# A Parallel Feature Tracker for Extended Image Sequences

**Richard Szeliski** and **Sing Bing Kang**
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square, Bldg. 700
Cambridge, MA 02139

**Heung-Yeung Shum**
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

## 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.

## 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 [16] 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. 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 [1, 9].

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. A longer version of this paper is available as [13].

## 2 Previous work

Feature tracking has a long history both in computer vision and photogrammetry (see [9] 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 *a priori* 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 typically by minimizing the *sum of squared differences* (SSD)

$$E(u,v) = \sum_{k,l} [I_1(x+u+k, y+v+l) - I_0(x+k, y+l)]^2 \quad (1)$$

To obtain sub-pixel registration accuracies, a number of possible extension to the basic search technique can be used [15]: interpolating the correlation surface $E(u, v)$, interpolating the intensities, the differential method [4], and phase correlation. The differential method uses a local Taylor series expansion of the intensity function to compute a sub-pixel improvement to the displacement estimate by solving the $2 \times 2$ system of equations

$$\begin{bmatrix} \sum_{k,l} I_x^2 & \sum_{k,l} I_x I_y \\ \sum_{k,l} I_x I_y & \sum_{k,l} I_y^2 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} \sum_{k,l} I_x e_{k,l} \\ \sum_{k,l} I_y e_{k,l} \end{bmatrix} \quad (2)$$

where $\nabla I_1 = (I_x, I_y)$ is the intensity gradient and $e_{k,l}$ is the term inside the brackets in (1), i.e., the intensity error at each pixel. 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.

The basic correlation technique works well when the motion is mostly (locally) translational between frames, and

0-8186-7190-4/95 $4.00 © 1995 IEEE

when there are not large photometric variations. A more general solution can be obtained by assuming a locally *affine* model for the motion [3, 8]. The differential method then corresponds to solving a $6 \times 6$ system of equations in the unknown parameters of the affine motion model [8]. However, because of the increased number of unknowns, either fairly large patches must be used [8], or a more restricted model (scaled rotation) must be used [3]. Shi and Tomasi [9] 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 [11]. 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

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

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

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

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

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 $(u_i, v_i)$ are known linear combinations of the $(\hat{u}_j, \hat{v}_j)$.

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 [11] 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 [1]. 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.

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 [7]. First, we compute the gradient of $E$ in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$, $g_j^u = \sum_i e_i I_{xi} w_{ij}$, $g_j^v = \sum_i e_i I_{yi} w_{ij}$, where $e_i = I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)$ is the intensity error at pixel $i$, $(I_{xi}, I_{yi}) = \nabla I_1(x_i + u_i, y_i + v_i)$ 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 $\mathbf{A}$ which contains entries of the form $a_{jk}^{uu} = \sum_i w_{ij} w_{ik} I_{xi}^2$, $a_{jk}^{uv} = \sum_i w_{ij} w_{ik} I_{xi} I_{yi}$, and $a_{jk}^{vv} = \sum_i w_{ij} w_{ik} I_{yi}^2$.

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 [4, 1]. This matrix (with $w_{ij} = 1$) is identical to the Hessian matrix used in the differential method, i.e., the matrix appearing on the left-hand side of (2). 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

$$(\mathbf{A} + \lambda\mathbf{I})\Delta\mathbf{u} = -\mathbf{g}, \quad (5)$$

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 [7]. To solve this large, sparse system of linear equations, we use preconditioned gradient descent

$$\Delta\mathbf{u} = -\alpha\mathbf{B}^{-1}\mathbf{g} = -\alpha\mathbf{d} \quad (6)$$

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 $\mathbf{d} = \mathbf{B}^{-1}\mathbf{g}$ is called the *preconditioned residual* or *direction* vector. The update rule is very close to that used in the differential method [4], except that the equations for computing the $\mathbf{g}$ and $\mathbf{A}$ include weights $w_{ij}$ and a step size is necessary because all updates occur in parallel. See [11] 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 [2]. 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).

## 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. 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 [5, 10]. 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 was done in [9]). Figure 1d shows the uncertainties in potential track positions as ellipses of various sizes (this idea of display is taken from [17]). 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

The squared error in a patch is also a strong indicator of the quality of a track [9]. 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 (or suspend) 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 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 [9], 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 account for 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 the simple patch based tracker described in [9] (we also present comparative results with another algorithm and more experimental results in [13]).

### 6.1 Simulation results

We applied our tracker to five synthetic motion sequences and compared its performance with that of Shi-Tomasi's tracker. Each frame in a sequence is derived from the first frame using a known affine transformation and bilinear intensity interpolation. The five sequences used to test our tracker are the translating tree, the diverging tree, the diverging tree with $\sigma = 10$ additive Gaussian noise, a rotating tree, and a diverging yosemite (each sequence has 10 frames). A more complete set of results and figures is given in [13].

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. Parts of the feature tracks that fall at or over the image boundary are automatically ignored. Figure 1b shows the minimum eigenvalue distribution; based on this distribution, 25 point features are chosen and subsequently tracked (Figure 1c).

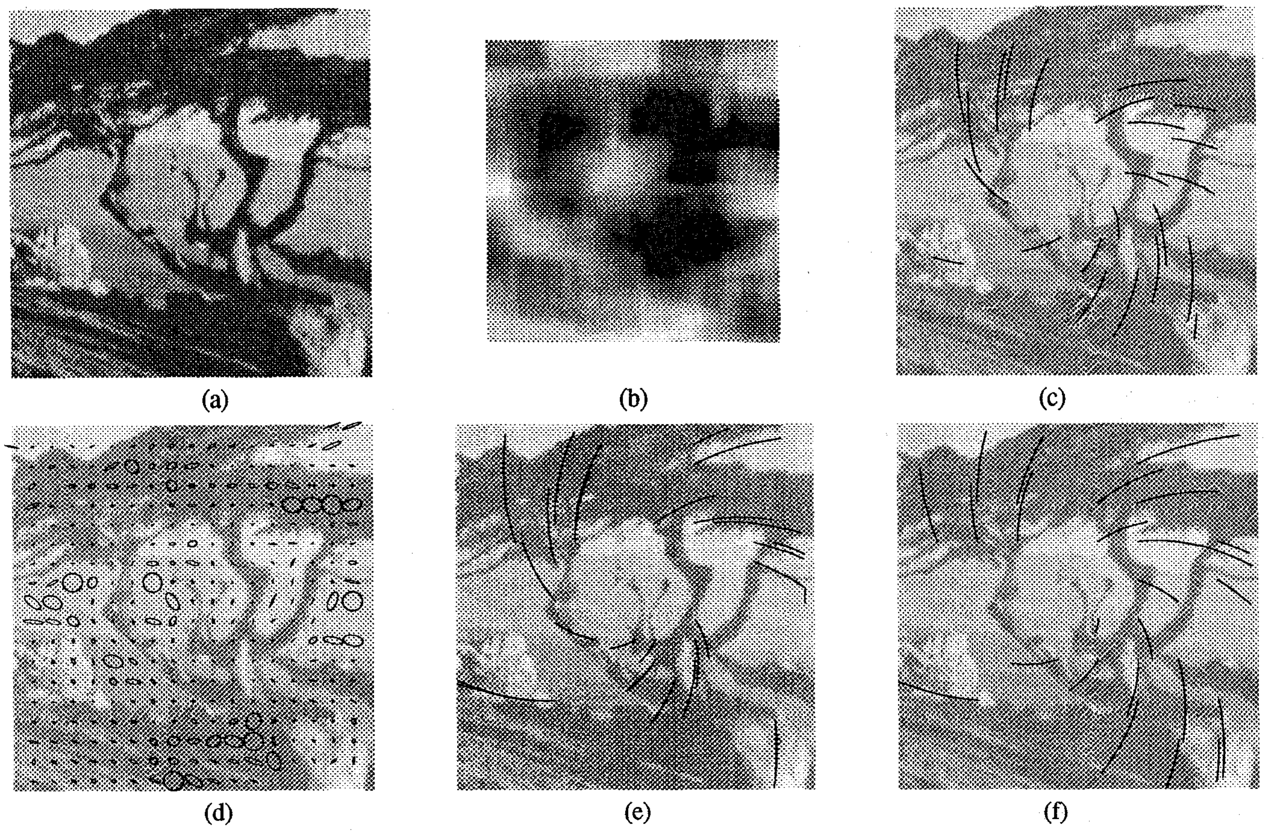


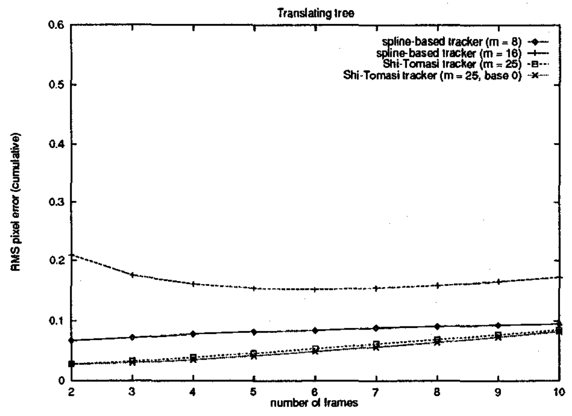
Figure 1: Rotating tree sequence: (a) frame 0, (b) minimum eigenvalues, (c) Shi-Tomasi tracker, (d) uncertainty ellipses, (e) spline-based tracker ($m = 8$), (f) spline-based tracker ($m = 16$). The amount of rotation is about $2.7°$ per frame.

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 Figure 1d, while the selected subtracks are shown in Figures 1e and (f) (for patch sizes of 8 and 16).

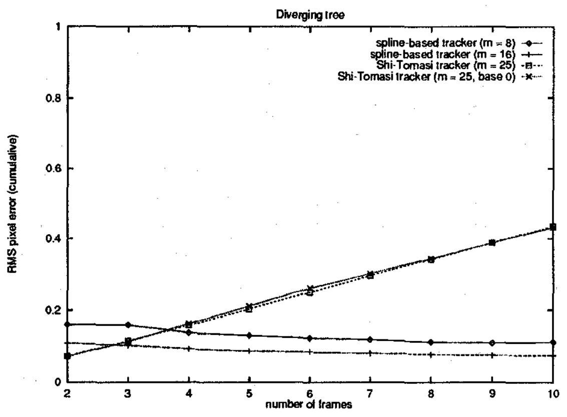
To obtain a more quantitative result, we computed the RMS pixel error in our trackers using the known affine motion, and plotted the results in Figure 2. The results for the Shi-Tomasi tracker generally exhibit a drift in tracking as shown by the increasing RMS pixel error with the number of frames. 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 of the Shi-Tomasi tracker for the case of the translating tree sequence is very small because the motion vector is approximately constant (difference of only one pixel in motion vector magnitude from one end to the other; this difference is significantly higher with the other sequences). 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 RMS pixel errors with increasing number of frames. The jump observed in Figure 2d at frame 8 can be attributed to severe misregistration at the image corners (e.g., the kink at the top left corner of the Figure 1f) whose tracks were not filtered out by the error-based and outside-of-image tests.

## 6.2 Results using a real image sequence

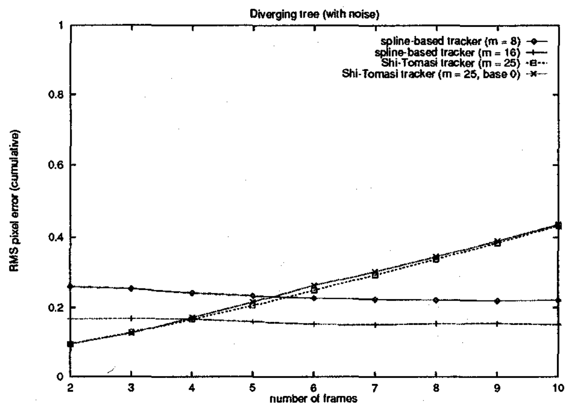
We have also applied the trackers to a real object sequence and recovered the object structure by applying an iterative non-linear least-squares structure-from-motion algorithm on the tracks [12]. The sequence is that of a rotating cube (Figure 3). The recovered 3-D feature points using the tracks from Shi-Tomasi's tracker are shown in Figure 3b. The recovered 3-D feature points using the tracks from our tracker are shown in Figure 3c. As can be seen from these figures, the structure estimates computed from our new feature tracker are less noisy.



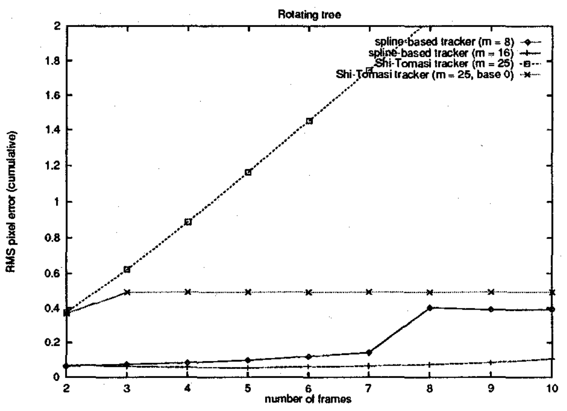
(a)



(b)

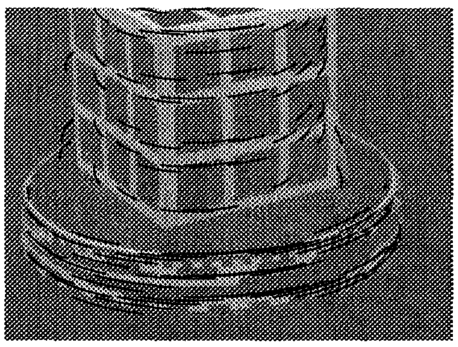


(c)

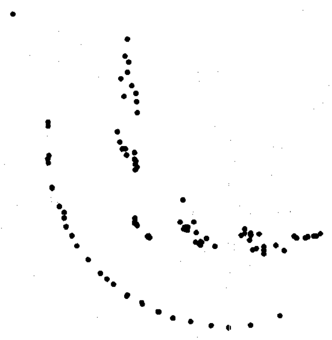


(d)

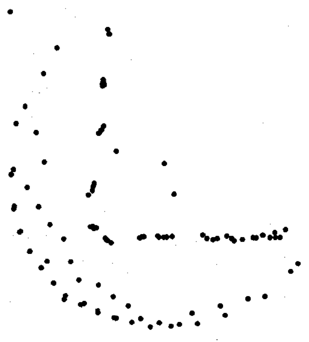
Figure 2: Comparison of RMS pixel error between the spline-based and Shi-Tomasi trackers: (a) translating tree sequence, (b) diverging tree sequence, (c) diverging tree sequence with $\sigma = 10$ noise, (d) rotating tree sequence. The results for the yosemite tree sequence (not shown) look similar to (b).



(a)



(b)



(c)

Figure 3: Cal-cube sequence: (a) image and recovered tracks (spline-based tracker), (b) top view of recovered shape (Shi-Tomasi tracker), (c) top view of recovered shape (spline-based tracker).

## 7 Discussion

Because of rounding effects during successive interframe template matching, the Shi-Tomasi tracker suffers from feature track drift. Our spline-based tracker has the clear advantage of not drifting with increasing number of frames in the image sequence. On the other hand, the spline-based tracker, in its present form, appears to be more sensitive to temporal photometric variation and additive noise. However, the problem of temporal photometric variation may be alleviated by correcting for bias or gain using bandpass filtering, while the problem of noise may be reduced using an adaptive window size scheme [6, 14].

A problem in comparing different trackers is that tracks may start at different image locations. The present simulated image sequences are basically affine transformations of the first image; more realistic image sequences would involve motion discontinuities, occlusions, and disocclusions. We are in the process of generating such sequences using an image synthesizer in order to obtain more reliable comparisons between trackers.

In future work, we plan 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 add an explicit occlusion model, which is not as straightforward because our splines are currently $C^0$ continuous.

## 8 Conclusions

We have described our spline-based tracker, which is based on the principle of local patch correlation with possible 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 [9], which we consider to be one of the most robust and accurate trackers to date. Using simulated image sequences with theoretically known feature motion, we have found that the spline-based tracker performs better in terms of pixel error accumulation as compared to the Shi-Tomasi tracker. Our preliminary results on the use of these tracks in structure from motion extraction also look promising.

## References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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.

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

[9] 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, Seattle, Washington, June 1994. IEEE Computer Society.

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

[11] 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, Seattle, Washington, June 1994. IEEE Computer Society.

[12] 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.

[13] R. Szeliski, S. B. Kang, and H.-Y. Shum. A parallel feature tracker for extended image sequences. Technical Report 95/2, Digital Equipment Corporation, Cambridge Research Lab, May 1995.

[14] R. Szeliski and H.-Y. Shum. Motion estimation with quadtree splines. Technical Report 95/1, Digital Equipment Corporation, Cambridge Research Lab, March 1995.

[15] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35:220–233, 1986.

[16] 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.

[17] 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.