International Journal of Computer Vision 22(3), 199–218 (1997)

Spline-Based Image Registration

RICHARD SZELISKI
Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399
szeliski@microsoft.com

JAMES COUGHLAN
Harvard University, Department of Physics, Cambridge, MA 02138
coughlan@hrl.harvard.edu

Received February 8, 1995; Revised May 17, 1995; Accepted August 31, 1995

Abstract. The problem of image registration subsumes a number of problems and techniques in multiframe image analysis, including the computation of optic flow (general pixel-based motion), stereo correspondence, structure from motion, and feature tracking. We present a new registration algorithm based on *spline* representations of the displacement field which can be specialized to solve all of the above mentioned problems. In particular, we show how to compute local flow, global (parametric) flow, rigid flow resulting from camera egomotion, and multiframe versions of the above problems. Using a spline-based description of the flow removes the need for overlapping correlation windows, and produces an explicit measure of the correlation between adjacent flow estimates. We demonstrate our algorithm on multiframe image registration and the recovery of 3D projective scene geometry. We also provide results on a number of standard motion sequences.

Keywords: motion analysis, multiframe image analysis, hierarchical image registration, optical flow, splines, global motion models, structure from motion, direct motion estimation

1. Introduction

The analysis of image sequences (*motion analysis*) is one of the more actively studied areas of computer vision and image processing. The estimation of motion has many diverse applications, including video compression, the extraction of 3D scene geometry and camera motion, robot navigation, and the registration of multiple images. A common problem is to determine correspondences between various parts of images in a sequence. This problem is often called *motion estimation*, *multiple view analysis*, or *image registration.*

Motion analysis subsumes a number of subproblems and associated solution techniques, including optic flow, stereo and multiframe stereo, egomotion estimation, and feature detection and tracking. Each of these approaches makes different assumptions about the nature of the scene and the results to be computed (computational theory and representation) and the techniques used to compute these results (algorithm).

In this paper, we present a general motion estimation framework which can be specialized to solve a number of these sub-problems. Like Bergen et al. (1992), we view motion estimation as an image registration task with a fixed computational theory (optimality criterion), and view each sub-problem as an instantiation of a particular global or local motion model. For example, the motion may be completely general, it can depend on a few global parameters (e.g., affine flow), or it can result from the rigid motion of a 3D scene. We also use coarse-to-fine (*hierarchical*) algorithms to handle large displacements.

The key difference between our framework and previous algorithms is that we represent the local motion

field using multi-resolution splines. This has a number of advantages over previous approaches. The splines impose an implicit smoothness on the motion field, removing in many instances the need for additional smoothness constraints (regularization). The splines also remove the need for correlation windows centered at each pixel, which are computationally expensive and implicitly assume a local translational model. Furthermore, they provide an explicit measure of the correlation between adjacent motion estimates.

The algorithm we develop to estimate structure and motion from rigid scenes differs from previous algorithms by using a general camera model, which eliminates the need to know the intrinsic camera calibration parameters. This results in estimates of *projective depth* rather than true Euclidean depth; the conversion to Euclidean shape, if required, can be performed in a later post-processing stage.

The remainder of the paper is structured as follows. Section 2 presents a review of relevant previous work. Section 3 gives the general problem formulations for image registration. Section 4 develops our algorithm for local motion estimation. Section 5 presents the algorithm for global (planar) motion estimation. Section 6 presents our novel formulation of structure from motion based on the recovery of projective depth. Section 7 generalizes our previous algorithms to multiple frames and examines the resulting performance improvements. Section 8 presents experimental results based on some commonly used motion test sequences. Finally, we close with a comparison of our approach to previous algorithms and a discussion of future work.

2. Previous Work

A large number of motion estimation and image registration algorithms have been developed in the past (Brown, 1992). These algorithms include optical flow (general motion) estimators, global parametric motion estimators, constrained motion estimators (*direct methods*), stereo and multiframe stereo, hierarchical (coarse-to-fine) methods, feature trackers, and feature-based registration techniques. We will use this rough taxonomy to briefly review previous work, while recognizing that these algorithms overlap and that many algorithms use ideas from several of these categories.

The general motion estimation problem is often called *optical flow* recovery (Horn and Schunck, 1981). This involves estimating an independent displacement vector for each pixel in an image. Approaches to this problem include gradient-based approaches based on the *brightness constraint* (Horn and Schunck, 1981; Lucas and Kanade, 1981; Nagel, 1987), correlation-based techniques such as the Sum of Squared Differences (SSD) (Anandan, 1989), spatio-temporal filtering (Adelson and Bergen, 1985; Heeger, 1987; Fleet and Jepson, 1990), and regularization (Horn and Schunck, 1981; Hildreth, 1986, Poggio et al., 1985). Nagel (1987) and Anandan (1989) provide comparisons and derive relations between different techniques, while Barron et al. (1994) provide some numerical comparisons.

Global motion estimators (Lucas, 1984; Bergen et al., 1992) use a simple flow field model parameterized by a small number of unknown variables. Examples of global motion models include affine and quadratic flow fields. In the taxonomy of Bergen et al. (1992), these fields are called parametric motion models, since they can be used locally as well (e.g., you can estimate affine flow at every pixel from filter outputs (Manmatha and Oliensis, 1992))[1]. Global methods are most useful when the scene has a particularly simple form, e.g., when the scene is planar.

Constrained (*quasi-parametric* (Bergen et al., 1992)) motion models fall between local and global methods. Typically, these use a combination of global egomotion parameters with local shape (depth) parameters. Examples of this approach include the *direct methods* of Horn and Weldon (1988) and others (Hanna, 1991; Bergen et al., 1992). In this paper, we use projective descriptions of motion and depth (Faugeras, 1992; Mohr et al., 1993; Szeliski and Kang, 1994) for our constrained motion model, which removes the need for calibrated cameras.

Stereo matching (Barnard and Fischler, 1982; Quam, 1984; Dhond and Aggarwal, 1989) is traditionally considered as a separate sub-discipline within computer vision (and, of course, photogrammetry), but there are strong connections between the two problems. Stereo can be viewed as a simplified version of the constrained motion model where the egomotion parameters (the *epipolar geometry*) are given, so that each flow vector is constrained to lie along a known line. While stereo is traditionally performed on pairs of images, more recent algorithms use sequences of images (*multiframe stereo* or *motion stereo*) (Bolles et al., 1987; Matthies et al., 1989; Okutomi and Kanade, 1993).

Hierarchical (coarse-to-fine) matching algorithms have a long history of use both in stereo matching

(Quam, 1984; Witkin et al., 1987) and in motion estimation (Enkelmann, 1988; Anandan, 1989; Singh, 1990; Bergen et al., 1992). Hierarchical algorithms first solve the matching problem on smaller, lower-resolution images and then use these to initialize higher-resolution estimates. Their advantages include both increased computation efficiency and the ability to find better solutions (escape from local minima).

Tracking individual features (corners, points, lines) in images has always been an alternative to iconic (pixel-based) optic flow techniques (Dreschler and Nagel, 1982; Sethi and Jain, 1987; Zheng and Chellappa, 1992). This has the advantage of requiring less computation and of being less sensitive to lighting variation. The algorithm presented in this paper is closely related to patch-based feature trackers (Lucas and Kanade, 1981; Rehg and Witkin, 1991; Tomasi and Kanade, 1992). In fact, our general motion estimator can be used as a parallel, adaptive feature tracker by selecting spline control vertices with low uncertainty in both motion components (Szeliski et al., 1995). Like Rehg and Witkin (1991), which is an affine-patch based tracker, it can handle large deformations in the patches being tracked.

Spline-based image registration techniques have been used in both the image processing and computer graphics communities. The work in Goshtasby (1986, 1988) applies surface fitting to discrete displacement estimates based on feature correspondences to obtain a smooth displacement field. Wolberg (1990) provides a review of the extensive literature in digital image warping, which can be used to resample images once the (usually global) displacements are known. Spline-based displacement fields have recently been used in computer graphics to perform *morphing* (Beier and Neely, 1992) (deformable image blending) using manually specified correspondences (see (Beymer et al., 1993) for a combination of motion estimation and morphing). Registration techniques based on elastic deformations of images (Burr, 1981; Bajcsy and Broit, 1982; Bajcsy and Kovacic, 1989; Amit, 1993) also sometimes use splines as their representation (Bajcsy and Broit, 1982).

3. General Problem Formulation

The general image registration problem can be formulated as follows. We are given a sequence of images $I_t(x, y)$ which we assume were formed by locally displacing a reference image $I(x, y)$ with horizontal and vertical displacement fields[2] $u_t(x, y)$ and $v_t(x, y)$, i.e.,

$$I_t(x + u_t, y + v_t) = I(x, y). \tag{1}$$

Each individual image is assumed to be corrupted with uniform white Gaussian noise. We also ignore possible occlusions ("foldovers") in the warped images.

Given such a sequence of images, we wish to simultaneously recover the displacement fields u_t and v_t and the reference image $I(x, y)$. The maximum likelihood solution to this problem is well known (Szeliski, 1989), and consists of minimizing the squared error

$$\sum_t \iint [I_t(x+u_t, y+v_t) - I(x, y)]^2 dx\, dy. \tag{2}$$

In practice, we are usually given a set of discretely sampled images, so we replace the above integrals with summations over the set of pixels $\{(x_i, y_i)\}$.

If the displacement fields u_t and v_t at different times are independent of each other and the reference intensity image $I(x, y)$ is assumed to be known, the above minimization problem decomposes into a set of independent minimizations, one for each frame. For now, we will assume that this is the case (returning to the problem of multiframe motion in Section 7), and only study the two frame problem, which can be rewritten as

$$E(\{u_i, v_i\}) = \sum_i [I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)]^2. \tag{3}$$

This equation is called the *Sum of Squared Differences* (SSD) formula (Anandan, 1989). Expanding I_1 in a first order Taylor series expansion in (u_i, v_i) yields the the *image brightness constraint* (Horn and Schunck, 1981; Anandan, 1989).

The above minimization problem typically has many local minima. Several techniques are commonly used to find a more globally optimal estimate. For example, the SSD algorithm performs the summation at each pixel over an $m \times m$ window (typically 5×5) (Anandan, 1989). More recent variations use adaptive windows (Okutomi and Kanade, 1992) and multiple frames (Okutomi and Kanade, 1993)). Regularization-based algorithms add smoothness constraints on the u and v fields to obtain good solutions (Horn and Schunck, 1981; Hildreth, 1986; Poggio et al., 1985). Multiscale or hierarchical (coarse-to-fine) techniques are often used to speed the search for the optimum displacement field.

Another decision that must be made is how to represent the (u, v) fields. Assigning an independent estimate at each pixel (u_i, v_i) is the most commonly made choice, but global motion descriptors are also possible (Lucas, 1984; Bergen et al., 1992) (see also Section 5). Constrained motion models which combine a global rigid motion description with a local depth estimate are also used (Horn and Weldon, 1988; Hanna, 1991; Bergen et al., 1992), and we will study these in Section 6.

Both local correlation windows (as in SSD) and global smoothness constraints attempt to disambiguate possible motion field estimates by aggregating information from neighboring pixels. The resulting displacement estimates are therefore highly correlated. While it is possible to analyze the correlations induced by overlapping windows (Matthies et al., 1989) and regularization (Szeliski, 1989), the procedures are cumbersome and rarely used.

3.1. Spline-Based Motion Model

The alternative to these approaches, which we introduce in this paper, is to represent the displacements fields $u(x, y)$ and $v(x, y)$ as two-dimensional *splines* controlled by a smaller number of displacement estimates $\hat{u}_j$ and $\hat{v}_j$ which lie on a coarser *spline control grid* (Fig. 1). The value for the displacement at a pixel i can be written as

$$u_i = u(x_i, y_i) = \sum_j \hat{u}_j B_j(x_i, y_i) = \sum_j \hat{u}_j w_{ij}, \quad (4)$$

where the $B_j(x, y)$ are called the *basis functions* and are only non-zero over a small interval (*finite support*).

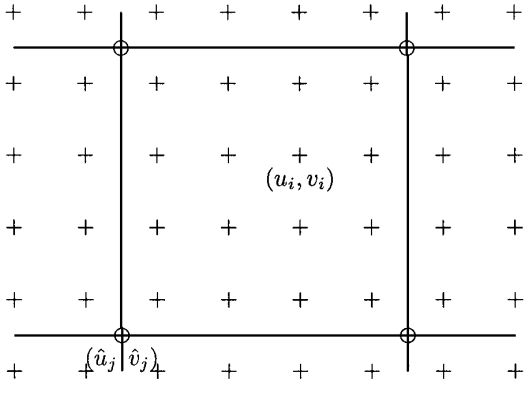


Figure 1. Displacement spline: the spline control vertices $\{(\hat{u}_j, \hat{v}_j)\}$ are shown as circles (○) and the pixel displacements $\{(u_i, v_i)\}$ are shown as pluses (+).

We call the $w_{ij} = B_j(x_i, y_i)$ *weights* to emphasize that the (u_i, v_i) are known linear combinations of the $(\hat{u}_j, \hat{v}_j)$[3].

In our current implementation, we make the spline control grid a regular subsampling of the pixel grid, $\hat{x}_j = mx_i$, $\hat{y}_j = my_i$, so that each set of $m \times m$ pixels corresponds to a single spline patch. The basis functions are spatially shifted versions of each other, i.e., $B_j(x, y) = B(\frac{x-\hat{x}_j}{m}, \frac{y-\hat{y}_j}{m})$. We have implemented five different interpolation functions:

1. block: $B(x, y) = 1$ on $[0, 1]^2$
2. linear: $B(x, y) = \begin{cases} (1 - x - y) \text{ on } [0, 1]^2, \\ (x + 1) \text{ on } [-1, 0] \times [0, 1], \\ (y + 1) \text{ on } [0, 1] \times [-1, 0] \end{cases}$
3. linear on sub-triangles: $B(x, y) = \max(0, 1 - \max(|x|, |y|, |x + y|))$
4. bilinear: $B(x, y) = (1 - |x|)(1 - |y|)$ on $[-1, 1]^2$
5. biquadratic: $B(x, y) = B_2(x)B_2(y)$ on $[-1, 2]^2$, where $B_2(x)$ is the quadratic B-spline

Figure 2 shows 3D graphs of the basis functions for the five splines.

How do spline representations compare to local correlation windows and to regularization? This question has previously been studied in the context of active deformable contours (*snakes*). The original work on snakes was based on a regularization framework (Kass et al., 1988), giving the snake the ability to model arbitrarily detailed bends or discontinuities where warranted by the data. More recent versions of snakes often employ the *B-snake* (Menet et al., 1990; Blake et al., 1993) which has fewer control vertices. A spline-based snake has fewer degrees of freedom, and thus may be easier to recover. The smooth interpolation function between vertices plays a similar role to regularization, although the smoothing introduced is not as uniform or stationary.

In our use of splines for modeling displacement fields, we have a similar tradeoff (see Section 9 for more discussion). We often may not need regularization (e.g., in highly textured scenes). Where required, adding a regularization term to the cost function (3) is straightforward, i.e., we can use a first order regularizer (Poggio et al., 1985)

$$\begin{aligned} E_1(\{\hat{u}_{k,l}, \hat{v}_{k,l}\}) \\ = \sum_{k,l} (\hat{u}_{k,l} - \hat{u}_{k-1,l})^2 \\ + (\hat{u}_{k,l} - \hat{u}_{k,l-1})^2 + \cdots \text{ terms in } \hat{v}_{kl} \end{aligned} \quad (5)$$

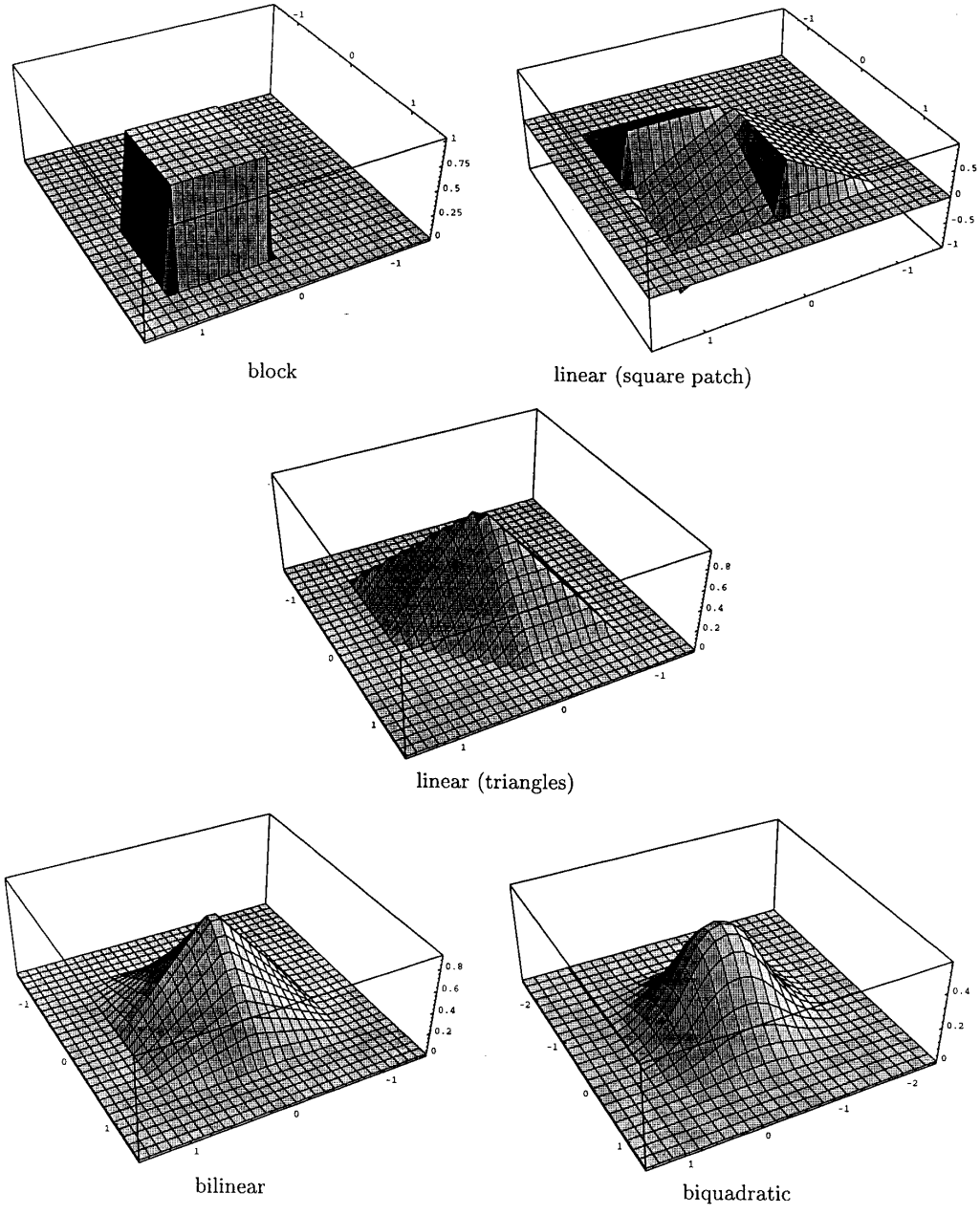


Figure 2. Spline basis functions.

or a second order regularizer (Terzopoulos, 1986)

$$
\begin{aligned}
E_2(\{\hat{u}_{k,l}, \hat{v}_{k,l}\}) &= h^{-2} \sum_{k,l} (\hat{u}_{k+1,l} - 2\hat{u}_{k,l} + \hat{u}_{k-1,l})^2 \\
&\quad + (\hat{u}_{k,l+1} - 2\hat{u}_{k,l} + \hat{u}_{k,l-1})^2 \\
&\quad + (\hat{u}_{k,l} - \hat{u}_{k,l-1} - \hat{u}_{k-1,l} + \hat{u}_{k-1,l-1})^2 \\
&\quad + \cdots \text{ terms in } \hat{v}_{kl} \qquad (6)
\end{aligned}
$$

where h is the patch size, and we index the spline control vertices with 2D indices (k, l).

Spline-based flow descriptors also remove the need for overlapping correlation windows, since each flow estimate $(\hat{u}_j, \hat{v}_j)$ is based on weighted contributions from all of the pixels beneath the support of its basis function (e.g, $(2m) \times (2m)$ pixels for a bilinear basis). As we will show in Section 4, the spline-based flow

formulation makes it straightforward to compute the uncertainty (covariance matrix) associated with the complete flow field. It also corresponds naturally to the optimal Bayesian estimator for the flow. The squared pixel errors correspond to Gaussian noise, and the spline model (and any associated regularizers) form the prior model (Szeliski, 1989). Correlation-based techniques with overlapping windows, on the other hand, do not have a similar direct connection to Bayesian estimators.

Before moving on to our different motion models and solution techniques, we should point out that the squared pixel error function (3) can be generalized to account for photometric variation (global brightness and contrast changes). Following (Lucas, 1984; Gennert, 1988; Fuh and Maragos, 1991), we can write

$$E'(\{u_i, v_i\}) = \sum_i [I_1(x_i + u_i, y_i + v_i) - cI_0(x_i, y_i) + b]^2, \quad (7)$$

where b and c are the (per-frame) brightness and contrast correction terms. Both of these parameters can be estimated concurrently with the flow field at little additional cost. Their inclusion is most useful in situations where the photometry can change between successive views (e.g., when the images are not acquired concurrently). Furthermore, the matching need not be applied directly to the raw intensity images. Both linear (e.g., low-pass or band-pass filtering (Burt and Adelson, 1983)) and non-linear pre-processing can be performed.

4. Local (General) Flow Estimation

To recover the local spline-based flow parameters, we need to minimize the cost function (3) with respect to the $\{\hat{u}_j, \hat{v}_j\}$. We do this using a variant of the Levenberg-Marquardt iterative non-linear minimization technique (Press et al., 1992). First, we compute the gradient of E in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$,

$$g_j^u \equiv \frac{\partial E}{\partial \hat{u}_j} = 2\sum_i e_i G_i^x w_{ij}$$
$$g_j^v \equiv \frac{\partial E}{\partial \hat{v}_j} = 2\sum_i e_i G_i^y w_{ij}, \quad (8)$$

where

$$e_i = I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i) \quad (9)$$

is the intensity error at pixel i,

$$\left(G_i^x, G_i^y\right) = \nabla I_1(x_i + u_i, y_i + v_i) \quad (10)$$

is the intensity gradient of I_1 at the displaced position for pixel i, and the w_{ij} are the sampled values of the spline basis function (4). Algorithmically, we compute the above gradients by first forming the displacement vector for each pixel (u_i, v_i) using (4), then computing the resampled intensity and gradient values of I_1 at $(x_i', y_i') = (x_i + u_i, y_i + v_i)$, computing e_i, and finally incrementing the g_j^u and g_j^v values of all control vertices affecting that pixel.

For the Levenberg-Marquardt algorithm, we also require the approximate Hessian matrix $\mathbf{A}$ where the second-derivative terms are left out[4]. The matrix $\mathbf{A}$ contains entries of the form

$$a_{jk}^{uu} = 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{u}_k} = 2\sum_i w_{ij} w_{ik} \left(G_i^x\right)^2$$
$$a_{jk}^{uv} = 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{v}_k} = 2\sum_i w_{ij} w_{ik} G_i^x G_i^y \quad (11)$$
$$a_{jk}^{vv} = 2\sum_i \frac{\partial e_i}{\partial \hat{v}_j}\frac{\partial e_i}{\partial \hat{v}_k} = 2\sum_i w_{ij} w_{ik} \left(G_i^y\right)^2.$$

The entries of $\mathbf{A}$ can be computed at the same time as the energy gradients.

What is the structure of the approximate Hessian matrix? The 2×2 sub-matrix $\mathbf{A}_{jj}$ corresponding to the terms a_{jj}^{uu}, a_{jj}^{uv}, and a_{jj}^{vv} encodes the local shape of the sum-of-squared difference correlation surface (Lucas, 1984; Anandan, 1989). This matrix is often used to compute an updated flow vector by setting

$$[\Delta\hat{u}_j \;\; \Delta\hat{v}_j]^T = -\mathbf{A}_{jj}^{-1}\left[g_j^u \;\; g_j^v\right]^T \quad (12)$$

(Lucas, 1984; Anandan, 1989; Bergen et al., 1992). The overall $\mathbf{A}$ matrix is a sparse multi-banded block-diagonal matrix, i.e., sub-blocks $\mathbf{A}_{jk}$ will be non-zero only if vertices j and k both influence some common patch of pixels.

The Levenberg-Marquardt algorithm proceeds by computing an increment $\Delta\mathbf{u}$ to the current displacement estimate $\mathbf{u}$ which satisfies

$$(\mathbf{A} + \lambda \,\mathrm{diag}(\mathbf{A}))\Delta\mathbf{u} = -\mathbf{g}, \quad (13)$$

where $\mathbf{u}$ is the vector of concatenated displacement estimates $\{\hat{u}_j, \hat{v}_j\}$, $\mathbf{g}$ is the vector of concatenated energy gradients $\{g_j^u, g_j^v\}$, and λ is a stabilization factor which varies over time (Press et al., 1992). In more detail, the value of λ is initialized to a small constant, say

$\lambda = 0.001$, and then incremented or decremented by a factor of 10 depending on whether the energy after the descent step increases or decreases. This allows the Levenberg-Marquardt algorithm to vary smoothly between the extremes of an inverse-Hessian and a steepest descent method.

For systems with small numbers of parameters, e.g., if only a single spline patch is being used (Section 5), the complete system of equations (13) can be solved at reasonable computational cost. However, for general flow computation, there may be thousands of spline control variables (e.g., for a 640 × 480 image with $m = 8$, we have $81 \times 61 \times 2 \approx 10^4$ parameters). In this case, iterative sparse matrix techniques have to be used to solve the above system of equations, in order to prevent excessive *fill in*.

In our current implementation, we use preconditioned gradient descent to update our flow estimates

$$\Delta \mathbf{u} = -\alpha\, \mathbf{B}^{-1}\mathbf{g} = -\alpha \mathbf{d} \tag{14}$$

where $\mathbf{B} = \hat{\mathbf{A}} + \lambda\, \mathrm{diag}(\mathbf{A})$, and $\hat{\mathbf{A}} = \mathrm{block\_diag}(\mathbf{A})$ is the set of 2×2 block diagonal matrices used in (12). In this simplest version, the update rule is very close to that used by Lucas (1984) and others, with the following differences:

1. the equations for computing the $\mathbf{g}$ and $\mathbf{A}$ are different (based on spline interpolation)
2. an additional diagonal term λ is added for stability[5]
3. there is a step size α.

The step size α is necessary because we are ignoring the off-block-diagonal terms in $\mathbf{A}$, which can be quite significant. An optimal value for α can be computed at each iteration by minimizing

$$\Delta E(\alpha \mathbf{d}) \approx \alpha^2 \mathbf{d}^T \mathbf{A}\mathbf{d} - 2\alpha \mathbf{d}^T \mathbf{g},$$

i.e., by setting $\alpha = (\mathbf{d} \cdot \mathbf{g})/(\mathbf{d}^T \mathbf{A}\mathbf{d})$. The denominator can be computed without explicitly computing $\mathbf{A}$ by noting that

$$\mathbf{d}^T \mathbf{A}\mathbf{d} = \sum_i \left(G_i^x \delta u_i + G_i^y \delta v_i\right)^2$$

where

$$\delta u_i = \sum_j w_{ij} \delta \hat{u}_j, \quad \delta v_i = \sum_j w_{ij} \delta \hat{v}_j,$$

and the $(\delta \hat{u}_j, \delta \hat{v}_j)$ are the components of $\mathbf{d}$.

To handle larger displacements, we run our algorithm in a coarse-to-fine (hierarchical) fashion. A Gaussian image pyramid is first computed using an iterated $\boxed{1}\boxed{2}\boxed{1}$ filter (Burt and Adelson, 1983). We then run the algorithm on one of the smaller pyramid levels, and use the resulting flow estimates to initialize the next finer level (using bilinear interpolation and doubling the displacement magnitudes).

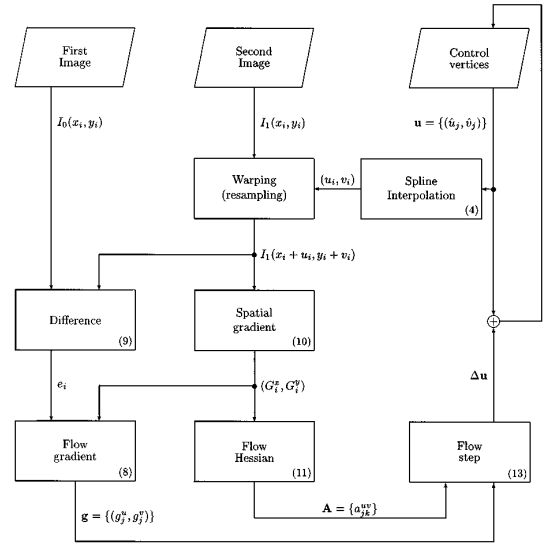


Figure 3. Block diagram of spline-based image registration. The numbers in the lower right corner of each processing box refer to the associated equation numbers in the paper.

Figure 3 shows a block diagram of the processing stages involved in our spline-based image registration algorithm.

Figure 4 shows an example of the flow estimates produced by our technique. The input image is 256 × 240 pixels, and we use 16 × 16 pixel spline patches, with 3 levels in the pyramid and 9 iterations at each level. The middle frame shows the distorted spline control grid, while the right frame shows the estimated flow vectors, which have been sampled on a 30 × 28 grid (every 8 pixels). The flow estimates are very good in the textured areas corresponding to the Rubik cube, the stationary boxes, and the turntable edges. Flow vectors in the uniform intensity areas (e.g., table and turntable tops) are fairly arbitrary. This example uses no regularization beyond that imposed by the spline patches, nor does it threshold flow vectors according to certainty. For a more detailed analysis, see Section 8.

5. Global (Planar) Flow Estimation

In many applications, e.g., in the registration of pieces of a flat scene, when the distance between the camera

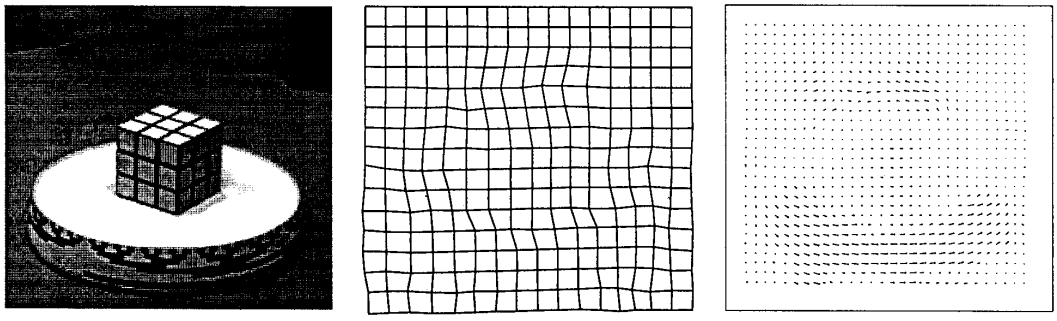


Figure 4. Example of general flow computation: first image, displaced spline control grid, estimated flow vectors.

and the scene is large (Bergen et al., 1992), or when performing a coarse registration of slices in a volumetric data set (Carlbom et al., 1991), a single global description of the motion model may suffice. A simple example of such a global motion is an affine flow (Koenderink and Doorn, 1991; Rehg and Witkin, 1991; Bergen et al., 1992)

$$\begin{aligned} u(x, y) &= (m_0 x + m_1 y + m_2) - x \\ v(x, y) &= (m_3 x + m_4 y + m_5) - y. \end{aligned} \tag{15}$$

The parameters $\mathbf{m} = (m_0, \ldots, m_5)^T$ are called the *global motion parameters*. Models with fewer degrees of freedom such as pure translation, translation and rotation, or translation plus rotation and scale (*similarity transform*) are also possible, but they will not be studied in this paper.

To compute the global motion estimate, we take a two step approach. First, we define the spline control vertices $\hat{\mathbf{u}}_j = (\hat{u}_j, \hat{v}_j)^T$ in terms of the global motion parameters

$$\begin{aligned} \hat{\mathbf{u}}_j &= \begin{bmatrix} \hat{x}_j & \hat{y}_j & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{x}_j & \hat{y}_j & 1 \end{bmatrix} \mathbf{m} - \begin{bmatrix} \hat{x}_j \\ \hat{y}_j \end{bmatrix} \\ &\equiv \mathbf{T}_j \mathbf{m} - \hat{\mathbf{x}}_j. \end{aligned} \tag{16}$$

Second, we define the flow at each pixel using our usual spline interpolation. Note that for affine (or simpler) flow, this gives the correct flow at each pixel if linear or bilinear interpolants are used[6]. For affine (or simpler) flow, it is therefore possible to use only a single spline patch[7].

Why use this two-step procedure then? First, this approach will work better when we generalize our motion model to 2D projective transformations (see below). Second, there are computational savings in only storing the w_{ij} for smaller patches. Lastly, we can obtain a better estimate of the Hessian at $\mathbf{m}$ at lower computational cost, as we discuss below.

To apply Levenberg-Marquardt as before, we need to compute both the gradient of the cost function with respect to $\mathbf{m}$ and the Hessian. Computing the gradient is straightforward

$$\mathbf{g}_\mathrm{m} \equiv \frac{\partial E}{\partial \mathbf{m}} = \sum_j \frac{\partial \hat{\mathbf{u}}_j}{\partial \mathbf{m}} \frac{\partial E}{\partial \hat{\mathbf{u}}_j} = \sum_j \mathbf{T}_j^T \mathbf{g}_j, \tag{17}$$

where $\mathbf{g}_j = (g_j^u, g_j^v)^T$. The Hessian matrix can be computed in a similar fashion

$$\mathbf{A}_\mathrm{m} = \frac{\partial^2 E}{\partial \mathbf{m}^T \partial \mathbf{m}} \approx \sum_{jk} \mathbf{T}_j^T \mathbf{A}_{jk} \mathbf{T}_k, \tag{18}$$

where the $\mathbf{A}_{jk}$ are the 2×2 submatrices of $\mathbf{A}$.

We can approximate the Hessian matrix even further by neglecting the off-diagonal $\mathbf{A}_{jk}$ matrices. This is equivalent to modeling the flow estimate at each control vertex as being independent of other vertex estimates. When the spline patches are sufficiently large and contain sufficient texture, this turns out to be a reasonable approximation.

To compute the optimal step size α, we let

$$\mathbf{d} = \mathbf{T}\mathbf{d}_\mathrm{m}, \tag{19}$$

where $\mathbf{T}$ is the concatenation of all the $\mathbf{T}_j$ matrices, and then set $\alpha = (\mathbf{d} \cdot \mathbf{g})/(\mathbf{d}^T \mathbf{A} \mathbf{d})$ as before.

Figure 5 shows a block diagram of the processing stages involved in the global motion estimation algorithms presented in this and the next section.

Examples of our global affine flow estimator applied to two different motion sequences can be seen in Figs. 6 and 7. These two sequences were generated

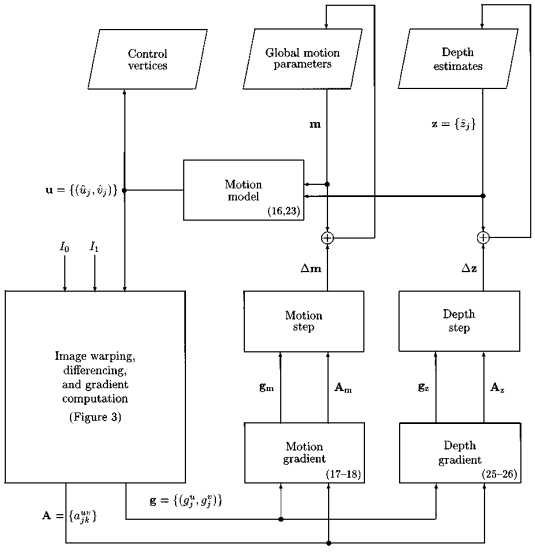


Figure 5. Block diagram of global motion estimation. The numbers in the lower right corner of each processing box refer to the associated equation numbers in the paper.

synthetically from a real base image, with Fig. 6 being pure translational motion, and Fig. 7 being divergent motion (Barron et al., 1994). As expected, the motion is recovered extremely well in this case (see Section 8 for quantitative results).

A more interesting case, in general, is that of a planar surface in motion viewed through a pinhole camera. This motion can be described as a 2D projective transformation of the plane

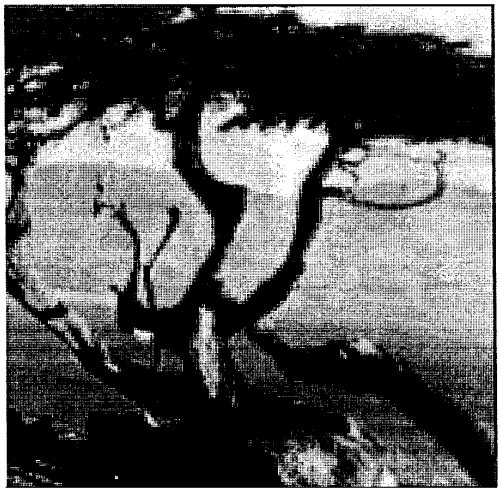
$$\begin{aligned} u(x,y) &= \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} - x \\ v(x,y) &= \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} - y. \end{aligned} \tag{20}$$

Our projective formulation[8] requires 8 parameters per frame, which is the same number as the quadratic flow field used in Bergen et al. (1992). However, our formulation allows for arbitrarily large displacements, whereas Bergen et al. (1992) is based on instantaneous (infinitesimal) motion. Our formulation also does not require the camera to be calibrated, and allows the internal camera parameters (e.g., zoom) to vary over time. The price we pay is that the motion field is no longer a linear function of the global motion parameters. We have been using our projective motion estimator to create large panoramic photo-mosaics for virtual reality applications (Szeliski, 1996).

To compute the gradient and the Hessian, we proceed as before. We use the equations

$$\begin{aligned} \hat{u}_j &= \frac{m_0 \hat{x}_j + m_1 \hat{y}_j + m_2}{m_6 \hat{x}_j + m_7 \hat{y}_j + 1} - \hat{x}_j \\ \hat{v}_j &= \frac{m_3 \hat{x}_j + m_4 \hat{y}_j + m_5}{m_6 \hat{x}_j + m_7 \hat{y}_j + 1} - \hat{y}_j \end{aligned} \tag{21}$$

to compute the spline control vertices, and use the bilinear interpolants to compute the flow at each pixel.



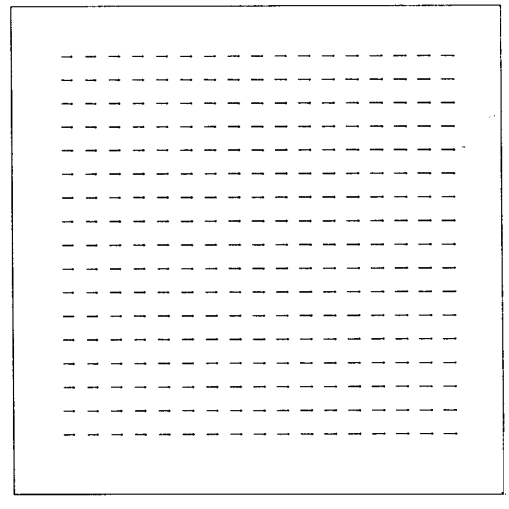


Figure 6. Example of affine flow computation: translation.

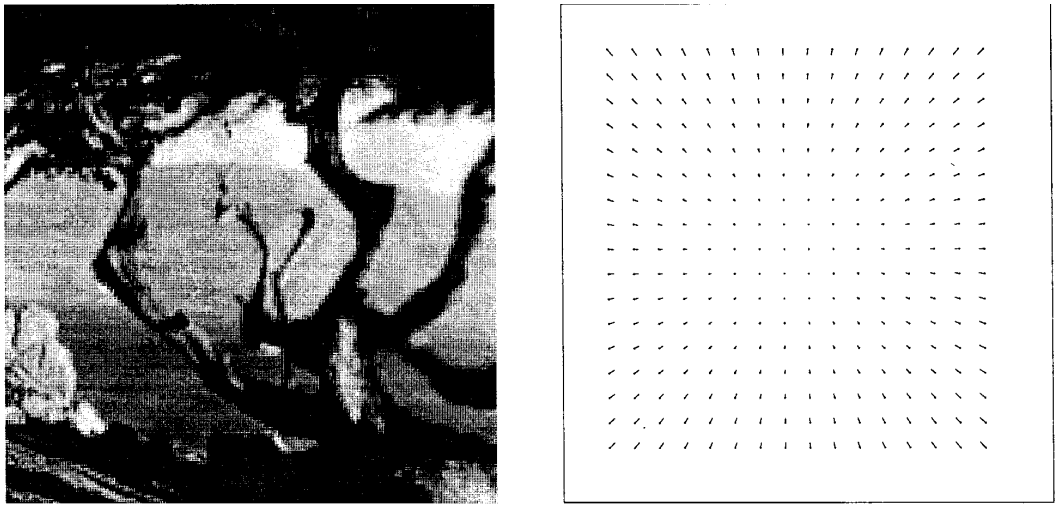


Figure 7. Example of affine flow computation: divergence (zoom).

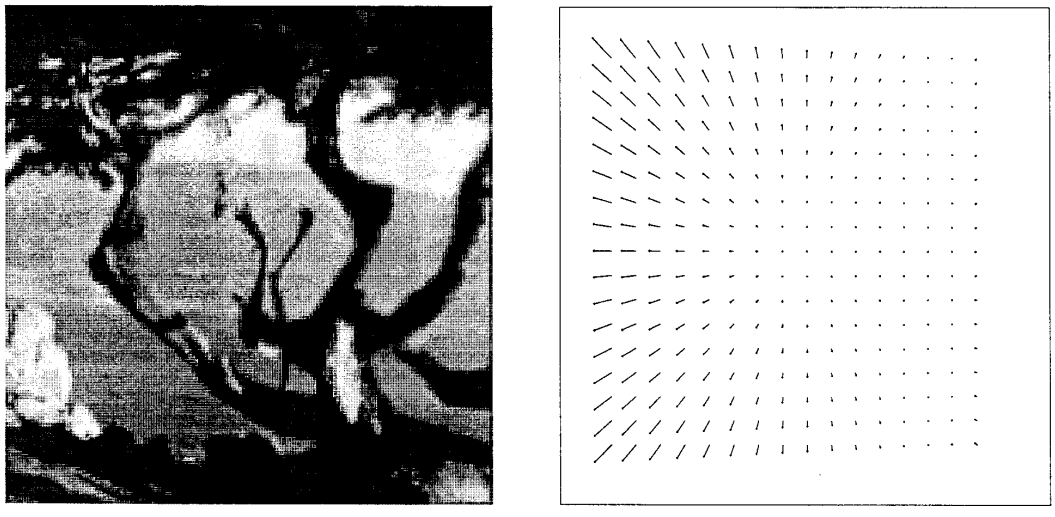


Figure 8. Example of 2D projective motion estimation.

This flow is not exactly equivalent to the true projective flow defined in Eq. (20), since the latter involves a division at each pixel. However, the error will be small if the patches are small and/or the perspective distortion (m_6, m_7) is small.

To compute the gradient, we note that

$$\delta\hat{\mathbf{u}}_j = \mathbf{T}_j \delta\mathbf{m} \tag{22}$$

with

$$\mathbf{T}_j = \frac{1}{D_j}\begin{bmatrix} \hat{x}_j & \hat{y}_j & 1 & 0 & 0 & 0 & -\hat{x}_j\frac{N_j^u}{D_j} & -\hat{y}_j\frac{N_j^u}{D_j} \\ 0 & 0 & 0 & \hat{x}_j & \hat{y}_j & 1 & -\hat{x}_j\frac{N_j^v}{D_j} & -\hat{y}_j\frac{N_j^v}{D_j} \end{bmatrix},$$

where $N_j^u = m_0\hat{x}_j + m_1\hat{y}_j + m_2$ and $N_j^v = m_3\hat{x}_j + m_4\hat{y}_j + m_5$ are the current numerators of (21), $D_j = m_6\hat{x}_j + m_7\hat{y}_j + 1$ is the current denominator, and $\mathbf{m} = (m_0, \ldots, m_7)^T$. With this modification to the affine case, we can proceed as before, applying (17)–(19) to compute the global gradient, Hessian, and the step size. Thus, we see that even though the problem is no longer linear, the modifications involve simply using an extra division per spline control vertex. Figure 8 shows an image which has been perspectively distorted and the accompanying recovered flow field. The motion is that of a plane rotating around its y axis and moving slightly forward, viewed with a wide angle lens.

6. Mixed Global and Local (Rigid) Flow Estimation

A special case of optic flow computation which occurs often in practice is that of *rigid motion*, i.e., when the camera moves through a static scene, or a single object moves rigidly in front of a camera. Commonly used techniques (*direct methods*) based on estimating the instantaneous camera egomotion $(\mathbf{R}(\omega), \mathbf{t})$ and a camera-centered depth $Z(x, y)$ are given in Horn and Weldon (1988), Hanna (1991), Bergen et al. (1992). This has the disadvantage of only being valid for small motions, of requiring a calibrated camera, and of sensitivity problems with the depth estimates[9].

Our approach is based on a *projective* formulation of structure from motion (Hartley et al., 1992; Faugeras, 1992; Mohr et al., 1993; Szeliski and Kang, 1994)

$$\begin{aligned} u(x, y) &= \frac{m_0 x + m_1 y + m_8\, z(x, y) + m_2}{m_6 x + m_7 y + m_{10} z(x, y) + 1} - x \\ v(x, y) &= \frac{m_3 x + m_4 y + m_9\, z(x, y) + m_5}{m_6 x + m_7 y + m_{10} z(x, y) + 1} - y. \end{aligned} \tag{23}$$

This formulation is valid for any pinhole camera model, even with time varying internal camera parameters. The local shape estimates $z(x, y)$ are *projective depth* estimates, i.e., the $(x, y, z, 1)$ coordinates are related to the true Euclidean coordinates $(X, Y, Z, 1)$ through some 3-D projective transformation (*collineation*) which can, given enough views, be recovered from the projective motion estimates.

Compared to the usual rigid motion formulation, we have to estimate more global parameters (11 instead of 6) for the global motion, so we might be concerned with an increased uncertainty in these parameters. However, we do not require our camera to be calibrated or to have fixed internal parameters. We can also deal with arbitrarily large displacements and non-smooth motion. Furthermore, situations in which either the global motion or local shape estimates are poorly recovered (e.g., planar scenes, pure rotation) do not cause any problems for our technique.

A special case of the rigid motion problem is when we are given a pair of calibrated images (*stereo*) or a set of *weakly calibrated* images (only the epipoles in the images are known (Hartley et al., 1992; Faugeras, 1992)). In either case, we can compute the m_l, resulting in a simple estimation problem in $z(x, y)$. We can then either proceed to resample (*rectify*) the images so that corresponding pixels lie along scanlines (Hartley and Gupta, 1993), or we can directly work with the original images, proceeding as below but omitting the $\mathbf{m}$ update step.

To compute the global and local flow estimates, we combine several of the approaches developed previously in the paper. First, we compute the 2D flows at the control vertices by evaluating Eq. (23) at the vertex locations $\{(\hat{x}_j, \hat{y}_j)\}$. We compute the gradients and Hessian with respect to the global motion parameters as before, with

$$\mathbf{T}_j = \frac{1}{D_j}\begin{bmatrix} \hat{x}_j & \hat{y}_j & 1 & 0 & 0 & 0 & -\hat{x}_j \frac{N_j^u}{D_j} \\ 0 & 0 & 0 & \hat{x}_j & \hat{y}_j & 1 & -\hat{x}_j \frac{N_j^v}{D_j} \\ & & & & & & -\hat{y}_j \frac{N_j^u}{D_j} \quad \hat{z}_j \quad 0 \quad -\hat{z}_j \frac{N_j^u}{D_j} \\ & & & & & & -\hat{y}_j \frac{N_j^v}{D_j} \quad 0 \quad \hat{z}_j \quad -\hat{z}_j \frac{N_j^v}{D_j} \end{bmatrix}, \tag{24}$$

$N_j^u = m_0 \hat{x}_j + m_1 \hat{y}_j + m_8 \hat{z}_j + m_2$, $N_j^v = m_3 \hat{x}_j + m_4 \hat{y}_j + m_9 \hat{z}_j + m_5$, and $D_j = m_6 \hat{x}_j + m_7 \hat{y}_j + m_{10} \hat{z}_j + 1$. The derivatives with respect to the depth estimates $\hat{z}_j$ are

$$\begin{aligned} g_j^z \equiv \frac{\partial E}{\partial \hat{z}_j} &= \frac{\partial E}{\partial \hat{u}_j}\frac{\partial \hat{u}_j}{\partial \hat{z}_j} + \frac{\partial E}{\partial \hat{v}_j}\frac{\partial \hat{v}_j}{\partial \hat{z}_j} \\ &= g_j^u \frac{m_8 - m_{10}\frac{N_j^u}{D_j}}{D_j} + g_j^v \frac{m_9 - m_{10}\frac{N_j^v}{D_j}}{D_j}. \end{aligned} \tag{25}$$

The Hessian matrix $\mathbf{A}_z$ for the $\mathbf{z} = \{\hat{z}_j\}$ parameters has components

$$a_{jk}^z = \left(\mathbf{p}_j^{uv/z}\right)^T \mathbf{A}_{jk} \mathbf{p}_k^{uv/z} \tag{26}$$

with

$$\mathbf{p}_j^{uv/z} = \left(p_j^{u/z}, p_j^{v/z}\right)^T \equiv \left(\frac{\partial \hat{u}_j}{\partial \hat{z}_j}, \frac{\partial \hat{v}_j}{\partial \hat{z}_j}\right)^T.$$

There are two ways at this point to estimate the $\mathbf{m}$ and $\mathbf{z}$ parameters. We can take simultaneous steps in $\Delta\mathbf{m}$ and $\Delta\mathbf{z}$, or we can alternate steps in $\Delta\mathbf{m}$ and $\Delta\mathbf{z}$. The former approach is the one we adopted in Szeliski and Kang (1994), since the full Hessian matrix had already been computed thus enabling a regular Levenberg-Marquardt step, and this proved to have faster convergence. In this work, we adopt the latter approach, since it proved to be simpler to implement.

The performance of our rigid motion estimation algorithm on a sample image sequence is shown in Fig. 9. As can be seen, the overall direction of motion (the *epipolar geometry*) has been recovered well, and the motion estimates look reasonable[10]. The computed depth map is shown in grayscale. The lower right flow

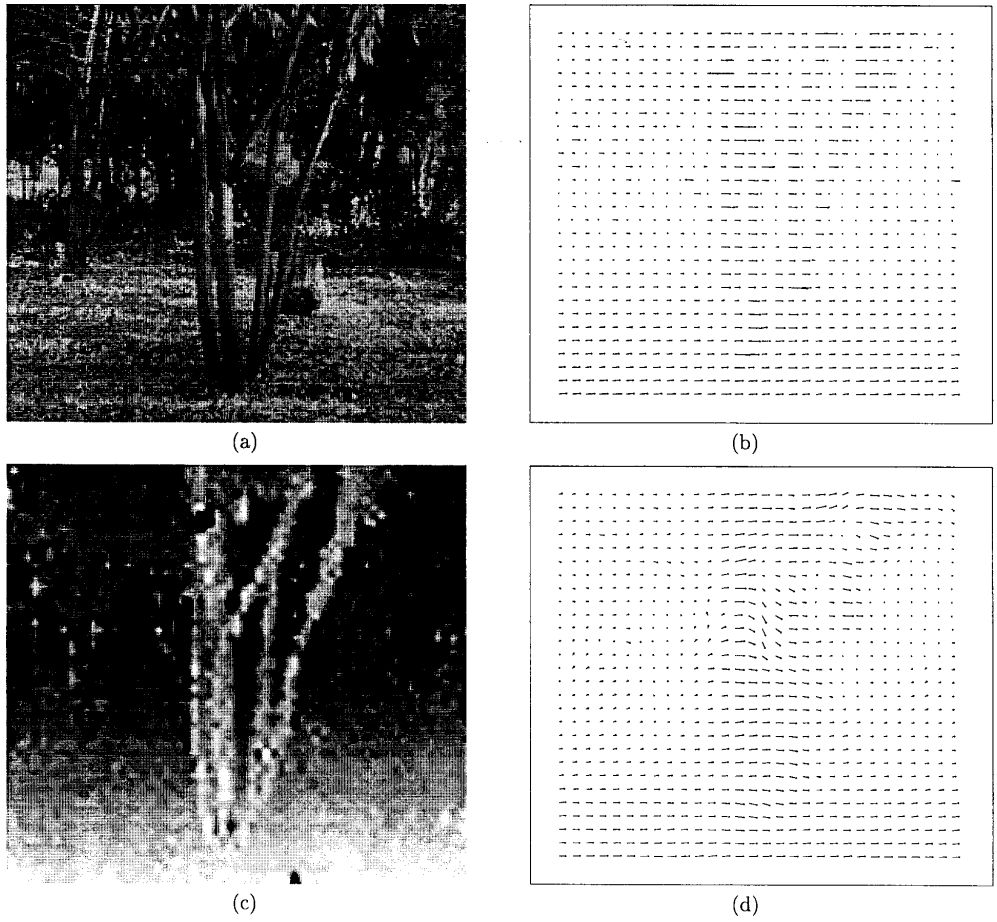


Figure 9. Example of 3D projective (rigid) motion estimation: (a) intensity image, (b) constrained rigid flow, (c) recovered depth map, (d) unconstrained local flow (for comparison).

field shows the local (general flow) model applied to the same image pair.

7. Multiframe Flow Estimation

Many current optical flow techniques use more than two images to arrive at local estimates of flow. This is particularly true of spatio-temporal filtering approaches (Adelson and Bergen, 1985; Heeger, 1987; Fleet and Jepson, 1990). For example, the implementation of Fleet and Jepson (1990) described in Barron et al. (1994) uses 21 images per estimate. Stereo matching techniques have also successfully used multiple images (Bolles et al., 1987; Matthies et al., 1989; Okutomi and Kanade, 1993). Using large numbers of images not only improves the accuracy of the estimates through noise averaging, but it can also disambiguate between possible matches (Okutomi and Kanade, 1993).

The extension of our local, global, and mixed motion models to multiple frames is straightforward. For local flow, we assume that displacements between successive images and a base image are known scalar multiples of each other,

$$\begin{aligned} u_t(x, y) &= s_t u_1(x, y) \\ v_t(x, y) &= s_t v_1(x, y), \end{aligned} \tag{27}$$

i.e., that we have linear flow (no acceleration).[11] We then minimize the overall cost function

$$E(\{u_i, v_i\}) = \sum_t \sum_i [I_t(x_i + s_t u_i, y_i + s_t v_i) - I_0(x_i, y_i)]^2. \tag{28}$$

This approach is similar to the *sum of sum of squared-distance* (SSSD) algorithm of Okutomi and Kanade

(1993), except that we represent the motion with a subsampled set of spline coefficients, eliminating the need for overlapping correlation windows.

The modifications to the flow estimation algorithm are minor and obvious. For example, the gradient with respect to the local flow estimate $\hat{u}_j$ in (8) becomes

$$g_j^u = 2 \sum_t s_t \sum_i e_{ti} G_{ti}^x w_{ij}$$

with e_{ti} and G_{ti}^x being the same as e_i and G_i^x with I_1 replaced by I_t. Given a block of images, estimating two sets of flows, one registered with the first image and another registered with the last, would allow us to do bidirectional prediction for motion-compensated video coding (Le Gall, 1991). Examples of the improvements in accuracy due to multiframe estimation are given in Section 8.

For global motion estimation, we can either assume that the motion estimates $\mathbf{m}_t$ are related by a known transform (e.g., uniform camera velocity), or we can assume an independent motion estimate for each frame. The latter situation seems more useful, especially in multiframe image mosaicing applications. The motion estimation problem in this case decomposes into a set of independent global motion estimation sub-problems.

The multiframe global/local motion estimation problem is more interesting. Here, we can assume that the global motion parameters for each frame $\mathbf{m}_t$ are independent, but that the local shape parameters $\hat{z}_j$ do not vary over time. This is the situation when we analyze multiple arbitrary views of a rigid 3-D scene, e.g., in the multiframe uncalibrated stereo problem (see Szeliski (1996) for some recent results). The modifications to the estimation algorithm are also straightforward. The gradients and Hessian with respect to the global motion parameters $\mathbf{m}_t$ are the same as before, except that the denominator D_{tj} is now different for each frame (since it is a function of $\mathbf{m}_t$).

The derivatives with respect to the depth estimates $\hat{z}_j$ are computed by summing over all frames

$$g_j^z = \sum_t p_{tj}^{u/z} g_{tj}^u + p_{tj}^{v/z} g_{tj}^v \tag{29}$$

where the $p_{tj}^{u/z}$ and $p_{tj}^{v/z}$ (which depend on $\mathbf{m}_t$) and the g_{tj}^u and g_{tj}^v (which depend on I_t) are different for each frame.

8. Experimental Results

In this section, we demonstrate the performance of our algorithms on the standard motion sequences analyzed in Barron et al. (1994). Some of the images in these sequences have already been shown in Figs. 4–9. The remaining images are shown in Figs. 10–14. We follow the organization of Barron et al. (1994), presenting quantitative results on synthetically generated sequences first, followed by qualitative results on real motion sequences.

Tables 1–5 give the quantitative results of our algorithms. In these tables, the top two rows are copied from Barron et al. (1994). The errors are reported as in Barron et al. (1994), i.e., by converting flow

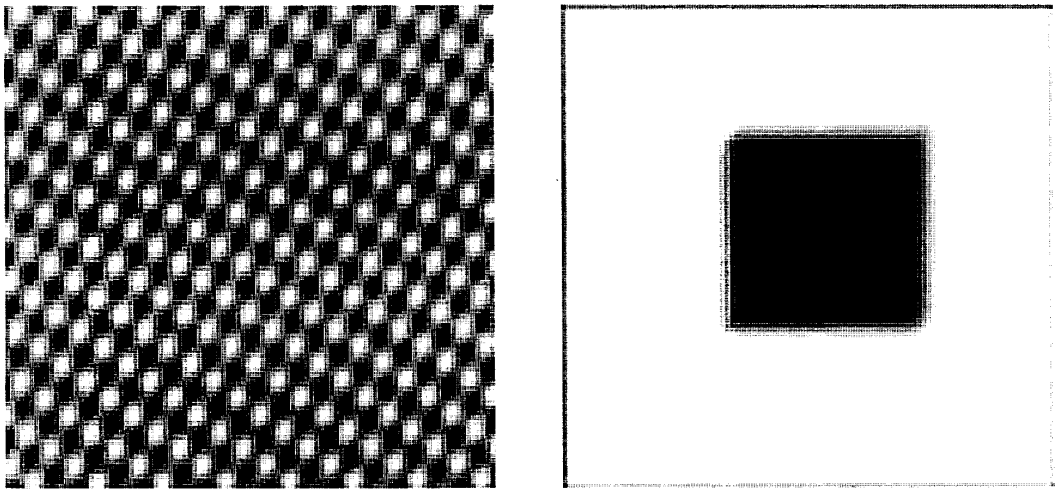


Figure 10. **Sinusoid 1** and **Square 2** sample images.

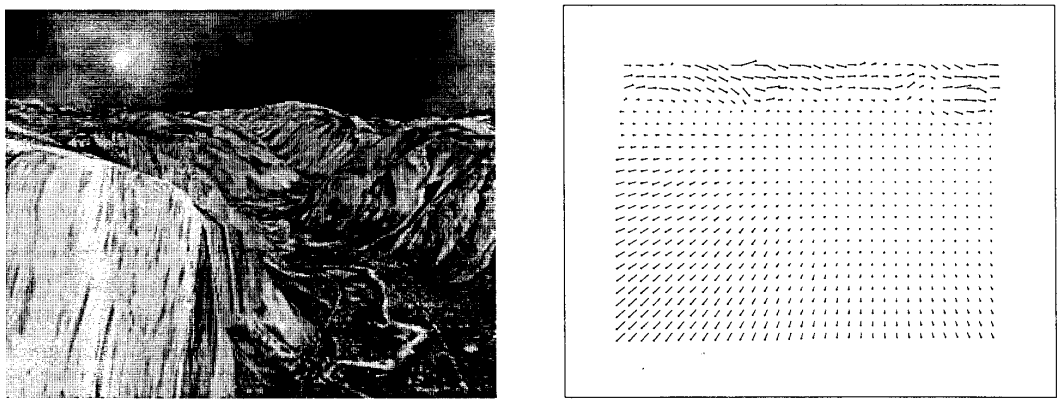


Figure 11. **Yosemite** sample image and flow (unthresholded).

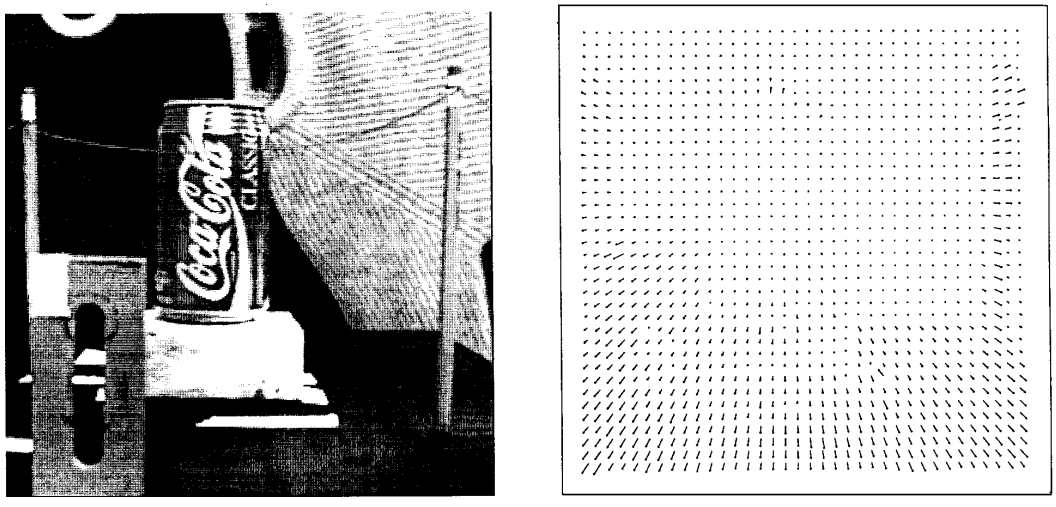


Figure 12. **NASA Sequence**, rigid flow.

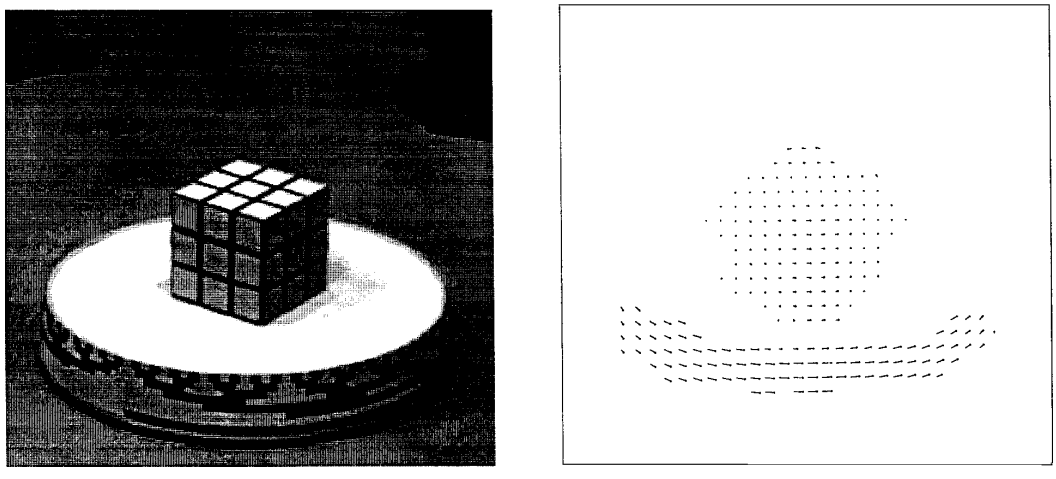


Figure 13. **Rubik Cube** sequence, sparse flow.

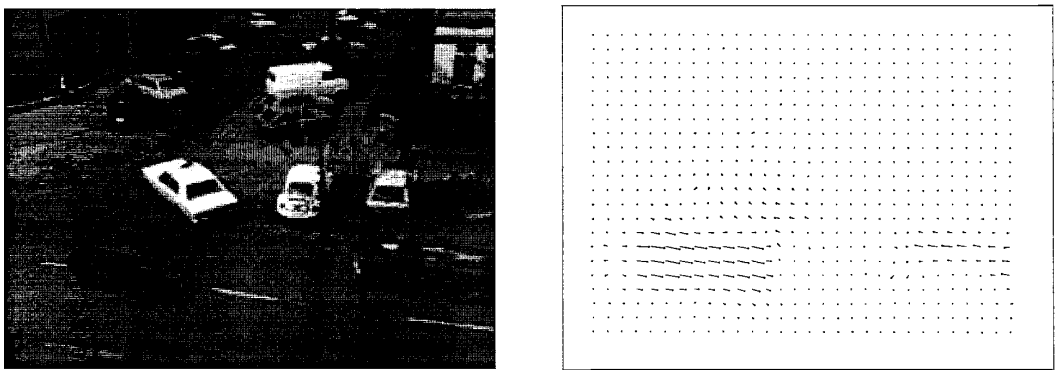


Figure 14. **Hamburg Taxi** sequence, dense local flow.

measurements into unit vector in $\mathcal{R}^3$ and taking the angle between them. The density is the percentage of flow estimates reported to have reliable flow estimates. The computation times on a DEC 3000 Model 500 AXP for these algorithms range from 0.5 second for the 100×100 **Sinusoid 1** image (single level, 9 iterations) to 9 seconds for the 300×300 **Nasa Sequence** (three levels, rigid flow, 9 iterations per level).

From the nine algorithms in Barron et al. (1994), we have chosen to show the Lucas and Kanade results, since their algorithm most closely matches ours and generally gives good results, and the Fleet and Jepson algorithm since it generally gave the best results. The most salient difference between our (local) algorithm and Lucas and Kanade is that we use a spline representation, which removes the need for overlapping correlation windows, and is therefore much more computationally efficient. The biggest difference with Fleet and Jepson is that they use the whole image sequence (20 frames) whereas we normally use only two (multi-frame results are shown in Table 3).

Table 1. Summary of **Sinusoid 1** results.

Technique	Average error	Standard deviation	Density
Lucas and Kanade (no thresholding)	2.47°	0.16°	100%
Fleet and Jepson ($\tau = 1.25$)	0.03°	0.01°	100%
Local flow ($n = 2, s = 2, L = 1, b = 0$)	0.17°	0.02°	100%
Local flow ($n = 3, s = 2, L = 1, b = 0$)	0.07°	0.01°	100%
Local flow ($n = 5, s = 2, L = 1, b = 0$)	0.03°	0.01°	100%
Local flow ($n = 7, s = 2, L = 1, b = 0$)	0.02°	0.01°	100%
Affine flow ($s = 2, L = 1, b = 0$)	0.13°	0.01°	100%
Affine flow ($s = 4, L = 1, b = 0$)	0.06°	0.01°	100%

Table 2. Summary of **Square 2** results.

Technique	Average error	Standard deviation	Density
Lucas and Kanade ($\lambda_2 \geq 5.0$)	0.14°	0.10°	7.9%
Fleet and Jepson ($\tau = 2.5$)	0.18°	0.13°	12.6%
Local flow ($T_e = 10^4$)	2.98°	1.16°	9.1%
Local flow ($s = 2, T_e = 10^4$)	1.78°	1.07°	10.1%
Local flow ($s = 2, \lambda_1 = 10^3, T_e = 10^4$)	0.47°	0.27°	23.8%
Local flow ($s = 2, \lambda_1 = 10^4$)	0.13°	0.10°	100%
Affine flow	0.03°	0.02°	100%

Table 3. Summary of **Translating Tree** results.

Technique	Average error	Standard deviation	Density
Lucas and Kanade ($\lambda_2 \geq 5.0$)	0.56°	0.58°	13.1%
Fleet and Jepson ($\tau = 1.25$)	0.23°	0.19°	49.7%
Local flow ($n = 2$)	0.35°	0.34°	100%
Local flow ($n = 3$)	0.30°	0.30°	100%
Local flow ($n = 5$)	0.24°	0.15°	100%
Local flow ($n = 8$)	0.19°	0.10°	100%
Affine flow	0.14°	0.06°	100%

Table 4. Summary of **Diverging Tree** results.

Technique	Average error	Standard deviation	Density
Lucas and Kanade ($\lambda_2 \geq 5.0$)	1.65°	1.48°	24.3%
Fleet and Jepson ($\tau = 1.25$)	0.80°	0.73°	46.5%
Local flow ($s = 4, L = 1$)	0.98°	0.74°	100%
Local flow ($s = 4, L = 1, \lambda_1 = 10^3$)	0.78°	0.47°	100%
Affine flow	2.51°	0.77°	100%

Table 5. Summary of **Yosemite** results.

Technique	Average error	Standard deviation	Density
Lucas and Kanade ($\lambda_2 \geq 5.0$)	3.22°	8.92°	8.7%
Fleet and Jepson ($\tau = 1.25$)	5.28°	14.34°	30.6%
Local flow ($s = 2$, $T_e = 3000$)	2.20°	5.87°	23.1%
Local flow ($s = 2$, $T_e = 2000$)	3.09°	7.59°	39.6%
Local flow, cropped ($s = 2$)	1.40°	1.52°	100%
Rigid flow, cropped ($s = 2$)	1.23°	1.11°	100%

As with any motion estimation algorithm, our algorithms require the selection of some relevant parameters. The most important of these are:

- n (= 2) the number of frames
- s (= 1) the step between frames, i.e., 1 = consecutive frames, 2 = every other frame, ...
- m (= 16) the size of the patch (width and height, m^2 pixels per patch)
- o (= bilinear) order of spline
- L (= 3) the number of coarse-to-fine levels
- b (= 3) the amount of initial blurring (# of iterations of a box filter).

Unless mentioned otherwise, we used the default values shown in parentheses above for the results in Tables 1–5. Bilinear interpolation was used for the flow fields (unless otherwise noted), as well as for intensity image resampling.

The simplest motions to analyze are two constant-translation sequences, **Sinusoid 1** and **Square 2** (Fig. 10). The translations in these sequences are (1.585, 0.863) and (1.333, 1.333) pixels per frame, respectively. Our local flow estimates for the sinusoid sequence are very good using only two frames (Table 1), and beat all other algorithms when 7 or more frames are used. For this sequence, we use a single level and no blurring and take a ($s = 2$) frame step for better results. To help overcome local minima for the multiframe ($n > 2$) sequences, we solve a series of easier subproblems (Xu et al., 1987). We first estimate two-frame motion, then use the resulting estimate to initialize a three-frame estimator, etc. Without this modification, performance on longer (e.g., $n = 8$) sequences would start to degrade because of local minima. The global affine model motion estimator performs well.

For the translating square (Table 2), our results are not as good because of the aperture problem, but with additional regularization, we still outperform all of the nine algorithms studied in Barron et al. (1994). To produce the sparse flow estimates (9–23% density), we first compute the minimum eigenvalues of the local Hessian matrices $\mathbf{A}_{jj}$ which are associated with each spline control vertex. We then interpolate these values over the whole grid, and select flow estimate which are above a threshold T_e. This process selects areas where both components of the motion estimate are well determined. The affine (global) flow for the square sequence works extremely well, outperforming all other techniques by a large margin.

The sequences **Translating Tree** and **Diverging Tree** were generated using a real image (Fig. 7) and synthetic (global) motion. Our results on the translating motion sequence (Table 3) are as good as any other technique for the local algorithm (note the difference in density between our results and the previous ones), and outperform all techniques for the affine motion model, even though we are just using two frames from the sequence. The results on the diverging tree sequence are good for the local flow, but not as good for the affine flow. These results are comparable or better than the other techniques in Barron et al. (1994) which produce 100% density.

The final motion sequence for which quantitative results are available is **Yosemite** (Fig. 11 and Table 5). The images in this sequence were generated by Lynn Quam using his texture mapping algorithm applied to an aerial photograph registered with a digital terrain model. There is significant occlusion and temporal aliasing, and the fractal clouds move independently from the terrain. Our results on this more realistic sequence are better than any of the techniques in Barron et al. (1994), even though we again only use two images. As expected, the quality of the results depends on the threshold T_e used to produce sparse flow estimates, i.e., there is a tradeoff between the density of the estimates and their quality. We also ran our algorithm on just the lower 176 (out of 252) rows of the images sequence. The dense (unthresholded) estimates are comparable to the thresholded full-frame estimates. The results using the rigid motion model are even better.

To study the effects of using different basis functions for the spline-based motion estimator, we ran our algorithm on the same cropped portion of the Yosemite sequence as above. Table 6 shows the resulting error statistics. From this table, it is evident that our spline-based estimator does significantly better than a block-based motion estimator (first line). However, there

Table 6. Summary of **Yosemite** results for varying spline basis functions.

Technique	Average error	Standard deviation	Density
Local flow, cropped ($s=2$, block)	3.11°	2.44°	100%
Local flow, cropped ($s=2$, linear square)	1.43°	1.54°	100%
Local flow, cropped ($s=2$, linear triangle)	1.50°	1.62°	100%
Local flow, cropped ($s=2$, bilinear)	1.40°	1.52°	100%
Local flow, cropped ($s=2$, biquadratic)	1.40°	1.41°	100%

does not seem to be a significant difference between the other orders of interpolation. This is somewhat surprising, since one might expect biquadratic splines to over-smooth the motion field, especially near the discontinuities. It appears that our default choice of using a bilinear interpolator, which can adequately model large projective distortions, is a reasonable choice.

To conclude our experimental section, we show results on some real motion sequences for which no ground truth data is available. The **SRI Trees** results have already been presented in Fig. 9 for both rigid and local (general) flow. Figure 12 shows the **NASA Sequence** in which the camera moves forward in a rigid scene (there is significant aliasing). The motion estimates look quite reasonable, as does the associated depth map (not shown)[12]. Figure 13 shows the sparse flow computed for the **Rubik Cube** sequence (the dense flows were shown in Fig. 4). The areas with texture and/or corners produce the most reliable flow estimates. Finally, the results on the **Hamburg Taxi** are shown in Fig. 14, where the independent motion of the three moving cars can be clearly distinguished. Overall, these results are comparable or better than those shown in Barron et al. (1994).

Further work needs to be done in the experimental evaluation of our algorithms. In addition to systematically studying the effects of the parameters n, s, m, o, L, and b, we should examine the effects of different preconditioners, and the usefulness of using conjugate gradient descent (see Szeliski and Shum (1996) for some more recent results).

9. Discussion

The spline-based motion estimation algorithms introduced in this paper are a hybrid of local optic flow algorithms and global motion estimators, utilizing the best features of both approaches. Like other local methods, we can produce detailed and accurate general flow estimates. Unlike correlation-based methods, however, we do not assume a local translational model in each correlation window. Instead, the pixel motion within each of our patches can model affine or even more complex motions (using linear or higher order splines). Bilinear interpolation of the four spline control vertices can provide a good approximation to local projective flow[13]. This is especially important when we analyze extended motion sequences, where local intensity patterns can deform significantly. When used to track features, our technique can be viewed as a generalization of affine patch trackers (Rehg and Witkin, 1991; Shi and Tomasi, 1994) where the patch corners are stitched together over the whole image (Szeliski et al., 1995).

Another major difference between our spline-based approach and correlation-based approaches is in computational efficiency. Each pixel in our approach only contributes its error to the 4 spline control vertices influencing its displacement, whereas in correlation-based approaches, each pixel contributes to m^2 overlapping windows. Furthermore, operations such as inverting the local Hessian or computing the contribution to a global model only occur at the spline control vertices, thereby providing an $O(m^2)$ speedup over correlation-based techniques. For typically-sized patches ($m = 8$), this can be significant. The price we pay for this efficiency is a slight decrease in the resolution of the computed flow field, especially when compared to locally adaptive widows (Okutomi and Kanade, 1992) (which are extremely computationally demanding). However, since window-based approaches produce highly correlated estimates anyway, we do not expect this difference to be significant. We have recently extended our approach to handle patches of varying size using the concept of quadtree splines (Szeliski and Shum, 1996).

Compared to spatio-temporal filtering approaches, we see a similar improvement in computational efficiency. Separable filters can reduce the complexity of computing the required local features from $O(m^3)$ to $O(m)$, but these operations must still be performed at each pixel. Furthermore, a large number of differently tuned filters are normally used. Since the final estimates are highly correlated anyway, it just makes more computational sense to perform the calculations on a sparser grid, as we do.

Because our spline-based motion representation already has a smoothness constraint built in, regularization, which requires many iterations to propagate

local constraints, is not usually necessary. If we desire longer-range smoothness constraints, regularization can easily be added to our framework. Having fewer free variables in our estimation framework leads to faster convergence when iteration is necessary to propagate such constraints.

Turning to global motion estimation, our motion model for planar surface flow can handle arbitrarily large motions and displacements, unlike the instantaneous model of Bergen et al. (1992). We see this as an advantage in many situations, e.g., in compositing multiple views of planar surfaces (Szeliski, 1996). Furthermore, our approach does not require the camera to be calibrated and can handle temporally-varying internal camera parameters. While our flow field is not linear in the unknown parameters, this is not significant, since the overall problem is non-linear and requires iteration.

Our mixed global/local (rigid body) model shares similar advantages over previously developed direct methods: it does not require camera calibration and can handle time-varying camera parameters and arbitrary camera displacements. Furthermore, experimental evidence from some related structure from motion research (Szeliski and Kang, 1994) suggests that our projective formulation of structure and motion converges more quickly than traditional Euclidean formulations.

Finally, our experimental results demonstrate that our techniques are competitive in quality with the best currently available motion estimators examined in Barron et al. (1994), especially when additional regularization is used.

10. Future Work and Conclusions

In recent work, we have extended our algorithm in a number of directions, and applied it to some interesting image registration problems. To improve the performance of our algorithm on difficult scenes with repetitive textures, we have added local search, i.e., extended the algorithm to evaluate several possible displacements instead of just relying on gradient descent (Anandan, 1989; Singh, 1990). We have added hierarchical basis functions to the algorithm as an alternative to coarse-to-fine estimation (Szeliski, 1990), and used these to construct adaptively sized patches using the concept of quadtree splines (Szeliski and Shum, 1996). We have also developed a parallel feature tracker based on our algorithm for extended motion sequences, using inter-frame motion prediction and keeping only motion estimates with high local confidence (Szeliski et al., 1995). In future work, we would like to address the problems of discontinuities and occlusions (Geiger et al., 1992), which must be resolved in order to improve the quality of the estimates.

In terms of applications, we have been using our global flow estimator to register multiple 2D images, e.g., to composite pieces of flat scenes such as whiteboards or panoramas into high-resolution stills (Szeliski, 1996). We have also used our local/global model to extract 3D projective scene geometry from multiple images as a basis for doing view interpolation (Szeliski, 1996; Szeliski and Kang, 1995). In the future, we would like to test our spline-based motion estimator as a predictor for motion-compensated video coding as an alternative to block-structured predictors such as MPEG.

To summarize, spline-based image registration combines the best features of local motion models and global (parametric) motion models. The size of the spline patches and the order of spline interpolation can be used to vary smoothly between these two extremes, and can even in some cases be determined locally (Szeliski and Shum, 1996). The resulting algorithm is more computationally efficient than correlation-based or spatio-temporal filter-based techniques while providing estimates of comparable quality. Purely global and mixed local/global estimators have also been developed based on this representation for those situations where a more specific motion model can be used.

Acknowledgments

This work was performed while the authors were employed at Digital Equipment Corporation.

Notes

1. The spline-based flow fields we describe in the next section can be viewed as local parametric models, since the flow within each spline patch is defined by a small number of control vertices.
2. We will use the terms *displacement field*, *flow field*, and *motion estimate* interchangeably.
3. In the remainder of the paper, we will use indices i for pixels and j for spline control vertices.
4. As mentioned in Press et al. (1992), inclusion of these terms can be destabilizing if the model fits badly or is contaminated by outlier points.
5. A Bayesian justification can be found in Simoncelli et al. (1991), and additional possible local weightings in [(Lucas, 1984), p. 20].

6. The definitions of $\hat{u}_j$ and $\hat{v}_j$ would have to be adjusted if biquadratic splines are being used.
7. Multiple affine patches can also be used to track features (Rehg and Witkin, 1991). Our general flow estimator (Section 4) performs a similar function, except in parallel and with overlapping windows (Szeliski et al., 1995).
8. In its full generality, we should have an m_8 instead of a 1 in the denominator. The situation $m_8 = 0$ occurs only for 90° camera rotations.
9. Bergen et al. (1992) mitigate the Z sensitivity problem by estimating $1/Z(x, y)$ instead.
10. We initialized the **m** vector in this example to a horizontal epipolar geometry. If unknown, the epipolar geometry can be recovered from a general 2-D motion field (Faugeras, 1992; Hartley et al., 1992; Hartley and Gupta, 1993; Szeliski and Kang, 1995).
11. In the most common case, a uniform temporal sampling ($s_t = t$) is assumed, but this is not strictly necessary (Okutomi and Kanade, 1993).
12. For this sequence and for the Yosemite sequence, we initialized the **m** vector to a forward looming motion.
13. To obtain an exact projective flow, we would have to use *rational splines* (Farin, 1992), which require a division per pixel and are therefore computationally expensive.

References

Adelson, E.H. and Bergen, J.R. 1985. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America*, A2(2):284–299.

Amit, Y. 1993. A non-linear variational problem for image matching. unpublished manuscript (from Newton Institute).

Anandan, P. 1989. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310.

Bajcsy, R. and Broit, C. 1982. Matching of deformed images. In *Sixth International Conference on Pattern Recognition (ICPR'82)*, IEEE Computer Society Press: Munich, Germany, pp. 351–353.

Bajcsy, R. and Kovacic, S. 1989. Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing*, 46(1):1–21.

Barnard, S.T. and Fischler, M.A. 1982. Computational stereo. *Computing Surveys*, 14(4):553–572.

Barron, J.L., Fleet, D.J., and Beauchemin, S.S. 1994. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77.

Beier, T. and Neely, S. 1992. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH'92)*, 26(2):35–42.

Bergen, J.R., Anandan, P., Hanna, K.J., and Hingorani, R. 1992. Hierarchical model-based motion estimation. In *Second European Conference on Computer Vision (ECCV'92)*, Santa Margherita Liguere, Springer-Verlag: Italy, pp. 237–252.

Beymer, D., Shashua, A., and Poggio, T. 1993. Example based image analysis and synthesis. A.I. Memo 1431, Massachusetts Institute of Technology.

Blake, A., Curwen, R., and Zisserman, A. 1993. A framework for spatio-temporal control in the tracking of visual contour. *International Journal of Computer Vision*, 11(2):127–145.

Bolles, R.C., Baker, H.H., and Marimont, D.H. 1987. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55.

Brown, L.G. 1992. A survey of image registration techniques. *Computing Surveys*, 24(4):325–376.

Burr, D.J. 1981. A dynamic model for image registration. *Computer Graphics and Image Processing*, 15(2):102–112.

Burt, P.J. and Adelson, E.H. 1983. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540.

Carlbom, I., Terzopoulos, D., and Harris, K.M. 1991. Reconstructing and visualizing models of neuronal dendrites. In *Scientific Visualization of Physical Phenomena*, N.M. Patrikalakis (Ed.), Springer-Verlag: New York, pp. 623–638.

Dhond, U.R. and Aggarwal, J.K. 1989. Structure from stereo—A review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510.

Dreschler, L. and Nagel, H.-H. 1982. Volumetric model and 3D trajectory of a moving car derived from monocular TV frame sequences of a stree scene. *Computer Graphics and Image Processing*, 20:199–228.

Enkelmann, W. 1988. Investigations of multigrid algorithms for estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing*, pp. 150–177.

Farin, G.E. 1992. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press: Boston, Massachusetts, 3rd edition.

Faugeras, O.D. 1992. What can be seen in three dimensions with an uncalibrated stereo rig? In *Second European Conference on Computer Vision (ECCV'92)*, Santa Margherita Liguere, Springer-Verlag: Italy, pp. 563–578.

Fleet, D. and Jepson, A. 1990. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104.

Fuh, C.-S. and Maragos, P. 1991. Motion displacement estimation using an affine model for image matching. *Optical Engineering*, 30(7):881–887.

Geiger, D., Ladendorf, B., and Yuille, A. 1992. Occlusions and binocular stereo. In *Second European Conference on Computer Vision (ECCV'92)*, Santa Margherita Liguere, Springer-Verlag, Italy, pp. 425–433.

Gennert, M.A. 1988. Brightness-based stereo matching. In *Second International Conference on Computer Vision (ICCV'88)*, IEEE Computer Society Press: Tampa, Florida, pp. 139–143.

Goshtasby, A. 1986. Piecewise linear mapping functions for image registration. *Pattern Recognition*, 19(6):459–466.

Goshtasby, A. 1988. Image registration by local approximation methods. *Image and Vision Computing*, 6(4):255–261.

Hanna, K.J. 1991. Direct multi-resolution estimation of ego-motion and structure from motion. In *IEEE Workshop on Visual Motion*, IEEE Computer Society Press: Princeton, New Jersey, pp. 156–162.

Hartley, R. and Gupta, R. 1993. Computing matched-epipolar projections. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, IEEE Computer Society Press: New York, pp. 549–555.

Hartley, R., Gupta, R., and Chang, T. 1992. Stereo from uncalibrated cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, IEEE Computer Society Press: Champaign, Illinois, pp. 761–764,.

Heeger, D.J. 1987. Optical flow from spatiotemporal filters. In *First International Conference on Computer Vision (ICCV'87)*, IEEE Computer Society Press: London, England, pp. 181–190.

Hildreth, E.C. 1986. Computing the velocity field along contours.

In *Motion: Representation and Perception*, N.I. Badler and J.K. Tsotsos (Eds.), North-Holland, New York, pp. 121–127.

Horn, B.K.P. and Schunck, B.G. 1981. Determining optical flow. *Artificial Intelligence*, 17:185–203.

Horn, B.K.P. and Weldon, E.J., Jr. 1988. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76.

Kass, M., Witkin, A., and Terzopoulos, D. 1988. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Koenderink, J.J. and van Doorn, A.J. 1991. Affine structure from motion. *Journal of the Optical Society of America A*, 8:377–385,538.

Le Gall, D. 1991. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):44–58.

Lucas, B.D. 1984. *Generalized Image Matching by the Method of Differences*. Ph.D. Thesis, Carnegie Mellon University.

Lucas, B.D. and Kanade, T. 1981. An iterative image registration technique with an application in stereo vision. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, Vancouver, pp. 674–679.

Manmatha, R. and Oliensis, J. 1992. Measuring the affine transform —I: Scale and rotation. Technical Report 92-74, University of Massachussets, Amherst, Massachussets.

Matthies, L.H., Szeliski, R., and Kanade, T. 1989. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209–236.

Menet, S., Saint-Marc, P., and Medioni, G. 1990. B-snakes: implementation and applications to stereo. In *Image Understanding Workshop*, Morgan Kaufmann Publishers: Pittsburgh, Pennsylvania, pp. 720–726.

Mohr, R., Veillon, L., and Quan, L. 1993. Relative 3D reconstruction using multiple uncalibrated images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, New York, pp. 543–548.

Nagel, H.-H. 1987. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324.

Okutomi, M. and Kanade, T. 1992. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162.

Okutomi, M. and Kanade, T. 1993. A multiple baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363.

Poggio, T., Torre, V., and Koch, C. 1985. Computational vision and regularization theory. *Nature*, 317(6035):314–319.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press: Cambridge, England, 2nd edition.

Quam, L.H. 1984. Hierarchical warp stereo. In *Image Understanding Workshop*, Science Applications International Corporation: New Orleans, Louisiana, pp. 149–155.

Rehg, J. and Witkin, A. 1991. Visual tracking with deformation models. In *IEEE International Conference on Robotics and Automation*, IEEE Computer Society Press: Sacramento, California, pp. 844–850.

Sethi, I.K. and Jain, R. 1987. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(1):56–73.

Shi, J. and Tomasi, C. 1994. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, IEEE Computer Society: Seattle, Washington, pp. 593–600.

Simoncelli, E.P., Adelson, E.H., and Heeger, D.J. 1991. Probability distributions of optic flow. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, IEEE Computer Society Press: Maui, Hawaii, pp. 310–315.

Singh, A. 1990. An estimation-theoretic framework for image-flow computation. In *Third International Conference on Computer Vision (ICCV'90)*, IEEE Computer Society Press: Osaka, Japan, pp. 168–177.

Szeliski, R. 1989. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers: Boston, Massachusetts.

Szeliski, R. 1990. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528.

Szeliski, R. 1996. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30.

Szeliski, R. and Kang, S.B. 1994. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28.

Szeliski, R. and Kang, S.B. 1995. Direct methods for visual scene reconstruction. In *IEEE Workshop on Representations of Visual Scenes*, Cambridge, Massachusetts, pp. 26–33.

Szeliski, R. and Shum, H.-Y. 1996. Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210.

Szeliski, R., Kang, S.B., and Shum, H.-Y. 1995. A parallel feature tracker for extended image sequences. In *IEEE International Symposium on Computer Vision*, Coral Gables, Florida, pp. 241–246.

Terzopoulos, D. 1986. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424.

Tomasi, C. and Kanade, T. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154.

Witkin, A., Terzopoulos, D., and Kass, M. 1987. Signal matching through scale space. *International Journal of Computer Vision*, 1:133–144.

Wolberg, G. 1990. *Digital Image Warping*. IEEE Computer Society Press: Los Alamitos, California.

Xu, G., Tsuji, S., and Asada, M. 1987. A motion stereo method based on coarse-to-fine control strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):332–336.

Zheng, Q. and Chellappa, R. 1992. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. Technical Report CAR-TR-628, Computer Vision Laboratory, Center for Automation Research, University of Maryland.