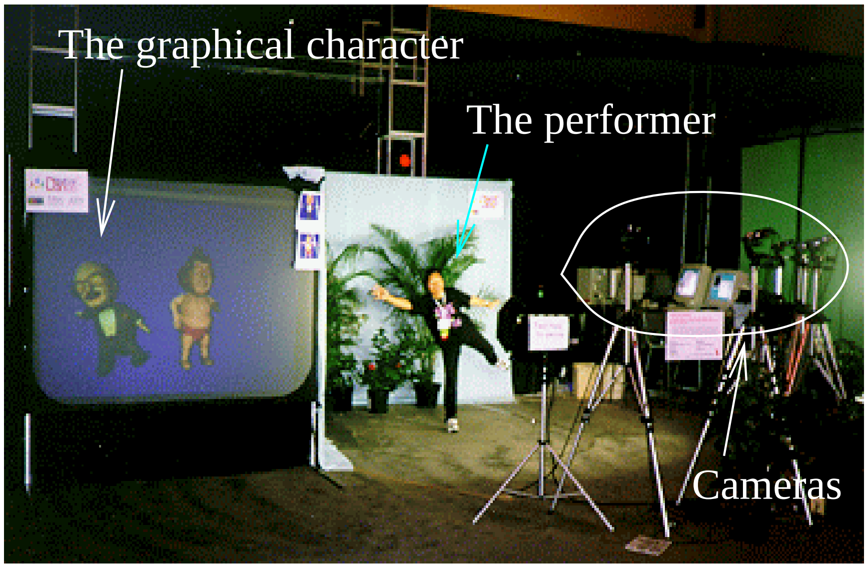

Figure 17: A snap-shot of the demonstration at SIGGRAPH.

4 Real-time Volume Reconstruction

Volume reconstruction techniques can be employed to recover 3D shape information of various objects, natural or man-built. An objective of our ongoing research is constructing, in real time, human body shape models for subsequent gesture and action recognition. Such models can be efficiently constructed using volume reconstruction methods. We utilize parallel and distributed algorithms for constructing an oct-tree representation of the volume of a person (or any object, for that matter) being observed. The volume reconstruction procedure utilizes a multi-perspective view of the scene, and consists of the following steps:

- camera calibration
- background modeling, and object silhouette extraction as described in the previous section

- volume reconstruction via silhouette visual cone intersection
- volumetric data interpretation and visualization

Notice that not all of the above steps have to be done in real-time. Camera calibration and background modeling are preliminary steps toward the volume reconstruction itself, and therefore can be done off-line.

4.1 Camera Calibration

An accurate camera calibration method is critical if the visual cone intersection procedure is to produce fine-detailed 3D shape estimation. We utilize an implementation of Tsai's camera calibration algorithm [18], using a non-coplanar calibration procedure. It accepts as input about 25 3D space points along with their corresponding projections to the image planes, and produces as output the estimated camera calibration parameters, both intrinsic and extrinsic. Our estimated average error in the object space is about 3 mm. The accuracy can be improved somewhat by computing the projections of the feature points with sub-pixel precision.

4.2 Background Modeling

The on-line silhouette extraction procedure uses background subtraction, which in turn employs a pre-computed background model. In the environment of our lab, the 3D scene is viewed by both color and gray-level cameras, and our volume reconstruction system is able to extract object silhouettes from both color and gray-level image sequences via two different kinds of background models. Both are statistical pixel-wise models, but they differ in the way they are built and used. The color model was described in the previous section.

The gray level model combines intensity and range data. The narrow-base stereo cameras in the Keck Lab are used to build a background range map using a simple correlation based stereo algorithm. For both gray level and range, each background pixel is modeled by a 3-tuple $< min, max, max - consecutive - difference >$. More detailed information about this background model is found in [10]. Notice that the use of the range model increases the robustness of the gray-level background subtraction, eliminating unwanted shadows.

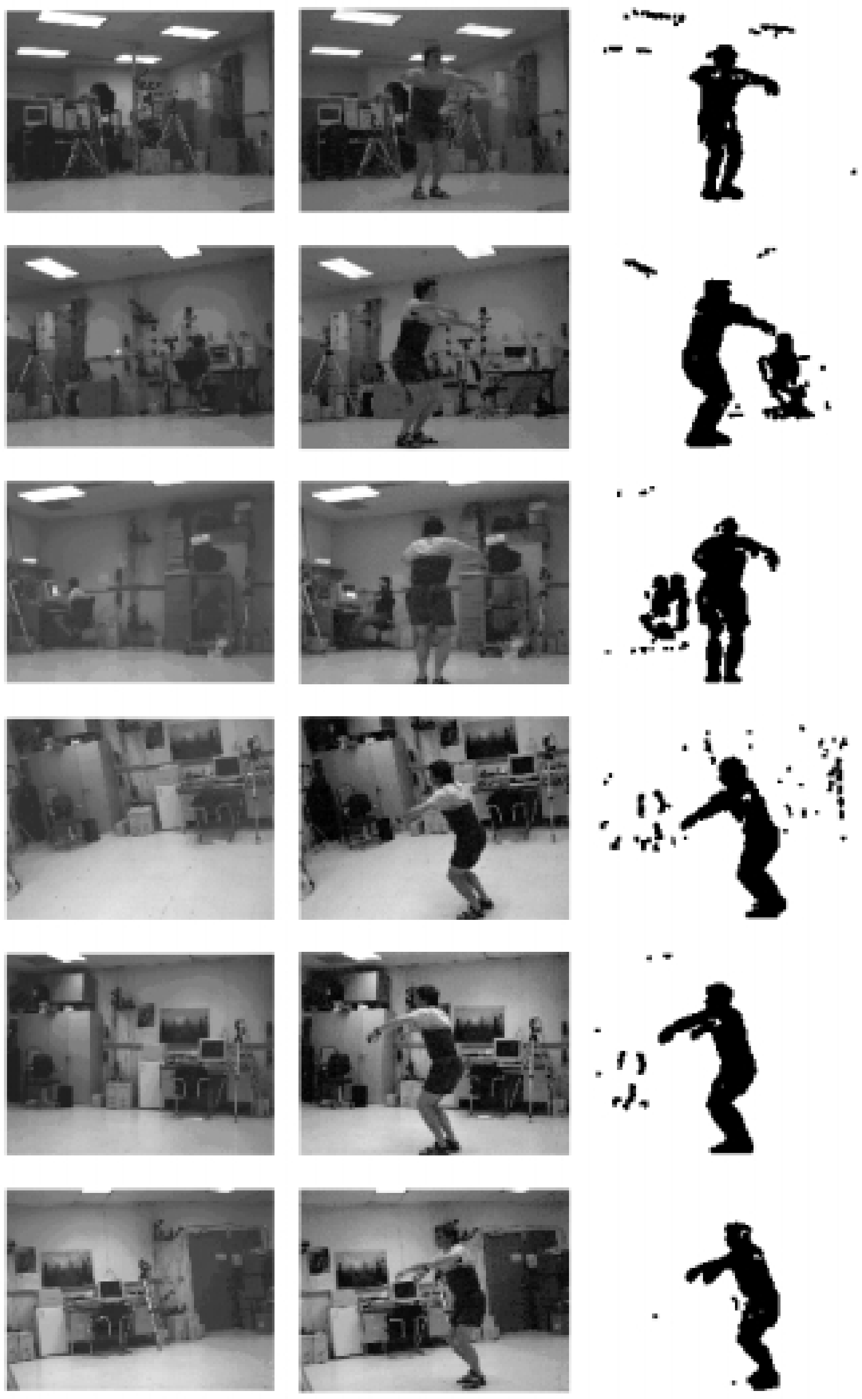

Figure 18: A multi-perspective snapshot of the background, a person in the "standing up" sequence, and the extracted silhouettes.

4.3 On-line Processing

Silhouette extraction and visual cone intersection are done on-line. The high-level picture is as follows: the computation is initiated at one of the nodes, which becomes the *manager*; the rest of the participating nodes become the em workers. Notice that there could be any number of participating nodes.

The worker processes grab frames, extract silhouettes, and intersect visual cones, while the manager process coordinates the overall volume reconstruction procedure and organizes the results. To put it simply, the workers reconstruct what they see, while the manager gathers the results and renders the volume. Notice that the manager may also capture and process frames, but this will reduce system performance unless the manager is a multi-processor computer that can dedicate some of its processors for volume rendering and other processors for capturing.

4.3.1 Silhouette Extraction

The silhouette extraction procedure uses background subtraction with adaptive thresholds. In the color case, a pixel is classified by applying suitable thresholds to its brightness distortion and chromaticity distortion values. In this way, it is possible to split pixels into four classes: original background, shaded background, highlighted background, and foreground. The foreground pixels form the candidate pool for the foreground object silhouette. In the gray-level case, the candidates for foreground regions are segmented in both the intensity and the disparity images. Then the significantly overlapping regions are intersected to form the foreground object silhouette. This produces a more accurate silhouette and eliminates unwanted shadows. In both cases (color and gray-level), some post-processing is done to reduce noise and make the extracted silhouette more precise. More details of both algorithms are found in [10] and [19]. Refer to Figure 18 to see the results of the background subtraction in the color case. Notice that the foreground object silhouette is well extracted and there is no ”shadow carpet” underneath. There is, however, some noise in the silhouette images primarily due to the fact that the background was not entirely static. This noise is dealt with by the robust visual cone intersection procedure.

4.3.2 Visual Cone Intersection

Visual cone intersection is done efficiently using a distributed algorithm that runs on a PC cluster. Each worker process is assigned a view for which it extracts the foreground object silhouette and builds a visual cone. This way, the visual cones are constructed in parallel. Once they are completed, the nodes exchange visual cones and build the final octree. Notice that each node builds a copy of the final octree in parallel. As soon as the octree is ready, the manager starts rendering andor storing the volume while the workers process the next frame and build the next set of visual cones. This process loops until the frame pool is exhausted or the manager terminates the computation. The following paragraphs describe each step of the algorithm in more detail.

Octree Construction from Multiple Views A visual cone is represented by an octree, which is built via the rapid octree construction algorithm suggested by Szeliski [20]. First, the background is subtracted from the newly captured frame to obtain the foreground object silhouette. Then the algorithm traverses the octree to a given depth and computes the occupancy attribute (opaque, transparent, or half-transparent) of each volume element in a hierarchical fashion. A voxel is transparent if its projection lies entirely outside the silhouette; and, similarly, a voxel is opaque if its projection lies entirely inside the silhouette. If voxel transparency cannot be decided at the present level, it is considered to be half-transparent, and the algorithm proceeds with computing transparencies for the voxel's children. Once the traversal is complete, the visual cone is ready to be exchanged with the rest of the cluster.

As soon as each worker process receives a complete set of the visual cones, the final octree is built as the intersection of the received octrees. The octree intersection algorithm traverses all given visual cone octrees at the same time in the depth-first search manner. The same branch is followed in all octrees until one of them reports a transparent leaf. At this point the final tree's branch is trimmed and another one is explored. The process continues until all branches are explored.

When the final octree is completed, the manager renders it in 3D, possibly using voxel coloring andor texture mapping techniques. The workers, however, proceed with the next frame, and the process repeats.

Voxel Projection and Transparency Test Details The above procedure has two potential "bottle necks": voxel projection and the transparency test. Without specialized parallel hardware, voxel projection could be expensive since it involves projection of eight voxel vertices with subsequent computing of a convex hexagon to represent the voxel's projection to the image plane. Intersecting the hexagonal projection with the silhouette image to decide the voxel's transparency is also a source of significant computation.

Szeliski proposes an efficient method to decide whether a voxel's projection is entirely inside or entirely outside the silhouette. Given a voxel, the system computes the bounding square s of the voxel's projection. Then the pre-computed half-distance transform map of the silhouette image is used to decide whether s entirely lies within the foreground/background region. The half-distance transform map is a one-sided version of the chessboard distance transform map. Each point in the half-distance transform map contains the size of the largest square rooted at that point that fits entirely within the foreground region. Figure 19 gives an example of a binary image and its half-distance transform.

```
. . . . . . . . . . . . . . .      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
. 1 1 . 1 1 1 1 1 . . . . . .      0 1 1 0 5 4 3 2 1 0 0 0 0 0 0
. . 1 1 1 1 1 1 1 1 . . . . .      0 0 1 4 5 4 3 2 1 1 0 0 0 0 0
. . . 1 1 1 1 1 1 . . . . . .      0 0 0 3 5 4 3 2 1 0 0 0 0 0 0
. . . 1 1 1 1 1 1 . . . . . .      0 0 0 2 4 4 3 2 1 0 0 0 0 0 0
. . . 1 1 1 1 1 1 . . . . . .      0 0 0 1 3 3 3 2 1 0 0 0 0 0 0
. . . . 1 1 1 1 1 1 . . . . .      0 0 0 0 2 2 3 2 1 1 0 0 0 0 0
. . . 1 1 1 1 1 1 . . . . . .      0 0 0 0 1 1 2 2 1 0 0 0 0 0 0
. . . . . . 1 1 1 1 1 1 1 . .      0 0 0 0 0 0 1 1 2 2 2 1 1 0 0
. . . . . . . . 1 1 1 1 . . .      0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
. . . . . . . . . . . . . . .      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 19: A binary image (left) and its half-distance transform map (right)

The half-distance transform maps are pre-computed for both the silhouette image (positive map) and its complement (negative map). The bounding box of the voxel's projection is tested against both positive (for inclusion) and negative (for exclusion) maps to determine the voxel's occupancy. If neither of the above tests is successful, the voxel's occupancy is undetermined, and transparencies of its children are left to decide at the next iteration of the algorithm.

Another time-consuming operation in the volume reconstruction procedure is the computation of voxel projections. There are at least two ways to speed it up: using specialized (parallel) hardware or employing lookup tables. With the absence of specialized hardware, we use pre-computed half-distance transform lookup tables (HDTLT) containing sizes of voxel projections that are used in the half-distance transform maps to determine inclusion or exclusion of a voxel. Usage of HDTLT dramatically speeds up the volume reconstruction procedure (by almost an order of magnitude, in our case) but its size makes it impractical to share it over the network. An HDTLT is a full octree and its size grows exponentially (as $O(8^d) = O(2^{3d})$) with respect to the depth parameter d. A typical size of an HDTLT of depth 8 is about 28MB. Therefore, visual cones need be computed "distributedly" and then shared with the manager of computation.

4.4 Volumetric Data Visualization and Interpretation

Currently, volume visualization and interpretation are done off-line. The application simply outputs all the leaves of the final octree to an 'Open Inventor' ascii file that can be opened and viewed off-line. Some snapshots of the reconstructed volume are shown in Figure 20. Notice that although the silhouette images were quite noisy, the 3D object was reconstructed correctly and almost all of the noise is gone due to the robustness of the visual cone intersection procedure.

A real-time application, however, needs a more efficient way of rendering the volume. To render an octree efficiently, we are considering techniques that employ graphics accelerators to run volume rendering algorithms similar to the ones described in [21] and [22]. If color information is available, some efficient texture mappingvoxel coloring methods [23] can be applied to better visualize the reconstructed volume.

4.5 Experimental Results

With the techniques described, we developed both sequential and distributed (parallel) systems. In all test cases the volume was reconstructed to the silhouette image resolution, which corresponds to 8 levels of the octree having the smallest voxel side of 1.5 cm. In the sequential case, the program runs on a Pentium II 300 MHz PC with inputs from multiple cameras simulated via disk files. The program was able to reconstruct the object's volume (given input from 6 virtual cameras supplying 320x240 silhouette images) in 20 ms on average. This figure does not include the time for frame grabbing, preprocessing (e.g. silhouette extraction) and volume rendering. The distributed system runs on a PC cluster consisting of Pentium III 400MHz computers inter-connected via a 100Mbit/s-bandwith TCP/IP network. In a test for three cameras, the visual cones were constructed in 10 ms; the visual cone exchange took about 100 ms, and the final octree construction took another 10 ms. For six cameras, the timing for visual cone construction and intersection did not change, but the communication overhead grew up to 200ms. Reduction of this term through oct-tree compression is the objective of our current implementation effort.

Acknowledgements The support of MURI under grant NAVY N-0001-495-10521 and Keck Foundation are gratefully acknowledged.

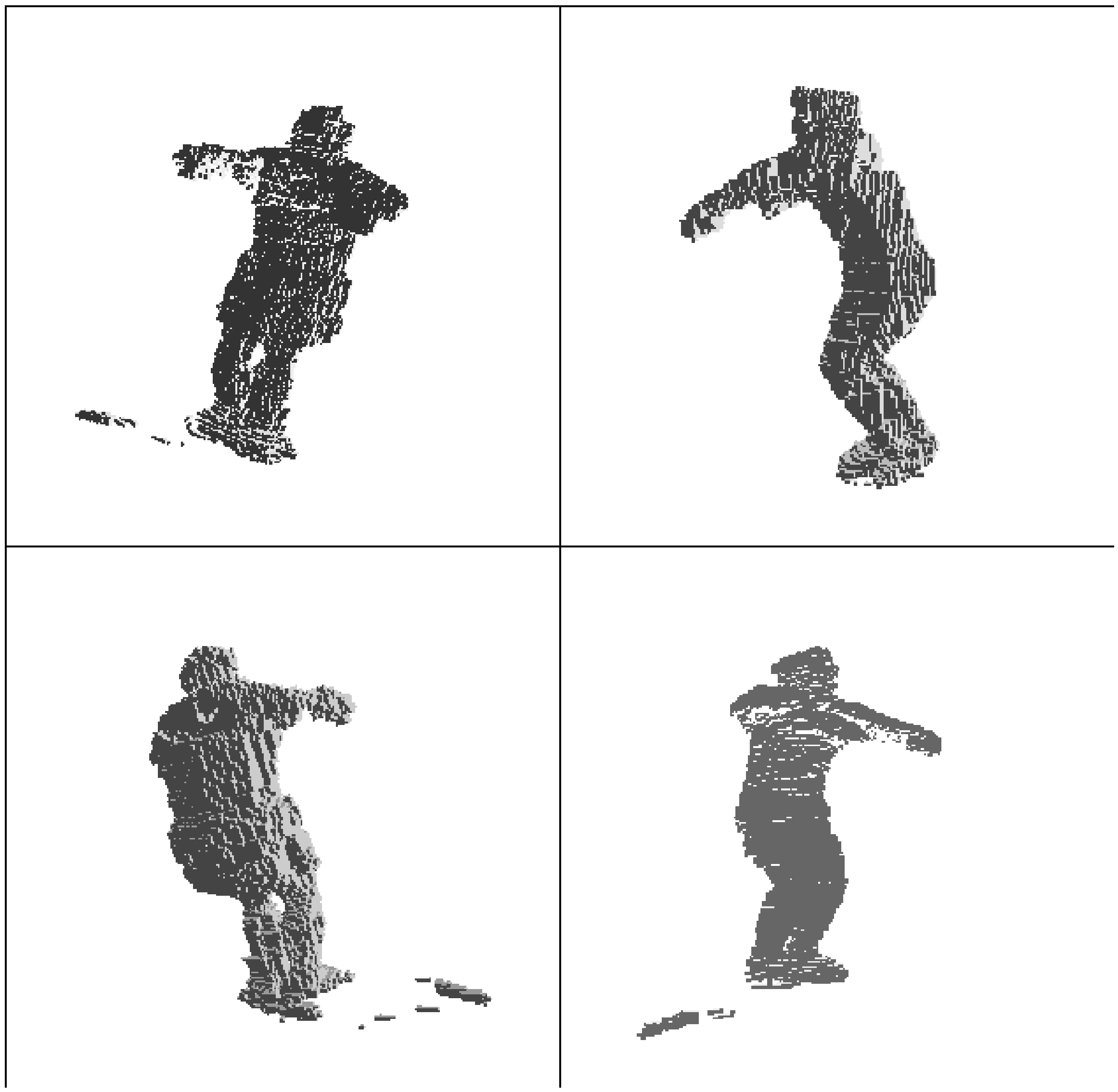

Figure 20: The reconstructed volume of a person's body viewed from different virtual points.

References

[1] P.J. Narayanan, P.W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. IEEE Int'l Conf. on Computer Vision*, pages 3–10. IEEE Computer Society Press, Los Alamitos, Calif., 1998.

[2] K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. Technical report, University of Rochester Computer Sciences Department, 1998.

[3] I. Essa, G. Abowd, and C. Atleson. Ubiquitous smart space. *A white paper submitted to DARPA*, 1998.

[4] S. Stillman, R. Tanawongsuwan, and I. Essa. A system for tracking and recognizing multiple people with multiple cameras. In *Proc. Second Int'l Conf. Audio- and Video-based Biometric Person Authentication*, pages 96–101, 1999.

[5] Ross Cutler and Larry Davis. Developing real-time computer vision applications for Intel Pentium III based Windows NT workstations. In *ICCV FRAME-RATE Workshop: Frame-rate Applications, Methods and Experiences with Regularly Available Technology and Equipment*, 1999.

[6] Ben Delaney. On the trail of the shadow woman: The mystery of motion capture. *IEEE Computer Graphics and Application*, 18(5):14–19, 1998.

[7] D.J. Sturman. Computer puppetry. *IEEE Computer Graphics and Application*, 18(1):38–45, 1998.

[8] T. Horprasert, I. Haritaoglu, C. Wren, D. Harwood, L.S. Davis, and A. Pentland. Real-time 3d motion capture. In *Proc. 1998 Workshop on Perceptual User Interface (PUI'98)*, San Francisco, 1998.

[9] J. Ohya and et al. Virtual metamorphosis. *IEEE Multimedia*, 6(2):29–39, 1999.

[10] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Who? when? where? what? a real-time system for detecting and tracking people. In *Proc. the thrid IEEE Int'l Conf. Automatic Face and Gesture Recognition (Nara, Japan)*, pages 222–227. IEEE Computer Society Press, Los Alamitos, Calif., 1998.

[11] C.R. Wren and A. Pentland. Dynamic modeling of human motion. In *Proc. the thrid IEEE Int'l Conf. Automatic Face and Gesture Recognition (Nara, Japan)*, pages 22–27. IEEE Computer Society Press, Los Alamitos, Calif., 1998.

[12] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1998.

[13] A. Utsumi, H. Mori, J. Ohya, and M. Yachida. Multiple-human tracking using multiple cameras. In *Proc. the thrid IEEE Int'l Conf. Automatic Face and Gesture Recognition (Nara, Japan)*. IEEE Computer Society Press, Los Alamitos, Calif., 1998.

[14] M. Yamada, K. Ebihara, and J. Ohya. A new robust real-time method for extracting human silhouettes from color images. In *Proc. the thrid IEEE Int'l Conf. Automatic Face and Gesture Recognition (Nara, Japan)*, pages 528–533. IEEE Computer Society Press, Los Alamitos, Calif., 1998.

[15] P.L. Rosin and T. Ellis. Image difference threshold strategies and shadow detection. In *Proc. the sixth British Machine Vision Conference*, 1994.

[16] Olivier Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint.* The MIT Press, Cambridge, Massachusetts, 1993.

[17] Dimitrios V. Papadimitriou. *Shape and Motion Analysis from Stereo for Model-Based Image Coding.* PhD thesis, Department of Electronic Systems Engineering, University of Essex, united Kingdom, May 1995.

[18] R. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc. the Computer Vision and Pattern Recognition*, 1986.

[19] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE ICCV'99 FRAME-RATE Workshop*, 1999.

[20] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, July 1993.

[21] D. Laur, P. Hanrahan, and Hierarchical Splatting. A progressive refinement algorithm for volume rendering. In *SIGGRAPH Proceedings*, 1991.

[22] B. Stander and J. Hart. A lipschitz method for accelerated volume rendering. In *Proceedings of the 1994 Symposium on Volume Visualization*, 1994.

[23] A. Prock and C. Dyer. Towards real-time voxel coloring. In *Proceedings of Image Understanding Workshop*, 1998.