Target Position and Trajectory Measurements by Videogrammetry

College of Aeronautics Report 0208

ISBN 1 861 941 048

Dr. Stephen Hobbs
School of Engineering
Cranfield University
Cranfield
Bedford MK43 0AL
s.e.hobbs@cranfield.ac.uk

10 Nov 2003

Target Position and Trajectory Measurements by Videogrammetry

College of Aeronautics report 0208
ISBN 1 861 941 048

Stephen E. Hobbs, Cranfield University, UK

Abstract

This report documents the algorithms, data processing and software for the video photogrammetry ("videogrammetry") system developed at Cranfield University. Cranfield's system has been used successfully since 1999 on a range of measurement projects. Videogrammetry typically uses two video cameras to film the motion of target objects in stereo, and then with suitable image processing and data analysis the targets' 3d trajectories are measured to good precision.

The main features of the Cranfield systems are that it is based on consumer electronics devices (e.g. digital camcorders and PC's), and that it is designed as an experimental tool. Using consumer electronics provides good performance at low cost. Its experimental character means that an expert user is required, but does allow great flexibility. Current system performance derives from the image resolution of 1 mrad per pixel over a field of view 720 by 576 pixels, and a frame rate of 25 Hz.

Two areas of work are described: (1) the mathematical models and algorithms used for calibration, position measurement and trajectory extraction, and (2) the software tools written to manipulate images and process the data. The model of the imaging system can be adapted for a wide range of applications, and is explicitly developed in this report for a single camera position and pose calibration, a two-camera system calibration, and a measurement system using two or more cameras (either the general non-linear case or a linear approximation). The image calibration (which converts image coordinates to geometrical angles of inclination and azimuth) is based on a 3rd order polynomial and achieves an accuracy equivalent to better than 1 pixel. Trajectories are obtained by either labelling targets or using kinematic rules. Two programs (`AVI1` and `mfitvid`) have been written for the videogrammetry system and are described in outline (for a user and to support future development). Commercial software provides the more general functions required by the system.

Experience with the videogrammetry system over a number of years gives confidence in its performance. Example results are provided to illustrate the type of measurements which are possible.

Contents

List of Figures

List of Tables

Notation

$\mathbf{a}$	position of midpoint of camera baseline
$\mathbf{a}_i$	position of camera i
A_{ik}	The numerator of the basic imaging equation for camera i and camera axis k
$\mathbf{b}$	the camera baseline (vector from camera 2 to camera 1)
b	length of the camera baseline
D_i	The denominator of the basic imaging equation for camera i
$\mathbf{e}_{ij}$	direction of camera i's jth unit axis
$\mathbf{p}$	position of the target being imaged
$\mathbf{p_0}$	position of a reference point in the measurement volume
$\mathbf{p'}$	position of the target relative to $\mathbf{p}_0$
α_i	inclination of the target measured by camera i
α_{i0}	inclination of the reference point for camera i
β_i	azimuth of the target measured by camera i
β_{i0}	azimuth of the reference point for camera i
ϕ_b	azimuth of the camera baseline
ϕ_i	azimuth of the view axis of camera i
θ_b	inclination of the camera baseline
θ_i	inclination of the view axis of camera i
ψ_i	rotation angle of azimuth direction for camera i about the view axis (from the plane containing the view axis and the vertical)

Table 1: Symbols used in the report.

Chapter 1

Introduction

Video photogrammetry ("videogrammetry") is a practical method of measuring the trajectory of a target in three dimensions and can be implemented using consumer electronics equipment which provide high performance at low cost. This report describes work on videogrammetry at Cranfield University, and in particular documents the algorithms and software used to calibrate the system, solve for 3D position of targets, and identify trajectories from a series of target position measurements for multiple targets.

The method reported here has been in use since 1999 at Cranfield for applications which include monitoring the wind-driven motion of vegetation, air traffic movements at Cranfield airport, suspended target motion, human locomotion, and trajectories of airflow tracers in wind tunnels. This experience gives us a reasonable degree of confidence in the techniques and their suitability as a research tool.

One function of this report is to act as a reference for the algorithms developed and thus they are described in full algebraic detail.

The fundamental limitations (e.g. measurement precision and rate) are determined largely by basic imaging geometry and the performance of the camera (including the effect of any data compression implemented). Digital camcorders for consumer use typically have frame rates of 25 Hz (in Europe) and use the DV compression format (nominal frame size of 720 columns by 576 rows, with the image encoded using a digital cosine transform algorithm - similar to the JPEG format). A typical shortest focal length setting for the lens (maximum wide angle) corresponds to an angular "resolution" of 1 mrad per pixel.

1.1 Examples of Videogrammetry Applications

Sample results from projects where videogrammetry has been applied demonstrate the system's scope and capability. The algorithms presented here are designed to track multiple targets simultaneously without needing extensive user manipulation of the data.

Figure 1.1 shows measurements of the motion in the direction parallel to the mean wind velocity of a wheat plant over a period of 1 minute. Several labels were placed on the plants and then their motion was tracked while they moved in the wind. These data allowed the effect of wind driven motion on

vegetation to be quantified for the first time. In general, videogrammetry allows the dynamics of motion to be analysed numerically for objects which are often difficult to measure by other methods.

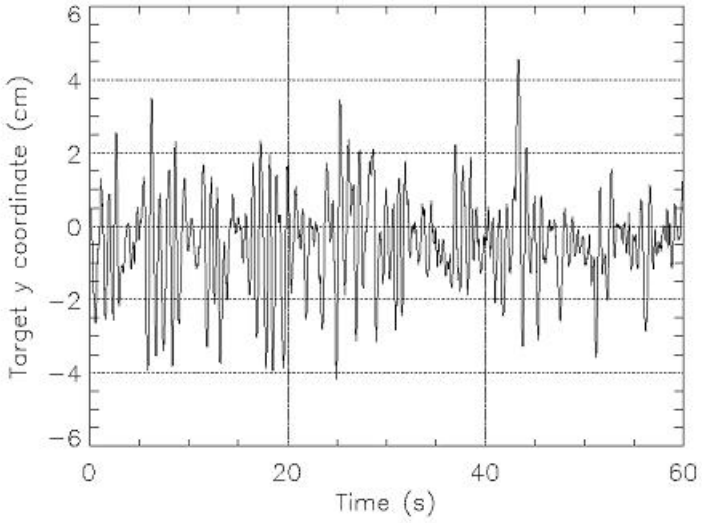


Figure 1.1: Example of motion measured for part of a wheat plant over a period of 1 minute [4].

Figure 1.2 shows the trajectories of a pair of bubbles used as airflow tracers in a vortex generated inside a wind tunnel. This quantifies observations which would otherwise only be available qualitatively. The figure also shows how it is possible to visualise the motion afterwards based on the measurements: this can be very helpful for analysing particular features of the motion.

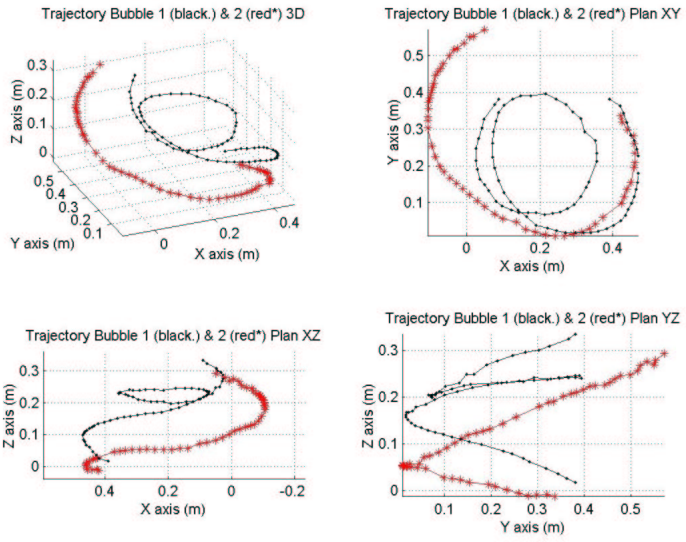


Figure 1.2: Various projections of trajectories measured for two bubbles inside a low-speed vertical vortex [2].

Chapter 2

Model of the Video System

The video system typically comprises two cameras viewing a measurement volume from different directions. This should allow the three-dimensional position of targets in the volume to be measured. The model developed in this chapter is based on a model of the geometry for a single camera which is then extended for the case of two (or more) cameras. Figure 2.1 summarises the system geometry and variables for a single camera.

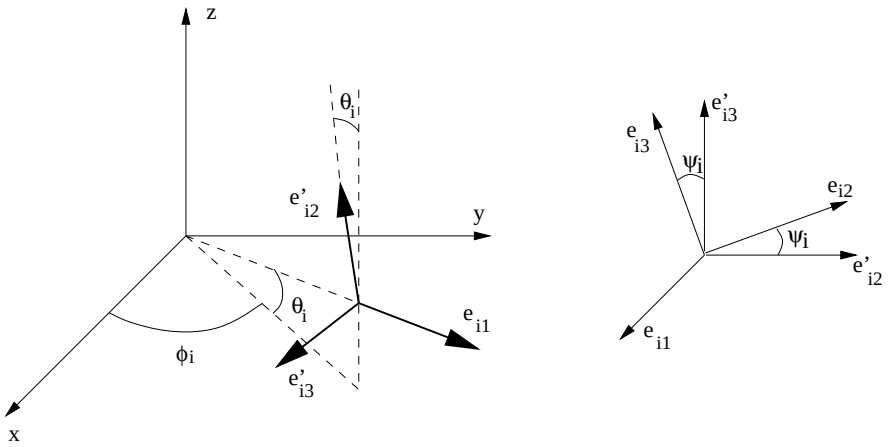


Figure 2.1: Video axes and angles for the imaging system model (camera i). e_{ij} are the camera axes, and $e\prime_{i2}$ and $e\prime_{i3}$ are intermediate vectors (corresponding to a camera with no "roll" about its viewing axis e_{i1}). The xyz coordinate system is that defined by the (position and pose) calibration targets.

The model of the video system is geometrical, and is expressed using vectors since this is a convenient notation. The vectors should be expressed using a coordinate system for the measurement system. This system can be chosen for experimental or calibration convenience. Note that the model developed in this chapter assumes that the camera image coordinates (row, column) can be translated into angles of inclination and azimuth using a calibration function (this function is developed separately in Appendix A for the cameras used for the experiments in 2000).

2.1 Camera Image Calibration

Underlying the models developed later in this report is a calibration of the image which converts image coordinates (e.g. column and row number) to geometrical viewing angles relative to the camera axes. Because of the form of the model developed, it turns out that the calibration functions are most useful if given in terms of $\tan\alpha$ and $\tan\beta$ rather than α and β directly (α and β are the inclination and azimuth angles respectively of the imaged point relative to the camera's viewing axes). Appendix A describes the camera image calibration method used and gives results for two cameras.

The calibrations are expressed as coefficients (a_i or b_i) of polynomials giving the angles (inclination or azimuth) as a function of image coordinate (row or column number relative to the image centre).

$$\begin{aligned}\tan\alpha &= a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \\ &\quad + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 \qquad (2.1)\\ \tan\beta &= b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \\ &\quad + b_6x^3 + b_7x^2y + b_8xy^2 + b_9y^3 \qquad (2.2)\end{aligned}$$

2.2 The Basic Model

The target is at $\mathbf{p}$, and the two cameras are at $\mathbf{a}_1$ and $\mathbf{a}_2$. Each camera (label $i = 1, 2$) has three orthogonal unit vectors defined for the viewing axis ($\mathbf{e}_{i1}$), the inclination (row) direction ($\mathbf{e}_{i2}$) and the azimuth (column) direction ($\mathbf{e}_{i3}$). Each camera measures the inclination and azimuth angles of the target in its own coordinate system. The unit axes for each camera can be expressed in terms of three angles relative to the measurement system's coordinate system; these angles are the inclination (θ_i) and azimuth (ϕ_i) of the viewing axis, and the roll angle (ψ_i) about the viewing axis of the camera's azimuth direction from the plane containing the view axis and the vertical. In terms of these angles the unit vectors for camera i are,

$$\begin{aligned}\mathbf{e}_{i1} &= (\cos\theta_i\cos\phi_i, \cos\theta_i\sin\phi_i,\ \sin\theta_i) \qquad (2.3)\\ \mathbf{e}_{i2} &= (-\sin\theta_i\cos\phi_i\cos\psi_i + \sin\phi_i\sin\psi_i, \\ &\quad -\sin\theta_i\sin\phi_i\cos\psi_i - \cos\phi_i\sin\psi_i, \cos\theta_i\cos\psi_i) \qquad (2.4)\\ \mathbf{e}_{i3} &= (\sin\theta_i\cos\phi_i\sin\psi_i + \sin\phi_i\cos\psi_i, \\ &\quad \sin\theta_i\sin\phi_i\sin\psi_i - \cos\phi_i\cos\psi_i, -\cos\theta_i\sin\psi_i) \qquad (2.5)\end{aligned}$$

Using these camera axes, the inclination (α_i) and azimuth (β_i) of the target as measured by camera i are,

$$\begin{aligned}\tan\alpha_i &= \frac{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i2}}{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i1}} \qquad (2.6)\\ \tan\beta_i &= \frac{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i3}}{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i1}} \qquad (2.7)\end{aligned}$$

Note that it may be more convenient in general to calibrate the cameras to give $\tan\alpha, \tan\beta$ as the measurements of target position instead of the angles directly.

The camera positions can be expressed in the model as shown here or using the baseline between the cameras and the vector from the midpoint of the baseline to the target. This second method is appropriate if a direct measurement of the baseline length is available, since a direct measurement may be more accurate than an estimate obtained by inverting the system model for a set of calibration points. In terms of the baseline the camera positions are

$$\mathbf{a}_1 = \mathbf{a} + \mathbf{b}/2 \tag{2.8}$$
$$\mathbf{a}_2 = \mathbf{a} - \mathbf{b}/2 \tag{2.9}$$

Thus

$$\mathbf{a} = \frac{1}{2}(\mathbf{a}_1 + \mathbf{a}_2) \tag{2.10}$$
$$\mathbf{b} = \mathbf{a}_1 - \mathbf{b}_2 \tag{2.11}$$

The baseline vector should be expressed in a form explicitly including its magnitude,

$$\mathbf{b} = b(\cos\theta_b \cos\phi_b, \cos\theta_b \sin\phi_b, \ \sin\theta_b) \tag{2.12}$$

If this formulation is used then each camera's position depends on six parameters and not just three as before. However, because b can be measured directly this should give a slightly more accurate system calibration.

2.2.1 Derivation of Camera Axes

The camera axes given above are based on the following two intermediate unit vectors $(\mathbf{e\prime}_{i2}, \mathbf{e\prime}_{i3})$ defined by the view axis and the measurement system's vertical axis. $\mathbf{e\prime}_{i2}$ is the intersection of the plane perpendicular to the view axis and the plane containing the view axis and the measurement system's vertical axis; $\mathbf{e\prime}_{i3}$ is perpendicular to $\mathbf{e}_{i1}$ and $\mathbf{e\prime}_{i2}$.

$$\mathbf{e\prime}_{i2} = (-\sin\theta_i \cos\phi_i, -\sin\theta_i \sin\phi_i, \ \cos\theta_i) \tag{2.13}$$
$$\mathbf{e\prime}_{i3} = (\sin\phi_i, -\cos\phi_i, 0) \tag{2.14}$$

$\mathbf{e}_{i2}$ and $\mathbf{e}_{i3}$ are obtained by rotating the primed axes by ψ about the view axis.

Chapter 3

Model Application and Inversion

Several means of applying the basic video model are used.

- System calibration. The model is inverted using measurements made for a set of calibration targets to estimate the camera positions and orientations.
- Position measurement. Using known camera positions and orientations the model is inverted to estimate target position from a set of measurements (e.g. using two cameras to find the target's position in three dimensions).
- More general use. The basic equation could be used for any imaging situation where camera image coordinates (expressed as angles) need to be related to geometrical position of the target relative to the camera and its attitude.

Model inversion for calibration or measurement is a non-linear problem in general (a linearised form of the equations is presented in the previous chapter). A standard way to invert a non-linear model is by maximum likelihood fitting of the model to some measured data [3]. The Levenburg-Marquardt [3] method used at Cranfield requires analytical expressions for the model, its derivatives with respect to all the model parameters, and a good first guess for the parameters to be estimated. The next section lists all relevant model derivatives.

3.1 Camera Position and Pose Calibration

The model developed in chapter 2 is a "forward" model, which gives the image coordinates of a target as a function of the camera position and pose. For the camera position and pose calibration, position and pose have to be estimated from measurements of image coordinates for a reference target - this is done by inverting the forward model.

Several methods of model inversion are possible, and the one used here is based on a non-linear least squares iterative minimisation procedure, Levenburg-Marquardt minimisation [3]. To implement this algorithm requires definition of a model (as is done in chapter 2) and its partial derivatives with respect to

all the model parameters. The next section gives the model and the partial derivatives required.

3.1.1 Model Derivatives

The basic model equations for a single-camera system can be written

$$\begin{aligned} \tan\alpha_i &= \frac{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i2}}{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i1}} \\ &= \frac{A_{2i}}{D_i} \qquad (3.1) \\ \tan\beta_i &= \frac{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i3}}{(\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i1}} \\ &= \frac{A_{3i}}{D_i} \qquad (3.2) \end{aligned}$$

where

$$\begin{aligned} A_{ki} &= (\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{ik} \qquad (3.3) \\ D_i &= (\mathbf{p}-\mathbf{a}_i)\,.\mathbf{e}_{i1} \qquad (3.4) \end{aligned}$$

Writing $y_{i2} = \tan\alpha_i$ and $y_{i3} = \tan\beta_i$, the derivatives of $y_{ik}(k = 2, 3)$ with respect to each of the model parameters for camera i are

$$\begin{aligned} \frac{\partial y_{ik}}{\partial p_j} &= \frac{e_{ikj}}{D_i} - \frac{A_{ki}}{D_i^2}(e_{i1j}) \qquad (3.5) \\ \frac{\partial y_{ik}}{\partial a_{ij}} &= -\frac{e_{ikj}}{D_i} + \frac{A_{ki}}{D_i^2}(e_{i1j}) \qquad (3.6) \\ \frac{\partial y_{ik}}{\partial \theta} &= \sum_{j=1}^{3}\left[\frac{(\mathbf{p}-\mathbf{a}_i)_j}{D_i}\,\frac{\partial e_{ikj}}{\partial\theta} - \frac{A_{ki}}{D_i^2}(\mathbf{p}-\mathbf{a}_i)_j\frac{\partial e_{i1j}}{\partial\theta}\right] \qquad (3.7) \\ \frac{\partial y_{ik}}{\partial \phi} &= \sum_{j=1}^{3}\left[\frac{(\mathbf{p}-\mathbf{a}_i)_j}{D_i}\,\frac{\partial e_{ikj}}{\partial\phi} - \frac{A_{ki}}{D_i^2}(\mathbf{p}-\mathbf{a}_i)_j\frac{\partial e_{i1j}}{\partial\phi}\right] \qquad (3.8) \\ \frac{\partial y_{ik}}{\partial \psi} &= \sum_{j=1}^{3}\left[\frac{(\mathbf{p}-\mathbf{a}_i)_j}{D_i}\,\frac{\partial e_{ikj}}{\partial\psi} - \frac{A_{ki}}{D_i^2}(\mathbf{p}-\mathbf{a}_i)_j\frac{\partial e_{i1j}}{\partial\psi}\right] \qquad (3.9) \end{aligned}$$

If the cameras positions are expressed in terms of the two-camera baseline $\mathbf{a}, \mathbf{b}$ then the derivatives are

$$\begin{aligned} \frac{\partial y_{ik}}{\partial a_j} &= \sum_{l=1}^{3}\frac{\partial y_{ik}}{\partial a_{il}}\,\frac{\partial a_{il}}{\partial a_j} \qquad (3.10) \\ \frac{\partial y_{ik}}{\partial b} &= \sum_{l=1}^{3}\frac{\partial y_{ik}}{\partial a_{il}}\,\frac{\partial a_{il}}{\partial b} \qquad (3.11) \end{aligned}$$

$$\frac{\partial y_{ik}}{\partial \theta_b} = \sum_{l=1}^{3} \frac{\partial y_{ik}}{\partial a_{il}} \frac{\partial a_{il}}{\partial \theta_b} \tag{3.12}$$

$$\frac{\partial y_{ik}}{\partial \phi_b} = \sum_{l=1}^{3} \frac{\partial y_{ik}}{\partial a_{il}} \frac{\partial a_{il}}{\partial \phi_b} \tag{3.13}$$

Note that the attitude angles θ, ϕ, ψ are those for camera i, i.e. the angles should be written θ_i, ϕ_i, ψ_i. The derivatives of the camera's direction vectors (again omitting the subscript i to indicate the camera) are

$$\frac{\partial e_{i11}}{\partial \theta} = -\sin\theta\cos\phi \tag{3.14}$$

$$\frac{\partial e_{i12}}{\partial \theta} = -\sin\theta\sin\phi \tag{3.15}$$

$$\frac{\partial e_{i13}}{\partial \theta} = \cos\theta \tag{3.16}$$

$$\frac{\partial e_{i11}}{\partial \phi} = -\cos\theta\sin\phi \tag{3.17}$$

$$\frac{\partial e_{i12}}{\partial \phi} = \cos\theta\cos\phi \tag{3.18}$$

$$\frac{\partial e_{i13}}{\partial \phi} = 0 \tag{3.19}$$

$$\frac{\partial e_{i11}}{\partial \psi} = 0 \tag{3.20}$$

$$\frac{\partial e_{i12}}{\partial \psi} = 0 \tag{3.21}$$

$$\frac{\partial e_{i13}}{\partial \psi} = 0 \tag{3.22}$$

$$\frac{\partial e_{i21}}{\partial \theta} = -\cos\theta\cos\phi\cos\psi \tag{3.23}$$

$$\frac{\partial e_{i22}}{\partial \theta} = -\cos\theta\sin\phi\cos\psi \tag{3.24}$$

$$\frac{\partial e_{i23}}{\partial \theta} = -\sin\theta\cos\psi \tag{3.25}$$

$$\frac{\partial e_{i21}}{\partial \phi} = \sin\theta\sin\phi\cos\psi + \cos\phi\sin\psi \tag{3.26}$$

$$\frac{\partial e_{i22}}{\partial \phi} = -\sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi \tag{3.27}$$

$$\frac{\partial e_{i23}}{\partial \phi} = 0 \tag{3.28}$$

$$\frac{\partial e_{i21}}{\partial \psi} = \sin\theta\cos\phi\sin\psi + \sin\phi\cos\psi \tag{3.29}$$

$$\frac{\partial e_{i22}}{\partial \psi} = \sin\theta\sin\phi\sin\psi - \cos\phi\cos\psi \tag{3.30}$$

$$\frac{\partial e_{i23}}{\partial \psi} = -\cos\theta\sin\psi \tag{3.31}$$

$$\frac{\partial e_{i31}}{\partial \theta} = \cos\theta\cos\phi\sin\psi \tag{3.32}$$
$$\frac{\partial e_{i32}}{\partial \theta} = \cos\theta\sin\phi\sin\psi \tag{3.33}$$
$$\frac{\partial e_{i33}}{\partial \theta} = \sin\theta\sin\psi \tag{3.34}$$
$$\frac{\partial e_{i31}}{\partial \phi} = -\sin\theta\sin\phi\sin\psi + \cos\phi\cos\psi \tag{3.35}$$
$$\frac{\partial e_{i32}}{\partial \phi} = \sin\theta\cos\phi\sin\psi + \sin\phi\cos\psi \tag{3.36}$$
$$\frac{\partial e_{i33}}{\partial \phi} = 0 \tag{3.37}$$
$$\frac{\partial e_{i31}}{\partial \psi} = \sin\theta\cos\phi\cos\psi - \sin\phi\sin\psi \tag{3.38}$$
$$\frac{\partial e_{i32}}{\partial \psi} = \sin\theta\sin\phi\cos\psi + \cos\phi\sin\psi \tag{3.39}$$
$$\frac{\partial e_{i33}}{\partial \psi} = -\cos\theta\cos\psi \tag{3.40}$$

If the camera positions of a two-camera system are written in terms of the mean camera position (**a**) and the baseline (**b**) then the derivatives for camera 1 are

$$\frac{\partial a_{ij}}{\partial a_j} = 1 \tag{3.41}$$
$$\frac{\partial a_{11}}{\partial b} = \frac{1}{2}\cos\theta\cos\phi \tag{3.42}$$
$$\frac{\partial a_{12}}{\partial b} = \frac{1}{2}\cos\theta\sin\phi \tag{3.43}$$
$$\frac{\partial a_{13}}{\partial b} = \frac{1}{2}\sin\theta \tag{3.44}$$
$$\frac{\partial a_{11}}{\partial \theta} = -\frac{b}{2}\sin\theta\cos\phi \tag{3.45}$$
$$\frac{\partial a_{12}}{\partial \theta} = -\frac{b}{2}\sin\theta\sin\phi \tag{3.46}$$
$$\frac{\partial a_{13}}{\partial \theta} = \frac{b}{2}\cos\theta \tag{3.47}$$
$$\frac{\partial a_{11}}{\partial \phi} = -\frac{b}{2}\cos\theta\sin\phi \tag{3.48}$$
$$\frac{\partial a_{12}}{\partial \phi} = \frac{b}{2}\cos\theta\cos\phi \tag{3.49}$$
$$\frac{\partial a_{13}}{\partial \phi} = 0 \tag{3.50}$$

The derivatives for camera 2 are the same as those for camera 1 except that all the derivatives with respect to the baseline change sign (i.e. $\frac{\partial a_{21}}{\partial b} = -\frac{\partial a_{11}}{\partial b}$, etc.). The angles θ, ϕ used here are the camera baseline orientation angles (θ_b, ϕ_b).

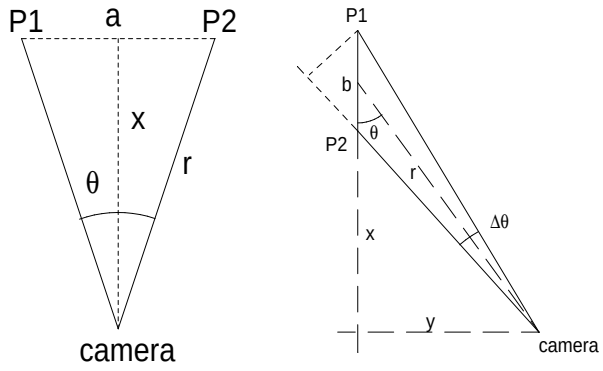
3.1.2 First-Guess Values

Iterative solution methods must generally be initialised with good first-guesses of the desired solution if the solution is to be efficient and to avoid secondary minima.

For calibration, the model parameters are best estimated using a crude direct measurement of the camera positions and attitudes.

3.1.3 Camera Parameter Measurement Uncertainty

The following sections provide simple relationships between the measurement accuracy of the camera and the uncertainty in the estimate of the camera position or attitude parameters (the approximations used are correct only to first order). The following figures illustrate the geometry assumed to derive these simple estimates of measurement uncertainty.



(a) Range error (b) Lateral errors

Figure 3.1: Symbols used in the derivation of the first order models for measurement uncertainty.

Radial distance from calibration targets

Consider a camera at a perpendicular distance x from two calibration targets which are distance a apart laterally (r is the slant range to one target) . θ is the angle between the lines of sight from the camera to the targets.

$$\tan\theta/2 = \frac{a}{2x} \tag{3.51}$$

$$x = \frac{a}{2\tan\theta/2} \tag{3.52}$$

$$\frac{dx}{d\theta} = \frac{-a}{4\sin^2\theta/2} \tag{3.53}$$

$$\sin\theta/2 = \frac{a}{2r} \tag{3.54}$$

$$\delta x \simeq \frac{dx}{d\theta}\delta\theta \tag{3.55}$$

$$= \frac{-r^2}{a}\delta\theta \tag{3.56}$$

This gives the uncertainty in x due to errors in measuring θ.

Lateral displacement wrt calibration targets

Consider a camera viewing two targets which are b apart along the line of sight, with the camera distance x from the midpoint. The angle subtended by the two points at the camera is

$$\Delta\theta = \frac{b\sin\theta}{r} \tag{3.57}$$

$$= \frac{by}{r^2} = \frac{by}{x^2+y^2} \tag{3.58}$$

$$\frac{d\Delta\theta}{dy} = b\frac{x^2-y^2}{(x^2+y^2)^2} \tag{3.59}$$

$$\simeq \frac{b}{x^2} \text{ assuming } x \gg y \tag{3.60}$$

thus

$$\delta y, \delta z \simeq \frac{x^2}{b}\delta\Delta\theta \tag{3.61}$$

This is the uncertainty in measuring y, z (the lateral position of the camera) due to errors in $\Delta\theta$. Note that the lateral separation of the targets leads only to a second order change in $\Delta\theta$.

Camera inclination and azimuth

The absolute position uncertainties can be converted directly to angular uncertainties for the camera's absolute inclination and azimuth by dividing the lateral position uncertainty by the range to the calibration targets.

$$\delta\theta, \delta\phi = \frac{\delta z, \delta y}{r} \simeq \frac{x^2}{br}\delta\Delta\theta \simeq \frac{x}{b}\delta\Delta\theta \tag{3.62}$$

Note that in general the pointing uncertainty due to position errors is greater than the direct angular resolution of the cameras.

Camera rotation angle

The uncertainty in the camera's rotation angle (about its line of sight) is given by the camera's spatial resolution at the target divided by the linear separation of the targets. Consider two targets (distance a apart) viewed in the image plane with the camera a distance r from this plane.

$$\delta\psi \simeq \frac{r\delta\theta}{a} \tag{3.63}$$

3.2 Target Position Measurement

If the camera positions and poses are known and the image has been calibrated, then it is possible to calculate the position of the target from measurements of the target's positions in the camera images available. The methods described below assume two images are available, although both methods can be generalised to accept measurements from more than two cameras. Two solution methods have been implemented:

- a Linearised model
- the Epi-polar line method

The linearised model requires an approximate solution for the target position (about which the equations are linearised), and then the position can be calculated in one step using linear algebra (a matrix multiplication). To use this technique, a separate method of identifying pairs of points from the two images must be implemented, i.e. the coordinates corresponding to the same target but viewed in the two different cameras.

The epi-polar line method is less direct, but has the advantage of automatically identifying pairs of points from the two images available even when many targets are visible by both cameras.

3.2.1 Linearised Model

A linearised version of the model is useful in many measurement situations and can be inverted directly to give target position as a linear function of the camera measurements.

The assumptions (which are more or less equivalent) made to linearise the equations are

1. The inclination and azimuth angles of the target relative to the cameras axes are small (so small angle approximations can be used although this is not really necessary).
2. The target position ($\mathbf{p}$) is close to a reference position ($\mathbf{p}_0$) which lies in the measurement volume close to the view axis of both (all) cameras (close is defined relative to the distance from the camera to the target).

The first approximation need not be used to approximate $\tan\alpha$ by α if the camera is calibrated in terms of $\tan\alpha$ rather than α. The target position can be written $\mathbf{p} = \mathbf{p}_0 + \mathbf{p}\prime$, and then the system equations become,

$$\tan\alpha_i = \frac{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i2}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.64}$$

$$= \frac{\mathbf{p}\prime.\mathbf{e}_{i2}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} + \frac{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i2}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.65}$$

$$\tan\beta_i = \frac{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i3}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.66}$$

$$= \frac{\mathbf{p}\prime.\mathbf{e}_{i3}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} + \frac{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i3}}{(\mathbf{p}_0 + \mathbf{p}\prime - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.67}$$

If the assumptions hold then $\mathbf{p}\prime$ is much smaller than $\mathbf{p}_0 - \mathbf{a}_i$ and can be ignored in the denominator. The constants (corresponding to the angular position of the reference point $\mathbf{p}_0$) can be written as offset angles giving,

$$\tan\alpha_i - \tan\alpha_{i0} \simeq \frac{\mathbf{p}\prime.\mathbf{e}_{i2}}{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.68}$$

$$\tan\alpha_{i0} = \frac{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i2}}{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.69}$$

$$\tan\beta_i - \tan\beta_{i0} \simeq \frac{\mathbf{p}\prime.\mathbf{e}_{i3}}{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.70}$$

$$\tan\beta_{i0} = \frac{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i3}}{(\mathbf{p}_0 - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.71}$$

These equations (for two or more cameras) have the form

$$\mathbf{b} = A\mathbf{p}\prime \tag{3.72}$$

$$\mathbf{p}\prime = A^{-1}\mathbf{b} \tag{3.73}$$

where $\mathbf{b}$ represents the measurements made using the cameras (and an offset which depends on calibration constants and the arbitrary reference position $\mathbf{p}_0$) and A is a matrix determined by the measurement system geometry. This equation can be inverted using singular value decomposition techniques [3] to give target position ($\mathbf{p}\prime$) in terms of the target's angular position measured by the cameras (α_i, β_i).

3.2.2 Epi-polar Line Method

The epi-polar line technique uses the fact that a point target detected in one image (image 1) can in general line anywhere along a line across the image from the other camera (image 2). The coordinates of this line can be calculated and then matches between the line and detected targets in image 2 are sought (Figure 3.2).

Camera i has unit vectors $\mathbf{e}_{i1}$ (viewing axis), $\mathbf{e}_{i2}$ (inclination), and $\mathbf{e}_{i3}$ (azimuth). If in camera 1 (wlog) a point is detected at a given position (column, row) then this corresponds to particular values of inclination ($\tan\alpha_1$) and azimuth ($\tan\beta_1$), so the target must lie somewhere along a line given by

$$\mathbf{p}(\lambda) = \mathbf{a}_1 + \lambda(\mathbf{e}_{11} + \tan\alpha_1\mathbf{e}_{12} + \tan\beta_1\mathbf{e}_{13}) \tag{3.74}$$

$$= \mathbf{a}_1 + \lambda\mathbf{d} \tag{3.75}$$

where

$$\mathbf{d} = \mathbf{e}_{11} + \tan\alpha_1\mathbf{e}_{12} + \tan\beta_1\mathbf{e}_{13} \tag{3.76}$$

For other cameras (i) in the system, this is imaged at a point with inclination and azimuth given by

$$\tan\alpha_i = \frac{(\mathbf{p} - \mathbf{a}_i).\mathbf{e}_{i2}}{(\mathbf{p} - \mathbf{a}_i).\mathbf{e}_{i1}} \tag{3.77}$$

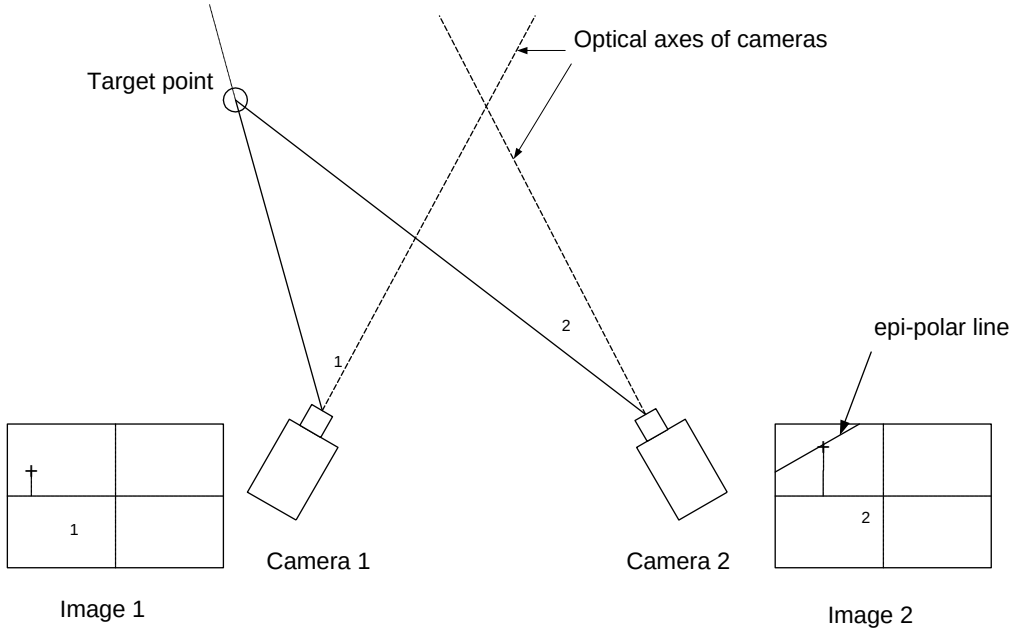


Figure 3.2: Conceptual description of the epi-polar line solution method. A point imaged in camera 1 corresponds, potentially, to a line in the image of camera 2. The closest match between points in camera 2 and the line is assumed to correspond to the correct solution.

$$
\begin{aligned}
& = \frac{(\mathbf{b}_i - \lambda\mathbf{d}).\mathbf{e}_{i2}}{(\mathbf{b}_i - \lambda\mathbf{d}).\mathbf{e}_{i1}} && (3.78)\\
& = \mathbf{f}_{i2}(\lambda) && (3.79)\\
\tan\beta_i & = \frac{(\mathbf{p} - \mathbf{a}_i).\mathbf{e}_{i3}}{(\mathbf{p} - \mathbf{a}_i).\mathbf{e}_{i1}} && (3.80)\\
& = \frac{(\mathbf{b}_i - \lambda\mathbf{d}).\mathbf{e}_{i3}}{(\mathbf{b}_i - \lambda\mathbf{d}).\mathbf{e}_{i1}} && (3.81)\\
& = \mathbf{f}_{i3}(\lambda) && (3.82)
\end{aligned}
$$

$\mathbf{b}_i = \mathbf{a}_1 - \mathbf{a}_i$ is the baseline between camera 1 and camera i.

Thus as the parameter λ (the range to the target from camera 1 along direction $\mathbf{e}_{11}$) varies a line is drawn across the image in camera i; this is the epi-polar line. The solution method is to vary λ between reasonable values (minimum and maximum plausible ranges for example) and to look for the best coincidence between the line and points in the image of camera i. It is assumed that the solution is given by the point lying closest to the epi-polar line.

Note that this method simultaneously finds the position of the target and matches up points from images of the different cameras. The linear solution method does not match up points, but does make more "symmetrical" use of the measurements. The epi-polar line method described here ensures that the target position exactly satisfies the point measurement for camera 1 but may only approximately satisfy measurements for other cameras, and is therefore

asymmetrical in its use of the measurements. An iterative method that used the epi-polar line method to find an approximate solution and match points, and then the linear method to improve the solution, should in principle give a better solution although the improvement may not normally be significant.

3.3 Algorithm Coding

Careful structuring of the way the functions are implemented in software provides more robust and versatile code. Some of the issues to consider are

- The standard code written for camera position and pose calibration implements algorithms from [3], where the model function takes a scalar input and returns a scalar value. In the calibration the inputs and outputs are more naturally treated as vectors so the scalar algorithms are modified as suggested by [3] to handle inputs and outputs which are vectors.
- For a conventional two-camera system with a baseline of about a metre a few metres from the measurement volume, the derivatives should all have approximately the same magnitude and so the numerical algorithms should be well behaved. If the lengths have numerical values much greater than 1 (e.g. several hundred metres) then the derivatives will no longer all be approximately the same magnitude and there may be problems with any matrix inversions required by the algorithms. (Such numerical problems often indicate situations where unrealistic measurements are being attempted.)

Chapter 4

Trajectory Extraction

The algorithms described in chapter 3 give measurements of the positions of targets at the time of each frame pair. For many applications, it is more useful to know the trajectory of individual targets over a period of time. Obtaining trajectories from individual measurements of position is not a trivial step. Several methods could be used: this chapter describes one method (labelling the targets) in outline and a second method (based on kinematic rules) in more detail.

4.1 Trajectory Measurement using Labelled Targets

In principle it is possible to "label" each target uniquely (using the term "label" in a general sense). The label may be by colour, size, correlation mask used for tracking, or region of the image, for instance, and should allow the targets to be uniquely identified. The position measurements can then be sorted using this label, and then the trajectories are obtained as a sequence of positions for each of the uniquely labelled targets.

If only a few targets are being tracked, or it is possible to arrange unique labels, then this method is relatively simple to implement. Early applications of the Cranfield videogrammety system used this method of obtaining trajectories. With many targets this method becomes more difficult to use.

4.2 Trajectory Measurement using Kinematic Rules

An alternative method of measuring trajectories is to use information about the kinematics of the tracked targets (e.g. position, velocity, etc.). This method does not require the targets to be individually labelled or widely separated and is suitable when several or many targets are to be tracked simultaneously. The method has been used in more recent videogrammetry experiments. This technique can be validated by using it for targets which have simple labels (as described above), and then ensuring that each trajectory obtained corresponds to a "target" which always has the same label.

The following three figures illustrate the kinematic rules method of trajectory identification (Appendix C provides details of how the algorithm has been implemented in software).

The starting point (Figure 4.1) is a set of measurements of target positions for each frame pair. So that the method can be visualised in a two dimensional figure, only one spatial coordinate is used in the figures (along the y axis), and the x axis is used for time (frame number). In practice, all three coordinates of the target's position are used to calculate displacements, separations, etc. From each frame pair, there may be several target position measurements: some will be real while others may be spurious noise.

Two steps are used to extract trajectories:

1. Form all plausible links between target positions in adjacent frames.
2. Edit the links to keep only the most likely ones. The remaining links constitute the trajectories sought.

4.2.1 Form all plausible links

The first step is to form all plausible links between positions measured in one frame and those measured in adjacent frames. A "link" is the pairing of a point in one frame with a point in an adjacent frame: each point can in principle belong to several links (both forward and backward in time). "Plausible" here means links that are shorter than a certain limit. A suitable limit can be obtained based on the maximum feasible target speed multiplied by the time between frames, plus a safety margin. Using this limit avoids the creation of unrealistic links which will only be discarded later. Figure 4.2 shows all plausible links for the targets of Figure 4.1, where the maximum link length is 1 (displacement along y axis). Isolated points form no links, and are discarded at this stage.

4.2.2 Edit links

The next step is to edit the links, keeping only the most likely one for any target in each frame. Many different rules could be applied to determine which is the most likely link; possible rules include:

- smallest displacement
- smallest change in acceleration
- most similar "label" statistics

The rule applied in program `AVI1` is currently (program version 0.36) to choose the link corresponding to the smallest displacement. Figure 4.3 shows the links remaining after this filter has been applied. Note that where the two trajectories almost overlap, the multiple links have all been resolved. The links which remain after this filter constitute the measured trajectories.

It is in principle possible for the algorithm's results to depend on the order in which it edits the links, which is unsatisfactory. For example, a given link may be the shortest forward link from, say, point p_n to point q_{n+1}; however this may not be the shortest reverse link from q_{n+1} since the reverse link to r_n

may be even shorter. Hence, searching first with point p will choose the link $p_n - q_{n+1}$ while searching first with point r would preserve $r_n - q_{n+1}$ and delete $p_n - q_{n+1}$. To avoid this problem, a link $p_n - q_{n+1}$ should only be kept to form part of a trajectory if it is simultaneously the shortest forward link from p_n and also the shortest reverse link from q_{n+1}.

4.2.3 Remarks

In practice, because of measurement noise or temporary changes in viewing conditions (e.g. lighting, aspect) it is common for a single real trajectory to only be recorded as fragments. These fragments of trajectories currently have to be edited manually to patch together an estimate of the real continuous trajectory. (This task is significantly easier than creating the trajectories in the first place.) It is possible to automate this partially, using estimates of the target's dynamics (e.g. minimise changes in acceleration or velocity) to identify which fragments should be merged. It is practical to leave much of this task to the user, since it is not generally too onerous and it would probably be very difficult to define rules for the software to cope with all likely circumstances.

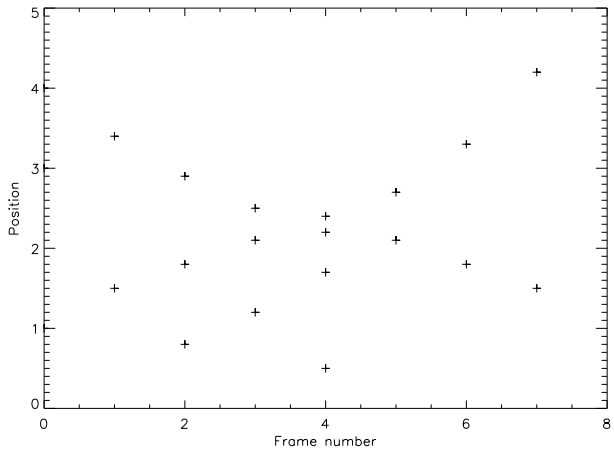


Figure 4.1: This figure shows several targets detected in each frame (numbered 0 to 7). Position is actually measured in 3D but only 1 coordinate (along the y axis) is shown here for clarity.

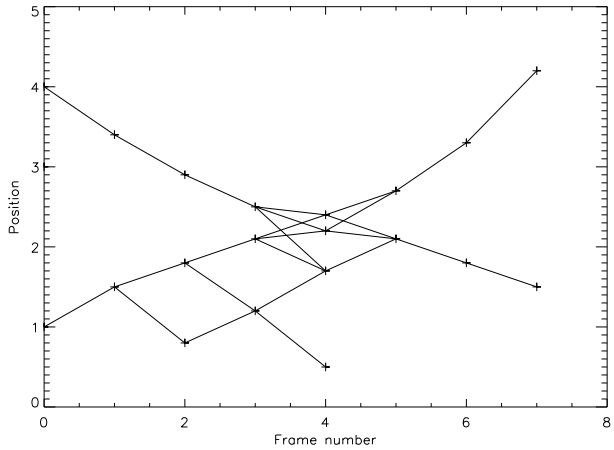


Figure 4.2: First step of the trajectory extraction for the targets of Figure 4.1: all plausible links are formed between the targets found (only links less than 1 unit long are deemed plausible). Isolated points (e.g. due to noise) form no links and are excluded from further analysis.

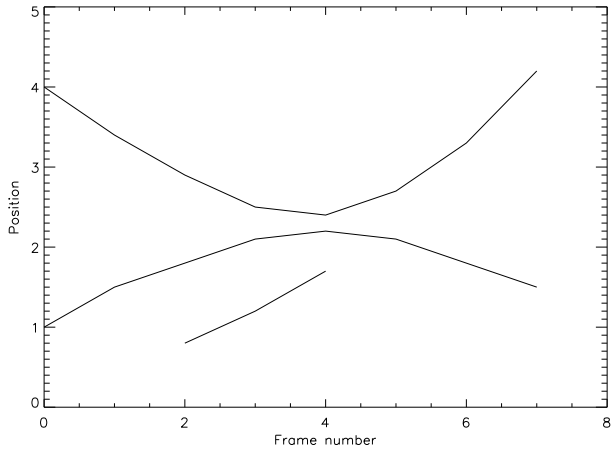


Figure 4.3: Second step of the trajectory extraction for the targets of Figure 4.1: only the most likely links (from Figure 4.2) are kept to make the trajectories. The rule used to determine "most likely" is to choose the link with smallest displacement.

Chapter 5

Practical Implementation

A system based on the algorithms described here has been used in several applications. This chapter outlines some of the issues which affect the quality of the measurements which can be made and is based on experience gained through the different applications attempted.

The applications on which the system has been used so far include:

- Measurement of wind-driven wheat motion at various growth stages
- Aircraft trajectory measurement on approach to Cranfield airport
- Measurement of the motion of a suspended target
- Human walking gait measurement
- Tracking tracers in a vortex airflow

These applications have raised a number of issues to be considered in experiment design and data processing. The system is an experimental system rather than an operational measurement device and requires a reasonable degree of operator skill to obtain good measurements.

5.1 Experiment Design

Good experiment design makes a difference to the quality of the measurements obtained. The following sections discuss some of the most important factors affecting the quality of the measurements.

5.1.1 Target Identification

The targets whose trajectories are to be measured need to be easily identifiable in the images recorded. If any labels are attached to improve identification, the labels should not affect the target's behaviour, e.g. labels should be lightweight and not too large.

5.1.2 Viewing Geometry

Camera viewing directions must keep the targets in view of at least two cameras at any time for position measurements to be possible: this favours a geometry where the viewing directions of the cameras are almost parallel so that the target is seem from the same aspect by all cameras. At the same time, the position measurements are most accurate if the viewing directions are as close to orthogonal as possible. In practice, it is not often possible to satisfy both these requirements simultaneously, and a compromise must be found. Geometries which work satisfactorily have used angles of 30°to 60°between the cameras' viewing directions.

5.1.3 Reference Points to Calibrate Camera Position and Pose

At some time during the experiment, a calibration target must be in the field of view of the cameras. In general it is easiest to use the same target for all cameras; if separate references are used then the coordinate systems of the different references must be related to each other. The reference target can be removed from the measurement volume once it has been imaged. The position and pose calibration does not need to be repeated unless the camera is moved (changing the zoom / wide angle setting does not require recalibrating camera position and pose).

A reference target must include a minimum of 3 points which span a plane (i.e. which are not co-linear). In practice it is better to use more points, and the points should span the field of view and also differ in range from the cameras (if the points are all at the same range from the camera then the estimate of camera position perpendicular to the viewing axis will be inaccurate). The larger the proportion of the field of view occupied by the reference target, the better.

5.1.4 Sychronisation

The video files from the cameras must be synchronised in time as well as registered to the same spatial coordinate system (which the camera position and pose calibration procedure achieves). To synchronise the videos, an event with good time accuracy which is visible in both videos must occur.

Suitable methods of synchronising the videos are a clapper board (as used in Holywood), switching room lights or a torch beam on or off, or a camera flash. Note that if the event is too brief (e.g. a camera flash) then it is not guaranteed to be seen by all cameras (as their shutters may have been closed at the time of the flash). Commercial camcorders maintain frame rates accurately, so that once cameras have been synchronised there is no need to repeat the synchronisation within an hour or so unless a camera is stopped / paused and then restarted.

5.1.5 Other Factors

Other relevant factors include the lighting conditions and target motion. Bright uniform lighting is the easiest to work with. In low light levels the cameras' exposure times increase which may blur images of moving targets.

Moving Targets

Moving targets can lead to image problems due to interlacing as well as blurring. Interlacing may be a problem on cameras which do not use a progressive scan to form the image, and so is sometimes a problem with consumer camcorders. The problem arises because the image frame is actually recorded as two sub-frames which are recorded at slightly different times, one made of the odd numbered lines and the other of the even numbered lines. If the camera's frame rate is 25 frame s^{-1} then the sub-frames are recorded $1/50s$ apart, and if the target has moved significantly (measured in numbers of pixels moved) in this time the two sub-frames will give an image of the target with jagged (vertical) edges. Functions to un-interleave the image can be used (effectively creating a video with a frame rate of 50 frame s^{-1} but with poorer vertical resolution); it is usually easier to track targets using this higher frame rate video.

5.2 Image Filtering

Good filtering of the images to highlight the target relative to its background significantly eases the data processing and improves the quality of the measurements. Several forms of image filtering have been used.

5.2.1 Temporal Filtering

Where the background is largely static it is useful to be able to isolate changes in the scene from the static background. This is easily done by subtracting one image from an earlier one (taking care to ensure the result is expressed in a format where all colour band signals are positive). Indoor experiments are particularly suitable for this technique.

5.2.2 Spectral Filtering

A standard method of identifying a target is to mark it with a coloured label. The label's colour should differ significantly from any other colours found in the scene, and the label should be large enough to occupy several pixels (at least 10, say).

The lighting and the camera's exposure setting affect the colour recorded by a camera, so the filter settings often need to be adjusted for the particular colours recorded in an individual experiment.

5.2.3 Correlation Mask

A more sophisticated method of tracking an object through a sequence of images is to use a correlation mask. A copy of a small section of the image including the object is taken and then correlated with later frames in the sequence. This method uses information about the colour and the shape of the target, and so in principle should be more selective and avoid spurious targets. In practice, a target's properties can change as it moves due to variations in lighting across the field of view and changes in aspect (i.e. the target is viewed from slightly

different directions as it moves), and these changes reduce the value of the correlation. Correlation values of 0.5 or better generally correspond to satisfactory matches between the mask and the object in the image.

5.3 Data Processing Sequence

There are several steps required to collect and process data to obtain measurements of target trajectories. With the system currently in use at Cranfield, several different programs are used; some are commercially available and others have been written specially (notably the programs referred to as `AVI1` and `mfitvid`, described in appendices C and B respectively). Figure 5.1 shows the various processing steps, including the three main tasks of (1) image calibration, (2) camera position and pose calibration, and (3) filming and tracking targets. Note that the diagram indicates which software tools are required for each of the processing stages.

More details on the use of the programs are available in Appendices C and B.

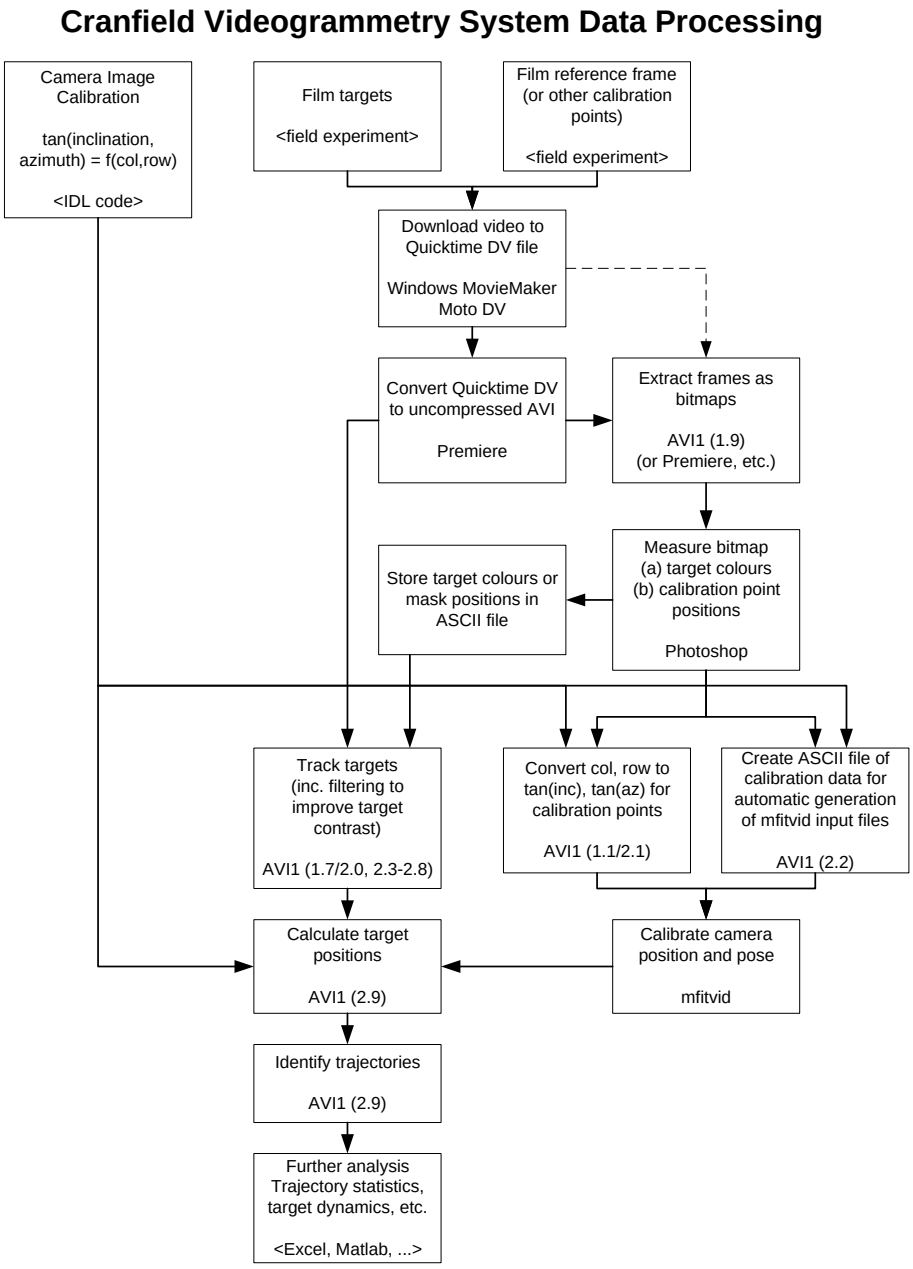


Figure 5.1: Videogrammetry data processing steps showing the software tools required at different stages.

Chapter 6

Discussion and Conclusions

From experience over several projects, the methods described in this report seem able to produce good quality measurements and are reasonably robust. The current system is clearly an experimental rather than operational system; however for research purposes it is very useful especially since it is based on the use of consumer digital camcorders (e.g. Sony Digital 8 models) which are readily available, robust, low cost, well-suited to fieldwork, and compatible with current personal computers.

Practical issues have a large influence on the quality of the measurements obtained (see Chapter 5). The fundamental accuracy achievable is determined by geometry, based on the angular resolution of the cameras used. As a guide, the camcorders used so far (Sony Digital 8) have a resolution of 1 mrad per pixel when on the wide-angle zoom setting; this is expected to reduce by a factor of 20 when used with the maximum optical zoom setting (because of the DV image compression algorithm and other factors 1 mrad is unlikely to be the strict "accuracy" of the cameras , but it is still a useful guide to the system's measurement accuracy).

The main application of the system uses a pair of cameras, however other configurations may be appropriate in other circumstances. A single camera can give good measurements if the targets' motion is constrained in one dimension. Systems using more than two cameras are also feasible, although the data processing is likely to be more complex.

A significant aspect of video analysis is that the user can see the events being measured as well as analyse quantitative data measured from the video. This is a major advantage of video recording of experiments and helps significantly with data analysis, e.g. to validate interpretations of measurements. Program `AVI1` includes functions for editing video files to aid this process, for example to merge two separate video files into one, or to annotate a video with plots of quantitative data relating to the event filmed.

Figure 6.1 shows a frame from an annotated video file. Coloured targets were attached to the aircraft's nose, tail and a wingtip. These targets were tracked and their locus calculated using the algorithms described in this report. The frame shown includes the original image with white crosses added to show the positions of the tracked objects, a filtered version of the image (almost wholly black except for small areas where the markers were found) and the annotation area which shows the experiment time in seconds and traces of the horizontal

coordinate of the aircraft's nose (sinusoidal trace) and the size of the detected target. An annotated video file like this is very helpful in understanding and interpreting the measurements made, and leads to much less ambiguous findings than would be the case if it were not possible to combine the video with results from quantitative analysis.

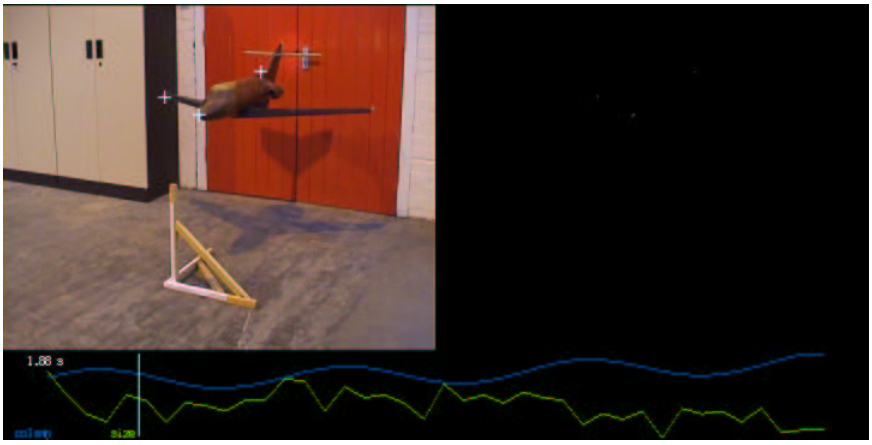


Figure 6.1: Example annotated video frame from a videogrammetry experiment produced using program `AVI1`.

6.1 Further Work

Future developments of the system are possible in several areas. Commercial packages are available which provide measurement systems based on video techniques. It is not currently the intention to compete directly with such packages; the objective for the current system is to provide high quality measurements based on readily available hardware (e.g. consumer electronics camcorders) in a system suitable for research use and development.

The process of making measurements is lengthy and requires many steps and several programs. The process could be simplified by some relatively straightforward integration of the different programs into a single piece of software. The effort required could only be justified however for a project of significant size.

Current work uses the epi-polar line solution method since this simultaneously solves for position and matches targets in the two images. If the target matching is known from some other source, then the linearized equations method (sometimes referred to as the direct linear transformation) is faster and uses the measurements more "symmetrically" (linearization does however give only an approximate solution). A hybrid of the epi-polar line and linearized methods could be developed to make optimal use of the measurements, although the practical benefits of this hybrid method might not justify the effort required.

Validation of the technique has been only partially carried out. All tests so far indicate that the measurements are of high quality with uncertainties corresponding to the errors estimated. Many applications require accuracy for relative, not absolute, position measurements, which significantly relaxes accu-

racy requirements. However, a thorough absolute validation of the technique is desirable, and is not too difficult in principle.

6.2 Acknowledgements

The work presented in this report draws on experience and results from a number of other people who have all made contributions to the Cranfield videogrammetry system. I am pleased to acknowledge the help of Cedric Seynat, Andreas Braunwart, Galder Bengoa, Mathieu Lalande, Tim Liggins (DataVision Ltd.), Keith Morrison and Davide Guiraud.

Bibliography

[1] Finchman W.H., Freeman M.H., *Optics*, 9^{th} Edition, Butterworth & Co (Publishers) Ltd, 1980, ISBN 0 407 93422 7

[2] Lalande, M., Dust devils on Mars. SOCRATES placement report, School of Engineering, Cranfield University, and ENSICA, Toulouse, September 2003.

[3] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P.: 1992, *Numerical Recipes in C, Second Edition*, Cambridge University Press: Cambridge, 994 pp.

[4] Seynat, C., Quantification of the effect of wind driven wheat motion on SAR interferometric coherence. PhD thesis, Cranfield University, 2000.

Appendix A

Camera Image Calibration

The calibrations reported here are based on the experiments and analysis of Cédric Seynat.

Videogrammetry relies on being able to convert the natural image coordinates (row, column) to geometrical angles (e.g. inclination and azimuth); the camera image calibration is therefore fundamental to videogrammetry.

This appendix describes a model used to calibrate the camera images and summarises the results of calibration experiments.

A.1 Calibration Model

The camera image calibration is expressed as coefficients of a function relating the geometrical angles to the image coordinates.

The video camera lens distorts the image so that the square grid used for calibration here appears as a characteristic barrel shape. This kind of image defect is well documented, and it is proved in [1] that the difference between the actual and aberrated height of an object is proportional to the cube of the image height. If no distortion were present in the image then the column and row number for a particular point would be proportional to the coordinates azimuth and inclination of that point, respectively. With the introduction of distortion in the image, the relation is no longer simple and needs to be modelled. With the theoretical treatment presented in [1] it is in principle possible to relate image coordinates to actual position, although this is not trivial.

Based on trials with several different polynomials and the analysis of [1], the model form chosen for the calibration is a 3^{rd} order polynomial in x and y (x is the column number and y is the row number, both measured relative to the centre of the image) (Equations A.1 and A.2) . α and β are the inclination and azimuth angles respectively of the point relative to the camera axes. Experiments show that this function (but not lower order polynomials) captures typical distortions by fitting the calibration data with an error corresponding to less than one pixel, which is roughly the measurement error for the coordinates of the calibration points in the image. A higher order polynomial is unnecessary.

The calibrations involve the tangent of the angles rather than the angles themselves because it is the tangent of the inclination or azimuth that is involved in the model of the imaging system.

$$\begin{aligned}\tan\alpha &= a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \\ &\quad + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 \qquad \text{(A.1)} \\ \tan\beta &= b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \\ &\quad + b_6x^3 + b_7x^2y + b_8xy^2 + b_9y^3 \qquad \text{(A.2)}\end{aligned}$$

The coefficients a_0 and b_0 are usually set to zero, thus forcing the camera's optical axis to pass through the centre of the image ($x = y = 0$).

A.2 Methodology

For the work at Cranfield, the image calibration has been performed for two different camcorders (each set to its wide angle setting). The two cameras are (1) SONY DCR-TR7000E (Digital 8 standard, using DV compression of the digital image data) and (2) CANON UC-X50Hi (Hi-8 analogue format). For the experiments, the camera is placed on a tripod at a known distance from a reference grid which is imaged with the camera. The grid is composed of a series of perpendicular lines precisely ticked. It was positioned so that lines on the grid were horizontal and vertical (the vertical alignment was checked with a plumb line). The camera itself was also carefully aligned vertically.

The grid-to-camera distance was measured from the grid to the inside edge of focusing ring of the camera. Although in principle the ideal distance should be measured from the focal plane of the camera, in practice this is not required as long as the same reference point is always used during all work involving measurements with the camera.

It is important to note that the grid images were taken with the zoom level set to its minimum (i.e. maximum wide angle setting). Obviously if a zoom is applied the relation between row and column number and inclination and azimuth angles changes. Since the cameras do not apparently provide a quantitative value for the zoom level, it is recommended that it is always set to its minimum or maximum when measurements are made.

A common origin for the reference grid and image was set to be at the centre of the image (thus defining the camera's line of sight), since the inclination and azimuth angles required are defined from this centre position. From measurements of the calibration points' positions the geometrical angles could be calculated, and then a least squares routine was used to estimate the model coefficients and fitting errors.

A.3 Calibration Results

The results of the calibration experiments are summarised in Table A.1 for the digital SONY camcorder, and in Table A.2 for the analogue CANON camcorder.

These values are based on measurements made by Cedric Seynat with a slight change to the reported range from the camera to the calibration target plane of an extra 15 mm (this is to allow for the distance from the camera's front lens ring to the estimated optical centre of the lens). The ranges used to calculate the tangent of the angles is thus 847 mm ($= 832 + 15$ mm) for the Sony camera,

Inclination coefficients		Azimuth coefficients	
a_0	$-3.80226.10^{-04}$	b_0	$9.89496.10^{-04}$
a_1	$-4.47296.10^{-06}$	b_1	$1.02754.10^{-03}$
a_2	$9.36606.10^{-04}$	b_2	$-1.38715.10^{-07}$
a_3	$7.00567.10^{-09}$	b_3	$-3.38471.10^{-08}$
a_4	$-2.85098.10^{-08}$	b_4	$3.36969.10^{-08}$
a_5	$3.49910.10^{-08}$	b_5	$1.14919.10^{-09}$
a_6	$9.08027.10^{-12}$	b_6	$2.41443.10^{-10}$
a_7	$2.18532.10^{-10}$	b_7	$1.20413.10^{-12}$
a_8	$-6.28508.10^{-12}$	b_8	$2.32910.10^{-10}$
a_9	$1.57032.10^{-10}$	b_9	$2.14570.10^{-11}$

Table A.1: Coefficients of the 3^{rd} order polynomials obtained with the digital SONY camcorder

Inclination coefficients		Azimuth coefficients	
a_0	$-4.76813.10^{-05}$	b_0	$-6.53703.10^{-04}$
a_1	$-3.64113.10^{-06}$	b_1	$1.32208.10^{-03}$
a_2	$1.21088.10^{-03}$	b_2	$2.97962.10^{-07}$
a_3	$5.96171.10^{-09}$	b_3	$-3.88573.10^{-10}$
a_4	$5.06823.10^{-09}$	b_4	$-1.08683.10^{-07}$
a_5	$-9.01054.10^{-08}$	b_5	$8.75926.10^{-09}$
a_6	$7.23817.10^{-12}$	b_6	$3.57502.10^{-10}$
a_7	$2.96875.10^{-10}$	b_7	$1.28913.10^{-11}$
a_8	$3.00965.10^{-11}$	b_8	$3.38045.10^{-10}$
a_9	$3.24447.10^{-10}$	b_9	$-3.95615.10^{-11}$

Table A.2: Coefficients of the 3^{rd} order polynomials obtained with the analogue CANON camcorder

and 805 mm (= 790 + 15 mm) for the Canon camera. Note also that although the model fitted allows a_0 and b_0 to be non-zero, the calibrations actually used ignore these small offsets, so that the camera's optical axis is assumed to pass through the centre of the image.

The average error ε_α and ε_β in the estimation of the angles is calculated as follows:

$$\begin{aligned} \varepsilon_\alpha &= \sqrt{\overline{\left(\widehat{\tan\alpha} - \tan\alpha_{true}\right)^2}} \\ \varepsilon_\beta &= \sqrt{\overline{\left(\widehat{\tan\beta} - \tan\beta_{true}\right)^2}} \end{aligned} \tag{A.3}$$

Here $\widehat{\tan\alpha}$ is the modelled value and $\tan\alpha_{true}$ is the true angle derived from the measured positions on the grid; the bar denotes averaging. The error is defined as the square root of the average square difference between the measured

and modelled tangent of the angle (α or β). The average is calculated over all the data points used to calculate the fitting function. The same definition applies to ε_β.

For quantitative analysis the values of ε_α and ε_β are compared to the average change in $\tan\alpha$ and $\tan\beta$ corresponding to one pixel in the vertical and horizontal directions respectively. For the digital SONY camcorder ($\mathcal{D}$=832mm), one pixel in the vertical direction corresponds to $\tan\alpha$=0.00096689 and one pixel in the horizontal direction corresponds to $\tan\beta$=0.00107644 (i.e. approximately 1 mrad in each direction). For the analog CANON camcorder ($\mathcal{D}$=790mm) these values are $\tan\alpha$=0.00124918 and $\tan\beta$=0.00136535 (roughly 1.3 mrad).

	ε_α (absolute value)	ε_α (in pixels)	ε_β (absolute value)	ε_β (in pixels)
Digital SONY camcorder				
3^{rd} ord. pol.	0.0006055	≈ 0.63	0.0008942	≈ 0.83
No distortion	0.002460	≈ 2.54	0.003174	≈ 2.95
Analog CANON camcorder				
3^{rd} ord. pol.	0.0007185	≈ 0.58	0.0005817	≈ 0.43
No distortion	0.002360	≈ 1.89	0.003356	≈ 2.46

Table A.3: Errors on the 3^{rd} order polynomial for the 2 video camcorders

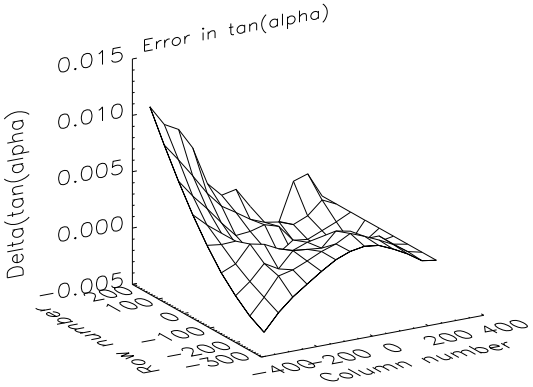
The 3^{rd} order polynomial distortion function provides an improvement compared to the no distortion case.

The final part of the analysis concerns the distribution of the error across the image. For that purpose the difference between the true values of $\tan\alpha$ and $\tan\beta$ and the estimated values $\widehat{\tan\alpha}$ and $\widehat{\tan\beta}$ is calculated at each point of measurement on the grid. The data points on the grid are irregularly spaced, so for plotting purposes with IDL a regularly spaced grid was generated from the irregular data, using a smooth quintic surface (procedure 'tri-surf' in IDL). The results are plotted in Figures A.1 to A.2.

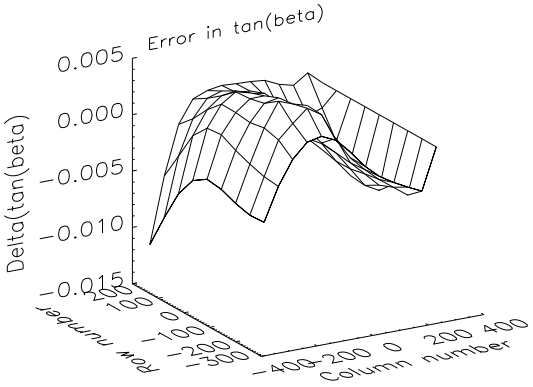
It is clear that the 3^{rd} order polynomial corrects for the effect of image distortion, particularly at the edges of the field of view. The remaining errors after correction are of the same order of magnitude or smaller than the measurement noise. A similar result is obtained for the analog CANON camcorder.

A.4 Standard Format for the Image Calibration Results

A simple ASCII file format has been defined to store the image calibration results, so that they can be read by the programs which need to use this information. An example of the format is shown in Table A.4 (file `imagecalwide0.txt`).

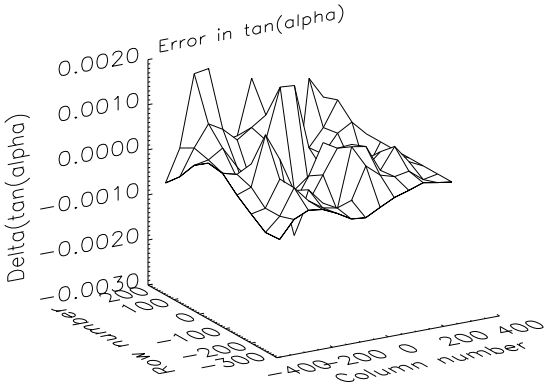


(a) Inclination angle

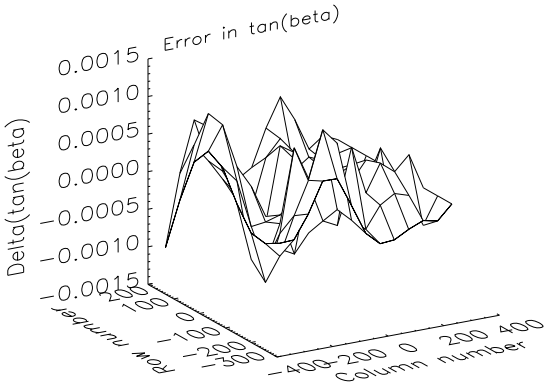


(b) Azimuth angle

Figure A.1: Distribution of the error across the image from the calibration of the SONY digital camcorder DCR TR7000E without any non-linearity corrections.



(a) Inclination angle



(b) Azimuth angle

Figure A.2: Distribution of the error across the image from the calibration of the SONY digital camcorder DCR TR7000E with 3rd order non-linearity corrections.

```
# imagecalwide0.txt
# Contains image calibration coefficients for the Sony camcorders in their wide-angle setting.
# File contains one data line summarising the contents (# of cameras, # of coefficients, etc.)
# then a line giving the image origin for each camera (origin_col, origin_row),
# and then the lines giving coefficients of the polynomials used to model distortions for
# each camera (inclination coefficients, then azimuth; excluding a[0] which is assumed to be 0).
# All comment lines are at the head of the file and start with '#', data lines start with a white
# space character.
#
# S.E. Hobbs, 14.05, 9 Aug 2001
2 10
359.5 287.5 359.5 287.5
-4.47296E-6 1.02754E-3 -4.47296E-6 1.02754E-3
9.36606E-4 -1.38715E-7 9.36606E-4 -1.38715E-7
7.00567E-9 -3.38471E-8 7.00567E-9 -3.38471E-8
-2.85098E-8 3.36969E-8 -2.85098E-8 3.36969E-8
3.49910E-8 1.14919E-9 3.49910E-8 1.14919E-9
9.08027E-12 2.41443E-10 9.08027E-12 2.41443E-10
2.18532E-10 1.20413E-12 2.18532E-10 1.20413E-12
-6.28508E-12 2.32910E-10 -6.28508E-12 2.32910E-10
1.57032E-10 2.14570E-11 1.57032E-10 2.14570E-11
```

Table A.4: Example format of the ASCII file to store the image calibration parameters (file `imagecalwide0.txt`). The values used are those reported in Table A.1.

Appendix B

Camera Position and Pose Calibration

This appendix discusses the second type of calibration required for videogrammetry, which is a measurement of the position and orientation ("pose") of each of the cameras involved (the image calibration discussed in Appendix A is the first type of calibration). The position and pose measurements for each camera must all be expressed in the same reference system. Several methods of measuring position and pose are available:

1. direct measurement
2. estimation by inverting image coordinates of a set of reference points
3. a combination of direct measurement and estimation by inversion

The image coordinate inversion method is in practice convenient for experimental use in many situations. The program (`mfitvid`) used in Cranfield's experiments to estimate position and pose is described here along with some test data used to validate the program.

B.1 Program `mfitvid`

`mfitvid` is written in Visual C++ as a 32-bit console application (i.e. it runs in an MS-DOS window from the command line). Within the program, a simple menu structure is used. The functions available include the basic ones needed to fit a non-linear model to data (maximum likelihood solution, using the Levenberg-Marquardt algorithm) plus other functions to allow more extensive investigation of the model and its inversion.

The program is based on a general model fitting program called `modelfit` which uses algorithms described in [3]. The `modelfit` software has been used in many applications since 1990 and within its limitations is reliable and robust. The forward model within `mfitvid` takes a point's position in 3-d as its input, and returns the tangents of that point's inclination and azimuth angles as its output: it thus naturally takes a 3-component vector as input and returns a 2-component vector as output. The standard algorithms described in [3] are

modified as suggested in [3] to operate with vector inputs and outputs, and require access to an external file containing the 3-d coordinates of the reference points used in the calibration (file `xvec.txt` is an example of this external file (see Table B.3 for its format)).

Table B.1 lists the main menu options for `mfitvid`.

Item	Option	Comments
0	Program help pages	Brief notes about the program
1	Load data ...	Loads data to which the model is fitted
2	List current data ...	
3	Fit model ...	Solves for the model parameters
4	Check evaluation ...	Runs the forward model
5	Evaluate ... model ...	
6	- as option 5 and write ...	
7	Evaluate ... solution ...	
8	1-d search ...	
9	Fit the model ... subsets ...	As 3, but repeats for subsets of the input data
Q	Return to DOS	Quit the program ('Q' and 'q' both work)

Table B.1: Main menu options for program `mfitvid`. The option labels are only shown in abbreviated form.

To use the program to estimate camera position and pose, options 1 (Load data into the program's working array) and 3 (Fit model to the current data) are used. Once a solution has been found, option 3 leads to a results sub-menu (Table B.2).

Item	Option	Comments
1	single parameter	
2	curvature and covariance	
3	Calculate errors ...	
4	principal results ...	Format required by `AVI1`
5	full results ...	As 4 plus covariance etc.
6	principle results tabulated ...	As 4 but all in 1 row
7	curvature and covariance to file	As 2 but written to file
8	repeat for different data	
Q	Return to main menu	'Q' and 'q' both work

Table B.2: Results analysis sub-menu options for program `mfitvid`. The option labels are only shown in abbreviated form.

Of the different output formats available, option 4 of the results analysis sub-menu generates an ASCII file formatted to be read directly by program `AVI1`. This is convenient for calculating target positions. The next section presents a test dataset used to validate `mfitvid` which also illustrates its operation. New users can use these values to check that `mfitvid` is being run correctly.

B.2 Simulation Input Values

The following data have been used to test `mfitvid.exe` for the calibration of a single camera system. The test data are stored in two files (`datatest.txt` and `xvec.txt`) for use by the program. These files allow the correct operation of the program to be tested and illustrate the data formats used. Currently, `mfitvid` assumes that file `xvec.txt` is available in directory `c:\video\ref0`.

The test data (in file `datatest.txt`) were generated using the forward model (menu option 4) of `mfitvid` with the camera model parameters set to position = (5 m, 5 m, 1 m), and attitude $\phi = 3.9$ rad, $\theta = -0.2$ rad, $\psi = 0.1$ rad. The positions of the calibration points are defined in file `xvec.txt`; this file should be modified (but keep the same name) to use other calibration points.

```
# ASCII file containing the vectors indexed by the modelfit variable x (default name xvec.txt).
# First 4 lines are comments and are ignored.
# First data line has two integers (number_of_components and number_of_points),
# then there are number_of_points lines of data, each with number_of_components values.
3 4
0.000 0.000 0.000
1.000 0.000 0.000
0.000 1.000 0.000
0.000 0.000 1.000
```

Table B.3: File `xvec.txt`, which contains the coordinates of the reference points and defines the fundamental coordinate system used for the measurements. `xvec.txt` must be in sub-directory `ref0` of `video` on drive `C` for the current version of `mfitvid` (version 1.10, using video1 v1.2).

```
# ASCII file containing test data to match the coordinate file xvec.txt for mfitvid.exe
# Each data line has {x y sig}, where y = model (x) and sig is the uncertainty on y
# S.E. Hobbs, 11.29, 8-May-00
0 0.05656 0.0005
1 -0.03259 0.0005
2 0.02976 0.0005
3 -0.14064 0.0005
4 0.05263 0.0005
5 0.07807 0.0005
6 0.19895 0.0005
7 -0.04764 0.0005
```

Table B.4: File `datatest.txt`, a test dataset for program `mfitvid` designed to be used with reference point data of Table B.3. The data are pairs of values for the tangents of the inclination and azimuth angles of each of the four reference points. The x value is actually an index which allows a vector model function to be used with scalars as suggested in [3].

B.3 Solution Method

To solve for the model parameters using program `mfitvid.exe` (menu option 3) a first guess has to be entered by the user. The first guesses used for the example dataset are listed in Table B.5.

parameter	value	quantity
$a[1]$	4.0	x coordinate of camera position (m)
$a[2]$	4.0	y coordinate of camera position (m)
$a[3]$	4.0	z coordinate of camera position (m)
$a[4]$	4.0	camera azimuth (ϕ, rad)
$a[5]$	0.0	camera inclination (θ, rad)
$a[6]$	0.0	camera rotation angle (ψ, rad)

Table B.5: First guess values used for the test data.

Results for the test case are given in Table B.6. These results are in the format suitable for input to program `AVI1` when it is used to calculate target positions, and were obtained using option 4 of `mfitvid`'s results analysis submenu (Table B.2).

```
# Results from modelfit, v 1.10, using video1 v1.2:  single camera model (uses c:\video\ref0\)

# Calculated on Fri Oct 17 16:32:09 2003
# Source data file:  testdat.txt in directory d:\video\test\
# skipped 0 input records; used records 0 to 7 of 0 to 7
# dof = 2, chisq = 0.000, p(X2>chisq) = 0.99995

# n.b.  SVD routines have deleted 0 (fit), 0 (error bounds) eigenvectors

# IF the model gives a good fit, is linear about the solution,
# and no eigenvectors have been deleted, the 1 standard deviation
# single-variable uncertainties for the solution vector are:
# a[1] = 4.9999, +/- 0.03299
# a[2] = 5.0000, +/- 0.03311
# a[3] = 0.9999, +/- 0.03607
# a[4] = 3.9000, +/- 0.00668
# a[5] = -0.2000, +/- 0.00525
# a[6] = 0.1000, +/- 0.00312
```

Table B.6: File `testres.txt`.

`mfitvid` thus estimates the camera position as (5.00 m, 5.00 m, 1.00 m) with uncertainties of about 3 cm, and its attitude as $\phi = 3.9000$ rad, $\theta = -0.2000$ rad, $\psi = 0.1000$ rad, with uncertainties of 3-7 mrad. These values agree well with those used to create the test data as expected.

The uncertainties can be checked against the first order models developed in chapter 3 and show satisfactory agreement.

In this case the model parameters are: $\delta\alpha = \delta\beta = 5.10^{-4}$ rad, $r = 7.1$ m, $a = 1.4$ m, $b = 0.7$ m. The estimated uncertainties are:

$$\begin{aligned} \delta x &= 0.018 \text{ m} \\ \delta y, \delta z &= 0.036 \text{ m} \\ \delta\psi &= 2.5 \text{ mrad} \\ \delta\theta, \delta\phi &= 5 \text{ mrad} \end{aligned}$$

Appendix C

Program AVI1

This appendix documents some of the algorithms used in program AVI1 (written to manipulate AVI files, and in particular to detect targets, track them, and calculate 3D trajectories using stereo videogrammetry). The appendix can be used as a reference either to understand the algorithms currently used, or to help develop improved versions.

The main stages in the data processing are:

1. Identify and track points of interest from a sequence of images.

2. Calculate the target position from the coordinates of points in images from 2 (or more) cameras.

3. Identify trajectories that best link together the target positions measured.

Note that the word "target" can mean either the point identified in an image or the point whose position has been measured in 3-d. The term "point" is generally used when referring to the position in an image, and "target" when referring to a physical object whose position is known or to be measured in 3-d. The program is written in C, and C data types are used in the description of data formats.

System calibration

Two stages of calibration are (largely) assumed in this appendix, and are described briefly in the first section here and in more detail in other parts of the report. These are:

1. Camera image calibration. It is assumed that the natural image coordinates (column, row) can be converted into the corresponding angles (azimuth, inclination) of the point away from the camera's optical axis. This calibration varies depending on the "zoom" setting of the lens.

2. Camera position and pose calibration. In order to determine target position from the angular measurements it is necessary to know the positions and orientations of the cameras involved. These can be estimated by imaging a set of points with known positions in the user coordinate system (e.g. on a reference target).

The remaining sections of this report describe the three main stages of the data processing (identified above).

C.1 System Calibration

Two calibrations are required to use the videogrammetry system. The camera image calibration needs to be performed only once for each camera (assumed to apply to all models of the camera so far) and each focal length setting (currently only calculated for the wide angle setting since this is most often used and the camera can reliably be reset to wide angle after use at other settings).

The camera pose and position calibration needs to be performed each time the camera is moved. A known reference target is used for this calibration; the reference target defines the coordinate system in which the target position measurements are to be made.

C.1.1 Camera Image Calibration

It is assumed that the natural image coordinates (column, row) can be converted into the corresponding angles (azimuth, inclination) of the point away from the camera's optical axis. This calibration varies depending on the "zoom" setting of the lens.

The calibration may be performed by taking an image of a grid of points for which the angles of inclination and azimuth can be calculated and then finding the coefficients of a least-squares fit of a function to convert image coordinates (column, row) into the angles (expressed as tan(azimuth), tan(inclination)). (See Appendix A.)

C.1.2 Position and Pose Calibration

The following table lists the steps to be taken to calibrate the camera position and pose. This process could in principle be automated, but for research use the current method is adequate. Two custom programs are used: AVI1 and mfitvid (mfitvid is described in Appendix B).

It is assumed that a suitable reference target has been filmed, and that the film has been transferred to a file on the computer (AVI1 v 0.36 and earlier needs the video file format to be .avi with video only, millions of colours, and no compression).

A good reference target is large enough to fill a significant portion of the image (10% or more) and extends in all three dimensions. Theoretically, at least 3 non-colinear points are needed, but in practice it is wise to use at least 5-6 points spread in inclination, azimuth and range.

	Step	Remarks
1	Extract an image of the reference target as a bitmap	AVI1 (option 1.9) can do this.
2	View the image (e.g. bitmap) with a program able to read out the image coordinates of points in the image	PaintShop Pro is suitable
3	Read off the image coordinates of all reference points on the target, and record their image coordinates and true position (in the coordinate system defined by the reference target) in an ASCII file with the format shown below (e.g. c1poscaldata.txt).	It doesn't matter where the image origin is since AVI1 can adjust for origins that are not lower left corner. PaintShop takes the upper left corner as origin.
4	Prepare the input files required by `mfitvid`. These files are (1) a data file, containing an index (counts from 0), the measured reference target positions as either tan(inclination) or tan(azimuth), and the uncertainty on the measurement, and (2) a file listing the coordinates of the reference points used. Both are ASCII files. Because of the way `mfitvid` currently works, it is necessary to place the coordinate file in directory `c:\video\ref0` (create this if necessary) and for it to have the name `xvec.txt`.	AVI1 (option 2.2) takes a file such as c1poscaldata.txt as input and creates the data file (e.g. `c1caldata.txt`) and coordinate file (must be `c:\video\ref0\xvec.txt`) needed for mfitvid. A file containing the coefficients of the camera image calibration is required. The wide angle calibrations for the Sony DCR-TR7000E (Digital 8 camcorder) is `imagecalwide0.txt` (see below). The datafile name is arbitrary but the coordinate reference file (`xvec.txt`) must have this name and be in directory `c:\video\ref0`. The sample files created by AVI1 should be renamed and moved to the right directory as necessary.
5	Load the calibration data into the working memory of program `mfitvid`	Run program `mfitvid`, and choose option 1. Load all available data points (skip none). (Check whether an extra data record has been created by listing the data with `mfitvid` option 2.)

6	Solve for the camera position and pose using `mfitvid`.	`mfitvid` option 3 starts the calculation; load all good data records available (should be an even number). Unless you know one of the camera coordinates, set all 6 parameters to "free" (flag = 1). The program converges more quickly and safely if you give a good guess for the camera position and pose (orientation / attitude). These values should be in the same coordinate system and units as defined by the reference target. The program usually converges in a few (<10) iterations. If the errors have been correctly specified and all other input data are good then the chisq value should be approximately the same as number of degrees of freedom (dof). The results can be inspected using option 1 of the results analysis sub-menu.
7	Create the position and pose calibration output file ready for use by `AVI1`.	Option 4 of `mfitvid`'s results analysis sub-menu creates the results file in the correct format (can be read by program `AVI1` when calculating target positions). The results should be checked for reasonable-ness; there is often good agreement with the guessed positions, but it is possible to obtain "mirror image" or other false solutions (if the first guess was poor).

Table. Steps required for the camera position and pose calibration using programs AVI1 and mfitvid.

```
% Camera position calibration data file.
% For camera 1, wheat expts 2000 jun 02, film c1jn02z02
%
% Exactly 9 comment lines (ignored, % prefix optional), then line with number of points
% followed by 1 data line for each point giving label (string <= 19 char, no white space),
% x y z values of point in user units and then col and row values for that point in the
% image, values separated only by spaces.  Row uses top left as origin (Paintshop).
%
% S.E. Hobbs, 25 April 2003
 6
 A   0.000   0.220   0.280   337   181
 B   0.000   0.000   0.280   385   226
 C   0.310   0.000   0.280   576   187
 D   0.310   0.220   0.280   512   148
 E   0.000   0.000   0.000   399   386
 F   0.310   0.000   0.000   574   339
```

Table. Example of ASCII file (`c1poscaldata.txt`) recording the positions of reference target points for the camera position and pose calibration in the format required by AVI1 to create the files for input to mfitvid.

```
# imagecalwide0.txt
# Contains image calibration coefficients for the Sony camcorders in their wide-angle setting.
# File contains one data line summarising the contents (# of cameras, # of coefficients, etc.)
# then a line giving the image origin for each camera (origin_col, origin_row),
# and then the lines giving coefficients of the polynomials used to model distortions for
# each camera (inclination coefficients, then azimuth; excluding a[0] which is assumed to be 0).
# All comment lines are at the head of the file and start with '#', data lines start with a white
# space character.
#
# S.E. Hobbs, 14.05, 9 Aug 2001
         2        10
          359.5            287.5            359.5            287.5
         -4.47296E-6       1.02754E-3      -4.47296E-6       1.02754E-3
          9.36606E-4      -1.38715E-7       9.36606E-4      -1.38715E-7
          7.00567E-9      -3.38471E-8       7.00567E-9      -3.38471E-8
         -2.85098E-8       3.36969E-8      -2.85098E-8       3.36969E-8
          3.49910E-8       1.14919E-9       3.49910E-8       1.14919E-9
          9.08027E-12      2.41443E-10      9.08027E-12      2.41443E-10
          2.18532E-10      1.20413E-12      2.18532E-10      1.20413E-12
         -6.28508E-12      2.32910E-10     -6.28508E-12      2.32910E-10
          1.57032E-10      2.14570E-11      1.57032E-10      2.14570E-11
```

Table. Image calibration coefficient file (imagecalwide0.txt) for the Sony DCR-TR7000E camcorder at its wide angle setting. This is used by AVI1 to convert image coordinates to angles of inclination and azimuth.

```
# Results from modelfit, v  1.07, using video1 v1.1: single camera model (uses c:\video\ref0\)

# Calculated on Fri Apr 25 11:24:07 2003
# Source data file: c1testdat.txt in directory c:\video\ref0\
#   skipped     0 input records; used records    0 to   11 of 0 to   12
# dof =    6, chisq = 104.479, p(X2>chisq) = 2.911854e-020

# n.b. SVD routines have deleted  0 (fit),  0 (error bounds) eigenvectors

# IF the model gives a good fit, is linear about the solution,
# and no eigenvectors have been deleted, the 1 standard deviation
# single-variable uncertainties for the solution vector are:
# a[1] =  -0.3832, +/-  0.02205
# a[2] =  -1.3223, +/-  0.01660
# a[3] =   0.8629, +/-  0.02141
# a[4] =   1.3260, +/-  0.01627
# a[5] =  -0.4560, +/-  0.01397
# a[6] =   0.1125, +/-  0.00969
```

Table. Example results file from program mfitvid, suitable for input to AVI1.

C.2 Point Tracking

The first task in processing the video data is to identify points in each image and then track them through a sequence of images. Two methods have been used: a colour threshold or a correlation mask.

C.2.1 Colour Threshold

This method defines a cuboid in colour space (blue, green, red components) and checks whether each pixel's colour is within this region. The result is a simple binary value (0 for colour outside the chosen region, 1 for an acceptable colour match).

In practice, this method requires careful tuning of the mean colour and bounds around the mean. It is difficult to account for changing lighting conditions as a target moves. Initial investigations of alternative colour representations (e.g. one that identifies colour brightness / intensity separately) have not yet shown any significant improvement on the simple thresholds.

It is useful to use some form of low-pass filter on the image after colour thresholding to associate outlier pixels with nearby clusters. Otherwise, spurious single pixel points are detected near to clusters associated with "real" points. The point position reported is usually calculated as the centroid of each cluster found.

C.2.2 Correlation Mask

A rectangular region of the image from the first frame is chosen. The correlation between this region (the mask) and portions of each later frame is calculated, and the maximum correlation is taken to be the point of the mask in the later frames. The correlation uses information from all three colour bands.

An option to allow the mask to be updated with the best correlation found is available, but this tends to lead to wandering of the mask relative to the desired point. A hybrid between a frozen and a changing mask may be possible and could deal with situations where, say, the visible mask shape changes as the target moves in the image.

Results so far suggest that the correlation mask is generally more reliable than the colour threshold although it is computationally more intensive (less than an order of magnitude more computation).

C.2.2.1 Correlation mask input parameters

Option 2.4 of AVI1 (v 0.34) takes the correlation mask parameters from an external ASCII file. This is useful if many targets are being tracked, or if analysis has to be repeated several times. An example input file for correlation mask parameters (caml7t0.txt) is given below.

```
# Target parameter data file for AVI1 correlation mask tracking
#
# Data from LIRMM expts 10 Apr 2002, initial image frame 0 of camleft7.avi (watch, shoulder, knee)
#
# The first 9 lines are comment lines (prefixed with #) to be discarded; then 1 line with the
# number of targets followed by 1 line for each target using the following format:
# <centre col> <centre row> <col half-size> <row half-size> <correlation step size> separated by spaces
#
# S.E. Hobbs, 14.03, 01 May 2002
4
591  478   6   6   1
572  325   8   8   1
595  322   6   6   1
569  193   5   5   1
```

Table. Example input file (caml7t0.txt) defining the AVI1 correlation mask parameters.

Note that the ASCII file parameters are slightly more versatile than the direct user input in that the mask does not need to be square.

C.2.3 Image column, row origin convention

Images stored in the avi file format have their origin at the bottom left corner. Therefore column indexes the image from left to right, and row indexes from bottom to top (both counters starting at 0).

Some graphics programs (e.g. Paintshop) take the image origin as the top left corner, and the row therefore needs inverting to be compatible with the avi convention.

C.2.4 Results Format

Two data formats are used to pass results for the point tracking. The earliest was based on an array of float, while more recent work uses a struct.

C.2.4.1 float array

Point number	target_list[point][0]	target_list[point][1]	target_list[point][2]
0	Number of points in this image	<not used>	<not used>
1	col	row	size
2	col	row	size
3	col	row	size
…	col	row	size
Nmaxtarg	col	row	size

Note that target_list[0][0] contains the total number of points stored, and that points are numbered from 1. The struct method numbers points from 0, and when targets are detected, targets (with position measured in 3d) are numbered from 0 also.

C.2.4.2 struct

A point_position_record struct is defined to store data specific to an individual frame, and then a frame_points_record is used to store information about the frame and each point detected in the frame. It is generally easier to modify data formats based on struct than on arrays, and the format can be naturally tailored to the type of data to be stored rather than forcing all parameters to a single type (e.g. float).

C.2.4.2.1 point_position_record

The point_position_record struct is used to record all information about a single point in an image.

float	float	float	float	short
col	row	correlation	size	mask_num

C.2.4.2.2 frame_points_record

The diagram below shows the frame_points_record and the associated point_position_record arrays.

Frame	short	short	point_position_record		
0	flag	N_point	*ppr	→	ppr[0..pmax]
1	flag	N_point	*ppr	→	ppr[0..pmax]
2	flag	N_point	*ppr	→	ppr[0..pmax]
…	flag	N_point	*ppr	→	ppr[0..pmax]
fmax	flag	N_point	*ppr	→	ppr[0..pmax]

Note that because of the use of pointers to struct in AVI1 to allow the structure sizes to be set at run-time (and so have more memory available), it is necessary to reserve space first for fpr[0..fmax] and then separately to reserve space for fpr[frame].ppr[0..pmax] for each frame = 0 .. fmax.

C.3 Target Position Calculation

The algorithms used in AVI1 to calculate target position from the point positions measured in each of two images are described in chapter 3 (epi-polar line method).

C.3.1 Data Format

Several data formats are used by the position calculation algorithm within AVI1 either for data input or for output.

C.3.1.1 Data Input

The point data from the original images are stored either in a float array (ipos[0..fmax][0..pmax][0..2]) or in the frame_points_record array fpr[0..fmax] (see section C.2.4.2 above). The following diagram shows how ipos (the float array format) is defined and used.

Each cell of the array (below) is actually a vector of 3 floats. For targets 1..pmax (≤ Nmaxtarg) the vector is used to store point information as:

float	float	float
ipos[frame][point][0] col	ipos[frame][point][1] row	ipos[frame][point][2] size / correlation

Vector ipos[0][0] is used as:

ipos[0][0][0] n_points	ipos[0][0][1] # of frames stored	ipos[0][0][2] max # of targets (Nmaxtarg)

i.e. the values used to size the whole float tensor are stored in the first vector.

Vector ipos[frame][0] is used as:

ipos[frame][0][0] # of points detected in this frame (n_points)	ipos[frame][0][1] <blank>	ipos[frame][0][2] <blank>

The float tensor (array of float*) is used as shown below.

Frame↓ Point→	0	1	…	Nmaxtarg
0	<see above>	point data	point data	point data
1	<see above>	point data	point data	point data
2	<see above>	point data	point data	point data
…	<see above>	point data	point data	point data
fmax	<see above>	point data	point data	point data

C.3.1.2 Results Output

Formats used for the position results are described below (as input formats for the target trajectory detection, section C.4.1).

C.4 Target Trajectory Detection

From a sequence of target positions measured in 3D space as a function of time, it is useful to identify positions that relate to the same target, and thus measure the target's trajectory. The algorithm described here uses kinematic information (e.g. it may assume that if there are positions which do not change much from one timestep to the next then these correspond to the same physical target and therefore form part of that target's trajectory). "Label" information is also available, e.g. colour or target size, and this can be used to check the integrity of the trajectories identified by these kinematic methods (i.e. if the trajectory really does correspond to a single physical object then the label information should not change significantly within the trajectory).

C.4.1 Input Information

The input information is assumed to be a set of records (one record per frame pair) giving the positions of all the targets (up to Nmaxtarg) positions measured from each pair of frames.

Target_position_record

Target position record:	x	y	z	λ	t

Set of 5 float values: (x, y, z, λ, t)

Frame record

The components of each frame record are: long integer (Ntarg, number of targets measured), then Ntarg Target_position_record (<tpr>, up to a maximum of Nmaxtarg). Nmaxtarg-Ntarg blank Target records follow, padding the frame record to a standard size.

Frame 0	Ntarg0	Target_position_record (tpr)	<tpr>	*... total of Nmaxtarg Target records per frame record; fill up frame record with blank targets*	<tpr>
Frame 1	Ntarg1	<tpr>			
Frame 2	Ntarg2				
Frame N	NtargN				

Using a formal record structure makes it easy to update the structure if it is desired to incorporate new information (e.g. size, colour or mask identification for the target).

C.4.2 Identification of Trajectories

Several steps are followed to identify trajectories. Chapter 4 describes the algorithm's principles; this section describes the data structures used in its implementation within AVI1.

The steps to identify trajectories are: (1) form all simple links (a "link" is defined in chapter 4, it is a pair of position measurements one time step apart), and (2) edit the links by keeping only the most plausible to leave a set of trajectories.

C.4.2.1 Form all plausible links

In general it may be possible to form a great many links, but many may not be physically plausible (e.g. because they correspond to motion at a speed much greater than is possible for the target). A maximum acceptable speed corresponds to setting a maximum acceptable link displacement.

The process of forming all plausible links is implemented in AVI1 as finding two targets in contiguous frames which are within the displacement threshold defined by the user. If several links are within the threshold store the closest M <= Nlinkmax. The algorithm is:

For frame 0 to N-1 (of 0..N), for each target in frame nframe,

(a) calculate ranges to all targets in frame nframe+1

(b) order ranges

(c) find links shorter than dmax (up to Nlinkmax)

(d) store the forward and reverse link information for the relationship

The following diagram illustrates the relating of target positions to make links, and then creating trajectories from chains of links.

User-defined displacement threshold: |----------|

	T1	T2	T3	T4	T5
Frame 0	o	o	o	o	o
Frame 1	o	o	o	o	
Frame 2	o	o	o	o	o
Frame 3	o	o	o	o	o
…	o	o	o	o	
	o	o	o	o	
	o	o	o	o	o
Etc.					

The 5 targets of frame 0 are shown schematically (T1..T5). All but T5 have neighbours within the user-defined displacement threshold – isolated targets like T5 will be discarded and do not contribute to a trajectory. The trajectories started by T1 and T2 touch in frames 2 and 3, so multiple links can be formed. The link editing stage of trajectory detection has to decide which links to keep and which to break.

The link information is stored in a separate array.

Frame 0	Ntarg in frame 0	Target_link record	Target_link record	*... total of Nmaxtarg Target_link records per frame record; fill up frame record with blank targets*	Target_link record
Frame 1	Ntarg1	Target			
Frame 2	Ntarg2				
Frame N	NtargN				

Target_link record

# of forward links (Nflink ≤ Nmaxlink)	Forward links (first Nflink of Nmaxlink are used)	# of reverse links (Nrlink ≤ Nmaxlink)	Reverse links (first Nrlink of Nmaxlink are used)

If the number of forward links is 0 and the number of reverse links is ≥1, then this is the end of a chain.

If the number of reverse links is 0 and the number of forward links is ≥1 then this is a chain start.

Link record

Target number (in the forward or reverse frame; note that the frame number is implicitly set by the base frame number and the sense of the link – forward or reverse)	Displacement between the two ends of the link

C.4.2.2 Find all trajectories

The next step of the trajectory formation algorithm is to find all trajectories using the link information available.

The algorithm is:

- Search the links array, from frame 0 to Nframe-1 for each target that starts a link.
- Follow the chain to its end (edit all branches to prevent duplication of targets).
- Store the links by recording the starting point (frame, target number) and length of the trajectory (number of frames).

(Note that a more general routine that would allow all possible links to be followed would be much more complex to implement and would probably not solve many real problems. What problems remain with the simple method proposed can be solved with some operator intervention.)

C.4.2.2.1 Edit all branches

In principle it is possible for a target position to be linked to several positions in adjacent frames (for either the forward links or the reverse links). In this case it is necessary to remove all but the most plausible forward and reverse links. The method is:

- If there's a branch, choose the strongest link (criteria = smallest displacement, or smallest change in acceleration ~ "jerk") and break all other links (count and record all links broken as a check on the algorithm). Apply this for all cases where there are >1 forward or >1 reverse links.

If the end of a link is found, store the result (starting point, length of chain), increment the count of trajectories found, and then search for the next starting point.

The format for storing the results of the trajectory search process is:

Array (0..Nmaxtarg x Nframe / 2) of results record (array [0..2] of long)

Row 0	N_count_trajectory (≤ Nmaxtarg x Nframe)	<blank>	<blank>
Trajectory 1	Frame number of start of trajectory	Target number in frame of start of trajectory	Length of trajectory (≥ 2)
Trajectory 2			
Trajectory N_count_trajectory			
Row Nmaxtarg x Nframe / 2	<blank>	<blank>	<blank>

(Note that the way this algorithm focuses only on the branch points implies that the topography of the links away from a branch point is not significant for tracking a target.)

C.4.2.3 Output the trajectories found

The final step in identifying trajectories, is to generate an ordered list of the trajectories found.

The method used is to search the list of targets (start points, lengths) and order by (increasing) start frame / target number for the link. Write out the results in a format ready for plotting (in Excel (list trajectory number, frame number, target number in first frame, position coordinates as columns, with a blank row between targets) initially, or Matlab or IDL for example).

- The trajectory array points to the start of each trajectory in the link array;
- The position coordinates are in the input target position array and are pointed to by the link array.

C.5 Utility Functions

AVI1 includes a variety of utility functions. These give more ways of manipulating video data or simplify some aspects of data processing. The utility functions include:

Function	Description
Image annotation	Allows an annotation area to be added to the foot of the video to show signal traces etc.
File copying (with filtering)	Various methods of copying AVI files are provided, including applying images filters for visualization.
Frame rate control	The playback rate of a video can be controlled (option 1.5)
File creation for mfitvid	The files needed by mfitvid can be created automatically (including applying camera image calibrations) from basic ASCII input files.
File merging	Two video files can be combined into one, for example to show simultaneous views from different cameras.

The following sections describe these utility functions.

C.5.1 Image annotation

An avi video file can be annotated with a plot area showing signal traces synchronized with the video (see Figure 6.1, chapter 6 for an example). This is a powerful way of illustrating phenomena and of analyzing behaviour, for example anomalous features of the data can be explored by combining the quantitative signals with the original video to allow more detailed study.

The program help option (main menu option 0) describes the format of the ASCII file used to input signal data to create the footer area containing the signal traces.

C.5.2 AVI file copying / filtering

.avi files can be copied or filtered (including copying a portion of the image for a subset of the original video frames). These operations can be combined with several image filtering operations. The filters available include:

- any linear combination of colour bands
- a cuboid can be defined in colour space to binary filter pixels in the images
- a correlation mask can be used
- the difference between frames can be calculated to detect change relative to a static background

Good filtering of the images helps significantly for the target tracking routines.

C.5.3 Frame rate control

Option 5 from the utilities sub-menu allows the frame rate to be changed from its default value. This is useful for setting a more convenient frame rate (usually to replay a sequence at slower than the real-time rate).

C.5.4 Automatic creation of mfitvid input files

Two ASCII input files are required by mfitvid.exe. These files are xvec.txt and the "data" file (containing an index referring to the input vector and output vector component, the signal, and the uncertainty on the signal). These files can be created manually or using option 2.2 of AVI1. An example of the camera position and pose calibration data required by option 2.2 (file lcamposcaldata.txt) is given below. This method automates tasks such as converting col, row coordinates into tan(inclination, azimuth) and writing the ASCII files with their strict format requirements. The next stage of program development may be to integrate the calibration function of mfitvid within AVI1.

```
% Camera position calibration data file.
% For left camera, LIRMM expt 10 April 2002.
%
% Exactly 9 comment lines (ignored, % prefix optional), then line with number of points
% followed by 1 data line for each point giving label (string <= 19 char, no white space),
% x y z values of point in user units and then col and row values for that point in the
% image, values separated only by spaces.
%
% S.E. Hobbs, 16 April 2002
 6
 O   0.000   0.000   0.000   272   475
 A   0.500   0.000   0.000   487   467
 B   0.528   0.000   0.318   500   318
 C  -0.028   0.000   0.318   259   318
 D   0.500   0.500   0.020   435   426
 F  -0.028   0.510   0.318   241   313
```

Table. File lcamposdaldata.txt: example of the data format for AVI1 to create input files for position and pose calibration using mfitvid.

C.5.5 File merging

Two video files can be merged into one, e.g. to combine the two stereo views available into just one file. This allows the two views to be shown conveniently in a way which preserves their synchronization. The two files may be combined "portrait" or "landscape", and by repeating the operation several times it is possible to create a two-dimensional array of views. Figure 6.1, chapter 6, is an example of file merging in the landscape mode.

C.6 Program Structure

AVI1 is written using Visual C++ (version 6.0) as a 32-bit console application (i.e. it runs as if from a MS-DOS window). Menus are used to make the various options available, either directly from the main menu or using sub-menus for a group of related functions. The following table shows the current menu structure.

It is planned to develop a full MS Windows application version of AVI1: the version numbers for this will start with 1.00. All functions will be preserved and additional functionality is anticipated (e.g. the ability to handle avi files with formats other than the uncompressed, no sound track standard currently assumed).

Main Menu Option	**Sub-menu Option**	**Remarks**
0. Program help pages	<no options>	Hardly used so far.
1. Utility functions	1. Convert row,col to incl,az	Checks image calibration; similar to but not identical to 2.1.
	2. Examine the avi file header	
	3. Examine the avi file footer	
	4. Create new blank avi file	
	5. Change the playback rate	
	6. Test font functions	
	7. Edit the filter parameters	Same as 2.0
	8. Print histograms of colour band intensities	
	9. Export frame as bitmap image	
2. Target tracking functions	0. Edit the filter parameters	
	1. Check the image calibration functions	
	2. Create files needed by mfitvid	Creates ASCII input files needed – data and reference files.
	3. Tracks using correlation mask	
	4. <as 3 but takes mask parameters from external ASCII file>	
	5. Tracks using colour thresholding method	Uses older and slower float array to store image data

	6. <as 5 but doesn't write cross-hairs>	
	7. Finds the centre of a label in an AVI file (thresholding method)	Uses the faster char array to store image data
	8. <as 7 but without writing cross-hairs to the avi file>	
	9. Position solution using epi-polar line method	
3. Read an AVI file and write frame data to a text file	<no options>	
4. File filtering / copying functions	0. Fast copy of avi file without filtering options	
	1. Copy or filter avi file	
	2. Take the difference between frames	
	3. Uninterleave avi half-frames	
5. Annotate an avi file	<no options>	
6. File merge functions	1. Merge files landscape	
	2. Merge files portrait	
	3. Create a red/green "3d" file	

Table. Program AVI1 version 0.34 menu structure.

The data analysis flow for Cranfield's videogrammetry system based largely on the operations available within program AVI1 are shown in the following diagram.

Cranfield Videogrammetry System Data Processing