Linear Sequence-to-Sequence Alignment

Flávio L.C. Pádua, Rodrigo L. Carceroni, Geraldo A.M.R. Santos, and Kiriakos N. Kutulakos, *Member*, *IEEE*

Abstract—In this paper, we consider the problem of estimating the spatiotemporal alignment between N unsynchronized video sequences of the same dynamic 3D scene, captured from distinct viewpoints. Unlike most existing methods, which work for $N = 2$ and rely on a computationally intensive search in the space of temporal alignments, we present a novel approach that reduces the problem for general N to the robust estimation of a single line in $\mathbb{R}^N$. This line captures all temporal relations between the sequences and can be computed without any prior knowledge of these relations. Considering that the spatial alignment is captured by the parameters of fundamental matrices, an iterative algorithm is used to refine simultaneously the parameters representing the temporal and spatial relations between the sequences. Experimental results with real-world and synthetic sequences show that our method can accurately align the videos even when they have large misalignments (e.g., hundreds of frames), when the problem is seemingly ambiguous (e.g., scenes with roughly periodic motion), and when accurate manual alignment is difficult (e.g., due to slow-moving objects).

Index Terms—Video synchronization, object tracking, epipolar geometry, spatiotemporal alignment, image and video registration.

1 Introduction

In this work, we consider the problem of spatiotemporal alignment of multiple video sequences of the same 3D scene, captured from distinct viewpoints. Typically, the temporal misalignment between video sequences occurs when the input sequences may have different frame rates (e.g., NTSC and PAL) or when there is a time shift between the sequences (e.g., when the cameras are not activated simultaneously). On the other hand, the spatial misalignment results from the different positions, orientations, and internal calibration parameters of all the cameras. Here, we use scene dynamics as well as static scene features to estimate the temporal synchronization (temporal alignment) and the spatial alignment between the sequences. Examples of applications where this demand is particularly critical include teleimmersion [1], video-based surveillance [2], video mosaicing [3], and video metrology from television broadcasts of athletic events [4].

While many existing methods [3], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] propose solutions for spatiotemporally aligning only two sequences, our focus here is on the general case of an unrestricted number of video sequences captured from distinct viewpoints. We believe that any general solution to this problem should handle the following cases:

- *Unknown frame rate:* The relative frame rate of the video sequences is unknown and unconstrained.
- *Arbitrary time shift:* The time shift between the sequences is unknown and can be arbitrarily large.
- *Unknown motion:* The 3D motion of objects in the scene is unknown and unconstrained.
- *Tracking failures:* Individual scene points cannot be tracked reliably over many frames.
- *Unknown epipolar geometry:* The relative camera geometry of the video sequences is unknown.
- *Scalability:* Computational efficiency should degrade gracefully with increasing number of sequences.
- *No static points:* No visible point in the scene remains stationary for two or more frames.

As a step toward this goal, we present a novel solution that operates under all of the above conditions except the last one. In particular, we assume that for every pair of video sequences, we can identify enough static scene points to get an initial estimate of the cameras' epipolar geometry. Moreover, in order to ensure that the parameters of that initial estimate remain constant during the application of our approach, we consider a scenario where the video cameras are stationary, with fixed (but *unknown*) intrinsic and extrinsic parameters. In this case, every corresponding set of N pixels is related by the same spatiotemporal transformation, whose spatial components are temporally invariant.

At the heart of our approach lies the concept of a *timeline*. Given N sequences, the timeline is a line in $\Re^N$ that completely describes all temporal relations between the sequences. A key property of the timeline is that even though knowledge of the timeline implies knowledge of the sequences' temporal alignment, we can compute points on the timeline without knowing this alignment. Using this property as a starting point, we reduce the temporal alignment problem for N sequences to the problem of robustly estimating a single N-dimensional line from a set of appropriately generated points in $\Re^N$.

- *F.L.C. Pádua is with the Department of Computer Engineering, Centro Federal de Educação Tecnológica de Minas Gerais, Av. Amazonas 7675, CEP 30510-000 Belo Horizonte, MG, Brazil. E-mail: cardeal@lsi.cefetmg.br.*
- *R.L. Carceroni and G.A.M.R. Santos are with the Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG 31270-010, Brazil. E-mail: {carceron, massahud}@dcc.ufmg.br.*
- *K.N. Kutulakos is with the Department of Computer Science, University of Toronto, 10 King's College Road, Rm 3303, Toronto, ON M5S 3G4, Canada. E-mail: kyros@cs.toronto.edu.*

Manuscript received 20 Apr. 2008; revised 14 Sept. 2008; accepted 13 Oct. 2008; published online 19 Dec. 2008.
Recommended for acceptance by H. Sawhney.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-04-0232.
Digital Object Identifier no. 10.1109/TPAMI.2008.301.

Spatiotemporal alignment algorithms can be divided into two main classes: *feature-based methods* and *direct methods*. Feature-based methods [5], [6], [12], [13], [14], [15], [16], [17], [18], [19], [20], [22], [23], [24], [25] use detected features as the main input for alignment (e.g., two-frame feature correspondences or multiframe feature trajectories). Direct methods, on the other hand, rely on colors, intensities, and intensity gradients to determine the spatiotemporal alignment of overlapping videos [3], [7], [8], [9], [10], [11], [21]. As a result, direct methods tend to align sequences more accurately when their intensities are similar, while feature-based methods are appropriate when scene appearance varies greatly from sequence to sequence (e.g., due to wide baselines, different magnification, or cameras with distinct spectral sensitivities). Our approach belongs to the class of feature-based methods.

Most existing feature-based methods [6], [14], [17], [18], [19], [20], [22], [23] conduct an explicit search in the space of all possible alignments and use constraints derived from correspondences between trajectories of scene points. Unfortunately, the combinatorial nature of that search requires several additional assumptions to make it manageable. These include assuming known frame rates; restricting N to be two; assuming that the temporal misalignment is an integer; and assuming that this misalignment falls within a small user-specified range (typically, less than 50 frames). Hence, even though most of these solutions can operate when no stationary scene points exist, efficiency considerations greatly limit their applicability. Unlike these techniques, our approach aligns N sequences directly, can handle arbitrarily large misalignments between them, and does not require any a priori information about their temporal relations.

More recently, feature-based techniques relying on space-time interest points have been proposed for pairwise video alignment [5], [13], [15]. These methods tend to fail on sequences from widely separated viewpoints (where corresponding interest points cannot be found), and on sequences that contain objects moving in front of a cluttered background (where the frequent emergence and occlusion of new interest points has a confounding effect).

Techniques for simultaneously aligning more than two sequences have received much less attention. Raguse and Heipke [24] propose a method where the temporal misalignment is modeled by a second order polynomial whose coefficients, along with the cameras' epipolar geometry, are computed in a single bundle adjustment stage. This method requires features to be tracked reliably across several frames and, if only two sequences are available, cannot handle motions along the epipolar plane. Anthony et al. [25] present a two-stage method for aligning three sequences when the trajectories of individual feature points can be recovered. The method relies on 2D shape heuristics in order to bring feature trajectories into a rough temporal alignment. This alignment is then refined by enforcing trifocal constraints. In contrast to both these methods, our work applies to any number of input sequences, including just two, and does not require reliable tracking across many frames.

Our approach is most closely related to the work of Caspi et al. [18]. In their work, the epipolar geometry and temporal misalignment between two sequences are recovered from the image trajectory of a single scene point visible in both sequences, and are subsequently refined using more features. To achieve this, they assume known frame rates and formulate a nonlinear optimization problem to jointly estimate epipolar geometry and temporal misalignment. Unfortunately, the highly nonlinear nature of this optimization necessitates good initial estimates for the temporal misalignment and the epipolar geometry. Importantly, that approach still assumes that a single scene point can be tracked reliably over the entire sequence. This may be difficult to achieve when aligning videos of complex scenes, where feature tracking can fail often because of occlusions or large interframe motions. Our solution, on the other hand, requires the ability to track scene points only across two consecutive frames of the same sequence. Moreover, it does not require the ability to establish feature correspondences between the sequences.

2 The Timeline Constraint

Suppose that a dynamic scene is viewed simultaneously by N perspective cameras located at distinct viewpoints. We assume that each camera captures frames with a constant, unknown frame rate. We also assume that the cameras are unsynchronized, i.e., they began capturing frames at different times with possibly distinct frame rates. In order to temporally align the resulting sequences, we must determine the correspondence between frame numbers in one "reference" sequence and frame numbers in all other sequences. This correspondence can be expressed as a set of linear equations,

$$t_i = \alpha_i t_r + \beta_i, \tag{1}$$

where t_i and t_r denote the frame numbers of the ith sequence and the reference sequence, respectively, and α_i, β_i are unknown constants describing the temporal dilation and temporal shift, respectively, between the sequences. In general, these constants will not be integers.

The pairwise temporal relations captured by (1) induce a global relationship between the frame numbers of the input sequences. We represent this relationship by an N-dimensional line $\mathcal{L}$ that we call the *timeline*:

$$\mathcal{L} = \{[\alpha_1 \dots \alpha_N]^T t + [\beta_1 \dots \beta_N]^T \mid t \in \Re\}. \tag{2}$$

A key property of the timeline is that even though knowledge of $\mathcal{L}$ implies knowledge of the temporal alignment of the sequences, we can compute points on the timeline without knowing the sequences' alignment. This observation leads to a simple algorithm for reconstructing the timeline from dynamic features in the scene that are visible in two or more of the sequences.

Specifically, let $\mathbf{q}_1(t_1)$ be the instantaneous projection of a moving scene point in camera 1 at frame t_1, expressed in homogeneous 2D coordinates (Fig. 1). Furthermore, let $\mathbf{q}_i(\cdot)$ be the trajectory traced by the point's projection in camera i and suppose that the fundamental matrix $\mathbf{F}_{1i}$ between cameras 1 and i is known for all i, $i = 2 \dots N$. If the scene point is visible to all cameras when frame t_1 is captured by camera 1, we have the following constraint:

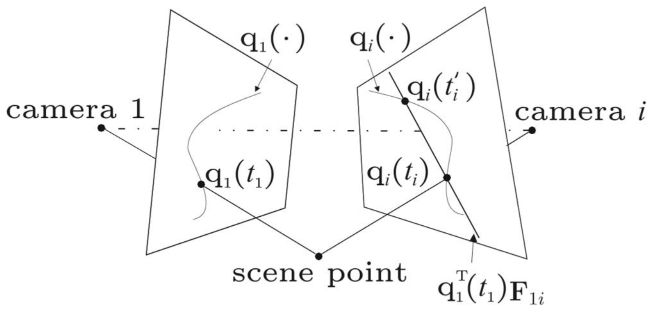

Fig. 1. Geometry of the **Timeline Constraint**. In this two-camera example, the point's trajectory in camera i intersects the epipolar line, $\mathbf{q}_1^T(t_1)\mathbf{F}_{1i}$, twice. Given the intersection points $\mathbf{q}_i(t_i)$ and $\mathbf{q}_i(t_i')$, we have the set $\mathcal{T}_{\mathbf{q}_1(t_1)} = \{[t_1 \;\; t_i]^T,\; [t_1 \;\; t_i']^T\}$.

Timeline Constraint. The set

$$\mathcal{T}_{\mathbf{q}_1(t_1)} = \{[t_1 \dots t_N]^T \mid \mathbf{q}_1^T(t_1)\mathbf{F}_{1i}\mathbf{q}_i(t_i) = 0,\; i = 2 \dots N\}$$

contains at least one point on the timeline $\mathcal{L}$.

Intuitively, the Timeline Constraint can be thought of as a procedure for generating a set $\mathcal{T}_{\mathbf{q}_1(t_1)}$ of "candidate" temporal alignments that is guaranteed to contain at least one point on the timeline. The constraint tells us that we can create such a set by: 1) intersecting the epipolar line of $\mathbf{q}_1(t_1)$ in camera i with the trajectory $\mathbf{q}_i(\cdot)$; 2) recording the frame number(s) corresponding to each intersection point for camera i; and 3) generating temporal alignment vectors from the recorded frame numbers.

To see why the Timeline Constraint holds, observe that if $[t_1 \dots t_N]^T$ is on the timeline, it must represent the "true" temporal alignment between the frame t_1 of pixel $\mathbf{q}_1(t_1)$ and the remaining cameras. Hence, pixels $\mathbf{q}_1(t_1)$ and $\mathbf{q}_i(t_i)$ must satisfy the epipolar constraint equation $\mathbf{q}_1^T(t_1)\mathbf{F}_{1i}\mathbf{q}_i(t_i) = 0$. Since, by definition, the set $\mathcal{T}_{\mathbf{q}_1(t_1)}$ contains *all* temporal alignments that satisfy the epipolar constraint equation across the N cameras, it must also contain the true alignment, which is a point on the timeline $\mathcal{L}$. In this respect, the Timeline Constraint can be thought of as the converse of the epipolar constraint for the case of N unaligned sequences.

In order to apply the Timeline Constraint, we must know the fundamental matrices $\mathbf{F}_{ij}$, describing the epipolar geometry between each pair (i, j) of cameras. In practice, we obtain an initial estimate of $\mathbf{F}_{ij}$ by finding "background features," i.e., points in the scene that remain stationary and are jointly visible by two or more cameras.[1] Once the timeline $\mathcal{L}$ is reconstructed from the estimated fundamental matrices, we jointly optimize $\mathcal{L}$ and the matrices $\mathbf{F}_{ij}$ using a linear, iterative refinement procedure. We describe the timeline reconstruction algorithm in the next section and consider the joint optimization of $\mathcal{L}$ and $\mathbf{F}_{ij}$ in Section 4.

3 Timeline Reconstruction

The Timeline Constraint leads directly to a voting-based algorithm for reconstructing the timeline of N sequences. The algorithm operates in two phases. In the first phase, we choose one of the image sequences to be the "reference" sequence. We then use the instantaneous position $\mathbf{q}_r(t_r)$ of each feature in the reference sequence and the entire trajectories of all features in the other sequences to estimate $\mathcal{T}_{\mathbf{q}_r(t_r)}$. In the second phase, we fit an N-dimensional line $\mathcal{L}$ to the union of the estimated sets $\mathcal{T}_{\mathbf{q}_r(t_r)}$. Therefore, to fully specify this algorithm, we must ask three questions: How do we compute the feature trajectories $\mathbf{q}_i(\cdot)$, for $i = 1, \dots, N$, how do we estimate the set $\mathcal{T}_{\mathbf{q}_r(t_r)}$ for each $\mathbf{q}_r(t_r)$, and how do we compute the timeline $\mathcal{L}$?

To compute the feature trajectories $\mathbf{q}_i(\cdot)$, we use a two-frame feature tracker. We treat this tracker as a "black-box" responsible for returning, for every pair of consecutive frames, a list of line segments between corresponding features. Each line segment connects the location of a feature that was detected in some frame of the ith sequence and was successfully tracked to its location in the next frame. Note that since our algorithm does not depend on a specific tracker, the tracker choice should ultimately depend on the scene's complexity and the properties of the features of interest.

Next, to compute the set $\mathcal{T}_{\mathbf{q}_r(t_r)}$ for a given $\mathbf{q}_r(t_r)$, our algorithm uses the initial estimates of the fundamental matrices $\mathbf{F}_{ij}$ between each pair (i, j) of cameras, as well as the line segments provided by the tracker. When a specific line segment intersects the epipolar line of $\mathbf{q}_r(t_r)$, it defines a possibly fractional frame number t_i corresponding to the instant that $\mathbf{q}_r(t_r)$'s epipolar line intersects the image trajectory of a scene point. Hence, t_i is the ith coordinate of a potential element of $\mathcal{T}_{\mathbf{q}_r(t_r)}$. To generate $\mathcal{T}_{\mathbf{q}_r(t_r)}$, we collect all of the t_i coordinates computed for all sequences and concatenate them so that they form N-dimensional vectors. These vectors represent candidate temporal alignments in a voting space. These steps are shown in Fig. 2.

Note that this approach may result in a large number of candidate temporal alignments being added to the voting space. This is because there may be several possible ways of "concatenating" the computed t_i coordinates into an N-dimensional vector. To avoid including an exponential number of vectors in $\mathcal{T}_{\mathbf{q}_r(t_r)}$, we only include vectors that are consistent with the cameras' epipolar geometries. In particular, let $[t_1 \dots t_N]^T$ be a candidate vector for a set of N cameras, where t_1 represents the temporal coordinate of a feature position in the reference camera. We include this vector in the voting space if for every pair of sequences i and j, the intersection points that defined t_i and t_j are within a fixed distance e from their corresponding epipolar lines (Fig. 3). In practice, we set e to be the average distance from the epipolar line of corresponding background features in the two views. This pruning criterion is conservative, i.e., it guarantees that the set of vectors placed in the voting space will be a superset of $\mathcal{T}_{\mathbf{q}_r(t_r)}$.

The set of candidate temporal alignments is the union of the sets $\mathcal{T}_{\mathbf{q}_r(t_r)}$ for all $\mathbf{q}_r(t_r)$. In general, this union will contain a large number of outliers, as illustrated in Fig. 2d. To reconstruct the timeline in the presence of outliers, we use the RANSAC algorithm [27]. The algorithm randomly chooses a pair of candidate temporal alignments to define the timeline $\mathcal{L}$, and then, computes the total number of candidates that fall within an ϵ-distance of this line. These

1. For pairs of cameras with wide baselines, this should be done using feature descriptors that are appropriate for wide-baseline stereo matching [26].

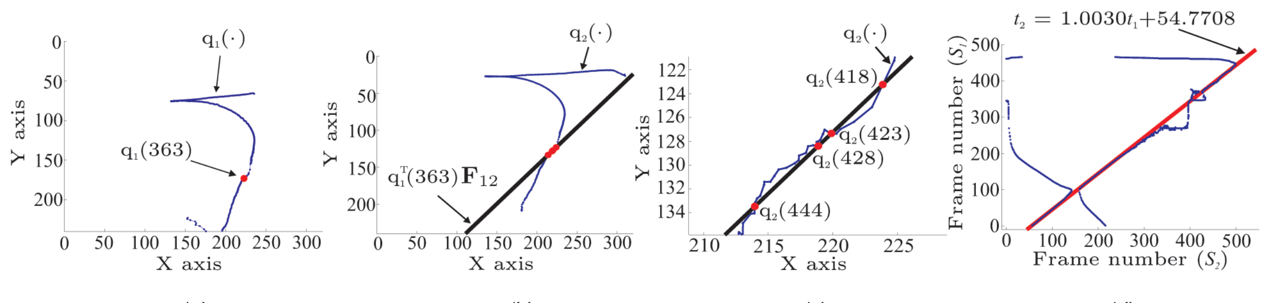

Fig. 2. (a) Trajectory of a feature in Sequence 1 of the *Car* data set, discussed in Section 5. The feature was the centroid of all pixels labeled as "foreground" by a color-based foreground-background detector. (b) Trajectory of the foreground pixel centroid along another viewpoint (Sequence 2 of the data set). Also shown is the epipolar line corresponding to pixel $\mathbf{q}_1(363)$ in (a). (c) Magnified view of the trajectory/epipolar line intersection in (b). The individual line segments connecting feature locations in adjacent frames are now visible. Note that the epipolar line of $\mathbf{q}_1(363)$ intersects multiple line segments along the trajectory. (d) Exploiting the Timeline Constraint for two-sequence alignment. Each point in the plot represents a candidate temporal alignment for the two sequences, i.e., an element of $\mathcal{T}_{\mathbf{q}_1(t_1)}$ for the feature location $\mathbf{q}_1(t_1)$ in (a). The reconstructed timeline, drawn as a red solid line, describes the temporal alignment of the two sequences.

two steps are repeated for a number of iterations. Therefore, the two critical parameters of the algorithm are the number k of RANSAC iterations and the distance ϵ. To determine k, we use the formula

$$k = \left\lceil \frac{\log(1-p)}{\log(1-r^2)} \right\rceil, \tag{3}$$

where p is a user-specified parameter between 0 and 1 and r is the probability that a randomly selected candidate is an inlier. Equation (3) expresses the fact that k should be large enough to ensure that with probability p, at least one randomly selected pair of candidates is an inlier. We used $p = 0.99$ and $r = 0.05$ for all experiments. To compute the distance ϵ, we observe that ϵ can be thought of as a bound on the distance between detected feature locations in the input cameras and their associated epipolar lines. This allows us to approximate ϵ by the average distance between static features in the scene and their associated epipolar lines.

The way the Timeline Constraint is formalized suggests that a scene point must be simultaneously visible (and tracked) in all N cameras. While this becomes a limitation when $N > 2$, it is easy to extend the basic idea to incorporate constraints where features are visible only in some of the views. For instance, a partial two-view match in a three-view problem would vote for a line (instead of just a point) in the 3D temporal space. Combined with appearance-based weights, for instance, such extension may make the proposed method more flexible. Unfortunately, in the current implementation, where all matches cast equally strong votes, this would only pollute the voting space.

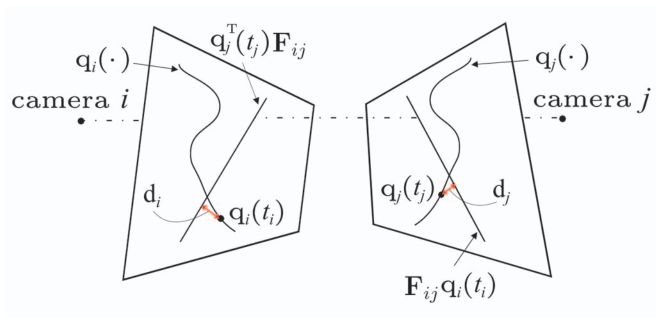

Fig. 3. Pruning candidate vectors. Two intersection points $\mathbf{q}_i(t_i)$ and $\mathbf{q}_j(t_j)$ in cameras i and j, respectively, are considered to be potential projections of the same scene point only if the distances d_i, d_j to each others' epipolar lines are within a fixed threshold. This threshold depends on the estimated fundamental matrix between cameras i and j and is estimated from corresponding points in the stationary background.

4 Timeline Refinement

While images of a dynamic scene may contain stationary points in the background, these points cannot be expected to represent the majority of detected features. Any procedure that attempts to estimate epipolar geometry from those features alone is likely to ignore a significant portion of the available image information. In practice, this will cause errors in the computed fundamental matrices, and ultimately, in the reconstructed timeline. Here, we show how to refine the matrices $\mathbf{F}_{ij}$ and the timeline $\mathcal{L}$ by incorporating all features detected in the sequences. Without loss of generality, we assume that camera 1 is our reference camera. We first describe a method for the case of two viewpoints ($N = 2$) and then generalize it for the case of $N > 2$.

Fig. 4 shows the geometry of the two-view refinement method. To make the problem linear, we approximate each trajectory $\mathbf{q}_2(\cdot)$ with a polygonal spline $\mathrm{s}_2(\cdot)$ using a method that guarantees an upper bound on the difference between the length of the original curve and the length of its polygonal

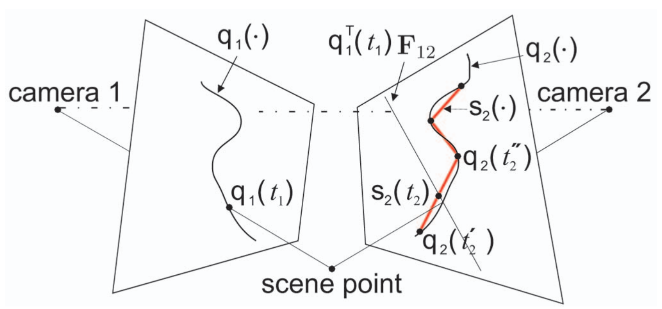

Fig. 4. Geometry of our two-view refinement method. The polygonal spline approximation of the trajectory $\mathbf{q}_2(\cdot)$ is shown in red.

approximation [28]. The spline is parameterized in a way that is consistent with the time parameter of trajectory $\mathbf{q}_2(\cdot)$, i.e., each point that lies both on the trajectory and the spline approximation keeps its original time coordinate.

Given this polygonal approximation, we refine $\mathbf{F}_{12}$ and the timeline parameters α and β as follows: For each point $\mathbf{q}_1(t_1)$ on the trajectory of camera 1, we

1. compute the intersection of epipolar line $\mathbf{q}_1^T(t_1)\mathbf{F}_{12}$ with the spline $\mathbf{s}_2(\cdot)$ in camera 2,
2. evaluate the consistency of each intersection point with the current estimate of α and β, and
3. refine $\mathbf{F}_{12}$, α, and β using algebraic constraints derived from the consistent intersection points.

More specifically, according to (1), an estimated temporal transformation with parameters α and β implies that any instantaneous feature $\mathbf{q}_1(t_1)$ in camera 1 should correspond to a feature in camera 2 whose temporal coordinate is

$$t_2 = \alpha t_1 + \beta. \tag{4}$$

Thus, an intersection point $\mathbf{s}_2(t_2)$ in camera 2 that lies on a spline segment with endpoints $\mathbf{q}_2(t_2')$ and $\mathbf{q}_2(t_2'')$ is consistent with the timeline parameters α, β only if

$$t_2' < \alpha t_1 + \beta < t_2''. \tag{5}$$

Every intersection point that satisfies (5) yields a linear constraint on α, β, and the entries of $\mathbf{F}_{12}$. To obtain this constraint, we note that an arbitrary intersection point $\mathbf{s}_2(t_2)$ on the spline segment is given by

$$\mathbf{s}_2(t_2) = \mathbf{q}_2(t_2') + (t_2 - t_2')\frac{\mathbf{q}_2(t_2'') - \mathbf{q}_2(t_2')}{t_2'' - t_2'}. \tag{6}$$

Since $\mathbf{s}_2(t_2)$ satisfies the epipolar constraint, we have

$$\mathbf{q}_1^T(t_1)\mathbf{F}_{12}\mathbf{s}_2(t_2) = 0. \tag{7}$$

Substituting (6) into (7), we obtain

$$\mathbf{q}_1^T(t_1)\mathbf{F}_{12}\left\{\mathbf{q}_2(t_2') + (t_2 - t_2')\frac{\mathbf{q}_2(t_2'') - \mathbf{q}_2(t_2')}{t_2'' - t_2'}\right\} = 0. \tag{8}$$

In general, since the timeline parameters α, β, and the matrix $\mathbf{F}_{12}$ may contain errors, (8) will be satisfied only approximately. We can therefore refine these estimated parameters by minimizing the algebraic error defined by (8). Specifically, rewrite (8) as follows:

$$\mathbf{q}_1^T(t_1)\{\mathbf{F}_{12}\mathbf{k}t_2 + \mathbf{F}_{12}\mathbf{m}\} = 0, \tag{9}$$

where

$$\mathbf{k} = \frac{\mathbf{q}_2(t_2'') - \mathbf{q}_2(t_2')}{t_2'' - t_2'} \tag{10}$$

and

$$\mathbf{m} = \mathbf{q}_2(t_2') - t_2'\mathbf{k}. \tag{11}$$

Now, let $\hat{\alpha} = \alpha + \Delta\alpha$, $\hat{\beta} = \beta + \Delta\beta$, and $\hat{\mathbf{F}}_{12} = \mathbf{F}_{12} + \Delta\mathbf{F}_{12}$ be the updated estimates of the parameters α, β, and $\mathbf{F}_{12}$, respectively. Substituting in (9), the factor enclosed by curly brackets becomes

$$\begin{aligned}\mathbf{F}_{12}(t_1\alpha\mathbf{k} + \beta\mathbf{k} + \mathbf{m}) + t_1\mathbf{F}_{12}\mathbf{k}\Delta\alpha + \mathbf{F}_{12}\mathbf{k}\Delta\beta +& \\ + \Delta\mathbf{F}_{12}(t_1\alpha\mathbf{k} + \beta\mathbf{k} + \mathbf{m}) +& \\ + t_1\Delta\alpha\Delta\mathbf{F}_{12}\mathbf{k} + \Delta\beta\Delta\mathbf{F}_{12}\mathbf{k}.&\end{aligned}$$

Disregarding the second order terms $t_1\Delta\alpha\Delta\mathbf{F}_{12}\mathbf{k}$ and $\Delta\beta\Delta\mathbf{F}_{12}\mathbf{k}$, we obtain the following linear constraint on $\Delta\mathbf{F}_{12}$, $\Delta\alpha$, and $\Delta\beta$:

$$\begin{aligned}\mathbf{q}_1^T(f_1)\{t_1\mathbf{F}_{12}\mathbf{k}\Delta\alpha + \mathbf{F}_{12}\mathbf{k}\Delta\beta + \Delta\mathbf{F}_{12}\mathbf{h}\} = -\mathbf{q}_1^T(t_1)\mathbf{F}_{12}\mathbf{h},& \\ \text{where} \quad \mathbf{h} = (t_1\alpha + \beta)\mathbf{k} + \mathbf{m}.&\end{aligned} \tag{12}$$

After straightforward algebraic manipulation, (12) may be rewritten as the inner product of two vectors: an 11-element row vector that contains only known coefficients and an 11-element column vector that contains the nine unknown coefficients of $\Delta\mathbf{F}$ followed by the scalar unknowns $\Delta\alpha$ and $\Delta\beta$. By taking all consistent intersection points into account, we obtain an overconstrained linear system in terms of the unknowns $\Delta\alpha$, $\Delta\beta$, and $\Delta\mathbf{F}_{12}$. In addition, we include in the system the linear constraints derived from static scene features. This allows us to use all available constraints, both from static features and dynamic features, in order to refine our spatiotemporal alignment. In practice, the number of equations in the system is frequently much larger than the 11 unknowns; we use standard numerical methods to solve this system [29], [30], [31], [32] and iterate until convergence.

Consider now a dynamic scene viewed by N distinct cameras, where $N > 2$. In this case, we simply use the two-view refinement technique for each camera pair (c_1, c_i), where c_i is the ith camera. The $N - 1$ two-view timelines computed in this fashion yield the N-view timeline in a straightforward way.

Note that the Timeline Refinement procedure outlined above is essentially a two-view method. In principle, it is possible to generalize this procedure to handle all N views simultaneously by computing and refining multiview tensors rather than pairwise fundamental matrices. Even though we implemented this generalization, we found it to be numerically unstable in preliminary experiments and did not pursue it any further.[2]

5 Experimental Results with Real Sequences

In order to demonstrate the advantages and limitations of the proposed approach, we performed experiments with several real two-view and three-view sequences. These sequences were chosen to be hard to handle with existing approaches because of one or more of the following complications:

1. feature trajectories that were periodic and overlapping;
2. feature trajectories that did not lie on a single plane;
3. sequences with large temporal misalignments;
4. frame rates that varied from sequence to sequence in an unknown way;
5. unreliable, frequently interrupted feature tracking.

2. Another possible N-view extension is to refine the epipolar geometry and the timeline of pairs of nonreference views, and then, robustly merge all two-view results into an N-view timeline. This is still under investigation, and it is not clear that the reliability and/or accuracy gains of such an approach are sufficient to justify the added computational cost.

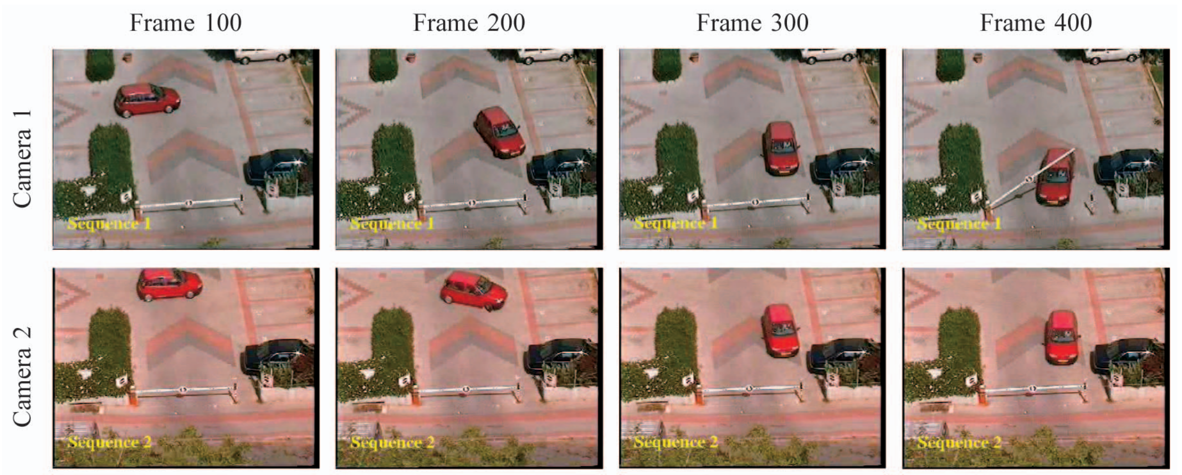

Fig. 5. Four representative frames (100, 200, 300, and 400) from the cameras 1 and 2 of the two-view *Car* data set [21]. Observe the spatial misalignment near image boundaries, where different static objects are visible in each sequence. The temporal misalignment is easily identified by comparing the position of the gate in frames 400.

Image dimensions in all data sets were 320×240 pixels. The sequences represented a wide variety of conditions, including sequences that ranged from 55 to 605 frames; temporal misalignments of 21-285 frames; relative frame rates between 1 and 2; image quality that ranged from quite high (i.e., sequences captured by laboratory-based color cameras) to rather low (i.e., clips from low-quality, MPEG-compressed video of a broadcast TV signal); and object motions ranging from several pixels per frame to less than a pixel. Since no single tracker was able to handle all of our data sets and since our algorithm does not depend on a specific tracker, we experimented with several—a simple color-based blob tracker, a blob tracker based on background subtraction, and the WSL tracker [33]. In each case, we treated the tracker as a "black box" that returned a list of corresponding features for every pair of consecutive frames.

Alignment accuracy was evaluated by measuring the average temporal misalignment. This is the average difference between the computed time of each frame and the frame's "ground-truth" time, i.e., when it was actually captured. Since our sequences were acquired with unsynchronized cameras, the ground-truth time of each frame could only be known to within ± 0.5 frames. This is because even if we could perfectly align the sequences at frame resolution, corresponding frames could have been captured up to 0.5-frame intervals apart. This bound on ground-truth accuracy should be taken into account when evaluating the results below.

5.1 Two-View *Car* Data Set

As a first test, we applied our technique to a two-view sequence used by Caspi and Irani [21] for evaluating their method (Fig. 5). The data were acquired by two cameras with identical frame rate of 25 fps, implying a unit ground-truth temporal dilation ($\alpha = 1$). The ground-truth temporal shift was $\beta = 55 \pm 0.5$ frames.

Most frames in the resulting sequences contain a single rigid object (a car) moving over a static background (a parking lot), along a fairly smooth trajectory. We therefore used a simple blob tracker that relied on foreground-background detection to label all foreground pixels in each frame. The centroid of the foreground pixels was the only "feature" detected and tracked (Figs. 2a and 2b). To compute the cameras' fundamental matrix, we used 26 manually selected correspondences between background pixels in the two views. Fig. 2d shows the timeline reconstructed using the RANSAC-based algorithm in Section 3, with the RANSAC parameter ϵ set to 2.0. The reconstructed timeline gives an average temporal misalignment of 0.66 frames, almost within the 0.5-frame uncertainty of the ground-truth measurements.

Applying the refinement procedure in Section 4 produced updated values of $\alpha = 1.0027$ and $\beta = 54.16$ for the timeline coefficients. These coefficients correspond to an improved average temporal misalignment of 0.35 frames, i.e., higher than the accuracy of the ground-truth alignment. Note that these results are at least as accurate as those of Caspi and Irani [21], even though we are solving a less constrained problem (i.e., α is unknown and scene planarity is not required). Moreover, the results were obtained from raw results of a tracker that was not particularly robust (e.g., the centroid of the foreground pixels drifts off the moving car for approximately 30 frames in each sequence).

5.2 Two-View *Robots* Data Set

In a second experiment, we used two cameras operating at 30 fps to acquire images of four small robots, as they executed small random movements on two planes (Fig. 6a). The ground-truth timeline coefficients were $\alpha = 1$ and $\beta = -284.5 \pm 2$. We used a uniform-color blob tracker to track these robots between consecutive frames. The resulting data were challenging for four reasons. First, the robots' interframe motion was imperceptibly small (roughly 0.25 pixels per frame), making precise manual alignment by a human observer virtually impossible. Second, the temporal shift of the sequences was large, making it inefficient to find this shift via exhaustive search. Third, the uniformly colored regions on each robot were small, causing our tracker to generate fragmented and noisy trajectories. Fourth, the robot's motion was designed to produce trajectories that self-intersect and are nonsmooth, complicating the shape of each blob's trajectory.

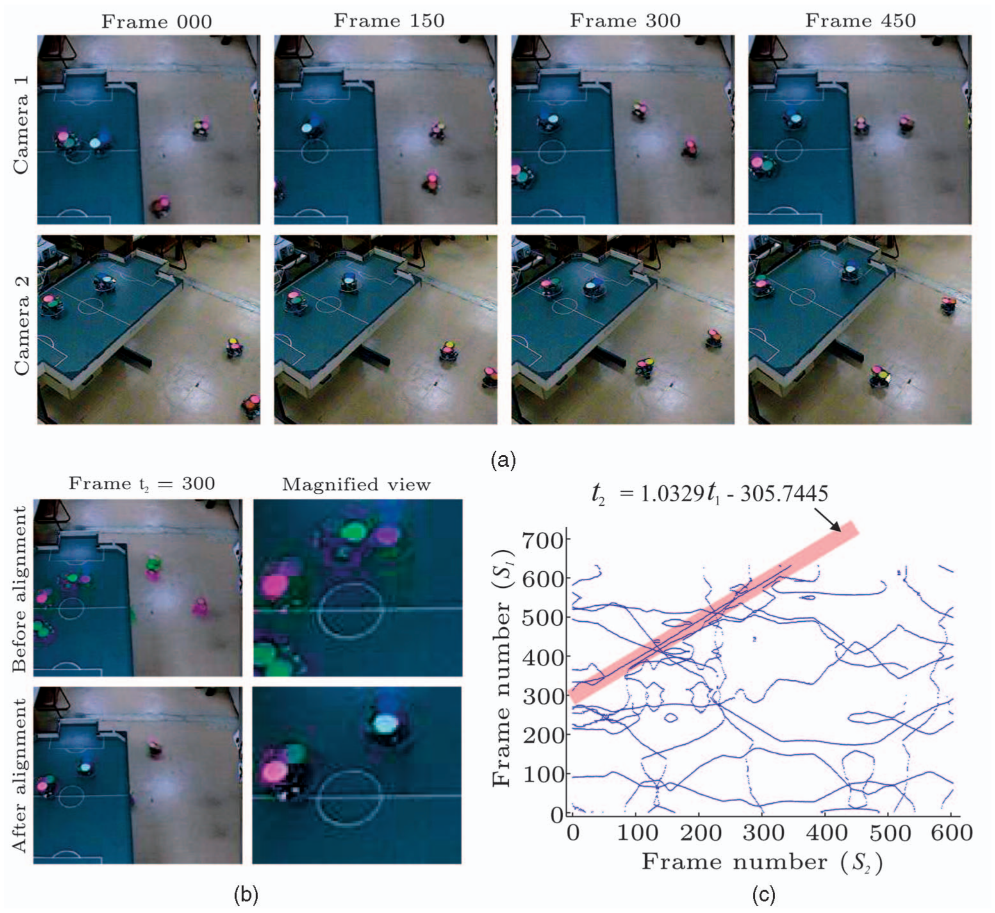

Fig. 6. (a) Four out of 605 frames from the two-view *Robots* data set. The spatial misalignment can be easily identified by observing the distinct orientations of the robots' soccer field. (b) **Before-alignment images** were created by superimposing the green band of a frame t_2 with the red and blue bands of frame $t_1^* = (t_2 - \beta^*)/\alpha^*$ using ground-truth timeline coefficients α^* and β^*. **After-alignment images** were created by replacing the green band of the images above them with that of frame $t_1 = (t_2 - \beta)/\alpha$, with α, β computed by our algorithm. For both types of images, deviations from the ground-truth alignment cause "double exposure" artifacts (i.e., when $t_1^* \neq t_2$ or $t_1^* \neq t_1$, respectively). (c) Voting space, timeline, and timeline equation recovered prior to refinement for the two-view *Robots* data set. Each point is an element of $\mathcal{T}_{\mathbf{q}_1(t_1)}$ for some feature $\mathbf{q}_1(t_1)$ in sequence 1.

The timeline reconstructed with $\epsilon = 2.0$ prior to refinement is shown in Fig. 6c. This line gives an average temporal misalignment error of 5.84 frames. Our refinement stage reduced this error to 4.43 frames, with $\alpha = 1.015$ and $\beta = -286.89$. Given the robots' image velocity, this translates to a misalignment of about one pixel. Fig. 6b confirms that the computed alignment is quite good, despite the robots' slow motion and the tracker's poor performance.

5.3 Two-View *Juggling* Data Set

In this data set, two people are observed by a wide-baseline camera pair while juggling five uniformly colored balls (Fig. 7a). Both sequences were acquired at a rate of 30 fps. This data set represents a difficult case for existing direct- or feature-based methods because:

1. the trajectories of different balls nearly overlap in 3D
2. individual trajectories are approximately periodic,
3. image velocities are quite large, up to 9 pixels per frame, making long-range feature tracking difficult, and
4. the ground-truth temporal shift between the sequences is $\beta = -41 \pm 0.5$ frames, or about 1.5 periods of a ball's motion.

This shift is likely to cause difficulties for techniques based on nonexhaustive search [17] or nonlinear optimization [18] because of the possibility of getting trapped in deep local minima. To make the alignment problem even more challenging, we modified this data set by deleting or adding frames to one of the sequences. These modifications were intended to simulate sequences with more than one frame rate (e.g., containing a slow-motion segment) and sequences that contain spurious clips (e.g., a TV commercial).

We used a uniform-color blob tracker to track four of the balls in each sequence, providing us with the location of four features per frame. No information about feature correspondences between cameras was given to the algorithm (i.e., color information was not used). Figs. 7c, 7d, and 7e show the reconstructed timelines before the refinement stage, with $\epsilon = 0.5$. The average temporal misalignment error was 0.75 frames for the original data set. The refinement stage brought this error down to 0.26 frames, with $\alpha = 1.0004$ and $\beta = -40.80$. In Fig. 8, we show the distribution of distances of inlier votes from the reconstructed timelines for the *Car*, *Robots*, and *Juggling* data sets.

5.4 Three-View *Soccer* Data Set

As a final experiment, we applied our technique to three video clips extracted from a single MPEG-compressed TV

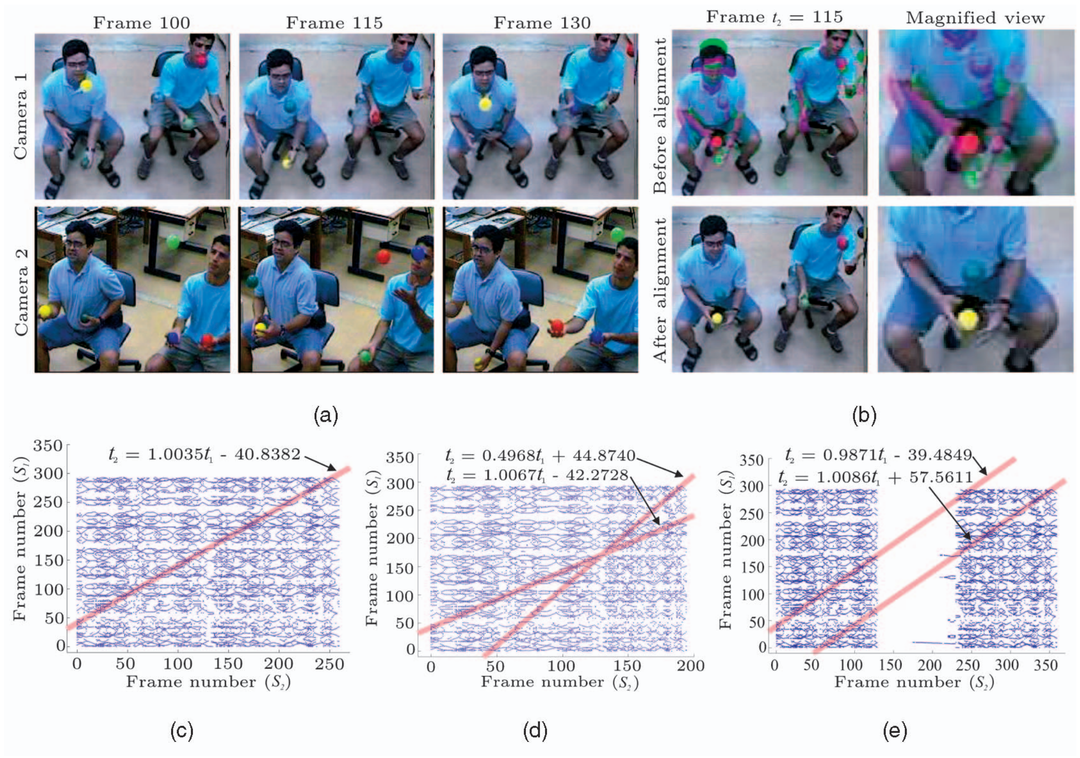

Fig. 7. (a) Three out of 260 frames from the two-view *Juggling* data set. (b) **Before-alignment images** were created by superimposing the green band of a frame t_2 with the red and blue bands of frame $t_1^* = (t_2 - \beta^*)/\alpha^*$ using ground truth timeline coefficients α^* and β^*. **After-alignment images** were created by replacing the green band of the images above them with that of frame $t_1 = (t_2 - \beta)/\alpha$, with α, β computed by our algorithm. For both types of images, deviations from the ground-truth alignment cause "double exposure" artifacts (i.e., when $t_1^* \neq t_2$ or $t_1^* \neq t_1$, respectively). (c)-(e) Voting spaces, timelines, and timeline equations recovered prior to refinement for the two-view *Juggling* data set: (c) *Juggling* data set without modification. (d) Simulation of a sequence with more than one frame rate. (e) Simulation of a sequence with spurious clips.

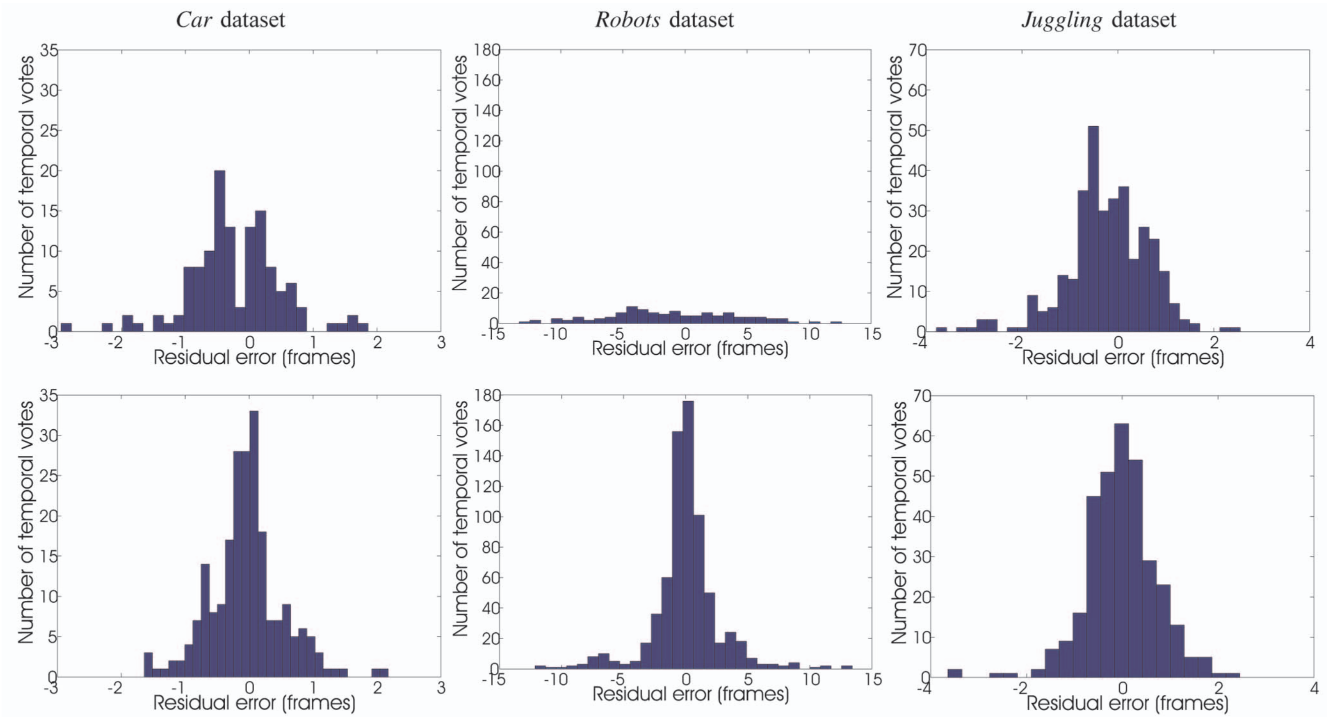

Fig. 8. Distribution of distances of inlier votes from the reconstructed timeline. **Top row:** Distribution before the timeline refinement stage. **Bottom row:** Distribution after the refinement stage. Note that the updated epipolar geometry and updated timeline parameters reduce the distance between inliers and the timeline and cause more votes to be labeled as inliers.

broadcast of a soccer match [34]. The clips were replays of the same goal filmed from three distinct viewpoints (Fig. 9a). Each sequence contained a significant panning motion to maintain the moving players within the field of view. To ensure that the pairwise fundamental matrices remained constant for all frames, we stabilized each sequence by

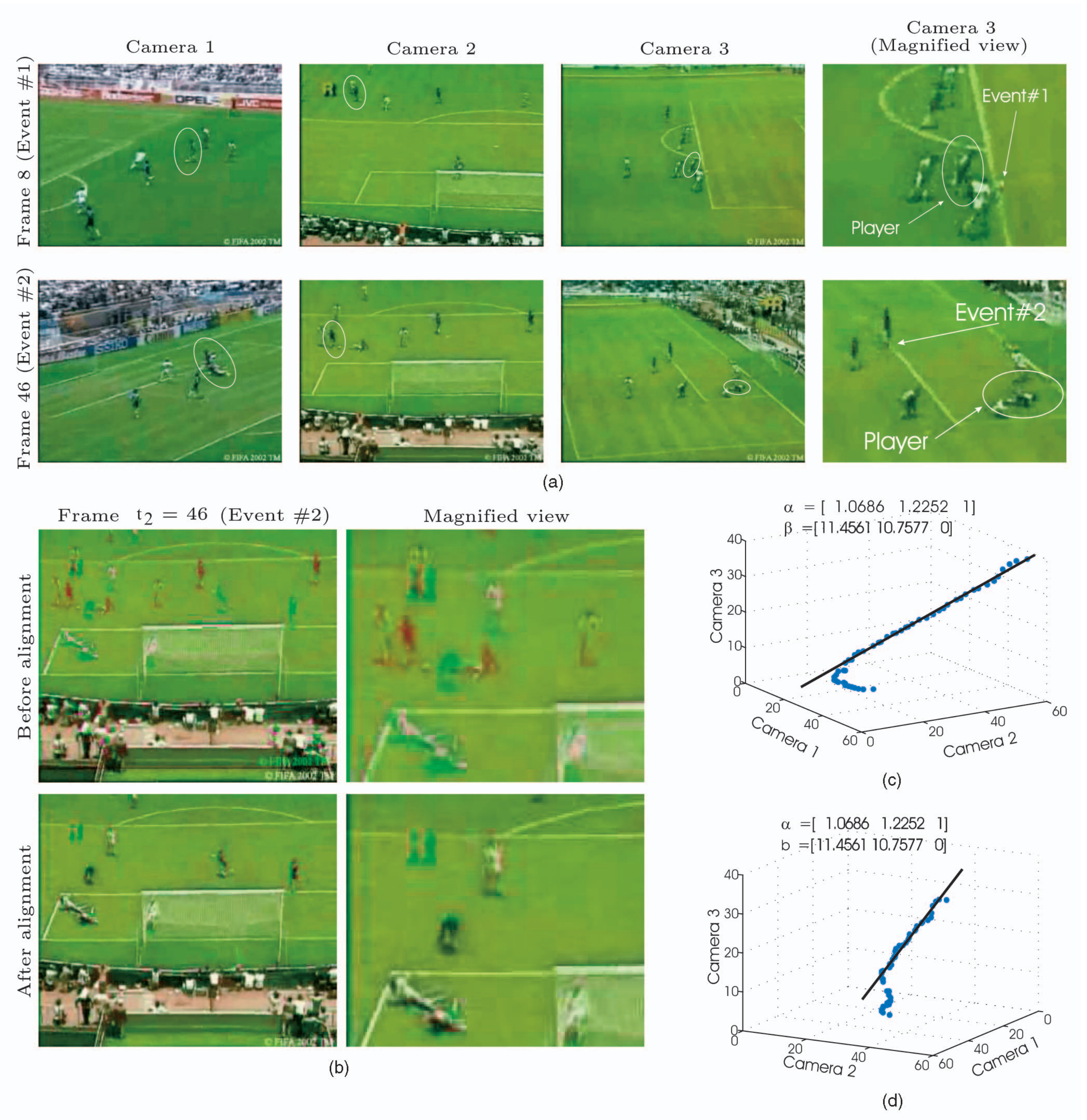

Fig. 9. (a) Two out of 55 frames from the three-view *Soccer* data set. Ellipses indicate the player being tracked by the WSL tracker. (b) **Before-alignment images** were created by superimposing the green band of a frame t_2 with the red and blue bands of frame $t_1^* = (t_2 - \beta^*)/\alpha^*$ using ground truth timeline coefficients α^* and β^*. **After-alignment images** were created by replacing the green band of the images above them with that of frame $t_1 = (t_2 - \beta)/\alpha$, with α, β computed by our algorithm. For both types of images, deviations from the ground-truth alignment cause "double exposure" artifacts (i.e., when $t_1^* \neq t_2$ or $t_1^* \neq t_1$, respectively). (c) and (d) Two views of the 3D voting space and 3D timeline computed for the *Soccer* data set.

computing the frame-to-frame homography using Brown and Lowe's system [35]. We used the WSL tracker to track the same player in each sequence, thereby obtaining one feature trajectory per camera. WSL was initialized manually in the first frame of each sequence. Even though it was able to track the chosen player for most frames, the player's small size and jitter artifacts caused by the video's poor quality resulted in noisy measurements of his location. These measurements were given as input to the basic timeline reconstruction algorithm with $\epsilon = 1.5$ and no timeline refinement.

Since this data set contained $N = 3$ views, the timeline is a 3D line with 3-vectors as its coefficients (see (2), Figs. 9c and 9d). To evaluate the timeline's accuracy in the absence of ground-truth information, we attempted to estimate the ground-truth alignment by visual inspection: we identified three easily distinguishable events (e.g., a player stepping on a field line, as shown in Fig. 9a) and recorded the frame where each event occurred in each sequence. These frames were used as "ground-truth" event times for each camera.

To evaluate the timeline's accuracy, we used it to predict the event times in cameras 1 and 2 from the ground-truth time in camera 3. The minimum difference between the predictions and the ground-truth times across all three events was 0.22 frames in camera 1 and 0.86 frames in camera 2; the maximum difference was 1.66 and 1.33 frames, respectively. This confirms that the sequences were aligned quite well (see Fig. 9b), despite the low quality of the videos and their unequal frame rates.

5.5 Discussion

Figs. 6c, 7c, 7d, and 7e suggest that a potential limitation of the proposed method is scalability with respect to the number of features. In Section 6, we present a large set of experiments with synthetic sequences that make this limitation clear. A key factor behind this limitation is that the timeline reconstruction algorithm in Section 3 does not weigh votes according to the visual similarity of the features that produced them. This means that the density of the voting space increases substantially when the number of tracked features increases. Moreover, features that move too slowly also tend to generate more "outlier votes" than fast-moving ones and so do features that move almost parallel to an epipolar line. Weighing each vote by a combination of feature similarity, speed, and direction of motion would go a long way toward increasing scalability with respect to the number of features.

We also believe that this limitation becomes less serious when the number of cameras increases. This is because epipolar constraints from nonreference views can be used to prune outliers from the voting space before applying RANSAC. Even more importantly, the probability that outliers will cluster accidentally along linear structures in the voting space decreases exponentially with the number of dimensions/cameras. This can be observed, to a limited extent, in Figs. 9c and 9d.

6 Sensitivity Analysis

To analyze the behavior of our approach in more detail, we ran a much larger set of experiments with synthetic scenes, where important parameters such as the number of features being tracked at any given instant, the noise in feature trajectories, and errors in the initial epipolar geometry were under control.

Our experiments with synthetic data aimed at answering the following questions:

- How does the algorithm scale with respect to the number of tracked features?
- How is the algorithm's reliability affected by errors in tracked trajectories?
- How is the algorithm's reliability affected by errors in initial estimates for the epipolar geometry?

In order to answer these questions, we performed experiments where random 3D feature trajectories were generated within the fields of view of two synthetic cameras. These trajectories followed a very simple 3D dynamics model and were projected on the cameras' image planes to obtain a set of corresponding 2D trajectories. The number of features was kept constant through each experiment.

TABLE 1
Simulation Parameters for Our Synthetic Data Experiments

Parameter	Description	Values used in experiments
F	Number of features	$\{1, 2, 4, 8, 16, 32\}$
T	Number of frames in sequence	256
G_a	Standard deviation of 3D acceleration	25mm/frame^2
G_l	Tracker localization error	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ pixels
G_e	Epipolar geometry error	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ pixels
p	RANSAC success probability	0.99
r	RANSAC inlier probability	0.05
ϵ	RANSAC tolerance parameter	0.5

Controlled levels of noise were added both to the 2D feature trajectories and the ground-truth epipolar geometry. The resulting 2D trajectories were then used as inputs to our spatiotemporal alignment algorithm. The intrinsic and extrinsic parameters of the synthetic cameras were identical to those in the real experiments in Sections 5.2 and 5.3.

1. *Generative model for 3D feature trajectories:* For each independent run, we simulate the simultaneous motion of F 3D features for a fixed period of $T = 256$ frames (Table 1). These features are viewed by a pair of cameras with identical ground-truth frame rates and a ground-truth temporal displacement of 32 frames. The initial position $\mathbf{m}_0$ of each feature is drawn randomly from a uniform distribution inside a spherical volume. The volume is fully contained in the field of view of both cameras, its center minimizes the sum of squared distances to both optical axes, and it has maximal radius. To generate a trajectory, we update the feature's instantaneous position $\mathbf{m}_t$ according to a randomly drawn acceleration vector $\vec{\mathbf{a}}_t$:

$$\mathbf{m}_1 = \mathbf{m}_0 + \vec{\mathbf{a}}_1, \tag{13}$$

$$\mathbf{m}_t - \mathbf{m}_{t-1} = \mathbf{m}_{t-1} - \mathbf{m}_{t-2} + \vec{\mathbf{a}}_t, \quad 2 \leq t \leq T_{\text{feature}} - 1. \tag{14}$$

   The orientation of vector $\vec{\mathbf{a}}_t$ is drawn uniformly from the Gauss sphere and its length is drawn from a normal distribution with mean zero and standard deviation $G_a = 25\ \text{mm/frame}^2$. This choice produces an average projected velocity of two pixels per frame for both cameras, which is approximately equal to that observed in some of our real sequences. The life span of a feature in frames, T_{feature}, is drawn from a uniform distribution in the interval $(0, T]$ and rounded up to the next integral value. By defining a potentially distinct life span for each feature, we simulate the fact that trackers often lose features, either because they become occluded, or because feature matching fails. To ensure that the number of features per frame remains fixed during a single run, we instantiate a new feature at a random position within the bounding sphere when a feature dies.
2. *Noisy image trajectories:* We simulate localization errors in the feature tracker by adding a random

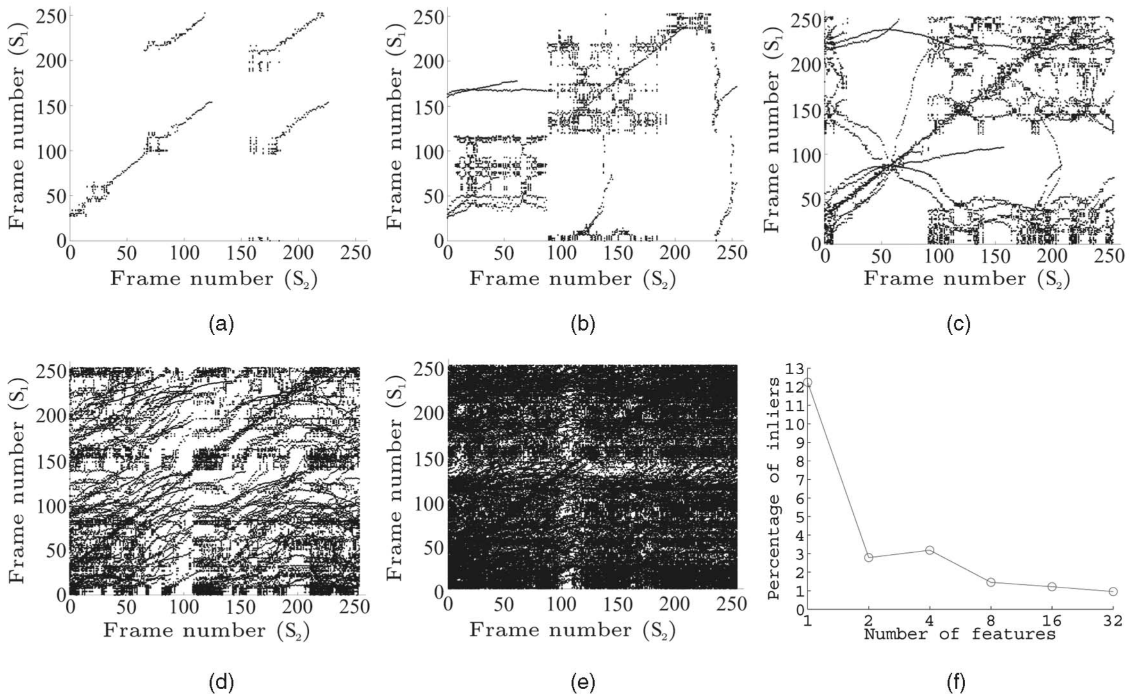

Fig. 10. Voting space examples for number of features (a) $F = 1$, (b) $F = 2$, (c) $F = 4$, (d) $F = 8$, and (e) $F = 16$. The epipolar geometry and tracker localization errors were two pixels in all cases (i.e., $G_e = 2, G_l = 2$). (f) Percentage of inliers for the voting spaces in (a)-(e). A vote is considered to be an inlier if its horizontal distance from the ground-truth timeline is less than one frame.

displacement to the projection of each feature. This displacement has a uniformly distributed orientation and a magnitude drawn from a normal distribution with mean zero and standard deviation G_l.

3. *Noisy initial fundamental matrices:* The timeline reconstruction algorithm in Section 3 requires an initial estimate of the fundamental matrix between the two views. In order to simulate the fact that this matrix may be inaccurate, we perturb the ground-truth fundamental matrix before each run to achieve a predefined epipolar geometry error G_e. For ground truth, we use the fundamental matrix computed in the two-view robots experiment in Section 5.2, normalized to unit Frobenius norm. Given a perturbed fundamental matrix and a set of 3D points, we define its reprojection error to be the root-mean-squared distance between the features' projections and their respective epipolar lines in the two views. The 3D points chosen are the points we used to initialize the fundamental matrix in the actual robot's experiment. To generate a matrix with a given reprojection error G_e, we begin with the ground-truth matrix, add the constant 10^{-5} to each element, measure the reprojection error, and iterate until the error becomes equal to G_e.
4. *RANSAC parameters:* The choice of RANSAC parameters determines the probability that a globally optimal solution will be found and has a major effect on computational cost. To keep this cost at a reasonable level, we constrain the temporal dilation parameter between the two cameras to the interval $[\frac{1}{5}, 5]$ in the timeline reconstruction procedure in Section 3. We then set the RANSAC success probability p to a very conservative level of 0.99. Along with the other RANSAC parameters (Table 1) and the constraint on temporal dilation, this causes 1,840 RANSAC iterations to be executed per run.
5. *Evaluation metrics:* We use the average absolute temporal alignment error ε_t as our basic accuracy metric:

$$\varepsilon_t = \frac{1}{T}\sum_{t_r=0}^{T-1} |(\alpha^* t_r + \beta^*) - (\alpha t_r + \beta)|, \qquad (15)$$

where t_r is the frame number of the reference camera, α^*, β^* are the parameters of the ground-truth timeline, and α, β are the parameters of the computed timeline. We used $\alpha^* = 1$ and $\beta^* = 32$ in all simulations. To illustrate the applicability of our approach in a variety of settings, we consider the percentage of simulation runs that produced timelines with an error below a bound ε_t. This allows us to assess its ability to compute highly accurate timelines ($\varepsilon_t \le 1$ frame) as well as its behavior in less challenging situations (e.g., $\varepsilon_t \le 2$ frames or $\varepsilon_t \le 5$ frames).

6.1 Accuracy versus Number of Features

As a first step, we evaluated the behavior of our temporal alignment and refinement methods for an increasing number F of moving features per frame. Fig. 10 shows examples of voting spaces generated in our simulations. Note that as the number of tracked features increases, the voting space becomes increasingly dense. The voting spaces for 16 and 32 features represent especially challenging

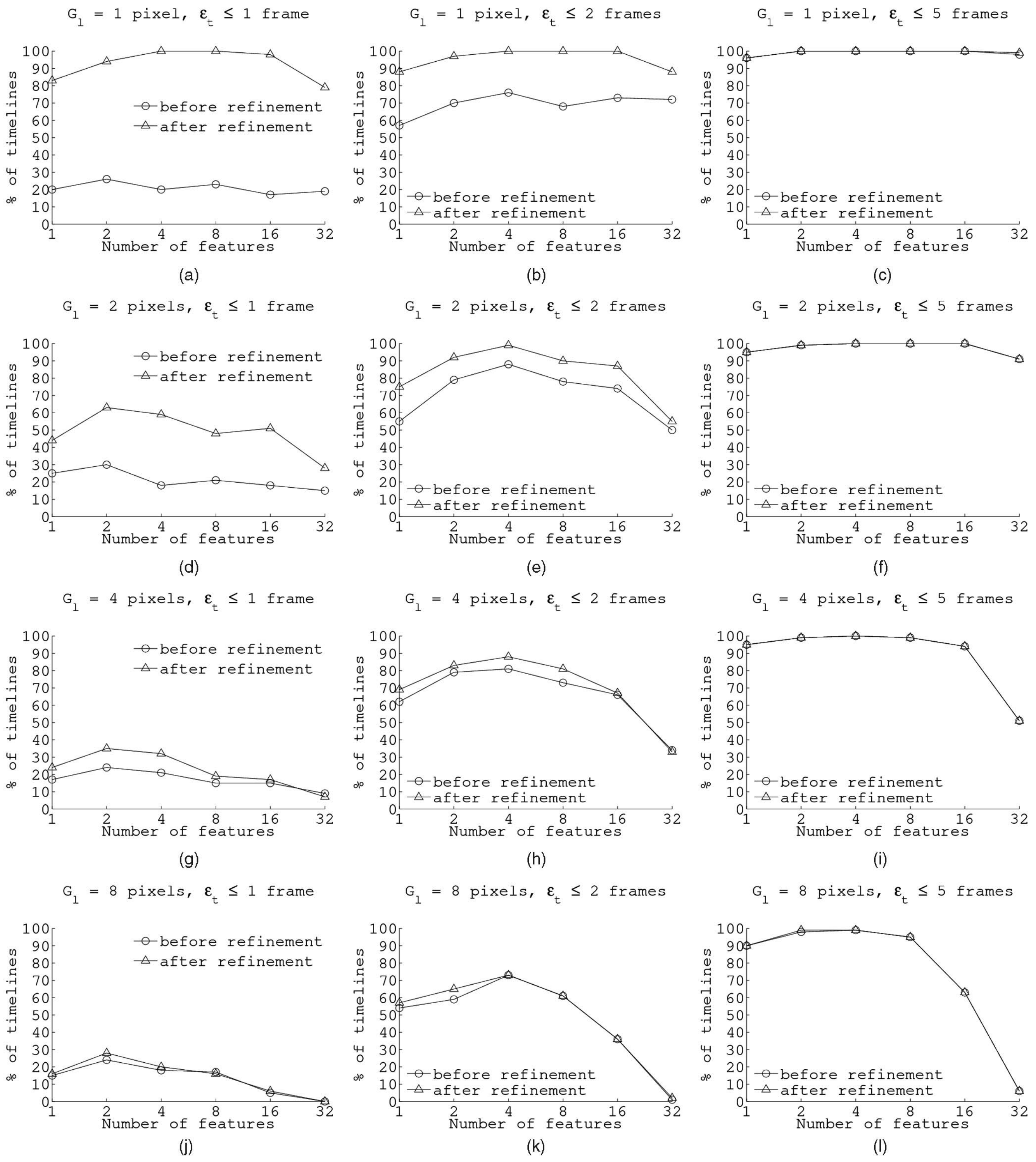

Fig. 11. Alignment error versus number of features for various levels of feature localization error. Each row of plots represents runs with a fixed level of localization error G_l and three different bounds on alignment error ε_t. Each column represents runs with a fixed bound on alignment error and four different levels of localization error. The epipolar geometry error was kept fixed at $G_e = 2$ pixels in all plots. (a), (b) and (c) Percentage of runs for which $G_l = 1$ pixel and the reconstructed timeline had an error below ε_t. (d), (e), (f), (g), (h), (i), (j), (k), (l) Percentage of such runs for different values of G_I and ε_t.

cases, where it is hard to identify the true timeline by inspection. Importantly, the density of votes has a major impact on the ratio of inliers: This ratio is less than 2 percent of the total votes when $F \geq 16$ (Fig. 10f).

Figs. 11 and 12 illustrate the impact of this increasing density on alignment accuracy. The figures show the percentage of runs[3] for which the reconstructed timeline

3. We executed 100 independent runs for each choice of simulation parameters.

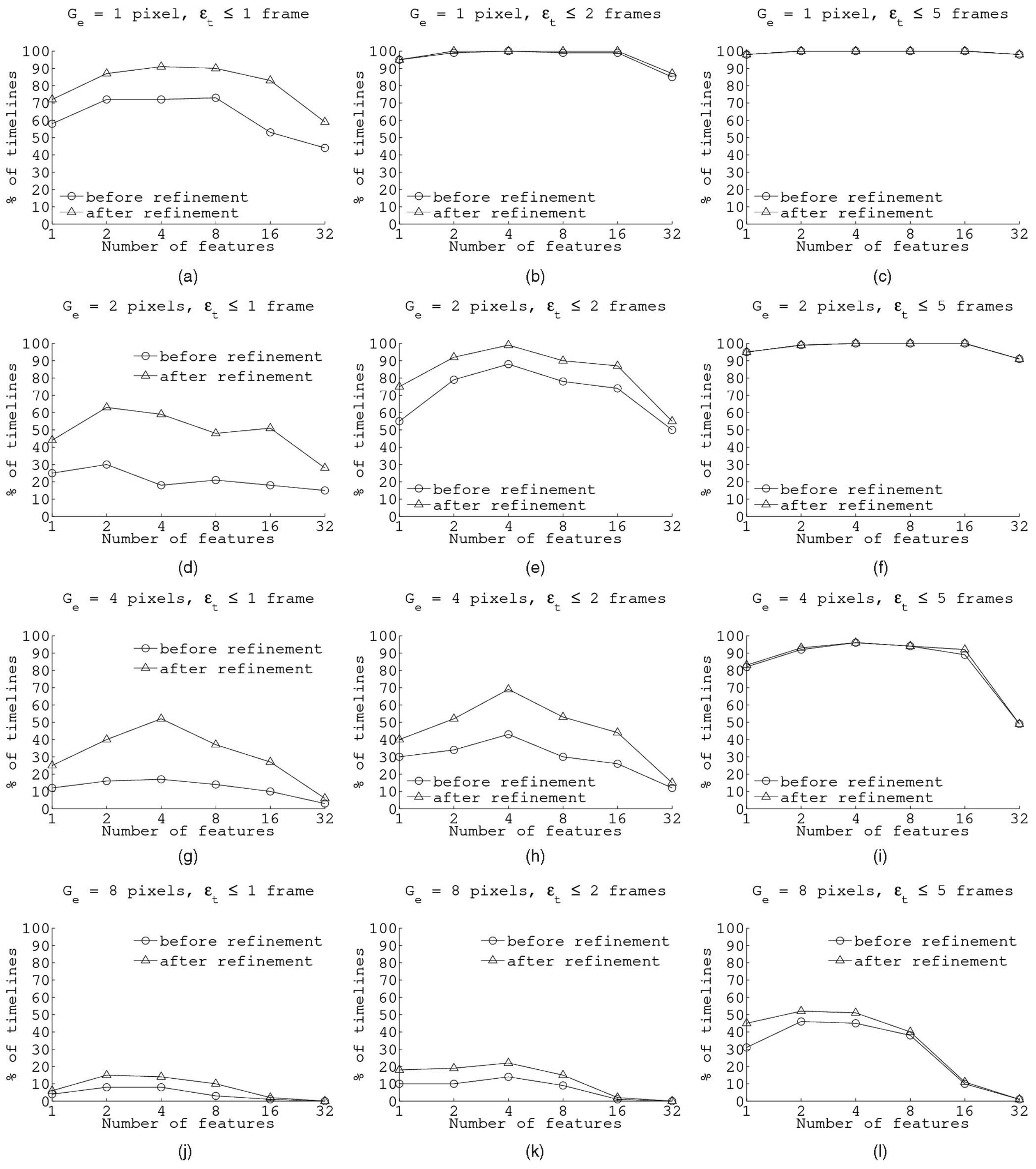

Fig. 12. Alignment error versus number of features for various levels of error in the initial epipolar geometry. Each row of plots represents runs with a fixed level of epipolar geometry error G_e and three different bounds on alignment error ε_t. Each column represents runs with a fixed bound on alignment error and four different levels of epipolar geometry error. The localization error was kept fixed at $G_l = 2$ pixels in all plots. (a), (b) and (c) Percentage of runs for which $G_e = 1$ pixel and the reconstructed timeline had an error below ε_t. (d), (e), (f), (g), (h), (i), (j), (k), (l) Percentage of such runs for different values of G_e and ε_t.

was below a specified bound on alignment error, as a function of the number of tracked features. A percentage near 100 percent for a given error bound implies a near-perfect ability to compute alignments within that bound. These figures lead to several observations about the behavior of the timeline reconstruction and refinement steps.

First, alignment accuracy decreases significantly when the number of features is too large ($F \geq 16$) or too small ($F = 1$). This is because when more features are added, the

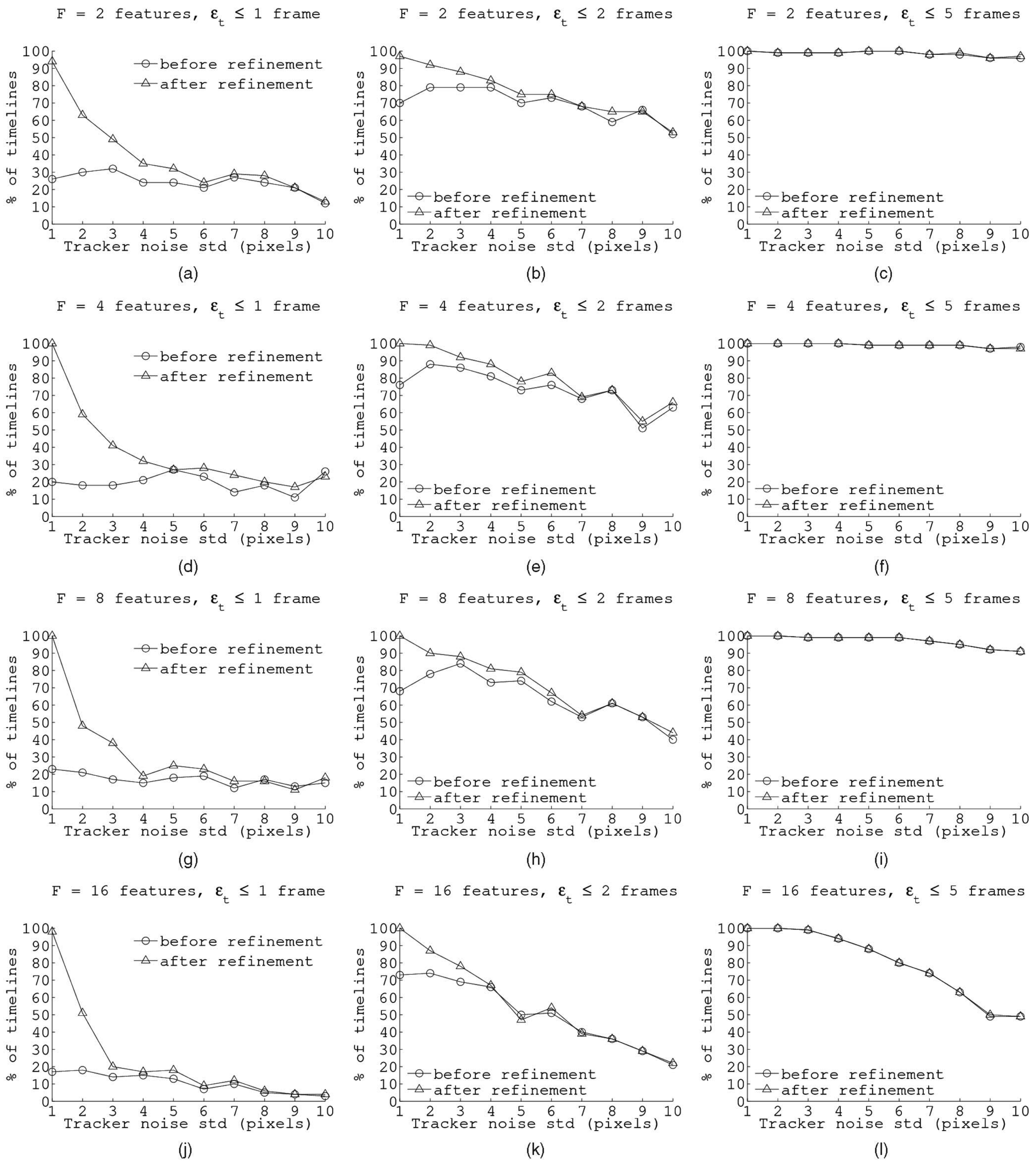

Fig. 13. Alignment error versus localization error for various numbers of features. Each row of plots represents runs with a fixed number F of features, and three different bounds on alignment error ε_t. Each column represents runs with a fixed bound on alignment error and four different numbers of features. The epipolar geometry error was kept fixed at $G_e = 2$ pixels in all plots. (a), (b) and (c) Percentage of runs for which $F = 2$ features and the reconstructed timeline had an error below ε_t. (d), (e), (f), (g), (h), (i), (j), (k), (l) Percentage of such runs for different values of F and ε_t.

number of outliers increases faster than the number of inliers, leading to inaccurate estimation of the timeline parameters for large voting space densities. On the other hand, when the number of features is very small (e.g., $F = 1$), there is an insufficient number of votes in the voting space to enable accurate fitting of the timeline. Second, the approach appears to give optimal results when $F = 4$. This behavior persists across all levels of noise in feature localization and/or epipolar geometry. For instance, even when four features are tracked with a rather high localization error ($G_l = 8$ pixels) and a moderate epipolar geometry error ($G_e = 2$ pixels), the refinement stage is able to compute an alignment within two frames of the ground truth 75 percent of the time (Fig. 11k). In effect, four features

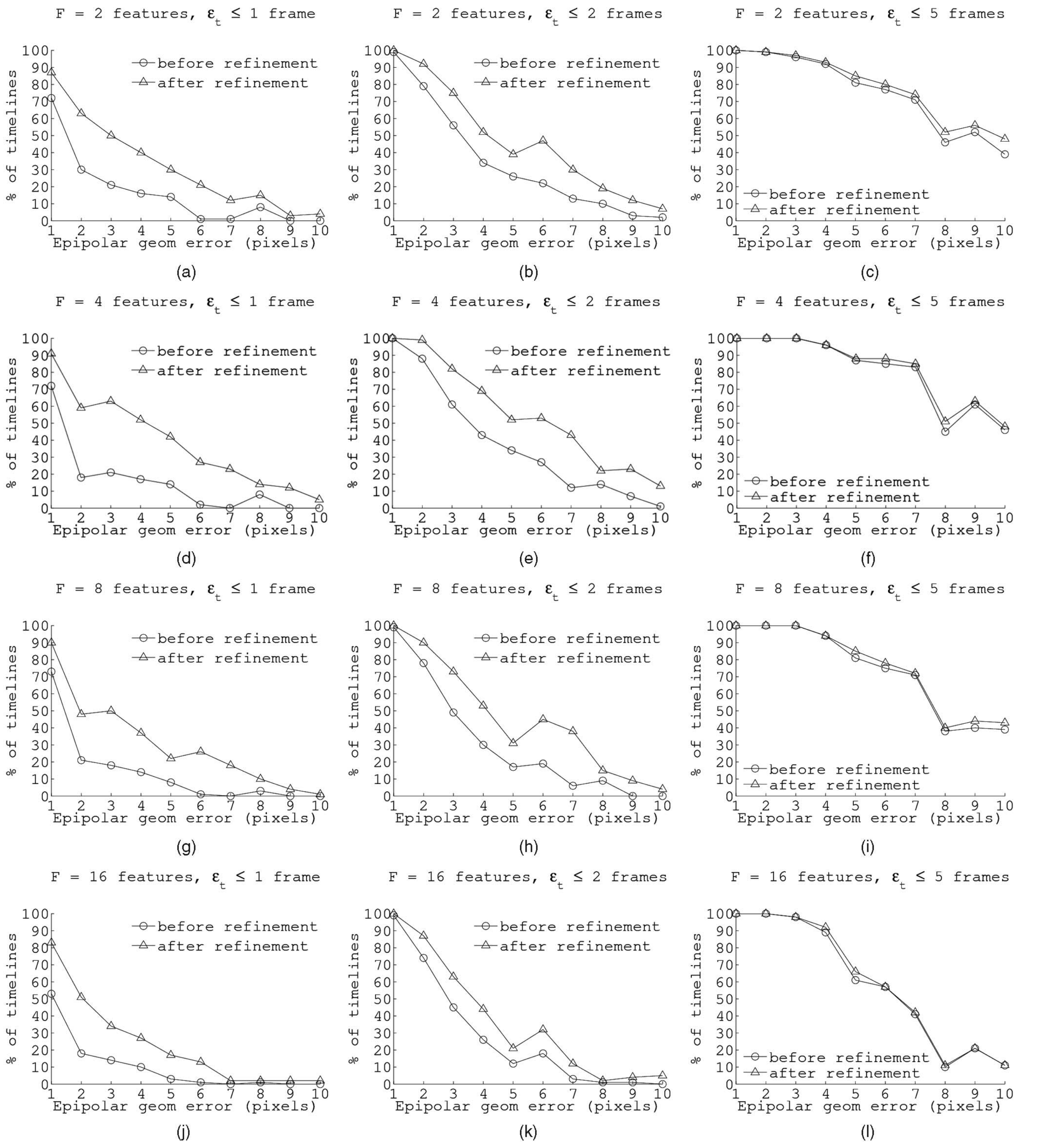

Fig. 14. Alignment error versus error in the initial epipolar geometry for various numbers of features. Each row of plots represents runs with a fixed number F of features and three different bounds on alignment error ε_t. Each column represents runs with a fixed bound on alignment error and four different numbers of features. The localization error was kept fixed at $G_l = 2$ pixels in all plots. (a), (b) and (c) Percentage of runs for which $F = 2$ features and the reconstructed timeline had an error below ε_t. (d), (e), (f), (g), (h), (i), (j), (k), (l) Percentage of such runs for different values of F and ε_t.

provide a good balance between voting space density and fitting accuracy. Third, for low levels of localization error (e.g., $G_l = 1$ pixel), alignment accuracy is much less sensitive to the number of tracked features, especially prior to timeline refinement (Figs. 11a, 11b, and 11c). Fourth, even though timeline refinement can lead to significant accuracy gains, this step is much more sensitive to the number of tracked features. As a result, these gains quickly diminish as F increases beyond eight features. Fifth, timeline refinement is particularly useful for computing highly accurate alignments ($\varepsilon_t \leq 1$ pixel), but has practically no impact when we need only a rough alignment of the input sequences ($\varepsilon_t \leq 5$).

6.2 Accuracy versus Tracking and Calibration Errors

Figs. 13 and 14 show the impact of localization and epipolar geometry error on alignment accuracy. These figures plot the percentage of runs for which the reconstructed timeline was below a specified bound on alignment error, as a function of localization error (Fig. 13) and error in the initial estimate of the fundamental matrix (Fig. 14).

As expected, our ability to achieve accurate alignments diminishes with increased noise levels. This degradation is especially pronounced when more features are present (Figs. 13j, 13k, 13l, 14j, 14k, and 14l). Two reasons explain this degradation in accuracy. First, as the noise levels increase, potential inliers are shifted from their "true" positions in the voting space, and the magnitude of these shifts is proportional to the noise level. This affects the line-fitting process in RANSAC and produces timelines with inaccurate parameters. Second, noise in localization and/or epipolar geometry causes a significant increase in outlier votes, which also affects negatively the accuracy of RANSAC estimation.

Our results also show that high noise levels reduce the impact of the timeline refinement stage. This is especially pronounced in the case of localization error: When a tracker cannot localize features to within 4-5 pixels, the refinement stage has a very small effect on alignment accuracy (first column of Fig. 13). On the other hand, this stage has a consistent, positive effect on accuracy for almost all noise levels in epipolar geometry (first and second columns of Fig. 14). Crucially, this stage is necessary for achieving alignments that are within one frame of the ground truth: For instance, in the experiments reported in Fig. 13, only up to 30 percent of runs produced alignments with such an accuracy prior to refinement, for almost all numbers of features and all noise levels.

7 Concluding Remarks

Our results suggest that the timeline reconstruction algorithm provides a simple and effective way to temporally align multiple video sequences. Unlike previous approaches, it is able to handle temporal dilations and large time shifts, with no degradation in accuracy, even when scene points move along 3D, overlapping, and near-periodic trajectories. Importantly, by reducing the alignment problem to a RANSAC-based procedure, our algorithms are able to tolerate large proportions of outliers in the data, high levels of noise, discontinuities in feature trajectories, complete absence of stereo correspondences for moving features, and sequences that contain multiple frame rates. We are currently investigating the combination of timeline reconstruction and multiview stereo for reconstructing important events in old video footage, where multiple replays of the same event are shown from different viewpoints.

Acknowledgments

An earlier version of this work appeared in [36]. The authors would like to thank Thomas El-Maraghi, Guilherme Pereira, and Matthew Brown for making available their tracking and image stabilization software. Rodrigo Carceroni, Flávio Pádua, and Geraldo Santos thank the support of CNPq-Brazil under Procs. No. 300592/2001-9, No. 478859/2003-1, No. 521259/2001-0, and No. 308195/2004-3; of Fapemig under Procs. No. EDT 162/07, No. CEX-227-04 and No. CEX 491/02; of PRPq-UFMG (Fundo Fundep RD), and of Capes-Brazil. Kiriakos Kutulakos gratefully acknowledges the support of the US National Science Foundation under Grant No. IRI-9875628, the Natural Sciences and Engineering Research Council of Canada under the RGPIN program, and the Alfred P. Sloan Foundation.

References

[1] S. Vedula, S. Baker, and T. Kanade, "Spatio-Temporal View Interpolation," *Proc. Eurographics Workshop Rendering,* pp. 65-76, 2002.

[2] L. Zelnik-Manor and M. Irani, "Event-Based Analysis of Video," *Proc. IEEE Computer Vision and Pattern Recognition Conf.,* vol. 2, pp. II-123-II-130, 2001.

[3] Y. Caspi and M. Irani, "Alignment of Non-Overlapping Sequences," *Proc. Int'l Conf. Computer Vision,* vol. 2, pp. 76-83, 2001.

[4] I. Reid and A. Zisserman, "Goal Directed Video Metrology," *Proc. European Conf. Computer Vision,* pp. 647-658, 1996.

[5] D. Wedge, D. Huynh, and P. Kovesi, "Using Space-Time Interest Points for Video Sequence Synchronization," *Proc. IAPR Conf. Machine Vision Applications,* pp. 190-194, 2007.

[6] L. Wolf and A. Zomet, "Wide Baseline Matching between Unsynchronized Video Sequences," *Int'l J. Computer Vision,* vol. 68, no. 1, pp. 43-52, 2006.

[7] M. Ushizaki, T. Okatani, and K. Deguchi, "Video Synchronization Based on Co-Occurrence of Appearance Changes in Video Sequences," *Proc. Int'l Conf. Pattern Recognition,* pp. 71-74, 2006.

[8] Y. Ukrainitz and M. Irani, "Aligning Sequences and Actions by Maximizing Space-Time Correlations," *Proc. European Conf. Computer Vision,* pp. 538-550, 2006.

[9] O. Shakil, "An Efficient Video Alignment Approach for Non-Overlapping Sequences with Free Camera Movement," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing,* vol. 2, pp. 257-260, 2006.

[10] C. Dai, Y. Zheng, and X. Li, "Accurate Video Alignment Using Phase Correlation," *IEEE Signal Processing Letters,* vol. 13, no. 12, pp. 737-740, Dec. 2006.

[11] C. Dai, Y. Zheng, and X. Li, "Subframe Video Synchronization via 3d Phase Correlation," *Proc. Int'l Conf. Image Processing,* pp. 501-504, 2006.

[12] K. Lee and R.D. Green, "Temporally Synchronising Image Sequences Using Motion Kinematics," *Proc. Image and Vision Computing New Zealand Conf.,* 2005.

[13] I. Laptev, S.J. Belongie, P. Perez, and J. Wills, "Periodic Motion Detection and Segmentation via Approximate Sequence Alignment," *Proc. Int'l Conf. Computer Vision,* vol. 1, pp. 816-823, 2005.

[14] D. Wedge, P. Kovesi, and D. Huynh, "Trajectory Based Video Sequence Synchronization," *Proc. Digital Image Computing: Techniques and Applications Conf.,* pp. 79-86, 2005.

[15] J. Yan and M. Pollefeys, "Video Synchronization via Space-Time Interest Point Distribution," *Proc. Advanced Concepts for Intelligent Vision Systems,* 2004.

[16] D.W. Pooley, M.J. Brooks, A.J. van den Hengel, and W. Chojnacki, "A Voting Scheme for Estimating the Synchrony of Moving-Camera Videos," *Proc. Int'l Conf. Image Processing,* vol. 1, pp. 413-416, 2003.

[17] C. Rao, A. Gritai, M. Shah, and T.S. Mahmood, "View-Invariant Alignment and Matching of Video Sequences," *Proc. Int'l Conf. Computer Vision,* vol. 2, pp. 939-945, 2003.

[18] Y. Caspi, D. Simakov, and M. Irani, "Feature-Based Sequence-to-Sequence Matching," *Int'l J. Computer Vision,* vol. 68, no. 1, pp. 53-64, 2006.

[19] L. Wolf and A. Zomet, "Correspondence-Free Synchronization and Reconstruction in a Non-Rigid Scene," *Proc. Workshop Vision and Modelling of Dynamic Scenes,* 2002.

[20] L. Wolf and A. Zomet, "Sequence to Sequence Self Calibration," *Proc. European Conf. Computer Vision,* vol. 2, pp. 370-382, 2002.
[21] Y. Caspi and M. Irani, "A Step Towards Sequence-to-Sequence Alignment," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 682-689, 2000.
[22] L. Lee, R. Romano, and G. Stein, "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 758-767, Aug. 2000.
[23] G. Stein, "Tracking from Multiple View Points: Self-Calibration of Space and Time," *Proc. DARPA Image Understanding Workshop,* pp. 521-527, 1998.
[24] K. Raguse and C. Heipke, "Photogrammetric Synchronization of Image Sequences," *Proc. ISPRS Commission V Symp. Image Eng. and Vision Metrology,* pp. 254-259, 2006.
[25] W. Anthony, L. Robert, and B. Prosenjit, "Temporal Synchronization of Video Sequences in Theory and in Practice," *Proc. Workshop Motion and Video Computing,* vol. 2, pp. 132-137, 2005.
[26] E. Tola, V. Lepetit, and P. Fua, "A Fast Local Descriptor for Dense Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.
[27] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, pp. 381-395, June 1981.
[28] J. Horst and I. Beichl, "A Simple Algorithm for Efficient Piecewise Linear Approximation of Space Curves," *Proc. Int'l Conf. Image Processing,* vol. 2, pp. 744-747, 1997.
[29] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge Univ. Press, 1988.
[30] K. Atkinson, *An Introduction to Numerical Analysis.* John Wiley and Sons, 1989.
[31] C. Tomasi, "Mathematical Methods for Robotics and Vision," Technical Report CS 205, Stanford Univ., 2000.
[32] J. Hefferon, *Linear Algebra.* Math. Dept. of Saint Michael's College, 2001.
[33] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 10, pp. 1296-1311, Oct. 2003.
[34] FIFA, "FIFA World Cup Archives: Goal of the Century," http://fifaworldcup.yahoo.com/02/en/pf/h/gotc/launch.html, 2002.
[35] M. Brown and D. Lowe, "Recognizing Panoramas," *Proc. Int'l Conf. Computer Vision,* pp. 1218-1225, 2003.
[36] R.L. Carceroni, F.L.C. Padua, G.A.M.R. Santos, and K.N. Kutulakos, "Linear Sequence-to-Sequence Alignment," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 746-753, 2004.

Flávio L.C. Pádua received the bachelor's degree in electrical engineering and the MSc and PhD degrees in computer science from the Universidade Federal de Minas Gerais (UFMG), Brazil, in 1999, 2002, and 2005, respectively. He has been an adjunct professor of computer engineering at the Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) since 2005. From 2001 to 2003, he worked at Oi S/A in Brazil, where he managed engineering projects for increasing the reliability and availability of telecommunication services. From 1998 to 1999, he participated in an undergraduate program at the Technical University of Berlin in Germany, sponsored by the governments of Brazil and Germany. During this period, he worked as a visiting scientist in the Institute for Machine Tools and Factory Management (IWF). His research interests include computer vision and video information processing, with special focus on visual motion analysis and 3D scene analysis from video.

Rodrigo L. Carceroni received the BS and MS degrees in computer science from the Universidade Federal de Minas Gerais (UFMG), Brazil, in 1983 and 1995, respectively, and the MS and PhD degrees in computer science from the University of Rochester in 1997 and 2001, respectively. Following completion of his doctoral studies, he returned to UFMG as a postdoctoral fellow (2001-2002), and was later hired as an assistant professor (2002-2006). Shortly, after obtaining tenure in early 2005, he took a one-year leave from UFMG to work as a postdoctoral fellow at the Grasp Lab at the University of Pennsylvania. In late 2006, he joined Google, Inc., where he is currently a senior software engineer. He was the program chair of the 2006 Brazilian Symposium on Computer Graphics and Image Processing and was twice recipient of the CNPq-Brazil Productivity in Research Award. His research interests include computer vision, computer graphics, and robotics.

Geraldo A.M.R. Santos received the BSc and MSc degrees in computer science from the Universidade Federal de Minas Gerais, Brazil, in 2002 and 2006, respectively. Since then, he has been working in industry on systems for video-based tracking and people counting. His research interests include computer vision and video analysis.

Kiriakos N. Kutulakos received the BA degree in computer science from the University of Crete, Greece, in 1988, and the MS and PhD degrees in computer science from the University of Wisconsin, Madison, in 1990 and 1994, respectively. Following his dissertation work, he joined the University of Rochester, where he was a US National Science Foundation (NSF) postdoctoral fellow and later an assistant professor until 2001. He is currently an associate professor of computer science at the University of Toronto. He won the Best Student Paper Award at CVPR '94, the David Marr Prize in 1999, a David Marr Prize Honorable Mention in 2005, and a Best Paper Honorable Mention at ECCV '06. He is the recipient of a CAREER Award from the US National Science Foundation, a Premier's Research Excellence Award from the government of Ontario, and an Alfred P. Sloan Research Fellowship. He served as program cochair of CVPR 2003 and is currently an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He is a member of the IEEE.