A Structure and Motion Toolkit in Matlab

"Interactive Adventures in S and M"

P. H. S. Torr

Microsoft Research, 7 JJ Thomson Avenue, Cambridge, CB3 0FB,UK

philtorr@microsoft.com

`http://research.microsoft.com/~philtorr/,`

June 2002

Technical Report
MSR-TR-2002-56

This document describes a set of Matlab functions for SAM (structure and motion recovery); its primary aim is to illustrate various methods I have developed over the past few years in this area, as it is easier to learn from a working example than from a paper. Eventually I hope to write a book on the subject which will be released together with Matlab code. This document is a sketch of some of the maths involved in the code so some of the descriptions are somewhat brief, the interested reader is referred to the bibliography. Requirements: Matlab 6, the optimization toolbox and the image processing tool box. The toolboxes are not necessary but and are used infrequently however you will have to hack around the required functions. The main vehicle of interaction is the torr tool GUI which is designed to help people get started. If the code is used in a paper please cite this document. Comments and feedback welcome. Especially typos, bugs and the like. Mail me if you would like to be informed of updates, bug fixes, patches.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Contents

Chapter 1

Introduction

This document is slightly different from a conventional book in that it is a living web document rather than a fixed entity. Publishing a conventional book is not so beneficial in a field which is changing as fast as computer vision. With the spread of the Internet it is possible to produce a book that evolves with the field (time permitting). Within this web-book a Matlab system for SAM is described. The purpose of this document is two fold, one to act as a manual to the Matlab SAM system, and two to act as a tutorial/reference on the practical aspects of designing a SAM system. As such only methods that have been found to work well are implemented and other methods as well as theoretical trivia are not included. As time passes I might flesh out some more of the sections, feedback on anything that is not clear will be always appreciated, although might not be enacted on straight away so please be patient. A lot of the text is taken from my thesis.

How to use this manual The SAM recovery system follows a natural progression, comprising the following phases:

1. **Feature detection**, Chapter 2. In version 1 only Harris corners are included. In future versions it is planned to include canny edge features, and other invariant features more suited to wide baseline matching.

2. **Feature matching**, Chapter 3. In version 1 only cross correlation, and Birch field and Tomasi are implemented. In future versions edge matching will appear and wide baseline invariant matching.

3. **Fundamental Matrix Estimation**, Chapter 4 for non robust and Chapter 5 for robust methods . Several methods are included for this including in the non-robust camp: linear, BOOSAM, non-linear and robust: RANSAC, MLESAC, MAPSAC.

4. **Rematching**, Chapter 6, not implemented in version 1.

5. **Self Calibration, and Recovery of 3D structure**, Chapter 7,to determine a Euclidean frame, structure and camera matrices.

The reader is also referred to Table 5.4. The structure of this document is to assign each phase of the reconstruction, e.g. corner detection, matching, self calibration etc. a separate chapter. Each chapter will begin with some background on the the technical details of that phase, following which will be a set of subsections, one per function. The subsection will explain the both the Matlab call to the function in some detail together with a mathematical description of what it does. This is a very hands-on approach to describing the SAM problem, but it is hoped that this document will serve as a useful teaching aid to those trying to understand an end to end system. The design is highly modular so that parts can be unplugged and new ideas plugged in, to aid research comparisons and testing. For additional information amongst the most useful books on the subject are [13, 25, 33].

1.1 A Quick Start

For those, like myself, who want to learn by example, a series of scripts to illustrate the usage of the functions have been devised. The interested user is encourage to study, modify and combine these scripts, experimenting with parameters and such like. The examples are:

1. **Features** Image loading, feature detection and display given in Section 2.4.
2. **Matching** Section 3.3 gives a script to load two images, generate corners, matches and display them.
3. **Estimation of the fundamental matrix** given in Section 4.1.2.

1.2 GUI or no GUI

The meat of the SAM functions can be run via the `torr_tool` which provides an easy interface to the bulk of the functions described herein. There are also a lot of stand alone programs provided to allow testing and understanding of the individual components. The GUI is described in Chapter 9

1.3 Mex files

Some of the computation in SAM can be highly intensive, to speed things up it has been necessary to implement some things as Mex files. These are provided precompiled for windows, but the source is provided for compilation onto other systems, however to do this you are on your own, and I advice you to read the Matlab manuals carefully if you are not familiar with how to compile Mex files.

1.4 Testing Methodology

For debugging and tutorial purposes a synthetic data generator is provided to generate synthetic sets of matches, motions and calibrations and is described in Chapter 8. I have a very clear idea on how to test algorithms such as the estimation of the fundamental matrix; this involves generation of realistic synthetic test sequences and then seeing how close the estimate is to the known ground truth. Thus I have designed many scripts that test each phase on synthetic data (with the exception of the corner matching/detection part which requires real images). There scripts also serve as useful debugging tools when the data is noise free, the estimate should be exactly the same as the ground truth.

1.5 A warning

Perhaps the weakest element of the whole toolkit is the matching process. Cross correlation of Harris corners initializes the whole process. For version two more robust versions of the correlation process will be tried, such as those of Lowe. In the meantime the GUI allows interactive addition of matches.

Chapter 2

Feature Detection

The corner detector used is that of Harris and Stephens [18] which calculates an interest operator defined according to an auto-correlation of Gaussian smoothed images. Corners only yield sparse information across the image but allow the system to be bootstrapped by allowing for the estimation of epipolar geometry. The size of the convolution mask gives a trade off between the localization of corners and the amount of noise excluded. A mask width of `width = 9` has been found to be suitable over a wide range of scenes. Auto-correlation may be defined as the sum of squares of the difference of image intensities

$$\delta I(\delta x, \delta y) = \sum_{ij \in \text{patch}} (I_1(i+\delta x, j+\delta y) - I_1(i,j))^2 \tag{2.1}$$

whose analytic Taylor expansion is

$$\delta I(\delta x, \delta y) = (\delta x, \delta y)\, \mathbf{N} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \tag{2.2}$$

where

$$\mathbf{N}(x,y) = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{2.3}$$

The two eigenvalues of $\mathbf{N}$ are proportional to the principal curvatures of $\mathbf{N}$ and functions of them have the property of rotational invariance. It is shown [18] that when the trace of the matrix is large there is an edge and when the determinant is large there is an edge or a corner [1]. A corner strength signal is

$$\Phi(x,y) = |\mathbf{N}(x,y)| - \kappa \text{Trace}^2 \mathbf{N}(x,y) \tag{2.4}$$

where $\kappa = 0.04$ is routinely used [2]. Corners are defined at the local maxima of two dimensional quadratic patches fitted to $\Phi(x,y)$, resulting in sub-pixel accuracy, in version 1 subpixel accuracy is not implemented.

[1] The term *corner* is used, even though this is truly an interesting point operator partly, for historical reasons

[2] Harris originally used the determinant divided by the trace, but he discarded this measure to avoid potential division by zero. The value $\kappa = 0.04$ was empirically arrived at as it gave the best result.

2.1 Sub Pixel Accuracy

Given that the corner has been detected at (x, y), subpixel accuracy may be gained by fitting a quadratic approximation to $\Phi(x, y)$

$$ax^2 + by^2 + cxy + dx + ey + f = \Phi(x, y) \tag{2.5}$$

using the nine pixels around (x, y) this leads to 9 equations in 6 unknowns that can be solved by least squares.

2.2 Detecting Corners, `torr_charris`

```
function [c_coord] =
torr_charris(im, ncorners, width, sigma, subpixel)
```

Input:

- `im`, is the image which is passed as an array of doubles (at present only grey level images are used, one day maybe I'll implement colour corners)

  ```
  [i1,map1] = imread([pathname1 filename1]);
  iii =  size(size(i1));
  if iii(2) == 3
      g1 = rgb2gray(i1);
      disp('converting to rgb');
  else
      g1 = i1;
  end
  im = double(g1);
  ```

- `ncorners`, is the number of corners requested, note that due to non-maximal suppression, sometimes fewer corners than `ncorners` may be returned.
- `width`, is the width of the Gaussian used to smooth the image.
- `sigma`, is the standard deviation of the Gaussian used to smooth the image.
- `subpixel`, is a boolean variable which is set to one if subpixel detection is requested, at the moment this is not implemented.

Output: `c_coord` a $n \times 2$ array of (x, y) positions.

2.3 Display: `display_corners_in_figure(handles)`

A simple function to display corners on an image in torr_tool, the meat of it is:

```
plot(ccr1(:,1), ccr1(:,2), 'g+','Parent', ax_handle2);
```

2.4 Feature Generation Example, `torr_cor_script`

To get an easy example of the corner detector in action run script `torr_cor_script`, this will read in the image j1.bmp (the default image) and display a little figure together with crosses showing Harris corners. The example is as simple as:

```
figure
i1 = imread('j1.bmp','bmp');
g1 = rgb2gray(i1);
d1 = double(g1);

ncorners = 500
width = 4
sigma = 1
subpixel = 0

[ccr1]  = torr_charris(d1, ncorners, width, sigma, subpixel);


imshow(g1);
%display corners
hold on
plot(ccr1(:,1), ccr1(:,2), 'g+');
hold off
```

2.5 To do List

1. Subpixel accuracy.
2. Colour corners
3. Canny Edges.
4. Other features

Chapter 3

Feature Matching

Feature matching is perhaps one of the weakest parts of the SAM edifice. The geometry and algorithms in the later chapters are well understood, but the whole algorithm is built on shaky foundations. Correlation matching can work very well for the small baseline case, when there is only a small change in illumination. However for significant changes in perspective and lighting the intensities of corresponding features in two images can undergo large changes, resulting in the failure of correlation matching, and hence the SAM algorithm. It is hoped in future version of the code to address this fundamental problem. An important thing to note is that image coordinates are represented as homogeneous vectors $(x, y, \texttt{m3})$, where `m3` is the variable used in the SAM for the third homogeneous coordinate of image points.

All corners within a certain disparity limit are compared over the two images. In the absence of *a priori* information, this limit is set to `max_disparity` pixels. In the course of the matching process there are often several candidate matches for each feature. Initially the one that is most correlated in image intensities at the corner positions is selected, in a similar manner to the Droid system [18]. As auto-correlation is used to define a feature, the strength of match is obtained by cross-correlation of image intensity over two `half_size` $\times$ `half_size` pixel patches centred on each feature,

$$C = \sum_{ij \in \text{patch}} \left(I_2(i,j) - I_1(i,j)\right)^2 \tag{3.1}$$

where $I_n(i,j)$ is the image intensity at coordinate (i,j) in the nth image. The match with the maximum strength is stored for each corner from the first to the second image. The same process is then applied in reverse from the second to the first image. Matches are accepted into the initial set if they are exhibit a maximum in both comparisons. This has the effect of removing corners which are ambiguous in that they have multiple candidate matches.

3.1 Correlation Matching: `torr_corn_matcher`

At the heart of this code is the function `patch_match`, this takes as input two corner positions and outputs the correlation between them, because this is very computationally intensive and involves loops it is implemented as a MEX file.

```
function [matches12,minc,mat12] =
 torr_corn_matcher(im1, im2, clist1, clist2, max_disparity,half_size)
```

Input:

- `im1, im2` the two input images, arrays of doubles as described in Section 2.2.
- `clist1, clist2` two $nc \times 2$ arrays of corner positions as described in Section 2.2, nc is the number of corners.
- `max_disparity` the size of the search window (square) in the next image.
- `half_size` the half size of the correlation window.

Output:

- `matches12` matches in an $n \times 4$ array of matches $(x, y, x'y')$, in this case n is the number of matches.
- `minc` is the minimum value of C for each corner.
- `mat12` is defined such that `mat(i) = j` means corner i matches to corner j.

3.1.1 Correlation of two patches, `patch_match`

```
Correlation = patch_match(im1,im2,x,y,x',y',half_size,minC));
```

Input:

- `im1, im2` the two input images, arrays of doubles as described in Section 2.2. These are needed to access the intensity values for correlation.
- x, y, x', y' the coordinates of two prospective matches.
- `half_size` the half size of the correlation window.
- `minC` is the minimum value of C so far for that corner (this allows an early jump out if the computed correlation goes greater than C as we sum over all the pixels).

Output: `Correlation` The correlation between the two corners.

3.2 Birchfield and Tomasi Correlation

Within this section an error measure is described based on a modification to that of Birchfield and Tomasi [5]. Comparative tests [42] have shown that this method gives better results than standard correlation. Referring to the notation of their paper, rather than compute $I_{\min}$ and $I_{\max}$ over the left and right pixel, which would only be correct if the epipolar lines lay exactly along the scanlines, we compute these two quantities over the 8-connected neighbourhood of each corresponding pixel in both images. Define the following quantities:

$$\begin{aligned} e_{12}(I_1(x,y), I_2(x',y')) &= \min_{-\frac{1}{2}\le\delta x,\delta y\le\frac{1}{2}} \{|\hat{I}_2(x'+\delta x, y'+\delta y) - I_1(x,y)|^2\} \\ e_{21}(I_1(x,y), I_2(x',y')) &= \min_{-\frac{1}{2}\le\delta x,\delta y\le\frac{1}{2}} \{|I_2(x',y') - \hat{I}_1(x+\delta x, y+\delta y)|^2\} \end{aligned}$$

where $I_i(x,y)$ is the intensity value at x,y in image i, and $\hat{I}_i(x,y)$ is the linearly interpolated function between sample points around x,y. Then the dissimilarity or match cost between two pixels is defined symmetrically as

$$e(x,y,x',y') = \min\{e_{12}(I_1(x,y), I_2(x',y')), e_{21}(I_1(x,y), I_2(x',y'))\}. \tag{3.2}$$

As pointed out in [5] the computation of this is simple, as the extreme points of a piecewise linear function must be its breakpoints; first compute

$$\begin{aligned} I_2^{\max}(x,y) &= \max_{\delta x,\delta y=-1,0,1} \frac{1}{2}\{(I_2(x,y) + I_2(x+\delta x, y+\delta y))\} \\ I_2^{\min}(x,y) &= \min_{\delta x,\delta y=-1,0,1} \frac{1}{2}\{(I_2(x,y) + I_2(x+\delta x, y+\delta y))\} \end{aligned}$$

With these quantities defined:

$$e_{12}(I_1(x,y), I_2(x',y')) = \max\{0, I_1(x,y) - I_2^{\max}(x,y), I_2^{\min}(x,y) - I_1(x,y)\} \tag{3.3}$$

similarly for e_{21}, thus allowing for computation of $e(x,y,x',y')$ This development is not the thrust of the paper; for want of space it is not discussed in detail, but this modification provides a very robust error, which is more tolerant to image sampling than either correlation or the shuffle metric recently advocated for space carving. The main drawback of the Birchfield and Tomasi approach is that it provides somewhat less discriminative power, thus I prefer SSD as presented above.

3.2.1 `birch_match`

The Birchfield and Tomasi algorithm is implemented by the MEX function `birch_match`, which takes the same form as the `patch_match` function.

3.3 Feature Matching Example: `torr_matcher_script`

This script loads two images j1.bmp, j2.bmp, detects corners and displays the result in a figure using `torr_display_matches`. The core code is again very simple (detect corners as in Section 2.4)

```
max_disparity = 20;
half_size = 2;

disp('detecting matches')
[matches12,minc12,mat12] =
 torr_corn_matcher(d1, d2, ccr1, ccr2,max_disparity,half_size);

max_corn_d = 0; %maximum corner motion
n_matches = length(matches12);

med_minc = median(minc12);


disp('the maximum corner disparity is used for edge matching')
max_corn_d


display_numbers = 0;
torr_display_matches(matches12,f1,display_numbers);
```

3.4 Match Display: `torr_display_matches`

This function displays the matches in a figure it may take 1,2 or 3 arguments.

```
function torr_display_matches(matches,display_numbers,f1)
```

Input:

1. `matches`: $n \times 4$ array of matches
2. `display_numbers`: if set to 1 then displays the index of each match.
3. `f1`: figure handle of the figure for the matches to be displayed in.

3.5 To do List

1. Invariant feature matching
2. Edge Matching
3. Guided matching using epipolar geometry.

Chapter 4

Determination of the Fundamental Matrix

This chapter is taken largely from my thesis, but provides some useful background. It was published in [52, 51, 53]. Other excellent sources for estimation of **F** are the review article by Zhang [64], and the book of Hartley and Zisserman [25] or [63]. The main function that you need to estimate the fundamental matrix, **F** is `torr_estimateF`, a wrapper for all the **F** estimation methods, which is described in the next section. There is a confusing plethora of ways to compute **F**, I suggest using a robust estimator like MAPSAC to get a first pass at **F** and then perform a constrained non-linear estimation afterwards to optimize, as described in the next section.

4.1 Fundamental Matrix Estimation Function: `torr_estimateF`

This is a wrapper for several methods to estimate the fundamental matrix, depending on what value of `method` is passed to the algorithm.

```
[f, f_sq_errors, n_inliers,inlier_index,F] =
torr_estimateF( matches, m3, f_optim_parameters, method,
set_rank2, f_init)
```

Input:

- `matches`
- matches in an $nm \times 4$ array of matches $(x, y, x'y')$, as defined in Section 3.1.
- `m3` is the third homogeneous coordinate.
- `f_optim_parameters` is a vector of parameters passed to the estimation algorithm.
- `method` can be

1. 'MAPSAC' (note that at present this is the same as 'MLESAC', assuming uniform priors). Calls `torr_mapsac_F`.
2. 'linear', described in Section 4.5.2, calls `torr_estf`.
3. 'Bookstein', an invariant total least squares fit described in Section 4.5.7, calls `torr_estf_bookstein`.
4. 'BooSam', fitting using the method of Section 4.5.7, followed by iterative least squares, using Sampson's weighting described in Section 4.7, calls `torr_estf_bookstein_sampson`.
5. 'non_linear' a constrained estimate of $\mathbf{F}$ enforcing $|\mathbf{F}| = 0$ as described in Section 4.9, call `torr_nonlinf_mincon2x2`.
6. 'lin+non_lin': 'linear' followed by 'non_linear'.

- `set_rank2`: boolean variable, if it is 1 then the SVD is used to enforce $|\mathbf{F}| = 0$ as described in Section 4.5.6.
- `f_init` this is an initial estimate of $\mathbf{F}$ and is only required for iterative (non-linear)methods, `f_init` is a vector as follows: $(f_1 \ldots f_9)$, as given by (4.14).

Output:

- `f` the solution vector $\mathbf{f} = (f_1 \ldots f_9)$, as given by (4.14).
- `f_sq_errors`, the squared error for each match (given as Sampson's error defined in Theorem 3).
- `n_inliers`, if a robust method is used like MAPSAC then this is the number of inliers detected.
- `inlier_index`, an array returned with robust methods, the ith element is 1 if the ith match is an inlier, 0 otherwise.
- `F`: $\mathbf{F}$ as a 3×3 matrix as given by (4.14).

4.1.1 Specific functions for estimating F

As explained `torr_estimateF` calls one of several specific functions to estimate $\mathbf{F}$. Most of the functions for estimating $\mathbf{F}$ have the same base parameters

```
function f = torr_estf(x1,y1,x2,y2, no_matches,m3,...)
```

Input:

- `x1,y1,x2,y2` are arrays of the n corresponding (x, y, x', y') coordinates. I could have passed them in one $n \times 4$ matrix but in the subsequent calculations it often makes things clearer to identity explicitly the coordinates.
- `no_matches` is the number of matches.
- `m3` is the third homogeneous coordinate.

Output: `f` the solution vector $\mathbf{f} = (f_1 \ldots f_9)$, as given by (4.14).

4.1.2 Example of Estimating F: `torr_test_F`

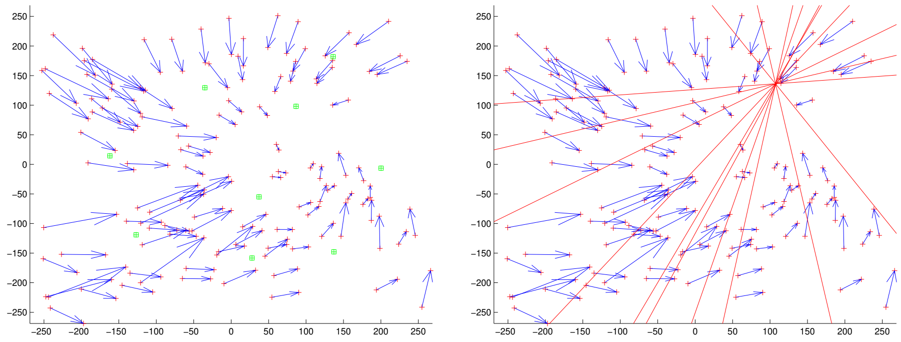
A simple script to show how to estimate **F**, the code first calls `torr_gen_2view_matches` which is described in Chapter 8. The user can then set the method by hand.

```
%a script to display the results of the F matrix...

%third homogeneous coordinate
m3 = 256;

%decide display method
compare = 1;

%choose your method here
method = 2
set_rank2 = 0;
compare = 1;

%generate synthetic data
[true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,true_R,true_t, true_E]
 = torr_gen_2view_matches;
no_matches = length(nx1);
matches = [nx1,ny1,nx2,ny2];


%first estimate F
[f, e1, n_inliers,inlier_index,estimateF]
= torr_estimateF( matches, m3, [], method, set_rank2);

%check errors
e = torr_errf2(f, nx1,ny1,nx2,ny2, no_matches, m3);

%display the result
if compare
    torr_compare_epipoles(estimateF,true_F,matches, m3)
else
    torr_display_epipoles(nF,matches, m3)
end
```

There are two display functions for matches and epipolar geometry, described in the next section, `torr_display_epipoles` or `torr_compare_epipoles`.

4.1.3 Display Functions for epipolar geometry

This functions take a fundamental matrix, and a set of matches, throws up two figures, with the matches displayed. The used can interactively click on one image, and an



(a) (b)

Figure 4.1: *Output of* `torr_display_epipoles`*: (a) A set of synthetic matches generated by* `torr_gen_2view_matches`, *displayed by* `torr_display_matches`*. The squares show using input points, with corresponding epipolar lines shown in (b)*

epipolar line will appear in the second image consistent with `F` and corresponding to the point clicked on. The function uses `ginput` and so input is terminated by a carriage return. An example of the output is shown in figure 4.1.

```
function torr_display_epipoles(F,matches, m3)
```

Input:

- `F`, the 3×3 fundamental matrix $\mathbf{F}$, for which epipolar lines are to be displayed (if you are only interested in displaying the matches, this can be arbitrary).
- `matches`, the $n \times 4$ match array.
- `m3` the third homogeneous coordinate.

```
function torr_compare_epipoles(Fmat1,Fmat2,matches, m3)
```

Input:

- `Fmat1, Fmat2`, two 3×3 fundamental matrices to be compared.
- `matches`, the $n \times 4$ match array.
- `m3` the third homogeneous coordinate.

When this function is called, two figures appear showing the disparity vectors of the matches. Click in figure 1 and two epipolar lines appear in the second figure, one in green (for `Fmat1`) and one in red (`Fmat2`). This function is particularly useful for comparing the estimated $\mathbf{F}$ to the ground truth (as in `torr_test_F`) or in comparing the result of two different algorithms for estimating $\mathbf{F}$.

4.2 An Overview of methods to estimate F

This chapter introduces the elementary concepts necessary for accurate estimation of the fundamental matrix $\mathbf{F}$ which encapsulates all the information on camera motion and camera parameters available from a given set of point correspondences [19, 12]. In subsequent chapters it is shown that the fundamental matrix can be used to guide the feature matching process and to initialize motion segmentation. The fundamental matrix has also been used to initialize structure. This now near ubiquitous employment of the fundamental matrix in vision algorithms prompted the content of this chapter. Specific issues that have not been seriously analysed previously are highlighted—issues such as robustness to gross outliers and the detection of degeneracy in the data–and in subsequent chapters they are addressed. It will be seen that without address to these issues, the estimate of $\mathbf{F}$ can be very poor.

In Section 4.3 the basic terminology that is used throughout this report is given. Section 4.4 describes the fundamental matrix and the distinction between the intrinsic and extrinsic parameters of a pair of cameras. Methods for estimating the fundamental matrix are then analysed. There are several key points to consider: first the choice of error function to be minimized, secondly the choice of parameterization for the fundamental matrix to enforce the constraint that the determinant is zero, and thirdly which estimator should be used. The latter is determined to a large extent by the first two. Three classes of estimators are described. An unnormalized least squares approach is described in Section 4.5, an iterative reweighted least squares method adapted from Sampson [41] and used by Weng *et al.* [60] is described in Section 4.7, and gradient descent estimators are set out in Section 4.8. Testing methodology and functions are described in Section 4.10

4.3 Homogeneous and Projective coordinates

In this section basic concepts and terminology which will be used in the rest of the work are defined. The language of projective geometry is adopted, about which the interested reader may consult Semple and Kneebone [43] for a text book on projective geometry, Mundy and Zisserman [35] for an elegant joining of projective geometry and computer vision, and Kanatani [27] for a discussion of some of the basic computational theory associated with the application of projective geometry.

Vectors will be denoted by boldface type: $\mathbf{a}$, and matrices by boldface and capitals: $\mathbf{A}$. The determinant of a matrix as $|\mathbf{A}|$, the Fröbenius norm of a matrix as $||\mathbf{A}||$.

The image plane is regarded as a 2-D projective space: $\mathcal{P}^2$, a point of $\mathcal{P}^2$ is designated as a triplet $\mathbf{x} = (x_1, x_2, x_3)$ of real numbers, not all of them equal to zero. This triplet is termed a homogeneous coordinate. It is beneficial to work with homogeneous coordinates as this often simplifies the mathematical derivations. Two points of $\mathcal{P}^2$ are equal if there exists a non-zero scalar λ such that $x_i = \lambda x_i$ for $i = 1, \ldots 3$, thus

$$(x_1, x_2, x_3) = \lambda(x_1, x_2, x_3) \tag{4.1}$$

We introduce the notation $\mathbf{a} \sim \mathbf{b}$ to indicate equality up to a scale factor when it is not obvious that we are dealing with homogeneous coordinates. The embedding

$\mathcal{R}^2 \subset \mathcal{P}^2$ is $(x, y) \mapsto (x, y, \zeta)$ where $\mathbf{x} = (x, y)$ is the inhomogeneous image point. The constant ζ is chosen to elicit the best numerical conditioning of our algorithms, as explained below. In the SAM algorithm its variable is `m3` (the third homogeneous coordinate.

A line in the 2-D projective space is also defined by a triplet of numbers $\mathbf{n} = (n_1, n_2, n_3)$, not all of them equal to zero. The line appears on the image plane as $n_1 x + n_2 y + n_3 = 0$. If $n_1 = n_2 = 0$ the line is interpreted to be the ideal line at infinity, l_∞, defined by the set $\mathcal{P}^2 \backslash \mathcal{R}^2$. The points $\mathbf{x} = (x, y, 0)$ that lie on this line are the points at infinity. It is important to draw a distinction between noise free or perfect quantities and their noisy counterparts. Underlined quantities $\underline{x}$ indicate the perfect or noise free quantities, to distinguish them from $x = \underline{x} + \delta x$, the value corrupted by noise. In the case of parameters, underlined quantities, $\underline{\mathbf{f}}$, indicate the true value, to distinguish them from their estimates, $\mathbf{f}$.

An image-image homography, projectivity or collineation is a linear one-to-one mapping that takes all the points/lines in $\mathcal{P}^2$ to the set of all the points/lines in $\mathcal{P}^2$ such that:

1. collinear points are mapped to collinear points,
2. concurrent lines are mapped to concurrent lines,
3. incidence is preserved.

It may be represented by a 3×3 nonsingular matrix $\mathbf{H}$ such that

$$\mathbf{x}^{'} = \mathbf{H}\mathbf{x} \tag{4.2}$$

for all image features in the first image denoted $\mathbf{x}$ and second image denoted $\mathbf{x}^{'}$.

Since the 3-D motion of a rigid object relative to a fixed camera is equivalent to the opposite 3-D motion of the camera relative to the object, it is henceforth assumed that a fixed scene is viewed relative to a moving camera. An XYZ-Cartesian camera coordinate system centred on the first camera is established, where the three dimensional coordinates of a world point are (x, y, z) in $\mathcal{R}^3$. Again $\mathcal{R}^3$ is regarded as a subset of projective three space $\mathcal{P}^3$. The embedding $\mathcal{R}^3 \subset \mathcal{P}^3$ is $(x, y, z) \mapsto (x, y, z, 1)$. The set $\mathcal{P}^3 \backslash \mathcal{R}^3$ is the plane at infinity Π_∞, the equation of which is the set of all points $\mathbf{X}$ such that $\mathbf{X} = (X, Y, Z, 0)$.

Features in space are termed *scene* features in contradistinction to those projected into a given view termed *image* features. The projection from $\mathcal{P}^3$ to $\mathcal{P}^2$ is a linear transformation represented by a 3×4 *camera matrix* $\mathbf{P}$.

The 3-D motion of the camera between the capture of two images is specified by the *motion parameters* $\{\mathbf{R}, \mathbf{t}\}$. That is, we regard the new $X'Y'Z'$-camera coordinate system, after the camera has moved, as obtained from the XYZ-camera coordinate system by first rotating the XYZ-coordinate system around its origin $\mathbf{0}$ by rotation matrix $\mathbf{R}$ and then translating it by vector $\mathbf{t}$, where the components of $\mathbf{R}$ and $\mathbf{t}$ are defined with respect to the XYZ-coordinate system. (The camera moves from $\mathbf{0}$ to $\mathbf{t}$ in the XYZ-coordinate system.) Let $\mathbf{x}$ represent the homogeneous coordinates of a point before the camera motion, and $\mathbf{x}'$ represent the homogeneous coordinates of a

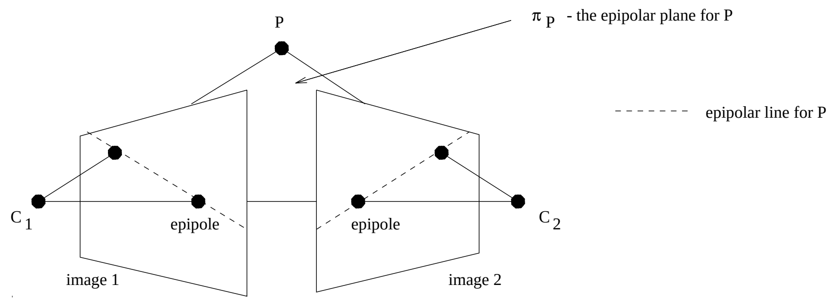


Figure 4.2: *Once the epipolar geometry is established it can be seen that each point in image one defines a line in image two upon which its correspondence must lie.*

point after the camera motion. A simple pin hole model for the camera is assumed, defining an epipolar geometry as shown in Figure 4.2.

The camera model will now be expressed using a general projective transform from $3D$ real projective space, $\mathcal{P}^3$, known as world space, to $2D$ real projective space known as image space. This transformation may be expressed in terms of homogeneous coordinates by a 3×4 matrix $\mathbf{P}$ known as the camera matrix. The camera matrix can be written as the product of $\mathbf{C}$, the 3×3 camera intrinsic parameter matrix giving the internal geometry of the camera, and an extrinsic matrix specifying the external orientation and position of the camera coordinate frame. If it is assumed that there is no shear in the axes, and non-linear effects such as radial distortion have been catered for [1], $\mathbf{C}$ is upper triangular with four degrees of freedom namely principal point, focal length and aspect ratio i.e.

$$\mathbf{C} = \begin{bmatrix} a & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1/f \end{bmatrix} \tag{4.3}$$

where a is the aspect ratio, (p_x, p_y) is the principal point, and f is the focal length. It is assumed (for simplicity of analysis) that $\mathbf{C}$ remains unchanged as the camera moves. If $\mathbf{x} \in \mathcal{P}^2$ is the image of $\mathbf{X} \in \mathcal{P}^3$ then

$$\mathbf{x} = \mathbf{C} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{X} \tag{4.4}$$

or

$$\mathbf{x} = \mathbf{P}\mathbf{X} \tag{4.5}$$

where $\mathbf{P} = \mathbf{C}[\mathbf{R} \mid \mathbf{t}] = [\mathbf{C}\mathbf{R} \mid \mathbf{C}\mathbf{t}]$, the extrinsic parameters (e.g. motion parameters from the origin) are $\{\mathbf{R}, \mathbf{t}\}$, such that

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} . \tag{4.6}$$

Given a vector $\mathbf{t}$ it is convenient to introduce the skew-symmetric matrix:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{4.7}$$

which allows vector products to be written as:

$$[\mathbf{t}]_\times \mathbf{v} = \mathbf{t} \times \mathbf{v} \tag{4.8}$$
$$\mathbf{v}^\top [\mathbf{t}]_\times = \mathbf{v} \times \mathbf{t}. \tag{4.9}$$

For any non-zero vector, $[\mathbf{t}]_\times$ has rank 2. Furthermore the null space of $[\mathbf{t}]_\times$ is generated by the vector $\mathbf{t}$ i.e. $\mathbf{t}^\top[\mathbf{t}]_\times = [\mathbf{t}]_\times \mathbf{t} = \mathbf{0}$. If $\mathbf{a}$ is any vector annihilated by $[\mathbf{t}]_\times$ then it is a scalar multiple of $\mathbf{t}$.

4.3.1 Function `torr_skew_sym`

A useful, but self explanatory routine to generate skew symmetric matrices:

```
function T = torr_skew_sym(t)

T = [0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0];
```

4.4 The Fundamental Matrix

Within this section the fundamental matrix is defined. It is a key concept when using point correspondences as it encapsulates all the geometric information about the camera positions contained within a set of point correspondences, and is equivalent to the epipolar geometry. The fundamental matrix is an extension of the the essential matrix formulated by Longuet-Higgins [31] to the case where calibration is unknown. The case for algorithms that do not require calibration has been strongly made in [12]. Camera calibration is at best difficult, possibly introducing correlated errors into the system, and at worst it is impossible. The fundamental matrix is now defined.

Theorem 1 *[12, 19] Given an uncalibrated camera let the set of homogeneous image points $\{\mathbf{x}_i\}, i = 1, \ldots n$, be transformed to the set $\{\mathbf{x}'_i\}$ on the image plane by the motion parameters $\{\mathbf{R}, \mathbf{t}\}$ such that $\mathbf{t} \neq \mathbf{0}$. Then there exists a 3×3 matrix $\mathbf{F}$ such that*

$$\mathbf{x}_i'^\top \mathbf{F} \mathbf{x}_i = 0 \tag{4.10}$$

for all i.

Proof. Appendix A.

The fundamental matrix may be written in terms of the intrinsic matrix and the essential matrix

$$\mathbf{F} = \mathbf{C}^{-\top} \mathbf{E} \mathbf{C}^{-1} \tag{4.11}$$

proved in Appendix A. Because $\mathbf{E}$ is rank two [31], the fundamental matrix at most of rank two. This leads to the following lemma.

Lemma 1 *If* $\mathbf{F}$ *is a fundamental matrix corresponding to a pair of image and* $\mathbf{x}$ *is a point in the first image, then* $\mathbf{Fx}$ *is the epipolar line in the second image corresponding to* $\mathbf{x}$.

The epipoles correspond to projections of the directions of translations in the two images, and can be recovered by the left and right nullspace of $\mathbf{F}$. The function `torr_get_right_epipole` is provided to calculate the epipole:

```
    % returns epipole such that Fmat1 * epipole = 0


function  epipole = torr_get_right_epipole(Fmat1,m3)

    [v,d] = eig(Fmat1);

    dd = [d(1,1)^2, d(2,2)^2, d(3,3)^2];
    [Y Index] = min(dd);


    epipole = v(:,Index);
    epipole = epipole * (m3/epipole(3));
    %Fmat1 * epipole
```

From $\mathbf{F}$ and the image correspondences it is straightforward to recover *projective* structure as has been pointed out in [12, 19]:

Theorem 2 ([12, 19]) *Given a set of image correspondences sufficient to determine the fundamental matrix, the corresponding world space coordinates are determined up to a collineation of projective* 3*-space* $\mathcal{P}^3$.

In the rest of the chapter non-robust estimators for recovering the fundamental matrix from corrupted data are reviewed.

4.5 Linear Methods: $\mathbf{O_1}$, $\mathbf{O_2}$

Implicit in Equation (4.10) is the assumption that image coordinates are noise free quantities. However, in real situations recovery of the fundamental matrix becomes non-trivial as the image coordinates are perturbed by noise, leading to inconsistency in the the set of constraints provided by (4.10).

In the ensuing sections previous work on estimation of the fundamental matrix is summarised, describing a linear method, an iterative method and a descent method for estimating $\mathbf{F}$. The choice of the algorithm is critical, as some perform substantially better than others. There are two key points in the estimation of the fundamental matrix. First is the choice of error function to be minimized: this involves consideration of the assumptions to be made about the noise distributions of the data (and, as will be seen later outlier distributions). Within this section linear methods are used to estimate $\mathbf{F}$, which although computational efficient, do not provide a maximum likelihood solution. In Sections 4.7 and 4.8 two different estimators that purport to give maximum likelihood solutions are explained. Second is the choice of parameterization for the fundamental matrix to enforce the constraint that its determinant is zero. It will be seen, in Section 4.8, that there is no ideal parameterization, but that an adequate one can be defined.

Well known results in linear regression are summarised next. Regression is used to study the relationship between parameters, and linear regression deals with the class of relationships that are linear in the parameters. Although bilinear in the image coordinates estimation of $\mathbf{F}$ is linear in the parameters. This can be seen by expanding

$$\underline{\mathbf{x}}_i'^{\top} \mathbf{F} \underline{\mathbf{x}}_i = 0 \tag{4.12}$$

to give

$$f_1 \underline{x}_i' \underline{x}_i + f_2 \underline{x}_i' \underline{y}_i + f_3 \underline{x}_i' \zeta + f_4 \underline{y}_i' \underline{x}_i + f_5 \underline{y}_i' \underline{y}_i + f_6 \underline{y}_i' \zeta + f_7 \underline{x}_i \zeta + f_8 \underline{y}_i \zeta + f_9 \zeta^2 = 0, \tag{4.13}$$

where the terms of the fundamental matrix are

$$\mathbf{F} = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}. \tag{4.14}$$

There are two main approaches to regression: ordinary (or classical) least squares [37] (common in statistics) in which the error is subsumed into one of the variables, and orthogonal regression [17] when there is error in all the variables. For ease of reference classical least squares is referred to as $\mathbf{O_1}$ and orthogonal regression as $\mathbf{O_2}$. The two methods are now outlined and then it is explained how they might be used to recover $\mathbf{F}$ from image correspondences.

4.5.1 Ordinary Least Squares Regression: $\mathbf{O_1}$

Consider the set of n measurement equations

$$a_i = \mathbf{d}_i^{\top} \underline{\mathbf{b}} + \epsilon \qquad\qquad i = 1 \ldots n, \tag{4.15}$$

where a_i is a measured scalar with error, ϵ, which is assumed to be Gaussian with standard deviation σ; $\mathbf{d}_i$ is a known p dimensional vector and $\underline{\mathbf{b}}$ is an unknown p dimensional vector of parameters which is to be recovered. If n is greater then p the set of equations is overdetermined. Noise corruption renders them inconsistent however and the least squares technique is used to find a solution. Let $\mathbf{D}$ be the matrix whose rows are $\mathbf{d}_i^{\top}$. From equation (4.15)

$$\mathbf{a} = \mathbf{D}\underline{\mathbf{b}} + \epsilon, \tag{4.16}$$

A common way of recovering an estimate $\mathbf{b}$ of $\underline{\mathbf{b}}$ is to use the Moore-Penrose [39] pseudo-inverse

$$\mathbf{b} = (\mathbf{D}^{\top}\mathbf{D})^{-1}\mathbf{D}^{\top}\mathbf{a}. \tag{4.17}$$

4.5.2 Orthogonal Least Squares Regression: $\mathbf{O_2}$

In ordinary least squares the error is assumed to be in only one coordinate, whereas in orthogonal least squares it is assumed that all coordinates are measured with error. Consider fitting a hyperplane $\mathbf{f} = (f_1, f_2, \ldots, f_p)$ through a set of n points in $\mathcal{R}^p$ with coordinates $\mathbf{z}_i = (z_{i_1}, z_{i_2}, \ldots, z_{i_p})$, with the centroid of the data taken as origin[1] (following [37]).

Assuming that the noise is Gaussian and that the elements of $\mathbf{z}$ have equal variance (if the points have unequal variance each element may be weighted by its standard deviation) the hyperplane with maximum likelihood, $\mathbf{f}$, is estimated by minimizing the perpendicular sum of Euclidean distances from the points to the plane [37, 30], as seen in Figure 4.3. This is accomplished by minimizing $\sum_{i=1}^{n}(\mathbf{f}^\top \mathbf{z}_i)^2$ subject to

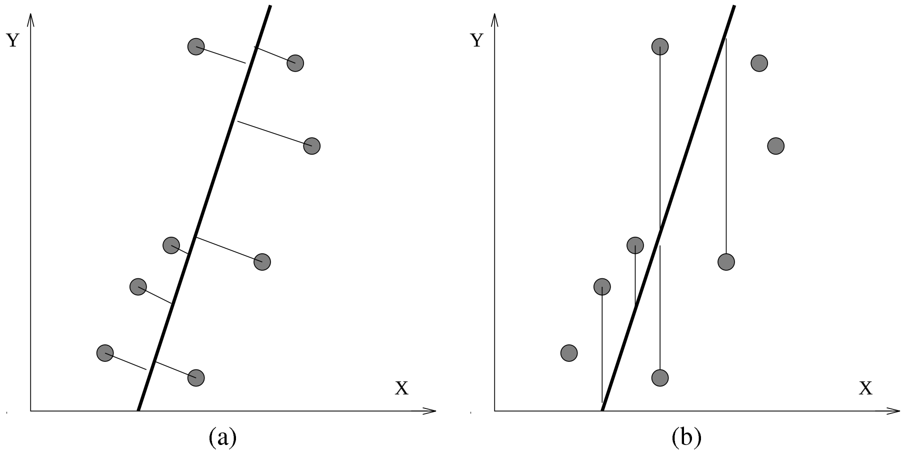


Figure 4.3: *(a) the orthogonal distance and (b) ordinary least squares distance. It can be seen that the latter distance becomes less stable as the angle between the line and y-axis decreases, leading to unstable solutions for the line.*

the constraint $\mathbf{f}^\top \mathbf{f} = 1$ [37, 30, 17]. This constraint ensures that the estimate will be invariant to equiform transformation of the inhomogeneous coordinates (an equiform transformation is a Euclidean transformation combined with a scaling). For example the best fitting line to a 2 dimensional scatter $(x_i, y_i),\ i = 1 \ldots n$ is estimated by minimizing $\sum_{i=1}^{n}(ax_i + by_i + c)^2$ subject to the constraint $a^2 + b^2 = 1$ [37]. This is accomplished by total least squares [17], minimizing

$$\sum_{i=1}^{n}(\mathbf{f}^\top \mathbf{z}_i)^2 \text{ subject to } \mathbf{f}^\top \mathbf{f} = 1 \ . \tag{4.18}$$

Let $\mathbf{Z}$ be the $n \times p$ measurement matrix with rows $\mathbf{z}_i$, and let $\mathbf{M} = \mathbf{Z}^\top \mathbf{Z}$ be the $p \times p$ moment matrix, with eigenvalues, in increasing order, $\lambda_1 \ldots \lambda_p$ and with $\mathbf{u}_1 \ldots \mathbf{u}_p$ the

[1] The best fitting hyperplane passes through the centroid of the data [37].

corresponding eigenvectors forming an orthonormal system. The best fitting hyperplane is given by the eigenvector $\mathbf{u}_1$ corresponding to the minimum eigenvalue λ_1 of the moment matrix. It can be shown that

$$\lambda_1 = \sum_{i=1}^{n} (\mathbf{u}_1^\top \mathbf{z}_i)^2 = \sum_{i=1}^{n} r_i^2 \tag{4.19}$$

which is the sum of squares of residuals r_i (in this case the perpendicular distances to the hyperplane).

In the instance of the fundamental matrix,

$$\mathbf{z} = \left(\begin{array}{ccccccccc} x'_i x_i & x'_i y_i & x'_i \zeta & y'_i x_i & y'_i y_i & y'_i \zeta & x_i \zeta & y_i \zeta & \zeta^2 \end{array} \right) \tag{4.20}$$

and the measurement matrix is

$$\mathbf{Z} = \mathbf{W} \begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 \zeta & y'_1 x_1 & y'_1 y_1 & y'_1 \zeta & x_1 \zeta & y_1 \zeta & \zeta^2 \\ x'_2 x_2 & x'_2 y_2 & x'_2 \zeta & y'_2 x_2 & y'_2 y_2 & y'_2 \zeta & x_2 \zeta & y_2 \zeta & \zeta^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_i x_i & x'_i y_i & x'_i \zeta & y'_i x_i & y'_i y_i & y'_i \zeta & x_i \zeta & y_i \zeta & \zeta^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_n x_n & x'_n y_n & x'_n \zeta & y'_n x_n & y'_n y_n & y'_n \zeta & x_n \zeta & y_n \zeta & \zeta^2 \end{bmatrix} \tag{4.21}$$

where $\mathbf{W}$ is a diagonal matrix of the weights given to each feature correspondence, corresponding to the inverse standard deviation of each error. (This is assumed to be homogeneous at present, in the next section its estimation by iteratively reweighted least squares is explained.) If the variances of the image coordinates are different along the two axes e.g. x-axis is σ_x^2 and y-axis is σ_y^2, the image coordinates are weighted by dividing them by their respective variances.

If $\mathbf{M} = \mathbf{Z}^\top \mathbf{Z}$ is the moment matrix then the estimate $\mathbf{f}$ minimizes $\mathbf{f}^\top \mathbf{M} \mathbf{f}$ subject to $\mathbf{f}^\top \mathbf{J} \mathbf{f} =$ constant, where

$$\mathbf{J} = \text{diag}(1, 1, 1, \ldots, 1, 0)\ , \tag{4.22}$$

is a normalization chosen to realise a solution from the equivalence class of solution with different scalings. This estimate is given by the eigenvector corresponding to the minimum eigenvalue of the centred moment matrix. Centring (*sic*) is a standard statistical technique that involves shifting the coordinate system of the data points so that the centroid lies at the origin. This can be effected by subtracting $\vec{1}\,\overline{z}_j$ from each column of $\mathbf{Z}$, where $\vec{1}$ is an n dimensional vector such that $\vec{1} = (1, 1, 1, \ldots, 1)^\top$ and $\overline{z}_j$ is the mean of that column. The proof is given in Appendix C.

The third homogeneous coordinate ζ is set to be equal to the estimate of the focal length measured in pixels, in order to improve the conditioning of the solution, and to maintain consistency of units. If no estimate is available then $\zeta = 256$, to ensure that it is of the same order of magnitude as the image coordinates. Because $\mathbf{F}$ is bilinear in the image coordinates the orthogonal regression result is not invariant to the choice of ζ. This is a phenomenon well known in conic fitting [6, 41] and stems from the poor normalization given in (4.22). This normalization is inappropriate for the fundamental matrix as it is not invariant to Euclidean transformations of the coordinate system. Table 4.1 gives the variance of the distance of *noise free points to their estimated epipolar lines* varying the value of ζ for 10 sets of n synthetic correspondences. It can be seen

n	25	50	100	200
$\zeta = 1$	8.94	7.60	7.45	4.33
$\zeta = 256$	5.16	3.19	1.37	0.48
$\zeta = 600$	6.02	3.36	1.55	0.89

Table 4.1: *Showing the variance of the distance of* noise free points *to epipolar lines varying ζ for 10 sets of n synthetic correspondences.*

that minimization of the algebraic distance is not invariant to ζ with respect to the distances of features on the image from their epipolar lines. Setting ζ to the same order of magnitude as the image coordinates produces the best result: in this case the images were 512×512 thus we set $\zeta = 256$. These problems do not arise when minimizing the geometric distances on the image plane, which will be described in the next section.

4.5.3 Linear Estimation: `torr_ls`

In Matlab linear estimation by SVD is easy as pie, and is provided by:

```
function [vec, error] = torr_ls(Z)
```

Input: `Z` is the $n \times p$ **Z** matrix.

Output:

- `vec` is the solution vector.
- `error` is $\lambda_1 = \sum r_i^2$.

4.5.4 Linear Estimation of F, `torr_estf`

Most of the functions for estimating **F** have the same parameters

```
function f = torr_estf(x1,y1,x2,y2, no_matches,m3)
```

with parameters defined as in Section 4.1.1.

4.5.5 The Shortcomings of the Linear Methods

Unfortunately both produce inaccurate results. Classical least squares to estimate **F**, as suggested by Tsai *et al.* [58] and Olsen [36], where a chosen variable (the 'observation') is regressed against, is unsuitable as there is a tacit assumption that there are only errors in this variable. The problem is shown in Figure 4.3 where it can be seen that the ordinary least squares distance, measured in the direction of one of the axes, becomes less stable as the line to be estimated tends to parallelism with this axis. That is, the variable chosen to regress against may have zero coefficient, in which case the estimate, $\mathbf{f}$, would be rendered meaningless.

Orthogonal least squares also produces poor results. This is due to three factors. First the residuals r_i that we minimize:

$$r_i = f_1 x'_i x_i + f_2 x'_i y_i + f_3 x'_i \zeta + f_4 y'_i x_i + f_5 y'_i y_i + f_6 y'_i \zeta + f_7 x_i \zeta + f_8 y_i \zeta + f_9 \zeta^2 \quad (4.23)$$

are not Gaussian; secondly the constraint that the determinant of $\mathbf{F}$ should be zero is not enforced; thirdly the choice of normalization in orthogonal regression is inappropriate to $\mathbf{F}$. In the next section *iteratively reweighted least squares* is described in which some compensation is made for the first and third shortcoming [6, 41]. In Section 4.8 we summarise parameterisations that may be used to overcome the second.

4.5.6 Imposing the cubic constraint $\det(\mathbf{F}) = 0$

The second constraint which $\mathbf{F}$ should satisfy is a cubic polynomial in the matrix elements imposing $\det(\mathbf{F}) = 0$. If it is not imposed then the epipolar lines do not all intersect in a single epipole. Assume that we have an estimate of the fundamental matrix, $\hat{\mathbf{F}}$. Typically, when performing only linear estimation to obtain $\hat{\mathbf{F}}$ this constraint is imposed by projecting the linear solution onto the space of Fundamental matrices such that $\det(\mathbf{F}) = 0$, such that the Frobenius norm $||\hat{\mathbf{F}} - \mathbf{F}||$ is a minimum. Let the singular value decomposition [17] of the recovered $\mathbf{F}$ be

$$\mathbf{F} = \mathbf{V}\Lambda\mathbf{U}^\top \, . \quad (4.24)$$

Due to noise $\mathbf{F}$ will have full rank with non zero singular values: $\Lambda = \text{diag}(\lambda^{\frac{1}{2}}{}_1, \lambda^{\frac{1}{2}}{}_2, \lambda^{\frac{1}{2}}{}_3)$. To approximate $\mathbf{F}$ by a rank two matrix, let $\Lambda^+ = \text{diag}(\lambda^{\frac{1}{2}}{}_1, \lambda^{\frac{1}{2}}{}_2, 0)$ then the reduced rank approximation of the fundamental matrix is

$$\mathbf{F} = \mathbf{V}\Lambda^+\mathbf{U}^\top \, . \quad (4.25)$$

This is easily coded in MATLAB via SVD as

```
[U,S,V] = svd(F);
S(3,3) = 0;
F = U*S*V';
f = reshape(F,9,1);
```

A problem is that smaller elements will have a relatively greater perturbation in relation to their size. Thus the method of [22] is adopted. This is not entirely satisfactory as ideally we would like to choose the $\hat{\mathbf{f}}$ which minimizes some Mahalanobis distance $(\hat{\mathbf{f}} - \mathbf{f})^\top \mathbf{M}_f^{-1} (\hat{\mathbf{f}} - \mathbf{f})$, taking into account the covariance of $\mathbf{F}$. Work on this is in progress.

4.5.7 Invariant linear fitting

Next an invariant linear method based on Bookstein is described noting that the fundamental matrix is like a 4D conic. We seek an estimation rule which is *general, simple to compute* and *invariant*. Simplicity suggests we seek a quadratic norm, $\mathbf{f}^\top \mathbf{J} \mathbf{f} =$ constant, on the parameters of $\mathbf{F}$ to enforce the scaling constraint as this will lead to a eigenvector solution. Invariance is to be with respect to Euclidean transformations of both image planes (possibly different transformations to different planes) i.e. if the coordinate system is changed in one or both of the images, then the best fitting $\tilde{\mathbf{F}}$ to the transformed points must be exactly the result of the same transformation(s) applied to the best fitting $\mathbf{F}$ of the original points.

Bookstein[6] suggested an invariant norm for conics under Euclidean transformations. It has been observed that the fundamental matrix is like a conic in the four dimensions of the joint image space $\Re^4$ [49]. Following Bookstein we seek a parametrization of $\mathbf{F}$ invariant to Euclidean transformations in the image planes (which is a subgroup of the Euclidean transformations in the joint image space $\Re^4$). Fortunately the construction of these invariants is a well studied problem [35].

Consider the transformations of the image coordinates $\mathbf{G}$ in image one such that $\mathbf{G}\tilde{\mathbf{x}} = \mathbf{x}$, and image two, $\mathbf{G}'\tilde{\mathbf{x}}' = \mathbf{x}'$, which leads to a transformation on $\mathbf{F}$ such that, $\tilde{\mathbf{F}} = \mathbf{G}'^\top \mathbf{F} \mathbf{G}$ with

$$\mathbf{G} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{G}' = \begin{bmatrix} \mathbf{R}' & \mathbf{t}' \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^\top & d \end{bmatrix}, \quad \tilde{\mathbf{F}} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{b}} \\ \tilde{\mathbf{c}}^\top & \tilde{d} \end{bmatrix}. \tag{4.26}$$

Thus it can be seen that

$$\tilde{\mathbf{F}} = \mathbf{G}'^\top \mathbf{F} \mathbf{G} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{b}} \\ \tilde{\mathbf{c}}^\top & \tilde{d} \end{bmatrix} = \begin{bmatrix} \mathbf{R}'^\top \mathbf{A} \mathbf{R} & \mathbf{R}'^\top \mathbf{A} \mathbf{t} + \mathbf{R}'^\top \mathbf{b} \\ \mathbf{t}'^\top \mathbf{A} \mathbf{R} + \mathbf{c}^\top \mathbf{R} & \mathbf{t}'^\top \mathbf{A} \mathbf{t} + \mathbf{t}'^\top \mathbf{b} + \mathbf{c}^\top \mathbf{t} + d \end{bmatrix}. \tag{4.27}$$

From this it can be seen that the norm cannot be any combination of f_3, f_6, f_7, f_8, f_9 as these can be transformed to arbitrary values by translations of the image coordinates. Unless of course $\mathbf{t}, \mathbf{t}' = 0$, in which case the norm $\sum_{i=1}^{i=9} f_i^2 = 1$ (amongst others) is invariant to rotations of the image plane. Discounting the non generic case, this leaves the elements of the upper left 2×2 submatrix of $\mathbf{F}$ to define the norm. Due to the special nature of rotation matrices it can be immediately seen that

$$\det(\tilde{\mathbf{A}}) = \det(\mathbf{R}'^\top \mathbf{A} \mathbf{R}) = \det(\mathbf{A}), \qquad ||(\tilde{\mathbf{A}})|| = ||(\mathbf{R}'^\top \mathbf{A} \mathbf{R})|| = ||(\mathbf{A})||$$

where $||()||$ corresponds to the Frobenius norm of the matrix. Thus we have the choice of the following norms, the determinant norm $\det(\mathbf{A}) = (f_1 f_5 - f_2 f_4)$, the Frobenius norm $||(\mathbf{A})|| = (f_1^2 + f_2^2 + f_4^2 + f_5^2)^{\frac{1}{2}}$. How many invariants can there be? Referring to [35] the counting argument states: "suppose there is a configuration space $\mathcal{S}$, on which a group G acts, then the number of functionally independent primitive scalar invariants is greater than or equal to $\dim \mathcal{S} - \dim G$ [2]. In this case $\dim \mathbf{A} = 4$ and $\dim(\mathbf{R}, \mathbf{R}') = 2$, thus we would expect at least two invariants [3].

[2] In general equality holds except in the special case of isotopies

[3] If only one rotation was applied to both images i.e. we knew the common orientation of the two images, then we could expect another invariant, which would correspond to trace $\mathbf{A}$

Which of these norms is most appropriate? In order to deduce this another desideratum is introduced; that the norm is positive definite. This is desirable because epipolar geometries for whose $\mathbf{F}$ the norm is zero can never be fitted at all, even if the data lie exactly upon them. Therefore we must say goodbye to the determinant norm $\det(\mathbf{A})$, which excludes all $\mathbf{F}$ for which $\det(\mathbf{A}) = 0$. The square of the Frobenius norm $||(\mathbf{A})||^2 = (f_1^2 + f_2^2 + f_4^2 + f_5^2)$ does not exclude general $\mathbf{F}$, rather it will fit all $\mathbf{F}$ except for data for which a linear or affine fundamental matrix $\mathbf{F}_A$ [35] is more suited;

$$\underline{\mathbf{x}}_i'^{\top} \mathbf{F}_A \underline{\mathbf{x}_i} = 0 \quad \text{where } \mathbf{F}_A = \begin{bmatrix} 0 & 0 & g_1 \\ 0 & 0 & g_2 \\ g_3 & g_4 & g_5 \end{bmatrix}. \tag{4.28}$$

Whether or not $\mathbf{F}_A$ is the more appropriate model can be determined by model selection methods [57]. If it is then an exact eigenvector solution [44] exists for $\mathbf{F}_A$ that minimizes reprojection error which should always be used rather than a more general algorithm for fitting $\mathbf{F}$. Thus we propose to minimize $\mathbf{f}^\top \mathbf{M} \mathbf{f}$ subject to $\mathbf{f}^\top \mathbf{J} \mathbf{f} = \text{constant}$, where $\mathbf{J} = \text{diag}(1, 1, 0, 1, 1, 0, 0, 0, 0)$, is the square of Frobenius normalization.

The square of the Frobenius norm $||(\mathbf{A})||^2 = (f_1^2 + f_2^2 + f_4^2 + f_5^2)$ is also invariant to choice of scale. Without loss of generality consider only the change of scaling in one of the images. If the coordinates in one image are rescaled by k, $(x, y) \to (\tilde{x}, \tilde{y}) = k(x, y)$, let $\mathbf{f}_k$ be the vector of coefficients of the best fitting $\mathbf{F}$ by this norm. The rescaling replaces the moment matrix $\mathbf{M}$ by $\mathbf{D}_k \mathbf{M} \mathbf{D}_k$ where $\mathbf{D}_k = \text{diag}(k, k, k, k, k, k, 1, 1, 1)$. Thus in the new coordinate system the minimization is of $\mathbf{f}^\top \mathbf{M}_k \mathbf{f}$ subject to $\mathbf{f}^\top \mathbf{J} \mathbf{f} = \text{constant}$, where $\mathbf{J} = \text{diag}(1, 1, 0, 1, 1, 0, 0, 0, 0)$ (because $\mathbf{D}_k \mathbf{J} \mathbf{D}_k = k^2 \mathbf{J}$). Since $\mathbf{D}_k$ is not singular the extremum is given by $\mathbf{D}_k^{-1} \mathbf{f}$, where $\mathbf{f}$ is the extremum before rescaling. But $\mathbf{D}_k^{-1} \mathbf{f}$ is simply the transformed version of $\mathbf{f}$. Hence the method presented is invariant under equiform transformations (Euclidean and scaling transformation in the images), and consequently choice of the third projective coordinate ζ.

4.5.8 Imposition of the quadratic constraint $f_1^2 + f_2^2 + f_4^2 + f_5^2 = K$

We wish to minimize $\mathbf{f}^\top \mathbf{M}_k \mathbf{f}$ subject to $\mathbf{f}^\top \mathbf{J} \mathbf{f} = \text{constant}$, where

$$\mathbf{J} = \text{diag}(1, 1, 0, 1, 1, 0, 0, 0, 0). \tag{4.29}$$

There are two eigenvector methods for conducting this minimization. The first is somewhat easier to implement (especially in MATLAB) and involves solving the generalized eigenvector problem:

$$\mathbf{J} \mathbf{f} - \lambda \mathbf{M} \mathbf{f} = 0. \tag{4.30}$$

The second is proposed by Bookstein, and is faster and more stable: First partition $\mathbf{f}$ into two components, $\mathbf{f}_1 = (f_1, f_2, f_4, f_5)$ comprising the four elements of $\mathbf{A}$, the second $\mathbf{f}_2$, comprising the other five elements. Let $\mathbf{M}$ be partitioned correspondingly:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{12}^\top & \mathbf{M}_{22} \end{bmatrix}, \quad \text{then} \quad \mathbf{f}^\top \mathbf{M} \mathbf{f} = \mathbf{f}_1^\top \mathbf{M}_{11} \mathbf{f}_1 + 2 \mathbf{f}_1^\top \mathbf{M}_{12} \mathbf{f}_2 + \mathbf{f}_2^\top \mathbf{M}_{22} \mathbf{f}_2,$$

as $\mathbf{M}$ and its partitions are all symmetric. We must minimize this subject to $\mathbf{f}_1^\top \mathbf{J}_{11} \mathbf{f}_1 =$ constant, where $\mathbf{J}_{11} = \text{diag}(1,1,1,1) = \mathbf{I}$. For any fixed $\mathbf{f}_1$, $\mathbf{f}^\top \mathbf{M} \mathbf{f}$ is minimal when

$$\frac{\partial \mathbf{f}^\top \mathbf{M} \mathbf{f}}{\partial \mathbf{f}_2} = 2\mathbf{M}_{12}^\top \mathbf{f}_1 + 2\mathbf{M}_{22}\mathbf{f}_2 = 0 \tag{4.31}$$

which implies

$$\mathbf{f}_2 = -\mathbf{M}_{22}^{-1}\mathbf{M}_{12}^\top \mathbf{f}_1 \tag{4.32}$$

Then

$$\mathbf{f}^\top \mathbf{M} \mathbf{f} = \mathbf{f}_1^\top (\mathbf{M}_{11} - \mathbf{M}_{12}\mathbf{M}_{22}^{-1}\mathbf{M}_{12}^\top)\mathbf{f}_1 = \mathbf{f}_1^\top \mathbf{Q} \mathbf{f}_1. \tag{4.33}$$

To minimize this for $\mathbf{f}_1^\top \mathbf{J}_{11} \mathbf{f}_1 =$ constant, let λ be a Lagrangian multiplier for the constraint. Then we must set the derivative with respect to $\mathbf{f}_1$ of $\mathbf{f}_1^\top \mathbf{Q} \mathbf{f}_1 - \lambda \mathbf{f}_1^\top \mathbf{f}_1$. This yields

$$\mathbf{Q}\mathbf{f}_1 = \lambda \mathbf{f}_1, \tag{4.34}$$

thus $\mathbf{f}_1$ may be recovered from as the eigenvector solution of (4.34). Note that unless the data lie on lines in the image $\mathbf{M}_{22}$ always has an inverse. As a general note, following in the style of the ellipse specific fitter [15], the other quadratic constraints can be imposed on the elements of $\mathbf{F}$ to restrict them to a certain subspace of fundamental matrices. Indeed it is a simple matter to also add an arbitrary linear constraint to $\mathbf{F}$ into the optimization as explained in [6].

4.5.9 Imposing Linear Constraints on F

The computation remains tractable when we place arbitrary linear constraints on the parameters. If these linear constraints satisfy $\mathbf{L}\mathbf{f} = 0$, then $\mathbf{f}$ is the eigenvector corresponding to the largest eigenvalue of the system

$$(\mathbf{I} - \mathbf{L}(\mathbf{L}^\top \mathbf{M}^{-1}\mathbf{L})^{-}\mathbf{L}^\top)\mathbf{J}\mathbf{f} - \lambda \mathbf{M}\mathbf{f} = 0. \tag{4.35}$$

where $\mathbf{A}^-$ denotes the generalized inverse of an arbitrary square matrix $\mathbf{A}$. A more efficient solution to this generalized eigensystem is given by Golub and Underwood [16], which reduces the problem to a smaller, symmetric system.

Setting linear constraints erodes the available degrees of freedom in various ways, some possibly useful e.g. (1) $f_1 = 0$, $f_5 = 0$: the two cameras are mounted on a lateral stereo rig, with the cameras free only to vary their angle of d_i or e_i)x [7]. (2) skew symmetry $f_2 = -f_4$, $f_3 = -f_7$ and $f_6 = -f_8$, this occurs under a pure translation, generally it would be preferable to fit a two parameter model directly. If the camera has both epipoles in the centre of the image (forward translation and cylcorotation) then $f_3, f_6, f_7, f_8 = 0$.

4.6 Bookstein function: `torr_estf_bookstein`

```
function f = torr_estf_bookstein(x1,y1,x2,y2, no_matches,m3)
```

with parameters defined as in Section 4.1.1.

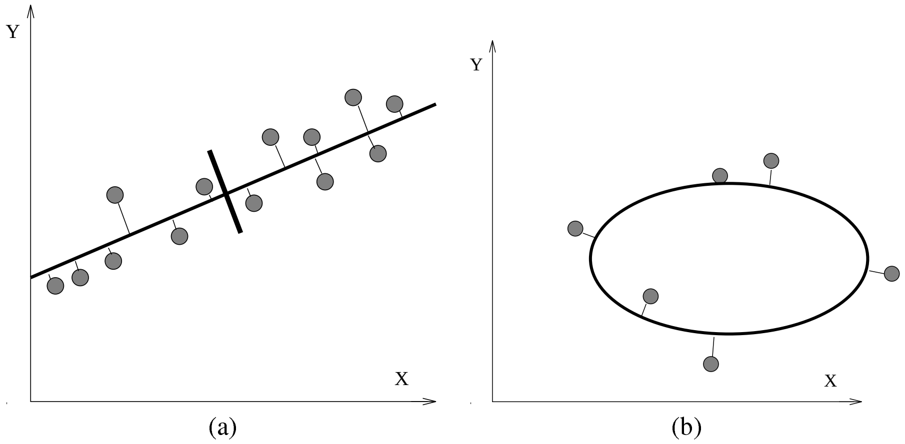


Figure 4.4: *Both (a) (b) show the perpendicular distances to the fit. Under Gaussian assumptions minimizing this gives the maximum likelihood estimate. (a) taking as solution the eigenvector of the moment matrix,* $\mathbf{M}$*, corresponding to the minimum eigenvalue minimizes the sum of squares of distances of the points to the line. The algebraic distance equals the 'geometric distance' to the line. (b) the eigenvector of the moment matrix does not minimize the sum of squares of points to a conic. The algebraic distance does not equal the 'geometric distance' to the line.*

4.7 Iteratively reweighted Least Squares: $\mathrm{S_1}$, $\mathrm{S_2}$

Within this section various error terms that might be minimized are discussed, and then it is shown how iterated least squares might be used to minimize the postulated error terms. The expression given in (4.23) is known as the *algebraic* distance as it does not possess any physical significance (i.e. it does not measure the distance of a feature to the quadric surface in 4D image coordinate space that is $\mathbf{F}$). The variance of r_i is heteroscedastic, meaning that it varies depending on the location of the feature correspondences.

Sampson [41] discovered a similar heteroscedascity in the algebraic residuals when fitting conics. If each point is perturbed by Gaussian noise then minimization of the algebraic distance is suboptimal. It can be shown [30] that the best fitting (maximum likelihood) quadratic curve is such that the sum of squares of perpendicular distances of points from it is a minimum. Furthermore this solution is invariant to Euclidean transformations of the coordinate system. This is clearer in two dimensions, in Figure 4.4 are two examples of minimization. The joins of the different points to the curve are not parallel and may not even be unique. A closed form solution is unobtainable. Sampson proposed using a first order approximation to this distance which can be readily calculated. Weng *et al* [61] extended this to the fundamental matrix as follows.

Theorem 3 ([61]) *Consider the problem of fitting a fundamental matrix to the data points* $\mathbf{z}_i, i = 1 \ldots n$*, defined in (4.20). Let* $\underline{\mathbf{f}}$ *be the exact fundamental matrix, writ-*

ten in vector form, and let $\mathbf{f}$ *be the estimate. If* $\mathbf{f}$ *is computed by the least squares optimization:*

$$\mathbf{f} = min_{\mathbf{f}} \sum_{i=1}^{n} \frac{(\mathbf{f}^\top \mathbf{z}_i)^2}{w_{S_i}} \tag{4.36}$$

where w_{S_i} *is the optimal weight (being the variance of the residual). Then , dropping subscripts for compactness, the optimal weighting is*

$$w_S = \frac{1}{\nabla r} \tag{4.37}$$

where the gradient, ∇r*, is easily computed:*

$$\begin{aligned} \nabla r &= (r_x^2 + r_y^2 + r_{x'}^2 + r_{y'}^2)^{\frac{1}{2}} \\ r_x &= f_1 x' + f_4 y' + f_7 \zeta \\ r_y &= f_2 x' + f_5 y' + f_8 \zeta \\ r_{x'} &= f_1 x + f_2 y + f_3 \zeta \\ r_{y'} &= f_4 x + f_5 y + f_6 \zeta, \end{aligned}$$

where r_x *denotes the partial derivative of* r *(given in Equation (4.23)) with respect to* x.

Proof: Refer to Appendix D.

The optimal weights outlined above require the fundamental matrix to be computed. To minimize (4.36) a method proposed for conics by Sampson [41] is adapted, noting that the fundamental matrix defines a quadratic in the image coordinates. Henceforth this method will be denoted $\mathbf{S_1}$. Sampson proposed an iterative approach, computing an algebraic fit to $\mathbf{f}$ by an eigenvalue method, then reweighting the algebraic distance from each sample point, in this case the image points $\{\mathbf{x}, \mathbf{x}'\}$ by $1/\nabla r^-(\mathbf{x}, \mathbf{x}')$, where ∇r^- is the gradient computed at the previous iteration, using unit weights on the first iteration.

Fit $\mathbf{f}$ by an eigenvalue method, then a new fit is computed as the solution of the weighted moment matrix $\mathbf{M}' = \sum_i \mathbf{z}_i \mathbf{z}_i^\top / w_i$, so that one computes

$$\min_{\mathbf{f}} \sum_{i=1}^{i=n} \frac{r_i^2}{w_i^2} = \min_{\mathbf{f}} \mathbf{f}^\top \mathbf{M}' \mathbf{f}, \quad \text{subject to} \quad \mathbf{f}^\top \mathbf{J} \mathbf{f} = 1.$$

Note that (a) the computation of one or more Sampson refinements is at the same cost as the initial eigenvector solution, and thus does not increase the order of magnitude of the complexity. (b) Any solution computed by a Sampson refinement of $\mathbf{F}$ is still invariant to equiform transformations. Finally for constrained minimization of $\mathbf{F}$ it has been proposed to hold the determinant of the top 2×2 constant [32], within the light of the current analysis it would seem more sensible to impose the constraint that the norm of the top 2×2 is constant, however once exact (as opposed to approximate) geometric distance is minimized then this should not make much difference. Recently Chojnacki et al [9] have reported excellent results on a variation of the Sampson distance for fitting $\mathbf{F}$, they do not consider invariant fitting however.

The optimal weights convert the algebraic distance of each point into the statistical distance in noise space, which is equivalent to the first order approximation of the geometric distance as shown in [41, 38]. The weighting breaks down at the epipole, the numerator and the denominator both approaching zero, indicating that there is less information about correspondences the closer they are to the epipole. In practice to remove unstable constraints all points within a pixel of our estimated epipole are excluded from that iteration of the calculation.

Contention: In the book of Hartley and Zisserman [25] there is a claim that the Sampson distance breaks down after 1 pixel of noise and that a highly non-linear triangulation method is necessary. I would dispute this and invite the discerning reader to try a comparison.

4.7.1 Error function: `torr_errf2`

To calculate the first order error of a set of matches for a given fundamental matrix **f**.

```
function e = torr_errf2(f, nx1,ny1,nx2,ny2, no_matches, m3)
```

Input: input parameters defined as in Section 4.1.1.

Output: `e` is a $n \times 1$ vector of squared errors with ith element $\frac{r_i^2}{w_i^2}$.

4.7.2 Weight function: `torr_grad_f`

`torr_grad_f` Calculates Sampson's weight for a set of matches and a given **f**.

```
function g = torr_grad_f(f, nx1,ny1,nx2,ny2, no_matches, m3)
```

Input: input parameters defined as in Section 4.1.1.

Output: `g` is a $n \times 1$ vector of squared errors with ith element w_i^2.

4.7.3 Sampson function: `torr_estf_bookstein_sampson`

```
function f = torr_estf_bookstein_sampson(x1,y1,x2,y2, no_matches,m3)
```

with parameters defined as in Section 4.1.1.

4.8 Parameterised Descent Methods: $\mathbf{N_1}$-$\mathbf{N_3}$

This section describes minimization whilst imposing the non-linear constraint that the determinant is zero. The iterative method given above has the deficiency that it does not directly enforce the constraint that the determinant of the fundamental matrix must be zero. Rather a procrustean device is used to convert the solution from one that is invalid to one that is valid. Another way of overcoming this deficiency is to perform a constrained non-linear minimization. As a general rule constrained non-linear minimizations are tricky and it is usually better to use a parameterisation that implicitly includes the constraint. For instance, $\mathbf{F}$ might parameterized as

$$\mathbf{F} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \\ \omega_4 & \omega_5 & \omega_6 \\ \omega_7\omega_1 + \omega_8\omega_4 & \omega_7\omega_2 + \omega_8\omega_5 & \omega_7\omega_3 + \omega_8\omega_6 \end{bmatrix}, \tag{4.38}$$

providing that the third row is not zero (if it is then it is a trivial matter to alter the parameterization). Luong *et al* [32] suggested a parameterization of $\mathbf{F}$ in terms of the non-homogeneous coordinates of the epipoles $\mathbf{e} = (e_1, e_2)$, $\mathbf{e}' = (e'_1, e'_2)$ and three of the four coefficients of the homography between the epipolar lines:

$$\mathbf{F} = \begin{bmatrix} b & a & -ae_2 - be_1 \\ -d & -c & ce_2 + de_1 \\ de'_2 - be'_1 & ce'_2 - ae'_1 & ae_2e'_1 + be_1e'_1 - ce_2e'_2 - de'_2e_1 \end{bmatrix} \tag{4.39}$$

where $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is the homography between the epipolar lines. As the homography has only three degrees of freedom, its matrix's determinant is constrained to be 1, giving $ad - bc = 1$, from which it can be seen that the fundamental matrix defined in Equation (4.39) has determinant zero. Given an estimate of $\mathbf{F}$ it is converted to the epipolar parameterization as follows:

$$\begin{aligned} a &= F_{12} \\ b &= F_{11} \\ c &= -F_{22} \\ d &= \frac{1 + bc}{a} \\ g &= F_{22}F_{11} - F_{12}F_{21} \\ e_1 &= \frac{F_{23}F_{12} - F_{22}F_{13}}{g} \\ e_2 &= \frac{F_{13}F_{21} - F_{11}F_{23}}{g} \\ e'_1 &= \frac{F_{32}F_{21} - F_{22}F_{31}}{g} \\ e'_2 &= \frac{F_{31}F_{12} - F_{11}F_{32}}{g}. \end{aligned}$$

The non-linear minimization is performed (using Powell's technique [59]) on the seven independent parameters: $a, b, c, e_1, e_2, e'_1, e'_2$ and once a minimum is attained the $\mathbf{F}$ matrix may be recovered from:

$$\begin{aligned} F_{11} &= b \\ F_{12} &= a \end{aligned}$$

$$\begin{aligned}
F_{13} &= -ae_2 - be_1 \\
F_{21} &= -d \\
F_{22} &= ce_2 + de_1 \\
F_{23} &= ce_2 + de_1 \\
F_{31} &= de'_2 - be'_1 \\
F_{32} &= ce'_2 - ae'_1 \\
F_{33} &= -ce'_2e_2 - de'_2e_1 + ae_2e'_1 + be_1e'_1.
\end{aligned}$$

Luong's parameterization breaks down when the epipoles are at infinity, because at infinity $ad - bc = 0$. To overcome this the coordinate system is changed. Prior to reparameterization of $\mathbf{F}$ the epipoles in image one and two are calculated from the nullspace of $\mathbf{F}$ and $\mathbf{F}^\top$ by observing:

$$\begin{aligned}
\mathbf{Fe} &= \mathbf{0} \\
\mathbf{F}^\top\mathbf{e}^{'} &= \mathbf{0}.
\end{aligned}$$

If the epipole is on or near the line at infinity all the coordinates in each image are transformed by an projective transformation that takes the epipoles well away from the line at infinity. Given the epipole is at $\mathbf{e} = (e_1, e_2, 0)^\top$ a rotation $\mathbf{A}$ about the axis $(e_2, -e_1, 0)$ by $\frac{\pi}{2}$ degrees is used,

$$\mathbf{A} = \frac{1}{2}\begin{bmatrix} 1 + e_2^2 - e_1^2 & -2e_1e_2 & -2e_1 \\ -2e_1e_2 & 1 - e_2^2 + e_1^2 & -2e_2 \\ 2e_1 & 2e_2 & 1 - e_2^2 - e_1^2 \end{bmatrix}, \tag{4.40}$$

which takes $\mathbf{e}^\top \mapsto (0, 0, 1)^\top$. If the image coordinates are transformed in image 1 by $\mathbf{A}$ and in image 2 by $\mathbf{A}'$ then

$$(\mathbf{A}'\mathbf{x}')^\top\mathbf{F}'(\mathbf{Ax}) = 0 \tag{4.41}$$

where $\mathbf{F}' = \mathbf{A}'^\top\mathbf{F}\mathbf{A}^{-1}$. The minimization on $\mathbf{F}'$ is conducted in the transformed coordinate system.

The non-linear minimization using the parameterization in (4.39) is referred to as $\mathbf{N_1}$, and using the parameterization in (4.38) is referred to as $\mathbf{N_2}$. Method $\mathbf{N_3}$ is a gradient descent minimization on the epipolar distance, without enforcing $|\mathbf{F}| = 0$. It is provided as a benchmark with which to compare $\mathbf{N_1}$ and $\mathbf{N_2}$.

Gradient descent methods are vulnerable to local minima if the starting point is not near the actual solution. The starting point we use is the output of the linear algorithm $\mathbf{O_2}$ described above, followed by the procrustean conversion of the estimate to a rank two matrix using SVD.

4.9 Constrained Estimation of F

Matlab provides a generic constrained estimator for $\mathbf{F}$ `fmincon`. This allows for the minimization of functions subject to constraints on the parameters, thus minimization of $\mathbf{F}$ can be undertaken subject to $|\mathbf{F}| = 0$ and the Frobenius norm of the top $2 \times 2 = 1$.

4.9.1 Constrained Estimator function: `torr_nonlinf_mincon2x2`

```
function f = torr_nonlinf_mincon2x2(f_init, nx1,ny1,nx2,
ny2, no_matches, m3)
```

with most parameters defined as in Section 4.1.1, except `f_init` which is an initial estimate of **f** furnished by some other estimator. The initial estimate should satisfy $|\mathbf{F}| = 0$ otherwise instability may result. Note `torr_nonlcon_f2x2` is the call back function for `torr_nonlinf_mincon2x2` that expresses the two constraints:

```
function [c,ceq] = torr_nonlcon_f2x2(f, nx1,ny1,nx2,ny2, m3)
%c = ...      % Compute nonlinear inequalities at f.
%ceq = ...    % Compute nonlinear equalities at f.

%g(1) = norm(f) -1.0;
%g(2) = f(1) * (f(5) * f(9) - f(6) * f(8)) - f(2) * (f(4) * f(9) -
f(6) * f(7)) + f(3) * (f(4) * f(8) - f(5) * f(7));
c = [];
%what norm should we use!
ceq(1) = sqrt(f(1)^2 + f(2)^2 + f(4)^2 + f(5)^2)- 1;
%ceq(1)= norm(f) -1.0;
ceq(2) = f(1) * (f(5) * f(9) - f(6) * f(8)) - f(2) * (f(4) * f(9) -
f(6) * f(7)) + f(3) * (f(4) * f(8) - f(5) * f(7));
```

4.10 Thoughts on Testing Estimators

Within this section tests a testing methodology is described for evaluating how good a method for fitting **F** is. For synthetic data, where the ground truth is known, an empirical measure of the goodness of fit is achieved by calculating the reprojection error of the *actual* noise free projections of the synthetic world points to **F** provided by each estimator. Traditionally the goodness of fit has been assessed by seeing how well the parameters fit the *observed* data. But we point out that this is the wrong criterion as the aim is to find the set of parameters that best fit the (unknown) *true data*. The parameters of the fundamental matrix themselves are not of primary importance, rather it is the structure of the corresponding epipolar geometry. Consequently it makes little sense to compare two solutions by directly comparing corresponding parameters in their fundamental matrices; one must rather compare the the difference in the associated epipolar geometry weighted by the density of the given matching points. This error metric is the first order approximation of the reprojection error of the noise free points to **F**: $\mathcal{E}_1 = \sum_{i=1}^{n}(\underline{w}_i \mathbf{f}^\top \underline{\mathbf{z}_i})^2$. The second statistic $\mathcal{E}_2$ is the average distance in pixels from the true epipole in each image to that yielded by the estimate of **F**.

4.10.1 Test script: `torr_evalFsc`

This script allows the user to estimate **F** with a method determined by the variable `method` and display $\mathcal{E}_1$ and $\mathcal{E}_2$. The code is simple:

```
%profile on

m3 = 256;
%user chooses the appropriate method
method = 3;
total_sse = 0;
epipole_distance = 0;
%
% randn('state',0)
% rand('state',0)
no_tests = 1;
for(i = 1:no_tests)
%     [true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2] = ...
%         torr_gen_2view_matches(foc, no_matches, noise_sigma, trans-
lation_mult, translation_adder, ...
%         rotation_multplier, min_Z,Z_RAN,m3);
    [true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,true_R,true_t, true_E] =
torr_gen_2view_matches;
 true_epipole = torr_get_right_epipole(true_F,m3);

 no_matches = length(nx1);

 matches = [nx1,ny1,nx2,ny2];
 perfect_matches = [x1,y1,x2,y2];
 set_rank2 = 0;

 %first estimate F
 [f, e1, n_inliers,inlier_index,nF] =
torr_estimateF( matches, m3, [], method, set_rank2);

 %check errors vs the true (noise free) points
 groundtrutherrors = torr_errf2(f, x1,y1,x2,y2, no_matches, m3);
 total_sse = total_sse + sum(groundtrutherrors);


 %calc noisy epipole
 noisy_epipole = torr_get_right_epipole(nF,m3);
 epipole_distance = epipole_distance + sqrt(norm(true_epipole -noisy_epipole));
end

disp(' the average sse vs the noise free points is')
total_sse/no_tests
%profile off

disp('RMS distance between true and estimated right epipole is')
epipole_distance
```

Chapter 5

Robust Estimation of F

Robust parameter estimation is essential to the realization of many computer vision algorithms. This chapter examines robust parameter estimators with specific emphasis on their relation to the fundamental matrix. A fully automatic approach requires not only an understanding of geometry, on which a wide range of work has been carried out in the past, but also the ability to deal with incorrect data (such as mismatched features) which will inevitably arise in a real system. It is often assumed that a standard least squares framework is sufficient to deal with *outliers* (data that does not agree with a postulated model). However, outliers can so distort a fitting process that the final result is arbitrary. We eschew the non-robust approach as unworkable except for carefully controlled scenes and examine in depth the robust approach.

5.1 Introduction

Geometric theory is the bedrock of computer vision as it provides models of the observed world. Computer vision algorithms generate interpretations (being instantiations of these geometric models) of the observed data. These algorithms are typically cast in terms of the minimization of an appropriate cost function, and in many cases this cost function is cast as the sum of squares of a set of residuals (the least squares solution). This is usually for one of two reasons. First, least squares is the maximum likelihood estimator when the errors are Gaussian. Secondly, the least squares principle is usually adopted as the fundamental postulate on computational grounds as speedy and stable algorithms exist to compute the solution in many cases.

When the data are contaminated by outliers, however, the first justification no longer holds, and the second justification becomes irrelevant as the solution provided may be far from the true one. Outliers which are inevitably included in the initial fit can so distort the fitting process that the resulting fit can be arbitrary. This is illustrated in Figure 5.1, taken from Fischler and Bolles [14], which shows the results using two estimators on a data set. The least squares estimator provides an erroneous solution, fit 1, whereas the best robust estimator, which will be described, gives a solution, fit 2, that well fits the six inliers. The coordinates of the data set are given in Table 5.1. This data

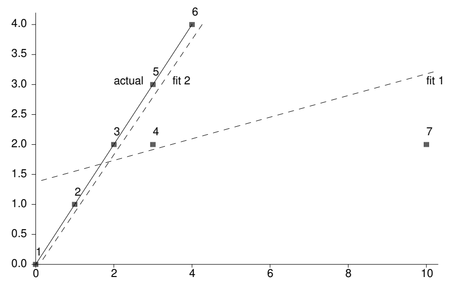


Figure 5.1: *Six of the seven points are valid data and can be fitted by the solid line under the assumption that no valid datum deviates from this line by more than* 0.8 *units. Using least squares and a "throwing out the worst residual heuristic" the points 6, 5 and 1, in table 5.1 are signified as being outliers. The dashed lines indicate the successive fits of the outlier algorithm.*

point	x	y	$x - \overline{x}$	$y - \overline{y}$
1	0.00	0.00	-3.28	-2.00
2	1.00	1.00	-2.28	-1.00
3	2.00	2.00	-1.28	0.00
4	3.00	2.00	-0.28	0.00
5	3.00	3.00	-0.28	1.00
6	4.00	4.00	0.71	2.00
7	10.00	2.00	6.71	0.00

Table 5.1: *The data set presented by Fischler and Bolles [14].*

set demonstrates the failings of naïve least squares and heuristic attempts to remove outliers. The data set possesses only one gross outlier (point 7) but this utterly distorts the estimated parameters; furthermore an attempt to remove this outlier by discarding the point with largest residual fails, even if the process is repeated four times, at each time re-evaluating the result after each removal (discarding over half of the valid data).

Much work has been done already on detecting outliers in the context of non-orthogonal regression (reviewed in [8]), method $\mathbf{O_1}$ in Section 4.5. Unfortunately, there is in this method a tacit assumption that all the error is concentrated in the dependent variable. In many engineering situations this does not occur. In some cases linear orthogonal regression, method $\mathbf{O_2}$, may be used, where the sum of squares of the algebraic distances are minimized. Little work has been done on outlier detection for orthogonal regression—the work of Shapiro and Brady [45] on hyperplanes is an exception—and no work for non-linear regression has been carried out. The estimation of the fundamental matrix falls into the last category. Within the rest of this chapter a number of robust methods are evaluated, with specific emphasis on their relative efficiency and breakdown points. The relative efficiency of a regression method is defined as the ratio between the lowest achievable variance for the estimated parameters (the

Cramér-Rao bound [30]) and the actual variance provided by the given method. An empirical measure of this is achieved by calculating the distance of the *actual* noise free projections of the synthetic world points to their epipolar lines for each estimator.

The breakdown point of an estimator is the smallest proportion of outliers that may force the value of the estimate outside an arbitrary range. For a normal least squares estimator one outlier is sufficient to alter arbitrarily the result, therefore it has a breakdown point of $1/n$ where n is the number of points in the set. An indication of the breakdown point is gained by conducting the tests with varying proportions of outliers.

5.2 Random Sampling Algorithms

An early example of a robust algorithm is the random sample consensus paradigm (RANSAC) [14]. Given that a large proportion the data may be useless the approach is the opposite to conventional smoothing techniques. Rather than using as much data as is possible to obtain an initial solution and then attempting to identify outliers, as small a subset of the data as is feasible to estimate the parameters is used (e.g. two point subsets for a line, seven correspondences for a fundamental matrix), and this process is repeated enough times on different subsets to ensure that there is a 95% chance that one of the subsets will contain only good data points. The best solution is that which maximizes the number of points whose residual is below a threshold. Once outliers are removed the set of points identified as non-outliers may be combined to give a final solution.

Use of the RANSAC method to estimate the epipolar geometry was first reported in Torr and Murray [52]. To estimate the fundamental matrix seven points are selected to form the data matrix $\mathbf{Z}$:

$$\mathbf{Z} = \mathbf{W} \begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1'\zeta & y_1'x_1 & y_1'y_1 & y_1'\zeta & x_1\zeta & y_1\zeta & \zeta^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_i'x_i & x_i'y_i & x_i'\zeta & y_i'x_i & y_i'y_i & y_i'\zeta & x_i\zeta & y_i\zeta & \zeta^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_7'x_7 & x_7'y_7 & x_7'\zeta & y_7'x_7 & y_7'y_7 & y_7'\zeta & x_7\zeta & y_7\zeta & \zeta^2 \end{bmatrix}. \tag{5.1}$$

The null space of the moment matrix $\mathbf{M} = \mathbf{Z}^\top\mathbf{Z}$ is dimension two, barring degeneracy ($\mathbf{Z}$ is 7×9). It defines a one parameter family of exact fits to the 7 correspondences: $\alpha\mathbf{F}_1 + (1-\alpha)\mathbf{F}_2$. Introducing the constraint $\det|\mathbf{F}| = 0$ leads to a cubic in α:

$$\det|\alpha\mathbf{F}_1 + (1-\alpha)\mathbf{F}_2| = 0 \tag{5.2}$$

Solving for α gives 1 or 3 solutions are obtained. The total number of consistent features for each solution is recorded, as outlined in Table 5.2.

Intuitively, it might be thought that the selection of 7 points and the solution of a cubic would lead to a very ill-conditioned result, and that the utilization of more than 7 points might improve matters, which was the approach followed in [11], where 8 points were used in each sample and $\mathbf{F}$ solved for by linear methods. This is not the case: in

Features	Fraction of Contaminated Data, ϵ						
p	5%	10 %	20 %	25 %	30 %	40 %	50 %
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95
6	3	4	10	16	24	63	191
7	3	5	13	21	35	106	382
8	3	6	17	29	51	177	766

Table 5.2: *The number m of subsamples required to ensure $\Upsilon \geq 0.95$ for given p and ϵ, where Υ is the probability that all the data points selected in one subsample are non-outliers.*

Luong *et al.* [32] it is shown that linear methods produce a biased solution, and, furthermore selecting more points exponentially increases the chance that the set contains an outlier. To illustrate this the performance of the two approaches was monitored for 100 trials each of 200 synthetically generated matches with added noise, using 7 and 8 point samplings. The average variance of the distance of point to the estimated epipolar lines was 1.4 pixels for 7 point sampling and 12.7 pixels for 8 point sampling!

In order to determine whether or not a feature pair is consistent with a given fundamental matrix, the epipolar distance of each correspondence in the image is compared to a threshold, which will be described later in Section 5.5.

The number of subsamples required is now calculated. Fischler and Bolles [14] and Rousseeuw [40] proposed slightly different means of calculation, but both give broadly similar numbers. Here, the method of calculation given in [40] is used. Ideally every possible subsample would be considered, but this is usually computationally infeasible, and so m the number of samples, is chosen sufficiently high to give a probability Υ in excess of 95% that a good subsample is selected. The expression for this probability Υ is

$$\Upsilon = 1 - (1 - (1 - \epsilon)^p)^m, \tag{5.3}$$

where ϵ is the fraction of contaminated data, and p the number of features in each sample. Table 5.2 gives some sample values of the number m of subsamples required to ensure $\Upsilon \geq 0.95$ for given p and ϵ. It can be seen from this that, far from being computationally prohibitive, the robust algorithm may require less repetitions than there are outliers, as it is not directly linked to the number but only the proportion of outliers. It can also been seen that the smaller the data set needed to instantiate a model, the less samples are required for a given level of confidence. If the fraction of data that is contaminated is unknown, as is usual, an educated worst case estimate of the level of contamination must be made in order to determine the number of samples to be taken, this can be updated as larger consistent sets are found e.g. if the worst guess is 50% and a set with 80% inliers is discovered, then ϵ could be reduced from 50% to 20%.

In general if the seven correspondence sample has an insufficient spread of disparities then the estimate of **F** obtained from that sample might not be unique. This is an example of degeneracy. Consider the seven correspondences shown in Figure 5.2.

1. Repeat for m samplings as determined in Table 5.2:
   (a) Select a random sample of the minimum number of data points to make a parameter estimate $\mathbf{F}$.
   (b) Calculate the distance d_i of each feature to the epipolar lines of $\mathbf{F}$.
   (c) In the case of the RANSAC estimator calculate the number of inliers consistent with $\mathbf{F}$, using the method prescribed in Section 5.5. (In the case of LMS calculate the median error).
2. Select the best solution i.e. the biggest consistent data set. In the case of ties select the solution which has the lowest standard deviation of inlying residuals.
3. Re-estimate the parameters using all the data that has been identified as consistent, a different more computationally expensive estimator may be used at this point e.g. Powell's method.

Table 5.3: *A brief summary of random sampling algorithm*

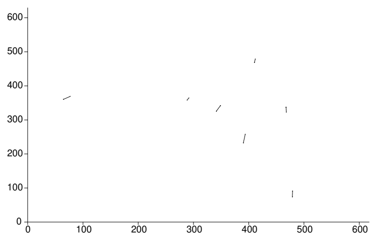


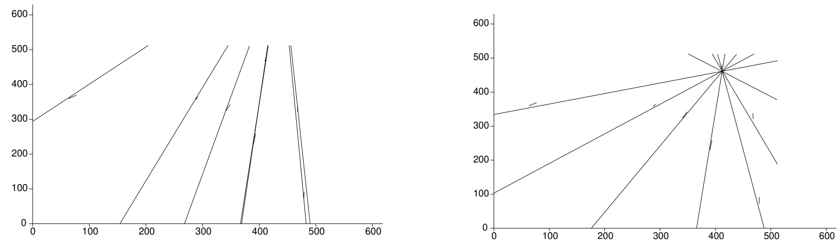
Figure 5.2: *A typical set of seven points selected during RANSAC.*

Two epipolar geometries that fit this data are shown, for one view, in Figure 5.3. Figure 5.3 (a) is the veridical epipolar geometry indigenous to the camera motion, (b) the estimated solution consistent with the cubic given in Equation (5.2). Clearly the result estimated from this sample will not have many other consistent correspondences that conform to the underlying motion. It is desirable to devise a scheme to determine whether any subsample is degenerate. How might the detection of degeneracy be incorporated into RANSAC? This question will be dealt with in some other work of mine [50].

5.2.1 Seven Point Function: `torr_F_constrained_fit`

The Matlab implementation of the seven point algorithm is given by

```
function [no_F, big_result] = torr_F_constrained_fit(x1,y1,x2,y2,m3)
```



(a) (b)

Figure 5.3: *Two epipolar geometries that exactly fit the data in Figure 5.2. (a) is the true epipolar geometry. (b) is a spurious epipolar geometry that fits the data Clearly the data set is degenerate as it admits to multiple solutions.*

Input: defined as in Section 4.1.1.

Output:

- `no_F` gives the number of real solution
- `big_result` a `no_F` $\times 9$ array of the solutions $\mathbf{f}$.

5.2.2 MAPSAC **Function:** `torr_mapsac_F`

```
function [f,f_sq_errors, n_inliers,inlier_index]
= torr_mapsac_F(x1,y1,x2,y2, no_matches, m3, no_samp, T)
```

Input:

- `x1,y1,x2,y2, no_matches, m3` defined as in Section 4.1.1.
- `no_samp` the maximum number of samples to be drawn.
- `T`, threshold for inliers.

Output:

- `f` defined as in Section 4.1.1..
- `f_sq_errors` the squared error for each match.
- `n_inliers` the number of inliers!
- `inlier_index` 1 if a match is inlying, 0 otherwise.

5.2.3 Least Median Estimator

Surprisingly, RANSAC originated in the field of computer vision and it was a few years later that a similar highly robust estimator was developed in the field of statistics, namely Rousseeuw's least median square (LMS) estimator [40]. The algorithms differ slightly in that the solution giving least median is selected as the estimate in [40]. Both estimators perform very well, RANSAC gives better performance for up to 35% above which LMS gives a marginally better performance.

5.3 Maximum Likelihood Estimation in the Presence of Outliers: MAPSAC

MAPSAC stands for Maximum a posteriori sample consensus, and is a generalization of my previous algorithm, MLESAC to the Bayesian case, although when there is a uniform prior on the parameters the two are the same. Within this section the maximum likelihood formulation is given for computing any of the multiple view relations, as the prior on $\mathbf{F}$ is not overly important in the case of estimation, and the MLE derivation is much simpler to follow than the MAP derivation e.g. [50, 62].

In the following we make the assumption, without loss of generality, that the noise in the two images is Gaussian on each image coordinate with zero mean and uniform standard deviation σ. Thus given a true correspondence the probability density function of the noise perturbed data is

$$\Pr(\mathbf{D}|\mathbf{M}) = \prod_{i=1\ldots n} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\left(\sum_{j=1,2}(\underline{x}_i^j - x_i^j)^2 + (\underline{y}_i^j - y_i^j)^2\right)/(2\underline{\sigma}^2)} \quad , \tag{5.4}$$

where n is the number of correspondences and $\mathbf{M}$ is the appropriate 2 view relation, e.g. the fundamental matrix or projectivity, and $\mathbf{D}$ is the set of matches. The negative log likelihood of all the correspondences $\mathbf{x}_i^{1,2}$, $i = 1..n$:

$$-\sum_{i=1\ldots n} \log(\Pr(\mathbf{x}_i^{1,2}|\mathbf{M},\sigma)) = \sum_{i=1\ldots n}\sum_{j=1,2} \left((\underline{x}_i^j - x_i^j)^2 + (\underline{y}_i^j - y_i^j)^2\right), \tag{5.5}$$

discounting the constant term. Observing the data, we infer that the true relation $\mathbf{M}$ minimizes this log likelihood. This inference is called "Maximum Likelihood Estimation".

Given two views with associated relation for each correspondence $\mathbf{x}^{1,2}$ the task becomes that of finding the maximum likelihood estimate, $\hat{\mathbf{x}}^{1,2}$ of the true position $\underline{\mathbf{x}}^{1,2}$, such that $\hat{\mathbf{x}}^{1,2}$ satisfies the relation and minimizes $\sum_{j=1,2}\left(\hat{x}_i^j - x_i^j\right)^2 + \left(\hat{y}_i^j - y_i^j\right)^2$. The MLE error e_i for the ith point is then

$$e_i^2 = \sum_{j=1,2} \left(\hat{x}_i^j - x_i^j\right)^2 + \left(\hat{y}_i^j - y_i^j\right)^2 \tag{5.6}$$

Thus $\sum_{i=1\ldots n} e_i^2$ provides the error function for the point data, and $\mathbf{M}$ for which $\sum_i e_i^2$ is a minimum is the maximum likelihood estimate of the relation (fundamental matrix, or projectivity). Hartley and Sturm [23] show how e, $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ may be found as the solution of a degree 6 polynomial. A computationally efficient first order approximation to these is given in Torr *et al.* [54, 55, 56].

The above derivation assumes that the errors are Gaussian, often however features are mismatched and the error on $\mathbf{m}$ is not Gaussian. Thus the error is modeled as a mixture model of Gaussian and uniform distribution:-

$$\Pr(e) = \left(\gamma \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{e^2}{2\sigma^2}) + (1-\gamma)\frac{1}{v}\right) \tag{5.7}$$

where γ is the mixing parameter and v is just a constant, σ is the standard deviation of the error on each coordinate. To correctly determine γ and v entails some knowledge of the outlier distribution; here it is assumed that the outlier distribution is uniform, with $-\frac{v}{2}..+\frac{v}{2}$ being the pixel range within which outliers are expected to fall (for feature matching this is dictated by the size of the search window for matches). Therefore the error minimized is the negative log likelihood:

$$-L = -\sum_i \log\left(\gamma\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\left(\sum_{j=1,2}(\underline{x}_i^j - x_i^j)^2 + (\underline{y}_i^j - y_i^j)^2\right)/(2\sigma^2) + (1-\gamma)\frac{1}{v}\right)\right) . \tag{5.8}$$

Given a suitable initial estimate there are several ways to estimate the parameters of the mixture model, most prominent being the EM algorithm [10, 34], but gradient descent methods could also be used. Because of the presence of outliers in the data the standard method of least squares estimation is often not suitable as an initial estimate, and it is better to use a robust estimate such as RANSAC which is described in the next section.

5.4 The robust estimators: MAPSAC

The RANSAC algorithm has proven very successful for robust estimation, but having defined the robust negative log likelihood function $-L$ as the quantity to be minimized it becomes apparent that RANSAC can be improved on.

One of the problems with RANSAC is that if the threshold T for considering inliers is set too high then the robust estimate can be very poor. Consideration of RANSAC shows that in effect it finds the minimum of a cost function defined as

$$C = \sum_i \rho\left(e_i^2\right) \tag{5.9}$$

where $\rho()$ is

$$\rho(e^2) = \begin{cases} 0 & e^2 < T^2 \\ \text{constant} & e^2 \geq T^2 \end{cases} . \tag{5.10}$$

In other words inliers score nothing and each outlier scores a constant penalty. Thus the higher T^2 is the more solutions with equal values of C tending to poor estimation e.g. if T were sufficiently large then all solutions would have the same cost as all the matches would be inliers. In Torr and Zisserman [55] it was shown that at no extra

1. Detect corner features using the Harris corner detector [18].
2. Putative matching of corners over the two images using proximity and cross correlation.
3. Repeat until `no_samp` samples have been taken or "jump out" occurs as described in Section 5.2.
   (a) Select a random sample of the minimum number of correspondences $S_m = \{\mathbf{x}_i^{1,2}\}$.
   (b) Estimate the image relation $\mathbf{M}$ consistent with this minimal set using the methods described in Section 5.2.
   (c) Calculate the error e_i for each datum.
   (d) Calculate C_2.
4. Select the best solution over all the samples i.e. that with lowest C_2. Store the set of correspondences S_m that gave this solution.
5. Minimize robust cost function over all correspondences, using iterative non-linear methods, as described in Section 4.9.

Table 5.4: *A brief summary of all the stages of estimation*

cost this undesirable situation can be remedied. Rather than minimizing C a new cost function can be minimized

$$C_2 = \sum_i \rho_2\left(e_i^2\right) \tag{5.11}$$

where the robust error term ρ_2 is

$$\rho_2(e^2) = \begin{cases} e^2 & e^2 < T^2 \\ T^2 & e^2 \geq T^2 \end{cases}. \tag{5.12}$$

This is a simple, redescending M-estimator [26]. It can be seen that outliers are still given a fixed penalty but now inliers are scored on how well they fit the data. We set $T = 1.96\sigma$ so that Gaussian inliers are only incorrectly rejected five percent of the time. The implementation of this new method (one incarnation of the general MAPSAC principle) yields a modest to hefty benefit to all robust estimations with absolutely no additional computational burden. *Once this is understood there is no reason to use* RANSAC *in preference to this method.* Similar schemes for robust estimation using random sampling and M-estimators were also proposed in [51] and [46].

5.5 Standard Deviation

This section gives a method for robustly estimating the standard deviation of the error term. Robust techniques to eliminate outliers are all founded upon some knowledge of the standard deviation σ of the error. Generally, given σ, outliers are calculated as follows:

$$z = \begin{cases} \text{non outlier} & |d| \leq t = 1.96\sigma \\ \text{outlier} & \text{otherwise,} \end{cases} \tag{5.13}$$

where $t = 1.96\sigma$ is a user defined threshold. In the case of the fundamental matrix there are two errors for each correspondence—the epipolar distances d_1, d_2 in each image. There are two options: either both can be tested by rule (5.13) and if either d_1, d_2 is greater than t then the correspondence is considered outlying; or, the two may be combined for a single test. The latter approach is followed, noting that $d_1^2 + d_2^2$ is approximated by a χ^2 variable with two degrees of freedom leads to the following 95% confidence test:

$$z_i = \begin{cases} \text{non outlier} & d_1^2 + d_2^2 \leq t = 5.99\sigma^2 \\ \text{outlier} & \text{otherwise,} \end{cases} \tag{5.14}$$

The standard deviation is related to the characteristics of the image, the feature detector and the matcher. Often the value of σ is unknown, in which case it must be estimated from the data. If there are no outliers in the data the σ can be estimated directly as the standard deviation of the residuals of a non-linear least squares minimization—e.g. $\mathbf{N_3}$. If there are outliers and they are in the minority, a first estimate of the variance can be derived from the median squared error of the chosen parameter fit [40]. It is known that $\text{med}_i|d_i|/\Phi^{-1}(0.75)$ is an asymptotically consistent estimator of σ when the d_i are distributed like $N(0, \sigma^2)$, where Φ is the cumulative distribution function for the Gaussian probability density function[1]. Empirically it has been shown [40] that when $n \approx 2p$ the correction factor of $\left(1 + \frac{5}{n-p}\right)$ improves the estimate of the standard deviation. Noting $1/\Phi^{-1}(0.75) = 1.4826$ the estimate of σ is

$$\sigma = 1.4826 \left(1 + \frac{5}{n-p}\right) \text{med}_i|d_i| \; . \tag{5.15}$$

The LMS algorithm is used to get the estimate of the median. The standard deviation can be estimated between each pair of images and the results filtered over time. Image pairs that give rise to unusually high standard deviations might possess independently moving objects. Given random perturbations of the image correspondences with unit standard deviation then the estimate of the standard deviation of **F** was found to be 1.3, this being a conflation of the image error and the error in the estimator.

[1] $N(0, \sigma^2)$ signifies a Gaussian or Normal distribution mean 0 and variance σ^2.

Chapter 6

Rematching

In version 1 this is not implemented, hopefully I will get round to it!

Epipolar Geometry and Feature Matching

Once the epipolar geometry has been estimated this can be used to aid matching. In [11, 2] the epipolar geometry is used to constrain the search area for a given match. Our proposal goes further in that we aim to conjoin the estimation of epipolar geometry and matching. As the match may be incorrect, it is desirable that, if in the course of the estimation process it is discovered that the feature is mismatched, then it can be rematched to another feature. In order to achieve this not only is a feature's initial match stored, based on cross correlation, but all its candidate matches that have a similarity score over a user defined threshold. After the robust estimation of the epipolar geometry all corners are rematched to the candidate with smallest epipolar distance. The epipolar geometry may then be further refined. In all the examples that follow the inliers and outliers are from the raw matches based only on cross correlation, in order to demonstrate the success of a given estimation.

Chapter 7

Self Calibration, establishing a projective frame

Once the fundamental matrix is estimated the projective structure is recovered. This chapter is a just sketch outline, and the user is referred for more details to [13, 25, 33]. All typos spotted please mail me!

It is possible to obtain only a projective reconstruction from the fundamental matrix, as shown in Section 7.1. A Euclidean reconstruction would be preferable. In order to achieve this the process has to stages (1) is to recover the camera positions and motions (modulo a scaling). Once this is done step (2) involves triangulation to recover the 3D points. Stage (1) requires some sort of calibration to convert from the uncalibrated fundamental matrix to the essential matrix. Here a *self calibration* process is implemented, described in Section 7.4.1, although there are other options (such as use of a calibration grid etc.). It is assumed only the focal length is unknown, and the Sturm [47] self calibration method used. Note that the focal length cannot be determined from a pure translation. Once the essential matrix is recovered, if the first camera is assumed to be at the origin of the coordinate system, then it is a simple matter to calculate the rotation and translation of the second camera relative to the first, described in Section 7.4.3. To improve the estimates of the camera intrinsic and extrinsic parameters a non linear optimization is used, see Section 7.4.5. Once these parameters are estimated then the camera projection matrices may be recovered and used to estimate the structure using generic projective methods as described in Section 7.1. The same method of triangulation can be used for projective or Euclidean reconstruction as it is the image error that is to be minimized.

7.1 Recovery of Projection Matrices

As laid out in Section 4.3, perspective projection from 3D to 2D by a 3×4 camera matrix $\mathbf{P}$

$$\mathbf{x} = \mathbf{P}\mathbf{X} \quad \text{and} \quad \mathbf{x}' = \mathbf{P}'\mathbf{X}$$

Thus if $\mathbf{P}, \mathbf{P}'$ are known then $\mathbf{X}$ may be recovered from from the matches $\mathbf{x}, \mathbf{x}'$. Fortunately, it is well known that given $\mathbf{F}$ then $\mathbf{P}, \mathbf{P}'$ can be estimated. One solution for establishing a projective frame is as follows [3]

1. Set $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$
2. Compute $\mathbf{F}$, Compute $\mathbf{e}'$ such that $\mathbf{e}'^\top \mathbf{F} = \mathbf{0}$, let $\mathbf{M} = [\mathbf{e}']_\times \mathbf{F}$.
3. Set the second projection matrix as

$$\mathbf{P}' = [\mathbf{M} + \mathbf{e}'\mathbf{b}^\top | c\mathbf{e}'] \tag{7.1}$$

   where $\mathbf{b}$ and c are an arbitrary 3-vector and scalar respectively, thus there are four degrees of freedom DOF in this choice (as there are 11 DOF in $\mathbf{P}$ and 7 DOF in $\mathbf{F}$: 11-7 = 4).
4. Normalize so that $\det \tilde{\mathbf{P}} > 0$, where $\tilde{\mathbf{P}}$ is the first 3×3 matrix of $\mathbf{P}$, this is useful for determining which side of the camera points are on in section 7.4.3.

This leads to projective reconstruction of the world, so called because a 3D projective transformation of the world coordinates $\mathbf{X}' = \mathbf{H}\mathbf{X}$, would lead to the same fundamental matrix. The following algorithm extracts the $\mathbf{P}$:

7.1.1 P,P$'$ from F

```
function [P1,P2] = torr_PfromF(FMat,m3)
```

Input: Input parameters defined as in Section 4.1.1.

Output: `P1, P2` the two 3×4 projection matrices.

7.2 Recovery of Projective Structure

Once the $\mathbf{P}$ matrices are recovered the structure $\mathbf{X}$ may be recovered by triangulation [24] however obtaining an optimal solution can be costly. This is because the optimal estimate would minimize the reprojection error of the 3D points i.e. minimize the sum of squares of Euclidean distance between the observed point in each image and the reprojection using the projection matrices and putative 3D structure i.e

$$\min_{\mathbf{X}} e_u(\mathbf{x}, \mathbf{P}\mathbf{X})^2 + e_u(\mathbf{x}', \mathbf{P}'\mathbf{X})^2 \tag{7.2}$$

where $e_u(\mathbf{a}, \mathbf{b})$ is the Euclidean distance between $\mathbf{a}$ and $\mathbf{b}$. This is equivalent to finding $(\hat{x}, \hat{y}, \hat{x}', \hat{y}')$ such that

$$\sum e = (x - \hat{x})^2 + (y - \hat{y})^2 + (\hat{x}' - x')^2 + (\hat{y}' + y')^2 \tag{7.3}$$

is a minimum and $(\hat{x}, \hat{y}, \hat{x}', \hat{y}')$ satisfies

$$\hat{\mathbf{x}}^\top \mathbf{F} \hat{\mathbf{x}} = 0 \tag{7.4}$$

where $\mathbf{x} = (x, y, 1)^\top$ and $\mathbf{x}' = (x', y', 1)^\top$.

This is a computationally expensive thing to do thus I have implemented a simpler scheme: (a) correct the point matches, using a first order correction based on Sampson (b) use a SVD method to estimate $\mathbf{X}$ from $\mathbf{x}$. First the SVD method is described. In section 7.3 the correction is described. Let $\mathbf{p}^{1-3}$ be the three rows of $\mathbf{P}$ and $\mathbf{p}'^{1-3}$ be the three rows of $\mathbf{P}'$ then it can be seen

$$\begin{aligned} x\mathbf{p}^{3\top}\mathbf{X} - \mathbf{p}^1\mathbf{X} &= 0 \\ y\mathbf{p}^{3\top}\mathbf{X} - \mathbf{p}^2\mathbf{X} &= 0 \\ x\mathbf{p}^{2\top}\mathbf{X} - y\mathbf{p}^1\mathbf{X} &= 0 \end{aligned}$$

thus an equation of the form $\mathbf{AX} = \mathbf{0}$ may be written with

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p}^{3\top} & - & \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} & - & \mathbf{p}^{1\top} \\ x'\mathbf{p}'^{3\top} & - & \mathbf{p}'^{1\top} \\ y'\mathbf{p}'^{3\top} & - & \mathbf{p}'^{2\top} \end{bmatrix} \tag{7.5}$$

thus once $\mathbf{A}$ is found $\mathbf{X}$ can be solved for using `torr_ls`, the function that does this is: `torr_triangulate` which is described next. However, in the noisy case, this should not be applied until the image coordinates have been corrected.

7.2.1 Quick triangulation function `torr_triangulate`

```
function X = torr_triangulate(matches, m3, P1, P2)
```

Input:

1. `matches` $n \times 4$ array of matches.
2. `m3` third homogeneous coordinate.
3. `P1, P2` the two 3×4 projection matrices.

Output: `X` $4 \times n$ array of homogeneous structure points.

7.3 Correction of the matches

Sampson's first order correction has been studied in detail in [25, 29], it is stated here without proof (until I get time to put one in) that the first order correction to a point is simply:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{x}' \\ \hat{y}' \end{pmatrix} = \begin{pmatrix} x \\ y \\ x' \\ y' \end{pmatrix} - \frac{r}{(\nabla r)^2} \begin{pmatrix} r_x \\ r_y \\ r_{x'} \\ r_{y'} \end{pmatrix} \tag{7.6}$$

contrary to the statement in [25] that this first order projection was only valid for one pixel noise, I have found it to give good results over a wide range of noise values. Once the points are corrected they should satisfy $\hat{\mathbf{x}}'\mathbf{F}\hat{\mathbf{x}} = 0$ thus the triangulation method given above should give equal results to that of [24] with only a fraction of the computational cost. The intuition here is to take a tangent plane to the manifold of the fundamental matrix and project points orthogonally down onto the manifold.

7.3.1 Correction function `torr_correctx4F`

This function generates the corrected matches as described in the previous subsection.

```
function [corrected_matches,sq_errors]
= torr_correctx4F(f, nx1,ny1,nx2,ny2, no_matches, m3)
```

Input: Input parameters defined as in Section 4.1.1.

Output:

1. `corrected_matches` the set of $n \times 4$ corrected matches.
2. `sq_errors` the $n \times 1$ array of squared errors.

7.3.2 Testing the two view match correction: `torr_test_correct_sc`

This simple script generates some random matches, estimates the fundamental matrix, and then corrects them to lie on the fundamental matrix, note how near to zero the resulting residuals are.

```
%torr_test_correct_sc.m
m3 = 256;
method = 2;


[true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,true_R,true_t, true_E] =
 torr_gen_2view_matches;

no_matches = length(nx1);

matches = [nx1,ny1,nx2,ny2];
set_rank2 = 0;


%first estimate F
[f, e1, n_inliers,inlier_index,nF] =
 torr_estimateF( matches, m3, [], method, set_rank2);
```

```
%next correct the points so that they lie on a fundamental matrix
[corrected_matches error2] =
 torr_correctx4F(f, nx1,ny1,nx2,ny2, no_matches, m3);

%check errors (should be near zero)
e = torr_errf2(f, corrected_matches(:,1), corrected_matches(:,2),
corrected_matches(:,3), corrected_matches(:,4),no_matches, m3)
```

7.4 Self Calibration

In section 7.1 it was shown that there is a four degree of freedom ambiguity in the recovery of the projection matrices. Ideally when resolving this ambiguity we would like to choose the projection matrices so that they would be as close to the true (Euclidean) matrices as possible. In order to do that a self calibration method is used, even if it is not totally accurate it should help to produce projection matrices that look near to the Euclidean ones, and hence structure that looks reasonable.

Road map

1. Recover $\mathbf{F}$, see last chapter.
2. Self Calibrate to get $\mathbf{C}$, Section 7.4.1.
3. Recover $\mathbf{E} = \mathbf{C}^\top \mathbf{E} \mathbf{C}^1$
4. Decompose $\mathbf{E}$ into $\mathbf{R}$ and $\mathbf{t}$.
5. Optimize $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{t}$.

7.4.1 Recovery of C

Once the fundamental matrix has been recovered the camera calibration matrix (4.3) can be recovered. At present only the case of self calibration with unknown focal length is considered implemented. This is because it can generally be assumed, for modern well engineered cameras, that the principal point (p_x, p_y) is at the centre of the image and that the aspect ratio is one. It is also assumed that the calibration matrix is the same between the images (i.e. no zoom). Peter Sturm recently presented a new method for self calibration in this case [47] [1]. His analysis is based on the $\mathbf{G}$ matrix, which is half way between the essential and fundamental matrices:

$$\mathbf{G} = \begin{bmatrix} a & 0 & 0 \\ 0 & 1 & -p_y \\ -p_x & -p_y & 1/f \end{bmatrix} \mathbf{F} \begin{bmatrix} a & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1/f \end{bmatrix} \tag{7.7}$$

and

$$\mathrm{diag}(1, 1, f)\mathbf{E}\,\mathrm{diag}(1, 1, f). \tag{7.8}$$

[1] It is important to note that the focal length cannot be determined in the case of pure translation.

At the moment the reader is referred to Sturm for the details of the self calibration method. One failure of the Sturm method is that we are fitting a 6 DOF $\mathbf{G}$ to a 7 DOF $\mathbf{F}$ which means that the $\mathbf{E}$ is not guaranteed to have two equal singular values, in order to fix this, once $\mathbf{E}$ is recovered the two non zero singular values are set to be equal and $\mathbf{E}$.

7.4.2 Sturm Self Calibration function

```
function [focal_length, E, CC_out] = torr_self_calib_f(F,CC)
```

Input:

1. `F`: the 3×3 fundamental matrix $\mathbf{F}$.
2. `CC`: the estimated calibration matrix.

Output:

1. `focal_length`: estimate of focal length.
2. `E`: estimate of essential matrix.
3. `CC_out`: estimate of calibration matrix.

7.4.3 Recovery of R and t

Given an Essential matrix it can be decomposed as follows [20, 33]. Suppose that the SVD of $\mathbf{E}$ is $\mathbf{E} = \mathbf{U}\Lambda\mathbf{V}^\top$. Define

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The two possible factorization $\mathbf{E} = \mathbf{T}\mathbf{R}_j$ as follows:

$$\mathbf{T} = \mathbf{U}\mathbf{Z}\mathbf{U}^\top \tag{7.9}$$

and

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \text{or} \quad \mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top.$$

Let $\mathbf{t}$ be the right epipole of $\mathbf{E}$, given by the third column of $\mathbf{U}$. Then, if the first projection matrix is set to

$$\mathbf{P} = \mathbf{C}[\mathbf{I}|\mathbf{0}] \tag{7.10}$$

there are four choices for $\mathbf{P}'$:

$$\mathbf{P}' = [\mathbf{R}_1|\mathbf{t}] \ \text{ or } \ \mathbf{P}' = [\mathbf{R}_1| - \mathbf{t}] \ \text{ or } \ \mathbf{P}' = [\mathbf{R}_2|\mathbf{t}] \ \text{ or } \ \mathbf{P}' = [\mathbf{R}_2| - \mathbf{t}] \tag{7.11}$$

It can be seen that two pairs of the solutions simply have the translation reversed. The other two pairs are called “twisted pairs” and in these the camera is rotated by 180 degrees about the line joining the camera centres. Richard Hartley [21, 25] proposed a scheme to resolve the ambiguity by looking at which side of the camera a reconstructed point lay. For only one of the four putative solutions for $\mathbf{P}'$ will the reconstructed point $\mathbf{X}$ lie in front of the camera in both views. In order to determine this the camera matrices must fist be normalized so that the determinate of the first 3×3 matrix is greater than zero. Let $\mathbf{x}$ and $\mathbf{x}'$ be the homogeneous image coordinate projections of a point $\mathbf{X}$, then $\mathbf{X}$ is in front of both cameras if

$$\mathbf{x}_3 \mathbf{X}_4 > 0 \quad \text{and} \quad \mathbf{x}'_3 \mathbf{X}_4 > 0$$

Once the rotation matrix is found it can written in terms of a minimal parametrization using Rodrigues’ formula

$$\mathbf{R} = \cos\omega \mathbf{I} + \sin\omega [\mathbf{l}]_\times + (1 - \cos\omega)\mathbf{l}\mathbf{l}^\top \tag{7.12}$$

where $\mathbf{l}$ is the axis of rotation, which may be recovered from the elements of $\mathbf{R}$ as:

$$\mathbf{l} = \begin{pmatrix} \mathbf{R}_{32} - \mathbf{R}_{23} \\ \mathbf{R}_{13} - \mathbf{R}_{31} \\ \mathbf{R}_{21} - \mathbf{R}_{12} \end{pmatrix} \tag{7.13}$$

and ω is the angle of rotation,

$$\omega = \arccos\left(\frac{\text{trace}\mathbf{R} - 1}{2}\right) \tag{7.14}$$

7.4.4 Function for R and t; `torr_linear_EtoPX`

```
function [P1,P2,R,t,rot_axis,rot_angle]
= torr_linear_EtoPX(E,matches,C,m3)
```

Input:

1. `E`: Essential matrix $\mathbf{E}$.
2. `matches` : $n \times 4$ array of matches.
3. `C`: calibration matrix $\mathbf{C}$.
4. `m3`: third homogeneous image coordinate.

Output:

1. `P1,P2`: estimated projection matrices$\mathbf{P}, \mathbf{P}'$
2. `R,t`: estimated rotation and translation $\mathbf{R}, \mathbf{t}$
3. `rot_axis,rot_angle`: angle and axis of rotation $\mathbf{l}$ and ω

7.4.5 Non-linear Optimization of G

Once extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ and the intrinsic parameter f have been recovered, they may be further optimized by gradient descent. There are 6 DOF in this formulation, a good minimal parametrization, represented by the 6×1 vector $\mathbf{g}$ is f the focal length, $\mathbf{t}$ the translation vector (parametrized in 2 spherical coordinates), $\mathbf{l}$ the axis of rotation (parametrized in 2 spherical coordinates), and ω the angle of rotation. Note the following two functions are provided for going from unit vectors to spherical coordinates and vice versa `torr_unit2sphere` and `torr_sphere2unit.`.

Given these six parameters $\mathbf{g}$ plus the calibration matrix $\mathbf{C}$, it is possible to calculate $\mathbf{F}$, the function provided to do this is `torr_g2F`. Once $\mathbf{F}$ is obtained the Sampson error may be computed in the usual way. The function provided to do this `torr_errg_sse` Thus it is possible to do gradient descent of the error function for $\mathbf{g}$. The function to do this is `torr_nonlinG`.

7.4.6 Non Linear minimization of g

```
function [g,f] =
 torr_nonlinG(g_init,nx1,ny1,nx2,ny2, no_matches, m3, C)
```

Input:

1. `nx1,ny1,nx2,ny2, no_matches, m3,`: defined as in Section 4.1.1.
2. `g_init`: initial estimate of $\mathbf{g}$ obtained from `torr_linear_EtoPX`.
3. `C`: the calibration matrix $\mathbf{C}$.

Output:

1. `f`: $\mathbf{f}$.
2. `g`: $\mathbf{g}$.

7.5 Testing Self Calibration `torr_test_calib_sc`

This script generates some synthetic matches as described in Chapter 8. Then a fundamental matrix is estimated, then calibration recovered (assuming everything is known bar the focal length). Then Essential matrix, rotation matrix and angle and axis of rotation are recovered. The script prints out the estimated and ground truth rotation axis, angle, translation and focal length. As a debugging exercise set `no_noise = 1` on line 25. The script runs on the noise free data, the astute reader will notice that the result and ground truth are the same.

```
%this is a script to test the self calibration stuff
%torr_test_calib_sc.m
%main()
```

```
%profile on
clear all;
m3 = 256;
method = 'mapsac';
method = 'linear';


%
 randn('state',0)
rand('state',0)
no_test = 1;
for(i = 1:no_test)

%     [true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2] = ...
%         torr_gen_2view_matches(foc, no_matches, noise_sigma,
 translation_mult, translation_adder, ...
%         rotation_multplier, min_Z,Z_RAN,m3);
[true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,true_R,true_TX, true_E] =
torr_gen_2view_matches;

no_matches = length(nx1);

%if we set this to one then the result should be the same as the groundtruth...
no_noise = 0;
if (no_noise)
    nx1 = x1;
    nx2 = x2;
    ny1 = y1;
    ny2 = y2;
end



matches = [nx1,ny1,nx2,ny2];
perfect_matches = [x1,y1,x2,y2];
set_rank2 = 1;


%first estimate F
[f, e1, n_inliers,inlier_index,nF] =
torr_estimateF( matches, m3, [], method, set_rank2);

%next correct the points so that they lie on a fundamental matrix
[corrected_matches error2] =
torr_correctx4F(f, nx1,ny1,nx2,ny2, no_matches, m3);

%now guess the camera calibration matrix
CC = diag(ones(3,1),0);
CC(3,3) = 1;
```

```
%next self calibrate for focal length
[focal_length, nE,CCC] = torr_self_calib_f(nF,CC);

%now we have an Essential matrix we can establish the camera frame...
[P1,P2,R,t,srot_axis,rot_angle,g]  = torr_linear_EtoPX(nE,matches,CCC,m3);

%next convert the 6 parameters of g to a fundamental matrix
f2 = torr_g2F(g,CCC);
disp('error before non-linear minimization')
e = torr_errf2(f2, nx1,ny1,nx2,ny2, length(nx1), m3);
norm(e)

[g,f] = torr_nonlinG(g ,nx1,ny1,nx2,ny2, no_matches, m3, CCC)


disp('error after')
 e2 = torr_errf2(f, nx1,ny1,nx2,ny2, length(nx1), m3);
norm(e2)


%the question now arises: how good is the fit? compare to groundtruth
true_rot_axis = [true_R(3,2)-true_R(2,3), true_R(1,3) - true_R(3,1), true_R(2,1) -
true_R(1,2)]';
true_rot_axis = true_rot_axis /norm(true_rot_axis);
true_rot_angle = acos( (trace(true_R)-1)/2);

true_t(1) = -true_TX(2,3);
true_t(2) = true_TX(1,3);
true_t(3) = -true_TX(1,2);
true_t = true_t/norm(true_t);


disp('true camera parameters')
true_t
true_rot_axis
true_rot_angle
true_C


rot_axis = torr_sphere2unit([g(2) g(3)]);
tt = torr_sphere2unit([g(5) g(6)]);
rot_angle = g(4);

CCC(3,3) = 1/g(1);

disp('estimated camera parameters')
tt
rot_axis
rot_angle
CCC
```

```
end
```

7.5.1 Displaying Structure, `torr_display_structure`

An example of synthetic matches and structure generated from them is given in figure 7.1. The 3D structure can be displayed using

```
f1 = torr_display_structure(X, P1, P2,display_numbers,f1).
```

The test diagnostics include, comparing the estimated structure with the ground truth, and examining the distance of the reprojected estimated structure to the ground truth image points. If the user sets `show_result = 1` these quantities are displayed. If the user sets `no_noise = 1` then the algorithm is tested on noise free points (for debugging). Note once the 3D points are displayed in a MATLAB figure, MATLAB provides a whole set of menu functions for altering the view of the 3D point set; furthermore it is easy to save the figure in any format e.g. encapsulated postscript.

Input:

1. `X`: either $3 \times n$ inhomogeneous 3D coordinates or $4 \times n$ homogeneous 3D coordinate.
2. `P1, P2`: two 3×4 projection matrices.
3. `f1`: figure handle of the figure for the matches to be displayed in.
4. `display_numbers`: if set to 1 then displays the index of each match.

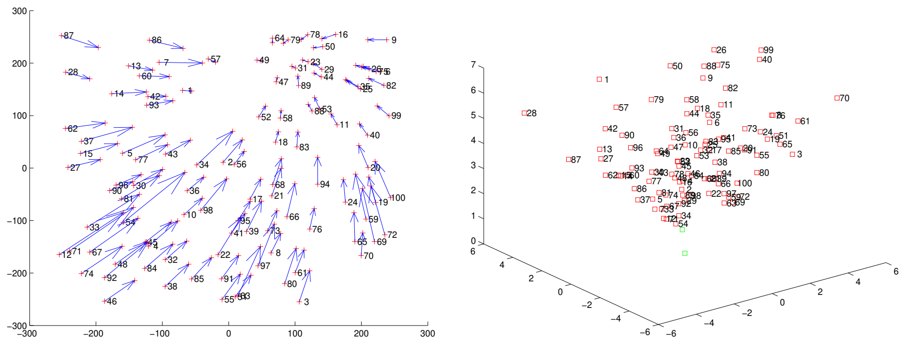
Output:

1. `f1`: : figure handle of the figure for the matches to be displayed in.

7.5.2 An example script for 3D structure generation

The script `torr_test_SFMsc` creates some synthetic matches, and then displays the 3D structure. The first half of the script follows `torr_test_calib_sc`; the second half is self explanatory and is listed here:

```
%next convert the 6 parameters of g to a fundamental matrix
f2 = torr_g2F(g,CCC);

%next correct the points so that they lie on the fundamental matrix
[corrected_matches error2] = torr_correctx4F(f2, nx1,ny1,nx2,ny2, no_matches, m3);

%next we need to obtain P1 & P2
[P1, P2] = torr_g2FP(g,CCC);
```



(a) (b)

Figure 7.1: *(a) A set of synthetic matches generated by* `torr_gen_2view_matches`, *displayed by* `torr_display_matches`. *(b) the 3D structure displayed by* `torr_display_structure`

```
%now use P matrices and corrected matches to get structure:
X = torr_triangulate(corrected_matches, m3, P1, P2);

torr_display_structure(X, P1, P2, 1);


%test
XX = [X(1,:) ./ X(4,:) ; X(2,:)  ./ X(4,:) ; X(3,:)  ./ X(4,:) ];

disp('ratio of estimated and true X');
XX ./true_X



show_result = 1;
if show_result
    disp('look at reprojection error to groundtruth points')
    x1_rp = P1 * X;
    x1_rp(1,:) = x1_rp(1,:) ./ x1_rp(3,:) * m3;
    x1_rp(2,:) = x1_rp(2,:) ./ x1_rp(3,:) * m3;
    (x1 - x1_rp(1,:)')'
    (y1 - x1_rp(2,:)')'

    x2_rp = P2 * X;
    x2_rp(1,:) = x2_rp(1,:) ./ x2_rp(3,:) * m3;
    x2_rp(2,:) = x2_rp(2,:) ./ x2_rp(3,:) * m3;
    (x2 - x2_rp(1,:)')'
    (y2 - x2_rp(2,:)')'
end
```

Chapter 8

Generating Synthetic Data

Data is randomly generated in $\mathcal{R}^3$ so that the imaged points lie in the range bounded by the the lines $y = -256$, $y = 256$, $x = -256$, $x = 256$ pixels for two imaginary cameras. The intrinsic parameters of the synthetic cameras can be represented by the matrix

$$\mathbf{C} = \begin{bmatrix} 1.00 & 0.00 & 0 \\ 0.00 & 1.00 & 0 \\ 0.00 & 0.00 & 1/f \end{bmatrix}, \tag{8.1}$$

corresponding to an aspect ratio of 1, with an optic centre at the image centre, and a focal length of $f =$ `foc`. The camera motion is a random translation and rotation, the translation is a multiple of the focal length. The rotation is generated using Rodrigues' formula

$$\mathbf{R} = \cos\omega \mathbf{I} + \sin\omega [\mathbf{l}]_\times + (1 - \cos\omega)\mathbf{l}\mathbf{l}^\top \tag{8.2}$$

where $\mathbf{l}$ is the axis of rotation (a random unit vector), and ω the angle of rotation about this axis.

The synthetic image positions are perturbed by Gaussian noise standard deviation by default 1.0 pixels and then quantized to the nearest pixel, simulating quite noisy image conditions.

8.0.3 Synthetic Two view match function

```
function [true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,
true_R,true_t, true_E] = ...
 torr_gen_2view_matches(foc, no_matches, noise_sigma,
translation_mult, translation_adder, ...
 rotation_multplier, min_Z,Z_RAN,m3)
```

Input:

1. `foc`: focal length.
2. `no_matches`: n the number of matches.

3. `noise_sigma`: σ of noise $e(\sigma)$ added to generated matches such that `nx1 = x1`: $+e(\sigma)$ etc.

4. `translation_adder, translation_mult`: the translation varies uniformly between `translation_adder` and `translation_adder + translation_mult`:.

5. `rotation_multplier`: the Euler angles of the rotation go between 0 and `rotation_multplier`:.

6. `min_Z, Z_RAN`: the depth varies uniformly between `min_Z` and min_Z + Z_RAN:.

7. `m3` third homogeneous coordinate.

Output:

1. `true_F`: true $\mathbf{F}$.

2. `x1,y1,x2,y2` true x, y, x', y'.

3. `nx1,ny1,nx2,ny2`: noisy x, y, x', y'.

4. `true_C, true_R,true_t, true_E`: true $\mathbf{C}$, $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{E}$.

8.0.4 A Script to generate and display synthetic matches

The script `torr_test_mat` generates some synthetic matches and then displays them.

```
[true_F,x1,y1,x2,y2,nx1,ny1,nx2,ny2,true_C,true_R,true_t, true_E]
 = torr_gen_2view_matches;
no_matches = length(nx1);
matches = [nx1,ny1,nx2,ny2];

%displayes matches
torr_display_matches(matches)
```

Chapter 9

The `Torr_tool` GUI

The GUI for the SAM library is the `torr_tool`. At present it is not fully finished and so the documentation and help in this chapter is not yet complete. The GUI puts together all the functions that have been described so far into an application, which can display the images and results. A rough and ready help can be obtained by typing `guide torr_tool` and using the property inspector to examine the call back routines of the buttons.

Some things to note:

1. The image coordinate system is chosen with origin at the centre of the image by setting the `XData` and `YData` properties `image`.
2. The demo images supplied are `j1.bmp, j2.bmp`.
3. `m3` the third homogeneous coordinate is 256.

9.1 Example

A quick tour

1. click on load demo images this loads `j1.bmp` and `j2.bmp`
2. click on load corners, and load `tab.cor` this displays Harris corners (detected by pressing Detect Corners). Observe the corners displayed by little crosses.
3. click on load matches, and load `table.matches`, this displays a whole load of correlation matches (detected by correlation matching)
4. click on load mat and F, load `table.fmatches` this loads a whole load of inlying matches to a precalculated **F** (via the MAPSAC button)
5. click on epipolar geometry, then click in the image and depress return, this shows the point clicked on and its corresponding epipolar line.
6. click on SFM to perform self calibration (here just for focal length) and create 3D structure for the inliers.

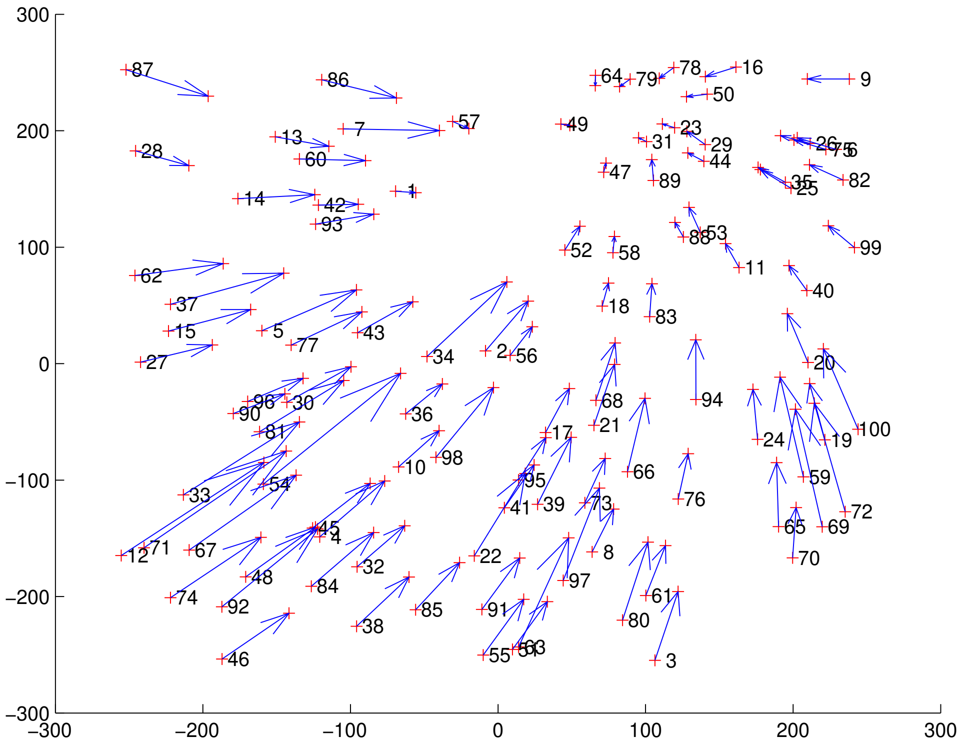


Figure 9.1:

9.2 Manual Addition of Matches

```
function [matches12,minc,mat12] =
torr_add_manual_matches(f,axes2,axes3,)
```

Input:

- `f` the vector form **f** of the fundamental matrix.
- `axes2, axes3` handles of the axes where matches are to be selected using `ginput`.

Output:

- `matches12` matches in an $n \times 4$ array of matches $(x, y, x'y')$, in this case n is the number of matches.
- `minc` is the minimum value of C for each corner.
- `mat12` is defined such that `mat(i) = j` means corner i matches to corner j.

Chapter 10

Conclusion and Future Work

The content of version 1 is just the bare minimum to do point based SAM recovery. The aim in version 2 is to complete the system: images in-dense 3D reconstruction out. Once the system is completed it will provide anyone with the tools to try out new developments in SAM by pulling out one module and replacing it with another. Key things on the critical path are

- Dealing with multiple images (loading in AVI's)
- Better Calibration.
- Much better feature detection and matching.
- Dense Stereo.
- Model selection.

Appendix A

Derivation of the Fundamental Matrix

This appendix introduces the elementary concepts concerning two view epipolar geometry, which is algebraically described in terms of, in the calibrated case the essential [31] matrix, and, in the uncalibrated case, the fundamental matrix [19, 12]. The essential matrix was developed for calibrated cameras and used as the starting point of the 8-point algorithm which recovered structure, it is now stated without proof:

Theorem 4 (Longuet-Higgins [31]) *Given a pin hole camera let the set of homogeneous image points* $\{\mathbf{x}_i\}, i = 1, \ldots N$, *be transformed to the set* $\{\mathbf{x}'_i\}$ *on the image plane by the motion parameters* $\{\mathbf{R}, \mathbf{t}\}$ *such that* $\mathbf{t} \neq \mathbf{0}$. *Then there exists a* 3×3 *essential matrix* $\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times$ *such that*

$$\mathbf{x}_i^{'\top} \mathbf{E} \mathbf{x}_i = 0 \tag{A.1}$$

for all i.

It can also be shown that $||\mathbf{E}|| = \sqrt{2}$ and that the singular values of $\mathbf{E}$ are 1,1 and 0, and so $|\mathbf{E}| = 0$. Given a perfect camera one can recover the motion parameters purely from point correspondences without knowledge of the scene structure, given that the points do not lie in special configuration.

The analysis for the calibrated case has been extended to that where the calibration is unknown. It is a remarkable fact that if neither the internal camera calibration nor the camera motion are known then there is still a set of linear equations linking the points in the two images. The case for algorithms that do not require calibration has been strongly made in [12]. Camera calibration is at best difficult possibly introducing correlated errors into the system, and at worst it is often impossible. The fundamental matrix constraint is now stated and proved:

Theorem 5 (Faugeras [12], Hartley [19]) *Given an uncalibrated camera let the set of homogeneous image points* $\{\mathbf{x}_i\}, i = 1, \ldots N$, *be transformed to the set* $\{\mathbf{x}'_i\}$ *on* *the image plane by the motion parameters* $\{\mathbf{R}, \mathbf{t}\}$ *such that* $\mathbf{t} \neq \mathbf{0}$. *Then there exists a* 3×3 *matrix* $\mathbf{F}$ *such that*

$$\mathbf{x}_i'^{\top} \mathbf{F} \mathbf{x}_i = 0 \tag{A.2}$$

for all i.

Proof. Let a real world point $\mathbf{x} = (x, y, z) \in \mathcal{R}^3$ and let $\mathbf{x}$ and $\mathbf{x}'$ be the homogeneous coordinates of the image points with the camera at $\mathbf{0}$ and $\mathbf{t}$ respectively. If $\mathbf{C}$ gives the camera's intrinsic parameters then the camera transformations are given by $\mathbf{P} = [\mathbf{C} \mid \mathbf{0}]$ and $\mathbf{P}' = [\mathbf{CR} \mid \mathbf{Ct}]$ before and after the motion. As before we determine the epipolar line of $\mathbf{x}$ in the second image by looking at the image under $\mathbf{P}'$ of the camera centre $(0, 0, 0, 1)$ and the point at infinity $(\mathbf{C}^{-1}\mathbf{x}, 0)$. The images of these two points under $\mathbf{P}'$ are $\mathbf{Ct}$ and $\mathbf{CRC}^{-1}\mathbf{x}$ respectively. Thus the epipolar line $\mathbf{n}$ is given by

$$\mathbf{n} = \mathbf{Ct} \times \mathbf{CRC}^{-1}\mathbf{x} \tag{A.3}$$

Since $\mathbf{x}'$ lies on this epipolar

$$\mathbf{x}'^{\top} \mathbf{F} \mathbf{x} = 0 \tag{A.4}$$

where we term $\mathbf{F}$ the fundamental matrix such that $\mathbf{F} = \mathbf{Ct} \times \mathbf{CRC}^{-1}$.

If $\mathbf{a}$ and $\mathbf{b}$ are 3-dimensional column vectors and $\mathbf{C}$ is a 3×3 matrix, then

$$\mathbf{Ca} \times \mathbf{Cb} \sim \mathbf{C}^*(\mathbf{a} \times \mathbf{b}) \tag{A.5}$$

where $\mathbf{C}^*$ is the adjoint of $\mathbf{C}$. This fact allows us to write the fundamental matrix in terms of the intrinsic matrix and the essential matrix:

$$\mathbf{F} = \mathbf{C}^{-\top} \mathbf{E} \mathbf{C}^{-1} \tag{A.6}$$

thus the fundamental matrix at most of rank two. In order to proceed only from image measurements $\mathbf{F}$ is the key concept, as it encapsulates all the geometric information on camera and motion contained within a set of point correspondences. The fundamental matrix is determined by and determines the epipolar transformation. Theorem 5 leads to the following lemma.

Lemma 2 *If* $\mathbf{F}$ *is a fundamental matrix corresponding to a pair of image and* $\mathbf{x}$ *is a point in the first image, then* $\mathbf{Fx}$ *is the epipolar line in the second image corresponding to* $\mathbf{x}$.

From $\mathbf{F}$ and the image correspondences it is straightforward to recover *projective* structure as has been pointed out by Faugeras [12] and Hartley [19]:

Theorem 6 (Faugeras [12],Hartley [19]) *Given a set of image correspondences sufficient to determine the fundamental matrix, the corresponding world space coordinates are determined up to a collineation of projective* 3*-space* $\mathcal{P}^3$.

Proof. We shall not present a formal proof, but note that if a set of points $\mathbf{x}_i \subset \mathcal{P}^3$ are visible to a pair of camera with transform matrices $\mathbf{P}$ and $\mathbf{P}'$, and if $\mathbf{G}$ is an arbitrary non-singular 4×4 matrix, then replacing $\mathbf{x}_i$ by $\mathbf{G}^{-1}\mathbf{x}_i$, $\mathbf{P}$ by $\mathbf{PG}$ and $\mathbf{P}'$ by $\mathbf{P}'\mathbf{G}$ preserves the object-point to image-space correspondences. As may be seen, the internal parameters of one of the cameras may be changed arbitrarily. Thus points that are consistent with Equation (A.4) can be said to move rigidly modulo a collineation.

Appendix B

Singular Value Decomposition and Least Squares

Let the singular value decomposition of $\mathbf{D}$ (eg. [48, 59]), be given by

$$\mathbf{D} = \mathbf{V}\Lambda\mathbf{U}^\top \quad , \tag{B.1}$$

where $\mathbf{V}$ is a $n \times p$ matrix whose columns are the left hand singular vectors of $\mathbf{D}$, $\mathbf{U}$ is a $p \times p$ matrix whose columns are the right hand singular vectors of $\mathbf{D}$ and Λ is the diagonal matrix of the corresponding singular values of $\mathbf{D}$: $\Lambda = \mathrm{diag}(\lambda^{\frac{1}{2}}{}_1, \lambda^{\frac{1}{2}}{}_2, \ldots, \lambda^{\frac{1}{2}}{}_p)$ in ascending order such that $\lambda^{\frac{1}{2}}{}_1$ is the minimum singular value. $\mathbf{U}$ is row and column orthogonal i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{I}, \mathbf{U}\mathbf{U}^\top = \mathbf{I}$; $\mathbf{V}$ is column orthogonal such that $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$. The SVD gives orthonormal bases for the null space and range of $\mathbf{D}$. The columns of $\mathbf{V}$ corresponding to zero singular values form a basis for the range of $\mathbf{D}$. The columns of $\mathbf{U}$ corresponding to zero singular values form a basis for the null space.

It is clear that the SVD of $\mathbf{D}$ is very closely related to the eigensystem of the moment matrix: $\mathbf{Z}^\top\mathbf{Z} = \mathbf{D}^\top\mathbf{D}$,

$$\begin{aligned} \mathbf{D} &= \mathbf{V}\Lambda\mathbf{U}^\top & \text{(B.2)} \\ \Rightarrow \mathbf{D}^\top\mathbf{D} &= \mathbf{U}\Lambda\mathbf{V}^\top\mathbf{V}\Lambda\mathbf{U}^\top & \text{(B.3)} \\ &= \mathbf{U}\Lambda^2\mathbf{U}^\top & \text{(B.4)} \\ \Rightarrow \mathbf{D}^\top\mathbf{D}\mathbf{U} &= \mathbf{U}\Lambda^2 \quad . & \text{(B.5)} \end{aligned}$$

Thus we see that the columns of $\mathbf{U}$ are eigenvectors of $\mathbf{Z}^\top\mathbf{Z}$, and that the squares of the singular values of $\mathbf{D}$ are the eigenvalues of $\mathbf{Z}^\top\mathbf{Z}$. The solution to the orthogonal regression problem:

$$\mathbf{f} = \min \mathbf{f}(\mathbf{f}^\top\mathbf{Z}^\top\mathbf{Z}\mathbf{f}) \tag{B.6}$$

is well known to be the eigenvector of $\mathbf{Z}^\top\mathbf{Z}$ corresponding to the minimum eigenvalue, this is simply $\mathbf{u}_1$ the first column of $\mathbf{U}$, which is the shortest principal axis. Thus we have selected the direction of minimum variation as normal to our fitted plane. The sum of squares of residuals in this direction is λ_1, similarly, if we were to choose $\mathbf{u}_2$ as our solution then the sum of squares of residuals would be λ_2. Thus if λ_2 is close to λ_1 we can see that there might not be a unique solution. The residuals, i.e. the projections of each point onto $\mathbf{u}_1$ can be efficiently calculated by noting that $\mathbf{DU} = \mathbf{V}\Lambda$, thus $r_i = \lambda^{\frac{1}{2}}{}_1\mathbf{V}(i,1)$ where $\mathbf{V}(i,1)$ is the element of $\mathbf{V}$ on the ith row and first column. Similarly $\mathbf{u}_k^\top \mathbf{z}_i = \lambda^{\frac{1}{2}}{}_k\mathbf{V}(i,k)$.

We shall prefer the use of the SVD rather than the eigen-decomposition of $\mathbf{Z}^\top\mathbf{Z}$ for numerical reasons [4]: namely that the SVD relates directly to the data matrix and the algorithms in existence for its computation are more stable than those that calculate the eigensystem of $\mathbf{Z}^\top\mathbf{Z}$, especially if $\mathbf{D}$ is ill-conditioned. Furthermore, in operating directly on the $n \times p$ data matrix we avoid the np^2 sums and products needed to calculate $\mathbf{Z}^\top\mathbf{Z}$. In passing we note that in the case of classical least squares the hat matrix is easily computed from the SVD:

$$\mathbf{H} = \mathbf{V}\mathbf{V}^\top \tag{B.7}$$

this form of calculation is preferable especially if $\mathbf{D}$ is of less than full rank.

Appendix C

Orthogonal Regression—affine case

Consider fitting a hyperplane $\mathbf{f} = (f_1, f_2, \ldots, f_p)$ through a set of n p-dimensional points with homogeneous coordinates $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{ip-1}, 1)$. This can alternatively be viewed as either fitting a hyperplane in p-dimensions through the origin using homogeneous coordinates or fitting a hyperplane in $p-1$ dimensions not through the origin using inhomogeneous coordinates $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{ip-1})$. The best fitting hyperplane $\mathbf{f}$ is estimated by minimizing the perpendicular sum of Euclidean distances from the points to the plane. This is accomplished by minimizing $\sum_{i=1}^{n}(\mathbf{f}^\top \mathbf{z}_i)^2$ subject to the constraint[1] $\sum_{i=1}^{p-1}(\mathbf{f})^2 = 1$. This constraint ensures that the estimate will be invariant to equiform transformation of the inhomogeneous coordinates. For example the best fitting line to a 2 dimensional scatter (x_i, y_i), $i = 1 \ldots n$ is estimated by minimizing $\sum_{i=1}^{n}(ax + by + c)^2$ subject to the constraint $a^2 + b^2 = 1$ [37].

Let $\mathbf{M} = \mathbf{Z}^\top \mathbf{Z}$ be the moment matrix then the estimate $\mathbf{f}$ minimizes $\mathbf{f}^\top \mathbf{M} \mathbf{f}$ subject to $\mathbf{f}^\top \mathbf{J} \mathbf{f} = \text{constant}$, where $\mathbf{J} = \text{diag}(1, 1, 1, \ldots, 1, 0)$. This estimate is given by the eigenvector corresponding to the minimum eigenvalue of the centred moment matrix. Centring is a standard statistical technique that involves shifting the coordinate system of the data points so that the centroid lies at the origin. This can be effected by subtracting $\vec{1}\,\overline{z}_j$ from each column of $\mathbf{Z}$. Where $\vec{1}$ is an n dimensional vector such that $\vec{1} = (1, 1, 1, \ldots, 1)^\top$ and $\overline{z}_j$ is the mean of that column.

Proof: Let us partition $\mathbf{f}$ into $(\mathbf{f}_1|\mathbf{f}_2)$ with components of length $p-1$ and 1 respectively, and let $\mathbf{M}$ be partitioned in a corresponding manner:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} , \tag{C.1}$$

then

$$\mathbf{f}^\top \mathbf{M} \mathbf{f} = \mathbf{f}_1^\top \mathbf{M}_{11} \mathbf{f}_1 + 2\mathbf{f}_1^\top \mathbf{M}_{12} \mathbf{f}_2 + \mathbf{f}_2^\top \mathbf{M}_{22} \mathbf{f}_2 \ . \tag{C.2}$$

[1]With non-affine higher order surfaces the constraint will be more complex involving higher order combinations of the coefficients.

For any fixed $\mathbf{f}_1^\top$, $\mathbf{f}^\top\mathbf{M}\mathbf{f}$ is minimal when

$$\frac{\partial \mathbf{f}^\top\mathbf{M}\mathbf{f}}{\partial \mathbf{f}_2^\top} = 0 \tag{C.3}$$

i.e.

$$2\mathbf{f}_1^\top\mathbf{M}_{12} + \mathbf{f}_2^\top\mathbf{M}_{22} = 0 \tag{C.4}$$

which implies

$$\mathbf{f}_2^\top = -2\mathbf{f}_1^\top\mathbf{M}_{12}\mathbf{M}_{22}^{-1} \ , \tag{C.5}$$

therefore

$$\mathbf{f}^\top\mathbf{M}\mathbf{f} = \mathbf{f}_1^\top(\mathbf{M}_{11} - \mathbf{M}_{12}\mathbf{M}_{22}^{-1}\mathbf{M}_{21})\mathbf{f}_1 \ . \tag{C.6}$$

Let us define

$$\tilde{\mathbf{M}} \stackrel{\text{def}}{=} \mathbf{M}_{11} - \mathbf{M}_{12}\mathbf{M}_{22}^{-1}\mathbf{M}_{21} \tag{C.7}$$

To minimize this for $\mathbf{f}^\top\mathbf{J}\mathbf{f} =$ constant, let κ be a Lagrangian multiplier for the constraint. Then we must set to zero the derivative with respect to zero of $\mathbf{f}_1^\top\tilde{\mathbf{M}}\mathbf{f}_1 - \kappa\mathbf{f}_1^\top\mathbf{J}\mathbf{f}_1$. This yields:

$$2\mathbf{f}_1^\top\tilde{\mathbf{M}} = 2\kappa\mathbf{f}_1^\top\mathbf{J} = 2\kappa\mathbf{f}_1^\top \tag{C.8}$$

so that κ is an eigenvalue of $\tilde{\mathbf{M}}$ with $\mathbf{f}_1^\top$ the corresponding eigenvector. The eigenvector of the best geometric fit usually corresponds to the smallest eigenvalue. It shall be shown elsewhere that the matrix $\tilde{\mathbf{M}}$ is related to the covariance matrix of the residuals. What form has $\tilde{\mathbf{M}}$? In the affine case it can be seen that $\mathbf{M} = \mathbf{Z}^\top\mathbf{Z}$ leads to $\mathbf{M}_{22} = 1$ thus

$$\tilde{\mathbf{M}} = \mathbf{M}_{11} - \mathbf{M}_{12}\mathbf{M}_{21} \tag{C.9}$$

using the fact

$$\begin{aligned}\sum(x-\overline{x})(y-\overline{y}) &= \sum(xy - \overline{x}y - \overline{y}x + \overline{x}\,\overline{y} \\ &= \sum(xy) - \sum x \sum y\end{aligned}$$

then equation (C.9) becomes

$$\begin{aligned}\tilde{\mathbf{M}} &= \begin{bmatrix} \sum z_{i1}^2 & \sum(z_{i1}z_{i2}) & \cdots \\ \sum(z_{i2}z_{i1}) & \sum z_{i2}^2 & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} - \begin{bmatrix} (\sum z_{i1})^2 & \sum z_{i1}\sum z_{i2} & \cdots \\ \sum z_{i2}\sum z_{i1} & (\sum z_{i2})^2 & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \\ &= \begin{bmatrix} \sum z_{i1}^2 - (\sum z_{i1})^2 & \sum(z_{i1}z_{i2}) - \sum z_{i1}\sum z_{i2} & \cdots \\ \sum(z_{i2}z_{i1}) - \sum z_{i2}\sum z_{i1} & \sum z_{i2}^2 - (\sum z_{i2})^2 & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix}\end{aligned}$$

from which it can be seen that $\tilde{\mathbf{M}}$ is the centred moment matrix.

A formal proof is not presented, rather it is noted that if a set of points $\mathbf{x}_i \subset \mathcal{P}^3$ are visible to a pair of camera with transform matrices $\mathbf{P}$ and $\mathbf{P}'$, and if $\mathbf{G}$ is an arbitrary non-singular 4×4 matrix, then replacing $\mathbf{x}_i$ by $\mathbf{G}^{-1}\mathbf{x}_i$, $\mathbf{P}$ by $\mathbf{PG}$ and $\mathbf{P}'$ by $\mathbf{P}'\mathbf{G}$ preserves the object-point to image-space correspondences. As may be seen, the internal parameters of one of the cameras may be changed arbitrarily. Thus points that are consistent with Equation (4.10) can be said to move rigidly modulo a collineation.

Appendix D

Variance of residuals

In Section 4.7 it was shown how each constraint may be reweighted by the variance of its corresponding residual, in order to provide a more statistically sound minimization measure. In this appendix an expression for the variance of each residual is derived, also bias in linear estimation is discussed.

Consider the problem of fitting a fundamental matrix to the data points $\mathbf{z}_i, i = 1 \dots n$, defined in (4.20). Let $\underline{\mathbf{f}}$ be the exact fundamental matrix, written in vector form and let $\mathbf{f}$ be the estimate. If $\mathbf{f}$ is computed by the least squares optimization:

$$\mathbf{f} = \min_{\mathbf{f}} \sum_{i=1}^{n} \frac{(\mathbf{f}^\top \mathbf{z}_i)^2}{w_i} \tag{D.1}$$

where w_i is the optimal weight (being the variance of the residual). Let $\mathbf{Z}$ be the matrix whose rows are $\mathbf{z}_i^\top / w_i$ and $\mathbf{M} = \mathbf{Z}^\top \mathbf{Z}$ be the p-dimensional symmetric moment matrix

$$\mathbf{M} = \sum_{i=1}^{n} \frac{\mathbf{z}_i \mathbf{z}_i^\top}{w_i^2} \tag{D.2}$$

If $\mathbf{M}$ has eigenvalues, in increasing order, $\lambda_1 \dots \lambda_p$ and corresponding eigenvectors $\mathbf{u}_1 \dots \mathbf{u}_p$ then

Theorem 7 (Weng *et al* **[61])** *If covariance matrix for* $\mathbf{f} = \mathbf{u}_1$ *is*

$$\Gamma_f = \mathcal{E}\{\delta \mathbf{u}_1 \delta \mathbf{u}_1^\top\} \tag{D.3}$$

the optimal weighting is

$$w_i = \mathbf{f}^\top \Gamma_{z_i} \mathbf{f}. \tag{D.4}$$

and that the covariance matrix for $\mathbf{f}$ *is*

$$\Gamma_f = \sigma^2 \sum_{k \neq 1}^{p} \frac{\mathbf{u}_k \mathbf{u}_k^\top}{\lambda_k}. \tag{D.5}$$

Proof: Following the analysis of Weng *et al* [61]. Let the homogeneous image points $\mathbf{x}_i = (x_i, y_i, \zeta)$ in the first image be matched to $\mathbf{x}'_i = (x'_i, y'_i, \zeta)$ in the second image. The noise in the data matrix $\mathbf{Z}$ is due to noise in the image coordinates, arising from spatial quantisation, feature detection errors, point mismatching and camera distortion. We shall assume that the noise in the image coordinates has zero mean and known variance. If the noise arises from many sources and is influenced by the sum of many factors, its distribution is roughly Gaussian by the central limit theorem. (In fact, as we shall see, steps can be taken to ensure that the methods we use are robust not only to outliers, but also to departure from the Gaussian distributions.) We shall further assume that the noise is uncorrelated between different image points and that the noise in the two components of the image coordinates is also uncorrelated. The covariance matrix of $\mathbf{f}$ is derived on the basis of first order perturbations. Consider two image coordinates x and y with small errors:

$$\begin{aligned} x &= \underline{x} + \delta x \\ y &= \underline{y} + \delta y. \end{aligned}$$

To first order

$$xy \approx \underline{xy} + \delta x \underline{y} + \delta y \underline{x}. \tag{D.6}$$

The perturbed data matrix $\mathbf{Z} = \underline{\mathbf{Z}} + \delta\mathbf{Z}$ gives rise to a perturbed moment matrix $\mathbf{M}$

$$\begin{aligned} \underline{\mathbf{M}} + \delta\mathbf{M} &= (\underline{\mathbf{Z}} + \delta\mathbf{Z})^\top (\underline{\mathbf{Z}} + \delta\mathbf{Z}) \\ &= \underline{\mathbf{Z}}^\top \underline{\mathbf{Z}} + \delta\mathbf{Z}^\top \underline{\mathbf{Z}} + \underline{\mathbf{Z}}^\top \delta\mathbf{Z} + O^2. \end{aligned}$$

Thus to first order

$$\delta\mathbf{M} = \delta\mathbf{Z}^\top \underline{\mathbf{Z}} + \underline{\mathbf{Z}}^\top \delta\mathbf{Z}, \tag{D.7}$$

where the rows of $\delta\mathbf{Z}$ are

$$\delta\mathbf{z}_i \stackrel{\text{def}}{=} \left(\delta x_i \underline{x}'_i + \delta x'_i \underline{x}_i \quad \delta y_i \underline{x}'_i + \delta y_i \underline{x}'_i \quad \delta x'_i \zeta \quad \delta y'_i \underline{x}_i + \delta x_i \underline{y}'_i \quad \delta y'_i \underline{y}_i + \delta y_i \underline{y}'_i \quad \delta y'_i \zeta \quad \delta x_i \zeta \quad \delta y_i \zeta \quad 0 \right).$$

Under the assumption that there is no cross correlation of error between correspondences we define the covariance matrix of any two rows of $\mathbf{Z}$ to be

$$\mathcal{E}(\delta\mathbf{z}_i \delta\mathbf{z}_j^\top) = \begin{cases} \mathbf{0} & \text{if } i \neq j \\ \Gamma_{z_i} & \text{if } i = j \end{cases} \tag{D.8}$$

where

$$\Gamma_{z_i} = \sigma^2 \begin{bmatrix} \underline{\mathbf{x}}'_i \underline{\mathbf{x}}'_i & 0 & 0 \\ 0 & \underline{\mathbf{x}}'_i \underline{\mathbf{x}}'_i & 0 \\ 0 & 0 & \underline{\mathbf{x}}'_i \underline{\mathbf{x}}'_i \end{bmatrix} + \sigma^2 \begin{bmatrix} \underline{x}^2 \mathbf{J} & \underline{xy} \mathbf{J} & \underline{x}\zeta \mathbf{J} \\ \underline{xy} \mathbf{J} & \underline{y}^2 \mathbf{J} & \underline{y}\zeta \mathbf{J} \\ \underline{x}\zeta \mathbf{J} & \underline{y}\zeta \mathbf{J} & \zeta^2 \mathbf{J} \end{bmatrix} \tag{D.9}$$

and with

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{D.10}$$

Given Γ_{z_i} we can derive the estimated variance of each residual. If $\underline{\mathbf{z}}_i$ is perturbed into $\mathbf{z}_i = \underline{\mathbf{z}}_i + \delta\mathbf{z}_i$, then $r_i = \underline{\mathbf{f}}^\top \mathbf{z}'$ can be written to a first approximation as

$$r_i = \underline{\mathbf{f}}^\top \delta\mathbf{z}_i \tag{D.11}$$

assuming that $\underline{\mathbf{f}}$ is known. As $\underline{\mathbf{f}}$ is unknown we must use its estimation $\mathbf{f}$ in its place, it is hoped that $\mathbf{f}$ will converge to $\underline{\mathbf{f}}$ as the algorithm progresses. The residual r_i is a random variable with mean zero and variance given by

$$\sigma_i^2 = \mathcal{E}(r_i^2) = \mathbf{f}^\top \mathcal{E}(\delta\mathbf{z}_i \delta\mathbf{z}_i^\top)\mathbf{f} = \mathbf{f}^\top \Gamma_{z_i} \mathbf{f}. \tag{D.12}$$

Thus we set $w_i = \sigma_i^2$ in equation (D.1), providing the maximum likelihood estimate of $\mathbf{f}$ (if the r_i follow a Gaussian distribution). By equating coefficients it can be shown that

$$\mathbf{f}^\top \Gamma_{z_i} \mathbf{f} = \sigma^2 (\nabla r_i)^2 \tag{D.13}$$

where the gradient, ∇r, is easily computed:

$$\begin{aligned}
\nabla r &= (r_x^2 + r_y^2 + r_{x'}^2 + r_{y'}^2)^{\frac{1}{2}} \\
r_x &= f_1 x' + f_4 y' + f_7 \zeta \\
r_y &= f_2 x' + f_5 y' + f_8 \zeta \\
r_{x'} &= f_1 x + f_2 y + f_3 \zeta \\
r_{y'} &= f_4 x + f_5 y + f_6 \zeta,
\end{aligned}$$

where r_x denotes the partial derivative of r (given in Equation (4.23)) with respect to x. The optimal weights involve the unperturbed points $\underline{\mathbf{z}}$, so we must approximate them by their sample values. This forms the basis of Sampson's method.

Next we shall derive the covariance matrix for $\mathbf{f} = \mathbf{u}_1$. If the moment matrix is perturbed by $\delta\mathbf{M}$ then the solution is perturbed by

$$\delta\mathbf{u}_1 = \sum_{k \neq 1}^{p} \frac{\mathbf{u}_k^\top \delta\mathbf{M}\mathbf{u}_1}{\lambda_k} \mathbf{u}_k. \tag{D.14}$$

Noting that

$$\mathbf{u}_k^\top \delta\mathbf{M}\mathbf{u}_1 = \sum_{i=1}^{n} \frac{\mathbf{u}_k^\top \mathbf{z}_i \delta\mathbf{z}_i^\top \mathbf{u}_1}{w_i}, \tag{D.15}$$

and defining vectors

$$\mathbf{s}_i = \sum_{k \neq 1}^{p} \frac{\mathbf{u}_k^\top \mathbf{z}_i}{\lambda_k} \mathbf{u}_k, \tag{D.16}$$

it can be seen that

$$\delta\mathbf{u}_1 = \sum_{i=1}^{n} \frac{\delta\mathbf{z}_i^\top \mathbf{u}_1 \mathbf{s}_i}{w_i}. \tag{D.17}$$

Hence the covariance matrix Γ_f is given by

$$\begin{aligned}\Gamma_f &= \sum_{i,j=1}^{n} \mathcal{E}\left(\frac{(\delta \mathbf{z}_i^\top \mathbf{u}_1 \delta \mathbf{z}_j^\top \mathbf{u}_1)\mathbf{s}_i \mathbf{s}_j^\top}{w_i w_j}\right) \\ &= \sum_{i,j=1}^{n} \mathcal{E}\left(\frac{(\mathbf{u}_1^\top \delta \mathbf{z}_i^\top \delta \mathbf{z}_j)\mathbf{u}_1 \mathbf{s}_i \mathbf{s}_j^\top}{w_i w_j}\right).\end{aligned}$$

Using (D.8) we see that

$$\Gamma_f = \sum_{i,j=1}^{n} \frac{\delta_{ij} \mathbf{u}_1^\top \Gamma_{z_i} \mathbf{u}_1 \mathbf{s}_i \mathbf{s}_j^\top}{w_i w_j},$$

where the Kronecker delta $\delta_{jk} = 0$ if $j \neq k$, $\delta_{jk} = 1$ if $j = k$. From (D.4) we see that

$$\begin{aligned} &= \sum_{i,j=1}^{n} \frac{\delta_{ij} w_i \mathbf{s}_i \mathbf{s}_i^\top}{w_i w_j} \\ &= \sum_{i=1}^{n} \frac{\mathbf{s}_i \mathbf{s}_i^\top}{w_i}.\end{aligned}$$

We are now in a position to expand the $\mathbf{s}$ vectors giving

$$\Gamma_f = \sum_{i=1}^{n} \frac{1}{w_i} \sum_{k,l \neq 1}^{p} \frac{\mathbf{u}_k^\top \mathbf{z}_i \mathbf{z}_i^\top \mathbf{u}_l}{\lambda_k \lambda_l} \mathbf{u}_k \mathbf{u}_k^\top. \tag{D.18}$$

Since $\mathbf{u}_k$ and $\mathbf{u}_l$ are eigenvectors of $\mathbf{M}$ we have:

$$\begin{aligned}\mathbf{u}_k^\top \mathbf{M} \mathbf{u}_l &= \sum_{i=1}^{n} \frac{\mathbf{u}_k^\top \mathbf{z}_i \mathbf{z}_i^\top \mathbf{u}_l}{w_i} \\ &= \lambda_k \delta_{kl}.\end{aligned}$$

and thus we obtain

$$\Gamma_f = \sigma^2 \sum_{k \neq 1}^{p} \frac{\mathbf{u}_k \mathbf{u}_k^\top}{\lambda_k}. \tag{D.19}$$

D.1 Bias in Linear Estimation

An estimate is statistically biased if the expectation of the error is zero, and statistically biased otherwise. Following [28] it can be seen that the true perturbation of $\mathbf{M}$ is

$$\delta \mathbf{M} = \delta \mathbf{Z}^\top \underline{\mathbf{Z}} + \underline{\mathbf{Z}}^\top \delta \mathbf{Z} + \delta \mathbf{Z}^\top \delta \mathbf{Z} \tag{D.20}$$

rather than that given in Equation (D.7), and its expectation is not zero:

$$\mathcal{E}(\delta \mathbf{M}) = \delta \mathbf{Z}^\top \delta \mathbf{Z} \;. \tag{D.21}$$

Kanatani suggests (in the case of conics [28]) that the bias may be removed by subtracting the bias term $\mathcal{E}(\delta\mathbf{M})$ from the moment matrix prior to finding the eigenvectors. In other words taking as our solution the eigenvector corresponding to the minimum eigenvalue of

$$\mathbf{M} - \mathcal{E}(\delta\mathbf{M}) \ . \tag{D.22}$$

Bibliography

[1] P. A. Beardsley. *Applications of projective geometry to robot vision*. PhD thesis, Oxford University, 1992.

[2] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential update of projective and affine structure from motion. Technical Report OUEL 2012/94, Dept of Eng Science, University of Oxford, 1994.

[3] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential update of projective and affine structure from motion. *International Journal of Computer Vision*, 23(3):235–259, 1997.

[4] D.A. Belsley. *Conditioning Diagnostics: Collinearity and weak data in regression*. Wiley, 1991.

[5] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE PAMI*, vol.20(4):401–405, 1998.

[6] F. Bookstein. Fitting conic sections to scattered data. *Computer Vision Graphics and Image Processing*, 9:56–71, 1979.

[7] M. Brooks, L. De Agapito, D. Huynh, and L. Baumela. Towards robust metric reconstruction via a dynamic uncalibrated stereo head. *Image and Vision Computing*, 16(14):989–1002, 1998.

[8] S. Chaterjee and A. S. Hadi. *Sensitivity Analysis in Linear Regression*. John Wiley, New York, 1988.

[9] W. Chojnacki, M. J. Brooks, A. van den Hengel, and D. Gawley. On the fitting of surfaces to data with covariances. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(2):1294–1303, 2000.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc.*, 39 B:1–38, 1977.

[11] R. Deriche, Z. Zhang, Q. T. Luong, and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In J. O. Ecklundh, editor, *Proc. 3rd European Conference on Computer Vision, LNCS 800/801, Stockholm*, pages 567–576. Springer-Verlag, 1994.

[12] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proc. 2nd European Conference on Computer Vision, LNCS 588, Santa Margherita Ligure*, pages 563–578. Springer–Verlag, 1992.

[13] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993.

[14] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, vol. 24:381–95, 1981.

[15] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999.

[16] G.H. Golub and R. Underwood. Stationary values of the ratio of quadratic forms subject to linear constraints. *Z. Angew. Math. Phys.*, 21:318–326, 1970.

[17] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1989.

[18] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Conf.*, pages 189–192, 1987.

[19] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In G. Sandini, editor, *Proc. 2nd European Conference on Computer Vision, LNCS 588, Santa Margherita Ligure*, pages 579–87. Springer–Verlag, 1992.

[20] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. 2nd European Conference on Computer Vision, LNCS 588, Santa Margherita Ligure*, pages 579–587. Springer-Verlag, 1992.

[21] R. I. Hartley. Cheirality invariants. In *Proc. DARPA Image Understanding Workshop*, pages 745–753, 1993.

[22] R. I. Hartley. In defence of the 8-point algorithm. In *Proc. 5th Int'l Conf. on Computer Vision, Boston*, pages 1064–1075, 1995.

[23] R. I. Hartley and P. Sturm. Triangulation. In *DARPA Image Understanding Workshop, Monterey, CA*, pages 957–966, 1994.

[24] R. I. Hartley and P. Sturm. Triangulation. In *American Image Understanding Workshop*, pages 957–966, 1994.

[25] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[26] P. J. Huber. Projection pursuit. *Annals of Statistics*, 13:433–475, 1985.

[27] K. Kanatani. *Geometric Computation for Machine Vision*. Oxford University Press, Oxford, 1992.

[28] K. Kanatani. Renormalization for unbiased estimation. In *Proc. 4th Int'l Conf. on Computer Vision, Berlin*, pages 599–606, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[29] K. Kanatani. Automatic singularity test for motion analysis by an information criterion. In *Proc. 4th European Conference on Computer Vision, LNCS 1064, Cambridge*, pages 697–708, Springer–Verlag, 1996. Buxton, B. and Cipolla R.

[30] M. G. Kendall and A. Stuart. *The Advanced Theory of Statistics*. Charles Griffin and Company, London, 1983.

[31] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, vol.293:133–135, 1981.

[32] Q. T. Luong, R. Deriche, O. D. Faugeras, and T. Papadopoulo. On determining the fundamental matrix: analysis of different methods and experimental results. Technical Report 1894, INRIA (Sophia Antipolis), 1993.

[33] S.J. Maybank. *Theory of Reconstruction From Image Motion*. Springer-Verlag, Berlin, 1993.

[34] G.I. McLachlan and K. Basford. *Mixture models: inference and applications to clustering*. Marcel Dekker. New York, 1988.

[35] J. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[36] S. I. Olsen. Epipolar line estimation. In G. Sandini, editor, *Proc. 2nd European Conference on Computer Vision, LNCS 588, Santa Margherita Ligure*, pages 307–311. Springer–Verlag, 1992.

[37] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philos. Mag. Ser. 6*, 2:559, 1901.

[38] V. Pratt. Direct least squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–152, 1987.

[39] J.O. Rawlings. *Applied Regression Analysis*. Wadsworth and Brooks, California, 1988.

[40] P. J. Rousseeuw. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[41] P.D. Sampson. Fitting conic sections to 'very scattered' data: An iterative refinement of the Bookstein algorithm. *Computer Vision, Graphics, and Image Processing*, 18:97–108, 1982.

[42] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1):7–42, 2002. Evaluation page http://www.middlebury.edu/stereo/eval/.

[43] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, Great Britain, 1952.

[44] L. S. Shapiro. *Affine Analysis of Image Sequences*. Cambridge University Press, Cambridge, England, 1995.

[45] L.S. Shapiro and J.M. Brady. Rejecting outliers and estimating errors in an orthogonal regression framework. Oxford Tech Report OUEL 1974/93, 1993.

[46] C. V. Stewart. Bias in robust estimation caused by discontinuities and multiple structures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-19,no.8:818–833, 1997.

[47] P. Sturm. On focal length calibration from two views. pages 145–150, booktitle =.

[48] R. A. Thisted. *Elements of Statistical Computing*. Chapman and Hall, New York, 1988.

[49] P. H. S. Torr. *Outlier Detection and Motion Segmentation*. PhD thesis, Dept. of Engineering Science, University of Oxford, 1995.

[50] P. H. S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *IJCV*, ?:?, 2002.

[51] P. H. S. Torr, P. A. Beardsley, and D. W. Murray. Robust vision. In J. Illingworth, editor, *Proc. 5th British Machine Vision Conference, York*, pages 145–155. BMVA Press, 1994.

[52] P. H. S. Torr and D. W. Murray. Outlier detection and motion segmentation. In P. S. Schenker, editor, *Sensor Fusion VI*, pages 432–443. SPIE volume 2059, 1993. Boston.

[53] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *IJCV*, 24(3):271–300, 1997.

[54] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *Int Journal of Computer Vision*, 24(3):271–300, 1997.

[55] P. H. S. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In U Desai, editor, *ICCV6*, pages 727–732. Narosa Publishing House, 1998.

[56] P. H. S. Torr, A Zisserman, and S. Maybank. Robust detection of degenerate configurations for the fundamental matrix. *CVIU*, 71(3):312–333, 1998.

[57] P.H.S. Torr. An assessment of information criteria for motion model selection. In *CVPR97*, pages 47–53, 1997.

[58] R.Y. Tsai and T.S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. 6:13–27, 1984.

[59] S. A. Teukolsky W. H. Press, B. P. Flannery and W. T. Vetterling. *Numerical Recipes in C, the art of scientific computing*. Cambridge University Press, Cambridge, 1988.

[60] J. Weng, N. Ahuja, and T. Huang. Optimal motion and structure estimation. *IEEE PAMI*, vol.15(9):864–884, 1993.

[61] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. 11:451–476, 1989.

[62] M. Werman and D. Keren. A bayesian method for fitting parametric and nonparametric model to noisy data. *IEEE PAMI*, vol.23(5):528–534, 2001.

[63] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition*. 1996.

[64] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *IJCV*, 27(2):161–195, 1997.

Index