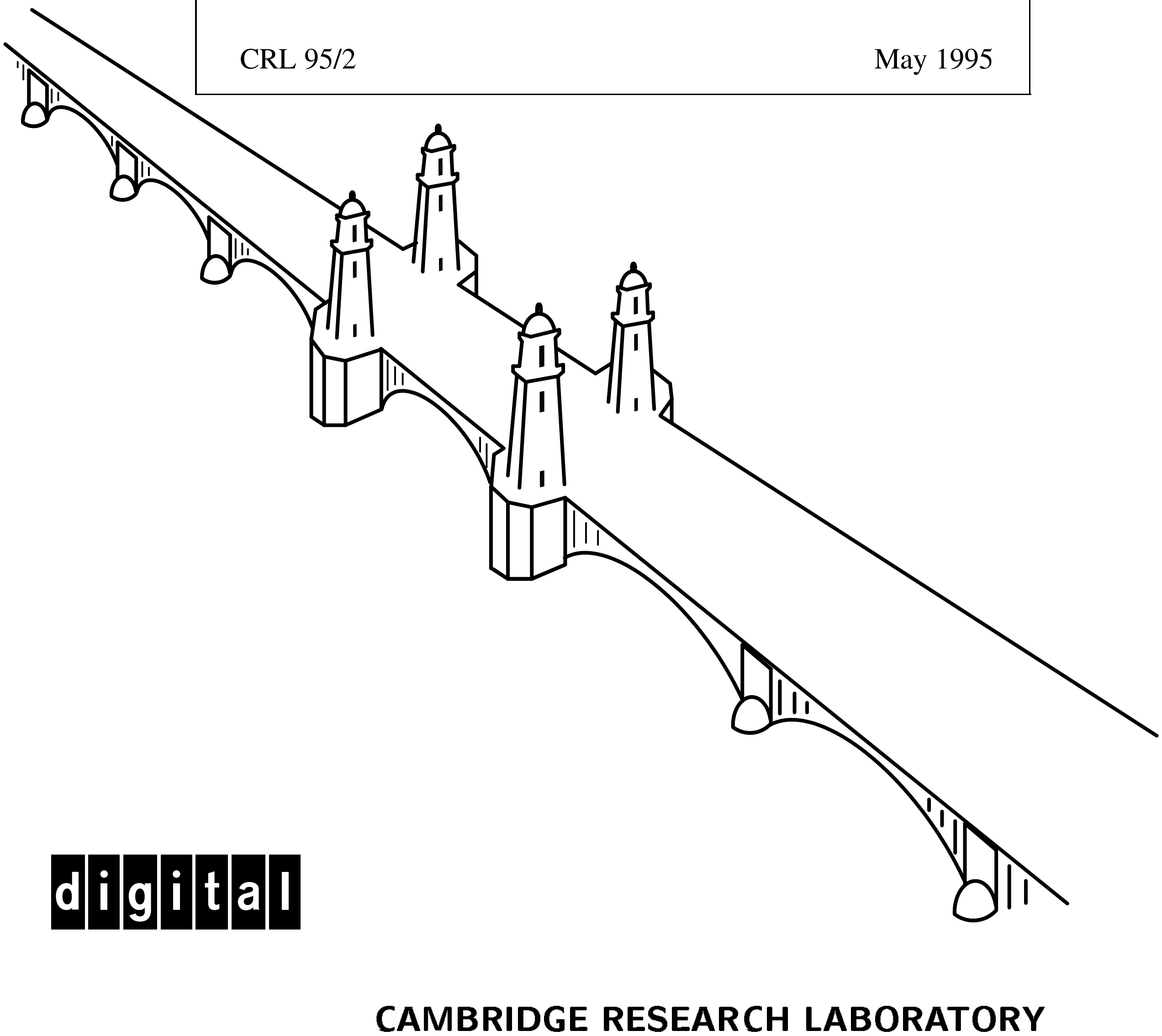
# A Parallel Feature Tracker for Extended Image Sequences


Richard Szeliski, Sing Bing Kang,
and Heung-Yeung Shum
Digital Equipment Corporation
Cambridge Research Lab



CRL 95/2 May 1995

digital


CAMBRIDGE RESEARCH LABORATORY
Technical Report Series

Digital Equipment Corporation has four research facilities: the Systems Research Center and the Western Research Laboratory, both in Palo Alto, California; the Paris Research Laboratory, in Paris; and the Cambridge Research Laboratory, in Cambridge, Massachusetts.

The Cambridge laboratory became operational in 1988 and is located at One Kendall Square, near MIT. CRL engages in computing research to extend the state of the computing art in areas likely to be important to Digital and its customers in future years. CRL's main focus is applications technology; that is, the creation of knowledge and tools useful for the preparation of important classes of applications.

CRL Technical Reports can be ordered by electronic mail. To receive instructions, send a message to one of the following addresses, with the word **help** in the Subject line:

| | |
|---|---|
| On Digital's EASYnet: | CRL::TECHREPORTS |
| On the Internet: | techreports@crl.dec.com |



digital™

Cambridge Research Laboratory
One Kendall Square
Cambridge, Massachusetts 02139

# A Parallel Feature Tracker for Extended Image Sequences

Richard Szeliski, Sing Bing Kang,
and Heung-Yeung Shum[1]
Digital Equipment Corporation
Cambridge Research Lab

CRL 95/2 May 1995

**Abstract**

This paper presents a feature tracker for long image sequences based on simultaneously estimating the motions and deformations of a collection of adjacent image patches. By sharing common corner nodes, the patches achieve greater stability than independent patch trackers. Modeling full bilinear deformations enables tracking in sequences which have large non-translational motions and/or foreshortening effects. We demonstrate the advantages of our technique with respect to previous algorithms using experimental results.

**Keywords:** motion analysis, multiframe feature tracking, affine patches


[1]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890

# Contents

# List of Figures

# 1 Introduction

Many tasks in computer vision and robotics require feature tracking, including tracking objects for grasping, tracking people in surveillance work, automatic vehicle convoying, and body tracking for video-based user interfaces. Feature tracking is also used extensively for the purpose of recovering structure from motion.

Much research in computer vision has been dedicated to developing robust and efficient means for tracking features in sequences of images. The current emphasis placed on long image sequences [Tomasi and Kanade, 1992] raises some new and interesting issues. One such issue is reliable tracking despite significant object image deformations due to object or camera motion and foreshortening effects. Another issue is the desire to track features whenever possible, namely at locations of high texture. A good feature tracker should not be restricted to track just features that fit specific templates such as corners.

In this paper, we develop a new algorithm for tracking features over long image sequences. Our algorithm is based on the principle of local patch correlation with possible bilinear deformations (a generalization of [Lucas and Kanade, 1981; Rehg and Witkin, 1991; Tomasi and Kanade, 1992; Shi and Tomasi, 1994]). In our tracker, adjacent patches share common nodes for better stability. The confidence of the recovered feature tracks is subsequently determined using local Hessian and squared error criteria [Anandan, 1989; Shi and Tomasi, 1994].

The structure of our paper is as follows. Section 2 reviews previous work. Section 3 describes our spline-based image registration algorithm. Section 4 describes how we assign confidence measures to various tracks using local Hessian and squared error criteria. Section 5 discusses the problem of maintaining and re-initializing tracks over long image sequences. Section 6 describes our experimental results, including a quantitative comparison of algorithms. We close with a discussion of the advantages of our technique and ideas for future work.

# 2 Previous work

Feature tracking has a long history both in computer vision and photogrammetry (see [Shi and Tomasi, 1994] for a recent review). Many techniques rely on finding specific kinds of features in the images, e.g., corners, and then finding correspondences between such features. A second class of techniques uses correlation, and thus has no preconceptions on what constitutes a feature. Our work falls into this second category.

A basic correlation-based feature tracker chooses a patch of pixels in the first image, and then searches for a corresponding patch in the second image either by maximizing the correlation

$$E(u,v) = \sum_{k,l} I_1(x+u+k, y+v+l) I_0(x+k, y+l) \tag{1}$$

or by minimizing the *sum of squared differences* (SSD)

$$E(u,v) = \sum_{k,l} [I_1(x+u+k, y+v+l) - I_0(x+k, y+l)]^2 \tag{2}$$

These approaches have been extensively studied and used. See [Ryan *et al.*, 1980; Burt *et al.*, 1982; Horn, 1983; Opitz, 1983; Wood, 1983] for some comparative analyses, and [Förstner, 1987] for a review of statistical aspects of photogrammetry.

To obtain sub-pixel registration accuracies, a number of possible extension to the basic search technique can be used [Tian and Huhns, 1986]: interpolating the correlation surface $E(u,v)$, interpolating the intensities, the differential method [Huang, 1981; Lucas and Kanade, 1981], and phase correlation [Kuglin and Hines, 1975]. The differential method uses a local Taylor series expansion of the intensity function to compute a sub-pixel improvement to the displacement estimate

$$\begin{aligned} E(u+\Delta u, v+\Delta v) &= \sum_{k,l} [I_1(x+u+\Delta u+k, y+v+\Delta v+l) - I_0(x+k, y+l)]^2 \\ &\approx \sum_{k,l} [I_1(x+u+k, y+v+l) + \nabla I_1 \cdot (\Delta u, \Delta v)^T - I_0(x+k, y+l)]^2 \\ &= \sum_{k,l} [\nabla I_1 \cdot (\Delta u, \Delta v)^T]^2 + \sum_{k,l} [\nabla I_1 \cdot (\Delta u, \Delta v)^T] e_{k,l} + E(u,v), \end{aligned}$$

where $\nabla I_1 = (I_x, I_y) = \nabla I_1(x+u+k, y+v+l)$ is the intensity gradient and $e_{k,l}$ is the term inside the brackets in (2), i.e., the intensity error at each pixel. Minimizing w.r.t. $(\Delta u, \Delta v)$, we obtain a $2 \times 2$ system of equations

$$\begin{bmatrix} \sum_{k,l} I_x^2 & \sum_{k,l} I_x I_y \\ \sum_{k,l} I_x I_y & \sum_{k,l} I_y^2 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} \sum_{k,l} I_x e_{k,l} \\ \sum_{k,l} I_y e_{k,l} \end{bmatrix}. \tag{3}$$

The matrix on the left hand side is often referred to as the *Hessian* of the system, and encodes the relative certainties in the flow estimates.[1] For tracking long sequences, a combination of search-based correlation and the differential method can be used [Zheng and Chellappa, 1992].

[1] More formally, the covariance matrix of the flow estimate is proportional to the inverse Hessian [Szeliski, 1989].

The basic correlation technique works well when the motion is mostly (locally) translational between frames, and when there are no large photometric variations. A more general solution can be obtained by assuming a locally *affine* model for the motion [Fuh and Maragos, 1991; Rehg and Witkin, 1991]. The differential method then corresponds to solving a $6 \times 6$ system of equations in the unknown parameters of the affine motion model [Rehg and Witkin, 1991]. It is also possible to model bias and gain variations in the intensity simultaneously with estimating the motion [Fuh and Maragos, 1991]. However, because of the increased number of unknowns, either fairly large patches must be used [Rehg and Witkin, 1991], or a more restricted model (scaled rotation) must be used [Fuh and Maragos, 1991]. Shi and Tomasi [Shi and Tomasi, 1994] also examined affine patch trackers, but concluded that in practice they were not as stable as pure translational trackers.

Our own previous work in motion estimation (reviewed in the next section) uses a spline-based description of the motion field [Szeliski and Coughlan, 1994]. This can be viewed as running a series of patch-based trackers in parallel, with a more complex local motion model (bilinear instead of affine), and the additional constraint that the motion estimates be continuous across patches. As we will demonstrate in this paper, this motion continuity constraint makes individual tracking results more reliable.

# 3 Spline-based image registration

Our algorithm for multi-frame feature tracking first computes a dense estimate of motion using direct image registration, and then selects certain points with high confidence as features to be tracked. In our framework, we register a new image $I_1$ to an initial *base image* $I_0$ using a sum of squared differences formula

$$E(\{u_i, v_i\}) = \sum_i [I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)]^2, \quad (4)$$

where the $\{u_i, v_i\}$ are the per-pixel *flow* estimates.[2]

Rather than representing the flow estimates $\{u_i, v_i\}$ as completely independent quantities (and thus having an underconstrained problem), we represent them using 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*

[2]The basic SSD algorithm can be made more robust to photometric variation by adding bias and gain parameters, and more robust to outliners using techniques from robust statistics (see [Szeliski and Shum, 1995a]).

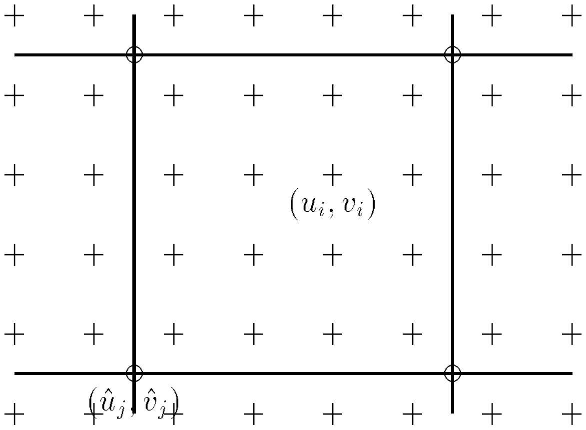



Figure 1: Displacement spline: the spline control vertices $\{(\hat{u}_j, \hat{v}_j)\}$ are shown as circles (○) and the pixel displacements $\{(u_i, v_i)\}$ are shown as pluses (+).

*grid* (Figure 1). The value for the displacement at a pixel $i$ can be written as

$$u(x_i, y_i) = \sum_j \hat{u}_j B_j(x_i, y_i) \quad \text{or} \quad u_i = \sum_j \hat{u}_j w_{ij}, \tag{5}$$

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 $(u_i, v_i)$ 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. We also use bilinear basis functions, $B_j(x, y) = \max((1 - |x - \hat{x}_j|/m)(1 - |y - \hat{y}_j|/m), 0)$ (see [Szeliski and Coughlan, 1994] for a discussion of other possible bases).

Our spline-based image registration algorithm has two major advantages over traditional motion estimators which use overlapping correlation windows [Anandan, 1989]. First, the overall amount of computation required is reduced by a factor of $m^2/4$, since each pixel only contributes to the flow estimates of 4 neighboring spline vertices (see below). Second, each patch in our spline-based approach can undergo large (bilinear) deformations, whereas traditional methods assume a pure locally translational model, making it impractical to match subsequent images in an extended sequence to a single base image.

[3] In the remainder of the paper, we will use indices $i$ for pixels and $j$ for spline control vertices.

## 3.1 Function minimization

To recover the local spline-based flow parameters, we need to minimize the cost function (4) 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 (4) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$,

$$\begin{aligned} g_j^u &\equiv \frac{\partial E}{\partial \hat{u}_j} = 2\sum_i e_i I_{xi} w_{ij} \\ g_j^v &\equiv \frac{\partial E}{\partial \hat{v}_j} = 2\sum_i e_i I_{yi} w_{ij}, \end{aligned} \tag{6}$$

where

$$e_i = I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i) \tag{7}$$

is the intensity error at pixel $i$,

$$(I_{xi}, I_{yi}) = \nabla I_1(x_i + u_i, y_i + v_i) \tag{8}$$

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 (5). Algorithmically, we compute the above gradients by first forming the displacement vector for each pixel $(u_i, v_i)$ using (5), 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 [Szeliski and Coughlan, 1994].

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

$$\begin{aligned} a_{jk}^{uu} &= 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{u}_k} = 2\sum_i w_{ij} w_{ik} I_{xi}^2 \\ a_{jk}^{uv} = a_{jk}^{vu} &= 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{v}_k} = 2\sum_i w_{ij} w_{ik} I_{xi} I_{yi} \\ a_{jk}^{vv} &= 2\sum_i \frac{\partial e_i}{\partial \hat{v}_j}\frac{\partial e_i}{\partial \hat{v}_k} = 2\sum_i w_{ij} w_{ik} I_{yi}^2. \end{aligned} \tag{9}$$

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

The $2 \times 2$ sub-matrix $\mathbf{A}_j$ 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 (for $w_{ij} = 1$) is identical to the Hessian matrix used in the differential method, i.e., the matrix appearing on the left-hand side of (3). The overall $\mathbf{A}$ matrix is a sparse multi-banded block-diagonal matrix, i.e., sub-blocks containing $a_{jk}$ will be non-zero only if vertices $j$ and $k$ both influence some common patch of pixels. In our current implementation, we never explicitly compute the off-diagonal blocks (see below).

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

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

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 $\lambda$ is a stabilization factor which varies over time [Press *et al.*, 1992]. To solve this large, sparse system of linear equations, we use preconditioned gradient descent

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

where $\mathbf{B} = \hat{\mathbf{A}} + \lambda\mathbf{I}$, and $\hat{\mathbf{A}} = \text{block\_diag}(\mathbf{A})$ is the set of $2 \times 2$ block diagonal matrices $\mathbf{A}_j$, and $\mathbf{d} = \mathbf{B}^{-1}\mathrm{g}$ is called the *preconditioned residual* or *direction* vector. The update rule is very close to that used in the differential method [Lucas, 1984], with the following differences:

1. the equations for computing the g and $\mathbf{A}$ are different (based on spline interpolation)

2. an additional diagonal term $\lambda$ is added for stability[4]

3. there is a step size $\alpha$.

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

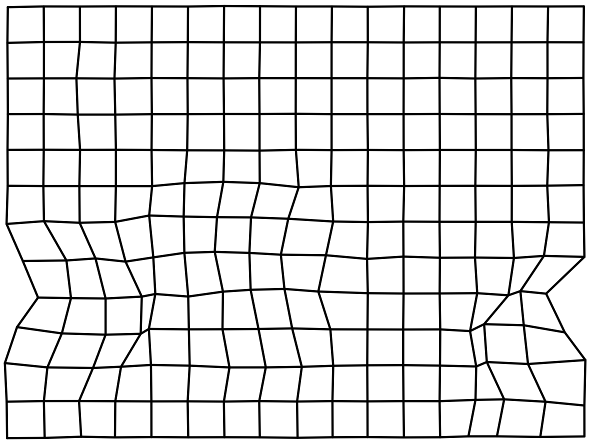
$$\Delta E(\alpha\mathbf{d}) \approx \alpha^2\mathbf{d}^T\mathbf{A}\mathbf{d} - 2\alpha\mathbf{d}^T\mathbf{g},$$

i.e., by setting $\alpha = (\mathbf{d} \cdot \mathbf{g})/(\mathbf{d}^T\mathbf{A}\mathbf{d})$. See [Szeliski and Coughlan, 1994] for more details on our algorithm implementation.

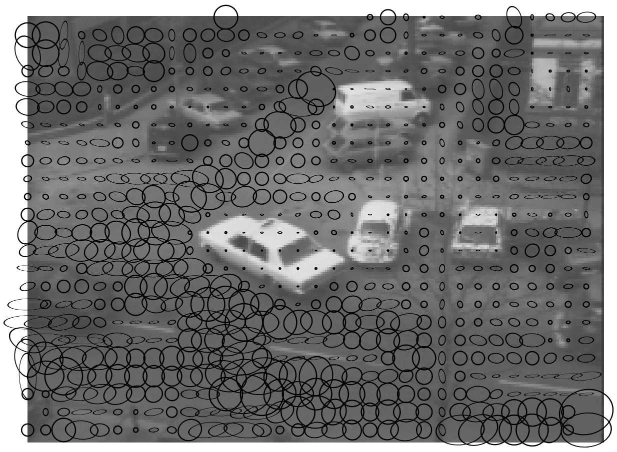
To handle larger displacements, we run our algorithm in a coarse-to-fine (hierarchical) fashion. A Gaussian image pyramid is first computed using an iterated 3-point filter [Burt and Adelson,

[4] A Bayesian justification can be found in [Simoncelli *et al.*, 1991], and additional possible local weightings in [Lucas, 1984, p. 20].

(a)



(b)



(c)

(d)

Figure 2: Spline-based image registration example (taxi sequence): (a) frame 0, (b) deformed spline control grid, (f) uncertainty ellipses, (g) top 50 tracks.

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). The result of applying our spline-based image registration algorithm to an image sequence with three independently moving cars is shown in Figure 2, with Figure 2b showing the shape of the deformed spline control grid.

# 4 Parallel feature tracking

To convert our spline-based image registration algorithm into a parallel feature tracker, we simply associate a scalar confidence value with each of the motion estimates $(\hat{u}_j, \hat{v}_j)$ and threshold out estimates with low confidence.[5] Two sources of confidence information are the structure of the local Hessian matrix $\mathbf{A}_j$ and the summed squared error within each spline patch.

To exploit the local Hessian information, we note that the eigenvectors and eigenvalues of $\mathbf{A}_j$ encode the directions of least and greatest certainty in the motion and their respective magnitudes. More formally, it can be shown that under small Gaussian noise, the inverse eigenvalues are proportional to the variance in the motion estimates along these two directions [Matthies *et al.*, 1989; Szeliski, 1989]. For most tracking applications, e.g., for structure from motion, good positioning in all directions is desired. We therefore use the inverse of the minimum eigenvalue as primary measure of feature uncertainty (as in [Shi and Tomasi, 1994]). Figure 2c shows the uncertainties in potential track positions as ellipses of various sizes (this display idea is taken from [Xiong and Shafer, 1995]). Regions with small circles indicate tracks with good positional accuracy, while elongated ellipses indicate the presence of the *aperture problem* (weak positional certainty in one direction). Features at locations with big uncertainty ellipses are poor candidates for tracking. Figure 2d shows the 50 best feature tracks selected on the basis of the minimum Hessian eigenvalue in the second frame.

The squared error in a patch is also a strong indicator of the quality of a track [Shi and Tomasi, 1994]. In particular, tracks which become occluded, or where large photometric effects are present, will result in an increased error score. We therefore use the patch error to monitor the quality of selected feature tracks, and terminate tracking when the error exceeds a threshold. Tracks are initially selected by choosing the tracks with the largest minimum eigenvalues, either choosing a predetermined number or a predetermined threshold on the values (see Section 6).

# 5 Tracking through long sequences

When tracking features through more than two images, e.g., for multi-frame structure from motion, we have two choices. We can either match successive pairs of images keeping track of the

[5] Depending on the application, we may also want to feed the complete set of tracks and confidences into the next stage, e.g., into a certainty-weighted structure from motion algorithm [Szeliski and Kang, 1994].

feature positions to sub-pixel position, or we can try to match all images to the initial (base) image. The first approach, taken by Shi and Tomasi [Shi and Tomasi, 1994], has the advantage that since the inter-frame motion is reduced (at least for a smooth sequence), a locally translational model of motion may be adequate. In our work, we have taken the second approach, i.e., we register all images to the base image. This has the advantage that small errors in tracking do not accumulate over time (see Section 6). A potential disadvantage is that slow variations in photometry (e.g., gradual brightening) are not as easily accommodated.

Matching all images to a base image means that the amount of inter-frame motion can be extremely large. For this reason, we use *motion prediction* to initialize the registration algorithm for each subsequent frame. We have studied two different prediction methods: linear flow, $\mathbf{u}_t = \frac{t}{t-1}\mathbf{u}_{t-1}$, and linear acceleration, $\mathbf{u}_t = \mathbf{u}_{t-1} + (\mathbf{u}_{t-1} - \mathbf{u}_{t-2})$. In practice, the second method (which can be used for $t > 2$) performs better, e.g., it can handle rotational motion, while linear flow cannot.

Another potential limitation to matching the first image is that there is no possibility for starting new tracks, e.g., in disoccluded regions. We overcome this limitation by periodically choosing a new frame as a base, while maintaining the previous tracker until most of the tracks have disappeared (due to excessive squared intensity errors). While this may result in more tracks than a pairwise tracker, the total amount of computation per track is comparable.

# 6 Experimental results

To determine the performance of our tracking algorithm, we tested it on a number of standard and customized image sequences. In this section, we present comparative results with a region-based tracker used in our previous structure from motion research [Szeliski and Kang, 1994], and with the simple patch based tracker described in [Shi and Tomasi, 1994].[6]

## 6.1 Simulation results

We applied our tracker to six synthetic motion sequences and compared its performance with that of Shi-Tomasi's tracker. Each frame in the first five sequences is derived from the first frame using

[6] We have not yet implemented the affine error metric which is used in [Shi and Tomasi, 1994] to monitor the quality of the tracks.

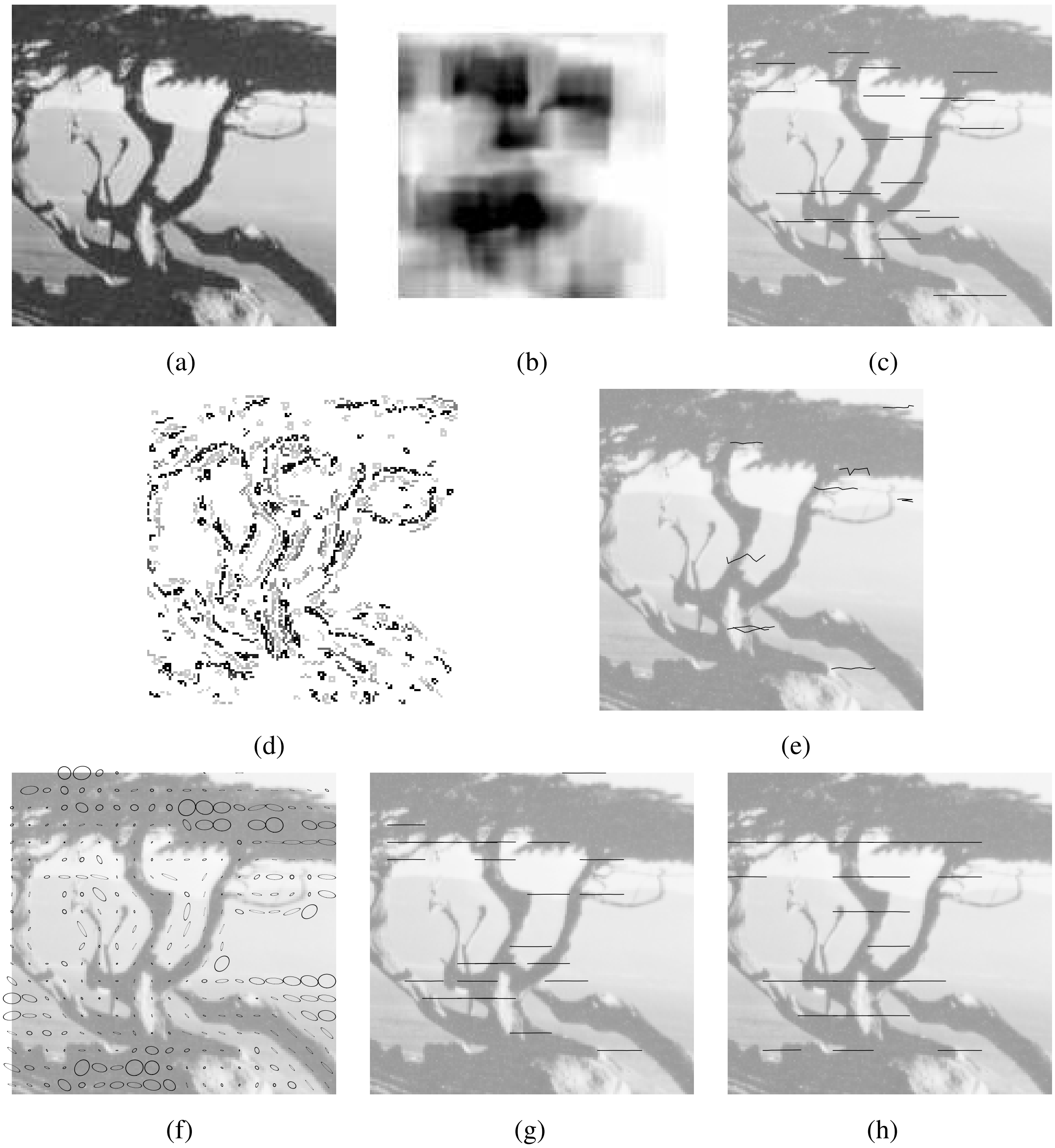
a known affine transformation and bilinear intensity interpolation. The first five sequences used to test our tracker are the translating tree (10 frames, Figure 3), the diverging tree (10 frames, Figure 4), the diverging tree with $\sigma = 10$ additive Gaussian noise (10 frames), a rotating tree (10 frames, Figure 5), and a diverging Yosemite (10 frames, Figure 6).[7] The sixth sequence used to test our tracker was created using Rayshade, which is a program for creating ray-traced color images [Kolb, 1994]. Its input is a text file that describes the properties of the camera, light source/s, objects (primitives such as spheres, cylinders, cones, and patches), and atmosphere in the scene.

The best 25 features are automatically picked for Shi-Tomasi's tracker to track; these features are chosen based on the minimum eigenvalue of the local Hessian, which is an indication of texturedness. The feature window size is 25x25 pixels. Each feature is separated from another by at least half the dimension of the feature window (i.e., 12 pixels). Parts of the feature tracks that fall at or over the image boundary are automatically ignored. Part (b) of each figure shows the minimum eigenvalue distribution, with darker regions indicating higher eigenvalues. Based on this distribution, 25 point features are chosen and subsequently tracked (part (c)).

We also present result for a tracker based on the monotonicity operator [Kories and Zimmermann, 1986]. This region-based tracker was used in our previous structure from motion research [Szeliski and Kang, 1994]. The monotonicity operator computes the number of neighboring pixels whose intensity is less than that of the central pixel and therefore maps each pixel into one of nine classes. Pixels of the same class with the same vicinity tend to form blobs which are used as features for tracking. As in [Kories and Zimmermann, 1986], the image is first bandpass filtered. We also impose a deadband of a few pixels to reduce the effects of noise [Szeliski and Kang, 1994]. Part (d) of each figure shows the monotonicity image, while part (e) shows the resulting tracks (all tracks are shown since there is no easy way to select the 25 best).

For our new tracker, we use the minimum eigenvalue in choosing the best 25 features to track, and the pixel match errors to determine the valid portion of each track. The uncertainty ellipse distribution for the rotating tree sequence is shown in part (f) of each Figure, while the selected subtracks are shown in parts (g) and (h) (for patch sizes of 8 and 16). Inspecting Figures 3 through 7, we see that both our tracker and the Shi-Tomasi tracker produce much better results than the monotonicity operator, which is therefore left out of the quantitative experiments which follow. The shape of the minimum eigenvalue surface and the shapes of the uncertainty ellipses are interesting,

[7] This is *not* the same texture-mapped Yosemite sequence as used in [Barron *et al.*, 1994].



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3: Translating tree sequence: (a) frame 0, (b) minimum eigenvalues, (c) Shi-Tomasi tracker, (d) result of monotonicity operator, (e) monotonicity tracker, (f) uncertainty ellipses, (g) spline-based tracker ($m = 8$), (h) spline-based tracker ($m = 16$).

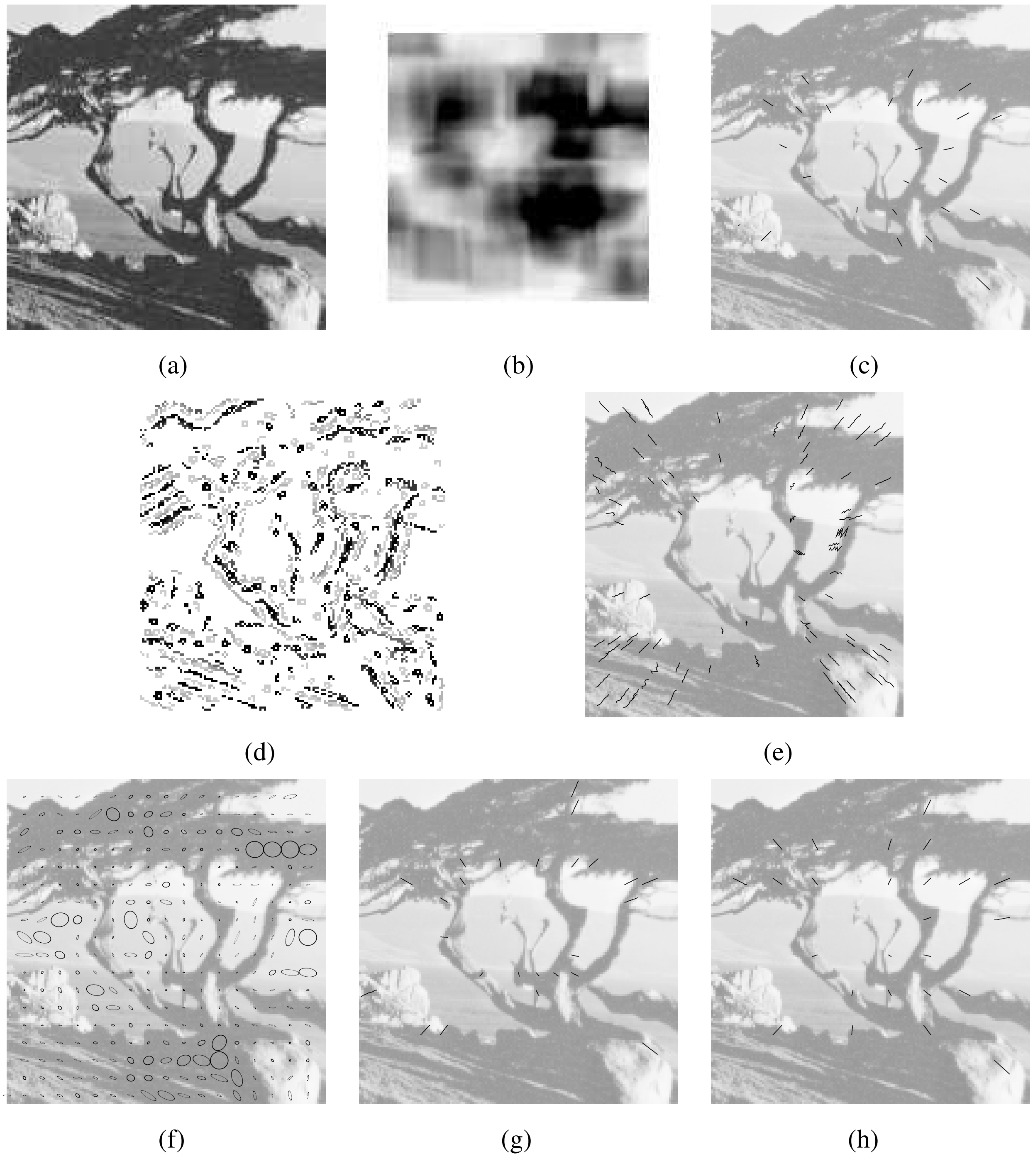


Figure 4: Diverging tree sequence: (a) frame 0, (b) minimum eigenvalues, (c) Shi-Tomasi tracker, (d) result of monotonicity operator, (e) monotonicity tracker, (f) uncertainty ellipses, (g) spline-based tracker ($m = 8$), (h) spline-based tracker ($m = 16$).

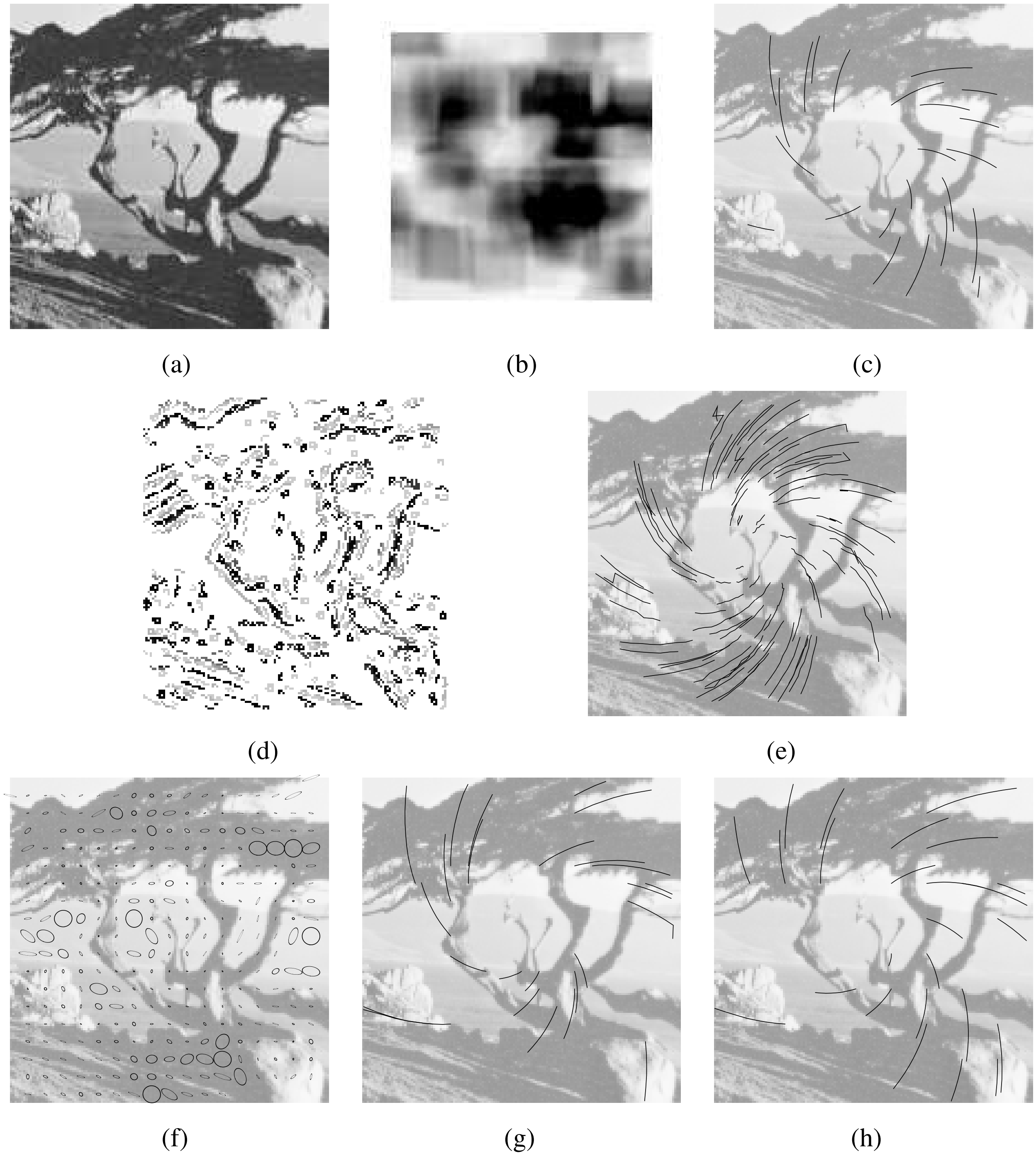


Figure 5: Rotating tree sequence: (a) frame 0, (b) minimum eigenvalues, (c) Shi-Tomasi tracker, (d) result of monotonicity operator, (e) monotonicity tracker, (f) uncertainty ellipses, (g) spline-based tracker ($m = 8$), (h) spline-based tracker ($m = 16$). The amount of rotation is about $2.7°$ per frame.

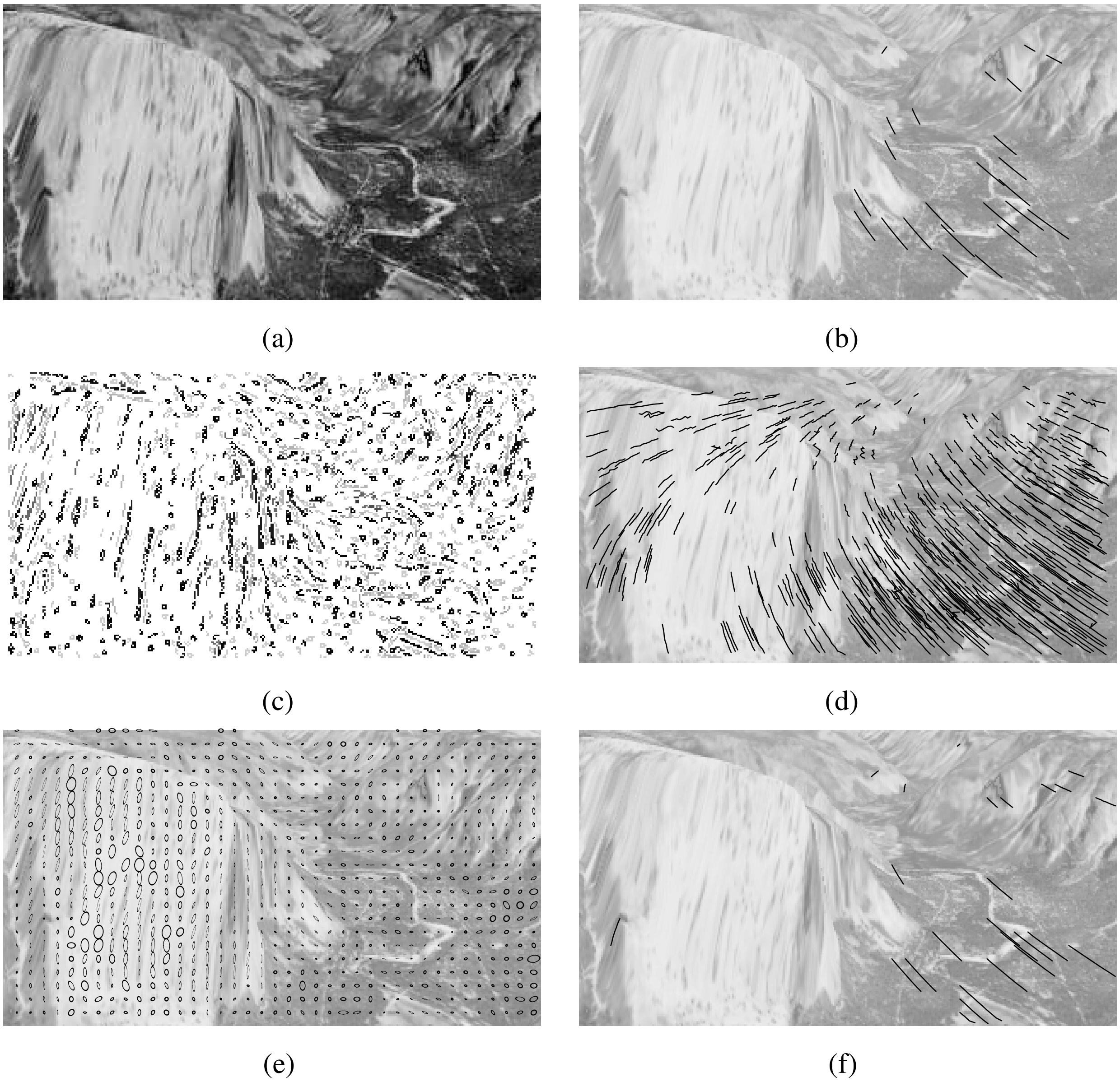
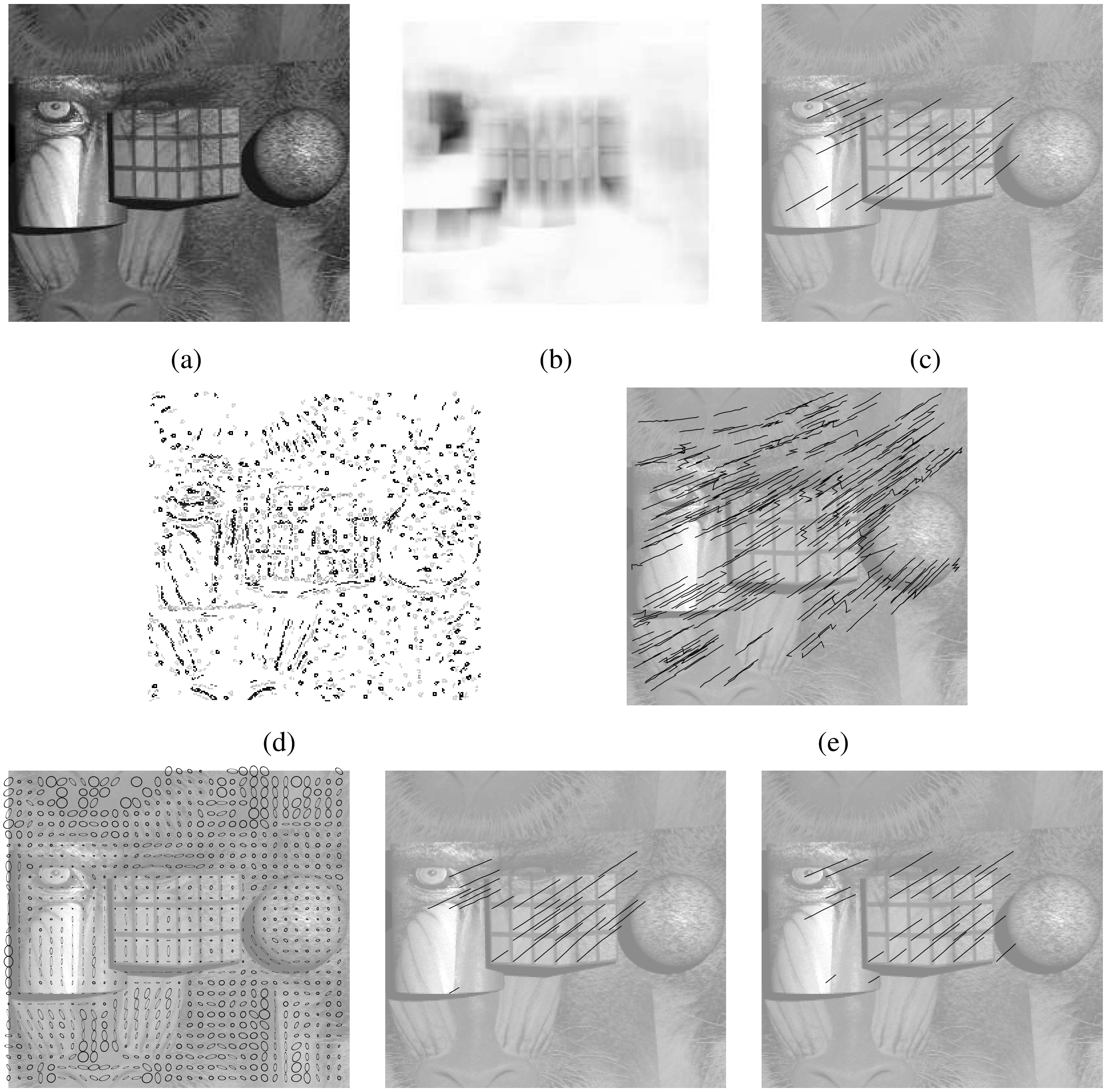


Figure 6: Yosemite (synthetic) sequence: (a) frame 0, (b) Shi-Tomasi tracker, (c) result of monotonicity operator, (d) monotonicity tracker, (e) uncertainty ellipses, (f) spline-based tracker ($m = 8$)



(a) (b) (c)

(d) (e)

(f) (g) (h)

Figure 7: Rayshade generated sequence: (a) frame 0, (b) minimum eigenvalues, (c) Shi-Tomasi tracker, (d) result of monotonicity operator, (e) monotonicity tracker, (f) uncertainty ellipses, (g) spline-based tracker ($m = 8$), (h) spline-based tracker ($m = 16$)

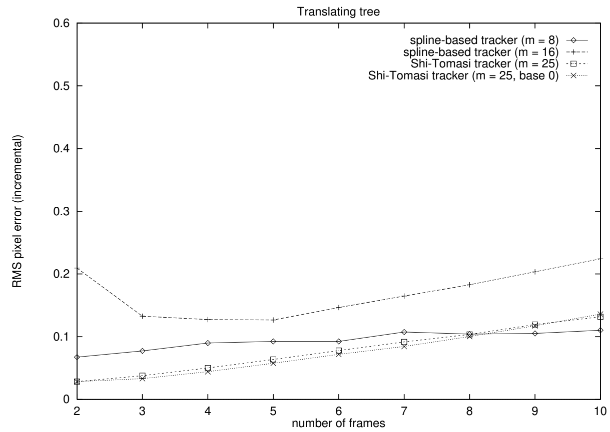
and correlate well with our intuitions of where good feature tracking might occur.

To obtain a more quantitative result, we computed the RMS pixel error in our trackers using the known motion, and plotted the results in Figure 8. To obtain a more robust estimate of tracker performance, i.e. to remove the influence of gross errors, we also computed the median of absolute errors, as shown in Figure 9. While the error statistics look more noisy, they are actually *much* lower overall (the figure axes have been scaled down by $\frac{1}{4}$ from Figure 8).

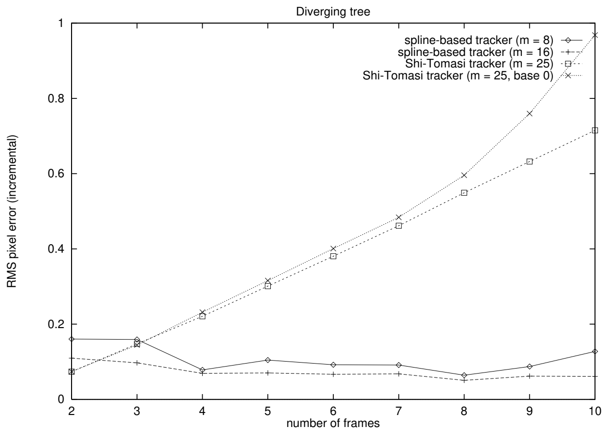
The results for the Shi-Tomasi tracker generally exhibit a drift in tracking as shown by the increasing error with the number of frames. This effect can be attributed to the rounding of template position during matching, despite the subpixel interframe motion estimation. This causes the estimation of the subsequent interframe motion to be that of a slightly different location. The drift in the Shi-Tomasi tracker for the case of the translating tree sequence is very small because the motion vector is approximately constant (less than one pixel of motion difference across the whole image).

Because our tracker matches the current frame with the *first* frame, the drift problem does not seem to occur. This is evident from the approximately constant pixel errors with increasing number of frames. To see if the Shi-Tomasi tracker would also benefit from matching to the first frame, we implemented this variant of the algorithm (labeled as "base 0" and shown as squares in the figures). One might expect that this would improve the performance, at least until the distortions in the templates become too large. Instead, the same accumulating error is seen as before, with the failure point of the algorithm depending on the rapidity of template distortion (about frame 8 or 9 for the diverging image sequences, and frame 2 for the rotating tree). This seems to indicate that the error in tracking is proportional to the distortions in the template.

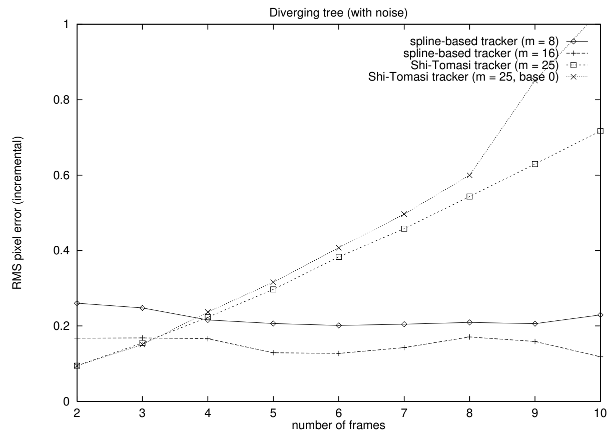
To test the robustness of the trackers to noise, we ran the same experiments at noise levels of $\sigma = 5, 10, 15$, and $20$ (for images whose dynamic range is $[0 \ldots 255]$). Both the Shi-Tomasi algorithm and the spline-based tracker were quite robust against this kind of noise, as evidenced by Figures 8c and 9c, which shows the results for $\sigma = 10$. In another experiment, we also tried limiting the starting locations for the Shi-Tomasi tracker to the same grid as our $m = 16$ spline-based tracker, to see if their freedom to choose optimal tracking locations was crucial to their performance. Our experiments (not shown for clarity) indicated that this had very little effect on the quality of the tracks.



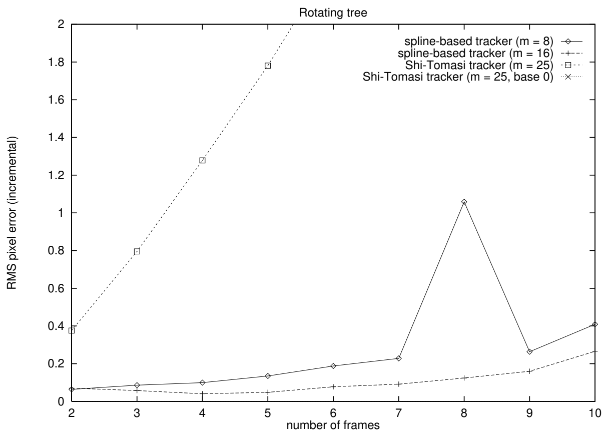

(a)



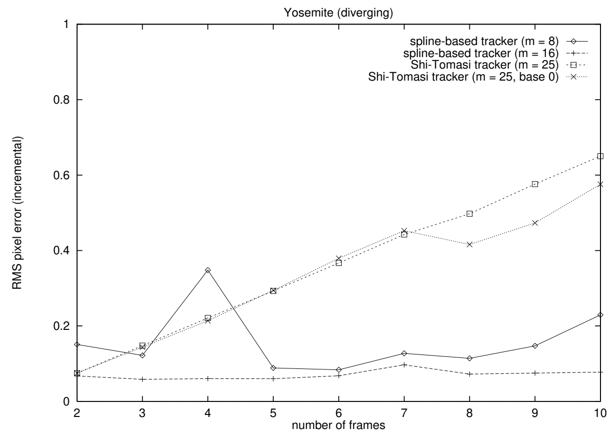

(b)



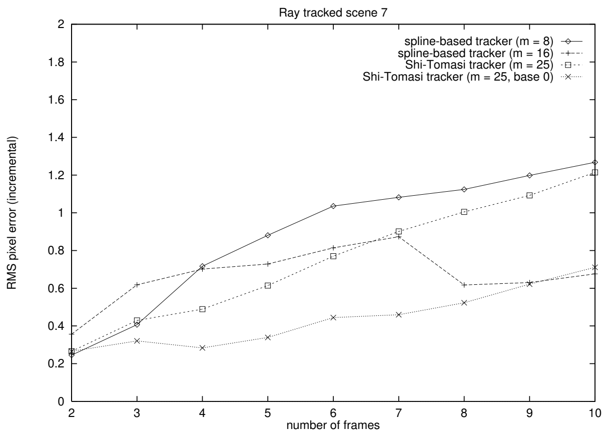

(c)




(d)




(e)




(f)

Figure 8: Comparison of RMS pixel error between the spline-based and Shi-Tomasi trackers: (a) translating tree sequence, (b) diverging tree sequence, (c) diverging tree sequence with $\sigma = 10$ noise, (d) rotating tree sequence, (e) yosemite sequence, (f) ray-traced sequence.

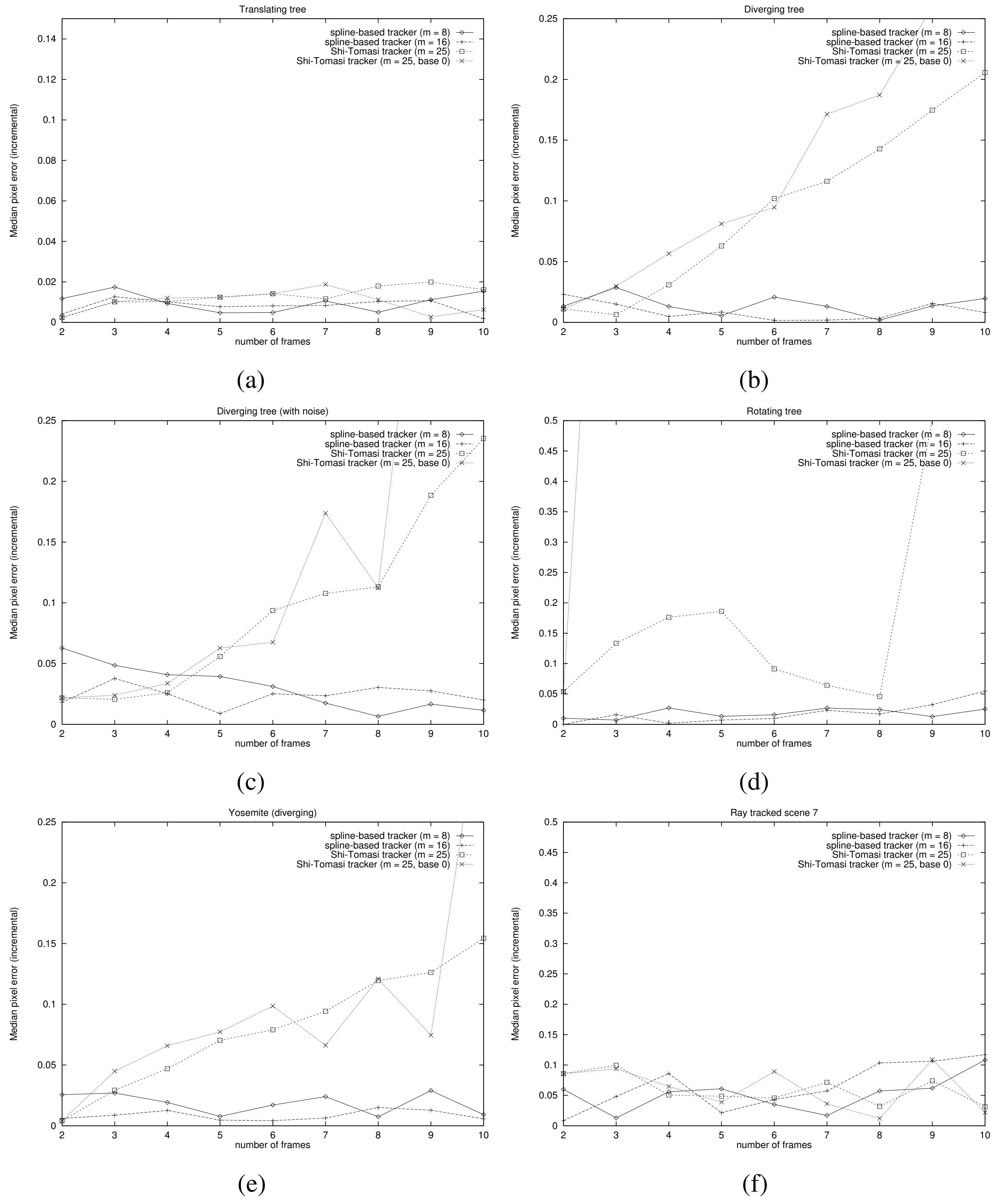



Figure 9: Comparison of median pixel error between the spline-based and Shi-Tomasi trackers: (a) translating tree sequence, (b) diverging tree sequence, (c) diverging tree sequence with $\sigma = 10$ noise, (d) rotating tree sequence, (e) yosemite sequence, (f) ray-traced sequence.

## 6.2 Results using a real image sequence

We have also applied the trackers to a real object sequence and recovered the object structure by applying an iterative non-linear least-squares structure-from-motion algorithm on the tracks [Szeliski and Kang, 1994]. The sequence is that of a rotating cube (Figure 10a). The recovered 3-D feature points using the tracks from Shi-Tomasi's tracker are shown in Figure 10f. We have also used our tracker on the rotating cube sequence. The uncertainty ellipse distribution for the sequence is shown in Figure 10d while the filtered tracks are shown in Figure 10e. The recovered 3-D feature points using the tracks from our tracker are shown in Figure 10g. As can be seen from these figures, the structure estimates computed from our new feature tracker are less noisy.

# 7 Discussion and Conclusions

This paper has described our spline-based tracker, which is based on the principle of local patch correlation with bilinear deformations. By sharing common corner nodes, the patches achieve greater stability than independent patch trackers. Modeling full bilinear deformations enables tracking in sequences which have significant non-translational motions and/or foreshortening effects.

We compared the performance of our spline-based tracker with Shi and Tomasi's tracker, which we consider to be one of the most robust and accurate trackers to date. Using simulated image sequences with theoretically known feature motions, we have found that the spline-based tracker performs better in terms of pixel error accumulation as compared to the Shi-Tomasi tracker. The deficiencies in their tracker seem to stem from template position rounding effects during successive interframe matching, and also from errors arising from the template distortion. Neither of these effects is present in our spline-based tracker.

To deal with the local minima which can trap our gradient descent technique, we are adding an optional exhaustive search component to our algorithm. At the beginning of each set of iterations, e.g., after inter-level transfers in the coarse to fine algorithm, or after splitting in the quadtree spline estimator, we search around the current $(u, v)$ estimate by trying a discrete set of nearby $(u, v)$ values (as in SSD algorithms [Anandan, 1989]). However, because we must maintain spline continuity, we cannot make the selection of best motion estimate for each patch independently. Instead, we average the motion estimates of neighboring patches to determine the motion of each spline control vertex.

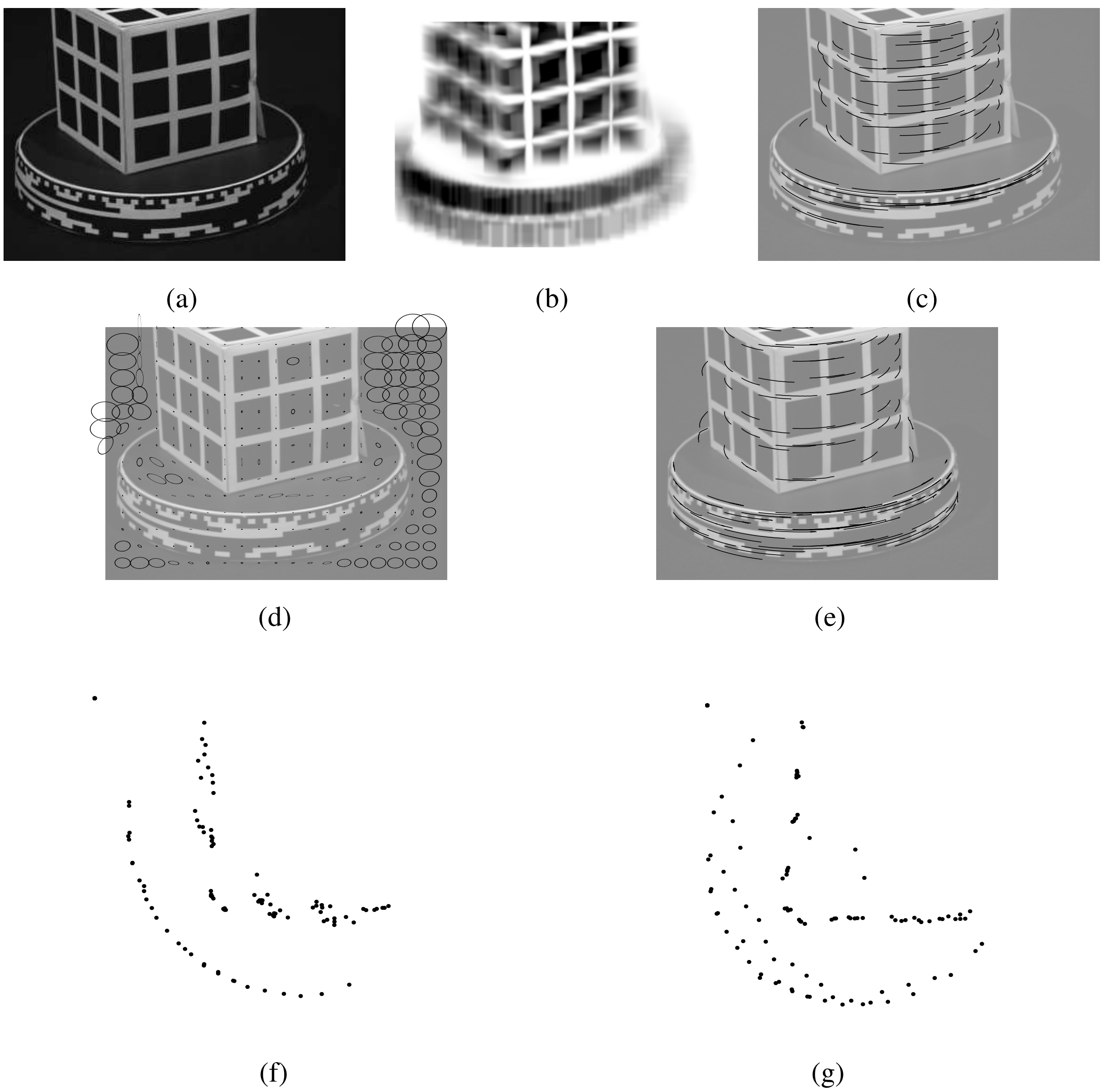



Figure 10: Cal-cube sequence: (a) first frame of cal-cube sequence, (b) distribution of minimum eigenvalues, (c) Shi-Tomasi tracker, (d) uncertainty ellipses, (e) spline-based tracker. Top view of recovered shape for: (f) Shi-Tomasi tracker, (g) spline-based tracker.

In future work, we would like to extend our algorithm to handle occlusions in order to improve the accuracy of the flow estimates. The first part, which is simpler to implement, is to simply detect *foldovers*, i.e., when one region occludes another due to faster motion, and to disable error contributions from the occluded background. The second part would be to handle tears, either by adding an explicit occlusion model [Geiger *et al.*, 1992; Geiger and Diamantaras, 1994], or by replacing the squared matching criterion with a non-quadratic penalty function to make the results more robust [Black and Anandan, 1993].

We would also like to investigate the use of adaptively-sized patches, which can dramatically improve the quality of matching results [Okutomi and Kanade, 1992]. For spline-based registration, this requires a means of allowing varying-sized patches to tessellate the image domain, while maintaining inter-patch continuity in the motion. Our solution to this problem used the novel concept of *quadtree splines* [Szeliski and Shum, 1995b], but we have not yet applied these ideas to feature tracking.

# References

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

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

[Black and Anandan, 1993] M. J. Black and P. Anandan. A framework for the robust estimation of optic flow. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 231–236, IEEE Computer Society Press, Berlin, Germany, May 1993.

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

[Burt *et al.*, 1982] P. J. Burt, C. Yen, and X. Xu. Local correlation measures for motion analysis: a comparative study. In *IEEE Conference on Pattern Recognition and Image Processing (PRIP'82)*, pages 269–274, IEEE Computer Society Press, 1982.

[Förstner, 1987] W. Förstner. Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision, Graphics, and Image Processing*, 40:273–310, 1987.

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

[Geiger and Diamantaras, 1994] D. Geiger and K. I. Diamantaras. Occlusion ambiguities in motion. In *Third European Conference on Computer Vision (ECCV'94)*, pages 175–180, Springer-Verlag, Stockholm, Sweden, May 1994.

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

[Horn, 1983] B. K. P. Horn. Non-correlation methods for stereo matching. *Photogrammetric Engineering and Remote Sensing*, 49(4):535–536, April 1983.

[Huang, 1981] T. S. Huang. *Image Sequence Analysis*. Springer-Verlag, Berlin/Heidelberg, 1981.

[Kolb, 1994] C. E. Kolb. Rayshade user's guide and reference manual. August 1994.

[Kories and Zimmermann, 1986] R. Kories and G. Zimmermann. A versatile method for the estimation of displacement vector fields from image sequences. In *IEEE Workshop on Motion: Representation and Analysis*, pages 101–106, IEEE Computer Society Press, 1986.

[Kuglin and Hines, 1975] C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. In *IEEE 1975 Conference on Cybernetics and Society*, pages 163–165, New York, September 1975.

[Lucas, 1984] B. D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Carnegie Mellon University, July 1984.

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

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

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

[Opitz, 1983] B. K. Opitz. Advanced stereo corrleation research. *Photogrammetric Engineering and Remote Sensing*, 49(4):533–534, April 1983.

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

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

[Ryan *et al.*, 1980] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322, May/June 1980.

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

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

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

[Szeliski and Coughlan, 1994] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 194–201, IEEE Computer Society, Seattle, Washington, June 1994.

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

[Szeliski and Shum, 1995a] R. Szeliski and H.-Y. Shum. *Motion Estimation with Quadtree Splines*. Technical Report 95/1, Digital Equipment Corporation, Cambridge Research Lab, March 1995.

[Szeliski and Shum, 1995b] R. Szeliski and H.-Y. Shum. Motion estimation with quadtree splines. In *Fifth International Conference on Computer Vision (ICCV'95)*, Cambridge, Massachusetts, June 1995.

[Tian and Huhns, 1986] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer*

*Vision, Graphics, and Image Processing*, 35:220–233, 1986.

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

[Wood, 1983] G. A. Wood. Realities of automatic correlation problems. *Photogrammetric Engineering and Remote Sensing*, 49(4):537–538, April 1983.

[Xiong and Shafer, 1995] Y. Xiong and S. Shafer. Hypergeometric filters for optical flow and affine matching. In *Fifth International Conference on Computer Vision (ICCV'95)*, Cambridge, Massachusetts, June 1995.

[Zheng and Chellappa, 1992] Q. Zheng and R. Chellappa. *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, June 1992.