VIDEO SEQUENCE SYNCHRONIZATION

This thesis is

presented to the

School of Computer Science and Software Engineering

for the degree of

Doctor of Philosophy

of

The University of Western Australia

By

Daniel Wedge

July 2, 2007

Declaration

This thesis is the author's own composition. All sources have been acknowledged and the author's contribution is clearly identified in this thesis. For any work in this thesis that has been co-published with other authors, the permission of all co-authors has been obtained to include this work in this thesis.

This thesis was completed during the course of enrollment in this degree at the University of Western Australia and has not previously been accepted for a degree at this or another institution.

Author	Date
Principal Supervisor	Date

Abstract

Video sequence synchronization is necessary for any computer vision application that integrates data from multiple simultaneously recorded video sequences. With the increased availability of video cameras as either dedicated devices, or as components within digital cameras or mobile phones, a large volume of video data is available for processing by a growing range of computer vision applications that process multiple video sequences. To ensure that the output of these applications is correct, accurate video sequence synchronization is essential.

Whilst hardware synchronization methods can embed timestamps into each sequence on-the-fly and require no post-processing, they require specialized hardware and it is necessary to set up the camera network in advance. On the other hand, computer vision-based software synchronization algorithms can be used to post-process video sequences recorded by cameras that are not networked, such as common consumer hand-held video cameras or cameras embedded in mobile phones, or to synchronize historical videos for which hardware synchronization was not possible.

The current state-of-the-art software algorithms vary in their input and output requirements and camera configuration assumptions. Many algorithms operate without requiring knowledge of the geometry relating the two cameras, however, for most of these algorithms it is necessary to specify the sequences' frame rate ratio. Only a handful of algorithms exist that recover the frame rate ratio of two sequences, and of these, most require the cameras' geometry to be known. Most synchronization algorithms require frame-to-frame object motion or object trajectories throughout an entire sequence to be provided as input data from which the

synchronization can be recovered. In this thesis, I present three algorithms for recovering both the frame offset and the ratio of the frame rates of pairs of video sequences. One of my algorithms does not require trajectory data as input, and only one of the three algorithms requires weak camera calibration.

Firstly, I present an algorithm that uses the motion of a single tracked object to synchronize two video sequences recorded by stationary cameras with fixed intrinsic parameters. The algorithm is unique in that it synchronizes a pair of video sequences using the trajectory of a single object tracked throughout each sequence and does not require camera calibration. A coarse-to-fine approach is used. At the coarse level, each sequence is divided into a number of sub-sequences, and corresponding sub-sequences from each sequence are determined. From this, an initial estimate of the ratio of frame rates is proposed and a voting scheme is used to provide bounds on the frame offset. The fine level of synchronization involves a search for the frame offset and frame rate ratio that minimize a measure of synchronization based on epipolar geometry. It is shown that this measure, whilst not new, is preferred for use in place of the reprojection error used by many synchronization algorithms.

Next, I describe an approach that synchronizes two video sequences where an object exhibits ballistic motions. Given the epipolar geometry relating the two cameras and the imaged ballistic trajectory of an object, the algorithm uses a novel iterative approach that exploits object motion to rapidly determine pairs of temporally corresponding frames. This algorithm accurately synchronizes videos recorded at different frame rates and takes few iterations to converge to sub-frame accuracy. Whereas the method presented by the first algorithm integrates tracking data from all frames to synchronize the sequences as a whole, this algorithm recovers the synchronization by locating pairs of temporally corresponding frames in each sequence.

Finally, I introduce an algorithm for synchronizing two video sequences recorded by uncalibrated stationary cameras. This approach is unique in that it recovers both the frame rate ratio and the frame offset of the two sequences by finding matching space-time interest points that represent events in each sequence; the algorithm does not require object tracking. RANSAC-based approaches that take

a set of putatively matching interest points and recover either a homography or a fundamental matrix relating a pair of still images are well known. This algorithm extends these techniques using space-time interest points in place of spatial features, and uses nested instances of RANSAC to also recover the frame rate ratio and frame offset of a pair of video sequences.

In this thesis, it is demonstrated that each of the above algorithms can accurately recover the frame rate ratio and frame offset of a range of real video sequences. Each algorithm makes a contribution to the body of video sequence synchronization literature, and it is shown that the synchronization problem can be solved using a range of approaches.

Preface

Some of the work presented in this dissertation has been already been published.

At the 2005 *Digital Image Computing: Techniques and Applications* conference, I introduced an algorithm that uses the motion of a single object to recover the frame offset of two video sequences recorded by cameras with unknown epipolar geometry [49]. Later, I extended this algorithm to also recover the frame rate ratio of the two sequences. This extended algorithm is presented in Chapter 3.

At the *Asian Conference on Computer Vision* in 2006, I presented a paper [47] that forms the basis of the basic algorithm given in Chapter 4. The extension to the algorithm presented in Chapter 4 was developed later and is introduced in this thesis.

A paper presented at the 2007 *IAPR Conference on Machine Vision Applications* [48] describes the general case of the algorithm presented in Chapter 5. The special case of the algorithm as described in Chapter 5 is introduced in this dissertation and has not been published previously.

Each of these papers was co-authored by myself and my two supervisors, Dr. Du Huynh and Dr. Peter Kovesi. In each case, I was the principal contributing author.

Acknowledgments

This thesis has occupied my life for the past four years. I am extremely grateful to my supervisors, Du Huynh and Peter Kovesi, for firstly allowing me to explore various research topics before settling on video sequence synchronization, and for their guidance and encouragement thereafter. I would like to thank them for the opportunities they have provided for me to attend and present at conferences. I would also like to thank Cara MacNish who supervised me for several months before I decided that my calling was in Computer Vision. A general thank you also goes out to all other staff in the School of Computer Science & Software Engineering at UWA who frequently provided encouragement.

This research was funded by an Australian Postgraduate Award and a top-up scholarship from the School of Computer Science & Software Engineering. Without this support, I would not have been able to perform this research. I would also like to acknowledge the University Research Committee for providing funding to allow me to travel to present a paper at the 2006 Asian Conference on Computer Vision.

Thank you to my friends who have travelled with me on this journey through university life. Though it's been a long journey and a lot of hard work, we've shared a lot of fun times that I'll always remember.

Lastly and most importantly, without the eternal support of my parents, I would not have come this far in my education and life. Mum and Dad, I owe everything to you and am forever grateful for your support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of computer vision researchers is to enable computers to understand the 3D world as captured by one or more cameras in a series of 2D images or video sequences. In today's world, still and video cameras can be found everywhere. The increased availability of cameras has been accompanied by an increase in the volume of captured image and video data. As such, techniques for processing such a large volume of data are essential. Computer vision algorithms for recovering structural information from still images are well-developed and more recently, many applications for processing video sequences have been introduced.

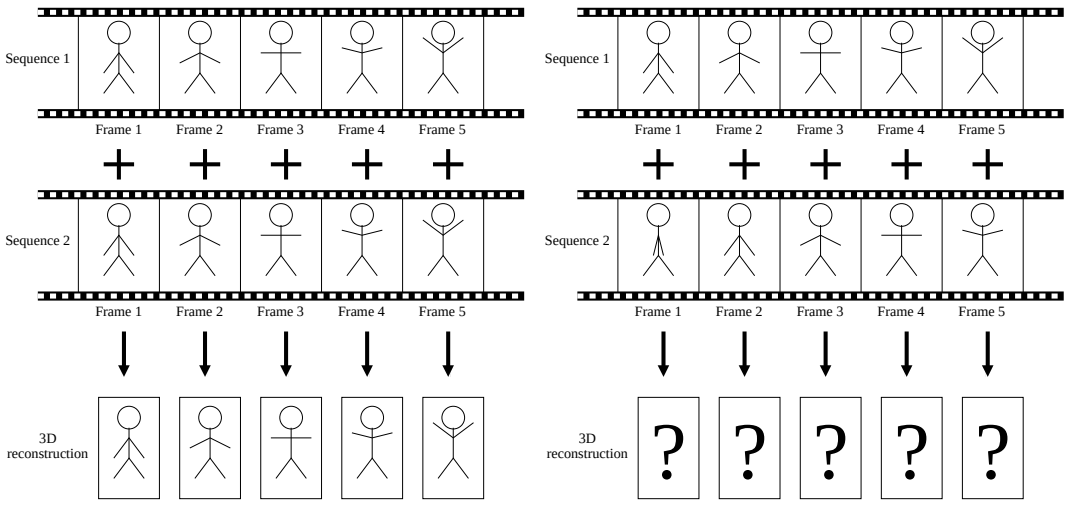
Video sequence synchronization is a fundamental problem required to be solved in order to allow video data from multiple independently recorded sources to be accurately integrated into a single model. For a simple demonstration of this, consider the human visual system. We take it for granted that the world viewed through each of our eyes is a snapshot of the world at that particular instant. However, consider what we would see if the visual information from one eye was delayed. Each eye would view slightly different world 'models', rather than different views of the same world model, and it would be extremely difficult, if not impossible, to accurately understand the 3D world from these unsynchronized views. Similarly, in computer vision applications, video sequence synchronization is essential to ensure consistency in the recovered data.

Many computer vision applications already exist that require video sequence synchronization and it is expected that many further methods will be developed in

the coming years. Video surveillance is a common example of a network of many video cameras recording a scene. In playing back a large number of video sequences for simultaneous viewing, it is advantageous if the videos are synchronized such that simultaneously recorded frames from the video sequences are displayed in sync.

Other applications may require synchronization for quantitative rather than qualitative purposes. Virtualized reality [15] involves performing an action in front of a network of cameras and reconstructing a 3D snapshot of the action at each time instant. Accurate video synchronization is essential in order to ensure consistency in the structure recovered from such video sequences. This is demonstrated in Figure 1(a), where a 3D model is successfully reconstructed from two synchronized cameras. However, the unsynchronized cameras in Figure 1(b) provide inconsistent data and a consistent 3D model cannot be reliably recovered.

Video sequence synchronization can also be useful in the comparison of human motions [31], such as dancing routines. Rao et al. demonstrated that the performances of two dancers executing the same actions can be analyzed such that the relative rates of execution of the actions can be recovered and potentially used as a training tool.



(a) Synchronized sequences (b) Unsynchronized sequences

Figure 1: In (a), the two video sequences are synchronized and the recovered 3D structure is consistent with the data from both views. In (b), a frame misalignment exists and the sequences are not synchronized. As a result of this, the 3D structure will be affected and the recovered orientations of the person's arms may be nonsensical.

The above applications use video sequences recorded in a controlled environment. A computer vision application that has taken advantage of the recent proliferation of digital cameras and online distribution of photos is the Photo Tourism system [37] developed by Snavely et al. Their system collects a large number of photos of a landmark from which a sparse 3D reconstruction of the landmark and the relative camera locations are automatically recovered. The user can interact with the system via a GUI that displays the relative positions of each camera and the photos taken at each location. As video cameras become increasingly popular and are embedded in ubiquitous devices such as mobile phones, it is possible that demand may arise for a similar application for processing many video sequences of an event recorded independently by people in different locations.

Video synchronization can be performed online in hardware or offline in software; this is discussed in more detail in the following chapter. I define the problem of video sequence synchronization as the process of recovering the temporal relationship between two or more video sequences. Most algorithms focus on solving for a linear temporal relationship between two video sequences. In this case, an initial temporal offset, Δ, exists between the sequences, and each camera may film at different frame rates; let α denote the ratio of the frame rates of the two cameras. Together, let α and Δ be known as the synchronization parameters of the pair of sequences. Then the two video synchronization problem can be expressed mathematically via the synchronization equation:

$$j = \alpha i + \Delta, \tag{1}$$

where i and j denote the indices of frames from each sequence recorded at the same instant in time. Synchronizing a network of multiple (i.e., more than two) video sequences can be approached in a pairwise manner.

In some applications, it may be useful to solve for a non-linear temporal relationship between sequences, though this is uncommon. For example, instead of synchronizing two videos of an action or event captured simultaneously by two cameras, the objective may be to synchronize videos of similar actions occurring at different times. Rao et al. [31] presented an example of this where the same action was performed by different dancers, with the rate of motion varying between

dancers. In this thesis, unless otherwise stated, I consider only a linear temporal relationship between sequences and as such, recovering the synchronization means that the linear relationship encapsulated in Equation (1) is solved.

1.1 Thesis structure and contributions

In this dissertation, I present three algorithms for synchronizing pairs of video sequences recorded by stationary cameras with fixed intrinsic parameters. My algorithms extend known techniques and introduce new approaches to the synchronization problem. Although I only consider synchronizing pairs of sequences in my algorithms, it is envisaged that the algorithms can be trivially extended to allow synchronization of sets of more than two video sequences in a pairwise manner.

A literature review is presented Chapter 2. I review existing video sequence synchronization algorithms, and examine the different assumptions made by each algorithm and the cases in which they can be used.

Next, each of Chapters 3, 4 and 5 introduces one of my synchronization algorithms. I do not summarize them here, instead, I place them into context in Section 2.3 after presenting the literature review. Each of these chapters is self-contained in that each chapter firstly presents an algorithm, examines results of applying that algorithm to various pairs of video sequences, and then discusses matters relevant to that algorithm.

In Chapters 3 and 4, manual tracking is used to recover the ball's trajectory in many video sequences. However, these algorithms are not reliant on manual tracking or any other manual inputs; the output of any suitable object tracking algorithm can be supplied as input for my synchronization algorithms. Just as a feature detection algorithm is not reliant on the camera used to capture the images to be processed, my algorithms are not reliant on manual tracking. This is demonstrated in Chapter 3 where the centroid of a moving light source is tracked automatically throughout some sequences and successful synchronization is achieved. These synchronization algorithms are shown to achieve synchronization when trajectory data is missing from each sequence, for example, where the tracked object is occluded or

moves out of the camera's field of view. However, as with other trajectory-based synchronization algorithms, it is essential that the object tracker does not produce outlying data points, e.g., where the tracker loses the object in clutter.

1.2 Notation

Throughout this thesis, the following notation conventions are used:

- The ratio of frame rates of a pair of sequences is always denoted by α.
- The frame offset of a pair of sequences is always denoted by Δ, whose values are measured in frames.
- The two video sequences are denoted by $\mathcal{S}$ and $\mathcal{S}'$.
- All vectors are column vectors and are denoted by lowercase boldface characters, e.g., $\mathbf{g}$. Vectors in $\mathcal{S}'$ are distinguished by a prime, e.g., $\mathbf{l}'_j$. Unless stated otherwise, vectors represent points and lines in homogeneous coordinates.
- Matrices are denoted by uppercase characters in a fixed-width font, e.g., `F`.
- Ground truth values of the synchronization parameters are denoted by an overhead bar, i.e., $\bar{\alpha}$ and $\bar{\Delta}$ denote the ground truth values of the frame rate ratio and frame offset, respectively, of a pair of sequences.

  In the experiments presented in the following chapters, unless otherwise stated, the ground truth frame rate ratio was known from the camera settings and the ground truth frame offset was obtained by manually locating frames in the sequences in which an event occurred such as a ball bouncing. In the example illustrated in Figure 2, the ball can be seen bouncing between frames of a sequence. Therefore, as the location of the bouncing event is available to half-frame accuracy, the ground truth frame offset for sequences containing a bouncing ball is accurate to half a frame.

(a) Frame 200 (b) Frame 201 (c) Frame 202 (d) Frame 203

Figure 2: Four frames from a sequence where the ball bounces between frames. The ball's shadow can also be seen, showing that the ball is in contact with the ground between frames 201 and 202; the bounce event is estimated to have occurred at 201.5 frames into the sequence.

- The recovered synchronization parameters are denoted by a hat, i.e., $\widehat{\alpha}$ and $\widehat{\Delta}$ denote the recovered frame rate ratio and frame offset, respectively, of a pair of sequences.

Chapter 2

Literature review

As outlined in the previous chapter, video sequence synchronization is essential for an ever-increasing range of applications. A number of video synchronization methods already exist and of these methods, a clear distinction can be made between hardware and software based approaches. In this chapter, hardware synchronization is outlined, and a large number of software synchronization algorithms are reviewed and compared. Also, the algorithms that I will present in Chapters 3, 4 and 5 are briefly introduced, and compared and contrasted with relevant algorithms in the current literature.

2.1 Hardware synchronization

There are several ways in which videos can be synchronized in hardware, however only a brief mention is given here such that a comparison can be made with software synchronization methods. One hardware synchronization approach involves sending a synchronization signal to each camera, and embedding a timestamp in each captured frame of the recorded sequences [14, 15]. In post-processing, the timestamps of each frame, and hence the synchronization, can be recovered.

The advantages of using hardware synchronization are that the synchronization is known exactly and that the amount of post-processing can be minimized. However, these advantages may be offset by the necessity of specialized and possibly

expensive hardware, and the requirement that the cameras must be set up for synchronization before recording. This procedure may involve laying cables from a central location to each camera or between cameras, which may not be practical if the cameras are widely separated or if insufficient time is available to set up the camera network before recording.

2.2 Software approaches

Software synchronization approaches use visual cues, e.g., lighting changes, or visible features, such as points on moving objects or object trajectories as a whole, for synchronization. Advantages of using software synchronization methods include the ability to synchronize videos without requiring the setup procedure or the equipment necessary for hardware synchronization. However, software synchronization involves significant offline post-processing of video data, and the approaches may be restricted in their application. For example, a certain camera model may be assumed or the ratio of frame rates of pairs of cameras must be known.

There are two general classes of software synchronization algorithms: feature-based algorithms and direct alignment algorithms. Feature-based algorithms operate by tracking objects or detecting features within sequences. From trajectory or feature correspondences, an algebraic or geometric error is computed and used as a measure of synchronization. Direct alignment algorithms do not attempt to match features between images; rather, most direct alignment methods use pixel intensities or gradients for synchronization.

2.2.1 Feature-based algorithms

Naturally, many feature-based synchronization methods are based on multiple view geometry [13]. A homography is a linear transformation that can be used to map points between two projective views of a reference plane. For scenes containing motion occurring on a reference plane, a homography can be used for synchronization since a point on the reference plane in one frame will be transferred to the corresponding location in the corresponding frame in the other sequence if the

synchronization is known. Alternatively, the fundamental matrix encapsulates the epipolar geometry relating two cameras; enforcing the epipolar constraint can be used as method of synchronizing two video sequences containing free motion in 3D space.

Reid and Zisserman [32] used a homography in synchronizing two videos of a critical passage of play in a soccer match, where the sequences were recorded at the same frame rate. Firstly, ground markings were used to estimate the homography relating the two views; this homography was then used to transfer moving points (players' shadows on the ground plane) between views. The two sequences were aligned by searching for the temporal offset that minimized the reprojection errors of moving points on the ground plane. Similarly, with object motion occurring mainly on the dominant ground plane, Stein [39] and Lee et al. [24] synchronized multiple surveillance videos by estimating a homography for every frame offset; the offset with the homography that yielded the lowest reprojection error was declared to be the actual frame offset of the two sequences. Nunziati et al. [29] also synchronized video sequences with motion occurring on the ground plane by tracking an object and identifying features on its trajectory. These features were used to propose temporally corresponding frames from each sequence from which the frame offset was recovered. The homography relating the views was also estimated from trajectory information surrounding each trajectory feature. Velipasalar and Wolf [46] demonstrated an alternative approach to synchronizing sequences containing motion occurring on a plane, using projective invariants to find initial trajectory and frame correspondences from which the frame offset was recovered.

A feature-based algorithm by Caspi and Irani [3, 4] synchronized two sequences recorded by cameras with fixed internal parameters filming at a known ratio of frame rates. The algorithm recovered the temporal offset of the two sequences by integrating multiple trajectory observations; it also estimated a homography or fundamental matrix relating the two views (depending on the type of motion contained in the scene). Corresponding trajectories from each sequence were proposed based on shape properties of the trajectories; each trajectory contributed multiple point correspondences between sequences. A RANSAC-based approach [9] recovered the

spatial and temporal transformations. At each RANSAC iteration, a proposed pair of corresponding trajectories were randomly selected. The spatial and temporal transformations were computed by initially performing an exhaustive search through a range of integer frame offsets, then commencing an iterative process that alternately refined the temporal offset, and the homography or fundamental matrix relating the two trajectories. Finally, the spatio-temporal transformations with the most inlying trajectories (as determined by a distance threshold) were refined using data from all inlying trajectories. Noguchi et al. [28] described a similar framework for recovering the frame offset of sequences where only one object was tracked throughout both sequences.

An algorithm developed by Carceroni et al. [1] employed the epipolar constraint to synchronize N video sequences (for $N \geq 2$) recorded by stationary cameras. A fundamental matrix was firstly estimated for each pairing of sequences with a reference sequence, using known corresponding stationary points from each view. Rather than tracking a number of points throughout entire sequences, feature points were only tracked between consecutive frames. Then, tentative sets of synchronized frames were proposed where the epipolar constraint was satisfied, i.e., a point in one sequence crossed the epipolar line due to the proposed corresponding point in the proposed corresponding frame in another sequence. The frame numbers of these tentative synchronized frames were used to form points in an N-dimensional space. A line was fitted to these points via RANSAC; the line encapsulated the ratio of frame rates and temporal misalignment between all pairs of video sequences.

Pooley et al. [30] solved for the synchronization of two sequences captured by two moving cameras using a fundamental matrix based algorithm. A set of stationary background points tracked through each video sequence was used to estimate the fundamental matrix between views. Instead of searching through the $\alpha - \Delta$ parameter space, they used a Hough transform on a reparameterized space of α and Δ, resulting in an estimate of the new parameters. A gradient descent method was then used to refine the parameters, using reprojection error as a measure of synchronization.

Tuytelaars and Van Gool [43] also presented a method to synchronize sequences

captured by two freely moving cameras. In their algorithm, five points undergoing non-rigid motion were tracked throughout each sequence and matching points from each sequence were determined. Instead of using a measure of synchronization based on 2D error (such as reprojection error), their method used a 3D analysis based on back-projecting lines into a 3D affine space. The distances between 3D lines at each offset were stored in a compatibility matrix, and from this the temporal offset was recovered.

Another algorithm for synchronizing sequences captured by moving cameras was introduced by Lei and Yang [25]. They synchronized three sequences by employing points, lines, and the trifocal tensor, where the frame rate ratios for pairs of cameras were known. Points and lines were tracked through each sequence and matched between sequences. A trifocal tensor was proposed at every pair of integer frame offsets (one frame offset between sequences 1 and 2, and another offset between sequences 1 and 3), and the error in reprojecting points and transferring lines between sequences calculated; the frame offset pair that minimized this error was proposed as an estimate of the correct synchronization. This estimate was refined to sub-frame accuracy by using the Levenberg-Marquardt algorithm to minimize the reprojection and transfer error. Lei and Yang also demonstrated that their algorithm could be extended to synchronize sequences recorded by a network of more than three cameras, by initially synchronizing 3 sequences, and then synchronizing further groups of 3 sequences where at least one of the sequences belonged to the first group of sequences to be synchronized. It was shown that the Levenberg-Marquardt algorithm could be applied to synchronize such sequences to sub-frame accuracy.

Whitehead et al. [50] introduced an alternative approach for synchronizing three or more weakly calibrated stationary cameras. Multiple objects were tracked throughout each sequence and feature points in each trajectory, such as sharp changes in direction, were located. An initial estimate of the synchronization parameters and trajectory correspondences from each sequence were established by locating inflection points from each sequence that satisfied the epipolar geometry.

Temporally corresponding frames from all sequences were then determined by locating frames where the object's location in all views satisfied the epipolar geometry, thus allowing the frame offset and frame rate ratio to be recovered for all pairs of sequences.

Whereas the aforementioned works used features visible in both sequences, another series of methods were developed for synchronize sequences recorded by two jointly moving cameras, where the cameras' fields of view did not overlap and the camera centres were almost coincidental. Caspi and Irani [2] showed that synchronization can be achieved if the inter-frame transformations within a sequence can be recovered. Since the cameras were jointly moving, similar transformations occurred in both sequences at the same instant in time. Their algorithm was based on the vectors of eigenvalues of similar transformation matrices differing only in scale. Alternative measures of transformation similarity were proposed by Spencer and Shah [38]. Later, Korah and Rasmussen [17] extended Caspi and Irani's algorithm to synchronize N sequences, for $N \geq 2$. They proposed a method to simultaneously recover the homographies relating each camera to a reference camera.

An alternative geometric approach by Zhou and Tao [54] used the projectively invariant cross-ratio [13] as a tool in recovering the temporal offset of two sequences. They determined an object's location in two frames of each sequence, and assumed that it was moving with a constant velocity. Then in one view, the epipolar lines for the object's location in the two frames in the other view were computed and the intersection points of these epipolar lines with the object's linear trajectory were determined. The cross ratio was then computed from these four collinear points, and the temporal offset recovered from the cross-ratio.

Some algorithms use an algebraic measure of synchronization rather than a geometric reprojection error. The factorization method of Tomasi and Kanade [40] showed that in the case of orthographic projection, a measurement matrix constructed from the image locations of feature points on a rigid object tracked through a number of frames has rank 3. Later, Costeira and Kanade [6] extended the factorization method to allow for independently moving objects. Wolf and Zomet [51, 52] used this approach to synchronize two sequences captured at the same frame rate.

A set of points were tracked throughout each sequence but trajectory correspondences between sequences were not determined; a measurement matrix was then constructed from these point trajectories. Wolf and Zomet proposed two measures of synchronization. One measure was based on the principal angles of subspaces spanned by the columns of matrices constructed from point trajectories, where the differences in the principal angles were to be minimized. The other measure was a rank-based constraint, where a number of singular values of the measurement matrix were expected to be zero due to the factorization rank constraint. These singular values may be non-zero due to noise, or if measurements were taken from frames captured at different instants in time. The sum of the singular values beyond the expected rank bound of the measurement matrix was used as a measure of synchronization, and the frame offset could be recovered. Tresadern and Reid [42] used a similar rank-based method in synchronizing to sub-frame accuracy sequences recorded by affine cameras; they also solved for an unknown ratio of frame rates. However, their algorithm varied in that it required a number of trajectory correspondences from each sequence to be known.

Rao et al. [31] developed an alternative rank-constraint based approach that synchronized sequences recorded by perspective cameras. In their algorithm, they solved for a non-linear temporal relationship between two video sequences, for example, two dancers performing the same routine at the same location, but at different and non-constant tempos. The idea of applying the synchronization problem to biological motions was earlier proposed by Giese and Poggio [10]. Rao's algorithm synchronized two video sequences from a single trajectory, with a set of matched background points from each sequence used to initialize the algorithm. Whereas Wolf and Zomet used the affine measurement matrix, Rao et al. used the $n \times 9$ measurement matrix used in the linear estimation of the fundamental matrix, where $n \geq 8$ is the number of corresponding points from each view. In the synchronized, noise-free case, this matrix should be of rank 8; the smallest singular value (the 9th) was used as a measure of departure from synchronization. Rao et al. used a coarse-to-fine approach, at each level determining pairs of corresponding frames from each sequence via a dynamic programming method known as dynamic time warping.

Previously, dynamic time warping has been used for speech recognition [33], which is essentially attempting to synchronize a 1D signal. Tresadern and Reid [41] also used the 9th singular value of this projective measurement matrix as a basis for a measure for synchronization in the case of perspective cameras.

A different trajectory-based approach to recovering the frame offset of sequences was outlined by Kuthirummal et al. [19]. Their algorithm required knowledge of the trifocal tensor relating three stationary cameras with fixed intrinsic parameters. It was assumed that the frame rates of all sequences were equal and that the frame offset of the first two sequences was known. Thus, only the frame offset of the third sequence relative to the first two was unknown. An object was tracked through each sequence and the trajectory data from the first two synchronized sequences was transferred to the third via the trifocal tensor. By taking the Fourier transform of this transferred trajectory and the actual observed trajectory in the third view, the unknown frame offset was recovered via the time shift property of the Fourier transform.

An alternative feature used for synchronization is the *frontier point*, a point on a curved surface where the tangent plane at the frontier point coincides with the epipolar plane [5]. Frontier points in silhouette images project to the boundary of the silhouette. Sinha and Pollefeys [35, 36] used frontier points to synchronize sequences of object silhouettes recorded by cameras operating at the same frame rate. A RANSAC-based approach firstly proposed a temporal offset and randomly selected lines tangential to the silhouette's convex hull. These lines were proposed as potential epipolar lines and their point of intersection was proposed as the epipole. Frontier points from other frames of the sequences were used to verify the proposed spatial and temporal models.

Yan and Pollefeys [53] described a synchronization algorithm based on the distribution of space-time interest points [22]. These features are often detected when objects collide or change directions. A number of significant space-time interest points were detected in each sequence, and for each sequence, a histogram was computed describing the temporal distribution of interest points throughout the sequence. Then, the histograms were cross-correlated over the range of possible

temporal offsets. The actual temporal offset was proposed to have occurred at the frame offset with the maximum correlation score.

2.2.2 Direct alignment algorithms

Direct alignment [3] uses pixel intensities in a video frame for synchronization and is suitable for videos containing significant lighting changes, e.g., fireworks, or flickering fire. A direct alignment algorithm by Caspi and Irani [3] synchronized two sequences in a coarse-to-fine manner. Firstly, a Gaussian sequence pyramid was computed, which is the video sequence equivalent of a Gaussian image pyramid. At each level of the pyramid, an iterative algorithm minimized the sum-of-squared differences in pixel intensities between sequences according to the current estimate of the spatio-temporal model.

Another intensity-based algorithm was proposed by Wolf and Zomet [52] who divided two sequences into a number of short sub-sequences. A matrix was constructed to record synchronization scores for matching each sub-sequence from one sequence to each sub-sequence from the other. The score was a rank-based measure computed from a matrix constructed from intensity gradients in the spatial and temporal dimensions. In the matrix of synchronization scores, correctly matched sub-sequences were located along a straight line; from the gradient of this line, the ratio of frame rates of the sequences was recovered, and the offset of the line from the origin represented the frame offset.

Shechtman and Irani [34] described a template-matching method for locating a template sequence (a video sequence with small spatial dimensions and consisting of few frames) within a much larger sequence. Although this was not a synchronization algorithm, it used the concept of correlating two video sequences. Ushizaki et al. [45] recovered the frame offset of two sequences by firstly computing regions of each frame containing intensity variations. Then for each sequence, a 1D signal was computed that described the pixel intensity of the region over time. The signals representing the two sequences were cross-correlated over a range of frame offsets; the normalized cross-correlation score reached a peak at the correct frame offset.

Ukrainitz and Irani [44] used a correlation-based approach to synchronizing sequences related by an affine spatial transformation. A Gaussian sequence pyramid was computed and an iterative algorithm used to recover the spatial and temporal parameters by maximizing a similarity measure. The similarity measure used in this algorithm was based on the normalized correlation score between pixel intensities within a pair of space-time patches, one patch from each sequence. Within one sequence, a patch was proposed around each pixel, and the corresponding patch from the other sequence computed from the estimate of the spatio-temporal model.

An alternative direct alignment algorithm was presented by Dai et al. [7, 8] who demonstrated that phase correlation, used to recover the spatial translation between two images, can be extended to 3D to also recover the frame offset of two video sequences. Since phase correlation is not robust to changes in viewpoint, their iterative algorithm assumed that the scene was planar and recovered a homography relating the two views. At each iteration, the algorithm refined the estimate of the homography, then applied the homography to one sequence and recomputed the frame offset via the 3D phase correlation method.

2.3 The algorithms presented in this thesis

In this section, my algorithms are introduced and I compare and contrast them with the relevant literature presented in the previous sections.

Chapter 3 describes an algorithm that synchronizes two long video sequences where a single object is tracked throughout each sequence. It uses a coarse-to-fine approach where at the coarse level, an estimate of the ratio of frame rates and bounds on the frame offset are proposed and at the fine level, an algebraic measure of synchronization is used to search for the synchronization parameters. This is the same measure as used by Rao et al. [31] However, my algorithm and Rao's algorithm vary in that my algorithm assumes that the frame rate of each camera is constant but Rao's algorithm can recover a non-linear temporal relationship (i.e., Rao et al. treat α in Equation (1) as a function of time whereas I treat it as a constant). Hence, Rao's algorithm is able to synchronize sequences of the same

actions performed by different people, however it requires weak camera calibration and the motions must be performed at the same location. My algorithm does not require camera calibration but it does require the recorded videos to contain the same motion.

The algorithm that I describe in Chapter 3 also shares some similarities with Tresadern and Reid's algorithm [42] in that the frame rate ratio and frame offset can be recovered from object trajectories alone, using a rank-based approach to synchronization. Their algorithm assumed an affine camera model and required multiple object trajectories to be computed and matched between sequences. A later algorithm by Tresadern and Reid [41] synchronized sequences recorded by perspective cameras, however it operated by computing a cost function for every pairing of frames and again required multiple points (in this case 8 or more) to be tracked throughout each sequence, though not all points were required to be in motion. Due to the coarse-to-fine approach employed by my algorithm, it is expected that the cost function will be computed fewer times compared to Tresadern and Reid's algorithm.

Chapter 4 describes an algorithm in which the known epipolar geometry and projectile motion are exploited such that pairs of corresponding frames from a pair of video sequences are recovered rapidly in a novel iterative manner. From multiple such frame correspondences, the algorithm is shown to recover the frame rate ratio of the two sequences, and the frame offset is recovered to sub-frame accuracy. This work is similar to the algorithm by Carceroni et al. [1] in that the known epipolar geometry is used to compute frame correspondences based on where an object crosses an epipolar line. Whereas Carceroni et al. used frame-to-frame object motion and did not attempt to track objects throughout entire sequences, my algorithm takes the trajectory of an object as tracked through two video sequences and exploits frame-to-frame motion relative to an epipolar line to rapidly converge to the correct synchronization. Unlike the method presented in Chapter 3 which integrates tracking data from all frames to synchronize the sequences as a whole, this algorithm recovers the synchronization by locating pairs of temporally corresponding frames in each sequence.

In Chapter 5, a synchronization algorithm based on the matching of space-time interest points is detailed; unlike the algorithms in Chapters 3 and 4, no object tracking is required. I review space-time interest points [22] in Section 5.2; they are considered to be the equivalent in video sequences of spatial interest points in still images. My algorithm describes a nested RANSAC-based framework for using putatively matching space-time interest points to recover the synchronization parameters and a spatial model (either a homography or a fundamental matrix) relating the two cameras. I also present a special case of the algorithm that recovers the synchronization parameters when the spatial model is known. My approach demonstrates that the synchronization problem can be considered as an extension to the common stereo matching problem. Yan and Pollefeys' previous work on synchronization using space-time interest points involved correlating the temporal distribution of space-time interest points from each sequence [53]. Whilst their algorithm is fast and simple, it does not attempt to find corresponding space-time interest points from each sequence.

In Table 1, the algorithms reviewed in this chapter and my algorithms outlined in this section are categorized based on their requirement for knowledge of a spatial relationship between the two sequences and whether the sequences are capable of recovering the frame rate ratio of two sequences.

2.4 Summary

In this chapter, a review of the relevant literature for synchronization has been presented. It has been shown that there is a wide variety of algorithms in their approaches to the synchronization problem and in the assumptions that are made. The material to be presented in the following chapters has been put into context with the relevant literature and each algorithm is shown to be unique in various aspects.

Cameras	Spatial r'ship	Recovers α?	Algorithms
2	Known	No	Kuthirummal et al. [19] Reid and Zisserman [32] Zhou and Tao [54]
		Yes	Pooley et al. [30] Rao et al. [31] Tuytelaars and Van Gool [43] Velipasalar and Wolf [46] **Chapter 4**
	Unknown	No	Caspi and Irani [3, 4] Caspi and Irani [2] Caspi and Irani [3] (direct method) Dai et al. [7, 8] Lee et al. [24] Noguchi and Kato [28] Nunziati et al. [29] Sinha and Pollefeys [35, 36] Stein [39] Ushizaki et al. [45] Wolf and Zomet [51, 52] Yan and Pollefeys [53]
		Yes	Tresadern and Reid [41, 42] Ukrainitz and Irani [44] Wolf and Zomet [52] (direct method) **Chapter 3** **Chapter 5**
3	Unknown	No	Lei and Yang [25]
N	Known	Yes	Whitehead et al. [50] Carceroni et al. [1]
	Unknown	No	Korah and Rasmussen [17] Lei and Yang [25]

Table 1: The reviewed algorithms are classified based on the number of video sequences that they can synchronize, whether the algorithms require the cameras' spatial relationship to be known prior to commencing the synchronization step (e.g., a homography or a fundamental matrix), and whether the algorithms can recover the frame rate ratio. Some papers describe both feature-based and direct alignment methods; the direct alignment method is listed separately in the table and identified. Algorithms that are presented in this thesis are denoted in bold.

Chapter 3

Synchronization from a single trajectory

3.1 Introduction

This chapter introduces an algorithm that synchronizes two video sequences from the imaged trajectories of a single object tracked throughout each sequence. In many video sequences, a dominant moving object is visible and can be tracked in most frames, e.g., a person walking or dancing (in which case, the person's head, hands, or feet may be tracked), or a ball being thrown or kicked in a sporting event. Sometimes, particularly in the latter example, the cameras capturing this motion may be widely separated, in which case there may be few or no common stationary background points available for camera calibration. Existing trajectory based synchronization algorithms require either stationary background points [31] or multiple trajectories [42] to achieve synchronization, though it is not always necessary to determine corresponding trajectories from each sequence [51].

The algorithm presented in this chapter recovers the frame rate ratio, α, and the frame offset, Δ, of two sequences where the trajectory of a single object moving with a significant vertical motion component, e.g., a series of ballistic trajectories, is known. The trajectory need not be continuous throughout the sequences, and no stationary background points are required for synchronization. My algorithm is shown to synchronize sequences where tracking data may be missing in a significant

number of frames. It is assumed that the two cameras recording the sequences are stationary and have fixed intrinsic parameters; it is also assumed that both cameras have similar vertical orientations.

The algorithm uses a coarse-to-fine approach. At the coarse level, each video sequence is divided into a number of sub-sequences, where a sub-sequence can be considered to be a trajectory segment consisting of the object moving upwards and then downwards, for example, a ballistic trajectory. A vertical motion constraint is used to propose pairs of matching sub-sequences, one from each video sequence. In this process, an initial estimate of the frame rate ratio is recovered. This estimate is then used to determine a range of frame offsets in which the actual frame offset is assumed to lie. The fine synchronization step involves processing the sequences at the (sub-)frame level, using a measure of synchronization derived from a measurement matrix constructed from image observations. Estimates of both α and Δ are refined using a simplex search method, the cost function of which is based on this measure of synchronization. The algorithm is coarse-to-fine in the sense that data are processed firstly at a coarse scale using one method, and a second method is used for processing at a finer scale, rather than repeating the same operation over a number of scales.

This algorithm is similar to the approaches of Rao et al. [31] and Tresadern and Reid [41, 42] in that it uses the smallest singular value of a measurement matrix as a measure of synchronization; however, there are some significant differences. Rao et al. solve for a nonlinear temporal relationship between sequences, whereas this algorithm solves for the linear relationship described in Equation (1). However, unlike Rao et al., my algorithm requires neither stationary background points nor the initial frame offset to be known for synchronization. Tresadern and Reid's algorithm does not require stationary background points; however, it requires multiple objects to be tracked throughout both sequences and corresponding trajectories from each sequence to be determined in order to recover both α and Δ. In comparison, my algorithm requires only one object to be tracked.

My algorithm also bears similarities to an algorithm by Caspi and Irani [3,4]. Although their algorithm is designed to recover the frame offset and either a homography or fundamental matrix from multiple hypothesised trajectory correspondences, in the simplest case, only one object must be tracked throughout each sequence, which is the case in this algorithm. The algorithms differ in that Caspi and Irani's algorithm explicitly estimates a fundamental matrix from a measurement matrix, from which they compute a measure of synchronization; my algorithm computes only the singular values of the same measurement matrix which is computationally cheaper; this is discussed in detail in Section 3.5.2. Further, Caspi and Irani use an iterative approach to recover the synchronization for which the results may be dependent on the initial state of the algorithm, whereas my algorithm uses a coarse-to-fine approach which is less likely to get stuck in a local minimum.

In this chapter, the proposed algorithm is introduced and results presented for synchronizing synthetic data sets and real video sequences. It is shown that the algorithm is successful in synchronizing uncalibrated video sequences which capture the trajectory of a single object with significant vertical motion.

3.2 Coarse synchronization

At the coarse level, each sequence is divided into a number of sub-sequences. An assumption is made such that upper and lower bounds of α are proposed from the lengths of the sub-sequences. Then, the direction of the object's vertical motion in each frame is used to establish tentative matching sub-sequences from each sequence, yielding an initial estimate of α. Finally, for this initial estimate of α, a range of Δ values are proposed that are used in the following fine synchronization step.

3.2.1 Sub-sequence labelling

A single moving object is tracked throughout two video sequences denoted by $\mathcal{S}$ and $\mathcal{S}'$. Sequence $\mathcal{S}$ is divided into N sub-sequences, denoted $s_1, \ldots, s_N$, and $\mathcal{S}'$ into N' sub-sequences $s'_1, \ldots, s'_{N'}$, where successive sub-sequences are separated by missing

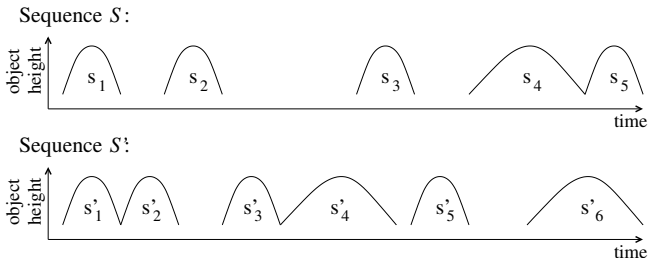


Figure 3: An example showing the labelled sub-sequences within two unsynchronized video sequences, where $\alpha = 1$ and each sub-sequence consists of the imaged trajectory of an object undergoing successive ballistic motions. It can be seen that if sub-sequence s_1 is aligned with sub-sequence s'_2, synchronization is achieved as demonstrated in Figure 4. Note that sub-sequence s'_4 has no corresponding sub-sequence in $\mathcal{S}$; this may be due to occlusion or a tracking failure.

trajectory data (due to occlusion or the tracked object leaving the field of view), or where the object bounces, i.e., where the direction of vertical motion changes from downwards to upwards. A bounce is a useful event to separate sub-sequences because its temporal location in each sequence is independent of perspective. In sports videos, a sub-sequence can be considered to contain a ball undergoing successive ballistic motions. More generally, a sub-sequence is a contiguous set of frames where the object firstly moves upwards and then moves downwards, and is visible in all such frames. Figure 3 shows an example of labelling sub-sequences within a pair of video sequences.

3.2.2 Temporal overlap assumption

It is expected that a pair of sub-sequences, s_n from $\mathcal{S}$ and s'_m from $\mathcal{S}'$, containing images of the same 3D motion, will be recorded over a similar length of time; note that this does not mean that they will have a similar number of frames since each sequence may be recorded at a different frame rate.

Sometimes, the time taken by one camera to record one sub-sequence may be significantly different to the time taken to record the sub-sequence containing the same 3D motion in the other video sequence, as, due to occlusion, one sub-sequence may be much shorter than the other sequence. It is assumed that the period of time taken to record s_n is no more than double the period of time taken to record s'_m, and vice versa; I refer to this as the *temporal overlap assumption*. For example, if s_n was

recorded in 4 seconds, then it is assumed that the corresponding sub-sequence s'_m was recorded in the range of 2 to 8 seconds. From this constraint and the number of frames in each sub-sequence, a range of frame rate ratios can be proposed in which the actual value of α is assumed to lie. It should be emphasized that this assumption does not mean that α is assumed to lie in the interval $[1/2, 2]$.

3.2.3 Determining the upper and lower bounds of α

The upper and lower bounds of α for each pair of sub-sequences are computed from the number of frames in the sub-sequences. Consider two sub-sequences s_n and s'_m having l_n and l'_m frames respectively. Then via the temporal overlap assumption, it is assumed that:

$$\alpha \in \left[\frac{l_n}{2l'_m}, \frac{2l_n}{l'_m}\right]. \tag{2}$$

Any frame rate ratio in this interval will satisfy the temporal overlap assumption for the pair of sub-sequences s_n and s'_m.

As an example, if $l_n = 30$ and $l'_m = 50$, then by Equation (2), $\alpha \in [3/10, 6/5]$. Consider the lower bound where $\alpha = 3/10$; in the time taken for 30 frames in s_n to be recorded, 100 frames will be recorded in $\mathcal{S}'$. Now due to the temporal overlap assumption, s'_m must be recorded in one half to twice the time period that was taken to record s_n, i.e., if s_n was recorded in 1 second, the time taken to record s'_m must be between 0.5 second to 2 seconds. From this constraint, s'_m must contain between 50 and 200 frames. It can be seen that the constraint holds, as s'_m contains 50 frames. At the upper bound, where $\alpha = 6/5$, there will be 25 frames recorded in $\mathcal{S}'$ in the time that the 30 frames of s_n are recorded. According to the temporal overlap assumption, the length of s'_m must be 12.5 to 50 frames, which again is true. For $\alpha \in (3/10, 6/5)$, the lower bound of the length of s'_m will increase from 12.5 to 50 frames, and the upper bound will increase from 50 to 200 frames. As l'_m will always lie in this interval, the bounds on α have been correctly calculated to satisfy the temporal overlap assumption.

For the sequences $\mathcal{S}$ and $\mathcal{S}'$, the upper and lower bounds of α are computed for all sub-sequence pairs. Then, from the set of the lower bounds of α, the median

lower bound value is determined and denoted by α^{lb}. Similarly, the median of the set of upper bound values is calculated and denoted by α^{ub}. The actual frame rate ratio is proposed to lie within the interval $[\alpha^{lb}, \alpha^{ub}]$.

3.2.4 A vertical velocity-based measure of matching sub-sequences

Vertical velocity is used to calculate a basic measure of how well two video sequences match at a given frame rate ratio. For a proposed pair of sub-sequences s_n and s'_m recorded at a frame rate ratio $\tilde{\alpha} \in [\alpha^{lb}, \alpha^{ub}]$, this measure describes how well the two sequences match if s_n and s'_m are images of the same 3D trajectory. The score incorporates the direction of the vertical velocity of the tracked ball, and also the number of other sub-sequences that are aligned if s_n and s'_m are aligned.

Vertical motion assumption

Clearly, the imaged trajectory of the ball in each view depends on the orientation of the cameras. As vertical velocity is used as the basis of the coarse measure of synchronization that is to be introduced, it is therefore important that both cameras' vertical orientations are similar. In this algorithm, it is not assumed that the cameras' vertical orientations are exactly the same; however, it is reasonable to assume that if both cameras are mounted, that the vertical axis of each image plane would be similar to the vertical direction in the real world.

Later, the validity of this assumption is tested in Section 3.4 where the imaged trajectories of an object are rotated to simulate rotating the vertical axis of the cameras.

Determining supporting sub-sequences

Consider a pair of sub-sequences, s_n and s'_m, that contain images of the same 3D trajectory. A temporal offset is applied to one sequence such that the first frames of s_n and s'_m are assumed to be recorded at the same time; these sub-sequences are now considered to be *aligned*. Aligning s_n and s'_m causes other sub-sequences in

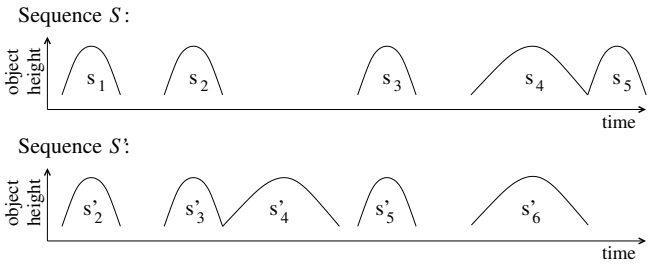


Figure 4: A temporal offset has been applied to the sequences shown in Figure 3 such that s_1 and s'_2 are now aligned. In this example, it is known that $\alpha = 1$.

$\mathcal{S}$ and $\mathcal{S}'$ to be aligned. Let s_q and s'_p be a pair of sub-sequences from $\mathcal{S}$ and $\mathcal{S}'$ respectively that are aligned as a result of aligning s_n and s'_m. If the directions of the vertical velocity components in s_q and s'_p match in any frame then (s_q, s'_p) is a *supporting sub-sequence pair* for s_n and s'_m.

Let the number of supporting sub-sequence pairs for s_n and s'_m be denoted by $P_{n,m}$. Now, consider Figure 4, which is modified from Figure 3 by deleting sub-sequence s'_1 and shifting $\mathcal{S}'$ to the left. Here, it can be seen that when s_1 and s'_2 are aligned, the directions of the vertical motions are aligned in the pair of sub-sequences s_2 and s'_3, denoted (s_2, s'_3), and also in the sub-sequence pairs (s_3, s'_5), (s_4, s'_6), and so on. Thus, there are 3 supporting sub-sequence pairs for s_1 and s'_2 and so in this example, $P_{1,2} = 3$.

Computing an alignment matrix

An alignment matrix $\mathtt{A}^{\tilde{\alpha}}$ encapsulates the relationship between all pairs of sub-sequences at a given frame rate ratio $\tilde{\alpha} \in [\alpha^{lb}, \alpha^{ub}]$, based on vertical velocity components. For a pair of sub-sequences, s_n and s'_m, an *alignment score* is computed that measures the matching vertical motions in the proposed corresponding frames of the two sequences. This score is stored in element (n, m) of $\mathtt{A}^{\tilde{\alpha}}$, denoted by $\mathtt{A}^{\tilde{\alpha}}_{n,m}$. If s_n and s'_m are images of the same 3D trajectory then the vertical motions will correspond and the alignment score is expected to be high; otherwise, it is likely that the alignment score will be low. The alignment score is computed as follows:

1. If $\tilde{\alpha} \neq 1$, it is necessary to interpolate trajectory data in one sequence such that the time period between consecutive frames in each sequence is the same,

therefore corresponding observations are considered to have been made at the same instant in time. It is assumed that the frame rate is sufficiently high that it is reasonable to use linear interpolation.

2. The sub-sequences s_n and s'_m are aligned by proposing that the first frames of s_n and s'_m were recorded at the same time. Let $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$ denote the sequences with the temporal offset applied as a result of aligning s_n and s'_m.

3. Corresponding frames from $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$ are deleted where trajectory information is not available in both sequences due to tracking failures or the sequences starting and finishing at different times. These frames are not useful because no data comparison can be made between sequences. Sequences $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$ now contain the same number of frames, denoted by $\tilde{f}_{n,m}$.

4. The alignment score is computed from the object's direction of vertical motion in corresponding frames of $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$. The vertical velocity is determined from consecutive frames in each sequence. Let v_k and v'_k be, respectively, the object's vertical velocity components in frame k of both $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$. Due to the interpolation process given in Step 1, both frames are proposed to have been recorded at the same instant in time. The alignment score for sub-sequences s_n from $\mathcal{S}$ and s'_m from $\mathcal{S}'$ at the frame rate ratio $\tilde{\alpha}$ is denoted by $\mathtt{A}^{\tilde{\alpha}}_{n,m}$ and computed via:

$$\mathtt{A}^{\tilde{\alpha}}_{n,m} = \frac{P_{n,m}}{\tilde{f}_{n,m}} \sum_{k=1}^{\tilde{f}_{n,m}} \delta(\mathrm{sgn}(v_k) - \mathrm{sgn}(v'_k)), \tag{3}$$

where $P_{n,m}$ is the number of supporting sub-sequence pairs for s_n and s'_m, $\mathrm{sgn}(v_k)$ is the sign of v_k, and

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Informally, the quantity expressed in Equation (3) is the percentage of frames in $\tilde{\mathcal{S}}_{n,m}$ and $\tilde{\mathcal{S}}'_{n,m}$ where the object's imaged vertical motions in the proposed corresponding frames are in the same direction, multiplied by the number of

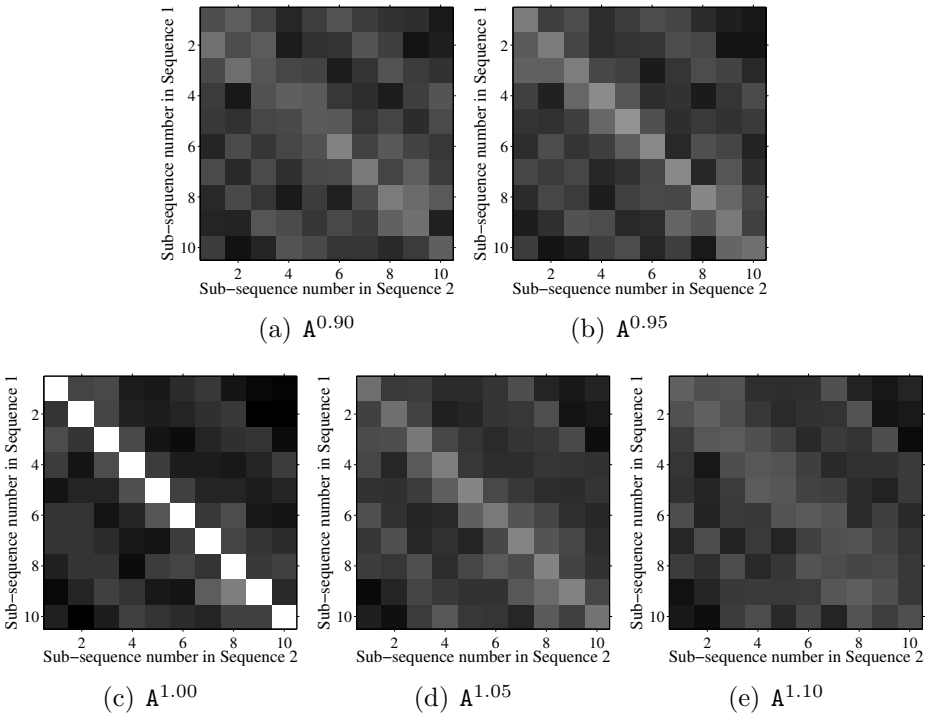
supporting sub-sequences, $P_{n,m}$. Although it may appear to be sufficient to use the percentage of frames with corresponding vertical motion and hence unnecessary to multiply the percentage by $P_{n,m}$, high alignment scores are sometimes found for pairs of sub-sequences only a few frames in length located near the ends of sequences (e.g., by aligning s_5 and s'_2 in Figure 4); such sub-sequence pairs often have few supporting sub-sequence pairs. Multiplying by $P_{n,m}$ ensures that a higher alignment score is assigned to correctly aligned sub-sequence pairs that have a high percentage of frames with matching directions of vertical motion.

3.2.5 Finding an initial estimate of α

The alignment matrix $\mathtt{A}^{\tilde{\alpha}}$ contains alignment scores for only one frame rate ratio $\tilde{\alpha}$. To determine an initial estimate of α, a number of alignment matrices must be computed for a range of frame rate ratios. Previously, the lower and upper bounds, α^{lb} and α^{ub} respectively, were proposed via the temporal overlap assumption. For each frame rate ratio $\tilde{\alpha} \in [\alpha^{lb}, \alpha^{lb} + 0.05, \ldots, \alpha^{ub}]$, an alignment matrix $\mathtt{A}^{\tilde{\alpha}}$ is computed. Separating successive values of $\tilde{\alpha}$ by 0.05 has shown empirically to provide a good compromise between computational efficiency and finding a good initial estimate of α for use in a later step.

For each alignment matrix, the *proposed corresponding sub-sequence pairs* are computed. A proposed corresponding sub-sequence pair is an element of the alignment matrix that contains the maximum value of both the row and the column that it is in; intuitively, this means that it is highly likely that this pair of sub-sequences correspond to the same trajectory segment in 3D space. The sub-sequence numbers of successive proposed corresponding sub-sequence pairs should increase monotonically, though not necessarily at the same rate since some sub-sequences may be missing in either one or both sequences. However, the previously described process of determining proposed corresponding sub-sequence pairs does not take this into account. The following method is used to remove sub-sequence pairs that do not satisfy this monotonicity constraint.

A linear relationship exists between the frame indices of frames recorded at the



(a) $\mathtt{A}^{0.90}$ (b) $\mathtt{A}^{0.95}$

(c) $\mathtt{A}^{1.00}$ (d) $\mathtt{A}^{1.05}$ (e) $\mathtt{A}^{1.10}$

Figure 5: An example of alignment matrices, $\mathtt{A}^{\tilde{\alpha}}$, shown graphically for a number of frame rate ratios $\tilde{\alpha}$. The pixel intensity is proportional to the value of the matrix element and all of the images use the same intensity scale.

same instant in time, as described mathematically in Equation (1). If, for every proposed corresponding sub-sequence pair, a 2D point is constructed from the frame numbers of the first frames of each of the proposed corresponding sub-sequences, then it is expected that a straight line can be fitted to these points. The 2D points representing incorrectly proposed corresponding sub-sequence pairs will not lie on this straight line; such incorrectly proposed sub-sequence pairs are determined via RANSAC [9] and deleted.

Figure 5 shows alignment matrices computed for synthetic sequences for a number of frame rate ratios. The synthetic sequences contained ten sub-sequences, and the actual frame rate ratio was 1.00. A dominant diagonal line of proposed corresponding sub-sequence pairs is visible in Figures 5(b), (c), and (d); it is seen that the brightness of this line in Figure 5(c), corresponding to the correct frame rate ratio, is much greater than in either of Figures 5(b) or (d). It is noted that the gradient of this line is not related to the sequences' frame rate ratio since it arises from sub-sequence correspondences rather than frame correspondences.

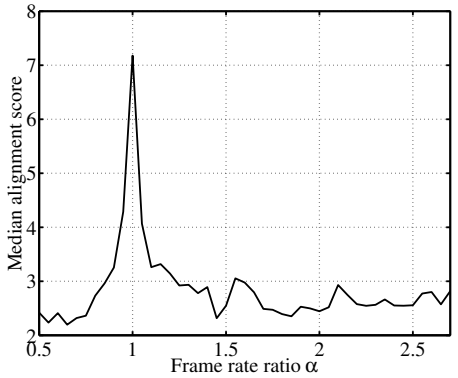


Figure 6: The median alignment score of the proposed corresponding sub-sequence pairs is shown over a range of frame rate ratios. It can be seen that the median alignment score reaches a maximum at the actual frame rate ratio, $\alpha = 1$.

Each proposed corresponding sub-sequence pair has an alignment score. At each value of $\tilde{\alpha}$, the median alignment score of the proposed corresponding sub-sequence pairs from the alignment matrix $\mathtt{A}^{\tilde{\alpha}}$ is computed. Let this median score be denoted by $a^{\tilde{\alpha}}$. Then the initial estimate of α, denoted by α^0, is determined via:

$$\alpha^0 = \underset{\tilde{\alpha}}{\operatorname{argmax}}\, a^{\tilde{\alpha}}.$$

Figure 6 shows the median alignment score of proposed corresponding sub-sequence pairs over a number of frame rate ratios for a synthetic data set containing ten sub-sequences. At the actual frame rate ratio, a maximum of the median score is clearly visible.

3.2.6 Finding an initial range of Δ values

Given α^0, the next step is to propose an estimate of the frame offset for that frame rate ratio. A voting scheme is used to determine a range of frame offsets that bracket the best frame offsets for α^0, based on the assumption that corresponding sub-sequences will overlap temporally.

Firstly, consider a pair of proposed corresponding sub-sequences s_n and s'_m. Since an estimate of α^0 is known, it is assumed that linear interpolation has been performed on the trajectory data of one of the sequences such that s_n and s'_m are considered to be recorded at the same frame rate and consist of l_n and l'_m frames respectively. Then, there is a range of frame offsets for which this pair of

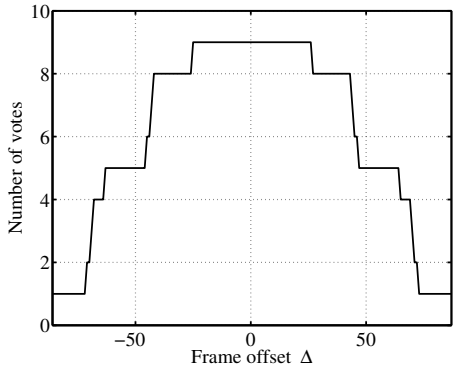


Figure 7: The distribution of votes as determined by the voting scheme. In this example, the each sequence contained ten sub-sequences, and $\alpha^0 = 1$. The actual frame offset is at $\Delta = 0$, with $\Delta^{lb} = -63$ and $\Delta^{ub} = 64$.

sub-sequences will overlap temporally. The upper and lower bounds of this range, denoted by $\Delta^{ub}_{n,m}$ and $\Delta^{lb}_{n,m}$ respectively, are given via:

$$\begin{aligned} \Delta^{ub}_{n,m} &= s^1_n - s'^1_m - l'_m \\ \Delta^{lb}_{n,m} &= s^1_n - s'^1_m + l'_n, \end{aligned}$$

where s^1_n and s'^1_m are the frame numbers of the first frames of s_n and s'_m in $\mathcal{S}$ and $\mathcal{S}'$ respectively. Votes are cast for each of the integer temporal offsets in the interval $[\Delta^{lb}_{n,m}, \Delta^{ub}_{n,m}]$. This process is repeated for each proposed corresponding sub-sequence pair. Finally, for each frame offset, the votes contributed by each pair of corresponding sub-sequences are summed. An example of the resulting distribution of votes is given in Figure 7.

Each temporal offset within the dominant peak of the vote distribution that has at least half of the maximum number of votes becomes a candidate temporal offset to be processed in the fine synchronization step. Let the upper bound of the set of eligible votes be denoted by Δ^{ub}, and the lower bound by Δ^{lb}; it is proposed that the actual offset Δ lies in the interval $[\Delta^{lb}, \Delta^{ub}]$. The size of this interval is similar to the lengths of the sub-sequences, hence it is expected to be small relative to the length of the sequences. This process significantly reduces the computation required in the subsequent search for the temporal offset Δ.

3.3 Fine synchronization

The fine synchronization step recovers the synchronization parameters, α and Δ, from the initial estimate of the frame rate ratio, α^0, and the interval $[\Delta^{lb}, \Delta^{ub}]$. This is a two-step process: first, an estimate of Δ is calculated for the frame rate ratio α^0; second, these values are used to initialize a search algorithm to accurately refine the values of α and Δ.

3.3.1 Finding the initial frame offset

To propose an initial estimate of the frame offset, firstly consider the votes corresponding to the initial estimate of the frame rate ratio, α^0. For each candidate integer frame offset $\tilde{\Delta} \in \{\Delta^{lb}, \Delta^{lb}+1, \ldots, \Delta^{ub}\}$, a measure of synchronization is computed. This measure is derived from two-view epipolar geometry.

Typically, the set of corresponding points used to estimate the fundamental matrix are stationary points visible in each of a pair of still images. In this synchronization algorithm, a different approach is taken by using the observed spatial locations of a *single* moving object that provides one point correspondence per pair of temporally corresponding video frames. The linear algorithm used to estimate the fundamental matrix involves constructing a measurement matrix $\mathtt{M}$ from 8 or more pairs of corresponding points [13], or in this case, trajectory observations from pairs of proposed temporally corresponding frames.

A measurement matrix is computed for a range of temporal offsets for the initial estimate of the frame rate ratio, α^0. If $\alpha^0 \neq 1$, trajectory information may be required at non-integer frame numbers, hence the tracking data are linearly interpolated to provide observations at these non-integer time intervals. The frame offset $\tilde{\Delta}$ is then applied to $\mathcal{S}$ and $\mathcal{S}'$, and frames where trajectory data are not available in both sequences are deleted, as in the process described in Step 3 of Section 3.2.4. Let $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{S}}'$ denote these modified sequences, and $\tilde{f}$ the number of frames in each of these sequences. For each of the proposed integer temporal offsets $\tilde{\Delta} \in \{\Delta^{lb}, \Delta^{lb}+1, \ldots, \Delta^{ub}\}$, the measurement matrix $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ is constructed from pairs of proposed temporally corresponding points $\mathbf{x}_t = (x_t, y_t)$ and

$\mathbf{x}'_{t\alpha^0+\tilde{\Delta}} = (x'_{t\alpha^0+\tilde{\Delta}}, y'_{t\alpha^0+\tilde{\Delta}})$, representing the object's location in frame t of $\tilde{\mathcal{S}}$ and frame $(t\alpha^0 + \tilde{\Delta})$ of $\tilde{\mathcal{S}}'$ respectively, where $t \in \{\tilde{t} \mid \tilde{t}, (\tilde{t}\alpha^0 + \tilde{\Delta}) \in \{1, 2, \ldots, \tilde{f}\}\}$. Note that $\tilde{\Delta}$ may be positive or negative. Let $\mathbf{f} \in \mathbb{R}^9$ be the vector formed by the elements of the fundamental matrix. Then, $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ and $\mathbf{f}$ satisfy:

$$\mathtt{M}_{\alpha^0,\tilde{\Delta}}\mathbf{f} = \mathbf{0}, \tag{4}$$

where

$$\mathtt{M}_{\alpha^0,\tilde{\Delta}} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_{t\alpha^0+\tilde{\Delta}}x_t & x'_{t\alpha^0+\tilde{\Delta}}y_t & x'_{t\alpha^0+\tilde{\Delta}} & y'_{t\alpha^0+\tilde{\Delta}}x_t & y'_{t\alpha^0+\tilde{\Delta}}y_t & y'_{t\alpha^0+\tilde{\Delta}} & x_t & y_t & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{5}$$

Each row of the matrix $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ takes on a different value of t. In order for $\mathbf{f}$ to be recoverable from Equation (4), it is necessary that $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ has at least 8 independent rows.

As suggested by Hartley [13], the trajectory coordinates are normalized to avoid numerical instability before constructing the measurement matrix. Ideally, the matrix $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ is of rank 8. However, when $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ contains more than 8 rows, it is usually of rank 9 due to the image coordinates being perturbed by noise, or incorrect point correspondences induced by unsynchronized sequences. Although vector $\mathbf{f}$ can be estimated via the singular value decomposition of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$, this is not necessary. Instead, the smallest, i.e., the 9th, singular value of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ is used as a measure of synchronization; it is calculated for each temporal offset in the interval determined by the voting scheme. The temporal offset with the minimum of the 9th singular value is deemed to be the frame offset recovered to integer accuracy. The 9th singular value of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ was previously used as a measure of synchronization by Rao et al. [31]. In Section 3.5.2, I discuss the advantages of using the 9th singular value for the measure of synchronization instead of the reprojection error.

Once the value of $\tilde{\Delta}$ (corresponding to the frame rate ratio α^0) with the minimum of the 9th singular value has been located to integer accuracy, it can be refined to sub-frame accuracy via the golden-section search (implemented using the MATLAB function `fminbnd`). Again, the 9th singular value of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ given in

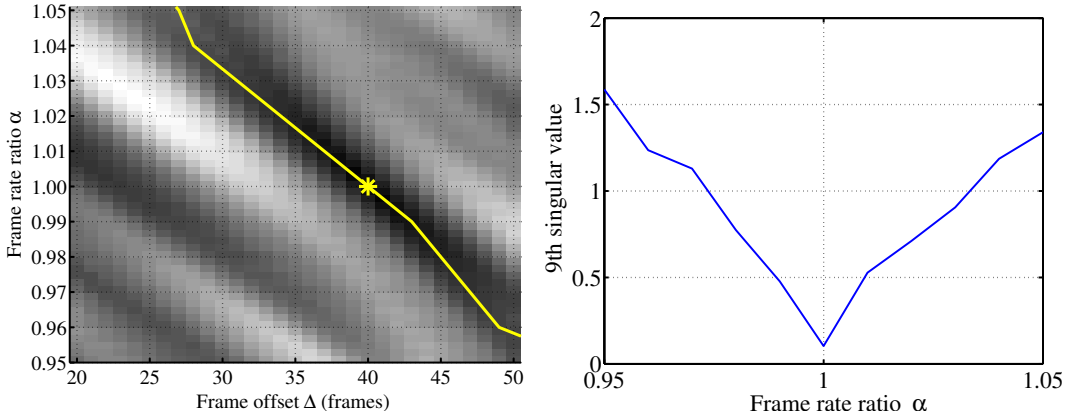
Equation (5) is used as a measure of synchronization and the object's trajectory is linearly interpolated where necessary. The search interval is two frames wide, with one frame on each side of the frame offset corresponding to the local minimum previously found. Now, let Δ^0 denote the sub-frame offset with the minimum of the 9th singular value. As this sub-frame temporal offset lies in a narrow interval and the curve's profile is convex, convergence is guaranteed. In practice, to reduce the width of the search window to within 0.05 frame, approximately seven iterations of the golden section search are required.

3.3.2 Searching for the actual synchronization

To further refine the synchronization parameters, Nelder-Mead simplex search is used (implemented via the MATLAB function `fminsearch`). The search is initialized using the values α^0 and Δ^0 computed above, and as before, the measure of synchronization used is the 9th singular value of the measurement matrix `M`.

In Figure 8(a), the variation of the 9th singular value over a region of Δ-α space is displayed. If the data were plotted in 3D, a short valley running along a straight line through the location of the actual synchronization would be visible, which in this example is at $\alpha = 1$ and $\Delta = 40$; the line plotted in Figure 8(a) indicates the bottom of this valley, and the asterisk indicates the global minimum within the valley. The 9th singular values at points along this line are displayed in Figure 8(b). It can be seen that as in Figure 15, a sharp minimum of the 9th singular value is observed at the actual synchronization. An explanation for the existence of this valley is presented in Section 3.5.1.

In the previous step, the bottom of the valley was located for the initial estimate of α^0. The aim now is to search for the minimum of the 9th singular value within the valley. Since the previous search step using a bounded one dimensional search located a point within the valley, the search region of this secondary search will remain inside the valley, as otherwise the value of the cost function will increase.



(a) (b)

Figure 8: (a) The 9th singular value of the measurement matrix calculated at various values of Δ and α. The pixel intensity is proportional to the magnitude of the 9th singular value. For each discretized α value, the Δ value with the smallest 9th singular value is located; a line is overlaid that passes through these (Δ, α) points. The actual 9th singular values along this line are plotted in (b).

3.4 Results

In this section, results are presented for synchronizing a number of synthetic and real video sequences. The synthetic sequences consisted of a set of parabolic trajectories generated in 3D space and projected onto two image planes representing virtual cameras; each sub-sequence in the virtual video sequences corresponded to one parabolic trajectory. Each parabolic trajectory was of a randomly generated length up to 100 frames, and the number of frames between successive trajectories was also randomly generated, again up to 100 frames. For the real video sequences, two different camera setups were used. In the real video sequences denoted `indoor` and `outdoor` in Table 2, both cameras recorded at 25fps with each frame containing two independently recorded half-height interlaced fields; the size of each half-height image was 720×288 pixels. For the other real video sequences, the resolution of each camera was 640×480 pixels, with one camera always recording at 15fps, and the other camera recording at either 15fps or 30fps. The ratio of frame rates for each pair of sequences is shown in Table 2.

In the real video sequences, an object was moving with significant vertical motion. In the sequences with names beginning with `dark`, a point light source was

tracked. The light source was held in one hand as the subject holding the light walked with normal arm motions around an irregular path in a dark room (in the `darkwalk` sequence pair), or manually moved the light along a path similar to a ballistic trajectory. In these sequences, the light source was tracked automatically by locating the centroid of the light in each frame. The other real video sequences contained a ball being thrown and bounced, or the ball being held in the hand of a person walking normally. The ball was manually tracked throughout the sequence by estimating its centroid in each frame. Trajectory data was unavailable in a significant number of frames due to occlusion or the ball or light source moving outside the camera's field of view.

Table 2 summarizes the results of synchronizing a number of real and synthetic video sequences. For synthetic data sets, the number of sub-sequences present in each sequence is given in the data set's name, e.g., `fifteen-a`. The actual frame rate ratio and frame offset for each pair of sequences are given by $\overline{\alpha}$ and $\overline{\Delta}$ respectively; the estimated frame rate ratio and frame offset are denoted by $\widehat{\alpha}$ and $\widehat{\Delta}$ respectively. Two values of $\widehat{\Delta}$ are given: one is for the case when both Δ and α are unknown, and the other is for when α is known. In the latter case, α^0 was set to the known value of α, and the recovered frame offset $\widehat{\Delta}$ is the value of Δ^0 as recovered in Section 3.3.1 (i.e., Δ^0 is estimated via the 1D golden section search, but not refined using the 2D Nelder-Mead method). In the case where $\overline{\alpha} \neq 1$, the trajectory data from one sequence was sub-sampled.

Figures 9, 10 and 11 show synchronized frames from the `backyardb`, `indoor` and `walk6` sequences respectively. In each of the frames shown in the figures, the tracked object's trajectory throughout the time interval of the displayed frames is overlaid. This is only a small segment of the trajectory used for synchronization. In the four data sets `darkrooma`, `darkroomb`, `darklaba`, and `darklabb`, a light was tracked throughout each sequence. Frames where the light turned on and off were used as cues for recovering the ground truth synchronization parameters, hence $\overline{\Delta}$ is only available to integer accuracy. The light was not occluded and rarely moved outside of either camera's field of view. In Figure 12, the entire trajectories of the light in each view of the `darkwalk` sequences are shown.

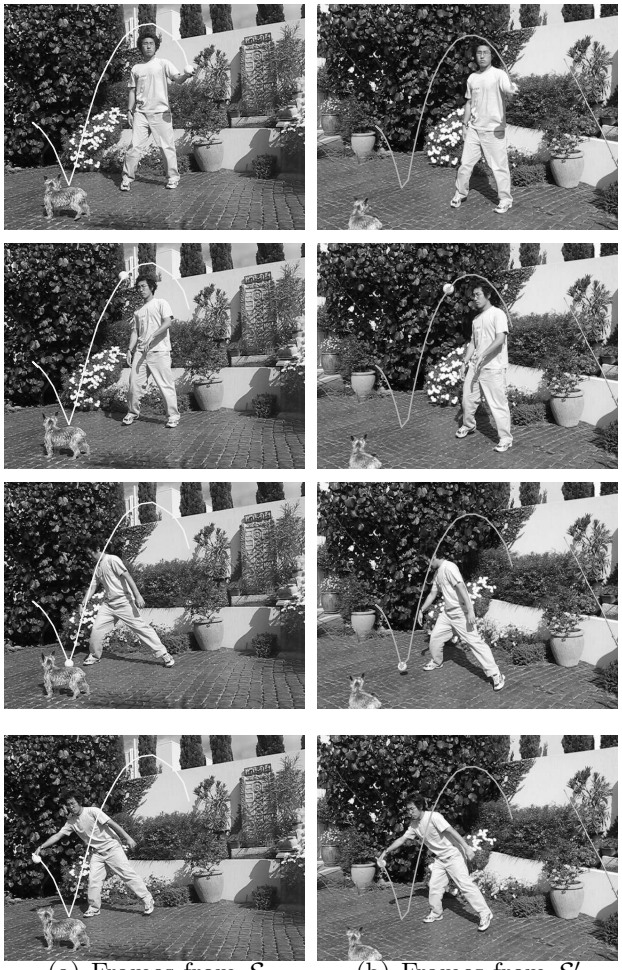
Data set name	$\mathcal{S}$ length	$\mathcal{S}'$ length	$\overline{\alpha}$	$\widehat{\alpha}$	$\overline{\Delta}$	$\widehat{\Delta}$ α unknown	$\widehat{\Delta}$ α known
ten-a	958	958	1	1.0000	0.0	0.00	0.00
ten-b	958	506	4/3	1.3334	200.0	199.99	200.01
fifteen-a	1715	1158	2	2.0000	−599.5	−599.51	−599.51
fifteen-b	1715	1298	6/5	1.2000	500.5	500.51	500.50
fifteen-c	1715	616	5/6	0.8334	1200.5	1200.47	1200.50
darklaba	433	808	2	2.0002	−59.0	−58.88	−58.82
darklabb	408	747	2	2.0001	−70.0	−69.36	−69.42
darkrooma	308	882	2	2.0000	258.0	258.24	258.20
darkroomb	449	615	1	1.0001	93.0	92.85	92.87
backyarda	449	698	2	2.0000	−172.5	−172.25	−172.24
backyardb	449	932	1	1.0000	39.5	39.19	39.21
indoor	794	838	1	1.0000	8.0	8.50	8.48
outdoor	1536	888	1	1.0000	738.0	737.73	737.74
darkwalk	400	900	2	2.0004	−130.5	−130.57	−130.46
walk6	434	405	1	1.0001	−30.5	−30.46	−30.44
walk7	449	787	2	2.0000	−50.5	−50.44	−50.44

Table 2: Results of synchronizing a number of synthetic and real video sequences; results for synchronizing synthetic data sets are presented in the top section of the table, whilst results for real sequences are given in the bottom section. To compute $\widehat{\Delta}$ where $\overline{\alpha}$ is known, the value of $\widehat{\Delta}$ is the value of Δ^0 computed in Section 3.3.1.

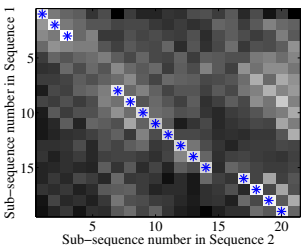
The results show that $\widehat{\alpha}$ is recovered very accurately, and $\widehat{\Delta}$ is recovered to acceptable accuracy, remembering that $\overline{\Delta}$ is at best known to the nearest half-frame for real video sequences. It can also be seen that if the algorithm recovers both $\widehat{\Delta}$ and $\widehat{\alpha}$, the parameter $\widehat{\Delta}$ is computed to a similar accuracy to the case where $\widehat{\alpha}$ was known, demonstrating that the synchronization accuracy is not significantly affected if the frame rate ratio is unknown. The results for synchronizing the synthetically generated data sets where the frame offset is known exactly shows that this algorithm can accurately recover the frame offset to sub-frame accuracy.

It was stated in Section 3.2.5 that potential values of α^0 were separated by 0.05. In Table 2, it can be seen that some synthetic data sets (`ten-b` and `fifteen-c`) had frame rate ratios that were not a multiple of 0.05. This shows that even though α^0 is only an estimate, accurate recovery of $\widehat{\alpha}$ can still be achieved.

In the `indoor` data set, there were 17 and 19 sub-sequences in $\mathcal{S}$ and $\mathcal{S}'$ respectively; the alignment matrix is shown in Figure 10(c). In this experiment, the



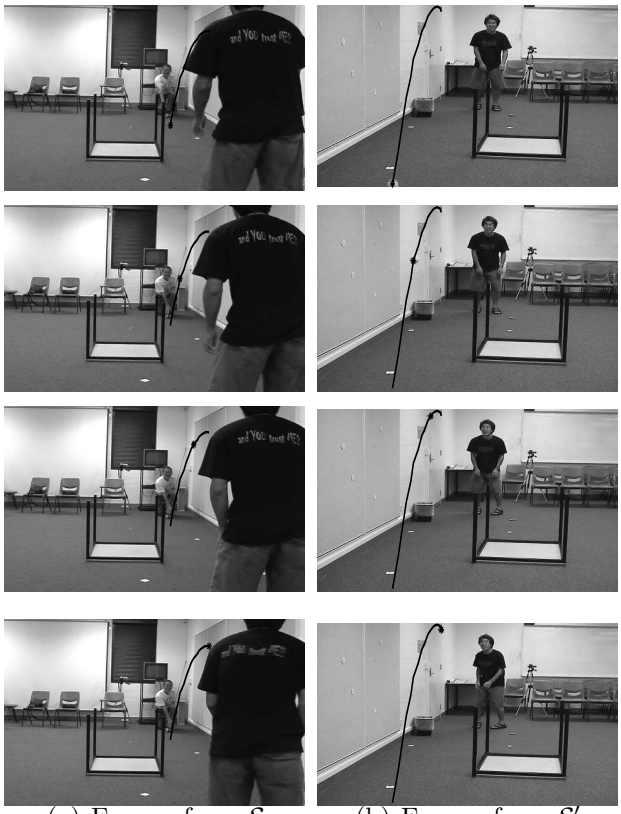
(a) Frames from $\mathcal{S}$ (b) Frames from $\mathcal{S}'$



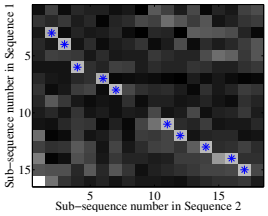
(c) Alignment matrix

Figure 9: (a) & (b) Synchronized pairs of frames from sequence `backyardb`, with a small segment of the ball's trajectory overlaid. In this example, $\overline{\alpha} = 1$ and $\overline{\Delta} = -39.5$ frames. The half-frame misalignment can be seen in these sequences by examining the trajectory where the ball bounces. The alignment matrix for this pair of sequences is shown in (c), where it can be seen that some sub-sequences visible in Sequence 1 are not visible in Sequence 2, and vice versa. The proposed corresponding sub-sequence pairs are indicated by asterisks in (c).

ball was frequently occluded by the people in the scene and the ball often moved outside the cameras' fields of view. Because of this, some sub-sequences in $\mathcal{S}$ did not have a corresponding sub-sequence in $\mathcal{S}'$ and vice versa, yet synchronization was still achieved. Also, the sequences had 847 pairs of corresponding frames after synchronization, but of these frames, there were only 130 frames where the ball was visible in both sequences. In the `fifteen-b` synthetic data set, even numbered sub-sequences were deleted from one sequence. Even with this significant loss of data, accurate synchronization was achieved as shown in Table 2.



(a) Frames from $\mathcal{S}$ (b) Frames from $\mathcal{S}'$



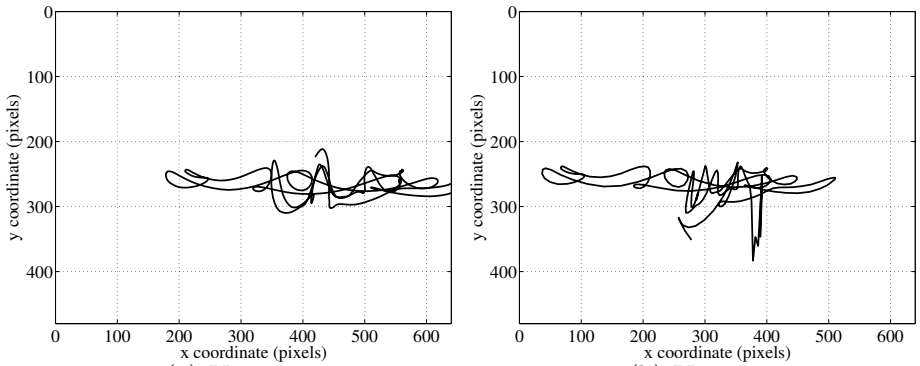
(c) Alignment matrix

Figure 10: Frames taken from the `indoor` sequences, where $\overline{\alpha} = 1$ and $\overline{\Delta} = 8.5$ frames. There are many occasions where the ball is occluded by the people in the scene, and when the ball is outside the field of view of at least one camera.

It has been stated previously in Section 3.2.4 that the cameras must have a similar vertical orientation. Experiments have shown that rotating the imaged trajectories in the sequences by a small amount does not significantly affect the accuracy of the synchronization; rotating the trajectories is equivalent to tilting the camera. The experiments included cases where the trajectories from corresponding sequences were rotated in the same direction, and in opposite directions. Rotating the trajectories from the `darkwalk` sequences, displayed in Figure 12, by up to 15 degrees did not affect the recovered synchronization. This example is significant because the motion in the sequences is predominantly horizontal, not vertical. As this algorithm initially determines sub-sequences based on the direction of vertical motion, it was possible that any slight changes in the camera orientation could significantly affect the determination of sub-sequences. Other sequences featuring more significant vertical motion, e.g., the `darklab` and `darkroom` sets of sequences,

(a) Frames from $\mathcal{S}$ (b) Frames from $\mathcal{S}'$ (c) Alignment matrix

Figure 11: Frames taken from the `walk6` sequences. Although it may seem that there is not significant vertical motion, the trajectory segment shown in each frame shows that there is sufficient motion for synchronization. For these sequences, $\overline{\alpha} = 1$ and $\overline{\Delta} = -30.5$.



(a) View 1 (b) View 2

Figure 12: The trajectories of a light held in the hand of a person walking around a room in the `darkwalk` sequences. It is noted that simple motion as displayed here contains sufficient vertical motion for sub-sequences to be proposed and correctly matched, and synchronization achieved.

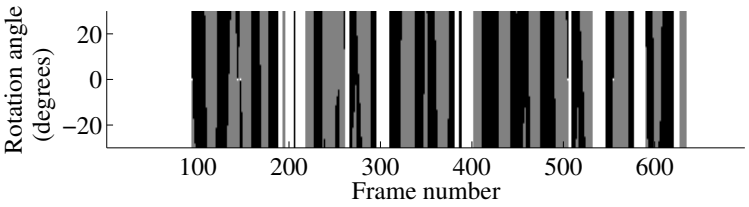


Figure 13: The effect of rotating trajectories on vertical velocity. To interpret this chart, consider a horizontal line corresponding to a particular rotation angle. Then, for each frame, grey indicates that the object is moving upwards, black indicates downward motion, and white denotes a tracking failure.

and the `indoor` and `outdoor` sequences, allowed a rotation of up to 30 degrees.

Figure 13 illustrates the effect of rotating the trajectories on the determination of sub-sequences. In this example, one of the `backyarda` sequences was rotated at angles varying from -30 degrees through to 30 degrees. It can be seen that at some locations, rotation has a significant effect on the direction of vertical velocity, however overall, the effect of rotation on the direction of vertical motion in the trajectories, and hence the determination of sub-sequences, is not significant.

Previously, it was stated in Section 3.2.6 that for each proposed corresponding sub-sequence pair, a vote is cast for the range of Δ values such that the sub-sequences overlap, hence it is expected that the maximum range of votes will be the sum of the number of frames in the two sub-sequences. The 9th singular value of `M` is computed for every integer $\tilde{\Delta}$ value as determined from the set of votes. If the lengths of the sub-sequences are small relative to the lengths of the sequences, the range of $\tilde{\Delta}$ values will also be small. Even though a brute-force search over this range is described in Section 3.3.1, in practice, this range is small, as demonstrated in Table 3.

The algorithm failed to synchronize sequences containing periodic motion or videos consisting of very short sub-sequences; these results are not listed in Table 2. In the case of non-periodic motion, each sub-sequence should only have one corresponding sub-sequence from the other sequence with a high alignment score. However, when periodic motion exists, a sub-sequence may have high alignment scores when paired with many different sub-sequences. Also, for non-periodic motion, each proposed corresponding sub-sequence pair will ideally have the same set

Data set	$\overline{l_n}$	$\overline{l'_m}$	Δ range	Frames
indoor	14.1	10.5	[-3,19]	23
outdoor	15.6	14.4	[724,747]	24
backyarda	23.3	22.0	[-194,-171]	24
backyardb	13.4	12.0	[-51,-28]	24
darkrooma	27.1	29.4	[244,274]	31
darkroomb	18.9	19.0	[71, 115]	45

Table 3: The mean number of frames, denoted by $\overline{l_n}$ and $\overline{l'_m}$ respectively, of sub-sequences in various experiments. The range of Δ values determined by the voting scheme and the number of frames in this range are shown in the last two columns.

of supporting sub-sequence pairs. In the case of periodic motion, though, there may appear to be many different valid sets of supporting sub-sequences. When this is combined with the numerous high alignment scores mentioned above, each row and column of the alignment matrix will contain many large spurious values, rather than a unique large value that corresponds to the correct pair of sub-sequences. This will cause the determination of proposed corresponding sub-sequence pairs to be incorrect, and the voting scheme will likely fail to set the correct range of Δ values.

3.5 Discussion

3.5.1 Reliance on the initial estimate of α

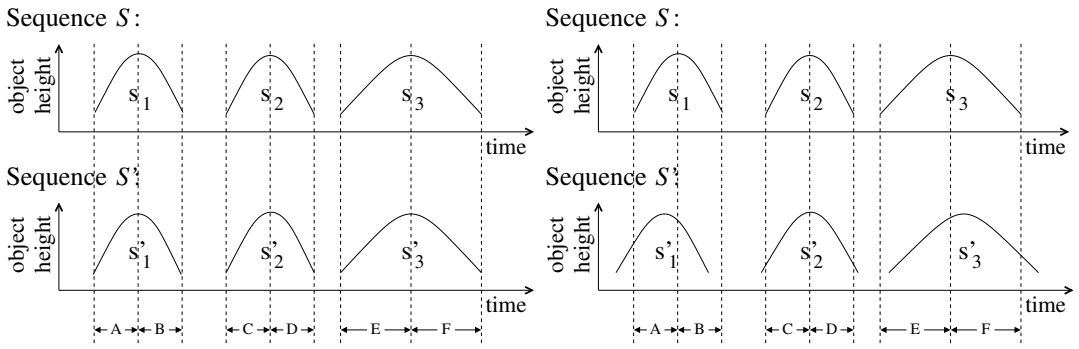
The accuracy of this algorithm relies heavily on α^0, the initial estimate of α. Since the initial estimate of the frame offset is dependent on α^0, and the following 2D search in Δ-α space is initialized from these values, accurate estimation of α^0 is crucial.

In Figure 8(a), it was shown that if the 9th singular value of `M` is plotted over a range of Δ-α space, a narrow valley exists around the location of the actual synchronization parameters. This occurs because a small change in the frame rate ratio causes one of the sequences to become stretched temporally relative to the other, causing the sequences to be unsynchronized; however, applying a small temporal shift to one sequence can partially counter the mis-synchronization. As the error

in the frame rate ratio increases, the temporal offset required for correcting the error induced by the frame rate ratio must also increase proportionally, resulting in the straight line segment shown in Figure 8(a). A simple diagram aiding in the explanation of this problem is shown in Figure 14.

In Figures 14(a) and (b), the region marked A denotes the time period in which the vertical motion is upwards in s_1 and region B denotes the period of downwards motion in s_1. Regions C and D are similarly defined for s_2, and E and F for s_3. Figure 14(a) shows an example where the proposed estimate of the frame rate ratio is $\tilde{\alpha} = 1$ and the frame offset has been recovered correctly; ignoring perspective effects, the tracked object's vertical motion will be in the same direction in corresponding frames of each sequence. Hence, in region A in $\mathcal{S}$, the vertical velocity in each frame will match the vertical velocity in region A in $\mathcal{S}'$; this will hold true for each region. In Figure 14(b), $\tilde{\alpha}$ is incorrect; however, in each of regions A to F, the vertical velocity components in each sequence correspond over a large percentage of each interval. It is noted that in this example, $\tilde{\alpha} > \overline{\alpha}$ and the corresponding compensatory frame offset $\tilde{\Delta}$ is less than the actual frame offset $\overline{\Delta}$. The values of $\tilde{\alpha}$ and $\tilde{\Delta}$ relative to $\overline{\alpha}$ and $\overline{\Delta}$, respectively, correspond qualitatively to (Δ, α) locations on the line displayed in Δ-α space in Figure 8(a). Although from this example, it is not possible to show directly how this causes the 9th singular value to remain small if a small temporal offset is applied to partially counteract the effect of a small change in α, in practice this hypothesis holds.

As the lengths of the sequences increase, the effect of a small change in the frame rate ratio is magnified, and sub-sequences that are temporally distant from the point where the sequences are stretched become more mis-aligned than those close to that point. This is demonstrated in Figure 14(b), where the midpoints of s_2 and s_2' are aligned. It is seen that the boundaries of s_2 and s_2' are closely aligned, however the boundaries of other sub-sequences located further from the alignment point are mis-aligned. Because of this effect, more of the observations used to construct `M` are inconsistent, which in turn causes the 9th singular value to increase. This means that for longer sequences, the width of the valley as described in Section 3.3.2 is reduced; for very long sequences, the search for the minimum of the 9th singular



(a) $\tilde{\alpha} = 1$ (b) $\tilde{\alpha} = 1.1$

Figure 14: An example of how applying a temporal offset can minimize the effect of mis-synchronization caused by an incorrect frame rate ratio; here, $\overline{\alpha} = 1$. In (a), the sequences are perfectly synchronized, i.e. $\tilde{\alpha} = \overline{\alpha}$. In (b), the initial estimate of $\tilde{\alpha}$ differs from $\overline{\alpha}$ by a small amount. The misalignment of sequences in (b) due to this error can be minimized by applying a small temporal offset; in this case, a negative offset is applied, resulting in $\mathcal{S}'$ shifting to the left. An explanation for the regions A through to F is given in the text of Section 3.5.1.

value may start outside of the valley and hence the algorithm will fail to synchronize the two sequences. Therefore, it is important that α^0 is estimated accurately. A possible solution for this problem is to divide longer sequences into a number of shorter sequences, and then attempt to synchronize the shorter sequences.

3.5.2 Why the 9th singular value is used instead of the reprojection error

As described in Section 3.3, the measure of synchronization used in the fine synchronization step of this algorithm is the 9th singular value of the measurement matrix given in Equation (5). Another potential measure of synchronization is the reprojection error of observed trajectory points from the corresponding epipolar lines. In this section, I outline some advantages of using the 9th singular value instead of the reprojection error.

In Figure 15, both the 9th singular value and the reprojection error are displayed over a range of frame offsets. It is noted that the global minimum of the 9th singular value coincides with that of the reprojection error, yet the profile of the curve of the 9th singular value is smoother. A possible explanation of why the

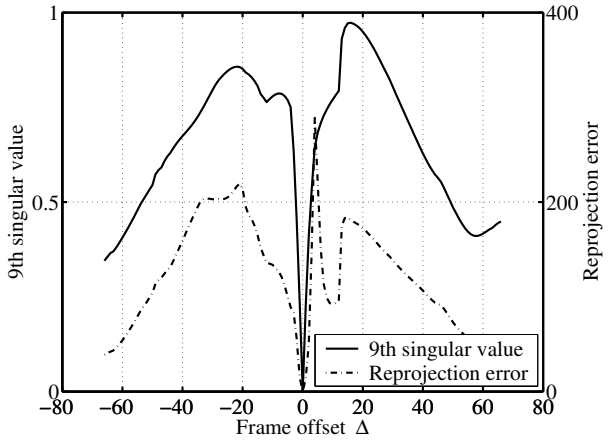


Figure 15: The 9th singular value and the reprojection error are compared over a range of frame offsets, with the frame rate ratio fixed at $\alpha = 1$. For both curves, a local minimum occurs at the actual temporal offset ($\Delta = 0$) within the interval determined by the voting scheme.

curve of the reprojection error is jagged is that when the epipolar geometry is estimated at consecutive integer frame offsets, the epipoles may be located in different positions in different frames because the measurements used to recover the epipolar geometry are inconsistent due to the mis-synchronization. As the reprojection error is dependent on the locations of the epipoles, the reprojection error computed in consecutive frames may be unstable which is reflected in the jaggedness of the curve.

The movement of the epipoles is illustrated in Figure 16. In each view, the other camera, mounted on a tripod, is visible, thus providing a simple way of confirming that the epipoles have been estimated correctly. For each frame offset, the fundamental matrix was estimated from the measurement matrix given in Equation (5), and the epipoles were computed from the fundamental matrix. It can be seen that the epipoles move significantly even over a small range of frame offsets.

A further advantage for using the 9th singular value as a measure of synchronization is that it is computationally cheaper to calculate compared to computing the vector $\mathbf{f}$ and the reprojection errors. The $m \times n$ measurement matrix $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ can be decomposed via the singular value decomposition (SVD) into a diagonal matrix $\mathtt{D}$ containing the singular values, and orthonormal matrices $\mathtt{U}$ and $\mathtt{V}$, such that:

$$\mathtt{M}_{\alpha^0,\tilde{\Delta}} = \mathtt{U}\mathtt{D}\mathtt{V}^{\mathsf{T}}.$$

(a) View 1 (b) View 2

Figure 16: The recovered epipoles are displayed as circles for integer frame offsets 5 frames on each side of the correct synchronization; epipoles for consecutive integer frame offsets are joined by a line, and the epipoles for the actual frame offset are denoted with an asterisk. The recovered epipoles coincide with the actual epipoles only at the correct frame offset.

However, it is not necessary to compute all of `U`, `D` and `V` in computing the SVD. To compute the 9th singular value of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$, only the `D` matrix must be computed, requiring approximately $2mn^2 + 2n^3$ floating point operations as stated by Golub and Van Loan [11]. Computing the reprojection error is a much more complicated process. Firstly, the `V` matrix must be computed, which when computed alongside the `D` matrix, requires approximately $2mn^2 + 11n^3$ flops. Then, the fundamental matrix is taken by forming a 3×3 matrix from the elements of the last column of `V`; a rank 2 constraint is enforced upon the resulting matrix via a further SVD operation. Finally, the reprojection error is computed via matrix multiplications for every feature. Clearly, it is far more computationally efficient to compute only the 9th singular value of $\mathtt{M}_{\alpha^0,\tilde{\Delta}}$ instead of the reprojection error.

3.5.3 Alternative methods for aligning sub-sequences

In Section 3.2.4, the process of aligning a pair of sub-sequences by aligning the first frames of the sub-sequences was described. This was one of three proposed methods of aligning two sub-sequences:

1. **Aligning the start of the sub-sequences.** The first frames of each of the sub-sequences to be aligned are considered to have been recorded at the same instant in time. This method assumes that the tracked object is not occluded or out of frame at the beginning of each sub-sequence.

2. **Aligning the midpoints of the sub-sequences.** The middle frames of each sub-sequence are considered to have been recorded at the same instant in time. Whereas the first method may be affected if the sub-sequences do not start at the same instant in time, this method may be more robust to such problems as it assumes that both the start and end of each sub-sequence are equally affected by missing trajectory data.

3. **Aligning the peak of the trajectory in each sub-sequence.** The frames from each sub-sequence with the maximum y component are considered to have been recorded at the same instant in time. This is different to Method 2 as the peak does not necessarily occur in the middle frame of the sequence. This method will work best in sequences where the trajectory has a sharp turning point.

To analyze the three alignment methods, experiments were conducted to examine the alignment matrices computed by each method for a number of video sequences where $\alpha = 1$. Five pairs of real video sequences were used; the sole synthetic data set used was named `fifty-t`. A short description of each data set is presented below:

- The `fifty-t` data set was created by projecting fifty parabolic trajectories in 3D space onto two virtual image planes; each trajectory represented by one sub-sequence in each sequence of up to 100 frames in length. Then, each sub-sequence had a random number of frames (up to 20 frames) deleted from each end of the sub-sequence, i.e., corresponding sub-sequences from each sequence had a different number of frames deleted. The trajectories had a sharp, well defined peak.

- In the `indoor` video sequences, two people are throwing a ball to each other. One camera is located behind each person, such that when the ball moves away from one camera, it is moving towards the other camera. Frames from these sequences are shown in Figure 16. In most sub-sequences, the peak of the trajectory was easily determined.

- Frames from the `outdoor` data set are illustrated in Figure 10. In these sequences, two people are bouncing a ball to each other. The cameras were placed to either side of one person, such that when the ball moves away from one camera, it is also moving away from the other camera. Hence, it is expected that the perspective effects are similar in each view. As the cameras are located high relative to the motion in the sequences, a sharp peak is not always visible for each ballistic trajectory; however, frequent bouncing events ensure that the first frames of many sub-sequences are correctly determined for both sequences.

- The `backyardb`, `walk6`, and `darkwalkb` sequences contain two adjacent cameras recording a person performing some action. Frames from the `backyardb` and `walk6` sequences are shown in Figures 9 and 11 respectively. As the cameras are adjacent, it expected that the perspective effects are similar in each view. Also, as the camera locations in these sequences are low relative to the motion in the sequences, the peaks of the trajectories are often easily identifiable.

Numerical results from the experiments are given in Tables 4 and 5. The ratio of alignment scores given in Table 4 is calculated from the median alignment score for the proposed corresponding sub-sequence pairs and the median alignment score for all other sub-sequence pairs. In Table 5, a *false positive* is defined as a sub-sequence pair that was incorrectly labelled as a proposed corresponding sub-sequence pair; a *false negative* is a sub-sequence pair that should have been labelled as a proposed corresponding sub-sequence pair, but was not.

Surprisingly, aligning the peaks of the trajectories in each sub-sequence gives the best results in some cases, typically in sequences where the cameras are level and "behind" the object motion, i.e., the tracked object moves away from both cameras in corresponding frames in each sequences, as occurs in the `outdoor`, `darkwalkb` and `walk6` sequences. In sequences with this camera setup, the perspective effects are similar in each view. However, it can be seen that the number of correctly proposed corresponding sub-sequences relative to the number of false positives produced by

Sequence	Alignment method	Ratio of alignment scores
fifty-t (synthetic data)	First frame	4.30
	Middle frame	4.42
	Trajectory peak	3.07
indoor	First frame	4.69
	Middle frame	4.36
	Trajectory peak	4.33
outdoor	First frame	3.15
	Middle frame	3.41
	Trajectory peak	4.19
backyardb	First frame	4.40
	Middle frame	4.72
	Trajectory peak	3.63
walk6	First frame	3.14
	Middle frame	2.69
	Trajectory peak	3.14
darkwalkb	First frame	5.92
	Middle frame	5.45
	Trajectory peak	5.92

Table 4: The ratio of the median alignment score for proposed corresponding sub-sequence pairs and the median alignment score of other sub-sequence pairs is shown for various video sequence pairs. For each pair of video sequences, three different methods of aligning sub-sequences were used.

this method is often larger than for other methods.

Comparing the method of aligning sub-sequences by the first frames of the sub-sequences with the method of aligning by the middle frames, it is shown that neither approach consistently produces a better ratio of the median alignment scores. It can also be seen in Table 5 that neither method consistently provides better results in correctly proposing corresponding sub-sequences.

It was expected that for the `fifty-t` data set, aligning the sub-sequences by their middle frames would provide the best results because deleting a random number of frames from the start of each sub-sequence and then aligning the sub-sequences by their first frames would cause mis-alignment. Aligning the sub-sequences by the middle frames was expected to minimize the mis-alignment, since a random number of frames would also be deleted from the end of each sub-sequence as well. However, the results show that the ratio of the median alignment scores

Sequence	Alignment method	Correct PCSPs	False positives	False negatives
fifty-t (synthetic data)	First frame	44	1	1
	Middle frame	42	2	3
	Trajectory peak	41	0	4
indoor	First frame	7	5	5
	Middle frame	9	3	3
	Trajectory peak	6	5	6
outdoor	First frame	5	2	6
	Middle frame	7	5	4
	Trajectory peak	8	3	3
backyardb	First frame	17	2	2
	Middle frame	16	1	3
	Trajectory peak	15	4	4
walk6	First frame	14	0	4
	Middle frame	8	4	10
	Trajectory peak	9	5	9
darkroomb	First frame	8	0	0
	Middle frame	8	0	0
	Trajectory peak	8	0	0

Table 5: The number of correctly proposed corresponding sub-sequence pairs (PCSPs), and the number of false negatives and false positives are given for the three alignment methods for various sequences.

was only marginally better than the ratio of scores produced when aligning the sub-sequences by their first frames; further, aligning the sub-sequences by their first frames resulted in fewer false negatives and fewer false positives.

The method of aligning sub-sequences by their first frames gives good results and outperforms the alternative method of aligning sub-sequences by their middle frames when proposing corresponding sub-sequence pairs for the `fifty-t` data set. Also, since sub-sequences may be separated by a bounce event, which is expected to occur in temporally corresponding frames from each sequence, it is sensible to use this event in the alignment process. Hence, as detailed in Section 3.2.4, it is preferred to align a sub-sequence pair by the first frames of each of the sub-sequences.

3.5.4 Alternative measures for computing alignment scores

It is noted that in Equation (3), only the direction of the object's vertical velocity component, and not its magnitude, is used in the computation of the alignment

score. An alternative score that incorporates the magnitude of the velocity may provide a more accurate measure of synchronization. I have experimented with using the correlation coefficient, computed from the vertical component of the object's motion in corresponding frames from each view, as an alignment score. Unfortunately, perspective effects cause this scheme to be less effective than expected.

Figure 17(a) shows an object moving in two parabolic segments. Two cameras view this motion, and Figures 17(b) and (c) display the scene as viewed by each camera. It can be seen that the object's motion in the parabolic segment closest to the camera appears larger than the motion in the distant segment. Figure 17(d) shows a scatterplot, where each point on the plot relates the vertical velocity component in frames from each view. Two distinct line segments can be seen on the plot, with each line corresponding to one parabolic segment in 3D space. The gradient of each line is different due to perspective effects; observed motion close to one camera appears larger than when viewed by a distant camera. As more parabolic segments are added, the number of line segments in Figure 17(d) increases, which causes the correlation coefficient to decrease. Hence, if this measure was to be used, it could be affected by the number of sub-sequences, which is undesirable. Whilst the simpler alignment score given in Equation (3) does not take into account the magnitude of vertical motion, it ignores perspective effects, hence it is a preferred measure of sub-sequence alignment.

3.5.5 Comparisons with other synchronization algorithms

As previously mentioned, this algorithm bears some similarities to other previously published algorithms by other authors. It is not trivial to compare these algorithms' performance in synchronizing the same video sequences as each algorithm makes different assumptions and has different input requirements as discussed in Section 3.1. In this section, the similarities and differences of these algorithms are highlighted, and their complexities are examined from a theoretical perspective.

Firstly, I present a basic complexity analysis of my algorithm. In the fine alignment step, an $m \times n$ measurement matrix `M` is constructed for each value of α and Δ, and a singular value decomposition (SVD) is then performed on each instance

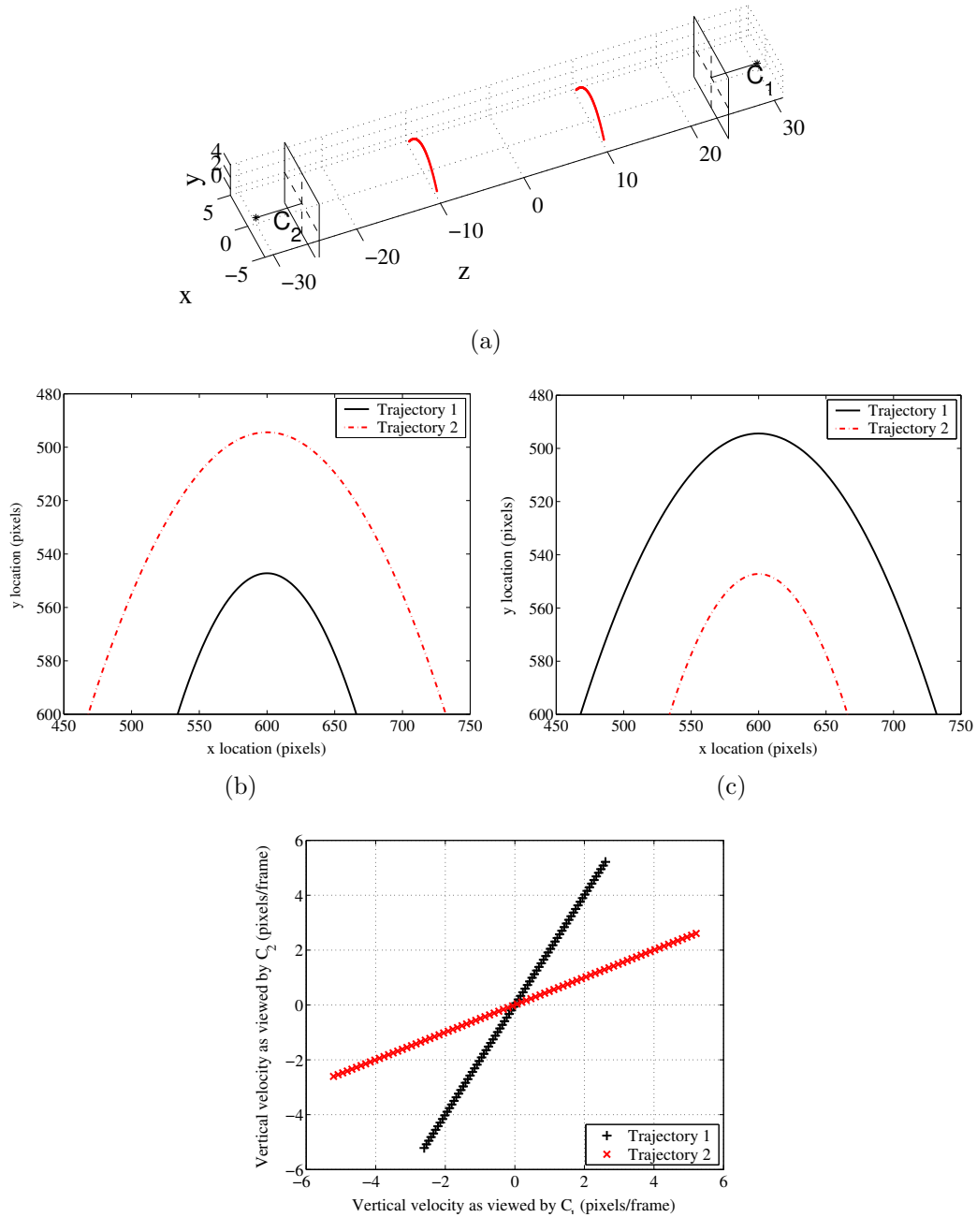


Figure 17: (a) Two virtual cameras, with optical centres C_1 and C_2, view two parabolic trajectory segments in 3D space. (b) The scene as viewed by C_1. (c) The scene as viewed by C_2. (d) A scatterplot of the vertical velocity v as viewed by C_1 vs the vertical velocity v' as viewed by C_2 in corresponding frames of each sequence.

of $\mathtt{M}$. It is noted that SVD is computationally expensive: computing only the diagonal matrix $\mathtt{D}$, where $\mathtt{M} = \mathtt{UDV}^\mathsf{T}$, requires $2m^2n^2 + 2mn^3$ floating point operations if $m \geq n$ [11]. Now, each of the m rows in $\mathtt{M}$ contains one observation, and as there is

one observation per frame, the number of frames is also m.

At the fine synchronization step in my algorithm, there will be at most m measurement matrices constructed, each with up to m observations. Hence, the complexity of the fine synchronization step is $O((m^2n^2 + mn^3)m) = O(m^3n^2 + m^2n^3)$. However, this does not include the computation of the alignment score expressed in Equation (3), which consists of computationally cheap integer additions and one integer division operation, and it is expected that the number of measurement matrices will be much less than m, as demonstrated in Table 3.

This algorithm is similar to a dynamic time warping algorithm by Rao et al. [31] in that the same measure of synchronization is used. However, the algorithms differ significantly in the restrictions made on the input trajectory: whereas my algorithm requires the tracked object to exhibit significant vertical motion, Rao et al. require the trajectory to be continuous (i.e., it is not clear how to handle missing trajectory data), and that the first observations in the trajectory data in each view were recorded at the same instant in time (i.e., the initial frame offset must be known).

Rao's algorithm implements a coarse-to-fine approach. At the finest level of detail, the dynamic time warping algorithm computes the SVD of a measurement matrix containing up to m observations for each of m^2 pairs of frames. Hence, the complexity of Rao's algorithm is $O((m^2n^2 + mn^3)m^2) = O(m^4n^2 + m^3n^3)$, which is m times greater than my algorithm's fine synchronization step.

Tresadern and Reid proposed two video synchronization algorithms, one for the affine camera model [42] and another for the perspective camera model [41]. The comparison discussed below is relevant to their algorithm for the perspective camera model only. My algorithm is similar to theirs in that a rank-based approach is used to synchronize the sequences from trajectory data alone. However, the methods of synchronization are different. My algorithm requires only a single trajectory to propose α^0 and searches throughout a small interval to estimate Δ^0 before iteratively refining both parameters. On the other hand, Tresadern and Reid's algorithm synchronizes sequences by determining pairs of temporally corresponding frames from at least 8 trajectory correspondences via an exhaustive search; α and Δ are then recovered from this set of frame correspondences.

Tresadern and Reid's algorithm will construct a $c \times 9$ measurement matrix for each pairing of frames (for c corresponding trajectories from each sequence), resulting in m^2 measurement matrices. As their algorithm only requires the diagonal matrix produced by SVD, the complexity of their algorithm is $O((c^2n^2 + cn^3)m^2) = O(c^2m^2n^2 + cm^2n^3)$. Since it is expected that $c \ll \sqrt{m}$, i.e., the number of correspondences is much less than the square root of the number of frames in each sequence, Tresadern and Reid's algorithm provides greater computational efficiency at the expense of requiring more pre-processing to track at least 8 objects and find trajectory correspondences between sequences.

A final comparison is provided between my algorithm and an algorithm by Caspi and Irani [3, 4] that recovers the frame offset of two sequences recorded by cameras with a known ratio of frame rates. Whereas the previous algorithms discussed in this section have used singular values as a measure of synchronization, Caspi and Irani's algorithm explicitly computes the fundamental matrix and uses reprojection error as a measure of synchronization. Recovering the fundamental matrix requires the SVD to compute the orthogonal `V` matrix which is computationally more expensive than calculating only the diagonal `D` matrix, requiring $2mn^2+11n^3$ flops [11]. Further, the fundamental matrix must be a rank 2 matrix which can be enforced via SVD. Then, the reprojection error is computed for every data point. These extra operations will be of order m: the step to enforce the rank 2 constraint will take constant time as the fundamental matrix is always a 3×3 matrix, and the number of operations required to compute the reprojection errors is linear in the number of points for which the reprojection error is computed, i.e., m. As the complexity of these steps is insignificant relative to computing the SVD of the measurement matrix, they are not considered further in this analysis.

Caspi and Irani's algorithm has a computationally expensive initialization step that estimates Δ via an exhaustive search. As it searches through m frames, the algorithm computes the SVD of an $m \times n$ measurement matrix. Hence, the complexity of the algorithm is $O((m^2n^2 + mn^3)m) = O(m^3n^2 + m^2n^3)$. This is the same as my algorithm that also recovers the frame rate ratio of the two sequences.

The complexities of the algorithms are summarized in Table 6.

Algorithms	Complexity
My algorithm	$O(m^3n^2 + m^2n^3)$
Rao et al. [31]	$O(m^4n^2 + m^3n^3)$
Tresadern and Reid [41]	$O(c^2m^2n^2 + cm^2n^3)$
Caspi and Irani [3, 4]	$O(m^3n^2 + m^2n^3)$

Table 6: A summary of the complexities of various synchronization algorithms.

Whilst all of these algorithms are intended for offline use, the complexity of the algorithm is still important, particularly since none of the algorithms operate in linear time. As the algorithm detailed in this chapter is more efficient than Rao's algorithm, it would be preferred for use where it is known that the ratio of frame rates of the two sequences is constant. For sequences where at least 8 corresponding trajectories are available in many frames of each sequence and where $c \ll \sqrt{m}$, Tresadern and Reid's algorithm gives a faster computation time. Finally, my algorithm would be expected to run in a similar time as Caspi and Irani's algorithm, yet it would also recover the frame rate ratio.

3.6 Conclusion

It has been shown that the frame offset and frame rate ratio of two unsynchronized videos can be recovered using the trajectory of only one object moving with significant vertical motion. Other similar algorithms require multiple object trajectories or stationary background points to be specified in order to recover both synchronization parameters.

A coarse-to-fine approach was presented that initially used the tracked object's vertical motion to determine matching sub-sequences, with a voting scheme used to provide an initial estimate of the synchronization parameters. A measure of synchronization derived from two-view epipolar geometry was calculated at each value of α and Δ as they were iteratively refined via a search in Δ-α space. It was demonstrated that this measure is preferred to the geometric reprojection error, and that a good approximation of the frame rate ratio can be computed from vertical motion data alone. The algorithm was shown to accurately recover the frame rate ratio and frame offset relating pairs of synthetic and real video sequences where

trajectory information was absent in a significant proportion of frames.

Chapter 4

Motion guided synchronization

4.1 Introduction

In this chapter, I present an algorithm that synchronizes two video sequences of an object undergoing ballistic motion. The epipolar geometry and the motion of the object are exploited such that the algorithm converges rapidly to find pairs of frames from each video sequence recorded at the same time. From the frame indices of multiple such pairs of frames, the values of α and Δ can be recovered. The algorithm requires the cameras to be weakly calibrated, and it is assumed that the cameras' intrinsic and extrinsic parameters remain fixed whilst the sequences are recorded. This algorithm is shown to accurately recover the ratio of frame rates and the temporal offset of the two sequences to sub-frame accuracy.

My algorithm is similar to that of Carceroni et al. [1] in that epipolar geometry is used to find temporally corresponding frames from each sequence. Their algorithm, proposed corresponding frames from each sequence by firstly taking a feature point detected in a frame of a reference sequence and using the known epipolar geometry to compute the corresponding epipolar line in the other video sequence. In this second sequence, feature points were detected and tracked between consecutive frames; features that crossed the computed epipolar line were deemed to have satisfied the epipolar constraint at some time instant between frames and thus became candidate matches for the point from which the epipolar line was computed. From these point matches, tentative frame correspondences from each sequence were proposed, from

which the synchronization parameters were recovered via RANSAC. Carceroni et al. demonstrated that their approach can be applied to $N \geq 2$ video sequences.

The algorithm presented here varies to that of Carceroni et al. in that an iterative approach is used to exploit object motion such that the algorithm rapidly converges to find the time instant at which the object crossed the epipolar line. Only a single object is required to be tracked, and the nature of the algorithm requires that the class of motion is restricted. In the basic algorithm presented in Section 4.2, it is assumed that the video sequences are short and contain the ball undergoing a single ballistic motion, i.e., the ball firstly moves upwards and then moves downwards. Later, in Section 4.3, the algorithm is extended to synchronize longer video sequences containing multiple such motions. My algorithm is designed to synchronize two video sequences, whereas Carceroni's algorithm can synchronize multiple video sequences recorded by weakly calibrated cameras.

Results are presented for applying the algorithm to synthetic and real data where the cameras remain stationary. It is shown that the algorithm can simultaneously recover the frame rate of the sequences to sub-frame accuracy and the ratio of frame rates of the two sequences. It is also demonstrated that the influence of noise on the coordinates of the trajectory used for synchronization is not significant.

In the following sections, frame i in video sequence 1 is denoted by $\mathcal{S}_i$ and similarly, frame j in sequence 2 is denoted by $\mathcal{S}'_j$. Throughout this chapter, the term *corresponding frames* is used to describe a pair of frames, one from each video sequence, that are recorded at the same instant in time. I use a ball as the moving object in this chapter, though it can be substituted for any object undergoing ballistic motion. To avoid confusion with other moving objects in the videos, the moving object tracked throughout the sequences is referred to as a ball throughout this chapter.

4.2 The basic algorithm

The basic algorithm consists of two steps: determining pairs of temporally corresponding frames from each sequence by exploiting object motion and epipolar

geometry, then estimating the ratio of frame rates and the frame offset from these pairs of frames. It is assumed that the fundamental matrix relating the two cameras has already been estimated from a number of corresponding stationary background points and that the ball's trajectory throughout each sequence has been recovered by an object tracking algorithm or manual tracking.

4.2.1 Finding a pair of corresponding frames

The first step is to establish a pair of corresponding frames from the two video sequences by exploiting epipolar geometry and the ball's motion. Let $\mathtt{F}$ denote the fundamental matrix relating the two cameras. For video sequences recorded by stationary cameras with fixed intrinsic parameters, $\mathtt{F}$ is invariant and the images of all corresponding stationary scene points satisfy the epipolar constraint in all frames of the video sequences. If the ball's location in frame $\mathcal{S}_i$ is known and denoted by $\mathbf{x}_i$, the corresponding epipolar line $\mathbf{l}'_i$ in $\mathcal{S}'_j$ can be computed via $\mathbf{l}'_i = \mathtt{F}\mathbf{x}_i$. If $\mathcal{S}_i$ and $\mathcal{S}'_j$ are corresponding frames, then the imaged locations of corresponding moving points in $\mathcal{S}_i$ and $\mathcal{S}'_j$ also satisfy the epipolar constraint. More specifically, the ball's imaged position $\mathbf{x}'_j$ in frame $\mathcal{S}'_j$ will lie on the epipolar line $\mathbf{l}'_i$. This fact is used to search for the correct alignment. It is noted that satisfying the epipolar constraint is a necessary, but not sufficient, requirement for synchronization. However, as will be seen, motion and trajectory constraints are applied to ensure that the epipolar constraint is satisfied only in temporally corresponding frames.

To find a pair of corresponding frames, first, a frame $\mathcal{S}_i$ is randomly chosen where the ball's vertical velocity is significant (explained later in Section 4.5). Rather than performing a brute force search through all frames $\mathcal{S}'_j$ to find the corresponding frame in $\mathcal{S}'$, the ball's vertical motion is exploited to reduce the search. A frame $\mathcal{S}'_j$ is selected such that the ball's direction of vertical motion is the same as in frame $\mathcal{S}_i$. It is assumed that the frame rate of the sequence is sufficiently high that the ball's inter-frame motion is approximately linear; thus, the ball's velocity in $\mathcal{S}'_j$ can be approximated from the ball locations in two consecutive frames. Then, from the ball's position in $\mathcal{S}'_j$ and its velocity, the number of frames until the ball crosses the epipolar line $\mathbf{l}'_i$ can be estimated. The following steps outline an iterative method

to estimate the value of j given an epipolar line $\mathbf{l}'_i$:

1. Calculate the ball's inter-frame velocity in frame $\mathcal{S}'_j$: $\mathbf{v}'_j = \mathbf{x}'_{j+1} - \mathbf{x}'_j$.

2. Let $\mathbf{t}'$ be the linear approximation of the ball's trajectory in $\mathcal{S}'_j$. Assuming a constant velocity model, $\mathbf{t}'$ is a straight line passing through $\mathbf{x}'_j$ and $\mathbf{x}'_{j+1}$, calculated via the cross product: $\mathbf{t}' = \mathbf{x}'_j \times \mathbf{x}'_{j+1}$.

3. Next, calculate the intersection point $\mathbf{p}'$ of the approximated trajectory $\mathbf{t}'$ and the epipolar line $\mathbf{l}'_i$, via: $\mathbf{p}' = \mathbf{t}' \times \mathbf{l}'_i$.

4. To estimate the number of frames until the ball crosses the epipolar line, firstly, let $\mathbf{d}'$ be the vector from $\mathbf{x}'_j$ to the intersection point $\mathbf{p}'$, i.e., $\mathbf{d}' = \mathbf{p}' - \mathbf{x}'_j$. Then, the ball is estimated to cross the epipolar line in $n = \|\mathbf{d}'\|/\|\mathbf{v}'_j\|$ frames.

5. Using the norm of vectors $\mathbf{d}'$ and $\mathbf{v}'_j$ and ignoring their orientations in the previous step introduces an ambiguity in the direction of the search. Figure 18(a) illustrates the case where the corresponding frame occurs in the future; the algorithm should look forwards in time to locate the frame when the ball crosses the epipolar line. Conversely, in Figure 18(b), the epipolar line crossing occurred in the past, hence, the algorithm should search backwards for the corresponding frame. The direction ambiguity can be resolved by examining the vectors $\mathbf{v}'_j$ and $\mathbf{d}'$. The search should be directed forwards if both $\mathbf{v}'_j$ and $\mathbf{d}'$ have the same orientation; otherwise, the search should move backwards. Thus, n is then modified via: $n \leftarrow n\ \mathrm{sgn}(\mathbf{v}'_\mathrm{j} \cdot \mathbf{d}')$.

6. The value of n is used to update j and to determine whether or not the process terminates at this iteration:

   - If $n \in [0, 1)$, the ball must have crossed the epipolar line in the time interval $[j, j+1)$. The synchronized frame can then be estimated to be frame $j \leftarrow j + n$. At this stage, the iteration can terminate and the temporal correspondence $\mathcal{S}_i \leftrightarrow \mathcal{S}'_j$ is established to sub-frame accuracy.
   - If $n \notin [0, 1)$, the frame in $\mathcal{S}'$ that is closest in time to frame $\mathcal{S}_i$ and not recorded after frame $\mathcal{S}_i$ must be $j + \lfloor n \rfloor$, where $\lfloor n \rfloor$ is the largest integer

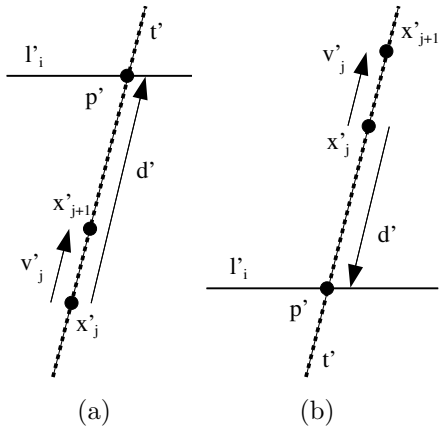


Figure 18: The ball's velocity is used to estimate where to search for the corresponding frame, assuming constant velocity. In (a), the corresponding frame is in the future, whereas in (b), it is in the past.

less than or equal to n. The strategy is to update j via $j \leftarrow j + \lfloor n \rfloor$ and repeat the process from Step 1. Although j is updated here by an integer, synchronization to sub-frame accuracy is achieved when the iteration terminates as described in the previous bullet point. Note that the integer update $\lfloor n \rfloor$ is enforced here because the ball's velocity is approximated using the forward, rather than the backward, difference.

4.2.2 The convergence of the algorithm

To demonstrate that the algorithm always converges, the four different states that can arise, as shown in Figure 19, are analyzed below:

1. State 1: The ball is below the epipolar line $\mathbf{l}'_i$, and moving upwards towards the epipolar line, as shown in Figure 19(a). In this state, the ball is decelerating due to gravity. Hence, the velocity will decrease as the ball moves closer to $\mathbf{l}'_i$. Thus, n, the number of frames to look forward, is underestimated as $\mathbf{v}'_j$, the magnitude of the velocity in frame j, is larger than at any frame between frame j and the ball crossing the epipolar line. Hence, at the next iteration, the algorithm will remain in State 1 until $0 \leq n < 1$, in which case convergence has been achieved.

2. State 2: The ball is above the epipolar line and moving upwards away from the

epipolar line, as shown in Figure 19(b). As in State 1, the ball is decelerating, but in this case, as the ball is moving away from the epipolar line, the number of frames to look backward for the point where the trajectory crosses the epipolar line will be overestimated. It is noted that in Step 6 in Section 4.2.1, if $n \notin [0, 1)$, j is updated via $j \leftarrow j + \lfloor n \rfloor$; however because $n < 0$, this will increase the overestimation of the number of frames to search backwards. Thus, at the next iteration, the algorithm will switch to State 1.

3. State 3: The ball is below the epipolar line and accelerating downwards away from the epipolar line, as shown in Figure 19(c). For the same reason given for State 1, $|n|$ will be underestimated. However, as $n < 0$ and $\lfloor n \rfloor$ is used to update j, an overestimate may occur. If $\lfloor n \rfloor$ is an underestimate, the algorithm will remain in State 3, and at the next iteration, the ball's location in frame $j + \lfloor n \rfloor$ will be closer to the epipolar line than in frame j, as is the case in State 1. A series of underestimations will eventually result in frame j being the first frame after the ball has moved below the epipolar line. Then, in this frame, $-1 \leq n < 0$. Thus, an overestimate will occur as $\lfloor n \rfloor = -1$, the ball in frame $j + \lfloor n \rfloor$ will be located above the epipolar line and the algorithm will move to State 4. In the following iteration in State 4, convergence will occur since $0 \leq n < 1$.

4. State 4: The ball is above the epipolar line and is accelerating downwards towards the epipolar line, as shown in Figure 19(d). As in State 2, n will be overestimated if $n > 1$ and the algorithm will move continue in State 3 at the next iteration. Otherwise, if $0 \leq n < 1$, the iteration will terminate due to convergence.

The analysis of the four cases above has not taken into account perspective foreshortening. When the ball is located close to a camera, its motion will appear to be exaggerated compared to when it is far from a camera. This may cause the expected underestimates and overestimates as described in the analysis to be magnified or reversed (i.e., what was expected to be an underestimate may become an overestimate, and vice versa). In practice, in a given iteration, if the ball is

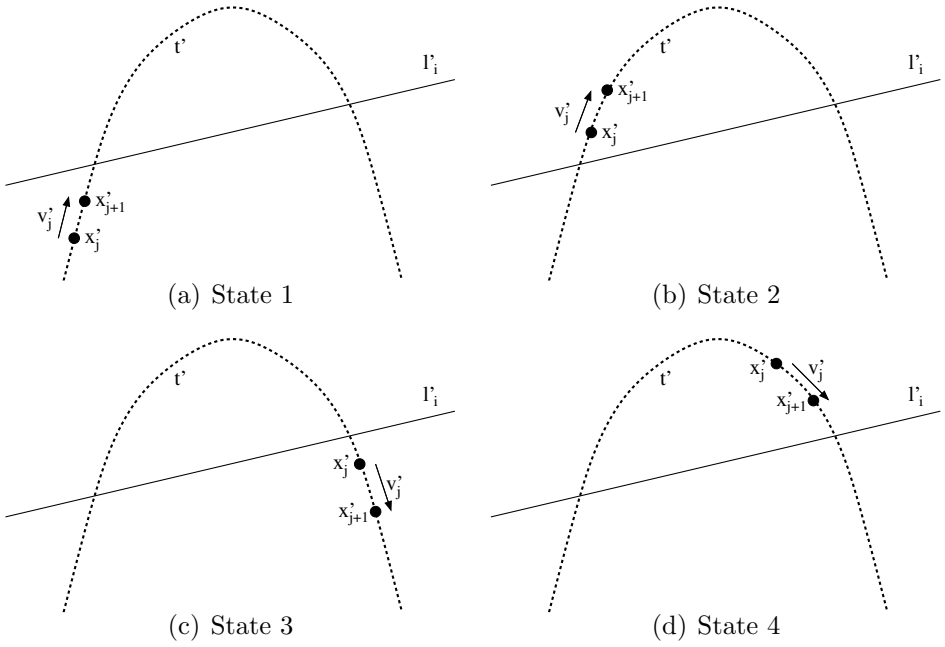


Figure 19: These four diagrams illustrate the four states that the algorithm may enter.

located within a few frames of the epipolar line, the perspective effects are not significant and convergence occurs as expected. As the distance between the ball and the epipolar line increases, the influence of perspective effects increases. The algorithm will still continue to search in the correct direction in time to locate the instant of the epipolar line crossing, but convergence may be achieved at a slower rate since the estimate of how many frames to look forwards or backwards will not be as accurate.

Figure 20 shows an example where the ball is moving upwards towards the epipolar line and decelerating due to gravity, with the iteration arbitrarily starting at $j = 2$, putting the algorithm in State 1. The velocity of the ball is shown by the line denoted by $\mathbf{v}_2$ and the ball is expected to cross the epipolar line in frame 11 based on the assumption of constant velocity. As expected from the above analysis, this is an underestimate of the number of frames to look forward. In the following iterations, this estimate of j is refined to frame 15 using the ball's velocity $\mathbf{v}_{11}$ at frame 11, then to frame 16, and finally, frame 16.20.

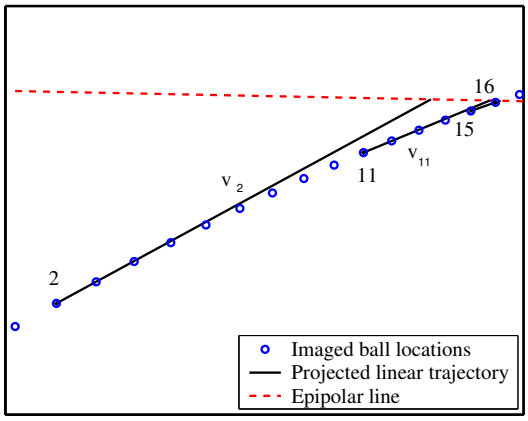


Figure 20: A demonstration of the convergence of the algorithm, as described in Section 4.2.2. The frame numbers of some trajectory coordinates used by the algorithm in the convergence process are displayed.

4.2.3 Combining observations to recover the synchronization

Given a frame $\mathcal{S}_i$, the iterative procedure in Section 4.2.1 estimates the corresponding frame j in $\mathcal{S}'$ to sub-frame accuracy. If this procedure is repeated for frames $\{i_1, \ldots, i_k\}$, where $k \geq 2$, the set of corresponding frames $\{j_1, \ldots, j_k\}$ from the second video sequence can be established. Since the values $\{i_1, \ldots, i_k\}$ and $\{j_1, \ldots, j_k\}$ are related via Equation (1), the values of α and Δ can be estimated via least-squares.

It has already been established that the trajectory data in frames $\{i_1, \ldots, i_k\}$ of $\mathcal{S}$ and frames $\{j_1, \ldots, j_k\}$ of $\mathcal{S}'$ satisfy the known epipolar geometry. If the recovered values of α and Δ are correct, then other temporally corresponding points on the ball's trajectory will also satisfy the epipolar geometry.

4.3 Extending the algorithm to handle multiple vertical motions

The method presented in the previous section deals with the case where the video sequences are short and contain an object moving upwards and then downwards. In this section, the basic algorithm is extended to handle longer video sequences

containing multiple upward and downward motions, thus broadening the range of video sequences that can be synchronized.

In Chapter 3, an algorithm was presented to synchronize a pair of video sequences containing multiple significant vertical motions. In the coarse synchronization step presented in Section 3.2, a method of proposing corresponding subsequences from each sequence was described. Sub-sequences were introduced in Section 3.2.1, and it can be seen that the video sequences required by the basic algorithm described in the previous subsection fit the definition of a sub-sequence.

To synchronize sequences containing many sub-sequences, the basic algorithm is extended here to allow corresponding frames to lie within any sub-sequence that has a proposed corresponding sub-sequence in the other sequence. The extended algorithm is outlined as follows:

1. The proposed corresponding sub-sequence pairs from each sequence are found using the algorithm from Section 3.2.

2. A pair of corresponding sub-sequences, (s_n, s'_m) is randomly selected. Then, a frame $\mathcal{S}_i$ containing trajectory data is randomly selected from s_n.

3. The ball's direction of vertical motion in frame $\mathcal{S}_i$ is computed. Then, the set of frames in s'_m where the ball has the same direction of vertical motion is determined and frame $\mathcal{S}'_j$ is randomly selected from these frames. If there are no frames in s'_m with the same direction of vertical motion, the algorithm discards this chosen frame $\mathcal{S}_i$ and repeats from Step 2 above.

4. The numbered steps in Section 4.2.1 are used to iteratively compute the frame $\mathcal{S}'_j$ corresponding to frame $\mathcal{S}_i$. Then, Steps 2 to 4 above are repeated multiple times to determine a set of frame correspondences from which α and Δ can be recovered.

A critical assumption made in this extension is that the proposed corresponding sub-sequences from each sequence are correctly identified. If the video sequences to be synchronized contain many short sub-sequences, some proposed corresponding sub-sequence pairs may be incorrect and the least squares method used to recover

α and Δ as outlined in Section 4.2.3 may not be suitable. Instead, RANSAC could be used to fit a straight line with gradient α and y-intercept Δ to the frame indices $\{i_1, \ldots, i_k\}$ and $\{j_1, \ldots, j_k\}$, thus ensuring that any outliers have no influence on the recovered synchronization parameters.

Thus, if the fundamental matrix relating the two cameras is known, this extension to the algorithm enables synchronization of extended video sequences containing multiple ballistic motions.

4.4 Results

The results for synchronizing synthetic and real data sets are presented in this section. The synthetic data allow the accuracy of the algorithm to be evaluated in the presence of added noise, whilst the experiments using real video sequences demonstrate that the algorithm can be applied in a practical setting. First, I analyze the effect of adding noise to the trajectories in synthetic sequences, and then to the stationary points used to estimate the fundamental matrix. Then, I present the results of synchronizing real video sequences. These results are divided into two subsections: firstly, using the basic algorithm from Section 4.2 to synchronize short video sequences containing a single ballistic motion, and secondly, using the extended version of the algorithm, presented in Section 4.3, to synchronize longer video sequences containing many such motions, some of which may be visible in only one video sequence due to occlusions or the ball moving out of frame in one view.

4.4.1 Simulated experiments

I used simulated trajectories to analyze the accuracy of my algorithm in the presence of noise. The setup used in my experiments on synthetic data is shown in Figure 21, where I simulated a ball being kicked towards a set of football goals. The fundamental matrix was estimated from 12 points located at the corners of the goals and line markings. The ball's maximum vertical speed was 26 pixels/frame at the point of the trajectory closest to the camera, and averaged 7 pixels/frame

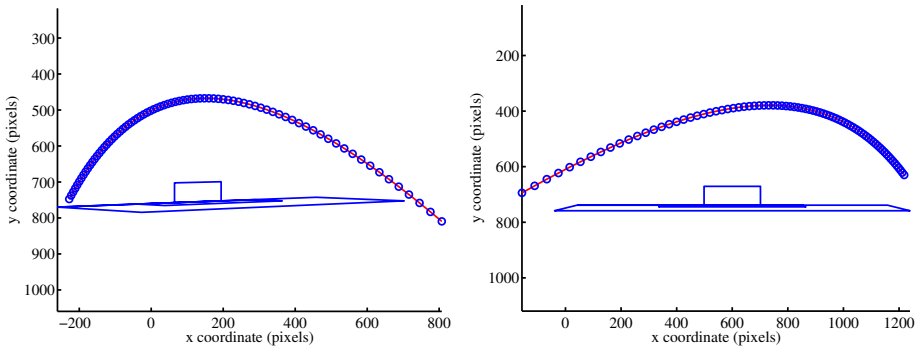


Figure 21: A simulated trajectory and football goals as viewed by two cameras, with the ball locations in each frame marked by circles.

σ	k	$\widehat{\alpha}$ error (%)	$\widehat{\Delta}$ error (frames)	Iterations
0.25	2	0.261	0.144	3.8
0.25	5	0.059	0.032	3.8
0.25	8	0.042	0.021	3.8
0.50	2	0.415	0.236	4.0
0.50	5	0.174	0.089	4.0
0.50	8	0.093	0.047	4.0
1.00	2	1.624	0.802	4.2
1.00	5	0.346	0.192	4.2
1.00	8	0.226	0.146	4.2

Table 7: Results for recovering $\widehat{\alpha}$ and $\widehat{\Delta}$ from k pairs of corresponding frames when isotropic Gaussian noise of standard deviation σ is added to the trajectory coordinates. The number of iterations required for convergence for each case is shown. The errors shown are the mean over 100 trials.

throughout both sequences.

To evaluate the accuracy of the algorithm in the presence of noise, isotropic Gaussian noise was added to the x and y components of the ball's position in each frame, and no smoothing process was applied to the trajectory. Table 7 shows the results for recovering the synchronization from trajectories perturbed by Gaussian noise of a range of standard deviations, σ. In these experiments, the ratio of frame rates and the frame offsets were fixed to isolate the influence of noise. The true values were $\bar{\alpha} = 5/6$ and $\bar{\Delta} = 3.5$ frames, and the sequences contained 80 frames. At each level of noise, I compared the results of using 2, 5, and 8 pairs of corresponding frames in estimating $\widehat{\alpha}$ and $\widehat{\Delta}$.

It can be seen from Table 7 that when noise is present, the accuracy of the

Goals	σ	$\widehat{\alpha}$ error (%)	$\widehat{\Delta}$ error (frames)	Iterations
Standard	0.25	0.430	0.189	3.8
	0.50	0.861	0.366	3.9
	1.00	1.703	0.690	3.6
Small	0.01	2.789	1.285	3.4
	0.05	7.873	3.121	3.8
Distant	0.25	1.168	0.538	3.8
	0.50	1.688	0.702	3.8
	1.00	3.512	1.621	3.6

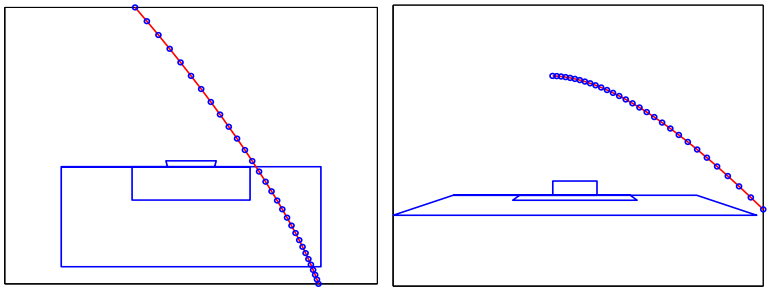
Table 8: The mean errors in estimating $\widehat{\alpha}$ and $\widehat{\Delta}$ when the fundamental matrix is estimated from points affected by noise. The synchronization parameters were calculated from 5 pairs of corresponding frames, over 100 trials.

recovered synchronization parameters, $\widehat{\alpha}$ and $\widehat{\Delta}$, is significantly improved by solving for more pairs of corresponding frames. Even with significant amounts of noise, the algorithm successfully computes the ratio of frame rates. The recovery of the frame offset is affected by noise more than the recovery of the frame rate ratio, however acceptable accuracy is still achieved. As expected, the rate of convergence decreases as the level of noise increases. In the noise free case, an average of 3.63 iterations were required for the algorithm to converge to sub-frame accuracy.

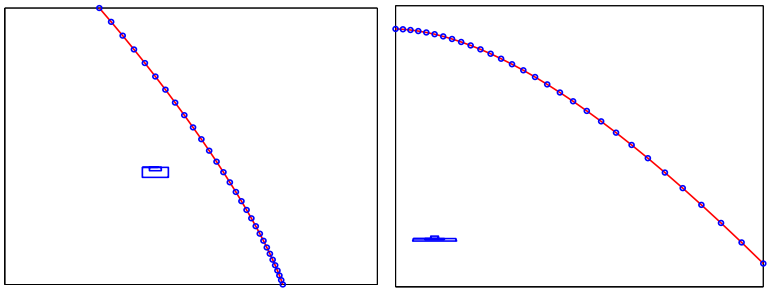
The accuracy of this algorithm relies heavily on the accurate estimation of the fundamental matrix; errors in computing epipolar lines will lead to errors in determining pairs of corresponding frames. The effect of estimating the fundamental matrix from stationary points perturbed by noise and the importance of the position of the selected stationary points relative to the trajectory were examined.

In the following experiments, the locations of the twelve points used to estimate the fundamental matrix were perturbed by isotropic Gaussian noise of various values of σ. Three sets of experiments were conducted: first, using the same football goals as in the previous experiment; second, with the goals one-tenth the size of those used previously; third, with the standard size goals located 100m from the trajectory in the virtual world. These three cases are illustrated in Figure 22.

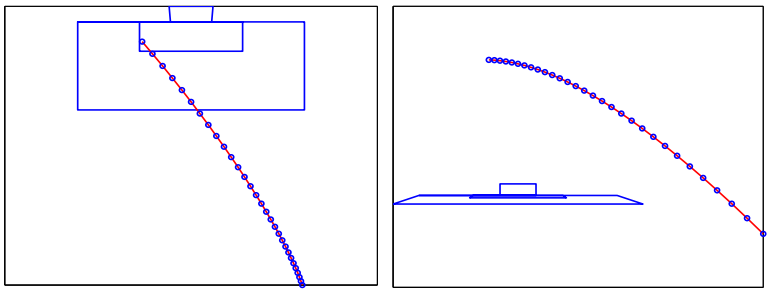
Table 8 shows the mean error in estimating the ratio of frame rates and frame offset over 100 trials. Again, I fixed the synchronization parameters at $\bar{\alpha} = 5/6$ and $\bar{\Delta} = 3.5$ frames, and no noise was added to the points in the trajectories.



(a) View 1 (normal) (b) View 2 (normal)



(c) View 1 (small) (d) View 2 (small)



(e) View 1 (distant) (f) View 2 (distant)

Figure 22: Three scenarios used to test the influence of noise on points used to estimate the fundamental matrix. (a) & (b): two views of large goals and a trajectory passing over the goals. (c) & (d): Goals one-tenth the size of the goals in the normal case, with the trajectory passing over the goals. (e) & (f): Large goals, but distant from the trajectory. The same trajectory was used in each experiment. Results are shown in Table 8.

From the results in Table 8, it is clear that suitable stationary points are required for estimating the fundamental matrix. This is highlighted in the Small and Distant cases. In the former case, the image of the goals spanned only ten pixels vertically, and less than one hundred horizontally, so the level of noise was significant. In the Distant case, the goals were placed 100m away from the trajectory in the virtual world, so the points used to estimate the fundamental matrix were not located close to the trajectory. Hence, in order to achieve accurate synchronization, it is essential

that the stationary points used for estimating the fundamental matrix are not only spatially well separated, but also surround the trajectory in each dimension. It can also be noted in this experiment that fewer iterations were required on average for convergence than in the experiments where noise was added to the trajectory data as detailed in Table 7; here, the trajectory was not perturbed by noise, leading to a more accurate estimation of the ball's velocity, and faster convergence.

4.4.2 Real video sequences containing a single ballistic trajectory

The algorithm was tested on real video sequences recorded by cameras capturing 25 interlaced frames per second. Each interlaced frame was separated into two independently recorded fields, so the data were captured at 50 fields per second; the resolution of each field was 720×288 pixels. For each data set, at least 12 stationary background points were manually selected from which the fundamental matrix was estimated. The centroid of the ball was manually tracked throughout the video sequences, although the method of tracking should not affect the synchronization algorithm.

Figures 23 to 26 show frames taken from the real video sequences, with the ball's trajectory overlaid. In these figures, the ball's trajectory throughout each of the short video sequences is displayed. In one view, some points on the trajectory are circled, and in the other view, the epipolar lines corresponding to these points are shown and the locations where these epipolar lines intersect with the trajectory are marked.

Table 9 summarizes the results of synchronizing the real video sequences. Where $\bar{\alpha}$ is given as 0.5, even-numbered frames were discarded from one sequence to test the estimation of $\widehat{\alpha}$. "Forced 1" indicates that I assumed that $\bar{\alpha} = 1$, and estimated $\widehat{\Delta}$ accordingly.

For each experiment where the frame rate was not altered artificially, the recovered synchronization parameters were used to resample the trajectory data of $\mathcal{S}$ (using linear interpolation of the trajectory data) to match the frame rate of $\mathcal{S}'$

Sequence pair	$\bar{\alpha}$	$\widehat{\alpha}$	$\bar{\Delta}$ (frames)	$\widehat{\Delta}$ (frames)
outdoor	1	0.94	4.5	4.73
	1	Forced 1	4.5	4.50
	0.5	0.51	2.75	2.59
indoor2	1	1.00	−8	−8.32
	1	Forced 1	−8	−8.36
	0.5	0.48	−3.5	−3.21
kick6	1	1.01	−20	−19.62
	1	Forced 1	−20	−19.31
	0.5	0.51	−9.5	−10.64
kick8	1	0.99	−67	−67.26
	1	Forced 1	−67	−67.36
	0.5	0.50	−33	−32.57

Table 9: The result of synchronizing pairs of real video sequences using the basic algorithm.

Sequence pair	Mean symmetric epipolar distance
outdoor	3.52 pixels^2
indoor2	10.71 pixels^2
kick6	5.32 pixels^2
kick8	1.05 pixels^2

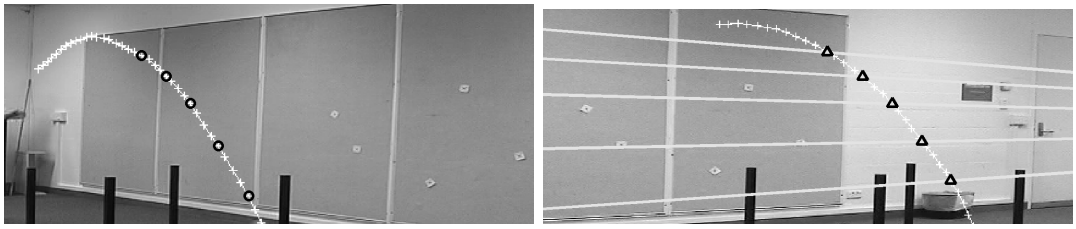
Table 10: The mean symmetric epipolar distance resulting from using the recovered synchronization parameters for pairs of real video sequences using the basic algorithm.

and nullify the frame offset. Then the ball's location in each frame of $\mathcal{S}$ was used to compute an epipolar line in the corresponding frame of $\mathcal{S}'$. The mean symmetric epipolar distance was then computed for all corresponding frames f, via:

$$d_{\text{sym}} = \frac{1}{N}\sum_{f}^{N} d_{\perp}(\mathbf{x}'_f, \mathtt{F}\mathbf{x}_f)^2 + d_{\perp}(\mathbf{x}_f, \mathtt{F}^{\mathsf{T}}\mathbf{x}'_f)^2, \tag{6}$$

where $d_{\perp}(\mathbf{x}, \mathbf{l})$ returns the perpendicular distance between the point $\mathbf{x}$ and the line $\mathbf{l}$. Table 10 displays the mean symmetric epipolar distance for each experiment.

The results demonstrate that for real video sequences, α and Δ can be recovered to acceptable accuracy. Further, if α is known, the accuracy of the recovered frame offset is improved. It can be seen that the corresponding frames used to recover α and Δ in the real video sequences are typically well-spaced. Whilst this is not required by the algorithm, it improves the accuracy of the least-squares estimation of α and Δ. The mean symmetric epipolar distance is also shown to be low, indicating



(a) (b)

Figure 23: Corresponding frames taken from the `indoor2` sequences, with the ball's trajectory overlaid. The five corresponding frames used for synchronization are indicated by circles in (a), with the corresponding epipolar lines shown in (b), and the ball's position indicated by triangles.

(a) (b)

Figure 24: Corresponding frames taken from the `outdoor` sequences, zoomed in for greater detail. The image markings are the same as in Figure 23.

that the supplied trajectory data and the recovered synchronization parameters satisfy the known epipolar geometry.

It is seen in Figures 23 to 26 that the magnitude of the ball's inter-frame vertical velocity is not large. However, it is sufficient for synchronization provided that the pairs of corresponding frames used for recovering α and Δ are not located near the peak of the trajectory, where the magnitude of the vertical velocity is at a minimum.

It is shown that although the ball may be tracked through many frames, few pairs of corresponding frames are required for synchronization, as demonstrated in Figures 23 to 26. In the real video sequences used in these experiments, the number of frames in which the ball was tracked ranged from 22 to 57 frames. These results confirm that the algorithm, whilst simple, is efficient.

(a) (b)

Figure 25: Corresponding frames taken from the `kick6` sequences. The image markings are the same as in Figure 23.

(a) (b)

Figure 26: Corresponding frames taken from the `kick8` sequences. The image markings are the same as in Figure 23.

4.4.3 Synchronizing longer real video sequences using the extended algorithm

In the previous experiments, the videos to be synchronized contained the ball moving upwards once and then moving downwards once. In Section 4.2.3, an extension to the algorithm was described that synchronizes longer video sequences containing multiple vertical motions. In this sub-section, the results of applying this algorithm to various real video sequences synchronized in Chapter 3 are presented.

Table 11 shows the results of synchronizing these of video sequences. The frame rate ratio and frame offset recovered by this algorithm are denoted by $\widehat{\alpha}$ and $\widehat{\Delta}$, respectively. Also, the frame rate ratio and frame offset recovered using the algorithm presented in Chapter 3 are provided for comparison purposes and are denoted by $\alpha^{\dagger}$ and $\Delta^{\dagger}$, respectively. Further, numerical results are also presented for both algorithms where the frame rate ratio is known, as indicated by "Forced 1" and "Forced 2". In these video sequences, the fundamental matrix was estimated from 12 to 16 manually selected points visible in each view and 8 pairs of corresponding frames were computed. These corresponding frames were contained within various

Sequence pair	$\bar{\alpha}$	$\widehat{\alpha}$	$\alpha^\dagger$	$\bar{\Delta}$	$\widehat{\Delta}$	$\Delta^\dagger$
`indoor`	1	0.9992	1.0000	8.0	8.42	8.50
		Forced 1			8.01	8.48
`outdoor`	1	1.0017	1.0000	738.0	738.14	737.73
		Forced 1			738.81	737.74
`backyarda`	2	2.0025	2.0000	−172.5	−172.05	−172.25
		Forced 2			−171.36	−172.24
`backyardb`	1	0.9985	1.0000	39.5	38.91	39.19
		Forced 1			39.29	39.21

Table 11: Using the extended algorithm to synchronize extended video sequences containing multiple vertical motions. The values $\alpha^\dagger$ and $\Delta^\dagger$ denote respectively the frame rate ratio and frame offset of the sequences as recovered by the algorithm described in Chapter 3.

sub-sequences evenly distributed throughout the video sequences.

It can be seen in Table 11 that the values of $\widehat{\alpha}$ and $\widehat{\Delta}$ are recovered accurately. However, the comparison with the values recovered by the algorithm from Chapter 3 indicate that that algorithm provides slightly better accuracy, particularly in the estimation of the frame rate ratio. It should be noted that, as demonstrated in Table 7, increasing the number of corresponding frames improves the accuracy of the algorithm where the trajectory data are affected by noise. As the image measurements are not expected to be significantly affected by noise, and since the algorithm from Chapter 3 typically recovers α and Δ from hundreds of pairs of corresponding frames rather than the 8 pairs used here, it is expected that the accuracy of that algorithm is greater than that of this algorithm.

To summarize, the extended algorithm presented in this chapter provides satisfactory synchronization performance using a fast and simple implementation, but it is not as accurate as the algorithm described in Chapter 3. In cases where the fundamental matrix is known, this algorithm could be used as a fast alternative to the method described in Sections 3.2.6 and 3.3.1 to initialize the 2D search for the minimum of the 9th singular value in Δ-α space.

4.5 Discussion

4.5.1 Degenerate cases

There are some situations in which degeneracies may occur in this algorithm due to the manner in which epipolar geometry is exploited. Firstly, care must be taken when choosing the frame $\mathcal{S}_i$ for synchronization. There may be a point on the trajectory such that the ball's approximated trajectory $\mathbf{t}'$ in $\mathcal{S}'_j$ is parallel to the epipolar line $\mathbf{l}'_i$ corresponding to the ball's location in $\mathcal{S}_i$, as shown in Figure 27. In this case, Step 3 in Section 4.2.1 yields a point $\mathbf{p}'$ at infinity, and there is no finite solution for the number of frames until the ball crosses the epipolar line. To resolve this problem, a different frame $\mathcal{S}_j$ should be chosen such that the ball has significant vertical velocity: i.e., its motion is predominantly vertical such that it is not moving parallel to $\mathbf{l}'_i$ (as $\mathbf{l}'_i$ is expected to be more horizontal than vertical), and the ball is not near the peak of the ballistic trajectory.

Choosing a frame $\mathcal{S}_i$ where the ball's location coincides with the epipole will result in a degeneracy in computing the corresponding epipolar line in the other view. In this case, another frame $\mathcal{S}_i$ should be randomly selected.

It is assumed that the vertical separation of the cameras is small relative to the lateral separation. Hence, when the ball has significant vertical velocity, it is expected that it will not move parallel to an epipolar line, which is likely to be almost horizontal. Sometimes, an epipolar line may not intersect with the trajectory due to errors in estimating `F`. If this occurs, another frame $\mathcal{S}_i$ should be chosen where the ball has significant vertical velocity and hence will not be at the peak of its trajectory; as a result, $\mathbf{l}'_i$ will not be tangential to the trajectory.

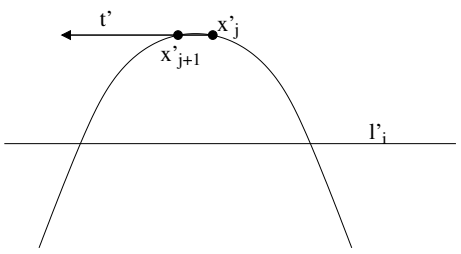


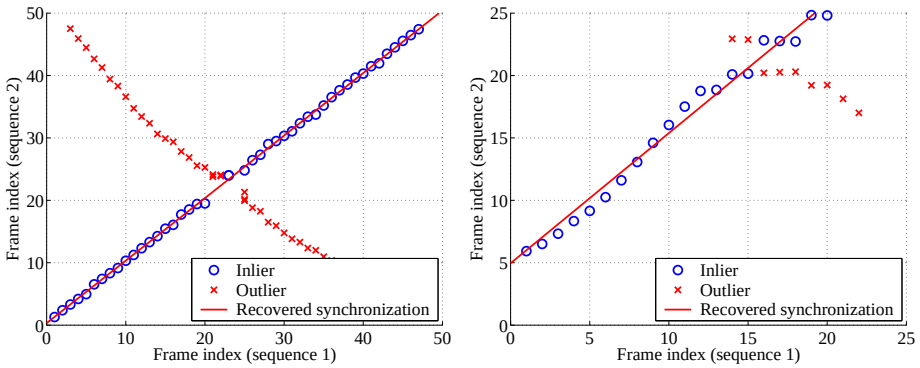
Figure 27: A degenerate case where the ball's approximated trajectory is parallel to the epipolar line. In this case, $\mathbf{t}'$ will intersect $\mathbf{l}'_i$ at a point at infinity.

4.5.2 Comparison with an algorithm by Carceroni et al.

As previously mentioned, this algorithm shares similarities with an algorithm by Carceroni et al. [1] in that both algorithms require weakly calibrated cameras and epipolar line crossings are used to propose temporally corresponding frames. However, Carceroni's algorithm is more general in that it can synchronize multiple video sequences containing arbitrary object motion. In this section, I address some problems that may be encountered in using Carceroni's algorithm to synchronize pairs of video sequences using the same data that my algorithm uses. It is important to note that in general use, it is likely that Carceroni's algorithm would derive extra tracking data from other moving objects visible in each sequence, rather than a single object tracked through a sequence.

Firstly, my algorithm differentiates between upwards and downwards motion. When determining frames in which an object crosses an epipolar line, it is likely that Carceroni's algorithm will propose two frame correspondences: one when the ball crosses the epipolar line and is moving upwards and one when it is moving downwards. This should not cause any problems when RANSAC is used to recover the synchronization parameters, since a simple constraint can be enforced to ensure that the recovered value of α is positive; i.e., it is assumed that time flows forwards in both sequences. This is demonstrated in Figure 28, where the distributions of the proposed corresponding frames for the `take8` and `outdoor` sequences are shown; the set of points in each plot that form a line with a positive gradient correspond to the correct corresponding frames, and those forming a line with a negative gradient are due to the incorrect epipolar line crossings.

Previously, it has been stated that my algorithm should only select frames where the ball has significant vertical velocity, i.e., where the ball's motion and the corresponding epipolar line are not almost parallel. Carceroni's algorithm does not enforce such a motion constraint. Errors in estimating the fundamental matrix may cause the gradient of the epipolar lines to be incorrect, thus affecting the location of where an object crosses an epipolar line. As a result, any proposed pairs of corresponding frames where the ball's motion is almost parallel to the epipolar line are sensitive to the accurate estimation of the fundamental matrix. As RANSAC is



(a) `take8` sequences (b) `outdoor` sequences

Figure 28: Proposed corresponding frames determined via Carceroni's algorithm for two pairs of video sequences.

employed to fit the timeline, their algorithm should not be significantly affected by the inclusion of outliers. However, experiments have identified a potential source of error introduced by RANSAC.

In Figure 28(b), a significant number of points can be seen that form two almost horizontal lines at the top of the plot. These points correspond to frames where the ball's velocity is almost parallel to the proposed corresponding epipolar lines. As the `outdoor` video sequences are short and the number of proposed corresponding frames is low, these points significantly increase the percentage of outliers. It can be seen that some of these points lie close to the straight line representing the synchronization parameters. If a large threshold distance is used by RANSAC to determine whether a point representing a pair of corresponding frames is an inlier, these erroneous points will affect the accuracy of the recovered synchronization parameters. Conversely, if the threshold is too low in order to eliminate these outliers, RANSAC may fit a straight line to the outlying points representing frames where the ball's motion is almost parallel to the epipolar lines, rather than fitting a line to the points representing actual corresponding frames.

It is noted that Carceroni et al. employ RANSAC to recover the synchronization parameters. Although it could be argued that RANSAC could also be used in a similar capacity in place of the least-squares method used in the basic case of my algorithm described in Section 4.2.3, this is not considered to be necessary when synchronizing short sequences containing only one ballistic trajectory. Whereas

Carceroni's algorithm attempts to find epipolar line crossings that may be due to any moving object within the video sequence, my algorithm requires a single moving object to be tracked; as such, any incorrectly proposed corresponding frames would be due to incorrect computation of the intersection point of the ball's trajectory with the epipolar line, rather than incorrectly matching inter-frame motions from different objects. Further, it is expected that my algorithm will locate the corresponding frames correctly due to the following reasons. In the basic case of my algorithm, it is assumed that each of the short video sequences contains only one contiguous set of frames where the ball moves upwards and one contiguous set of frames where the ball moves downwards. Hence, no ambiguity arises from the direction of the ball's motion. If the ball is moving upwards in a frame from $\mathcal{S}$, the corresponding frame from $\mathcal{S}'$ must lie within the contiguous set of frames in $\mathcal{S}'$ where the ball moves upwards. Recall that the initial selection of frames requires that the ball's direction of vertical motion is the same in the selected frames of both sequences. As my algorithm converges as described in Section 4.2.2, the correct corresponding frame should be located. Thus, since no outlying pairs of corresponding frames are expected to be computed, it is not necessary for the basic algorithm to use a random sampling method to eliminate outliers.

4.6 Conclusion

An algorithm has been presented that uses object motion to recover the ratio of frame rates and the temporal offset of video sequences recorded by two weakly calibrated stationary cameras with fixed intrinsic parameters. The basic case of the algorithm was shown to converge rapidly to locate a pair of corresponding frames within a pair of short video sequences containing an object undergoing ballistic motion. This basic case was extended by incorporating the concept of sub-sequences, introduced in the previous chapter, to allow the algorithm to synchronize pairs of longer video sequences containing several ballistic motions.

Experiments on synthetic and real data have shown that the algorithm provides

a simple and fast method of synchronization. The recovered synchronization parameters were shown to be accurate via comparison with manual synchronization and an analysis of the reprojection error of temporally corresponding trajectory locations using the known epipolar geometry. As expected, as the level of noise affecting the trajectory coordinates or the stationary background points used to estimate the fundamental matrix increases, the algorithm's accuracy decreases gradually and gracefully. The fundamental matrix plays an important role in my algorithm and needs to be estimated accurately. The results of synchronizing a number of extended real sequences have shown that this algorithm can accurately and rapidly recover the frame rate ratio and frame offset of a pair of video sequences.

Chapter 5

Synchronization from space-time interest points

5.1 Introduction

The synchronization algorithms introduced in Chapters 3 and 4 utilized trajectory information to recover the synchronization parameters. In particular, in Chapter 4, pairs of temporally corresponding frames were determined via the epipolar constraint; from these pairs of frames and Equation (1), the frame rate ratio, α, and the frame offset, Δ, of the sequences were recovered. In this chapter, an algorithm is proposed that determines pairs of temporally corresponding frames by finding corresponding *space-time interest points* from each video sequence. The algorithm synchronizes pairs of video sequences without requiring camera calibration or object tracking. Space-time interest points, as introduced by Laptev [20, 22], are locations in video sequences where a significant change in pixel intensity exists both spatially, i.e., within a frame of a sequence, and temporally, i.e., between consecutive frames of a sequence. Such interest points are commonly detected where events, such as objects merging, splitting, or changing direction, occur. Space-time interest points are presented in more detail in Section 5.2.

There has been little previous work in using space-time interest points for video sequence synchronization. As mentioned in Chapter 2, Yan and Pollefeys [53] recovered the frame offset of a pair of video sequences by cross-correlating the temporal

distribution of space-time interest points at each integer frame offset; the frame offset yielding the greatest correlation score was returned as the recovered frame offset. This process did not determine pairs of corresponding space-time interest points from each sequence; rather, it assumed that an event that was visible in both sequences resulted in a similar number of space-time interest points being detected in each sequence.

Laptev et al. [21] employed space-time interest points for detecting periodic motion within a single sequence. Firstly, pairs of space-time interest points that were proposed to be periodically equivalent were determined. That is, the interest points that were matched were due to the same event occurring in different periods of the periodic motion. Each pair of points were separated temporally by some time offset Δ; for points that were in fact periodically equivalent, Δ was a multiple of the period length. Then, a RANSAC-based approach was used to recover pairs of periodically equivalent points. At each RANSAC iteration, two pairs of space-time interest points with similar Δ values were selected and a reduced fundamental matrix was estimated; the number of inliers was the number of pairs of points that supported the proposed value of Δ and the proposed reduced fundamental matrix. A property of the reduced form of the fundamental matrix introduced by Laptev et al. is that periodically equivalent interest points lie on the same epipolar line. Thus, the epipolar lines coincide with the direction of the periodic motion.

In this chapter, an algorithm is presented that synchronizes two sequences recorded by stationary cameras with fixed intrinsic parameters. Neither weak camera calibration nor object tracking are required. Firstly, space-time interest points are detected in each sequence, and putatively matching interest points from each sequence are proposed. Next, a two step nested RANSAC approach firstly proposes a *temporal model*, i.e., the synchronization parameters α and Δ, that relates the two sequences. Then, the best *spatial model*, defined as either a homography for sequences containing planar motion or a fundamental matrix for sequences containing free object motion in 3D space, is recovered for the proposed temporal model. A special case of the algorithm is also presented for use when the spatial model is known. In this case, a single instance of RANSAC is used to recover the temporal

model only.

This algorithm is similar to the periodic motion detection algorithm by Laptev et al. [21] in that a temporal model is proposed first and a spatial model is then fitted to the putative matches that satisfy the proposed temporal model. However, whereas Laptev's algorithm searches for constant-rate periodic motion within the same video sequence, my algorithm synchronizes two views of the same event recorded at different frame rates. A more detailed comparison is provided in Section 5.7.2.

The structure of this chapter is as follows. Firstly, space-time interest points are introduced. Then, three interest point descriptors are described, two utilizing the widely used SIFT descriptor, and the third based on the *local N-jet* which is computed from derivatives of the space-time volume; local N-jets are introduced in Section 5.3.1. Next, a method of determining putatively matching space-time interest points points is outlined. The algorithm that recovers the spatial and temporal models relating the two sequences is then presented, and results given for using all descriptors for synchronizing pairs of sequences containing either motion on a plane or free motion in 3D space. Finally, the ordering of the two RANSAC steps in the algorithm is discussed, and comparisons with the aforementioned algorithms by Yan and Pollefeys, and Laptev et al. are presented.

5.2 Space-time interest points

Space-time interest points, introduced by Laptev [20, 22], are locations in greyscale video sequences where a large variation in pixel intensities exists in space (within each frame) and time (between frames). They may be considered to be the equivalent in video sequences of spatial interest points in still images, e.g., Harris corners [12]. Interest points may be detected where a variety of events occur, such as where and when an object appears to change direction, where objects appear to merge or separate, or where an object is occluded or dis-occluded. It is likely that interest points detected due to occlusion and dis-occlusion events in one sequence would either appear at a different time and location in the other sequence, or not

at all. However, it is expected that the following steps of determining putatively matching space-time interest points from each sequence and then rejecting outlying putative matches via RANSAC will ensure that these events do not affect the synchronization process.

To detect space-time interest points, a second moment matrix, $\mathtt{M}$, is constructed for each pixel location (x, y) in each frame t of a sequence $\mathcal{S}$:

$$\mathtt{M} = \begin{bmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{bmatrix}, \tag{7}$$

where L_u, the first order derivative in the u dimension of the video sequence, is computed via:

$$L_u(x, y, t; \sigma^2, \tau^2) = \frac{\partial}{\partial u}(g(x, y, t; \sigma^2, \tau^2) * \mathcal{S}), \tag{8}$$

and $g(x, y, t; \sigma^2, \tau^2)$ is a separable Gaussian kernel, given by:

$$g(x, y, t; \sigma^2, \tau^2) = \frac{\exp(-(x^2 + y^2)/(2\sigma^2) - t^2/(2\tau^2))}{\sqrt{(2\pi)^3 \sigma^4 \tau^2}}, \tag{9}$$

where $*$ denotes convolution and σ^2 and τ^2 denote the independent spatial and temporal variances, respectively. Then, space-time interest points are located at positive local maxima of the corner function H, where:

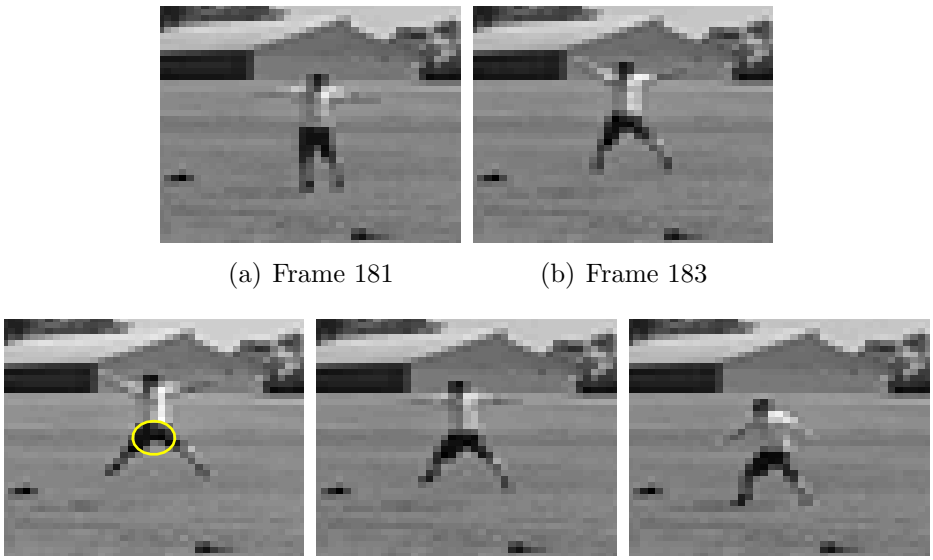
$$H = \det(\mathtt{M}) - k \ \mathrm{tr}^3(\mathtt{M}). \tag{10}$$

Laptev suggests using $k \approx 0.005$. The positive local maxima of H correspond to (x, y, t) locations within the video sequence where the three eigenvalues of $\mathtt{M}$ are significant. The eigenvalues of $\mathtt{M}$ do not need to be explicitly calculated, since H can be computed directly from the elements of $\mathtt{M}$. However, the eigenvalues, λ_1, λ_2, and λ_3, are implied in the computation of the determinant and trace of $\mathtt{M}$ which are contained within H due to Equation (10), since [11]:

$$\det(\mathtt{M}) = \lambda_1 \lambda_2 \lambda_3, \tag{11}$$

and

$$\mathrm{tr}(\mathtt{M}) = \lambda_1 + \lambda_2 + \lambda_3. \tag{12}$$



(a) Frame 181 (b) Frame 183

(c) Frame 185 (d) Frame 187 (e) Frame 189

Figure 29: An example of the location of a space-time interest point, identified by the circle in (c), within a series of frames of a video sequence.

Let $\mathbf{x}_i = (x_i, y_i, t_i)^\mathsf{T}$ denote a space-time interest point in the space-time volume where (x_i, y_i) is the pixel location of the interest point in frame t_i of the video sequence.

Figure 29 shows a space-time interest point detected within a video sequence. It may be convenient to firstly consider that 2D spatial interest point detectors often locate features at the corners of objects. In this example, consider the set of video frames as a 3D image volume. The frames displayed show the subject performing a star jump on a grassy surface. As time passes, the grassed area visible between the person's legs expands from a narrow triangle in frame 181 to a wider triangle in frame 185, before becoming slightly narrower. Simultaneously, this triangle moves upwards between frame 181 and 185, and then moves downwards after the peak of the jump. In the image volume, these triangles would form a pyramid-like shape, with the interest point detected at the apex of the pyramid, which is a corner in 3D space.

5.3 Interest point descriptors

In this section, three types of interest point descriptors are outlined. A descriptor derived from the local N-jet is introduced first, followed by the SIFT descriptor and another proposed SIFT-based descriptor. A descriptor is required to be computed for every space-time interest point in each sequence. The synchronization algorithm is not dependent on the selection of the descriptor type but the same descriptor class must be computed for each point in each sequence. In the following section, putatively matching space-time interest points from each sequence are proposed by matching these descriptors.

5.3.1 The local jet descriptor

Consider the spatio-temporal pixel intensity information contained within the video sequence as a 3D volume. Then, the intensity values in the local volume surrounding a space-time interest point can be approximated via the Taylor series. The coefficients of the truncated Taylor series of degree N, i.e., the partial derivatives of the volume of up to and including order N, at the location of the space-time interest point are known as the *local N-jet*. Koenderink and van Doorn provide a discussion of the physical interpretations of the derivatives of various orders where local N-jets are used in the spatial domain [16].

Laptev and Lindeberg demonstrated that local 4-jets, i.e., the 34 partial derivatives of the image volume up to and including the fourth order, can be used to describe space-time interest points for the process of determining matching space-time interest points from different video sequences [23]. For practical purposes, the local 4-jet is represented as a 34-vector, $\mathbf{j}$, where:

$$\begin{aligned}\mathbf{j} \;=\; & [L_x, L_y, L_t, L_{xx}, L_{xy}, L_{xt}, L_{yy}, L_{yt}, L_{tt},\\ & L_{xxx}, L_{xxy}, L_{xxt}, L_{xyy}, L_{xyt}, L_{xtt}, L_{yyy}, L_{yyt}, L_{ytt}, L_{ttt},\\ & L_{xxxx}, L_{xxxy}, L_{xxxt}, L_{xxyy}, L_{xxyt}, L_{xxtt}, L_{xyyy}, L_{xyyt}, L_{xytt}, L_{xttt},\\ & L_{yyyy}, L_{yyyt}, L_{yytt}, L_{yttt}, L_{tttt}].\end{aligned}$$

An extension to the local 4-jet was proposed by Laptev and Lindeberg [23],

who described a method of incorporating local 4-jets computed over multiple spatial and temporal scales into a single descriptor. For a space-time interest point detected with spatial and temporal scales σ_0 and τ_0 respectively, local 4-jets were computed for each combination of σ and τ values, where $\sigma \in \{\sigma_0/2, \sigma_0, 2\sigma_0\}$ and $\tau \in \{\tau_0/2, \tau_0, 2\tau_0\}$. As each local 4-jet consists of 34 elements and there are 9 spatio-temporal scale combinations, the multi-scale local 4-jet descriptor is a 306-vector. As I only use the multi-scale local 4-jet in the following sections, the term *local jet* is used to refer to the 306-element multi-scale local 4-jet descriptor described here.

5.3.2 The SIFT descriptor

The widely used scale-invariant feature transform (SIFT) descriptor, developed by Lowe [26], is a 128-vector computed from the histograms of pixel intensity gradients in still images. The descriptors are invariant to rotation and changes in scale, and robust to a change in viewpoints. Previously, the SIFT descriptor has been successfully used for representing regions in still images. In this algorithm, it is used as an alternative descriptor for finding putatively matching space-time interest points. Since a SIFT descriptor is computed from spatial information from only one frame of a video sequence, it contrasts significantly with the local jet that is computed from spatio-temporal information.

To compute the SIFT descriptor for a given space-time interest point detected in frame t_i of a video sequence, the affine invariant region is first computed around the (x_i, y_i) location of the space-time interest point in that frame. This region is a window surrounding the interest point such that the second moment matrix of the region is isotropic, i.e., the magnitudes of the minimum and maximum eigenvalues of the second moment matrix are equal. Further details on affine invariant regions are provided by Mikolajczyk and Schmid [27].

Once the affine invariant region has been computed, the next step is to compute the SIFT descriptor for that region in frame t_i. The orientations of the gradients at each sample point surrounding the location of the centre of the affine invariant region are incorporated into orientation histograms, from which the descriptor vector is computed. Lowe provides further details on how to compute SIFT descriptors [26].

Sometimes, the location of an affine invariant region may not coincide exactly with the spatial location of a space-time interest point. When this occurs, the SIFT descriptor is computed for the affine invariant region closest to the detected space-time interest point in the frame where that interest point is detected. A threshold distance of 3 pixels between the location of the space-time interest point and its corresponding SIFT descriptor is enforced; if no affine invariant region exists within this distance, the space-time interest point is discarded. As SIFT descriptors are computed only for the affine invariant regions within a frame, the locations of space-time interest points with SIFT descriptors coincide with the locations of spatial interest points.

5.3.3 The 3SIFT descriptor

A significant difference between the local jet and the SIFT descriptor is that the former is computed from space-time pixel intensity information whereas the latter incorporates pixel data from only a single frame. I propose that the previously described process for computing a SIFT descriptor for each space-time interest point be extended by using *multiple frames* of a video sequence to yield a descriptor that incorporates temporal data. Where locations within an image with matching SIFT descriptors must have a similar visual appearance up to an affine transformation, matching features represented by the proposed SIFT descriptor computed over multiple frames must also have similar appearances at the spatial location of the detected interest point in frames before and after the frame containing the space-time feature.

As in the previous sub-section, an affine invariant region is computed around the location in frame t_i of the video sequence containing the space-time interest point. Then, SIFT descriptors are computed for this region in frames $\{t_i - 2, t_i, t_i + 2\}$ of the video sequence. These 128-element SIFT descriptor vectors are concatenated to produce a single 384-element descriptor vector. It is noted that the affine invariant regions in frames $t_i - 2$ and $t_i + 2$ are not necessarily located at a space-time interest point or even at a spatial interest point. However these regions should have a similar visual appearance in corresponding frames from each sequence and hence

should have similar SIFT descriptor vectors.

In order to differentiate between the SIFT descriptor described in the previous section and the SIFT-based descriptor computed for multiple frames described here, henceforth, the term *SIFT descriptor* shall refer to a SIFT descriptor computed from a single frame of a video sequence, and *3SIFT descriptor* will be used to denote a descriptor computed from multiple frames via the method described in this section.

5.4 Determining putative matches

Putatively matching space-time interest point pairs are proposed from the descriptor vectors for each interest point by firstly computing a distance measure for every pair of space-time interest points, and then determining putative matches from these distances. Given that we now have

$$S_{STIP} = \{\mathbf{x}_i = (x_i, y_i, t_i) \mid t_i > 0, 1 \leq i \leq N\}$$

and

$$S'_{STIP} = \{\mathbf{x}'_j = (x'_j, y'_j, t'_j) \mid t'_j > 0, 1 \leq j \leq N'\},$$

i.e., a set of N and N' space-time interest points detected in video sequences 1 and 2 respectively, the goal is to establish a set of putative matches, P, from S_{STIP} and S'_{STIP} via the following steps:

1. Initialize the set of putative matches, P, to an empty set: $P \leftarrow \varnothing$.

2. For each pairing of points, $\mathbf{x}_i \in S_{STIP}$ and $\mathbf{x}'_j \in S'_{STIP}$, the corresponding descriptor vectors are normalized to unit vectors and their Euclidean distance, d_{ij}, is computed. The normalization procedure is used to achieve invariance in contrast between the two frames of the video sequences. Apart from the Euclidean distance function adopted here, other distance measures can also be used; a discussion of measures suitable for use with multi-scale local jets can be found in Laptev and Lindeberg's paper [23].

3. The algorithm greedily selects the pair of putatively matching space-time interest points with the lowest distance score d_{ij}. Let the two putatively matching points be denoted by $\mathbf{x}_i$ and $\mathbf{x}'_j$.

4. • If local jets are the descriptor vectors selected to represent the space-time interest points, the pair of points $\mathbf{x}_i$ and $\mathbf{x}'_j$ are declared as a putative match. The set of putative matches, P, is then updated via:

$$P \leftarrow P \cup \{(\mathbf{x}_i, \mathbf{x}'_j)\}. \tag{13}$$

   • If the distance measure d_{ij} is computed from SIFT or 3SIFT descriptors, a stronger condition on matching can be achieved by enforcing Lowe's suggestion [26] that the ratio of the distance of the closest matching point to the second-closest matching point should be no more than 0.8. More specifically, for $\mathbf{x}_i$, the second best distance score, d_{ik}, corresponding to the point $\mathbf{x}'_k \in S'_{STIP}$, is determined; for $\mathbf{x}'_j$, the second best distance score, d_{hj}, corresponding to the point $\mathbf{x}_h \in S_{STIP}$, is also computed. Then, if both $d_{ij}/d_{ik} < 0.8$ and $d_{ij}/d_{hj} < 0.8$, $\mathbf{x}_i$ and $\mathbf{x}'_j$ are said to be a putative match and P is updated as in Equation (13).

   It is noted that if 3SIFT descriptors are used, this ratio constraint applies to the entire 384-element 3SIFT vector. Although the 3SIFT descriptor consists of three 128-element SIFT descriptors and the ratio constraint could be applied independently to each 128-element component, this is not enforced in practice.

5. To enforce a one-to-one mapping and to prevent the matching process from entering an infinite loop, all remaining pairs of interest points containing either $\mathbf{x}_i$ or $\mathbf{x}'_j$ are removed from further consideration, via:

$$S_{STIP} \leftarrow S_{STIP} \backslash \{\mathbf{x}_i\}$$

   and

$$S'_{STIP} \leftarrow S'_{STIP} \backslash \{\mathbf{x}'_j\}$$

   Then, the process repeats from Step 3, using the reduced sets S_{STIP} and S'_{STIP}, until either $S_{STIP} = \varnothing$ or $S'_{STIP} = \varnothing$.

5.5 Recovering the synchronization

RANSAC [9] is a random sampling method for fitting a model to a data set containing outliers. The synchronization algorithm that I present here employs RANSAC to recover the temporal model from the set of putatively matching space-time interest points, P. Firstly, I outline the general case of the algorithm where neither the temporal nor the spatial models are known; in this case, the algorithm estimates both models via two nested instances of RANSAC. Secondly, in some applications such as video surveillance, the cameras may be permanently mounted and the spatial model, either a homography or a fundamental matrix, may be known. For these applications, I describe a special case of the algorithm that exploits the known spatial model to improve the putative matches and then uses only a single instance of RANSAC to recover the synchronization parameters.

5.5.1 Recovering the spatial and temporal models

To recover the temporal and spatial models that relate two video sequences, my algorithm uses two nested instances of RANSAC. The first, outer instance recovers the temporal model, i.e., the synchronization parameters α and Δ. The inner instance estimates the spatial model, either a homography or a fundamental matrix for sequences containing planar motion and free motion in 3D space, respectively. The structure of the standard RANSAC algorithm is displayed in Figure 30, and is contrasted with Figure 31 which outlines the nested approach presented in this algorithm. It can be seen that the step in which inliers are computed in the standard RANSAC algorithm is replaced in my algorithm by another complete instance of RANSAC. My algorithm is described in detail below:

1. Two pairs of putative matches are randomly selected from the set of putative matches, P. Let the temporal components of one pair of putatively matching space-time interest points be denoted by t_i and t'_j, and let the temporal components of the other pair of putatively matching space-time interest points be denoted by t_k and t'_l. Then for each putative match, *temporal component pairs* are constructed representing the two interest points' temporal components in

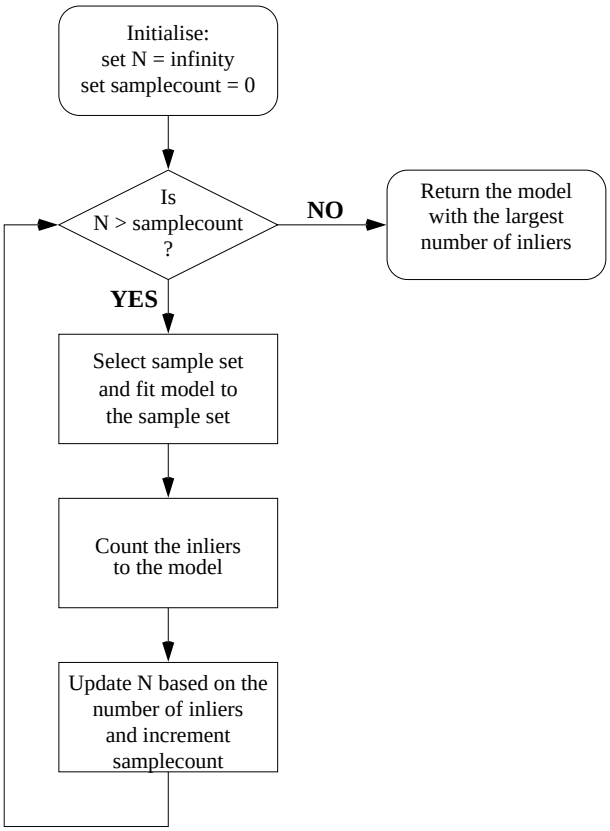


Figure 30: A generic RANSAC framework used to recover a single model. N is the number of samples to try, based on the probability that at least one sample is free of outliers, the probability that a data point is an inlier, and the size of the sample set. Hartley and Zisserman provide further details on computing N adaptively [13].

a 2D space, i.e., (t_i, t'_j) and (t_k, t'_l). An example of the temporal component pairs for all putative matches is displayed in Figure 32(a). Next, a straight line is fitted to these two temporal component pairs. This straight line is the temporal model as it encapsulates the synchronization parameters α and Δ in the line's gradient and y-intercept, respectively, via Equation (1).

2. Next, the inliers for the temporal model are determined. Putative matches whose temporal component pairs lie within a threshold perpendicular distance from the straight line proposed in the previous step are *temporal inliers* as their temporal components are consistent with the values of α and Δ. Figure 32(b) provides an example of fitting a temporal model to a set of temporal component pairs and determining temporal inliers. A strict distance threshold should be employed as we are searching for putative matches that rigidly

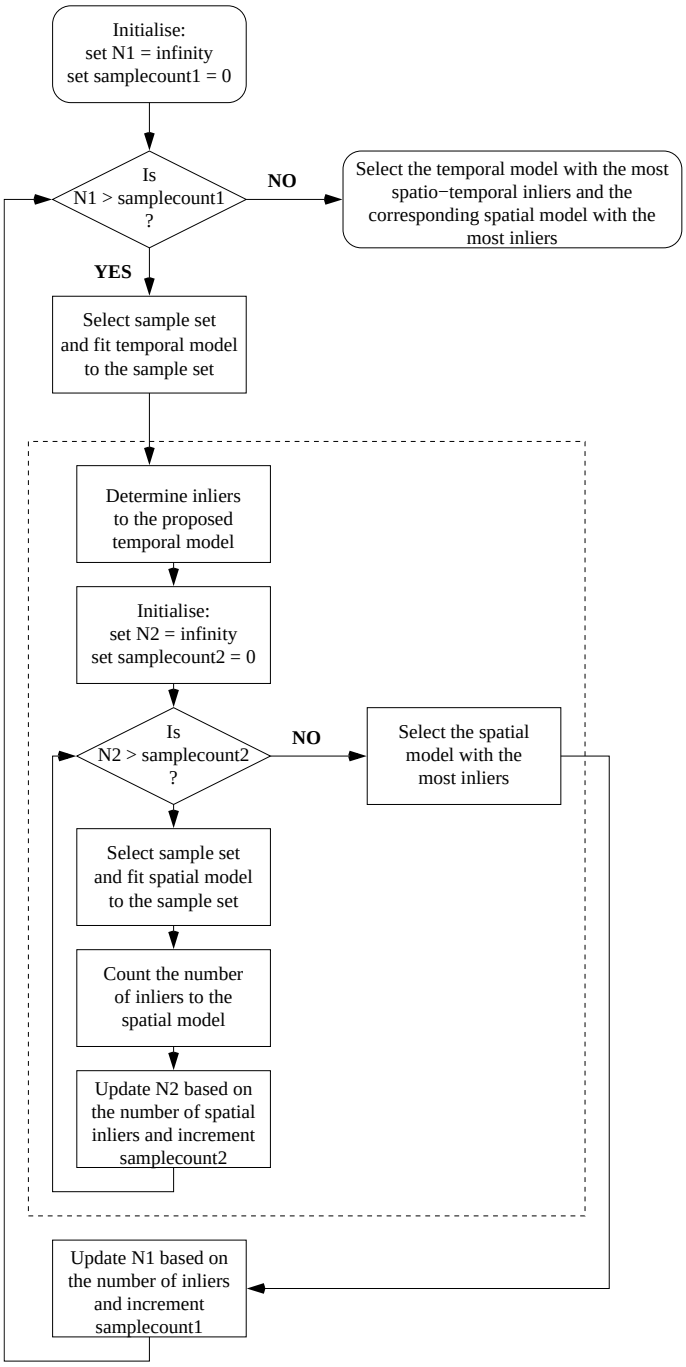


Figure 31: The nested RANSAC framework used by this algorithm. This varies from the framework displayed in Figure 30 in that the secondary RANSAC step contained within the dashed box replaces the step in which inliers are counted.

satisfy the proposed temporal model, rather than a line of best fit through all temporal component pairs. The temporal inliers are used as the input to the inner instance of RANSAC which estimates the spatial model in the next step.

3. Using the spatial components of the temporal inliers, a RANSAC step is used to fit the spatial model, either a homography fitted from 4 temporally inliers or a fundamental matrix from 8 temporal inliers if the linear method is used [13]. The class of spatial model is dependent on the type of motion contained within the video sequences. In this step, RANSAC may propose many spatial models for the proposed temporal model; the model resulting in the most inliers is selected. As the spatial model is fitted to the temporal inliers, the inliers to the recovered spatial model are *spatio-temporal inliers*. It is noted that not all temporal inliers are spatio-temporal inliers, as demonstrated in Figures 32(c), (d) and (e).

4. Once the inner RANSAC step has recovered the best spatial model corresponding to the temporal model proposed in this iteration, the number of spatio-temporal inliers is used as the support for the proposed temporal model. Then, the estimated number of iterations required to recover the temporal model is then updated adaptively, based on the probability of finding an outlier-free sample and the maximum support for all proposed temporal models [13]. If insufficient iterations have been completed, the algorithm then repeats from Step 1. Otherwise, the temporal model and the corresponding spatial model with the most spatio-temporal inliers are returned.

It may not appear to be necessary to compute the spatial model. However, enforcing the spatio-temporal inliers to satisfy both the temporal and spatial models provides another step for rejecting outliers. Sometimes, as shown in Figure 32(b), there may be many temporal outliers. If the spatial model was not considered in this algorithm, the synchronization process would be reduced to using RANSAC to fit a straight line to the temporal component pairs, with the line encapsulating the synchronization parameters. It should be emphasized that this process would fit a straight line to the set of inliers as determined by RANSAC, rather than computing a line of best fit through all temporal inliers. If no spatial model was recovered, it is possible that RANSAC may find an incorrect solution where the recovered straight line representing the temporal model has more inliers than the

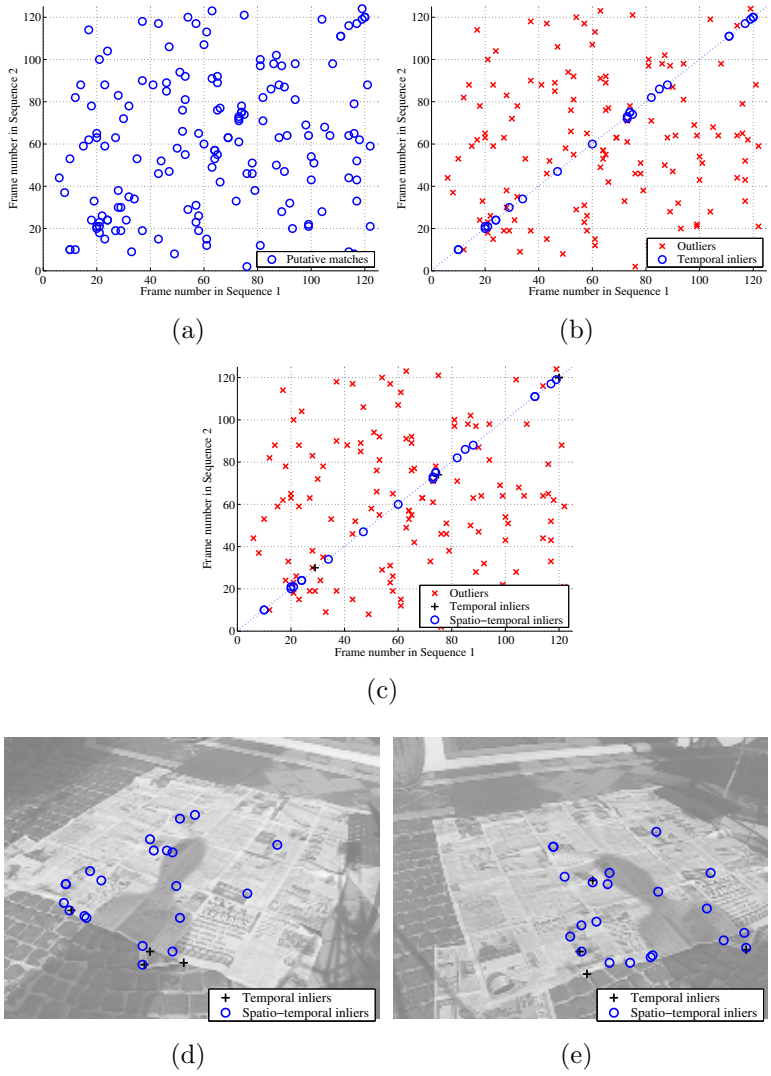


Figure 32: The process of determining spatio-temporal inliers from putative matches. (a) Firstly, temporal component pairs are constructed for all putative matches. Some putative matches occur in the same frames in each sequence; due to display limitations, such matches are represented by a single data point. (b) A straight line is fitted to two randomly selected temporal component pairs, and the temporal inliers determined. Here, the correct temporal model has been fitted. (c) The temporal inliers are used to fit a spatial model, from which spatio-temporal inliers, a subset of the temporal inliers, are determined. The spatial components of these temporal inliers are overlaid on a frame taken from each sequence and displayed in (d) and (e).

line representing the actual synchronization parameters; however, it is unlikely that the spatial components of all of the temporal inliers would form a consistent spatial model.

5.5.2 A special case: recovering only the temporal model

In some applications, the spatial model relating the two video sequences may be known and so it would not be necessary to recover the spatial model via the above process. Here, I present a special case of the above algorithm for recovering only the temporal model. The known spatial model is used to improve the process of determining putative matches:

1. Firstly, the process of determining putative matches is modified to exploit the information provided by the spatial model. To compute the set of putative matches, P, the Euclidean distance between interest point descriptors is calculated for every pair of space-time interest points as described in Step 2 of Section 5.4. However, if for a given pair of interest points, $\mathbf{x}_i$ and $\mathbf{x}'_j$, the reprojection error due to the known homography or fundamental matrix is above a specified threshold distance, the pair of points $\mathbf{x}_i$ and $\mathbf{x}'_j$ is immediately rejected from being a potential putative match. The putative matches are then computed from the remaining pairs of space-time interest points via the remainder of the procedure as described previously in Section 5.4.

2. Next, the temporal component pairs in 2D space are constructed from the temporal components of the putative matches. Then, RANSAC is applied to recover the synchronization parameters from these temporal component pairs. At each iteration, two temporal component pairs are randomly selected and a straight line representing the synchronization parameters is fitted to these points. The inliers to the resulting proposed synchronization parameters are then determined. This is the same process used to determine a temporal model in Step 1 of the previous sub-section. As the spatial components of the putative matches satisfy the known spatial model and the recovered temporal model, the inliers to the proposed synchronization parameters are spatio-temporal inliers.

Since the putative matches must satisfy the known spatial model, the special case presented in this sub-section essentially reduces the synchronization problem

to the simple problem of fitting a straight line to the temporal components of the inlying putative matches.

5.6 Results

The algorithm was tested on pairs of real video sequences recorded by stationary cameras with fixed internal parameters. In the series of sequences denoted `shadow` in the following tables, the shadow of a moving person was projected onto a textured planar surface; a homography relating the two views and the temporal model were recovered. On the other hand, the `park` sequences contained free 3D motion, hence a fundamental matrix was recovered in place of a homography. The pair of sequences labelled `star` also contained free motion in 3D space where a person moved within the scene and performed star jumps in various locations. The motion in this sequence was closer to the camera in this sequence than in the `park` sequences.

The resolution of the input video sequences was 200×150 pixels. The original resolution of each sequence was 640×480 pixels, however as the detection of space-time interest points is expensive in terms of computation time and memory requirements, the sequences were resampled to 200×150 pixels using bicubic interpolation. The videos contained only greyscale pixel data. Each sequence was between 125 and 540 frames in length, and each video was recorded at 15 or 30 frames per second, hence the frame rate ratio for each sequence pair was known exactly.

5.6.1 Free parameters

My algorithm and the experiments I conducted involved setting a number of free parameters. These parameters are outlined and their values are given below:

- In each sequence, the 200 most significant space-time interest points were detected, i.e., the space-time locations of the 200 strongest local maxima of the measure H, given in Equation (10), were used.

- For all sequences, the temporal variance τ^2 was set to 2. The spatial variance differed between sequences. For the `shadow` sequences, the spatial variance was $\sigma^2 = 2$. For the `park` and `star` sequences, $\sigma^2 = 4$. The estimated spatio-temporal extents of features occurring in the recorded video sequences were used to set the above variances.
- In determining the inliers for the temporal model, a threshold distance of 1 frame was used by RANSAC. This threshold is low in the context of fitting a straight line to a set of 2D points, however, it is important that the temporal component pairs of inlying putative matches closely correspond to the proposed synchronization parameters.
- When computing SIFT and 3SIFT descriptors for space-time interest points, the centre of the affine invariant region was enforced to be within 3 pixels of the space-time interest point for which the SIFT or 3SIFT descriptor was computed, as stated in Section 5.3.2.
- In the special case of the algorithm, the reprojection error threshold used when applying the known spatial model to determine putative matches was 4 pixels.

In this section, I firstly present results for the general case of the algorithm, i.e., where both the temporal and spatial models are recovered, using each of the three interest point descriptors. A comparison is also presented where video sequences containing planar motion are synchronized firstly using the general case of the algorithm where the ground plane homography relating the sequences is unknown, and secondly where the homography is estimated manually and the special case of the algorithm is used to recover the temporal model.

5.6.2 Results using local jet descriptors

The results for using the multi-scale local jet descriptors to determine putatively matching space-time interest points are given in Table 12. A symmetric transfer error measure is provided for the `shadow` sequences listed in this and later results tables. This error, in units of pixels^2, was calculated by firstly estimating the

ground plane homography from 14 manually selected significant features visible in each sequence, and then computing the mean symmetric transfer error of inlying pairs of space-time interest points. Equation (14) describes the computation of the transfer error:

$$d_{\text{trans}} = \frac{1}{N}\sum_{i}^{N} d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i + \mathtt{H}\mathbf{x}_i)^2, \tag{14}$$

where $d(\mathbf{a}, \mathbf{b})$ denotes the distance between the points $\mathbf{a}$ and $\mathbf{b}$.

These experiments confirm that this algorithm provides results comparable to manually synchronizing video sequences. It is noted that the algorithm failed to synchronize the `park8` sequences, though this was not due to the sequences having different frame rates; the `shadow10` and `star` sequences both had a non-unity ratio of frame rates and were successfully synchronized.

Figure 33 shows synchronized frames taken from each view of pairs of sequences containing planar motion. For each sequence pair, the frame recorded by the first camera has been rectified using the recovered homography such that it appears to have been viewed by the second camera. This visual comparison also demonstrates that the recovered homography is accurate. In Figure 34, synchronized frames from pairs of sequences containing free motion are shown. The epipolar geometry has been recovered from the space-time features whose spatial components are illustrated in each view; epipolar lines corresponding to these points are overlaid in the other frame from each sequence. In one view of each sequence pair, the other camera is visible, and the recovered epipole is located close to the imaged location of the camera, indicating that the recovered epipolar geometry is satisfactory.

5.6.3 Results using SIFT descriptors

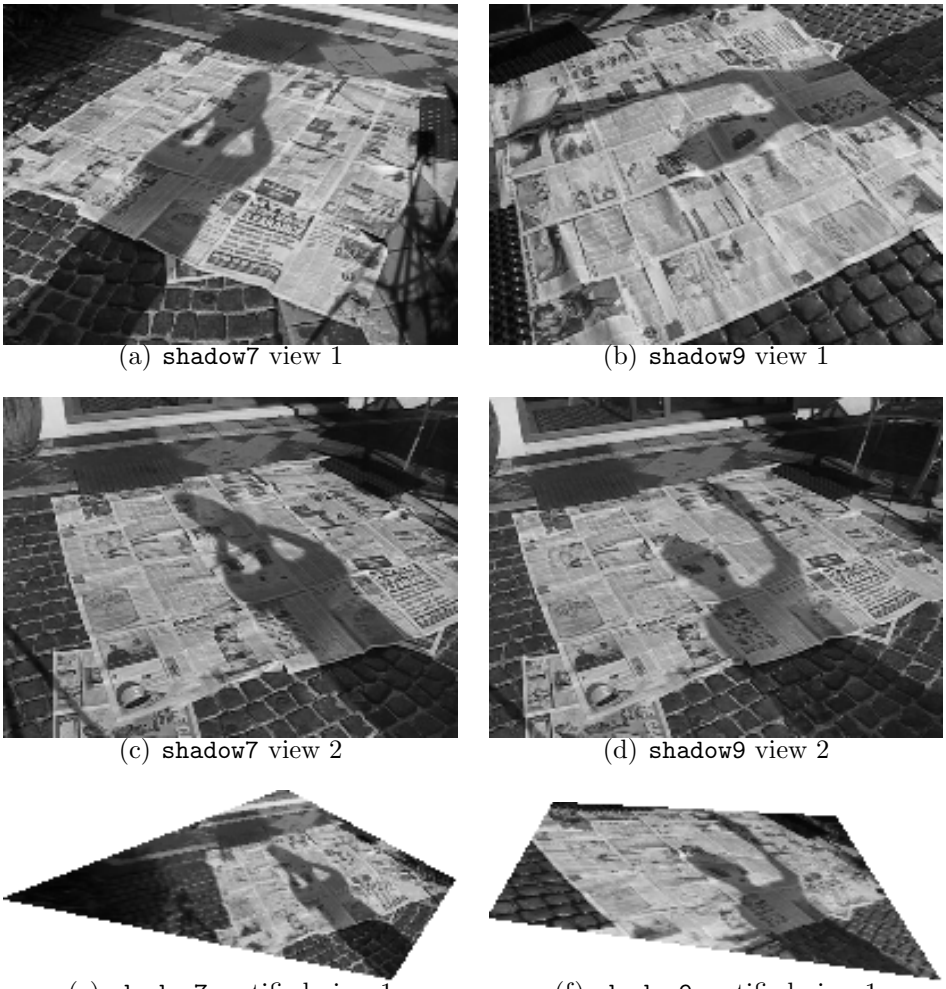
Table 13 lists the results of synchronizing the pairs of video sequences where the putative matches were proposed by matching SIFT descriptors representing each space-time interest point. The algorithm failed to synchronize any of the `park` video sequences when SIFT descriptors were used to represent space-time interest points; these results are not displayed in the table. A possible cause for this failure is the poor localisation of the SIFT descriptors, although this was not a problem in the

Sequence pair	$\bar{\alpha}$	$\widehat{\alpha}$	$\bar{\Delta}$	$\widehat{\Delta}$	Transfer error
shadow6	1	0.9973	0.0	0.7116	2.0780
shadow7	1	0.9978	0.0	0.0490	3.9459
shadow8	1	0.9961	0.0	0.4071	10.8315
shadow9	1	1.0004	0.0	0.3330	22.8190
shadow10	0.5	0.5004	94.0	93.5359	18.1046
park1	1	0.9999	−10.0	−9.8060	
park2	1	1.0021	0.0	−0.3130	
park3	1	0.9978	0.0	0.4293	
park4	1	1.0009	0.0	0.1401	
park5	1	1.0019	0.0	−0.3589	
park6	1	0.9987	0.0	−0.4379	
park7	1	1.0025	0.0	−0.4855	
park8	0.5	Failed	0.0	Failed	N/A
park9	1	1.0001	24.5	23.7990	
park10	1	1.0029	13.0	12.6256	
park11	1	0.9938	16.0	17.8868	
park12	1	0.9989	105.5	105.8763	
park13	1	1.0047	60.5	59.9576	
park14	1	0.9990	36.5	37.3511	
park15	1	1.0006	−29.5	−29.2951	
star	2	2.0026	0.0	0.1069	N/A

Table 12: The results of synchronizing the real video sequences where putative matches were computed from local jet descriptors. The "Transfer error" column denotes the symmetric transfer error of the spatial components of pairs of space-time interest points computed using a homography estimated from manually selected points; the error is in units of pixels^2.

`shadow` sequences. It was previously mentioned that a SIFT descriptor must be computed at a location within some threshold distance of the space-time interest point; Table 14 demonstrates that most space-time interest points satisfied this requirement. The failure may also have been due to the motion occurring in 3D space where the stationary background was much further away from the camera than the foreground motion. In this situation, corresponding space-time interest points from each sequence that were detected on the boundary of a foreground object may appear to have significantly different backgrounds, particularly when the cameras were widely separated. Hence, the descriptors of such points may vary considerably and as such, reliable matching may not be possible.

Removing Lowe's restriction on putative matches resulted in far more putative



(a) `shadow7` view 1 (b) `shadow9` view 1

(c) `shadow7` view 2 (d) `shadow9` view 2

(e) `shadow7` rectified view 1 (f) `shadow9` rectified view 1

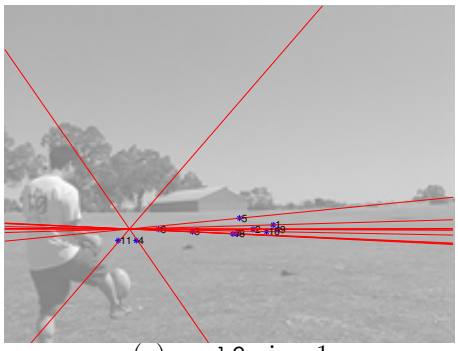
Figure 33: The result of synchronizing sequences containing planar motion. The pair of images (a) and (c) were recorded at the same instant in time, as were the images (b) and (d). The rectified views shown in (e) and (f) are the result of applying the recovered homography to the images in (a) and (b) such that those images appear to have been viewed from the same viewpoints as (c) and (d) respectively.

matches being declared but did not improve the synchronization process.

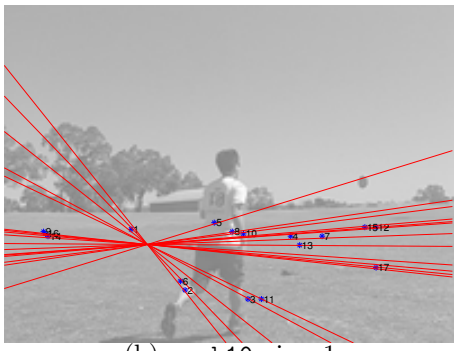
5.6.4 Results using 3SIFT descriptors

As with the single-frame SIFT descriptors, determining putative matches via matching 3SIFT descriptors was successful in the planar motion case but was unsuccessful in synchronizing the video sequences containing free motion in 3D space.

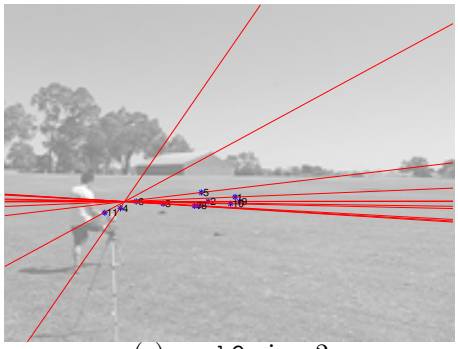
As the 3SIFT descriptors were computed for the same affine invariant regions at which the SIFT descriptors were computed, the number of 3SIFT descriptors



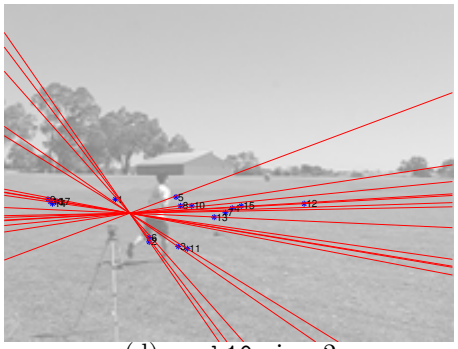
(a) `park9` view 1



(b) `park10` view 1



(c) `park9` view 2



(d) `park10` view 2

Figure 34: The result of synchronizing sequences containing free object motion. In each image, the spatial locations of inlying space-time interest points are displayed, and epipolar lines are overlaid corresponding to the spatial locations of inlying space-time interest points in the other sequence. In (c) and (d), the camera that captured frames (a) and (b) can be seen atop a tripod; the recovered epipole is close to the actual camera location. The pair of images (a) and (c) were recorded at the same instant in time, as were the images (b) and (d).

Sequence pair	$\bar{\alpha}$	$\widehat{\alpha}$	$\bar{\Delta}$	$\widehat{\Delta}$	Transfer error
`shadow6`	1	0.9984	0.0	0.6269	15.1876
`shadow7`	1	1.0026	0.0	−0.3454	4.0933
`shadow8`	1	1.0054	0.0	−0.1292	17.0871
`shadow9`	1	1.0010	0.0	0.5776	21.7699
`shadow10`	0.5	0.4983	94.0	94.3035	10.6653
`star`	2	2.0001	0.0	0.3189	N/A

Table 13: The results of synchronizing real video sequences where putative matches were computed from SIFT descriptors.

computed for each sequence was the same as the number of SIFT descriptors.

When comparing the number of outliers, temporal inliers, and spatio-temporal inliers for each sequence pair, it was noticed that there was no significant advantage in using the 3SIFT descriptor. As expected, where the frame rates of the sequences

Sequence pair	Number of interest points	SIFTs computed in Sequence 1	SIFTs computed in Sequence 2
shadow6	200	171	175
shadow7	200	179	168
shadow8	200	174	171
shadow9	200	170	167
shadow10	200	169	176
park1	200	145	174
park2	200	140	172
park3	200	130	174
park4	200	133	125
park5	200	160	183
park6	200	124	183
park7	200	124	166
park8	200	136	140
park9	200	169	145
park10	200	186	112
park11	200	140	119
park12	200	145	179
park13	200	160	163
park14	200	152	153
park15	200	166	172
star	200	200	196

Table 14: The number of space-time interest points computed for each sequence are shown along with the number of corresponding SIFT descriptors that were able to be computed for those interest points.

were the same, the 3SIFT descriptor performed no worse than the SIFT descriptors; unfortunately, the 3SIFT descriptor did not perform significantly better than the SIFT descriptor. Further, using the 3SIFT descriptor could cause problems in sequences where $\alpha \neq 1$ as the interval between frames used in computing each 3SIFT descriptor is constant; recall that in Section 5.3.3, for a space-time interest point in frame t_i of a sequence, the 3SIFT descriptor was computed from frames $\{t_i - 2, t_i, t_i + 2\}$ of the sequence. Ideally, frames $\{t_i - a, t_i, t_i + a\}$ would be used to compute the 3SIFT descriptor, where a is proportional to the frame rate of the sequence, but of course this is not possible if the frame rate ratio is unknown.

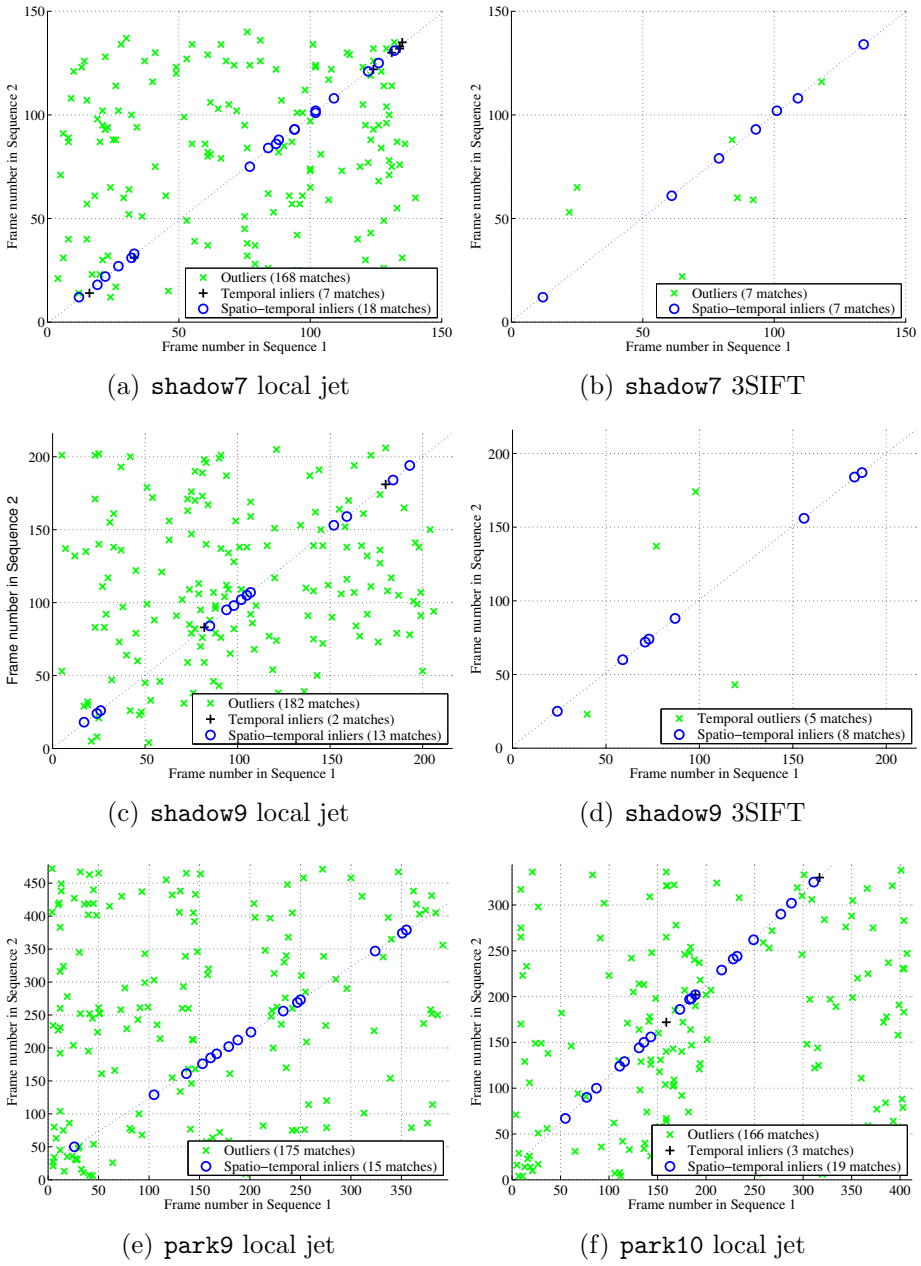
Sequence pair	$\bar{\alpha}$	$\widehat{\alpha}$	$\bar{\Delta}$	$\widehat{\Delta}$	Transfer error
`shadow6`	1	1.0033	0.0	−0.1437	10.8505
`shadow7`	1	1.0006	0.0	−0.3270	8.5310
`shadow8`	1	1.0009	0.0	0.1173	8.6287
`shadow9`	1	1.0016	0.0	−0.0291	10.3017
`shadow10`	0.5	0.5012	94.0	93.2718	7.9029
`star`	2	2.0001	0.0	0.3189	N/A

Table 15: The results of synchronizing the real video sequences where putative matches were computed from 3SIFT descriptors.

5.6.5 Summary of results for the general case

In Figure 35, the classification of temporal component pairs into outliers, temporal inliers, and spatio-temporal inliers are shown for the `shadow` and `park` sequences that are shown in Figures 33 and 34 respectively. Results for using both the local jet and 3SIFT descriptors are displayed for the `shadow` sequences; only the results of using the local jets are shown for the `park` sequences since these sequences were not successfully synchronized when using the 3SIFT descriptors. In the cases where the number of temporal inliers is not displayed, all temporal inliers are spatio-temporal inliers. From these figures, it can be seen that the percentage of inliers is relatively low. Normally, one would expect that it would not be possible to fit a model to a data set with such a low percentage of inliers. It is thought that because the general case of this algorithm employs two independent constraints to determine inliers, the percentage of inliers can be much lower than would normally be required to recover a model via RANSAC.

It is also noted that using the 3SIFT descriptors results in significantly fewer putative matches than using the multi-scale local jet descriptors. This is due to enforcing Lowe's recommendation that the ratio of the Euclidean distances between the SIFT descriptors of the best matching interest point and the second best match should be less than 0.8, as described in Section 5.4. Hence, this process results in only strong matches being declared as putative matches. Conversely, not applying this constraint when determining putative matches using the multi-scale local jet descriptors means that there may be many putative matches with poor distance scores.



(a) `shadow7` local jet (b) `shadow7` 3SIFT

(c) `shadow9` local jet (d) `shadow9` 3SIFT

(e) `park9` local jet (f) `park10` local jet

Figure 35: The classification of putative matches into outliers, temporal inliers, and spatio-temporal inliers for various pairs of video sequences. The putative matches were computed by comparing the specified descriptors for each interest point.

The results show that the general case of this algorithm is successful in accurately recovering the temporal model and either a homography induced by a plane or a fundamental matrix relating the two views. I expect that the localization of the epipoles as shown in Figures 34(c) and (d) would be improved if the fundamental matrix was estimated from features more evenly distributed spatially throughout

the frames.

It is noted that the symmetric transfer error of putatively matching space-time interest points is large relative to the resolution of the video sequences. Previously, Kovesi noted that the localisation of some features returned by the Harris corner detector may be affected by the Gaussian smoothing operation [18]. As the process of detecting space-time interest points is similar to that of the Harris corner detector [12], similar localisation issues may arise from the use of space-time interest points.

The time taken to synchronize the `shadow6` to `shadow8` sequences using a MATLAB implementation of this algorithm averaged between 4 and 5 minutes per pair of sequences on a 3GHz Pentium IV machine with 1GB of RAM. These sequences contained between 125 and 150 frames. The detection of the space-time interest points consumed approximately 94% of the computation time.

5.6.6 Results of synchronization where the spatial model is known

In Section 5.5.1, an algorithm was outlined for synchronizing sequences where neither the temporal nor the spatial models are known. Alternatively, in cases where the spatial model is known, the special case described in Section 5.5.2 can be used to exploit this information and improve the putative matching process. Experiments were conducted using each method to synchronize the `shadow` sequences containing planar motion. The homography relating the two views was estimated by manually selecting 14 significant features on the ground plane common to each view; the same homographies were used to compute the reprojection errors in the previous subsections.

In Table 16, it is shown that prior knowledge of the spatial model significantly improves the quality of the set of putative matches. As all putative matches satisfy the supplied spatial model and are spatial inliers, the classification of putative matches into inliers and outliers as displayed in the table is due to the temporal model. It is noted that the total number of putative matches (the sum of the inliers

Sequence name	Homography unknown		Homography known	
	Inliers	Outliers	Inliers	Outliers
`shadow6`	16	181	60	54
`shadow7`	19	174	73	58
`shadow8`	15	181	49	77
`shadow9`	12	185	39	56
`shadow10`	6	187	14	94

Table 16: The number of spatio-temporal inliers and outliers are shown for the case where the homography relating the `shadow` sequences was recovered by the algorithm, and where the homography was known. The multi-scale jet descriptors were used to determine putative matches.

and outliers) is lower in experiments where the homography is known. This can be attributed to the process of determining putative matches discarding pairs of space-time interest points whose spatial components do not satisfy the supplied homography. In the case where the spatial model is unknown, no pairs of interest points are discarded by the putative matching process.

Hence, if the spatial model is known prior to synchronization, it is advantageous to use it in the special case of the algorithm to improve the percentage of inlying putative matches that are passed to the RANSAC step used to recover the temporal model.

5.7 Discussion

5.7.1 The importance of RANSAC ordering

In the process of estimating the spatial and temporal models in the general case of the algorithm, the temporal model is proposed first, and the spatial model is then recovered from the set of temporal inliers. In sequences containing a significant number of outliers, this ordering is important. Since increasing the size of a sample set causes the probability of the sample set containing an outlier to increase, it is desirable to choose a smaller sample set. In this algorithm, this is achieved by firstly fitting a temporal model which can be proposed from only two putative matches, rather than a spatial model which requires at least four matches.

A further point of interest is how the inner model is affected when an outlier

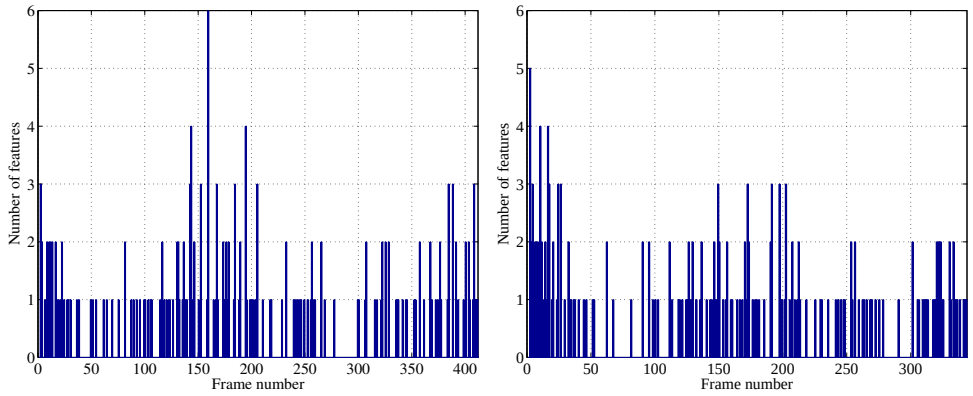
is contained in the sample used to propose the model in the outer instance of RANSAC. It is expected that if a temporal model is proposed from a pair of putative matches where at least one match is an outlier, there will not be many temporal inliers. In fact, there may be insufficient temporal inliers from which to propose a spatial model, in which case the temporal model is immediately discarded. If we were to firstly attempt to recover the spatial model, there are a number of possible temporal models that could be proposed from the spatially inlying points (as it is assumed that all putative matches used to propose the spatial model are *spatial inliers*). For example, if a homography was recovered first from a sample of four putative matches, there are at least four spatial inliers from which at least $\binom{4}{2} = 6$ temporal models can be proposed. Hence, it is likely that an incorrect spatial model would lead to many incorrect temporal models being proposed, which is clearly inefficient.

It is noted that the accuracy of the estimated spatial model is heavily dependent on the estimated temporal model at each RANSAC iteration. Whilst an inaccurate temporal model may produce many temporal inliers, these temporal inliers are unlikely to yield a consistent spatial model, hence the number of spatio-temporal inliers is expected to be low. This is not a problem, as due to the nature of RANSAC, it is expected that there are sufficient correct putative matches such that the correct temporal and spatial models will be recovered in at least one iteration.

5.7.2 Comparison with existing algorithms

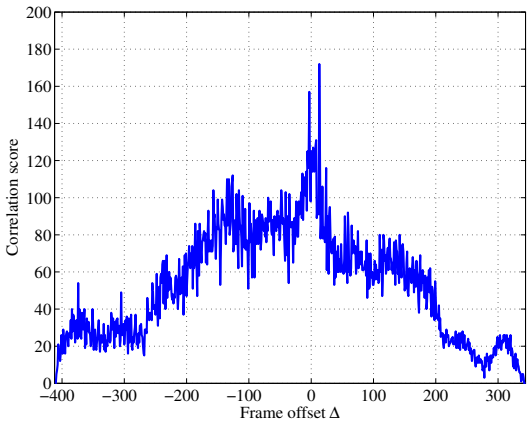
An algorithm by Yan and Pollefeys [53] that also uses space-time interest points assumes a known frame rate ratio and recovers only the frame offset of two sequences to integer accuracy. For each sequence, a histogram of the number of detected space-time interest points in each frame of the sequence is constructed. Then, at each integer frame offset, the cross-correlation score of these histograms is computed; at the actual frame offset, the distribution of interest points in each sequence should be similar, resulting in a high correlation score.

An example of using Yan and Pollefeys' algorithm to synchronize the `park10` pair of sequences is given in Figure 36. It can be seen that even though the distributions



(a) Distribution of space-time interest points in Sequence 1

(b) Distribution of space-time interest points in Sequence 2



(c) Correlation score over the range of possible integer frame offsets

Figure 36: An example of recovering the frame offset of two sequences using Yan and Pollefeys' algorithm. In (a) and (b), the temporal distributions of the detected space-time interest points are shown for the `park10` sequences. The result of cross-correlating these distributions is given in (c). The maximum correlation score occurs at $\Delta = 13$ frames, which corresponds to the actual frame offset. It is noted that a second local maximum is located close to this global maximum.

of interest points do not appear to have many unique and significant features, e.g., peaks and valleys, Yan and Pollefeys' algorithm successfully recovers the frame offset. It is noted that an incorrect frame offset near the actual frame offset also has a high correlation score. This is a potential source of error and is addressed by Yan and Pollefeys in their paper [53].

Although Yan and Pollefeys' algorithm is shown to synchronize sequences based only on the distribution of space-time interest points, their algorithm will fail in the case where a similar motion generating the same number of space-time interest

points is regularly repeated at different locations, e.g., a person walking and turning regularly but in different directions and at different angles each time. Since the distribution of space-time interest points will be periodic and because there is no attempt to determine corresponding space-time interest points from each sequence, this type of motion will appear to be identical to periodic motion to Yan and Pollefeys' algorithm. By fitting a spatial model to the putatively matched space-time interest points, my algorithm is able to differentiate between this motion and periodic motion.

A further case that may cause their algorithm to fail is where view-dependent events occur, e.g., occlusions, where interest points will be generated at different time instants of the two video sequences, or in one sequence only. This may significantly affect the correlation score, and hence the recovered frame offset. My algorithm is more robust in that such events are unlikely to be declared as putative matches, and further, because these events are unlikely to occur at the same time and place in both sequences, they will not satisfy the temporal and spatial models and will hence be discarded.

A recent periodic motion detection algorithm by Laptev et al. [21] shares some similarities with my algorithm. In Laptev's algorithm, periodically equivalent features are detected from a single video sequence. These points can be either space-time interest points as described in this algorithm, or spatial feature points detected in regions of a frame that contain non-constant motion in surrounding frames of the video sequence. Laptev et al. assume that the periodic motion is performed at the same rate throughout the sequence, hence only the period length is recovered. In the context of synchronizing a pair of video sequences as my algorithm does, this is equivalent to recovering only the frame offset of the two sequences and not the frame rate ratio. Laptev et al. also recover a spatial model which is a reduced fundamental matrix that can be recovered from only two pairs of periodically equivalent points.

The differences in the algorithms are that my algorithm operates by matching space-time interest points between two sequences, rather than finding periodically equivalent interest points from the same sequence. It can be seen in Figures 33(b) and (d) that the sequences synchronized by my algorithm can have significantly

different viewpoints which makes the process of finding matching interest points more challenging. Also, my algorithm can recover a standard fundamental matrix that encapsulates the epipolar geometry relating the two sequences rather than a reduced matrix that only satisfies periodically equivalent points. Further, as Laptev et al. search for periodic motion within one video sequence, it is assumed that the motion occurs at the same rate within the sequence; my algorithm is able to recover the ratio of the frame rates of two video sequences.

A possible case where my algorithm may fail includes sequences containing repeated or periodic motions which often generate many similar space-time interest points in both sequences. Consequently, the number of outliers would be excessive and incorrect temporal and spatial models may be returned by the RANSAC process. This problem is common to all video synchronization algorithms.

5.8 Conclusion

It has been shown that this algorithm successfully synchronizes pairs of video sequences by matching space-time interest points from each sequence and using RANSAC to recover the synchronization parameters from these points. The general case of the algorithm consists of an extension to well-known techniques used to recover a homography or fundamental matrix relating a pair of still images from a set of putatively matching interest points. This algorithm is unique as it operates without requiring object tracking or knowledge of the epipolar geometry or the homography relating the cameras. However, in applications such as video surveillance where the spatial model may be known, it has been shown to be advantageous to use the spatial model to improve the quality of the putative matches.

The accuracy of the recovered frame offset and frame rate ratio are shown to be comparable with manual synchronization. In each experiment, the recovered homography or fundamental matrix relating the two sequences was confirmed to be accurate via visual methods and in the case of the sequences containing planar motion, numerical analysis. It was shown that using local jets for representing space-time interest points is successful in most cases. However, although SIFT descriptors

have been successfully used for many applications, they were not suitable for use in this algorithm for synchronizing scenes containing non-planar motion.

Chapter 6

Conclusions and future work

This thesis has demonstrated that the synchronization problem can be approached in a variety of manners; in an algebraic approach as described in Chapter 3, or using geometric approaches as outlined in Chapters 4 and 5. My three algorithms are shown to draw on some aspects of some existing algorithms, e.g., the measure of synchronization used in Chapter 3. However, each of my algorithms introduces a new approach to the synchronization problem such as the iterative approach given in Chapter 4, or applies techniques from other domains to the synchronization problem, as in the algorithm in Chapter 5 that extends a well-known technique from the image domain and demonstrates its application to synchronizing video sequences.

The results of using the three algorithms to synchronize a variety of real video sequences have demonstrated that the algorithms can accurately synchronize a range of video sequences subject to the specified constraints of each algorithm. The results of synchronizing synthetic data sets in Chapters 3 and 4 demonstrate that those algorithms are accurate and are not significantly affected by the presence of noise.

The major contributions presented in this thesis are summarized below:

- It is demonstrated by the algorithms presented in Chapters 3 and 4 that only a single object is required to be tracked in order to recover both the frame rate ratio and frame offset of two video sequences. Further, the algorithm given in Chapter 3 does not require the cameras' epipolar geometry to be known. No

other feature-based algorithm is known to use the motion of a single object to recover both α and Δ relating a pair of sequences recorded by cameras with unknown epipolar geometry.

- The coarse-to-fine approach outlined in Chapter 3 involved a coarse synchronization step that recovered a good approximation of the frame rate ratio from only sub-sequence length information and the direction of vertical motion of a single object in each frame. Thus, it is demonstrated that this minimal information is sufficient to recover an estimate of the frame rate ratio of a pair of sequences.

- In many algorithms used for synchronizing two video sequences, a number of point correspondences from each sequence are detected and either the reprojection error of these points or the 9th singular value of a measurement matrix constructed from the point locations is used as a measure of synchronization. It was shown that an advantage of using the 9th singular value in place of the reprojection error is that it is cheaper computationally. More importantly, it provides greater consistency in the error measure since the reprojection error relies on accurate estimation of the epipolar geometry; mis-synchronization causes the epipoles to be mislocated, which directly affects the reprojection error and hence the synchronization algorithm.

- Video sequences recorded by stationary cameras with fixed intrinsic parameters can be synchronized without first requiring weak camera calibration as demonstrated by the algorithms presented in Chapters 3 and 5.

- It was shown that for sequences of a projectile moving with significant vertical motion recorded by weakly calibrated cameras, a rapidly converging iterative algorithm, as described in Chapter 4, can be applied to locate corresponding frames from each video sequence by exploiting the known epipolar geometry and the object's motion.

- In Chapter 5, a standard method used to recover a homography or fundamental matrix from still images was extended to the space-time domain, allowing

the synchronization parameters to also be recovered. In this manner, the synchronization problem was recast in the framework of a well-known technique.

- It was shown that video sequences can be synchronized by finding matching *events* represented by space-time interest points in video sequences, rather than by matching object trajectories. Previous algorithms utilizing space-time interest points either did not attempt to match interest points between sequences [53] or did not apply the technique to the synchronization problem [21].

6.1 Suggestions for future work

All of the algorithms presented in this dissertation are applicable to synchronizing pairs of video sequences. A logical step would be to extend the techniques presented in this thesis to synchronize more than two video sequences. The trivial method of doing this would be to synchronize the sequences pairwise, designating one sequence as a reference sequence. Alternatively, it may be possible to synchronize all possible pairs of video sequences, i.e., recovering $\binom{N}{2}$ sets of synchronization parameters for a set of N sequences. Then, methods of minimizing the errors in the set of recovered synchronization parameters could be investigated. Another approach could investigate whether it is possible to adapt these approaches to synchronize 3 sequences by employing 3-view geometric methods, e.g., using the trifocal tensor in place of the fundamental matrix.

The algorithm given in Chapter 3 relies on dividing the video sequences into sub-sequences based on the vertical motion of the tracked object in the scene. In order to generalize the algorithm to handle video sequences that do not contain significant vertical motions, it would be desirable to develop a more general method of determining sub-sequences. It may be possible to use events such as those detected by space-time interest points to define the first and last frames of sub-sequences.

A further extension could incorporate the ideas presented in this thesis into an algorithm capable of synchronizing video sequences recorded by moving cameras whose intrinsic parameters may change throughout the sequences. In order to

achieve this, it may be necessary to determine a set of stationary features (i.e., lines and points) from which camera calibration could be performed for each frame and the effects of camera motion minimized, before attempting to synchronize the sequences.

With the expected increase in the quantity of recorded video sequences in the future and the increase in computing power available to process this vast amount of video data, it is envisaged that video sequence synchronization will play an important role in the future of computer vision applications.

Bibliography

[1] R. L. Carceroni, F. L. C. Pádua, G. A. M. R. Santos, and K. N. Kutulakos. Linear sequence-to-sequence alignment. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1:746–753, 2004.

[2] Y. Caspi and M. Irani. Alignment of non-overlapping sequences. *International Journal of Computer Vision*, 48(1):39–52, Jun 2002.

[3] Y. Caspi and M. Irani. Spatio-temporal alignment of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1409–1424, November 2002.

[4] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. *International Journal of Computer Vision*, 68(1):53–64, 2006.

[5] R. Cipolla, K. E. Åström, and P. J. Giblin. Motion from the frontier of curved surfaces. In *Proceedings of the International Conference on Computer Vision*, pages 269–275, 1995.

[6] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[7] C. Dai, Y. Zheng, and X. Li. Accurate video alignment using phase correlation. *IEEE Signal Processing Letters*, pages 737–740, 2006.

[8] C. Dai, Y. Zheng, and X. Li. Subframe video synchronization via 3D phase correlation. In *Proceedings of the International Conference on Image Processing*, pages 501–504, 2006.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, Jun 1981.

[10] M. A. Giese and T. Poggio. Synthesis and recognition of biological motion patterns based on linear superposition of prototypical motion sequences. In *Proceedings of the IEEE Workshop on Multi-View Modeling and Analysis of Visual Scene*, pages 73–80, 1999.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 3rd edition, 1996.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[14] T. Kanade, P. J. Narayanan, and P. W. Rander. Virtualized reality: Concepts and early results. In *Proceedings of IEEE Workshop on Representation of Visual Scenes*, pages 69–76, 1995.

[15] I. Kitahara, H. Saito, S. Akimichi, T. Onno, Y. Ohta, and T. Kanade. Large-scale virtualized reality. In *Computer Vision and Pattern Recognition Technical Sketches*, 2001.

[16] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367–375, 1987.

[17] T. Korah and C. Rasmussen. Aligning sequences from multiple cameras. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2005.

[18] P. Kovesi. Phase congruency detects corners and edges. In *Proceedings of Digital Image Processing: Techniques and Applications*, pages 309–318, 2003.

[19] S. Kuthirummal, C. V. Jawahar, and P. J. Narayanan. Video frame alignment in multiple views. In *Proceedings of the International Conference on Image Processing*, 2002.

[20] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005.

[21] I. Laptev, S. J. Belongie, P. Pérez, and J. Wills. Periodic motion detection and segmentation via approximate sequence alignment. In *Proceedings of the International Conference on Computer Vision*, 2005.

[22] I. Laptev and T. Lindeberg. Space-time interest points. In *Proceedings of the International Conference on Computer Vision*, pages 1:432–439, 2003.

[23] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. In *ECCV Workshop on Spatial Coherence for Visual Motion Analysis*, 2004.

[24] L. Lee, R. Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):758–767, August 2000.

[25] C. Lei and Y.-H. Yang. Tri-focal tensor-based multiple video synchronization with subframe optimization. *IEEE Transactions on Image Processing*, 15(9):2473–2480, September 2006.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[27] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[28] M. Noguchi and T. Kato. Geometric and timing calibration for unsynchronized cameras using trajectories of a moving marker. In *IEEE Workshop on Applications of Computer Vision*, 2007.

[29] W. Nunziati, J. Alon, S. Sclaroff, and A. Del Bimbo. View registration using interesting segments of planar trajectories. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 75–80, 2005.

[30] 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. In *Proceedings of the International Conference on Image Processing*, 2003.

[31] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *Proceedings of the International Conference on Computer Vision*, 2003.

[32] I. Reid and A. Zisserman. Goal-directed video metrology. In *Proceedings of the European Conference on Computer Vision, LNCS 1065*, pages 647–658. Springer, 1996.

[33] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978.

[34] E. Shechtman and M. Irani. Space-time behavior based correlation. In *Proceedings of Computer Vision and Pattern Recognition*, 2005.

[35] S. N. Sinha and M. Pollefeys. Synchronization and calibration of camera networks from silhouettes. In *Proceedings of the International Conference on Pattern Recognition*, 2004.

[36] S. N. Sinha and M. Pollefeys. Visual-hull reconstruction from uncalibrated and unsynchronized video streams. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 349–356, 2004.

[37] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[38] L. Spencer and M. Shah. Temporal synchronization from camera motion. In *Proceedings of the Asian Conference on Computer Vision*, 2004.

[39] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *Proceedings of Computer Vision and Pattern Recognition*, pages 521–527, 1999.

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

[41] P. Tresadern and I. Reid. Uncalibrated and unsynchronized human motion capture: A stereo factorization approach. In *Proceedings of Computer Vision and Pattern Recognition*, pages 128–134, 2004.

[42] P. A. Tresadern and I. Reid. Synchronizing image sequences of non-rigid objects. In *Proceedings of the British Machine Vision Conference*, 2003.

[43] T. Tuytelaars and L. Van Gool. Synchronizing video sequences. In *Proceedings of Computer Vision and Pattern Recognition*, pages 762–768, 2004.

[44] Y. Ukrainitz and M. Irani. Aligning sequences and actions by maximizing space-time correlations. In *Proceedings of the European Conference on Computer Vision*, pages 538–550, 2006.

[45] M. Ushizaki, T. Okatani, and K. Deguchi. Video synchronization based on co-occurrence of appearance changes in video sequences. In *Proceedings of the International Conference on Pattern Recognition*, pages 71–74, 2006.

[46] S. Velipasalar and W. Wolf. Frame-level temporal calibration of video sequences from unsynchronized cameras by using projective invariants. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 462–467, 2005.

[47] D. Wedge, D. Huynh, and P. Kovesi. Motion guided video sequence synchronization. *Lecture Notes in Computer Science — ACCV 2006*, 3852:832–841, 2006.

[48] D. Wedge, D. Huynh, and P. Kovesi. Using space-time interest points for video sequence synchronization. In *IAPR Conference on Machine Vision Applications*, pages 190–194, 2007.

[49] D. Wedge, P. Kovesi, and D. Huynh. Trajectory based video sequence synchronization. In *Proceedings of Digital Image Computing: Techniques and Applications*, pages 79–86, 2005.

[50] A. Whitehead, R. Laganiere, and P. Bose. Temporal synchronization of video sequences in theory and in practice. In *Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05)*, pages 132–137, 2005.

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

[52] L. Wolf and A. Zomet. Wide baseline matching between unsynchronized video sequences. *International Journal on Computer Vision*, 68(1):43–52, 2006.

[53] J. Yan and M. Pollefeys. Video synchronization via space-time interest point distribution. In *Proceedings of Advanced Concepts for Intelligent Vision Systems*, 2004.

[54] C. Zhou and H. Tao. Dynamic depth recovery from unsynchronized video streams. In *Proceedings of Computer Vision and Pattern Recognition*, pages 351–358, 2003.