Linear Sequence–to–Sequence Alignment

Rodrigo L. Carceroni, Flávio L. C. Pádua, Geraldo A. M. R. Santos
Dept. de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, MG, CEP 31270-010, Brazil
{carceron,cardeal,massahud}@dcc.ufmg.br

Kiriakos N. Kutulakos
Dept. of Computer Science
University of Toronto
Toronto M5S 3G4, Canada
kyros@cs.toronto.edu

Abstract

We present a novel approach for temporally aligning N unsynchronized sequences of a dynamic 3D scene, captured from distinct viewpoints. Unlike existing methods, which work for $N = 2$ and rely on a computationally-intensive search in the space of temporal alignments, we reduce the problem for general N to the robust estimation of a single line in $\Re^N$. This line captures all temporal relations between the sequences and can be computed without any prior knowledge of these relations. Experimental results show that our method can accurately align sequences even when they have large mis-alignments (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).

1. Introduction

Many applications today benefit from the availability of simultaneous video recordings of the same physical event. Examples include tele-immersion [1], video-based surveillance [2], video mosaicing [3], and video metrology from television broadcasts of athletic events [4]. A critical task in all of these applications is *temporal alignment*—placing every frame of every video recording onto a single, global timeline. Unfortunately, even though video synchronization hardware can be attached to cameras for subsequent recording, it cannot be used to align pre-recorded videos or to align multiple clips in one video stream (e.g., regular and slow-motion clips of the same penalty kick).

We believe that any general solution to the temporal alignment 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.

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 $\Re^N$.

Most existing solutions to the temporal alignment problem conduct an explicit search in the space of all possible alignments [5–10]. Unfortunately, the combinatorial nature of this 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 fifty 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 in a single step, can handle arbitrarily-large misalignments between them, and does not require any *a priori* information about their temporal relations.

Our work is most closely related to the approach of Caspi, Simakov and Irani [5]. In their approach, the epipolar geometry and temporal misalignment between two sequences are recovered from the image trajectory of a single scene point that is visible in both sequences, and are

subsequently refined using more features. To achieve this, they assume that frame rates are known and formulate a non-linear optimization problem to jointly estimate epipolar geometry and temporal misalignment. Unfortunately, the highly non-linear nature of this optimization necessitates good initial estimates for both the temporal misalignment and the epipolar geometry. Importantly, the approach 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 inter-frame 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 a different time with possibly-distinct frame rates. In order to temporally align the resulting sequences, we must determine the correspondence between frame numbers in one sequence and frame numbers in all other sequences. This correspondence can be expressed as a set of linear equations,

$$t_i = \alpha_i \, t_1 \, + \, \beta_i \tag{1}$$

where t_i is the frame number of the i-th sequence and α_i, β_i are unknown constants describing temporal dilation and temporal shift, respectively, between the i-th sequence and the first. In general, these constants will not be integers.

The pairwise temporal relations captured by Eq. (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*:

$$\mathcal{L} = \left\{ [\alpha_1 \, \ldots \alpha_N \,]^T \, t + [\beta_1 \, \ldots \beta_N \,]^T \;\; | \;\; t \in \Re \right\}. \tag{2}$$

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 $\mathbf{q}$ be the instantaneous projection of a moving scene point in camera 1 at frame t_1, expressed in homogeneous 2D coordinates (Figure 1). Furthermore, let $\mathbf{q}_i(t_i)$ 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. If the scene point is visible to all cameras when frame t_1 is captured by camera 1, we have the following constraint:

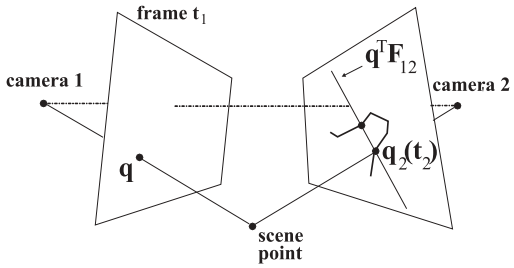

Figure 1. Geometry of the Timeline Constraint. In this two-camera example, the point's trajectory intersects the epipolar line, $\mathbf{q}^T\mathbf{F}_{12}$, twice. If the intersection points were $\mathbf{q}_2(t_2)$ and $\mathbf{q}_2(t_2')$, we have $\mathcal{T}_\mathbf{q} = \{\, [t_1 \;\; t_2]^T \,,\; [t_1 \;\; t_2']^T \,\}$.

Timeline Constraint: The set

$$\mathcal{T}_\mathbf{q} = \left\{ [t_1 \, \ldots t_N \,]^T \;\; | \;\; \mathbf{q}^T \, \mathbf{F}_{1i} \, \mathbf{q}_i(t_i) = 0, \; i = 2 \ldots N \right\}$$

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 $\mathcal{T}_\mathbf{q}$ 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 $\mathbf{q}$ in camera i with the trajectory $\mathbf{q}_i(t_i)$, (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 $[t_1 \, \ldots t_N \,]^T$ is on the timeline it must represent the "true" temporal alignment between the frame of pixel $\mathbf{q}$ and the remaining cameras. Hence, pixels $\mathbf{q}$ and $\mathbf{q}_i(t_i)$ must satisfy the epipolar constraint equation, $\mathbf{q}^T\mathbf{F}_{1i}\mathbf{q}_i(t_i) = 0$. Since, by definition, the set $\mathcal{T}_\mathbf{q}$ 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 cameras' epipolar geometry. 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. 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.

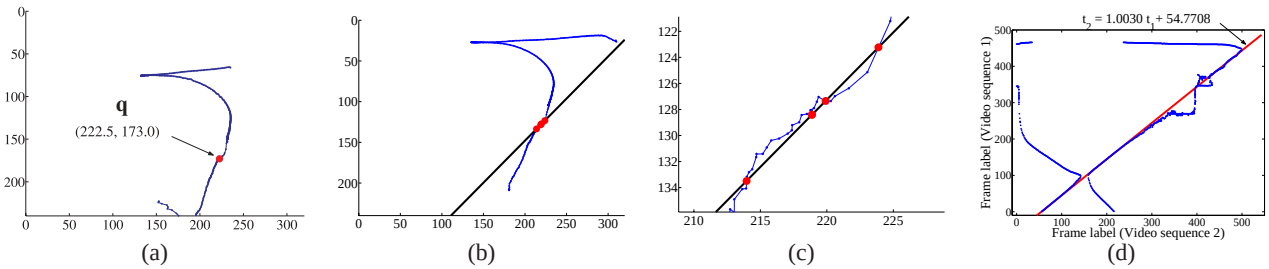

Figure 2. (a) Trajectory of a feature in Sequence 1 of the *Car* dataset (Figure 4). The feature was the centroid of all pixels labeled as "foreground" by a color-based foreground-background detector. (b) Trajectory of the foreground pixel centroid in Sequence 2 of the dataset. Also shown is the epipolar line corresponding to pixel $\mathbf{q}$ 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 $\mathbf{q}$ intersects multiple line segments along the trajectory. (d) Exploiting the Timeline Constraint for two-sequence alignment. Each point represents a candidate temporal alignment, i.e., an element of $\mathcal{T}_\mathbf{q}$ for some feature location, $\mathbf{q}$, in (a). The reconstructed timeline, drawn as a solid line, describes the temporal alignment of the two sequences in the *Car* dataset.

The algorithm operates in two phases. In the first phase, we choose one of the image sequences to be the "reference" sequence and use pixels $\mathbf{q}$ from that sequence to estimate $\mathcal{T}_\mathbf{q}$ for each $\mathbf{q}$. In the second phase, we fit an N-dimensional line $\mathcal{L}$ to the union of the estimated sets $\mathcal{T}_\mathbf{q}$. To fully specify this algorithm we must therefore ask three questions: how do we select the pixels $\mathbf{q}$, how do we estimate the set $\mathcal{T}_\mathbf{q}$ and how do we compute the line $\mathcal{L}$?

To select the pixels $\mathbf{q}$, we apply a feature detector to every frame of the reference sequence and compute a distinct set $\mathcal{T}_\mathbf{q}$ for each location returned by the detector. Hence, the selection of $\mathbf{q}$ does not involve any temporal processing of the reference sequence.

To estimate the set $\mathcal{T}_\mathbf{q}$ for a given $\mathbf{q}$, we use a two-frame feature tracker and rely on an initial estimate of the fundamental matrices, $\mathbf{F}_{ij}$. Specifically, by applying the tracker to every pair of adjacent frames in the i-th sequence, we obtain a collection of line segments. Each line segment connects the location of a feature that was detected in some frame of the i-th sequence and was successfully tracked to the next frame (Figure 2a-c). When such a line segment intersects the epipolar line of $\mathbf{q}$, it defines a possibly-fractional frame number, t_i, corresponding to the instant that $\mathbf{q}$'s epipolar line intersects the image trajectory of a point in the scene. Hence, t_i is the i-th coordinate of a potential element of $\mathcal{T}_\mathbf{q}$. To generate $\mathcal{T}_\mathbf{q}$, we collect all the t_i coordinates computed for all sequences and concatenate them so that they form valid N-dimensional vectors.[1]

The set of candidate temporal alignments is the union of the sets $\mathcal{T}_\mathbf{q}$ for all $\mathbf{q}$. In general, this union will contain a large number of outliers (Figure 2d). To reconstruct the timeline in the presence of outliers, we use the RANSAC algorithm [11]. 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 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.

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 the reference camera.

Let $\mathbf{q}$ be the projection along camera 1 of a scene point in

[1] Note that if the epipolar line of $\mathbf{q}$ intersects two line segments in each of $N-1$ cameras, we have a total of 2^{N-1} possible ways of "concatenating" the computed t_i coordinates into an N-dimensional vector. To avoid including an exponential number of vectors in $\mathcal{T}_\mathbf{q}$, we only include vectors that are consistent with the cameras' epipolar geometry. In particular, we concatenate coordinates t_i and t_j for cameras i and j, respectively, only if the intersection points that defined them are near each others' epipolar lines. Note that our concatenation procedure is conservative, i.e., it guarantees that the set of vectors generated this way will be a superset of $\mathcal{T}_\mathbf{q}$.

frame t_1. Furthermore, suppose that the point's projection in camera i traces a known linear trajectory,

$$\mathbf{q}_i(t_i) = (1 - t_i)\mathbf{r}_i \; + \; t_i\mathbf{s}_i \tag{4}$$

with $\mathbf{r}_i, \mathbf{s}_i$ known pixels on the image plane of camera i. This trajectory jointly constrains the estimated timeline $\mathcal{L}$ and the estimated fundamental matrix, $\mathbf{F}_{1i}$. Specifically, we use our estimate of $\mathcal{L}$ to compute the location on this trajectory corresponding to pixel $\mathbf{q}$ in frame t_1:

$$\mathbf{q}_i^* = [1 - (\alpha_i t_1 + \beta_i)]\,\mathbf{r}_i \; + \; (\alpha_i t_1 + \beta_i)\mathbf{s}_i \;\; . \tag{5}$$

This location must also satisfy the epipolar constraint,

$$\mathbf{q}^T \, \mathbf{F}_{1i} \, \mathbf{q}_i^* = 0 \;\; . \tag{6}$$

By combining Eqs. (5) and (6) we obtain a homogeneous equation that "couples" the estimated timeline parameters α_i, β_i and the elements of the fundamental matrix $\mathbf{F}_{1i}$:

$$(1 - \alpha_i t_1 - \beta_i)\, \mathbf{q}^T \mathbf{F}_{1i} \mathbf{r}_i \; + \; (\alpha_i t_1 + \beta_i)\, \mathbf{q}^T \mathbf{F}_{1i} \mathbf{s}_i = 0 \;\; . \tag{7}$$

In practice, errors in $\mathcal{L}$ and $\mathbf{F}_{1i}$ will cause Eq. (7) not to be satisfied exactly, representing a non-zero algebraic distance between pixel $\mathbf{q}_i^*$ and its associated epipolar line. To refine the current estimate of the timeline and of the cameras' epipolar geometry, we expand Eq. (7) by introducing unknown refinement terms $\Delta\mathbf{F}_{1i}, \Delta\alpha_i$ and $\Delta\beta_i$

$$\begin{aligned} \mathbf{F}_{1i} &\longleftarrow \mathbf{F}_{1i} + \Delta\mathbf{F}_{1i} \\ \alpha_i &\longleftarrow \alpha_i + \Delta\alpha_i \\ \beta_i &\longleftarrow \beta_i + \Delta\beta_i \end{aligned}$$

and solving for these terms in order to minimize algebraic distance across all scene points and all input frames.[2]

In general, the image trajectory of an arbitrarily-moving scene point will not be linear. To handle this general case, we apply the above refinement procedure to the line segments returned by the two-frame tracker we use to estimate $\mathcal{T}_{\mathbf{q}}$ (Section 3). This leads to an iterative refinement algorithm that consists of two steps. In the first step, the current estimates of $\mathbf{F}_{1i}$ and $\mathcal{L}$ are used to select, for every feature detected in camera 1, one line segment for each sequence i. In the second step, the estimates of $\mathbf{F}_{1i}$ and $\mathcal{L}$ are updated using the linear method outlined above. These two steps are repeated until convergence.

5. Experimental Results

To demonstrate the effectiveness of our timeline reconstruction algorithm, we tested it on several challenging two- and three-view datasets (Figure 4). Image dimensions in all datasets were about 320×240 pixels. The sequences represented a wide variety of conditions, including sequences that ranged from 55 to 605 frames; temporal misalignments of 21 to 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 a 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 handle all of our datasets 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 of El-Maraghi, Fleet and Jepson [12]. 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 can be 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 lower bound on ground-truth accuracy is critical in evaluating the results below.

Two-view *Car* dataset As a first test, we applied our technique to a two-view sequence used by Caspi and Irani [13] for evaluating their method (Figure 4). The data was acquired by two cameras with identical frame rate of 25fps, 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 (Figures 2a and 2b). To compute the cameras' fundamental matrix we used twenty six correspondences between background pixels in the two views.

Figure 2d shows the timeline reconstructed using the RANSAC-based algorithm of 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 of 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., below the accuracy of the ground-truth alignment. Note that these results are at least as accurate as those of Caspi and Irani, even though we are solving a less constrained problem (i.e., α is unknown and scene planarity is

[2] For every scene point and every frame in which it is visible, we obtain one equation in the 10 unknown parameters that completely determine $\Delta\mathbf{F}_{1i}$, $\Delta\alpha_i$, and $\Delta\beta_i$. This equation is not linear because it contains the second-order terms $(\Delta\alpha_i \Delta\mathbf{F}_{1i})$ and $(\Delta\beta_i \Delta\mathbf{F}_{1i})$. In our implementation, we simplify computations by ignoring these terms and solving the resulting over-determined system of linear equations.

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).

Two-view *Robots* dataset In a second experiment, we used two cameras operating at 30fps to acquire images of four small robots, as they executed small random movements on two planes (Figure 4). 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 was challenging for four reasons. First, the robots' inter-frame 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 that are non-smooth, complicating the shape of each blob's trajectory.

The timeline reconstructed with $\epsilon = 2.0$ prior to refinement is shown in Figure 5a. 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. Figure 3 confirms that the computed alignment is quite good, despite the robots' slow motion and the tracker's poor performance.

Two-view *Juggling* dataset In this dataset, two people are observed by a wide-baseline camera pair while juggling five uniformly-colored balls (Figure 4). Both sequences were acquired at a rate of 30fps. This dataset 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 cyclical, (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 non-exhaustive search [6] or non-linear optimization [5] because of the possibility of getting trapped in deep local minima. To make the alignment problem even more challenging, we modified this dataset 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). Figures 5d-f show the reconstructed timelines before the refinement stage, with $\epsilon = 0.5$. The average temporal misalignment error was 0.75 frames for the original dataset. The refinement stage brought this error down to 0.26 frames, with $\alpha = 1.0004$ and $\beta = -40.80$.

Three-view *Soccer* dataset As a final experiment, we applied our technique to three video clips extracted from a single MPEG-compressed TV broadcast of a soccer match [15]. The clips were replays of the same goal filmed from three distinct viewpoints (Figure 4). 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 computing the frame-to-frame homography using Brown and Lowe's system [14]. 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 dataset contained $N = 3$ views, the timeline is a 3D line with 3-vectors as its coefficients (see Eq. (2) and Figures 5b and 5c). 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 in Figure 4) 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 Figure 3), despite the low quality of the videos and their unequal frame rates.

6. Concluding Remarks

Our results suggest that timeline reconstruction provides a simple and effective method for temporally aligning 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 three-dimensional, overlapping and almost–cyclical 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 fea-

tures, and sequences that contain multiple frame rates. We are currently investigating the combination of timeline reconstruction and multi-view stereo for reconstructing important events in old video footage, where multiple replays of the same event are shown from different viewpoints.

Acknowledgments We 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 and No. 521259/2001-0; of Fapemig under Proc. No. CEX 491/02; of PRPq-UFMG (Fundo Fundep RD), and of Capes-Brazil. Kiriakos Kutulakos gratefully acknowledges the support of the National Science Foundation under Grant No. IRI-9875628, of the Natural Sciences and Engineering Research Council of Canada under the RGPIN program, and of the Alfred P. Sloan Foundation.

References

[1] S. Vedula, S. Baker, and T. Kanade, "Spatio-temporal view interpolation," in *Proc. Eurographics Workshop on Rendering*, pp. 65–76, 2002.

[2] L. Zelnik-Manor and M. Irani, "Event-based analysis of video," in *Proc. CVPR*, v. 2, pp. 123–130, 2001.

[3] Y. Caspi and M. Irani, "Alignment of non-overlapping sequences," in *Proc. 8th ICCV*, v. 2, pp. 76–83, 2001.

[4] I. Reid and A. Zisserman, "Goal directed video metrology," in *Proc. 4th ECCV*, pp. 647–658, 1996.

[5] Y. Caspi, D. Simakov, and M. Irani, "Feature-based sequence-to-sequence matching," in *Proc. Workshop on Vision and Modelling of Dynamic Scenes*, 2002.

[6] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood, "View-invariant alignment and matching of video sequences," in *Proc. 9th ICCV*, pp. 939–945, 2003.

[7] L. Wolf and A. Zomet, "Correspondence-free synchronization and reconstruction in a non-rigid scene," in *Proc. Workshop on Vision and Modelling of Dynamic Scenes*, 2002.

[8] L. Wolf and A. Zomet, "Sequence-to-sequence self calibration," in *Proc. 7th ECCV*, v. 2, pp. 370–382, 2002.

[9] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: Establishing a common coordinate frame," *IEEE T-PAMI*, v. 22, pp. 758–767, 2000.

[10] G. Stein, "Tracking from multiple view points: Self-calibration of space and time," in *Proc. IUW*, pp. 521–527, 1998.

[11] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM*, v. 24, n. 6, pp. 381–395, 1981.

[12] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," in *Proc. CVPR*, v. 1, pp. 415–422, 2001.

[13] Y. Caspi and M. Irani, "A step towards sequence-to-sequence alignment," in *Proc. CVPR*, v. 2, pp. 682–689, 2000.

[14] M. Brown and D. Lowe, "Recognising panoramas," in *Proc. 9th ICCV*, pp. 1218–1225, 2003.

[15] FIFA, "FIFA World Cup archives: Goal of the century," *http://fifaworldcup.yahoo.com/02/en/pf/h/gotc/launch.html*, 2002.

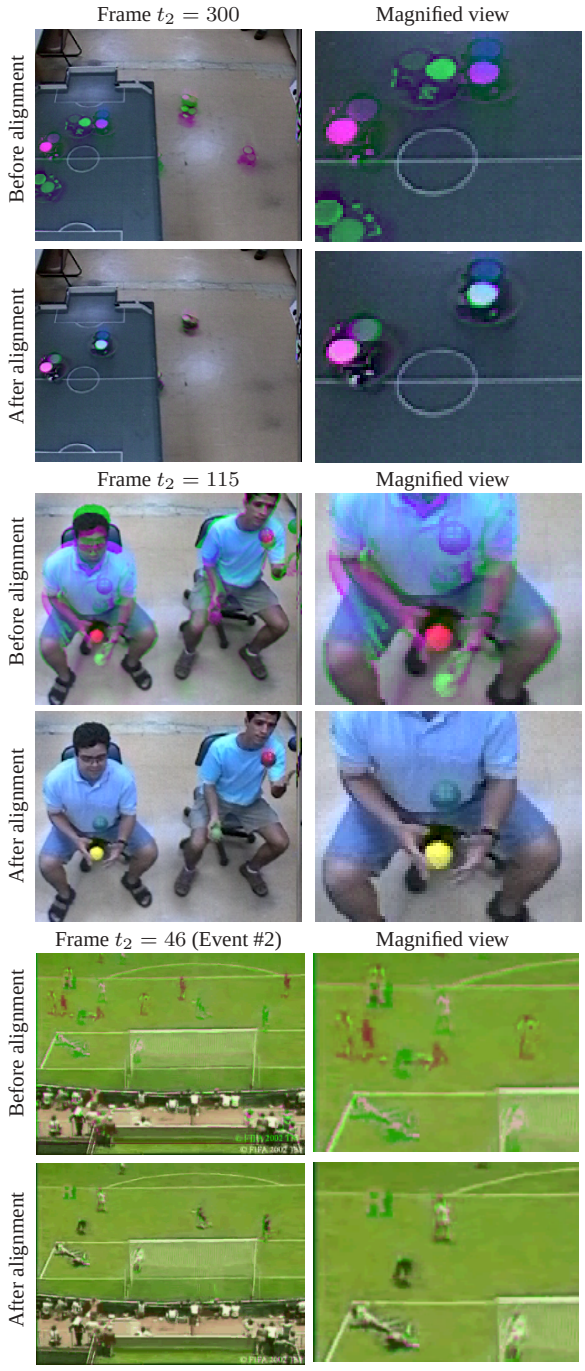

Figure 3. Using synthetic images to assess alignment quality for the datasets of Figure 4. **Before alignment images** were created by superimposing the green band of a frame t_2 with the red and blue bands of frame $t_1^* = (t_2 - \beta^*)/\alpha^*$ using ground truth timeline coefficients α^* and β^*. **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).

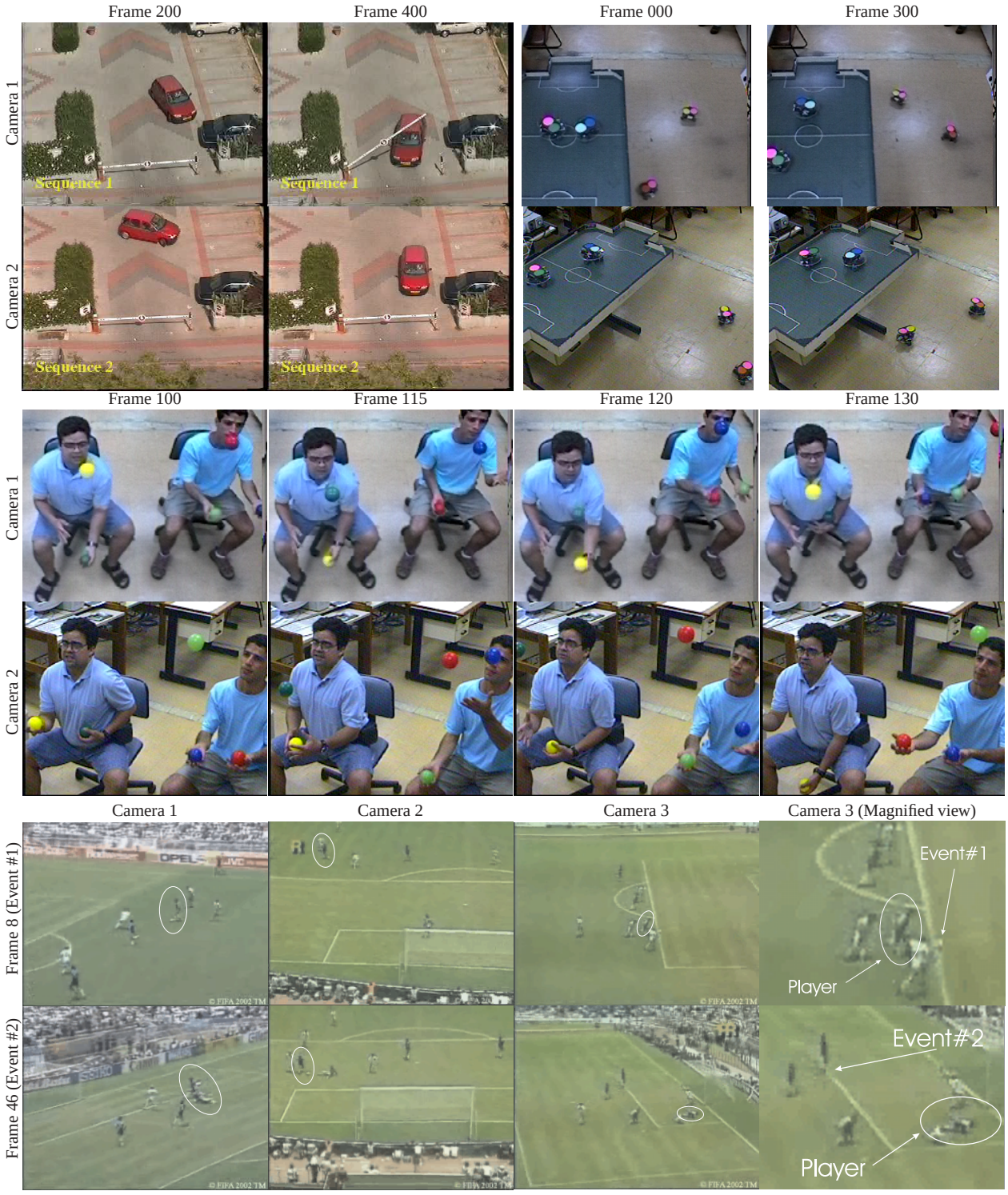

Figure 4. Top left: Two out of 470 frames from the two-view *Car* dataset. **Top right:** Two out of 605 frames from the two-view *Robots* dataset. **Middle:** Four out of 260 frames from the two-view *Juggling* dataset. **Bottom:** Two out of 55 frames from the three-view *Soccer* dataset. Ellipses indicate the player being tracked by the WSL tracker. These datasets, along with more experimental results and videos, are available at *http://www.cs.toronto.edu/~kyros/research/timeline/* .

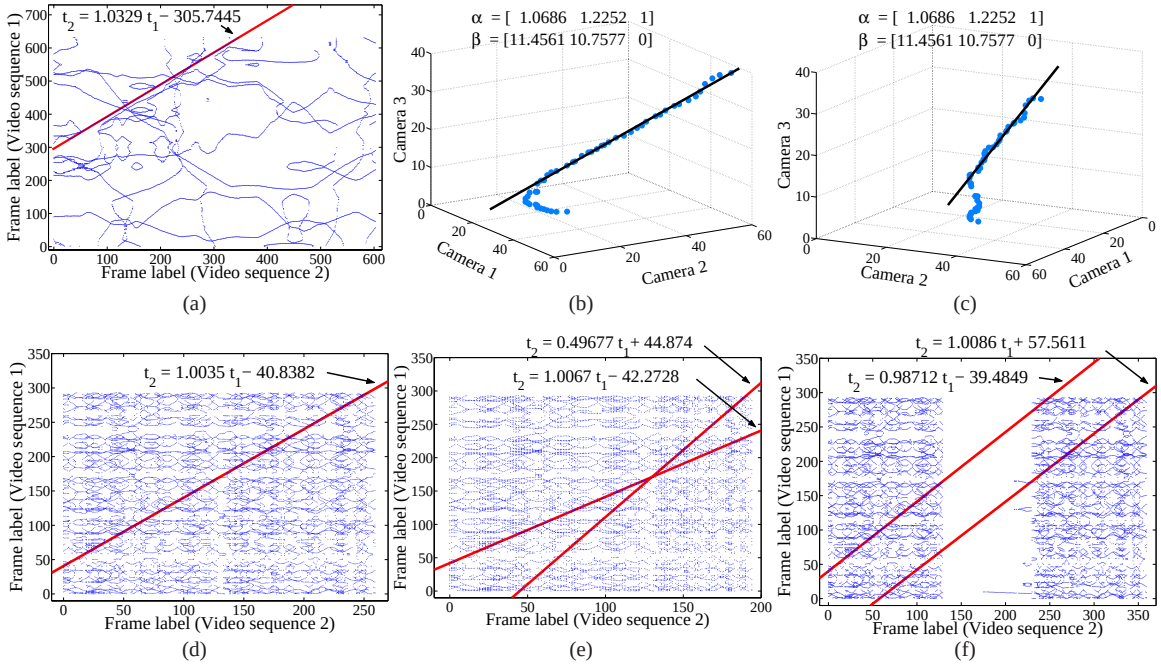

Figure 5. Voting spaces, timelines, and timeline equations recovered prior to refinement. Each point is an element of $\mathcal{T}_{\mathbf{q}}$ for some feature $\mathbf{q}$ in the reference sequence. (a) *Robots* dataset. (b),(c) Two views of the 3D voting space and 3D timeline computed for the *Soccer* dataset. (d) *Juggling* dataset. (e) Modified *Juggling* dataset: all odd-numbered frames from the last half of Sequence 1 were removed to simulate a 2× increase in frame rate. Timelines were computed by applying RANSAC to fit two lines to the voting space. (f) Modified *Juggling* dataset: 100 spurious frames were inserted in Sequence 2 and timelines were reconstructed as in (e). Note that the reconstructed timelines contain sufficient information to "join" the separated segments of the original sequence.

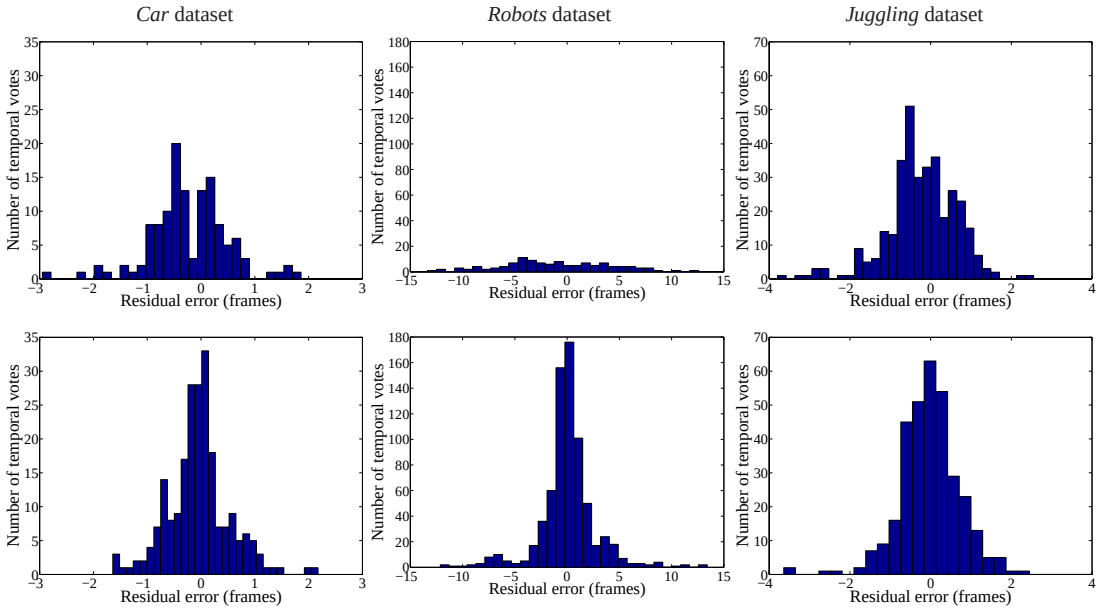

Figure 6. Distribution of distances of inlier votes from the reconstructed timeline. **Top row:** Distribution before the timeline refinement stage. **Bottom row:** Distribution after the refinement stage. Note that the updated epipolar geometry and updated timeline parameters reduce the distance between inliers and the timeline and cause more votes to be labeled as inliers.