NOTE

A Parallel Feature Tracker for Extended Image Sequences

Sing Bing Kang

Digital Equipment Corporation, Cambridge Research Lab, One Kendall Square, Building 700, Cambridge, Massachusetts 02139

Richard Szeliski

Microsoft Corporation, One Microsoft Way, Redmond, Washington 98052-6399

and

Heung-Yeung Shum

The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890

Received June 19, 1995; accepted March 14, 1996

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 [16, 21, 30, 23]). 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, 23].

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.

1. INTRODUCTION

Many tasks in computer vision and robotics require feature tracking, including tracking objects for grasping, tracking people in surveillance work, automatic vehicle convoys, 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 [30] 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 [16, 21, 30, 23]). 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, 23].

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 [23] 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 \( I_0 \), and then searches for a corresponding patch in the second image \( I_1 \) either by maximizing the correlation

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

or by minimizing the sum of squared differences (SSD)

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

These approaches have been extensively studied and used. See [22, 5, 10, 19, 31] for some comparative analyses and [6] for a review of statistical aspects of photogrammetry.

To obtain subpixel registration accuracies, a number of possible extensions to the basic search technique can be used [29]: interpolating the correlation surface \( E(u, v) \), interpolating the intensities, the differential method [11, 16], and phase correlation [14]. The differential method uses a local Taylor series expansion of the intensity function to compute a subpixel improvement to the displacement estimate

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

\[
= \sum_{k,l} [I_1(x + u + k, y + v + l)]^2 + \nabla I_1 \cdot (\Delta u, \Delta v)^T - I_0(x + k, y + l)]^2
\]

\[
= \sum_{k,l} [\nabla I_1 \cdot (\Delta u, \Delta v)^T]^2
\]

\[
+ \sum_{k,l} [\nabla I_1 \cdot (\Delta u, \Delta v)^T] e_{k,l} + E(u, v),
\]

where \( \nabla I_1 = (I_{x1}, I_{y1}) = \nabla I_1(x + u + k, y + v + l) \) 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

\[
\begin{bmatrix}
\sum_{k,l} I_1^2 & \sum_{k,l} I_1 I_{x1} \\
\sum_{k,l} I_1 I_{y1} & \sum_{k,l} I_{y1}^2
\end{bmatrix}
\begin{bmatrix}
\Delta u \\
\Delta v
\end{bmatrix} = \begin{bmatrix}
\sum_{k,l} I_1 e_{x1} \\
\sum_{k,l} I_1 e_{y1}
\end{bmatrix}.
\]

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 [7, 21]. The differential method then corresponds to solving a 6 \times 6 system of equations in the unknown parameters of the affine motion model [21]. It is also possible to model bias and gain variations in the intensity simultaneously with estimating the motion [7]. However, because of the increased number of unknowns, either fairly large patches must be used [21], or a more restricted model (scaled rotation) must be used [7]. Shi and Tomasi [23] 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 [26]. 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 multiframe 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

\[
E([u_i, v_i]) = \sum_i [I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)]^2,
\]

where \( \{u_i, v_i\} \) 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 grid (Fig. 1). The value for the displacement at a pixel \( i \) can be written as

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

\(^2\)The basic SSD algorithm can be made more robust to photometric variation by adding bias and gain parameters, and more robust to outliers using techniques from robust statistics.
where the \( B_j(x, y) \) are called the basis functions and are only nonzero 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 \((\hat{u}_i, \hat{v}_i)\).

In our current implementation, we make the spline control grid a regular subsampling of the pixel grid, \( x\hat{=} jmx_i, \ y\hat{=} my_j \), so that each set of \( m \times m \) pixels corresponds to a single spline patch. We also use bilinear basis functions, \( B_j(x, y) = \max((1 - |x - x_j|)/m)(1 - |y - y_j|)/m, 0) \) (see [26] for a discussion of other possible bases).

Our spline-based image registration algorithm has the following advantage over traditional motion estimators which use overlapping correlation windows [1]. 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.

3.1 Function Minimization

To recover the local spline-based flow parameters, we need to minimize the cost function (4) with respect to the \((\hat{u}_i, \hat{v}_i)\). We do this using a variant of the Levenberg–Marquardt iterative nonlinear minimization technique [20]. First, we compute the gradient of \( E \) in (4) with respect to each of the parameters \( \hat{u}_j \) and \( \hat{v}_j \),

\[
g_u^{ij} = \frac{\partial E}{\partial \hat{u}_j} = 2 \sum_i e_i I_{xi} w_{ij} \]

\[
g_v^{ij} = \frac{\partial E}{\partial \hat{v}_j} = 2 \sum_i e_i I_{yi} w_{ij},
\]

where

\[
e_i = I_0(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)
\]

is the intensity error at pixel \( i \),

\[
(I_{xi}, I_{yi}) = \nabla I_i(x_i + u_i, y_i + v_i)
\]

is the intensity gradient of \( I_i \) 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 \( I_i \) at \((x'_i, y'_i) = (x_i + u_i, y_i + v_i)\), computing \( e_i \), and finally incrementing the \( g_u^{ij} \) and \( g_v^{ij} \) values of all control vertices affecting that pixel [26].

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

\[
a_u^{ij} = 2 \sum k \frac{\partial e_i}{\partial \hat{u}_k} \frac{\partial e_i}{\partial \hat{u}_k} = 2 \sum_i w_{ik} w_{jk} I_{xi}^2
\]

\[
a_v^{ij} = 2 \sum k \frac{\partial e_i}{\partial \hat{u}_k} \frac{\partial e_i}{\partial \hat{u}_k} = 2 \sum_i w_{ik} w_{jk} I_{yi}^2
\]

\[
a_{uv}^{ij} = 2 \sum k \frac{\partial e_i}{\partial \hat{u}_k} \frac{\partial e_i}{\partial \hat{u}_k} = 2 \sum_i w_{ik} w_{jk} I_{xi} I_{yi}
\]

The entries of \( A \) can be computed at the same time as the energy gradients.

The \( 2 \times 2 \) submatrix \( A_j \) corresponding to the terms \( a_u^{ij}, a_v^{ij}, \) and \( a_{uv}^{ij} \) encodes the local shape of the sum-of-squared difference correlation surface [15, 1]. 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 \( A \) matrix is a sparse multi-banded block-diagonal matrix, i.e., subblocks containing \( a_{jk} \) will be nonzero 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 \( \Delta u \) to the current displacement estimate \( u \) which satisfies

\[
(A + \lambda I) \Delta u = -g.
\]

where \( u \) is the vector of concatenated displacement estimates \( \{\hat{u}_i, \hat{v}_i\} \), \( g \) is the vector of concatenated energy gradients \( \{g_u^{ij}, g_v^{ij}\} \), and \( \lambda \) is a stabilization factor which varies over time [20]. To solve this large, sparse system of linear equations, we use preconditioned gradient descent

\[
\Delta u = -\alpha B^{-1}g = -\alpha d,
\]
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 \( \mathbf{A} \), and \( \mathbf{d} = \mathbf{B}^{-1} \mathbf{g} \) is called the \textit{preconditioned residual} or \textit{direction} vector. The update rule is very close to that used in the differential method [15], 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 \( \lambda \) is added for stability
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

\[
\Delta E(\alpha \mathbf{d}) = \alpha^2 \mathbf{d}^T \mathbf{A} \mathbf{d} - 2 \alpha \mathbf{d}^T \mathbf{g},
\]

i.e., by setting \( \alpha = (\mathbf{d} \cdot \mathbf{g})/(\mathbf{d}^T \mathbf{A} \mathbf{d}) \). See [26] 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 [4]. 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 Fig. 2, with Fig. 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 associate a scalar confidence value with each of the motion estimates \( \hat{u}_i \),
and threshold out estimates with low confidence. Two sources of confidence information are the structure of the local Hessian matrix $A_j$ and the summed squared error within each spline patch.

To exploit the local Hessian information, we note that the eigenvectors and eigenvalues of $A_j$ 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 [17, 25]. 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 [23]). Figure 2c shows the uncertainties in potential track positions as ellipses of various sizes (this display idea is taken from [32]). 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 [23]. 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 the best $m$ tracks or all tracks whose minimum eigenvalue exceeds a threshold (see Section 6).

5. TRACKING THROUGH LONG SEQUENCES

When tracking features through more than two images, e.g., for multiframe structure from motion, we have two choices. We can either match successive pairs of images keeping track of the feature positions to subpixel position, or we can try to match all images to the initial (base) image. The first approach, taken by Shi and Tomasi [23], has the advantage that since the interframe motion is reduced (at least for a smooth sequence), a locally translational model of motion may be adequate. Note that for tracking, Shi and Tomasi use the translational model, which is equivalent to the Lucas–Kanade method with a threshold on the minimum Hessian eigenvalues. 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 interframe 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, $u_t = (t/(t-1))u_{t-1}$, and linear acceleration, $u_t = u_{t-1} + (u_{t-1} - u_{t-2})$. 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 occluded 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 [27] and with the simple patch-based tracker described in [23].

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 a known affine transformation and bilinear intensity interpolation. The first five sequences used to test our tracker are the translating tree (10 frames, Fig. 3), the diverging tree (10 frames, Fig. 4), the diverging tree with $\sigma = 10$ additive Gaussian noise (10 frames), a rotating tree (10 frames, Fig. 5), and a diverging Yosemite (10 frames, Fig. 6). The sixth sequence used to test our tracker was created using Rayshade, which is a program for creating ray-traced color images [12]. Its input is a text file that describes the properties of the camera, light

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 [27].

6 We did not implement the affine error metric which is used in [23] to monitor the quality of the tracks.

7 This is not the same texture-mapped Yosemite sequence as used in [2]; rather, it is an affine transform of the first frame from the Yosemite sequence.
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 $25 \times 25$ 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 [13]. This region-based tracker was used in our previous structure from motion research [27]. 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 [13], the image is first bandpass filtered. We also impose a dead-monotonicity operator computes the number of neighboring pixels whose intensity is less than that of the central band of a few pixels to reduce the effects of noise [27].

**FIG. 4.** Diverging tree sequence: (a) frame 0, (b) frame 9, (c) minimum eigenvalues, (d) Shi-Tomasi tracker, (e) result of monotonicity operator, (f) monotonicity tracker, (g) uncertainty ellipses, (h) spline-based tracker ($m = 8$), (i) spline-based tracker ($m = 16$).
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 to choose 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 Figs. 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, 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 Fig. 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 Fig. 9. While the error statistics look more noisy, they are actually much lower overall (the figure axes have been scaled down by $\frac{1}{2}$ from Fig. 8).

We also computed the percentage displacement and angular errors [2] over all the frames of the selected features for each of the sequences (listed in Tables 1 and 2, respectively). These metrics are actually better indications of the effectiveness of the spline-based method. Notice the significantly worse performance of the Shi–Tomasi tracker for the rotating tree sequence; this is due to its local translational model assumption. The spline-based tracker does not exhibit significant degradation of performance for the

![Image](78x106 to 553x556)

**FIG. 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 ($m = 8$).
FIG. 7. Rayshade generated sequence: (a) frame 0; (b) frame 9; (c) minimum eigenvalues; (d) Shi–Tomasi tracker; (e) result of monotonicity operator; (f) monotonicity tracker; (g) uncertainty ellipses; (h) spline-based tracker ($m = 8$); (i) spline-based tracker ($m = 16$).

same sequence. Note also that these numbers may appear incompatible with those in the graphs (Fig. 8). This is because the “best” features are at different locations, and hence have different displacements. While the magnitude displacement error may be larger, the percentage error (of the correct displacement) may be smaller. It is interesting to note that low angular errors do not automatically imply low percentage displacement errors. The large errors for the diverging sequence are primarily due to the relatively small actual displacements (on average 0.5 pixels, Table 1). The spline-based tracker did not perform as well in the Rayshade sequence because of the significant motion discontinuities.

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

FIG. 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.
FIG. 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.

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 “$\times$” 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
TABLE 1
Average Percentage Displacement Errors for Selected Tracked Features over 10 Frames

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spline-based (m = 8)</td>
<td>4.0%</td>
<td>17.6%</td>
<td>6.2%</td>
<td>8.3%</td>
<td>12.3%</td>
</tr>
<tr>
<td></td>
<td>Spline-based (m = 16)</td>
<td>4.5%</td>
<td>12.9%</td>
<td>2.4%</td>
<td>3.4%</td>
<td>11.4%</td>
</tr>
<tr>
<td></td>
<td>Shi–Tomasi (m = 25)</td>
<td>3.6%</td>
<td>80.4%</td>
<td>82.1%</td>
<td>22.3%</td>
<td>13.8%</td>
</tr>
<tr>
<td></td>
<td>Shi–Tomasi (m = 25, base 0)</td>
<td>3.5%</td>
<td>80.7%</td>
<td>145.5%</td>
<td>17.8%</td>
<td>9.0%</td>
</tr>
<tr>
<td>Mean true displacement (pixels per frame)</td>
<td>2.1</td>
<td>0.5</td>
<td>3.4</td>
<td>2.2</td>
<td>4.4</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 2
Average Angular Errors for Selected Tracked Features over 10 Frames

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spline-based (m = 8)</td>
<td>0.4°</td>
<td>12.8°</td>
<td>1.2°</td>
<td>2.7°</td>
<td>1.7°</td>
</tr>
<tr>
<td></td>
<td>Spline-based (m = 16)</td>
<td>0.3°</td>
<td>4.3°</td>
<td>0.5°</td>
<td>1.1°</td>
<td>1.3°</td>
</tr>
<tr>
<td></td>
<td>Shi–Tomasi (m = 25)</td>
<td>0.1°</td>
<td>4.4°</td>
<td>11.5°</td>
<td>1.4°</td>
<td>0.9°</td>
</tr>
<tr>
<td></td>
<td>Shi–Tomasi (m = 25, base 0)</td>
<td>0.2°</td>
<td>5.2°</td>
<td>26.9°</td>
<td>1.8°</td>
<td>0.8°</td>
</tr>
</tbody>
</table>

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 \([0 \cdots 255]\)). Both the Shi–Tomasi algorithm and the spline-based tracker were quite robust against this kind of noise, as evidenced by Figs. 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 brevity) indicated that this had very little effect on the quality of the tracks.

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 nonlinear least-squares structure-from-motion algorithm on the tracks [27]. The sequence is that of a rotating cube (Fig. 10a). The recovered 3D feature points using the tracks from Shi–Tomasi’s tracker are shown in Fig. 10f. We have also used our tracker on the rotating cube sequence. The uncertainty ellipse distribution for the sequence is shown in Fig. 10d while the filtered tracks are shown in Fig. 10e. The recovered 3D feature points using the tracks from our tracker are shown in Fig. 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 nontranslational motions and/or foreshortening effects.

We compared the performance of our spline-based tracker with Shi and Tomasi’s tracker (basically, the Lucas–Kanade algorithm with the best tracks selected according to the minimum Hessian eigenvalue), 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 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 have added an optional neighborhood search component to our algorithm. At the beginning of each set of iterations, e.g., after interlevel transfers in the coarse to fine algorithm, we search around the current \((u, v)\) estimate by trying a discrete set of nearby \((u, v)\) values (as in SSD algorithms [1]). However, because we must maintain spline continuity, we cannot make the selection of a 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.

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 [9; 8], or by replacing the squared match-
ing criterion with a nonquadratic penalty function to make the results more robust [3].

We would also like to investigate the use of adaptively sized patches, which can dramatically improve the quality of matching results [18]. For spline-based registration, this requires a means of allowing varying-sized patches to tessellate the image domain, while maintaining interpatch continuity in the motion. Our solution to this problem used the novel concept of quadtree splines [28], but we have not yet applied these ideas to feature tracking.

REFERENCES


