TOWARDS THE ULTIMATE MOTION CAPTURE TECHNOLOGY

Bradley Stuart, Patrick Baker and Yiannis Aloimonos
Computer Vision Laboratory
Center for Automation Research
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
{brad,pbaker,yiannis}@cfar.umd.edu

Abstract Recent developments in camera and computer technology have made multiple-camera systems less expensive and more usable. Using such systems, we can generate 3-D models of human activity for use in surveillance, as avatars, or for 3-D effects generation. Some approaches to model generation are voxel coloring, space carving, silhouette intersection, and the combination of multiple stereo reconstructions.

Our attempt to overcome various shortcomings of the above approaches has led to the use of image derivatives and motion to determine the shape and motion of the activity in view. Direct computations of the gradient directions and the image motion normal to the gradient provide the information to generate a 3D + motion model consistent with all the image data. Data structures encode visibility information from each of the cameras surrounding the scene, allowing efficient determination of the subsets of measurements to be combined in a modified space-carving system.

The main contributions of this paper are the following: the development of a system for combining multiple image gradient measurements to determine the 3-D iso-brightness direction and its consistency, a system for combining multiple normal flow measurements to determine the motion normal to the iso-brightness direction, and a data structure based on the rays passing through the centers of projection and the image pixels, forming an unbounded projective grid through the space of the scene and allowing efficient determination and updating of scene point visibility.

Reconstructions of human motion using twenty cameras are presented. The resulting 3D dynamic models of human action can be used: (a) directly as avatars performing specific activities, (b) for creating large libraries of human action models that can be used for animation, and (c) for further statistical/geometric processing that will yield sophisticated models of generic action.

N. Magnenat-Thalmann et al. (eds.), *Deformable Avatars*

1. PREVIOUS WORK

The question of how to generate new views of a three-dimensional (3-D) scene or object has been addressed in a number of different ways. The methods used fall into three rough categories, depending on whether the method uses only two-dimensional (2-D) data, builds a 2-D depth map ($2\frac{1}{2}$-D sketch), or generates new views from a full 3-D model of the scene.

1.1. VOXEL CARVING

Much recent work has concerned the integration of a large number of views into a single 3-D object model. In the technique called space carving or voxel coloring, space is broken up into small cells (volume elements, or voxels) and the images of these cells are checked for consistency. Cells with inconsistent images are removed from the set of cells which constitute the object [2] [7] [10].

The issue of cell visibility can be addressed in a variety of ways. In [10], the camera configuration is controlled so that voxels are visited in near-to-far order. The space carving technique [7] allows arbitrary camera placement, but cameras are not used until they are passed by the plane sweeping through the scene. Generalized voxel coloring [2] maintains a "layered depth image" to store visibility information, and efficiently determine which voxels are revealed by the carving operation.

Different criteria for voxel consistency have also been used. The technique of volume intersection (shape from silhouettes) uses the logical AND of bit mask images to define consistent voxels. Seitz develops a color consistency test which is based on hue.

1.2. MORPHING

Other approaches to the problem of generating new views from a given set of views are image interpolation and morphing. In image interpolation, interposed frames are created by smoothly varying pixel values between two or more source frames. Interpolation gives unsatisfactory results unless the source frames are very close together.

Morphing improves on this technique by imposing a mesh on the source frames, and moving control points on this mesh smoothly between the frames. Intermediate frames are generated by texture mapping each mesh cell as it moves, and varying the cell texture maps smoothly between the source frames. Defining the mesh requires the establishment of correspondences between points in the source images. Both image interpolation and morphing are strictly 2-D techniques; they do not perform well in the presence of occlusions and 3-D structure in the scene.

1.3. STEREO TECHNIQUES

Stereo techniques use two or more images to compute 3-D structure in the scene. The epipolar constraint is used along with some technique (such as correlation) to find corresponding points on epipolar lines. Additional constraints are added to speed searching or deal with ambiguities. Examples of these are ordering constraints, inter-line constraints and depth smoothness constraints [13]. Once correspondences has been determined, the distance to a point in the scene is then an inverse function of the disparity in the image coordinates. Stereo techniques work well only with a limited range of camera separations. If the cameras are too close, images are too similar, disparities are small, and accuracy suffers. If the cameras are too widely separated, correspondences become difficult to find and the additional constraints employed to find them start to be violated more and more. Narayanan et al. [8] demonstrate techniques combining multiple stereo pairs and filling in the holes that appear behind a single depth map. In [12], optical flow values are back-projected onto these models and 3-D flow values are inferred. Alternately, the optical flow values can be used to augment incomplete 3-D models to make them more accurate.

1.4. STRUCTURE FROM MOTION

Structure-from-motion techniques use the relative motion between camera and scene to determine the depth of points in the scene [11]. The technique is based on motion parallax; when the camera translates relative to the scene, points near the camera have a greater apparent motion than points far away. This translational flow is compounded with the apparent motion due to rotation of the camera. This rotation gives no depth information, but its confusion with the translation makes the structure-from-motion problem considerably more difficult. Structure-from-motion techniques can be based on either optical flow or normal flow. Either way, the technique assumes that flows are due only to the motion of the camera; independent motion in the scene is not allowed. Furthermore, inaccuracies in the determination of camera motion lead directly to inaccuracies in the resulting depth map.

2. CAMERA SETUP AND IMAGE FORMATION

Up to sixteen cameras are installed on each of four walls, for a total of sixty-four cameras. Sixteen of these cameras are single-ccd RGB filtered color cameras; the rest are standard gray-scale cameras. All cameras send 8 bits per pixel of digital data at 640x480 pixels and 60 frames per second. Frame rates up to 85 fps are possible with a restricted region of interest. Data is fed from

each camera to a dedicated video capture card; four such cards are installed in each of 16 Pentium II PC's. Each PC has one gigabyte of RAM to allow real-time capture of 3000 uncompressed frames per computer [3]. Projection matrices are computed by nonlinear optimization on up to 25 points of a large calibration object, or more accurately by the method detailed in [1].

3. RAY CARVING

The process of voxel carving typically begins with the division of space into cubes of some fixed size. This introduces restrictions on the shape and size of the space that can be carved, as well as restrictions on the resolution of the cameras that can be usefully employed.

3.1. VOXELS OR RAYS

Instead of splitting the scene arbitrarily with a regular grid, we base our division of space on the rays which pass through the pixels in each camera. Each camera defines a pyramid-shaped bundle of rays passing through the scene. Saito and Kanade [9] define a grid based on the rays from two selected cameras. Here, we do not define any special cameras, using the rays from all cameras equally. Furthermore, we use a continuous (floating point) rather than discrete representation of the intervals on the ray which are solid or transparent. Each filled interval is represented by a pair of numbers, the distances to the endpoints of the interval. These distances are adjusted as portions of the scene are carved away.

The carving method of Seitz [10] restricts the camera positions to be "occlusion compatible." This means that it must be possible to separate the scene from the cameras by a flat or concave surface, and sweep the surface out, processing voxels from near to far. Cameras which violate this ordering can be excluded until they are passed by the sweep surface. One of any pair of cameras which can "see" each other must be excluded until the sweep surface passes the line joining their centers. The asymmetries induced by excluding one camera can be alleviated by sweeping in multiple directions.

One chief advantage of the ray representation is that it provides a compact description of visibility. A point on a ray is visible if its distance is less than or equal to the distance to the beginning of the ray's first filled interval. Rays can be processed in arbitrary or random order; it is not necessary to sweep a surface through the scene.

3.2. COMPUTING DISTANCES ALONG RAYS

Carving efficiently using ray information requires the ability to convert image positions along the ray into 3-D distances and positions. All projective transformations of a line can be defined by specifying the transformations of only three points on that line. Using this fact, we can specify the transformations back and forth among 3-D coordinates along a ray, 2-D image coordinates in any other camera's view of that ray, and 1-D line coordinates of distance along the ray.

The visibility information for the ray is supplemented with the information needed to render the ray into all of the other views. This is analogous to the rendering of epipolar lines needed to determine stereo correspondences. The projective mapping between the ray in 3-D and its images in all the other views is completely determined by defining the mapping for three distinct points on the ray. The camera center is one such point, selected because it is shared by all rays in one image. For a 3×4 projection matrix $\tilde{\mathbf{P}} = [\mathbf{P}\tilde{\mathbf{p}}]$, where $\mathbf{P}$ is a 3×3 matrix and $\tilde{\mathbf{p}}$ is a 3×1 vector, Faugeras [4] gives its position as

$$\mathbf{C} = \mathbf{P}^{-1}\tilde{\mathbf{p}}. \tag{1}$$

The second point selected is the vanishing point for the given ray — the vector parallel to the ray. Again from Faugeras, for homogeneous pixel coordinate $\mathbf{x}$ this direction is the homogeneous vector $\left[\mathbf{D}^{\top}0\right]^{\top}$, where

$$\mathbf{D} = \mathbf{P}^{-1}\mathbf{x}. \tag{2}$$

The third point $\mathbf{U}$ is taken at a unit distance from the camera center, in the direction of the vanishing point. In homogeneous coordinates,

$$\mathbf{U} = \begin{bmatrix} \mathbf{C} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix}, \tag{3}$$

where $\mathbf{D}$ has been scaled so that $||\mathbf{D}||_2 = 1$.

By projecting these three points into each of the other images, we are able to convert positions along the image of the ray directly into 3-D distances along the ray itself. Specifically, we define projectivities between the homogeneous image coordinates in the i^{th} camera $\mathbf{c} = \tilde{\mathbf{P}}_i\mathbf{C}$, $\mathbf{d} = \tilde{\mathbf{P}}_i\mathbf{D}$, and $\mathbf{u} = \tilde{\mathbf{P}}_i\mathbf{U}$, and the homogeneous line coordinates $[0,1]^{\top}$, $[1,0]^{\top}$ and $[1,1]^{\top}$, respectively. This projectivity will also map a point $\mathbf{x} = \tilde{\mathbf{P}}_i\mathbf{X}$ lying on the line to the line coordinate $[x_1, x_2]$, where $\frac{x_1}{x_2}$ is the distance from the camera center $\mathbf{C}$ defining this ray to $\mathbf{X}$. This projectivity is defined by the matrix product

$$\mathbf{L} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} \mathbf{c} \times \mathbf{n} \\ \mathbf{n} \times \mathbf{d} \end{bmatrix}. \tag{4}$$

The second matrix in this definition ensures that the points **c** and **d** map correctly, while the first defines the scale so that **u** maps to the point $[1, 1]$. The vector **n** in the second matrix is an arbitrary vector defining the null-space of the matrix L. For points$\mathbf{x} = \tilde{\mathbf{P}}_i\mathbf{X}$ on the ray, $\mathbf{Lx}$ gives the homogeneous coordinate measuring the 3-D distance from the camera center **C** to the point **X**.

Requiring that the point **u** maps to the line coordinate $[1, 1]$ gives us

$$\begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} \mathbf{c} \times \mathbf{n} \\ \mathbf{n} \times \mathbf{d} \end{bmatrix} \mathbf{u} \equiv \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \text{or} \qquad s_1[\mathbf{cnu}] = s_2[\mathbf{ndu}],$$

which is satisfied by setting $s_1 = [\mathbf{ndu}]$ and $s_2 = [\mathbf{cnu}]$.

Applying this projectivity to points which do not lie strictly on the line introduces errors when taking the resulting line coordinates as the depth values at the given point. The null-space of **L** defines the sets of points in the plane which project to the same line coordinates. For points not on the ray, we take their coordinate to be the same as the coordinate of the perpendicular projection onto the ray. To achieve this, we define the null-space vector to be the point at infinity in the direction perpendicular to the line containing **c** and **d**. This is given by taking the cross product $\mathbf{c} \times \mathbf{d}$, and setting its third coordinate to zero. Thus

$$\mathbf{n} = [c_2d_3 - c_3d_2, -c_1d_3 + c_3d_1, 0]^\top .$$

Once the distance coordinate along the line is known, it is a simple matter to convert this to a 3-D coordinate. For any distance coordinate λ,

$$\mathbf{X}_\lambda = \begin{bmatrix} \mathbf{C} \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix}, \tag{5}$$

where again, **D** has been scaled to unity.

3.3. CARVING ALONG A RAY

Ray carving requires projecting a given ray into all the other views. It is more computationally efficient to continue carving down the length of the ray until the visible end of the ray is no longer inconsistent. While updating the visible end of the current ray, moving its front point away from the camera, rays from the other cameras which view this front point are also updated. The carving algorithm proceeds as follows:

1 Determine the 3-D point x which is visible on this ray R as in (5).

2 Project the principal points (**C**,**D**,**U**) of the ray R into the other views i, giving image coordinates $(r, c)_i$ and defining the matrices $\mathbf{L}(R)_i$ as in (4). Recall that **C** is common to all rays from a particular view; its projected coordinate (the epipole) can be saved. Saving other values for the images of the rays is memory intensive.

3 Image coordinates $(r, c)_i$ are used to access predefined rays S_i, the rays which view x in the other cameras.

4 Determine the depth boundaries at which the ray R crosses the edges of the pixels $(r, c)_i$ in each of the views. One of the views, i_{lub} defines the least upper bound on this depth.

5 If the pixels $(r, c)_i$ are inconsistent with the pixel defining the ray R (see below), the portion of the current ray between the visible point and the least upper bound defined above must be removed. The removed region extends onto the second least (unique) depth value found in the previous step.

6 In addition, the ray $S_{i_{lub}}$ defined in the view i_{lub} is also carved, removing from it the region defined by $S_{i_{lub}}$'s intersection with the pixel defining ray R.

7 The position x on ray R is updated, along with the definition of $S_{i_{lub}}$, the ray containing the portion which was carved away. Other rays S_i do not need to be recomputed, or carved, as the point x has moved to a new pixel only in view i_{lub}.

Since visibility information is encoded and updated for each ray, they can be visited in arbitrary order. Our choice is to visit all the rays defined by one image before moving on to the next. Cycling through the images continues until the scene converges and new points are no longer carved away.

4. INTENSITY GRADIENTS AND EDGES IN THREE DIMENSIONS

Typically, voxel carving algorithms have used color consistency as a test function in determining what portions of the scene need to be removed from the object model. Color has several advantages that a simple gray level does not. Color is less sensitive to variation in viewpoint, camera intensity response, and differences in lighting.

Gray-scale images are not without their own properties that can perform equally well. Intensity gradients can perform better than simple intensities, as they are less sensitive to the camera intensity response and differences in lighting. However, intensity gradients are not independent of viewpoint — a simple rotation of the camera in place will induce an opposite rotation of the gradient vectors. Furthermore, gradients which appear at a 3-D point in one image may not be present at all in another image, as at the occluding boundary of a smooth object. As we will see, these viewpoint dependent characteristics can be overcome by computing the 3-D iso-intensity contour.

Intensity gradients arise in images for several reasons. A surface in the scene may reflect or emit different intensities due to texture intrinsic to the surface or due to lighting differences such as shadows. An occluding boundary (discontinuity) in the scene produces a gradient when the occluding and occluded objects are of different colors. Specular reflections and other departures from a Lambertian reflectance model also produce intensity gradients. We do not deal with specularities here, but they could be addressed by including a lighting model of the scene, for example.

The gradient vector $dI/d\mathbf{x} = [I_x, I_y]^\top$ is the intensity change per unit length (usually a single pixel width) in the direction of steepest ascent. The normal to this vector is the direction in which the image intensity remains constant; the iso-intensity contour. Projecting the image intensity values back along their rays into three dimensions, we have a space-filling gradient field, without regard to where the physical surfaces lie. In Euclidean 3-D coordinates $\mathbf{X}$, this field has the brightness gradient

$$\frac{dB}{d\mathbf{X}} = k \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_2}{\partial X_1} \\ \frac{\partial x_1}{\partial X_2} & \frac{\partial x_2}{\partial X_2} \\ \frac{\partial x_1}{\partial X_3} & \frac{\partial x_2}{\partial X_3} \end{bmatrix} \frac{dI}{d\mathbf{x}}, \tag{6}$$

where $k = dB/dI$, i.e. we assume a linear relationship between image intensity and scene brightness.

Using the 3×4 projection matrix $\mathbf{P}$ and homogeneous coordinates $\hat{\mathbf{X}}^\top = \begin{bmatrix} \mathbf{X}^\top & 1 \end{bmatrix}$, we define $\hat{\mathbf{x}} = \mathbf{P}\hat{\mathbf{X}}$. The image of $\mathbf{X}$ is then at the 2-D coordinates

$$x_1 = \frac{\hat{x}_1}{\hat{x}_3}, x_2 = \frac{\hat{x}_2}{\hat{x}_3}. \tag{7}$$

The derivatives of (6) are then given by

$$\begin{aligned} \frac{\partial x_i}{\partial X_j} &= \frac{\partial}{\partial X_j}\left(\frac{\hat{x}_i}{\hat{x}_3}\right) \\ &= \left(\frac{\partial \hat{x}_i}{\partial X_j}\hat{x}_3 - \frac{\partial \hat{x}_3}{\partial X_j}\hat{x}_i\right)/\hat{x}_3^2 \\ &= \left(P_{i,j}\hat{x}_3 - P_{3,j}\hat{x}_i\right)/\hat{x}_3^2. \end{aligned}$$

As in the 2-D case, the gradient vector is normal to the manifold of constant brightness. This is the iso-brightness plane, which must contain the direction of constant brightness lying on the actual surface in the scene.

Each of the views gives a measurement of the image gradient, and therefore a plane on which the iso-brightness direction must lie. With two such planes, a unique line is defined for the edge direction in 3-D. With more than two planes, we can use a fitting technique to find the best direction for the 3-D

edge. Furthermore, we can use the quality of this fit to determine whether these image gradients do in fact define a consistent edge in 3-D.

The fitting technique used is a principal component analysis. For N 3-D gradient vectors $\mathbf{g}_i = dB_i/d\mathbf{X}$, what is required is to find the vector $\mathbf{x}$ of unit length which minimizes the sum of projection lengths $\mathbf{x} \cdot \mathbf{g}_i$. This is done by finding the eigenvector for the least eigenvalue of the matrix

$$\mathbf{M} = \sum_{i=1}^{N} (\mathbf{g}_i \mathbf{g}_i^\top) \tag{8}$$

where $(\mathbf{g}_i \mathbf{g}_i^\top)$ is the 3×3 outer product of the vector $\mathbf{g}_i$ with itself.

If the third eigenvalue is not small relative to the first and second, there is no vector which is a satisfactory approximation to the normal of the inputs. In that case, the 3-D location selecting these image measurements must not be a part of a consistent object; we carve the location away.

At surface discontinuities, the image gradient is due to a boundary edge for the object. The 3-D gradient describes the tangent plane passing through the camera center and the limb of the object. The surface at the limb may be a sharp corner, or it the may be a smooth surface. In either case, the tangent plane containing the camera center and the edge will be consistent with any iso-intensity direction on the surface; the 3-D gradient is normal to *all* directions on the surface. If the edge is due to a sharp corner, the same edge may be visible in several views and the iso-intensity contour can be determined precisely. In the case of a smooth surface edge, the iso-intensity direction will have to be determined through texture edges on the surface seen from other viewpoints. A smooth surface with a smooth texture will not have a single iso-intensity direction, but at the limb of the object it can be limited to the tangent plane.

5. THE NORMAL MOTION FIELD IN THREE DIMENSIONS

5.1. NORMAL FLOW

The motion constraint equation is the mathematical formulation of the statement that the brightness of a point in the scene remains constant for small motions. Formally,

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{9}$$

This equation relates the component of the optical flow parallel to the image gradient to the time derivative of the intensity. Since this optical flow component is normal to the edges of the image, it is termed *normal flow*.

The tangential component of the optical flow cannot be determined directly without additional assumptions. These extra constraints introduce biases into

the computation of flow [5] [6]. Flow smoothness constraints may be marginally acceptable in applications where the camera is moving in a rigid environment. However, with a stationary camera and a non-rigid scene, we expect regions of zero flow to be near regions of quite high flow values. Refusing to compute flow near discontinuities will leave us without the most informative parts of the scene. Rather than accepting additional biases or loss of measurements in order to compute a quantity which is not well-defined in the image, we choose to work with the normal component of the flow.

5.2. THREE DIMENSIONS

Just as in the 2-D case, we start from the premise that motion along the iso-brightness contour cannot be measured locally. This is simply the restatement of the aperture problem in the 3-D framework. The case is not so bad in three dimensions, as we can still determine motion in two out of three principal axes.

The normal motion on an iso-brightness contour is by definition limited to the plane of normals to the contour at a given point. The component of the motion parallel to the contour cannot be measured directly, and doesn't concern us here. Each measurement of normal flow in the image set defines a line in the image, called the normal flow constraint line. Both the optical flow and the projection of the 3-D normal flow must lie along this constraint line. This line, together with the camera center, defines a constraint plane through scene space. The constraint planes from the various views intersect the plane of normals in a set of lines, all of which intersect in the single point defining the 3-D normal flow. If the lines fail to intersect, this indicates that the normal flow measurements in the images are inconsistent.

In practice, there are errors in normal flow measurements, and in gradient measurements. We need to take these errors into account when designing the algorithm which determines the consistency and value of the 3-D normal flow. When more than two normal flow measurements are available, the problem of finding the best intersection point is an optimization problem. The point which minimizes the sum of squared distances from the constraint planes (including the plane of edge normals) is found, while the error measurement determines whether the selected point is sufficiently consistent to use as the 3-D normal flow.

We find the constraint plane for a given image point $\mathbf{x}$, normal flow $[n_1, n_2]$, and the iso-intensity direction defined in the previous section. This plane is defined in homogeneous coordinates by the camera center $\mathbf{C}$, the direction vector $\mathbf{D} = \mathbf{P}^{-1}(x_1 + n_1, x_2 + n_2, 1)^\top$ — as in (2) — and the iso-intensity direction unit vector $\mathbf{W}$. Using homogeneous coordinates for the point $\mathbf{C}$, and representing the vectors $\mathbf{D}$ and $\mathbf{W}$ by the point at infinity in the vector direction, the

plane $\mathbf{p}$ containing these (homogeneous) points is given by the determinant

$$\mathbf{p} = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 \\ C_1 & C_2 & C_3 & 1 \\ D_1 & D_2 & D_3 & 0 \\ W_1 & W_2 & W_3 & 0 \end{vmatrix} \tag{10}$$

$$= [(\mathbf{D} \times \mathbf{W}), -[\mathbf{DWC}]]. \tag{11}$$

Likewise, the plane of normals to $\mathbf{W}$ through $\mathbf{X}$ combines the other two eigenvectors of the matrix $\mathbf{M}$ of (8), $\mathbf{U}$ and $\mathbf{V}$. Noting that $\mathbf{W} = \pm\mathbf{U} \times \mathbf{V}$, the homogeneous coordinate of this plane is

$$\mathbf{q} = [(\mathbf{U} \times \mathbf{V}), -[\mathbf{UVX}]] \tag{12}$$

$$\equiv [\mathbf{W}, -\mathbf{W} \cdot \mathbf{X}]. \tag{13}$$

To find the best candidate for the intersection of these planes, we need to weight the planes equally in Euclidean space. This involves scaling the homogeneous coordinates $\mathbf{p}$ by $1/\| [p_1, p_2, p_3]\|_2$. Scaled this way, the homogeneous coordinate can be viewed as a unit vector in the direction normal to the plane, and the negative of the distance from the origin to the plane along this vector. The point $\mathbf{X}'$ is then constrained to have $X'_4 = 1$ and is the least squares solution minimizing the error function

$$E = \sum_{i=1}^{i<N} (\mathbf{p}_i \cdot \mathbf{X}')^2. \tag{14}$$

This gives three constraint equations for $k \in \{1, 2, 3\}$ of the form

$$-\sum_{i=1}^{N} p_{i,k} p_{i,4} = \sum_{j=1}^{3} X'_j \sum_{i=1}^{N} p_{i,k} p_{i,j}. \tag{15}$$

The normal motion vector is then $\mathbf{N} = \mathbf{X}' - \mathbf{X}$. In places where the error measure of (14) is excessive, the normal flow values can be deemed inconsistent, and the 3-D point $\mathbf{X}$ can be carved away. Alternately, the reprojection of the vector $\mathbf{N}$ into each of the images can be checked for consistency with the normal flow measurement in that image. The projection of $\mathbf{X}'$ should lie on (near) the motion constraint line.

6. RESULTS

Here we present some results of the algorithm. Input data was obtained from sixteen cameras widely separated around the room, with a person walking through the scene. Cameras were calibrated using images of a known calibration object, and images were corrected for radial distortions. These reconstructions used images of 320 × 240 pixels. Four of the input images are shown in

Figure 1; Figure 1(a) and (b) are two two raw data images, (c) is an image of the background, and (d) is the foreground silhouette.

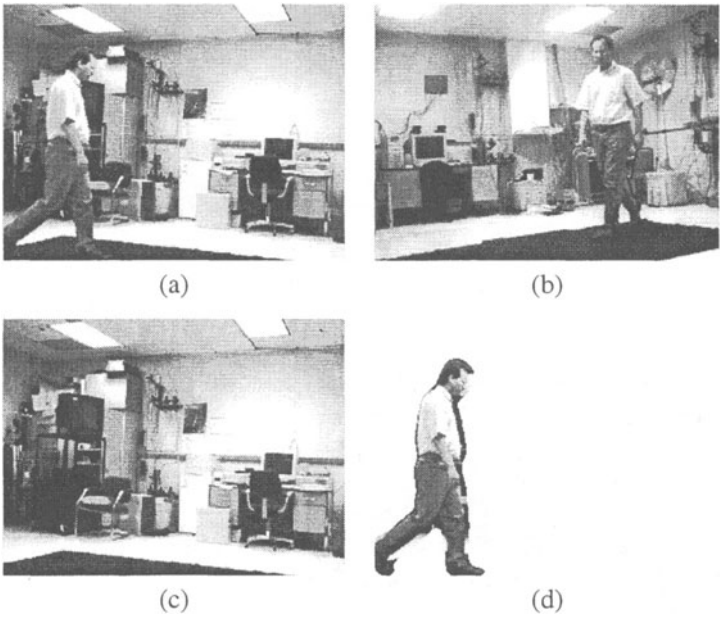

Figure 1 Input Data

In Figure 2, images (a) and (b) show two views of a moving person, after the data structure has been initialized with the silhouette intersection from twenty views. The foreground/background separation procedure purposely favors the foreground, as any missing foreground in a view will generate holes in the initial volume. This also adds a shell around the volume which needs to be carved away, along with any concavities in the moving human. Figures 2(c) and (d) show a depth map of the scene before and after the ray carving. Figures 2(e) and (f) show the agreement in the iso-intensity direction before and after the carving. Light colors represent maximum agreement, darker colors indicate inconsistency. Figures 2(g) through (l) show five views of the scene from viewpoints between the cameras. A virtual floor and shadow are added to the scene, as a simple example of the effects available using full 3-D structure.

7. DISCUSSION AND FUTURE WORK

This paper presents the general framework of ray carving, a method of generating 3-D models by which regions of space with inconsistent images are removed. Previous work in this field has relied on color information as a test of consistency. This paper presents two new criteria for this determination;

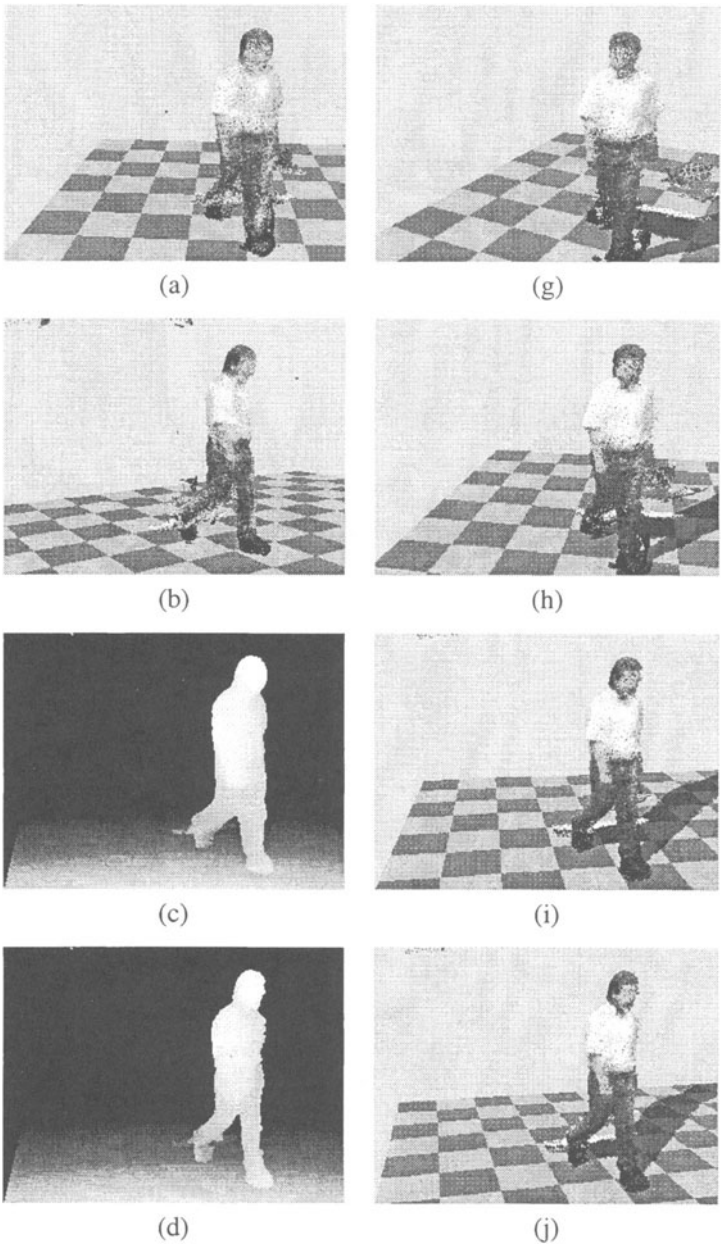

Figure 2 Results

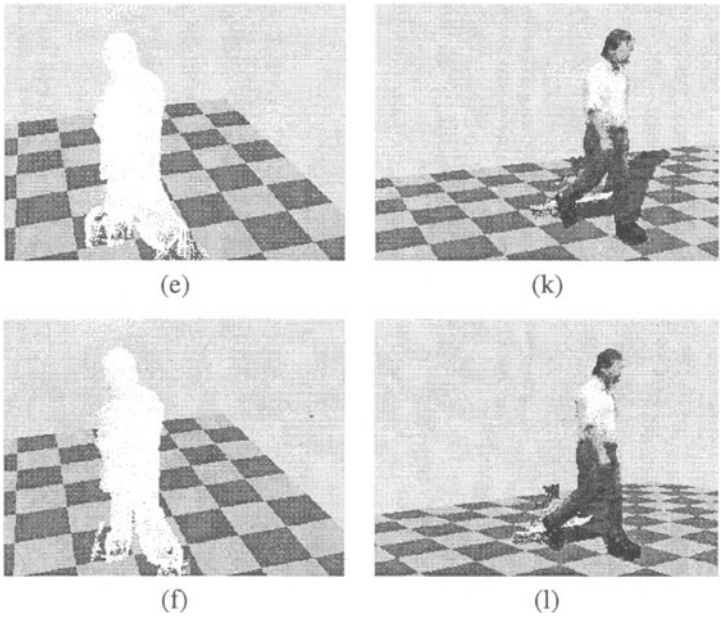

Figure 3 Results (cont.)

consistency of edge gradients in 3-D and normal flow in 3-D. Use of these techniques on sequences of moving people is shown.

This technique is not specific to human motion; it gives the shape and surface motion of general deformable objects. The next step in progressing toward the ultimate motion capture technology for human motion in particular is to use this moving object model to fit trajectories of points in a human model. Simply put, using the recovered models we can create a 3-D motion field sequence representing implicitly the specific animation. Our algorithm implicitly uses this representation during carving. This approach allows motion capture to occur without the need for markers, specialized backgrounds, or sensors on the body. It can capture the complex motions of clothing and nonrigid portions of the body. Ultimately, databases of human activity can be created and used in graphics and recognition systems.

References

[1] P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *Proc. IEEE Workshop on Omnidirectional Vision*, pages 134–141, Hilton Head Island, SC, 2000. IEEE Computer Society.

[2] W. Bruce Culbertson, Thomas Malzbender, and Greg Slabaugh. Generalized voxel coloring. In *Vision Algorithms: Theory and Practice*, Corfu, Greece, 1999. IEEE.

[3] L. Davis, E. Borovikov, R. Cutler, D. Harwood, and T. Horprasert. Multiperspective analysis of human action. In *Proc. of Third International Workshop on Cooperative Distributed Vision*, Kyoto, Japan, 1999.

[4] O. D. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, MA, 1992.

[5] C. Fermüller, R. Pless, and Y. Aloimonos. Statistical biases in optic flow. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 561–566, 1999.

[6] C. Fermüller, R. Pless, and Y. Aloimonos. The Ouchi illusion as an artifact of biased flow estimation. *Vision Research*, 40:77–96, 2000.

[7] K. N. Kutulakos and S. M. Seitz. What do N photographs tell us about 3D shape? Computer Science Technical Report 692, University of Rochester, 1998.

[8] P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. International Conference on Computer Vision*, pages 3–10, Bombay, 1998.

[9] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 49–54, 1999.

[10] S. M. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–1073, 1997.

[11] M. E. Spetsakis and J. Aloimonos. A unified theory of structure from motion. In *Proc. DARPA Image Understanding Workshop*, pages 271–283, 1990.

[12] Sundar Vedula, Simon Baker, Peter Rander, Robert collins, and Takeo Kanade. Three-dimensional scene flow. In *Proc. International Conference on Computer Vision*, Corfu, Greece, 1999.

[13] Z. Zhang, O. D. Faugeras, and N. Ayache. Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints. In *Proc. Second International Conference on Computer Vision*, pages 177–186, 1988.