THE UNIVERSITY OF ADELAIDE

The Automated Synchronisation of Independently Moving Cameras

Daniel William Pooley

School of Computer Science
The University of Adelaide

May 2008

TABLE OF CONTENTS

Abstract

Computer vision is concerned with the recovery of useful scene or camera information from a set of images. One classical problem is the estimation of the 3D scene structure depicted in multiple photographs. Such estimation fundamentally requires determining how the cameras are related in space. For a dynamic event recorded by multiple video cameras, finding the temporal relationship between cameras has a similar importance. Estimating such synchrony is key to a further analysis of the dynamic scene components. Existing approaches to synchronisation involve using visual cues common to both videos, and consider a discrete uniform range of synchronisation hypotheses. These prior methods exploit known constraints which hold in the presence of synchrony, from which both a temporal relationship, and an unchanging spatial relationship between the cameras can be recovered.

This thesis presents methods that synchronise a pair of independently moving cameras. The spatial configuration of cameras is assumed to be known, and a cost function is developed to measure the quality of synchrony even for accuracies within a fraction of a frame. A Histogram method is developed which changes the approach from a consideration of multiple synchronisation hypotheses, to searching for seemingly synchronous frame pairs independently. Such a strategy has increased efficiency in the case of unknown frame rates. Further savings can be achieved by reducing the sampling rate of the search, by only testing for synchrony across a small subset of frames. Two robust algorithms are devised, using Bayesian inference to adaptively seek the sampling rate that minimises total execution time. These algorithms have a general underlying premise, and should be applicable to a wider class of robust estimation problems. A method is also devised to robustly synchronise two moving cameras when their spatial relationship is unknown. It is assumed that the motion of each camera has been estimated independently, so that these motion estimates are unregistered. The algorithm recovers both a synchronisation estimate, and a 3D transformation that spatially registers the two cameras.

Declaration

Daniel Pooley
October 9, 2008

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 *International Conference on Image Processing, Barcelona, Spain, September 15-19, 2003*, volume 1, pages 413–416. IEEE Computer Society Press.

D. Pooley, M. Brooks and A. van den Hengel. RATSAC: an adaptive method for accelerated robust estimation, and its application to video synchronisation. In *Digital Image Computing: Techniques and Applications, Adelaide, Australia, December 3-5, 2007*. IEEE Computer Society Press.

Acknowledgements

I would chiefly like to thank my supervisors Professor Michael Brooks and Associate Professor Anton van den Hengel, for their support, guidance and motivation throughout my candidature. In addition, thanks go to Professor Wojciech Chojnacki for his assistance with a variety of mathematical queries, as well as to Dr. Darren Gawley, Dr. Chunhua Shen, Dr. Thorsten Thormählen, Dr. John Bastian, and Mr. Rhys Hill for productive theoretical discussions. I am also appreciative of the support provided by the Cooperative Research Centre for Sensor Signal and Information Processing. Finally, special thanks go to my family and friends, in recognition of their encouragement, moral support, and for providing welcome distractions when most needed.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF EXAMPLES

LIST OF THEOREMS

NOTATION

k	A scalar (non-bold lower case).
K	A random variable (non-bold upper case).
X	An event (non-bold sans-serif upper case).
$\boldsymbol{S}$	A set (bold upper case).
$\boldsymbol{v}$	A vector (bold lower case).
$\mathbf{\mathsf{A}}$	A matrix (bold sans-serif upper case).
$\mathbf{\mathsf{A}}^\top$	The transpose of matrix $\mathbf{\mathsf{A}}$.
$\mathbf{\mathsf{A}}^{-1}$	The inverse of matrix $\mathbf{\mathsf{A}}$.
$[\boldsymbol{v}]_i$	The element i of vector $\boldsymbol{v}$.
$[\mathbf{\mathsf{A}}]_{i,j}$	The element (i, j) of matrix $\mathbf{\mathsf{A}}$.
$[\mathbf{\mathsf{A}}]_{i,*}$	The row i of matrix $\mathbf{\mathsf{A}}$.
$[\mathbf{\mathsf{A}}]_{*,j}$	The column j of matrix $\mathbf{\mathsf{A}}$.
$Tr(\mathbf{\mathsf{A}})$	The trace of matrix $\mathbf{\mathsf{A}}$.
$Rank(\mathbf{\mathsf{A}})$	The rank of matrix $\mathbf{\mathsf{A}}$.
$\\|\mathbf{\mathsf{A}}\\|_{\mathrm{F}}$	The Frobenius norm of matrix $\mathbf{\mathsf{A}}$.
$P(\mathsf{X})$	The probability of event X.
$\sim$	Equality up to scale of two vectors or matrices.
$\approx$	Approximate equality up to scale of two vectors or matrices.

Chapter 1

INTRODUCTION

Computer vision concerns the recovery of useful properties of a scene from images or video sequences, such as the classical problem of reconstructing a 3D model. Consider two cameras with known 3D locations, orientations, and internal properties, both viewing a common stationary scene. The 3D location of a scene point can be determined if its corresponding 2D locations in a pair of images recorded by the cameras have been identified. Much attention has been given in recent years to the analysis of video sequences depicting *dynamic* scenes. These scenes may contain multiple independently moving objects, or other effects such as illumination which change over time. Analysis of a dynamic scene viewed by multiple cameras requires knowing how the resulting video sequences should be synchronised.

A set of video sequences is considered synchronised if their relationship in time is known. Such synchronisation is readily available if every frame in each video sequence has been accurately time-stamped by a common timekeeping device. Given the time-stamp of a frame in one video sequence, the frames with the closest time-stamps in the remaining video sequences can be located.

A simple example of where synchronisation can be applied is trajectory reconstruction. Consider video sequences recorded by cameras viewing a moving scene point. Given a pair of frames with the same time-stamp, the 3D location of the moving point can be determined by the same means used for stationary scene points. If this is repeated for many time-stamps, the path taken by the moving point through the scene is recovered.

1.1 Synchronisation

The task of determining synchrony is equivalent to determining a consistent set of time-stamps for video sequences depicting the same dynamic scene. For simplicity, it is assumed that time-stamps are represented with a single number, expressing the elapsed interval since some point in time. One example of this is the Unix time-stamp, where time is expressed as the number of milliseconds since 00:00:00 (UTC) on 1-1-1970.

If a video camera records at a fixed frame rate, the time difference between successive frames remains constant. The function that assigns a time-stamp to each frame of the resulting video sequence is therefore linear in the indices of the frames. Consequently,

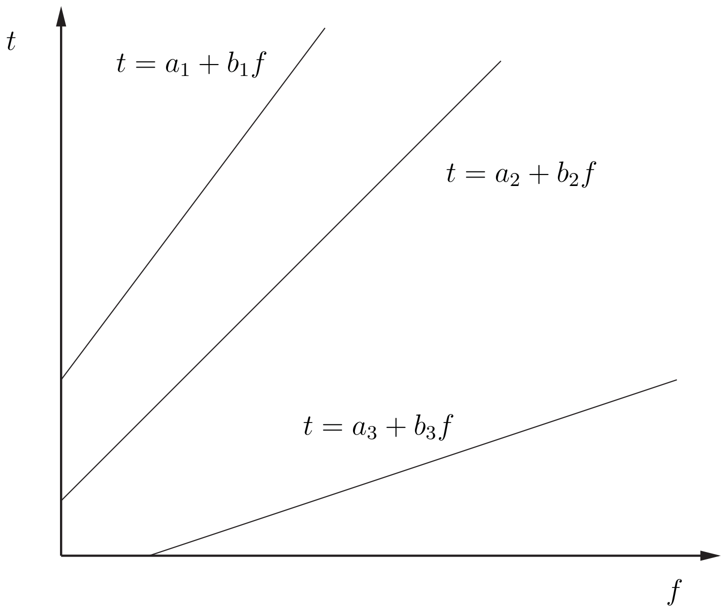


Figure 1.1: Time-stamping 3 video sequences

the time-stamping of n video sequences can be described by n linear equations, where the equation for video sequence i is given by

$$t = a_i + b_i f, \quad \forall i \in [1 \dots n].$$

Here, f is a frame index from video sequence i, and t is the resulting time-stamp for frame f. The constant a_i is the point in time that the camera started recording, and is therefore the time-stamp for frame 0, while b_i is the elapsed time between frames, or the inverse of the camera's frame rate. The time-stamping of 3 video sequences is illustrated in figure 1.1.

A set of video sequences can be synchronised during recording with the use of specialised hardware, requiring a physical connection. A system designed for this purpose is described in [35], in which cameras are given a global synchronisation signal to capture images simultaneously. A time code is then associated with the set of resulting frames.

This thesis is concerned with the problem of automatically synchronising video sequences without specialised hardware, by analysing and processing the visual content of the recorded frames. Typically, a 3D scene does not contain a visual cue that indicates precise time. Thus, when using only visual information to synchronise video sequences, the resulting time-stamps will not have a known correlation with real-world time. Both the units of the time-stamps, effectively the real-world time between integer values of t, and the real-world time corresponding to $t = 0$ will be unknown. For typical applications of synchronisation, such as trajectory reconstruction, this does not matter. It makes little difference how the time-stamps correlate with real-world time, as any choice still admits a consistent set of time-stamps for the video frames. The correlation between time-stamps and real-world time can be arbitrarily chosen. If n video sequences are to be synchronised, a convenient choice is to choose $t = 0$ to correspond with the real-world time that camera n started recording. Similarly, t can be chosen to measure time in units that equal the real-world time between camera n recording successive frames. This defines

$$a_n = 0.0, \quad b_n = 1.0.$$

Now the time-stamps for video sequence n will be equal to its frame indices. In effect, camera n is now being used not only to record visual information, but also as a time-keeping device by which the remaining video sequences will be time-stamped. Since the alignment between time and frame indices for video sequence n is now specified, time-stamps need only be found for the remaining video sequences. Synchronising n video sequences, without establishing a correspondence with real-world time, therefore only requires the estimation of $n - 1$ linear relations.

In the specific case of a pair of video sequences, synchronisation may be expressed in terms of just a single linear equation. If the frame indices from the first and second video sequences are denoted f and f' respectively, then

$$t = a_1 + b_1 f,$$
$$t = a_2 + b_2 f'.$$

By combining these equations to eliminate t, the relationship between frame indices is

$$f' = (a_1 - a_2)b_2^{-1} + b_1 b_2^{-1} f.$$

By a simplification of these terms,

$$a = (a_1 - a_2)b_2^{-1},$$
$$b = b_1 b_2^{-1},$$

The linear relation describing synchronisation for a pair of video sequences, which describes a line of synchrony in the space of (f, f'), is given by

$$f' = a + bf. \tag{1.1}$$

Given (a, b), equation 1.1 defines frame pairs (f, f') which have the same time-stamp, and are therefore assumed to be synchronous. Note a faster derivation can be obtained by choosing $a_2 = 0$ and $b_2 = 1$ as described above, in which case (a, b) are simply given by (a_1, b_1). The parameter a is the *frame offset*, specifying which frame the second camera captured as the first camera started recording; and b is the *frame rate ratio* of the two cameras. Note that a can positive, negative, or zero. Conversely, b is assumed to be positive, since a negative b value indicates that the frames in one of the video sequences are in reverse order.

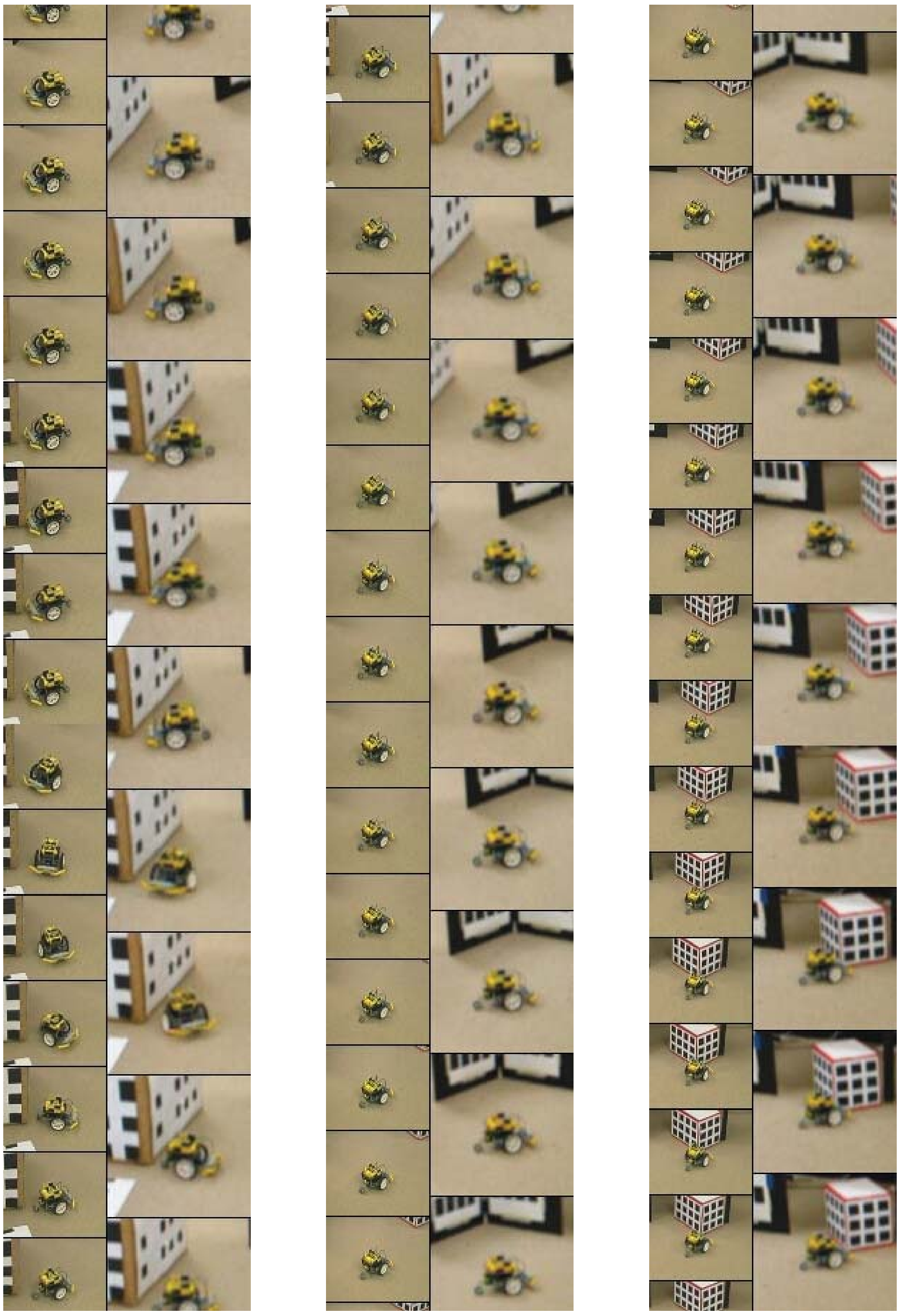


Figure 1.2: Side-by-side display of synchronised video frames, cropped to a region of interest

Synchronisation is illustrated visually in figure 1.2. Cropped frame regions from video sequences with different frame rates are displayed side by side such that their vertical positions correlate with their estimated time-stamps.

1.2 Outline

Chapter 2 provides an overview of existing synchronisation methods, and explains the required underlying camera geometry and estimation algorithms. A description is also given as to how the approach of this thesis differs from previous methods. The fundamental component for achieving the accurate synchronisation of independently moving cameras is presented in chapter 3. A cost function is developed which uses corresponding trajectories of moving scene points, and the performance of synchronisation via an exhaustive search across the parameter space is assessed. In chapter 4, a method is introduced which searches for potentially synchronous frame pairs, and uses histogram methods to achieve an initial estimate of synchronisation with greater efficiency. These histogram methods are employed to achieve robust synchronisation in chapter 5, with new robust algorithms that seek a trade-off between efficiency and reliability. Chapter 6 investigates the possibility of using moving scene points to achieve both synchronisation, and combine scene reconstructions from two cameras. Finally, a summary of contributions and scope for further improvements is provided in chapter 7.

Chapter 2

PREVIOUS WORK

As shown in section 1.1, the temporal relationship between two video sequences recorded by cameras with constant frame rates can be described by a single line with parameters (a, b). The estimation of these parameters via an analysis of the video frames has been previously examined in the literature, under a variety of different constraints and using different visual cues. This chapter will describe a number of existing methods, and provide explanations as to how the corresponding visual cues can offer a means for achieving synchronisation.

All existing methods have one common requirement. The visual cue used to synchronise the video sequences must indicate some change over time, and this change must be observable to both cameras. Clearly, two video sequences exhibiting no common change over time cannot be synchronised. A suitable visual cue will provide a means of determining whether or not a pair of frames appear to be synchronous.

Many current approaches make use of well-known spatial constraints that relate corresponding locations in synchronous frames. An overview of these constraints and their relevance to automated synchronisation is given in section 2.1. A description of existing synchronisation algorithms, and the circumstances in which they can be applied, is then presented in section 2.2

2.1 Spatial Constraints

The formation of an image recorded by a camera is typically modelled by perspective projection, which mathematically represents the use of a pinhole camera. A detailed description is given in [14]. The advantage of this model is its elegant simplicity, in which the projection is described by a linear relationship.

A scene point can be represented by a 4×1 vector. If this vector is termed $\boldsymbol{q}$, the Euclidean 3D coordinates of the point are given by $(\frac{[\boldsymbol{q}]_1}{[\boldsymbol{q}]_4}, \frac{[\boldsymbol{q}]_2}{[\boldsymbol{q}]_4}, \frac{[\boldsymbol{q}]_3}{[\boldsymbol{q}]_4})$. Note this representation is *scale-invariant*, as any two such vectors equal up to scale will represent the same 3D scene point. A pinhole camera projects this scene point to a 2D point in an image, where

$$\boldsymbol{p} = \mathsf{M}\boldsymbol{q}$$

describes the resulting image point location. The vector $\boldsymbol{p}$ has size 3×1, and specifies 2D Euclidean coordinates given by $(\frac{[\boldsymbol{p}]_1}{[\boldsymbol{p}]_3}, \frac{[\boldsymbol{p}]_2}{[\boldsymbol{p}]_3})$. The 3×4 projection matrix M encapsulates

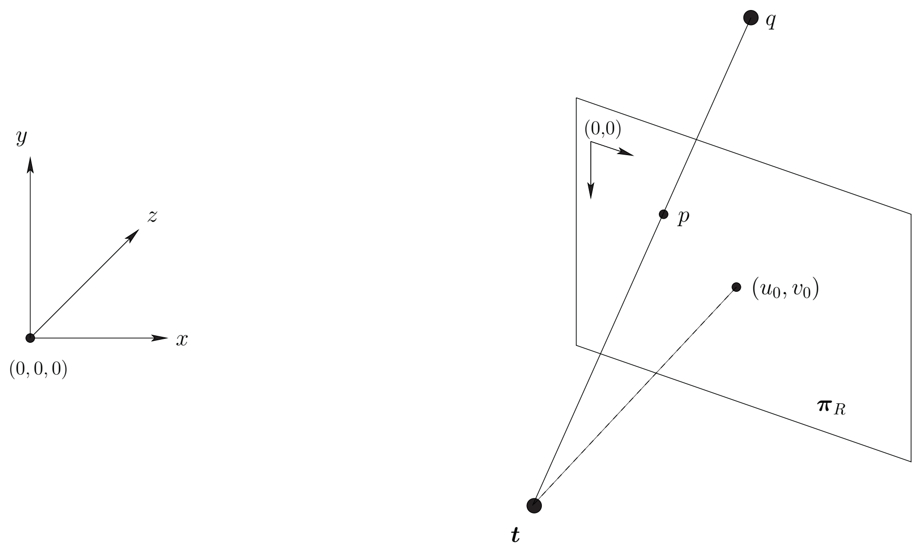


Figure 2.1: The perspective projection of a scene point

the properties of the camera, and can be expressed as

$$\mathsf{M} \sim \mathsf{K}\left[\mathsf{R} \,|\, -\mathsf{R}\boldsymbol{t}\right],$$

where $\boldsymbol{t}$ is the 3D location of the camera's optical centre and R is a 3×3 rotation matrix specifying the orientation of the camera with respect to the coordinate system of the 3D scene. The calibration matrix K is an upper-triangular 3×3 matrix specifying the internal camera settings, or intrinsic parameters.

The process of perspective projection is illustrated in figure 2.1, which displays both the 3D and 2D coordinate systems. A ray in space joining the scene point $\boldsymbol{q}$ with the camera's optical centre is shown. The resulting 2D location $\boldsymbol{p}$ is the image of $\boldsymbol{q}$, and is determined by the point at which this ray intersects the retinal plane $\boldsymbol{\pi}_R$. The ray in space passing through the optical centre and perpendicular to the retinal plane is called the optical axis, and the distance between the optical centre and the retinal plane is defined as the focal length. Increasing or decreasing the focal length corresponds to the camera zooming in or out respectively.

Geometrically, the matrix K describes a transformation relating the coordinate system of the image to the coordinate system of the retinal plane, and is given as

$$\mathsf{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The location (u_0, v_0) is called the principal point, and specifies in image coordinates where the optical axis intersects the retinal plane. The magnitudes of the scalars f_u and f_v are ratios of the camera's focal length with the width and height of the pixels respectively. A negative value for f_u or f_v denotes a reflection of the horizontal or vertical axis for the image coordinate space. Alternatively, they can both be assumed to be positive, though the reflection of an image axis must then be included in the rotation matrix R. The skew parameter s determines how 'rectangular' the pixel locations are on the retinal plane. An ideal camera should have a skew of 0, and $|f_u| = |f_v|$, as otherwise the image is a non-Euclidean distortion of what was projected onto the retinal plane.

2.1.1 Essential and Fundamental Matrices

The epipolar constraint is a simple matrix equation that projections of a scene point into two cameras must satisfy. It also describes some aspects of the spatial geometry relating these cameras.

In the case of known intrinsic parameters, image points can be expressed in the coordinate system of the retinal plane. The epipolar constraint can now be derived algebraically as follows. The projection of a scene point $\boldsymbol{q}$ into two cameras is given by

$$\begin{aligned}\boldsymbol{p} &\sim \mathsf{R}\,[\mathsf{I}\,|-\boldsymbol{t}]\,\boldsymbol{q},\\ \boldsymbol{p}' &\sim \mathsf{R}'\,[\mathsf{I}\,|-\boldsymbol{t}']\,\boldsymbol{q},\end{aligned}$$

where $\sim$ denotes equality up to scale for a pair of matrices or vectors. Note that both R and R' are rotation matrices, so $\mathsf{R}^\top = \mathsf{R}^{-1}$, and similarly for R'. The right hand sides can therefore be simplified by correcting for the rotations, giving

$$\begin{aligned}\mathsf{R}^\top\boldsymbol{p} &\sim [\mathsf{I}\,|-\boldsymbol{t}]\,\boldsymbol{q},\\ \mathsf{R}'^\top\boldsymbol{p}' &\sim [\mathsf{I}\,|-\boldsymbol{t}']\,\boldsymbol{q}.\end{aligned}$$

The optical centre of the second camera can be expressed in terms of the relative translation between the cameras. If $\emptyset$ denotes the zero matrix, then the second projection can be re-expressed as

$$\begin{aligned}\mathsf{R}'^\top\boldsymbol{p}' &\sim [\mathsf{I}\,|\boldsymbol{t}_\Delta-\boldsymbol{t}]\,\boldsymbol{q} \quad \text{where } \boldsymbol{t}_\Delta = \boldsymbol{t}-\boldsymbol{t}',\\ &\sim [\mathsf{I}\,|-\boldsymbol{t}]\,\boldsymbol{q} + [\emptyset\,|\boldsymbol{t}_\Delta]\,\boldsymbol{q}.\end{aligned}$$

Note that the first term is a scalar multiple of the rotation-adjusted first projection, and that the second can be simplified due to the presence of the zero matrix, leading to

$$\mathsf{R}'^\top\boldsymbol{p}' \sim k\mathsf{R}^\top\boldsymbol{p} + [\boldsymbol{q}]_4\boldsymbol{t}_\Delta$$

for some non-zero scalar k. Both sides of this equation can be pre-multiplied by a matrix $[\boldsymbol{t}_\Delta]_\times$, which is an anti-symmetric matrix such that $[\boldsymbol{t}_\Delta]_\times\boldsymbol{u}$ equates to the cross product $\boldsymbol{t}_\Delta \times \boldsymbol{u}$. This leads to

$$[\boldsymbol{t}_\Delta]_\times\mathsf{R}'^\top\boldsymbol{p}' \sim k[\boldsymbol{t}_\Delta]_\times\mathsf{R}^\top\boldsymbol{p} + [\boldsymbol{q}]_4[\boldsymbol{t}_\Delta]_\times\boldsymbol{t}_\Delta.$$

The rightmost term disappears, since $\boldsymbol{t}_\Delta \times \boldsymbol{t}_\Delta = \mathbf{0}$. Pre-multiplying by $\boldsymbol{p}^\top \mathsf{R}$ gives

$$\boldsymbol{p}^\top \mathsf{R}[\boldsymbol{t}_\Delta]_\times \mathsf{R}'^\top \boldsymbol{p}' \sim \boldsymbol{p}^\top \mathsf{R}[\boldsymbol{t}_\Delta]_\times \mathsf{R}^\top \boldsymbol{p}.$$

Since $[\boldsymbol{t}_\Delta]_\times$ is an anti-symmetric matrix, the right hand side equates to 0, leaving the constraint

$$\boldsymbol{p}^\top \mathsf{E} \boldsymbol{p}' = 0, \quad \text{where } \mathsf{E} = \mathsf{R}[\boldsymbol{t}_\Delta]_\times \mathsf{R}'^\top.$$

This expression for the epipolar constraint was first introduced by Longuet-Higgins [31], along with the linear algebra eight-point algorithm for estimating E from eight corresponding image point pairs. Note that the scale of E can not be recovered from these point matches, since E and $k\mathsf{E}$ will satisfy the same epipolar constraints for any non-zero scalar k. A method is also given for recovering the relative rotation and translation between the cameras, and subsequently the scene point locations. The matrix E, called the *essential matrix*, has 5 degrees of freedom; 3 for rotation, and 2 to describe the direction (but not the magnitude) of relative translation between cameras. Since only the direction of the relative translation is available, the distance between the cameras is unknown. As this distance is undefined, it reflects the fact that the scale of a scene can not be determined from image correspondences alone. Additionally, the world coordinate system is also not defined by the essential matrix. A recovered reconstruction of scene features is therefore only unique up to a translation, rotation and scale. It is demonstrated in [27] that it is both necessary and sufficient for an essential matrix to be rank 2 with a pair of equal non-zero singular values.

Similarly to the calibrated case, an analogous constraint arises when the intrinsic parameters are unknown. If the relative translation and rotation between two cameras are encapsulated by an essential matrix E, and the cameras have calibration matrices K and K', then the projections of a scene point $\boldsymbol{p}$ and $\boldsymbol{p}'$ satisfy

$$\boldsymbol{p}^\top \mathsf{F} \boldsymbol{p}' = 0, \quad \text{where } \mathsf{F} \sim \mathsf{K}^{-\top} \mathsf{E} \mathsf{K}'^{-1}.$$

The matrix F is called the *fundamental matrix*, and was introduced by Faugeras [16] and Hartley et. al. [26]. It is noted by both that projection matrices cannot be recovered uniquely, since a projection matrix pair $(\mathsf{M}, \mathsf{M}')$ yield the same epipolar geometry as $(\mathsf{MH}, \mathsf{M}'\mathsf{H})$, for any full rank 4×4 matrix H. Algorithms are given in each paper for recovering projection matrices and scene points up to this ambiguity, using a set of image correspondences. If certain assumptions about the calibration matrix hold, the 4×4 ambiguity can be partly resolved. Hartley demonstrates that a single focal length parameter for each projection matrix can be recovered from F if the other intrinsic parameters are known [22]. An alternative method is given by Newsam et. al. [36], along with a complete classification of degenerate camera configurations. It is shown by Faugeras et. al. [17] that if the intrinsic parameters of a moving camera remain constant, they can all be estimated from at least two unique fundamental matrices relating pairs of

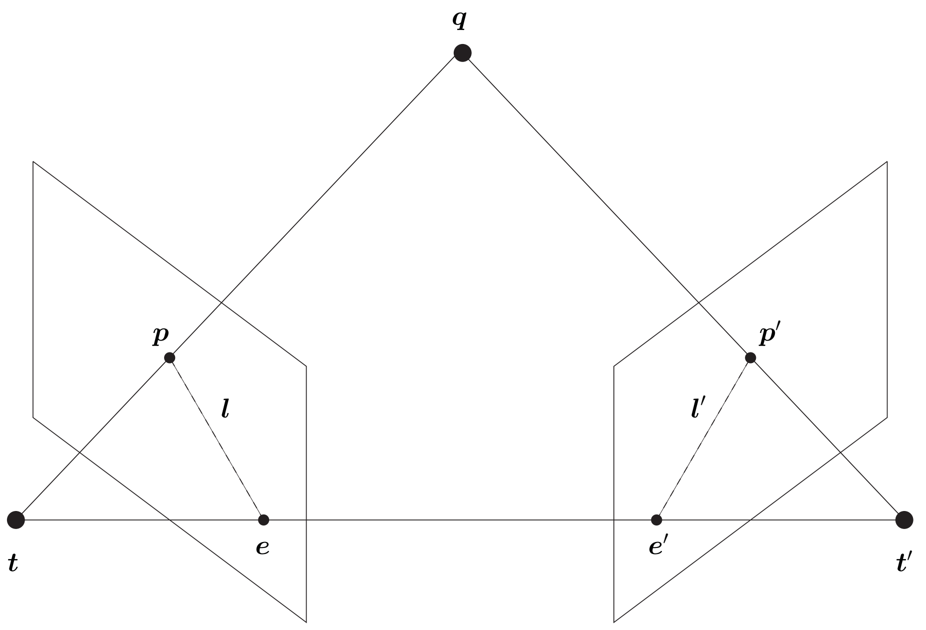


Figure 2.2: Visualisation of the epipolar constraint

images. Once the intrinsic parameters are known, as in the essential matrix case, a scene reconstruction can be chosen which is unique up to translation, rotation, and scale.

The geometry corresponding to the epipolar constraint is illustrated in figure 2.2. An image point $\boldsymbol{p}'$ in the second (right) camera is mapped to a corresponding *epipolar line* $\boldsymbol{l}$ in the image of the first (left) camera, given by $\boldsymbol{l} = \mathsf{F}\boldsymbol{p}'$. The algebraic form of the epipolar constraint therefore specifies that the image point $\boldsymbol{p}$ must lie along this line. Additionally, all epipolar lines in the first camera must pass through the epipole $\boldsymbol{e}$, which specifies the projection of the second camera's optical centre. Similarly, the point $\boldsymbol{p}$ in the first camera has a corresponding epipolar line $\boldsymbol{l}'$, given by $\mathsf{F}^\top\boldsymbol{p}$, which passes through both $\boldsymbol{p}'$ and $\boldsymbol{e}'$.

Estimation of the fundamental matrix from corresponding image points is a crucial step in the reconstruction of a scene from two uncalibrated cameras, and hence has been extensively studied by the computer vision community. In [25], Hartley adapts the 8-point algorithm to the fundamental matrix case, by adding a prior step in which the feature points in each image are transformed so that their centroid is at the origin, and their average distance from the centroid is $\sqrt{2}$. This often improves the quality of F, estimated by minimising the sum of squared algebraic errors, given by

$$\sum_{i=1}^{m}(\boldsymbol{p}_i^\top \mathsf{F}\boldsymbol{p}_i')^2, \text{ where } \|\mathsf{F}\|_{\mathrm{F}} = 1. \tag{2.1}$$

The operator $\|.\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix, and m is the total number of image correspondences. As in the essential matrix case, this minimisation can be solved by linear algebra methods. This is possible since each summand in equation 2.1 is the square of a term which is linear in the unknown elements of F. The normalising transformations typically remove an order of magnitude difference in the scales of the elements of the image point vectors. It is noted in [25] that such a normalisation will reduce the condition number of the quadratic form associated with equation 2.1. An alternative justification for the improvement in quality is given by Chojnacki et. al [8], in which image point locations are considered to be instances of random variables. The normalisation and estimation can then be equated to minimising a cost function in which each squared residual is scaled by the inverse of its variance. A fundamental matrix estimate then has a higher associated cost if it indicates the residuals have a lower variance.

Further clarification comes from considering equation 2.1 for image points that have not been normalised. Suppose that, for the true fundamental matrix, the sum of squared algebraic errors is slightly more than m. On average, each squared sum is slightly more than 1. A 3×3 matrix F with $[\mathsf{F}]_{3,3} = 1$, and 0 elsewhere, unconditionally yields a sum of m. Such a matrix is degenerate, and specifies *all* epipolar lines to be the line at infinity. In a sense, this could be considered the *worst possible* solution, yet it admits a lower algebraic error than the truth. In contrast, a prior normalisation typically downscales the coordinates, and also the error in image point measurements, such that each squared summand is much smaller than 1 for the true F. Accordingly, their sum will be much smaller than m.

An estimation of the fundamental matrix with linear algebra methods is advantageous due to its speed, yet the cost function given in equation 2.1 has no meaningful geometric interpretation. An initial estimate of the fundamental matrix minimising equation 2.1 can typically be refined by using iterative methods to minimise a cost function consisting of the squared sum of non-linear expressions.

Given a fundamental matrix F and a pair of points from the left and right images $(\boldsymbol{p}, \boldsymbol{p}')$, F maps the point $\boldsymbol{p}'$ to its corresponding epipolar line in the frame of point $\boldsymbol{p}$. The orthogonal distance from $\boldsymbol{p}$ to the epipolar line is given by

$$\mathcal{E}_{\mathrm{ORTH}}(\boldsymbol{p}, \boldsymbol{p}', \mathsf{F}) = \frac{|\boldsymbol{p}^\top \mathsf{F} \boldsymbol{p}'|}{\sqrt{[\mathsf{F}\boldsymbol{p}']_1^2 + [\mathsf{F}\boldsymbol{p}']_2^2}}.$$

Similarly, the distance from point $\boldsymbol{p}'$ to its corresponding epipolar line is computed as $\mathcal{E}_{\mathrm{ORTH}}(\boldsymbol{p}', \boldsymbol{p}, \mathsf{F}^\top)$. The *symmetric epipolar error* is defined as the square root of the sum of the squares of these two distances:

$$\mathcal{E}_{\mathrm{SYMM}}(\boldsymbol{p}, \boldsymbol{p}', \mathsf{F}) = \sqrt{\mathcal{E}_{\mathrm{ORTH}}(\boldsymbol{p}, \boldsymbol{p}', \mathsf{F})^2 + \mathcal{E}_{\mathrm{ORTH}}(\boldsymbol{p}', \boldsymbol{p}, \mathsf{F}^\top)^2}.$$

One of the earliest fundamental matrix papers by Faugeras [17] suggests improving a fundamental matrix estimate by minimising the sum of squared symmetric epipolar errors.

Hartley and Zisserman propose the so-called *Gold Standard* method for improving an estimate of the fundamental matrix [21]. This equates to finding F by minimising the cost function

$$\sum_{i=1}^{m} d(\boldsymbol{p}_i, \hat{\boldsymbol{p}}_i)^2 + d(\boldsymbol{p}'_i, \hat{\boldsymbol{p}}'_i)^2,$$
$$\text{where } \hat{\boldsymbol{p}}_i^\top \mathsf{F} \hat{\boldsymbol{p}}'_i = 0,\ \forall i,\ Rank(\mathsf{F}) = 2.$$

The image points $\boldsymbol{p}_i$ and $\boldsymbol{p}'_i$ represent the measured image feature locations. The image point pair $\hat{\boldsymbol{p}}_i$ and $\hat{\boldsymbol{p}}'_i$ are the assumed true point locations, and are included as parameters in the minimisation. The function $d(.)$ measures the 2D Euclidean distance between two points. The minimum of this function equates to a maximum likelihood solution, under the assumption that every image point is perturbed with uniform Gaussian noise. Due to the constraints on this cost function, the unknowns are often parameterised by a pair of projection matrices (in which the first is prescribed and fixed) and a set of scene points.

Another distance that measures conformity to the epipolar constraint is the *Sampson* error, which provides a first-order approximation to reprojection error [21]. Sampson error is defined as

$$\mathcal{E}_{\text{SAMP}}(\boldsymbol{p}, \boldsymbol{p}', \mathsf{F}) = \frac{|\boldsymbol{p}^\top \mathsf{F} \boldsymbol{p}'|}{\sqrt{[\mathsf{F}\boldsymbol{p}']_1^2 + [\mathsf{F}\boldsymbol{p}']_2^2 + [\mathsf{F}^\top \boldsymbol{p}]_1^2 + [\mathsf{F}^\top \boldsymbol{p}]_2^2}}. \tag{2.2}$$

This expression for Sampson error in the fundamental matrix case is suggested by Luong and Faugeras [33], and is experimentally compared to symmetric epipolar error.

The epipolar constraint has particular relevance to automated synchronisation since it also applies to *moving* scene points. Consider such a moving point, projecting to locations $\boldsymbol{p}_i$ in frame i of the first camera, and $\boldsymbol{p}'_j$ in frame j of the second camera. Given frame offset a, and frame rate ratio b, these frames are synchronous if $j = a + bi$. In this case, these image points are projections of the same scene location. The epipolar constraint therefore specifies that

$$\boldsymbol{p}_i^\top \mathsf{F} \boldsymbol{p}'_j = 0, \quad \text{if } j = a + bi,$$

Conversely, in asynchronous frames the image features are generated by the projections of different scene locations, and in general will not satisfy the epipolar constraint, as illustrated in figure 2.3. By extension, just as measures of epipolar error can be used to assess the quality of a fundamental matrix, epipolar errors of moving points provide a means of assessing an estimate of synchronisation parameters.

2.1.2 Homography Matrices

A 3×3 homography matrix is a transformation that relates two projections of a planar scene. Representing the plane by a 4×1 vector $\boldsymbol{\pi}$, a scene point $\boldsymbol{q}$ lies on this plane if and only if $\boldsymbol{\pi}^\top \boldsymbol{q} = 0$. The space of scene points satisfying this equation can be represented by

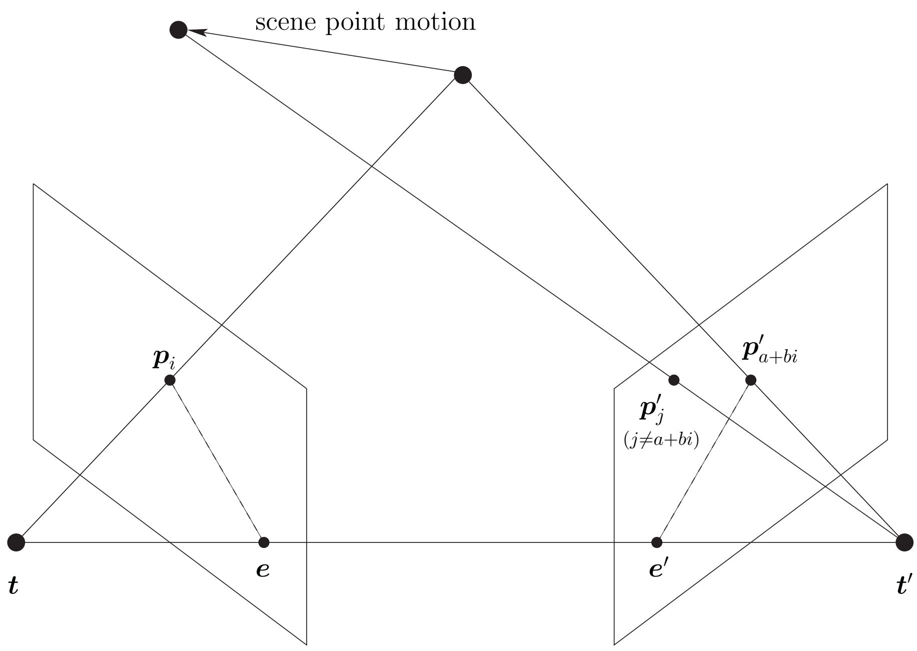


Figure 2.3: Visualisation of the epipolar constraint for projections of a moving scene point in synchronous and asynchronous frame pairs

a 4×3 matrix $\mathbf{N}$. The columns of $\mathbf{N}$ span the left null-space of $\boldsymbol{\pi}$, which can be considered a 4×1 matrix. Specifically, $\mathbf{N}^\top \boldsymbol{\pi} = \mathbf{0}$, where $\mathbf{0}$ is the zero-vector. Note that $\mathbf{N}$ must have rank 3, since $\boldsymbol{\pi}^\top \boldsymbol{q} = 0$ only imposes *one* linear constraint on the 4×1 vector $\boldsymbol{q}$. *Any* scene point $\boldsymbol{q}$ lying on the plane can be expressed as a linear combination of the columns of $\mathbf{N}$. For such a $\boldsymbol{q}$, there exists a 3×1 vector $\boldsymbol{k}$ such that $\boldsymbol{q} = \mathbf{N}\boldsymbol{k}$. The projection of $\boldsymbol{q}$ into two cameras represented by projection matrices $\mathbf{M}$ and $\mathbf{M}'$ is therefore given by

$$\begin{aligned} \boldsymbol{p} &\sim \mathbf{M}\mathbf{N}\boldsymbol{k}, \\ \boldsymbol{p}' &\sim \mathbf{M}'\mathbf{N}\boldsymbol{k}. \end{aligned}$$

Note that since $\mathbf{M}$,$\mathbf{M}'$ and $\mathbf{N}$ are all rank 3, both $\mathbf{MN}$ and $\mathbf{M}'\mathbf{N}$ are full-rank 3×3 matrices. Consequently, the vector $\boldsymbol{k}$, which defines the scene point on the plane, can be eliminated from these equations, leading to

$$\begin{aligned} &\boldsymbol{p}' \sim \mathbf{H}\boldsymbol{p}, \\ &\text{where } \mathbf{H} = \mathbf{M}'\mathbf{N}(\mathbf{MN})^{-1}. \end{aligned} \tag{2.3}$$

The homography matrix, denoted here as $\mathbf{H}$, relates corresponding image locations exactly. If $\mathbf{H}$ is known, a feature location in one image can be mapped to it's precise corresponding location in the other image, as shown by the relation in equation 2.3. Note that $\mathbf{H}$ is defined only by the projection matrices and the null-space of $\boldsymbol{\pi}$. Consequently, this relation holds only for the projections of scene points lying on the plane.

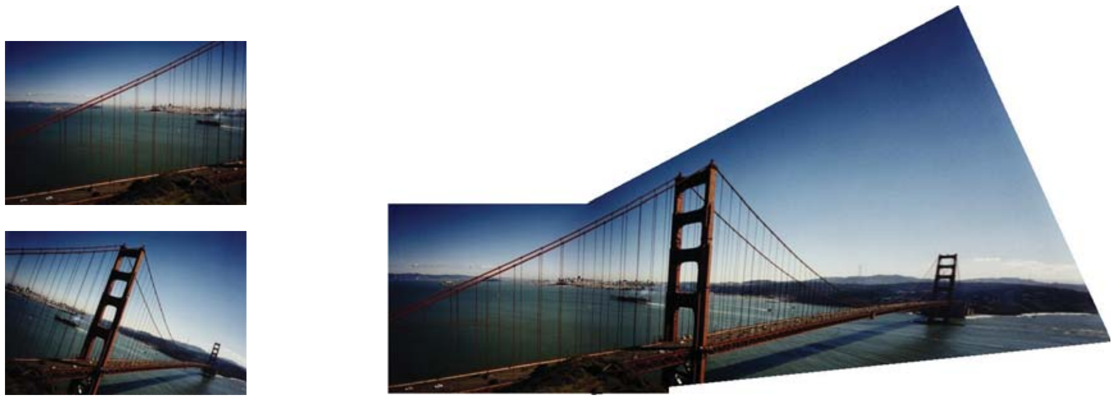


Figure 2.4: An application of the homography matrix; original images and the resulting mosaic

Two images of a non-planar scene can also be related by a homography matrix, providing the optical centres of the two projection matrices are coincident. In this situation, the two projections of a scene point $\boldsymbol{q}$ are given by

$$\begin{aligned} \boldsymbol{p} &\sim \mathbf{KR}\,[\mathbf{I}| - \boldsymbol{t}]\,\boldsymbol{q}, \\ \boldsymbol{p}' &\sim \mathbf{K}'\mathbf{R}'\,[\mathbf{I}| - \boldsymbol{t}]\,\boldsymbol{q}. \end{aligned}$$

By applying the inverse of the calibration and rotation matrices to the image point locations and equating, this leads to

$$\boldsymbol{p}' \sim \mathbf{H}\boldsymbol{p}, \quad \text{where } \mathbf{H} = \mathbf{K}'\mathbf{R}'\mathbf{R}^{\top}\mathbf{K}^{-1}. \tag{2.4}$$

A homography matrix with this form, comprising the rotation and calibration matrices for the two projections, is equal to the homography matrix obtained by assuming that the scene lies on the plane at infinity. Note however that in the case of coincident optical centres, this relation holds for any pair of corresponding image locations, not just those associated with scene points on a particular plane.

Since a homography matrix uniquely relates corresponding image points, it can be used to combine multiple images of a scene taken by cameras that observe (or approximately observe) the constraints mentioned above. Such a mosaic is depicted in figure 2.4, combining two images recorded by a rotating camera.

A homography matrix can be estimated from corresponding image points. As with the fundamental matrix, a linear algebra estimation is possible, though in this case it requires a minimum of four image point correspondences. A homography matrix estimated this way can be subsequently improved by an iterative minimisation with more sophisticated error measures. One such error is the distance between an image point and it's transformed matching location. Specifically, such a measure is given by the Euclidean distance between $\boldsymbol{p}'$ and $\mathbf{H}\boldsymbol{p}$ in one image, or $\boldsymbol{p}$ and $\mathbf{H}^{-1}\boldsymbol{p}'$ in the other image. The sum of these squared distances gives a symmetric error. Additionally, there exist Sampson and Gold Standard errors, analogous to the fundamental matrix case. The Gold Standard error is used to improve initial homography matrix estimates and generate mosaics in [3]. Further details on all these error measures can be found in [21].

Since the homography matrix relates noise-free image points exactly, other forms of estimation are also possible. In [61], just two pairs of corner features are used to estimate a homography matrix. Each corner can be considered the intersection of two lines. Two corner pairs therefore provide four corresponding line pairs, which are sufficient to uniquely compute a homography matrix. It is also possible to compute an estimate that minimises the sum of squared differences in intensities, for *all* corresponding pixel locations, rather than just a sparse set of features. Such an approach is used in [45].

Homography matrices can also be used to recover the calibration matrices of a non-translating camera. A homography matrix relating two images recorded by such a camera comprises calibration matrices and a relative rotation, as shown in equation 2.4. Since a rotation matrix is known to be orthogonal, this places a constraint on the intrinsic parameters. Hartley shows how to recover the unchanging intrinsic parameters of a purely rotating camera [24], from at least two unique homography matrices. Under a single additional constraint on the intrinsic parameters (such as zero skew), only one homography matrix is needed. The case of varying intrinsic parameters is examined by de Agapito et. al., using both non-linear [13] and linear [12] optimisations. In the non-linear case, not all intrinsic parameters can be assumed to vary. This provides constraints which ensure the estimation is over-determined. The linear case requires the stricter constraint that some intrinsic parameters are known. At a minimum, the skew is assumed to be 0. If additional constraints are available, such as known principal points or aspect ratios, less homography matrices are needed to estimate the remaining intrinsic parameters for each image. Note that once the intrinsics parameters of a rotating camera are known, the relative rotation between images can be recovered.

As is the case with fundamental matrices, a homography matrix places constraints on the locations of the projections of moving scene points. If cameras are relatively close together, or viewing an approximately planar scene, a moving point will project to locations related by the associated homography matrix in synchronous frames. Generally, this constraint will not be satisfied for image locations in asynchronous frames.

2.1.3 Robust Estimation

Feature-based estimation of a spatial transformation is typically initialised by choosing conspicuous locations in each image, such as those obtained with a Harris corner detector [20]. A tentative list of correspondences can be established by examining the pixel intensities around each point, and comparing via some measure of similarity. A number of such *correlation measures* are compared in [42]. In the case of a video sequence, correspondences can be found using the assumption that the displacement of a feature is small between successive frames. One such *tracking* method is described in [48]. Given image point $\boldsymbol{p}$ in one frame, the corresponding location $\boldsymbol{p'}$ in the next frame is sought that minimises a measure of difference between pixel regions around $\boldsymbol{p}$ and $\boldsymbol{p}'$. The location $\boldsymbol{p}'$ is found by an iterative minimisation using image gradient information. The Scale Invariant Feature Transform (SIFT) [32] can be used to locate features in images at varying scales. An image is transformed by Gaussian smoothing with multiple standard deviations. For some pre-chosen k, an 'intensity' value at (x, y, σ) is defined as the difference between Gaussian convolutions of the image, at point (x, y), with standard deviations of σ and $k\sigma$. Local extrema in this space exhibiting the appearance of a well-defined corner in (x, y) are chosen as features. Rather than using a small surrounding window of intensities to describe a feature, a distribution of the gradient magnitudes and orientations within a small window is used. Orientation is measured relative to the orientation of the gradient at the feature location. A normalised distribution of these gradient measures provides a description of the feature which is invariant to both rotation, and certain changes in illumination, of the original image. Such a description facilitates the identification of correspondences in images which differ by these properties.

The feature-based estimation strategies outlined in the previous sections assume that all the correspondences contributing to the estimation are correct, yet establishing such correspondences is an error-prone process. Features arising from different scene points may have a locally similar visual appearance, and be matched incorrectly as a consequence. Since such mismatches can significantly reduce the quality of the final estimate, there is a need for *robust* methods, which can identify such incorrect correspondences. An underlying model, such as a fundamental matrix or homography matrix, can then be computed using only the correct data. The estimation will therefore be robust, in the sense that it is unaffected by the incorrect data.

In [18], Fischler and Bolles introduce a strategy called Random Sample Consensus (RANSAC), which can be applied to a large variety of parameter estimation problems. The set of measured data consists of *inliers*, those which approximately fit the model to be estimated, and *outliers*, erroneous data samples which are only present due to errors in gathering the data. When dealing with image correspondences, an incorrect correspondence qualifies as an outlier. RANSAC achieves robustness by considering many estimates of the unknown parameters, assessing each individually, and selecting the one for which the most data are classified as inliers. Each iteration requires selecting a small random subset of the data, and using this subset to estimate the underlying model. The size of each subset is typically minimal, containing the smallest possible number of datum from which a candidate model estimate can be computed. In the context of homography matrix estimation, each datum would be a pair of corresponding image points, and four such pairs would be chosen at random for each subset. Given an over-estimate of the proportion of outliers in the data, the number of iterations is chosen such that there is a high probability that at least one of the randomly selected data subsets consists entirely of inliers. It is expected that such a subset should produce a reasonable estimate of the underlying model parameters. Each estimate of the parameters is assessed by counting the number of data with an associated error less than some threshold, denoted here as e_{th}. If the error associated with datum i is denoted e_i, and the total data set has size m, RANSAC can be considered as a search for the parameters that minimises

$$\sum_{i=1}^{m} \left\{ \begin{array}{lcl} 0 & : & e_i^2 < e_{th} \\ 1 & : & e_i^2 \geq e_{th} \end{array} \right. .$$

Upon completion, with the best estimate identified, this threshold test can be used to distinguish between inliers and outliers. The data classified as inliers can then all be used to generate a better estimate of the parameters. The threshold e_{th} can be chosen using prior knowledge of the distribution of inlying errors. For example, if the inlier errors are assumed to have a Gaussian distribution with 0 mean and known standard deviation σ, a threshold of $e_{thr} = 3.84\sigma^2$ should admit 95% of inliers. Note that erroneous data should in general fail the threshold test, and be classified correctly as outliers. Those wrongly classified as inliers should not significantly degrade the subsequent estimation, since their low associated errors indicates that they are reasonably consistent with approximately correct model parameters.

Rousseuw proposes the Least Median of Squares (LMedS) method [40], which is procedurally similar to RANSAC. Random subsets of the data are used to generate candidate parameter estimates. It differs in that the estimates are assessed by measuring the median squared residual, thereby searching for parameters which minimise

$$med \left\{ e_i^2 \,|\, i \in 1 \ldots m \right\} .$$

If inliers account for more than 50% of the data, then for approximately correct parameters, the median is a sample of the error measures associated with the inliers. Once the best estimate is found, inliers and outliers can again be classified using a threshold test. Unlike RANSAC, the threshold need not be chosen using prior knowledge of the distributions, and can instead be derived from the median.

M-estimator Sample Consensus (MSAC) is introduced by Torr and Zisserman in [50]. Like RANSAC and LMedS, multiple iterations are performed, with each iteration estimating the model parameters from a small randomly chosen subset of data. Each estimated model is assessed with the expression

$$\sum_{i=1}^{m} \left\{ \begin{array}{lcl} e_i^2 & : & e_i^2 < e_{th} \\ e_{th} & : & e_i^2 \geq e_{th} \end{array} \right. ,$$

Note that every datum i with $e_i^2 \geq e_{th}$ will yield a summand of the same value. The data classified as inliers however, produce summands equal to their associated error terms e_i^2. Using this expression, two model estimates classifying the same number of inliers will be ranked according to how well each fits its associated inlying data. Furthermore, this expression may be subsequently minimised, refining the initial model estimate, and possibly allowing data to be reclassified as inliers or outliers as appropriate. Torr and Zisserman also present a method called MLESAC [51], which models the distribution of errors associated with both inliers and outliers. Error terms for inliers are assumed to have a zero-mean Gaussian distribution with known variance. The outliers are assumed to have a uniform distribution, over the search range used to find candidate matching image features. The overall distribution of the data can therefore be expressed as a linear combination of these two distribution types, if the proportion of inliers is known. As with the previous methods, estimates are produced using random subsets of the data. For each estimate, the proportion of inliers is determined using Expectation Maximisation, which defines the data distribution. The likelihood of the error terms is then computed and used to assess the model estimate. In [49], Torr introduces Maximum a-posteriori Sample Consensus (MAPSAC). Prior probability distributions can be assumed for the minimal parameterisation of a 'true' correspondence (of which data are noisy estimates), the model to be computed, and for inlier and outlier classification. A model estimate can then be assessed by the posterior, computed by weighting the likelihood (such as that used in MLESAC) by the prior terms. If the prior probability distributions indicate that a model estimate, its associated inlier classifications, or 'true' correspondences are unlikely, the posterior will specify the model estimate is of lower quality.

Even if a set of data used for a robust estimation contains a high proportion of inliers, the probability of a randomly chosen subset containing *no* outliers will typically be small. The majority of iterations during a robust estimation will therefore produce model parameters of unacceptable quality. Additionally, since the random subsets are typically chosen to be of minimal size, the model parameters are usually generated by fast linear algebra methods to precisely fit the data subset. For a large data set, evaluating the model and classifying the data will dominate the time required to execute each iteration. Most of the execution time in the above methods is therefore spent evaluating poor parameter estimates. Chum et. al. have introduced a number of strategies to decrease this cost. The $T_{d,d}$ test [10] quickly identifies those model estimates which are likely to be of poor quality. Such models can then be ignored without requiring a full evaluation. RANSAC is modified so that each model estimate is first assessed using a small random subset of the data of size d, and discarded if any are classified as outliers. Using such a strategy, the majority of poor estimates can be identified quickly. An application-specific test is described in [11], in which a fundamental matrix estimate generated by seven randomly chosen correspondences is ignored if it corresponds to a configuration where not all seven scene points lie in front of both cameras. Speedup can also be achieved by using an alternative sampling strategy called PROSAC [9], by associating a measure of quality with each datum (such as a correlation measure in the case of matches). Random samples are initially drawn from a set containing only the 'best' data. This set is grown over time if no parameter estimate satisfying the termination condition is found. Since the better data are more likely to be inliers, a suitable parameter estimate can often be found with far fewer iterations.

Just as sets of image correspondences may be contaminated with mismatches, a set of visual cues used for synchronisation may also be mismatched between video sequences. Robust methods can be used to classify such outlying data, while obtaining an initial estimate of the synchronisation parameters (a, b). Once the outlying data is classified, it can be ignored, permitting the video sequences to be accurately synchronised with just the inlying visual cues.

2.2 Existing Synchronisation Methods

In [39], Reid and Zisserman examine a notorious goal scored by England during extra time of the 1966 World Cup final. This task involves determining the path of the soccer ball throughout the scene, which requires analysing frames recorded simultaneously by different video cameras. A soccer pitch is approximately planar, so a homography matrix relating the images of this plane in any two frames can be computed from corresponding stationary features, such as the intersection of line markings. Moving features on the ground, in this case the points on players' shadows, should also be related by this homography matrix if the frames are synchronous. The video sequences are synchronised by searching for a frame offset such that the moving point correspondences are best related by the computed homography matrices. Note that this method requires the careful selection of both stationary and moving corresponding points on the ground plane only.

Stein presents a fully automatic synchronisation algorithm [43] for a pair of cameras observing objects that move close to the ground plane. The cameras are stationary, so the homography matrix relating the views of the ground plane remains unchanged over time. The centroids of moving objects in each frame are identified with background subtraction methods, and a 1D search is then conducted to find the frame offset. For each estimate, a list of all pairs of centroids in approximately synchronous frames is made. The homography matrix is estimated from these matches using robust methods. The quality of both spatial and temporal parameters is measured by computing the 2D distance between the centroid locations in one camera, and the hypothesised matching centroids from the other camera, transformed by the homography matrix. The 20^{th} percentile of the sorted list of these errors provides the measure of quality. So through a search of the frame offset, and a repeated random sampling of hypothesised matches, this method seeks to minimise the function

$$Percentile_{20}\left(\left\{d(\boldsymbol{p}_{h,i}, \mathbf{H}\boldsymbol{p}'_{k,j}) \,|\, \forall i, j, h, k,\ where\, |a + bi - j| < t\right\}\right),$$

where t is the threshold for assuming approximate synchrony. The vector $\boldsymbol{p}_{h,i}$ represents the location of centroid h in frame i of the first video sequence. The vector $\boldsymbol{p}'_{k,j}$ is the location of centroid k in frame j of the second video sequence. The function $d(\dots)$ maps the vectors $\boldsymbol{p}_{h,i}$ and $\boldsymbol{p}'_{k,j}$ to Euclidean points and measures the distance between them. The advantage of this method is that no stationary matches are required. Moving objects, however, are still approximately constrained to the ground plane, and the cameras must remain stationary to facilitate background subtraction.

A featureless synchronisation method using homography matrices is described by Caspi and Irani [6]. Their method differs from the previous ones in that it assumes that the entire frames, not just the ground plane images, are related by a homography matrix. This requires that either the entire scene is approximately planar, or that the optical centres of the cameras are almost coincident. The method seeks to find a homography matrix and synchronisation parameters which minimise the sum of squared differences between corresponding pixels in synchronous frames, measured as

$$\sum_{x',y',i} \left(I'_{a+bi}(x',y') - I_i(\mathcal{N}_{Euc}(\mathbf{H}[x',y',1]^T))\right)^2 . \tag{2.5}$$

The function $\mathcal{N}_{Euc}$ maps a projective point to the corresponding Euclidean point, by dividing the first two elements of the projective point by the third. The terms $I_k(x,y)$ and $I'_k(x,y)$ refer to the pixel value at location (x,y), in frame k of the first and second video sequences respectively. The synchronisation parameters which minimise the expression given in equation 2.5 are found using Gauss-Newton minimisation in a coarse-to-fine sampling of the data. Unlike previous methods, this strategy not only makes use of moving objects, but other changes in the scene such as illumination. The constraint of a single homography relationship between the sequences requires that the cameras either remain stationary or move rigidly as a pair such as two fixed cameras mounted on a moving rig. This method is illustrated in figure 2.5, which depicts selected frames from two video sequences of a bouncing ball. Note that the difference image for the overlapping image regions is dark for the synchronous frames, indicating similar pixel intensities. The difference image for the asynchronous frames has brighter regions, due to the motion of the ball and its shadow.

Rather than using scene changes, video sequences can also be synchronised by using constraints on the motion of cameras, as shown in a paper by Caspi and Irani [5]. Two video cameras are rigidly held together, and moved while recording. Since the optical centres are close together, it can be assumed that all pairs of synchronous frames are related by the same homography matrix. For each frame, the fundamental matrix or homography matrix describing the camera motion to the next frame in the same video sequence is computed. A search is then made for a frame offset such that synchronous frames have transformations to their successors that are most consistent with the fact that the cameras are related by an unchanging homography matrix. Once the best frame offset is found, the homography matrix relating the two cameras can be estimated, even in the case where the cameras view nothing in common.

Caspi et. al. present a feature based synchronisation method [7]. Moving image features are detected and tracked independently in each video sequence, providing a set of 2D trajectories associated with each camera. A set of hypothesised matching trajectories is constructed, consisting of all possible trajectory pairs. Alternatively, a smaller set may

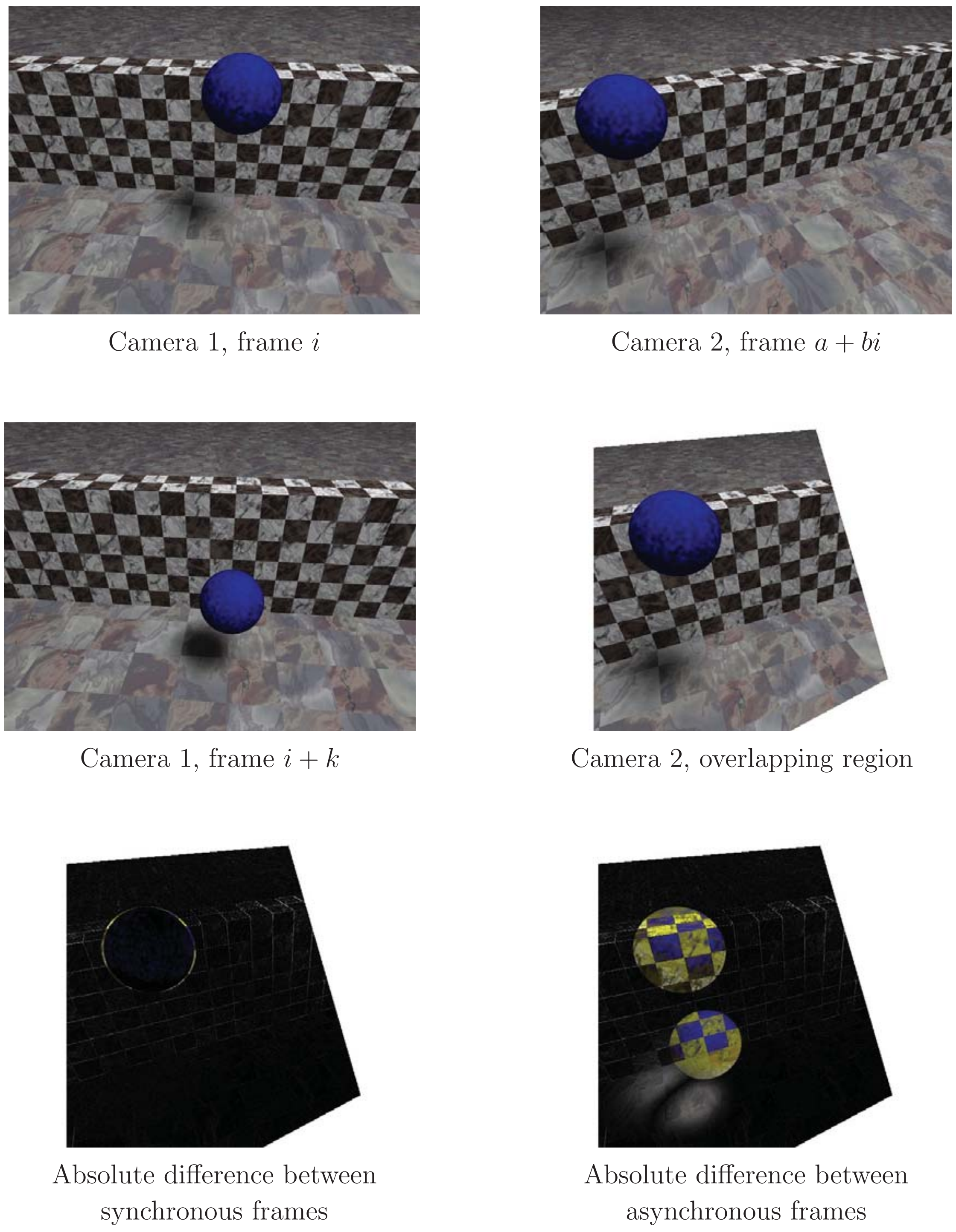

Camera 1, frame i
Camera 2, frame $a + bi$
Camera 1, frame $i + k$
Camera 2, overlapping region
Absolute difference between synchronous frames
Absolute difference between asynchronous frames

Figure 2.5: An example of pixel-based synchrony with a synthetic scene of a bouncing ball

be used by assuming that corresponding trajectories have similar properties such as colour and shape. Robust estimation is used to achieve synchronisation by repeatedly choosing a trajectory pair at random. For each pair, a search is conducted to find the frame offset. Each frame offset value specifies a set of image point correspondences, drawn from the trajectory pair at synchronous frames. These are used to compute a fundamental matrix or homography matrix. As is the case in [6], this requires that the cameras remain stationary, or move consistently together such that the spatial relationship remains unchanged over time. This is similar to the method used by Stein in [43], but differs in that whole trajectories are considered as correspondences, rather than independently matching points in each frame pair. This reduces the number of potential matches, thereby reducing the number of iterations required for the robust estimation. It is noted that a typical pair of recording cameras will not capture frames at precisely the same times. For frame i in the first video sequence, the synchronous frame index in the second video sequence, computed as $a + bi$, will not be an integer. To resolve this, tracked image features are linearly interpolated to define their locations 'between frames'. This allows the computation of error terms, such as epipolar errors, in the case where $a + bi$ is not an integer. It also permits the frame offset to be refined, synchronising the video sequences to an accuracy within a fraction of a frame.

Wolf and Zomet propose a strategy in [59] that does not require corresponding features between the video sequences, nor estimation of the spatial transformation relating the cameras. Features moving through the image are identified and tracked in each video sequence. It is assumed that each 3D feature observed by one camera can be expressed as a linear combination of 3D features observed by the other camera. The coefficients of the linear combination for each scene point remain unchanged over time. Note this is equivalent to the case where each camera observes possibly differing features on the same set of rigidly moving objects. Each camera is described by an orthographic projection matrix, which remains constant over time. Given a frame offset a, a matrix M is constructed from the set of image points in both video sequences. Each pair of rows in M contains the image locations for one of the tracked features from either video sequence. Each column of M contains the set of image points for frame i in the first video sequence, and the synchronous frame $a + bi$ in the second video sequence. An upper bound on the rank of M can be assumed, and a search is conducted to find the frame offset for which the associated matrix M best conforms to this rank constraint.

A similar rank constraint is used by Tresadern and Reid to synchronise a pair of stationary cameras [52]. Moving scene features are tracked in each sequence, and correspondences are established between the resulting 2D trajectories. Assuming affine projection, image correspondences for a pair of frames can be arranged in a matrix that is known to have rank 3 if the frames are synchronous. For each frame in the first video sequence, a search is conducted to find a corresponding frame from the second video sequence such that this rank constraint is best satisfied. This search provides a number of frame index pairs (f, f'), a subset of which are presumed to be synchronous. RANSAC is used to robustly obtain the synchronisation parameters (a, b) that approximately relate the majority of these frame pairs. The synchronisation estimate is then refined, by seeking a new set of hypothesised synchronous frame pairs, using image point interpolation to approximate the projection of a moving scene point for times between integer frame index values.

The typical linear constraint relating synchronous frame indices is relaxed in a paper by Rao et. al. [38]. Assuming stationary cameras, moving scene features are identified and tracked independently in each video sequence. Once corresponding trajectories are identified, each pair of hypothesised synchronous frames has an associated error. This error is given by the minimum cost associated with an algebraic estimation of the fundamental matrix relating the cameras. These errors therefore define a surface in the space of the frame indices of the two video sequences, $\mathcal{E}(f, f')$. A path of low errors through this surface is identified, comprising a set of hypothesised synchronous frame pairs. Each such pair is required to be adjacent to another pair on the path, but no specific underlying temporal model is assumed, permitting a non-linear synchrony of the video sequences. This algorithm can therefore be used even for video sequences depicting different scenes, providing they share similar motion patterns. The path of synchronous frames is identified in a coarse-to-fine manner, at first using only a low sampling of frames from each video sequence.

A correspondence free synchronisation strategy is proposed by Yan and Pollefeys [60]. Each video sequence is uniformly sampled for *space-time interest points*. Such points are analogous to Harris corners, but exhibit a corner-like appearance in the 3D volume of pixel values indexed by image location and time. A common event generating a space time interest point is a sudden change in disparity of a corner feature. Figure 2.6 depicts such an event, for a camera observing a moving grey square. The resulting 3D volume of grey pixels is shown by key images and a wire-frame. The change in disparity half way through the sequence generates 3D corners in this volume. If a pair of stationary cameras both observe a sudden change in direction of a moving scene point, we expect a space-time interest point to be generated in each of the pair of synchronous frames. The distribution of these features within each video sequence is estimated, and a histogram over time is built. A search is performed for a frame offset which best correlates the two histograms.

The case of a pair of independently moving cameras is examined in by Tuytelaars and Van Gool [55]. Their method assumes scaled orthographic projection, and uses 5 corresponding pairs of trajectories tracked in the video sequences. Each trajectory pair corresponds to a scene point, and it is assumed that these scene points exhibit non-rigid motion over time. Four of the scene points define the world coordinate system at each instance in time, essentially providing a projection matrix for each frame of the video sequences. The fifth scene point then provides a cue for synchronisation. A 1D search is conducted for a frame offset such that the images of the fifth scene point in synchronous frames back-project to intersecting rays in 3D space.

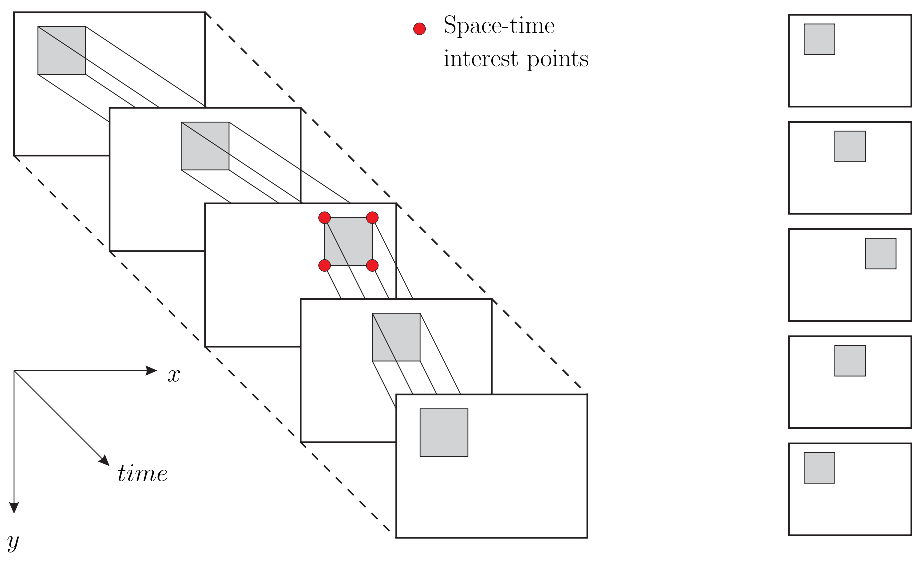


Figure 2.6: Key frames from a video sequence of a moving grey square (right), and resulting space-time interest points (left)

Carceroni et. al. consider the case of an arbitrary number of stationary cameras [4]. Moving scene points are tracked in each video sequence, and their locations for times in between frames are approximated by a linear interpolation. For each moving scene point feature in frame i of the first video sequence, the corresponding epipolar line in every other camera is computed. The intersection of this epipolar line with a 2D trajectory defines a hypothesised frame index j, possibly synchronous with i. In the case of more than two cameras, such frame pairs can be combined to produce a set of vectors, where each vector has the form $(f_1, f_2, \ldots, f_k)$. The frame indices f_1 to f_k denote a set of frames, one per camera, which are hypothesised to all have been recorded at the same time. RANSAC is used to robustly estimate the synchronisation parameters relating each camera with the first.

Two papers by Wedge et. al. describe methods for synchronising a pair of stationary cameras, using a visual cue provided by the vertical motion of a single moving object. In [58], each video sequence is divided into sub-sequences. Synchronisation is estimated for each sub-sequence pair independently, by finding a frame offset such that the vertical direction of the object's motion is consistent in synchronous frames. Consensus using a histogram approach determines a constrained range for the frame offset that will synchronise the entire video sequences. For each integer frame offset in this range, the fundamental matrix relating the cameras is estimated from moving image correspondences in synchronous frames. The frame offset for which an estimated fundamental matrix has the lowest associated algebraic error is determined. This initial estimate is then refined by using image point interpolation to approximate the location of a moving image feature between frames, and performing a binary search to determine a frame offset between integer values. In [57], a different approach is used. The fundamental matrix F relating a pair of stationary cameras is estimated from stationary scene point correspondences. A frame i is chosen from the first video sequence in which the moving object exhibits significant vertical motion. The location of the moving object in this frame, denoted $\boldsymbol{p}_i$, is mapped to its corresponding epipolar line in the second camera, given by $\mathsf{F}\boldsymbol{p}_i$. A frame j is sought from the second camera, such that the moving point location $\boldsymbol{p}'_j$ lies along this line. Such a j is found by initially choosing a frame in which the vertical direction of the moving object matches that of frame i. The distance between $\mathsf{F}\boldsymbol{p}_i$ and $\boldsymbol{p}'_j$ is used to iteratively predict a more accurate value for j. Linear point interpolation is used to identify a hypothesised synchronous (i, j) frame pair where j may lie between integer values. This process is repeated to find many such frame pairs, and the synchronisation parameters are estimated using linear least squares. This is similar to the method used by Carceroni [4], but is only suitable for simple 3D motions since *every* such frame pair (i, j) is assumed to be approximately correct.

Lei and Yang present an algorithm for the synchronisation of three independently moving cameras in [30]. Corresponding point and line features are determined between the video sequences, and then tracked independently in each. At least a subset of these features are presumed to be projections of moving scene features. A uniform search is conducted over frame offsets relating the second and third cameras with the first. Each combination of frame offsets sampled represents a hypothesised synchronisation of the three cameras, and defines a set of synchronous frame triplets. The spatial geometry relating each such triplet of frames is estimated from the point and line correspondences. A synchronisation estimate can then be assessed based on the consistency of the spatial relationships with their associated point and line correspondences.

Serrat describes a method for the synchronisation of a pair of cameras with independent but restricted motion [41]. Specifically, the paths of the cameras through the scene are assumed to be almost coincident, and with similar orientations. The synchrony of a pair of frames is assessed using a pixel-based method, similar to that in [6]. A homography is estimated that minimises the sum of squared pixel intensity differences for corresponding image locations. Note that this method differs from that of [6] in that a unique homography is estimated for *each* pair of frames. While the cameras may therefore move independently, this independence is limited. Synchronisation is determined by considering a number of frames from the first video sequence. For each such frame i, a corresponding frame j is sought from the second video that exhibits a low associated difference in pixel values. Similarly to [38], no specific linear synchronisation model is assumed. This permits the synchronisation of two video sequences, recorded with cameras undergoing similar motions through the scene, possibly at different times. This method does not specifically seek to establish a correspondence in *time* between the cameras, but rather determine which frames were recorded while the cameras were at approximately coincident locations.

2.3 Problem Formulation

The focus of this thesis is this synchronisation of independently moving cameras. As described in section 2.2, the synchronisation of more than two video sequences has been examined for both stationary [4] and moving [30] cameras. Here, only a *pair* of cameras will be considered. This is justified by observing that any scheme for synchronising a pair of cameras can be trivially extended to three cameras or more, by combining pair-wise synchrony to achieve a globally consistent timestamping of all video sequences. An initial estimate of the global synchrony can then be refined by considering corresponding visual cues from every camera pair.

Two of the methods summarised in section 2.2 address the problem of synchronising independently moving cameras under particular constraints. In [55], a scaled orthographic projection is assumed. This equates to assuming that the rays in space projecting a 3D scene to an image are parallel, and such an assumption is therefore unsuitable for some camera motions. The pixel-based method in [41] assumes that camera motions are approximately coincident. Conversely, this thesis will allow full perspective projection, as described in section 2.1. The only constraint assumed on the camera motion is that a moving object must be observable to both cameras for some period of time. This amounts to assuming that the video sequences have *some* overlap in time, and that the moving object is observable in both sequences for some subset of synchronous frames.

A similar assumption of fully independent motion is also used in [30], where spatial transformations relating synchronous frames are estimated from tracked corresponding features. A different approach will be used here. The transformations relating synchronous frames do not provide a complete spatial description of the cameras. The precise motions of the cameras, described by associating a projection matrix with every frame, can not be recovered from such transformations alone. A complete spatial description is necessary for many applications, such as the recovery of the 3D motion of features through the scene. While the method in [30] does not *preclude* a full spatial registration of the cameras, the estimation of many transformations relating synchronous frames is redundant if a complete spatial description of the cameras can be estimated.

This thesis will demonstrate that, in the case where the cameras are spatially registered, an accurate and efficient synchronisation can be realised, based on visual cues from moving objects. The underlying methods used to achieve this are described in chapters 3 and 4. It will also be shown that such an estimation can be robust, much like the stationary camera cases described in [43] and [7]. Chapter 5 will show how to achieve a fast robust recovery of synchronisation, while identifying corresponding projections of moving scene points in the camera pair. Finally, chapter 6 will describe a means for using moving scene points to assist in determining how the motions for each camera are related in space. It will therefore be shown that moving scene points provide a visual cue that enables both the robust estimation of synchrony, and a spatial registration of the cameras.

Chapter 3

SYNCHRONISING A PAIR OF MOVING CAMERAS

Automatic video synchronisation has previously been analysed assuming certain scene or camera constraints. In particular, the majority of previous methods place strong constraints on the motion of cameras. They must either be stationary, or move rigidly as a pair. In this chapter, we consider the situation where a pair of cameras move independently as they record images of a scene consisting of both static and dynamic objects. It is assumed that a subset of the dynamic objects are visible to both cameras at the same time, for a significant number of frames. These objects provide a visual cue which can be used to estimate the frame offset, and optionally the frame rate ratio.

Since the cameras have independent motion, pixel based methods such as the one described in [6] are inappropriate as there is no simple spatial transformation that relates corresponding pixels in synchronous frames. Instead, we use a feature-based method that makes use of image points (such as corners) tracked throughout each video sequence.

In [43, 7], both the synchronisation parameters and a spatial transformation are estimated using information from only the moving objects. For independently moving cameras, any two frames from the same video sequence have different timestamps, so the images of a moving scene feature in these frames are projections of different 3D locations. Accordingly, the moving objects place no constraints on the spatial transformations between frames from the *same* video sequence. To recover an estimate of both temporal and spatial parameters, information from stationary scene objects must also be used. We therefore assume that the images of stationary scene features have been identified and tracked in each video sequence. Furthermore, it is assumed that sufficient stationary scene correspondences have been established between the video sequences, permitting the computation of the fundamental matrix associated with *any* pair of frames.

In this chapter it is also assumed that extended tracks for moving scene points have been correctly associated between the video sequences, but not matched at a point-to-point level. This level of matching is illustrated in figure 3.1.

3.1 The Evolution of a Cost Function

In a pair of synchronous frames, the projections onto the images of a moving scene point originate from a common 3D location. These image points will therefore be related by the fundamental matrix which also relates projections of the stationary scene features. In

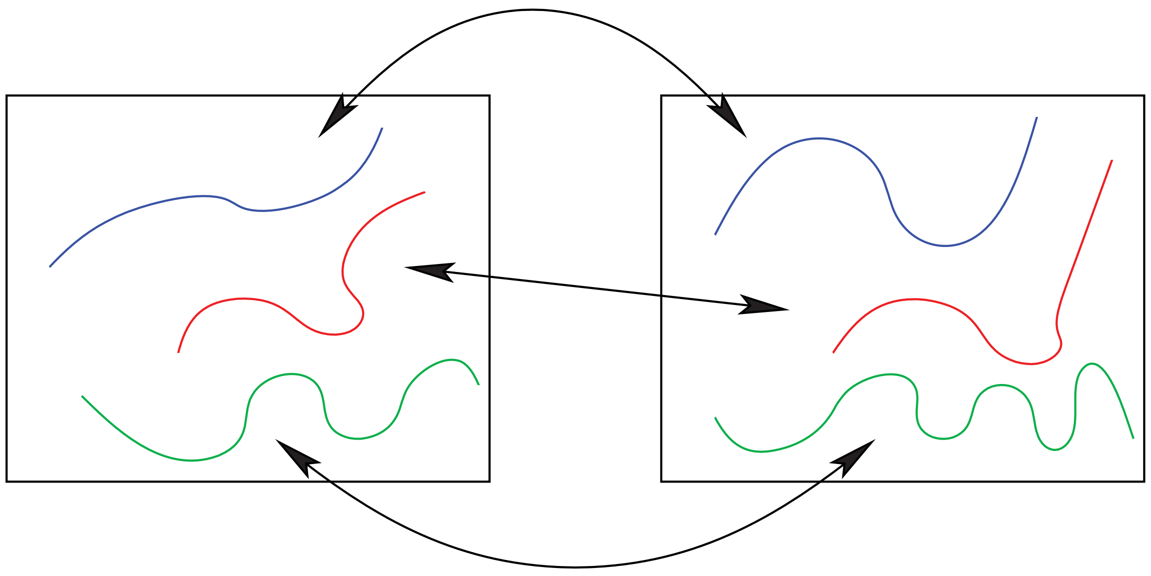
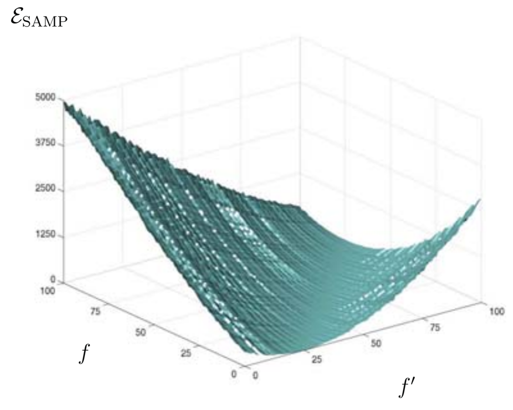


Figure 3.1: Corresponding trajectories between a pair of video sequences, identified prior to synchronisation

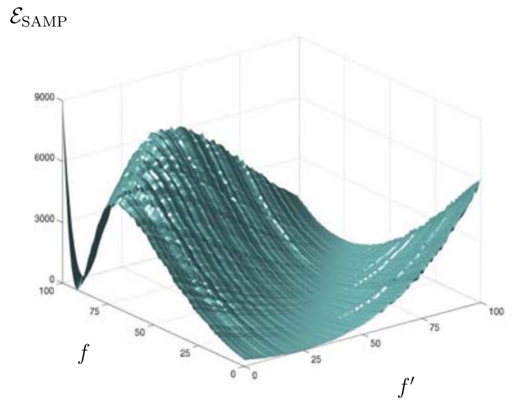
asynchronous frames, these points will in general not conform to this epipolar constraint. The epipolar errors associated with such a pair of image points therefore provide a means by which the quality of the synchronisation parameters can be measured.

Figure 3.2 shows two examples of epipolar error measurement. In each example, the squared Sampson error of the projections of a moving scene point has been measured for every possible frame pair. The horizontal axes specify the frame indices for the two cameras, and the Sampson error is measured along the vertical axis. It can be assumed that this error measure will be low for approximately synchronous frames, and that such frame pairs have the linear relationship defined by the synchronisation parameters (a, b) described in section 1.1. Synchronising a pair of video sequences requires finding a long straight valley in the surface described by the epipolar errors. Note that the second 'valley' in figure 3.2(b) is indicative of the fact that, although synchronous frames have low epipolar error, not all frame pairs with low epipolar error are necessarily synchronous.

Now suppose that video sequences 1 and 2 have n and n' frames, respectively, and exhibit corresponding moving scene points, numbered from 1 to m. Let point h project to locations $(\boldsymbol{p}_{h,0}, \boldsymbol{p}_{h,1}, \ldots, \boldsymbol{p}_{h,n})$ in the frames of video sequence 1 and $(\boldsymbol{p}'_{h,0}, \boldsymbol{p}'_{h,1}, \ldots, \boldsymbol{p}'_{h,n'})$ in the frames of video sequence 2. Frames i and j from the first and second video sequences have associated projection matrices M_i and M'_j, which can be used to generate a fundamental matrix relating the two views, as shown in section 2.1.1. $\mathsf{F}_{i,j}$ denotes the fundamental matrix mapping an image point in frame j of video sequence 2 to its corresponding epipolar line in frame i of video sequence 1.



(a)



(b)

Figure 3.2: Two cases of epipolar errors for synchronous and asynchronous frame pairs

A simple cost function can be obtained by computing the sum of squared symmetric epipolar errors, obtained by considering each of the m moving scene points, and their images in each frame of the first video sequence. The function is given by

$$\mathcal{S}_{\mathrm{SIMPLE}}(a,b) = \sum_{h=1}^{m}\sum_{i=0}^{n-1} \mathcal{E}_{\mathrm{SYMM}}(\boldsymbol{p}_{h,i}, \boldsymbol{p}'_{h,a+bi}, \mathsf{F}_{i,a+bi})^2.$$

Although we expect this function to be low for accurate synchronisation estimates of (a, b), there are some issues that must still be addressed. Note that certain epipolar errors may not be measurable. Image points may be occluded in certain frames, and for some values of frame index i in the first video sequence, $a + bi$ may lie outside the range of $[0, n' - 1]$. The simplest solution is to ignore these cases by setting the corresponding epipolar error to 0. A side-effect of this is that the cost function may reward synchronisation parameters that lead to a smaller overlap in time. This problem is illustrated by a simple case in example 3.1.

The solution we use to counter the problem of unmeasurable epipolar errors is to compute the *average* of squared epipolar errors, rather than the sum. Note that in the case of example 3.1, if the average of squared epipolar errors is used, the cost function equates to $(ka)^2$, which is parabolic and therefore does not suffer from local maxima.

Another problem arises from the fact that, for a frame rate ratio $b > 1$, some frames in the second video sequence may be skipped altogether. These skipped frames contain important image point information, which should ideally be included in the cost function. It therefore makes more sense to measure epipolar errors for every frame in both video sequences.

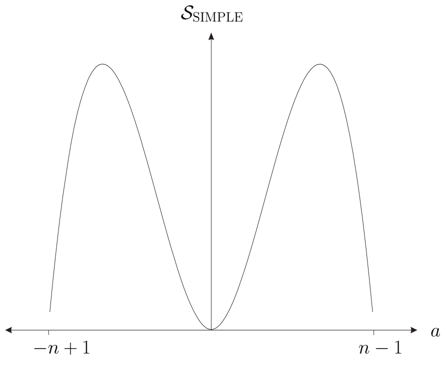
To account for immeasurable epipolar errors, and using the average of squared epipolar errors, the following notation is used. A function measuring epipolar error is denoted

Example 3.1 (Problematic Sum of Squares Example) *Consider the scenario where $a = 0.0$, $b = 1.0$, $n = n'$. We examine the case where a single moving point generates epipolar errors that are equal for every frame of the first video sequence, and that increase linearly as the offset estimate differs from the truth. A point moving vertically at constant speed in front of two rectified stationary cameras can have this effect. Now, for an integer estimate of a,*

$$\mathcal{S}_{\text{SIMPLE}}(a,1) = \sum_{i=0, a+i\in[0..n-1]}^{n-1} (ka)^2.$$

Although this equation appears to be parabolic, note that as a differs from 0, fewer epipolar distances are measurable. For an integer estimate of $a \in [-n+1, n-1]$, the number of measurable epipolar distances is $(n-|a|)$. It therefore follows that

$$\begin{aligned}\mathcal{S}_{\text{SIMPLE}}(a,1) =& (n-|a|)(ka)^2\\ =& nk^2a^2 - sign(a)k^2a^3, a \in [-n+1, n-1].\end{aligned}$$



The cost function is therefore piecewise cubic, not quadratic. It has local maxima at $a = \pm\frac{2n}{3}$. This means that an estimate $a > \frac{2n}{3}$ is considered better than the estimate $a = \frac{2n}{3}$, despite the facts that all measurable epipolar errors are higher, and the synchronisation estimate is worse.

$\mathcal{E}_{IDENT}$, where *IDENT* specifies the type of epipolar distance measured. The list of inputs to function $\mathcal{E}_{IDENT}$ is denoted ζ. Note this generalisation is used since epipolar error measures developed in later sections use differing inputs. We define $\mathcal{E}_{IDENT}{}^{\star}$ as the corresponding function which equates to $\mathcal{E}_{IDENT}$ if the epipolar error is measurable, and 0 otherwise. Specifically,

$$\mathcal{E}_{IDENT}{}^{\star}(\boldsymbol{\zeta}) = \begin{cases} \mathcal{E}_{IDENT}(\boldsymbol{\zeta}) & : \ \mathcal{E}_{IDENT}(\boldsymbol{\zeta}) \text{ is defined} \\ 0 & : \ \text{otherwise} \end{cases}$$

The function $\mathcal{W}$ specifies how many epipolar errors are measurable, for all m moving scene points, and every frame in the two video sequences. Note that the value of $\mathcal{W}$ will not only depend on how many frames each moving scene point is visible in, but also on the synchronisation parameters (a, b). The cost function to assess the quality of synchronisation parameters (a, b), by computing the average of measurable epipolar errors, where *DIST* is one of $\{\text{ORTH}, \text{SYMM}, \text{SAMP}\}$, is

$$\begin{aligned} \mathcal{S}_{DIST}(a,b) = & \frac{1}{\mathcal{W}(a,b)} \sum_{h=1}^{m} \sum_{i=0}^{n-1} \mathcal{E}_{DIST}{}^{\star}(\boldsymbol{p}_{h,i}, \boldsymbol{p}'_{h,a+bi}, \mathsf{F}_{i,a+bi})^2 + \\ & \frac{1}{\mathcal{W}(a,b)} \sum_{h=1}^{m} \sum_{j=0}^{n'-1} \mathcal{E}_{DIST}{}^{\star}(\boldsymbol{p}'_{h,j}, \boldsymbol{p}_{h,a'+b'j}, \mathsf{F}^{\top}_{a'+b'j,j})^2, \\ & \text{where } a' = -ab^{-1}, \quad b' = b^{-1}. \end{aligned} \tag{3.1}$$

Note that (a', b') are the parameters that give the inverse of the line defined by (a, b).

3.2 Handling Indices Between Integer Values by Interpolation

Consider the cost functions defined by 3.1. One significant problem still remains. Image points and fundamental matrices are only defined for integer frame indices. Given an integer frame index i, and general synchronisation parameters (a, b), the value of $a + bi$ is not necessarily an integer. Methods for measuring epipolar distances for such values of $a + bi$ are therefore required.

A simple solution is to linearly interpolate the epipolar distances as measured in the two closest frames. We therefore define

$$\mathcal{E}_{DIST\text{-RI}}(\boldsymbol{p}, \boldsymbol{p}'_0, \mathsf{F}_0, \boldsymbol{p}'_1, \mathsf{F}_1, q) = (1-q)\mathcal{E}_{DIST}(\boldsymbol{p}, \boldsymbol{p}'_0, \mathsf{F}_0) + q\mathcal{E}_{DIST}(\boldsymbol{p}, \boldsymbol{p}'_1, \mathsf{F}_1)$$

to be the function that linearly interpolates between two epipolar distances, where *DIST* is one of $\{\text{ORTH}, \text{SYMM}, \text{SAMP}\}$. The function that counts the number of measurable interpolated epipolar errors, for all scene points and all frames in the two video sequences, is denoted $\mathcal{W}_\text{I}$. The corresponding cost function to assess synchronisation parameters is

$$\begin{aligned}\mathcal{S}_{DIST\text{–RI}}(a,b) = \frac{1}{\mathcal{W}_\text{I}(a,b)} \sum_{h=1}^{m}\sum_{i=0}^{n-1} \mathcal{E}_{DIST\text{–RI}}{}^{\star}(&\boldsymbol{p}_{h,i},\, \boldsymbol{p}'_{h,\lfloor a+bi\rfloor},\, \mathbf{F}_{i,\lfloor a+bi\rfloor},\\ &\boldsymbol{p}'_{h,\lfloor a+bi\rfloor+1},\, \mathbf{F}_{i,\lfloor a+bi\rfloor+1},\\ &a+bi-\lfloor a+bi\rfloor)^2 +\\ \frac{1}{\mathcal{W}_\text{I}(a,b)} \sum_{h=1}^{m}\sum_{j=0}^{n'-1} \mathcal{E}_{DIST\text{–RI}}{}^{\star}(&\boldsymbol{p}'_{h,j},\, \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor},\, \mathbf{F}^{\top}_{\lfloor a'+b'j\rfloor,j},\\ &\boldsymbol{p}_{h,\lfloor a'+b'j\rfloor+1},\, \mathbf{F}^{\top}_{\lfloor a'+b'j\rfloor+1,j},\\ &a'+b'j-\lfloor a'+b'j\rfloor)^2,\\ \text{where } a' = -ab^{-1},\quad b' = b^{-1}.\end{aligned}$$

Note that an epipolar distance associated with a particular moving scene point may now be measured in frame i from the first video sequence, where $a+bi$ is not an integer. This requires that $a+bi$ is within the range $[0,n']$, and the moving scene point has been tracked in the two frames from the second video sequence with indices closest to $a+bi$.

This method of residual interpolation has a serious potential drawback, however. In certain circumstances, the cost function may be locally flat, limiting the accuracy of the synchronisation. A simple case demonstrating this possibility is presented in example 3.2.

Example 3.2 (Problematic Residual Interpolation Example) *Consider the spatial configuration described in example 3.1. A point moves vertically at constant speed in front of two rectified stationary cameras, with equal frame rate. As a result, epipolar error measures $\mathcal{E}_\text{ORTH}$ increase linearly as the offset estimate differs from the truth. Since the cameras are rectified, epipolar lines are horizontal. Symmetric and Sampson errors therefore also increase linearly as the offset changes.*
Assuming the true frame offset is given by $a = 0.5$, then the epipolar errors for candidate offset estimates $\hat{a} = 0$ and $\hat{a} = 1$ are equal. If a cost function using an interpolation of residuals is used, the function is flat between these estimates if $\mathcal{W}_\text{I}$ (the number of measurable errors) is unchanged. Otherwise, the minimum will occur at whichever of these two estimates has a higher value of $\mathcal{W}_\text{I}$.
In this case, achieving synchronisation to within a fraction of a frame is impossible via an interpolation of residuals, regardless *of how many scene points with this class of motion are tracked.*

Another possibility is to interpolate the information used in computing the epipolar errors, rather than interpolating the errors themselves. This approach is used in [7], where image points are linearly interpolated to approximate their locations between integer frame indices. Such a strategy is insufficient for independently moving cameras, since fundamental matrices also change between frames. A simple alternative is to interpolate epipolar lines.

Consider frame i in video sequence 1, where $k = a + bi$ for some estimate of (a, b) is not an integer. The two integer frame indices closest to k in video sequence 2 are given by $\lfloor k \rfloor$ and $\lfloor k \rfloor + 1$. The images of moving point h in these frames map to epipolar lines in frame i of video sequence 1, given by

$$\begin{aligned} \boldsymbol{l}_{h,i,\lfloor k \rfloor} &= \mathsf{F}_{i,\lfloor k \rfloor} \boldsymbol{p}'_{h,\lfloor k \rfloor}, \\ \boldsymbol{l}_{h,i,\lfloor k \rfloor+1} &= \mathsf{F}_{i,\lfloor k \rfloor+1} \boldsymbol{p}'_{h,\lfloor k \rfloor+1}. \end{aligned}$$

Since we wish to measure the epipolar distance for a value of k between two integers, we will use a line that in some sense lies *between* the lines $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$. Such a line can be obtained by performing a linear interpolation, thereby producing another line which passes through the point at which $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$ intersect.

The line vectors $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$ have a scale-invariant representation. If one of these vectors is multiplied by a non-zero scalar, it still represents the same line in 2D space, yet the result of a linear interpolation to estimate a line between integer frame indices will be different. To account for this ambiguity, each epipolar line is first normalised so that the sum of the squares of its first two coordinates is 1. The first two elements of the resulting vector then equate to $\sin \theta$ and $\cos \theta$, where θ represents the line's orientation. We denote this normalisation function as $\mathcal{N}_{\mathrm{L}}$, defined as

$$\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}) = \frac{\boldsymbol{l}}{\sqrt{[\boldsymbol{l}]_1^2 + [\boldsymbol{l}]_2^2}}$$

An interpolation of the lines $\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_{h,i,\lfloor k \rfloor})$ and $\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_{h,i,\lfloor k \rfloor+1})$ is denoted $\hat{\boldsymbol{l}}_{h,i,k}$. This interpolated line will have an orientation defined by an interpolation of two points on the unit circle that describe the orientations of $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$. This is illustrated in figure 3.3, where the two epipolar lines and interpolated line have orientations denoted by $\theta_{h,i,\lfloor k \rfloor}$, $\theta_{h,i,\lfloor k \rfloor+1}$, and $\hat{\theta}_{h,i,k}$ respectively. It should also be noted that $|\boldsymbol{p}^\top \mathcal{N}_{\mathrm{L}}(\boldsymbol{l})|$ is the orthogonal distance from point $\boldsymbol{p}$ to line $\boldsymbol{l}$.

The orientation of a line vector is not unique. For line vector $\boldsymbol{l}$, the orientation is given by θ, where the first two elements of $\mathcal{N}_{\mathrm{L}}(\boldsymbol{l})$ are $\sin \theta$ and $\cos \theta$ respectively. Note that $\mathcal{N}_{\mathrm{L}}(-\boldsymbol{l})$ represents the same line in 2D space, but has an orientation that differs from that of $\mathcal{N}_{\mathrm{L}}(\boldsymbol{l})$ by $\pm\pi$. Because of this ambiguity, there are two different possible interpolations of the lines $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$. Negating one of these lines will change which of the angles between the pair the interpolated line will pass through. If $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$ are precisely parallel, scaling one of them by -1 will change whether the interpolation lies between or outside the pair.

Since the epipolar lines $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$ are generated by a tracked moving point in consecutive frames, it is reasonable to assume that they should have similar orientations. Prior to computing the interpolation, one of the line vectors is negated if this reduces the difference in their orientations. This equates to negating one of $\boldsymbol{l}_{h,i,\lfloor k \rfloor}$ and $\boldsymbol{l}_{h,i,\lfloor k \rfloor+1}$ if the

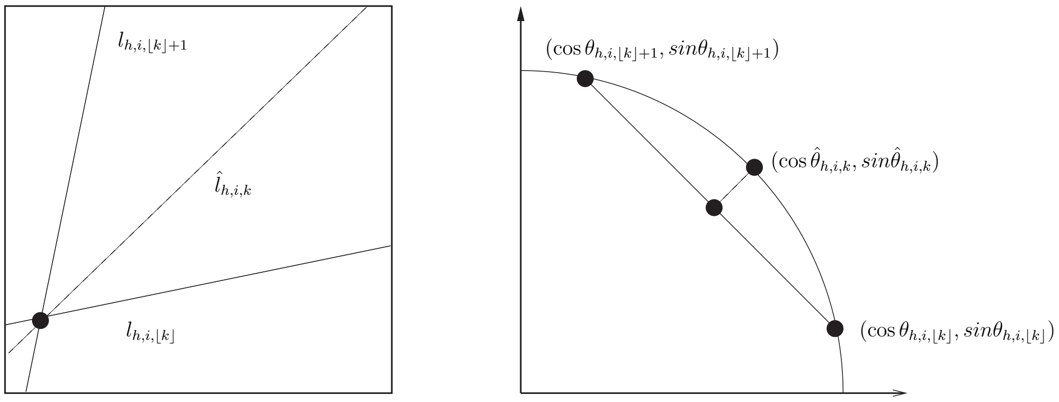

$l_{h,i,\lfloor k\rfloor+1}$
$\hat{l}_{h,i,k}$
$l_{h,i,\lfloor k\rfloor}$
$(\cos\theta_{h,i,\lfloor k\rfloor+1}, sin\theta_{h,i,\lfloor k\rfloor+1})$
$(\cos\hat{\theta}_{h,i,k}, sin\hat{\theta}_{h,i,k})$
$(\cos\theta_{h,i,\lfloor k\rfloor}, sin\theta_{h,i,\lfloor k\rfloor})$

Figure 3.3: Epipolar Line Interpolation

inner product of their first two elements is negative.

The function to interpolate two lines is therefore defined as

$$\mathcal{I}_{\mathrm{L}}(\boldsymbol{l}_0, \boldsymbol{l}_1, k) = \begin{cases} (1-k)\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0) + k\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_1) & : \quad [\boldsymbol{l}_0]_1[\boldsymbol{l}_1]_1 + [\boldsymbol{l}_0]_2[\boldsymbol{l}_1]_2 > 0 \\ (1-k)\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0) - k\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_1) & : \quad \text{otherwise} \end{cases} \tag{3.2}$$

Note that it would also be possible to interpolate two lines $\boldsymbol{l}_0$ and $\boldsymbol{l}_1$ by a linear interpolation of their orientations θ_0 and θ_1. Such a method however, does not uniquely define a line when $\boldsymbol{l}_0$ and $\boldsymbol{l}_1$ are parallel. The case of parallel lines would then have to be handled separately. For this reason, a linear interpolation of the normalised line vectors is preferable. Once an interpolated epipolar line is computed, the epipolar error in the corresponding frame can be measured, given by

$$\mathcal{E}_{\mathrm{ORTH-LI}}(\boldsymbol{p}, \boldsymbol{p}'_0, \mathsf{F}_0, \boldsymbol{p}'_1, \mathsf{F}_1, k) = \boldsymbol{p}^\top \mathcal{N}_{\mathrm{L}}(\mathcal{I}_{\mathrm{L}}(\mathsf{F}_0\boldsymbol{p}'_0, \mathsf{F}_1\boldsymbol{p}'_1, k)).$$

The resulting synchronisation cost function is

$$\begin{aligned} \mathcal{S}_{\mathrm{ORTH-LI}}(a,b) = {} & \frac{1}{\mathcal{W}_{\mathrm{I}}(a,b)} \sum_{h=1}^{m}\sum_{i=0}^{n-1} \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}_{h,i},\ \boldsymbol{p}'_{h,\lfloor a+bi\rfloor}, \mathsf{F}_{i,\lfloor a+bi\rfloor}, \\ & \qquad \boldsymbol{p}'_{h,\lfloor a+bi\rfloor+1}, \mathsf{F}_{i,\lfloor a+bi\rfloor+1}, \\ & \qquad a+bi-\lfloor a+bi\rfloor)^2 + \\ & \frac{1}{\mathcal{W}_{\mathrm{I}}(a,b)} \sum_{h=1}^{m}\sum_{j=0}^{n'-1} \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}'_{h,j},\ \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor}, \mathsf{F}^\top_{\lfloor a'+b'j\rfloor,j}, \\ & \qquad \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor+1}, \mathsf{F}^\top_{\lfloor a'+b'j\rfloor+1,j}, \\ & \qquad a'+b'j-\lfloor a'+b'j\rfloor)^2, \\ & \text{where } a' = -ab^{-1}, \quad b' = b^{-1}. \end{aligned}$$

To highlight the difference between $\mathcal{E}_{\mathrm{ORTH-LI}}$ and $\mathcal{E}_{\mathrm{ORTH-RI}}$, a noiseless synthetic case is illustrated in figure 3.4. Errors are measured for k which varies between the

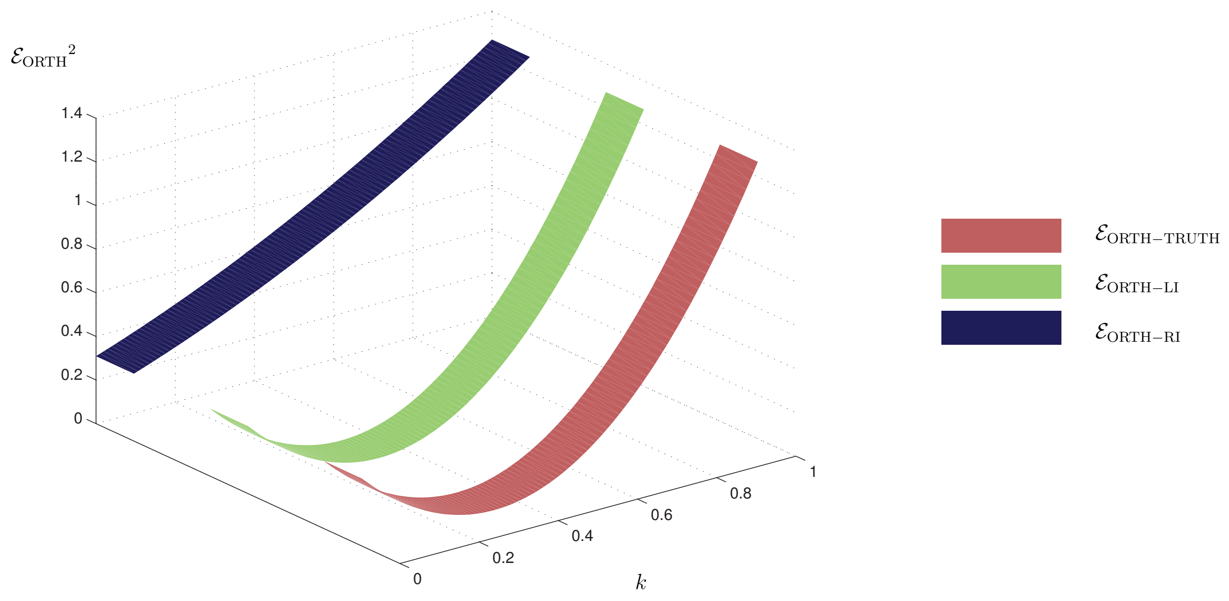


Figure 3.4: A Comparison of Interpolated Epipolar Errors

integer frame indices 0 and 1 from the first video sequence. Interpolated epipolar errors are measured in a frame from the second video sequence. These errors are compared against the true epipolar errors $\mathcal{E}_{\text{ORTH}-\text{TRUTH}}$ calculated from the known trajectories of the moving cameras and scene points. Note that $\mathcal{E}_{\text{ORTH}-\text{RI}}$ appears approximately linear, whereas $\mathcal{E}_{\text{ORTH}-\text{LI}}$ is almost indistinguishable from $\mathcal{E}_{\text{ORTH}-\text{TRUTH}}$, with a minimum value at approximately the same value of k. This demonstrates that, compared with $\mathcal{E}_{\text{ORTH}-\text{RI}}$, $\mathcal{E}_{\text{ORTH}-\text{LI}}$ can yield epipolar errors between integer frame indices which are closer to the true errors $\mathcal{E}_{\text{ORTH}-\text{TRUTH}}$.

3.3 Estimating the Frame Offset via an Exhaustive Search

In this section it is assumed that the frame rates of the cameras, or optionally just their ratio, are known. An initial estimate of the frame offset a can be found with a discrete uniform sampling over a constrained range of values. For each sample value of a, some cost function $\mathcal{S}_{COST}$ is evaluated. The value with the smallest associated cost is then used as the initial estimate of a, which can subsequently be refined by minimising the selected cost function with a minimiser such as Levenberg-Marquardt. An exhaustive search is necessary since the cost is a function of scene point and camera motions, which may be arbitrarily complex and give rise to multiple local minima.

The density of the sampling is chosen so as to test sample values of a that are a distance of $min(1, b)$ apart. Such a change in offset is equivalent to shifting the video sequence with the highest frame rate one frame forward in time. In some sense, such a

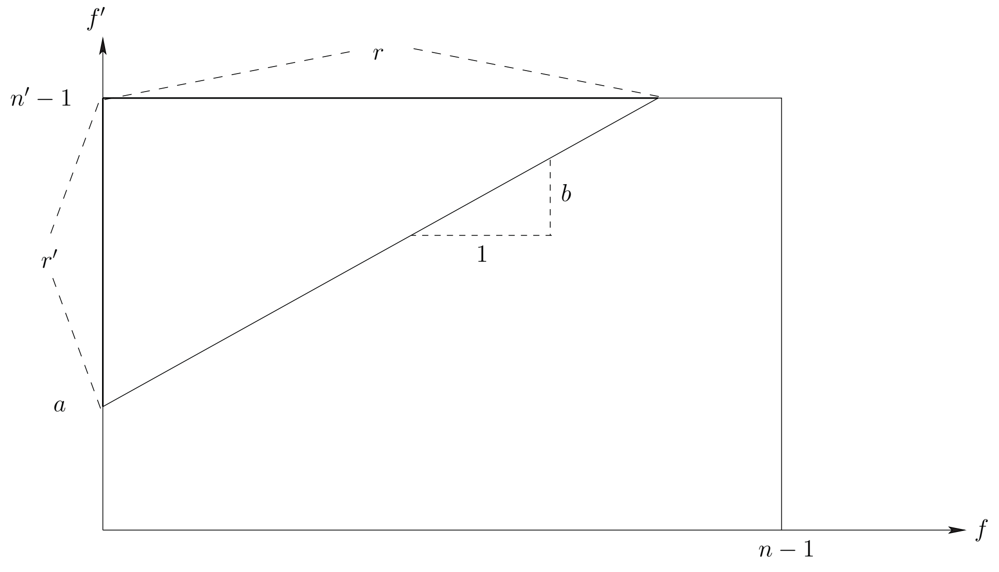


Figure 3.5: Ranges of frames in each video sequence captured while both cameras recorded

sampling will synchronise the video sequences at the frame level.

Each candidate sample of a, along with the known frame rate ratio b, defines a hypothesised temporal alignment of the video sequences. Accordingly, it defines a range of frames in each video sequence that correspond to the period of time that both cameras were recording. The lengths of these frame ranges are denoted (r, r') for the first and second video sequences, respectively, and are defined by the projection of the synchronisation line within a rectangle bounded by $(0, 0)$ and $(n-1, n'-1)$ onto each axis. An example of this is illustrated in figure 3.5. The search for the frame offset a will be restricted to values such that (r, r') are above pre-chosen thresholds (r_{min}, r'_{min}), indicating the video sequences have a significant overlap in time. These thresholds allow the user to make use of prior knowledge of the video sequences to constrain the search range of a. For example, if we choose $r_{min} = \frac{1}{2}n$, and $r'_{min} = \frac{1}{2}n'$, the search for offset a will be restricted to values indicating that neither camera was recording for longer than twice the time that both were recording. If the thresholds are both set to 0, every frame offset indicating at least one frame of overlap will be tested.

The frame range sizes (r, r') can be expressed in terms of parameters (a, b). A frame offset of 0 indicates that the cameras started recording at the same point in time. We define a_{end} as the frame offset indicating that the cameras finished recording at the same point in time, and compute it using the equation

$$a_{end} = (n'-1) - b(n-1). \tag{3.3}$$

Each frame range size remains constant for frame offsets between 0 and a_{end}, but they decrease linearly for estimates of a outside of these values. Specifically, if $a_{end} > 0$, the formulae for (r, r') are

$$r' = \begin{cases} 0 & : \quad a < -b(n-1) \\ a + b(n-1) & : \quad -b(n-1) \le a < 0 \\ b(n-1) & : \quad 0 \le a < a_{end} \\ n' - 1 - a & : \quad a_{end} \le a < n' - 1 \\ 0 & : \quad a \ge n' - 1 \end{cases} \quad , r = \frac{r'}{b}$$

Note that (r, r') have maximum values of $(n-1, b(n-1))$. The thresholds (r_{min}, r'_{min}) must be less than or equal to these values, or else the search range for a will be empty. If $a_{end} \le 0$, then (r, r') are given by

$$r' = \begin{cases} 0 & : \quad a < -b(n-1) \\ a + b(n-1) & : \quad -b(n-1) \le a < a_{end} \\ n' - 1 & : \quad a_{end} \le a < 0 \\ n' - 1 - a & : \quad 0 \le a < n' - 1 \\ 0 & : \quad a \ge n' - 1 \end{cases} \quad , r = \frac{r'}{b}$$

Analogously to the previous case, (r, r') have maximum values of $(\frac{n'-1}{b}, n'-1)$. The pair (r_{min}, r'_{min}) must be chosen to be less than or equal to these maxima.

Once appropriate (r_{min}, r'_{min}) have been chosen, the range of a satisfying each of the constraints $(r \ge r_{min})$ and $(r' \ge r'_{min})$ can be computed. The innermost of these two ranges specifies the constrained search range for frame offset a. The process is illustrated for $(a_{end} > 0)$ in figure 3.6. Specifically, the constrained range of frame offset a is

$$\begin{aligned} a_{min} &= max\{br_{min} - b(n-1), r'_{min} - b(n-1)\}, \\ a_{max} &= min\{(n'-1) - br_{min}, (n'-1) - r'_{min}\}. \end{aligned} \tag{3.4}$$

With a sample rate and search range now defined for offset a, the estimate that minimises the cost function is found. The entire process is summarised in algorithm 3.1.

Once the initial estimate is available, it can be refined to achieve synchronisation with accuracy to within a fraction of a frame, by using the chosen cost function $\mathcal{S}_{COST}$ with a minimiser such as Levenberg-Marquardt.

3.4 Estimating Both Synchronisation Parameters via an Exhaustive Search

If the frame rates of the cameras are unknown, both (a, b) must be estimated. A uniform discrete sampling over (a, b) is inappropriate, since b may theoretically increase without bound. Choosing upper and lower bounds for the sampling is therefore problematic. Furthermore, since b is a *ratio*, it makes sense to consider reciprocal values. If $\hat{b}$ is one value sampled, $\hat{b}^{-1}$ should be considered as well. A uniform sampling does not permit this.

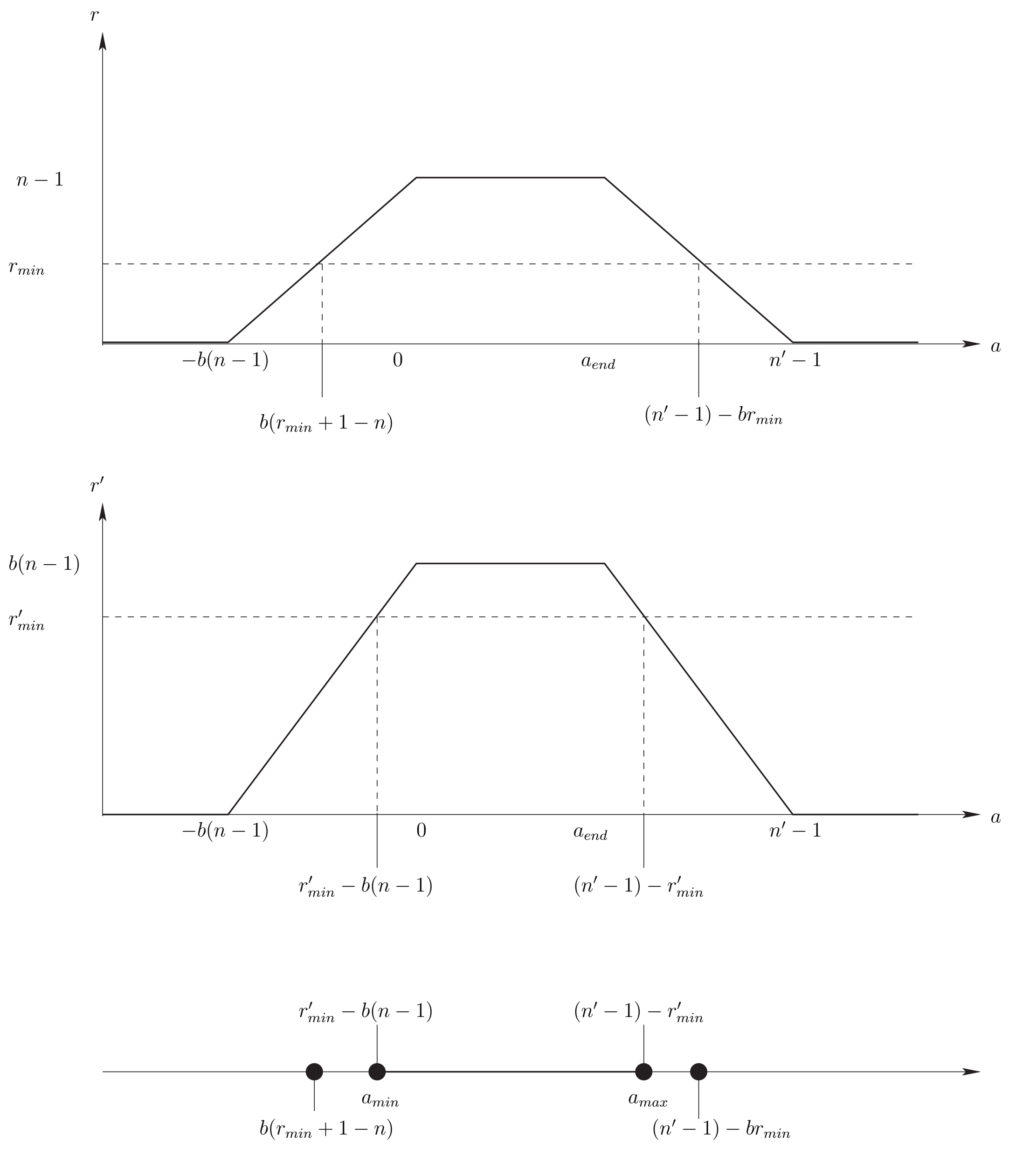


Figure 3.6: Frame ranges for different values of offset a ($a_{end} > 0$)

Algorithm 3.1 (Estimation of a with an Exhaustive Search)

```
Compute $a_{end}$ as defined in equation 3.3
Choose
    $$0 \leq r_{min} \leq \begin{cases} n-1 & : \quad a_{end} > 0 \\ \frac{n'-1}{b} & : \quad \text{otherwise} \end{cases}$$
And
    $$0 \leq r'_{min} \leq \begin{cases} b(n-1) & : \quad a_{end} > 0 \\ n'-1 & : \quad \text{otherwise} \end{cases}$$
Compute $(a_{min}, a_{max})$ as defined in equation 3.4
$\hat{a} \leftarrow a_{min}$
$\hat{a}_{best} \leftarrow a_{min}$
while $\hat{a} \leq a_{max}$ do
    if $\mathcal{S}_{COST}(\hat{a}, b) < \mathcal{S}_{COST}(\hat{a}_{best}, b)$ then
        $\hat{a}_{best} \leftarrow \hat{a}$
    end if
    $\hat{a} \leftarrow \hat{a} + min(1, b)$
end while
$a \leftarrow \hat{a}_{best}$
```

An alternative parameterisation for the line defined by (a, b) is given by

$$\alpha = \frac{-a}{b+1}, \quad \beta = \frac{n + n' - 2 - a}{b+1}.$$

The parameter α specifies the intersection of the line defined by (a, b) with a line of gradient -1 passing through the origin. Similarly, β defines the location of the intersection with a line of gradient -1 passing through the point $(n-1, n'-1)$. This is illustrated in figure 3.7.

This parameterisation is convenient since it places simple bounds on the values of α and β. The value of α must be in the range of $(1-n', n-1)$, and β must lie between $(0, n'+n-2)$. A uniform sampling over all integer values of (α, β) in these ranges has another convenient property. The set of temporal alignments sampled is equivalent, regardless of which video sequence is chosen to be the first, with frame index f, and which is chosen to be the second. Furthermore, by defining β as shown in figure 3.7, a similar equivalence occurs even if the order of frames in each video sequence is reversed.

Given values for α and β, the original parameters a and b can be recovered as

$$a = -\alpha\frac{n + n' - 2}{\beta - \alpha}, \quad b = \frac{(n + n' - 2) - (\beta - \alpha)}{\beta - \alpha}. \tag{3.5}$$

To fit the requirement of a positive frame rate ratio, we require $(\alpha \leq \beta \leq n + n' - 2 + \alpha)$.

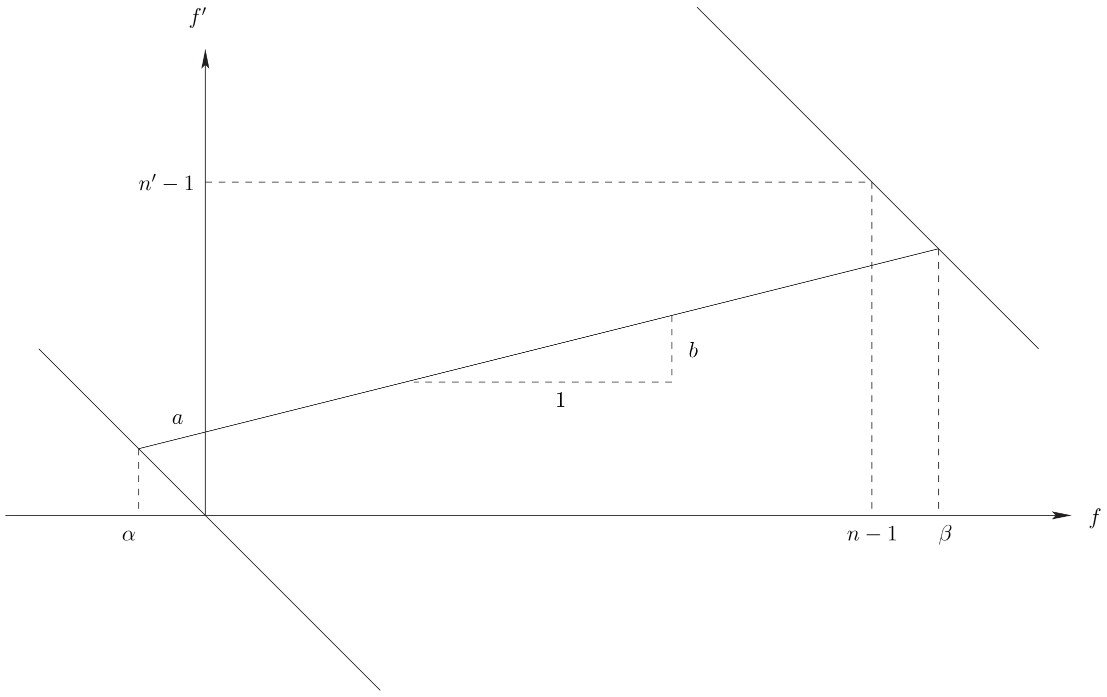


Figure 3.7: Line parameterisation (α, β).

From equation 3.5, it can be seen that a uniform search across integer values of α and β will sample reciprocal frame rate ratios. Consider the expression for b in equation 3.5, for some integer values of α and β. An estimate of synchronisation (α', β') for which

$$(\beta' - \alpha') = (n + n' - 2) - (\beta - \alpha)$$

will equate to a frame rate ratio of b^{-1}. Note that for integer values of α and β, there exist integer values of α' and β' satisfying this relation.

As in the case for a known frame rate ratio, the frame ranges (r, r') can be expressed in terms of the synchronisation parameters. For $\alpha > 0$,

$$r = \tfrac{\beta-\alpha}{g+\alpha-\beta} r'.$$

$$r' = \begin{cases} 0 & : \ \beta < \alpha \\ n' - 1 & : \ \alpha \le \beta \le (n-1) \\ n' - 1 + \frac{g(n-1-\beta)}{\beta-\alpha} & : \ (n-1) < \beta \le (g - \frac{n-1}{n'-1}\alpha) \\ 0 & : \ \beta > (g - \frac{n-1}{n'-1}\alpha) \end{cases},$$

$$\text{where} \quad g = n + n' - 2.$$

Similarly, for $\alpha < 0$,

$$r = \begin{cases} 0 & : \beta < (-\frac{n-1}{n'-1}\alpha) \\ n-1-\frac{g(n-1-\beta)}{g+\alpha-\beta} & : (-\frac{n-1}{n'-1}\alpha) \leq \beta < (n-1) \\ n-1 & : (n-1) \leq \beta \leq (g+\alpha) \\ 0 & : (g+\alpha) < \beta \end{cases},$$

$$r' = \tfrac{g+\alpha-\beta}{\beta-\alpha}r,$$

$$\text{where} \quad g = n + n' - 2.$$

Finally, for $\alpha = 0$,

$$r = \begin{cases} 0 & : \beta < 0 \\ \frac{(n'-1)\beta}{g-\beta} & : 0 \leq \beta < (n-1) \\ n-1 & : n-1 \leq \beta \leq g \\ 0 & : \beta > g \end{cases},$$

$$r' = \begin{cases} 0 & : \beta < 0 \\ n'-1 & : \beta \leq (n-1) \\ \frac{(n-1)(g-\beta)}{\beta} & : (n-1) \leq \beta \leq g \\ 0 & : \beta > g \end{cases},$$

$$\text{where} \quad g = n + n' - 2.$$

Unlike the case for a known frame rate ratio, we can always choose any threshold r_{min} in the range of $[0, n-1]$, and likewise any r'_{min} in $[0, n'-1]$. Since both parameters (a, b) are unknown, an estimate can always be found that specifies the video sequences have a complete overlap in time. Once these thresholds have been chosen, the corresponding search ranges for α and β can be calculated.

For a positive α, some value of β can always be chosen such that $r' = (n'-1)$. Consequently, the upper bound for α is only dependent on r_{min}. Similarly, the lower bound for α is only dependent on r'_{min}. For any value of α, the frame range sizes r and r' attain their maximum at $\beta = (n-1)$. Consequently, the upper bound for α is the value at which $r = r_{min}$ for $\beta = (n-1)$. The lower bound for α is the value at which $r' = r'_{min}$ for $\beta = (n-1)$. This equates to a search range for α defined by

$$\begin{aligned} \alpha_{min} &= -\frac{(n-1)(n'-1-r'_{min})}{r'_{min}+n-1}, \\ \alpha_{max} &= \frac{(n'-1)(n-1-r_{min})}{r_{min}+n'-1}. \end{aligned} \tag{3.6}$$

For every candidate value $\hat{\alpha}$ within this range, we need to compute the corresponding search range of β. The lower bound of β is given by

$$\begin{aligned} \beta_{min} &= \begin{cases} \beta_{min-1} & : \quad \hat{\alpha} \geq 0 \\ max\{\beta_{min-2}, \beta_{min-3}\} & : \quad \hat{\alpha} < 0 \end{cases}, \\ \text{where} \quad \beta_{min-1} &= \hat{\alpha} + \tfrac{g r_{min}}{n'-1+r_{min}}, \\ \beta_{min-2} &= \hat{\alpha} + \tfrac{g(r_{min}-\hat{\alpha})}{n'-1+r_{min}}, \\ \beta_{min-3} &= \hat{\alpha} - \tfrac{g\hat{\alpha}}{n'-1-r'_{min}}, \\ g &= n + n' - 2. \end{aligned} \tag{3.7}$$

Similarly, the upper bound for β is

$$\begin{aligned} \beta_{max} &= \begin{cases} \beta_{max-1} & : \quad \hat{\alpha} \leq 0 \\ min\{\beta_{max-2}, \beta_{max-3}\} & : \quad \hat{\alpha} > 0 \end{cases}, \\ \text{where} \quad \beta_{max-1} &= \hat{\alpha} + g - \tfrac{r'_{min} g}{r'_{min}+n-1}, \\ \beta_{max-2} &= \hat{\alpha} + g - \tfrac{g\hat{\alpha}}{n-1-r_{min}}, \\ \beta_{max-3} &= \hat{\alpha} + g - \tfrac{g(\hat{\alpha}+r'_{min})}{r'_{min}+n-1}, \\ g &= n + n' - 2. \end{aligned} \tag{3.8}$$

Having defined the ranges for both parameters that produce (r, r') greater than or equal to the minimum thresholds, the search to find the best estimates of α and β can now be conducted. Note that the ranges for α and β admit frame rate ratios of 0 or infinity if $r_{min} = 0$, or $r'_{min} = 0$. This is reasonable since we are only searching for an *initial* estimate of the parameters (a, b). Given synchronisation cost function $\mathcal{S}_{COST}(a, b)$, we define $\mathcal{S}_{COST^*}(\alpha, \beta)$ as the corresponding cost function that assesses parameters in the space of (α, β). The resulting search is summarised in algorithm 3.2.

As in the case of a known frame rate ratio, the initial estimate can be refined by applying a minimiser such as Levenberg-Marquardt to the relevant cost function.

3.5 Synthetic Tests

This section presents a comparison of the cost functions described in section 3.1, and assesses the performance of algorithms 3.1 and 3.2 which estimate synchronisation parameters via a uniform search of the parameter space.

3.5.1 Test Configurations

The algorithms are assessed for various synchronisation parameters. Cameras orbit a unit sphere at different elevations, with optical centre paths shown in figure 3.8. The cameras move along the illustrated trajectories at constant speed, maintaining a distance of 2.25 from the vertical axis of the sphere, and rotating so that the optical axis of each frame intersects the centre of the sphere.

Algorithm 3.2 (Estimation of (α, β) with an Exhaustive Search)

```
Choose $r_{min}$ such that $0 \leq r_{min} \leq n - 1$
And $r'_{min}$ such that $0 \leq r'_{min} \leq n' - 1$
Compute $(\alpha_{min}, \alpha_{max})$ as defined in equation 3.6
$(\hat{\alpha}_{best}, \hat{\beta}_{best}) \leftarrow (\alpha_{min}, n - 1)$
$\hat{\alpha} \leftarrow \alpha_{min}$
while $\hat{\alpha} \leq \alpha_{max}$ do
  Compute $(\beta_{min}, \beta_{max})$ as defined in equations 3.7 and 3.8
  $\hat{\beta} \leftarrow \beta_{min}$
  while $\hat{\beta} \leq \beta_{max}$ do
    if $\mathcal{S}_{COST^*}(\hat{\alpha}, \hat{\beta}) < \mathcal{S}_{COST^*}(\hat{\alpha}_{best}, \hat{\beta}_{best})$ then
      $(\hat{\alpha}_{best}, \hat{\beta}_{best}) \leftarrow (\hat{\alpha}, \hat{\beta})$
    end if
    $\hat{\beta} \leftarrow \hat{\beta} + 1$
  end while
  $\hat{\alpha} \leftarrow \hat{\alpha} + 1$
end while
$(\alpha, \beta) \leftarrow (\hat{\alpha}_{best}, \hat{\beta}_{best})$
```

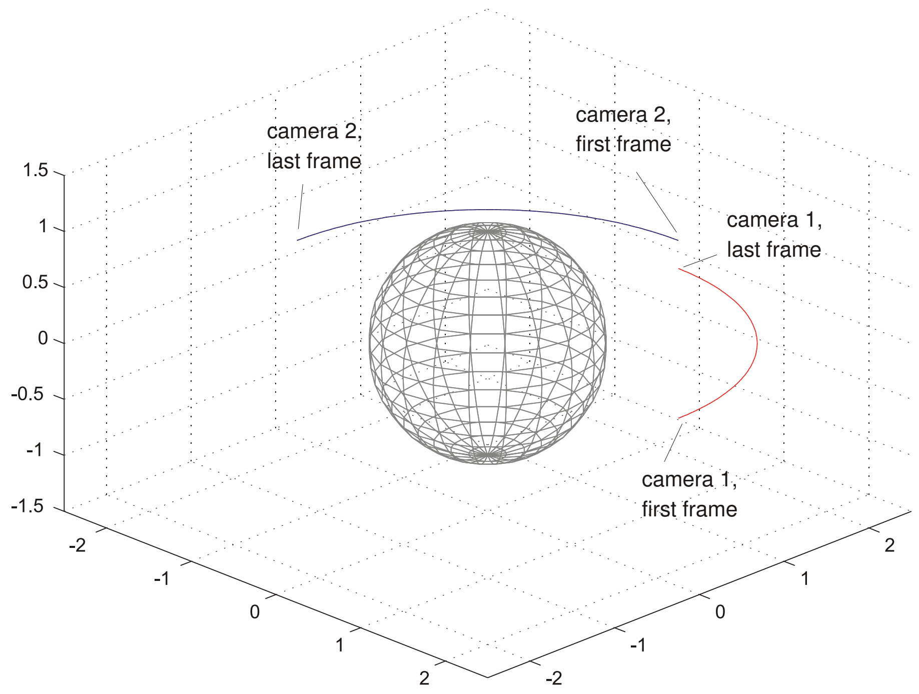


Figure 3.8: Spatial configuration for synthetic synchronisation tests

In each test, 100 stationary scene points are randomly chosen within the unit sphere. These scene points are projected to each video frame, for an image size of 500×500, and Gaussian noise is added to each resulting location. The standard deviation of the noise is chosen such that the average squared distance to the true location is one pixel. Linear trajectories are chosen at random for the moving scene points, each having a length between 1 and 2. These moving scene points are also projected to the video frames, and as with the projections of stationary scene points, Gaussian noise is added to each 2D location.

The projection matrices are precomputed by forming an initial reconstruction of the stationary point cloud using just 2 frames. This point cloud is used to estimate a larger subset of projection matrices for both video sequences. The subset of projection matrices and the 3D points are then refined using bundle adjustment with sparse methods as described in [23]. Finally, the remaining projection matrices are computed, and Resection-Intersection [54] is used, whereby the sets of projection matrices and 3D scene points are iteratively refined.

Each of the temporal configurations is tested 1000 times, with new stationary and moving scene points chosen randomly for each test.

3.5.2 Initial Estimates of Synchronisation Parameters

Algorithms 3.1 and 3.2 provide initial estimates of the synchronisation parameters, for known and unknown frame rate ratios respectively. We begin by assessing their performance across a variety of temporal configurations, using different cost functions defined earlier. To assess *initial* estimates, errors are measured in the space of the parameters over which the search is conducted. In the known frame rate ratio case, algorithm 3.1 returns an estimate of the frame offset, $\hat{a}$. Denoting the true frame offset as $\bar{a}$, the error measure used to rate the performance of this search is

$$E_a = \frac{|\hat{a} - \bar{a}|}{min(b, 1)}.$$

Note that since algorithm 3.1 uniformly samples frame offsets distance $min(b, 1)$ apart, E_a measures the distance from the initial estimate to the true frame offset $\bar{a}$ in units of the sampling rate. The purpose of measuring this error for an initial estimate is *not* to demonstrate a highly accurate level of synchronisation. Rather, a small E_a indicates that, of all frame offsets values sampled during the search, the one with the lowest associated error is adjacent, or at least nearby, to the sample closest to $\bar{a}$.

For an unknown frame rate ratio, algorithm 3.2 provides initial estimates $\hat{\alpha}$ and $\hat{\beta}$. Accordingly, the error measure used to assess the performance of this search is

$$E_{\alpha,\beta} = \sqrt{(\hat{\alpha} - \bar{\alpha})^2 + (\hat{\beta} - \bar{\beta})^2},$$

Setup				Percentage of tests where $E_a < 2$			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\text{ORTH-RI}}$	$\mathcal{S}_{\text{SYMM-RI}}$	$\mathcal{S}_{\text{SAMP-RI}}$	$\mathcal{S}_{\text{ORTH-LI}}$
80	100	10.63	1.1875	99.6	99.5	99.6	99.5
80	100	42.3	1.1875	97.3	97.3	97.3	97.3
20	100	10.63	4.9375	99.3	99.4	99.4	99.3

Setup				Percentage of tests where $E_{\alpha,\beta} < 2$			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\text{ORTH-RI}}$	$\mathcal{S}_{\text{SYMM-RI}}$	$\mathcal{S}_{\text{SAMP-RI}}$	$\mathcal{S}_{\text{ORTH-LI}}$
80	100	10.63	1.1875	97.8	97.7	97.6	97.7
80	100	42.3	1.1875	92.3	92.4	92.5	92.4
20	100	10.63	4.9375	98.9	99.0	98.9	99.0

Table 3.1: Synthetic initial estimate performance for a single moving scene point

where $\bar{\alpha}$, and $\bar{\beta}$ are the true values of the parameters being estimated. Note that measuring a 2D distance in the parameter space makes sense since algorithm 3.2 performs a uniform search in α and β, sampling values at distance 1 apart in either parameter.

The results are given in table 3.1, for synchronisation estimates obtained using just a single moving scene point. The percentages of tests are shown for which a search across parameter space yields $E_a < 2$ and $E_{\alpha,\beta} < 2$. This is the lowest integer value such that a significant majority of tests (above 90%) satisfy the error threshold in all cases. Note that for each temporal configuration, a similar percentage of tests achieve this threshold regardless of the cost function used. This demonstrates that, out of all the synchronisation parameters sampled by a uniform search, one of the closer samples to truth is typically identified as the initial estimate.

3.5.3 Video Synchronisation Error

Each of the algorithms 3.1 and 3.2 provide an estimate of the synchronisation parameters, denoted $(\hat{a}, \hat{b})$. Such estimates can subsequently be refined using Levenberg-Marquardt and any of the previously used cost functions. The result is assessed by examining the accuracy of the synchronisation for each frame in the two video sequences. The error in synchronised frame indices is therefore computed as follows. For frame i in video sequence 1, $(\bar{a} + \bar{b}i)$ and $(\hat{a} + \hat{b}i)$ are the true and estimated synchronous frame indices from video sequence 2 respectively. The absolute difference between these two values, computed as $|(\bar{a} + \bar{b}i) - (\hat{a} + \hat{b}i)|$, measures how well frame i has been synchronised and is denoted the *frame synchronisation error*. Similarly, frame j in video 2 has a frame synchronisation error given by $|(\bar{a}' + \bar{b}'j) - (\hat{a}' + \hat{b}'j)|$, where $(\bar{a}', \bar{b}')$ and $(\hat{a}', \hat{b}')$ provide the inverse lines of the true and estimated synchronisation parameters respectively. The quality of the estimated synchronisation parameters is assessed by finding the maximum of these frame synchronisation errors, for all frame indices (not just integers) within the period when either the true or estimated parameters indicate that both cameras were recording. This corresponds to a range of frames in video sequence 1 $[i_{min}, i_{max}]$, and a range of frame indices in video sequence 2 $[j_{min}, j_{max}]$, given by

$$\begin{aligned}
i_{min} &= max\{0, min\{-\hat{a}\hat{b}^{-1}, -\bar{a}\bar{b}^{-1}\}\},\\
i_{max} &= min\{n-1, max\{(n'-1-\hat{a})\hat{b}^{-1}, (n'-1-\bar{a})\bar{b}^{-1}\}\},\\
j_{min} &= max\{0, min\{\bar{a}, \hat{a}\}\},\\
j_{max} &= min\{n'-1, max\{\bar{a}+\bar{b}(n-1), \hat{a}+\hat{b}(n'-1)\}\}.
\end{aligned}$$

Note that, since $(\hat{a}, \hat{b})$ and $(\bar{a}, \bar{b})$ specify lines in the space of the frame indices of the two cameras, the maximum frame synchronisation errors within these constraints will always be found at the boundaries of the ranges. The maximum such error is denoted the *video synchronisation error* (VSE), and is computed as

$$\begin{aligned}
VSE(\bar{a}, \bar{b}, \hat{a}, \hat{b}) = \max\{&|(\bar{a}+\bar{b}i_{min}) - (\hat{a}+\hat{b}i_{min})|,\\
&|(\bar{a}+\bar{b}i_{max}) - (\hat{a}+\hat{b}i_{max})|,\\
&|(\bar{a}'+\bar{b}'j_{min}) - (\hat{a}'+\hat{b}'j_{min})|,\\
&|(\bar{a}'+\bar{b}'j_{max}) - (\hat{a}'+\hat{b}'j_{max})|\}.
\end{aligned}$$

This error therefore represents the *worst case* when trying to identify synchronous frame pairs from the two video sequences.

The results are shown in table 3.2, for synchronisation parameters estimated using just one moving scene point. Each entry corresponds to the median of VSE measures across 1000 tests, for a particular temporal configuration and cost function. Table 3.3 displays the percentage of tests where the VSE measure was less than 0.5. For all frames captured while both cameras were recording, a synchronisation estimate within this threshold will correctly identify the closest synchronous frame from the other video sequence.

All cost functions exhibit a reasonable performance, often achieving synchronisation accurate to within a fraction of a frame, with a greater than 60% probability of meeting the criterion for success in all cases. Note that in all but one case, the cost function $\mathcal{S}_{\text{ORTH-LI}}$ has the best median performance. Additionally, figure 3.9 shows the proportion of tests for which each cost function achieved the best synchronisation estimate (lowest VSE measure). It is clear that in all configurations, $\mathcal{S}_{\text{ORTH-LI}}$ achieved the best synchronisation more frequently that the other cost functions. This affirms that epipolar line interpolation can provide better synchronisation estimates than residual interpolation, even for a more sophisticated measure such as Sampson error. For this reason, and the theoretical issues illustrated in example 3.2, the remainder of this section will focus on the performance of $\mathcal{S}_{\text{ORTH-LI}}$.

Setup				Median of VSE measures (Known b)			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\mathrm{ORTH-RI}}$	$\mathcal{S}_{\mathrm{SYMM-RI}}$	$\mathcal{S}_{\mathrm{SAMP-RI}}$	$\mathcal{S}_{\mathrm{ORTH-LI}}$
80	100	10.63	1.1875	0.058	0.057	0.057	0.064
80	100	42.3	1.1875	0.103	0.101	0.102	0.096
20	100	10.63	4.9375	0.1	0.09	0.092	0.089

Setup				Median of VSE Measures (Unknown b)			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\mathrm{ORTH-RI}}$	$\mathcal{S}_{\mathrm{SYMM-RI}}$	$\mathcal{S}_{\mathrm{SAMP-RI}}$	$\mathcal{S}_{\mathrm{ORTH-LI}}$
80	100	10.63	1.1875	0.186	0.186	0.184	0.173
80	100	42.3	1.1875	0.3	0.3	0.3	0.284
20	100	10.63	4.9375	0.37	0.352	0.349	0.248

Table 3.2: VSE measures for a single moving scene point

Setup				Percentage of tests where $VSE < 0.5$ (Known b)			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\mathrm{ORTH-RI}}$	$\mathcal{S}_{\mathrm{SYMM-RI}}$	$\mathcal{S}_{\mathrm{SAMP-RI}}$	$\mathcal{S}_{\mathrm{ORTH-LI}}$
80	100	10.63	1.1875	92.5	92.8	92.7	92.2
80	100	42.3	1.1875	86.5	86.5	86.5	86.7
20	100	10.63	4.9375	91.3	91.7	91.7	91.8

Setup				Percentage of tests where $VSE < 0.5$ (Unknown b)			
n	n'	$\bar{a}$	$\bar{b}$	$\mathcal{S}_{\mathrm{ORTH-RI}}$	$\mathcal{S}_{\mathrm{SYMM-RI}}$	$\mathcal{S}_{\mathrm{SAMP-RI}}$	$\mathcal{S}_{\mathrm{ORTH-LI}}$
80	100	10.63	1.1875	75.8	75.8	76.6	74.4
80	100	42.3	1.1875	66.5	66.8	66.4	68.5
20	100	10.63	4.9375	62.1	62.9	62.8	70.4

Table 3.3: Percentage of successful synchronisations for a single moving scene point

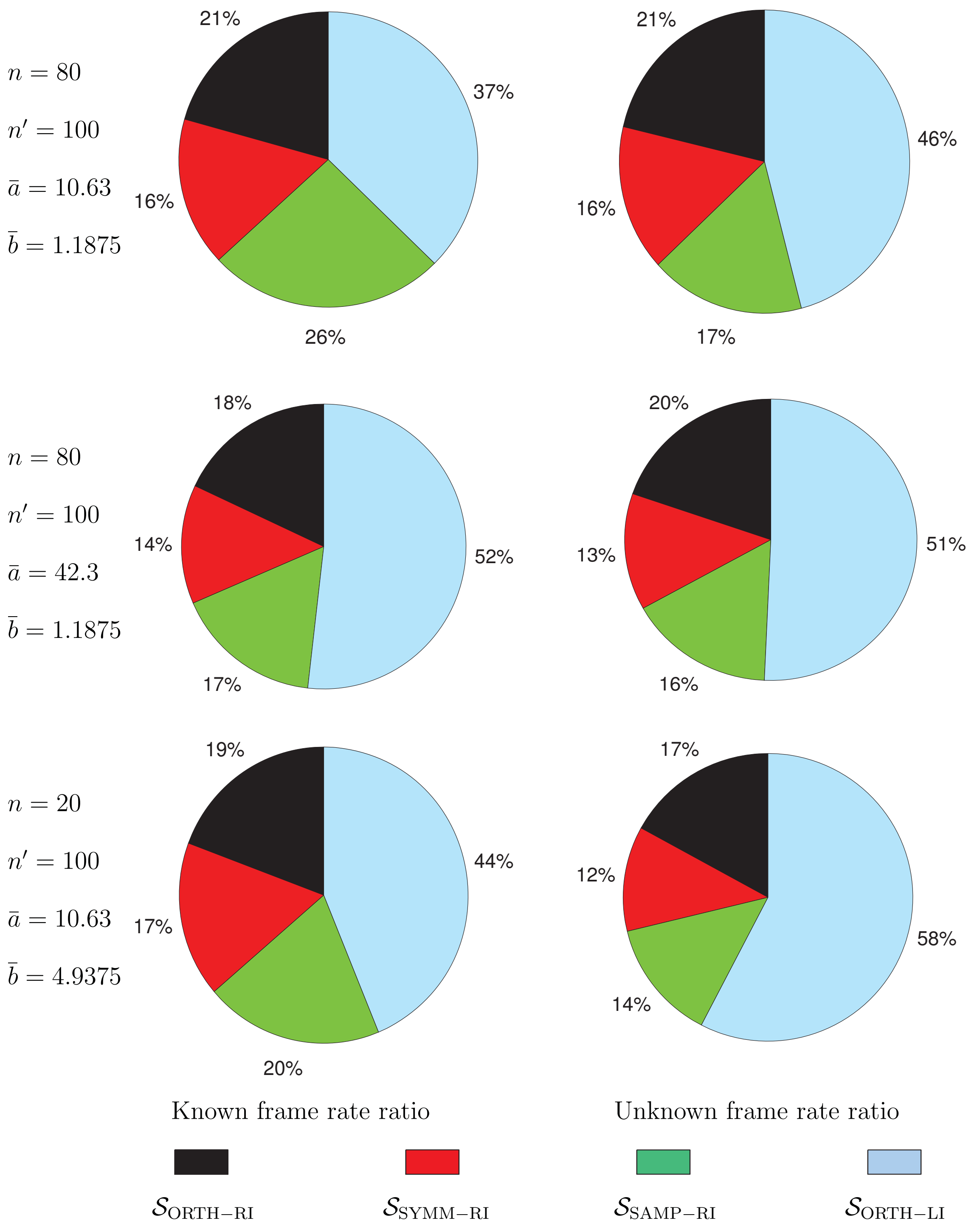


Figure 3.9: Proportions of tests in which each cost function achieved the best synchronisation

Figure 3.10 shows the VSE measures for $\mathcal{S}_{\mathrm{ORTH-LI}}$ for various numbers of moving scene points. Note that as m increases, the median VSE decreases, but with diminishing returns. In all configurations, even for $m = 5$, the median VSE is below 5% of a frame if b is known, and below 10% otherwise. The number of successful synchronisation tests achieving a VSE less than 0.5 can also be seen to rapidly approach 100% as m increases.

These results also show that for given video sequence lengths (measured as the number of frames), better synchronisation is possible if the video sequences have a larger overlap in time. In this case, more frames provide visual cues to refine the synchronisation estimate. It should be noted that the first case, with $a = 10.63$, $b = 1.1875$, and the third case, with $a = 10.63$, $b = 4.9375$, have been devised to have identical points in time at which the cameras start and stop recording. The decrease in performance for the third case must therefore be due to the fact that less frames are present in the first video sequence, and that a larger frame rate ratio produces less accurate epipolar line interpolations.

Note that the previous experiments all test a single class of scene point motion. Points move linearly at constant speed. To demonstrate the applicability of these methods to other scenarios, the tests are repeated for points with a piece-wise linear motion. At some randomly chosen point in time when either camera is recording, the scene point changes direction. The results are shown in figure 3.11. Note that the median VSE measures are very similar to the linear motion case, but that the number of tests for which the VSE is less than 0.5 increases in all cases where $m = 1$. Moving scene points have more complicated motions, providing additional visual cues for synchronisation and helping to avoid near-degenerate cases.

3.6 Real Video Sequence Pair Tests

Algorithms 3.1 and 3.2 were tested on a real video sequence pair, using the cost function $\mathcal{S}_{\mathrm{ORTH-LI}}$. Two video sequences of a Lego robot were recorded by moving hand-held cameras. Projection matrix estimates and moving point trajectories were obtained by using the ICARUS software [19]. The two projection matrix sets were then registered by manually identifying corresponding scene points, and robustly estimating a 4×4 projective transformation to spatially align them. A manual estimate of synchronisation is provided by identifying frame pairs depicting particular events. The average of frame offset values specified by each such frame pair provides the manual synchronisation estimate.

Assuming the manual synchronisation provides truth, VSE measures are computed for both known and unknown frame rate ratio searches. The results for the Lego robot sequence are given in table 3.4. Results for the 'initial' step are the approximate estimates given by the discrete parameter search. The 'final' results are for synchronisation estimates after a post-search minimisation with Levenberg-Marquardt. Note that, for the case where b is known, the initial synchronisation estimate has a VSE of precisely 0. This is due to the manual estimate of the frame offset a equating to one of the values sampled during the search, which coincidentally occurred due to the choice of minimum

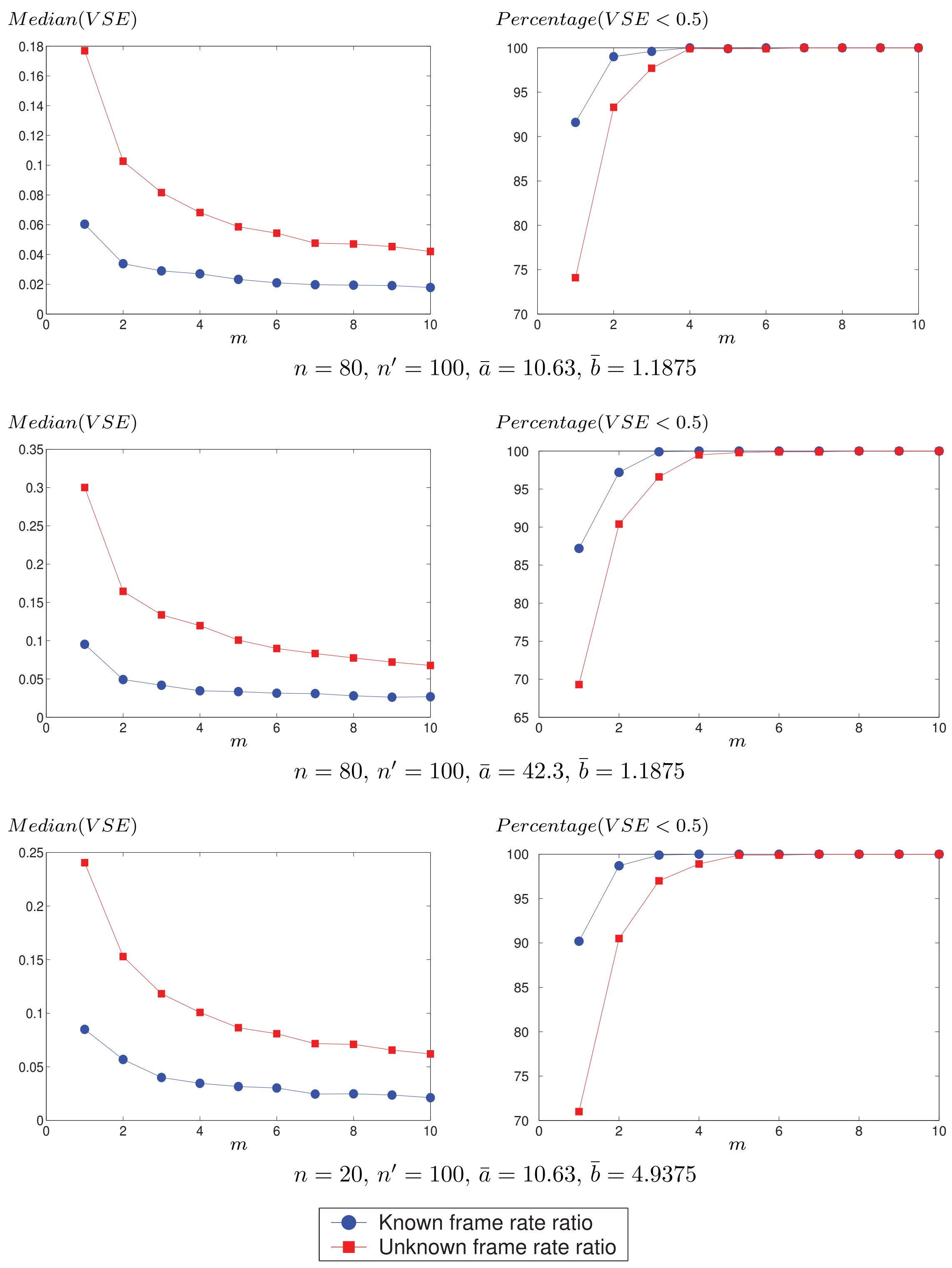


Figure 3.10: VSE results for synthetic test cases synchronised by a search of the parameter space, using scene points with linear motion

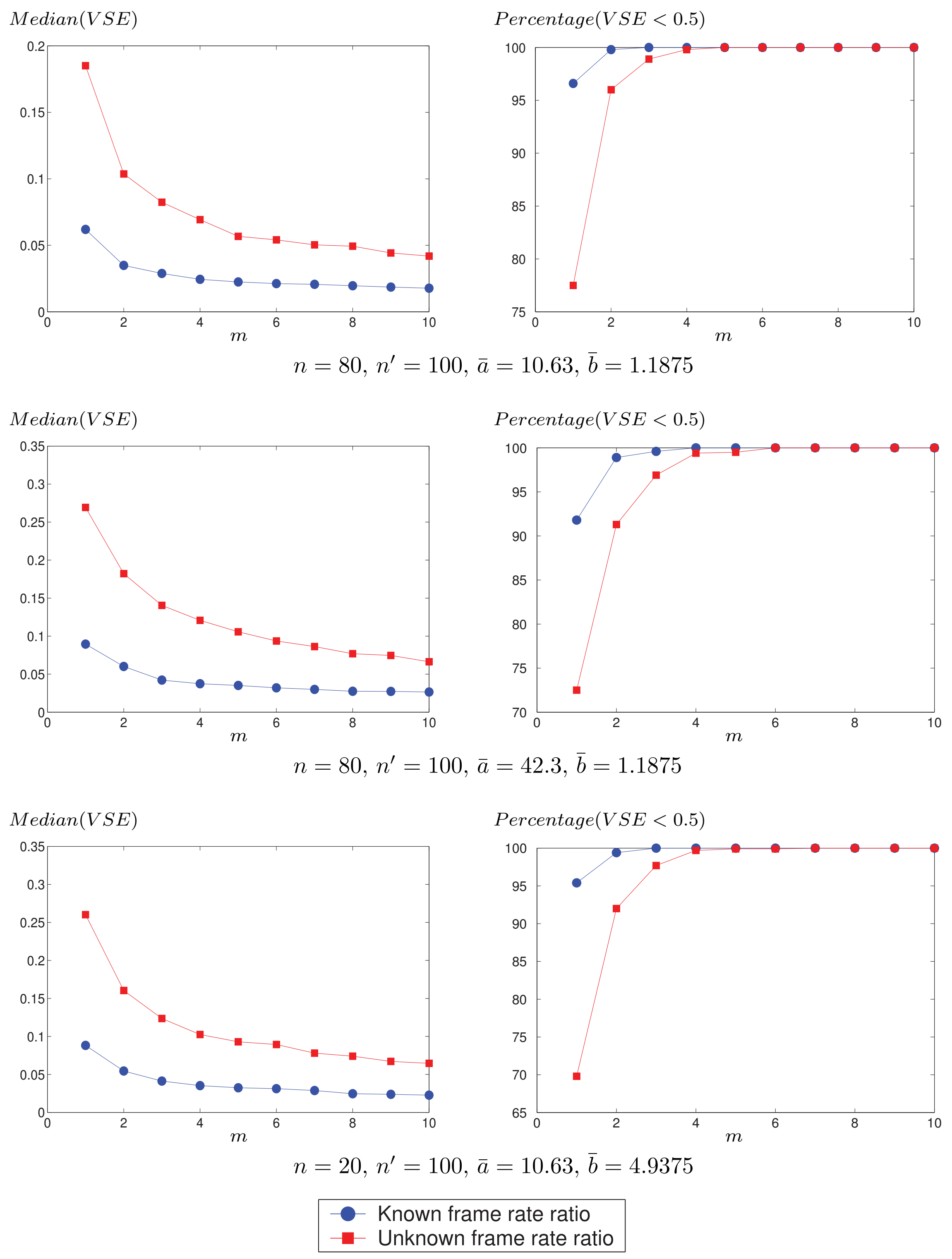


Figure 3.11: VSE results for synthetic test cases synchronised by a search of the parameter space, using scene points with piecewise-linear motion

Setup: $\bar{a} = -8.1$, $\bar{b} = 0.6$, $n = 201$, $n' = 136$				
Method	Step	$\hat{a}$	$\hat{b}$	VSE
b known	initial	-8.1	-	0
	final	-8.032	-	0.113
b unknown	initial	-8.887	0.606	1.158
	final	-8.214	0.601	0.362

Table 3.4: VSE measures for the synchronisation of the Lego robot sequences via a search of the parameter space

frame range sizes (r_{min}, r'_{min}), as described in section 3.3. In both cases, the low initial VSE measures indicate that the parameter space search successfully returns a reasonable approximate estimate of synchronisation. The final VSE measures are both within half a frame, which is a reasonable limit for the accuracy of manual synchronisation.

Figure 3.12 shows two examples of side-by-side frames from the Lego sequences, synchronised with a parameter search over just the offset a. The vertical placement of the frames corresponds to how they are offset in time. The difference in image sizes for the two cameras is due to the fact that the cameras have different frame rates. The images are scaled such that the ratio of their heights matches the frame rate ratio b.

3.7 Conclusions

In this chapter, cost functions have been developed for the automatic synchronisation of video sequences captured by cameras undergoing general motions. An average least-squares approach is used to avoid incorrectly rewarding a smaller time overlap. An analysis of a regular search in parameter space has been performed, incorporating the notion of minimum frame range sizes to restrict the extent of the search and avoid incorrect minima at the extremities of the search space. Synthetic tests demonstrate that a strategy of interpolating epipolar lines can often outperform a method which interpolates residuals to assess the cost function for frame indices between integer values. Furthermore, the resulting cost function can be employed in both a regular search across parameter space, and a subsequent minimisation by Levenberg-Marquardt. This typically achieves a synchronisation estimate accurate to within a small fraction of a frame, even when the frame rate ratio is unknown. These results have also been affirmed with tests on a real video sequence pair.

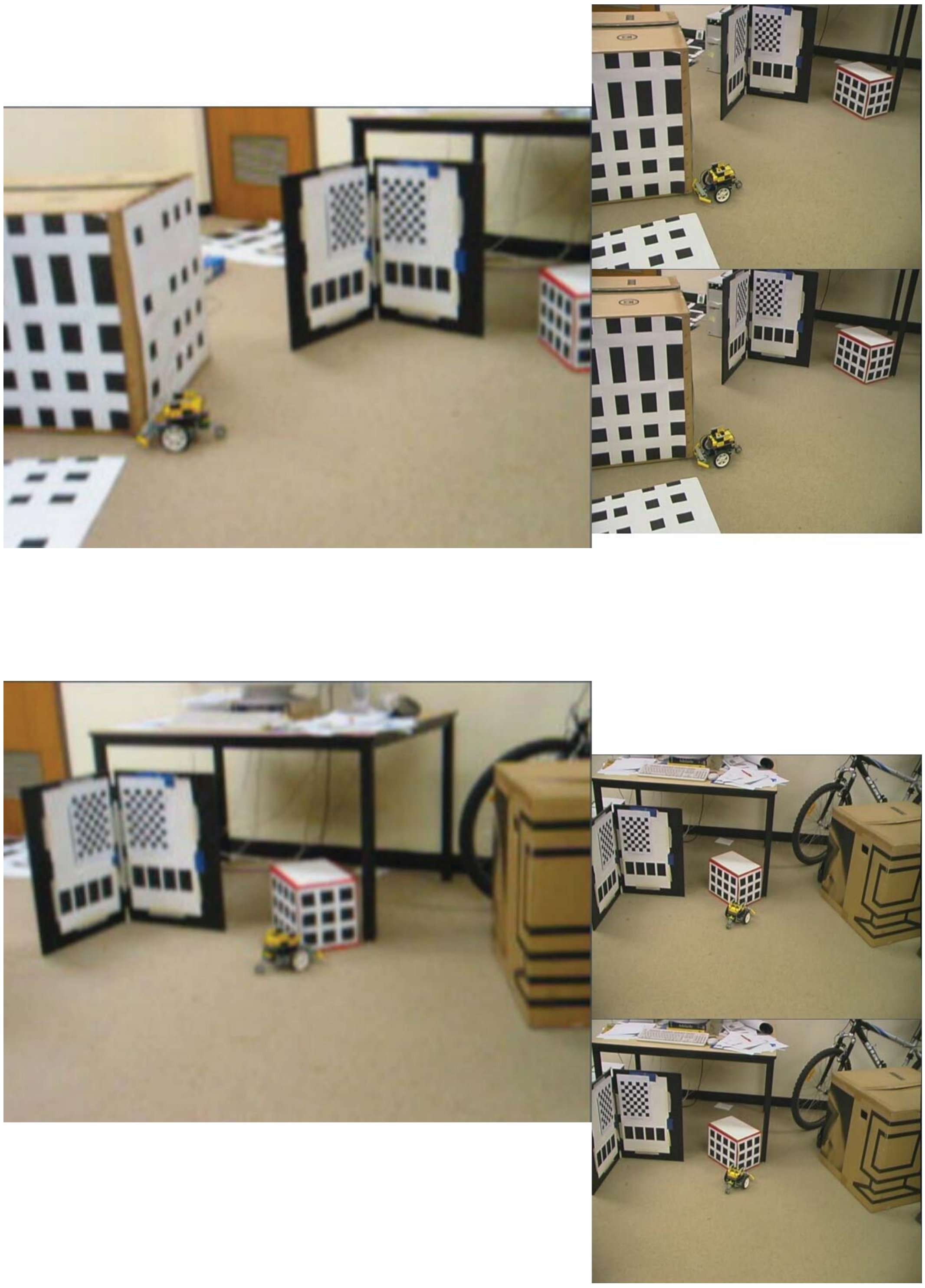


Figure 3.12: Approximately synchronous frames from the Lego robot sequences, synchronised using a uniform search for the frame offset

Chapter 4

SYNCHRONY PAIRS

In chapter 3, cost functions were developed to assess an estimate of synchronisation parameters (a, b) for independently moving cameras. Algorithms were devised to then estimate (a, b) by an exhaustive search of the parameter space. When the frame rate ratio b is known, the execution time for such an approach has quadratic complexity in the number of frames. Assume for simplicity that the video sequences have equal frame rates, and an equal number of frames, such that $b = 1$ and $n = n'$. Additionally, consider a search for the frame offset a across all integer values indicating that both cameras were recording simultaneously for at least one point in time. This requires setting the constraints on frame ranges (r_{min}, r'_{min}), as described in section 3.3, to 0. If each of the m moving scene points is visible in every frame of both video sequences, the complexity is $\mathcal{O}(mn^2)$. Note that previous methods employing a uniform search for the frame offset and using a cost function which is linear in the number of frames will have a similar complexity. If the frame rate ratio is unknown, and again assuming $n = n'$ and $r_{min} = r'_{min} = 0$, the complexity is $\mathcal{O}(mn^3)$. For lengthy video sequences, an algorithm with cubed complexity can be prohibitively expensive.

In this chapter, we propose an alternative approach, using a search for frame pairs that appear to be exactly synchronous. Each specifies a point in (f, f') space, which denote a frame index from each video sequence. Such a point is termed a *synchrony pair*. The line parameters a and b can then be chosen to best fit the set of these points. Excepting degenerate cases, this approach has lower complexity than an exhaustive search of the parameter space when b is unknown.

4.1 Finding Synchrony Pairs by Epipolar Line Interpolation

We now consider the case described in chapter 3, of moving video cameras viewing both stationary and dynamic scene objects. It is assumed that the projections of moving scene points have been tracked independently in each video sequence, and that the resulting trajectories have been correctly identified as correspondences. Adopting the same notation as used in chapter 3, moving scene feature points are labelled with integers from 1 to m. The projection of moving scene point h has a 2D location denoted $\boldsymbol{p}_{h,i}$ in frame i of the first video sequence, and location $\boldsymbol{p}'_{h,j}$ in frame j of the second video sequence. The epipolar geometry relating frames i and j from the first and second video sequences respectively is given by the fundamental matrix $\mathsf{F}_{i,j}$.

A pair of frame indices can be hypothesised as synchronous if the projections of a moving scene point *exactly* satisfy the epipolar constraint. Specifically, (i, j) is a synchrony pair if, for moving scene point h,

$$\boldsymbol{p}_{h,i}^{\top} \mathsf{F}_{i,j} \boldsymbol{p}'_{h,j} = 0.$$

Note that, in general, a pair of video cameras will not record frames simultaneously. Due to such a temporal misalignment, as well as mismeasurements in image point locations, and errors in the fundamental matrix estimates, it is highly unlikely that *any* pair of integer frame indices will perfectly satisfy this constraint. To overcome this, we use the epipolar line interpolation method described in section 3.2 to extend the search to frame indices between integer values.

Given an integer frame index i from video sequence 1, and consecutive frames j and $j+1$ from video sequence 2, the epipolar lines generated by moving scene point h in frame i of video sequence 1 are given by

$$\begin{aligned} \boldsymbol{l}_{h,i,j} &= \mathsf{F}_{i,j} \boldsymbol{p}'_{h,j}, \\ \boldsymbol{l}_{h,i,j+1} &= \mathsf{F}_{i,j+1} \boldsymbol{p}'_{h,j+1}. \end{aligned}$$

The function $\mathcal{I}_{\mathrm{L}}$ defined in equation 3.2 provides an estimate of the epipolar line between integer frame indices $(j, j+1)$. Assume there exists a hypothesised frame index k between integer values $(j, j+1)$ such that the corresponding interpolation precisely passes through the point $\boldsymbol{p}_{h,i}$. The frame index pair (i, k) can be said to exhibit zero epipolar error for moving scene point h, under the assumed model of epipolar line interpolation. Specifically, (i, k) will be selected as a synchrony pair if

$$\exists k \in [j, j+1) \text{ such that } \boldsymbol{p}_{h,i}^{\top} \mathcal{I}_{\mathrm{L}}(\boldsymbol{l}_{h,i,j}, \boldsymbol{l}_{h,i,j+1}, k-j) = 0. \tag{4.1}$$

Two cases illustrating such synchrony pairs are shown in figure 4.1.

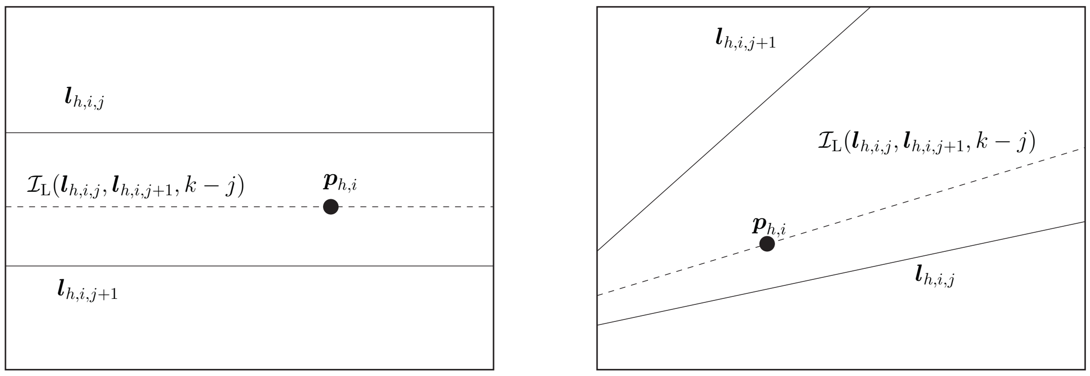
A scalar k satisfying equation 4.1 almost always exists, but not necessarily within the required range $[j, j+1)$. An expansion of $\mathcal{I}_{\mathrm{L}}$ shows that the function to find a scalar k such that an interpolation of the lines $(\boldsymbol{l}_0, \boldsymbol{l}_1)$ passes through some known point $\boldsymbol{p}$, is given by

$$\mathcal{P}_{\mathrm{I}}(\boldsymbol{p}, \boldsymbol{l_0}, \boldsymbol{l_1}) = \begin{cases} \frac{\boldsymbol{p}^{\top} \mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0)}{\boldsymbol{p}^{\top} (\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0) - \mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_1))} & : \quad [\boldsymbol{l}_0]_1 [\boldsymbol{l}_1]_1 + [\boldsymbol{l}_0]_2 [\boldsymbol{l}_1]_2 > 0 \\ & : \\ \frac{\boldsymbol{p}^{\top} \mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0)}{\boldsymbol{p}^{\top} (\mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_0) + \mathcal{N}_{\mathrm{L}}(\boldsymbol{l}_1))} & : \quad \text{otherwise} \end{cases}.$$

It follows that (i, k) is a synchrony pair if

$$k = j + \mathcal{P}_{\mathrm{I}}(\boldsymbol{p}_{h,i}, \mathsf{F}_{i,j} \boldsymbol{p}'_{h,j}, \mathsf{F}_{i,j+1} \boldsymbol{p}'_{h,j+1}) \in [j, j+1). \tag{4.2}$$

This style of approach, searching for frame indices in which image points precisely satisfy the epipolar constraint, is similar to the methods used in [4] and [57]. As described



$l_{h,i,j}$
$\mathcal{I}_L(l_{h,i,j}, l_{h,i,j+1}, k-j)$
$p_{h,i}$
$l_{h,i,j+1}$
$l_{h,i,j+1}$
$\mathcal{I}_L(l_{h,i,j}, l_{h,i,j+1}, k-j)$
$p_{h,i}$
$l_{h,i,j}$

Figure 4.1: Example synchrony pairs using epipolar line interpolation

in chapter 2, these methods work by interpolating image points to approximate their locations between frames. The alternative approach in this chapter, published earlier in [37], is more general. An interpolation of epipolar lines can be applied in the case of either stationary or independently moving cameras.

The test described by equation 4.2 can be performed for each moving scene point, testing each frame i in video sequence 1 with every consecutive frame pair $(j, j+1)$ in video sequence 2. A reciprocal search can also be conducted by considering every frame j in video sequence 2 with every frame pair $(i, i+1)$ in video sequence 1. The procedure outlining this exhaustive search for synchrony pairs is summarised in algorithm 4.1.

Once the set of synchrony pairs has been constructed, synchronisation parameters a and b can be chosen such that the line given by equation 1.1 best fits the points in the set. This is not a simple case of least-squares line fitting. We can expect exactly synchronous frame pairs to exhibit zero epipolar error, but not all frame pairs exhibiting zero epipolar error are necessarily synchronous. A simple counter-case is given in example 4.1. A method is needed for fitting line parameters (a, b) which is robust against a high proportion of outliers.

4.2 Searching Across a Subset of Frames and Moving Points

The previous section describes a method for finding synchrony pairs via an exhaustive search of possibly synchronous frame indices. In many cases, such a full search may be unnecessary, and time constraints may require a faster solution. Note that if the frame rate ratio is known, only one accurate synchrony pair is needed to define the synchronisation parameters. When b is unknown, then two accurate synchrony pairs, distant in frame index space, should be sufficient. Even when secondary lines or curves are present, as is the case in example 4.1, a subset of these synchrony pairs may still suffice. Note that such a subset may still contain erroneous synchrony pairs and therefore

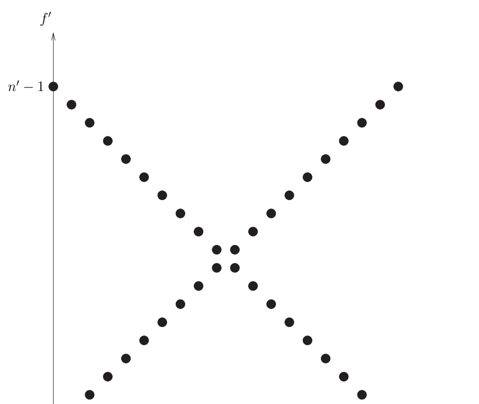
Algorithm 4.1 (An Exhaustive Search for Synchrony Pairs)

```
Initialise the synchrony pair set to $\emptyset$ (empty set)
for $h = 1$ to $m$ do
  for $i = 0$ to $n - 1$ do
    for $j = 0$ to $n' - 2$ do
      $k = j + \mathcal{P}_{\mathrm{I}}(\boldsymbol{p}_{h,i}, \mathsf{F}_{i,j}\boldsymbol{p}'_{h,j}, \mathsf{F}_{i,j+1}\boldsymbol{p}'_{h,j+1})$
      if $k \in [j, j+1)$ then
        add the frame pair $(i, k)$ to the set of synchrony pairs
      end if
    end for
  end for
  for $j = 0$ to $n' - 1$ do
    for $i = 0$ to $n - 2$ do
      $k = i + \mathcal{P}_{\mathrm{I}}(\boldsymbol{p}'_{h,j}, \mathsf{F}^{\top}_{i,j}\boldsymbol{p}_{h,i}, \mathsf{F}^{\top}_{i+1,j}\boldsymbol{p}_{h,i+1})$
      if $k \in [i, i+1)$ then
        add the frame pair $(k, j)$ to the set of synchrony pairs
      end if
    end for
  end for
end for
```

Example 4.1 (Non-Synchronous Entries in the Synchrony Pair Set)
Consider a pair of stationary cameras that record for the same period of time at the same frame rate. The resulting configuration is

$$a = 0, \quad b = 1, \quad n = n'.$$

We denote the times at which the video sequences start and stop recording as 0 *and* t_{end} *respectively. A single moving scene point visible to both cameras moves linearly in 3D space from location* $[x_0, y_0, z_0, 1]^\top$ *to* $[x_1, y_1, z_1, 1]^\top$ *for the first half of the time period. During the second half, the point linearly returns to its original location.*
The moving scene point therefore has the same location at times t *and* $t_{end} - t$*. As a result, the image point in frame* i *of video sequence 1 will have zero epipolar error with frames* i *and* $(n' - 1 - i)$ *in the second camera. The resulting set of synchrony pairs will form a cross as shown below. Linear least-squares line fitting is inappropriate in this case.*



still require a robust line-fitting process to estimate the parameters. The simplest means to find a subset of synchrony pairs is to restrict the search to a subset of frames. This can be achieved by only considering a small number of frames from the first video sequence. For each such frame i, a search for synchrony pairs is conducted for every consecutive $(j, j+1)$ frame pair in video sequence 2. As with the full search, a reciprocal search can be conducted considering only a small number of frames from the second video sequence. The question then arises as to how to choose the subsets of frames. Two obvious possibilities are a random sampling and a uniform sampling of frames from each video sequence.

In the random case, two sets are conceptually considered, each containing all possible combinations of frame indices from one of the video sequences, and the moving scene points which are visible in these frames. These sets are given as

$$\begin{aligned} \boldsymbol{S} &= \left\{(i,h) | i \in [0, n-1],\, h \in [1, m],\, \boldsymbol{p}_{h,i} \text{ is visible}\right\}, \\ \boldsymbol{S}' &= \left\{(j,h) | j \in [0, n'-1],\, h \in [1, m],\, \boldsymbol{p}'_{h,j} \text{ is visible}\right\}. \end{aligned}$$

Denoting $SIZE(.)$ as the number of elements in a set, if a speedup factor of k is required, then $k^{-1}SIZE(\boldsymbol{S})$ and $k^{-1}SIZE(\boldsymbol{S}')$ elements can be randomly selected from $\boldsymbol{S}$ and $\boldsymbol{S}'$ respectively. Since each element denotes a frame index from one video sequence, and a moving scene point, a search for synchrony pairs can be conducted for this set element, considering all frames of the other video sequence. This will reduce the total execution time to a proportion of k^{-1} compared to the full search.

For a uniform search, each moving scene point is considered independently. For a given scene point, if v_h and v'_h denote the number of frames in the first and second video sequences where the point is visible, then $\lfloor k^{-1}v_h \rfloor$ and $\lfloor k^{-1}v'_h \rfloor$ such frames are uniformly sampled, where $\lfloor . \rfloor$ denotes the floor operator. For each frame sampled, a search for synchrony pairs is conducted considering all consecutive frame pairs from the other video sequence. Note that if all moving points are visible for the same ranges of frames in both video sequences, the uniform sampling will yield the same frame subsets for each moving scene point.

4.3 Histogram Methods

Given a set of hypothesised synchronous frame pairs, a robust method is required to identify the synchronisation parameters. Chapter 2 described previous methods, specifically [52] and [4], which employ RANSAC to robustly estimate the synchronisation parameters from such a set of frame index pairs. Another approach is to build a histogram of the hypothesised synchronisation parameters, according to each synchrony pair found by algorithm 4.1.

The histogram approach is attractive for a number of reasons. The RANSAC algorithm is based on repeated random sampling of the data set. Conversely, for a set of synchrony pairs obtained via an exhaustive search, the construction of a histogram is a deterministic process with no random sampling. Furthermore, a set of synchrony pairs may contain subsets which form multiple lines. Not all such lines will necessarily indicate a negative frame rate ratio, like that depicted in example 4.1. In the case of a low number of moving scene points, using RANSAC to estimate the synchronisation parameters may wrongly identify such a line, since it could be consistent with a significant proportion of the synchrony pairs. By building a histogram, such multiple lines will be associated with distinct peaks. The most significant peak can then be chosen to provide the initial synchronisation estimate.

4.3.1 Known Frame Rates

A synchrony pair (i, j) provides a hypothesised linear constraint on the synchronisation parameters (a, b). For a known frame rate ratio b, (i, j) specifies a hypothesised value for the frame offset, given by

$$\hat{a} = j - bi.$$

A histogram built from these frame offset estimates should have peaks which correspond to synchronisation parameters with many nearby synchrony pairs. The possible range of a is given by $[-b(n-1), n'-1]$, which will be partitioned into a number of histogram cells. A count is maintained for each cell, denoting how many synchrony pairs specify a frame offset within this cell's range. Upon finding a synchrony pair (i, j), the hypothesised frame offset $\hat{a}$ is determined, and mapped to a non-negative real number given by

$$\mathcal{H}_a(\hat{a}) = w_c^{-1}[\hat{a} + b(n-1)],$$

where w_c is the width of each cell. Note this function represents a 1D affine transformation of $\hat{a}$, and maps the hypothesised frame offset to a real-valued cell index. The corresponding cell to be incremented has an integer label given by $\lfloor \mathcal{H}_a \rfloor$. When the tallest histogram peak is identified, the centre of the corresponding histogram cell is used to provide the initial estimate of the frame offset.

All that remains is to choose an appropriate cell width w_c. The first natural case to consider is that where $w_c = min(1, b)$, matching the sampling rate of the search across parameter space used in chapter 3. From a theoretical analysis of the behaviour of such a histogram, however, it emerges that this choice has undesirable properties. Consider a case where $b > 1$, $\bar{a}$ is the true frame offset, and the frame indices (i, j) are synchronous. Due to mismeasurements in image point locations, and errors introduced by the interpolation of epipolar lines, $(i + i_{err}, j)$ is wrongly identified as a synchrony pair, for some small value i_{err}. The hypothesised offset according to this synchrony pair is

$$\hat{a} = j - b(i + i_{err}) = \bar{a} - bi_{err},$$

Even if the synchrony pair is accurate to within a single frame, such that $|i_{err}| \leq 1$, the error in the offset estimate $|\bar{a} - \hat{a}|$, and hence in the histogram cell index, can be as high as b. Since b is theoretically unbounded, such a property is undesirable.

To choose an appropriate value for w_c, we restrict the analysis to only those synchrony pairs which are approximately correct. Specifically, the errors in such frame pairs are only due to slight mismeasurements in image point locations, fundamental matrices, and the epipolar line interpolations. Secondary sets of synchrony pairs, arising from complicated camera or scene point motions, such as that depicted in example 4.1, are not considered.

Consider the case where frame pair (i, j) are synchronous, such that $j = \bar{a} + bi$, and *either* i or j is an integer. Selecting integer frame i from the first video sequence, and testing for synchrony between successive frames from the second video sequence, yields a synchrony pair $(i, j + j_{err})$. In the case where j is an integer, the reciprocal search yields a synchrony pair of the form $(i + i_{err}, j)$. To form a well-defined peak on the histogram, a significant number of such synchrony pairs should correspond to the correct histogram cell.

Ideally, the cell width w_c would be chosen by considering the distributions of the error terms i_{err} and j_{err}, however these are dependent on the motions of both scene points and cameras, and will typically be unknown until a frame offset estimate is available. Instead, the cell width is determined by considering synchrony pairs accurate to within a single frame. Assuming the true frame offset $\bar{a}$ corresponds precisely with the centre of a histogram cell, a synchrony pair of the form $(i, j + j_{err})$ with $|j_{err}| < 1$ is guaranteed to increment this cell if $w_c \geq 2$. Similarly, a synchrony pair of the form $(i + i_{err}, j)$ where $|i_{err}| < 1$ will increment the correct cell if $w_c \geq 2b$. The proposed solution is to choose the average of these values, yielding a cell width of $b + 1$.

A choice of $w_c = b + 1$ has a correlation with the (α, β) parameterisation described in section 3.4. If both b and β are known, the frame offset a is given by

$$a = (n + n' - 2) - (b + 1)\beta. \tag{4.3}$$

Note that integer values of β therefore specify values of the frame offset which are spaced $b+1$ distance apart. Building a histogram of frame offset values with cell width $w_c = b+1$ is therefore equivalent to building a histogram of β values with a cell width of 1. In this case, a synchrony pair (i, j) provides a hypothesised estimate of β given by

$$\hat{\beta} = \frac{(n + n' - 2) + bi - j}{b + 1}.$$

This parameterisation is preferred, since it has simpler properties. The range of β is $[0, n+n'-2]$, regardless of the frame rate ratio. Furthermore, if a synchrony pair provides a candidate estimate $\hat{\beta}$, the histogram cell to be incremented is given simply by $\lfloor \hat{\beta} \rfloor$.

Additionally, we consider again the case of errors in the synchrony pairs. If frame indices (i, j) are truly synchronous, but $(i + i_{err}, j)$ is found as a synchrony pair, then the difference between hypothesised value $\hat{\beta}$ and true value $\bar{\beta}$ is

$$\left|\hat{\beta} - \bar{\beta}\right| = \frac{b}{b + 1} \left|i_{err}\right| \leq \left|i_{err}\right|.$$

Similarly, if a synchrony pair $(i, j + j_{err})$ is found, then this difference will be

$$\left|\hat{\beta} - \bar{\beta}\right| = \frac{1}{b+1}\left|j_{err}\right| \leq \left|j_{err}\right|.$$

Note that the absolute difference between the hypothesised $\hat{\beta}$ and true $\bar{\beta}$ is *never* larger than the frame index error terms i_{err} and j_{err}. Accordingly, for a synchrony pair such that $|i_{err}| < 1$ or $|j_{err}| < 1$, either the correct cell, or one of its immediate neighbours, will be incremented.

Once the histogram is constructed from the set of synchrony pairs, a measure is computed for each cell. This measure is defined as the sum of the cell's value and the values of the two neighbouring cells. Such smoothing is used to counter the possibility that the height of a peak may be reduced, simply due to the true value of β being close to one of the histogram cell boundaries. The centre of the cell with the highest such measure is used as the estimate for β, and the corresponding frame offset a is given by equation 4.3. This estimate of the frame offset can then be refined using a minimiser such as Levenberg-Marquardt to minimise $\mathcal{S}_{\text{ORTH-LI}}$, the cost function introduced in section 3.2.

4.3.2 Unknown Frame Rates

If the frame rate ratio is unknown, a synchrony pair (i, j) specifies the single linear constraint

$$((n + n' - 2) - (i + j))\hat{\alpha} + (i + j)\hat{\beta} - (n + n' - 2)i = 0.$$

This constraint corresponds to a line in the space of α and β. Since any synchrony pair found during the search is bounded by the first and last frames of the video sequences ($i \in [0, n-1]$, $j \in [0, n'-1]$), the gradient of this line in the space of (α, β) is guaranteed to always be negative, 0, or infinite.

The space of (α, β) can be partitioned into a discrete grid, where each grid element corresponds to a cell in a 2D histogram. The synchrony pair (i, j) defines a line of these cells, each of which has its value incremented as a consequence. This is a form of the Hough Transform for lines, a good survey of which can be found in [28].

The consequence of an error in a synchrony pair is similar to the 1D histogram case. The correct synchronisation parameters are denoted $(\bar{\alpha}, \bar{\beta})$, and frame indices (i, j) are truly synchronous. If a noisy synchrony pair $(i + i_{err}, j)$ is found, this specifies a 1D family of candidate solutions $(\hat{\alpha}, \hat{\beta})$, all of which satisfy

$$((n + n' - 2) - (i + i_{err} + j))\hat{\alpha} + (i + i_{err} + j)\hat{\beta} - (n + n' - 2)(i + i_{err}) = 0. \qquad (4.4)$$

We define error value $\delta_{\alpha,\beta}$ as

$$\delta_{\alpha,\beta} = \underset{(\hat{\alpha},\hat{\beta}) \in \boldsymbol{L}}{min}\ max\left\{\left|\hat{\alpha} - \bar{\alpha}\right|, \left|\hat{\beta} - \bar{\beta}\right|\right\},$$

where $\boldsymbol{L}$ denotes the (infinite) set of (α, β) points which satisfy equation 4.4. For $(\hat{\alpha}, \hat{\beta})$ satisfying the linear constraint in equation 4.4, the right hand side expression is minimal at the point $(\hat{\alpha}, \hat{\beta})$ where

$$\delta_{\alpha,\beta} = |\hat{\alpha} - \bar{\alpha}| = \left|\hat{\beta} - \bar{\beta}\right|.$$

Note that since the constraint on $(\hat{\alpha}, \hat{\beta})$ specifies a line of negative gradient, $\hat{\alpha} - \bar{\alpha}$ and $\hat{\beta} - \bar{\beta}$ must both have the same sign. We therefore define scalar s as

$$s = \begin{cases} -1 & : \quad (\hat{\alpha} < \bar{\alpha}) \vee (\hat{\beta} < \bar{\beta}) \\ 1 & : \quad \text{otherwise} \end{cases},$$

where $\vee$ denotes the logical 'or' operator. Rearranging the expressions for $\delta_{\alpha,\beta}$ gives

$$\begin{aligned} \hat{\alpha} &= s\delta_{\alpha,\beta} + \bar{\alpha}, \\ \hat{\beta} &= s\delta_{\alpha,\beta} + \bar{\beta}. \end{aligned}$$

Substituting these expressions into the linear constraint on $(\hat{\alpha}, \hat{\beta})$ given in equation 4.4, and rearranging, gives

$$\begin{aligned} &(gs)\delta_{\alpha,\beta} = i_{err}(g - (\bar{\beta} - \bar{\alpha})) - ((g - (i+j))\bar{\alpha} + (i+j)\bar{\beta} - gi), \\ &\text{where} \quad g = n + n' - 2. \end{aligned}$$

Note that the right-most term is the linear constraint specified by the true synchrony pair (i, j) with true parameters $(\bar{\alpha}, \bar{\beta})$, and therefore equates to 0. Therefore,

$$\delta_{\alpha,\beta} = \frac{s i_{err}(n + n' - 2 - (\bar{\beta} - \bar{\alpha}))}{n + n' - 2} = \frac{|i_{err}|\,(n + n' - 2 - (\bar{\beta} - \bar{\alpha}))}{n + n' - 2} \leq |i_{err}|\,.$$

Note that s and i_{err} must have the same sign since $\delta_{\alpha,\beta}$, and all other terms in this expression, are known to be non-negative. A similar argument applies if we consider a noisy synchrony pair $(i, j + j_{err})$, and results in the expression

$$\delta_{\alpha,\beta} = \frac{|j_{err}|\,(\bar{\beta} - \bar{\alpha})}{n + n' - 2} \leq |j_{err}|\,.$$

Note that in both cases, $\delta_{\alpha,\beta}$ is less than or equal to the error in one of the frame indices. For a synchrony pair of the form $(i + i_{err}, j)$, with $|i_{err}| < 1$, or $(i, j + j_{err})$, with $|j_{err}| < 1$, $\delta_{\alpha,\beta}$ is also less than 1. This means there exists a point $(\hat{\alpha}, \hat{\beta})$, satisfying equation 4.4, for which both $|\hat{\alpha} - \bar{\alpha}| < 1$ and $|\hat{\beta} - \bar{\beta}| < 1$. The line of solutions specified by equation 4.4 is then guaranteed to pass through the correct histogram cell, or one of its immediate neighbours.

To increment a line of cells, a straightforward line-drawing algorithm is used. This requires iterating through a range of either α or β, and incrementing the corresponding cell. For a synchrony pair (i, j), to fit the requirements of a positive frame rate ratio, the constrained ranges are

$$\alpha \in [-j, i], \quad \beta \in [i, n + n' - 2 - j].$$

To ensure a continuous line of cells is incremented, the line drawing algorithm iterates through the largest of these ranges. Assuming for simplicity that the number of cells incremented is equal to the largest of these two ranges, and that synchrony pairs are uniformly distributed across the video sequences, the expected number of cells incremented per synchrony pair is

$$\begin{aligned} &\frac{1}{(n-1)(n'-1)} \int_0^{n'-1} \int_0^{n-1} max\left\{i+j, n+n'-2-(i+j)\right\} \, di \, dj \\ &= \frac{min\{n-1, n'-1\}^2}{12\, max\{n-1, n'-1\}} + \frac{1}{2}(n+n'-2) + \frac{1}{4} max\{n-1, n'-1\}. \end{aligned} \tag{4.5}$$

After completion, the centre of the histogram element with the most support within a 3×3 window of cells is chosen for the initial estimate of (α, β). This estimate can subsequently be refined using Levenberg-Marquardt to minimise the function $\mathcal{S}_{\text{ORTH-LI}}$.

For illustrative purposes, some example synchrony pair sets along with the resulting 2D histograms are shown in figure 4.2. Note that in each case, a single prominent peak is present.

4.3.3 Histogram Method Complexity

Assuming that all moving scene points are visible in all frames, a full search for synchrony pairs requires $mn(n'-1) + mn'(n-1)$ steps, where each step tests for the existence of a synchrony pair. Note the execution time complexity is therefore quadratic in the number of frames. Incrementing a single cell in a 1D histogram has constant cost, and can therefore be considered negligible. Consequently, finding an initial synchronisation estimate in this way has complexity $\mathcal{O}(mnn')$ for a known frame rate ratio.

The unknown frame rate ratio case is less straightforward. Finding a synchrony pair causes a line of cells in the 2D histogram to be incremented. This incurs a cost which is linear in the number of frames. If synchrony pairs are only found along the line defined by the true parameters (a, b), the number of synchrony pairs found will be a linear function of the number of frames. The associated complexity will then be quadratic in the number of frames. Additional synchrony pairs forming secondary lines or curves may be present due to variations in scene point or camera motions, as illustrated in example 4.1. It is reasonable to assume that the frequency of such motions is low, since most natural motions will not involve sweeping back and forth through the scene with high repetition. There may also be regions in the space of frame indices (f, f') which exhibit a high 'density' of synchrony pairs. These will occur due to motions which are ill-conditioned or degenerate for a particular moving scene point. Across these ranges of frames, this moving point offers little benefit as a cue for synchronisation. Should efficiency be of

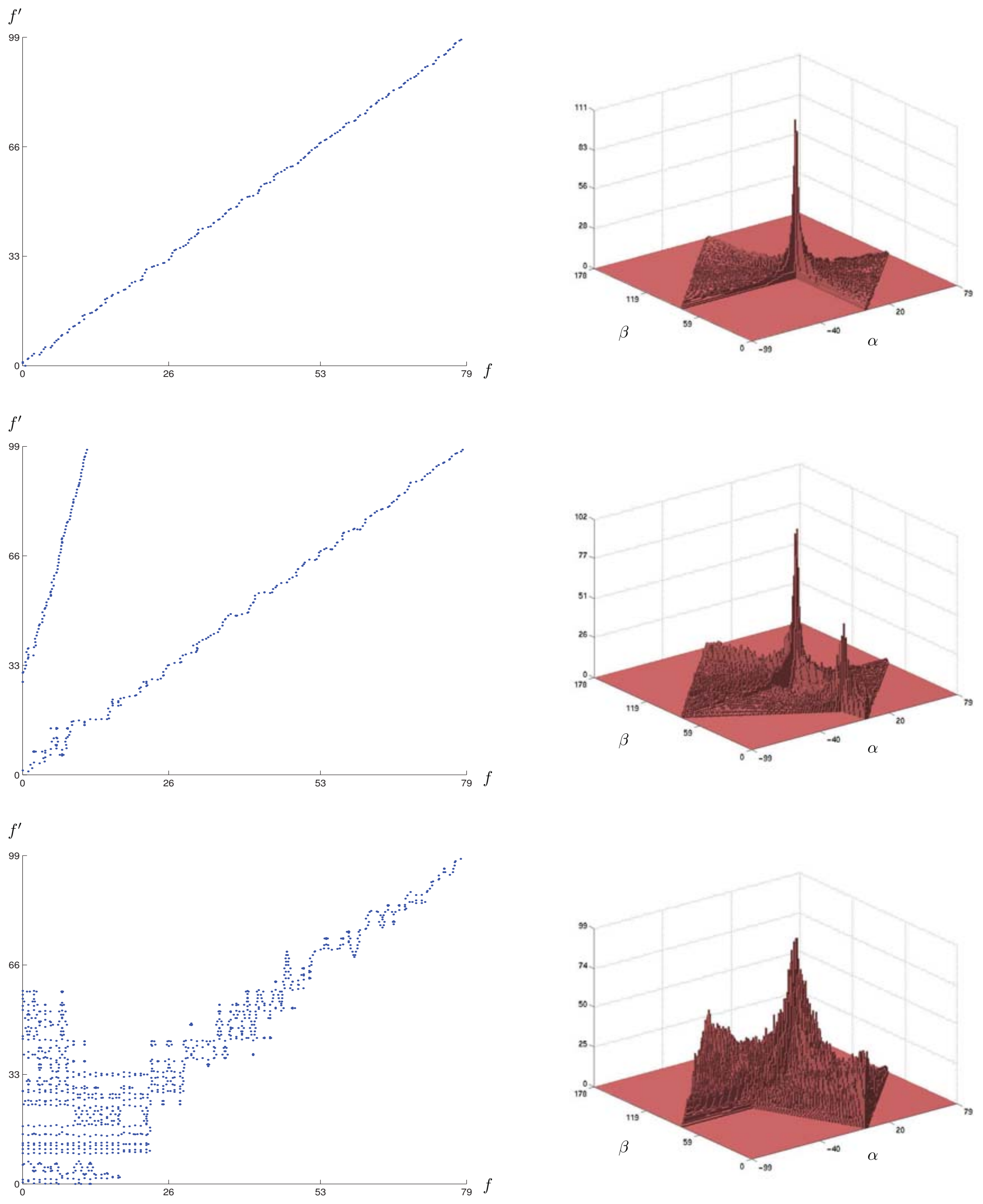


Figure 4.2: Example synchrony pair sets (left) and resulting 2D histograms (right)

Setup				Percentage of tests where $Err_\beta < 2$		
n	n'	$\bar{a}$	$\bar{b}$	Full	Uniform	Random
80	100	10.63	1.1875	99.9	99.1	98.9
80	100	42.3	1.1875	98.7	97.7	96.7
20	100	10.63	4.9375	100.0	99.9	99.6

Setup				Percentage of tests where $Err_{\alpha,\beta} < 2$		
n	n'	$\bar{a}$	$\bar{b}$	Full	Uniform	Random
80	100	10.63	1.1875	96.9	91.1	90.3
80	100	42.3	1.1875	91.1	84.8	79.2
20	100	10.63	4.9375	98.5	93.3	89.2

Table 4.1: Synthetic initial estimate performance for synchrony pair searches using a single moving scene point

particular concern, such degenerate cases could easily be detected by examining nearby epipolar errors, and excluded from the histogram.

For typical cases, with well-defined natural motions, the complexity for an unknown frame rate ratio is therefore also $\mathcal{O}(mnn')$, the same as that for when b is known.

4.4 Synthetic Tests

To test the synchrony pair search and histogram methods, the same temporal and spatial configurations used in chapter 3 are considered. True and estimated synchronisation parameters are denoted $(\bar{\alpha}, \bar{\beta})$ and $(\hat{\alpha}, \hat{\beta})$ respectively. The quality of the initial estimate is assessed by measuring its distance to truth in parameter space. In the case of an unknown frame rate ratio, this equates to the term $E_{\alpha,\beta}$ defined in section 3.5.2, and is defined as

$$E_{\alpha,\beta} = \sqrt{(\hat{\alpha} - \bar{\alpha})^2 + (\hat{\beta} - \bar{\beta})^2}.$$

For a known frame rate ratio (requiring just a 1D histogram), the distance in parameter space is given by

$$E_\beta = |\hat{\beta} - \bar{\beta}|,$$

As noted in chapter 3, the purpose of such measurements is not to illustrate an accurate synchronisation. Rather, here they indicate that the located peak of a histogram is typically ‘close’ to the correct cell.

Table 4.1 shows the percentage of tests out of 1000 in which the initial estimate was reasonably close in parameter space to the truth, for a single linear moving scene point. For each temporal configuration used in the tests, the true frame offset and frame rate ratio are denoted $(\bar{a}, \bar{b})$. The threshold of 2 was found to be the lowest integer value such that a significant majority of tests in all cases satisfy the threshold. A comparison of the unknown frame rate ratio case with the results in table 3.1 shows that a full search for synchrony pairs has a similar rate of success to a search of the parameter space. The random and uniform subset searches are configured to only use one tenth of the available basis frame and moving point pairs, yet still yield an approximately accurate histogram peak in a significant majority of cases.

The quality of final synchronisation estimates for linear moving scene points are shown in figures 4.3 and 4.4, for known and unknown frame rate ratios respectively. In each case, three temporal configurations were tested 1000 times each, for different numbers of moving scene points. For each case, the results of a full synchrony pair search are compared to uniform and random subset searches configured to sample just 10% of all possible (frame, point) pairs. Note that as m increases, the median Video Synchronisation Errors (VSEs), as defined in section 3.5.3, become indistinguishable for the three methods. For all three methods, and both a known and unknown frame rate ratio, the percentage of tests achieving a VSE smaller than 0.5 approaches 100% as more moving scene points are tracked. Recall that a VSE satisfying this threshold indicates that, for any frame from either video sequence captured when both cameras were recording, the synchronisation estimate will correctly identify the closest to synchronous frame from the other video sequence. Given the level of speedup that can be achieved, the benefit of subset searches is clear. Comparing with figure 3.10, it can be seen that for the full synchrony pair search, median VSEs are similar to those obtained by a search across the parameter space. This indicates that both search types return acceptable initial estimates that will tend to be refined to a synchronisation estimate accurate to within a small fraction of a frame.

4.5 Real Video Sequence Pair Tests

The synchronisation of video sequences using histogram methods built from a full search for synchrony pairs has been tested on the Lego robot sequences used in section 3.6. A manual synchronisation estimate is assumed to provide truth. The resulting synchrony pair set and histograms are shown in figure 4.5. Note that each histogram is dominated by a single peak corresponding to the approximate synchronisation.

The synchronisation estimates and associated VSE measures are given in table 4.2, where the estimated synchronisation is denoted $(\hat{a}, \hat{b})$. Initial results refer to the synchronisation estimate provided by locating the histogram peak, for both a known and unknown frame rate ratio. Final results are given by a subsequent Levenberg-Marquardt minimisation of the cost function $\mathcal{S}_{\mathrm{ORTH-LI}}$ developed in section 3.2. A comparison with table 3.4 indicates that the histogram methods are slightly less accurate than a search

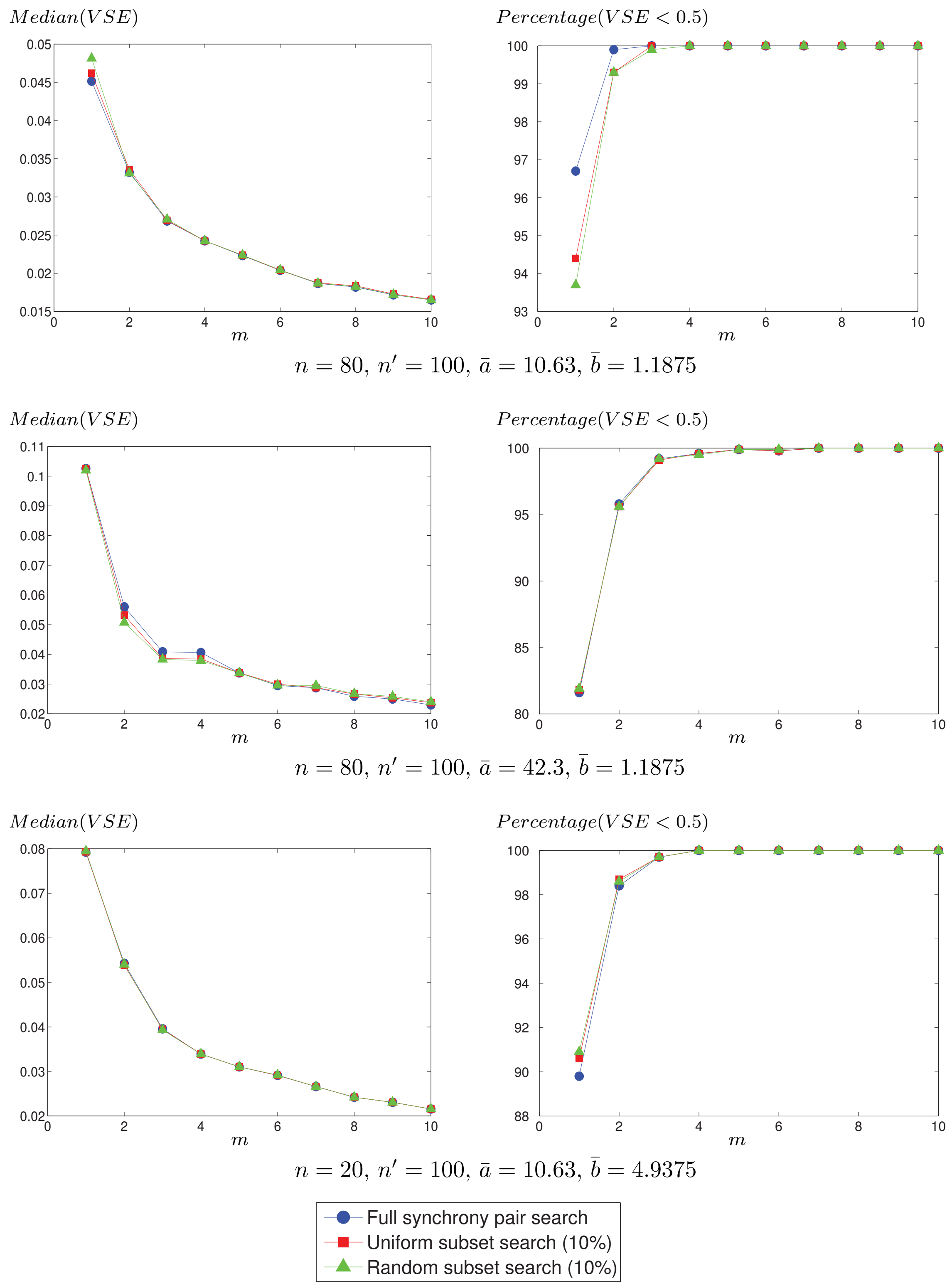


Figure 4.3: VSE results for synthetic test cases synchronised by a search for synchrony pairs, with a known frame rate ratio and scene points with linear motion

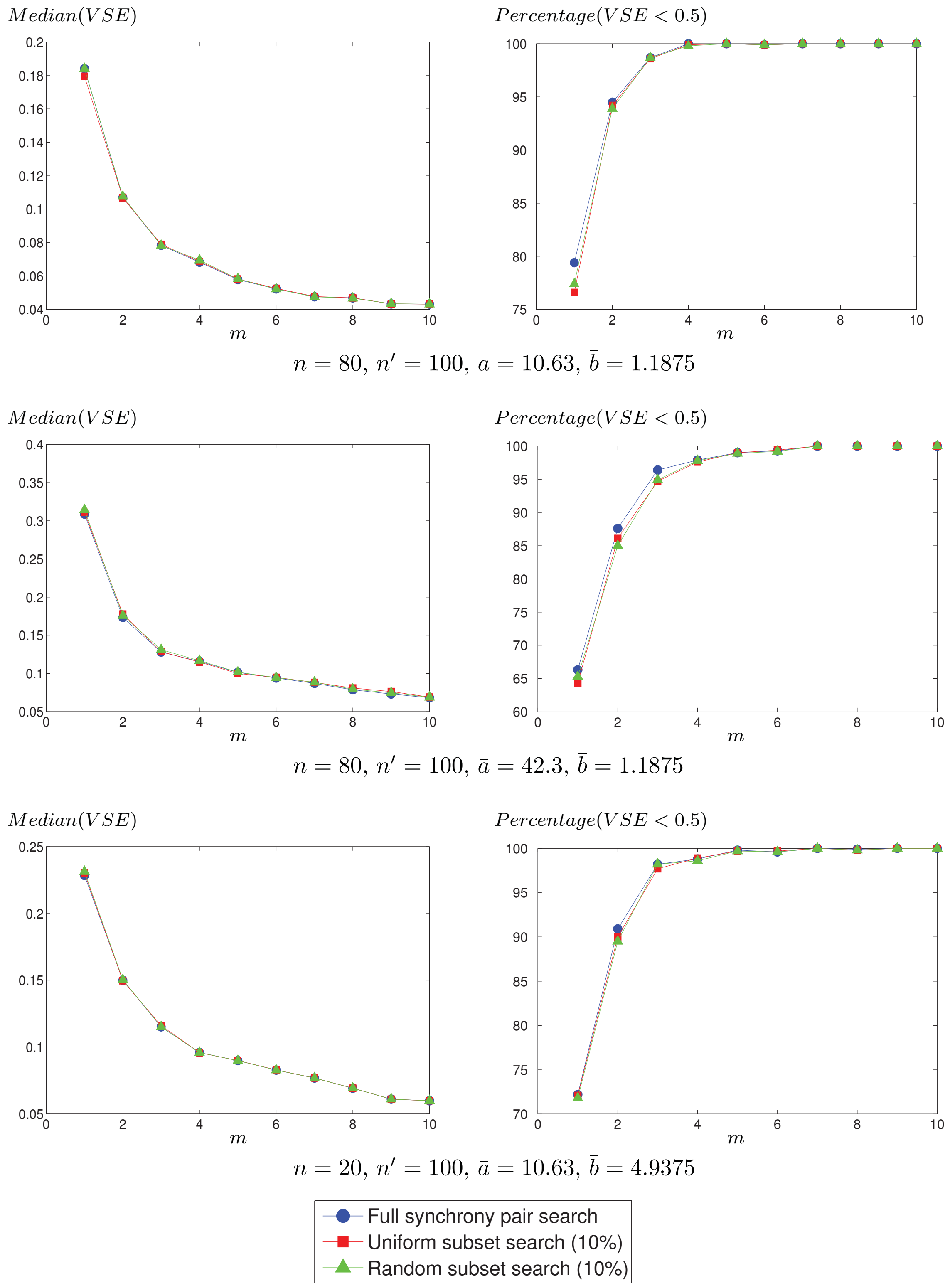


Figure 4.4: VSE results for synthetic test cases synchronised by a search for synchrony pairs, with an unknown frame rate ratio and scene points with linear motion

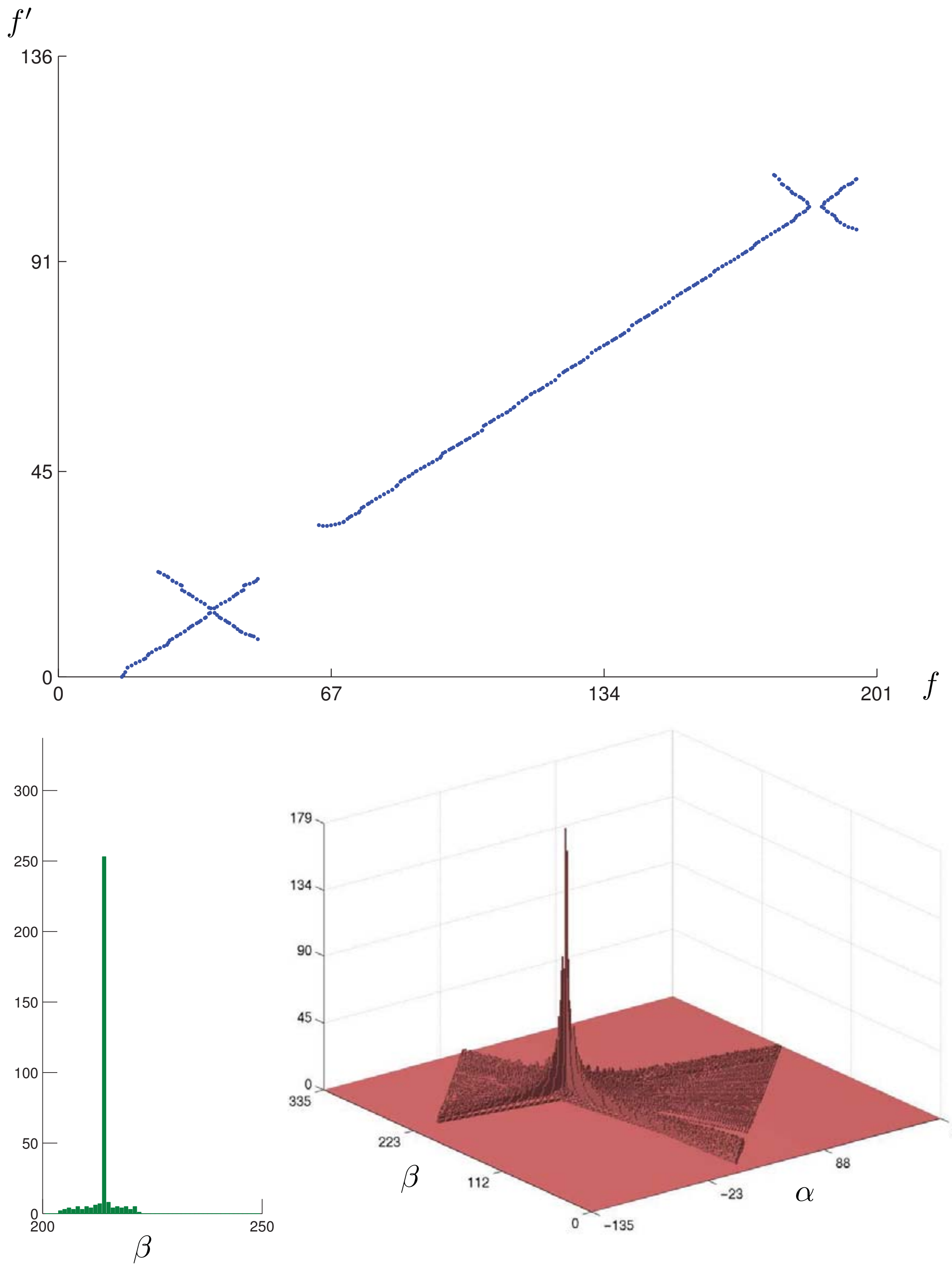


Figure 4.5: Synchrony pair set (top), and resulting 2D (bottom right) and section of 1D (bottom left) histograms for the Lego robot example

Setup: $\bar{a} = -8.1$, $\bar{b} = 0.6$, $n = 201$, $n' = 136$				
Method	Step	$\hat{a}$	$\hat{b}$	VSE
b known	initial	-8.2	-	0.166
	final	-8.032	-	0.113
b unknown	initial	-7.178	0.595	1.44
	final	-8.214	0.601	0.362

Table 4.2: VSE measures for the synchronisation of the Lego robot sequences via a search for synchrony pairs

across the parameter space. The difference, however, is small, and may merely reflect the fact that the two strategies sample the parameter space at different locations. Note that the final estimates, for both known and unknown b are identical to those obtained with a parameter space search. In this respect, the histogram based methods are equally qualified to provide an initial estimate of synchronisation for this example.

4.6 Conclusions

The concept of using a linear interpolation of epipolar lines to identify synchrony pairs has been introduced. This permits the identification of possibly synchronous frame pairs in the case of independently moving cameras. Histogram methods have been developed which enable the identification of approximate synchronisation estimates, even when the set of synchrony pairs has a high proportion of members distant from the line of synchrony. In the case of unknown frame rates, such a histogram technique is typically an order of magnitude faster than a parameter space search, excepting degenerate or high-frequency scene point motions. Further savings in time can be achieved by randomly or uniformly sampling the frames in each video sequence during the synchrony pair search.

The performance of this approach has been assessed, and shown to be comparable to a parameter space search, particularly as more moving scene points are used. The applicability of these histogram methods to non-synthetic cases has been demonstrated with the use of a real video sequence pair.

Chapter 5

ROBUST SYNCHRONISATION

Some of the feature-based synchronisation methods reviewed in chapter 2 make use of robust methods so as to achieve synchronisation when matching features between the video sequences are unavailable or contaminated with mismatches. In this chapter, the algorithms described in chapter 4 are extended to include such robustness.

Previous methods robustly synchronise a pair of video sequences by performing an exhaustive search for the frame offset, and attempting to robustly fit either a homography matrix or fundamental matrix to a set of candidate matches. As mentioned in chapter 3, these methods are inappropriate in the case of synchronising independently moving cameras by using moving scene points. Different pairs of frames will have differing spatial relationships. Accordingly, as with chapters 3 and 4, it is assumed in this chapter that the spatial relationship between the cameras has been determined in advance using stationary scene features. An estimate of the fundamental matrix relating any pair of frames from the two video sequences is therefore assumed to be available. The robust methods described in this chapter are only used to estimate the synchronisation parameters.

Similarly to chapters 3 and 4, image features generated by the projections of moving scene points provide the visual cue used to estimate the synchronisation parameters. It is assumed that the projections of a set of moving scene points have been identified, and tracked independently in each video sequence. As a result, a set of 2D trajectories, corresponding to moving scene features, is associated with each camera. A difficult case will be considered for testing. Rather than being provided with a set of hypothesised matching trajectory pairs, it is assumed that no useful correlation measure relating trajectories is available. All possible pairs of trajectories must therefore be considered as potential matches. In this situation, an additional complication arises. If two trajectories in one video sequence are visible in overlapping ranges of frames, they must correspond to different moving scene points, and should not both be matched to the same trajectory from the other video sequence. Matches may therefore be *mutually exclusive*, in the sense that identifying one trajectory pair as a match can preclude other trajectory pairs from also being considered as matches. As described in section 2.2, a robust estimation of synchrony from an exhaustive set of trajectory matches has previously been examined for the stationary camera case [7].

It should be noted that the histogram methods employed in chapter 4 already exhibit a form of robustness. Given a number of corresponding trajectories, the additional presence of a few mismatching trajectory pairs is unlikely to generate a higher peak on the histogram. Incorrectly matched trajectories will rarely produce a linear set of synchrony pairs, as illustrated by example in figure 5.1. Multiple mismatches are particularly unlikely to have linear synchrony pair sets coincide. A difficulty arises in the case where no prior matching information is available. A global histogram made from all possible pairs of trajectories could contain peaks accumulated from mutually exclusive matches. This can be a problem since the number of mismatches contributing to the histogram grows quadratically as more moving scene points are tracked.

Another issue with building a global histogram arises from the accuracy of the result. The recovered synchronisation estimate defined by the largest histogram peak is typically only approximate. For the correct matches, their associated epipolar errors measured by say, $\mathcal{S}_{\text{ORTH-LI}}$ as defined in section 3.2, may be uncharacteristically high. Therefore choosing a means to distinguish between correct and incorrect matches becomes problematic, particularly since the error measure associated with a corresponding pair of trajectories will be dependent on properties of the underlying moving scene point.

For these reasons, a RANSAC-style approach is proposed. The following sections present general algorithms for efficient robust estimation, considering both a random and exhaustive sampling of hypothesised correspondences. Issues specific to synchronisation, and the results of tests with both synthetic and real video sequence pairs follow.

5.1 Efficient Robust Estimation with Random Sampling

The following notation is used. The unknown parameters to be robustly estimated are given by the vector $\boldsymbol{\theta}$. In the case of synchronisation, this vector will contain the frame offset a, and the frame rate ratio b. Equivalently, the (α, β) parameterisation described in section 3.4 may be used. If the frame rate ratio b is known, $\boldsymbol{\theta}$ may only contain either a or β. As described in section 2.1.3, data consistent with the model $\boldsymbol{\theta}$ are termed inliers, whereas the inconsistent data are termed outliers. The inlying and outlying data are given by the set $\{\boldsymbol{x}_i | i \in [1 \ldots n_x]\}$. In the case of fundamental matrix estimation, each $\boldsymbol{x}_i$ would represent a hypothesised pair of matching image points. For synchronisation, each $\boldsymbol{x}_i$ is a pair of trajectories. Typical robust algorithms perform many iterations, producing a candidate estimate of the unknown parameters on each. The parameter estimate generated by iteration k of the robust algorithm is denoted $\boldsymbol{\theta}_k$. The function $\mathcal{R}(\boldsymbol{\theta})$ uses some robust statistic to measure the quality of a parameter estimate $\boldsymbol{\theta}$. We seek a $\boldsymbol{\theta}$ that minimises this measure. Outliers and inliers are determined by means of a classification function $\mathcal{C}(\boldsymbol{\theta}, \boldsymbol{x}_i)$, which returns 1 if $\boldsymbol{x}_i$ is an inlier according to the estimate $\boldsymbol{\theta}$, and 0 otherwise. A robust estimation process seeks to both compute an estimate of $\boldsymbol{\theta}$, and classify each x_i as either an inlier or outlier. Accordingly, a single iteration of the robust algorithm is considered a success if it yields an acceptable estimate of $\boldsymbol{\theta}$ and classification of the data. Otherwise, the iteration is considered a failure.

The robust algorithms described in section 2.1.3 repeatedly take random subsets of potential matches, and use these to produce estimates of the unknown parameter vector

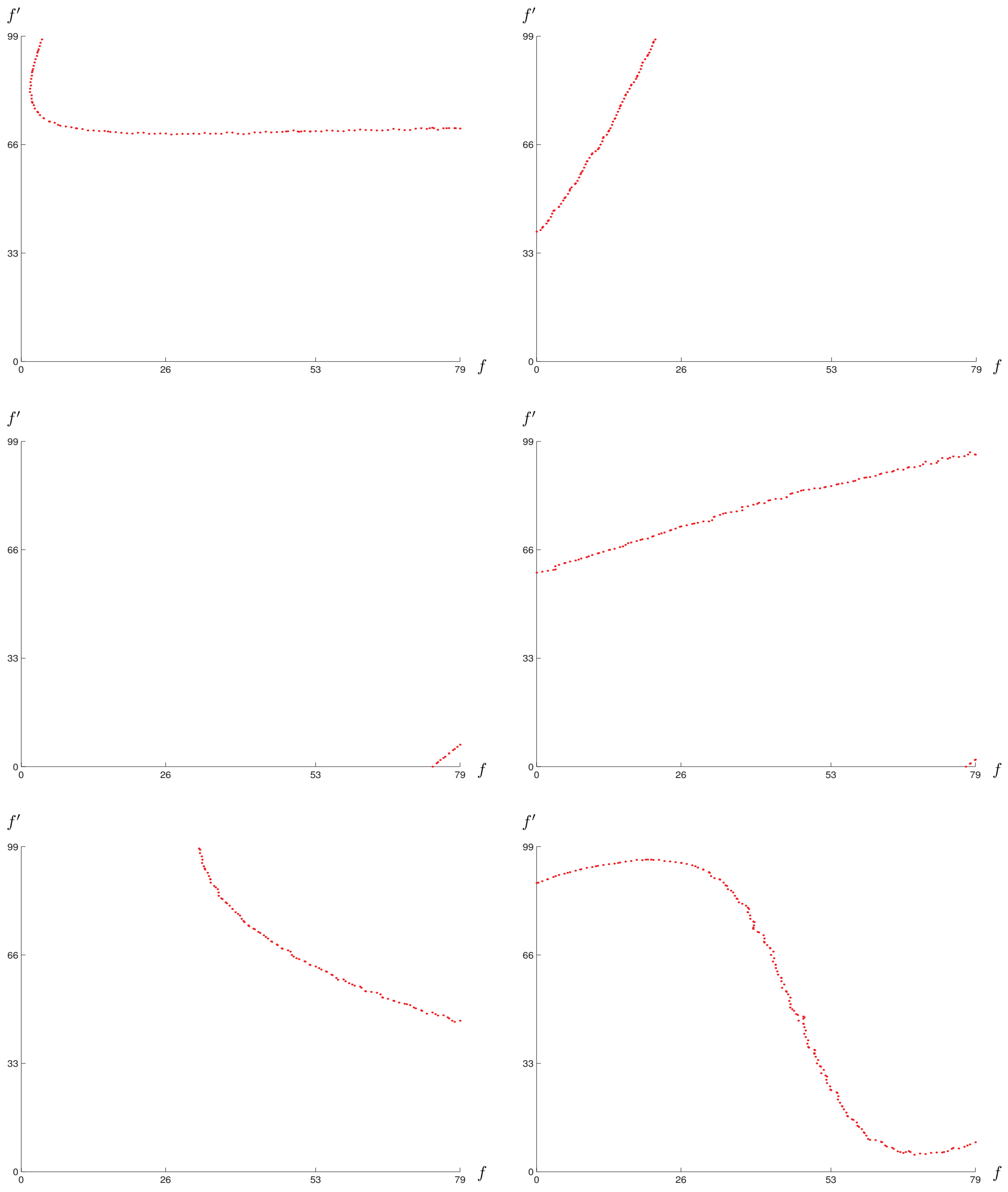


Figure 5.1: Example synchrony pair sets from incorrectly matched trajectories

$\boldsymbol{\theta}$. These subsets are of minimal size, choosing the smallest possible set of data necessary to estimate $\boldsymbol{\theta}$. Each estimate is assessed using some robust cost measure. The termination condition is typically derived by assuming a certain number of inliers exist, and considering the probability of randomly selecting a subset of data consisting entirely of inliers. The random subset size and assumed number of inliers are denoted c and μ respectively. The probability of randomly selecting a subset consisting only of inliers is denoted $\mathcal{G}(\mu)$. In [40], $\mathcal{G}(\mu)$ is approximated by $(\mu n_x^{-1})^c$, though this expression is typically an overestimate. In [10] it is noted that a random subset should be chosen without a repetition of elements, and that the correct expression is

$$\mathcal{G}(\mu) = \frac{(n_x - c)!\mu!}{(\mu - c)!n_x!}.$$

Note that the algorithms in chapters 3 and 4 can achieve synchronisation with just a single pair of matching trajectories. In this case, where $c = 1$, these two forms for $\mathcal{G}(\mu)$ are equivalent. Whichever expression for $\mathcal{G}(\mu)$ is assumed, the probability of never choosing a subset containing no outliers in η iterations is given by

$$p_f = (1 - \mathcal{G}(\mu))^\eta.$$

By choosing a low probability of failure, such as $p_f = 0.001$, this expression can be rearranged to solve for η. This specifies the number of iterations required such that, with probability $(1 - p_f)$, a random subset consisting only of inliers is chosen on at least one iteration. The number of iterations is given by

$$\eta = \frac{log(p_f)}{log(1 - \mathcal{G}(\mu))}.$$

For most vision-based problems, producing $\boldsymbol{\theta}$ estimates is fast. Since the random subsets are of the minimal size necessary for the estimation, linear algebra methods are typically employed. The resulting $\boldsymbol{\theta}$ will precisely fit each datum in the random subset, so no minimisation of more sophisticated cost functions is necessary. Consequently, optimisations designed to speed up robust estimation have focused on reducing the time needed to evaluate an estimate of $\boldsymbol{\theta}$. A particularly relevant example is the $T_{d,d}$ test introduced by Chum and Matas [10]. This test is used to discard estimates of $\boldsymbol{\theta}$ that do not classify d randomly chosen data as inliers. There also exists a class of problems, of which synchronisation is one, where computing $\boldsymbol{\theta}$ itself is costly. Given even a single pair of matching trajectories, recovering a synchronisation estimate typically has an algorithmic complexity on the order of $\mathcal{O}(n^2)$, where n is the number of frames in the video sequences. For some problems, the cost of computing $\boldsymbol{\theta}$ can be reduced at the expense of the reliability of the result. Again, synchronisation is one such case. In section 4.2 it was noted that a search for synchrony pairs need not be exhaustive. Performing the search for only a uniform or random subset of frames may suffice. We therefore consider the case where *either* estimation or evaluation can be accelerated by the use of some process or test. In general, such a speedup process will be adjustable, permitting faster iterations but increasing the probability that each will fail to correctly classify the data.

It is assumed that the speedup process is adjusted by varying a parameter ψ, which is analogous to d in the $T_{d,d}$ test. For the robust synchronisation algorithms described later in this chapter, ψ specifies how many frames in each video sequence are considered in the search for synchrony pairs. Considering the adjustable nature of the speedup, the natural question arises as to which value of ψ to choose, such that the robust estimation is completed in the least possible time. It is assumed here that a lower ψ value will decrease the average time required for each iteration, but also increase the probability that each iteration will fail. The failure may occur even for an iteration on which a random subset with no outliers is chosen. This could be because of an accelerated estimation, resulting in a less accurate $\boldsymbol{\theta}$, or because of a test such as the $T_{d,d}$ test which causes an accurate estimate of $\boldsymbol{\theta}$ to be wrongly discarded. As a consequence, more iterations will be required for lower values of ψ, to ensure the robust estimation succeeds with probability $(1 - p_f)$.

Two functions are of relevance when attempting to choose the optimal ψ. The first, $\mathcal{T}(\psi, \mu)$, specifies the average time required to compute $\boldsymbol{\theta}$ using the random subset of data, and evaluate the result. The second, $\mathcal{P}(\psi, \mu)$, specifies the probability that an iteration will succeed in yielding an acceptable $\boldsymbol{\theta}$ estimate and data classification, given a random subset which consists entirely of inliers. If the computation step is accelerated by the speedup process, with ψ expressed as proportion such that 0 implies instantaneous computation, and 1 is a full non-accelerated computation, then it is reasonable to assume that

$$\mathcal{P}(0, \mu) = 0, \quad \mathcal{P}(1, \mu) = 1.$$

The former is an obvious necessity, and the latter is consistent with typical robust methods, which assume an iteration will be considered a success for *any* random subset consisting only of inliers. In this case, we can typically expect both $\mathcal{P}(\psi, \mu)$ and $\mathcal{T}(\psi, \mu)$ to be monotonically increasing in ψ.

The probability of randomly selecting a subset of inliers, estimating $\boldsymbol{\theta}$ successfully, and correctly classifying the data, is given by $(\mathcal{G}(\mu)\mathcal{P}(\psi, \mu))$. Thus, the probability of never achieving this in η iterations is

$$p_f = (1 - \mathcal{G}(\mu)\mathcal{P}(\psi, \mu))^{\eta}.$$

Consequently, the required number of iterations to have the robust estimation succeed with a probability of $(1 - p_f)$ for some ψ is

$$\eta = \frac{log(p_f)}{log(1 - \mathcal{G}(\mu)\mathcal{P}(\psi, \mu))}. \tag{5.1}$$

For problems that admit faster $\boldsymbol{\theta}$ estimations or evaluations, the number of iterations is of little relevance. What matters is the total time required for the robust estimation to succeed with a probability of $(1 - p_f)$. Since partial iterations are typically an impossibility, we should ideally choose the ψ which minimises the total time, given by

$$\mathcal{T}_{TOTAL}(\psi,\mu,p_f) = \mathcal{T}(\psi,\mu)\left\lceil \frac{log(p_f)}{log(1-\mathcal{G}(\mu)\mathcal{P}(\psi,\mu))} \right\rceil. \tag{5.2}$$

Aside from the $\lceil . \rceil$ (ceiling) operator, an approximation to this expression is used to find the optimal d for the $T_{d,d}$ test in [10]. Indeed, the ceiling operator can be ignored if it complicates the process of choosing ψ, since the time taken for a fractional iteration is often irrelevant. In such a case, the minimum of $\mathcal{T}_{TOTAL}$ is independent of p_f. To cause the robust algorithm to terminate in the least possible time, the ideal choice for ψ is given by

$$\psi^* = \underset{\psi}{arg\,min} \quad \mathcal{T}_{TOTAL}(\psi,\mu,p_f). \tag{5.3}$$

Equivalently, if the $\lceil . \rceil$ operator is ignored, by rearranging equation 5.2 it can be seen that ψ^* is the value for which the robust estimation is most likely to succeed for any specific period of time.

Rather than assuming a specific number of inliers, many implementations of robust algorithms use an adaptive approach as described in [21]. Initially, μ is chosen to be minimal, by choosing $\mu = c$. If on iteration k, $\mathcal{R}(\boldsymbol{\theta}_k)$ is found to be the lowest error encountered so far, the number of inliers μ can be reevaluated as

$$\mu = \sum_{i=1}^{n_x} \mathcal{C}(\boldsymbol{\theta}_k, \boldsymbol{x}_i).$$

The revised value for μ then provides a different estimate for the required number of iterations η. Such an approach is useful, since it no longer requires an a-priori assumption regarding the true number of inliers. Additionally, if no iteration has succeeded once the probability of failure is reached for the *true* number of inliers, more iterations will proceed. A new algorithm will now be described, which aims to combine the benefits of an adaptive μ with accelerated iterations.

An updated μ may have a considerable effect on the choice of ψ which minimises equation 5.3. Accordingly, when a best-so-far $\boldsymbol{\theta}$ is found, ψ^* should be reevaluated. If ψ^* varies throughout the robust estimation, the termination condition can no longer be directly determined using equation 5.1. Iterations use different values of ψ^* and hence have differing probabilities of success. Instead, it is necessary to maintain a history of the ψ^* values used for each iteration. It may appear that maintaining such a history is unwieldy, since robust estimation can require a very large number of iterations. In practice, this should not be a problem. A new value of ψ^* is only chosen when a $\boldsymbol{\theta}_k$ is found that surpasses all before it, and such estimates will only be found infrequently. When using a RANSAC cost, the number of such estimates is guaranteed to be less than n_x. A concise representation is therefore possible. Assume that n_ψ separate values for ψ^* have been used, and that for each t between 1 and n_ψ, $\psi^*{=}\psi_t$ was used for k_t iterations.

After k iterations, the probability that all previous iterations failed is given by

$$\mathcal{F}(\mu) = \prod_{t=1}^{n_\psi} (1 - \mathcal{G}(\mu)\mathcal{P}(\psi_t, \mu))^{k_t}, \quad \text{where } \sum_{t=1}^{n_\psi} k_t = k. \tag{5.4}$$

Robust estimation may terminate when $\mathcal{F}(\mu)$ is smaller than or equal to the required probability of failure p_f. Evaluating $\mathcal{F}(\mu)$ is not necessary after every iteration. An accumulated probability of failure for the previous iterations, denoted $\hat{p}_f$, can be maintained. After an iteration yielding a best-so-far $\boldsymbol{\theta}$, $\mathcal{F}(\mu)$ can be evaluated, and the result stored in $\hat{p}_f$. For other iterations, it is sufficiently to multiply $\hat{p}_f$ by the probability that the current iteration failed.

In addition to altering the form of the termination condition, an adaptive μ also warrants additional consideration of the equations used to choose ψ^*. Throughout most of the robust estimation, μ will be an underestimate, which can have significant consequences on the choice of ψ^*. This is particularly true in earlier iterations, when μ will be low. While using an underestimate for μ is ideal for the termination condition, it is unnecessarily pessimistic for choosing ψ^*.

To resolve this problem, it is convenient to consider the pessimistic μ as a *lower bound* on the number of inliers. The true number of inliers is typically unknown before a successful termination, and is therefore described by a random variable N, and a prior discrete probability distribution $P(N = q)$. For example, when estimating a fundamental matrix from point correspondences, it may be appropriate to assume that each matching pair of points is an inlier with a constant probability. The probabilities $P(N = q)$ can then be assumed to follow a binomial distribution. Since the lower bound is known throughout the robust estimation, it is also convenient to denote n_{max} as the assumed upper bound on the number of inliers. For many problems, it will be sufficient to assume $n_{max} = n_x$, although the synchronisation case described later requires a lower value.

Whenever μ is updated, a new ψ^* is chosen seeking to minimise the time required for the remaining iterations. Determining the expected time until termination is difficult, since μ may change on later iterations. A simpler approach is to model the expected remaining time required for the *true* number of inliers, according to the distribution $P(N = q)$, and assuming the new ψ^* will be used for all remaining iterations. This time can be computed by considering each possible number of inliers q in the range $[\mu, n_{max}]$. If q is the true number of inliers, and the overall probability of failure is p_f, then the *remaining* iterations must only fail with probability

$$\mathcal{F}_{REM}(q) = min(1, \mathcal{F}(q)^{-1} p_f). \tag{5.5}$$

The *min* operator is required since otherwise this expression may be greater than 1, if the previous iterations already satisfy the overall probability of failure p_f. The estimated time for these iterations, for some choice of speedup ψ, is

$$\mathcal{T}_{REM}(\psi, q) = \mathcal{T}(\psi, q) \left\lceil \frac{log(\mathcal{F}_{REM}(q))}{log(1 - \mathcal{G}(q)\mathcal{P}(\psi, q))} \right\rceil.$$

The expected time until the termination condition is reached is therefore given by summing the $\mathcal{T}_{REM}$ expressions for each q, and weighting each summand by the probability that q is correct, given the observed lower bound μ. If the lower bound μ is considered an instance of a random variable N', then the assumed optimal speedup ψ^* is given by

$$\psi^* = \underset{\psi}{arg\,min} \quad \sum_{q=\mu}^{n_{max}} P(N = q|N' = \mu)\mathcal{T}_{REM}(\psi, q). \tag{5.6}$$

Note that choosing this ψ^* may result in suboptimal speedups since the true number of inliers is unknown, though this is an unavoidable consequence of assuming a prior distribution $P(N = q)$. Also, once the probability of failure is reached for an assumed number of inliers q, $\mathcal{F}_{REM}(q)$ is 1, and the corresponding summand in equation 5.6 will be 0. This means that once a sufficient number of iterations have passed, the speedup ψ^* will be chosen without consideration of q values that would indicate a correct estimate should already have been found with probability $(1 - p_f)$.

The probability terms $P(N = q|N' = \mu)$ can be determined using Bayesian inference, and are given by

$$P(N = q|N' = \mu) = \frac{P(N' = \mu|N = q)P(N = q)}{\sum_{j=\mu}^{n_{max}} P(N' = \mu|N = j)P(N = j)}. \tag{5.7}$$

The conditional probabilities $P(N' = \mu|N = q)$ represent the probabilities of observing the lower bound μ, given the history of ψ^* values used in the preceding iterations, and assuming the data set contains q inliers. The form of these probabilities is problem specific, and will typically require additional assumptions. The denominator has a constrained range since it is assumed that $P(N' = \mu|N < \mu) = 0$, reflecting an assumption that no iteration will classify more inliers than the true number.

If finding a reasonable form for the conditional probabilities is intractable, some simple alternatives may be considered. One alternative is to choose a new speedup value under the additional assumption that all previous iterations have failed. This requires defining S, which describes the event that at least one of the previous iterations succeeded. The complement of this, that no previous iterations have succeeded, is denoted $\neg\mathsf{S}$. The probability terms $P(N = q|N' = \mu)$ in equation 5.6 are replaced with

$$P(N = q|N' = \mu \cap \neg\mathsf{S}).$$

As with the probability terms $P(N = q|N' = \mu)$, these can be determined using Bayesian inference, which specifies

$$P(N = q|N' = \mu \cap \neg\mathsf{S}) = \frac{P(N' = \mu \cap \neg\mathsf{S}|N = q)P(N = q)}{\sum_{j=\mu}^{n_{max}} P(N' = \mu \cap \neg\mathsf{S}|N = j)P(N = j)}. \tag{5.8}$$

The assumption that previous iterations failed can be justified by noting that, for all $q > \mu$,

$$P(N' = \mu \cap \neg\mathsf{S}|N = q) = P(N' = \mu|N = q).$$

This equality is a simple consequence of the definition of event S. Under the hypothesis that $N = q$, the fact that all previous iterations classified fewer inliers than this indicates that those iterations have failed. Note that this equality means the numerators for the probabilities given in equations 5.7 and 5.8 are identical for all $q > \mu$. The only numerator for which this equality does not hold is the case of $q = \mu$.

Assuming that all previous iterations have failed, even for the case where $q = \mu$, allows for a simplification of the expression in equation 5.8. Using the definition of conditional probability, $P(A|B)P(B) = P(A \cap B)$, this expression can be modified to

$$\begin{aligned} P(N = q|N' = \mu \cap \neg\mathsf{S}) &= \frac{P(N' = \mu \cap \neg\mathsf{S} \cap N = q)}{\sum_{j=\mu}^{n_{max}} P(N' = \mu \cap \neg\mathsf{S} \cap N = j)} \\ &= \frac{P(N' = \mu|N = q \cap \neg\mathsf{S})P(N = q \cap \neg\mathsf{S})}{\sum_{j=\mu}^{n_{max}} P(N' = \mu|N = j \cap \neg\mathsf{S})P(N = j \cap \neg\mathsf{S})} \\ &= \frac{P(N' = \mu|N = q \cap \neg\mathsf{S})P(\neg\mathsf{S}|N = q)P(N = q)}{\sum_{j=\mu}^{n_{max}} P(N' = \mu|N = j \cap \neg\mathsf{S})P(\neg\mathsf{S}|N = j)P(N = j)}. \end{aligned} \tag{5.9}$$

Some common probabilities in this expression can now be eliminated. An iteration is considered successful if it correctly classifies each datum x_i as an inlier or outlier. The event $\neg\mathsf{S}$ indicates an assumption that this has never occurred. It is therefore assumed that every iteration has either produced an inaccurate estimate of $\boldsymbol{\theta}$, or has wrongly discarded an accurate estimate due to some speedup mechanism being employed. The observed lower bound μ is therefore independent of the true number of inliers. Consequently, $P(N' = \mu|N = q \cap \neg\mathsf{S})$ has the same value for all q. Specifically,

$$P(N' = \mu|N = q \cap \neg\mathsf{S}) = P(N' = \mu|\neg\mathsf{S}).$$

Note that these terms then cancel in equation 5.9. Furthermore, $P(\neg\mathsf{S}|N = q)$ is the probability that all previous iterations failed, assuming there are q inliers. This probability is equal to $\mathcal{F}(q)$, defined in equation 5.4. Equation 5.9 can therefore be simplified to

$$P(N = q|N' = \mu \cap \neg\mathsf{S}) = \frac{\mathcal{F}(q)P(N = q)}{\sum_{j=\mu}^{n_{max}} \mathcal{F}(j)P(N = j)}. \tag{5.10}$$

Note that using this formula will adjust the probability terms more in favour of lower values of q as additional iterations are completed. This occurs because the assumption that previous iterations failed, indicated by event $\neg\mathsf{S}$, is less likely to occur for higher hypothesised numbers of inliers. Such a trend is appropriate since μ is more likely to be the correct number of inliers as the number of iterations increases.

Summarily, this approach is a means to periodically choose an appropriate trade-off between the speed of an iteration, and the probability that it will successfully estimate $\boldsymbol{\theta}$

and correctly classify each datum x_i. We term this process *Random Adaptive Trade-off Sample Consensus* (RATSAC). Since the algorithm is based on the typical approach of repeatedly sampling and evaluating subsets of matches, it can be used in tandem with any of the robust cost functions described in section 2.1.3. A concise description of the process is given in algorithm 5.1.

Algorithm 5.1 (Random Adaptive Trade-off Sample Consensus)

```
Choose a small probability of failure $p_f < 1$
Choose an assumed upper bound on the number of inliers $n_{max}$
Number of inliers $\mu \leftarrow c$
Accumulated probability of failure $\hat{p}_f \leftarrow 1$
Lowest error $\epsilon^* \leftarrow \infty$
Best parameter estimate $\boldsymbol{\theta}_{best} \leftarrow \mathbf{0}$
Iteration count $k \leftarrow 0$
Compute $\psi^*$ as described in equation 5.6
while $\hat{p}_f > p_f$ do
  $\hat{p}_f \leftarrow \hat{p}_f(1 - \mathcal{G}(\mu)\mathcal{P}(\psi^*, \mu))$
  $k \leftarrow k + 1$
  Record $\psi$ to be used for this iteration
  Choose $c$ random samples from $\{\boldsymbol{x}_i\}$
  Compute $\boldsymbol{\theta}_k$ from the random subset, according to $\psi^*$ if applicable
  Evaluate $\mathcal{R}(\boldsymbol{\theta}_k)$, according to $\psi^*$ if applicable
  if $\mathcal{R}(\boldsymbol{\theta}_k) < \epsilon^*$ then
    $\epsilon^* \leftarrow \mathcal{R}(\boldsymbol{\theta}_k)$
    $\boldsymbol{\theta}_{best} \leftarrow \boldsymbol{\theta}_k$
    if $\mu \neq \sum_{i=1}^{n_x} \mathcal{C}(\boldsymbol{\theta}_k, \boldsymbol{x}_i)$ then
      $\mu \leftarrow \sum_{i=1}^{n_x} \mathcal{C}(\boldsymbol{\theta}_k, \boldsymbol{x}_i)$
      if $n_{max} < \mu$ then $n_{max} \leftarrow \mu$
      $\hat{p}_f \leftarrow \mathcal{F}(\mu)$ (equation 5.4)
      Recompute $\psi^*$ for the new $\mu$ (equation 5.6)
    end if
  end if
end while
Return $\boldsymbol{\theta}$ and $\{\mathcal{C}(\boldsymbol{\theta}_{best}, \boldsymbol{x}_i) | i \in [1 \ldots n_x]\}$
```

5.2 Efficient Robust Estimation with Exhaustive Sampling

A typical robust estimation algorithm repeatedly chooses small random subsets of data. This approach is used because using all possible minimally sized data subsets to estimate $\boldsymbol{\theta}$ and classify the data is prohibitively expensive. If the minimal size c is low however, an exhaustive sampling may be feasible. In such a case, *every* possible data subset of size c can be used to estimate $\boldsymbol{\theta}$, and each can be evaluated. This definitely applies in the case of synchronising video sequences recorded by independently moving cameras, as just a single pair of corresponding trajectories is sufficient to estimate (a, b).

The natural progression from section 5.1 is to consider how such an exhaustive sampling may benefit by adjusting a speedup parameter ψ. If it is assumed that there are μ inliers, then the number of data subsets containing no outliers will be $\binom{\mu}{c}$. The total number of data subsets is $\binom{n_x}{c}$. If every possible subset is used *once* to estimate and evaluate $\boldsymbol{\theta}$, then the probability that all such iterations fail to correctly classify the data is

$$p_f = (1 - \mathcal{P}(\psi, \mu))^{\binom{\mu}{c}}$$

Consider the case where the estimation and evaluation of $\boldsymbol{\theta}$ is deterministic for each data subset. If μ is known, and assuming $\mathcal{P}(\psi, \mu)$ and $\mathcal{T}(\psi, \mu)$ are monotonically increasing in ψ, then this expression can be solved to find the smallest ψ, and hence the fastest time, such that an exhaustive sampling of data subsets succeeds with probability $(1 - p_f)$. A more interesting possibility presents itself if we assume that multiple iterations for a data subset containing only inliers have independent probabilities of success. In this case, multiple estimations of $\boldsymbol{\theta}$ with the same data subset lowers the probability that all iterations fail.

Assuming such independence, if every data subset is used to estimate and evaluate $\boldsymbol{\theta}$ for η iterations, the probability that all iterations fail is

$$p_f = ((1 - \mathcal{P}(\psi, \mu))^{\eta})^{\binom{\mu}{c}}.$$

Note the change in notation from the previous section. Here η refers to the number of iterations for *each* data subset, not the total number. Given a particular choice of p_f,ψ and μ, the number of iterations required for each data subset is

$$\eta = \frac{log(p_f)}{\binom{\mu}{c} log(1 - \mathcal{P}(\psi, \mu))} + \lfloor \mathcal{P}(\psi, \mu) \rfloor$$

The addition of $\lfloor \mathcal{P}(\psi, \mu) \rfloor$ is necessary since otherwise, for $\mathcal{P}(\psi, \mu) = 1$, the estimated value for η is 0. The total time to perform $\lceil \eta \rceil$ iterations for every data subset is

$$\begin{aligned} \mathcal{T}_{TOTAL}(\psi, \mu, p_f) &= \binom{n_x}{c} \mathcal{T}(\psi, \mu) \lceil \eta \rceil \\ &= \binom{n_x}{c} \mathcal{T}(\psi, \mu) \left\lceil \frac{log(p_f)}{\binom{\mu}{c} log(1 - \mathcal{P}(\psi, \mu))} + \lfloor \mathcal{P}(\psi, \mu) \rfloor \right\rceil . \end{aligned}$$

Unlike RATSAC, where the use of $\lceil . \rceil$ is often unnecessary, here it is critical. A 'partial' iteration for *every* data subset can have a considerable effect on the estimated total time. If a constant μ is assumed, ψ can be chosen to minimise this expression.

The proposed algorithm proceeds in a breadth-first type strategy. Multiple sweeps are performed, where each sweep consists of iterating over *every* possible data subset of size c. Each is used to estimate $\boldsymbol{\theta}$ and evaluate the result. The algorithm terminates at the end of a sweep, once the accumulated probability that all iterations have failed is smaller than p_f. As is the case with RATSAC, additional considerations are needed for an adaptive termination condition.

The number of inliers μ is reevaluated whenever a $\boldsymbol{\theta}$ is found surpassing all estimates evaluated before it, according to the robust cost measure $\mathcal{R}$. The new μ however, is only employed once the next sweep begins. This ensures that each data subset consisting entirely of inliers has an equal probability of successfully estimating $\boldsymbol{\theta}$ and classifying the data. Consequently, it is only necessary to maintain the accumulated probability that all previous iterations failed for a *single* data subset consisting only of inliers. This probability is denoted $\hat{p}_s$. If n_ψ different values for ψ have been used, and k_t sweeps have been performed using a speedup of ψ_t,the accumulated probability of failure $\hat{p}_s$ after k sweeps is given by

$$\hat{p}_s = \mathcal{F}_S(\mu) = \prod_{t=1}^{n_\psi} (1 - \mathcal{P}(\psi'_t, \mu))^{k'_t}, \quad \text{where } \sum_{t=1}^{n_\psi} k'_t = k. \tag{5.11}$$

Note that this expression for $\hat{p}_s$ may not explicitly contain μ. Indeed, if the nature of the speedup is such that the expression for $\mathcal{P}(\psi, \mu)$ does not contain the number of inliers, it is never necessary to recompute $\hat{p}_s$ when a best-so-far solution is found. In this case, maintaining a history of ψ values is unnecessary. The algorithm may terminate at the end of a sweep when

$$\hat{p}_s^{\binom{\mu}{c}} \leq p_f.$$

As with RATSAC, a discrete probability distribution is used to describe the true number of inliers, given by $P(N = q)$, so that the speedup value ψ is not chosen solely assuming a pessimistic value for μ. An optimal ψ is chosen that minimises the expected time until the termination condition is reached for the *true* number of inliers. At any stage of the robust estimation, the true number of inliers can be assumed to lie between μ and n_{max}. For any q in this range, the required probability of failure for the *remaining* iterations is

$$\mathcal{F}_{REM}(q) = min(1, p_f \mathcal{F}_S(q)^{-\binom{q}{c}}). \tag{5.12}$$

If the remaining iterations fail with probability $\mathcal{F}_{REM}(q)$, then the overall probability of failure, assuming q inliers, is p_f. The time required for these iterations is

$$\mathcal{T}_{REM}(\psi, q) = \binom{n_x}{c} \mathcal{T}(\psi, q) \left\lceil \frac{log(\mathcal{F}_{REM}(q))}{\binom{q}{c} log(1 - \mathcal{P}(\psi, q))} + \lfloor \mathcal{P}(\psi, q) \rfloor \right\rceil.$$

The expected time until the termination condition is reached for the true number of inliers is computed by summing $\mathcal{T}_{REM}$ for each q, and weighting each by the probability that there are q inliers, given the observed lower bound μ. Specifically, the speedup value ψ^* is chosen as

$$\psi^* = \underset{\psi}{arg\,min} \quad \sum_{q=\mu}^{n_{max}} P(N = q | N' = \mu) \mathcal{T}_{REM}(\psi, q) \tag{5.13}$$

As with RATSAC, the probability weights can be modified to include an additional assumption that all previous iterations failed. Assuming q inliers, the probability of failure is given by

$$P(\neg \mathsf{S} | N = q) = \mathcal{F}_S(q)^{\binom{q}{c}}.$$

Additionally, it is assumed again that $P(N' = \mu | N = q \cap \neg \mathsf{S})$ is the same value regardless of q. Under these assumptions, the probability terms in equation 5.13 can be replaced with

$$P(N = q | N' = \mu \cap \neg \mathsf{S}) = \frac{\mathcal{F}_S(q)^{\binom{q}{c}} P(N = q)}{\sum_{j=\mu}^{n_{max}} \mathcal{F}_S(j)^{\binom{j}{c}} P(N = j)}. \tag{5.14}$$

It should be noted that, unlike RATSAC, an exhaustive sampling can guarantee success providing probability $\mathcal{P}(\psi, \mu)$ reaches a certainty in the valid range of ψ. If $p_f = 0$, and functions $\mathcal{P}$ and $\mathcal{T}$ are monotonically increasing, the algorithm defaults to choosing the smallest ψ^* such that $\mathcal{P}(\psi, \mu) = 1$ and only performing one sweep.

Since the above method exhaustively samples *all* possible data subsets, it is termed Comprehensive Adaptive Trade-off Sample Consensus (CATSAC). The full process is listed in algorithm 5.2.

5.3 Robust Synchronisation Details

Before either RATSAC or CATSAC can be applied to the problem of synchronisation, $\mathcal{C}(\boldsymbol{\theta}, \boldsymbol{x}_i)$, $\mathcal{R}(\boldsymbol{\theta})$, $\mathcal{P}(\psi, \mu)$ and $\mathcal{T}(\psi, \mu)$ must first be defined, along with the assumed distribution for the number of inliers.

For simplicity of notation, it is assumed that each datum $\boldsymbol{x}_i$ is a vector of length 2, such that (h, h') specifies a correspondence between trajectory h in the first video sequence, and h' in the second video sequence. The total number of moving points tracked through each

Algorithm 5.2 (Comprehensive Adaptive Trade-off Sample Consensus)

```
Choose a small probability of failure $p_f < 1$
Choose an assumed upper bound on the number of inliers $n_{max}$
Number of inliers $\mu \leftarrow c$
Accumulated probability of failure $\hat{p}_s \leftarrow 1$
Lowest error $\epsilon^* \leftarrow \infty$
Best parameter estimate $\boldsymbol{\theta}_{best} \leftarrow \mathbf{0}$
Compute $\psi^*$ as described in equation 5.13
$\hat{\mu} \leftarrow \mu$
Sweep count $k \leftarrow 0$
**while** $\hat{p}_s^{\binom{\mu}{c}} > p_f$ **do**
  $k \leftarrow k + 1$
  **for** each subset of size $c$ **do**
    Compute $\boldsymbol{\theta}$, with speedup $\psi^*$ if applicable
    Evaluate $\mathcal{R}(\boldsymbol{\theta})$, with speedup $\psi^*$ if applicable
    **if** $\mathcal{R}(\boldsymbol{\theta}) < \epsilon^*$ **then**
      $\epsilon^* \leftarrow \mathcal{R}(\boldsymbol{\theta})$
      $\boldsymbol{\theta}_{best} \leftarrow \boldsymbol{\theta}$
      **if** $\mu \neq \sum_{i=1}^{n_x} \mathcal{C}(\boldsymbol{\theta}, \boldsymbol{x}_i)$ **then**
        $\hat{\mu} \leftarrow \sum_{i=1}^{n_x} \mathcal{C}(\boldsymbol{\theta}, \boldsymbol{x}_i)$
      **end if**
    **end if**
  **end for**
  $\hat{p}_s \leftarrow \hat{p}_s(1 - \mathcal{P}(\psi^*))$
  **if** $\hat{\mu} \neq \mu$ **then**
    $\mu \leftarrow \hat{\mu}$
    **if** $n_{max} < \mu$ **then** $n_{max} \leftarrow \mu$
    Recompute $\hat{p}_s \leftarrow \mathcal{F}_S(\mu)$ if necessary (equation 5.11)
    Recompute $\psi^*$ (equation 5.13)
  **end if**
**end while**
Return $\boldsymbol{\theta}$ and $\{\mathcal{C}(\boldsymbol{\theta}_{best}, \boldsymbol{x}_i) | i \in [1 \ldots n_x]\}$
```

video sequence are denoted $\dot{m}$ and $\dot{m}'$. It is assumed that no tentative correspondence information is available, so all possible trajectory pairs are considered potential matches. The size of the data set used for RATSAC or CATSAC is then $\dot{m}\dot{m}'$. The resulting set of matches is denoted $\boldsymbol{M}$.

Analogously to the notation used in chapter 3, the image point $\boldsymbol{p}_{h,i}$ denotes the image location associated with trajectory h, in frame i of the first video sequence. Similarly, $\boldsymbol{p}'_{h',j}$ is the image point associated with trajectory h' in frame j of the second video sequence. $\mathsf{F}_{i,j}$ is the fundamental matrix describing the epipolar geometry relating frames i and j of the two video sequences. Additionally, we define visibility measures $v_{h,i}$ and $v'_{h',j}$, such that $v_{h,i}$ is 1 if trajectory h is visible in frame i of video sequence 1, and 0 otherwise. Similarly, $v'_{h',j}$ is 1 if trajectory h' is visible in frame j of video sequence 2, and 0 otherwise. The terms v_h and $v'_{h'}$ specify the numbers of frames in which trajectories h and h' are visible.

5.3.1 Inlier Classification Function $\mathcal{C}$

A hypothesised matching pair of trajectories is assessed using the interpolated epipolar error $\mathcal{E}_{\mathrm{ORTH-LI}}$ developed in section 3.2. Adopting the notation from section 3.1, $\mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}$ denotes an equivalent error which equates to 0 in the event that $\mathcal{E}_{\mathrm{ORTH-LI}}$ is unmeasurable. Given synchronisation estimate (a, b), the number of measurable epipolar errors for all frames in both sequences, for trajectory pair (h, h') is denoted $\dot{\mathcal{W}}(a, b, h, h')$. A trajectory pair (h, h') therefore has an average interpolated epipolar error given by

$$\begin{aligned}\dot{\mathcal{S}}(a,b,h,h') = \frac{1}{\dot{\mathcal{W}}(a,b,h,h')} \sum_{i=0}^{n-1} \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(&\boldsymbol{p}_{h,i},\ \boldsymbol{p}'_{h',\lfloor a+bi \rfloor}, \mathsf{F}_{i,\lfloor a+bi \rfloor}, \\ &\boldsymbol{p}'_{h',\lfloor a+bi \rfloor+1}, \mathsf{F}_{i,\lfloor a+bi \rfloor+1}, \\ &a+bi-\lfloor a+bi \rfloor)^2 + \\ \frac{1}{\dot{\mathcal{W}}(a,b,h,h')} \sum_{j=0}^{n'-1} \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(&\boldsymbol{p}'_{h',j},\ \boldsymbol{p}_{h,\lfloor a'+b'j \rfloor}, \mathsf{F}^{\top}_{\lfloor a'+b'j \rfloor, j}, \\ &\boldsymbol{p}_{h,\lfloor a'+b'j \rfloor+1}, \mathsf{F}^{\top}_{\lfloor a'+b'j \rfloor+1, j}, \\ &a'+b'j-\lfloor a'+b'j \rfloor)^2. \\ \text{where } a' = -ab^{-1}, \quad b' = b^{-1}.\end{aligned} \tag{5.15}$$

This cost represents the sample variance of the interpolated epipolar errors. Under the assumption that such summands have a zero-mean Gaussian distribution, the expression $\dot{\mathcal{W}}(a, b, h, h')\dot{\mathcal{S}}(a, b, h, h')$ has an order $\dot{\mathcal{W}}(a, b, h, h')$ Chi-squared distribution. As explained in section 3.1, there are two causes for an epipolar error being unmeasurable. A trajectory will typically not be tracked throughout an entire video sequence, so image points $\boldsymbol{p}_{h,i}$ and $\boldsymbol{p}'_{h',j}$ may be undefined for certain frame indices i and j. Image points are also undefined for frame indices beyond the boundaries of the video sequences. For frame index i in video sequence 1, the supposedly synchronous frame $a + bi$ in video sequence 2 may lie outside the range of $[0, n' - 1]$.

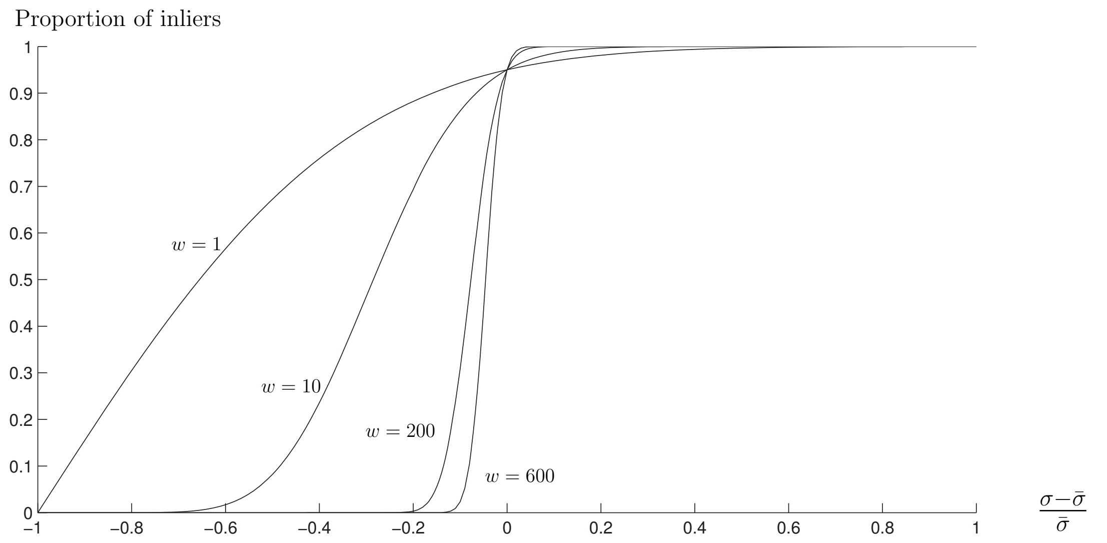


Figure 5.2: Effects of a misestimation of σ on a threshold test for Chi-squared distributions of different order

The simplest inlier classification test is a threshold. For an assumed standard deviation σ, a threshold can be determined for $\dot{\mathcal{S}}(a, b, h, h')$ that admits a specific proportion of the 'best' inliers. Two issues arise here. First, the threshold is dependent on the order of the distribution $\dot{\mathcal{W}}(a, b, h, h')$. Two matches with different numbers of measurable epipolar errors will be assessed with different thresholds. Second, these orders can be high, well into the hundreds even for video sequences with just a few seconds of footage. For such high $\dot{\mathcal{W}}$ values, the distributions associated with a sample variance are very narrow around σ^2. Since σ is rarely known to high precision, the effects of a slight misestimation are far greater as the number of measurable epipolar errors increases.

Figure 5.3.1 illustrates the consequences of misestimating the standard deviation. The horizontal axis gives the relative error in σ, where truth is given by $\bar{\sigma}$. The vertical axis shows the proportion of inliers that will actually be admitted when using a threshold test designed to admit 95% of inliers. Curves are shown for Chi-squared distributions of various orders, denoted w. Note the effect of under-estimating σ by as little as 10%. For cases where $w = 1$, which applies in the case of say, fundamental matrix estimation, 92% of inliers will be admitted instead of 95%. Robust estimation may still perform well in such a case. For a higher $w = 200$, which could occur for synchronisation, the proportion of inliers passing the threshold test will be just 30%. For an even higher $w = 600$, which could arise from just 12 seconds of footage, only 2% of inliers will pass the threshold test.

A simple solution to the problem mentioned above is to use the same threshold for every $\dot{\mathcal{S}}(a, b, h, h')$, regardless of the number of measurable summands. We choose this threshold such that 95% of inliers will be admitted for the case where $\dot{\mathcal{W}}(a,b,h,h') = 1$. The threshold test is therefore

$$\mathcal{C}_{\mathrm{T}}(a,b,h,h') = \begin{cases} 1 & : \quad \dot{\mathcal{S}}(a,b,h,h') \leq 3.84\sigma^2 \\ 0 & : \quad \text{otherwise} \end{cases}$$

Note that if the assumed σ is greater than roughly $\frac{1}{2}\bar{\sigma}$, at least 95% of inliers should still be admitted for cases with high values of $\dot{\mathcal{W}}$. It may seem that allowing such leniency is dangerous. If $\sigma = \bar{\sigma}$, larger proportions of inliers will pass the threshold test as $\dot{\mathcal{W}}$ increases. Fortunately, this should not be accompanied with a significant risk of admitting incorrect matches. For high values of $\dot{\mathcal{W}}$, an incorrect match is less likely to pass the threshold test as the greater amount of visual information will tend to generate a higher value for $\dot{\mathcal{S}}(a,b,h,h')$.

The inlier classification function does not consist of just a threshold test. To cover the case where all possible pairs of trajectories are admitted as potential matches, care must be taken to ensure that the set of inliers resulting from $\mathcal{C}(a,b,h,h')$ does not contain mutually exclusive matches. If two trajectories in one video sequence are visible in overlapping ranges of frames, it is reasonable to assume that they should not both be matched to the same trajectory from the other video sequence. For simplicity, preference is always given to the match with the lowest associated error $\dot{\mathcal{S}}(a,b,h,h')$.

Given two trajectory pairs (h,h') and (g,g'), the function $\mathcal{L}$ counts the number of frames in the each video sequence in which these trajectory matches are both visible. Specifically,

$$\mathcal{L}(h,h',g,g') = \sum_{i=0}^{n-1} v_{h,i} v_{g,i} + \sum_{j=0}^{n'-1} v'_{h',j} v'_{g',j}.$$

The exclusivity test that returns 0 if a trajectory pair (g,g') should preclude (h,h') from being classified as an inlier, and 1 otherwise, is given by

$$\mathcal{D}(a,b,h,h',g,g') = \begin{cases} 0 & : \quad ((h=g) \veebar (h'=g')) \wedge (\mathcal{C}(a,b,g,g')=1) \wedge \\ & \quad \left(\dot{\mathcal{S}}(a,b,h,h') \geq \dot{\mathcal{S}}(a,b,g,g')\right) \wedge (\mathcal{L}(h,h',g,g') \geq 0) \\ 1 & : \quad \text{otherwise} \end{cases}.$$

The symbol $\wedge$ denotes the 'and' operator, and $\veebar$ denotes the 'exclusive or' operator. For (h,h') to be classified as an inlier, $\mathcal{D}(a,b,h,h',g,g')$ must equate to 1 for *all* other matches (g,g') in the set $\boldsymbol{M}$. The test to classify a hypothesised match (h,h') as an inlier or outlier is

$$\mathcal{C}(a,b,h,h') = \mathcal{C}_{\mathrm{T}}(a,b,h,h') \prod_{(g,g)\in \boldsymbol{M}} \mathcal{D}(a,b,h,h',g,g').$$

Note that from the definition of $\mathcal{D}$, the classification of (h,h') may be dependent on the classification of other matches with a lower associated error. Classification of matches must therefore proceed in ascending order of $\dot{\mathcal{S}}(a,b,h,h')$.

5.3.2 Robust Estimation Function $\mathcal{R}$

Following the approach of MSAC [50], an estimate of synchronisation parameters (a, b) is assessed by how well it fits the set of matches classified as inliers, along with a penalty term for the remaining matches. The sum of squared epipolar errors associated with match (h, h') is given by

$$\dot{\mathcal{S}}_{SUM}(a, b, h, h') = \dot{\mathcal{S}}(a, b, h, h')\dot{\mathcal{W}}(a, b, h, h').$$

The matches can be partitioned into inlier and outlier sets defined by the functions $\mathcal{M}_{\mathrm{I}}$ and $\mathcal{M}_{\mathrm{O}}$, where

$$\begin{aligned}\mathcal{M}_{\mathrm{I}}(a, b) &= \left\{(h, h') \in \boldsymbol{M} \mid \mathcal{C}(a, b, h, h') = 1\right\},\\ \mathcal{M}_{\mathrm{O}}(a, b) &= \left\{(h, h') \in \boldsymbol{M} \mid \mathcal{C}(a, b, h, h') = 0\right\}.\end{aligned}$$

The sizes of these sets are given by functions $\mathcal{N}_{\mathrm{I}}(a, b)$ and $\mathcal{N}_{\mathrm{O}}(a, b)$ respectively. The robust cost used to assess an estimate of (a, b) is defined as

$$\mathcal{R}(a, b) = \mathcal{N}_{\mathrm{I}}(a, b)\frac{\sum_{(h,h')\in\mathcal{M}_{\mathrm{I}}(a,b)} \dot{\mathcal{S}}_{SUM}(a, b, h, h')}{\sum_{(h,h')\in\mathcal{M}_{\mathrm{I}}(a,b)} \dot{\mathcal{W}}(a, b, h, h')} + 3.84\sigma^2\mathcal{N}_{\mathrm{O}}(a, b). \tag{5.16}$$

The robust cost is therefore the number of matches classified as inliers, multiplied by their overall average epipolar error, with an additional penalty term equal to the number of outliers multiplied by the threshold.

For two synchronisation estimates with an equal number of classified outliers, $\mathcal{R}$ will be lower for the estimate with the lowest average interpolated epipolar error associated with the data classified as inliers. Consequently, as well as providing a means to measure the quality of parameter (a, b) during robust estimation sampling, $\mathcal{R}$ can also be subsequently minimised directly with a minimiser such as Levenberg-Marquardt, to improve the synchronisation estimate and possibly reclassify the trajectory correspondences. For a range of synchronisation parameters (a, b) in which the classification of matches is unchanging, minimising $\mathcal{R}$ is equivalent to minimising the cost function $\mathcal{S}_{\mathrm{ORTH-LI}}$ defined in chapter 3.

5.3.3 Estimating Synchronisation From a Single Match

Before specifying the speedup functions $\mathcal{T}(\psi, \mu)$ and $\mathcal{P}(\psi, \mu)$, the process by which a single match is used to estimate the synchronisation must first be described.

The estimation of synchronisation parameters (a, b) consists of a search for synchrony pairs using a variation of the histogram methods described in chapter 4. This is followed by a Levenberg-Marquardt minimisation to improve the estimate. Unlike in chapter 4, a unique histogram is built for each trajectory pair (h, h'). It is therefore unnecessary to use a histogram which covers the full range of frames from the two video sequences. Considering match (h, h'), where trajectory h is visible in frames $[i_{min} \ldots i_{max}]$, and trajectory

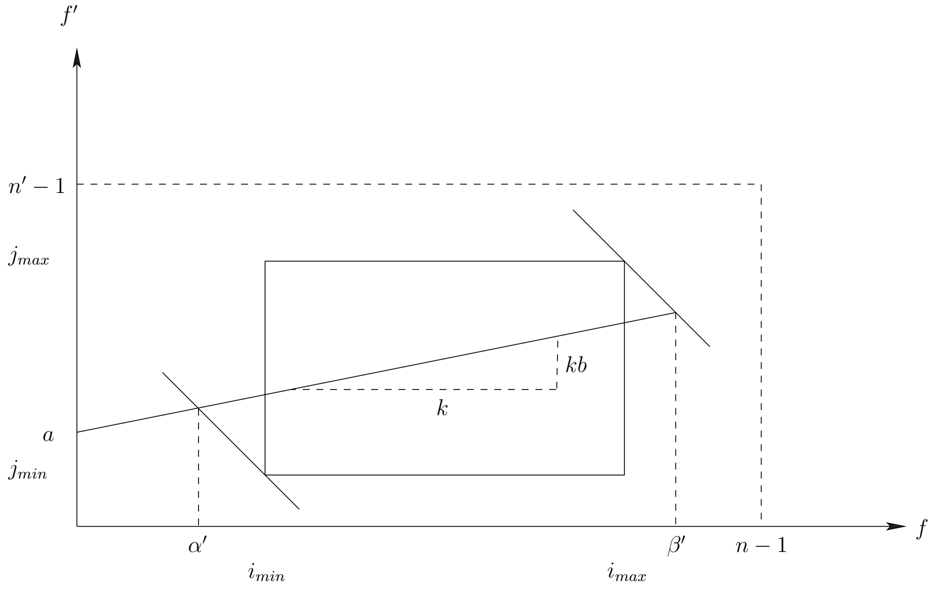


Figure 5.3: Line parameterisation (α', β').

h' in $[j_{min} \dots j_{max}]$, the synchronisation can be parameterised by

$$\alpha' = \frac{i_{min} + j_{min} - a}{b+1}, \quad \beta' = \frac{i_{max} + j_{max} - a}{b+1}.$$

Note the similarity to the (α, β) parameterisation introduced in section 3.4. The value of α' specifies the intersection of line of synchrony with a line of gradient -1 passing through point (i_{min}, j_{min}). Similarly, β' specifies the intersection with a line of gradient -1 passing through (i_{max}, j_{max}). This is illustrated in figure 5.3.

Since the line of synchrony must have a positive gradient $(b \geq 0)$, and any synchrony pairs found for the trajectory pair (h, h') will have frame indices bounded by (i_{min}, i_{max}) and (j_{min}, j_{max}), α' and β' are constrained to the ranges

$$\alpha' \in [i_{min} - (j_{max} - j_{min}), i_{max}],$$
$$\beta' \in [i_{min}, i_{max} + (j_{max} - j_{min})].$$

When estimating synchronisation using a matching trajectory pair (h, h'), such a parameterisation permits smaller histograms. In the case where the frame rate ratio b of two cameras is unknown, a 2D histogram is used, with each cell covering a range in parameter space (α, β) of size 1. This yields a 2D histogram with $(v_h + v'_{h'} - 2)^2$ cells. Similarly to chapter 4, synchronisation can be parameterised by β' if the frame rate ratio is known. A 1D histogram with $(v_h + v'_{h'} - 2)$ cells is used.

A search for synchrony pairs is performed, randomly sampling $\lceil \psi^{-1} v_h \rceil$ and $\lceil \psi^{-1} v'_{h'} \rceil$ frames from the first and second video sequences respectively. For each selected frame, a test for a synchrony pair is conducted using all consecutive frame pairs from the other video sequence. As in chapter 4, the histogram element with the most support within a 3×1 or 3×3 window provides the initial synchronisation estimate. Although uniform sampling has been shown to have better performance for a single search, it is entirely deterministic, so there is no benefit to performing multiple iterations for the same trajectory pair (h, h').

A synchronisation estimate obtained using histogram methods is only approximate, and therefore unsuitable for evaluating and classifying potential matches. This estimate should therefore be refined, using a Levenberg-Marquardt minimisation with an appropriate cost function. The minimisation is performed using the (α, β) parameterisation, though (a, b) is used in the following function definition to avoid an unwieldy subscript notation.

Note that there is much redundancy in measuring every possible epipolar error associated with a trajectory pair (h, h') and a synchronisation estimate (a, b). Consecutive frames are likely to have associated errors of the same order of magnitude, and only a subset of errors are typically needed to find a superior estimate. To facilitate speedup, a coarse sampling of the error measures is used. Given the initial synchronisation estimate (α, β), the range of frames with measurable residuals in each video sequence can be computed in closed-form. Subsets of frames are chosen from each video sequence, denoted $V_{h,h'}$ and $V'_{h,h'}$, such that the chosen frame indices are evenly distributed across these ranges. If the number of measurable residuals are w_v and w'_v, then the size of the sets are given by

$$\begin{aligned} SIZE(V_{h,h'}) &= max(min(5, w_v), \tfrac{1}{20} w_v), \\ SIZE(V'_{h,h'}) &= max(min(5, w'_v), \tfrac{1}{20} w'_v), \end{aligned}$$

where $SIZE(.)$ denotes the size of a set. This can be interpreted as typically choosing one in 20 frames, which equates to a little more than one per second for standard video camera frame rates. The $max(.)$ and $min(.)$ operators ensure that no fewer than 5 frames are chosen for each video sequence, unless fewer than this are available. Given the synchronisation estimate (α, β), the number of measurable epipolar errors, for frames in the sets $V_{h,h'}$ and $V'_{h,h'}$, is denoted $\overline{\mathcal{W}}(a, b, h, h')$. The synchronisation estimate is refined by minimising the following cost function:

$$\begin{aligned} \overline{\mathcal{S}}(a, b, h, h') = \frac{1}{\overline{\mathcal{W}}(a, b, h, h')} \sum_{i \in V_{h,h'}} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}_{h,i}, \boldsymbol{p}'_{h',\lfloor a+bi \rfloor}, \mathsf{F}_{i,\lfloor a+bi \rfloor}, \\ & \qquad \boldsymbol{p}'_{h',\lfloor a+bi \rfloor+1}, \mathsf{F}_{i,\lfloor a+bi \rfloor+1}, \\ & \qquad a + bi - \lfloor a + bi \rfloor)^2 + \\ \frac{1}{\overline{\mathcal{W}}(a, b, h, h')} \sum_{j \in V'_{h,h'}} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}'_{h',j}, \boldsymbol{p}_{h,\lfloor a'+b'j \rfloor}, \mathsf{F}^{\top}_{\lfloor a'+b'j \rfloor,j}, \\ & \qquad \boldsymbol{p}_{h,\lfloor a'+b'j \rfloor+1}, \mathsf{F}^{\top}_{\lfloor a'+b'j \rfloor+1,j}, \\ & \qquad a' + b'j - \lfloor a' + b'j \rfloor)^2. \end{aligned} \tag{5.17}$$

$$\text{where } a' = -ab^{-1}, \quad b' = b^{-1}.$$

Note that the sets of frame indices for which errors are measured do not change throughout the minimisation. The majority of these should remain measurable if (h,h') is a true correspondence, and providing the initial (α,β) is reasonably accurate. For the majority of truly corresponding (h,h'), this minimisation typically terminates after several iterations. The maximum number of iterations before termination is therefore set to 15.

After the coarse minimisation, three tests are used to determine if the synchronisation estimate should be discarded, causing the robust algorithm RATSAC or CATSAC to proceed to the next iteration. The first such test is a visibility constraint. We assume that a correct match (h,h') should generate a synchronisation estimate indicating a significant overlap in time of the ranges of frames in which the trajectories are visible. If the number of measurable residuals for the refined estimate are w_v and w'_v, the estimate is discarded if both

$$w_v < \tfrac{1}{4}v_h, \quad w'_v < \tfrac{1}{4}v'_{h'}.$$

The motivation for this test is based on the observation that a synchronisation estimate is more likely to have a low associated error $\dot{\mathcal{S}}_{\text{ORTH-LI}}$ across a short range of frames. The majority of poor synchronisation estimates are therefore expected to have few measurable errors. This test detects such estimates quickly, and the corresponding iteration of RATSAC or CATSAC can terminate without incurring the expense of assessing every trajectory pair.

The second test is based on the quality of the epipolar error measures. If trajectory pair (h,h') are projections of the same moving scene point, it is reasonable to assume that (h,h') is classified as an inlier by an accurate synchronisation estimate. Accordingly, the match (h,h') must satisfy the threshold test $\mathcal{C}_{\text{T}}$. The synchronisation estimate is therefore discarded if

$$\dot{\mathcal{S}}(a,b,h,h') > 3.84\sigma^2.$$

The third test involves a coarse sampling of measurable residuals for all matches, in the same manner as that used for the single match minimisation. Given the refined (α,β) values, subsets of frames $V_{h,h'}$, $V'_{h,h'}$ are chosen for *every* trajectory pair (h,h'). From the coarse error estimates, defined by $\overline{\mathcal{S}}(a,b,h,h')$, inliers and outliers are classified. The classification function equivalent to $\mathcal{C}$, but using the coarse error measures $\overline{\mathcal{S}}(a,b,h,h')$, is denoted $\overline{\mathcal{C}}$. The function to classify trajectory pairs as inliers or outliers, based on their associated coarse $\overline{\mathcal{S}}(a,b,h,h')$ measures, is

$$\overline{\mathcal{C}}(a,b,h,h') = \overline{\mathcal{C}}_{\text{T}}(a,b,h,h') \prod_{(g,g)\in \boldsymbol{M}} \overline{\mathcal{D}}(a,b,h,h',g,g'),$$

where

$$\overline{\mathcal{D}}(a,b,h,h',g,g') = \begin{cases} 0 & : \quad ((h=g)\,\underline{\vee}\,(h'=g')) \wedge (\overline{\mathcal{C}}(a,b,g,g')=1) \wedge \\ & \quad \left(\overline{\mathcal{S}}(a,b,h,h') \geq \overline{\mathcal{S}}(a,b,g,g')\right) \wedge (\mathcal{L}(h,h',g,g') \geq 0) \\ 1 & : \quad \text{otherwise} \end{cases},$$

and

$$\overline{\mathcal{C}}_{\mathrm{T}}(a,b,h,h') = \begin{cases} 1 & : \quad \overline{\mathcal{S}}(a,b,h,h') \leq 3.84\sigma^2 \\ 0 & : \quad \text{otherwise} \end{cases} .$$

Based on the errors measured over a coarse sampling of frames, the set of trajectory pairs can be tentatively partitioned into inlier and outlier sets, given by functions

$$\overline{\mathcal{M}}_{\mathrm{I}}(a,b) = \left\{(h,h') \in \boldsymbol{M} \mid \overline{\mathcal{C}}(a,b,h,h') = 1\right\},$$
$$\overline{\mathcal{M}}_{\mathrm{O}}(a,b) = \left\{(h,h') \in \boldsymbol{M} \mid \overline{\mathcal{C}}(a,b,h,h') = 0\right\}.$$

The sizes of the sets defined by functions $\overline{\mathcal{M}}_{\mathrm{I}}$ and $\overline{\mathcal{M}}_{\mathrm{O}}(a,b)$ are given by the functions $\overline{\mathcal{N}}_{\mathrm{I}}(a,b)$ and $\overline{\mathcal{N}}_{\mathrm{O}}(a,b)$ respectively. An approximation of the robust cost function $\mathcal{R}$, measured using only a coarse sampling of frames for each trajectory pair, is given by

$$\overline{\mathcal{R}}_(a,b) = \overline{\mathcal{N}}_{\mathrm{I}}(a,b)\frac{\sum_{(h,h')\in\overline{\mathcal{M}}_{\mathrm{I}}(a,b)} \overline{\mathcal{S}}_{\mathrm{SUM}}(a,b,h,h')}{\sum_{(h,h')\in\overline{\mathcal{M}}_{\mathrm{I}}(a,b)} \overline{\mathcal{W}}(a,b,h,h')} + 3.84\sigma^2\overline{\mathcal{N}}_{\mathrm{I}}(a,b), \tag{5.18}$$

where $\overline{\mathcal{S}}_{\mathrm{SUM}} = \overline{\mathcal{S}}_(a,b,h,h')\overline{\mathcal{W}}(a,b,h,h')$.

A full evaluation of the synchronisation parameters, using all measurable errors from each match (h,h') only proceeds if this coarse robust cost $\overline{\mathcal{R}}_(a,b)$ is less than the minimum robust cost found on all previous iterations. Note that performing this test enables synchronisation estimates with a high associated robust cost to be detected quickly. Such estimates can safely be discarded.

The process of estimating and evaluating a synchronisation estimate from a single match (h,h') is summarised in algorithm 5.3. Note this amounts to a single iteration of either RATSAC or CATSAC.

5.3.4 Speed Parameter Functions $\mathcal{P}(\psi,\mu)$ and $\mathcal{T}(\psi,\mu)$, and Probability Distributions

The probability function $\mathcal{P}(\psi,\mu)$ represents the probability that an evaluation of synchronisation parameters succeeds for a correct match. This will be highly dependent on the nature of synchrony pair sets that are generated by the motions of the cameras and scene points. Assuming well-defined non-degenerate motions, it is assumed for simplicity that the probability of success is given by

$$\mathcal{P}(\psi,\mu) = \psi^{\frac{1}{10}}.$$

This function is therefore monotonically increasing, with $\mathcal{P}(0,\mu) = 0$, and $\mathcal{P}(1,\mu) = 1$ as desired. Furthermore, $\mathcal{P}(0.1,\mu) \approxeq 0.8$, indicating a high probability of success for a random sampling rate of 0.1, as demonstrated in chapter 4.

Since the set of matches will contain exhaustive trajectory pairings, the probability of each match being correct is unknown. The prior distribution for the number of inliers $P(N = q)$ is therefore assumed to be uniform. Choosing an appropriate n_{max} is more

Algorithm 5.3 (Accelerated Iteration for Estimating Synchronisation)

```
RATSAC or CATSAC specify a trajectory correspondence $(h, h')$
RATSAC or CATSAC specify the lowest error found so far, $\epsilon^*$
RATSAC or CATSAC specify speedup parameter $\psi^*$
Randomly select $\psi^* \lceil v_h \rceil$ frames from video sequence 1
Randomly select $\psi^* \lceil v'_{h'} \rceil$ frames from video sequence 2
Search for synchrony pairs for each selected frame
**if** frame rate ratio $b$ is known **then**
  Build a 1D histogram from the synchrony pairs
  Determine $a$ from the tallest histogram peak
  Compute coarse frame sets $V_{h,h'}$ and $V'_{h,h'}$
  Refine $a$ by minimising $\overline{\mathcal{S}}(a, b, h, h')$ (equation 5.17)
**else**
  Build a 2D histogram from the synchrony pairs
  Determine $(a, b)$ from the tallest histogram peak
  Compute coarse frame sets $V_{h,h'}$ and $V'_{h,h'}$
  Refine $(a, b)$ by minimising $\overline{\mathcal{S}}(a, b, h, h')$ (equation 5.17)
**end if**
Compute measurable error counts $w_v, w'_v$
**if** $w_v < \frac{1}{4} v_h$ and $w'_v < \frac{1}{4} v'_{h'}$ **then**
  Discard the estimate (proceed to next iteration)
**end if**
**if** $\dot{\mathcal{S}}(a, b, h, h') > 3.84\sigma^2$ (equation 5.15) **then**
  Discard the estimate (proceed to next iteration)
**end if**
Compute coarse frame sets $V_{u,u'}$ and $V_{u,u'}$ for *all* matches $(u, u') \in \boldsymbol{M}$
Compute coarse error $\overline{\mathcal{S}}(a, b, u, u')$ for all matches $(u, u') \in \boldsymbol{M}$
**if** $\overline{\mathcal{R}}(a, b) > \epsilon^*$ (equation 5.18) **then**
  Discard the estimate (proceed to next iteration)
**end if**
Compute full robust cost $\mathcal{R}(a, b)$ (equation 5.16)
```

problematic. A choice of $n_{max} = n_x$ is clearly inappropriate due to the exhaustive nature of the matches. Ideally, we should use the size of the largest subset of matches such that no two trajectory pairs in the subset are mutually exclusive, and for which there exists a line of synchrony (a, b) permitting each such match to have measurable associated epipolar errors. Even without the latter condition, this is equivalent to the maximum clique problem in graph theory, which is known to be NP-Complete. For simplicity, it is initially assumed that $n_{max} = min(\dot{m}, \dot{m}')$. Even though this may be an underestimate, it is a useful upper bound assuming long 'unbroken' trajectories. If a number of inliers μ is classified during the execution of either RATSAC or CATSAC, such that $\mu > n_{max}$, n_{max} is increased accordingly. Mutually exclusive matches also complicate the probability terms in equations 5.6 and 5.13 when choosing ψ^*. Accordingly, the assumption of previous failure is used, as defined in equations 5.10 and 5.14.

The function $\mathcal{T}(\psi, \mu)$ measures the time taken to estimate parameters (a, b) from a single pair of matching trajectories, and evaluate the result. Since the units of time are arbitrary, it is assumed that the evaluation of a single epipolar residual, or the search between 2 frames for a single possible synchrony pair takes time 1. Estimating (a, b) requires a reduced search for synchrony pairs, controlled by parameter ψ, a subsequent coarse minimisation, and three tests to determine if the estimate should be discarded. Consequently, the total time can be separated into

$$\mathcal{T}(\psi, \mu) = \mathcal{T}_{\text{SEARCH}}(\psi, \mu) + \mathcal{T}_{\text{HIST}}(\psi, \mu) + T_{\text{MIN}} + \mathcal{T}_{\text{TEST}}(\psi, \mu) + \mathcal{T}_{\text{EVAL}}(\psi, \mu)$$

The time required to search for synchrony pairs is $\mathcal{T}_{\text{SEARCH}}$. The time required to reset the histogram to 0, add synchrony pairs to the histogram, and find the maximum after the search is completed is given by $\mathcal{T}_{\text{HIST}}$. The post-search minimisation, testing, and evaluation times are given by T_{MIN}, $\mathcal{T}_{\text{TEST}}$ and $\mathcal{T}_{\text{EVAL}}$ respectively.

A full search for synchrony pairs across all frames would require $(2nn' - n - n')$ units of time. Since the trajectories may only be visible for a smaller ranges of frames, the search need only be conducted over these intervals. For trajectories (h, h') in the first and second video sequences respectively, recall that the total number of frames in which a point is visible are given by

$$v_h = \sum_{i=0}^{n-1} v_{h,i}, \quad v'_{h'} = \sum_{j=0}^{n'-1} v'_{h',j}.$$

A full search for synchrony pairs within these ranges requires iterating over each frame, and considering all successive pairs of frames from the other video sequence. This therefore takes $v_h(v'_{h'} - 1) + v'_{h'}(v_h - 1)$ units of time. The average time taken to perform this search can therefore be expressed as the expected value of this term, across all possible trajectory pairs. Furthermore, each search is scaled by proportion ψ since only a random subset of frames is selected. The expression $\mathcal{T}_{\text{SEARCH}}$ is therefore

$$\mathcal{T}_{\text{SEARCH}}(\psi, \mu) = 2\psi \frac{1}{n_x} \left(\sum_{(h,h')\in \boldsymbol{M}} v_h(v'_{h'} - 1) + v'_{h'}(v_h - 1) \right).$$

The time to initialise the histogram cells and find the peak after the search will be dependent on the histogram size, and therefore whether or not the frame rate ratio is known. The value t_s represents the time required to perform these operations on a single cell. The time required to increment a histogram cell is denoted t_a. To measure the cost of all additions to the histogram, the expected number of synchrony pairs must be known. Note that this is typically unavailable, since it is a function of the scene point and camera motions. For simplicity, it is assumed that correct matches yield an average of $n_i = 2$ synchrony pairs per frame, and incorrect matches produce an average of $n_o = 1$ synchrony pairs per frame. The expected number of synchrony pairs per frame is therefore

$$\mathcal{N}_{\text{S}}(\mu) = n_x^{-1}\mu n_i + (1 - n_x^{-1}\mu) n_o.$$

For a known frame rate ratio, each synchrony pair requires incrementing just one cell. The total cost for histogram operations is therefore given by

$$\mathcal{T}_{\text{HIST}}(\psi, \mu) = \frac{t_s}{n_x} \left(\sum_{(h,h')\in \boldsymbol{M}} v_h + v'_{h'} - 2 \right) + \frac{t_a \psi}{n_x} \left(\sum_{(h,h')\in \boldsymbol{M}} v_h + v'_{h'} \right) \mathcal{N}_{\text{S}}(\mu).$$

For an unknown frame rate ratio, a line of cells must be incremented per synchrony pair. It is assumed that each trajectory occupies a continuous range of frames, and that, for a trajectory pair (h, h'), synchrony pairs are uniformly distributed across the ranges of frames in which these trajectories are visible. By considering the (α', β') parameterisation for the histogram, and according to equation 4.5 given in section 4.3.2, the average number of cells incremented by a synchrony pair found for trajectory pair (h, h') is

$$\mathcal{A}_{\text{C}}(h, h') = \frac{min\{v_h - 1, v'_{h'} - 1\}^2}{12\, max\{v_h - 1, v'_{h'} - 1\}} + \frac{1}{2}(v_h + v'_{h'} - 2) + \frac{1}{4} max\{v_h - 1, v'_{h'} - 1\}.$$

The associated histogram operation cost is therefore

$$\mathcal{T}_{\text{HIST}}(\psi, \mu) = \frac{t_s}{n_x} \left(\sum_{(h,h')\in \boldsymbol{M}} (v_h + v'_{h'} - 2)^2 \right) + \frac{t_a \psi}{n_x} \left(\sum_{(h,h')\in \boldsymbol{M}} \mathcal{A}_{\text{C}}(h, h')(v_h + v'_{h'}) \right) \mathcal{N}_{\text{S}}(\mu).$$

The values for t_s and t_a need not be known to high accuracy, since the relative cost of histogram operations is very small. For a known frame rate ratio, $t_s = 0.01$ is assumed. For an unknown frame rate ratio, a higher $t_s = 0.025$ is used due to the additional cost of evaluating whether a cell is a peak. For both cases, $t_a = 0.005$ is assumed.

The time T_{MIN} is dependent on the initial estimate of the synchronisation, as well as the visibility of the trajectory pair used to estimate $\boldsymbol{\theta}$. An estimate of the synchronisation defines a number of frames in each video sequence for which epipolar errors are measurable. It is assumed that, over many iterations, these numbers of frames are uniformly distributed between $(\lfloor\frac{1}{4}v_h\rfloor, v_h)$ and $(\lfloor\frac{1}{4}v'_{h'}\rfloor, v'_{h'})$. The expected number of frames sampled in each video sequence per iteration of the minimisation are

$$\begin{aligned} s_f =& \frac{1}{n_x}\sum_{(h,h')\in\boldsymbol{M}}(1+v_h-\lfloor\tfrac{1}{4}v_h\rfloor)^{-1}\sum_{i=\lfloor\frac{1}{4}v_h\rfloor}^{v_h} max(min(5,i),\tfrac{1}{20}i)\,, \\ s'_f =& \frac{1}{n_x}\sum_{(h,h')\in\boldsymbol{M}}(1+v'_{h'}-\lfloor\tfrac{1}{4}v'_{h'}\rfloor)^{-1}\sum_{j=\lfloor\frac{1}{4}v'_{h'}\rfloor}^{v'_{h'}} max(min(5,j),\tfrac{1}{20}j). \end{aligned}$$

Assuming the maximum of 15 iterations is typically realised, the time for the minimisation step is

$$T_{\text{MIN}} = 15(s_f + s'_f).$$

The time $\mathcal{T}_{\text{TEST}}$ specifies the required time to perform the measurability range test, threshold test, and coarse robust cost test. The range test is assumed to be trivial since ranges can be computed in closed-form. Just as (s_f, s'_f) denote the expected number of frames sampled for a low-coarse minimisation, (c_f, c'_f) denote the expected number of frames *not* sampled, and are given by

$$\begin{aligned} c_f =& -s_f + \left(\frac{1}{n_x}\sum_{(h,h')\in\boldsymbol{M}}(1+v_h-\lfloor\tfrac{1}{4}v_h\rfloor)^{-1}\sum_{i=\lfloor\frac{1}{4}v_h\rfloor}^{v_h} i\right)\,, \\ c'_f =& -s'_f + \left(\frac{1}{n_x}\sum_{(h,h')\in\boldsymbol{M}}(1+v'_{h'}-\lfloor\tfrac{1}{4}v'_{h'}\rfloor)^{-1}\sum_{j=\lfloor\frac{1}{4}v'_{h'}\rfloor}^{v'_{h'}} j\right)\,. \end{aligned}$$

The threshold test requires the evaluation of every measurable epipolar error for the match being considered, and therefore takes an average time of $c_f + s_f + c'_f + s'_f$. The probability that an incorrect match passes the threshold and range tests is denoted p_t, and assigned to $\frac{1}{4}$ or $\frac{1}{2}$ for known and unknown frame rate ratios respectively. It is expected that this will usually be an over-estimate, and so this measure is conservative. The probability that a correct match passes these tests must be at least $\mathcal{P}(\psi,\mu)$ by definition, since a successful iteration must satisfy the range and cost conditions. Even an unsuccessful iteration for a correct match may pass these tests, due to the presence of a secondary set of synchrony pairs. It is therefore assumed that the probability of a correct match passing the tests is given by

$$\mathcal{P}(\psi,\mu) + (1-\mathcal{P}(\psi,\mu))p_t.$$

The probability of a synchronisation estimate generated from *any* match passing these tests is

$$\mathcal{N}_{\text{P}}(\psi,\mu) = \mu n_x^{-1}(\mathcal{P}(\psi,\mu) + (1-\mathcal{P}(\psi,\mu))p_t) + (1-\mu n_x^{-1})p_t.$$

If the synchronisation estimate passes the threshold test, the coarse robust cost test is necessary, which requires a low sampling of epipolar errors from *all* matches. It seems intuitive to consider the number of errors sampled to be $n_x(s_f + s'_f)$, however this is only appropriate if all matches occupy the same range of frames in each video sequence. If the ranges of frames in which matches are visible are more distributed, the number of measurable errors will typically be lower, for both coarse and non-coarse samplings. To approximate this behaviour, the variables d_f and d'_f are introduced. These specify the average number of matches per frame for which at least one trajectory is visible, for the first and second video sequences respectively. Since all pairings of trajectories are contained in the set of matches, d_f and d'_f are given by

$$d_f = SIZE\left(\{i|\,\exists\, v_{h,i}=1\}\right)^{-1}\sum_{h=1}^{\dot{m}}\dot{m}' v_h\,,$$

$$d'_f = SIZE\left(\{j|\,\exists\, v'_{h',j}=1\}\right)^{-1}\sum_{h'=1}^{\dot{m}'}\dot{m} v'_{h'}\,.$$

The expected number of errors measured in coarse sampling is assumed to be $(s_f d_f + s'_f d'_f)$. Note that when all matches occupy the same range of frames, both d_f and d'_f equate to $\dot{m}\dot{m}'$, yielding a sampling of size $n_x(s_f+s'_f)$ as appropriate. For more distributed matches, the sampling will be reduced as required. The total time for the testing step is therefore

$$\mathcal{T}_{\text{TEST}}(\psi,\mu) = (c_f + c'_f + s_f + s'_f) + \mathcal{N}_{\text{P}}(\psi,\mu)(s_f d_f + s'_f d'_f).$$

The final term, $\mathcal{T}_{\text{EVAL}}$, specifies the required time for a full evaluation of the robust cost function $\mathcal{R}$. Note that some of the epipolar errors associated with the synchronisation estimate have already been evaluated in the previous step. Computing the remainder requires an average time of $(c_f d_f + c'_f d'_f)$. The difficulty lies in determining how frequently this is necessary. The probability that a synchronisation estimate passes both the threshold test and the coarse robust cost test is needed.

It is assumed for simplicity that, regardless of ψ, all coarse robust cost measures $\overline{\mathcal{R}}$ are samples from the same distribution. It is also assumed for the purposes of timing that a full evaluation of $\mathcal{R}$ would yield the same cost as the coarse sampling.

Consider the estimation and evaluation of $k-1$ synchronisation parameter vectors, only for those iterations which pass the measurability and threshold tests. On the next such iteration, a full evaluation is only necessary if the coarse robust cost is lower than all that preceded it. Assuming that all the costs for these iterations are randomly sampled from the same distribution, this occurs with probability k^{-1}. Over k such iterations, the expected number of full evaluations required is given by

$$\sum_{j=1}^{k} \frac{1}{j} = \frac{\Gamma'(k+1)}{\Gamma(k+1)} + \gamma, \tag{5.19}$$

where the fractional term is the Digamma function, and γ is the Euler-Mascheroni constant.

The probability that an iteration requires a full evaluation is therefore dependent on the number of preceding iterations which passed both the range and threshold tests. This number is denoted ξ_d.

The assumed total number of remaining iterations $\mathcal{N}_{\mathrm{R}}(\psi, \mu)$ has a different form for RATSAC and CATSAC. In the case of RATSAC, for the given inputs ψ and μ to function $\mathcal{T}(\psi, \mu)$, $\mathcal{N}_{\mathrm{R}}$ is given by

$$\mathcal{N}_{\mathrm{R}}(\psi, \mu) = \left\lceil \frac{log(\mathcal{F}_{REM}(\mu))}{log(1 - \mathcal{G}(\mu)\mathcal{P}(\psi, \mu))} \right\rceil,$$

for $\mathcal{F}_{REM}$ as defined in equation 5.5. When using CATSAC, $\mathcal{N}_{\mathrm{R}}$ is given by

$$\mathcal{N}_{\mathrm{R}}(\psi, \mu) = \binom{n_x}{c} \left\lceil \frac{log(\mathcal{F}_{REM}(\mu))}{\binom{\mu}{c} log(1 - \mathcal{P}(\psi, \mu))} + \lfloor \mathcal{P}(\psi, \mu) \rfloor \right\rceil,$$

for $\mathcal{F}_{REM}$ given in equation 5.12.

Given $\mathcal{N}_{\mathrm{R}}$, the expected total number of iterations (both completed and remaining) passing both threshold and range tests is

$$\mathcal{N}_{\mathrm{T}}(\psi, \mu) = \mathcal{N}_{\mathrm{R}}(\psi, \mu)\mathcal{N}_{\mathrm{P}}(\psi, \mu) + \xi_d.$$

According to equation 5.19, and the preceding assumptions, and rounding $\mathcal{N}_{\mathrm{T}}$ to the nearest higher integer value, the expected number of remaining synchronisation estimates requiring a full robust cost evaluation is given by

$$\frac{\Gamma'(\lceil \mathcal{N}_{\mathrm{T}}(\psi, \mu) \rceil + 1)}{\Gamma(\lceil \mathcal{N}_{\mathrm{T}}(\psi, \mu) \rceil + 1)} - \frac{\Gamma'(\xi_d + 1)}{\Gamma(\xi_d + 1)}.$$

This yields an average time for the evaluation step of

$$\mathcal{T}_{\mathrm{EVAL}}(\psi, \mu) = \mathcal{N}_{\mathrm{R}}(\psi, \mu)^{-1} \left(\frac{\Gamma'(\lceil \mathcal{N}_{\mathrm{T}}(\psi, \mu) \rceil + 1)}{\Gamma(\lceil \mathcal{N}_{\mathrm{T}}(\psi, \mu) \rceil + 1)} - \frac{\Gamma'(\xi_d + 1)}{\Gamma(\xi_d + 1)} \right) (c_f d_f + c'_f d'_f).$$

This expression is derived under the assumption that coarse robust costs from synchronisation estimates passing the threshold and range tests are drawn from the same distribution. This is obviously incorrect, since different values of ψ may have been used for different iterations. In practice however, it has convenient properties. The estimated number of remaining iterations requiring full evaluations is expected to increase quickly at first, then slow as more iterations are completed. Furthermore, in one sense this is a conservative measure. Consider the case where a low value of ψ is chosen at the start of the robust estimation. After some iterations have passed, a higher ψ value is chosen. For this higher ψ value, the expression for $\mathcal{T}_{\text{EVAL}}$ is too low. For the higher ψ value, more iterations estimating synchronisation from a correct trajectory correspondence are likely to require a full evaluation. In this case, the expression for $\mathcal{T}_{\text{EVAL}}$ should bias the algorithm in favour of choosing a higher ψ.

With the functions $\mathcal{P}(\psi)$ and $\mathcal{T}(\psi)$ now defined, it is possible to choose appropriate ψ values for the robust estimation. Since ψ is neatly bounded, and represents a proportion of frames to be sampled, an analysis of the gradient of the total time function is unnecessary (and complicated, due to $\lfloor . \rfloor$ and $\lceil . \rceil$). To find an approximate ψ minimising the total execution time, a uniform discrete sampling is used. The candidate values for ψ are of the form $max(v_h, v'_{h'})^{-1}w$, for all integers w such that ψ lies between 0 and 1. Such a search is fast compared to an iteration of the robust estimation, and therefore causes no significant decrease in speed.

5.4 Synthetic Tests

In testing RATSAC and CATSAC, the same camera configurations and temporal settings as those in chapters 3 and 4 are employed. For each temporal configuration, RATSAC and CATSAC are tested 1000 times using both linear and piece-wise linear moving scene points, and assuming both a known and unknown frame rate ratio. In each test, 10 trajectories are present in each video sequence, though only 5 trajectories in each sequence are projections of moving scene points viewed by both cameras. All possible pairs of trajectories are considered as matches. Since all trajectories occupy the same range of frames, no more than 10 matches, correct or incorrect, can be found in each test.

Tables 5.1 and 5.2 show the numbers of correct and incorrect matches identified by RATSAC and CATSAC. The true synchronisation parameters for each case are denoted $(\bar{a}, \bar{b})$. These results were obtained by finding an initial synchronisation estimate with the specified robust algorithm, and then refining the estimate via a direct minimisation of the robust cost function $\mathcal{R}$ using Levenberg-Marquardt. A reclassification of matches may have occurred at this stage.

The first point to note is that every case tested has a high probability of correctly classifying the set of matches. The results suggest that a correct match was used to generate an acceptable synchronisation estimate in all tests, since each resulted in a correct classification of the 5 true matches. If a correct classification of true matches is considered a measure for success, and given the total number of tests is $24,000$, then these results clearly indicate that the probability of failure p_f has been satisfied, thereby validating the choice of the function $\mathcal{P}(\psi, \mu)$.

For both algorithms and both types of scene point motion, the second setup with $a = 42.3$ has a greater probability of including an incorrect match. This is to be expected,

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.5	0.5
80	100	42.3	1.1875	100	98.3	1.7
20	100	10.63	4.9375	100	99.7	0.3

(a) Known frame rate ratio, linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.7	0.3
80	100	42.3	1.1875	100	98.1	1.9
20	100	10.63	4.9375	100	99.8	0.2

(b) Unknown frame rate ratio, linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	100	0
80	100	42.3	1.1875	100	99.5	0.5
20	100	10.63	4.9375	100	99.8	0.2

(c) Known frame rate ratio, piecewise-linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	100	0
80	100	42.3	1.1875	100	99.6	0.4
20	100	10.63	4.9375	100	100	0

(d) Unknown frame rate ratio, piecewise-linear scene point motion

Table 5.1: Correct and incorrect matches classified for synthetic test cases, by using RATSAC followed by a minimisation of the robust cost function $\mathcal{R}$

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.7	0.3
80	100	42.3	1.1875	100	97.8	2.2
20	100	10.63	4.9375	100	99.6	0.4

(a) Known frame rate ratio, linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.8	0.2
80	100	42.3	1.1875	100	98.3	1.7
20	100	10.63	4.9375	100	99.7	0.3

(b) Unknown frame rate ratio, linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.9	0.1
80	100	42.3	1.1875	100	99.8	0.2
20	100	10.63	4.9375	100	99.9	0.1

(c) Known frame rate ratio, piecewise-linear scene point motion

Setup				Percentage of tests with classified matches		
n	n'	$\bar{a}$	$\bar{b}$	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	100	0
80	100	42.3	1.1875	100	99.7	0.3
20	100	10.63	4.9375	100	99.9	0.1

(d) Unknown frame rate ratio, piecewise-linear scene point motion

Table 5.2: Correct and incorrect matches classified for synthetic test cases, by using CATSAC followed by a minimisation of the robust cost function $\mathcal{R}$

as the true synchronisation parameters indicate the video sequences have a smaller overlap in time. An incorrect match is more likely to have a low associated error across a smaller range of frames. Although the percentage of tests admitting a single incorrect match may seem significant, (reaching 2.2% for one configuration), it should be noted that the proportion of correct matches in each test is just 0.05. Even when all 5 correct matches are properly classified, there are 25 possible pairings of the remaining trajectories which would constitute incorrect matches. Additionally, it should be noted that the percentage of tests admitting a single incorrect match is typically lower for piecewise-linear scene point motions. Piecewise-linear motions should logically lead to a lower probability that an incorrectly matched pair of trajectories has a low associated error, since it is less likely that *both* linear pieces of the motion yield low epipolar errors for a correct synchronisation of the video sequences.

These results clearly demonstrate the suitability of RATSAC, CATSAC, and the functions $\mathcal{C}$ and $\mathcal{R}$ for classifying matches. However, they provide no explicit information about the quality of the resulting synchronisation estimates. The median Video Synchronisation Errors (VSEs), as described in section 3.5.3, are presented in table 5.3. As expected, these follow the same trends as for non-robust estimation, with a lower median VSE being attained for a known frame rate ratio, and for configurations where the video sequences have a greater overlap in time. The results are of the same order of magnitude to those obtained for $m = 5$ in chapter 3, thereby demonstrating that both RATSAC and CATSAC, followed by a minimisation of $\mathcal{R}$, typically achieve synchronisation to within a small fraction of a frame.

To further analyse the quality of synchrony, the percentages of tests for which the resulting VSE measure was less than 0.5 are listed in table 5.4. A justification for the threshold of 0.5 is given in section 3.5.3. As with the median VSE measures, the results in table 5.4 follow the same trends found in chapters 3 and 4. More tests satisfy the VSE < 0.5 threshold if the frame rate ratio is known. Typically, this condition is also satisfied more often for the piecewise-linear case. Piecewise-linear motions are less likely to generate a near-degenerate case than linear motions, so each correct trajectory match will typically provide more useful visual cues to refine the synchronisation estimate. Note that there may be a correlation between the percentage of tests failing the synchronisation threshold of (VSE < 0.5), and the percentage of tests that admitted a single incorrect match. A comparison of tables 5.4, 5.1, and 5.2 shows this is not necessarily true. Some tests wrongly admitting a single incorrect match still achieve a VSE below 0.5, and other tests with no incorrect matches still do not achieve this fine an estimate of synchrony. The former case can be attributed to incorrect matches that yield low epipolar errors for true synchrony, and may in fact be degenerate matches. The latter case implies that the epipolar error based cost function is reasonably flat around the true synchronisation parameters for all of the identified correct matches. A better synchronisation estimate could therefore be found if additional matching moving scene points are tracked in each video sequence.

Setup				RATSAC		CATSAC	
n	n'	$\bar{a}$	$\bar{b}$	Known b	Unknown b	Known b	Unknown b
80	100	10.63	1.1875	0.022	0.059	0.022	0.061
80	100	42.3	1.1875	0.035	0.099	0.033	0.103
20	100	10.63	4.9375	0.032	0.086	0.03	0.089

(a) Median VSE measures for linear scene point motions

Setup				RATSAC		CATSAC	
n	n'	$\bar{a}$	$\bar{b}$	Known b	Unknown b	Known b	Unknown b
80	100	10.63	1.1875	0.022	0.058	0.022	0.058
80	100	42.3	1.1875	0.034	0.106	0.035	0.107
20	100	10.63	4.9375	0.032	0.093	0.032	0.095

(b) Median VSE measures for piecewise-linear scene point motions

Table 5.3: Median VSE measures for synthetic test cases, for both CATSAC and RATSAC followed by a minimisation of the cost function $\mathcal{R}$

Setup				RATSAC		CATSAC	
n	n'	$\bar{a}$	$\bar{b}$	Known b	Unknown b	Known b	Unknown b
80	100	10.63	1.1875	99.9	99.9	100	99.8
80	100	42.3	1.1875	100	99	99.9	99
20	100	10.63	4.9375	100	99.5	100	99.7

(a) Linear scene point motions

Setup				RATSAC		CATSAC	
n	n'	$\bar{a}$	$\bar{b}$	Known b	Unknown b	Known b	Unknown b
80	100	10.63	1.1875	100	100	100	100
80	100	42.3	1.1875	100	99.4	100	98.9
20	100	10.63	4.9375	100	99.9	100	99.9

(b) Piecewise-linear scene point motions

Table 5.4: Percentages of synthetic tests for which CATSAC and RATSAC, followed by a minimisation of $\mathcal{R}$, achieve synchronisation with a VSE measure less than 0.5

Setup				RATSAC Timing (seconds) - Known b			
n	n'	$\bar{a}$	$\bar{b}$	Initial ψ	Average time	Average time ($\psi = 1$)	Saving
80	100	10.63	1.1875	0.01	0.077	0.518	85.1%
80	100	42.3	1.1875	0.01	0.064	0.529	87.9%
20	100	10.63	4.9375	0.02	0.057	0.180	68.3%

Setup				RATSAC Timing (seconds) - Unknown b			
n	n'	$\bar{a}$	$\bar{b}$	Initial ψ	Average time	Average time ($\psi = 1$)	Saving
80	100	10.63	1.1875	0.01	0.141	0.673	79%
80	100	42.3	1.1875	0.01	0.132	0.673	80.4%
20	100	10.63	4.9375	0.03	0.088	0.257	65.8%

Setup				CATSAC Timing (seconds) - Known b			
n	n'	$\bar{a}$	$\bar{b}$	Initial ψ	Average time	Average time ($\psi = 1$)	Saving
80	100	10.63	1.1875	0.01	0.072	0.384	81.3%
80	100	42.3	1.1875	0.01	0.058	0.393	85.2%
20	100	10.63	4.9375	0.03	0.055	0.133	58.6%

Setup				CATSAC Timing (seconds) - Unknown b			
n	n'	$\bar{a}$	$\bar{b}$	Initial ψ	Average time	Average time ($\psi = 1$)	Saving
80	100	10.63	1.1875	0.03	0.141	0.5	71.8%
80	100	42.3	1.1875	0.03	0.134	0.501	73.2%
20	100	10.63	4.9375	0.06	0.057	0.191	70.1%

Table 5.5: Initial ψ choices, average times (seconds), and percentages of time saved by using an adaptive speedup, for synthetic test cases with linear scene point motions

The final results presented in this section are concerned with the efficiency of RATSAC and CATSAC. Table 5.5 presents the initial ψ chosen for each configuration, the average time required to complete the robust estimation (the subsequent minimisation of $\mathcal{R}$ with Levenberg-Marquardt is omitted from the timing), the average time required for $\psi = 1$, and the percentage of time saved by choosing the an adaptive ψ. Results are presented for linear scene point motions only, since the piecewise-linear times are similar enough to be considered redundant.

The proportion of time saved is calculated as the difference in times, divided by the time for $\psi = 1$. Note that all savings are high, indicating that choosing ψ according to the cost functions $\mathcal{T}(\psi, \mu)$ and $\mathcal{P}(\psi, \mu)$ yields a considerable benefit. For a known frame rate ratio, CATSAC is typically faster than RATSAC, yielding slightly lower times for all cases. CATSAC is also faster for the third configuration when b is unknown. This is

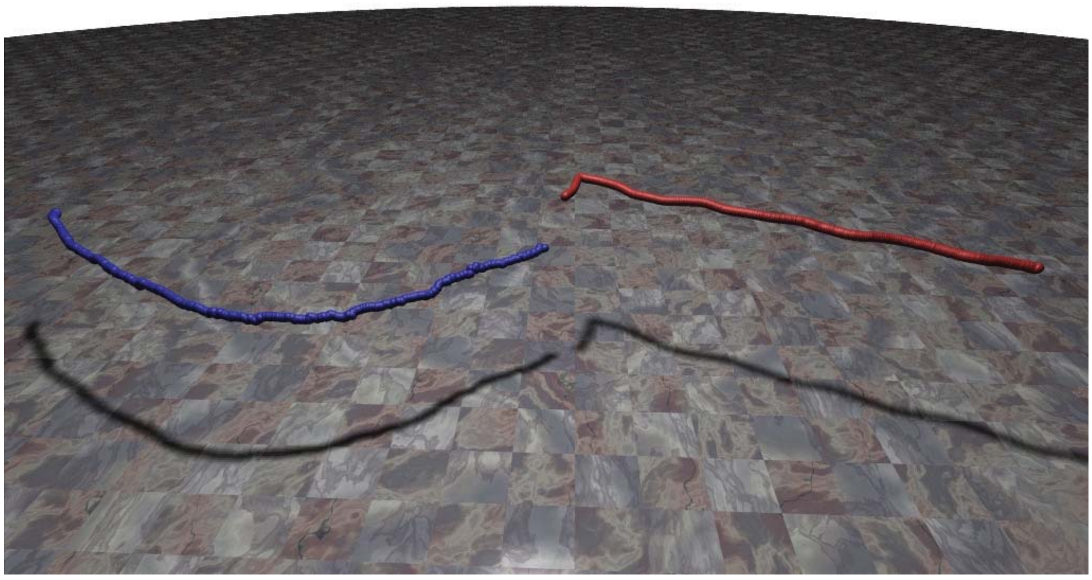


Figure 5.4: Relative positions of the moving cameras' optical centres for the bouncing balls example

most likely due to the difference in ψ, which is greater for CATSAC than for RATSAC in the case where the frame rate ratio is unknown. For both methods, greater ψ values are chosen for the third configuration, where $b = 4.9375$. This can be attributed to the shorter length of the first video sequence. Accordingly, the search for synchrony pairs is faster, and does not dominate the total iteration time to as great an extent as it does in the other two setups.

Although the time taken to achieve robust synchronisation may seem quite short, the required computation will grow quadratically in both the number and length of tracked moving scene points. Choosing optimal ψ values will reduce the time required to synchronise longer video sequences with more moving objects.

5.5 Real Video Sequence Pair Tests

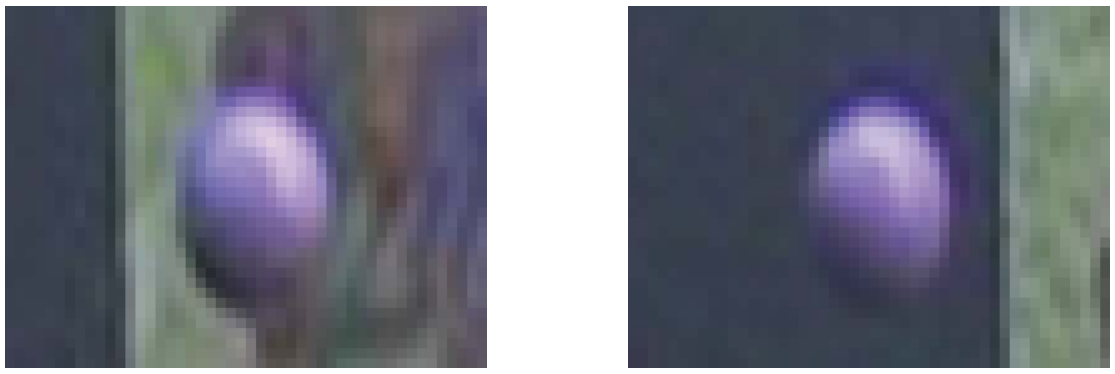
To assess the performance of RATSAC and CATSAC, two hand-held moving cameras recorded footage of three bouncing balls. The independent camera motions and a sparse set of scene points were estimated using the Voodoo camera tracker, which provides implementations of the methods described in [47] and [46]. The two sets of projection matrices were registered, as with the Lego sequence in chapter 3, using a manual identification of corresponding stationary scene points followed by the robust estimation of a 4×4 transformation. The optical centres of the registered projection matrices are shown in figure 5.4.

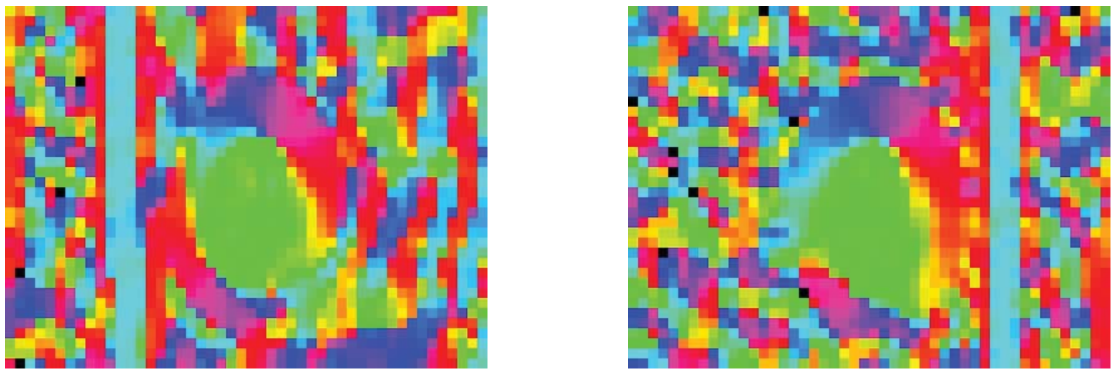
The projections of moving scene points are defined by the centres of the bouncing balls. These were determined by using a combination of gradient and colour measures. The gradient measure was devised by considering the illumination of the balls, and their reflectivity. Figure 5.5(a) depicts magnified images of the balls, which clearly show a brighter region due to the nature of the reflection. Pixel brightness decreases away from this region towards the ball images' periphery. Figure 5.5(b) provides a visualisation of the intensity gradient direction, by mapping this direction to a fully saturated hue colour. Note that the gradient direction change is mostly slow and smooth within the ball pixels, but more 'chaotic' elsewhere. The gradient measure used to detect balls in the images is depicted in figure 5.5(c). This measure is the absolute difference between the gradient direction for one pixel, and that of its immediate neighbour in the direction of steepest intensity descent. By choosing the neighbouring pixel in this way, the measure is also low at the 'peak' of the pixel intensities.

To locate the balls in each frame, a binary image was formed, marking pixels which satisfy both a threshold on the gradient direction measure, and a set of lenient colour based constraints. After morphology and hole-filling, connected components were identified, and filtered on the basis of aspect-ratio and size. Components were also discarded if a significant proportion of their pixels failed to satisfy stricter colour constraints. To determine the image point associated with a ball, the median of a components' pixel locations along each axis was assumed to approximate the projection of the ball's centre.

To group component centres into trajectories in each video sequence, a primitive tracker was constructed. A measure based on both distance and colour similarity between components in successive frames permits correlation based matching. The colour similarity measure helps to avoid ball components from being incorrectly matched with non-ball components that happen to satisfy the feature detection process. Due to occlusions and intermittent failure of the tracker, 22 trajectories were found in the first video sequence, and 19 in the second video sequence. Two of the trajectories in the first video sequence do not represent ball locations, and instead correspond to stationary scene regions which happen to satisfy the feature detection and tracking requirements. The resulting set of matches has size 418 since all possible pairs of trajectories from the two video sequences are included.

Both RATSAC and CATSAC were tested 1000 times for the bouncing ball video sequences, for both known and unknown frame rate ratios. For each test, the initial synchronisation estimate was refined by a minimisation of the robust cost function $\mathcal{R}$ using Levenberg-Marquardt. As with the synthetic tests, this minimisation may cause matches to be reclassified. The quality of the resulting matches and synchronisation are shown in table 5.6. The median VSE measures are computed by comparing the estimated synchronisation against a manual estimate, obtained by visually identifying approximately synchronous frame pairs. For both methods, the median VSE measures indicate synchrony close to one tenth of a frame for a known frame rate ratio, and under one fifth for an unknown frame rate ratio. This is well within the range of the accuracy


(a) Cropped regions of frames around ball locations


(b) Image gradient direction mapped to hue

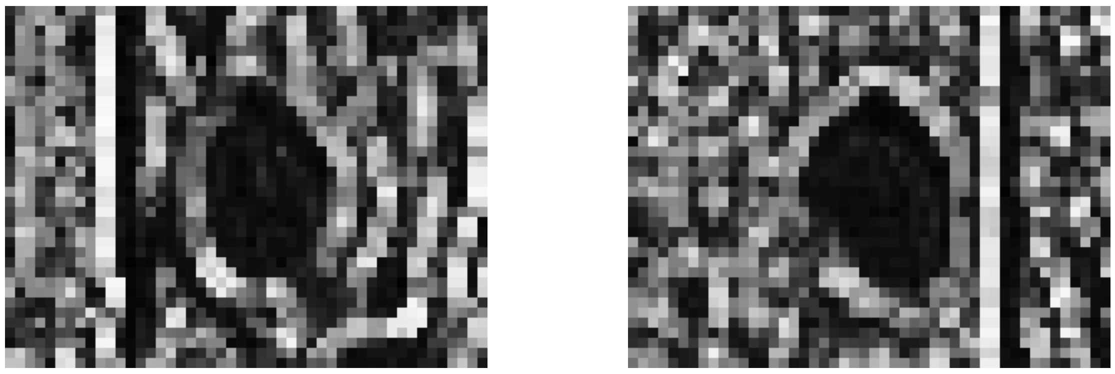

(c) Measure of difference in gradient direction

Figure 5.5: Two examples of the gradient measure used to assist in ball detection

Setup		Percentage of tests with classified matches			VSE
Method	b	19 Correct	20 Correct	0 Incorrect	Median
RATSAC	Known	100	0	100	0.116
RATSAC	Unknown	100	0	100	0.175
CATSAC	Known	100	0	100	0.116
CATSAC	Unknown	99.9	0.1	100	0.175

Table 5.6: VSE measures and matches identified for the bouncing balls example, for both RATSAC and CATSAC, followed by a minimisation of $\mathcal{R}$

of a manual synchronisation. According to the manual synchronisation, there are 19 pairs of trajectories which should be classified as inliers. An additional trajectory pair may also be considered a correct match, but would not be classified as an inlier by the manual synchronisation estimate, which indicates that one trajectory begins just one quarter of a frame in time after the other ends. Accordingly, it is expected that an approximately correct synchronisation should yield either 19 or 20 correct matches. Note that this occurred in all tests. Additionally, in *every* test, no incorrect match was wrongly classified. This clearly demonstrates that both RATSAC and CATSAC, when used with the robust cost function $\mathcal{R}$, typically yield accurate classification of both correct and incorrect matches in a real video sequence case.

An assessment of the speedups is provided in table 5.7. For both methods, the initial ψ value is small, indicating a preference for a low-sampled synchrony pair search. Compared with the synthetic tests, the proportions of time saved by using such a low sampling are much lower. This is likely due to the shorter trajectory sizes, which causes evaluation to dominate the average execution time per iteration. Unlike the synthetic tests, RATSAC has a significantly lower average execution time. This can be explained by an analysis of the ψ values chosen. For both methods, 0.0625 is chosen when the frame rate ratio is unknown. Assuming this value is used throughout the entire process, and that 19 matches are classified before the associated termination condition is reached, RATSAC will perform 198 iterations. CATSAC, by testing every match, is guaranteed to perform at least a single sweep, and must therefore evaluate a minimum of 418 synchronisation estimates. Note that this example alone should not imply a preference for the RATSAC method, as evidenced by the faster CATSAC times during synthetic testing as shown in table 5.5.

The quality of the synchronisation is further illustrated by figure 5.6, which shows cropped regions of frames side-by-side, with their vertical placement determined by synchronisation estimated with RATSAC for an unknown frame rate ratio. Figure 5.7 shows a reconstruction of the 3D ball trajectories, for matches classified by CATSAC. The re-

Setup		Timing (seconds)			
Method	b	Initial ψ	Average time	Average time ($\psi = 1$)	Saving
RATSAC	Known	0.0417	0.064	0.078	17.9%
RATSAC	Unknown	0.0625	0.085	0.136	37.5%
CATSAC	Known	0.0625	0.117	0.204	42.6%
CATSAC	Unknown	0.0625	0.139	0.321	56.7%

Table 5.7: Initial ψ choices, average times (seconds) and percentages of time saved by using an adaptive speedup, for the bouncing balls example

constructions were obtained by assuming that the balls' motion for each match can be described by a 3D bezier curve parameterised by time. Such curves can then be estimated using linear algebra methods, and further refined to minimise reprojection error, incorporating information from both video sequences. Each match was manually associated with one of the three balls, and the resulting locations have been coloured accordingly to aid in the visualisation of the 3D motions.

5.6 Conclusions

A framework for choosing an appropriate level of speedup for robust estimations has been introduced. A trade-off is sought between the speed and reliability of each iteration, such that the overall execution time is reduced. The benefits of an adaptive speedup are combined with an adaptive termination condition, and Bayesian inference is used to select a new speedup parameter whenever a superior model estimate is found. This approach forms the basis for two new algorithms, RATSAC and CATSAC, which can be used for random and exhaustive sampling of data subsets respectively. The underlying premise is general enough to be applicable to a variety of speedup mechanisms and robust estimation problems.

Specific instances of these algorithms have been devised for the robust synchronisation of video sequences recorded by moving cameras, employing an accelerated search for synchrony pairs. These algorithms have been tested on both a real video sequence pair, and a set of synthetic cases which cover a variety of temporal configurations and scene point motions.

The synthetic tests validate the approximate models for time and probability, and demonstrate that a random sampling of frames for the synchrony pair search can result in a hefty speedup. Tests on the bouncing balls example verify that both algorithms are practically applicable. Furthermore, the accuracy of match classification and low VSE measures in all test configurations establishes the suitability of the robust cost function $\mathcal{R}$ in assessing a synchronisation estimate, and distinguishing between inliers and outliers.

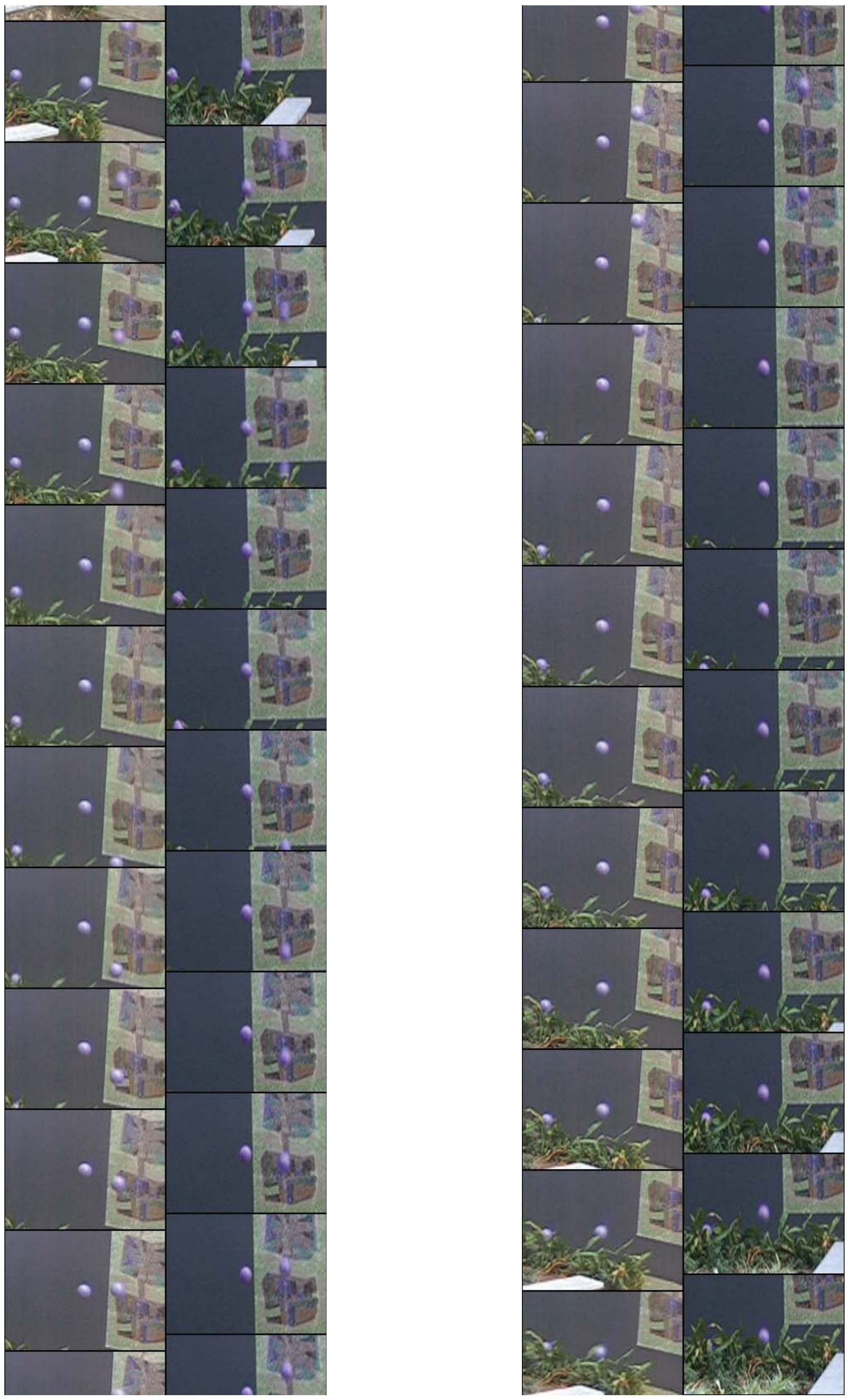


Figure 5.6: Side-by-side display of synchronised video frames, cropped to a region around a match identified by RATSAC assuming a known frame rate ratio

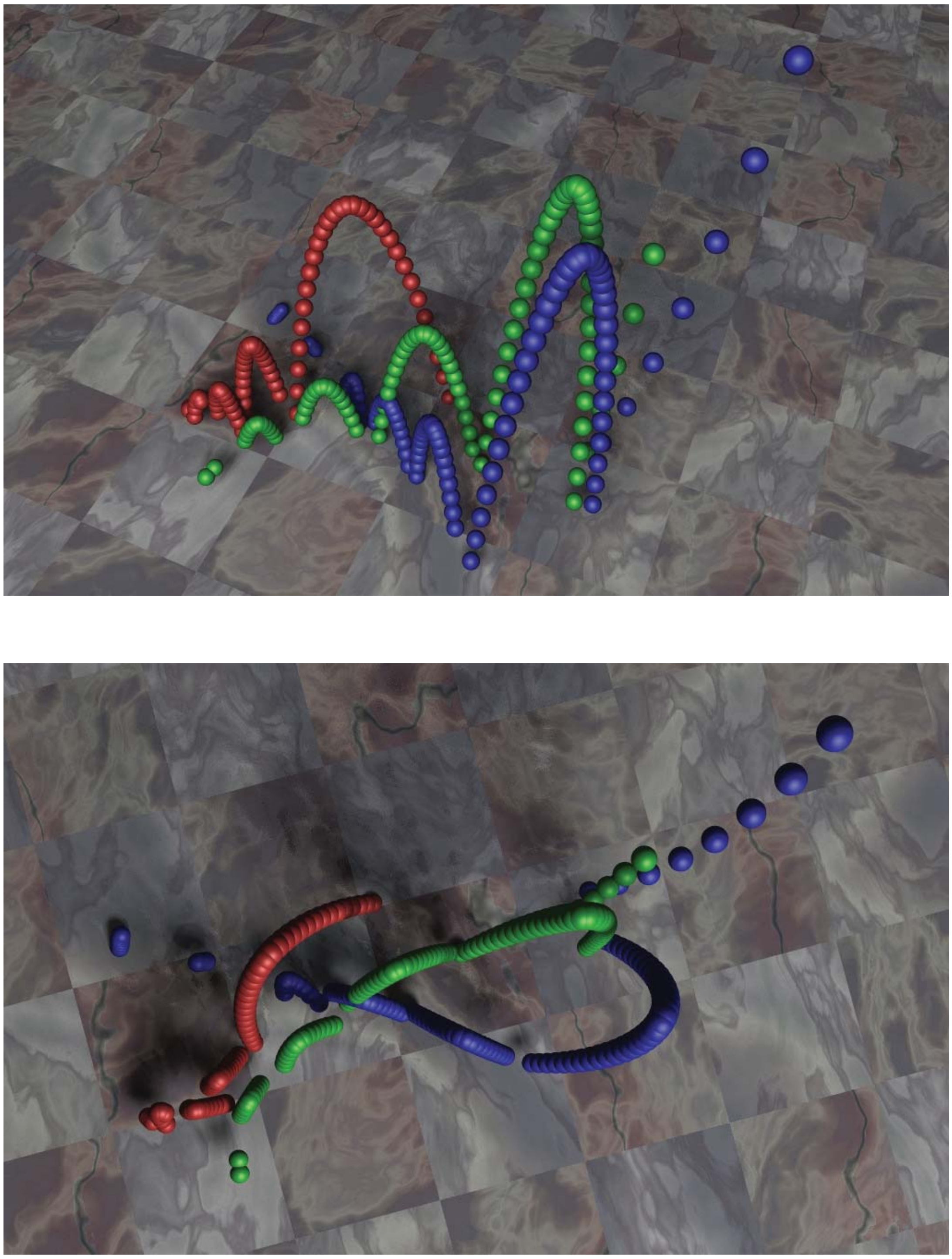


Figure 5.7: A reconstruction of 3D ball trajectories, projected to virtual cameras, for matches identified by CATSAC assuming an unknown frame rate ratio

Chapter 6

ESTIMATING A PROJECTIVE TRANSFORMATION WITH MOVING SCENE POINTS

In previous chapters, methods have been presented to synchronise a pair of independently moving cameras. These methods have assumed that all spatial information concerning the motion of the cameras through space is known. We now consider the case where the spatial information is incomplete. Specifically, the sets of projection matrices describing the motion for each camera are uncorrelated.

6.1 Problem Formulation

Consider a scene consisting of both stationary and dynamic scene points, observed by two independently moving video cameras. It is assumed for now that the video sequences are perfectly synchronised such that the frame t in video 1 sequence has the same time-stamp as frame t in video sequence 2. We define

$\bar{\boldsymbol{x}}_h$	as the stationary scene point h $(h \in [1, s])$,
$\bar{\boldsymbol{y}}_{h,t}$	as the moving scene point h at time t $(h \in [1, m])$,
$\bar{\mathsf{P}}_t$	as the projection matrix describing camera 1 at time t,
$\bar{\mathsf{P}}'_t$	as the projection matrix describing camera 2 at time t.

Image point information is also available, defined as

$$\begin{aligned}\bar{\boldsymbol{q}}_{h,t} &\sim \bar{\mathsf{P}}_t \bar{\boldsymbol{x}}_h,\\ \bar{\boldsymbol{q}}'_{h,t} &\sim \bar{\mathsf{P}}'_t \bar{\boldsymbol{x}}_h,\\ \bar{\boldsymbol{p}}_{h,t} &\sim \bar{\mathsf{P}}_t \bar{\boldsymbol{y}}_{h,t},\\ \bar{\boldsymbol{p}}'_{h,t} &\sim \bar{\mathsf{P}}'_t \bar{\boldsymbol{y}}_{h,t}.\end{aligned}$$

Consider each set of projection matrices distorted by an arbitrary projective transformation, such that

$$\begin{aligned}\bar{\mathsf{M}}_t &= \bar{\mathsf{P}}_t \mathsf{N},\\ \bar{\mathsf{M}}'_t &= \bar{\mathsf{P}}'_t \mathsf{N}',\end{aligned}$$

for some pair of full rank 4×4 matrices $(\mathsf{N}, \mathsf{N}')$. The set of 3D stationary points is also distorted by each of the transformations, defining

$$\bar{\boldsymbol{z}}_j \sim \mathbf{N}^{-1}\bar{\boldsymbol{x}}_h,$$
$$\bar{\boldsymbol{z}}'_j \sim \mathbf{N}'^{-1}\bar{\boldsymbol{x}}_h.$$

Given noisy estimates of $(\bar{\boldsymbol{q}}_{h,t}, \bar{\boldsymbol{q}}'_{h,t})$, structure from motion algorithms such as those used by ICARUS [19] and Voodoo [47], provide noisy estimates of the projection matrices and 3D scene points for each video sequence. Consequently,

$$\boldsymbol{z}_j \approx \bar{\boldsymbol{z}}_j,$$
$$\boldsymbol{z}'_j \approx \bar{\boldsymbol{z}}'_j,$$
$$\mathbf{M}_t \approx \bar{\mathbf{M}}_t,$$
$$\mathbf{M}'_t \approx \bar{\mathbf{M}}'_t$$

are all available, where $\approx$ denotes approximate equality up to scale. Note this provides two separate sets of 3D points, $\{\boldsymbol{z}_j\}$ and $\{\boldsymbol{z}'_j\}$. The unknown projective transformation that can join these reconstructions together is denoted

$$\bar{\mathbf{H}} \sim \mathbf{N}^{-1}\mathbf{N}'.$$

We denote a noisy estimate of this matrix as $\mathbf{H}$. An estimate of $\mathbf{H}$ can be found by using corresponding 3D scene points from the two point sets, each of which supplies the constraint that

$$\boldsymbol{z}_j \sim \mathbf{H}\boldsymbol{z}'_j.$$

A minimum of 5 such correspondences are needed. In this chapter, we consider the case where there are insufficient stationary point correspondences to estimate $\mathbf{H}$, and derive constraints for projections of the moving scene points that can be used as an alternative.

Sufficient stationary 3D point correspondences may be unavailable due to the fact that only a sparse set of features has been tracked in each video, or because of constraints in the scene as illustrated in example 6.1.

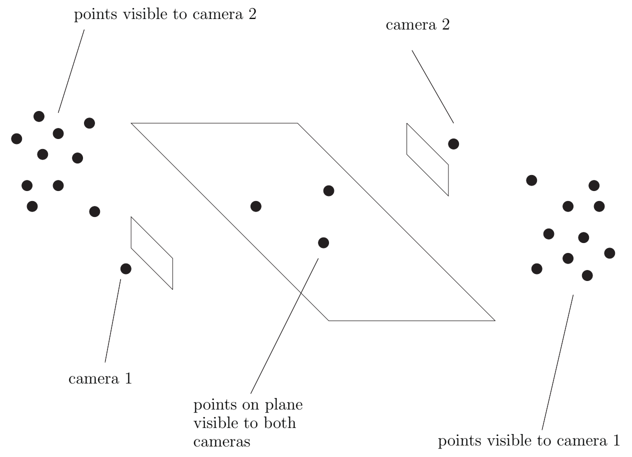
6.2 A Constraint For Moving Scene Point Correspondences

At time t, a moving scene point projects to estimated image location $\boldsymbol{p}_t$ in the first camera, and $\boldsymbol{p}'_t$ in the second camera. Since projection matrix estimates $\mathbf{M}_t$ and $\mathbf{M}'_t$ are known, the rays in space that form these projections can be computed. We seek a projective transformation $\mathbf{H}$ that distorts one of the rays such that the pair of rays intersect in 3D space.

This introduces the question of how to best represent a ray in 3D space. One possibility is to parameterise a ray by some pair of 3D points that it passes through. Suppose a ray in space passes through the points $\boldsymbol{p} = [p_1, p_2, p_3, 1]^\top$ and $\boldsymbol{q} = [q_1, q_2, q_3, 1]^\top$. The matrix

$$\mathbf{R} = \begin{bmatrix} p_1 & p_2 & p_3 & 1 \\ q_1 & q_2 & q_3 & 1 \end{bmatrix} \tag{6.1}$$

Example 6.1 (A Case of Insufficient Stationary Correspondences) *Consider two moving cameras on opposite sides of a sports field. Each camera can view objects behind the other camera (such as spectator stands), but the only stationary 3D points visible to both cameras lie on the playing field. Planar correspondences are insufficient to estimate a projective transformation.*


points visible to camera 2
camera 2
camera 1
points on plane visible to both cameras
points visible to camera 1

is a simple representation of the ray. Unfortunately, the representation is far from unique. Note that pre-multiplying R by any full-rank 2×2 matrix produces a matrix which represents the same ray in space. This ambiguity introduces unnecessary complexity when attempting to derive a constraint for moving scene points.

An alternative representation is the Plücker ray, which, for a given ray in 3D space, is unique up to scale. The following sections describe this representation, and its applicability to the problem described above. A number of theorems and proofs are also provided in appendix A. While similar theorems are doubtless available elsewhere, they have been included for completeness, and to assist readers unfamiliar with Plücker rays.

6.2.1 Plücker Rays

A ray in 3D space passing through distinct points $\boldsymbol{p}$ and $\boldsymbol{q}$ can be represented as a 4×4 anti-symmetric matrix given by

$$\mathsf{R} = \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top,$$

and is defined as the Plücker matrix. This can be seen to represent a ray in space, since the set of planes along which the ray lies is equal to the left or right null-space of the matrix, as described in theorem A.1.

The dual of the Plücker matrix can be formed in a similar way, using planes instead of points. If a ray is defined by the intersection of the two planes $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$, then the dual

Plücker matrix for this ray is given by

$$\mathsf{R}^* = \boldsymbol{\pi}_1\boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2\boldsymbol{\pi}_1^\top.$$

By theorem A.2, for a given ray in 3D space, the Plücker matrix, and the dual Plücker matrix, are unique up to a scale.

A Plücker matrix is anti-symmetric and has the form

$$\mathsf{R} = \begin{bmatrix} 0 & r_1 & r_2 & r_3 \\ -r_1 & 0 & r_4 & -r_5 \\ -r_2 & -r_4 & 0 & r_6 \\ -r_3 & r_5 & -r_6 & 0 \end{bmatrix},$$

and can therefore be parameterised using only the upper triangular elements. The ordering of the labels for these elements is irrelevant, but has been chosen here to be row-wise, with the fifth negated for convenience in later equations. This is the same ordering as that used in [21]. To facilitate a conversion between matrix and vector forms, $\mathcal{V}_{Ray}$ is defined as the function such that

$$\boldsymbol{r} = [r_1, r_2, r_3, r_4, r_5, r_6]^\top = \mathcal{V}_{Ray}(\mathsf{R}).$$

Note that $\mathcal{V}_{Ray}$ is a linear function, so

$$\mathcal{V}_{Ray}(\mathsf{R} + \mathsf{R}') = \mathcal{V}_{Ray}(\mathsf{R}) + \mathcal{V}_{Ray}(\mathsf{R}')$$

For convenience, an operator notation is also adopted, whereby

$$(\boldsymbol{p} \curlywedge \boldsymbol{q}) = \mathcal{V}_{Ray}(\boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top).$$

This is similar to the notation used in [21], in which the $\wedge$ operator defines the formation of a ray from two points. Since $\wedge$ denotes a logical 'and' operator, $\curlywedge$ is used here instead. For points $\boldsymbol{p} = [p_1, p_2, p_3, p_4]^\top$ and $\boldsymbol{q} = [q_1, q_2, q_3, q_4]^\top$, $(\boldsymbol{p} \curlywedge \boldsymbol{q})$ expands to

$$(\boldsymbol{p} \curlywedge \boldsymbol{q}) = \left[\begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix}, \begin{vmatrix} p_1 & p_3 \\ q_1 & q_3 \end{vmatrix}, \begin{vmatrix} p_1 & p_4 \\ q_1 & q_4 \end{vmatrix}, \begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix}, -\begin{vmatrix} p_2 & p_4 \\ q_2 & q_4 \end{vmatrix}, \begin{vmatrix} p_3 & p_4 \\ q_3 & q_4 \end{vmatrix} \right]^\top$$

For convenience, a matrix J is introduced, defined as

$$\mathsf{J} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that pre-multiplying a vector by J will reverse the order of its elements. Given Plücker ray vector $\boldsymbol{r}$ and dual $\boldsymbol{r}^*$, theorem A.3 specifies that

$$\boldsymbol{r} \sim \mathsf{J}\boldsymbol{r}^*$$

Additionally, theorem A.4 specifies that two rays $\boldsymbol{r}$ and $\boldsymbol{r}'$ intersect if and only if

$$\boldsymbol{r}^\top \mathsf{J}\boldsymbol{r}' = 0. \tag{6.2}$$

Furthermore, by theorem A.5, a vector $\boldsymbol{r}$ represents a valid ray in space if and only if

$$\boldsymbol{r}^\top \mathsf{J}\boldsymbol{r} = 0.$$

Results equivalent to theorems A.1, A.2, A.3, A.4 and A.5 can all be found in [21]. The equivalence between between equation 6.2 and the co-planarity of four scene points, as used in the proofs for theorems A.4 and A.5, can also be found in [21].

6.2.2 The Ray Projection Matrix

A general ray in 3D space projects to a straight line in a 2D image. Given such a ray, and a 3×4 projection matrix, the resulting line in the image can be computed according to theorem A.7. A corresponding expression for projecting a ray vector is made possible by introducing a 6×3 *ray projection matrix.* Given a 3×4 projection matrix

$$\mathsf{M} = \begin{bmatrix} \boldsymbol{m}_1^\top \\ \boldsymbol{m}_2^\top \\ \boldsymbol{m}_3^\top \end{bmatrix},$$

the corresponding ray projection matrix is given by

$$\hat{\mathsf{M}} = \mathsf{J}[(\boldsymbol{m_2} \curlywedge \boldsymbol{m_3}), (\boldsymbol{m_3} \curlywedge \boldsymbol{m_1}), (\boldsymbol{m_1} \curlywedge \boldsymbol{m_2})].$$

This definition is the transpose of that used in [21] if J is omitted. The differing definition is used since it permits a more elegant form of later equations. The ray projection matrix can be used to project rays in 3D space to lines in an image, or to back-project from an image point to recover the ray of projection. According to theorem A.8, a ray vector $\boldsymbol{r}$ projects to a line in an image given by $\hat{\mathsf{M}}^\top \mathsf{J}\boldsymbol{r}$. Theorem A.9 specifies that the ray of 3D points projecting to image location $\boldsymbol{p}$ is given by $\hat{\mathsf{M}}\boldsymbol{p}$. Similar proofs for theorems A.7 and A.8 are given in [21].

The ray projection matrix provides an elegant way of defining the fundamental matrix relating two cameras. Given two projection matrices M and M', we denote the corresponding ray projection matrices as $\hat{\mathsf{M}}$ and $\hat{\mathsf{M}}'$, and define $\boldsymbol{p}$ and $\boldsymbol{p}'$ as corresponding image points. The ray projection matrix $\hat{\mathsf{M}}'$ maps an image point $\boldsymbol{p}'$ in the second camera to the ray in space that forms this projection. The matrix $\hat{\mathsf{M}}^\top \mathsf{J}$ then projects this ray to a line in the first camera that passes through $\boldsymbol{p}$. Combining these steps, we arrive at the equation

$$\boldsymbol{p}^\top \hat{\mathsf{M}}^\top \mathsf{J} \hat{\mathsf{M}}' \boldsymbol{p}' = 0.$$

The matrix $\hat{\mathsf{M}}^\top \mathsf{J} \hat{\mathsf{M}}'$ maps image point $\boldsymbol{p}'$ to its corresponding epipolar line in the first camera, and vice versa. This matrix must therefore be equal to the fundamental matrix relating the two cameras. An equivalent expression can be found in [15].

6.2.3 Ray Transformation

A full-rank 4×4 projective transformation H describes a distortion applied to 3D points in the scene. For

$$\mathsf{H} = \begin{bmatrix} \boldsymbol{h}_1^\top \\ \boldsymbol{h}_2^\top \\ \boldsymbol{h}_3^\top \\ \boldsymbol{h}_4^\top \end{bmatrix},$$

the corresponding *ray transformation* matrix is given by

$$\hat{\mathsf{H}} = \begin{bmatrix} (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_4)^\top \\ (\boldsymbol{h}_2 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_4 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_3 \curlywedge \boldsymbol{h}_4)^\top \end{bmatrix}.$$

This matrix applies the same distortion to rays that H applies to 3D points, as specified by theorem A.10.

Not every 6×6 matrix $\hat{\mathsf{H}}$ corresponds to a projective transformation that distorts ray vectors. Every ray $\boldsymbol{r}$ satisfies the constraint that $\boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0$. Similarly, the transformed rays must also satisfy this constraint. Therefore

$$\boldsymbol{r}^\top \hat{\mathsf{H}}^\top \mathsf{J} \hat{\mathsf{H}} \boldsymbol{r} = 0 \quad \forall \boldsymbol{r} \quad \text{where} \quad \boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0.$$

Therefore, both $\hat{\mathsf{H}}^\top \mathsf{J} \hat{\mathsf{H}}$ and J describe the same quadratic surface in 6D space. The space of symmetric matrices representing this surface is $k\mathsf{J}$ for any non-zero scalar k. Therefore, the matrix $\hat{\mathsf{H}}$ must satisfy the constraint that

$$\hat{\mathsf{H}}^\top \mathsf{J} \hat{\mathsf{H}} \sim \mathsf{J}.$$

This equation imposes 20 quadratic constraints on the elements of $\hat{\mathsf{H}}$. Where $[\hat{\mathsf{H}}]_{*,u}$ denotes column u of matrix $\hat{\mathsf{H}}$, 18 of these constraints are given by

$$[\hat{\mathsf{H}}]_{*,u}^\top \, \mathsf{J} \, [\hat{\mathsf{H}}]_{*,v} = 0, \quad \text{where } u \geq v, \quad u + v \neq 7.$$

The remaining two constraints are given by

$$\begin{aligned} [\hat{\mathsf{H}}]_{*,1}^\top \, \mathsf{J} \, [\hat{\mathsf{H}}]_{*,6} - [\hat{\mathsf{H}}]_{*,3}^\top \, \mathsf{J} \, [\hat{\mathsf{H}}]_{*,4} &= 0, \\ [\hat{\mathsf{H}}]_{*,2}^\top \, \mathsf{J} \, [\hat{\mathsf{H}}]_{*,5} - [\hat{\mathsf{H}}]_{*,3}^\top \, \mathsf{J} \, [\hat{\mathsf{H}}]_{*,4} &= 0. \end{aligned}$$

Together with scale invariance, this means that $\hat{\mathsf{H}}$ has 15 degrees of freedom, the same number as a 4×4 projective transformation.

6.2.4 The Moving Point Constraint

Recalling the original problem, we have sets of projection matrices such that M_i describes the first camera for frame i, and M'_j describes the second camera for frame j. We seek a 4×4 projective transformation H that will spatially register the two sets of projection matrices, and equivalently their associated scene reconstructions. The equivalent ray projection matrices are denoted $\hat{\mathsf{M}}_i$ and $\hat{\mathsf{M}}'_j$ for the first and second video sequences respectively. The 6×6 ray transformation equivalent to H is denoted $\hat{\mathsf{H}}$. Assuming frame i from the first video sequence is exactly synchronous with frame j from the second video sequence, and a moving scene point projects to locations $\boldsymbol{p}$ and $\boldsymbol{p}'$ respectively, then

$$\boldsymbol{p}^\top \hat{\mathsf{M}}_i^\top \mathsf{J} \hat{\mathsf{H}} \hat{\mathsf{M}}'_j \boldsymbol{p}' = 0, \tag{6.3}$$

where the 3×3 matrix $\hat{\mathsf{M}}_i^\top \mathsf{J} \hat{\mathsf{H}} \hat{\mathsf{M}}'_j$ is the fundamental matrix describing the epipolar geometry relating the projection matrices $\mathsf{M}_i \mathsf{H}$ and M'_j. This constraint can theoretically be used to estimate the matrix $\hat{\mathsf{H}}$, and equivalently the 4×4 projective transformation H that spatially registers the two video sequences.

Estimation of the 6×6 ray transformation matrix has been examined before by Bartoli and Sturm in [2]. The 6×6 matrix is referred to as the *line motion* matrix, and is estimated using corresponding pairs of rays in two 3D reconstructions. Note that the relation for corresponding rays is ($\boldsymbol{r}' \sim \hat{\mathsf{H}} \boldsymbol{r}$), which specifies five linear constraints on $\hat{\mathsf{H}}$ per pair of rays. The relation in equation 6.3 of ($\boldsymbol{r}^\top \mathsf{J} \hat{\mathsf{H}} \boldsymbol{r}' = 0$) only specifies one linear constraint on $\hat{\mathsf{H}}$ per ray pair.

The correlation of equation 6.3 with the epipolar constraint is illustrated in example 6.2, which demonstrates that estimating the fundamental matrix relating a pair of stationary cameras is *exactly* equivalent to estimating a subset of the elements of the ray transformation matrix $\hat{\mathsf{H}}$.

6.3 Estimation of the Ray Transformation Matrix

As illustrated in the previous section, a single moving scene point provides one constraint on the ray transformation matrix per pair of synchronous frames. The associated algebraic error measure for a point pair $(\boldsymbol{p}, \boldsymbol{p}')$ and ray projection matrix pair $(\hat{\mathsf{M}}, \hat{\mathsf{M}}')$ is

$$\mathcal{H}_{\text{ALG}}(\hat{\mathsf{H}}, \boldsymbol{p}, \boldsymbol{p}', \hat{\mathsf{M}}, \hat{\mathsf{M}}') = \boldsymbol{p}^\top \hat{\mathsf{M}}^\top \mathsf{J} \hat{\mathsf{H}} \hat{\mathsf{M}}' \boldsymbol{p}'.$$

Using the notation from chapter 5, it is assumed that a number of moving points have been observed by both cameras, and the resulting trajectories have been identified as a match. Specifically, a moving point projects to image location $\boldsymbol{p}_{h,i}$ in frame i of the first

Example 6.2 (Estimating $\hat{\mathsf{H}}$ for Stationary Cameras) *If we consider the case of two stationary cameras, and for simplicity define*

$$\mathsf{M}_i = \mathsf{M}'_j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \forall i, j,$$

then the corresponding ray projection matrices are

$$\hat{\mathsf{M}}_i^\top = \hat{\mathsf{M}}_j'^\top = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \forall i, j.$$

We wish to find the projective transformation that registers the two cameras. If the ray transformation $\hat{\mathsf{H}}$ is defined as

$$\hat{\mathsf{H}} = \begin{bmatrix} \hat{h}_{1,1} & \hat{h}_{1,2} & \hat{h}_{1,3} & \hat{h}_{1,4} & \hat{h}_{1,5} & \hat{h}_{1,6} \\ \hat{h}_{2,1} & \hat{h}_{2,2} & \hat{h}_{2,3} & \hat{h}_{2,4} & \hat{h}_{2,5} & \hat{h}_{2,6} \\ \hat{h}_{3,1} & \hat{h}_{3,2} & \hat{h}_{3,3} & \hat{h}_{3,4} & \hat{h}_{3,5} & \hat{h}_{3,6} \\ \hat{h}_{4,1} & \hat{h}_{4,2} & \hat{h}_{4,3} & \hat{h}_{4,4} & \hat{h}_{4,5} & \hat{h}_{4,6} \\ \hat{h}_{5,1} & \hat{h}_{5,2} & \hat{h}_{5,3} & \hat{h}_{5,4} & \hat{h}_{5,5} & \hat{h}_{5,6} \\ \hat{h}_{6,1} & \hat{h}_{6,2} & \hat{h}_{6,3} & \hat{h}_{6,4} & \hat{h}_{6,5} & \hat{h}_{6,6} \end{bmatrix},$$

then the constraint described in equation 6.3 becomes

$$\begin{aligned} 0 &= \boldsymbol{p}^\top \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathsf{J} \begin{bmatrix} \hat{h}_{1,1} & \hat{h}_{1,2} & \hat{h}_{1,3} & \hat{h}_{1,4} & \hat{h}_{1,5} & \hat{h}_{1,6} \\ \hat{h}_{2,1} & \hat{h}_{2,2} & \hat{h}_{2,3} & \hat{h}_{2,4} & \hat{h}_{2,5} & \hat{h}_{2,6} \\ \hat{h}_{3,1} & \hat{h}_{3,2} & \hat{h}_{3,3} & \hat{h}_{3,4} & \hat{h}_{3,5} & \hat{h}_{3,6} \\ \hat{h}_{4,1} & \hat{h}_{4,2} & \hat{h}_{4,3} & \hat{h}_{4,4} & \hat{h}_{4,5} & \hat{h}_{4,6} \\ \hat{h}_{5,1} & \hat{h}_{5,2} & \hat{h}_{5,3} & \hat{h}_{5,4} & \hat{h}_{5,5} & \hat{h}_{5,6} \\ \hat{h}_{6,1} & \hat{h}_{6,2} & \hat{h}_{6,3} & \hat{h}_{6,4} & \hat{h}_{6,5} & \hat{h}_{6,6} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{p}' \\ &= \boldsymbol{p}^\top \begin{bmatrix} \hat{h}_{4,3} & -\hat{h}_{4,5} & \hat{h}_{4,6} \\ -\hat{h}_{2,3} & \hat{h}_{2,5} & -\hat{h}_{2,6} \\ \hat{h}_{1,3} & -\hat{h}_{1,5} & \hat{h}_{1,6} \end{bmatrix} \boldsymbol{p}'. \end{aligned}$$

camera, and $\boldsymbol{p}'_{h',j}$ in frame j of the second camera, where h and h' uniquely identify the trajectory in each video sequence. A set of trajectory pairs known, or assumed, to be correspondences is denoted $\boldsymbol{M}_C$.

Proceeding in a similar fashion to fundamental matrix or homography estimation, as described in chapter 2, we consider finding the 6×6 matrix $\hat{\mathbf{H}}$ which minimises the sum of squares of these $\mathcal{H}_{\mathrm{ALG}}$ measures. Since frame pairs are unlikely to be exactly synchronous, given synchronisation parameters (a, b) and frame i from the first video, the frame from the second video with integer index closest to $a + bi$ (denoted $\lfloor a + bi \rceil$) is used. The estimate $\hat{\mathbf{H}}$ is therefore chosen to be the matrix that minimises the expression

$$\begin{aligned}\mathcal{R}_{\mathrm{ALG}}(\hat{\mathbf{H}}, a, b) = & \sum_{(h,h')\in\boldsymbol{M}_C} \sum_{i=0}^{n-1} \mathcal{H}_{\mathrm{ALG}}(\hat{\mathbf{H}}, \boldsymbol{p}_{h,i}, \boldsymbol{p}_{h',\lfloor a+bi \rceil}, \mathbf{M}_i, \mathbf{M}'_{\lfloor a+bi \rceil})^2 \quad + \\ & \sum_{(h,h')\in\boldsymbol{M}_C} \sum_{j=0}^{n'-1} \mathcal{H}_{\mathrm{ALG}}(\mathbf{J}\hat{\mathbf{H}}^\top\mathbf{J}, \boldsymbol{p}'_{h',j}, \boldsymbol{p}_{h,\lfloor a'+b'j \rceil}, \mathbf{M}'_j, \mathbf{M}_{\lfloor a'+b'j \rceil})^2 \\ & \text{where } a' = ab^{-1}, \quad b' = b^{-1}, \\ & \text{and } \|\hat{\mathbf{H}}\|_{\mathrm{F}} = 1.\end{aligned} \tag{6.4}$$

As in the case of the synchronisation cost functions presented in chapters 3 and 5, summands corresponding to out-of-bounds frame indices or occluded image points are assumed to be 0.

Note that expressing $\mathcal{R}_{\mathrm{ALG}}$ as a function of the elements of the 4×4 projective transformation $\mathbf{H}$ yields a quartic function, and is therefore not amenable to linear algebra methods. Conversely, the cost function is quadratic in the elements of the matrix $\hat{\mathbf{H}}$.

The cost function described by equation 6.4 is inappropriate for reliably estimating $\hat{\mathbf{H}}$. The algebraic error has no meaningful geometric interpretation, since the projection matrices, and equivalently 3D rays, differ from truth by unknown *projective* distortions. Additionally, each projection matrix is only defined up to a single scale. Without a means to specify an appropriate scale for each projection matrix, this ambiguity equates to only defining each squared summand up to a positive scale. Estimating $\hat{\mathbf{H}}$ in this fashion would typically produce a result which does not satisfy

$$\hat{\mathbf{H}}^\top \mathbf{J} \hat{\mathbf{H}} \sim \mathbf{J}. \tag{6.5}$$

It is also unlikely that the matrix $\hat{\mathbf{H}}$ minimising $\mathcal{R}_{\mathrm{ALG}}$ will be 'close' to satisfying the relation in equation 6.5, particularly for cases where few of the summands are measurable, since this relation specifies a high number of independent quadratic constraints. Furthermore, the quadratic constraints arising from equation 6.5 are not positive definite or semi-definite, so they have no equivalent linear form. The quadratic constraints need to be included in the estimation process, to ensure that $\hat{\mathbf{H}}$ satisfies equation 6.5, and therefore corresponds exactly to some 4×4 projective transformation $\mathbf{H}$.

If synchronisation parameters (a, b) are known (or assumed), and the elements of $\hat{\mathbf{H}}$ are stored in vector $\boldsymbol{h}$, the problem can be described as

$$\begin{aligned} \boldsymbol{h}^* = \underset{\boldsymbol{h}}{arg\,min} \quad & \boldsymbol{h}^\top \mathbf{A} \boldsymbol{h}, \\ \text{where } & \boldsymbol{h}^\top \boldsymbol{h} = 1, \\ & \boldsymbol{h}^\top \mathbf{C}_k \boldsymbol{h} = 0, \ \forall k \in 1 \ldots 20. \end{aligned}$$

The matrix $\mathbf{A}$ is the quadratic form defined by $\mathcal{R}_{\text{ALG}}(\hat{\mathbf{H}})$, and each $\mathbf{C}_k$ is a constraint matrix arising from the relation given in equation 6.5.

Optimisation in the presence of multiple quadratic equality constraints is a difficult problem, since it is known to be non-convex with a disconnected solution space. A common strategy for dealing with such problems is to treat all quadratic terms of the unknowns as independent variables. Each of the quadratic constraints in the original problem becomes a linear constraint in the new parameterisation. Specifically, if we denote $\mathbf{X} = \boldsymbol{h}\boldsymbol{h}^\top$, the problem becomes

$$\begin{aligned} \boldsymbol{X}^* = \underset{\mathbf{X}}{arg\,min} \quad & Tr(\mathbf{A}\mathbf{X}), \\ \text{where } & Tr(\mathbf{X}) = 1, \\ & Tr(\mathbf{C}_k \mathbf{X}) = 0, \ \forall k \in 1 \ldots 20, \\ & \mathbf{X} \succeq \emptyset, \ \mathbf{X}^\top = \mathbf{X}, \ Rank(\mathbf{X}) = 1, \end{aligned} \tag{6.6}$$

where $Tr(.)$ denotes the trace of a matrix. Without the rank constraint, this amounts to a semi-definite programming problem, an overview of which is given by Vandenberghe and Boyd in [56]. Such problems are solvable in polynomial time using interior-point methods such as that of Alizadeh [1]. In practise however, the optimal solution found by this method will not satisfy the rank constraint. Although strategies to find a rank 1 solution have been investigated [34], finding the global minimum is not guaranteed. In particular, note that the cost associated with a rank 1 solution will typically be higher than the optimal cost of a solution with higher rank. Also note that the number of unknowns has increased quadratically.

Because of these difficulties, it is natural to consider both reducing the number of unknown parameters, and using only a minimally sized set of data. The former will reduce the number of higher-order terms to be treated as independent variables. The latter ensures that a rank 1 matrix $\mathbf{X}$ exists which minimises equation 6.6.

The number of unknown parameters in $\hat{\mathbf{H}}$ can be reduced by using stationary scene point correspondences to constrain the solution. It is therefore assumed that a set of scene features have been viewed by both cameras, and that their projections have been identified and tracked independently in each video sequence. As described in section 6.1, it is assumed that the locations of these scene features are known, but *uncorrelated*. Specifically, $\boldsymbol{z}_k$ is a 4×1 vector describing the location of stationary scene point k recovered using image information from the first video sequence. Similarly, $\boldsymbol{z}'_k$ describes the location of stationary scene point k, recovered using the second video sequence. We seek a ray transformation matrix $\hat{\mathsf{H}}$, such that the corresponding 4×4 projective transformation satisfies

$$\boldsymbol{z}_k \sim \mathsf{H}\boldsymbol{z}'_k \quad \forall k$$

Corresponding scene points are assumed to occupy a region of the scene that has little 3D variation. Estimating a 4×4 H from these matches alone is therefore ill-conditioned at best, and degenerate at worst. A degenerate example is when moving cameras observe a common area of a ground plane from different directions, as depicted in example 6.1.

It is assumed that three stationary scene point correspondences exist. Upon an initial inspection, estimating a ray transformation when three stationary matches are available may seem redundant. It was noted in sections 2.1.1 and 2.1.2 that methods have been developed to recover the intrinsic parameters of a camera from fundamental matrices and homography matrices. Similarly, the intrinsic parameters of a moving camera can be determined from a set of projection matrices, assuming certain constraints. As an example, a method to estimate intrinsic parameters common to every projection is given in [53]. If the intrinsic parameters associated with each frame of a video sequence are known, the projective distortion of a scene reconstruction can be removed. Two such scene reconstructions will differ only by rotation, translation, and as single scale. A 4×4 projective transformation H aligning these reconstructions can be described by seven parameters. Since each scene point correspondence $(\boldsymbol{z}_k, \boldsymbol{z}'_k)$ provides three constraints on the elements of H, it could be computed from these stationary matches alone. If a reliable transformation can be estimated, the two sequences can be spatially registered, and synchronisation can be robustly estimated with the algorithms given in chapter 5.

Depending on the available constraints on the intrinsic camera parameters, there exist camera motions which do not permit a Euclidean reconstruction to be realised. A number of these motion classes, denoted Critical Motion Sequences, are described in [44]. In such degenerate cases, estimating a ray transformation matrix with both stationary and moving point constraints allows the two reconstructions (and equivalent cameras) to be spatially registered. Constraints on the intrinsic parameters for *both* cameras can then be used to achieve a Euclidean reconstruction, which when combined may resolve the degeneracy. Additionally, even in non-degenerate cases where Euclidean reconstructions are achieved, moving point constraints may provide some benefit. The constraints on intrinsic parameters may yield reconstructions which are a poor approximation to Euclidean, even if they are aesthetically acceptable. Accordingly, the area of space occupied by moving objects may be poorly registered. In this case, using the synchronisation algorithms from previous chapters could fail due to uncharacteristically high epipolar errors.

The following section derives equations for estimating a 4×4 transformation to spatially register a pair of independently moving cameras. A projective transformation is sought using only the three stationary scene point correspondences, and moving point constraints.

6.3.1 Applying The Three Stationary Scene Point Constraints

Two scene point sets are available, each of size 3, with 3D point locations $\{\boldsymbol{z}_k\}$ and $\{\boldsymbol{z}'_k\}$ observed by the first and second cameras respectively. For each k, the scene points $\boldsymbol{z}_k$ and $\boldsymbol{z}'_k$ correspond with the same scene feature. We seek a 4×4 projective transformation H that will spatially align the two cameras, such that

$$\mathsf{H}\boldsymbol{z}'_k = \boldsymbol{z}_k \quad \forall k \in 1, 2, 3.$$

To conveniently parameterise any matrix 4×4 matrix H satisfying the stationary scene point correspondences, we introduce two 4×4 transformations, Q and Q', defined such that

$$\begin{aligned}
\mathsf{Q}\boldsymbol{z}_1 &\sim \mathsf{Q}'\boldsymbol{z}'_1 \sim \boldsymbol{e}_1, \\
\mathsf{Q}\boldsymbol{z}_2 &\sim \mathsf{Q}'\boldsymbol{z}'_2 \sim \boldsymbol{e}_2, \\
\mathsf{Q}\boldsymbol{z}_3 &\sim \mathsf{Q}'\boldsymbol{z}'_3 \sim \boldsymbol{e}_3.
\end{aligned} \tag{6.7}$$

The vectors $\boldsymbol{e}_k$ are standard unit vectors with 1 at element k, and 0 elsewhere. The 4×1 vector describing the plane defined by the three scene points $\{\boldsymbol{z}_k\}$ is denoted $\boldsymbol{\pi}_z$. Similarly, the plane defined by the three scene points $\{\boldsymbol{z}'_k\}$ is denoted $\boldsymbol{\pi}'_z$. A simple choice for the matrices Q and Q', such that the relations in 6.7 are satisfied, is

$$\begin{aligned}
\mathsf{Q}^{-1} &= [\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{\pi}_z], \\
\mathsf{Q}'^{-1} &= [\boldsymbol{z}'_1, \boldsymbol{z}'_2, \boldsymbol{z}'_3, \boldsymbol{\pi}'_z].
\end{aligned} \tag{6.8}$$

The first three columns of Q^{-1} and Q'^{-1} are chosen to ensure that the relations given by equation 6.7 are satisfied. The choice for the fourth columns ensure that Q^{-1} and Q'^{-1} are non-singular.

Recalling that we require a 4×4 matrix H satisfying

$$\mathsf{H}\boldsymbol{z}'_k = \boldsymbol{z}_k \quad \forall k \in 1, 2, 3,$$

it follows that

$$\begin{aligned}
&\mathsf{H}\boldsymbol{z}'_k = \boldsymbol{z}_k \\
&\mathsf{H}\mathsf{Q}'^{-1}\mathsf{Q}'\boldsymbol{z}'_k = \mathsf{Q}^{-1}\mathsf{Q}\boldsymbol{z}_k \\
&\mathsf{Q}\mathsf{H}\mathsf{Q}'^{-1}\mathsf{Q}'\boldsymbol{z}'_k = \mathsf{Q}\boldsymbol{z}_k \\
&\mathsf{G}(\mathsf{Q}'\boldsymbol{z}'_k) = \mathsf{Q}\boldsymbol{z}_k \\
&\forall k \in 1 \ldots 3, \\
&\text{where } \mathsf{G} = \mathsf{Q}\mathsf{H}\mathsf{Q}'^{-1}.
\end{aligned}$$

Conceptually, if H correctly aligns the reconstructions described by $\{\boldsymbol{z}_k\}$ and $\{\boldsymbol{z}'_k\}$, then G aligns the reconstructions given by $\{\mathsf{Q}\boldsymbol{z}_k\}$ and $\{\mathsf{Q}'\boldsymbol{z}'_k\}$. Furthermore, due to the choice of Q and Q', the transformation G is constrained such that

$$\mathsf{G}\boldsymbol{e}_1 \sim \boldsymbol{e}_1,$$
$$\mathsf{G}\boldsymbol{e}_2 \sim \boldsymbol{e}_2,$$
$$\mathsf{G}\boldsymbol{e}_3 \sim \boldsymbol{e}_3.$$

The matrix G is therefore known to have the form

$$\mathsf{G} = \begin{bmatrix} d_1 & 0 & 0 & t_1 \\ 0 & d_2 & 0 & t_2 \\ 0 & 0 & d_3 & t_3 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \tag{6.9}$$

For any full-rank G of this form, the corresponding H given by $\mathsf{H} = \mathsf{Q}^{-1}\mathsf{G}\mathsf{Q}'$ will precisely relate the three stationary scene point correspondences.

6.3.2 Applying The Moving Scene Point Constraints

A matrix G, of the form described in equation 6.9, is only defined up to scale. Matrices G and $u\mathsf{G}$ describe the same projective transformation, for any non-zero scalar u. Accordingly, 6 pairs of moving scene point correspondences in synchronous frames are needed to define G. For a matrix G given by equation 6.9, the corresponding 6×6 ray transformation matrix is denoted $\hat{\mathsf{G}}$, and has the form

$$\hat{\mathsf{G}} = \begin{bmatrix} g_1 & 0 & g_2 & 0 & g_3 & 0 \\ 0 & g_4 & g_5 & 0 & 0 & g_6 \\ 0 & 0 & g_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_8 & g_9 & g_{10} \\ 0 & 0 & 0 & 0 & g_{11} & 0 \\ 0 & 0 & 0 & 0 & 0 & g_{12} \end{bmatrix}$$

For the 4×4 projective transformations Q and Q' defined by equation 6.8, the corresponding ray transformations are denoted $\hat{\mathsf{Q}}$ and $\hat{\mathsf{Q}}'$. Given synchronous frame pair (i, j) and matching moving point trajectories (h, h'), a pair of image correspondences provide the constraint

$$\boldsymbol{p}_{h,i}^{\top}(\hat{\mathsf{M}}_i^{\top}\hat{\mathsf{Q}}^{\top})\mathsf{J}\hat{\mathsf{G}}(\hat{\mathsf{Q}}'\hat{\mathsf{M}}_j')\boldsymbol{p}'_{h',j} = 0.$$

Note that this constraint is linear in the 12 unknown elements of $\hat{\mathsf{G}}$. Six constraints of this form from matching image points in synchronous frames are required. These six constraints alone will not uniquely determine $\hat{\mathsf{G}}$. The remaining constraints are five quadratic equations arising from the relation

$$\hat{\mathsf{G}}^{\top}\mathsf{J}\hat{\mathsf{G}} \sim \mathsf{J}.$$

Additionally, three more quadratic constraints can be found by combining pairs of the existing five, or by considering the equivalent relation

$$\hat{\mathbf{G}}\mathbf{J}\hat{\mathbf{G}}^\top \sim \mathbf{J}.$$

Although the extra three quadratic constraints are redundant, none are a *linear* combination of the previous five, and hence will prove useful later. The quadratic constraints are

$$\begin{aligned} g_1g_{12} - g_7g_8 &= 0, & g_4g_{11} - g_7g_8 &= 0, \\ g_1g_6 + g_3g_4 &= 0, & g_1g_{10} + g_2g_8 &= 0, \\ g_4g_9 + g_5g_8 &= 0, & g_5g_{11} + g_7g_9 &= 0, \\ g_2g_{12} + g_7g_{10} &= 0, & g_3g_{12} + g_6g_{11} &= 0. \end{aligned} \tag{6.10}$$

If the unknown elements of $\hat{\mathbf{G}}$ are placed in vector $\boldsymbol{g}$, and denoting $\mathbf{0}$ as the zero vector, the problem equates to finding $\boldsymbol{g}$ such that

$$\mathbf{A}\boldsymbol{g} = \mathbf{0}, \quad \boldsymbol{g}^\top\mathbf{C}_\kappa\boldsymbol{g} = 0, \quad \forall\kappa \in 1\ldots8. \tag{6.11}$$

Each row of the 6×12 matrix $\mathbf{A}$ specifies a linear constraint on $\boldsymbol{g}$ derived from a moving point correspondence in a synchronous frame pair. Each 12×12 matrix $\mathbf{C}_\kappa$ is a quadratic form equivalent to one of the quadratic constraints given in equation 6.10.

The constraints on vector $\boldsymbol{g}$ described by matrix $\mathbf{A}$ can be applied using linear algebra methods. The linear space of solutions satisfying $\mathbf{A}\boldsymbol{g} = \mathbf{0}$ can be found using a singular value decomposition of the matrix $\mathbf{A}$. Assuming non-degeneracies, this restricts the solution to

$$\boldsymbol{g} = \mathbf{N}_A\boldsymbol{x}, \tag{6.12}$$

for a 12×6 matrix $\mathbf{N}_A$, the orthogonal columns of which span the right null-space of matrix $\mathbf{A}$. For *any* non-zero 6×1 vector $\boldsymbol{x}$, the corresponding vector $\boldsymbol{g}$ given by equation 6.12 will satisfy the linear relation $\mathbf{A}\boldsymbol{g} = \mathbf{0}$. This reduces the problem described by equation 6.11 to finding $\boldsymbol{x}$ such that

$$\begin{aligned} &\boldsymbol{x}^\top\hat{\mathbf{C}}_\kappa\boldsymbol{x} = 0, \quad \forall\kappa \in 1\ldots8. \\ &\text{where } \hat{\mathbf{C}}_\kappa = \mathbf{N}_A^\top\mathbf{C}_\kappa\mathbf{N}_A. \end{aligned} \tag{6.13}$$

As mentioned previously, solving simultaneous multivariate quadratics is a difficult problem. To find a solution, the *relinearisation* method of Kipnis and Shamir [29] is used. Relinearisation, proposed as a method for defeating quadratic based cryptosystems, applies constraints to multiple higher order terms to find a solution. The specific application of relinearisation to this problem is described below.

Initially, the matrix $\mathbf{Y} = \boldsymbol{x}\boldsymbol{x}^\top$ is considered. Treating the elements as independent variables yields a problem of finding the 6×6 matrix $\mathbf{Y}$ such that

$$\begin{aligned} &Tr(\mathbf{Y}\hat{\mathbf{C}}_\kappa) = 0, \quad \forall\kappa \in 1\ldots8, \\ &Rank(\mathbf{Y}) = 1, \quad \mathbf{Y}^\top = \mathbf{Y}. \end{aligned} \tag{6.14}$$

Note that a positive semi-definite constraint is unnecessary here. A symmetric matrix with rank 1 is either positive or negative semi-definite, and a negative semi-definite rank 1 matrix still provides a solution since

$$Tr(\mathbf{Y}\hat{\mathbf{C}}_\kappa) = 0 \quad \Leftrightarrow \quad Tr(-\mathbf{Y}\hat{\mathbf{C}}_\kappa) = 0.$$

Additionally, note that given a rank 1 matrix $\mathbf{Y}$ satisfying equation 6.14, any non-zero row or column of $\mathbf{Y}$ specifies a vector $\boldsymbol{x}$ satisfying equation 6.13. Since $\mathbf{Y}$ is symmetric, it can be defined by its upper or lower triangular elements only. These elements can be stored row-wise in a 21×1 vector $\boldsymbol{y}$. Such conversions between symmetric matrix and vector representations are used repeatedly during relinearisation.

To facilitate conversions between vector and matrix representations, we define index mapping functions $\mathcal{IR}$ and $\mathcal{IC}$. Given element j in the vector representation, the corresponding (row,column) indices in the upper triangular part of a symmetric matrix is given by $(\mathcal{IR}(j), \mathcal{IC}(j))$. Although the size of the symmetric matrix is technically required to perform such a mapping, this can be determined by context, and is therefore not included as an input to the functions. Similarly, the function $\mathcal{IV}$ maps a (row,column) index pair from the symmetric matrix to its corresponding index in the vector representation.

Given these index mapping functions, each linear constraint $Tr(\mathbf{Y}\hat{\mathbf{C}}_\kappa) = 0$ corresponds to a linear constraint on the vector $\boldsymbol{y}$, given by

$$\begin{aligned} &\boldsymbol{c}_\kappa^\top \boldsymbol{y} = 0, \quad \forall \kappa \in 1 \dots 8, \\ &\text{where } [\boldsymbol{c}_\kappa]_\xi = \begin{cases} [\hat{\mathbf{C}}_\kappa]_{\mathcal{IR}(\xi),\mathcal{IC}(\xi)} & : \ \mathcal{IR}(\xi) = \mathcal{IC}(\xi) \\ [\hat{\mathbf{C}}_\kappa]_{\mathcal{IR}(\xi),\mathcal{IC}(\xi)} + [\hat{\mathbf{C}}_\kappa]_{\mathcal{IC}(\xi),\mathcal{IR}(\xi)} & : \ \mathcal{IR}(\xi) < \mathcal{IC}(\xi) \end{cases} \end{aligned}$$

From the eight linear constraints on $\boldsymbol{y}$, we can produce a linear system

$$\mathbf{B}\boldsymbol{y} = \mathbf{0},$$

for an 8×21 matrix $\mathbf{B}$, with row κ given by $\boldsymbol{c}_\kappa^\top$. Note these linear constraints will not produce a unique solution up to scale, since we require $\mathbf{Y}$ to have a rank of 1. The rank constraint on $\mathbf{Y}$ can be expressed as a set of quadratic constraints. For a symmetric rank 1 matrix, the determinant of every 2×2 sub-matrix obtained by selecting a pair of rows and pair of columns equates to 0. For every $(1 \leq s \leq t \leq u \leq v \leq 6)$,

$$\begin{aligned} [\mathbf{Y}]_{s,t}[\mathbf{Y}]_{u,v} &= [\mathbf{Y}]_{s,v}[\mathbf{Y}]_{t,u}, \\ [\mathbf{Y}]_{s,t}[\mathbf{Y}]_{u,v} &= [\mathbf{Y}]_{s,u}[\mathbf{Y}]_{t,v}. \end{aligned}$$

Considering all such quadruples of indices (s, t, u, v) produces 252 quadratic constraints, each represented by a quadratic form $\mathbf{D}_\kappa$, although some are redundant due to repeated indices. Indeed, some of these quadratic forms will be the zero matrix. We therefore seek a vector $\boldsymbol{y}$ such that

$$\mathbf{B}\boldsymbol{y} = \mathbf{0}, \quad \boldsymbol{y}^\top \mathbf{D}_\kappa \boldsymbol{y} = 0, \quad \forall \kappa \in 1 \dots 252. \tag{6.15}$$

Note the similarity to the original problem of finding $\boldsymbol{g}$ satisfying equation 6.11. Each element of a vector $\boldsymbol{y}$ satisfying the relations in equation 6.15 will equate to a quadratic function of the elements of vector $\boldsymbol{g}$. Proceeding in the same fashion as with the original problem of identifying $\boldsymbol{g}$, the right null-space of matrix $\mathbf{B}$ is computed using singular value decomposition. The orthogonal vectors spanning this null-space define the columns of matrix $\mathbf{N}_B$. This provides the relation

$$\boldsymbol{y} = \mathbf{N}_B \boldsymbol{z},$$

Given the known 21×13 matrix $\mathbf{N}_B$, this equation yields a vector $\boldsymbol{y}$ satisfying $\mathbf{B}\boldsymbol{y} = \mathbf{0}$, for *any* non-zero 13×1 vector $\boldsymbol{z}$. We therefore seek $\boldsymbol{z}$ such that

$$\begin{aligned} &\boldsymbol{z}^\top \hat{\mathbf{D}}_\kappa \boldsymbol{z} = 0, \quad \forall \kappa \in 1 \ldots 252. \\ &\text{where } \hat{\mathbf{D}}_\kappa = \mathbf{N}_B^\top \mathbf{D}_\kappa \mathbf{N}_B. \end{aligned}$$

Note that the problem of finding $\boldsymbol{z}$ satisfying these quadratics has a similar form to the earlier problem of finding $\boldsymbol{x}$ satisfying equation 6.13. The difference is that both the size of the unknown vector, and the size of the set of constraints, are larger.

The final step of relinearisation is to consider all quadratic terms of $\boldsymbol{z}$ as independent variables. Defining the matrix $\mathbf{V} = \boldsymbol{z}\boldsymbol{z}^\top$, the problem becomes one of finding $\mathbf{V}$ such that

$$\begin{aligned} &Tr(\mathbf{V}\hat{\mathbf{D}}_\kappa) = 0, \quad \forall \kappa \in 1 \ldots 252, \\ &\text{where } Rank(\mathbf{V}) = 1, \quad \mathbf{V}^\top = \mathbf{V}. \end{aligned}$$

Once again, a vector representation is considered, where the 91×1 vector $\boldsymbol{v}$ contains the upper triangular elements of the 13×13 matrix $\mathbf{V}$. Each of the constraints on $\mathbf{V}$ has a corresponding vector constraint given by

$$\begin{aligned} &\boldsymbol{d}_\kappa^\top \boldsymbol{v} = 0, \forall \kappa \in 1 \ldots 252, \\ &\text{where } [\boldsymbol{d}_\kappa]_\xi = \begin{cases} [\hat{\mathbf{D}}_\kappa]_{\mathcal{IR}(\xi),\mathcal{IC}(\xi)} & : \ \mathcal{IR}(\xi) = \mathcal{IC}(\xi) \\ [\hat{\mathbf{D}}_\kappa]_{\mathcal{IR}(\xi),\mathcal{IC}(\xi)} + [\hat{\mathbf{D}}_\kappa]_{\mathcal{IC}(\xi),\mathcal{IR}(\xi)} & : \ \mathcal{IR}(\xi) < \mathcal{IC}(\xi) \end{cases} \end{aligned}$$

The key to success in relinearisation is the increase in the number of constraints relative to the size of the unknown vector. Under favourable circumstances, the set of linear constraints on $\boldsymbol{v}$ is large enough to yield a solution without the consideration of the rank constraint. Matrix $\mathbf{\Theta}$ is defined as the 252×91 matrix with row κ given by $\boldsymbol{d}_\kappa^\top$. We therefore seek a vector $\boldsymbol{v}$ such that

$$\mathbf{\Theta}\boldsymbol{v} = \mathbf{0}. \tag{6.16}$$

Although the number of constraints is greater than the size of $\boldsymbol{v}$, the solution is under-determined. Matrix $\mathbf{\Theta}$ has been determined by random test cases to have a rank of 78. In [29], it is proposed to handle such a situation by also considering the quadratic constraints on $\mathbf{V}$ derived from the fact that $\mathbf{V}$ is a rank 1 symmetric matrix. This requires the formulation of even larger linear systems, and, for this particular problem, yields no benefit. This may be related to the fact that the quadratic constraints in the original problem given by equation 6.11 equate to 0, and that the solution $\boldsymbol{g}$ is only defined up to scale. In fact, it should be noted that the above description of relinearisation is specific to this case, and that extra considerations are required for a problem with non-zero constant terms.

Although the process of relinearisation, as described in [29], has failed to solve the original problem given in equation 6.11, the development of a new step in the process provides a means for computing vector $\boldsymbol{v}$. The means for determining vector $\boldsymbol{v}$ is now described, specific to this problem. The right null-space of matrix $\mathbf{\Theta}$ is spanned by the orthogonal columns of the 91×13 matrix $\mathbf{N}$. The matrix $\mathbf{N}$ can be determined by a singular value decomposition of $\mathbf{\Theta}$. A vector $\boldsymbol{v}$ satisfies the linear system given in equation 6.16, for $\boldsymbol{v}$ given by

$$\boldsymbol{v} = \mathbf{N}\boldsymbol{w},$$

for *any* non-zero 13×1 vector $\boldsymbol{w}$. Note however, that the corresponding matrix $\mathbf{V}$ is not necessarily rank 1. The inclusion of the rank constraint on $\mathbf{V}$ is made possible by the fact that vector $\boldsymbol{w}$ is the same size as any row or column of matrix $\mathbf{V}$. A single column of the unknown matrix $\mathbf{V}$ can be expressed in terms of the vector $\boldsymbol{w}$. For column u, the relation is

$$\begin{aligned} &[\mathbf{V}]_{*,u} = \mathbf{N}_u \boldsymbol{w}, \\ &\text{where } [\mathbf{N}_u]_{\xi,*} = [\mathbf{N}]_{\mathcal{IV}(\xi,u),*}, \end{aligned} \tag{6.17}$$

where operators $[\mathbf{M}]_{*,u}$ and $[\mathbf{M}]_{v,*}$ denote column u and row v of a matrix $\mathbf{M}$. Equation 6.17 specifies that each column of $\mathbf{V}$ can be found by pre-multiplying $\boldsymbol{w}$ by a 13×13 matrix comprised of a subset of the rows of $\mathbf{N}$. Furthermore, since we require $\mathbf{V}$ to be rank 1, any two of these columns must be equal up to scale. Choosing a pair of such columns, say (u, v),

$$\mathbf{N}_u \boldsymbol{w} \sim \mathbf{N}_v \boldsymbol{w}.$$

In general, these matrices will be non-singular, so

$$\mathbf{N}_v^{-1} \mathbf{N}_u \boldsymbol{w} \sim \boldsymbol{w}.$$

This tells us that the solution for $\boldsymbol{w}$ that defines a corresponding rank-1 matrix $\mathbf{V}$ is an eigenvector of $\mathbf{N}_v^{-1}\mathbf{N}_u$, associated with a real eigenvalue. If this matrix has separate real eigenvalues, the space of possible solutions (up to scale) is reduced to a finite set of odd numbered size between 1 and 13. The set of solutions is known to be of odd size since complex eigenvalues are only present as conjugate pairs for a matrix with real-valued elements. Random testing has demonstrated that, in general, the set of eigenvectors associated with real eigenvalues is the same up to scale, regardless of the choice of (u, v), and that each such solution yields a rank-1 matrix $\mathbf{V}$. It should be noted however, that this may not occur for cases where the original problem of finding $\boldsymbol{g}$ is degenerate.

The presence of multiple solutions is to be expected since we are seeking the intersection of a linear space (the moving point constraints) with curved surfaces (the quadratic constraints). A similar multiplicity can occur when computing a fundamental matrix from a minimum of seven image point correspondences.

Each $\boldsymbol{w}$ solution can be used to find the corresponding solution $\boldsymbol{g}$ to the original problem. Given $\boldsymbol{w}$, $\boldsymbol{v}$ and hence V are known. The vector $\boldsymbol{z}$ is then given (up to scale) by any row or column of V. Similarly, $\boldsymbol{z}$ defines $\boldsymbol{y}$ and Y, from which any row or column defines $\boldsymbol{x}$. Finally, $\boldsymbol{g}$ can be recovered from $\boldsymbol{x}$ by the linear relation given in equation 6.12.

To summarise, candidate solutions for the unknown elements of $\hat{\mathsf{G}}$ can be computed using 6 moving point correspondences, and by considering linear constraints on a reduced set of quadratic and quartic functions of the elements of $\hat{\mathsf{G}}$, followed by an eigenvalue decomposition of a 13×13 matrix.

The increase in problem size during relinearisation should clarify why stationary point matches are used to constrain the solution. If we consider the case where no stationary point matches are used, 15 moving point correspondences are required to estimate a projective transformation. An equivalent relinearisation process to the one described above yields vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of sizes 21×1 and 231×1 respectively. Even assuming that the constraints from relations

$$\hat{\mathsf{H}}^\top \mathsf{J} \hat{\mathsf{H}} \sim \mathsf{J}, \quad \hat{\mathsf{H}} \mathsf{J} \hat{\mathsf{H}}^\top \sim \mathsf{J}$$

form independent linear constraints on $\boldsymbol{y}$, this still yields a vector $\boldsymbol{z}$ of size 191×1. The resulting size of vector $\boldsymbol{v}$ is 18336×1. Computing the null-space of constraints on such a large vector may be acceptable if it is only required once. However, even if relinearisation can be used to determine a solution, the computational cost will simply be too high when such estimations are an atomic component of an iterative robust algorithm.

6.3.3 Recovering The Projective Transformation G

Section 6.3.2 describes a means to compute the 6×6 ray transformation $\hat{\mathsf{G}}$ using a set of stationary and moving scene point correspondences associated with a pair of cameras. Given $\hat{\mathsf{G}}$, the corresponding 4×4 projective transformation G can be determined in closed form. Recalling that the form of G is

$$\mathsf{G} = \begin{bmatrix} d_1 & 0 & 0 & t_1 \\ 0 & d_2 & 0 & t_2 \\ 0 & 0 & d_3 & t_3 \\ 0 & 0 & 0 & d_4 \end{bmatrix},$$

and that every element of a ray transformation matrix can be expressed as a particular quadratic equation of the elements in a 4×4 transformation, it follows that

$$(d_1d_2, d_1d_3, d_1d_4, d_2d_3, d_2d_4, d_3d_4) = \pm(g_1, g_4, g_7, g_8, g_{11}, g_{12}).$$

Note the ambiguity of sign is necessary since $\boldsymbol{g}$ is only defined up to scale, and the left hand side consists of quadratic terms. Considering triplets of these vector elements leads to multiple formulas for each d_i^2, given by

$$\begin{aligned}
&d_1^2(g_8, g_{11}, g_{12}) = s(g_1g_4, g_1g_7, g_4g_7),\\
&d_2^2(g_4, g_7, g_{12}) = s(g_1g_8, g_1g_{11}, g_8g_{11}),\\
&d_3^2(g_1, g_7, g_{11}) = s(g_4g_8, g_4g_{12}, g_8g_{12}),\\
&d_4^2(g_1, g_4, g_8) = s(g_7g_{11}, g_7g_{12}, g_{11}g_{12}),\\
&\text{where } s = \pm 1.
\end{aligned}$$

The common ambiguity of sign, represented by s, can be resolved by noting that we require each d_i^2 to be non-negative. Furthermore, we need not be concerned about 0 elements since a full rank G strictly requires that all of the above terms are non-zero. Now given any d_i^2 term, we can solve for the diagonal of G by choosing any of

$$\begin{aligned}
(d_1, d_2, d_3, d_4) &= \tfrac{\pm 1}{\sqrt{d_1^2}}(sd_1^2, g_1, g_4, g_7),\\
(d_1, d_2, d_3, d_4) &= \tfrac{\pm 1}{\sqrt{d_2^2}}(g_1, sd_2^2, g_8, g_{11}),\\
(d_1, d_2, d_3, d_4) &= \tfrac{\pm 1}{\sqrt{d_3^2}}(g_4, g_8, sd_3^2, g_{12}),\\
(d_1, d_2, d_3, d_4) &= \tfrac{\pm 1}{\sqrt{d_4^2}}(g_7, g_{11}, g_{12}, sd_4^2).
\end{aligned}$$

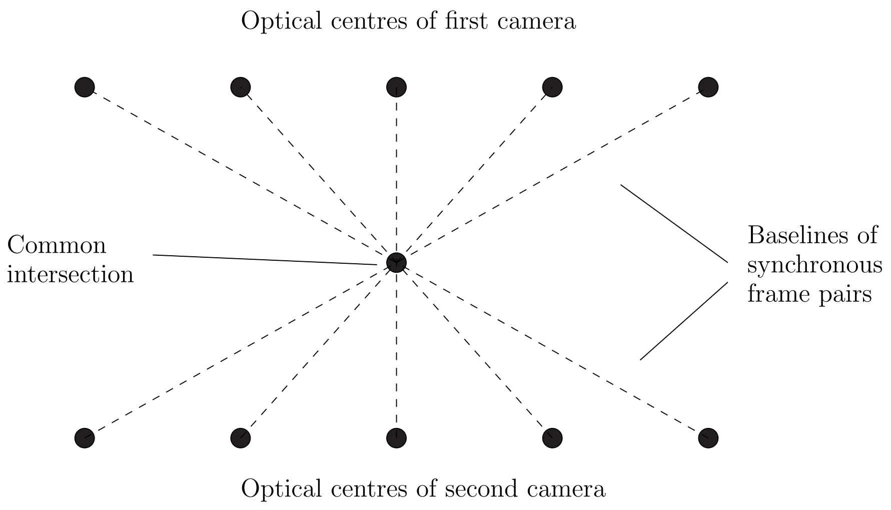
The choice of sign in any of the above equations is arbitrary, since G and $-\mathsf{G}$ will generate the *same* corresponding 6×6 matrix $\hat{\mathsf{G}}$. Given (d_1, d_2, d_3, d_4) and s, the remaining elements can be computed using the relations

$$\begin{aligned}
t_1(d_2, -d_3) &= s(g_3, g_6),\\
t_2(d_1, -d_3) &= s(g_2, g_{10}),\\
t_3(d_1, -d_2) &= s(g_5, g_9).
\end{aligned}$$

6.3.4 A Degenerate Camera Pair Motion

Not all pairs of independent camera motions are appropriate for using moving point correspondences to achieve a spatial registration. In particular, consider the case where the baselines connecting the optical centres of synchronous frame pairs have a common point of intersection. Simple cases where this occurs can be two cameras with parallel linear motions on opposite sides of a scene, as illustrated in figure 6.1.

Such camera motions do not permit a unique solution for registering the reconstructions. Consider the corresponding scene point triplets $\{\mathsf{Q}\boldsymbol{z}_k\}$, $\{\mathsf{Q}'\boldsymbol{z}'_k\}$ associated with the


Optical centres of first camera
Common intersection
Baselines of synchronous frame pairs
Optical centres of second camera

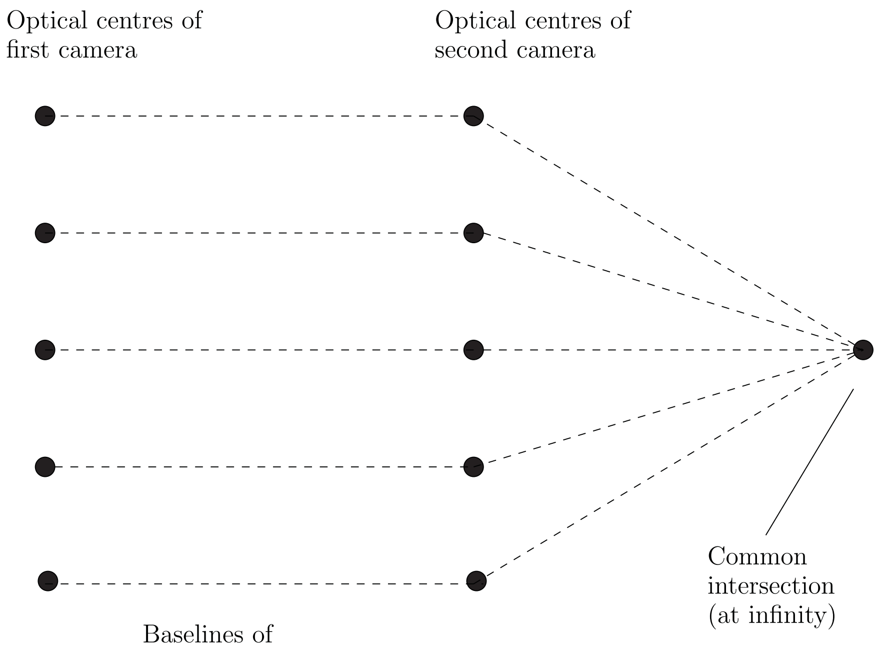

Optical centres of first camera
Optical centres of second camera
Common intersection (at infinity)
Baselines of synchronous frame pairs

Figure 6.1: Two top-down views of degenerate camera pair motions for spatial registration using moving point correspondences

first and second camera respectively, with Q and Q' as defined in section 6.3.1. By the definition of Q and Q', these triplets have corresponding locations on the plane at infinity. A matrix G is sought such that projection matrices sets $\{\mathsf{M}_i\mathsf{Q}^{-1}\}$ and $\{\mathsf{M}'_j\mathsf{Q}'^{-1}\mathsf{G}^{-1}\}$ are correctly aligned. The common point of baseline intersection for these aligned cameras is denoted $\boldsymbol{q}$. If G satisfies the set of moving point correspondence constraints, then so does DG, where

$$\mathsf{D} = \begin{bmatrix} 1 & 0 & 0 & kq_1 \\ 0 & 1 & 0 & kq_2 \\ 0 & 0 & 1 & kq_3 \\ 0 & 0 & 0 & 1+kq_4 \end{bmatrix},$$

$$\text{where } \boldsymbol{q} = [q_1, q_2, q_3, q_4]^\top, \qquad \text{for any } kq_4 \neq -1.$$

The degeneracy arises from the fact that

$$\mathsf{D}\boldsymbol{x} = \boldsymbol{x} + [\boldsymbol{x}]_4 k\boldsymbol{q}.$$

Accordingly, transforming a 3D point vector by pre-multiplying by D will simply shift it towards or away from point $\boldsymbol{q}$. Since $\boldsymbol{q}$ denotes the intersection of baselines for all synchronous frame pairs, it also lies on all epipolar planes associated with moving point correspondences. The transformation D will therefore preserve the set of epipolar planes, such that the constraints derived from moving point correspondences are still satisfied. Furthermore, D preserves the locations of all points at the plane at infinity, so the stationary point correspondences also remain satisfied.

6.4 Robustly Estimating a Projective Transformation

Temporal and spatial registration requires a different approach here to that used in previous chapters. Since the synchronisation parameters are no longer the only unknown variables, the histogram methods described in chapter 4 are no longer appropriate. Instead, a search across synchronisation parameter space is required, as used in most existing synchronisation algorithms (see section 2.2 for details). It is assumed for the remainder of this chapter that the pair of video sequences have known frame rates. Spatial and temporal registration therefore requires the estimation of the 4×4 transformation G, and a frame offset a. An extension to the unknown frame rate case is trivial, but incurs an order of magnitude increase in time. It is also assumed that correct stationary scene point correspondences are provided as input, allowing the estimation of Q and Q' as described in section 6.3.1. Such correspondences could readily be determined in the case of a ground plane using robust homography estimation. Should only a tentative set of stationary scene point correspondences be available, the robust estimation could easily be extended to also classify these correspondences as inliers and outliers. As with chapter 5, all possible pairs of trajectories (h, h') associated with moving scene features are considered as candidate matches. The set of such pairs is denoted $\boldsymbol{M}$. The description of the robust estimation is restricted to cases where the frame rate ratio $b \geq 1$. The case of $b < 1$ is redundant, since an equivalent problem with $b \geq 1$ can be defined simply by alternating which of the video sequences is considered the first, and which is considered the second, as shown in section 1.1.

6.4.1 A Robust Algorithm Template

The proposed algorithm consists of searching uniformly across a discrete set of frame offset values. For each frame offset value a, a robust estimation of the 4×4 projective transformation G is performed. This is achieved by repeatedly choosing a small number c of trajectory pairs (h, h') at random from the set $\boldsymbol{M}$. The selected random subset of trajectory pairs is denoted $\boldsymbol{M}_C$. Moving point correspondences for trajectory pairs (h, h') in synchronous frames are selected to compute G, by use of the relinearisation method described in section 6.3.2. Note the relinearisation process can yield multiple solutions for G on each iteration. Each such solution must be assessed independently.

In chapter 5, only one trajectory correspondence was used to estimate the synchronisation parameters on each iteration. Estimating both the frame offset a, and the projective transformation G, however, may require choosing $c > 1$. This then warrants additional consideration, since not all random subsets $\boldsymbol{M}_C$ are appropriate. A particular value of the frame offset a may not admit any synchronous frame pairs for a given trajectory correspondence, as illustrated in figure 6.2.

Since all pairs of trajectories (h, h') are included in the set of matches $\boldsymbol{M}$, a randomly chosen subset $\boldsymbol{M}_C$ may contain trajectory correspondences which are mutually exclusive. As described in chapter 5, two trajectories in the same video sequence, both visible for at least one frame in common, must correspond to distinct moving scene features. These two trajectories should therefore not both correspond to the same trajectory from the other video sequence. If a randomly chosen subset $\boldsymbol{M}_C$ contains such mutually exclusive matches, it can be assumed to contain at least one outlier, and is therefore inappropriate for estimating G. Such a subset $\boldsymbol{M}_C$ should then be discarded.

The formula in [40] to compute a number of iterations for robust estimation is at best an approximation, and only suitable for a large set of candidate matches. Since matches in this chapter denote pairs of entire trajectories, the set may be too small for such an approximation to be appropriate. This was not an issue in chapter 5 as only a single trajectory correspondence (h, h') was used to estimate synchronisation on each iteration.

As described in [10], the probability of randomly selecting c distinct correct matches from a set of size n_x, of which number μ are inliers is given by

$$\frac{(n_x - c)!\mu!}{n_x!(\mu - c)!}.$$

Given an estimate of the frame offset a, a trajectory correspondence (h, h') should only be included in the data subset $\boldsymbol{M}_C$ if the frame offset a indicates that these trajectories

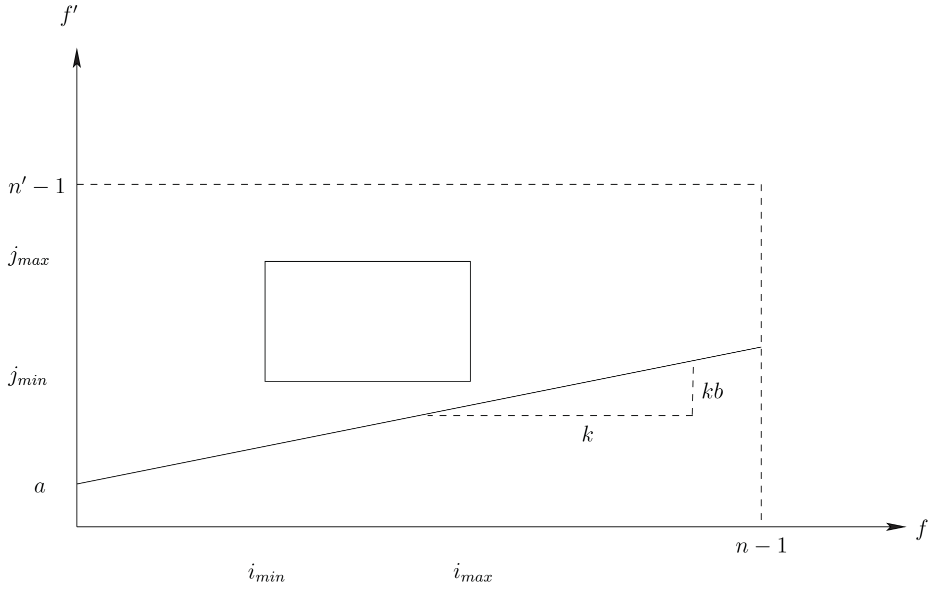


Figure 6.2: A trajectory pair visible in frames $i_{min} \ldots i_{max}$ in video 1, and $j_{min} \ldots j_{max}$ in video 2, which can not be a match according to frame offset a

are visible for ranges of frames which overlap in time. Trajectory correspondences like that shown in figure 6.2 should not be selected. Trajectory correspondences are selected randomly from the set which have overlapping visibilities according to offset estimate a. This set is denoted $\boldsymbol{M}_V$, and has size denoted $\mathcal{N}_v(a)$. Assuming all inliers are contained in the set $\boldsymbol{M}_V$, the probability of randomly selecting c inliers is

$$\mathcal{G}_v(a,\mu) = \frac{(\mathcal{N}_v(a)-c)!\mu!}{\mathcal{N}_v(a)!(\mu-c)!}.$$

Similarly to chapter 5, the termination condition for the robust algorithm is based on a chosen probability of failure p_f. For each frame offset a, we require at least one iteration to choose a subset $\boldsymbol{M}_C$ consisting only of inliers with probability $(1-p_f)$. The number of iterations required to achieve this probability, for a particular frame offset a and assuming μ inliers is

$$\mathcal{I}(a,\mu) = \left\{ \begin{array}{rcl} 0 & : & \mu > \mathcal{N}_v(a) \\ max(\lceil ln(p_f)\, ln(1-\mathcal{G}_v(a,\mu))^{-1}\rceil, 1) & : & \text{otherwise} \end{array} \right. .$$

The possibility of 0 iterations arises since, if it assumed μ trajectory correspondences are inliers, a value for a such that $SIZE(\boldsymbol{M}_V) < \mu$ is known to be incorrect. The $max(.)$ operator is necessary since if $\mathcal{N}_v(a) = \mu$, the ratio of logarithms equates to 0.

On each iteration, an estimate of $\mathbf{G}$ and a is used to classify every trajectory correspondence as either an inlier or an outlier. If the number of classified inliers is greater than that on all previous iterations, the assumed number of inliers μ is revised. This provides an adaptive termination condition, similar to that described in chapter 5. This requires an extra consideration as to how to iterate over frame offset values. The set of frame offset values are constrained to lie between a_{min} and a_{max}, such that every value in this range satisfies the minimum frame range size constraints r_{min} and r'_{min}, as defined in section 3.3. A breadth-first strategy is proposed. The algorithm iterates over candidate values of a, performing *one* iteration of the robust estimation for each frame offset a that has not yet been tested $\mathcal{I}(a, \mu)$ times. This process is then repeated until the required number of iterations $\mathcal{I}(a, \mu)$ is achieved for each frame offset value. This breadth-first approach is attractive since it should reduce the required execution time. If each frame offset value was considered strictly in turn, a higher number of iterations would be required for those values distant from truth. Such values would typically not classify a high number of inliers, and the assumed number of inliers μ would therefore remain low until a frame offset close to truth was considered.

6.4.2 Estimation and Evaluation with the Data Subset $\boldsymbol{M}_C$

The data subset $\boldsymbol{M}_C$ contains a total of c trajectory correspondences. These were chosen based on a hypothesised estimate of the frame offset a. This section describes the process by which $\mathbf{G}$ is estimated, and how the resulting parameters a and $\mathbf{G}$ are both refined by a Levenberg-Marquardt minimisation.

From the data subset $\boldsymbol{M}_C$, a total of six image point correspondences from synchronous frames are needed. The proposed method selects pairs of synchronous frames uniformly, but with a random offset in time. Specifically, for each trajectory pair $(h, h') \in \boldsymbol{M}_C$, a defines a range of frames in the first video (i_{min}, i_{max}), for which image points $p_{h,i}$ and $p'_{h',\lceil a+bi \rfloor}$ are both visible. The operator $\lceil k \rfloor$ denotes the 'closest' integer to real value k. If q correspondences are required from a trajectory pair (h, h'), then frames from the first video are chosen to be $q^{-1}(i_{max} - i_{min} + 1)$ frames apart. This uniform sampling is randomly offset between the first being equal to i_{min}, and the last being equal to i_{max}. For each such frame i, the image point pair $p_{h,i}$ and $p'_{h',\lceil a+bi \rfloor}$ denote a correspondence to be used as a moving point constraint.

Assuming smooth camera motions, the uniform sampling of frame indices ensures the moving point constraints from any trajectory pair (h, h') are well distributed in time. This increases the probability that these constraints are independent, typically yielding a better estimate of $\mathbf{G}$. The random offset in the frame sampling is used to avoid any two iterations of the robust estimation with the same set $\boldsymbol{M}_C$ being deterministic. If a selected set of moving point constraints is degenerate for the purposes of estimating $\mathbf{G}$, a subsequent iteration choosing moving point constraints from the same subset $\boldsymbol{M}_C$ is then less likely to be degenerate. Each moving point constraint is chosen by selecting a frame from the first video sequence, and computing the closest to synchronous integer frame index from the second video sequence. This is preferable to the reciprocal case of selecting an integer frame index from the second video sequence, and rounding a real-valued frame index from the first video sequence to the nearest integer. The preference is due to the fact that $b \geq 1$ specifies the second camera has a higher frame rate. Rounding to the nearest integer frame index typically corresponds to a smaller shift in time for the second camera.

The selected moving point correspondences are used to estimate the 4×4 transformation $\mathbf{G}$ to achieve a hypothesised spatial alignment. For convenience of notation, the corresponding 6×6 ray transformation is given by the function $\mathcal{R}_T(\mathbf{G})$. The fundamental matrix relating a frame from the first video sequence with another from the second video sequence can be expressed in terms of the ray transformation $\mathbf{G}$, and the ray projection matrix for each of the two frames. Specifically, for frames i and j from the first and second video sequences respectively, the associated fundamental matrix is

$$\mathcal{F}(\mathbf{G}, i, j) = (\hat{\mathbf{M}}_i^\top \hat{\mathbf{Q}}^\top) \mathbf{J} \mathcal{R}_T(\mathbf{G})(\hat{\mathbf{Q}}' \hat{\mathbf{M}}_j'),$$

where $\hat{\mathbf{Q}}$ and $\hat{\mathbf{Q}}'$ are the ray transformation matrices corresponding to $\mathbf{Q}$ and $\mathbf{Q}'$ respectively. Note that since the fundamental matrices relating frame pairs are defined, the interpolated epipolar error function $\mathcal{E}_{\text{ORTH-LI}}$, developed in chapter 3, can be used to assess the quality of $\mathbf{G}$ and a. Using the notation from section 3.1, $\mathcal{E}_{\text{ORTH-LI}}{}^\star$ denotes the equivalent function, but equates an unmeasurable interpolated epipolar error to 0.

On each iteration of the robust estimation, the relinearisation process produces a number of candidate estimates of $\mathbf{G}$. Each estimate of $\mathbf{G}$ is refined, along with the frame offset, by using Levenberg-Marquardt to minimise a cost function measuring the average of squared $\mathcal{E}_{\text{ORTH-LI}}{}^\star$ distances for all trajectory pairs (h, h') in the randomly chosen subset $\boldsymbol{M}_C$. As mentioned in section 5.3.3, epipolar errors in consecutive frames are likely to be of the same order of magnitude for a given trajectory correspondence (h, h'). Only a subset of such error measures are typically needed to refine a synchronisation estimate.

The frame offset a defines a range of integer frame indices in each video sequence for which trajectory pair (h, h') is visible, and epipolar errors $\mathcal{E}_{\text{ORTH-LI}}$ are measurable. The sizes of these frame ranges are denoted w_h and $w'_{h'}$ for the first and second video sequences respectively. For each match (h, h') in $\boldsymbol{M}_C$, sets of frames $V_{h,h'}$ and $V'_{h,h'}$ are constructed, containing frame indices uniformly sampled from these ranges for the first and second video sequence respectively. As with section 5.3.3, the sizes of these sets is chosen as

$$\begin{aligned} SIZE(V_{h,h'}) &= max(min(5, w_{h,h'}), \tfrac{1}{20} w_{h,h'}), \\ SIZE(V'_{h,h'}) &= max(min(5, w'_{h,h'}), \tfrac{1}{20} w'_{h,h'}). \end{aligned}$$

Unless there is very little overlapping visibility, these frame sets are large enough to permit a refinement of both $\mathbf{G}$ and a. For a given frame offset a, the number of measurable $\mathcal{E}_{\text{ORTH-LI}}$ measures for frames in sets $V_{h,h'}$ and $V'_{h,h'}$ is denoted $\overline{\mathcal{W}}_{\text{G}}(a, h, h')$. The average squared $\mathcal{E}_{\text{ORTH-LI}}$ measure for trajectory correspondence (h, h'), using the coarse sampling of frames in $V_{h,h'}$ and $V'_{h,h'}$, is given by

$$\begin{aligned}\overline{\mathcal{S}}_G(a,h,h',\mathbf{G}) = \tfrac{1}{\overline{\mathcal{W}}_{\mathrm{G}}(a,h,h')} \sum_{i\in V_{h,h'}} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}_{h,i},\ \boldsymbol{p}'_{h',\lfloor a+bi\rfloor}, \mathcal{F}(\mathbf{G}, i, \lfloor a+bi\rfloor), \\ & \qquad \boldsymbol{p}'_{h',\lfloor a+bi\rfloor+1}, \mathcal{F}(\mathbf{G}, i, \lfloor a+bi\rfloor+1), \\ & \qquad a+bi-\lfloor a+bi\rfloor)^2 + \\ \tfrac{1}{\overline{\mathcal{W}}_{\mathrm{G}}(a,h,h')} \sum_{j\in V'_{h,h'}} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}'_{h',j}, \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor}, \mathcal{F}(\mathbf{G}, \lfloor a'+b'j\rfloor, j)^{\top}, \\ & \qquad \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor+1}, \mathcal{F}(\mathbf{G}, \lfloor a'+b'j\rfloor+1, j)^{\top}, \\ & \qquad a'+b'j-\lfloor a'+b'j\rfloor)^2, \\ \text{where } a' = -ab^{-1}, \quad b' = b^{-1}. & \end{aligned} \tag{6.18}$$

The sum of squared $\mathcal{E}_{\mathrm{ORTH-LI}}$ measures for a particular trajectory correspondence (h,h'), frame offset a, and frame sets $V_{h,h'}$ and $V'_{h,h'}$ is given by

$$\overline{\mathcal{S}}_{\mathrm{G-SUM}}(a,h,h',\mathbf{G}) = \overline{\mathcal{S}}_G(a,h,h',\mathbf{G})\overline{\mathcal{W}}_{\mathrm{G}}(a,h,h').$$

An initial estimate of projective transformation $\mathbf{G}$ and frame offset a is refined by minimising the average squared $\mathcal{E}_{\mathrm{ORTH-LI}}$ measure for *all* trajectories in the randomly chosen set $\boldsymbol{M}_C$, using the coarse sampling of frames contained in sets $V_{h,h'}$ and $V'_{h,h'}$. The function to be minimised is

$$\overline{\mathcal{S}}_{\mathrm{G-TOTAL}}(a,\mathbf{G}) = \frac{\sum_{(h,h')\in\boldsymbol{M}_C} \overline{\mathcal{S}}_{\mathrm{G-SUM}}(a,\mathbf{G},h,h')}{\sum_{(h,h')\in\boldsymbol{M}_C} \overline{\mathcal{W}}_{\mathrm{G}}(a,h,h')} \tag{6.19}$$

After the minimisation, the refined estimate of $(a,\mathbf{G})$ is first assessed by computing an average squared $\mathcal{E}_{\mathrm{ORTH-LI}}$ measure across all frames, for each trajectory correspondence $(h,h')\in\boldsymbol{M}_C$. For frame offset a, the number of measurable $\mathcal{E}_{\mathrm{ORTH-LI}}$ distances for trajectory pair (h,h') is denoted $\mathcal{W}_{\mathrm{G}}(a,h,h')$. The average squared $\mathcal{E}_{\mathrm{ORTH-LI}}$ measure for (h,h') is given by

$$\begin{aligned}\mathcal{S}_{\mathrm{G}}(a,\mathbf{G},h,h') = \tfrac{1}{\mathcal{W}_{\mathrm{G}}(a,h,h')} \sum_{i=0}^{n-1} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}_{h,i},\ \boldsymbol{p}'_{h',\lfloor a+bi\rfloor}, \mathcal{F}(\mathbf{G}, i, \lfloor a+bi\rfloor), \\ & \qquad \boldsymbol{p}'_{h',\lfloor a+bi\rfloor+1}, \mathcal{F}(\mathbf{G}, i, \lfloor a+bi\rfloor+1), \\ & \qquad a+bi-\lfloor a+bi\rfloor)^2 + \\ \tfrac{1}{\mathcal{W}_{\mathrm{G}}(a,h,h')} \sum_{j=0}^{n'-1} & \mathcal{E}_{\mathrm{ORTH-LI}}{}^{\star}(\boldsymbol{p}'_{h',j}, \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor}, \mathcal{F}(\mathbf{G}, \lfloor a'+b'j\rfloor, j)^{\top}, \\ & \qquad \boldsymbol{p}_{h,\lfloor a'+b'j\rfloor+1}, \mathcal{F}(\mathbf{G}, \lfloor a'+b'j\rfloor+1, j)^{\top}, \\ & \qquad a'+b'j-\lfloor a'+b'j\rfloor)^2, \\ \text{where } a' = -ab^{-1}, \quad b' = b^{-1}. & \end{aligned}$$

If the subset $\boldsymbol{M}_C$ consists only of inliers, it is reasonable to assume that, for an approximately correct estimate of a and $\mathbf{G}$, each trajectory pair (h,h') in $\boldsymbol{M}_C$ should be classified as an inlier. As with chapter 5, for (h, h') to be classified as an inlier, it is necessary (but not sufficient) for $\mathcal{S}_{\mathrm{G}}(a, h, h', \mathbf{G})$ to satisfy a threshold test. Analogously to chapter 5, the threshold test is defined as

$$\mathcal{C}_{\mathrm{G-T}}(a, \mathbf{G}, h, h') = \begin{cases} 1 & : \quad \mathcal{S}_{\mathrm{G}}(a, \mathbf{G}, h, h') \leq 3.84\sigma^2 \\ 0 & : \quad \text{otherwise} \end{cases} ,$$

for a pre-chosen standard deviation σ. The refined estimate of $(a, \mathbf{G})$ is discarded if $\mathcal{C}_{\mathrm{G-T}}(a, \mathbf{G}, h, h') = 0$ for *any* trajectory pair (h, h') in the data subset $\boldsymbol{M}_C$.

Trajectory correspondences in the set $\boldsymbol{M}$ can be classified as inliers or outliers by means of a threshold test, and choosing the inliers such that none are mutually exclusive. A pair of matches (h, h') and (g, g') can only be mutually exclusive if they are both visible in at least one frame in either sequence. The number of frames in which both are visible is given by

$$\mathcal{L}(h, h', g, g') = \sum_{i=0}^{n-1} v_{h,i} v_{g,i} + \sum_{j=0}^{n'-1} v'_{h',j} v'_{g',j}.$$

Note this is identical to the definition of $\mathcal{L}$ used in chapter 5. A trajectory pair (g, g') precludes (h, h') from being an inlier if the function $\mathcal{D}_{\mathrm{G}}(a, \mathbf{G}, h, h', g, g')$ equates to 0, for $\mathcal{D}_{\mathrm{G}}$ given by

$$\mathcal{D}_{\mathrm{G}}(a, \mathbf{G}, h, h', g, g') = \begin{cases} 0 & : \quad ((h = g) \veebar (h' = g')) \wedge (\mathcal{C}_{\mathrm{G}}(a, \mathbf{G}, g, g') = 1) \wedge \\ & \quad\; (\mathcal{S}_{\mathrm{G}}(a, \mathbf{G}, h, h') \geq \mathcal{S}_{\mathrm{G}}(a, \mathbf{G}, g, g')) \wedge (\mathcal{L}(h, h', g, g') \geq 0) \\ 1 & : \quad \text{otherwise} \end{cases} ,$$

where $\veebar$ and $\wedge$ denote the logical operators ‘exclusive or’ and ’and’ respectively. The classification function $\mathcal{C}_{\mathrm{G}}(a, \mathbf{G}, h, h')$ returns 1 if (h, h') is considered an inlier according to the estimates of $(a, \mathbf{G})$, and 0 otherwise. The classification function is defined as

$$\mathcal{C}_{\mathrm{G}}(a, \mathbf{G}, h, h') = \mathcal{C}_{\mathrm{G-T}}(a, \mathbf{G}, h, h') \prod_{(g,g) \in \boldsymbol{M}} \mathcal{D}_{G}(a, \mathbf{G}, h, h', g, g'),$$

Note that, as discussed in chapter 5, the classification of a trajectory correspondence may be dependent on the classification of other correspondences with lower associated error measures. Correspondences must be classified in ascending order of $\mathcal{S}_{\mathrm{G}}$.

On the basis of the classification function $\mathcal{C}_{\mathrm{G}}$, the set of matches $\boldsymbol{M}$ can be partitioned into inlier and outlier sets given by functions

$$\mathcal{M}_{\mathrm{GI}}(a, \mathbf{G}) = \left\{(h, h') \in \boldsymbol{M} \mid \mathcal{C}_{\mathrm{G}}(a, \mathbf{G}, h, h') = 1\right\},$$
$$\mathcal{M}_{\mathrm{GO}}(a, \mathbf{G}) = \left\{(h, h') \in \boldsymbol{M} \mid \mathcal{C}_{\mathrm{G}}(a, \mathbf{G}, h, h') = 0\right\}.$$

The number of elements in the inlier and outlier sets are given by $\mathcal{N}_{\mathrm{GI}}(a, \mathbf{G})$ and $\mathcal{N}_{\mathrm{GO}}(a, \mathbf{G})$ respectively. The robust cost measure used to assess the refined estimate of $(a, \mathbf{G})$ is given by

$$\mathcal{R}_{\mathrm{G}}(a,\mathbf{G}) = \mathcal{N}_{\mathrm{GI}}(a,\mathbf{G})\frac{\sum_{(h,h')\in\mathcal{M}_{\mathrm{GI}}(a,\mathbf{G})}\mathcal{S}_{\mathrm{G-SUM}}(a,\mathbf{G},h,h')}{\sum_{(h,h')\in\mathcal{M}_{\mathrm{GI}}(a,\mathbf{G})}\mathcal{W}_{\mathrm{G}}(a,\mathbf{G},h,h')} + 3.84\sigma^2\mathcal{N}_{\mathrm{GO}}(a,\mathbf{G}), \tag{6.20}$$

where $\mathcal{S}_{\mathrm{G-SUM}}(a,\mathbf{G},h,h') = \mathcal{S}_{\mathrm{G}}(a,\mathbf{G},h,h')\mathcal{W}_{\mathrm{G}}(a,\mathbf{G},h,h')$.

Note this is the equivalent function to $\mathcal{R}$ used in chapter 5 and therefore has the same properties. Given two estimates of $(a,\mathbf{G})$ with the same number of associated classified inliers, preference is given to the one with the lowest average interpolated epipolar error.

The majority of iterations during the robust estimation will yield poor estimates of $(a,\mathbf{G})$. To reduce the expense of a full evaluation for poor estimates, a coarse sampling of epipolar errors is measured for every trajectory pair (h,h') in the total set of matches $\boldsymbol{M}$. This is achieved by defining coarse frame sets $V_{h,h'}$ and $V'_{h,h'}$ for every $(h,h')\in\boldsymbol{M}$, and measuring their associated coarse errors $\overline{\mathcal{S}}_{\mathrm{G}}$, as defined in equation 6.18. On the basis of these error measures, trajectory correspondences can be classified as inliers or outliers, using the function

$$\overline{\mathcal{C}}_{\mathrm{G}}(a,\mathbf{G},h,h') = \overline{\mathcal{C}}_{\mathrm{G-T}}(a,\mathbf{G},h,h')\prod_{(g,g)\in\boldsymbol{M}}\overline{\mathcal{D}}_{G}(a,\mathbf{G},h,h',g,g'),$$

where the function $\overline{\mathcal{C}}_{\mathrm{G-T}}$ determines if a trajectory correspondence (h,h') has an average interpolated epipolar error satisfying a threshold test, for frames in the sets $V_{h,h'}$ and $V'_{h,h'}$. This function is defined as

$$\overline{\mathcal{C}}_{\mathrm{G-T}}(a,\mathbf{G},h,h') = \begin{cases} 1 & : \quad \overline{\mathcal{S}}_{\mathrm{G}}(a,\mathbf{G},h,h') \leq 3.84\sigma^2 \\ 0 & : \quad \text{otherwise} \end{cases}$$

Similarly, $\overline{\mathcal{D}}_{G}(a,\mathbf{G},h,h',g,g')$ determines if the trajectory correspondence (g,g') precludes (h,h') from being classified as an inlier, based on their associated errors $\overline{\mathcal{S}}_{\mathrm{G}}$ computed using the coarse frame sets. The function $\overline{\mathcal{D}}_{G}$ is given by

$$\overline{\mathcal{D}}_{\mathrm{G}}(a,\mathbf{G},h,h',g,g') = \begin{cases} 0 & : \quad ((h=g)\veebar(h'=g')) \wedge (\overline{\mathcal{C}}_{\mathrm{G}}(a,\mathbf{G},g,g')=1) \wedge \\ & \quad \big(\overline{\mathcal{S}}_{\mathrm{G}}(a,\mathbf{G},h,h') \geq \overline{\mathcal{S}}_{\mathrm{G}}(a,\mathbf{G},g,g')\big) \wedge (\mathcal{L}(h,h',g,g')\geq 0) \\ 1 & : \quad \text{otherwise} \end{cases}$$

Using the coarse classifying function $\overline{\mathcal{C}}_{\mathrm{G}}$, inlier and outlier sets are given by

$$\overline{\mathcal{M}}_{\mathrm{GI}}(a,\mathbf{G}) = \left\{(h,h')\in\boldsymbol{M} \mid \overline{\mathcal{C}}_{\mathrm{G}}(a,\mathbf{G},h,h')=1\right\},$$
$$\overline{\mathcal{M}}_{\mathrm{GO}}(a,\mathbf{G}) = \left\{(h,h')\in\boldsymbol{M} \mid \overline{\mathcal{C}}_{\mathrm{G}}(a,\mathbf{G},h,h')=0\right\},$$

with the sizes of the sets given by $\overline{\mathcal{N}}_{\mathrm{GI}}(a,\mathbf{G})$ and $\overline{\mathcal{N}}_{\mathrm{GO}}(a,\mathbf{G})$ respectively. An approximation to the robust cost function $\mathcal{R}_{\mathrm{G}}$ as defined in equation 6.20, is therefore

$$\overline{\mathcal{R}}_{\mathrm{G}}(a,\mathbf{G}) = \overline{\mathcal{N}}_{\mathrm{GI}}(a,\mathbf{G})\frac{\sum_{(h,h')\in\overline{\mathcal{M}}_{\mathrm{GI}}(a,\mathbf{G})}\overline{\mathcal{S}}_{\mathrm{G-SUM}}(a,\mathbf{G},h,h')}{\sum_{(h,h')\in\overline{\mathcal{M}}_{\mathrm{GI}}(a,\mathbf{G})}\overline{\mathcal{W}}_{\mathrm{G}}(a,\mathbf{G},h,h')} + 3.84\sigma^2\overline{\mathcal{N}}_{\mathrm{GO}}(a,\mathbf{G}),$$

where $\overline{\mathcal{S}}_{\mathrm{G-SUM}}(a,\mathbf{G},h,h') = \overline{\mathcal{S}}_{\mathrm{G}}(a,\mathbf{G},h,h')\overline{\mathcal{W}}_{\mathrm{G}}(a,\mathbf{G},h,h')$.

An estimate of (a, G) is first assessed by computing $\overline{\mathcal{R}}_{\mathrm{G}}$. A full evaluation of (a, G), achieved by computing $\mathcal{R}_{\mathrm{G}}$, only proceeds if $\overline{\mathcal{R}}_{\mathrm{G}}(a, \mathsf{G})$ yields a smaller value than the minimum of the robust costs $\mathcal{R}_{\mathrm{G}}$ for all previous iterations.

The robust estimation of G and a is summarised in algorithm 6.1. Once the robust estimation is completed for all frame offset values a, the best solution found is refined using a direct minimisation of the robust cost function $\mathcal{R}_{\mathrm{G}}$ with Levenberg-Marquardt.

6.5 Real Video Sequence Pair Tests

A robust estimation of both a and G was tested using the bouncing balls video sequences described in chapter 5. Although the video sequences view a common area of the scene with considerable 3D variation, this makes it a useful test case as a variety of stationary scene point correspondences can be used to assess the resulting estimate of the spatial alignment G.

Finding well-dispersed accurate stationary matches on the ground plane is difficult for this video sequence pair, due to occlusions and camera motions which cause many areas of the ground plane to only be visible for a limited number of frames. Consequently, the reconstructed 3D scene point locations may be of low accuracy. For this reason, two separate triplets of stationary matches are considered. The first, denoted in the forthcoming results as GP, consists of three scene point correspondences on the ground plane, although the area they cover is small compared to the region of the scene observable to both cameras. The second case, denoted WP, consists of three stationary scene point correspondences more widely dispersed over the scene, two of which are highly elevated and visible across a wider range of frames in both video sequences.

For each triplet of stationary point correspondences, algorithm 6.1 was tested 100 times, with each test being followed by a minimisation of the robust cost function $\mathcal{R}_{\mathrm{G}}$ using Levenberg-Marquardt. A refinement of a, G, and the match classification can occur during the minimisation. The first and second video sequences have 305 and 298 frames respectively. Minimum frame range sizes r_{min} and r'_{min} were both set to 275, which defines a range of frame offset values bounded by $a_{min} = -29$ and $a_{max} = 23$. This choice for r_{min} and r'_{min} reflects a prior assumption that the cameras started recording within approximately one second. This may seem restrictive, but it is helpful for multiple tests due to the computational cost of the relinearisation step.

The ball detection and tracking described in section 5.5 provides 22 trajectories in the first video sequence, and 19 in the second video sequence. The set of matches $\boldsymbol{M}$ for this example therefore has size 418. A manual estimate of synchrony for the bouncing balls example admits 19 correct matches for the moving scene points, with an additional correct match being admissible for a frame offset differing from the manual estimate by one quarter of a frame. A successful classification of matches should therefore identify either 19 or 20 of the correct matches as inliers, and classify the remaining trajectory pairs as outliers.

Algorithm 6.1 (Robust Estimation of $\boldsymbol{a}$ and $\mathbf{G}$)

```
Choose stationary matches, compute $\mathbf{Q}, \mathbf{Q}'$, and corresponding $\hat{\mathbf{Q}}$, $\hat{\mathbf{Q}}'$
Choose a small probability of failure $p_f < 1$, and random subset size $c$
Number of inliers $\mu \leftarrow c$
Choose minimum frame range sizes $r_{min}, r'_{min}$
Compute frame offset range $a_{min}, a_{max}$
Lowest error $\epsilon^* \leftarrow \infty$
Iteration counts $k_a \leftarrow 0$, $\forall a \in \{a_{min} + k\,min(1,b) \leq a_{max} \,|\, k \in \mathbb{N}\}$
while $\exists\, k_a < \mathcal{I}(a,\mu)$ do
  for each $a \in \{a_{min} + k\,min(1,b) \leq a_{max} \,|\, k \in \mathbb{N}\,,\; k_a < \mathcal{I}(a,\mu)\}$ do
    Determine the set of usable matches $\boldsymbol{M}_V$
    $k_a \leftarrow k_a + 1$
    Randomly choose $c$ correspondences from $\boldsymbol{M}_V$ to produce subset $\boldsymbol{M}_C$
    if $\boldsymbol{M}_C$ contains conflicting matches then proceed to next iteration
    Select 6 correspondences in synchronous frames from the subset $\boldsymbol{M}_C$
    Compute estimates of $\mathbf{G}$ as described in section 6.3
    for each estimate $\mathbf{G}$ do
      $\hat{a} \leftarrow a$
      Refine $(\hat{a}, \mathbf{G})$ by minimising $\overline{\mathcal{S}}_{\mathrm{G-TOTAL}}$ (equation 6.19)
      if $\mathcal{C}_{\mathrm{G-T}}(\hat{a}, \mathbf{G}, h, h') = 1\;\; \forall (h,h) \in \boldsymbol{M}_C$ then
        if $\overline{\mathcal{R}}_{\mathrm{G}}(\hat{a}, \mathbf{G}) < \epsilon^*$ then
          $\epsilon \leftarrow \mathcal{R}_{\mathrm{G}}(\hat{a}, \mathbf{G})$
          if $\epsilon < \epsilon^*$ then
            $(a^*, \mathbf{G}^*, \epsilon^*) \leftarrow (\hat{a}, \mathbf{G}, \epsilon)$
            $\mu \leftarrow max\left(\mu, \sum_{(h,h') \in \boldsymbol{M}} \mathcal{C}_{\mathrm{G}}(a^*, \mathbf{G}^*, h, h')\right)$
          end if
        end if
      end if
    end for
  end for
end while
Return $a^*, \mathbf{G}^*$ and $\{\mathcal{C}_{\mathrm{G}}(a^*, \mathbf{G}^*, h, h') | (h,h') \in \boldsymbol{M}\}$
```

Case	Percentage of tests with classified matches			
	17 Correct	18 Correct	19 Correct	0 Incorrect
GP	0	4	96	100
WP	1	1	98	100

Table 6.1: Matches identified for the bouncing balls example, for a robust estimation of both a and $\mathbf{G}$, followed by a minimisation of $\mathcal{R}$

Case	Median VSE	Maximum VSE
GP	0.1207	0.25
WP	0.1213	0.142

Table 6.2: Synchronisation quality for the bouncing balls example, for a robust estimation of both a and $\mathbf{G}$, followed by a minimisation of $\mathcal{R}$

Results concerning the match classification are shown in table 6.1. In every test, all incorrect matches were classified as outliers. Additionally, 19 correct matches were classified as inliers in over 95% of tests for both cases. Note that the remaining tests still identified a significant majority of the correct matches.

Measures of the synchronisation quality are shown in table 6.2. In each test, the resulting frame offset a is compared to the manual synchronisation estimate. The quality of the synchronisation in each test is determined by computing the Video Synchronisation Error (VSE) defined in section 3.5.3, assuming the manual synchronisation estimate coincides with truth. The median and maximum VSE across 100 tests is given, for both the GP and WP stationary point triplets. Note that every test produced a synchronisation estimate which differs from the manual estimate by only a fraction of a frame. In the GP case, the highest VSE is 0.25, which occurs in 4 of the 100 tests. This occurs due to the presence of the single correct match which, for the manual synchronisation estimate, has no measurable epipolar errors. In the 4 tests for which the VSE is 0.25, this match is classified as an inlier. This classification did not occur in the WP case, as evidenced by the lower VSE measures.

Neither synchronisation nor classification results explicitly provide a measure of quality for the spatial alignment. To assess the quality of the estimated transformation $\mathbf{G}$, manually identified stationary scene point matches were used. After each test, image point locations from both video sequences were used to reconstruct a sparse set of 20 stationary scene features. A recovered scene point is then projected to each frame where an estimate of the location of its image is available. The reprojection error, defined as the 2D Euclidean distance between the measured and reprojected image locations, is computed.

Case	Median RMS-RPE	Maximum RMS-RPE
GP	16.8	41.6
WP	6.3	26.9

Table 6.3: Spatial alignment quality for the bouncing balls example, for a robust estimation of both a and $\mathbf{G}$, followed by a minimisation of $\mathcal{R}$

The root-mean-squared reprojection error (RMS-RPE), for all 20 scene points, provides a measure of quality by which the spatial alignment $\mathbf{G}$ can be assessed. Table 6.5 shows the maximum and median RMS-RPE measures.

Even a casual comparison of these results shows that the two cases produce spatial alignments of differing quality. In the WP case, an RMS-RPE of 6.3 is achieved in 98 of the tests. The maximum error of 26.9 occurs for the test which only classified 17 of the correct matches as inliers. While an error of 6.3 can be considered high, it still suggests that an *approximate* spatial alignment has been achieved. This alignment is illustrated in figure 6.3, which shows the optical centre locations for the second camera, transformed according to two distinct spatial alignments. The blue path shows the optical centre locations as determined by a projective transformation estimated from a cloud of stationary scene point matches. This is the spatial alignment that was used to register the cameras in chapter 5. The red path shows the optical centre locations as determined by algorithm 6.1, followed by a minimisation of $\mathcal{R}_{\mathrm{G}}$, using the WP triplet of stationary matches. A comparison with figure 5.4 shows that both of the projective transformations yield similar relative camera positions.

The GP case produces much worse spatial alignments. This may in part be due to the smaller area between the triplet of stationary matches. A deeper analysis of this case reveals another problem. The median error of 16.8 is obtained in 83 of the 100 tests. The next most common result is an RMS-RPE of 38.4, which occurs in 13 of the tests. These two results represent significantly different spatial alignments, but both admit the classification of 19 correct matches, and accurate synchronisation. This indicates that the GP case must be ill-conditioned for the purposes of estimating the spatial alignment.

A possible cause for the ill-conditioned nature of this example is depicted in figure 6.4, which shows baselines of a sparse set of approximately synchronous frame pairs, where the cameras are spatially registered with the projective transformation used in chapter 5. The baselines are only shown across the range in time during which the bouncing balls were observed. Note that throughout much of this time, the baselines appear to be roughly parallel, approximating the degeneracy described in section 6.3.4. This raises the possibility that the WP case may also be poorly conditioned, but that the associated stationary scene point constraints may simply be more 'independent' from the moving point constraints.

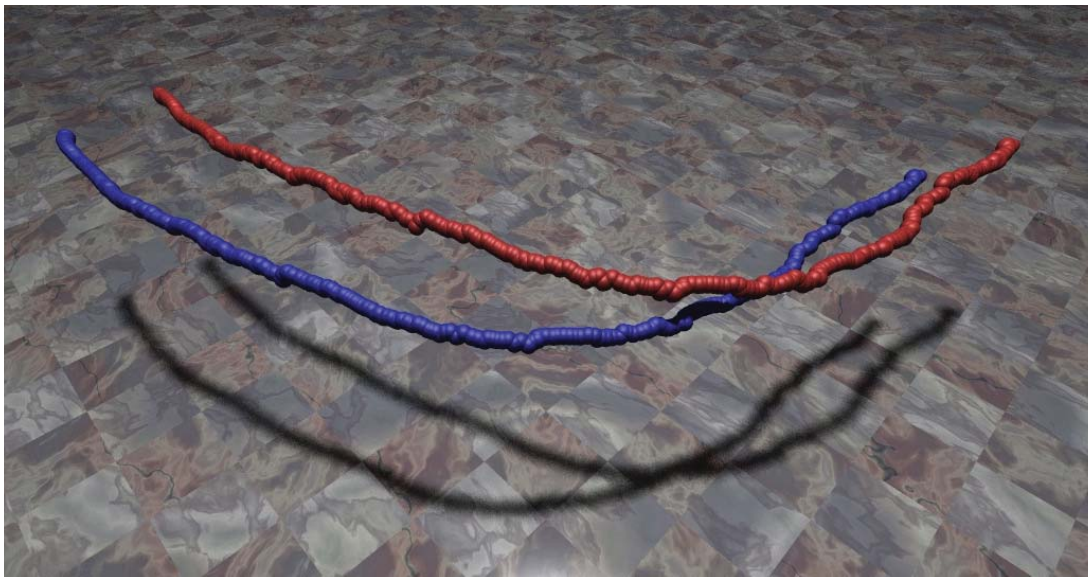


Figure 6.3: A comparison of optical centre locations for two potential alignments of the second camera in the bouncing balls example

Figure 6.4: Baselines of approximately synchronous frame pairs for the bouncing balls example

It should also be noted that superior alignments could hypothetically be found with an additional step. The projective transformation $\mathbf{G}$ is constrained to precisely relate a triplet of stationary scene point matches. Better results may be achievable if this constraint is relaxed, by minimising a cost function comprising a mixture of epipolar errors for moving scene points, and reprojection errors for the three stationary scene point correspondences.

6.6 Conclusions

A robust algorithm has been developed which uses moving point information to achieve both temporal and spatial registration for a pair of cameras. This algorithm can be employed in cases where two cameras observe the same dynamic scene, but where the region of static scene structure in common has little 3D variation.

The algorithm uses an adaptation of relinearisation to compute a spatial alignment from a minimal set of moving and stationary scene point constraints in the presence of known or assumed synchrony. The synchronisation is achieved by repeatedly testing across a set of hypothesised frame offsets. Efficiency is improved by terminating an iteration early if the matches used to generate the model fail to classify a cost threshold, or if a coarse evaluation using all matches suggests the estimate is inferior to one found before it.

Tests on a real video pair demonstrate that this algorithm can indeed find approximate spatial alignments. It has also been shown that, even in ill-conditioned spatial cases, accurate synchronisation and match classification is still possible.

Chapter 7
CONCLUSION

Algorithms have been developed which can be applied to the problem of synchronising a pair of independently moving video cameras. It has been demonstrated that, with appropriate numbers of moving scene points observable to both cameras, the resulting synchronisation is typically accurate to within a small fraction of a frame. The contributions of each chapter towards this problem are given in section 7.1. Possible directions for future work on this problem are given in section 7.2.

7.1 Summary of Contributions

In chapter 3, a new cost function to synchronise video sequences recorded by independently moving cameras was introduced. Moving scene points observed by both cameras provide the cue for synchronisation, with measurements for frame indices between integer values made possible by a linear interpolation of epipolar lines. An empirical analysis of this epipolar line interpolation demonstrated its superiority over interpolating residuals from nearest frame pairs. Testing on synthetic and real video sequence examples demonstrate the resulting cost function can be used to achieve accurate synchronisation for both a known and unknown frame rate ratio.

Chapter 4 introduced the concept of synchrony pairs. By assuming a linear interpolation of epipolar lines, a search can be made for pairs of frame indices for which the projections of a moving scene point precisely satisfy the epipolar constraint. Histogram methods were devised that could be used to efficiently obtain an approximate estimate of synchronisation even when the frame rates of the cameras are unknown. Testing demonstrated that this approach offers comparable synchronisation quality to searching uniformly across the parameter space. It was also shown that a greater increase in speed could be achieved by randomly or uniformly sampling frames to generate smaller synchrony pair sets, typically without sacrificing the quality of synchronisation estimates.

The new robust algorithms RATSAC and CATSAC were presented in chapter 5. These algorithms can be applied to problems that admit reducing the time required for each iteration of the robust estimation, at the risk of obtaining a lower quality result. Algorithms were devised for both an exhaustive and random sampling of data, where both methods seek to find the best trade-off between the speed and reliability of the iterations. A prior distribution regarding the number of inliers is required, and Bayesian inference is used to determine how much to speed up the iterations, such that the robust estimation can terminate in the least possible time. When applied to the robust synchronisation of independently moving cameras, both algorithms yield accurate synchronisation, typically within a fraction of a frame, for both a known and unknown frame rate ratio. Despite considering an exhaustive pairing of moving point trajectories as matches, high accuracy is achieved for inlier and outlier classification. By adaptively seeking the best speedup values, a considerable reduction in execution time is possible. This reduction is diminished however, for moving scene points tracked over shorter ranges of frames, as evidenced by the real video sequence pair example.

Chapter 6 introduced the concept of using moving object cues to both achieve synchronisation, and combine two reconstructions of a scene. This is useful in the case where there is insufficient spatial overlap between the reconstructions to achieve an alignment with stationary object information alone. The reconstructions are combined by finding a projective transformation to spatially align the projection matrices associated with each video sequence. Such transformations are found by using three stationary scene point matches between the reconstructions, and six 3D ray pairs from moving scene point correspondences, each of which should intersect when the cameras are both synchronised and spatially aligned. The resulting set of linear and quadratic equations are solved using an adaptation of the relinearisation method. This step then forms the core of a robust algorithm, which repeatedly searches across a range of frame offset values to find the best solution. Repeated testing on a real video sequence pair showed not only that approximate spatial alignment is achievable, but also that synchronisation and correct match classification can occur even when the spatial alignment is poor.

7.2 Future Work

The epipolar line interpolation method described in chapter 3 assumes that epipolar line orientations are similar for successive frame pairs. Such an assumption typically holds, but may be violated in cases where a moving scene point exhibits high disparity in the video sequences, or moves close to the epipoles. In such circumstances, the interpolation direction, and hence the associated epipolar error measure, will be incorrect. The possibility of using more sophisticated methods to avoid such mistakes should be investigated.

Epipolar lines are also assumed to change linearly over the time between frames in a video sequence. This assumption has been shown to permit video sequences to be synchronised to within a fraction of a frame. Using a linear interpolation model may reduce the accuracy of epipolar error measurements for frame indices between integer values. The use of higher order models should be investigated, since they may permit an even finer level of synchronisation.

Chapter 4 concerns the building of a histogram from synchrony pairs. In the case where the frame rate ratio is unknown, each synchrony pair causes a line of cells in a 2D histogram to be incremented. An alternative approach would be to examine epipolar errors around a synchrony pair, and use these errors to compute an estimate of the frame rate ratio. Given such an estimate, only one cell need be incremented. Alternatively, if the accuracy of the frame rate ratio estimate is known, a corresponding short line segment of cells may be incremented.

Both the robust and non-robust methods presented in this thesis assume that moving scene points have been reliably tracked in each video sequence. The tracking of *stationary* scene point features can be enhanced by using known relations, such as the epipolar constraint. In the case of moving scene points observed by independently moving cameras, this type of enhancement is only possible if the video sequences recorded by the cameras are synchronised, and the tracking of moving scene points is not performed independently in each sequence. This therefore raises the possibility of devising an algorithm to achieve both synchronisation, and a constrained tracking of moving scene points.

Some of the methods presented in later chapters may prove useful outside of the problem of synchronisation. The RATSAC method combines an adjustable speedup with an adaptive assumption concerning the number of inliers, and could be readily applied to any robust estimation for which a constructive speedup mechanism exists. CATSAC has similar wider applicability, but only for problems where an unknown model can be estimated from a small number of matches. In chapter 6, the method of relinearisation [29] was adapted to successfully solve systems of multivariate quadratic equations, which are under-constrained in the sense that a plurality of solutions may exist. Through the addition of a final step, the space of solutions is reduced to a finite set, where each solution is defined up to scale. This may be of benefit to other computer vision problems which involve small numbers of quadratic constraints.

Appendix A

PLÜCKER RAY THEOREMS

Theorem A.1 (The null-space of a Plücker Matrix) *A ray described by Plücker matrix* $\mathsf{R} = \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top$ *lies along plane* $\boldsymbol{\pi}$ *if and only if* $\mathsf{R}\boldsymbol{\pi} = \mathbf{0}$.

Proof (i) *If ray* R *lies along plane* $\boldsymbol{\pi}$,

$$\boldsymbol{p}^\top\boldsymbol{\pi} = 0, \quad \boldsymbol{q}^\top\boldsymbol{\pi} = 0.$$

Therefore,

$$\begin{aligned}\mathsf{R}\boldsymbol{\pi} &= (\boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top)\boldsymbol{\pi}\\ &= \boldsymbol{p}\boldsymbol{q}^\top\boldsymbol{\pi} - \boldsymbol{q}\boldsymbol{p}^\top\boldsymbol{\pi}\\ &= 0\boldsymbol{p} - 0\boldsymbol{q} = \mathbf{0}.\end{aligned}$$ □

Proof (ii) *If* $\mathsf{R}\boldsymbol{\pi} = \mathbf{0}$,

$$\boldsymbol{p}\boldsymbol{q}^\top\boldsymbol{\pi} = \boldsymbol{q}\boldsymbol{p}^\top\boldsymbol{\pi}.$$

If $\boldsymbol{q}^\top\boldsymbol{\pi} = u$, *and* $\boldsymbol{p}^\top\boldsymbol{\pi} = v$,
then $u\boldsymbol{p} = v\boldsymbol{q}$.
So either $(u, v) = (0, 0)$, *or* $\boldsymbol{p} \sim \boldsymbol{q}$.
If $\boldsymbol{p} \sim \boldsymbol{q}$, $\mathsf{R} = \emptyset$ *and does not represent a ray in space.*
Hence $\boldsymbol{q}^\top\boldsymbol{\pi} = \boldsymbol{p}^\top\boldsymbol{\pi} = 0$. □

Theorem A.2 (Uniqueness up to Scale) *Given the Plücker matrix defined by* $\mathsf{R} = \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top$, *and Plücker matrix* R' *defined by any other distinct points along the ray,* $\mathsf{R} \sim \mathsf{R}'$.

Proof *We choose two points along the ray as* $(u_1\boldsymbol{p}+v_1\boldsymbol{q})$ *and* $(u_2\boldsymbol{p}+v_2\boldsymbol{q})$. *These 3D points are only different beyond a scale factor if* $(u_1v_2 \neq u_2v_1)$.

$$\begin{aligned}\mathsf{R}' =&(u_1\boldsymbol{p} + v_1\boldsymbol{q})(u_2\boldsymbol{p} + v_2\boldsymbol{q})^\top - (u_2\boldsymbol{p} + v_2\boldsymbol{q})(u_1\boldsymbol{p} + v_1\boldsymbol{q})^\top\\ =&(u_1v_2 - u_2v_1)\boldsymbol{p}\boldsymbol{q}^\top - (u_1v_2 - u_2v_1)\boldsymbol{q}\boldsymbol{p}^\top\\ =&(u_1v_2 - u_2v_1)\mathsf{R} \sim \mathsf{R}.\end{aligned}$$ □

And similarly for the dual.

Theorem A.3 (Duality Constraint) *Given Plücker matrix* R, *and its dual* R^*, *with vector representations* $\boldsymbol{r} = \mathcal{V}_{Ray}(\mathsf{R})$, *and* $\boldsymbol{r}^* = \mathcal{V}_{Ray}(\mathsf{R}^*)$, *then*

$$\boldsymbol{r} \sim \mathsf{J}\boldsymbol{r}^*.$$

Proof *For some pair of points* $(\boldsymbol{p}, \boldsymbol{q})$ *on the ray, and some pair of planes* $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ *intersecting along the ray,*

$$\mathsf{R} = \quad \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top \quad = \begin{bmatrix} 0 & r_1 & r_2 & r_3 \\ -r_1 & 0 & r_4 & -r_5 \\ -r_2 & -r_4 & 0 & r_6 \\ -r_3 & r_5 & -r_6 & 0 \end{bmatrix},$$

$$\mathsf{R}^* = \quad \boldsymbol{\pi}_1\boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2\boldsymbol{\pi}_1^\top \quad = \begin{bmatrix} 0 & r_1^* & r_2^* & r_3^* \\ -r_1^* & 0 & r_4^* & -r_5^* \\ -r_2^* & -r_4^* & 0 & r_6^* \\ -r_3^* & r_5^* & -r_6^* & 0 \end{bmatrix}$$

Since $\boldsymbol{p}^\top\boldsymbol{\pi}_1 = \boldsymbol{q}^\top\boldsymbol{\pi}_1 = \boldsymbol{p}^\top\boldsymbol{\pi}_2 = \boldsymbol{q}^\top\boldsymbol{\pi}_2 = 0$, $\mathsf{R}\mathsf{R}^* = \emptyset$. *Expanding* $\mathsf{R}\mathsf{R}^*$ *and equating it to the zero matrix, we find that*

$$r_i r_j^* = r_{7-i}^* r_{7-j} \quad \forall i, j = 1 \ldots 6.$$

Choosing some $k \in 1 \ldots 6$ *such that* $r_k \neq 0$,

$$r_j^* = \frac{r_{7-k}^*}{r_k} r_{7-j} \quad \forall j \in 1 \ldots 6$$

Therefore, $\boldsymbol{r^*} \sim \mathsf{J}\boldsymbol{r}$. □

Theorem A.4 (Ray Intersection) *Two rays with vector representations* $\boldsymbol{r}$ *and* $\boldsymbol{r}'$ *intersect if and only if* $\boldsymbol{r}^\top \mathbf{J} \boldsymbol{r}' = 0$.

Proof *We define 4 points in 3D space*

$$
\begin{aligned}
\boldsymbol{p} =& [p_1, p_2, p_3, p_4]^\top, \\
\boldsymbol{q} =& [q_1, q_2, q_3, q_4]^\top, \\
\boldsymbol{p}' =& [p'_1, p'_2, p'_3, p'_4]^\top, \\
\boldsymbol{q}' =& [q'_1, q'_2, q'_3, q'_4]^\top.
\end{aligned}
$$

The rays $\boldsymbol{r}$ *and* $\boldsymbol{r}'$ *are given by*

$$
\begin{aligned}
\boldsymbol{r} =& (\boldsymbol{p} \curlywedge \boldsymbol{q}) = \left[\begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix}, \begin{vmatrix} p_1 & p_3 \\ q_1 & q_3 \end{vmatrix}, \begin{vmatrix} p_1 & p_4 \\ q_1 & q_4 \end{vmatrix}, \begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix}, -\begin{vmatrix} p_2 & p_4 \\ q_2 & q_4 \end{vmatrix}, \begin{vmatrix} p_3 & p_4 \\ q_3 & q_4 \end{vmatrix} \right]^\top, \\
\boldsymbol{r}' =& (\boldsymbol{p}' \curlywedge \boldsymbol{q}') = \left[\begin{vmatrix} p'_1 & p'_2 \\ q'_1 & q'_2 \end{vmatrix}, \begin{vmatrix} p'_1 & p'_3 \\ q'_1 & q'_3 \end{vmatrix}, \begin{vmatrix} p'_1 & p'_4 \\ q'_1 & q'_4 \end{vmatrix}, \begin{vmatrix} p'_2 & p'_3 \\ q'_2 & q'_3 \end{vmatrix}, -\begin{vmatrix} p'_2 & p'_4 \\ q'_2 & q'_4 \end{vmatrix}, \begin{vmatrix} p'_3 & p'_4 \\ q'_3 & q'_4 \end{vmatrix} \right]^\top.
\end{aligned}
$$

The rays intersect if and only if the 4 *points are coplanar, or equivalently, if* $|\mathbf{W}| = 0$, *for matrix* $\mathbf{W}$ *defined as*

$$
\mathbf{W} = [\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{p}', \boldsymbol{q}']^\top,
$$

$$
\begin{aligned}
|\mathbf{W}| =& p_1 \begin{vmatrix} q_2 & q_3 & q_4 \\ p'_2 & p'_3 & p'_4 \\ q'_2 & q'_3 & q'_4 \end{vmatrix} - p_2 \begin{vmatrix} q_1 & q_3 & q_4 \\ p'_1 & p'_3 & p'_4 \\ q'_1 & q'_3 & q'_4 \end{vmatrix} + p_3 \begin{vmatrix} q_1 & q_2 & q_4 \\ p'_1 & p'_2 & p'_4 \\ q'_1 & q'_2 & q'_4 \end{vmatrix} - p_4 \begin{vmatrix} q_1 & q_2 & q_3 \\ p'_1 & p'_2 & p'_3 \\ q'_1 & q'_2 & q'_3 \end{vmatrix} \\
=& p_1 q_2 r'_6 + p_1 q_3 r'_5 + p_1 q_4 r'_4 - p_2 q_1 r'_6 + p_2 q_3 r'_3 - p_2 q_4 r'_2 \\
& - p_3 q_1 r'_5 - p_3 q_2 r'_3 + p_3 q_4 r'_1 - p_4 q_1 r'_4 + p_4 q_2 r'_2 - p_4 q_3 r'_1 \\
=& (p_3 q_4 - p_4 q_3) r'_1 + (p_4 q_2 - p_2 q_4) r'_2 + (p_2 q_3 - p_3 q_2) r'_3 + \\
& (p_1 q_4 - p_4 q_1) r'_4 + (p_1 q_3 - p_3 q_1) r'_5 + (p_1 q_2 - p_2 q_1) r'_6 \\
=& r_6 r'_1 + r_5 r'_2 + r_4 r'_3 + r_3 r'_4 + r_2 r'_5 + r_1 r'_6 \\
=& \boldsymbol{r}^\top \mathbf{J} \boldsymbol{r}'.
\end{aligned}
$$

□

Theorem A.5 (Ray Constraint) *A non-zero vector $\boldsymbol{r} \in \Re^6$ represents a ray in 3D space if and only if $\boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0$.*

Proof (i) $\boldsymbol{r} = (\boldsymbol{p} \curlywedge \boldsymbol{q})$ *for some pair of points $\boldsymbol{p}$ and $\boldsymbol{q}$ along the ray. See previous proof, with $\boldsymbol{p} = \boldsymbol{p}'$, and $\boldsymbol{q} = \boldsymbol{q}'$, then $|\mathsf{W}| = \boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0$.* □

Proof (ii) *Given vector $\boldsymbol{r}$ such that $\boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0$, we define R as the matrix such that*

$$\mathcal{V}_{Ray}(\mathsf{R}) = \boldsymbol{r}.$$

We also define matrix R' such that

$$\mathcal{V}_{Ray}(\mathsf{R}') = \mathsf{J}\boldsymbol{r}.$$

We need to show that either R or R' is a valid Plücker ray matrix. If so, then $\mathsf{R}' \sim \mathsf{R}^*$.

$$\mathsf{R}\mathsf{R}' = \begin{bmatrix} 0 & r_1 & r_2 & r_3 \\ -r_1 & 0 & r_4 & -r_5 \\ -r_2 & -r_4 & 0 & r_6 \\ -r_3 & r_5 & -r_6 & 0 \end{bmatrix} \begin{bmatrix} 0 & r_6 & r_5 & r_4 \\ -r_6 & 0 & r_3 & -r_2 \\ -r_5 & -r_3 & 0 & r_1 \\ -r_4 & r_2 & -r_1 & 0 \end{bmatrix} = \emptyset$$

We know that $\mathsf{R}\mathsf{R}' = \emptyset$, since the expansion of equations for non-leading diagonal elements of the right hand side all cancel. The expressions for the leading diagonal elements are all $\pm\frac{1}{2}\boldsymbol{r}^\top \mathsf{J} \boldsymbol{r} = 0$. Now choose any non-zero element of $\boldsymbol{r}$. The two columns of R' that this element appears in are denoted $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$.
Now $\mathsf{R}\boldsymbol{\pi}_1 = \mathbf{0}$, $\mathsf{R}\boldsymbol{\pi}_2 = \mathbf{0}$, and $(\boldsymbol{\pi}_1\boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2\boldsymbol{\pi}_1^\top)$ is a valid ray matrix.

$$\mathsf{R}(\boldsymbol{\pi}_1\boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2\boldsymbol{\pi}_1^\top) = \emptyset.$$

From the proof on the duality constraint,

$$\boldsymbol{r} = \mathsf{J}(\boldsymbol{\pi}_1 \curlywedge \boldsymbol{\pi}_2) = (\boldsymbol{\pi}_1 \curlywedge \boldsymbol{\pi}_2)^* \qquad \square$$

Theorem A.6 (Inner Product) *Given two Plücker rays, with vector representations*

$$\boldsymbol{r} = (\boldsymbol{p} \curlywedge \boldsymbol{q}), \quad \boldsymbol{r}' = (\boldsymbol{a} \curlywedge \boldsymbol{b}),$$

the inner product of these vectors is such that

$$\boldsymbol{r}^\top \boldsymbol{r}' = \boldsymbol{a}^\top(\boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top)\boldsymbol{b}.$$

Proof

$$\begin{aligned}
\boldsymbol{r} =& \left[\begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix}, \begin{vmatrix} p_1 & p_3 \\ q_1 & q_3 \end{vmatrix}, \begin{vmatrix} p_1 & p_4 \\ q_1 & q_4 \end{vmatrix}, \begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix}, -\begin{vmatrix} p_2 & p_4 \\ q_2 & q_4 \end{vmatrix}, \begin{vmatrix} p_3 & p_4 \\ q_3 & q_4 \end{vmatrix} \right]^\top \\
=& [r_1, r_2, r_3, r_4, r_5, r_6]^\top \\
\boldsymbol{r}' =& \left[\begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}, \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}, \begin{vmatrix} a_1 & a_4 \\ b_1 & b_4 \end{vmatrix}, \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}, -\begin{vmatrix} a_2 & a_4 \\ b_2 & b_4 \end{vmatrix}, \begin{vmatrix} a_3 & a_4 \\ b_3 & b_4 \end{vmatrix} \right]^\top
\end{aligned}$$

Let $\mathsf{R} = \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top$, *so that*

$$\boldsymbol{r} = \mathcal{V}_{Ray}(\mathsf{R}).$$

$$\begin{aligned}
\boldsymbol{a}^\top \mathsf{R} \boldsymbol{b} =& \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} 0 & r_1 & r_2 & r_3 \\ -r_1 & 0 & r_4 & -r_5 \\ -r_2 & -r_4 & 0 & r_6 \\ -r_3 & r_5 & -r_6 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
=& \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} r_1 b_2 + r_2 b_3 + r_3 b_4 \\ -r_1 b_1 + r_4 b_3 - r_5 b_4 \\ -r_2 b_1 - r_4 b_2 + r_6 b_4 \\ -r_3 b_1 + r_5 b_2 - r_6 b_3 \end{bmatrix} \\
=& (a_1 b_2 - a_2 b_1) r_1 + (a_1 b_3 - a_3 b_1) r_2 + (a_1 b_4 - a_4 b_1) r_3 + \\
& (a_2 b_3 - a_3 b_2) r_4 + (a_4 b_2 - a_2 b_4) r_5 + (a_3 b_4 - a_4 b_3) r_6 \\
=& r_1 r_1' + r_2 r_2' + r_3 r_3' + r_4 r_4' + r_5 r_5' + r_6 r_6' \\
=& \boldsymbol{r}^\top \boldsymbol{r}'. \qquad \square
\end{aligned}$$

Theorem A.7 (Projection to a Line Matrix) *Given Plücker ray matrix* R *and* 3×4 *projection matrix* M*, the image line* $\boldsymbol{l} = [l_1, l_2, l_3]^\top$ *describing the projection of the ray is defined by*

$$\mathsf{L} = \begin{bmatrix} 0 & l_3 & -l_2 \\ -l_3 & 0 & l_1 \\ l_2 & -l_1 & 0 \end{bmatrix} = \mathsf{MRM}^\top.$$

Proof *Choose two 3D points along the ray,* $\boldsymbol{p}'$ *and* $\boldsymbol{q}'$*, such that* $\mathsf{R} = \boldsymbol{p}'\boldsymbol{q}'^\top - \boldsymbol{q}'\boldsymbol{p}'^\top$*. These 3D points project to image locations* $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *defined by*

$$\begin{aligned} \boldsymbol{p} = [p_1, p_2, p_3]^\top &= \mathsf{M}\boldsymbol{p}', \\ \boldsymbol{q} = [q_1, q_2, q_3]^\top &= \mathsf{M}\boldsymbol{q}'. \end{aligned}$$

The image line $\boldsymbol{l}$ *passing through points* $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *such that* $\boldsymbol{l}^\top\boldsymbol{p} = \boldsymbol{l}^\top\boldsymbol{q} = 0$ *is given by*

$$\boldsymbol{l} = [l_1, l_2, l_3]^\top = \left[\begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix}, \begin{vmatrix} p_3 & p_1 \\ q_3 & q_1 \end{vmatrix}, \begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix} \right]^\top.$$

Now, expanding the equation for line matrix L*,*

$$\begin{aligned} \mathsf{L} &= \mathsf{MRM}^\top \\ &= \mathsf{M}\boldsymbol{p}'\boldsymbol{q}'^\top\mathsf{M}^\top - \mathsf{M}\boldsymbol{q}'\boldsymbol{p}'^\top\mathsf{M}^\top \\ &= \boldsymbol{p}\boldsymbol{q}^\top - \boldsymbol{q}\boldsymbol{p}^\top \\ &= \begin{bmatrix} 0 & (p_1q_2 - p_2q_1) & (p_1q_3 - p_3q_1) \\ (p_2q_1 - p_1q_2) & 0 & (p_2q_3 - p_3q_2) \\ (p_3q_1 - p_1q_3) & (p_3q_2 - p_2q_3) & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & l_3 & -l_2 \\ -l_3 & 0 & l_1 \\ l_2 & -l_1 & 0 \end{bmatrix} \qquad \square \end{aligned}$$

Theorem A.8 (Ray Projection) *Given the ray projection matrix* $\hat{\mathsf{M}}$*, and Plücker ray vector* $\boldsymbol{r}$*, the projection of the ray to a line* $\boldsymbol{l}$ *in the camera described by* $\hat{\mathsf{M}}$ *is given by*

$$\boldsymbol{l} = \hat{\mathsf{M}}^\top \mathsf{J}\boldsymbol{r}.$$

Proof *A ray in space is represented by the Plücker matrix* R *and the Plücker vector* $\boldsymbol{r}$ *such that* $\boldsymbol{r} = \mathcal{V}_{Ray}(\mathsf{R})$*. Define a* 3×4 *projection matrix as*

$$\mathsf{M} = \begin{bmatrix} \boldsymbol{m}_1^\top \\ \boldsymbol{m}_2^\top \\ \boldsymbol{m}_3^\top \end{bmatrix}.$$

Line $\boldsymbol{l} = [l_1, l_2, l_3]^\top$ *in the image can be computed according to the formula from the previous theorem:*

$$\begin{bmatrix} 0 & l_3 & -l_2 \\ -l_3 & 0 & l_1 \\ l_2 & -l_1 & 0 \end{bmatrix} = \mathsf{MRM}^\top.$$

The elements of the line are therefore

$$\begin{aligned} l_1 &= \boldsymbol{m}_2^\top \mathsf{R}\boldsymbol{m}_3, \\ l_2 &= \boldsymbol{m}_3^\top \mathsf{R}\boldsymbol{m}_1, \\ l_3 &= \boldsymbol{m}_1^\top \mathsf{R}\boldsymbol{m}_2. \end{aligned}$$

From the inner product theorem, this equates to

$$\begin{aligned} l_1 &= (\boldsymbol{m_2} \curlywedge \boldsymbol{m}_3)^\top \boldsymbol{r}, \\ l_2 &= (\boldsymbol{m_3} \curlywedge \boldsymbol{m}_1)^\top \boldsymbol{r}, \\ l_3 &= (\boldsymbol{m_1} \curlywedge \boldsymbol{m}_2)^\top \boldsymbol{r}. \end{aligned}$$

Accordingly, the formula for line vector $\boldsymbol{l}$ *is*

$$\boldsymbol{l} = \begin{bmatrix} (\boldsymbol{m_2} \curlywedge \boldsymbol{m}_3)^\top \\ (\boldsymbol{m_3} \curlywedge \boldsymbol{m}_1)^\top \\ (\boldsymbol{m_1} \curlywedge \boldsymbol{m}_2)^\top \end{bmatrix} \boldsymbol{r} = \begin{bmatrix} (\boldsymbol{m_2} \curlywedge \boldsymbol{m}_3)^\top \\ (\boldsymbol{m_3} \curlywedge \boldsymbol{m}_1)^\top \\ (\boldsymbol{m_1} \curlywedge \boldsymbol{m}_2)^\top \end{bmatrix} \mathsf{JJ}\boldsymbol{r} = \hat{\mathsf{M}}^\top \mathsf{J}\boldsymbol{r} \qquad \square$$

Theorem A.9 (Ray Back-Projection) *Given a ray projection matrix $\hat{\mathsf{M}}$, and an image location $\boldsymbol{p} = [x, y, 1]^\top$, the set of 3D points projecting to this image location form a ray in space defined by*

$$\boldsymbol{r} = \hat{\mathsf{M}}\boldsymbol{p}.$$

Proof *We define projection matrix*

$$\mathsf{M} = \begin{bmatrix} \boldsymbol{m}_1^\top \\ \boldsymbol{m}_2^\top \\ \boldsymbol{m}_3^\top \end{bmatrix}.$$

as the 3×4 projection matrix corresponding to $\hat{\mathsf{M}}$, and scene point $\boldsymbol{q}$ as some 3D point such that

$$\boldsymbol{p} \sim \mathsf{M}\boldsymbol{q}.$$

This gives the constraints

$$(\boldsymbol{m}_1^\top - x\boldsymbol{m}_3^\top)\boldsymbol{q} = 0, \quad (\boldsymbol{m}_2^\top - y\boldsymbol{m}_3^\top)\boldsymbol{q} = 0.$$

So, all points projecting to image location $\boldsymbol{p}$ lie on the planes described by $(\boldsymbol{m}_1^\top - x\boldsymbol{m}_3^\top)$ and $(\boldsymbol{m}_2^\top - y\boldsymbol{m}_3^\top)$. The dual of the ray along which points project to location $\boldsymbol{p}$ is given by

$$\begin{aligned}\mathsf{R}^* &= (\boldsymbol{m}_1 - x\boldsymbol{m}_3)(\boldsymbol{m}_2 - y\boldsymbol{m}_3)^\top - (\boldsymbol{m}_2 - y\boldsymbol{m}_3)(\boldsymbol{m}_1 - x\boldsymbol{m}_3)^\top \\ &= (\boldsymbol{m}_1\boldsymbol{m}_2{}^\top - \boldsymbol{m}_2\boldsymbol{m}_1{}^\top) + \\ &\quad (\boldsymbol{m}_2\boldsymbol{m}_3{}^\top - \boldsymbol{m}_3\boldsymbol{m}_2{}^\top)x + \\ &\quad (\boldsymbol{m}_3\boldsymbol{m}_1{}^\top - \boldsymbol{m}_1\boldsymbol{m}_3{}^\top)y.\end{aligned}$$

Using the vector representation for rays,

$$\begin{aligned}\boldsymbol{r}^* &= \mathcal{V}_{Ray}(\mathsf{R}^*) \\ &= [(\boldsymbol{m}_2 \curlywedge \boldsymbol{m}_3), (\boldsymbol{m}_3 \curlywedge \boldsymbol{m}_1), (\boldsymbol{m}_1 \curlywedge \boldsymbol{m}_2)]\boldsymbol{p} \\ &= \mathsf{J}\hat{\mathsf{M}}\boldsymbol{p}.\end{aligned}$$

From the duality constraint, the ray in space $\boldsymbol{r}$ is given by

$$\boldsymbol{r} = \mathsf{J}\boldsymbol{r}^* = \hat{\mathsf{M}}\boldsymbol{p}. \qquad \square$$

Theorem A.10 (Ray Transformation) *Given projective transformation* H*, and ray* $\boldsymbol{r}$ *passing through 3D points* $\boldsymbol{p}$ *and* $\boldsymbol{q}$*, the ray* $\boldsymbol{r}'$ *passing through* $\mathsf{H}\boldsymbol{p}$ *and* $\mathsf{H}\boldsymbol{q}$ *is*

$$\boldsymbol{r}' = [r'_1, r'_2, r'_3, r'_4, r'_5, r'_6]^\top = \hat{\mathsf{H}}\boldsymbol{r} \quad \textit{where}$$

$$\mathsf{H} = \begin{bmatrix} \boldsymbol{h}_1^\top \\ \boldsymbol{h}_2^\top \\ \boldsymbol{h}_3^\top \\ \boldsymbol{h}_4^\top \end{bmatrix}, \quad \hat{\mathsf{H}} = \begin{bmatrix} (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_4)^\top \\ (\boldsymbol{h}_2 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_4 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_3 \curlywedge \boldsymbol{h}_4)^\top \end{bmatrix}.$$

Proof *We define two Plücker matrices* R *and* R' *such that* $\boldsymbol{r} = \mathcal{V}_{Ray}(\mathsf{R})$ *and* $\boldsymbol{r'} = \mathcal{V}_{Ray}(\mathsf{R}')$*. Then,*

$$\begin{aligned} \mathsf{R}' &= \mathsf{H}\boldsymbol{p}\mathsf{H}\boldsymbol{q}^\top - \mathsf{H}\boldsymbol{q}\mathsf{H}\boldsymbol{p}^\top \\ &= \mathsf{H}\boldsymbol{p}\boldsymbol{q}^\top\mathsf{H}^\top - \mathsf{H}\boldsymbol{q}\boldsymbol{p}^\top\mathsf{H}^\top \\ &= \mathsf{H}\mathsf{R}\mathsf{H}^\top. \end{aligned}$$

For each element r'_i*, using the inner product theorem,*

$$\begin{array}{lllll} r'_1 & = & \boldsymbol{h}_1^\top\mathsf{R}\boldsymbol{h}_2 & = & (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_2)^\top\boldsymbol{r} \\ r'_2 & = & \boldsymbol{h}_1^\top\mathsf{R}\boldsymbol{h}_3 & = & (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_3)^\top\boldsymbol{r} \\ r'_3 & = & \boldsymbol{h}_1^\top\mathsf{R}\boldsymbol{h}_4 & = & (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_4)^\top\boldsymbol{r} \\ r'_4 & = & \boldsymbol{h}_2^\top\mathsf{R}\boldsymbol{h}_3 & = & (\boldsymbol{h}_2 \curlywedge \boldsymbol{h}_3)^\top\boldsymbol{r} \\ r'_5 & = & \boldsymbol{h}_4^\top\mathsf{R}\boldsymbol{h}_2 & = & (\boldsymbol{h}_4 \curlywedge \boldsymbol{h}_2)^\top\boldsymbol{r} \\ r'_6 & = & \boldsymbol{h}_3^\top\mathsf{R}\boldsymbol{h}_4 & = & (\boldsymbol{h}_3 \curlywedge \boldsymbol{h}_4)^\top\boldsymbol{r} \end{array}.$$

In vector form, this gives

$$\boldsymbol{r}' = \begin{bmatrix} (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_1 \curlywedge \boldsymbol{h}_4)^\top \\ (\boldsymbol{h}_2 \curlywedge \boldsymbol{h}_3)^\top \\ (\boldsymbol{h}_4 \curlywedge \boldsymbol{h}_2)^\top \\ (\boldsymbol{h}_3 \curlywedge \boldsymbol{h}_4)^\top \end{bmatrix}\boldsymbol{r} = \hat{\mathsf{H}}\boldsymbol{r}. \qquad \square$$

BIBLIOGRAPHY

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstructions. In *Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, December 8-14, 2001*, volume 1, pages 287–292. IEEE Computer Society Press.

[3] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, June 23-25, 1998*, pages 885–891. IEEE Computer Society Press.

[4] R. Carceroni, F. Pádua, G. Santos, and K. Kutulakos. Linear sequence-to-sequence alignment. In *Conference on Computer Vision and Pattern Recognition, Washington DC, June 27 - July 2, 2004*, volume 1, pages 746–753. IEEE Computer Society Press.

[5] Y. Caspi and M. Irani. Alignment of non-overlapping sequences. In *International Conference on Computer Vision, Vancouver, July 7-14, 2001*, volume 2, pages 76–83, Los Alamitos, CA. IEEE Computer Society Press.

[6] Y. Caspi and M. Irani. A step towards sequence-to-sequence alignment. In *Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, June 13-15, 2000*, volume 2, pages 682–689. IEEE Computer Society Press.

[7] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. In *ECCV Workshop on Vision and Modelling of Dynamic Scenes (VAMODS), 2002.*

[8] W. Chojnacki, M.J. Brooks, A. van den Hengel, and D. Gawley. Revisiting hartley's normalised eight-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(9):1172–1177, September 2003.

[9] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Conference on Computer Vision and Pattern Recognition, San Diego, California, June 20-26, 2005*, volume 1, pages 220–226. IEEE Computer Society Press.

[10] O. Chum and J. Matas. Randomized RANSAC with $T_{d,d}$ test. In *British Machine Vision Conference, Cardiff, UK, 2002*, pages 448–457. British Machine Vision Association.

[11] O. Chum, T. Werner, and J. Matas. Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint. In *International Conference on Pattern Recognition, Cambridge, UK, August 23-26, 2004*, pages 112–115. IEEE Computer Society Press.

[12] L. de Agapito, R.I. Hartley, and E. Hayman. Linear calibration of a rotating and zooming camera. In *Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, June 23-25, 1999*, pages 15–21. IEEE Computer Society Press.

[13] L. de Agapito, E. Hayman, and I. Reid. Self-calibration of a rotating camera with varying intrinsic parameters. In *British Machine Vision Conference, Southampton, UK, 1998*, pages 105–114. British Machine Vision Association.

[14] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[15] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *International Conference on Computer Vision, Massachusetts, USA, June 20-23, 1995*, pages 951–962. IEEE Computer Society Press.

[16] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *European Conference on Computer Vision, Santa Margherita Ligure, Italy, May 19-22, 1992*, pages 563–578. Springer-Verlag.

[17] O.D. Faugeras, Q.-T. Luong, and S.J. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision, Santa Margherita Ligure, Italy, May 19-22, 1992*, pages 321–334. Springer-Verlag.

[18] 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*, 24:381–395, 1981.

[19] S. Gibson, J. Cook, T.L.J. Howard, and R.J. Hubbold. Icarus: Interactive reconstruction from uncalibrated image sequences. In *ACM Siggraph 2002 Conference Abstracts and Applications, San Antonio, Texas, July, 2002 Proceedings.*

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

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[22] R.I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *European Conference on Computer Vision, Santa Margherita Ligure, Italy, May 19-22, 1992*, pages 579–587. Springer-Verlag.

[23] R.I. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision, Second Joint European - US Workshop, Azores, Portugal, October 9-14, 1993*, pages 237–256. Springer-Verlag.

[24] R.I. Hartley. Self-calibration from multiple views with a rotating camera. In *European Conference on Computer Vision, Stockholme, Sweden, May 2-6, 1994*, volume 1, pages 471–478. Springer-Verlag.

[25] R.I. Hartley. In defence of the 8-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[26] R.I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Conference on Computer Vision and Pattern Recognition, Urbana-Champaign, Illinois, June 15-18, 1992*, pages 761–764. IEEE Computer Society Press.

[27] T. Huang and O. Faugeras. Some properties of the e-matrix in two-view motion estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(12):1310–1312, December 1989.

[28] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[29] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem. *Lecture Notes in Computer Science*, 1666:19–30, 1999.

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

[31] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[32] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[33] Q.-T. Luong and O.D. Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75, January 1996.

[34] J. Moore and D. Jiang. A rank preserving flow algorithm for quadratic optimization problems subject to quadratic equality constraints. In *International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, April 21-24, 1997*, volume 1, pages 67–70. IEEE Computer Society Press.

[35] P.J. Narayanan, P. Rander, and T. Kanade. Synchronous capture of image sequences from multiple cameras. Technical report, The Robotics Institute, Carnegie Mellon University, December 1995.

[36] G.N. Newsam, D.Q. Huynh, M.J. Brooks, and H.-P. Pan. Recovering unknown focal lengths in self-calibration: An essentially linear algorithm and degenerate configurations. In *International Archives of Photogrammetry and Remote Sensing, Vienna, Austria, July 9-19, 1996*, volume 31-B3, pages 575–580.

[37] 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 *International Conference on Image Processing, Barcelona, Spain, September 14-17, 2003*, volume 1, pages 413–416. IEEE Computer Society Press.

[38] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *International Conference on Computer Vision, Nice, France, October 14-17, 2003*, volume 2, pages 939–945. IEEE Computer Society Press.

[39] I. Reid and A. Zisserman. Goal-directed video metrology. In *European Conference on Computer Vision, Cambridge, UK, April 14-18, 1996*, volume 2, pages 647–658. Springer-Verlag.

[40] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., 1987.

[41] J. Serrat, F. Diego, F. Lumbreras, and J. Álvarez. Synchronization of video sequences from free-moving cameras. In *Iberian Conference on Pattern Recognition and Image Analysis, Girona, Spain, June 6-8, 2007*, volume 2, pages 620–627. Springer-Verlag.

[42] P. Smith, D. Sinclair, R. Cipolla, and K. Wood. Effective corner matching. In *British Machine Vision Conference, Southampton, UK, 1998*, pages 545–556. British Machine Vision Association.

[43] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *DARPA Image Understanding Workshop, Monterey, CA, November, 1998*, pages 1037–1042. Morgan Kauffman.

[44] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 17-19, 1997*, pages 1100–1105. IEEE Computer Society Press.

[45] R. Szeliski and S.B. Kang. Direct methods for visual scene reconstruction. In *IEEE Workshop on Representation of Visual Scenes, June 1995*, pages 26–33.

[46] T. Thormählen, H. Broszio, and P. Mikulastik. Robust linear auto-calibration of a moving camera from image sequences. In *Asian Conference on Computer Vision, Hyderabad, India, January 13-16, 2006*, volume 2, pages 71–80. Springer-Verlag.

[47] T. Thormählen, H. Broszio, and A. Weissenfeld. Keyframe selection for camera motion and structure estimation from multiple views. In *European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004*, pages 523–535. Springer-Verlag.

[48] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[49] P. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):35–61, 2002.

[50] P. Torr and A. Zisserman. Robust computation and parameterization of multiple view relations. In *International Conference on Computer Vision, Bombay, India, January 4-7 1998*, pages 727–732. IEEE Computer Society Press.

[51] P. Torr and A. Zisserman. MLESAC: a new robust estimator with application to estimating image geomtry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[52] P. Tresadern and I. Reid. Synchronizing image sequences of non-rigid objects. In *British Machine Vision Conference, Norwich, UK, 2003*, volume 2, pages 629–638. British Machine Vision Association.

[53] B. Triggs. Autocalibration and the absolute quadric. In *Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 17-19, 1997*, pages 609–614. IEEE Computer Society Press.

[54] B. Triggs, P. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *International Workshop on Vision Algorithms, International Conference on Computer Vision, Corfu, Greece, September 21-22, 1999*, pages 298–372. Springer-Verlag.

[55] T. Tuytelaars and L. Van Gool. Synchronizing video sequences. In *Conference on Computer Vision and Pattern Recognition, Washington DC, June 27 - July 2, 2004*, volume 1, pages 762–768. IEEE Computer Society Press.

[56] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[57] D. Wedge, D. Huynh, and P. Kovesi. Motion guided video sequence synchronization. In *Asian Conference on Computer Vision, Hyderabad, India, January 13-16, 2006*, volume 2, pages 832–841. Springer-Verlag.

[58] D. Wedge, P. Kovesi, and D. Huynh. Trajectory based video sequence synchronization. In *Digital Image Computing: Techniques and Applications, Cairns, Australia, December 6-8, 2005*. IEEE Computer Society Press.

[59] 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 (VAMODS), 2002*.

[60] J. Yan and M. Pollefeys. Video synchronisation via space-time interest point distribution. In *Advanced Concepts for Intelligent Vision Systems, Brussels, Belgium, August 31 - September 3, 2004*.

[61] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2D mosaic from a set of images. In *Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 17-19, 1997*, pages 420–425. IEEE Computer Society Press.