Three-Dimensional Scene Flow

Sundar Vedula[†], Simon Baker[†], Peter Rander[†‡], Robert Collins[†], and Takeo Kanade[†]

[†]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
[‡]Zaxel Systems Inc., Ten 40th Street, Pittsburgh, PA 15201

Abstract

Scene flow is the three-dimensional motion field of points in the world, just as optical flow is the two-dimensional motion field of points in an image. Any optical flow is simply the projection of the scene flow onto the image plane of a camera. In this paper, we present a framework for the computation of dense, non-rigid scene flow from optical flow. Our approach leads to straightforward linear algorithms and a classification of the task into three major scenarios: (1) complete instantaneous knowledge of the scene structure, (2) knowledge only of correspondence information, and (3) no knowledge of the scene structure. We also show that multiple estimates of the normal flow cannot be used to estimate dense scene flow directly without some form of smoothing or regularization.

1 Introduction

Optical flow is a two-dimensional motion field in the image plane. It is the projection of the three-dimensional motion of the world. If the world is completely non-rigid, the motions of the points in the scene may all be independent of each other. One representation of the scene motion is therefore a dense three-dimensional vector field defined for every point on every surface in the scene. By analogy with optical flow, we refer to this three-dimensional motion field as *scene flow*.

In this paper, we present a framework for the computation of *dense, non-rigid* scene flow directly from optical flow. Our approach leads to efficient linear algorithms and a classification of the task into three major scenarios:

1. Complete instantaneous knowledge of the structure of the scene, including surface normals and rates of change of depth maps. In this case, only one optical flow is required to compute the scene flow.

2. Knowledge only of stereo correspondences. In this case, at least two optical flows are needed to compute the scene flow, but more improve robustness.

3. No knowledge of the surface. In this case, several optical flows can be used in a reconstruction algorithm to estimate the scene structure (and then scene flow).

For each scenario, we propose an algorithm and demonstrate it on a collection of video sequences of a dynamic, non-rigid scene. We also show that multiple estimates of the normal flow cannot be used to estimate scene flow directly, without some form of regularization or smoothing.

One possible application of scene flow is as a predictor for efficient and robust stereo. Given a reconstructed model of the scene at a certain time, one would like to obtain an estimate of the structure at the next time step using minimal computation. This would allow: (1) more efficient computation of the structure at the next time step because a first estimate would be available to reduce the search space, and (2) more robust computation of the structure because the predicted structure can be integrated with the new stereo data. Other applications of scene flow include various dynamic rendering and interpretation tasks, from the generation of slow-motion replays, to the understanding and modeling of human actions.

1.1 Related Work

Computing the three-dimensional motion of a scene is a fundamental task in computer vision that has been approached in a wide variety of ways. If the scene is rigid and the cameras are calibrated, the three-dimensional scene structure and relative motion can be computed (up to a scale factor) from a single monocular video sequence using *structure-from-motion* [Ullman, 1979]. If the scene is only piecewise rigid, extensions to structure-from-motion algorithms can be used. See, for example, [Zhang and Faugeras, 1992a] and [Costeira and Kanade, 1998].

Although restricted forms of non-rigidity can be analyzed using the structure-from-motion paradigm [Avidan and Shashua, 1998], general non-rigid motion cannot be estimated from a single camera without additional assumptions about the scene. However, given strong enough *a priori* assumptions about the scene, for example in the form of a deformable model [Pentland and Horowitz, 1991] [Metaxas and Terzopoulos, 1993] or the assumption that the motion minimizes the deviation from a rigid body motion [Ullman, 1984], recovery of three-dimensional non-rigid motion from a monocular view is possible. See [Penna, 1994] for a recent survey of monocular non-rigid motion estimation, and the assumptions used to compute it.

Another common approach to recovering three-dimensional motion is to use multiple cameras and combine stereo and motion in an approach known as *motion-stereo*. Nearly all motion-stereo algorithms assume that the scene is rigid. See, for example, [Waxman and Duncan, 1986], [Young and Chellappa, 1999], and [Zhang and Faugeras, 1992b]. A paper which explicitly combines two optical flow fields is that of [Shi *et al.*, 1994]. In this paper, both the analysis and implementation are only applicable to certain simple motions of the camera (i.e. translations).

A few motion-stereo papers do consider non-rigid motion, including [Liao *et al.*, 1997] and [Malassiotis and Strintzis, 1997]. The former uses a relaxation-based algorithm to co-operatively match features in both the temporal and spatial domains. It therefore does not provide dense motion. The latter uses a grid which acts as a deformable model in a generalization of the monocular approaches mentioned above. Besides requiring *a priori* models of the scene, most deformable-model based approaches to motion-stereo would be too inefficient for our stereo-prediction application.

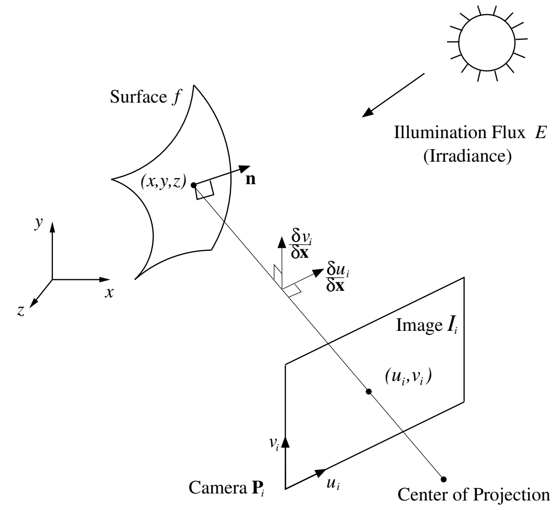

Figure 1: A non-rigid surface $f(x,y,z;t) = 0$ is moving with respect to a fixed world coordinate system (x,y,z). The normal to the surface is $\mathbf{n} = \mathbf{n}(x,y,z;t)$. The surface is assumed to be Lambertian with albedo $\rho = \rho(x,y,z;t)$ and the illumination flux (irradiance) is E. The i^{th} camera is fixed in space, has a co-ordinate frame (u_i,v_i), is represented by the 3×4 camera matrix $\mathbf{P}_i$, and captures the image sequence $I_i = I_i(u_i,v_i;t)$.

2 Image Formation Preliminaries

Consider a non-rigidly moving surface $f(x,y,z;t) = 0$ imaged by a fixed camera i, with 3×4 projection matrix $\mathbf{P}_i$, as illustrated in Figure 1. There are two aspects to the formation of the image sequence $I_i = I_i(u_i,v_i;t)$ captured by camera i: (1) the relative camera and surface geometry, and (2) the illumination and surface photometrics.

2.1 Relative Camera and Surface Geometry

The relationship between a point (x,y,z) on the surface and its image coordinates (u_i,v_i) in camera i is given by:

$$u_i = \frac{[\mathbf{P}_i]_1\,(x,y,z,1)^{\mathrm{T}}}{[\mathbf{P}_i]_3\,(x,y,z,1)^{\mathrm{T}}} \quad (1)$$

$$v_i = \frac{[\mathbf{P}_i]_2\,(x,y,z,1)^{\mathrm{T}}}{[\mathbf{P}_i]_3\,(x,y,z,1)^{\mathrm{T}}} \quad (2)$$

where $[\mathbf{P}_i]_j$ is the j^{th} row of $\mathbf{P}_i$. Equations (1) and (2) describe the mapping from a point $\mathbf{x} = (x,y,z)$ on the surface to its image $\mathbf{u}_i = (u_i,v_i)$ in camera i. Without knowledge of the surface, these equations are not invertible. Given f, they can be inverted, but the inversion requires intersecting a ray in space with the surface f.

The differential relationships between $\mathbf{x}$ and $\mathbf{u}_i$ can be represented by a 2×3 Jacobian matrix $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$. The 3 columns of the Jacobian matrix store the differential change in projected image co-ordinates per unit change in x, y, and z. A closed-form expression for $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$ as a function of $\mathbf{x}$ can be derived by differentiating Equations (1) and (2) symbolically. The Jacobian $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$ describes the relationship between a small change in the point on the surface and its image in camera i via $\Delta \mathbf{u}_i \approx \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \Delta \mathbf{x}$. Similarly, the inverse Jacobian $\frac{\partial \mathbf{x}}{\partial \mathbf{u}_i}$ describes the relationship between a small change in a point in the image of camera i and the point it is imaging in the scene via $\Delta \mathbf{x} \approx \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i} \Delta \mathbf{u}_i$.

Since image co-ordinates do not map uniquely to scene co-ordinates, the inverse Jacobian cannot be computed without knowledge of the surface. If we know the surface (and its gradient), the inverse Jacobian can be estimated as the solution of the following two sets of linear equations:

$$\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (3)$$

$$\frac{\partial f}{\partial \mathbf{u}_i} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i} = \boldsymbol{\nabla} f \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i} = (\,0\;\;0\,)\,. \quad (4)$$

Equation (3) expresses the constraint that a small change in $\mathbf{u}_i$ must lead to a small change in $\mathbf{x}$ which when projected back into the image gives the original change in $\mathbf{u}_i$. Equation (4) expresses the constraint that a small change in $\mathbf{u}_i$ does not lead to a change in f since the corresponding point in the world should still lie on the surface.

The 6 linear equations in Equations (3) and (4) can be decoupled into 3 for $\frac{\partial \mathbf{x}}{\partial u_i}$ and 3 for $\frac{\partial \mathbf{x}}{\partial v_i}$. Unique solutions exist for both $\frac{\partial \mathbf{x}}{\partial u_i}$ and $\frac{\partial \mathbf{x}}{\partial v_i}$ if and only if:

$$\left(\frac{\partial u_i}{\partial \mathbf{x}} \times \frac{\partial v_i}{\partial \mathbf{x}} \right) \cdot \boldsymbol{\nabla} f \neq 0. \quad (5)$$

Since $\boldsymbol{\nabla} f$ is parallel to the surface normal $\mathbf{n}$, the equations are degenerate if and only if the ray joining the camera center of projection and $\mathbf{x}$ is tangent to the surface.

2.2 Illumination and Surface Photometrics

At a point $\mathbf{x}$ in the scene, the irradiance or illumination flux measured in the direction $\mathbf{m}$ at time t can be represented by $E = E(\mathbf{m}; \mathbf{x}; t)$ [Horn, 1986]. This 6D irradiance function E is what is described as the *plenoptic function* in [Adelson and Bergen, 1991].

We denote the net directional irradiance of light at the point (x, y, z) on the surface at time t by $\mathbf{s} = \mathbf{s}(x, y, z; t)$. The net directional irradiance $\mathbf{s}$ is a vector quantity and is given by the (vector) surface integral of the irradiance E over the visible hemisphere of possible directions:

$$\mathbf{s}(x, y, z; t) \;=\; \int_{S(\mathbf{n})} E(\mathbf{m}; x, y, z; t)\, \mathrm{d}\mathbf{m} \qquad (6)$$

where $S(\mathbf{n}) = \{\mathbf{m} : \|\mathbf{m}\| = 1 \text{ and } \mathbf{m}\cdot\mathbf{n} \leq 0\}$ is the hemisphere of directions from which light can fall on a surface patch with surface normal $\mathbf{n}$.

We assume that the surface is Lambertian with albedo $\rho = \rho(\mathbf{x}; t)$. Then, assuming that the point $\mathbf{x} = (x, y, z)$ is visible in the i^{th} camera, and that the intensity registered in image I_i is proportional to the radiance of the point that it is the image of (i.e. image irradiance is proportional to scene radiance [Horn, 1986]), we have:

$$I_i(\mathbf{u}_i; t) \;=\; -C \cdot \rho(\mathbf{x}; t) \left[\mathbf{n}(\mathbf{x}; t) \cdot \mathbf{s}(\mathbf{x}; t)\right] \qquad (7)$$

where C is a constant that only depends upon the diameter of the lens and the distance between the lens and the image plane. The image pixel $\mathbf{u}_i = (u_i, v_i)$ and the surface point $\mathbf{x} = (x, y, z)$ are related by Equations (1) and (2).

3 Two-Dimensional Optical Flow

Suppose $\mathbf{x}(t)$ is the 3D path of a point on the surface and the image of this point in camera i is $\mathbf{u}_i(t)$. The 3D motion of this point is $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ and the 2D image motion of its projection is $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$. The 2D flow field $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ is usually known as optical flow. As the point $\mathbf{x}(t)$ moves on the surface, it is natural to assume that its albedo $\rho = \rho(\mathbf{x}(t); t)$ remains constant; i.e. we assume that

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} \;=\; 0. \qquad (8)$$

(For a deformably moving surface, it is only the surface properties like albedo that distinguish points anyway). The basis for optical flow algorithms is then the equation:

$$\frac{\mathrm{d}I_i}{\mathrm{d}t} \;=\; \boldsymbol{\nabla} I_i \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial I_i}{\partial t} \;=\; -C \cdot \rho(\mathbf{x}; t)\, \frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{s}] \qquad (9)$$

where $\boldsymbol{\nabla} I_i$ is the spatial gradient of the image, $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ is the optical flow, and $\frac{\partial I_i}{\partial t}$ is the instantaneous rate of change of the image intensity $I_i = I_i(\mathbf{u}_i; t)$.

The term $\mathbf{n} \cdot \mathbf{s}$ depends upon both the shape of the surface ($\mathbf{n}$) and the illumination ($\mathbf{s}$). To avoid explicit dependence upon the structure of the three-dimensional scene, it is often assumed that:

$$\mathbf{n} \cdot \mathbf{s} \;=\; \int_{S(\mathbf{n})} E(\mathbf{m}; \mathbf{x}; t)\, \mathbf{n} \cdot \mathrm{d}\mathbf{m} \qquad (10)$$

is constant ($\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{s}] = 0$). With uniform illumination or a surface normal that does not change rapidly, this assumption holds well (at least for Lambertian surfaces).

In either scenario $\frac{\mathrm{d}I_i}{\mathrm{d}t}$ goes to zero, and we arrive at the *Normal Flow* or *Gradient Constraint* Equation, used by "differential" optical flow algorithms [Barron *et al.*, 1994]:

$$\boldsymbol{\nabla} I_i \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial I_i}{\partial t} \;=\; 0. \qquad (11)$$

Using this constraint, a large number of algorithms have been proposed for estimating the optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$. See [Barron *et al.*, 1994] for a recent survey.

4 Three-Dimensional Scene Flow

In the same way that optical flow describes an instantaneous motion field in an image, we can think of scene flow as a three-dimensional flow field $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ describing the motion at every point in the scene. The analysis in Section 2.1 was only for a fixed time t. Now suppose there is a point $\mathbf{x} = \mathbf{x}(t)$ moving in the scene. The image of this point in camera i is $\mathbf{u}_i = \mathbf{u}_i(t)$. If the camera is not moving, the rate of change of $\mathbf{u}_i$ is uniquely determined as:

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \;=\; \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}. \qquad (12)$$

Inverting this relationship is, again, impossible without knowledge of the surface f. To invert it, note that $\mathbf{x}$ depends not only on $\mathbf{u}_i$, but also on the time, indirectly through the surface $f = f(\mathbf{x}; t)$. That is $\mathbf{x} = \mathbf{x}(\mathbf{u}_i(t); t)$. Differentiating this expression with respect to time gives:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \;=\; \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}. \qquad (13)$$

This equation says that the motion of a point in the world is made up of two components. The first is the projection of the scene flow on the plane tangent to the surface and passing through $\mathbf{x}$. This is obtained by taking the instantaneous motion on the image plane (the optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$), and projecting it out into the scene using the inverse Jacobian $\frac{\partial \mathbf{x}}{\partial \mathbf{u}_i}$.

The second term is the contribution to scene flow arising from the three-dimensional motion of the point in the scene imaged by a fixed pixel. It is the instantaneous motion of $\mathbf{x}$ along the ray corresponding to $\mathbf{u}_i$. The magnitude of $\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}$ is (proportional to) the rate of change of the depth of the surface f along this ray. A derivation of $\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}$ is presented in Appendix A.

There are three major ways of computing scene flow, depending upon what is known about the scene at that instant:

$t = 1$ $t = 2$ $t = 3$ $t = 4$ $t = 5$

Figure 2: A sequence of images that show the scene motion. For lack of space, we only present scene flow results for $t = 1$ in this paper. The extended sequence is presented to help the reader visualize the three-dimensional motion.

1. Completely known instantaneous structure of the scene, including surface normals, depth maps, and the temporal rate of change of these depth maps.
2. Knowledge only of stereo correspondences. Since we are working in a calibrated setting, this is equivalent to having the depth maps. However, it does not include the surface normals and the temporal rates of change of the depth maps.
3. Completely unknown scene structure. We do not even know correspondence information.

Each of these cases leads to a different strategy for estimating the scene flow. It seems intuitive that less knowledge of scene structure requires the use of more optical flows, and indeed this result does follow from the amount of degeneracy in the linear equations used to compute scene flow. We now describe algorithms for each of the three cases. We also demonstrate their validity using flow results computed from multiple image sequences (captured from various viewpoints) of a non-rigid, dynamically changing scene. One such image sequence is shown in Figure 2.

4.1 Complete Knowledge of Surface Geometry

If the surface f is completely known (with high accuracy), the surface gradient $\boldsymbol{\nabla} f$ can be computed at every point. The inverse Jacobian $\frac{\partial \mathbf{x}}{\partial \mathbf{u}_i}$ can then be estimated by solving the set of 6 linear equations in Equations (3) and (4). Given the inverse Jacobian, the scene flow can be estimated from the optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ using Equation (13):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i}\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}. \tag{14}$$

Computing $\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}$ requires the temporal derivative of the surface depth map, and is described in Appendix A.

Complete knowledge of the scene structure thus enables us to compute scene flow from one optical flow, (and the rate of change of the depth map corresponding to this image.) These two pieces of information correspond to the two components of the scene flow; the optical flow is projected onto the tangent plane passing through $\mathbf{x}$, and the rate of change of depth map is mapped onto a component along the ray passing through the scene point $\mathbf{x}$ and the center of projection of the camera.

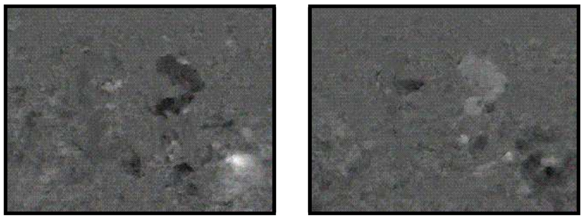

Figure 3: The horizontal and vertical optical flows at $t = 1$ for the same camera used in Figure 2. Darker pixels indicate motion to the left and top of the frames respectively.

Note that we assume that the surface is locally planar when computing the inverse Jacobian. Since the surface is known, it is possible to project the "flowed" point in the image and intersect this ray with the surface. We currently do not perform this to save an expensive ray-surface intersection for every pixel.

If only one optical flow is used, the scene flow can be computed only for those points in the scene that are visible in that image. It is possible to use multiple optical flows in multiple cameras for better visibility, and for greater robustness. Also, flow is recovered only when the change in depth map is valid - that is, when an individual pixel sees neighboring parts of the surface as time changes. If the motion of a surface is large relative to its size, then a pixel views different surfaces, and flow cannot be computed.

Consider the scene shown as a sequence in Figure 2. For lack of space, we only present scene flow results for time $t = 1$ in this paper. The scene flow is computed using depth maps from the model obtained from the volumetric merging of multiple range images, each computed using stereo, as described in [Rander *et al.*, 1996]. Another input is the optical flow shown in Figure 3, which was computed using a hierarchical version of the Lucas-Kanade optical flow algorithm. The final input is the temporal rate of change of the depth map, estimated from the difference between two independently computed volumetric models.

Figure 4 shows the result of computing scene flow for the visible set of points on the model. The original points are shown in light grey. The points that the original points have "flowed to" by adding the scene flow are shown in black. The darker points therefore represent a prediction

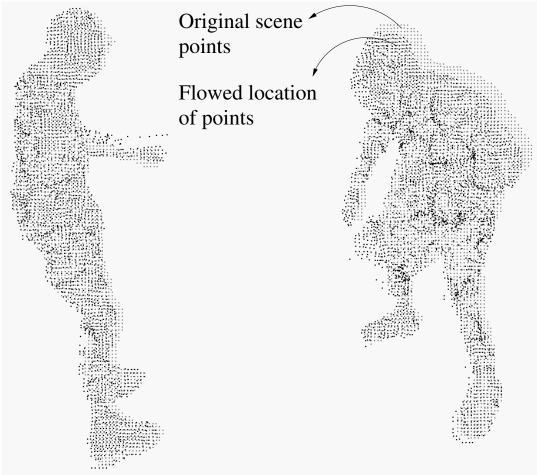

Figure 4: The computed scene flow. The original points on the model are shown in grey. The locations that these points have "flowed" to are shown in black. These flowed points form a prediction of the model at time $t = 2$.

of the model at time $t = 2$. They could be used to enhance the efficiency and robustness of shape recovery. In Figure 4, it is seen that the bending motion of the player on the right (the player with the ball) is recovered, as is the downward (and somewhat sideways) motion of the ball. The major motion of the player of the left (the player facing the ball) is the upward motion of his left arm, which is partially recovered. No flow is recovered for some points on the arm because the arm moves very fast relative to its size. Many pixels see completely different surfaces even during one time-step. Therefore the rate of change of depth information for the points on those surfaces is invalid yielding no flow estimate for those points.

4.2 Known Image Correspondences

The second major case is when the structure of the scene is not completely known, but correspondences between images are available. In our calibrated setting, correspondences can be used to compute depth maps, but these depth maps may be too noisy to estimate surface normals and temporal rates of change. This situation is common. For example, it is typical in image based modeling and rendering problems. While these problems typically only consider static scenes, scene flow can be used as a means of extending image based modeling methodologies into the temporal domain for dynamic scenes.

If the surface is not completely known, it is not possible to solve for $\frac{d\mathbf{x}}{dt}$ directly from one camera. Instead, consider the implicit linear relation between the scene and the optical flow, as described by the Jacobian in Equation (12).

This set of equations provides two linear constraints on $\frac{d\mathbf{x}}{dt}$. Therefore, if we have $N > 2$ cameras, we can solve for $\frac{d\mathbf{x}}{dt}$, by setting up the system of equations $\mathbf{B}\mathbf{x} = \mathbf{U}$,

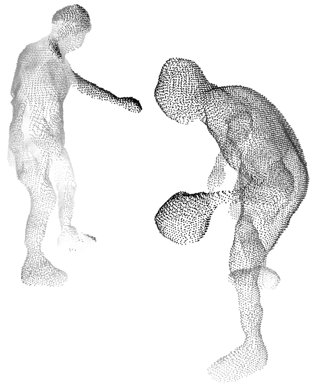

Figure 5: The magnitude of the scene flow is displayed for points on the model (the locations of which are obtained by projecting depth maps from 4 cameras into the scene). The magnitude of the scene flow is displayed as intensity. It can be seen that the largest motion occurs on the ball and the arm of the person at the rear, while the smallest motion is near the feet of the players.

where:

$$\mathbf{B} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z} \\ \frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} & \frac{\partial v_1}{\partial z} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial u_N}{\partial x} & \frac{\partial u_N}{\partial y} & \frac{\partial u_N}{\partial z} \\ \frac{\partial v_N}{\partial x} & \frac{\partial v_N}{\partial y} & \frac{\partial v_N}{\partial z} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \frac{\partial u_1}{\partial t} \\ \frac{\partial v_1}{\partial t} \\ \cdot \\ \cdot \\ \frac{\partial u_N}{\partial t} \\ \frac{\partial v_N}{\partial t} \end{bmatrix} \qquad (15)$$

This gives us $2N$ equations in 3 unknowns, and so for $N \geq 2$ we have an overconstrained system and can find an estimate of the scene flow. (This system of equations is degenerate if and only if the point $\mathbf{x}$ and the N camera centers are co-linear.) A singular value decomposition of $\mathbf{B}$ gives the solution that minimizes the sum of least squares of the error obtained by re-projecting the scene flow onto each of the optical flows.

We implemented the above algorithm and applied it to the same sequence that was used in the previous section, but without using the surface normal or rate of change of the depth map. We used optical flows from 15 different cameras. The use of this many optical flows ensures that every point at which we desire to compute scene flow is viewed by at least 2 or 3 cameras.

Figure 5 shows the magnitude of the computed scene flow. The absolute values of the computed flows in the x, y, and z directions are averaged and displayed as the intensity for each point. Since image correspondences along with camera calibration give us depth maps, the scene flow is computed for a set of points, the locations of which are obtained by projecting depth maps from 4 widely separated cameras into the scene. It is seen that the motion of the ball,

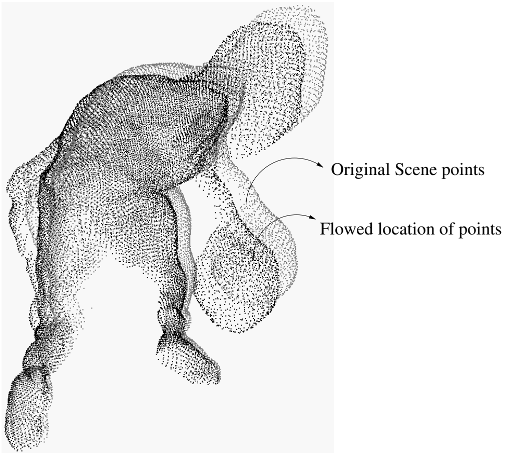

Figure 6: The initial point cloud is shown in light grey, while the set of points that these points have flowed to are shown in black. As can be seen, there is significant three-dimensional motion of the player downwards and to their right.

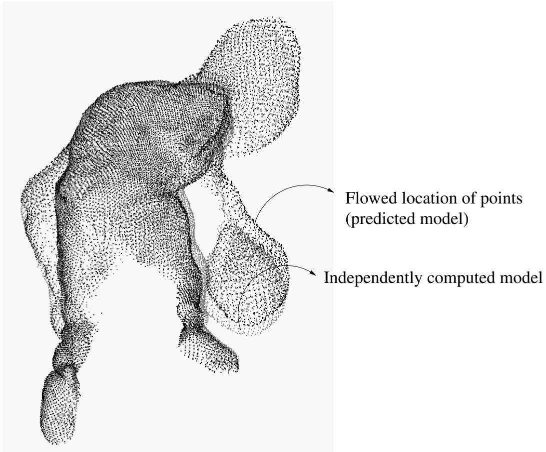

Figure 7: A comparison of the flowed points (black) with an independently computed model at the next next time instant (light grey). It is seen that the flowed points are a good approximation of the model at the next time instant.

and the vertical motion of the left arm of the person at the rear are the most significant.

A close up of the player holding the ball is displayed in Figures 6 and 7. In Figure 6, the light grey points represent the model at $t = 1$, which are displaced by the estimated scene flow to give the darker colored points. The same *flowed* points are shown in Figure 7, except that the light grey points now represent the model at $t = 2$, computed independently using stereo and volumetric merging [Rander *et al.*, 1996]. The figures clearly show that the displacement of points using the scene flow results in them moving almost exactly onto the "true" model. Hence, scene flow may be used as a predictor for the structure of the scene at subsequent time intervals.

4.3 No Knowledge of the Surface

If the point $\mathbf{x}$ lies on the surface, Equation (12) must hold for every camera i. Therefore, it is possible to use the degree to which Equation (12) is consistent across cameras as information for a flow-based reconstruction algorithm. Such an approach would, however, be very susceptible to outliers. A single large magnitude flow which was wrong could always make the equations inconsistent. We therefore take a slightly different approach.

The solution of Equation (12) can be written in the following form:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \;=\; \left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \mu\, \mathbf{r}_i(\mathbf{u}_i) \qquad (16)$$

where $\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star}$ is the pseudo-inverse of $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$, $\mathbf{r}_i(\mathbf{u}_i)$ is the direction of a ray through the pixel $\mathbf{u}_i$ (see Appendix A), and μ is an unknown constant that depends upon instantaneous properties of the surface f. (Equation (16) holds because $\mathbf{r}_i(\mathbf{u}_i)$ is in the null-space of $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$). Therefore, we have the following constraint on the the scene flow:

$$\mathbf{m}_i(\mathbf{x}) \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \equiv \left[\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \times \mathbf{r}_i(\mathbf{u}_i)\right] \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = 0 \quad (17)$$

where $\mathbf{m}_i(\mathbf{x}) = \left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \times \mathbf{r}_i(\mathbf{u}_i)$ is a vector which is perpendicular to the plane defined by the camera center and the optical flow in the image plane. Hence, if $\mathbf{x}$ is actually a point on the surface, the vectors $\mathbf{m}_i(\mathbf{x})$ should all lie in a plane (the one perpendicular to the scene flow $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$). We form a measure of how coplanar the vectors are by computing the 3×3 matrix:

$$M(\mathbf{x}) \;=\; \sum_i \hat{\mathbf{m}}_i \hat{\mathbf{m}}_i^T \qquad (18)$$

where $\hat{\mathbf{m}}_i$ is $\mathbf{m}_i$ normalized to a unit vector. The normalization makes the algorithm less susceptible to incorrect large magnitude flows. The smallest eigenvalue $\lambda = \lambda(\mathbf{x})$ of M is a measure of non-coplanarity. We therefore use $N - \lambda(\mathbf{x})$ as a measure of the likelihood that $\mathbf{x}$ lies on the surface. (N is the number of cameras.)

We discretize the scene into a three-dimensional array of voxels, as was done in the Voxel Coloring algorithm of [Seitz and Dyer, 1997]. We then compute $N - \lambda(\mathbf{x})$ for each voxel, ignoring visibility as in [Collins, 1996]. Ignoring visibility in this way does not significantly affect the performance because our coplanarity measure is not significantly affected by outliers. (This algorithm could be extended to keep track of the visibility in a similar manner to [Seitz and Dyer, 1997] if so desired.)

We present the results of this algorithm in Figure 8. We used the data from all 51 cameras of the CMU Virtualized Reality dome [Narayanan *et al.*, 1998] (Some of the data

Figure 8: A volume rendering of the coplanarity measure $N - \lambda(\mathbf{x})$. As can be seen, the gross scene structure is recovered fairly well. Note, however, that this algorithm only recovers structure where the scene is moving. Hence, certain parts of the scene, such as the legs, are not recovered as well as others. The information provided by $N - \lambda(\mathbf{x})$ could be combined with traditional sources of information to further enhance the robustness of stereo.

from one camera is presented in Figure 2). For all 51 cameras, we computed the optical flow from $t = 1$ to $t = 2$. The measure of coplanarity $N - \lambda(\mathbf{x})$ was then computed for each voxel and thresholded. Figure 8 contains a volume rendering of the results. As can be seen, the gross structure of the scene is recovered. Note, however, that this flow-based reconstruction algorithm can only recover structure where the scene is actually moving. This is the reason that certain parts of the scene, such as the legs of the people, are not fully recovered.

4.4 Three-Dimensional Normal Flow Constraint

Optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ is a two dimensional vector field, and so is often divided into two components, the *normal flow* and the *tangent flow*. The normal flow is the component in the direction of the image gradient $\boldsymbol{\nabla} I_i$, and the tangent flow is the component perpendicular to the normal flow. The magnitude of the normal flow can be estimated directly from Equation (11) as:

$$\frac{1}{|\boldsymbol{\nabla} I_i|}\boldsymbol{\nabla} I_i \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \; = \; -\frac{1}{|\boldsymbol{\nabla} I_i|}\frac{\partial I_i}{\partial t}. \tag{19}$$

Estimating the tangent flow is an ill-posed problem. Hence, some form of local smoothness is required to estimate the complete optical flow [Barron *et al.*, 1994]. Since the estimation of the tangent flow is the major difficulty in most algorithms, it is natural to ask whether the normal flows from several cameras can be used to estimate the 3D scene flow without having to use some form of regularization.

The Normal Flow Constraint Equation (11) can be rewritten as:

$$\boldsymbol{\nabla} I_i \cdot \left[\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right] + \frac{\partial I_i}{\partial t} \; = \; 0. \tag{20}$$

This is a scalar linear constraint on the components of the scene flow $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. Therefore, at first glance it seems likely that it might be possible to estimate the scene flow directly from three such constraints. Unfortunately, differentiating Equation (7) with respect to $\mathbf{x}$ we see that:

$$\boldsymbol{\nabla} I_i \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \; = \; -C \cdot \boldsymbol{\nabla}\left(\rho(\mathbf{x};t)\left[\mathbf{n}(\mathbf{x};t)\cdot \mathbf{s}(\mathbf{x};t)\right]\right). \tag{21}$$

Since this expression is independent of the camera i, and instead only depends on properties of the scene (the surface albedo ρ, the scene structure $\mathbf{n}$, and the illumination $\mathbf{s}$), the coefficients of $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ in Equation (20) should ideally always be the same. Hence, any number of copies of Equation (20) will be linearly dependent. In fact, if the equations turn out not to be linearly dependent, this fact can be used to deduce that $\mathbf{x}$ is not a point on the surface. (See Section 4.3.)

This result means that it is impossible to compute 3D scene flow independently for each point on the object, without some form of regularization of the problem. We wish to emphasize, however, that this result does not mean that is it not possible to estimate other useful quantities directly from the normal flow, as for example is done in [Negahdaripour and Horn, 1987] and other "direct methods."

5 Conclusions

Three-dimensional scene flow is a fundamental property of dynamic scenes. It can be used as a prediction mechanism to build more robust stereo algorithms, and for various scene interpretation and rendering tasks. We have presented a framework for computing scene flow from optical flow, assuming various instantaneous properties of the scene are known. We intend to extend our framework to incorporate knowledge of structure computed independently at the next time instant. We also plan to investigate other algorithms for computing scene flow that do not use optical flow, and develop methods of quantitatively evaluating their accuracy.

A Computing $\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}$

The term $\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i}$ is the 3D motion of the point in the scene imaged by the pixel $\mathbf{u}_i$. Suppose the depth of the surface measured from the i^{th} camera is $d_i = d_i(\mathbf{u}_i)$. Then, the point $\mathbf{x}$ can be written as a function of $\mathbf{P}_i$, $\mathbf{u}_i$, and d_i as follows. The 3×4 camera matrix $\mathbf{P}_i$ can be written as:

$$\mathbf{P}_i \; = \; [\, \mathbf{R}_i \; \mathbf{T}_i \,] \tag{22}$$

where $\mathbf{R}_i$ is a 3×3 matrix and $\mathbf{T}_i$ is a 3×1 vector. The center of projection of the camera is $-\mathbf{R}_i^{-1}\mathbf{T}_i$, the direction of the ray through the pixel $\mathbf{u}_i$ is $\mathbf{r}_i(\mathbf{u}_i) = \mathbf{R}_i^{-1}(u_i, v_i, 1)^{\mathrm{T}}$, and the direction of the camera z-axis is $\mathbf{r}_i(\mathbf{0}) = \mathbf{R}_i^{-1}(0,0,1)^{\mathrm{T}}$. Using simple geometry, (see Figure 9) we therefore have:

$$\mathbf{x} \; = \; -\mathbf{R}_i^{-1}\mathbf{T}_i + d_i\left[\frac{\|\mathbf{r}_i(\mathbf{0})\|\,\mathbf{r}_i(\mathbf{u}_i)}{\mathbf{r}_i(\mathbf{0})\cdot \mathbf{r}_i(\mathbf{u}_i)}\right]. \tag{23}$$

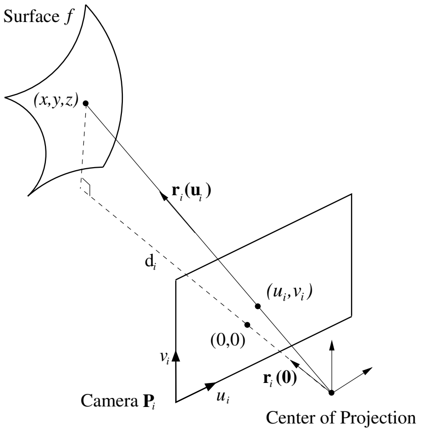

Figure 9: Given the camera matrix $\mathbf{P}_i$ and the distance d_i to the surface, the direction of the ray through the pixel $\mathbf{u}_i$ and the direction of the z-axis of the camera can be used to derive an expression for the point $\mathbf{x}$. This expression can be symbolically differentiated to give $\frac{\partial \mathbf{x}}{\partial t}\big|_{\mathbf{u}_i}$ as a function of $\mathbf{x}$, $\mathbf{P}_i$, and $\frac{\partial d_i}{\partial t}$.

(Care must be taken to choose the sign of $\mathbf{P}_i$ correctly so that the vector $\mathbf{r}_i(\mathbf{u}_i)$ points out into the scene.) If camera $\mathbf{P}_i$ is fixed, we have:

$$\left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{u}_i} = \left[\frac{\|\mathbf{r}_i(\mathbf{0})\|\, \mathbf{r}_i(\mathbf{u}_i)}{\mathbf{r}_i(\mathbf{0}) \cdot \mathbf{r}_i(\mathbf{u}_i)}\right] \frac{\partial d_i}{\partial t}. \quad (24)$$

So, the magnitude of $\frac{\partial \mathbf{x}}{\partial t}\big|_{\mathbf{u}_i}$ is proportional to the rate of change of the depth map and the direction is along the ray joining $\mathbf{x}$ and the center of projection of the camera.

References

[Adelson and Bergen, 1991] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In Landy and Movshon, editors, *Computational Models of Visual Processing*. MIT Press, 1991.

[Avidan and Shashua, 1998] S. Avidan and A. Shashua. Non-rigid parallax for 3D linear motion. In *Proc. of the DARPA IUW*, 1998. (To appear in CVPR '99).

[Barron *et al.*, 1994] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

[Collins, 1996] R.T. Collins. A space-sweep approach to true multi-image matching. In *Proc. of CVPR '96*, pages 358–363, 1996.

[Costeira and Kanade, 1998] J.P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998.

[Horn, 1986] B.K.P. Horn. *Robot Vision*. McGraw Hill, 1986.

[Liao *et al.*, 1997] W.-H. Liao, S.J. Aggrawal, and J.K. Aggrawal. The reconstruction of dynamic 3D structure of biological objects using stereo microscope images. *Machine Vision and Applications*, 9:166–178, 1997.

[Malassiotis and Strintzis, 1997] S. Malassiotis and M.G. Strintzis. Model-based joint motion and structure estimation from stereo images. *CVIU*, 65(1):79–94, 1997.

[Metaxas and Terzopoulos, 1993] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE PAMI*, 15(6):580–591, 1993.

[Narayanan *et al.*, 1998] P.J Narayanan, P.W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. of the Sixth ICCV*, pages 3–10, 1998.

[Negahdaripour and Horn, 1987] S. Negahdaripour and B.K.P. Horn. Direct passive navigation. *PAMI*, 9(1):168–176, 1987.

[Penna, 1994] M.A. Penna. The incremental approximation of nonrigid motion. *CVGIP*, 60(2):141–156, 1994.

[Pentland and Horowitz, 1991] A.P. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE PAMI*, 13(7):730–742, 1991.

[Rander *et al.*, 1996] P.W. Rander, P.J Narayanan, and T. Kanade. Recovery of dynamic scene structure from multiple image sequences. In *Proc. of the 1996 Intl. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 305–312, 1996.

[Seitz and Dyer, 1997] S.M. Seitz and C.M. Dyer. Photorealistic scene reconstrcution by space coloring. In *Proc. of CVPR '97*, pages 1067–1073, 1997.

[Shi *et al.*, 1994] Y.Q. Shi, C.Q. Shu, and J.N. Pan. Unified optical flow field approach to motion analysis from a sequence of stereo images. *Pattern Recognition*, 27(12):1577–1590, 1994.

[Ullman, 1979] S. Ullman. *The Interpretation of Visual Motion*. MIT Press, 1979.

[Ullman, 1984] S. Ullman. Maximizing the rigidity: The incremental recovery of 3-D shape and nonrigid motion. *Perception*, 13:255–274, 1984.

[Waxman and Duncan, 1986] Allen M. Waxman and James H. Duncan. Binocular image flows: Steps toward stereo-motion fusion. *IEEE PAMI*, 8(6):715–729, 1986.

[Young and Chellappa, 1999] G.S. Young and R. Chellappa. 3-D motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness. *IEEE PAMI*, 12(8):735–759, 1999.

[Zhang and Faugeras, 1992a] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

[Zhang and Faugeras, 1992b] Z. Zhang and O. Faugeras. Estimation of displacements from two 3-D frames obtained from stereo. *IEEE PAMI*, 14(12):1141–1156, 1992.