Three-Dimensional Scene Flow

Sundar Vedula, Simon Baker,
Peter Rander, *Member*, *IEEE*,
Robert Collins, *Member*, *IEEE*, and
Takeo Kanade, *Fellow*, *IEEE*

Abstract—Just as optical flow is the two-dimensional motion of points in an image, scene flow is the three-dimensional motion of points in the world. The fundamental difficulty with optical flow is that only the normal flow can be computed directly from the image measurements, without some form of smoothing or regularization. In this paper, we begin by showing that the same fundamental limitation applies to scene flow; however, many cameras are used to image the scene. There are then two choices when computing scene flow: 1) perform the regularization in the images or 2) perform the regularization on the surface of the object in the scene. In this paper, we choose to compute scene flow using regularization in the images. We describe three algorithms, the first two for computing scene flow from optical flows and the third for constraining scene structure from the inconsistencies in multiple optical flows.

Index Terms—Scene flow, three-dimensional dense nonrigid motion, optical flow, the brightness constancy constraint, normal flow, three-dimensional normal flow.

1 Introduction

One of the fundamental properties of any scene is its motion. When a dynamically changing scene is imaged by a video camera, the *optical flow* between two neighboring frames tells us something about the motion of the scene. However, optical flow is just 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 nonrigid, the motions of the points in the scene may all be independent of each other. The scene motion is therefore a dense three-dimensional vector field defined for each point on every surface in the scene. By analogy with optical flow, we refer to this three-dimensional motion field as the *scene flow* [22].

Computing the three-dimensional motion of a scene is a basic task in computer vision that has been approached in a wide variety of ways. If the scene is rigid, 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* [18]. If the scene is only piecewise rigid, extensions to structure-from-motion algorithms can be used. See, for example, [6].

Although restricted forms of nonrigidity can be analyzed using the structure-from-motion paradigm [2], general nonrigid motion cannot be estimated from a single camera without additional assumptions. Given strong enough a priori assumptions about the scene, for example, in the form of a deformable model [12], [15] or the assumption that the motion minimizes the deviation from a rigid body motion [19], recovering three-dimensional nonrigid motion from a monocular view is possible. See [14] for a survey of monocular nonrigid motion estimation.

- S. Vedula can be reached at 970 Corte Madera Ct. #204, Sunnyvale, CA 94085. E-mail: srv@cs.cmu.edu.
- S. Baker, R. Collins, and T. Kanade are with the The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. E-mail: {simonb, rcollins, tk}@cs.cmu.edu.
- P. Rander is with the National Robotics Engineering Consortium, 10 40th Street, Pittsburgh, PA 15201. E-mail: rander@cs.smu.edu.

Manuscript received 15 Oct. 2002; revised 24 Feb. 2004; accepted 7 Sept. 2004; published online 14 Jan. 2005.
Recommended for acceptance by M. Irani.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 117587.

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, [23], [24]. A few motion-stereo papers do consider nonrigid motion, including [9], [11]. The first of these papers uses a relaxation-based algorithm to cooperatively match features in both the temporal and spatial domains. It therefore does not provide dense motion. The second uses a grid which acts as a deformable model in a generalization of the monocular approaches mentioned above.

The well-known difficulty with optical flow is that only the normal flow can be computed directly from the image measurements, without some form of smoothing or regularization. In this paper, we first show that the same fundamental limitation applies to scene flow. However, if many cameras are used to image the scene, it is impossible to compute scene flow directly from the image measurements without any smoothing or regularization (except at points with more than 1D structure, such as corner points.) This result holds for the Lambertian reflectance model and any generalization of it, which includes most common reflectance functions.

Given this result, there are then two options if we wish to compute dense scene flow: 1) perform the regularization in the images or 2) perform the regularization on the surface of the object in the scene. Since our original scene flow paper [22], a few authors have proposed elegant formulations of scene flow in the scene domain, perhaps most notably [5]. In this paper, we investigate scene flow algorithms where the regularization is performed in the image plane. We describe three algorithms: The first two algorithms compute scene flow from (regularized) optical flows assuming the scene shape is known. One algorithm uses a single camera, the other uses multiple cameras. The third algorithm computes scene structure from the inconsistencies in multiple optical flows. All three algorithms require fully calibrated cameras.

2 Background

Consider a nonrigidly moving surface $S^t(\mathbf{x}) = 0$ imaged by a fixed camera C_i with 3×4 projection matrix $\mathbf{P}_i$, as illustrated in Fig. 1. There are two aspects to the formation of the image sequence $I_i^t = I_i^t(u_i, v_i)$ captured by camera C_i: 1) the geometry and 2) the photometrics. We now describe each of these components in turn and then review the brightness constancy basis for optical flow.

2.1 Relative Camera and Surface Geometry

The relationship between a 3D world point $\mathbf{x} = (x, y, z)^{\mathrm{T}}$ on the surface and the 2D image coordinates $\mathbf{u}_i = (u_i, v_i)^{\mathrm{T}}$ of its projection in camera C_i is given by:

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

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

where $[\mathbf{P}_i]_j$ is the jth row of $\mathbf{P}_i$. Without knowing the surface, these equations are not invertible. If S^t is known, they can be inverted, but the inversion requires intersecting a ray with $S^t(\mathbf{x}) = 0$.

At fixed time t, the differential relationship between $\mathbf{x}$ and $\mathbf{u}_i$ is represented by a 2×3 Jacobian matrix $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$. The three columns of the Jacobian matrix store the differential change in projected image coordinates 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 (1) and (2) symbolically [20]. 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.

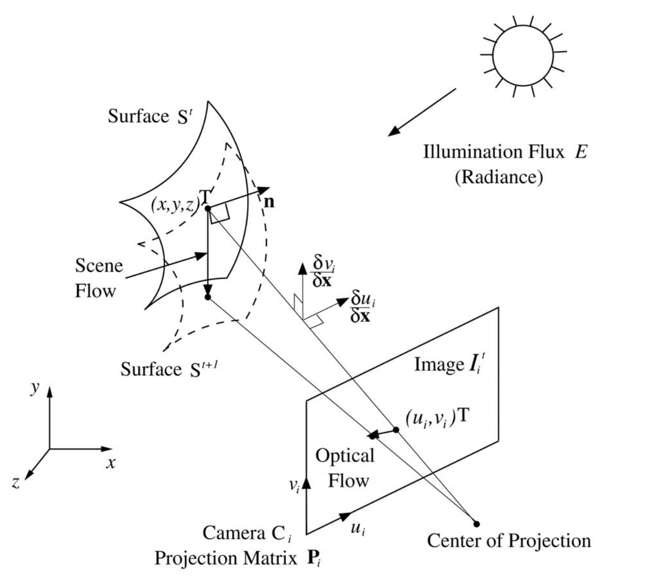
Surface S^t
Illumination Flux E (Radiance)
$(x,y,z)^T$
n
Scene Flow
$\frac{\delta v_i}{\delta \mathbf{x}}$
$\frac{\delta u_i}{\delta \mathbf{x}}$
Surface S^{t+1}
Image I_i^t
$(u_i,v_i)^T$
Optical Flow
v_i
u_i
y
x
z
Camera C_i
Projection Matrix $\mathbf{P}_i$
Center of Projection

Fig. 1. A nonrigid surface $S^t(\mathbf{x}) = 0$ is moving with respect to a fixed coordinate system $\mathbf{x} = (x, y, z)^{\mathrm{T}}$. 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 (radiance) is E. The camera C_i is fixed in space, has a coordinate frame $(u_i, v_i)^{\mathrm{T}}$, is represented by the 3×4 projection matrix $\mathbf{P}_i$, and captures the image sequence $I_i^t = I_i^t(u_i, v_i)$. The scene flow is the instantaneous 3D motion of every point on every surface in the scene, illustrated here for a single point $(x, y, z)^{\mathrm{T}}$. Optical flow is the 2D projection of the scene flow into an image.

2.2 Illumination and Surface Photometrics

Suppose that the illumination flux or radiance of light in the scene is $E = E(\mathbf{m}; \mathbf{x}; t)$. The radiance E depends upon the scene point $\mathbf{x}$ where it is measured, the direction $\mathbf{m}$ in which it is measured, and the time t at which it is measured [7]. This 6D radiance function E is just a simplified version of the *plenoptic function* [1].

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

$$\mathbf{r}(x, y, z; t) = \int_{H(\mathbf{n})} E(\mathbf{m}; x, y, z; t)\, \mathrm{d}\mathbf{m}, \tag{3}$$

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

Assume that the surface S^t is Lambertian with albedo $\rho = \rho(\mathbf{x}; t)$. If the point $\mathbf{x} = (x, y, z)^{\mathrm{T}}$ is visible in camera C_i and the intensity registered in image I_i^t is proportional to the radiance of the point that it is the image of (i.e., image irradiance is proportional to scene radiance [7]), we have:

$$I_i^t(\mathbf{u}_i) = -K \cdot \rho(\mathbf{x}; t)\, [\mathbf{n}(\mathbf{x}; t) \cdot \mathbf{r}(\mathbf{x}; t)], \tag{4}$$

where K is a constant that only depends upon the diameter of the lens and the distance between the lens and the image plane [7]. The pixel $\mathbf{u}_i$ and 3D point $\mathbf{x}$ are related by (1) and (2).

2.3 Two-Dimensional Optical Flow

Suppose that $\mathbf{u}_i(t)$ is the 2D path of the image of the point $\mathbf{x}(t)$ in camera C_i. The optical flow is the 2D motion $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ in the image plane of camera C_i. As the 3D point $\mathbf{x}(t)$ on the surface moves in the scene, we assume that that its albedo $\rho = \rho(\mathbf{x}(t))$ remains constant; i.e., we assume that:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = 0. \tag{5}$$

The basis for optical flow algorithms is then the time-derivative of (4):

$$\frac{\mathrm{d}I_i^t}{\mathrm{d}t} = \nabla I_i^t \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial I_i^t}{\partial t} = -K \cdot \rho(\mathbf{x})\, \frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{r}], \tag{6}$$

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

The term $\mathbf{n} \cdot \mathbf{r}$ depends upon both the shape of the surface ($\mathbf{n}$) and the illumination ($\mathbf{r}$). To avoid explicit dependence upon the structure of the three-dimensional scene, it is often assumed[1] that

$$\mathbf{n} \cdot \mathbf{r} = \int_{H(\mathbf{n})} E(\mathbf{m}; \mathbf{x})\, \mathbf{n} \cdot \mathrm{d}\mathbf{m} \tag{7}$$

is constant ($\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{r}] = 0$). There are at least two common scenarios where this assumption holds well: 1) the illumination is constant or 2) the surface normal does not change rapidly. Assuming that $\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{r}] = 0$, we arrive at the *Optical Flow Constraint* equation:

$$\nabla I_i^t \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial I_i^t}{\partial t} = 0. \tag{8}$$

Using this constraint, an example of the *brightness constancy* assumption, a large number of differential algorithms have been proposed to estimate the optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$. The most important thing to note about (8), however, is that it only provides a constraint on the optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ in the direction of the image gradient ∇I_i^t, or the normal component of the optical flow:

$$\frac{1}{|\nabla I_i^t|} \nabla I_i^t \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = -\frac{1}{|\nabla I_i^t|} \frac{\partial I_i^t}{\partial t}. \tag{9}$$

Hence, to compute dense optical flow, some form of regularization or smoothing is needed. Most research on optical flow has focused on different methods of performing this regularization [3].

3 Three-Dimensional Scene Flow

Just as an optical flow is a two-dimensional motion field in an image, the scene flow is a three-dimensional flow field describing the motion at every point in the scene [22]. If $\mathbf{x} = \mathbf{x}(t)$ is the 3D path of a point in the world, its instantaneous scene flow is $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. If the image of this point in camera C_i is $\mathbf{u}_i = \mathbf{u}_i(t)$, the optical flow is simply the projection of the scene flow into the image plane:

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}, \tag{10}$$

where $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$ is the Jacobian introduced in Section 2.1. This equation tells us how optical flow can be computed if the scene flow is known, but not the other way around.

3.1 Is Scene Flow Computable from Multiple Normal Flows?

Since only the normal component of the optical flow can be computed without regularization, as shown earlier, perhaps the most important question to ask is whether it is possible to combine the normal flows from several cameras to estimate the scene flow without having to regularize the problem? The gradient constraint equation (8) can be rewritten as

$$\nabla I_i^t \cdot \left[\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \right] + \frac{\partial I_i^t}{\partial t} = 0. \tag{11}$$

1. Note that the assumption that $\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{r}] = 0$ is required in addition to the Lambertian assumption to derive the optical flow constraint in (8). This "hidden assumption" is often overlooked.

This is a linear constraint on the three unknown components of the scene flow $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. At first glance, therefore, it seems likely that it might be possible to estimate the scene flow directly from three such constraints. Differentiating[2] (4) with respect to $\mathbf{x}$, we see that:

$$\nabla I_i^t \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} = -K \cdot \nabla(\rho(\mathbf{x};t)\,[\mathbf{n}(\mathbf{x};t)\cdot \mathbf{r}(\mathbf{x};t)]). \tag{12}$$

Since this expression is independent of the camera C_i and instead only depends on properties of the scene (the surface albedo ρ, the scene structure $\mathbf{n}$, and the illumination $\mathbf{r}$), the coefficients of $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ in (11) should ideally always be the same. Hence, any number of copies of the equation 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, as will be shown in Section 3.4.

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. The derivation above uses the Lambertian model; however, it naturally also holds for any reflectance function that is a generalization of the Lambertian model in the sense that it equals the Lambertian model for certain (limiting) parameter settings. Most reasonable reflection models, such as the Torrance-Sparrow model [17], are generalization of the Lambertian model in this sense. Note, however, that this result does not imply that it is not possible to estimate other useful quantities directly from the normal flow [13].

There are two ways to perform the regularization required to compute scene flow: 1) on the surface of the object in the scene and 2) in the image (i.e., by first computing optical flow). In [5], Carceroni and Kutulakos showed how the brightness constancy equation can be generalized to the temporal domain and the motion of a 3D patch (which they refer to as a *surfel*) can be determined directly from the input images. Implicitly, this algorithm regularizes over the surface of the 3D object. In this paper, we consider the second option. We assume that optical flow has first been computed for each camera and then compute the scene flow. In particular, we describe three different ways in which the optical flows can be combined: 1) single camera, known scene geometry, 2) multiple cameras, known scene geometry, and 3) multiple cameras, unknown scene geometry. We now discuss each of these options in turn.

3.2 Single Camera, Known Scene Geometry

A natural question to ask is whether the scene flow can be computed from a single optical flow. As we showed in [22], theoretically, it can be. Computing scene flow requires inverting (10). 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:

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

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. See [22] for the full details of the single camera algorithm.

Using (13) to estimate the scene flow is very noisy. In particular, the rate of change of depth map (needed in the second term) is undefined around depth discontinuities. Moreover, multiple cameras are needed to estimate the rate of change of the depth map anyway. Hence, single camera scene flow is mainly only of theoretical interest. See [22] for example results.

2. Strictly, $\rho(\mathbf{x};t), [\mathbf{n}(\mathbf{x};t)\cdot\mathbf{r}(\mathbf{x};t)]$ is defined only on the 2D scene surface and, so, its 3D gradient is undefined. However, the definition can be extended into 3D by making it constant on all $\mathbf{x}$ that project to the same $\mathbf{u}_i$.

3.3 Multiple Cameras, Known Scene Geometry

With multiple optical flows, inverting (10) is far easier. First, the volume containing the geometry of the scene is partitioned into voxels. Then, the occupancy of each voxel is determined using the voxel coloring algorithm [16], where each voxel is projected into the various images and deemed to be occupied or not depending on the consistency of measurements from the various images. At the end of the voxel coloring approach, suppose the recovered scene geometry is:

$$S^t = \{\mathbf{x}_j^t \mid j = 1, \ldots, V^t\}, \tag{14}$$

where V^t is the total number of voxels recovered. In the process, voxel coloring also recovers the visibility, i.e., for each voxel $\mathbf{x}_j^t$, we compute the set of cameras $\mathrm{Vis}(\mathrm{x}_\mathrm{j}^\mathrm{t})$ that can see $\mathbf{x}_j$.

Equation (10) expresses the fact that any optical flow $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ is the projection of the scene flow $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. Assuming that the optical flow has been computed, it provides two linear constraints on the three unknowns in the scene flow. Therefore, if we have two or more cameras viewing a particular point in the scene, the scene flow can be recovered. In practice, we use as many optical flows as are available to estimate the scene flow.

Suppose there are N cameras in the set $\mathrm{Vis}(\mathbf{x_j})$ for a particular voxel. If we have $N > 2$, we can solve for $\frac{\mathrm{d}\mathrm{x_j}}{\mathrm{d}t}$ by setting up the system of equations $\mathbf{B}\frac{\mathrm{d}\mathbf{x_j}}{\mathrm{d}t} = \mathbf{U}$, where

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

This system of equations is degenerate if and only if the voxel $\mathbf{x_j}$ and the N camera centers are collinear. A singular value decomposition of $\mathbf{B}$ gives the solution that minimizes the sum of least squares of the error obtained by reprojecting the scene flow onto each of the optical flows. Note that it is critical that only the cameras in the set $\mathrm{Vis}(\mathbf{x_j})$ are used.

3.3.1 *Desirable Properties of the Scene Flow*

Suppose F^t is the recovered scene flow for the voxel model, i.e.,

$$F^t = \frac{\mathrm{d}S^t}{\mathrm{d}t} = \frac{\mathrm{d}\{\mathbf{x}_j^t \mid j = 1, \ldots, V^t\}}{dt}. \tag{16}$$

Note that S^t is defined in an asymmetric way; the voxel model at time $t+1$ is not even used. This seems undesirable. We would want that the voxel model at time t flowed forward to time $t+1$ to give us the voxel model at time $t+1$. Consider the following two properties:

Inclusion: Every voxel at time t should flow to a voxel at time $t+1$, i.e., $\forall t, i\ X_i^t + F_i^t \in S^{t+1}$.

Onto: Every voxel at time $t+1$ should have a voxel at time t that flows to it, $\forall t, i,\ \exists j$ s.t. $X_j^t + F_j^t = X_i^{t+1}$.

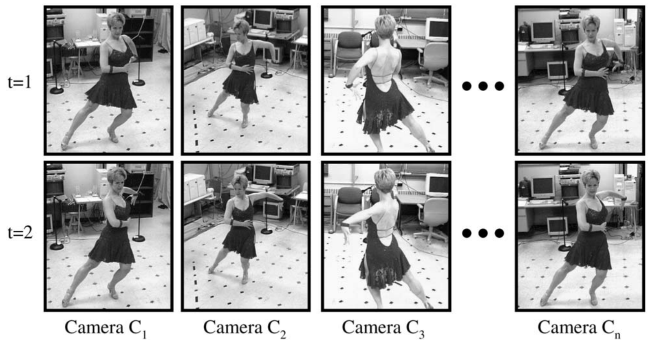

Fig. 2. The input to the scene flow algorithm consists of 17 image pairs (one set at $t = 1$, the other at $t = 2$) extracted from 17 video sequences captured in the CMU 3D Room [8]. At the moment the images were captured, the dancer was twisting her torso to the left and stretching out her right arm.

These properties immediately imply that the voxel model at time t flowed forward to time $t + 1$ is *exactly* the voxel model at time $t + 1$:

$$\{X_i^t + F_i^t \mid i = 1, \ldots, V^t\} = S^{t+1}. \tag{17}$$

This means that the scene shape will be continuous at $t + 1$ as the voxel model is *flowed* forward. In [20], we show how to enforce these properties without forcing the scene flow to be one-to-one by introducing what we call *duplicate voxels*, additional voxels introduced to satisfy the flow properties without altering the 3D geometry itself. See [20], [21] for the full details.

3.3.2 *Results*

Consider a dynamic event consisting of a dancer performing a Paso Doble dance sequence, shown in Fig. 2. The event is captured as a time-varying sequence from 17 cameras and images from a snapshot at two time instants are shown for four of these cameras. The image sequences were captured using the CMU 3D Room [8]. At the moment the images were captured, the dancer was twisting her torso to the left and stretching out her right arm.

Optical flow is computed between each pair of these images. We use a hierarchical version of the Lucas-Kanade algorithm [10], although other optical flow algorithms could have been used [3]. (This is a major advantage of regularizing in the image.) Fig. 3 shows one pair of horizontal and vertical optical flow fields computed for one pair of images. Darker pixels indicate a greater

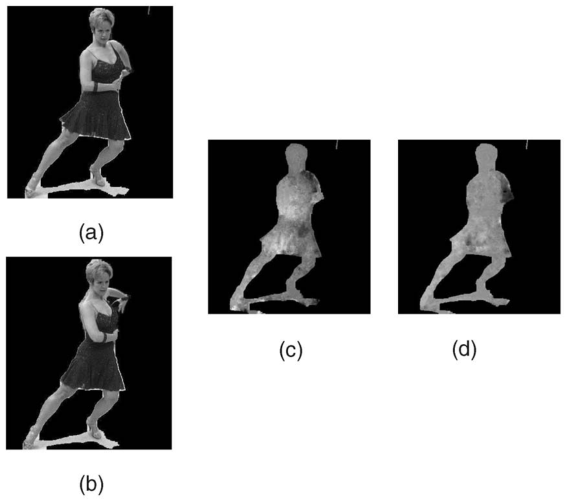

Fig. 3. Two input images (shown after background subtraction) and the horizontal and vertical 2D optical flow fields computed using a hierarchical version of the Lucas-Kanade algorithm [10]. Darker pixels indicate larger flow to the right and bottom in the horizontal and vertical flow fields, respectively.

motion to the right and bottom, while lighter pixels indicate motion to the left and upwards.

The results of scene flow computed on an $80 \times 80 \times 120$ voxel grid are included in Figs. 4, 5, and 6. In Fig. 4, the two colored voxel models, one for each time, are displayed. Fig. 5 shows two snapshots of the scene flow. Scene flow is a dense flow field, so a motion vector is computed for every voxel in the scene. The close-up snapshot on the right shows the motion of the voxels as the dancer raises and stretches out her arm. Fig. 6 shows the same flow vectors overlaid on the shape. Notice that the motion of the dancer is highly nonrigid. Also notice the smoothness of the flow field (for example, on the dancer's left hand, where the optical flow is noisy and by no means perfect, but the scene flow is much smoother across the surface). Clearly, redundancies across multiple cameras have resulted in a scene flow that is less noisy than any one of the input optical flows. It is also possible to use robust schemes for better elimination of outliers, such as those in [4].

3.4 Computing Shape from Inconsistencies of Multiple Optical Flows

Just as voxel coloring determines the scene shape based on consistency of pixel image measurements, we can also think of

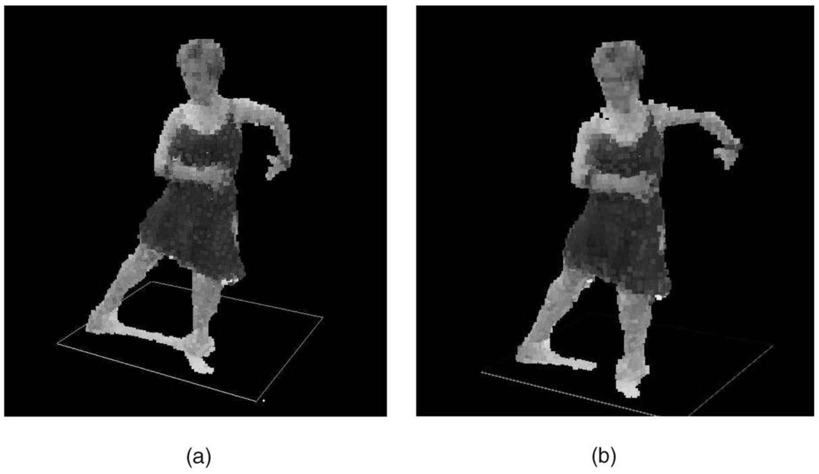

Fig. 4. The results of applying *voxel coloring* [16] to the inputs in Fig. 2. The voxel models are displayed with the voxels colored with the average color of the pixels that the voxel projects to in the inputs.

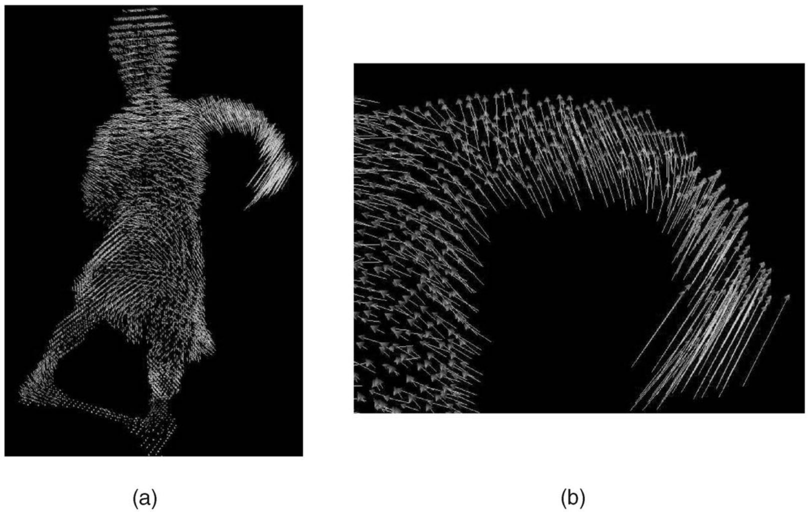

Fig. 5. The final scene flow computed from the inputs in Fig. 2: (a) the complete model and (b) a close up of the dancer's arm. Notice that the overall motion of the dancer is highly nonrigid.

shape as being the surface that yields consistent image optical flow measurements. If the point $\mathbf{x}$ lies on the surface, (10) must hold for every camera i. Therefore, it is possible to use the degree to which (10) 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 (10) can be written in the following form:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \mu\, \mathbf{r}_i(\mathbf{u}_i), \tag{18}$$

where $\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star}$ is the pseudoinverse of $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$, $\mathbf{r}_i(\mathbf{u}_i)$ is the direction of a ray through the pixel $\mathbf{u}_i$, and μ is an unknown constant that depends upon instantaneous properties of the surface f. (Equation (18) 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:

$$\mathbf{m}_i(\mathbf{x}) \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \equiv \left[\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^{\star} \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \times \mathbf{r}_i(\mathbf{u}_i)\right] \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = 0, \tag{19}$$

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:

$$M(\mathbf{x}) = \sum_i \hat{\mathbf{m}}_i \hat{\mathbf{m}}_i^T, \tag{20}$$

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 noncoplanarity. We use $N - \lambda(\mathbf{x})$ as a measure of the likelihood that $\mathbf{x}$ lies on the surface, where 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 [16]. We then compute $N - \lambda(\mathbf{x})$ for each voxel. We ignore visibility, but this does not significantly affect the performance because our coplanarity measure is not significantly affected by outliers. (The algorithm could be extended to keep track of the visibility, similar to the flow computation algorithm in the previous section.)

3.4.1 Results

We present the results of this algorithm in Fig. 7, for the same sequence shown in Fig. 2. For all 51 cameras, we computed the

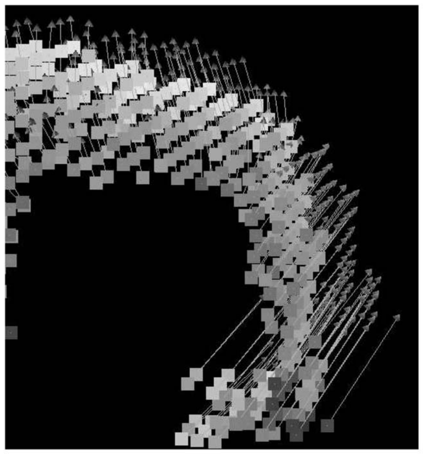

Fig. 6. The scene flow shown together with the voxel model; i.e., a combination of Fig. 4 and Fig. 5b.

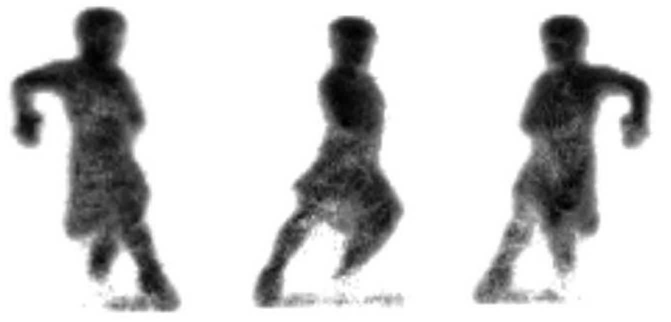

Fig. 7. Three volume renderings of the coplanarity measure $N - \lambda(\mathbf{x})$ show that the gross scene structure is recovered fairly well. Note, however, that this algorithm only recovers structure where the scene is moving. Hence, certain parts of the scene, such as the legs, are not recovered as well as others. This information could be combined with traditional sources of information to further enhance the robustness of stereo. See the accompanying movie, `volume.mpg`, available on the Computer Society's Digital Library at http://www.computer.org/publications/dlib, for a flyby movie of the volume rendering generated.

optical flow from $t = 1$ to $t = 2$. The measure of coplanarity $N - \lambda(\mathbf{x})$ was then computed for each voxel and thresholded. Fig. 7 contains three volume renderings of the results. (The accompanying movie, `volume.mpg`, available on the Computer Society's Digital Library at http://www.computer.org/publications/dlib, contains a flyby movie.) 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 dancer, are not fully recovered. In practice, the information in $N - \lambda(\mathbf{x})$ would probably be best used to enhance the robustness of a intensity-based 3D reconstruction algorithm.

4 Conclusions

Scene flow, or dense nonrigid 3D motion, is a fundamental property of dynamic scenes. In this paper, we have shown how the computation of scene flow is an ill-posed problem, needing some form of smoothing or regularization. We have also presented algorithms to compute scene flow from both a single and from multiple cameras. In these algorithms, the regularization is performed in image space in the optical flow algorithm. In contrast, Carceroni and Kutulakos [5] perform the regularization in the scene itself. Finally, we showed that the consistency of the scene flow projected into multiple viewpoints (optical flows) can be used to impose constraints on the 3D shape of the scene.

One obvious area for future work is the development of better scene flow algorithms, both algorithms that just compute scene flow, and algorithms than compute both scene flow and structure simultaneously. Another major area for future work is in developing applications. While we have used scene flow for "spatio-temporal view interpolation" [21], other possible applications include modeling the nonrigid deformations of clothes and around human joints. A final suggestion is to investigate the best way to arrange the cameras to obtain the best possible scene flow.

Acknowledgments

Elements of this work appeared previously in [22]. The authors thank the anonymous reviewers of the paper, as well as the *TPAMI* reviewers, for their useful comments and suggestions.

References

[1] E. Adelson and J. Bergen, "The Plenoptic Function and the Elements of Early Vision," *Computational Models of Visual Processing,* Landy and Movshon, eds., pp. 3-20, 1991.

[2] S. Avidan and A. Shashua, "Nonrigid Parallax for 3D Linear Motion," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 62-66, 1998.

[3] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l J. Computer Vision,* vol. 12, no. 1, pp. 43-77, 1994.

[4] M. Black and P. Anandan, "A Framework for the Robust Estimation of Optical Flow," *Proc. Fourth Int'l Conf. Computer Vision,* pp. 231-236, 1993.

[5] R. Carceroni and K. Kutulakos, "Multi-View Scene Capture by Surfel Sampling: From Video Streams to Nonrigid 3D Motion, Shape and Reflectance," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 60-67, 2001.

[6] J.P. Costeira and T. Kanade, "A Multibody Factorization Method for Independently Moving Objects," *Int'l J. Computer Vision,* vol. 29, no. 3, pp. 159-179, 1998.

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

[8] T. Kanade, H. Saito, and S. Vedula, "The 3D Room: Digitizing Time-Varying 3D Events by Synchronized Multiple Video Streams," Technical Report CMU-RI-TR-98-34, Robotics Inst., Carnegie Mellon Univ., 1998.

[9] 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,* vol. 9, pp. 166-178, 1997.

[10] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. Seventh Int'l Joint Conf. Artificial Intelligence,* pp. 674-679, 1981.

[11] S. Malassiotis and M.G. Strintzis, "Model-Based Joint Motion and Structure Estimation from Stereo Images," *Computer Vision Image Understanding,* vol. 65, no. 1, pp. 79-94, 1997.

[12] D. Metaxas and D. Terzopoulos, "Shape and Nonrigid Motion Estimation through Physics-Based Synthesis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 6, pp. 580-591, June 1993.

[13] S. Negahdaripour and B.K.P. Horn, "Direct Passive Navigation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, no. 1, pp. 168-176, Jan. 1987.

[14] M.A. Penna, "The Incremental Approximation of Nonrigid Motion," *Computer Vision, Graphics, and Image Processing,* vol. 60, no. 2, pp. 141-156, 1994.

[15] A.P. Pentland and B. Horowitz, "Recovery of Nonrigid Motion and Structure," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 7, pp. 730-742, July 1991.

[16] S.M. Seitz and C.R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *Int'l J. Computer Vision,* vol. 35, no. 2, pp. 151-173, 1999.

[17] K. Torrance and E. Sparrow, "Theory for Off-Specular Reflection from Rough Surfaces," *J. Optical Soc. of Am.,* vol. 57, pp. 1105-1114, Sept. 1967.

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

[19] S. Ullman, "Maximizing the Rigidity: The Incremental Recovery of 3-D Shape and Nonrigid Motion," *Perception,* vol. 13, pp. 255-274, 1984.

[20] S. Vedula, "Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events," PhD thesis, Robotics Inst., Carnegie Mellon Univ., 2001.

[21] S. Vedula, S. Baker, and T. Kanade, "Spatio-Temporal View Interpolation," *Proc. 13th ACM Eurographics Workshop Rendering,* pp. 65-76, June 2002.

[22] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-Dimensional Scene Flow," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 722-729, 1999.

[23] A.M. Waxman and J.H. Duncan, "Binocular Image Flows: Steps toward Stereo-Motion Fusion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 715-729, Nov. 1986.

[24] G.S. Young and R. Chellappa, "3-D Motion Estimation Using a Sequence of Noisy Stereo Images: Models, Estimation, and Uniqueness," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 8, pp. 735-759, Aug. 1999.