Hierarchical Spline-Based Image Registration

Richard Szeliski
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square, Bldg. 700
Cambridge, MA 02139

James Coughlan
Harvard University
Department of Physics
Cambridge, MA 02138

Abstract

The problem of image registration subsumes a number of topics 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 a spline representation 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.

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. The 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 sub-problems 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, the results to be computed, and the techniques used to compute these results.

In this paper, we present a general motion estimation framework which can be specialized to solve a number of these subproblems. Like Bergen *et al.* [5], 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 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 remainder of the paper is structured as follows. Section 2 presents a review of relevant previous work. Section 3 gives the general problem formulation. 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. 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. A longer version of this paper is available as [19].

2 Previous work

A large number of motion estimation and image registration algorithms have been developed in the past. These algorithms include optical flow estimators, global parametric motion estimators, constrained motion estimators (*direct methods*), stereo and multiframe stereo, hierarchical (coarse-to-fine) methods, and feature trackers. We will use this rough taxonomy to briefly review previous work.

The general motion estimation problem is often called *optical flow* recovery [9] and involves estimating an independent displacement vector for each pixel in an image. Approaches to this problem include gradient-based techniques using the *brightness constraint* [9, 11, 13], correlation-based techniques such as the Sum of Squared Differences (SSD) [2], spatio-temporal filtering [1, 8], and regularization [9]. Nagel [13] and Anandan [2] provide comparisons and derive relations between different techniques, while Barron *et al.* [4] provide some numerical comparisons.

1063-6919/94 $3.00 © 1994 IEEE

Global motion estimators [5] 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.* [5], these fields are called parametric motion models, since they can be used locally as well.[1] Global methods are most useful when the scene has a particularly simple form, e.g., when the scene is planar.

Constrained (*quasi-parametric* [5]) 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 [10]. In this paper, we use projective descriptions of motion and depth [7, 20] for our constrained motion model, which removes the need for calibrated cameras.

Stereo [3] can be viewed as a simplified version of a 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*) [12, 14].

Hierarchical (coarse-to-fine) matching algorithms have a long history of use both in stereo matching [16, 21] and in motion estimation [2, 5]. 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 alternative to iconic (pixel-based) optic flow techniques. The algorithm presented in this paper is closely related to patch-based feature trackers [11, 17]. Like [17], which is an affine-patch based tracker, it can handle large deformations in the patches being tracked.

3 General problem formulation

The general image registration problem can be formulated as follows. Given a sequence of images $I_t(x, y)$ which we assume were formed by locally displacing a reference image $I(x, y)$,

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

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 and consists of minimizing the squared error

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

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 $\{(x_i, y_i)\}$,

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

This equation, simplified for the two frame problem, is called the *Sum of Squared Differences* (SSD) formula [2]. Expanding I_1 in a first order Taylor series expansion in (u_i, v_i) yields the the *image brightness constraint* [9, 2].

The above minimization problem will typically have many locally optimal solutions. The choice of method for finding the best estimate efficiently is what typically differentiates between various motion estimation algorithms. For example, the SSD algorithm performs the summation at each pixel over a 5×5 window [2]. Regularization-based algorithms add smoothness constraints on the u and v fields to obtain good solutions [9]. Multiscale or hierarchical (coarse-to-fine) techniques are often used to speed the search for the optimum. Assigning an independent estimate at each pixel (u_i, v_i) is most common, but global motion descriptors are also possible [5] (Section 5).

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*. The value for the displacement at a pixel i can be written as

$$\mathbf{u}(x_i, y_i) = \sum_j \hat{\mathbf{u}}_j B_j(x_i, y_i) \quad \text{or} \quad \mathbf{u}_i = \sum_j \hat{\mathbf{u}}_j w_{ij}, \tag{4}$$

where the $B_j(x, y)$ are called the *basis functions* and are only non-zero over a small interval (*finite support*). We call the $w_{ij} = B_j(x_i, y_i)$ *weights* to emphasize that the $\mathbf{u}_i = (u_i, v_i)^T$ are known linear combinations of the $\hat{\mathbf{u}}_j = (\hat{u}_j, \hat{v}_j)^T$.

In our current implementation, the basis functions are spatially shifted versions of each other, i.e., $B_j(x, y) = B(x - \hat{x}_j, y - \hat{y}_j)$. We have studied five different interpolation functions: (1) block, (2) linear, (3) linear on triangles, (4) bilinear, and (5) biquadratic [19]. We also impose the condition that the spline control grid is a regular subsampling of the pixel grid, $\hat{x}_j = m x_i$, $\hat{y}_j = m y_i$, so that each set of $m \times m$ pixels corresponds to a single spline patch.

Spline-based flow descriptors 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). 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, where the squared pixel errors correspond to Gaussian noise, while the spline model (and any associated regularizers) form the prior model.

[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.

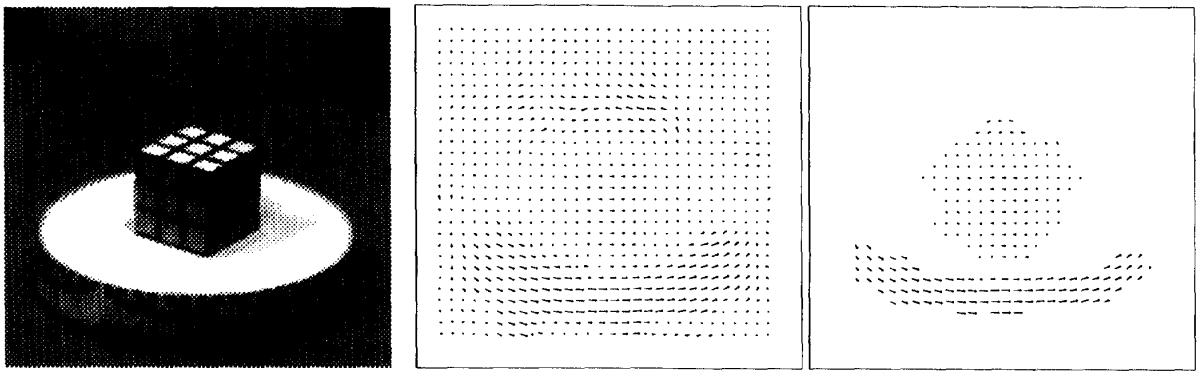

Figure 1: Example of general flow computation. Middle frame shows dense flow, right frame shows sparse (thresholded) flow.

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 [15]. First, we compute the gradient of E in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$,

$$\mathbf{g}_j \equiv \frac{\partial E}{\partial \hat{\mathbf{u}}_j} = 2 \sum_i e_i \mathbf{G}_i w_{ij}, \tag{5}$$

where

$$e_i = I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i) \tag{6}$$

is the intensity error at pixel i,

$$\mathbf{G}_i = \nabla I_1(x_i + u_i, y_i + v_i) \tag{7}$$

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).

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

$$\mathbf{A}_{jk} \equiv 2 \sum_i \frac{\partial e_i}{\partial \hat{\mathbf{u}}_j} \frac{\partial e_i}{\partial \hat{\mathbf{u}}_k}^T = 2 \sum_i w_{ij} w_{ik} \mathbf{G}_i \mathbf{G}_i^T. \tag{8}$$

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 $\mathbf{A}_{jj}$ encodes the local shape of the sum-of-squared difference correlation surface [2, 11]. This matrix is often used to compute an updated flow vector by setting

$$\Delta \hat{\mathbf{u}}_j = -\mathbf{A}_{jj}^{-1} \mathbf{g}_j \tag{9}$$

[2, 5, 11]. 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

$$(\mathbf{A} + \lambda \operatorname{diag}(\mathbf{A})) \Delta \mathbf{u} = -\mathbf{g}, \tag{10}$$

where $\mathbf{u}$ is the vector of concatenated displacement estimates $\hat{\mathbf{u}}_j$, $\mathbf{g}$ is the vector of concatenated energy gradients $\mathbf{g}_j$, and λ is a stabilization factor which varies over time [15]. For systems with small numbers of parameters, e.g., if only a single spline patch is being used (Section 5), this system of equations 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 our current implementation, we use preconditioned gradient descent to update our flow estimates

$$\Delta \mathbf{u} = -\alpha \, \mathbf{B}^{-1} \mathbf{g} = -\alpha \hat{\mathbf{g}} \tag{11}$$

where $\mathbf{B} = \hat{\mathbf{A}} + \lambda \mathbf{I}$, and $\hat{\mathbf{A}} = \text{block\_diag}(\mathbf{A})$ is the set of 2×2 block diagonal matrices used in (9). This update rule is very close to that used by [11] 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. (3) there is a step size α, which is necessary because we are ignoring the off-block-diagonal terms in $\mathbf{A}$. The step α can either be set conservatively, or it 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 $\alpha = (\mathbf{d} \cdot \mathbf{g})/(\mathbf{d}^T \mathbf{A} \mathbf{d})$, where $\mathbf{d} = \hat{\mathbf{g}}$ is the descent direction.

To handle larger displacements, we run our algorithm in a coarse-to-fine (hierarchical) fashion. A Gaussian image pyramid is first computed [6]. 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 1 shows an example of the flow estimates produced by our technique. The input image is 256×240 pixels, and

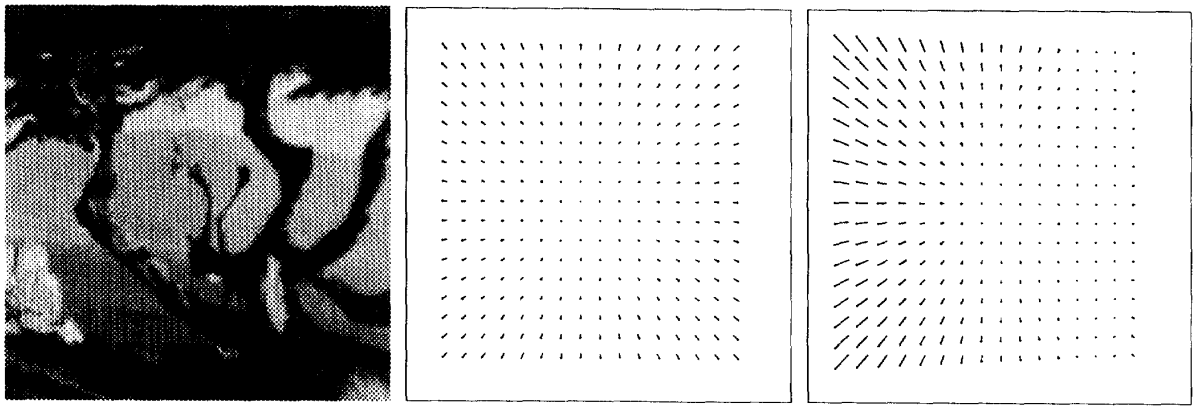

Figure 2: Examples of affine and 2D projective flow estimates

the flow is displayed on a 30 × 28 grid. We used 16 × 16 pixel spline patches and 3 levels in the pyramid, with 9 iterations at each level. 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. Thresholding the flow estimates based on the minimum eigenvalue of the local Hessian matrices produces the flow shown in the rightmost frame (Section 8). This example uses no regularization beyond that imposed by the spline patches.

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 and the scene is large [5], or when performing a coarse registration of slices in a volumetric data set, a single global description of the motion model may suffice. A simple example of such a global motion is an affine flow [5, 17]

$$\begin{aligned} u(x,y) &= (m_0 x + m_1 y + m_2) - x \\ v(x,y) &= (m_3 x + m_4 y + m_5) - y. \end{aligned} \tag{12}$$

The parameters $\mathbf{m} = (m_0, \ldots, m_5)^T$ are called the *global motion parameters*.

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

$$\hat{\mathbf{u}}_j = \mathbf{T}_j \mathbf{m} - \hat{\mathbf{x}}_j \text{ with } \mathbf{T}_j = \begin{bmatrix} \hat{x}_j & \hat{y}_j & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{x}_j & \hat{y}_j & 1 \end{bmatrix}. \tag{13}$$

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

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

$$\mathbf{g}_m \equiv \frac{\partial E}{\partial \mathbf{m}} = \sum_j \frac{\partial \hat{\mathbf{u}}_j}{\partial \mathbf{m}} \frac{\partial E}{\partial \hat{\mathbf{u}}_j} = \sum_j \mathbf{T}_j^T \mathbf{g}_j. \tag{14}$$

The Hessian matrix can be computed in a

$$\mathbf{A}_m \equiv \frac{\partial^2 E}{\partial \mathbf{m}^T \partial \mathbf{m}} \approx \sum_{jk} \mathbf{T}_j^T \mathbf{A}_{jk} \mathbf{T}_k, \tag{15}$$

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 reasonable approximation.

An example of our global affine flow estimator applied to a motion sequence can be seen in Figure 2. This sequence was generated synthetically from a real base image with divergent motion [4]. 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

$$\begin{aligned} u(x,y) &= \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} - x \\ v(x,y) &= \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} - y. \end{aligned} \tag{16}$$

Our projective formulation requires 8 parameters per frame, which is the same number as the quadratic flow field used in [5]. However, our formulation allows for arbitrarily large displacements, whereas [5] is based on instantaneous (infinitesimal) motion.

To compute the gradient and the Hessian, we proceed as before. We evaluate (16) at the $\hat{\mathbf{x}}_j$ to compute the spline control

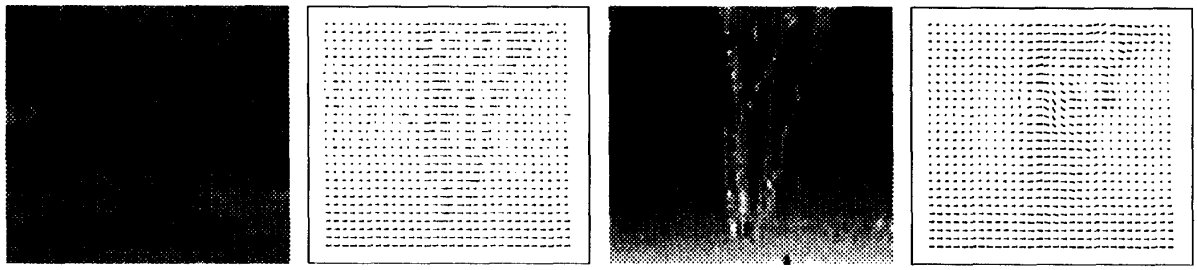

Figure 3: Example of 3D projective (rigid) motion estimation: intensity image, constrained rigid flow, recovered depth map, and unconstrained local flow (for comparison)

vertices $\hat{\mathbf{u}}_j$, and use the B-spline interpolants to compute the flow at each pixel. This flow is not exactly equivalent to the true projective flow defined in (16), 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. We then compute $\mathbf{T}_j = \frac{\partial \hat{\mathbf{u}}_j}{\partial \mathbf{m}}$ by differentiating (16) (the expression is similar to the one in (13) but requires two extra divisions), and then proceed to evaluate (14) and (15). The rightmost frame in Figure 2 shows the projective flow recovered from the tree image which had been perspectively distorted.

6 Rigid (direct) 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 [5, 10]. This has the disadvantage of only being valid for small motions, of requiring a calibrated camera, and of sensitivity problems with the depth estimates.[2]

Our approach is based on a *projective* formulation of structure from motion [7, 20]

$$u(x,y) = \frac{m_0 x + m_1 y + m_8\, z(x,y) + m_2}{m_6 x + m_7 y + m_{10} z(x,y) + 1} - x$$

$$v(x,y) = \frac{m_3 x + m_4 y + m_9\, z(x,y) + m_5}{m_6 x + m_7 y + m_{10} z(x,y) + 1} - y. \quad (17)$$

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.

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 (17) at the vertex locations $\hat{\mathbf{x}}_j$. We compute the gradients and Hessian with respect to the global motion parameters as before using $\mathbf{T}_j = \frac{\partial \hat{\mathbf{u}}_j}{\partial \mathbf{m}}$ from (17). The derivatives with respect to the depth estimates $\hat{z}_j$ and the Hessian matrix $\mathbf{A}_z$ for the $\mathbf{z} = \{\hat{z}_j\}$ parameters are similarly computed using the chain rule. Iterations in $\Delta\mathbf{m}$ and $\Delta\mathbf{z}$ are performed in alternate steps.

The performance of our rigid motion estimation algorithm on a sample image sequence is shown in Figure 3. As can be seen, the overall direction of motion (the *epipolar geometry*) has been recovered well, and the motion estimates look reasonable.[3] The computed depth map is shown in grayscale. The rightmost flow 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 [1, 8]. For example, the implementation of [8] described in [4] uses 21 images per estimate. Stereo matching techniques have also successfully used multiple images [12, 14]. Using large numbers of images not only improves the accuracy of the estimates through noise averaging, but it can also disambiguate between possible matches [14].

The extension of our local, global, and mixed motion models to multiple frames is straightforward. For local flow, we assume that we have linear flow (no acceleration), i.e.,

$$u_t(x,y) = s_t u_1(x,y) \quad \text{and} \quad v_t(x,y) = s_t v_1(x,y). \quad (18)$$

[2] [5] mitigate the Z sensitivity problem by estimating $1/Z(x, y)$ instead.

[3] We initialized the $\mathbf{m}$ vector in this example to a horizontal epipolar geometry.

We then minimize the overall cost function

$$E(\{u_i, v_i\}) = \sum_t \sum_i [I_t(x_i + s_t u_i, y_i + s_t v_i) - I_0(x_i, y_i)]^2. \tag{19}$$

This approach is similar to the *sum of sum of squared-distance* (SSSD) algorithm of [14], 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 $\hat{\mathbf{u}}_j$ in (5) becomes

$$\mathbf{g}_j = 2 \sum_t s_t \sum_i e_{ti} \mathbf{G}_{ti} w_{ij}$$

with e_{ti} and $\mathbf{G}_{ti}$ being the same as e_i and $\mathbf{G}_i$ with I_1 replaced by I_t. 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 motion estimation problem in the latter 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. 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. The derivatives with respect to the depth estimates $\hat{z}_j$ are computed by summing over all frames.

8 Experimental results

In this section, we demonstrate the performance of our algorithms on some of the standard motion sequences analyzed in [4]. Three of the images in these sequences have already been shown in Figures 1–3. Two other images are shown in Figure 4. We follow the organization of [4], presenting quantitative results on synthetically generated sequences first, followed by qualitative results on real motion sequences. A more complete set of experimental results is presented in [19].

Tables 1–3 give the quantitative results of our algorithms. In these tables, the top two rows are copied from [4]. The errors are reported as in [4], i.e., by converting 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.

From the nine algorithms in [4], we decided 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.

Technique	Avg. Error	Std. Dev.	Dens.
Lucas and Kanade (no thresh.)	2.47°	0.16°	100%
Fleet and Jepson ($\tau = 1.25$)	0.03°	0.01°	100%
local flow ($s = 2, L = 1, b = 0$)	0.17°	0.02°	100%
affine flow ($s = 2, L = 1, b = 0$)	0.13°	0.01°	100%

Table 1: Summary of **Sinusoid 1** results

Technique	Avg. Error	Std. Dev.	Dens.
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 ($n = 2$)	0.34°	0.36°	100%
local flow ($n = 3$)	0.28°	0.23°	100%
local flow ($n = 8$)	0.51°	0.15°	100%
affine flow	0.17°	0.12°	100%

Table 2: Summary of **Translating Tree** 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 (multiframe results are shown in Table 2).

As with many motion estimation algorithms, 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, (1 = consecutive frames)
- m [16] the size of the patch (width and height)
- L [3] the number of coarse-to-fine levels
- b [3] initial blurring (# of iterations of a box filter)

Unless mentioned otherwise, we used the default values shown in brackets above for the results in Tables 1–3. Bilinear interpolation was used for the flow fields.

The simplest motions to analyze are constant-translation sequences such as **Sinusoid 1**, which is a plaid formed by adding two sine waves translating at $(1.585, 0.863)$ pixels per frame [4]. Our local flow estimate for this sequence are very good for a two-frame method (Table 1) and improve slightly when the global affine model is used. For this sequence, we used a single level and no blurring, and took a $(s = 2)$ frame step for better results.

The **Translating Tree** sequence was generated using a real image (Figure 2) and synthetic (global) motion. Our results on this sequence (Table 2) 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 improvement from using additional frames is modest, and at some point the performance degrades due to the inability to escape from local

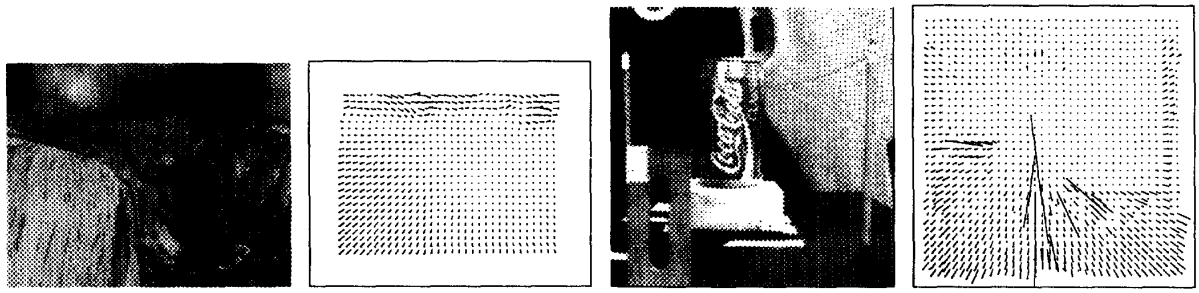

Figure 4: **Yosemite** sample image and flow (unthresholded). **NASA Sequence** and rigid flow estimates

Technique	Avg. Error	Std. Dev.	Dens.
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 ($s = 2, T_e = 3000$)	2.19°	5.86°	23.1%
local flow ($s = 2, T_e = 2000$)	3.06°	7.54°	39.6%
local flow, cropped ($s = 2$)	2.45°	3.05°	100%
rigid flow, cropped ($s = 2$)	3.77°	3.32°	100%

Table 3: Summary of **Yosemite** results

minima (see Section 9 for ideas on how to overcome this).

The final motion sequence for which quantitative results are available is **Yosemite** (Figure 4 and Table 3). 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 [4], even though we again only use two images. To produce the sparse flow estimates, we set a threshold T_e on the minimum eigenvalues of the local Hessian matrices $\mathbf{A}_{jj}$ interpolated over the whole grid (this selects areas where both components of the motion estimate are well determined). 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.

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 Figure 3 for both rigid and local (general) flow. The right half of Figure 4 shows the **NASA Sequence** in which the camera moves forward in a rigid scene (there is significant aliasing). The motion estimates are reasonable, except for some gross errors (huge vectors) which occur in the bright areas where the motion cannot be determined. The rightmost frame in Figure 1 shows the sparse flow computed for the **Rubik Cube** sequence (the dense flows are shown in the middle frame). The areas with texture and/or corners produce the most reliable flow estimates. Overall, these results are comparable or better than those shown in [4].

Much work remains to be done in the experimental evaluation of our algorithms. In addition to systematically studying the effects of the parameters n, s, m, L, and b (introduced previously), we plan to study the effects of different spline interpolation functions, regularization, preconditioning, and conjugate gradient descent, to see which of these would improve our results.

9 Discussion and Conclusions

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 local flow estimates which perform well in the presence of independently moving objects and large depth variations. 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. This is especially important when we analyze extended motion sequences, where local intensity patterns can deform significantly. Our technique can be viewed as a generalization of affine patch trackers [17] where the patch corners are stitched together over the whole image.

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.

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.

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.

Turning to global motion estimation, our motion model for planar surface flow can handle arbitrarily large motions and displacements, unlike the instantaneous model of [5]. Furthermore, our approach does not require the camera to be calibrated and can handle temporally-varying internal camera parameters. 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.

Overall, our experimental results suggest that our techniques are competitive in quality with the best currently available motion estimators, in some cases (as in the realistic Yosemite sequence) outperforming previously reported results.

9.1 Future work

To overcome the current problems with multiframe estimation, we plan to implement a strategy where only the first two images are initially matched, and subsequent images are added one at a time, with previous flow/motion estimates being used to initialize the system. To deal with more difficult scenes (e.g., with repetitive textures), a local search component [2] may have to be added.

We also plan to study hierarchical basis functions as an alternative to coarse-to-fine estimation. This approach has proven to be very effective in other vision problems such as surface reconstruction and shape from shading where smoothness or consistency constraints need to be propagated over large distances [18]. Finally, we plan to address the problems of discontinuities and occlusions, which must be resolved for any motion analysis system to be truly useful.

In terms of applications, we are currently using our global flow estimator to register multiple 2D images, e.g., to align successive microscope slice images or to composite pieces of flat scenes such as documents or whiteboards seen with a video camera. We plan to use our local/global model to extract 3D projective scene geometry from multiple images. We would also like to study the performance of our local motion estimator in extended motion sequences as a parallel feature tracker, i.e., by using only estimates with high local confidence. Finally, we would like to test our spline-based motion estimates as predictors 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. 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.

References

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Amer.*, A 2(2):284–299, February 1985.

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Intern. J. Comput. Vis.*, 2(3):283–310, January 1989.

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

[4] J. L. Barron *et al.* Performance of optical flow techniques. *Intern. J. Comput. Vis.*, 12(1):43–77, January 1994.

[5] J. R. Bergen *et al.* Hierarchical model-based motion estimation. In *ECCV'92*, pp. 237–252, Santa Margherita Liguere, Italy, May 1992.

[6] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. Comm.*, COM-31(4):532–540, April 1983.

[7] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *ECCV'92*, pp. 563–578, Santa Margherita Liguere, Italy, May 1992.

[8] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *Intern. J. Comput. Vis.*, 5:77–104, 1990.

[9] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artif. Intell.*, 17:185–203, 1981.

[10] B. K. P. Horn and E. J Weldon Jr. Direct methods for recovering motion. *Intern. J. Comput. Vis.*, 2(1):51–76, 1988.

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *IJCAI-81*, pp. 674–679, Vancouver, BC, 1981.

[12] L. H. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *Intern. J. Comput. Vis.*, 3:209–236, 1989.

[13] H.-H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artif. Intell.*, 33:299–324, 1987.

[14] M. Okutomi and T. Kanade. A multiple baseline stereo. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(4):353–363, April 1993.

[15] W. H. Press *et al. Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 2nd Edition, 1992.

[16] L. H. Quam. Hierarchical warp stereo. In *Image Understanding Workshop*, pp. 149–155, New Orleans, LA, December 1984.

[17] J. Rehg and A. Witkin. Visual tracking with deformation models. In *IEEE Intern. Conf. Robotics and Automation*, pp. 844–850, Sacramento, CA, April 1991.

[18] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(6):513–528, June 1990.

[19] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. Technical Report 94/1, Digital Equipment Corporation, Cambridge Research Lab, April 1994.

[20] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *J. Visual Commun. and Image Repres.*, 5(1):10–28, March 1994.

[21] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *Intern. J. Comput. Vis.*, 1:133–144, 1987.