1-1-2003

Automatic target matching

Michael Leslar
Ryerson University

AUTOMATIC TARGET MATCHING

by

Michael Leslar, B.Eng,
Ryerson Polytechnic University, 2001

A thesis

presented to Ryerson University

in partial fulfillment of the

requirement for the degree of

Master of Applied Science

in the Program of

Civil Engineering

Toronto, Ontario, Canada, 2003

Your file Votre référence
ISBN: 0-612-87145-2
Our file Notre référence
ISBN: 0-612-87145-2

AUTHOR'S DECLARATION

Automatic Target Matching by Michael Leslar, 2003
Master Of Applied Science, Civil Engineering Dept., Ryerson University

ABSTRACT

Many photogrametric processes require a large number of points to be collected from numerous digital images. It is imperative that these points be collected accurately, so that precise real-world coordinates may be assigned to points captured in the image. To this end, many techniques have been developed to locate, track and identify image targets. This thesis outlines many of these techniques and presents a target matching solution that has been developed in C++, for the subpixel target location program INDMET. The target matching solution is composed of three elements: an epipolar line program, a cross correlation program and a template least squares matching program. The epipolar line program is used to limit the search area in the right image of a given stereo pair, to the vicinity of a single line. The cross correlation program searches this line to locate possible targets and the template least squares matching program is used to determine the target centre of a black and white image target, once it has been located. It was found that these three programs, working together, had between a 20 and 70 percent chance of locating the correct target, depending on the similarity of elliptical targets in each image. Once found, the program could calculate the target centre to an accuracy of approximately 1/10th of a pixel.

ACKNOWLEDGEMENTS

I would like to acknowledge the significant amount of assistance provided to me by Dr. Michael Chapman, P.Eng. In addition, the digital images provided by the Applanix Corporation were invaluable in the completion and testing of the target matching program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 Introduction

Automatic target matching can be loosely defined as the process where a computer can use preprogrammed algorithms to identify, locate and track the objects of interest that occur in multiple digital images. In photogrammetry, it often occurs that numerous points need to be located for the purposes of rectifying digital imagery. This can be a tedious and time consuming task. Added to this is the danger that one misidentified point could destroy the accuracy of the final results. This can be extremely prohibitive to performing a photogrammetric solution in a given situation. The development of a computer program which utilizes the well established algorithms and concepts of automatic target matching could be very useful in solving this problem. By incorporating these concepts into the previously developed software package INDMET, it is hoped that a solution to this problem may be found.

1.1 Objective and Scope

This thesis developed a complete automatic target matching system using the methodologies for the calculation of epipolar lines, cross correlation matching and template least squares matching. These methodologies were programmed and applied using the pre-existing target locator, INDMET. Due to the vast number of target matching algorithms available, the solution presented here is the only possible solution. The development of a complete target matching program was beyond the scope of this thesis; therefore, the previously programmed user interface from the software package INDMET was used as the platform upon which a target matching solution was built.

This thesis also contains a literature review, which describes many of the common target matching algorithms, and methodologies that have been developed, as well as some of their common uses. It is, however, beyond the scope of the thesis to delve too deeply into the applications of automatic target matching. A brief summary of these applications in Section 1.2 is all that will be presented.

1.2 Applications of Automatic Target Matching

Automatic target matching is a multidisciplinary field which is concerned with the identification, location and/or tracking of targets by intelligent computer programs. Figure 1.1 shows the fields of study for which image matching is of interest and indicates how these fields are interconnected.

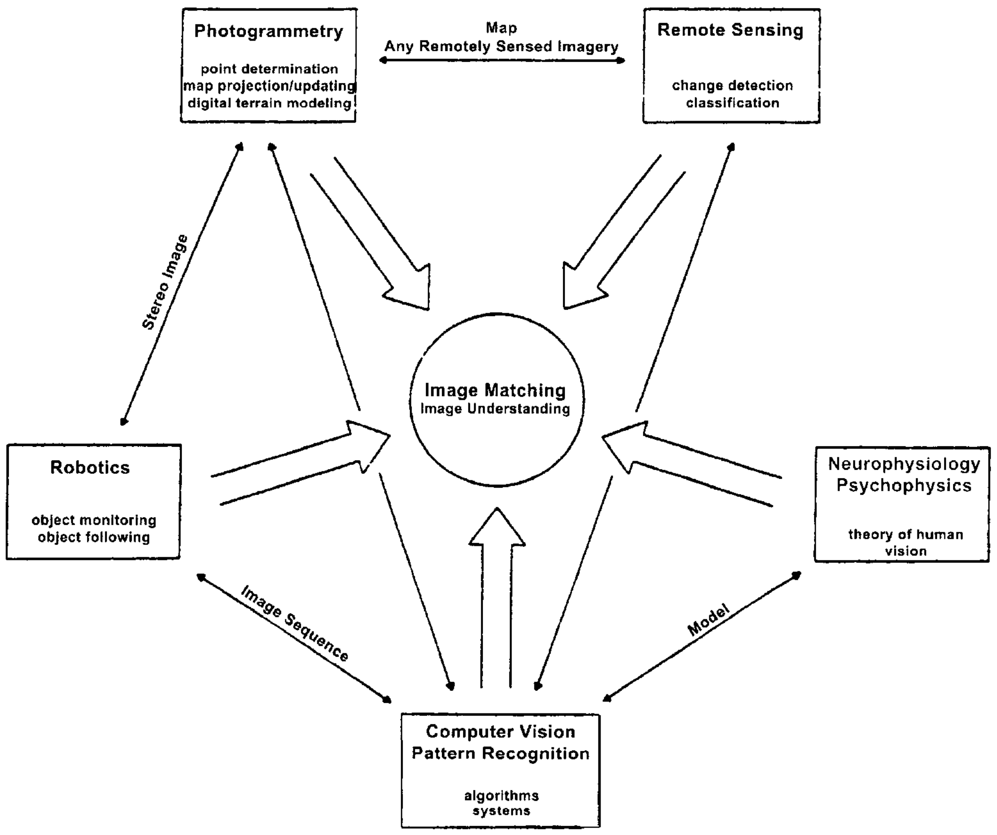


Fig 1.1: Applications of Image Matching (Luhmann and Altrogge, 1986.)

Each of these professions needs computers to automatically perform target matching for different reasons. Photogrammetry is concerned with performing high precision measurements on digital imagery for the purpose of calculating the real-world coordinates of those objects that appear in those images. Remote sensing, on the other hand, is interested in automatically detecting broad pattern changes between two time lapsed images. These changes in pattern can be taken as indications of such things as changes in environmental conditions and human land usage patterns. Neurophysiology, for example, uses automatic target matching to develop better theories on how the human visual system works. Computer vision specialists are concerned with developing systems that are capable of performing menial tasks faster, longer and more efficiently than humans do now. Examples of such tasks would be manufacturing, quality control, driving and piloting. Finally, robotics specialists are interested in developing machines that can mimic human actions and behaviors for the purposes of creating automated workers.

Just as there are numerous algorithms and theories concerning target matching, there also exist numerous uses and applications of target matching. As previously stated, photogrammetry is a field in which vast numbers of points need to be gathered on a regular basis. One example of such a use would be industrial metrology, which is the reason that INDMET exists. Industrial metrology involves the precise alignment of high-tech industrial machinery. This alignment is critical since a misalignment can cause premature machine failure, which can be very costly. Automatic target matching can be used to locate targets on such machinery, greatly reducing the time required to perform these alignments. Another application, similar to industrial metrology, would be as-built surveys of industrial sites. Figure 1.2 shows an image of an industrial piping set up. The lower right of Figure 1.2 shows an actual image taken of the piping, while the upper left

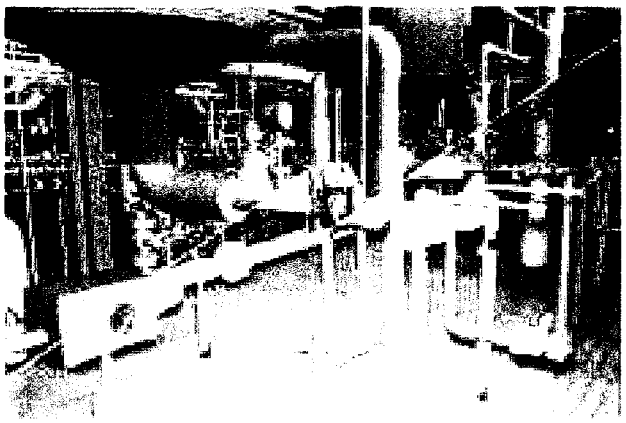


Fig 1.2: As-Built Survey of Industrial Piping (Russell, 2003.)

shows a three-dimensional (3-D) model that was created after points were gathered from the images. The number of points needed to produce this detailed model would have been excessive. An automatic target matching model would make the gathering of these points faster and more efficient. A third photogrammetry application of automatic target matching involves the collection of image tie points for the purpose of creating orthographic images from multiple aerial or satellite images. Figure 1.3 demonstrates how the 92 Landsat satellite images of the Vatnajökull glacier fit together. These images were collected to monitor changes in the glacier, which is Europe's largest

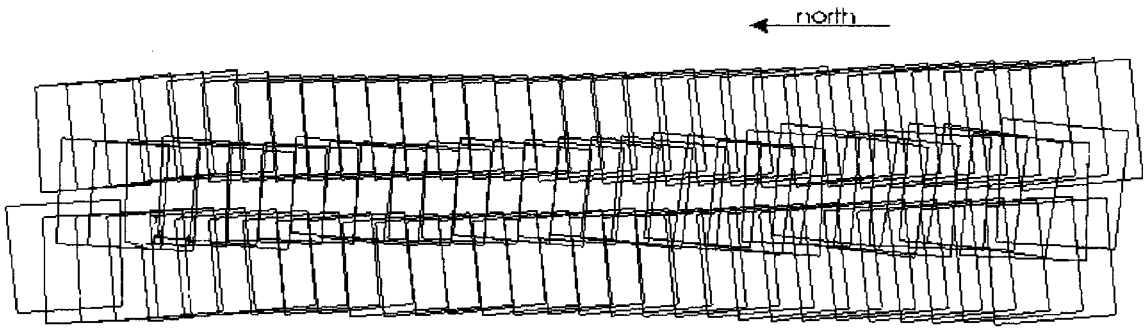


Fig 1.3: Satellite Image Block with Terrestrial Control of Vatnajökull Glacier (Bacher et al., 1999.)

and is located off the coast of Iceland. The use of an automatic target matching system, which can recognize and use ground features as targets, would make the collection of these points fast and easy.

Many applications of automatic target matching exist outside of the photogrammetry field. The computer vision community has developed many automatic target matching algorithms for the purpose of automating many common tasks currently performed by humans. Figure 1.4 is a stereo pair of images that were taken for the purpose of allowing a car to steer itself.

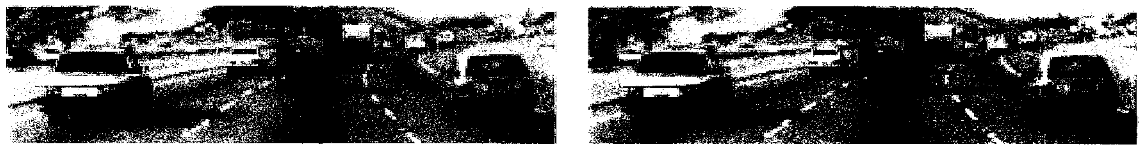


Fig 1.4: Example Of Vehicle Vision Stereo Pair (Olson, 2000.)

If the onboard computer of the vehicle in question was able to detect and track solid objects in the images, such as other vehicles on the road, the vehicle itself would be able to react to avoid collisions. Of course, the point has not yet been reached where a reliable automatic system has been developed and proven safe for vehicle use. Another use that the computer vision community has made of target matching algorithms is for fingerprint

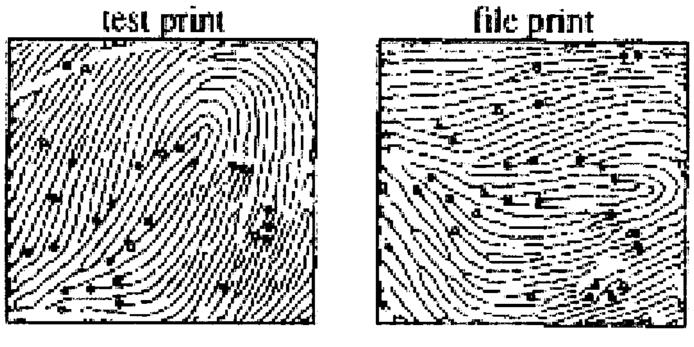


Fig 1.5: Example Of Fingerprint Stereo Pair (Mital et al., 1996.)

searches. Figure 1.5 shows the type of fingerprint images that would be compared, to determine if they are close matches. This type of automatic target matching routine would be used by police departments to narrow the search through digital fingerprint databases. Lastly, computer vision specialists have used automatic target matching to perform electronic component verification. This implies that newly assembled electronic boards would be passed under a camera to obtain an image, which would be compared to a master image, thereby checking that all electronic components are present.

Only a few examples of the applications of automatic target matching have been presented in this section. Applications of the type of uses that two of the five fields of study mentioned earlier have been presented. The other three fields have equally plentiful examples of how automatic target matching is used. Since the purpose of this thesis is to develop an automatic target matching program for the purposes of photogrammetry, this section has been mainly limited to photogrammetric applications of this technology. The computer vision applications were added simply to provide some contrast to the photogrammetric work. Automatic target matching has been applied to a wide variety of subjects and the number of applications will only increase over time.

1.3 Thesis Organization

The thesis begins by examining what automatic target matching is, who uses it and how it is applied to real world situations. The definition of automatic target matching has already been given in the introduction to this chapter; the other two questions will be dealt with in the next section. Knowing what it is and how it is used leads to the question of how can it be performed. To answer this, a detailed investigation into mathematical methods for the matching of targets in different images is needed. This research presents many of the common methods used in the identification, location and/or tracking of targets in digital images. An extensive literature review can be found in Chapter 2.

Chapter 3 is dedicated to describing the foundations of the software package INDMET. This is important since INDMET will be the platform upon which the target matching solution will be built. It was determined, through research, that a program consisting of subroutines that calculate epipolar lines, performs cross correlation analysis, and uses template least squares matching, would create a suitable matching routine. Mathematical methods of performing these three subroutines, as well as other issues related to the programming of these subroutines are presented in Chapters 4 and 5, respectively. After programming of the three target matching modules in the C++ language was completed, and these modules were successfully incorporated into INDMET, testing began. Using data supplied by the Applanix Corporation, each module was tested individually to ensure that it is performing as expected. The results of this testing can be found in Chapter 6. Conclusions and recommendations were made and presented in Chapter 7.

2.0 Automatic Target Matching

There is no complete unifying theory that has yet been created to automatically extract needed information from digital images. Approaches that work well to solve one problem have proven to not be general enough to solve other problems. This lack of a generalized theory on how automatic target matching can be performed, has led to the development of a wide diversity of approaches by both the photogrammetry community and the computer vision community.

The idea of target matching is a very general notion because it is unknown in advance which phenomena in one observed set correspond to thatin another set. In matching, it is important to realize that corresponding phenomena are actually the same object, viewed from different positions. It is also true that, in many cases, the counterpart of an element in one image might not even appear in some or all of the other images contained in that set. In addition, a correspondence problem can be created by the fact that no transformation is known and the image set has to be adjusted.

The term matching refers to identifying corresponding visual phenomena in image sequences, caused by the same phenomena in object space (Lemmens, 1988.). The purpose of target matching is surface recovery of 3-D object space from conjugate image pairs. It provides a better depth determination than passive mono techniques and its passive state makes it more generally applicable than active ranging (Lemmens, 1988.). Through the 3-D surface description of objects in the real world, it becomes possible, for instance, to compile orthophotos from single frames or 3-D impressions of spatial mono images by superposition. An example of what is meant by a 3-D impression of spatial mono images would be satellite imagery. Many of these approaches to target matching have been developed for the use of the United States military. They are used in applications such as guiding fire-and-forget missiles to their targets. For the application that is intended here, these approaches need only locate and match specific elliptical, black and white targets. The methods that are detailed here can be adapted for this purpose.

The algorithms that have been developed to match targets in digital imagery can be divided into three distinct categories. These categories are based on the way that an individual can look at a digital image. The first way would be to consider an image as simply a 2-D spatial distribution of functions of Electromagnetic intensities (Lemmens, 1988.). In other words, a group of grey values or signals that have been measured and recorded from a real life object. The second category can be referred to as feature-based matching. This method relies on matching features that exist in both images, such as points, lines or areas. The last category takes feature matching a little further and examines the relationships between phenomena. It is called relational matching.

2.1 The Human Visual System

Before methods designed to allow a computer to perform target matching are discussed, it seems appropriate to outline how we, as humans, perform target matching using our own eyes. All of the methods that have been developed to perform target matching are, in effect, a very poor attempt to reproduce human recognition and identification abilities. Human vision is a complex and efficient system that has an almost universal image processing capability. By comparison, the most state of the art artificial vision system can only cope with very specific tasks and applications. While many computer target matching systems can store and compare a specific pattern, the human visual system has the ability to learn, adapt and draw conclusions, when trying to locate or identify an object. Added to that, is the human's ability to store high-capacity information and draw upon this knowledge-based memory.

Much of photogrammetry deals with the extraction of 3-D information from 2-D images. Human stereo vision works in a similar fashion. Correspondence between the left and the right eyes produces an internal 3-D picture, which is Korn's so-called 'cyclopic eye' (Luhmann and Altrogge, 1986.). Due to its complexity, the human visual system is a matter of interest for many diverse fields. These fields were previously discussed in Section 1.4.

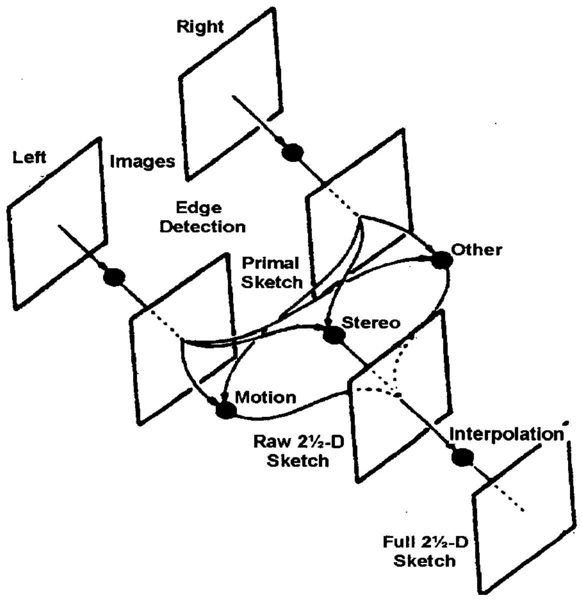


Fig 2.1: The Human Early Visual Processing System (Luhmann and Altrogge, 1986.)

Using a three-stage hierarchical structure, Marr and Poggio (1979) demonstrated their theory of how human vision actually functions. Once lines and edges are extracted from the view, the human eye performs a correspondence analysis of these features. This creates what is referred to as the 'primal sketch'. Then a raw 2½-D sketch is created, that uses all of the primitive features of the image. This is where motion, stereo and textural features enter the process. Interpolation is then used to recover 3-D surface information. The random dot stereograms that were developed by Julesz in 1971 have been used to prove this theory and Grimson successfully implemented this model in a computer vision system in 1981. Figure 2.1 outlines this human early visual processing system. A network of photosensitive cells is used by the human visual system to detect lines and edges. The sensitivity of these cells can be described by a combination of Gaussian functions (Luhmann and Altrogge, 1986.). High-level structures are created through the combination of linear features, and symbolic descriptions are used for the correspondence of images.

2.2 Signal Based Matching

Signal matching, or area matching, as it is sometimes called, relies on the relationship between the brightness or grey values in the neighbourhood of the target point. The resemblance between grey values is used to determine whether the image points are similar. Signal matching may be considered the digital continuation of the analog approach, which used cathode ray tubes and was introduced by Hobrough in the late 1950's (Lemmens, 1988). This type of matching is frequently applied since it has the advantage of being fully compatible with the tried and tested concepts of signal processing. Radiometric restoration, smoothing and edge detection are particularly enhanced through the use of a signal based solution. When low resolution image pairs are used, signal matching works well for smooth terrain with low relief. However, signal matching is not very reliable for high resolution image pairs where features, such as artificial buildings exist.

The disadvantages of signal matching are twofold. Firstly, only in the case of a simple shift does a correct match occur. Secondly, the grey value distributions don't correspond to physical entries. In other words, the first disadvantage refers to the geometric transformations and the second disadvantage refers to radiometric transformations.

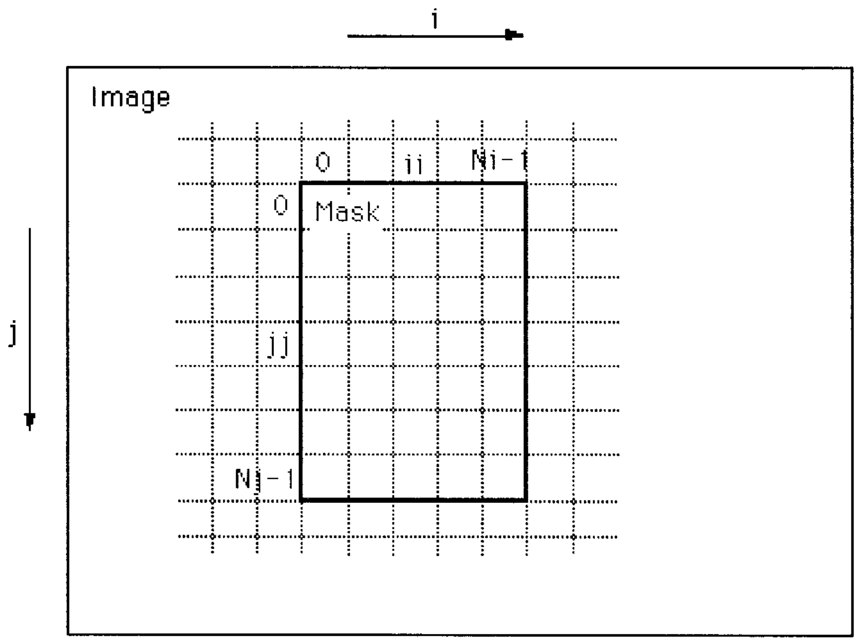


Fig 2.2: An Example of a Window Mask for Image Sample Extraction (Bourke, 1996.)

The simplest way of accomplishing this would be to use a statistical measure, such as cross correlation, to search the second image, comparing each search window with the template taken from the first image. The size of the search area is dependant on the size of the window mask chosen by the user and the precision to which the exterior orientation parameters are known. An example of a window mask is shown in Figure 2.2. The windows are based around the pixel located at the window's centre and the windows on the first and second images that best match, are taken as corresponding points. It is usual that a threshold value is set, which determines when the two matches are sufficiently similar.

2.2.1 Euclidean Distance

This type of matching is also known as the sum of squared distance (ssd) technique. Along with cross correlation analysis, Euclidean distance is a standard statistical technique that is used in target matching. This method is based on the assumption of square target and search areas. Using N to denote a single dimension of these target and search areas, indicates that the number of pixels in the area under consideration is simply N^2. Keeping this assumption in mind, the mathematical formula for performing the Euclidean distance technique can be expressed as follows.

$$ssd_{ij} = \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} (S_{i+x,\, j+y} - T_{x,y})^2 \qquad \textbf{[2.1]}$$

Where T denotes the target area and S denotes the search area. The coordinates of each pixel in the target area are given by x and y, while the coordinates of the pixels in the search area are represented by i and j respectively.

Unlike cross correlation analysis, where the maximum correlation value is looked for, the Euclidean distance technique searches for the minimum value of ssd_{ij}. When the minimum occurs, the best matching target in the search area will have been found. It is possible to expand Equation 2.1 so that minimizing the ssd_{ij} surface can be equivalent to maximizing the expression (Dew and Holmlund, 2000.). This expanded form is shown as Equation 2.2.

$$pseudo\_ssd_{ij} = 2 \cdot \sum_{x=1}^{N} \sum_{y=1}^{N} (S_{i+x,\, j+y} \cdot T_{x,y}) - \sum_{y=1}^{N} (S_{i+x,\, j+y})^2 \qquad \textbf{[2.2]}$$

If bias and normalization terms are added to the terms T and S in Equation 2.2, then the second half of the equation ($\sum S^2$) becomes invariant and the pseudo_ssd value becomes equivalent to the cross correlation value (ρ). Therefore, the only real separation between the Euclidean distance technique and cross correlation analysis is the addition of the normal and biasing terms. This would seem to indicate that cross correlation is a better search engine, since it takes these factors into account. The addition of normalization and bias terms are not necessarily a good thing. In low contrast search areas, where the correlation surface has shallow slopes and broad maxima, normalization may degrade the accuracy of the surface peak location (Dew and Holmlund, 2000.). In fact, in the extreme case of no contrast, the cross correlation expression becomes undefined. Therefore, the Euclidean distance technique should provide better quality results in this situation.

2.2.2 Cross Correlation

Also known as grey value correlation analysis, this type of signal analysis is very common in image analysis. It is used as a similarity measure to search the image, finding regions that closely resemble a given target area. Since the spatial type of this analysis was used in the addition to INDMET, a mathematical model of how cross correlation analysis is performed is discussed in Chapter 4. For now, this section will deal with the topics of autocorrelation and how Fast Fourier Transforms can be used in cross correlation analysis.

Autocorrelation is when the correlation is calculated between a series and a lagged version of itself. If a high correlation value is found, it possibly could indicate a periodicity in the signal of the corresponding time duration. To find the correlation coefficient of a series, denoted by $\{x_o,\ldots,x_{n-1}\}$, at a time delayed lag of k, the following formula can be used (Bourke, 1996.).

$$autocorr(k) = \frac{\sum_{i=0}^{n-1} (x_i - \bar{x}) \cdot (x_{i+k} - \bar{x})}{\sum_{i=0}^{n-1} (x_i - \bar{x})^2} \qquad \textbf{[2.3]}$$

Where $\bar{x}$ is the mean average of the image pixels series and n is the number of terms in the series. The counter i+k will eventually expand past the end of the series, at which point two options will be available. The series can either be considered to become zero or, as is standard with the Fourier approach, it could wrap back on itself (Bourke, 1996.). This would mean that the index would become (i+k) mod (n). When the correlation coefficient is calculated for all lags, k = {0,...,n-1}, the resulting series is called the autocorrelation series, or the correlogram (Bourke, 1996.). This new series could be calculated using Equation 2.3 or using the Fourier transform shown below

$$fourier[attocorr(k)] = |fourier[x_i]|^2 \qquad \textbf{[2.4]}$$

In other words, one could transform the series into the frequency domain, taking the modulus of each spectral coefficient and then perform the inverse transformation. Depending on the normalization factor used with the Fast Fourier Transform (FFT), scaling by N may be needed. This particular method of performing autocorrelation is especially useful for long series where the time to compute each correlation value can become excessive. It is important to note that the Fourier model shown here is a special case and that the Fourier transform for the cross correlation function is the product of the

Fourier transform of the first series and the complex conjugate of the Fourier transform of the second series (Bourke, 1996.).

Many spatially based cross correlation algorithms have been used in template matching solutions because no simple and efficient expression of the normalized form of correlation has been developed in the frequency domain. Lewis (1995) proposed a correlation technique employing Fast Fourier Transforms, which he used to track target images. His model is based on the denominator of the spatial cross correlation technique, which can be found as Equation 4.6 in Chapter 4. Assuming that we have images g_L and g_R, in which the mean value has already been removed.

$$g_L'(x, y) \equiv g_L(x, y) - \overline{g_L} \quad \textbf{[2.5a]}$$

$$g_R'(x, y) \equiv g_R(x, y) - \overline{g_R} \quad \textbf{[2.5b]}$$

Rewriting Equation 4.7 from Chapter 4 into this form gives the following equation

$$\overset{num}{\gamma}(i, j) = \sum g_L'(x, y) \cdot g_R'(x-i, y-j) \quad \textbf{[2.6]}$$

Where, x, y are the image coordinates and i,j are the position of the masking window containing the relevant feature. Assuming that the mask window has a size of M^2 pixels and the feature of interest, which is contained in the window, has a size of N^2 pixels, then Equation 2.6 requires $N^2 (M - N + 1)^2$ additions and $N^2 (M - N + 1)^2$ multiplications. Equation 2.6 is a direct convolution of the image with the reversed feature $g_R'(-x,-y)$. It can be computed through the Fourier transform equation

$$fourier^{-1}\{fourier(g_L) \cdot fourier^{*}(g_R)\} \quad \textbf{[2.7]}$$

The complex conjugate reverses the feature via the Fourier transform property (Lewis, 1995.)

$$fourier \cdot \overset{*}{g_L}(-x) = F^*(\omega) \qquad [2.8]$$

To implement most FFT algorithm, it is necessary that images g_L and g_R be padded with zeros to a common power of two. To compute the transform then requires $12M^2\log_2(M)$ multiplications and $18M^2\log_2(M)$ additions or subtractions. This means that if M is significantly larger than N, the spatial domain only requires N^2M^2 total operations and is, therefore, faster than operations in the spectral domain. The transforms in the spectral domain become more efficient the closer N is to M and when N and M are both large numbers.

2.2.3 Least Squares Matching

Due to the fact that a template least squares matching solution was implemented in the addition to INDMET, a model of this type of matching is presented in Chapter 4. Least squares matching is an extremely versatile approach to target matching. As such, it is widely used and applied in many target matching solutions. Least squares matching is a technique where the optimum match is defined by the transformation of the elements of one array into another, which minimizes the residual differences between the remaining grey values. This produces the optimal transformation, which is required to find the optimal match.

There are four factors that contribute to grey values. These factors are illumination, reflectivity, geometry and viewpoint. All of these factors should be considered when performing a target matching solution, if unambiguous matches are to be made. However, it is almost certain that only the position and orientations of the camera will be known. Attempts have been made to describe the surface reflectivity, by introducing an optical density function (Lemmens, 1988.). The formulation of this function is very complex and to obtain a tractable method, simplifications are needed. Added to this, is the fact that there will always exist geometric and radial distortions in the camera that need to be corrected. To improve matching accuracy and deal with these distortions, Gruen (1985) and Rosenholm (1987) recommend the least squares method. The study conducted by Rosenholm (1987) involved the effect of window size on the precision and reliability of least squares matching solutions. He found that window sizes of 21x21 pixels and 31x31 pixels were optimal for precision and that larger window sizes optimized reliability.

There are two main disadvantages to using least squares matching. The resampling that is required can be time-consuming, tying up computer resources. The second disadvantage is that very accurate initial approximates are required for the least squares iteration process to converge. On the other hand, least squares can return highly precise answers. The precision of the answers returned by this method has reach 1/20th of a pixel. Measures that indicate the precision and reliability of each answer are easily obtainable from the least squares calculation. Least squares matching requires no image rectification before it is applied since simultaneous geometrical image shaping and radiometric adjustment are performed by the adjustment. It is able to perform area based and edge based analysis. It can function in hierarchical mode (Gruen, 1985.). It is usable as a derivative operator based matching procedure, since the first order derivatives that can generate the vector of observations leads to the coefficients of the design matrix, which are the second order derivatives (Gruen, 1985.). Least squares can eliminate low signal regions that will not provide a reliable solution.

2.3 Feature-Based Matching

Unlike signal matching procedures, features can be detected in both images, which allow the primal sketch to be created. Features can be points, lines or areas (Lemmens, 1988). These features are usually combined with the epipolar geometry to limit the amount of area in which features are to be considered. The search for points is done by one of several characteristic point detectors, or interest operators, that have been developed over the years. The search for lines and shapes is commonly accomplished through the use of edge detectors combined with line following and vectorization techniques (Lemmens, 1988). Since this thesis is primarily concerned with automatically gathering points on stereo pairs of images, little attention will be paid to the matching of lines or areas.

To use feature matching techniques, several criteria must be met. The first is that the feature must be well defined on the image. This is needed for the correspondence analysis. Localization is the next criteria. The positional precision of the feature must be defined. Next, the attributes that are suitable for matching must be defined. Finally, the noise, geometric and radiometric distortion tolerances must be determined.

Selecting corresponding feature takes several steps, the first of which is the determination of an upper bound for the parallaxes. This is done to limit the search space and reduce computation time. Then a measure of the similarity between the two points is determined and used as a set of initial weights or costs (Lemmens, 1988). An example of this would be to define a window of n pixels by n pixels around each feature in each image and perform a cross correlation analysis with possible counterparts. The value of the correlation coefficient ρ can then be used as a likelihood measure. It is possible, however, that this similarity measure might not be enough to lead to a unique match. Techniques based on relaxation, minimal path computation using dynamic programming, robust statistics and simulated annealing have been shown to be successful at solving this problem (Lemmens, 1988).

The success or failure of a matching routine based on features in the images is influenced by three properties (Lemmens, 1988). The discreetness of individual points, which is a measure of the distinction of the point from its surroundings, is critical to a feature matching solution. The more discreet the point, the better chance the computer has to match it in the second image. Next, the similarity, which is a measure of how closely two points resemble each other. Finally, the consistency provides a measure of how well a particular match conforms to surrounding matches, based on some general object model. An example of such a model would be if the object surface is smooth, attended by a limited number of surface discontinuities or the surface is a tiled plane (Lemmens, 1988).

These properties indicate that there must be three stages to a feature-based target matching solution. Firstly, distinct features must be located and selected from the image. This corresponds to the distinction property and can be accomplished through use of some sort of interest operator. Next, candidate features, which may form possible matches, should be compared and selected based on one or more similarity measures. Finally, the candidate list must be thinned, consistent with some pre-chosen object model, until only unique matches remain. This corresponds to the consistency check, which is designed to determine which candidates are the correct matches.

Many feature-matching based routines have been created over the years. These routines employ a wide variety of techniques to find targets, but they all involve one principle, the use of image features to locate targets. The following subsections detail many of the common feature matching routines, which are available. Each routine by itself is not a target matching solution, they are designed to be mixed and matched, depending on the user's particular matching needs. Some, like epipolar geometry, limit the search area, while others locate points, but they all depend on image features to function.

2.3.1 Epipolar Geometry

A method for calculating Epipolar lines is described in Section 3.1. The epipolar constraint is the most common method for solving the correspondence problem in digital photogrammetry. Forlani et al used epipolar geometry to automatically determine the location of tie points in digital close range images. The idea behind the epipolar constraint is to project the intersection of the epipolar plane onto the image plane. The epipolar plane is formed from the rays emanating from the object (A) pointing toward the optical centres (O1, O2) of the camera stations (see Figure 2.3). By locating the target on

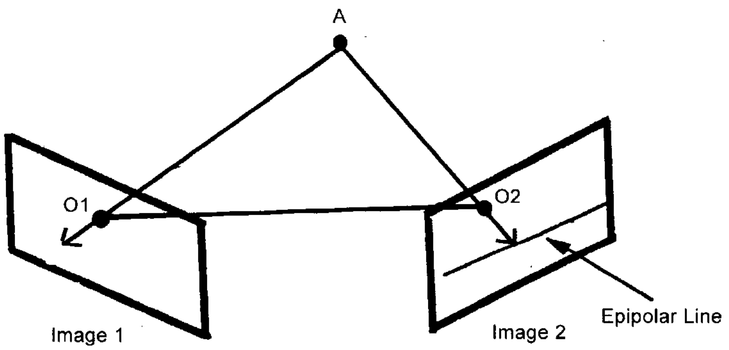


Fig 2.3: The Epipolar Matching Method

the first image and projecting this intersection of planes onto the second image plane, a line will be formed that should be coincident with the corresponding target on the second image. This method contains two steps. First, the estimation of the relative orientation parameters of the cameras in question must be performed. Second, the matching of corresponding target images must be performed. To match a target image in one view with the corresponding image in another view, the coordinates of the first target view and the orientation parameters are used to calculate coordinates for the target in the second view (Chen et al., 1994.).

Problems may arise with using the epipolar line method due to a number of factors. Firstly, geometric and radiometric distortions might interfere with the line-target intersection. This, coupled with poor estimation of the camera orientation parameters, could throw the epipolar line off the target. Secondly, the object might not appear in all images in the set, due to the fact that some other object might cause the target to be occluded. Thirdly, ambiguities may arise due to the fact that other target images may lie along the epipolar line. This can have the effect of causing incorrect matches. The cost of computing the epipolar lines increases exponentially as the number of images in the set increases.

A number of methods have been developed to deal with the problems that occur when using epipolar geometry. The first uses multi-view constraints, such as the geometric epipolar constraints and uniqueness constraints. These constraints improve the reliability of matching by doing such things as eliminating targets that are occluded or ambiguous in some images. It is also possible that these constraints could be used to combine the matching process with a bundle adjustment (Chen et al., 1994.). The orientation information is usually difficult to accurately obtain. Without this information the correspondence problem cannot be solved. This problem can be overcome by combining epipolar target matching with the bundle adjustment technique.

When computing the epipolar line, one of two matrices must be created from known information. The fundamental matrix is used when interior and/or exterior image information is unavailable. This information covers such topics as the orientation angles, camera station locations, lens distortion and the location of the principal point within the image. It encodes all the geometric constraints available, given two images locked in a ridge scene. To define the fundamental matrix, it is first necessary to define two images, I_1 and I_2, with points locations, $p_1 = (x_1, y_1, z_h)^T$ and $p_2 = (x_2, y_2, z_h)^T$. If these points are homogeneous coordinates, then the relationship between these points, assuming they are projected from the same scene point, is as follows.

$$p_2^T \cdot F \cdot p_1 = 0 \qquad \textbf{[2.9]}$$

Where F is the 3x3 fundamental matrix. Both the photogrammetry community and the computer vision community have developed many techniques for the computation of F. Zhang (1998) presents many of these techniques as well as methods for computing how well the fundamental matrix has been determined. The second matrix, called the essential matrix, can be created when the interior and/or exterior image information is known. It is equivalent to the matrix when the image orientations are known. The essential matrix can be defined as the following.

$$E = R \cdot [T]_\times \qquad \textbf{[2.10]}$$

Where R is the rotation matrix that can convert between the two images and $[T]_x$ is the cross product of the translation vector between the two images. What is meant by the cross product of the translation vector is demonstrated below.

$$T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}, \ [T]_\times = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \qquad \textbf{[2.11]}$$

2.3.2 The Förstner Interest Operator

Förstner developed an interest operator in 1986, based on evaluation of the covariance matrix (C) of the gradient images of a neighbourhood of pixels. For the purposes of this Section, the gradient images will be called g_x and g_y. The assumption will also be made that a 5x5 pixel sample window has been extracted from the original image. The covariance matrix is therefore the following

$$C = \begin{bmatrix} \sum g_x^2 & \sum g_x g_y \\ \sum g_x g_y & \sum g_y^2 \end{bmatrix} \qquad \textbf{[2.12]}$$

The gradient terms g_x and g_y can be calculated using normal, Roberts, Pewitt or Sobel gradient techniques. Since the covariance matrix of the gradients is being used to find the match, it determines the precision to which the match is made. This becomes a distinct advantage because it allows the user to select features beforehand, based on their suitability to give precise matches. Unfortunately, this type of interest operator is perturbed by edges, where the match will not be defined in the direction along the edge. To combat this, the error ellipse associated with the match, must be kept small and closely equivalent to a circle.

The error ellipse is determined by the eignvalues λ_1 and λ_2, where λ_1 is greater than λ_2. Equation 2.13 can be used as a measure of how elongated the error ellipse has become.

$$E = \left(\frac{\lambda_1}{\lambda_2}\right)^2 \qquad \textbf{[2.13]}$$

Alternatively, the eignvalues computation can be avoided by computing a direct measure (Lemmens, 1988).

$$q = 1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 = \frac{4 \cdot \lambda_1 \cdot \lambda_2}{(\lambda_1 + \lambda_2)^2} \qquad \textbf{[2.14]}$$

$$\lambda_1 \cdot \lambda_2 = C_{xx} \cdot C_{yy} - C_{xy}{}^2 \qquad \textbf{[2.15]}$$

$$\lambda_1 + \lambda_2 = C_{xx} + C_{yy} \qquad \textbf{[2.16]}$$

Using Equations 2.14, 2.15 and 2.16 the direct measure for determining the shape of the error ellipse becomes

$$q = \frac{4 \cdot \det(C)}{(C^T)^2} \qquad \textbf{[2.17]}$$

The size of the error ellipse can be determined by Equation 2.18.

$$w = \frac{\det(C)}{C^T} \qquad \textbf{[2.18]}$$

Unlike the Moravec operator (see Section 2.3.5), this operator is not dependant on the orientation of the feature in question, it is rotationally invariant. It also has been proven that this operator has some useful properties when detecting features such as edges and circles (Förstner and Gülch, 1987.). This operator has been used with cross correlation as a measure of similarity and centre of gravity calculations to determine subpixel locations of the centre of features (Lemmens, 1988).

2.3.3 The Dreschler Interest Operator

Grey values may be looked at in the same way that digital elevation models are viewed. In other words, they can be seen to create a curved plane, where the actual grey value is a 'height' above a datum. Differential geometry tells us that Gaussian curvature is invariant with respect to transformation (Lemmens, 1988). This is a very good measure to have when performing stereo matching. A second differentiable, smooth, surface can be represented by the following equation

$$r(u,v) = (x(u,v), y(u,v), z(u,v)) \tag{2.19}$$

Where x, y and z are Cartesian coordinates and u and v are surface coordinates. Using the explicit expression for a smooth surface, g = f (x, y), allows the following equation, equivalent to Equation 2.19, to be written

$$r(u,v) = (u, v, g(u,v)) \tag{2.20}$$

This means that the second partial derivatives of Equation 2.20 become

$$\begin{aligned} r_{UU} &= (0,\ 0,\ g_{UU}) \\ r_{UV} &= (0,\ 0,\ g_{UV}) \\ r_{VV} &= (0,\ 0,\ g_{VV}) \end{aligned} \tag{2.21}$$

The curvature information is now enclosed in a symmetric tensor (Lemmens, 1988).

$$\begin{bmatrix} \frac{g_{UU}}{\sqrt{1+g_U{}^2+g_V{}^2}} & \frac{g_{UV}}{\sqrt{1+g_U{}^2+g_V{}^2}} \\ \frac{g_{UV}}{\sqrt{1+g_U{}^2+g_V{}^2}} & \frac{g_{VV}}{\sqrt{1+g_U{}^2+g_V{}^2}} \end{bmatrix} \tag{2.22}$$

The principal curvature can now be found from eignvalue analysis of the symmetric matrix. This is true because the third coordinate of the partial derivatives is the only one to differ from zero and the kind of curvature, not its absolute value, is what is being sought.

$$\begin{bmatrix} g_{XX} & g_{XY} \\ g_{XY} & g_{YY} \end{bmatrix} \qquad \textbf{[2.23]}$$

The principal curvatures, k_1 and k_2, are now calculated using Equation 2.24. Note k_1 must be greater than k_2.

$$k_1, k_2 = \frac{1}{2} \cdot (g_{XX} + g_{YY} \pm \sqrt{(g_{XX} + g_{YY})^2 - 4 \cdot (g_{XX} \cdot g_{YY} - g_{XY}{}^2)}) \qquad \textbf{[2.24]}$$

The principal direction of the curvature can be calculated using Equation 2.25.

$$D = \frac{1}{2} \cdot a\tan\left(\frac{2g_{XX}}{(g_{XX} - g_{YY})}\right) \qquad \textbf{[2.25]}$$

The signs of k_1 and k_2 can now be used to perform a similarity check. An example case would be that k_1 is positive and k_2 is negative, which would indicate a saddle point.

2.3.4 The Marr-Hildreth Interest Operator

Marr and Hildreth based their interest operator on the operation of the human visual system. It is designed to detect grey level changes across a digital image. It does this by smoothing the input image with a Gaussian kernel and then locating the edges by searching for *zero-crossings* of the Laplacian of the smoothed image. It uses the second derivative of the Laplacian of a Gaussian function.

$$\nabla^2 S(x, y) \times G(x, y) \qquad \textbf{[2.26]}$$

Where S(x,y) is the 2-D normal distribution in 1-D form (Lemmens, 1988). S(x,y) is given by the following equation.

$$S(x, y) = \frac{1}{\sigma(2\pi)^{0.5}} e^{\frac{-x^2}{2\sigma^2}} \qquad \textbf{[2.27]}$$

The Laplacian operator ∇^2, which is used on the image G, can be calculated by Equation 2.28.

$$\nabla^2 G = \frac{d^2 G}{dx^2} + \frac{d^2 G}{dy^2} = g_{XX} + g_{YY} \qquad \textbf{[2.28]}$$

The repeated convolution of a two by two unweighted smoothing filter can be used to approximate the needed Gaussian.

By convoluting the Gaussian with the Laplacian, ∇^2S can be found. When a point of inflection is found, the Laplacian operator will give zero. Generally, on a raster image, the point of inflection can be precisely found, since it will likely lie within a single pixel. Therefore, searching for a positive and a negative response and then linearly interpolating between these values will identify the point of zero crossing. The sign and orientation of the line that contains the zero crossing can be used to perform the similarity check. The sign is determined by taking the first operator response, either positive or negative, when moving along an epipolar line (Lemmens, 1988).

2.3.5 The Moravec Interest Operator

The Moravec interest operator is designed to locate unique points in images by using the assumption that there will be high variances in all directions outward from the point in question (Luhmann and Altrogge, 1986). To reduce computation time, the test for high variance is only conducted on samples that are taken in the four cardinal directions as well as the two diagonal directions. The variance is calculated as the sum of the squared grey value difference between neighbouring pixels (Lemmens, 1988). Originally the Moravec interest operator was defined for a 5x5 window and the equations are as follows:

$$M_1(i,j) = \frac{1}{20}\sum_{K=-2}^{2}\sum_{l=-2}^{1}(g(i+k,j+l) - g(i+k,j+l+1))^2 \qquad \textbf{[2.29a]}$$

$$M_2(i,j) = \frac{1}{20}\sum_{K=-2}^{1}\sum_{l=-2}^{2}(g(i+k,j+l) - g(i+k+1,j+l))^2 \qquad \textbf{[2.29b]}$$

$$M_3(i,j) = \frac{1}{16}\sum_{K=-2}^{1}\sum_{l=-2}^{1}(g(i+k,j+l) - g(i+k+1,j+l+1))^2 \qquad \textbf{[2.29c]}$$

$$M_4(i,j) = \frac{1}{16}\sum_{K=-2}^{1}\sum_{l=-2}^{1}(g(i+k,j+l+1) - g(i+k+1,j+l))^2 \qquad \textbf{[2.29d]}$$

The feature that is the object of the search (M), will be located at the minimum of these four values.

$$M = \min(M_i);\ i = \{1,2,3,4\} \qquad \textbf{[2.30]}$$

A threshold is set, so that any point (i,j) that exceeds the threshold will be excepted as a characteristic point (Lemmens, 1988). This will cause a series of responses from the operator. A distinct point requires that non-maximum suppression be used. The test for similarity between characteristic points of conjugate images that this operator offers, involves the use of the sum of the squares of the differences between surrounding windows.

Dreschler-Fischer (1987) detailed how this method of point matching could be modified so that is could operate on colour images. Basically, two methods were suggested, the first involved the breaking of the single grey value into its three component colours, and operating on each of these colours separately. The colour band that gives the maximum response is the one to consider when point matching. The second approach involves calculating the directional variances of the spectral band vectors. Here, the squared differences of the vectors are calculated and, like the single band approach, the response should exceed some predetermined threshold (Lemmens, 1988).

This method of point matching has many inherent problems. While it is easy to implement, it does not find the actual feature. A shift is introduced that is proportionally equal to the size of the operator. If the target is of low resolution, an extended operator signal could result. Lastly, the operator is rotationally dependant.

2.4 Relation-Based Matching

Also known as structural matching, this method of performing target matching has become increasingly popular over the last few years. This type of matching establishes a correspondence or homomorphism between primitives of two structural sets. A structural description is defined as a set of primitives and their interrelationships (Wang, 1998.). This means that a structural description of an image might consist of image features and information on how these features relate to each other. Due to the nature of structural image matching techniques, no a-prior information is required and therefore, structural matching may be considered a fully automated technique. This is true because structural image matching techniques are able to utilize topological and geometrical relations among image features. The main problems to be solved when implementing a structural matching solution involve the efficient acquisition of structural descriptions and the operational approach for their matching (Wang, 1998.).

Primitives, which are the bases for structural matching, are simple image features that when combined form more complex features. Therefore, points, lines and areas are primitives. Each primitive and relation can be described by several attributes (Wang, 1998.). To describe a point primitive, one might refer to its coordinates, i.e. (x , y), or its grey value (g) or possibly its gradient (t). The relation between two lines can possibly be described by their point of intersection (p), if there is one, or vectors that lie on the line (V_1 , V_2), or the angle at which the lines intersect (θ).

The overall goal of structural matching is to find the best possible match, or correspondence, between the primitives and relations of the two structural sets. The sheer number of primitives and relations that can be generated by the two images can be enormous. Added to this is the fact that the two images will probably be partially over lapping. Therefore, the correct match may consist of only a partial correspondence between all the primitives and relations. Since one primitive or relation may match several, one or no primitives or relations from the second image, there exists the possibility that several target candidates may exist. This means that the time needed by the computer to search for a suitable match may be excessive. To ensure that the structural matching search is performed quickly and correctly, thought must be given to such topics as evaluation functions, search methods, correctness checks and efficient extraction of structural descriptions.

Evaluation functions are designed to test the goodness-of-fit of a possible match. They give a quantifiable value, which is necessary for the computer to select the best match, based on the structural sets under consideration. An example of an evaluation function would be maximum-likelihood estimation. Assuming that you have two structural sets, D_L and D_R, the theory of maximum-likelihood estimation indicates that the best match h_b should have the maximum conditional probability among all possible matches. Using Bayes' formula the probability function can be written as follows:

$$P\left(\frac{h_i}{D_L, D_R}\right) = \frac{P\left(\frac{D_L, D_R}{h_i}\right) \cdot P(h_i)}{P(D_L, D_R)} \qquad \textbf{[2.31]}$$

Equation 2.31 can be maximized by simply maximizing the term $P(D_L, D_R / h_i)$. Therefore, the joint probability $P(D_L, D_R / h_i)$ can become the evaluation criteria, upon which the goodness-of-fit between two structural sets can be judged. In reality, the joint probability distribution $P(D_L, D_R / h_i)$ is calculated through the similarity of the attributes of the primitives and relations (Wang, 1998.). A different weight is usually assigned to each attribute.

The method used to search the image becomes very important in a structure-based matching solution because it, above all else, determines how fast the program will run. Most structural matching routines that currently exist employ a tree structure to search their images. Tree structures can be broken down into two types. The first is a blind search, which treats every node in the tree structure equally. This means that the search could potentially take a long time, since the program will methodically investigate each node on the tree structure in the order that they appear. The second type is an informed tree structure search. This type of tree search uses some measure to assess the probability of finding a match and testes the nodes that have the highest probability of success. Vosselman (1994) performed an investigation into tree search methods.

Once the search has located a possible match and the evaluation function has refined that match, a check must be performed to see if the match is actually correct. The check can be performed by using the magnitude of the match's probability and any existing geometric conditions (Wang, 1998.). The coplanarity condition is an example of a check that can be performed, when two perspective images have been used. Once the coplanarity condition has been enforced, the parameters of the coplanar equation and the variance can be determined. The variance can be used to ascertain the correctness of the match, since the a-priori variance of the points is available from the point extraction algorithm (Wang, 1998.).

Extraction of points, lines and areas, along with their corresponding relationships from an image, is vital to performing a structural matching solution. Due to the large size of the majority of the original images used in matching, a Gaussian image pyramid is created to reduce the amount of information fed into the program. The structural matching routine will only be applied to the highest level of the pyramid. This level should contain all the prepossessed information, which will have undergone such procedures as edge-preserving smoothing. After structural matching has successfully been performed on this pyramid layer, the matched point will be transferred to the next level and new points will be added between each pyramid level until the original image is reached. In this way, points can be extracted from an image for structural matching purposes. Extraction of other lines and areas can be performed by previously described procedures, such as the Förstner Interest operator.

2.4.1 Multiresolution Image Matching

This technique is also known as coarse-to-fine image matching. As discussed previously, pyramid matching can be a very useful technique for quickly locating and extracting image features. Like epipolar geometry, multiresolution image matching is a technique that is designed to reduce the amount of image area that is to be searched.

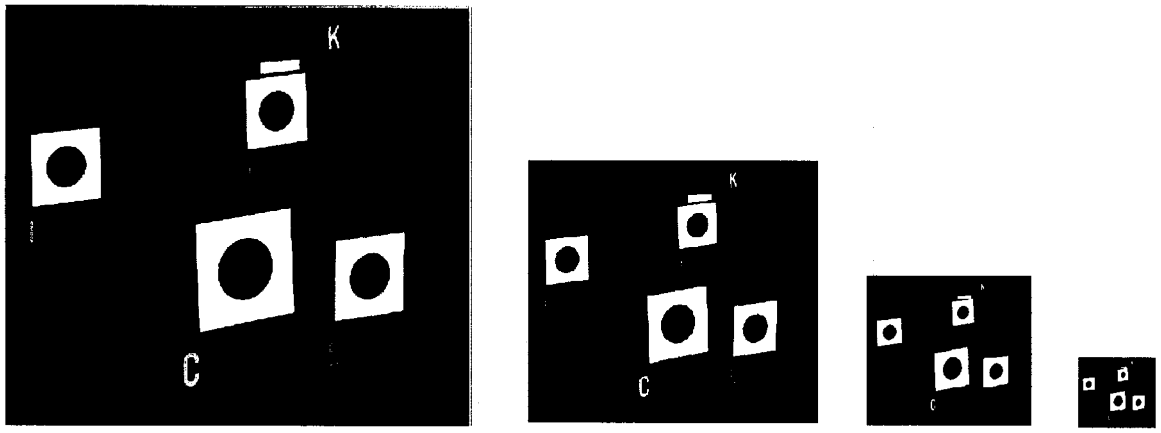


Fig 2.4: A Regular Image Pyramid of a Target Field

Unlike epipolar geometry, which simply limits the search area, multiresoultion image matching adapts the image to the search process, by minimizing the noise and extraneous detail that will interfere with the matching process. At the same time, disk space is not sacrificed. Using even the smallest possible reduction factor of 2, the amount of data storage only increases by 30% of the original image.

The idea behind an image pyramid is that the same scene will be represented at different resolutions, as shown in Figure 2.4. This idea of a stack of digital images depicting the same scene with decreasing resolution is closely related to the concept of scale space, where the scale is simply an additional continuous dimension of the image in question (see Figure 2.5). The reduction of the original image can be performed by using either a Gaussian or a Mean smoothing filter. These filters both perform averaging on a neighbourhood of pixels, allowing these pixels to be compressed into a single grey value. The difference is that the Gaussian filter uses a different kernel from the Mean filter; it is specifically designed to approximate a 3-D bell-curve or 'hump'.

Since all levels of the pyramid are derived from the original image, it is this image that will have the greatest resolution and, therefore, will be located at the base of the

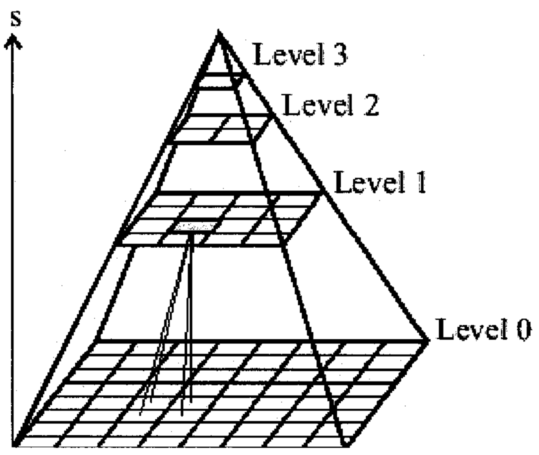


Fig 2.5: Structure of a Regular Image Pyramid (Kropatsch et al., 2000.)

pyramid. Using a low pass filter, such as Gaussian filter, the image pyramid can be created through iteration of the following formula.

$$L_{i+1} = G_{\sigma} \cdot L_{i} \qquad [2.31]$$

Where G_σ is the Gaussian kernel for the particular standard deviation σ. The variable L_i represents the pyramid image at level i and L_{i+1} represents the next pyramid level. The degree to which the image is smoothed at each level, depends on the standard deviation that is used. In many cases G_σ is approximated by an M x M binomial filter. The value of M in the binomial filter determines what the standard deviation is for the approximated Gaussian filter. If $M = 3$, then $\sigma = 0.71$ or if $M = 5$, $\sigma = 1$. No matter what filter is used the size of G_σ is referred to as the reduction window. The second control factor on the creation of an image pyramid is the reduction factor (n). The reduction factor controls how low the resolution will go on each iteration and how pixels relate to each other between pyramid layers. Once the iteration process has been completed, an image matching routine, such as the Förstner Interest operator, is used on the apex image to locate those image features that are of interest.

There are two relationships between pixels that exist in this scheme. Each pixel has a 'horizontal' neighborhood relationship within the levels of the pyramid and a 'vertical' father-son relation between adjacent levels (Kropatsch et al., 2000.). Each pixel in the pyramid, except for the base image, has a set of children. These children are always contained in a higher resolution image than their father. Each pixel, except for the image at the apex, has a father in the level above it. These relationships are shown in Figure 2.6. This type of pyramid is known as a regular image pyramid.

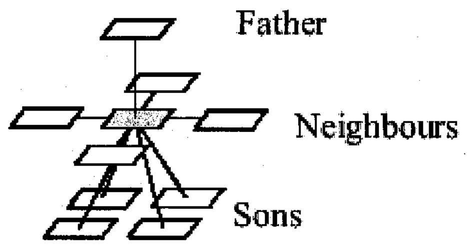


Fig 2.6: Relationships Of A Pixel
(Kropatsch et al., 2000.)

Another type of image pyramid exists, the so-called irregular image pyramid. In the irregular image pyramid scheme cells are considered instead of pixels. These cells can contain the original image grey values to symbolic information, such as the representation of an image edge that was extracted with some feature extraction algorithm. Also, the neighbor and parent relationships become generalized due to the fact that the cells contain generalized information. These relationships are now contained in a graph, where the nodes of the graph correspond to the cells of the pyramid and the edges indicate relationships. These edges need to be labeled so the 'vertical' father-son relationships can be distinguished from the 'horizontal' neighborhood relationships. A more general creation process is needed to allow for non-grey value data. This process can still include the idea of smoothing followed by the selection of surviving cells, but new criteria need to be used to accomplish this. An example of this would be a pyramid of edge segment cells, where an edge strength is assigned to each cell (Kropatsch et al., 2000.). The routine could then check the neighbors of a cell to see if any have a stronger edge strength. The strongest cells then survive to move up the pyramid and the weaker neighbors get assigned to that cell as children.

These irregular image pyramids, which can process image features in a hierarchical way, might also be called feature pyramids. However, digital photogrammetry uses the term feature pyramid to refer to the extraction of features from all levels of an iconic regular pyramid. Therefore, all features that have been extracted from the same level of this image pyramid form a similar level of the feature pyramid. In a sense, this means that the photogrammetric feature pyramid represents a hierarchical structure that contains symbolic information about the digital image. The major difference between feature and irregular pyramids is that the feature pyramid contains fewer relationships than in the irregular pyramid structure. This is due to the fact that the feature pyramids, while maintaining the 'horizontal' neighbor relations, does not explicitly contain any information about the 'vertical' parental relationships. Feature pyramids do not contain this 'vertical' information because the features forming individual levels were created independently. While this might seem like a disadvantage of feature pyramids, the inherent smoothing associated with irregular pyramids, which causes the strongest features to graduate up the pyramid, can cause some important image features to disappear from certain pyramid levels. This means that relevant features could be lost from consideration.

2.4.2 Tree Search Matching

Also called relational matching, this type of search procedure is very similar to the irregular pyramid previously discussed. It is designed to find common features by comparing the attribute values between two relational descriptions. The search tree contains all possible combination of features, but only certain paths along the tree will be followed. Since common tree search algorithms have combinatorial or exponential complexity, the time needed to complete the search rapidly increases as the number of levels in the tree increase. Due to the inherently large nature of a search tree it is common practice to incorporate heuristic search strategies with the standard search method. This is done to reduce the average computation for a tree search.

Figure 2.7 represents the mapping of two structural sets, $p_i \in P$ and $q_j \in Q$, where all possible mappings of P to Q are shown. These two relational descriptions could describe many things. In image matching, it is usual that P would represent features and

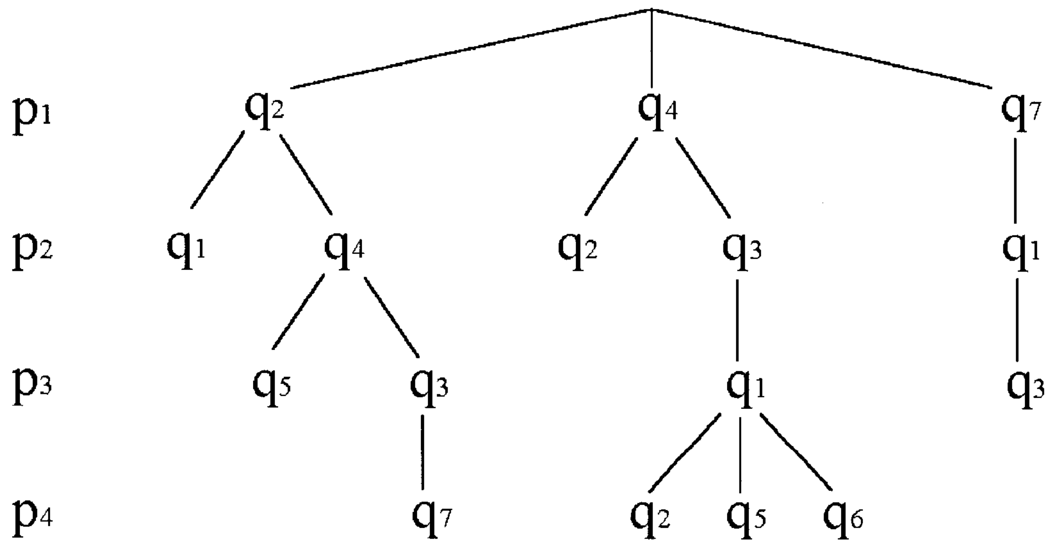


Fig 2.7: Search Tree Containing Relationships Between Two Relational Descriptions (Vosselman, 1994.)

Q would represent the segmented image. This would mean that the features of set P are called the units of the tree search and the segments are called the labels (Vosselman, 1994.). In each level in the search tree, which is shown in Figure 2.7, a label (q_j) is sought for each unit (p_i). Once a label q_j is found for unit p_1 then a label for unit p_2 is sought on that particular branch of the tree. An example of this would be if label q_4 were chosen for unit p_1 then only labels q_2 and q_3 are considered for unit p_2. Therefore, each mapping of the root node to a leaf node represents a different mapping of P to Q (Vosselman, 1994.). The idea behind a tree search algorithm is to identify the path along the tree structure that contains the best similarity between the attributes of corresponding features and their relations. Once the search is completed the location of the object can be inferred from the corresponding features (Vosselman, 1994.).

Cost or merit values are usually incorporated with search tree edges, to guide the algorithm along the optimal path. Examples of such merit functions would be gradient strengths or similarity values. These types of searches, which take advantage of a merit or cost function, are called informed searches. Tree searches that do not take advantage of a cost or merit function must use some sort of geometric reasoning function to search the branches of the tree. These types of searches are known as blind search methods.

Both types of searches perform several tests at each node. These tests ensure a variety of conditions are met, including a check to make sure the label has not been used before, that the label is consistent with prior labels and a check to make sure that the attributes of the unit and label are compatible. If these tests are successful, then the nodes of the next lower level will be generated. The order in which these lower level nodes are generated is unique to the tree search method that is used.

Blind search methods scan the search tree in a systematic order, looking for the optimal path. There are two ways in which this is done. The first method of searching the tree is called a depth-first search. This type of search, as the name implies, starts at the root node and scans down each path until a node fails a test. Once this occurs, the algorithm moves back up to a higher level in the tree and continues along other branches. The second method of scanning the search tree is called the breath-first search. This type of search scans all the nodes in a particular level and only proceeds to the next level after all nodes on the current level have been explored. These blind search methods usually take a considerable amount of time to find a solution. That is why, if possible, an informed search method is preferable.

Informed searched methods use information about the features to locate the best path through the search tree. This information is usually in the form of costs or merits. The optimal path in such a system would be the path, from root to leaf, which has the lowest sum of the costs, or the highest sum of merits. As stated previously, these costs or merits are associated with the edges of the search tree, not the nodes. The three most common methods of performing an informed search are the hill climbing approach, best first search and the A^* algorithm. These three routines differ in the amount of information that is needed to select the nodes to be expanded. Also, the A^* algorithm defines the admissibility as the condition that occurs when the sum of the non-explored edges never exceeds the actual costs (Vosselman, 1994.).

Several search strategies have been developed to reduce the search effort and speed up the tree search method. The most common of these strategies are unit ordering, forward checking, relaxation of the admissibility and beam search. Since each unit is labeled at each level of the tree and the ordering of these labels will not affect the final solution, these labels can be reordered so that the minimum number of nodes are present. This is how unit ordering can be used to reduce the number of nodes and, therefore, the time taken by the program to search the tree. Forward checking is used at every optimal path node to ensure that there is one label left for each node left in the tree. This prevents wrong labeling occurring, which will only be caught at the end of the optimal path and saves time because the search process will not have to start over again. Relaxation of the admissibility has to do with relaxing the criteria that ensures the costs of the sub-trees below a node (the future costs), never exceeds the costs of the optimal path through the tree, the so called 'actual future costs'. If the user can guarantee that the overestimates of the future costs will never exceed ε, then in the worst case, the A^* algorithm will only select a path that is ε higher than the optimal path. This strategy is alright if the tree is extremely large and there are many near optimal solutions that will be acceptable. The beam search method selects only the best candidates when a node is expanded and discards the rest. This strategy can be dangerous since it is possible that the optimal path might be pruned. Both the beam search method and the relaxation of the admissibility strategies can result in the failure of the routine to find a match. They are, however, sometimes necessary, since the search tree can grow too large for the available computer resources.

2.4.3 Automatic Target Recognition By Matching Oriented Edge Pixels

Olsen and Huttenlocher (1997) described a method for performing target recognition and matching through the use of an objects edge map. It is difficult to model small irregularly shaped targets. In order to do this, target objects and images need to be represented by their edge maps, where a local orientation is associated with each edge pixel. To model a 3-D object, a series of two-dimensional views are needed. To accomplish a full 3-D view of the object, translations, rotations and scaling of the 2-D views are required. A form of the Hausdorff measure is used to determine which positions of object models contain a possible target. This Hausdorff measure contains both location and orientation information. A hierarchical cell decomposition of the transformation space is used to efficiently determine the positions of these possible targets. A catalog of object models is used to match the possible target locations. This decreases computation time required by this method.

There are many benefits to implementing this type of target recognition model. Since edges are robust to changes in sensing conditions and edge based techniques can be used to locate many types of image pattern, the technique is general enough to be applied in a wide range of subjects (Olsen and Huttenlocher, 1997.). Using the pixelated edge of the target, instead of straight-line approximations, allow small irregularly shaped objects to be located. Also, many matching techniques have already been developed for edge maps. These techniques are able to handle such obstacles to matching as occlusion, image noise and clutter. Edge maps are also capable of working with many of the intelligent search strategies that have been developed to rule out much of the search space with little computer processing work. A large drawback to this matching model is that images with considerable clutter can cause numerous false alarms of the target location. To combat this, a modified version of the Hausdorff measure is used.

The unmodified form of the Hausdorff measure can be described as follows

$$h(M,I) = \max_{m \subset M} \min_{i \subset I} \| m - i \| \qquad \textbf{[2.32]}$$

Where M and I are point sets, and the measurement is being made from M to I. The effect of this equation is that the maximum distance of a point in set M is located from the nearest point in set I. This allows the Hausdorff measure to determine the quality of a match between an object model and an image (Olsen and Huttenlocher, 1997.). To generalize the Hausdorff measure to incorporate oriented edge pixels, it is necessary to consider each edge pixel in sets I and M to be a vector. This vector contains three terms, p_x and p_y give the location of the point and p_o represents the local orientation of the point. Since our concern is focused on the edge points, the x and y values of the vector fall into discrete sets (Olsen and Huttenlocher, 1997.). Similarly, the orientation parameters p_o can be mapped into discrete sets. The Hausdorff measure now must be expanded to include pixel orientation. This new measure should reduce to the old Hausdorff measure presented in Equation 2.32, when the pixel orientations are all the same. It also should use the old measure as a lower bound. A measure that fulfils these requirements is as follows.

$$h_\alpha(M, I) = \max_{m \subset M} \min_{i \subset I} \max \left\{ \left\| \begin{matrix} m_x - i_x \\ m_y - i_y \end{matrix} \right\|, \frac{|m_o - i_o|}{\alpha} \right\} \qquad \textbf{[2.33]}$$

The form of Equation 2.33 is the same as that of the Hausdorff measure, but it now computes the distance between M and I based on the maximum distances in translation and orientation of the edge pixels. In this measure, α denotes the normalized factor that makes the orientation values implicitly comparable with the location values (Olsen and Huttenlocher, 1997.). This means that a maximum deviation in translation and orientation for two pixels to match, can be set, allowing the number of pixel matches to be counted and kept. By adjusting the threshold of an acceptable match and the parameter α, the proximity required by the Hausdorff measure can be accomplished. To obtain a partial measure of oriented points that is robust to occlusion, the parameter α is set to one (α=1), and L_∞ norm is applied. This means the Hausdorff measure can be simplified to a form that is closely related to Equation 2.32.

$$h(M, I) = \max_{m \subset M} \min_{i \subset I} \| m - i \|_\infty \qquad [2.34]$$

A hierarchical cell decomposition of the transformation space is used to search the image for possible targets. To start, the image is made discrete to improve efficiency. This is done in such a way as to ensure that adjacent transformations do not map any object pixel more than one pixel apart in the image. It is done this way so that no good matches are missed. The image is then divided into rectilinear cells on the discrete grid of transformations. These rectilinear cells have three dimensions, the orientations, the shift in x and the shift in y. The discrete transformation that is closest to the cell centre is tested to determine if a match has been found. If the match at the centre is poor, the cell is pruned and further consideration in this cell ceases. Otherwise, the cell is divided into sub-cells and each sub cell is similarly considered. When a cell is found that contains only one transformation, it is tested and, no matter the outcome, no further subdivisions will be made in this cell.

The probability of a false alarm using this hierarchical cell decomposition method is given by the formula that follows.

$$1 - \prod_{i=0}^{W} (1 - P_K(i))^{d_i} \qquad [2.35]$$

Where P_K is the probability of a false alarm of size K in a window containing i edge pixels. The number of image windows containing i edge pixels is represented by the term d_i and W is the size of the window in pixels. Therefore, i must be greater or equal than zero but less than or equal to W ($0 \leq i \leq W$). Estimating the probability of a false alarm requires that the discrete nature of the transform space must be taken into account. Unlike transformation, rotation and scaling do not move every pixel a uniform distance. However, discrete rotations and scales can be considered to move the farthest pixels no more than one pixel in the image. This corresponds to the requirements mentioned for the correct operation of the hierarchical search strategy.

2.4.4 Relaxation Matching

Relaxation labeling is a technique that was developed to cope with uncertainty in sensory data interpretation so that the best possible match could be made in a stereo pair of images. Contextual information is used as an aid to classify sets of interdependent objects (Pajares et al., 2000.). Unique labels are assigned to matches taken from a list of features that are suspected of corresponding to the given template. In this way, each feature is assigned a value based on its disparity, so that it remains consistent with the predefined constraints that have been set by the user. There are two main types of relaxation matching. The first involves optimizing an energy function, which is formulated from the applicable constraints, so that the minimum value is found. The second technique uses initial probability estimates, which are established from a local stereo correspondence process and computed from the similarity in the feature values, to iteratively update these probabilities based on the matching probabilities of neighbouring features and the preset constraints.

One method for optimizing an energy function was proposed by Barnard in1987. He proposed to use sympathetic annealing to optimize the 1-D intensity profiles of the two images in the stereoscopic set. He defined an energy function as follows.

$$E_{ij} = \left|I_L(i,j) - I_R(i, j + D(i,j))\right| + \lambda \cdot \left|\nabla D(i,j)\right| \qquad \textbf{[2.36]}$$

Where I_L denotes the intensity function of the left image at the ith row and jth column. I_R denotes the intensity function of the right image at the ith row and (j+D(i,j))th column. D(i,j) is the disparity value, which is simply a horizontal shift in this case. The disparity value in this equation is the constraint that must be optimized, so that the minimum change in the disparity value is found (D(i,j), $\forall$i,j). This type of constraint is commonly known as the continuity constraint. The advantage to using this type of intensity-based technique in relaxation matching is that a dense disparity map will be output by the procedure. This means that a dense depth or range map can also be created. The problem with using this technique is, like all optimization problems, whether the routine will converge to the global minimum, or will local minima confuse it. The use of a multiresolution approach has been reported to improve the speed at which this method operates and helps avoid local minima (Robert et al.,1992.).

The probabilistic approach to relation matching was developed in a Bayesian framework. Assuming that a set of objects, labelled A, are to be matched, where N is the number of objects in the set and $A = \{a_1, a_2, a_3, \ldots, a_N\}$. The object of this procedure is to match this scene to a mathematical model. Each object in A will be assigned a corresponding label. These labels will be represented by θ. Therefore, the object a_i will take on the label θ_i. Each θ can take on any of the M model labels that comprise the set $\mathbf{\Omega} : \mathbf{\Omega} = \{\omega_1, \omega_2, \omega_3, \ldots, \omega_M\}$. The notation $\omega_{\theta i}$ means that this model label is associated with a particular object label, θ_i (Pajares et al., 2000.). Once labelling has been completed, every object label will have one unambiguous value. For the sake of simplicity, two indices must now be defined. The value N_0 is an indice that can take on values as follows, $N_0 = \{1, 2, \ldots, N\}$. Similarly, the value N_i is an indice that can take on values as follows, $N_i = \{1, 2, \ldots, i-1, i+1, \ldots, N\}$.

The implementation of the probabilistic approach consists of a number of steps. The first step involves the organization of image features into pairs of edge segments. Each pair must contain a difference in the direction of the gradient of less than ±45° and can be characterized by its initial matching probabilities. These initial matching probabilities can be calculated from the following formula.

$$p(x) = \frac{1}{|\Sigma|^{0.5}} \cdot \exp\left[-\frac{1}{2}(x-\mu)^t \, \Sigma^{-1} (x-\mu) \right] \qquad \textbf{[2.37]}$$

Where, μ is the corresponding learned cluster centre, Σ is the covariance matrix, x is the dimensional difference measurement vector and t indicates that the transpose needs to be taken. Next, let the variable npair represent the number of match pairs that have been modified. To start with, npair is given the value of zero. The probabilistic relaxation technique for object labeling will then be performed. This technique operates by giving the label θ_i of an object a_i the value $\omega_{\theta i}$, if, and only if, it is the most probable label that contains all the unary measurements and binary relationships between the image objects (Pajares et al., 2000.). This probability is computed from the following formula.

$$P(\theta_i = \omega_{\theta i} \mid x_{j, j \in N_0}, A_{ij, j \in N_i}) = \max_{\omega \in \Omega} P(\theta_i = \omega_\lambda \mid x_{j, j \in N_0}, A_{ij, j \in N_i}) \qquad \textbf{[2.38]}$$

Where $P(\theta_i = \omega_{\theta i})$ and $P(\theta_i = \omega_\lambda)$ represent the probabilities that object label θ_i matches the model labels $\omega_{\theta i}$ and ω_λ, respectively. At the nth iteration the probability of a match can be computed from Equation 2.39.

$$P^{(n+1)}(\theta_i = \omega_{\theta i}) = \frac{P^{(n)}(\theta_i = \omega_{\theta i}) \cdot Q^{(n)}(\theta_i = \omega_{\theta i})}{\sum_{\omega\lambda \in \Omega} P^{(n)}(\theta_i = \omega_{\theta i}) \cdot Q^{(n)}(\theta_i = \omega_{\theta i})} \qquad [2.39]$$

$$Q^{(n)}(\theta_i = \omega_{\theta i}) = \prod_{j \in N_i} \sum_{\omega\beta \in \Omega} P^{(n)}(\theta_j = \omega_\beta) p(A_{ij} \mid \theta_i = \omega_\alpha, \theta_j = \omega_\beta) \qquad [2.40]$$

Where, $P^{(n)}(\theta_i = \omega_{\theta i})$ is the probability of a match at iteration n, between label θ_i and model label $\omega_{\theta i}$. The quantity $Q^{(n)}(\theta_i = \omega_\alpha)$ indicates the amount of participation from surrounding objects at iteration n, for the probability of a match between label θ_i and model label $\omega_{\theta i}$. The density function $p(A_{ij} \mid \theta_i = \omega_\alpha, \theta_j = \omega_\beta)$ is a compatibility coefficient that will take on a value between 0 and 1. It quantifies the compatibility between this match $(\theta_i = \omega_\alpha)$ and a neighbouring match $(\theta_j = \omega_\beta)$. A test is now made to determine if the absolute value of the difference between $P^{(n+1)}(\theta_i = \omega_{\theta i})$ and $P^{(n)}(\theta_i = \omega_{\theta i})$ is greater than some user defined error value (ε). If the error is too great then npair is

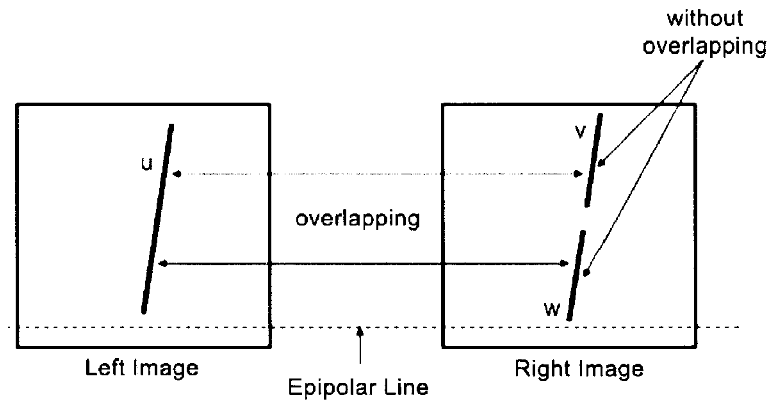


Fig 2.8: Broken Edge Segments that Match Original Line (Pajares et al., 2000.)

increased and the iteration process starts over again.

This procedure has many possible outcomes. A left edge segment can, in the unambiguous case, be assigned to a unique right edge segment, or in the ambiguous case, it can be matched to several right edge segments (Figure 2.8). The unambiguous case provides only one probability value, so identifying the line is easy, however the ambiguous case will provide many probabilities and choosing the correct segment is simply a matter of choosing the highest probability. A threshold value will also have been set by the user, which limits probabilities that are examined when determining the correct line.

3.0 INDMET

INDMET was designed for performing high precision target location in industrial metrology. The black circular targets on retroreflective backings were chosen because they have been shown to be effective for close range applications (Cosandier and Chapman, 1992). When these targets are viewed at an angle, they appear as ellipses and are recorded as such by the camera. The edge observations are determined by using a single dimensional sub-pixel edge locator based on moments. A non-linear least squares adjustment is then used to determine the centre coordinates based on these edge observations. INDMET is capable of locating the centre of a target to accuracies of 1/20th to 1/50th of a pixel.

3.1 Targets

To achieve subpixel accuracy, it is absolutely imperative that any target used appears in several pixels. This is the reason that specialty targets were designed, to be used with INDMET, over other conventional surveying targets. The target consists of a

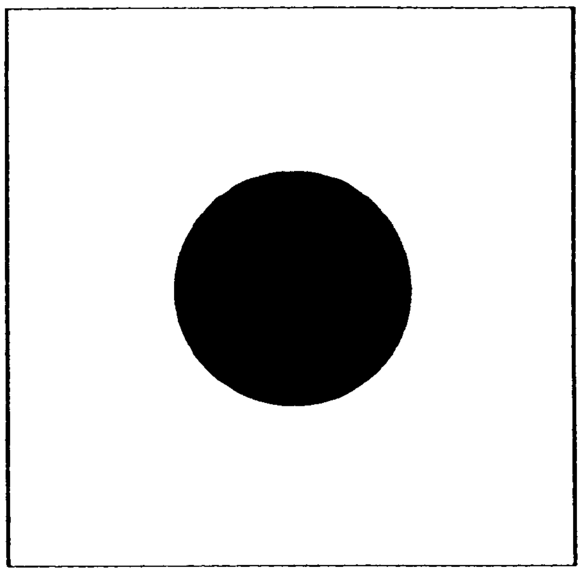
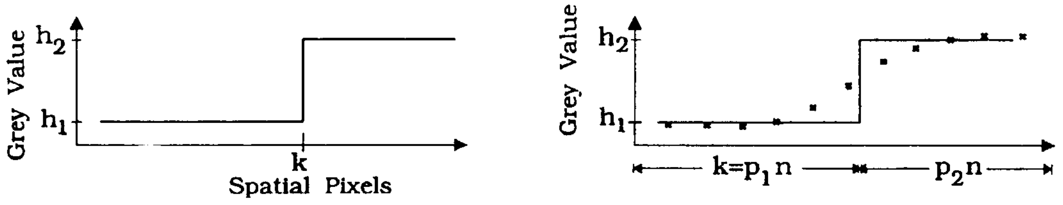


Fig 3.1: Circular Target (actual size)
(Cosandier and Chapman, 1992)

black circle placed on a white background. An example target is displayed in Figure 3.1. Circle roundness and non-uniformities in background intensity are important problems that have to be addressed if subpixel accuracies are to be achieved. To address these problems, a retro-reflective film was used for the target background. It was decided that 3M Scotchlite #7610 would suffice for the retro-reflective film (Cosandier and Chapman, 1992). This material has a high degree of reflectance and reflects light directly back (within 1°) toward the source. This means that the light source has to be rather close to the camera and that the retro-reflectivity sharply drops if the light source is more than 50° from the perpendicular to the surface (Cosandier and Chapman, 1992).

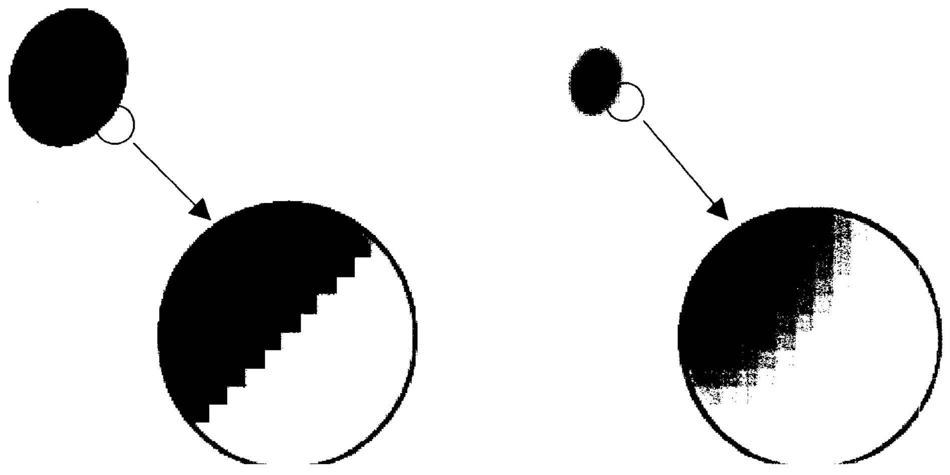
To achieve subpixel accuracy within the image, it is imperative that the targets be well defined. In an ideal world the contrast difference that distinguishes the ellipses edge, would be as simple as black and white. Unfortunately, there will always be a



a) Ideal Edge b) Observed Edge

Fig 3.2: Representations Of Target Edges (Cosandier and Chapman, 1992)

region between the black ellipse and the white background. Figure 3.2a shows the ideal edge, where at a pixel value k, the grey values in the image will jump from black to white. Figure 3.2b represents reality, where the plotted points represent the rise that occurs in the CCD pixel array and the solid line indicates an average location of the image edge. There are a number of reasons why the ideal edge and the observed edge do not match. These reasons include the edge failing to fall on the border between pixels,



a) Well Defined Edge b) Poorly Defined Edge

Fig 3.3: Digital Image Target Edges

the edge not being perpendicular to the operator axis and the inherent noise in CCD arrays. The idea is to try and minimize the number of pixels over which this rise occurs. Figure 3.3a shows a target that is well defined, where the contrast edge is only two pixels deep. Figure 3.3b shows a poorly defined target. This edge is a very gradual transition from white to black, which makes identifying the actual edge very difficult. Without a precise location for the edge of the target, the calculated target centre will lose precision, probably to the point where subpixel accuracy is lost.

Another problem that can be encountered is when the target becomes too skewed in the image. As stated before, the retro–reflectivity of the target drops off sharply if the light source is greater than 50° from the perpendicular to the target surface. An example of this type of poorly defined target is shown in Figure 3.4. The result of this skewness is that light was not reflected back at the camera and therefore the contrast difference between the ellipse black and white target background is not great. This will also hurt the ability to achieve subpixel accuracy on the calculation of the target centre.

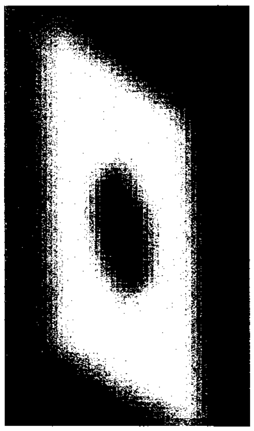


Fig 3.4: Skewed Circular Target

3.2 Mathematical Basis for INDMET

The target described above can be best approximated by a rotated ellipse on the image plane. To be valid, the distance between the camera and the target must be sufficiently great to completely display the target. The equation for the rotated ellipse is as follows.

$$\left[\frac{\Delta x \cdot \cos\theta + \Delta y \cdot \sin\theta}{a}\right]^2 + \left[\frac{\Delta y \cdot \cos\theta + \Delta x \cdot \sin\theta}{b}\right]^2 = 1 \qquad [3.1]$$

$$Where \quad \Delta x = x - xc, \quad \Delta y = y - yc$$

In Equation 3.1, x and y are the edge coordinates in pixels, xc and yc are the centre of the ellipse in pixels, a and b are the horizontal and vertical ellipse axes in pixels and θ is the rotation of the ellipse in radians.

Equation 3.1 is implemented in an iterative non-linear least-squares adjustment. Due to the fact that the orientation of a circle is ambiguous, convergence on θ fails when a is approximately equal to b (Cosandier and Chapman, 1992). To solve this problem, the weighted constraint $1/\sigma^2$ was added. Therefore, when θ equals zero, σ becomes $\pi/2$. Equation 3.1 must also be weighted. To do this it is assumed that a and b will be nearly equivalent, θ is close to zero and the measuring accuracies on the x and y axes are the same (Cosandier and Chapman, 1992). Using these assumptions Equation 3.1 can be simplified to become Equation 3.2.

$$f(r) \approx \left[\frac{r}{R}\right]^2 \qquad \textbf{[3.2]}$$

Where r is the measured radius and R is the true radius. Taking the partial derivative and then assuming that r ≈ R, Equation 3.3 was created.

$$\delta f(r) \approx \frac{2}{R}\sigma_r \qquad \textbf{[3.3]}$$

This equation is used to scale the adjustment's residuals into values that will be useful in post-processing. INDMET uses the RMS of the residuals as the indicator of the measurement quality of each target (Cosandier and Chapman, 1992).

Due to the fact that a non-linear least squares adjustment is used to calculate xc, yc, a, b and θ, initial approximates need to be made. The edge locator requires that fairly accurate (i.e. 20% of the ellipse size) initial approximates have to be used (Cosandier and Chapman, 1992). The centre of the ellipse, represented by the terms xc and yc, can be fairly accurately approximated by calculating the centroid of the circular portion of the target. The calculation of the centroid is greatly disturbed by dark patches outside the target. To solve this problem, the user can either select a region within the target or use the auto box routine to have the computer construct a box within the target. The ellipse axes are approximated by searching outwards from the selected point, to the inverse target background (Cosandier and Chapman, 1992). The ellipse rotation is estimated to be zero.

It is absolutely imperative that precise edge coordinate observations are made if subpixel accuracy is to be achieved. The one-dimensional edge operator that is used by INDMET was first presented by Tabatabai and Mitchell and is based on the method of moments. In this method, the edge location, k, is the number of pixels from the vertical axis with intensity h_1. To calculate the edge observations, the following equations are used.

$$\overline{m}_i = \frac{1}{n}\sum_{j=1}^{n} x_j^i, \quad i = 1 \quad to \quad 3 \qquad [3.4]$$

$$\sum_{j=1}^{2} p_j h_j^i = \overline{m}_i \qquad [3.5]$$

$$p_1 + p_2 = 1 \qquad [3.6]$$

Where $\overline{m}_i$ is the ith order sample moment, x is the array of grey values, p is the unit length of each edge and h is the brightness values of each edge. Using the equations above, expressions for p_1, h_1 and h_2 can be formed.

$$h_1 = \overline{m}_1 - \overline{\sigma}\sqrt{\frac{p_2}{p_1}} \qquad [3.7]$$

$$h_2 = \overline{m}_1 - \overline{\sigma}\sqrt{\frac{p_1}{p_2}} \qquad [3.8]$$

$$p_1 = \frac{1}{2}\left[1 + \overline{s}\sqrt{\frac{1}{4 + \overline{s}^2}}\right] \qquad [3.9]$$

Where,

$$\overline{s} = \frac{\overline{m}_3 + 2\overline{m}_1^3 + 3\overline{m}_1\overline{m}_2}{\overline{\sigma}^3} \qquad [3.10]$$

$$\overline{\sigma}^2 = \overline{m}_2 - \overline{m}_1^2 \qquad [3.11]$$

The edge can be now calculated by ($k = n \cdot p_1$). These equations will only calculate to sub-pixel accuracy if p_1 is a non-integer value. To account for this, the first pixel is positioned at ½ and pixel locations are incremented by 1 (Cosandier and Chapman, 1992).

4.0 A Target Matching Solution for INDMET

A target matching solution has been added to INDMET. This solution incorporates an epipolar line module, a cross correlation module and a template least squares module. The solution assumes that a stereo pair of images is present. The user will choose a target on the left template image and the corresponding target will be found on the right image. The coordinates of the user chosen point and the image orientations will be used by the epipolar line module to calculate the epipolar line, along which the corresponding target lies. This limits the search area to a 1-D case, greatly decreasing search time. The cross correlation module will then be used to search the right image, comparing each search window with the template image taken from the left image. Finally, the template least squares module will be used to locate the target centre to subpixel accuracy.

This Section describes the mathematical theory behind these three modules and is designed to walk the reader through the calculation process. Information specific to the programming of the routines can be found in Chapter 5, 'Algorithm Implementation In INDMET'.

4.1 A Method For The Calculation Of Epipolar Lines

Epipolar lines are created through the intersection of the epipolar plane and the two image planes (see Figure 4.1). The epipolar plane is created by the basis vector; the vector joining the two image centres, and the ray vectors intersecting the image planes (see Figure 4.2). This implies that corresponding image points must lie on corresponding epipolar lines. The advantage to doing this is that a match for the point on the left image

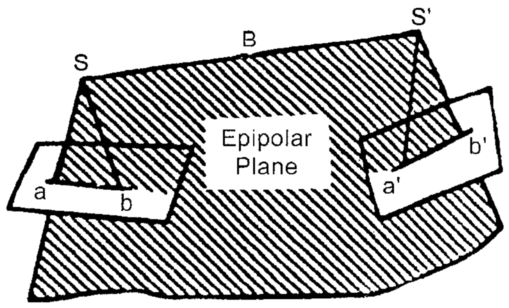


Fig 4.1: Intersection Of Epipolar Plane
With Images (Lü and Zhang, 1988.)

should be found to lie somewhere along the corresponding epipolar line on the right image. This has come to be known as the epipolar constraint.

Figure 4.2 shows the epipolar plane SS'A. Where S is the image centre for the left image, S' is the image centre for the right image and A is the object point. Figure 4.1 also shows the left image, labelled p. Image t is the normalized form of p, which has been rotated so that it is parallel to B. Using this projective geometry, it is possible to form the following relationship (Lü and Zhang, 1988.).

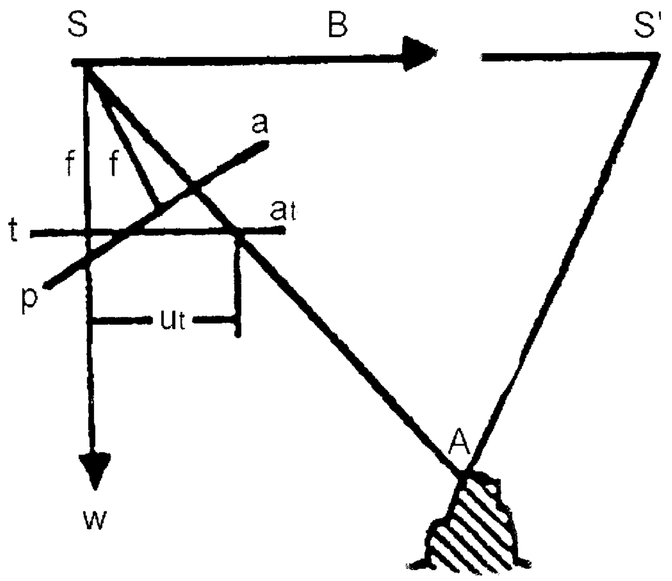


Fig 4.2: Epipolar Geometry (Lü and Zhang, 1988.)

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ -f \end{bmatrix} \quad [4.1]$$

Where u, v, w are the object coordinates of point A in image p; a, b, c are the direction cosines that orient image p in relation to the basis vector B; x, y are the coordinates of point A in image p, and f is the focal length of the camera when it captured image p.

It is then possible to derive the coordinates of point A on the normal image t, using Equation 4.1 and Figure 4.2.

$$u_t = u\left(\frac{-f}{w}\right) = -f\,\frac{a_1x + a_2y - a_3f}{c_1x + c_2y - c_3f} \quad [4.2a]$$

$$v_t = v\left(\frac{-f}{w}\right) = -f\,\frac{b_1x + b_2y - b_3f}{c_1x + c_2y - c_3f} \quad [4.2b]$$

Alternatively, the coordinates of point A in the original image p can be calculated by the following formula.

$$x = -f\,\frac{a_1u_t + b_1v_t - c_1f}{a_3u_t + b_3v_t - c_3f} \quad [4.3a]$$

$$y = -f\,\frac{a_2u_t + b_2v_t - c_2f}{a_3u_t + b_3v_t - c_3f} \quad [4.3b]$$

On the original image p, it is expected that all epipolar lines will be convergent, while on image t, the normal image, the epipolar lines are expected to be mutually parallel (see Figure 4.3). Given that the exterior orientations of the camera at S and S' are known, all

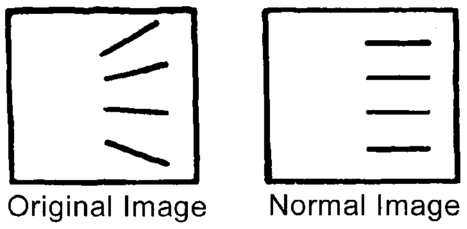


Fig 4.3: Epipolar Lines (Lü and Zhang, 1988.)

the points on image p can be projected onto the normal image by Equations 4.2. This is conventional digital rectification. It is important to realize that these projection equations rearrange the epipolar lines on the normal image. This means that the transfer of the epipolar line to the normal image and then to the right image must be done point by point.

Using the coplainarity condition for three lines, the following relationship can be made.

$$\overline{SS'} \cdot (\overline{Sa} \times \overline{Sb}) = 0 \qquad \textbf{[4.4a]}$$

Alternatively, the coplainarity condition can be written as:

$$\begin{vmatrix} B & 0 & 0 \\ U_a & V_a & W_a \\ U_b & V_b & W_b \end{vmatrix} = B \begin{vmatrix} V_a & W_a \\ V_b & W_b \end{vmatrix} = 0 \qquad \textbf{[4.4b]}$$

By combining Equation's 4.1 and 4.4b, the following determinant can be created.

$$\begin{vmatrix} b_1 x_a + b_2 y_a - b_3 f & c_1 x_a + c_2 y_a - c_3 f \\ b_1 x_b + b_2 y_b - b_3 f & c_1 x_b + c_2 y_b - c_3 f \end{vmatrix} = 0 \qquad \textbf{[4.5]}$$

Given the user defined point (x_a, y_a) and using an appropriate value for x_b, it becomes possible to find the epipolar line on the left image, by solving Equation 4.5 for y_b. Once these two points are known, it becomes a simple matter to transfer the line to the right image. By transferring the two points on the epipolar line in the left image to the normal image t (Equations 4.2) and given the fact that epipolar lines in the normal image are mutually parallel, it becomes possible to use those points on the right image. After that, all that needs to be done is to reorient the two points from the normal image into the plane of the right image using Equations 4.3.

4.2 Cross Correlation Analysis

Cross Correlation analysis is one of the standard techniques used in target matching. To perform this analysis, it is necessary to create a template image sample, and a test image sample. The correlation coefficient, denoted by ρ, reflects the degree of correlation between these two samples. The value of ρ will lie between –1 and +1, where a –1 indicates that the test sample is the exact opposite of the template sample and a +1 indicates that the test sample is a perfect match for the template sample. Another way of stating this is that if |ρ| = 1, then there exists perfect linear correlation and if you plotted the template image grey values against the test image grey values, the result would be a straight line. The closer that ρ is to zero, the less similarity or the less correlation the two images have.

An estimate of the correlation coefficient can be computed in the same way as the sample variance and the sample covariance would be. The following formula is applied.

$$\hat{\rho} = \frac{s_{LR}}{s_L \cdot s_R} \quad \textbf{[4.6]}$$

Where $\hat{\rho}$ is the estimate of the correlation coefficient, s_{LR} is the sample covariance between the left template image sample, and the right test image sample, s_L is the estimated standard deviation of the left template image sample and s_R is the estimated standard deviation of the right test image sample. To obtain the standard deviations and the covariance information, the following formulas are used.

$$s_{LR} = \sum \frac{(g_L - \bar{g}_L) \cdot (g_R - \bar{g}_R)}{n-1} \quad \textbf{[4.7]}$$

$$s_L^{\,2} = \sum \frac{(g_L - \bar{g}_L)^2}{n-1} \quad \textbf{[4.8]}$$

$$S_R^{\;2} = \sum \frac{(g_R - \bar{g}_R)^2}{n-1} \qquad [4.9]$$

Where g_L is the grey value of the pixels in the template left image sample, g_R is the grey value of the pixels in the test right image sample and n is the number of grey values in the sample.

4.3 A Method For Performing Template Least Squares Target Matching

There are many statistical estimation models that can be used to perform a template matching solution. The model presented here is a simple least square adjustment that performs grey level matching. While it is possible to apply this type of estimation model to a multi-photo arrangement, this model is described for a two-photo system only.

Given two grey level patches that are of equal size and contain an identical number of pixels, the following grey level matching observation equation can be formulated.

$$-e(x, y) = R(x, y) - L(x, y) \qquad [4.10]$$

Where, e is the true error vector, L is the grey level function of the left image patch and R is the grey level function of the right image patch. Assuming that the left template image will be fixed and that an affine transformation can be used for the right image, the following equations can also be formulated.

$$x_L = x_{L0} \qquad [4.11a]$$

$$y_L = y_{L0} \qquad [4.11b]$$

$$x_R = T_{XR} + S_X \cdot x_{R0} + U_X \cdot y_{R0} \qquad [4.11c]$$

$$y_R = T_{YR} + U_Y \cdot x_{R0} + S_Y \cdot y_{R0} \qquad [4.11d]$$

Where x_{L0}, y_{L0} are the initial pixel coordinates of the left image patch, x_{R0}, y_{R0} are the initial pixel coordinates of the right image patch, T_{XR}, T_{YR} are the calculated shift parameters of the right image patch, S_X, S_Y are the calculated scale parameters for the right image patch and U_X, U_Y are the calculated sheering parameters for the right image patch. It is important to note that the parameters x_{L0}, y_{L0}, x_{R0}, y_{R0} refer to the centralized pixel coordinates of the template left image patch and the test right image patch (Gruen and Baltsavias, 1986). By linearizing Equation 4.10, the following equation can be created.

$$-e(x,y) = R^0(x,y) - L^0(x,y) + R_X \cdot \Delta T_{XR} + R_Y \cdot \Delta T_{YR} + R_X \cdot x_R \cdot \Delta S_X + R_Y \cdot y_R \cdot \Delta S_Y + R_X \cdot y_R \cdot \Delta U_X + R_Y \cdot x_R \cdot \Delta U_Y \qquad \textbf{[4.12]}$$

Where

$$R_X = \frac{\partial R^0(x,y)}{\partial x_R} \qquad \textbf{[4.13a]}$$

$$R_Y = \frac{\partial R^0(x,y)}{\partial y_R} \qquad \textbf{[4.13b]}$$

Rewriting this equation in matrix form, results in the following

$$-e(x,y) = Ax - l; P \qquad \textbf{[4.14]}$$

Where

$$A = \begin{bmatrix} R_X & R_Y & R_X \cdot x_R & R_Y \cdot y_R & R_X \cdot y_R & R_Y \cdot x_R \\ & & & \vdots & & \end{bmatrix}$$

$$x = \begin{bmatrix} \Delta T_{XR} \\ \Delta T_{YR} \\ \Delta S_X \\ \Delta S_Y \\ \Delta U_X \\ \Delta U_Y \end{bmatrix}$$

$$l = \begin{bmatrix} L^0(x,y) - R^0(x,y) \\ \vdots \end{bmatrix}$$

P is a weight matrix that can be used to force the adjustment into assigning more value to some of the observations. In this case, the weight matrix will be left as an identity matrix ensuring all observations are treated equally. The least squares solution for Equation 4.14 can be calculated as follows.

$$\hat{x} = (A^T PA)^{-1} \cdot (A^T Pl) \qquad \textbf{[4.15]}$$

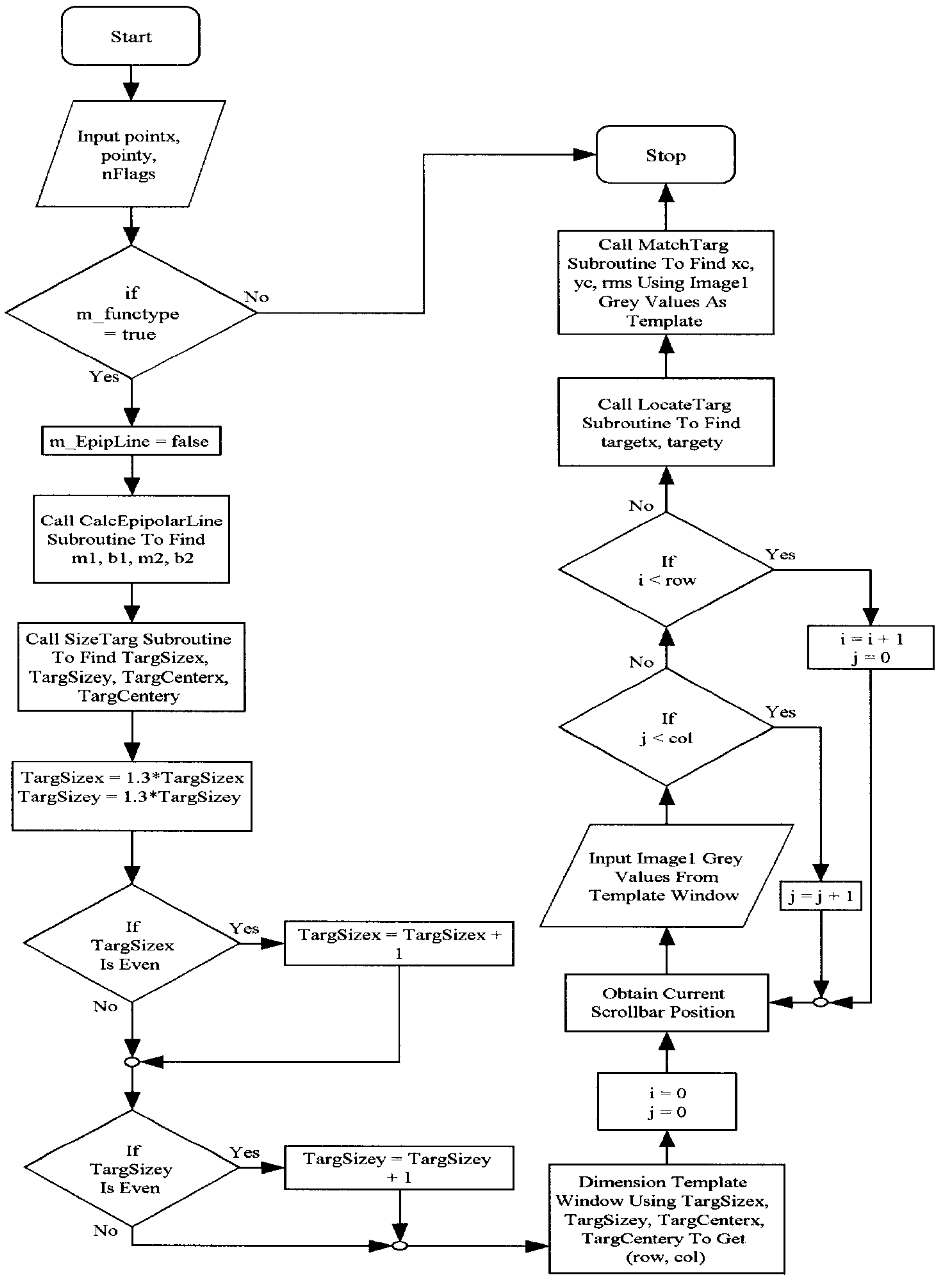
It is important in most engineering application to know how well the estimates of Equation 4.15 are determined. This normally could be accomplished through the use of the covariance matrix which can be incorporated into the least squares formulation. However, due to the fact that the covariance matrix would contain error estimates calculated in the spatial domain and not the spectral domain, a different method is needed for error estimation. This alternative method of error estimation involves the use of vector l which is defined above. By squaring and then summing the terms of vector l, it is possible to obtain an error estimate. Through the use of Equation 4.16, this error estimate can be calculated.

$$Error \approx \frac{\sqrt{\sum_{i=0}^{n} (l_i)^2}}{n} = \frac{\sqrt{\sum_{i=0}^{n} (L^0(x,y) - R^0(x,y))^2}}{n} \qquad \textbf{[4.16]}$$

Where n is the number of pixels used to determine the estimates of Equation 4.15 and Error is the root mean square error estimate which determines the goodness-of-fit of the data.

5.0 Algorithm Implementation in INDMET

Three new subroutines have been added to Indmet. These subroutines give Indmet the added ability to perform a complete target matching solution, which can determine the centre of a target to subpixel accuracies. The flow chart in Figure 5.1



Start
Input pointx, pointy, nFlags
if m_functype = true
No
Yes
m_EpipLine = false
Call CalcEpipolarLine Subroutine To Find m1, b1, m2, b2
Call SizeTarg Subroutine To Find TargSizex, TargSizey, TargCenterx, TargCentery
TargSizex = 1.3*TargSizex
TargSizey = 1.3*TargSizey
If TargSizex Is Even
Yes
No
TargSizex = TargSizex + 1
If TargSizey Is Even
Yes
No
TargSizey = TargSizey + 1
Dimension Template Window Using TargSizex, TargSizey, TargCenterx, TargCentery To Get (row, col)
i = 0
j = 0
Obtain Current Scrollbar Position
Input Image1 Grey Values From Template Window
If j < col
Yes
No
j = j + 1
If i < row
Yes
No
i = i + 1
j = 0
Call LocateTarg Subroutine To Find targetx, targety
Call MatchTarg Subroutine To Find xc, yc, rms Using Image1 Grey Values As Template
Stop

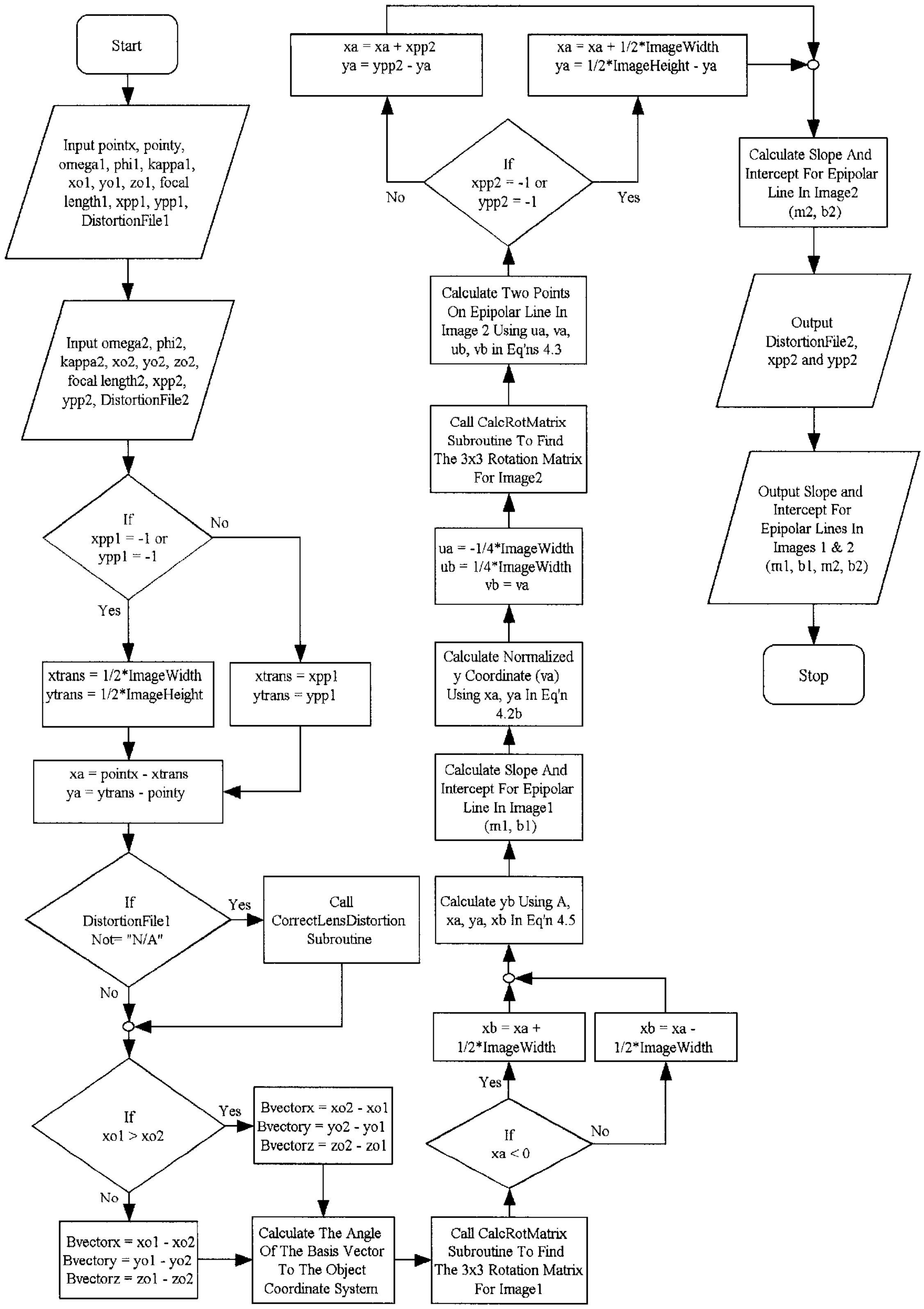
Fig 5.1: Portion of INDMET control file OnLButtonDown that operate the automatic target matching functions

diagrams the overall structure of the automatic target matching program that has been created and added to INDMET. It incorporates and utilizes the three subroutines that have been programmed to calculate epipolar lines, perform cross correlation analysis and perform template least squares matching. The programming considerations, including the logical order flow chart diagrams, for these three subroutines are presented in the subsections to this chapter.

5.1 Epipolar Lines

An epipolar line subroutine was added to INDMET to allow the computation of an epipolar line, based on the target chosen and the image's exterior orientation data. This orientation data is entered through the use of an exterior orientation dialog box, which appears when the images are first opened. The module is based on the mathematical method presented in Section 4.1 and the code for the module and the images used in the testing of this module can be found in the appendix. Figure 5.2 present the flow diagram for the logical operation of the epipolar line subroutine.

Using images of a target field, it was found that the epipolar line routine was not passing through the corresponding targets on the images. It was determined that the cause of this problem was principal point and lens distortion errors associated with the camera. The principal point problem was simply corrected by having the user input the principal point in pixels from the upper left corner of the image and shifting the image coordinate system so that the origin is at the principal point. To correct lens distortion, a new subroutine was added that reads in lens distortion data and then computes and applies the appropriate correction. Since the lens distortion data that was received with the images was in the form of regularly spaced radial distance readings of the distortion,



Start
Input pointx, pointy, omega1, phi1, kappa1, xo1, yo1, zo1, focal length1, xpp1, ypp1, DistortionFile1
Input omega2, phi2, kappa2, xo2, yo2, zo2, focal length2, xpp2, ypp2, DistortionFile2
If xpp1 = -1 or ypp1 = -1
No
Yes
xtrans = 1/2*ImageWidth ytrans = 1/2*ImageHeight
xtrans = xpp1 ytrans = ypp1
xa = pointx - xtrans ya = ytrans - pointy
If DistortionFile1 Not= "N/A"
Yes
Call CorrectLensDistortion Subroutine
No
If xo1 > xo2
Yes
Bvectorx = xo2 - xo1 Bvectory = yo2 - yo1 Bvectorz = zo2 - zo1
No
Bvectorx = xo1 - xo2 Bvectory = yo1 - yo2 Bvectorz = zo1 - zo2
Calculate The Angle Of The Basis Vector To The Object Coordinate System
Call CalcRotMatrix Subroutine To Find The 3x3 Rotation Matrix For Image1
If xa < 0
Yes
No
xb = xa + 1/2*ImageWidth
xb = xa - 1/2*ImageWidth
Calculate yb Using A, xa, ya, xb In Eq'n 4.5
Calculate Slope And Intercept For Epipolar Line In Image1 (m1, b1)
Calculate Normalized y Coordinate (va) Using xa, ya In Eq'n 4.2b
ua = -1/4*ImageWidth ub = 1/4*ImageWidth vb = va
Call CalcRotMatrix Subroutine To Find The 3x3 Rotation Matrix For Image2
Calculate Two Points On Epipolar Line In Image 2 Using ua, va, ub, vb in Eq'ns 4.3
If xpp2 = -1 or ypp2 = -1
No
Yes
xa = xa + xpp2 ya = ypp2 - ya
xa = xa + 1/2*ImageWidth ya = 1/2*ImageHeight - ya
Calculate Slope And Intercept For Epipolar Line In Image2 (m2, b2)
Output DistortionFile2, xpp2 and ypp2
Output Slope and Intercept For Epipolar Lines In Images 1 & 2 (m1, b1, m2, b2)
Stop

Fig 5.2: Diagram of the operation of the Epipolar Line function

an interpolator to fill in the missing data was necessary. It was decided that a simple linear interpolator, applied between distortion points, was the best option. The linear interpolator has the advantage of working in a general context, where all forms of distortion can be represented. It has the disadvantage of inaccuracy, since linear interpolation between points, can displace a great distance from the actual distortion value. This becomes even more pronounced when the data points are farther apart.

5.2 Cross Correlation Analysis

A cross correlation subroutine was added to Indmet so that images can be scanned to find targets along a given epipolar line. Since the routine needs to orient the template and test image samples along the appropriate epipolar line, two sets of line parameters, m and b, are required. These lines represent the epipolar lines in each of the two images. The mathematical basis for this analysis is discussed in Section 4.2 and the code for the subroutine, as well as the images used in the testing of this subroutine, can be found in the appendix. Figures 5.3a, 5.3b and 5.3c present the flow diagram for the logical operation of the cross correlation subroutine.

Due to the fact that the pixel areas are aligned with the axes and not the epipolar line, rotation and interpolation become necessary to obtain a proper sample along the aforementioned epipolar line. To reorient the pixels under consideration, coordinates around the centre of the image sample are rotated so that they are aligned with the epipolar line. An interpolator is then used to determine a grey value for each of these coordinates. The interpolator is the same bi-linear interpolator that is described in the template least squares matching section. It was used in this application because it was

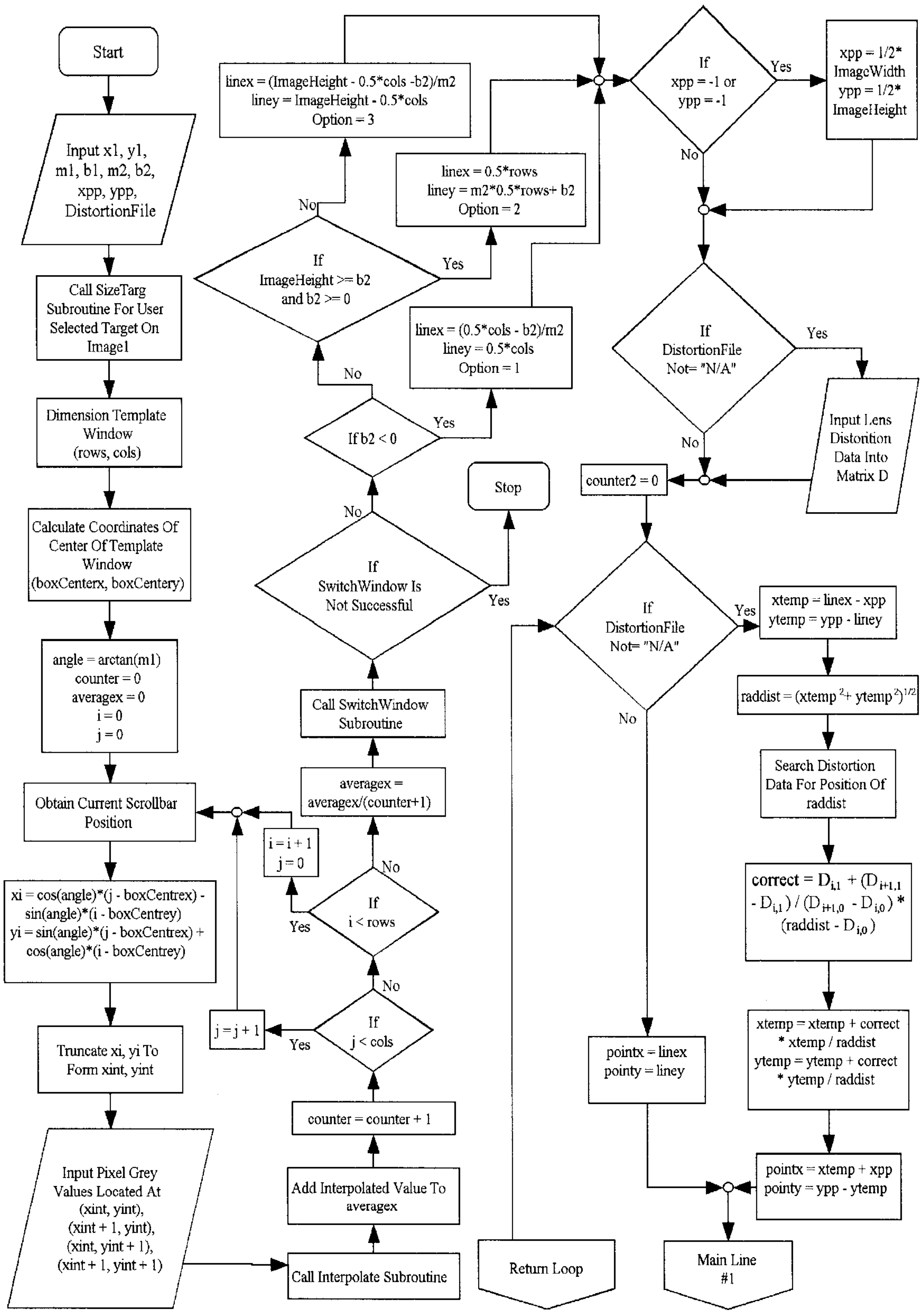
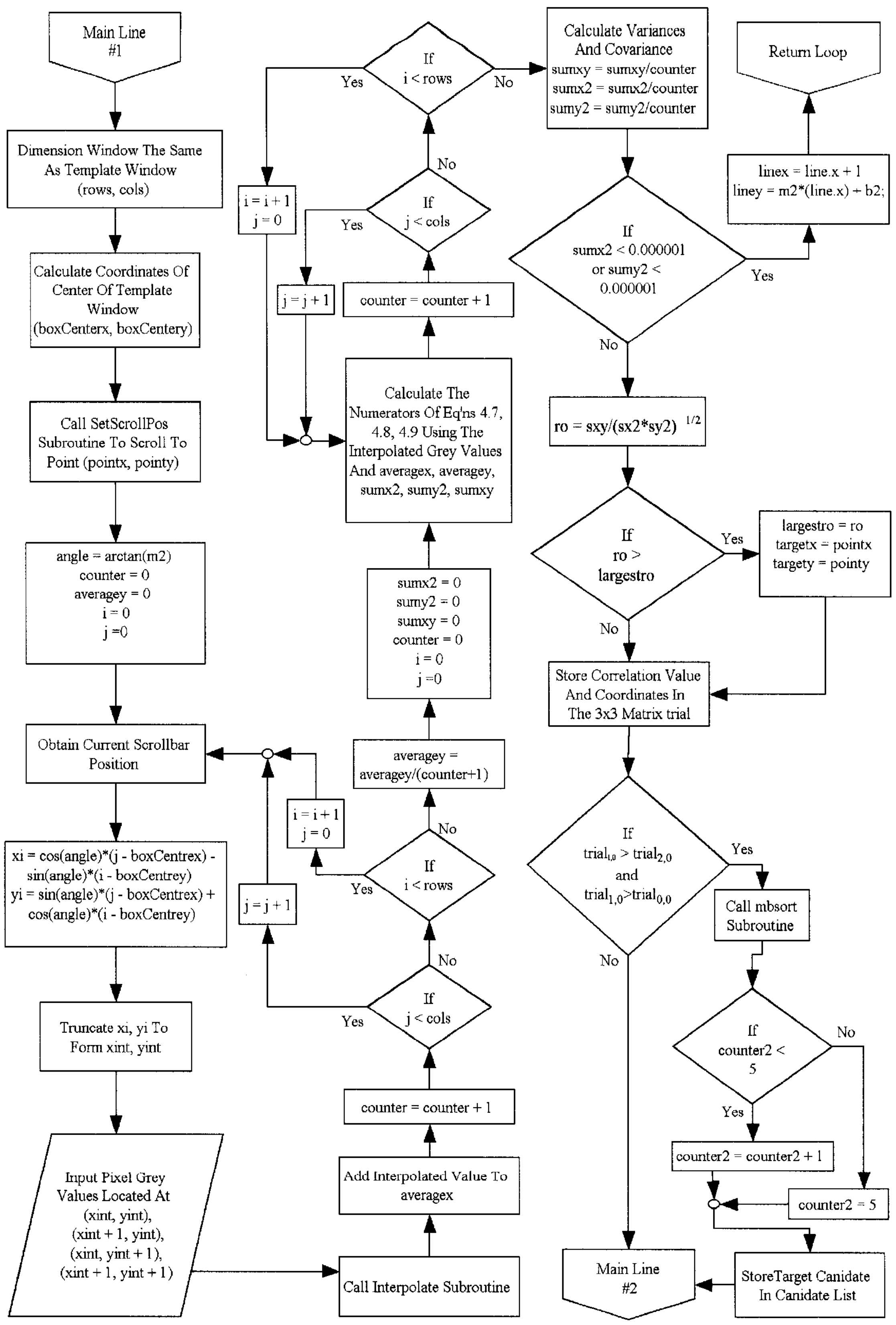

Start
Input x1, y1, m1, b1, m2, b2, xpp, ypp, DistortionFile
Call SizeTarg Subroutine For User Selected Target On Image1
Dimension Template Window (rows, cols)
Calculate Coordinates Of Center Of Template Window (boxCenterx, boxCentery)
angle = arctan(m1) counter = 0 averagex = 0 i = 0 j = 0
Obtain Current Scrollbar Position
xi = cos(angle)*(j - boxCentrex) - sin(angle)*(i - boxCentrey) yi = sin(angle)*(j - boxCentrex) + cos(angle)*(i - boxCentrey)
Truncate xi, yi To Form xint, yint
Input Pixel Grey Values Located At (xint, yint), (xint + 1, yint), (xint, yint + 1), (xint + 1, yint + 1)
Call Interpolate Subroutine
Add Interpolated Value To averagex
counter = counter + 1
If j < cols
Yes
j = j + 1
No
If i < rows
Yes
i = i + 1 j = 0
No
averagex = averagex/(counter+1)
Call SwitchWindow Subroutine
If SwitchWindow Is Not Successful
Yes
Stop
No
If b2 < 0
Yes
linex = (0.5*cols - b2)/m2 liney = 0.5*cols Option = 1
No
If ImageHeight >= b2 and b2 >= 0
Yes
linex = 0.5*rows liney = m2*0.5*rows+ b2 Option = 2
No
linex = (ImageHeight - 0.5*cols -b2)/m2 liney = ImageHeight - 0.5*cols Option = 3
If xpp = -1 or ypp = -1
Yes
xpp = 1/2* ImageWidth ypp = 1/2* ImageHeight
No
If DistortionFile Not= "N/A"
Yes
Input Lens Distortion Data Into Matrix D
No
counter2 = 0
If DistortionFile Not= "N/A"
Yes
xtemp = linex - xpp ytemp = ypp - liney
raddist = (xtemp ²+ ytemp²)^1/2
Search Distortion Data For Position Of raddist
correct = D_{i,1} + (D_{i+1,1} - D_{i,1}) / (D_{i+1,0} - D_{i,0}) * (raddist - D_{i,0})
xtemp = xtemp + correct * xtemp / raddist ytemp = ytemp + correct * ytemp / raddist
pointx = xtemp + xpp pointy = ypp - ytemp
No
pointx = linex pointy = liney
Return Loop
Main Line #1

Fig 5.3a: Diagram of the operation of the Cross Correlation function



Main Line #1
Dimension Window The Same As Template Window (rows, cols)
Calculate Coordinates Of Center Of Template Window (boxCenterx, boxCentery)
Call SetScrollPos Subroutine To Scroll To Point (pointx, pointy)
angle = arctan(m2) counter = 0 averagey = 0 i = 0 j =0
Obtain Current Scrollbar Position
xi = cos(angle)*(j - boxCentrex) - sin(angle)*(i - boxCentrey) yi = sin(angle)*(j - boxCentrex) + cos(angle)*(i - boxCentrey)
Truncate xi, yi To Form xint, yint
Input Pixel Grey Values Located At (xint, yint), (xint + 1, yint), (xint, yint + 1), (xint + 1, yint + 1)
Call Interpolate Subroutine
Add Interpolated Value To averagex
counter = counter + 1
If j < cols
Yes
No
j = j + 1
If i < rows
Yes
No
i = i + 1 j = 0
averagey = averagey/(counter+1)
sumx2 = 0 sumy2 = 0 sumxy = 0 counter = 0 i = 0 j =0
Calculate The Numerators Of Eq'ns 4.7, 4.8, 4.9 Using The Interpolated Grey Values And averagex, averagey, sumx2, sumy2, sumxy
counter = counter + 1
If j < cols
Yes
No
j = j + 1
i = i + 1 j = 0
If i < rows
Yes
No
Calculate Variances And Covariance sumxy = sumxy/counter sumx2 = sumx2/counter sumy2 = sumy2/counter
If sumx2 < 0.000001 or sumy2 < 0.000001
Yes
No
linex = line.x + 1 liney = m2*(line.x) + b2;
Return Loop
ro = sxy/(sx2*sy2) 1/2
If ro > largestro
Yes
No
largestro = ro targetx = pointx targety = pointy
Store Correlation Value And Coordinates In The 3x3 Matrix trial
If trial1,0 > trial2,0 and trial1,0>trial0,0
Yes
No
Call mbsort Subroutine
If counter2 < 5
Yes
No
counter2 = counter2 + 1
counter2 = 5
StoreTarget Canidate In Canidate List
Main Line #2

Fig 5.3b: Diagram of the operation of the Cross Correlation function continued

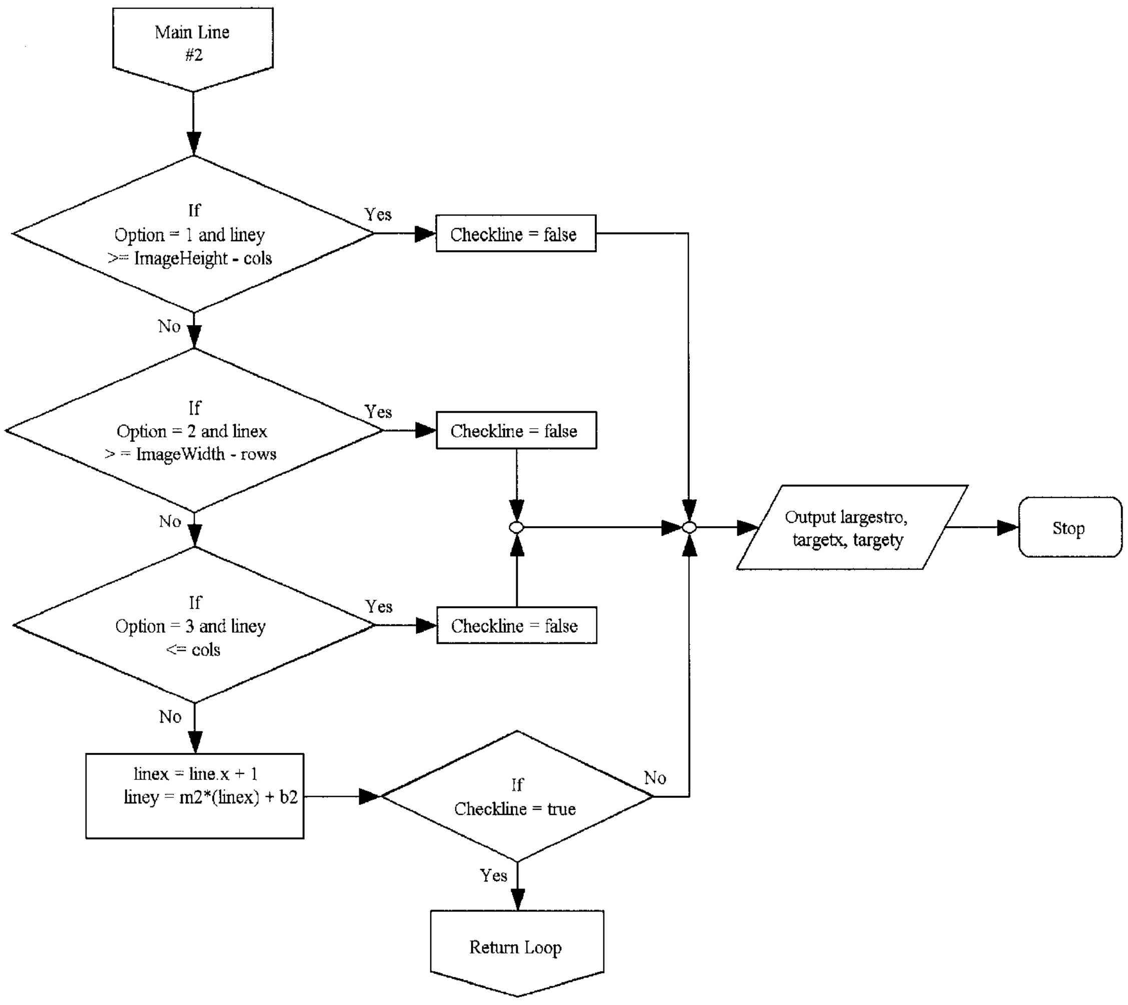


Fig 5.3c: Diagram of the operation of the Cross Correlation function continued

convenient; being that it had already been programmed for use in the template least squares matching routine.

Similar to the lens distortion correction, mentioned in the epipolar line section above, a method to match the epipolar line to the image had to be added. In this case, the ‘perfect’ epipolar line had to be reshaped to fit on the distorted image. To do this, a distortion is added to the coordinates of every point that is calculated from the line. This ensures that the centre of the target, which is being searched for, lies close to the coordinates calculated from the line. Exactly as was laid out in the previous Section, for the lens distortion correction, the correction data points are read in and a linear interpolator is used to determine the distortion for the point under scrutiny. Exactly for the reasons previously stated, a linear interpolator was chosen to do this job. Unlike before, the distortion is added to the point, instead of being removed, since the 'perfect' line is now being related back to the 'imperfect' image.

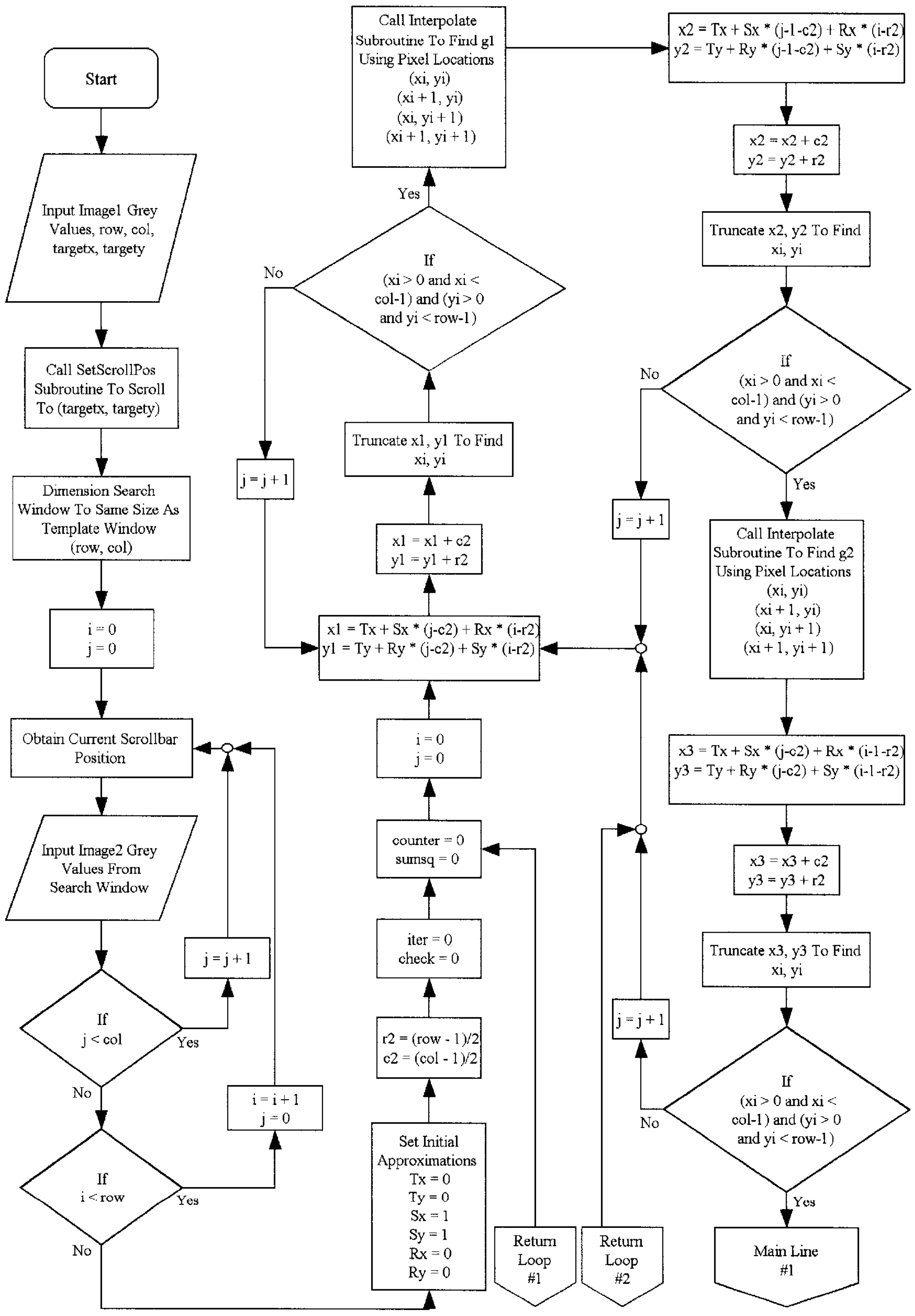
5.3 Template Least Squares Matching

The template least squares matching subroutine was created and added to Indmet to provide an alternative method of subpixel target location. Given that there are now two views of the same target loaded into Indmet, the required information to perform template least square matching is at hand. The way in which this information is used to obtain a subpixel target location in the second image is described in Section 4.3. The code for the program that was written to perform this operation, as well as the images used in the testing of this module, can be found in the appendix. Figures 5.4a, 5.4b and 5.4c present the flow diagram for the logical operation of the template least squares matching subroutine.

To perform the template least squares matching routine, it was necessary to program an interpolator that could determine grey values at decimal pixel locations. This allows the routine to shift its window under scrutiny by fractions of a pixel, so that subpixel accuracy can be achieved when determining the target centre. The interpolator that was used is the bi-linear interpolator shown in Equation 5.1.

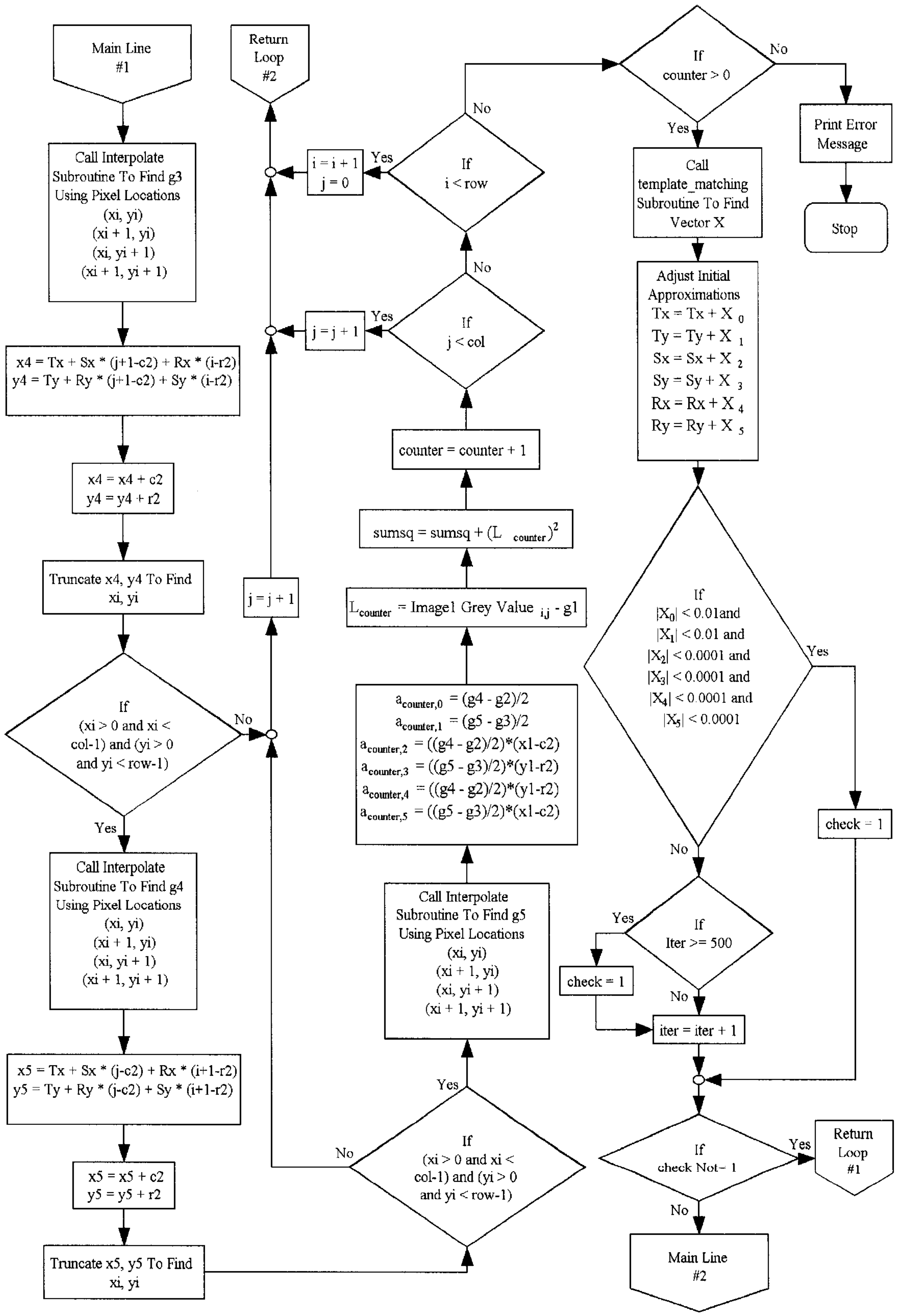
$$g_{new} = (1 - x) \cdot (1 - y) \cdot g_1 + x \cdot (1 - y) \cdot g_2 + (1 - x) \cdot y \cdot g_3 + x \cdot y \cdot g_4 \qquad \textbf{[5.1]}$$

Where, g_1 to g_4 are the grey values of the four pixels that contain the point in question and x, y are the decimal coordinates of the point. This is not the only type of interpolator



Start
Input Image1 Grey Values, row, col, targetx, targety
Call SetScrollPos Subroutine To Scroll To (targetx, targety)
Dimension Search Window To Same Size As Template Window (row, col)
i = 0
j = 0
Obtain Current Scrollbar Position
Input Image2 Grey Values From Search Window
If j < col
Yes
No
j = j + 1
If i < row
Yes
No
i = i + 1
j = 0
Set Initial Approximations
Tx = 0
Ty = 0
Sx = 1
Sy = 1
Rx = 0
Ry = 0
r2 = (row - 1)/2
c2 = (col - 1)/2
iter = 0
check = 0
counter = 0
sumsq = 0
i = 0
j = 0
x1 = Tx + Sx * (j-c2) + Rx * (i-r2)
y1 = Ty + Ry * (j-c2) + Sy * (i-r2)
x1 = x1 + c2
y1 = y1 + r2
Truncate x1, y1 To Find xi, yi
If (xi > 0 and xi < col-1) and (yi > 0 and yi < row-1)
No
Yes
j = j + 1
Call Interpolate Subroutine To Find g1 Using Pixel Locations (xi, yi) (xi + 1, yi) (xi, yi + 1) (xi + 1, yi + 1)
x2 = Tx + Sx * (j-1-c2) + Rx * (i-r2)
y2 = Ty + Ry * (j-1-c2) + Sy * (i-r2)
x2 = x2 + c2
y2 = y2 + r2
Truncate x2, y2 To Find xi, yi
If (xi > 0 and xi < col-1) and (yi > 0 and yi < row-1)
No
Yes
j = j + 1
Call Interpolate Subroutine To Find g2 Using Pixel Locations (xi, yi) (xi + 1, yi) (xi, yi + 1) (xi + 1, yi + 1)
x3 = Tx + Sx * (j-c2) + Rx * (i-1-r2)
y3 = Ty + Ry * (j-c2) + Sy * (i-1-r2)
x3 = x3 + c2
y3 = y3 + r2
Truncate x3, y3 To Find xi, yi
If (xi > 0 and xi < col-1) and (yi > 0 and yi < row-1)
No
Yes
j = j + 1
Return Loop #1
Return Loop #2
Main Line #1

Fig 5.4a: Diagram of the operation of the Template Least Squares Matching function



Main Line #1
Call Interpolate Subroutine To Find g3 Using Pixel Locations (xi, yi) (xi + 1, yi) (xi, yi + 1) (xi + 1, yi + 1)
x4 = Tx + Sx * (j+1-c2) + Rx * (i-r2)
y4 = Ty + Ry * (j+1-c2) + Sy * (i-r2)
x4 = x4 + c2
y4 = y4 + r2
Truncate x4, y4 To Find xi, yi
If (xi > 0 and xi < col-1) and (yi > 0 and yi < row-1)
Yes
No
Call Interpolate Subroutine To Find g4 Using Pixel Locations (xi, yi) (xi + 1, yi) (xi, yi + 1) (xi + 1, yi + 1)
x5 = Tx + Sx * (j-c2) + Rx * (i+1-r2)
y5 = Ty + Ry * (j-c2) + Sy * (i+1-r2)
x5 = x5 + c2
y5 = y5 + r2
Truncate x5, y5 To Find xi, yi
Return Loop #2
i = i + 1
j = 0
j = j + 1
j = j + 1
If i < row
Yes
No
If j < col
Yes
No
counter = counter + 1
sumsq = sumsq + (L counter)^2
L counter = Image1 Grey Value i,j - g1
a counter,0 = (g4 - g2)/2
a counter,1 = (g5 - g3)/2
a counter,2 = ((g4 - g2)/2)*(x1-c2)
a counter,3 = ((g5 - g3)/2)*(y1-r2)
a counter,4 = ((g4 - g2)/2)*(y1-r2)
a counter,5 = ((g5 - g3)/2)*(x1-c2)
Call Interpolate Subroutine To Find g5 Using Pixel Locations (xi, yi) (xi + 1, yi) (xi, yi + 1) (xi + 1, yi + 1)
If (xi > 0 and xi < col-1) and (yi > 0 and yi < row-1)
Yes
No
If counter > 0
Yes
No
Print Error Message
Stop
Call template_matching Subroutine To Find Vector X
Adjust Initial Approximations
Tx = Tx + X 0
Ty = Ty + X 1
Sx = Sx + X 2
Sy = Sy + X 3
Rx = Rx + X 4
Ry = Ry + X 5
If |X0| < 0.01and |X1| < 0.01 and |X2| < 0.0001 and |X3| < 0.0001 and |X4| < 0.0001 and |X5| < 0.0001
Yes
No
check = 1
If Iter >= 500
Yes
No
check = 1
iter = iter + 1
If check Not= 1
Yes
No
Return Loop #1
Main Line #2

Fig 5.4b: Diagram of the operation of the Template Least Squares Matching function continued

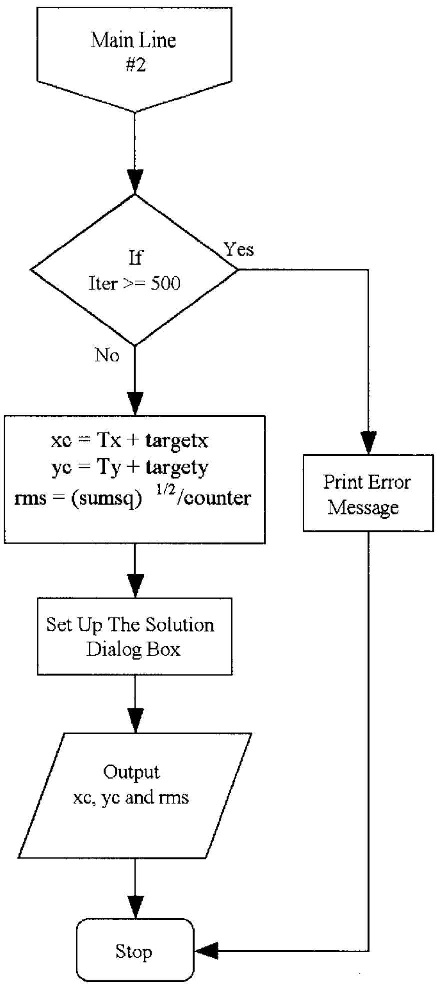


Fig 5.4c: Diagram of the operation of the Template Least Squares Matching function continued

that could have been used. Other types, such as a sinc function interpolator are common when interpolating between image pixels. The advantage of using a bi-linear interpolator is that it is a fast method of interpolation that requires little computational expense. This becomes a valuable asset considering that every pixel in the template and test regions have to be interpolated.

Another challenge that was encountered during the writing of the template least squares matching routine, involved the determination of the derivatives for the left and right image, grey level functions. Since no mathematical functions were formulated to describe these grey value functions, it was necessary to use numerical derivation to calculate these derivatives. To perform this type of derivation, it was necessary extract, in addition to the pixels own grey value, the four grey values that were directly above, below, to the right and left of the pixel in question. The derivatives that were used are shown in Equation 5.2 below:

$$\frac{\partial R^0(x,y)}{\partial x_R} = \frac{(g_{x+1} - g_{x-1})}{2} \qquad \textbf{[5.2a]}$$

$$\frac{\partial R^0(x,y)}{\partial y_R} = \frac{(g_{y+1} - g_{y-1})}{2} \qquad \textbf{[5.2b]}$$

Where R(x,y) is the grey value function of the right search image and x, y are coordinates of the point under consideration. Using the surrounding pixels, it was possible to determine the slope of the function at any particular point in the search window, with one exception. Due to the fact that the pixels at the edge of the search window are not completely surrounded by other pixels, it was necessary to remove these pixels from direct least squares computation. They were only used to determine the point derivatives of the pixels in the adjoining layer.

Template least squares matching is, by nature, an iterative process. It is necessary, therefore, that appropriate exit conditions be chosen. This is important so that the answer is sufficiently close to the target centre, while at the same time not tying up computer resources and thereby slowing execution of the program. As shown in Section 4.3, the adjustment has unknown parameters. These parameters are the shift, in the x and y directions, the scale in the x and y direction and the rotation in the x and y directions. The adjustment provides corrections for these parameters, based on the initial assumptions of

$$T_x = 0$$
$$T_y = 0$$
$$S_x = 1$$
$$S_y = 1$$
$$R_x = 0$$
$$R_y = 0$$

Where T_x and T_y are the shift parameters, S_x and S_y are the scale factors, and R_x and R_y are the rotation parameters. The program is adjusted so that the iteration loop that encompasses least squares adjustment will exit when the calculated corrections satisfy the following criteria

$$|\Delta T_x| < 0.01$$
$$|\Delta T_y| < 0.01$$
$$|\Delta S_x| < 0.0001$$
$$|\Delta S_y| < 0.0001$$
$$|\Delta R_x| < 0.0001$$
$$|\Delta R_y| < 0.0001$$

These criteria were chosen based on the first derivatives of Equations 4.11c and 4.11d from Chapter 4 and the fact that answers more accurate than 1/100th are not possible using this method. Taking the first derivatives of Equation 4.11c, with respect to T_x, S_x and R_x, then splitting this derivative into its component differentials, results in the following equations.

$$dx = dT_x \quad \textbf{[5.3a]}$$
$$dx = x \cdot dS_x \quad \textbf{[5.3b]}$$
$$dx = y \cdot dR_x \quad \textbf{[5.3c]}$$

Now, having already chosen 1/100th of a pixel as the absolute value of the smallest accuracy possible and assuming that the largest target that will be matched is 100 pixels by 100 pixels, the exit limits for the template matching routine can be determined.

$$|dTx| = |dx| = 0.01 \qquad \textbf{[5.4a]}$$

$$|dSx| = \frac{|dx|}{x} = \frac{0.01}{100} = 0.001 \qquad \textbf{[5.4b]}$$

$$|dRx| = \frac{|dx|}{y} = \frac{0.01}{100} = 0.001 \qquad \textbf{[5.4c]}$$

Similarly, Equation 4.11d is differentiated and split, to obtain the error limits for the remaining three adjustment parameters. These error limits will provide answers that will be sufficiently accurate, while limiting the number of iterations so as to limit the amount of time each matching session takes. If these criteria are not met within a predetermined iteration limit of 50, the program prints an error message to the screen and returns control of the program to the user, so more target matching can be done.

6.0 Testing Procedures and Results

To ascertain if the three modules that have been added to INDMET are operating as expected, tests have been performed. These tests have been conducted on each module separately to ensure that every part of the target matching solution is functioning as expected. The tests involve the use of digital images that were kindly supplied by the Applanix Corporation. A box-like target rig, upon which white targets containing black circles have been logically dispersed, is the subject of these images. Information on focal lengths, camera orientations, camera positions and camera distortions, associated with the images, has also been provided. This information is imperative for the proper functioning of the target matching solution.

The tests come in three stages. The epipolar line test is designed to determine how well the calculated epipolar line overlaps the desired target. The cross correlation test is concerned with the success rate of locating targets well enough for the template least squares module to operate. Finally, the template least squares tests are concerned with finding how much of the target must be in the search window for a reliable match to be made and how accurately subpixel values can be determined from the target images.

6.1 The Epipolar Line Module

The accuracy with which epipolar lines are calculated was tested by determining the distance from the closest point on the calculated line, to the centre of the corresponding target in the second image of the stereo pair. To perform this test, two stereo pairs were used, and ten targets were located within each stereo pair. This means that an overall total of twenty targets were used. The centres of these targets were determined, using the subpixel moment method that was already a part of the INDMET program. Each stereo pair that was used demonstrates a unique situation. The first stereo pair, which uses AHO as the left template image and CHO as the right search image, demonstrates a situation where two images are taken at equal heights, but one image is at a significant angle to the rig, while the other image is more or less flat. This means that targets in the skewed image, i.e. AHO will be far more elliptical than the targets in the second image. In the other situation, both images are flat to the target rig, but were taken at differing heights above the datum. This means that the targets in each image are more or less similar in respect to shape and size. All four of these images can be found in the appendix, along with their associated orientation and distortion data.

Table 6.1: Results of Epipolar Line Routine for Images AHO and CHO

Target	Point On Epipolar Line (Pixels)		Target Centre (Pixels)				Difference (Pixels)	
No.	X	Y	X	Y	RMS	Type	X	Y
101	1464	1482	1465.081	1485.241	0.034	Auto Box	-1.081	-3.241
102	1453	469	1453.454	471.502	0.032	Auto Box	-0.454	-2.502
104	1067	1462	1069.487	1465.011	0.032	Auto Box	-2.487	-3.011
117	711	505	710.564	508.852	0.044	Auto Box	0.436	-3.852
119	398	1462	396.427	1465.909	0.028	Auto Box	1.573	-3.909
120	379	631	379.763	633.775	0.045	Auto Box	-0.763	-2.775
218	767	2681	767.466	2684.742	0.034	Auto Box	-0.466	-3.742
221	447	2670	448.85	2668.641	0.019	Box	-1.85	1.359
237	555	2491	553.369	2492.576	0.02	Box	1.631	-1.576
312	2760	2897	2763.238	2898.328	0.017	Box	-3.238	-1.328

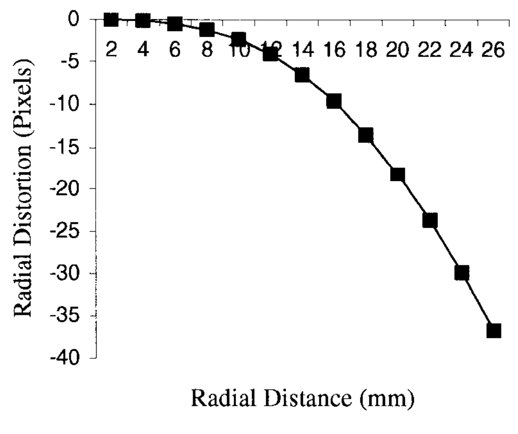
Table 6.2: Results of Epipolar Line Routine for Images CHO and CLO

Target	Point On Epipolar Line (Pixels)		Target Centre (Pixels)				Difference (Pixels)	
No.	X	Y	X	Y	RMS	Type	X	Y
206	1021	3192	1022.043	3187.659	0.037	Auto Box	-1.043	4.341
217	721	3592	722.942	3595.645	0.038	Auto Box	-1.942	-3.645
236	506	3268	506.567	3263.936	0.045	Auto Box	-0.567	4.064
321	3225	2771	3223.304	2765.179	0.022	Box	1.696	5.821
336	2910	3290	2913.444	3294.582	0.042	Auto Box	-3.444	-4.582
407	2985	1653	2987.106	1657.43	0.028	Auto Box	-2.106	-4.43
408	3013	869	3015.978	869.296	0.035	Auto Box	-2.978	-0.296
411	2624	981	2626.594	976.998	0.028	Auto Box	-2.594	4.002
419	3276	1654	3278.472	1652.901	0.045	Auto Box	-2.472	1.099
420	3318	897	3314.687	891.823	0.044	Auto Box	3.313	5.177

Table 6.1 shows, out of ten trials, the epipolar line is never more that 4 pixels away from the target centre. Table 6.2, on the other hand, shows a maximum deviation of almost 6 pixels. This amount of deviation is acceptable, since the target images that were used are already assumed to appear in multiple pixels, so that INDMET's subpixel routines can find the centre. In fact, the larger the target in the image, the more accurately the target centre should be determined. Therefore, if the target is only 6 pixels wide, it is highly dubious that the results could be obtained from either subpixel locator and, if they are, it is doubtful they will be of much practical use.

To achieve this level of accuracy, when creating the epipolar line, the module has to incorporate corrections for lens distortion and the principal point offset. While calculating a straight epipolar line is useful, comparing it to large images that contain these errors will result in significant deviation of the line from the target in the image. For the test, the image that was used contained lens distortions that are shown in Figure 6.1. The radial lens distortions contained in this image can reach up to 37 pixels near the edges. This represents a significant deviation from the epipolar line and can cause the matching process to fail, if not corrected.

Fig 6.1: Radial Lens Distortions for Epipolar Line Test



6.2 The Cross Correlation Module

The ability of the cross correlation module to locate the required target was tested by simply running the matching algorithm and recording the results. The same sets of stereo pairs, as well as the same twenty targets were used. The results of this test are shown in Tables 6.3 and 6.4. The cross correlation routine stores the top five correlation values in a list. The table column entitled, 'List Position', refers to the position of the actual corresponding target in this list. The entries that are labelled 'Did Not Appear' refers to the fact that the corresponding target in the second image of the stereo pair did not appear as one of the top five correlation values.

Table 6.3: Results of Cross Correlation Routine for Images AHO and CHO

Target No.	List Position	Matched?	X (Pixels)	Y (Pixels)	RMS (Pixels)
101	Did Not Appear	No	-	-	-
102	3	No	-	-	-
104	Did Not Appear	No	-	-	-
117	2	No	-	-	-
119	5	No	-	-	-
120	1	Yes	380.251	633.654	0.16
218	2	No	-	-	-
221	Did Not Appear	No	-	-	-
237	1	Yes	554.027	2492.684	0.315
312	Did Not Appear	No	-	-	-

Table 6.4: Results of Cross Correlation Routine for Images CHO and CLO

Target No.	List Position	Matched?	X (Pixels)	Y (Pixels)	RMS (Pixels)
206	1	No	-	-	-
217	1	Yes	723.331	3595.901	0.202
236	1	No	-	-	-
321	Did Not Appear	No	-	-	-
336	1	Yes	2913.147	3295.219	0.376
407	1	Yes	2987.123	1657.349	0.16
408	2	No	-	-	-
411	1	Yes	2626.783	977.353	0.247
419	1	Yes	3278.454	1653.781	0.241
420	Did Not Appear	No	-	-	-

Table 6.3 clearly shows that 4 out of 10 matches did not appear in the top five. These targets, while having high correlation values, did not have high enough correlation

values to make the list. Another 4 out of the 10 matches made the top five correlation list. The remaining two targets went through the entire target matching solution and were correctly identified and matched. This represents a matching success rate of approximately 20%, when one image sits at a skewed angle with respect to the other.

Table 6.4, on the other hand, shows a much higher success rate. Out of the ten targets that were tested in this stereo pair, no less than five were successfully matched. Two more of these targets appeared number one in the cross correlation list, but due to the fact that the template least squares matching routine reached its preset iteration limit, no target centre information was generated. One of the targets was number two on the list, indicating that it was a strong contender as a target match. Finally, only two failed to make the list. Therefore, due to the fact that a total of 7 targets appeared first in the list, a matching success rate of 70% was achieved, for the case where image targets have similar elliptical dimensions.

Unfortunately, targets are not the only things that the module correlates as a match. The target field is set up on a steel cube structure, where the targets are attached to the posts. The cross correlation module routinely finds high correlation values when scanning over these posts. In many instances, these posts can make the top five list which confuses the module. The results of Table 6.4 partially occur because the image geometry of this stereo pair dictates that the epipolar lines will run fairly vertical, meaning that far fewer of the prominent vertical posts will be crossed by the routine. It also means that fewer targets will be encountered by the routine, since these targets are attached to the posts.

It should be noted that the images used in this cross correlation test were of poor quality. The targets were generally blurred and the target contrast edges were not well defined. These images were used since images of better quality, that had all the necessary associated orientation and distortion information, could not be obtained.

6.3 The Template Least Squares Matching Module

The first test that was performed involved using two identical images of a target to test whether the least square adjustment was adequately correcting for shifts in the target. Therefore, the pixel window was shifted from the centre of the target on the right image by increments to ensure that the program corrected each shift. This was done in the four cardinal directions, along the axes, to make sure that all possible shifts were dealt

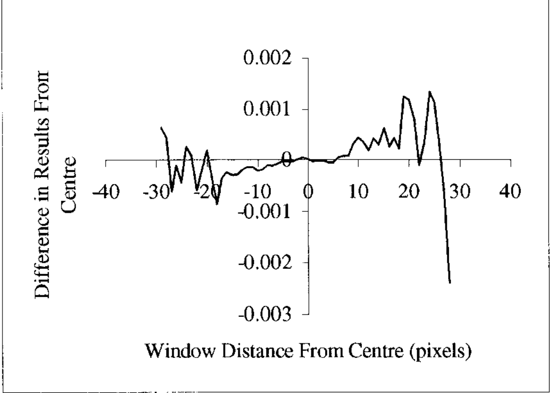


Fig 6.2a: Difference in Results of X Coordinate Due to X Axis Shift in Window

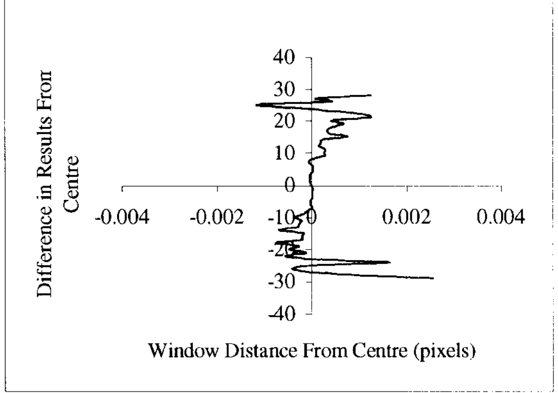


Fig 6.2b: Difference in Results of Y Coordinate Due to X Axis Shift in Window

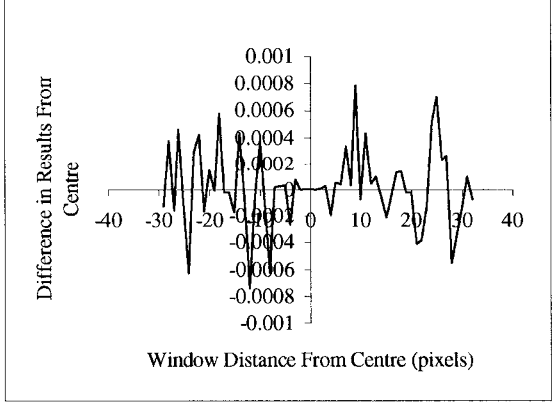


Fig. 6.2c : Difference in Results of X Coordinate Due to Y Axis Shift in Window

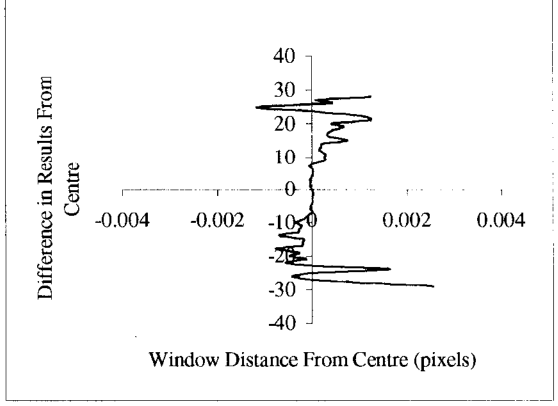


Fig 6.2d: Difference in Results of Y Coordinate Due to Y Axis Shift in Window

with. It was found that the template least squares matching subroutine had a pull-in range of approximately half the target size. The four graphs in Figure 6.2, show the difference between the centre coordinates produced by the template least squares matching program and the known target centre. These graphs show how the calculated points vary as the comparison window is moved first along the x axis, then along the y axis. The graphs of Figure 6.2 demonstrate how the calculated error increases, as parts of the target edge are lost to the adjustment. These graphs also show that there is little variation in the answers that are being returned by the program, and that these errors are well within the accuracy range of INDMET's original sub-pixel location routine (1/20th to 1/50th of a pixel). The routine also provides estimations of the accuracy for which the target centre is located. Using the root mean square (RMS) formulation on the difference between the grey values of the template and the search images allows this estimate to be calculated. Figure 6.3 shows that for each comparison window location, the estimated RMS error for the match was always below 0.004 of a pixel. Again, this value is well within the expected accuracy for the system. Lastly, the number of iterations, which is an indication of the running time of the program, must be kept low to ensure that the routine is practical. The graphs shown in Figure 6.4, represent the number of iterations of the routine at each window location. The graphs end when the iteration limit is reached. As can be seen in the graphs of Figure 6.4, the number of iterations that the routine requires is less than one hundred, right up to the point where the routine can no longer match the target accurately.

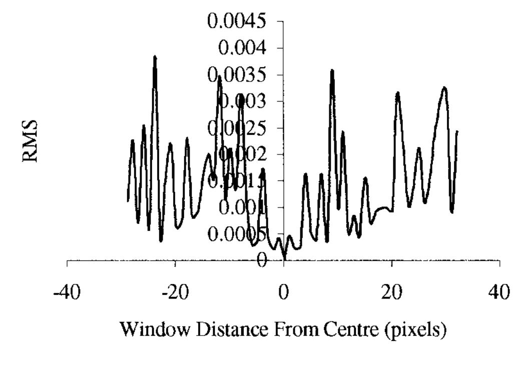


Fig 6.3a: RMS Values as Target Window Shifts Along X Axis

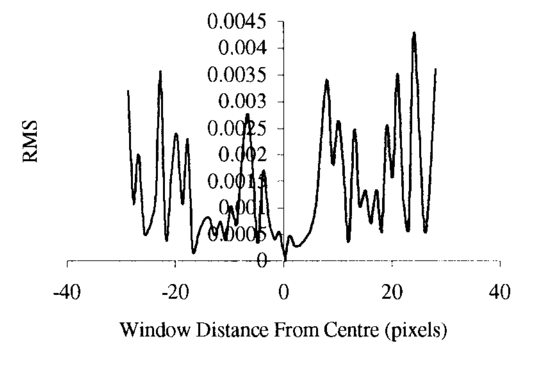


Fig 6.3b: RMS Values as Target Window Shifts Along Y Axis

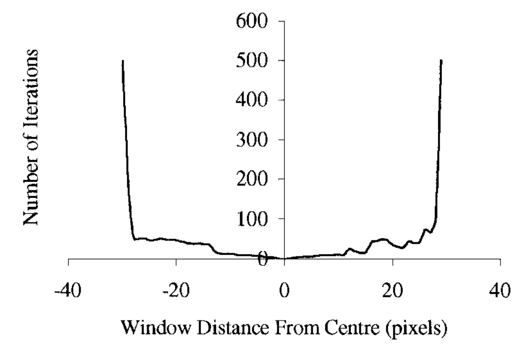


Fig 6.4a: Iterations as Target Window Shifts Along X Axis

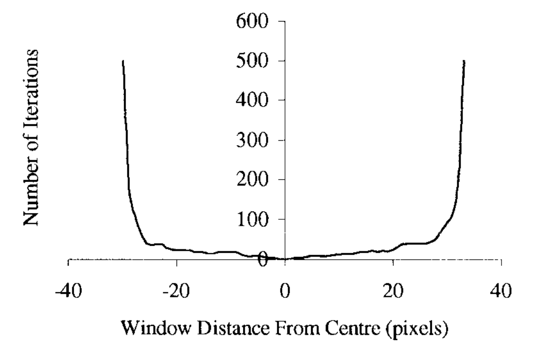


Fig 6.4b: Iterations as Target Window Shifts Along Y Axis

The second test that was preformed involved seeing how the template least squares matching routine compared to INDMET's moment routine. An image containing four targets was used and the four targets were compared to one another. Each target served as the template image as well as the search area image. Table 6.5 lists the results of this analysis, where the targets serving as the template image are listed in the first column and the targets serving as the search area image, are listed in the first row. The column labelled 'Moment Method' presents the target centre as calculated by the original INDMET subpixel centre location routine. As the table shows, the two routines do not precisely agree in terms of the centre of the targets. The two methods are within less

Table 6.5: Template Least Squares Matching Results for Four Targets

Target No.	1	2	3	4	Moment Method
1	-	xc = 92.277 pixels yc = 237.579 pixels RMS = 0.142 pixels	xc = 93.043 pixels yc = 237.768 pixels RMS = 0.103 pixels	Failed To Converge	xc = 92.329 pixels yc = 237.887 pixels RMS = 0.011 pixels
2	xc = 406.632 pixels yc = 154.362 pixels RMS = 0.121 pixels	-	xc = 406.716 pixels yc = 154.121 pixels RMS = 0.113 pixels	Failed To Converge	xc = 405.923 pixels yc = 154.181 pixels RMS = 0.012 pixels
3	xc = 545.957 pixels yc = 401.252 pixels RMS = 0.127 pixels	xc = 545.169 pixels yc = 400.916 pixels RMS = 0.167 pixels	-	Failed To Converge	xc = 545.258 pixels yc = 401.227 pixels RMS = 0.009 pixels
4	xc = 359.725 pixels yc = 390.798 pixels RMS = 0.114 pixels	xc = 358.752 pixels yc = 390.241 pixels RMS = 0.116 pixels	xc = 359.825 pixels yc = 390.315 pixels RMS = 0.114 pixels	-	xc = 359.152 pixels yc = 390.933 pixels RMS = 0.007 pixels

than a pixel of one another and the RMS error for the template least squares indicates that these values are determined poorer than the values determined from the moment method. In fact, the results show that the template least squares matching routine can locate targets to approximately 1/10th of a pixel, while the original INDMET moment routine was able to achieve accuracies of 1/20th of a pixel or better. This discrepancy could possibly be caused by the fact that the simplest least square adjustment was used to match the targets. The use of a more complicated mathematical model might be suggested by this discrepancy. It is also of note that template four failed to converge when the other targets were being used as the template. This is almost certainly due to the fact that target four is nearly twice the size of the other targets. Being twice the size of the other targets means, parts of the target edges might have been clipped out of the comparison window, since the routine determines the window's size based on the template target's dimensions.

7.0 Conclusions and Recommendations

7.1 Conclusions

The thesis presented here demonstrated a single target matching solution to locate specific targets for the purpose of high precision photogrammetric measurement. This solution included the programming and detailed testing of three specific subroutines in the C++ language. Testing was conducted on poor quality images to prove that these subroutines could handle images of this type and still function. Completion of this thesis has provided future Ryerson students with access to high precision target location software. While researching this topic, a large volume of information was gathered on automatic target matching algorithms. This information has been presented here to educate readers in the goals and methodologies of target matching, as well as provide a platform upon which future researchers may build.

Target matching is an ever-expanding field of study. Numerous organizations and professions specifically deal with a machine's ability to locate, track and identify targets from digital images. These groups include photogrammetrists, computer vision specialists, robotics experts, neurophysiologists and those interested in remote sensing. Through these groups, numerous matching techniques, search methods and other mathematical theories have been developed. These techniques can be broken down into three categories; signal based matching, feature based matching and relation based matching. Each of the techniques, which are contained within these categories, tries to duplicate a specific aspect of the human visual system. The first purpose of this thesis has been to describe many of these techniques, illustrating how each uses different properties, inherent to the digital image, to locate, track, or identify specific items within the image. It now seems appropriate to summarize these categories.

Signal-based matching uses the relationship between the brightness, or grey values in the neighbourhood of the target point. The resemblance between grey values is used to determine whether the image points are similar. The most popular signal matching routines include least squares matching, cross correlation and Euclidean distance. One of the advantages of using signal based matching is that it is fully compatible with the tried and tested concepts of signal processing. Radiometric restoration, smoothing and edge detection are particularly enhanced through the use of a signal based solution. The two main disadvantages that exist with signal matching routines are that a match only occurs if the target image has undergone a simple shift and that grey value distributions do not correspond to physical entities.

Feature-based matching uses image features, such as points, lines and areas, to locate and match targets. Most feature based systems are combined with the epipolar geometry to limit the amount of area that is to be considered. The actual searching is performed by one of a number of interest operators that have been created over the years. The most common of these operators are the Förstner, Dreschler, Marr-Hildreth and Moravec interest operators. The success of feature based matching depends on three key properties. First, the discreetness of individual points is critical. The more discreet the point, the better chance the computer has to match it in the second image. Second, the similarity, which is a measure of how closely two points resemble each other. Third, the consistency provides a measure of how well a particular match conforms to surrounding matches, based on some general object model.

Relation-based matching is the most complex of the three matching schemes. It uses image features and information on how these features relate to each other, to create structural sets. These structural sets are used to compare the images and establish the location of image targets. Due to the nature of structural image matching techniques, no a-prior information is required and, therefore, relational matching may be considered a fully automated technique. This is true because relation based matching techniques are able to utilize topological and geometrical relations among image features. The main disadvantages of relation based matching are that efficient acquisition of structural descriptions is difficult and the operations used to match them are very complex.

The second purpose of this thesis was to demonstrate how a precise target matching solution might be performed. The software package INDMET was used as the basis for applying this target matching solution. The software upgrade included three major sub-routines involving epipolar lines, cross correlation and template least squares matching. These routines limited the search area, search for target matches and performed subpixel target location respectively. It was found that this target matching solution was able to locate the correct targets between 20% and 70% of the time, depending on how similar the elliptical targets appeared in the images. Once found, the program was able to perform subpixel target centre location, accurate to about 1/10th of a pixel. Different mathematical models for the cross correlation and template least squares matching modules might produce better results. Automatic target matching is a field where there is always a better way to perform the matching. Despite the fact that this is a multidisciplinary field, with many people working on ways to improve computer vision, no model, algorithm or program has yet been able to approach the human ability to recognize and associate objects. This will probably remain the case until people are able to create a machine that can think and reason for itself. Until then, automatic target matching will continue to develop, adding new algorithms to the plethora that already exist.

7.2 Recommendations

Due to the nature of target matching and the plethora of matching routines available, there are many changes to a target matching solution that are possible. The routine presented in this thesis has a number of areas where improvement is possible. The cross correlation module has problems identifying the true target from the other targets and similar objects that lie along the epipolar line. Some sort of consistency check would seem to be needed. An example of such a check would be the coplanarity condition. Since the images can be linked by the epipolar line, which is simply a representation of the intersection of the epipolar plane with the image planes, the true targets in each image must lie precisely in the same epipolar plane. Another option would be to use a different form of the cross correlation method, such as the Fast Fourier Transform method that is outlined in Section 2.2.2.

The template least squares matching module also has a number of features that could be changed to improve its performance. Even though a simple least square adjustment was used, a more complicated mathematical model might bring better results. Gruen and Baltsavias (1986) presented a model that not only adjusts the search area target, but also adjusts the template target. This might bring better results when the template target is significantly smaller than the search area target or when the template target is significantly skewed. Also, considerable interpolation is required for the purposes of resampling the image whenever a shift is made. The interpolator that was chosen was of a bilinear nature. This type of interpolator is fast but rather simplistic. It is unable to compensate for aliasing errors, which represent the bulk of the interpolation errors. A better interpolator would be the so-called 'sinc' function. This type of interpolator has a smoothing effect on the interpolated pixels, producing a clearer interpolated image. Clear interpolated images mean that the target edges will be better defined, allowing more accurate determination of target locations to subpixel values.

References

Applanix Corporation. 2002. *Camera Orientation Data.* Personal Communication. www.applanix.com

Bacher, U., Bludovsky, S., Dorrer, E. and Münzer. 1999. *Precision Aerial Survey of Vatnajökull, Iceland by Digital Photogrammetry.* In: Altan, M.O. & Gruendig, L. (eds.): Towards a Digital Age. – Proceedings of the Third Turkish German Joint Geodetic Days, Istanbul, pp. 127-136.

Barnard, S. T. 1987. *Stereo Matching by Hierarchical Microcanonical Annealing.* Proceedings of the International Joint Conference on Artificial Intelligence, Milan, Italy, pp. 832-835.

Bourke, P. 1996. *Cross Correlation.* http://astronomy.swin.edu.au/~pbourke/analysis/correlate/auto

Chen, J., Clarke, T.A., Robinson, S. and Grattan, K.T.V. 1994. *An Optimized Target Matching Based on A 3-D Space Intersection and A Constrained Search for Multiple Camera Views.* Conference on Videometrics III, Boston, USA, SPIE Volume 2350, pp 324-335.

Cosandier, D. and Chapman, M. 1992. *High Precision Target Location for Industrial Metrology.* Presented at Videometrics, SPIE OE/Technology. Boston, Mass., Volume 1820, pp. 111-122.

Dew, G. and Holmlund, K. 2000. *Investigations of Cross-Correlation and Euclidean Distance Target Matching Techniques in The MPEF Environment.* EUMETSAT (Europe's Meteorological Satellite Organisation), Proceedings of the Fifth International Winds Workshop, Session V, Lorne, Australia. pp. 235-244.

Dreschler-Fischer, L. 1987. *A Blackboard System For Dynamic Stereo Matching.* Intelligent Autonomous Systems, Elsevier Science Publishers, Amsterdam. pp. 189-202

Förstner, W. 1984. *Quality Assessment of Object Location and Point Transfer Using Digital Image Correlation Techniques.* invited paper to Commission III, 15th ISPRS Congress, Rio De Janeiro, pp. 169 - 191

Förstner, W. and Gülch, E. 1987. *A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features.* Proceedings of ISPRS Intercommission Conference on fast processing of photogrammetric data, Interlaken, pp.281 - 305

Forlani G. Giussani A., Scaioni M. 1996. *Target Detection and Epipolar Geometry for Image Orientation in Close-Range Photogrammetry.* ISPRS, Commission V, pp. 518 – 523

Grimson, W. E. L. 1981. *From Images to Surfaces: A Computational Study of the Human Early Visual System.* MIT Press, Cambridge, MA.

Gruen, A.W. 1985. *Adaptive Least Squares Correlation: A Powerful Image Matching Technique*, South Africa Journal of Photogrammetry, Remote Sensing and Cartography. pp. 175-187.

Gruen, A.W. and Baltsavias, E.P. 1986. *High Precision Image Matching for Digital Terrain Model Generation.* ISPRS, Commission III, Helsinki Finland.

Julesz, B. 1971. *Foundations of Cyclopean Perception.* Univ. of Chicago Press, Chicago, IL.

Kropatsch, W., Bischof, H., and Englert, R. 2000. *Digital image analysis: selected techniques and applications.* Springer, New York. pp. 211-230.

Lemmens, M. 1988. *A Survey On Stereo Matching Techniques.* Proceedings of the ISPRS, vol. 27, pp. 11-23.

Lewis, J.P. 1995. *Fast Normalized Cross-Correlation.* Vision Interface, pp. 120-123

Luhmann, T. and Altrogge, G. 1986. *Interest-Operator For Image Matching.* Invited PAPER, symposium COMM. III, Rovaniemi, pp.459-472.

Marr, D. and Poggio, T. 1979. *A Computational Theory Of Human Stereo Vision.* Proceedings of the Royal Society of London, Volume B 204, pp 301-328.

Milgram, D.L. and Rosenfeild, A. 1978. *Algorithms and Hardware Technology for Image Recognition.* Final report to U.S. Army Night Vision and Electro-Optics Lab., Fort Belvoir, Va.

Mital, D.P., Teoh, E.K. and Amarasinghe S.K. 1996. *Automated Matching Technique for Identification of Fingerprints.* SPIE Proceedings, Volume 2908, pp. 58-64.

Olsen C.F. and Huttenlocher D.P. 1997. *Automatic Target Recognition by Matching Oriented Edge Pixels.* IEEE Transactions On Image Processing, Vol.6, No.1, pp.103-113.

Olson, C.F. 2000. *Maximum-Likelihood Image Matching.* IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 24, No. 6, pp. 853-857.

Pajares, G., Manuel de la Cruz, J. and Lopez-Orozco, J.A. 2000. *Relaxation Labeling In Stereo Image Matching.* Pattern Recognition, pp. 53-68.

Robert, L., Deriche, R. and Faugeras, O. D. 1992. *Dense Depth Map Reconstruction Using Multiscale Regularization.* Proceedings Of The Singapore International Conference on Image Processing, Singapore, pp. 123-127.

Bibliography

Candocia, F. and Adjouadi, M. 1997. *A Similarity Measure for Stereo Feature Matching.* IEEE Transactions On Image Processing, Vol. 6, No. 10, pp. 1460-1464

Chai, J.and Ma, S. 1998. *Robust Epipolar Geometry Estimation Using Genetic Algorithm.* Pattern Recognition Letters, pp. 829 - 838

Chen, J. Clarke, T.A. & Robson. S., 1993. *An alternative to the epipolar method for automatic target matching in multiple images for 3-D measurement.* Optical 3-D measurements techniques II, Pub. Wichmann, Karlsruhe, pp. 197-204.

Cho,W. and Shenk,T. 1992. *Resampling Digital Imagery to Epipolar Geometry.* Presented At the XVII Congress of ISPRS, Columbus, Ohio, pp.37-43

Di Stefano, L., Marchionni, M., Mattoccia, S. and Neri, G. 2002. *A Fast Area-Based Stereo Matching Algorithm.* 15th IAPR/CIPRS International Conference on Vision Interface, Calgary, Canada.

Förstner, W. 2000. *New Orientation Procedures.* Proceedings of the 19th ISPRS Congress, Amsterdam.

Hahn, M. and Förstner, W. 1988. *The Applicability of a Feature Based and a Least Squares Matching Algorithm for DEM Generation.* International Archives of Photogrammetry and Remote Sensing, Volume 27, No. 3, pp.137-150

Jin, L.W., Chan, K.P. and Xu, B.Z. 1995. *Consistent Relaxation Matching For Hand Written Chinese Character Recognition.* Proceedings of the Fifth IEEE International Conference on Image Processing and Its Application, Edinburgh, U.K., pp. 55-59

Okutomi, M. and Kanade, T. 1992. *A Locally Adaptive Window For Signal Matching.* International Journal of Computer Vision, Vol. 7, No. 2, pp.143-162

Pla, F. and Marchant, J.A. 1997. *Matching Feature Points in Image Sequences through a Region-Based Method.* Computer Vision And Image Understanding, Vol. 66, No. 3, pp. 271–285

Sainath, S. and Sarkar S. 1998. *An Approximate Algorithm For Structural Image Matching.* Proceedings of the IEEE International Conference on Image Processing, Chicago, Illinois, Vol. 1, pp. 798-802

Suel, M., O'Gorman, L. and Sammon, M.J. 2000. *Practical Algorithms For Image Analysis Description, Examples, Code.* Cambridge University Press, New York, New York, U.S.A., pp. 106-109

Zhang, Z. 1988. *A New Approach of Epipolar-line Matching for Improving the Performance of SODAMS system.* ISPRS 16th, Commission III, Kyoto, Japan.

Zitnick, C.L. and Kanade, T. 1999. *A Cooperative Algorithm for Stereo Matching and Occlusion Detection.* Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

Appendices

Rosenholm, D. 1987. *Empirical Investigation of Optimal Window Size Using the Least Squares Image Matching Method.* Photogrammetria, pp. 113-125.

Russell D. 2003. *Beca Simons Photogrammetry Web Site.* http://www.becasimons.co.nz/Photogram.htm

Tabatabai, A.J. and Mitchell, O.R. 1984. *Edge Location to Subpixel Values in Digital Imagery.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-6, No. 2, pp.188-201.

Vosselman, G. 1994. *Use Of Tree Search Methods In Digital Photogrammetry.* International Archive Of Photogrammetry And Remote Sensing, Commission III Symposium, Munich. pp. 886-893.

Wang, Y. 1998. *Principles And Applications Of Structural Image Matching.* ISPRS Journal Of Photogrammetry & Remote Sensing, Volume 53, pp. 154-165.

Lü,Y. and Zhang, Z. 1988. *Fast Implementation For Generating Epipolar Line Images With One-Dimensional Resampling.* 16th ISPRS Congress, IAPRS 27, B3, Com. III, Kyoto Japan, pp. 511-520.

Zhang, Z. 1998. *Determining The Epipolar Geometry And Its Uncertainty: A Review.* International Journal Of Computer Vision, Volume 27 No.2, Kluwer Academic Publishers, Netherlands, pp. 161-195.

Appendix A

Source Code For The INDMET Upgrade

To keep the size of the appendices within reason, the complete program code will not be presented here. The main program functions are listed in their entirety, but the numerous other functions and procedures, which are used by the main functions, are not. These little routines will be introduced by their function headers and accompanied by brief descriptions of their function. The main control functions are also listed here. These control functions set up and use the main functions to perform the target matching solution. They also will only be introduced by function heading and briefly described. The full code for all these functions can be found on the CD accompanying this thesis.

Main Functions

```
/*****************************************************************
CALCEPIPOLARLINE : CALCULATES THE PARAMETERS OF THE EPIPOLAR
LINE

        range:  pointx - x coordinate of the template target
                pointy - y coordinate of the template target
                m1 - returns the slope of epipolar line on the template image
                b1 - returns the y intercept of the epipolar line on the template
                     image
                m2 - returns the slope of epipolar line on the search image
                b2 - returns the y intercept of the epipolar line on the search image
                xpp - the x coordinate of the principal point
                ypp - the y coordinate of the principal point
                DistortionFile - the file containing lens distortion data

This function computes the slope and y intercept parameters of the epipolar line. It is
designed to operate in conjuction with CalcEpipolarInput(). The CalcEpipolarInput()
function collects the organizes the size and orientation data for use here. Through use of
the global variables sizeTotal, orien1[4][2] and orien2[4][2], the information is relayed to
this function.
**********************************************Michael Leslar 06/18/2002*/

void CEpipolarDlg::CalcEpipolarLine(double pointx, double pointy, double &m1, double
&b1, double &m2, double &b2, double &xpp, double &ypp, CString &DistortionFile){
        double A[3][3];
        double b_vector[3];
        double beta1, beta2;
        double xa, ya, xb, yb, focal;
        double ua, va, ub, vb;
        double pi = 4*atan(1);
        double xtrans, ytrans;
```

```
// sets the coordinate system to the principal point
// or if no principal point entered, then sets
// coordinate system to the image center
if((orien1[4][0] == -1)||(orien1[4][1] == -1)){
        xtrans = 0.5*sizeTotal.cx;
        ytrans = 0.5*sizeTotal.cy;
}
else{
        xtrans = orien1[4][0];
        ytrans = orien1[4][1];
}

xa = pointx - xtrans;
ya = ytrans - pointy;

// if a lens distortion file entered, CorrectLensDistortion corrects
// the coordinates entered
if(DistortionFile1 != "N/A")
        CorrectLensDistortion(DistortionFile1, xa, ya,1);

// condition statement to ensure that the direction of
// the basis vector corresponds to the geometry
if(orien1[0][0] > orien2[0][0]){
        b_vector[0] = orien2[0][0] - orien1[0][0];
        b_vector[1] = orien2[1][0] - orien1[1][0];
        b_vector[2] = orien2[2][0] - orien1[2][0];
}
else{
        b_vector[0] = orien1[0][0] - orien2[0][0];
        b_vector[1] = orien1[1][0] - orien2[1][0];
        b_vector[2] = orien1[2][0] - orien2[2][0];
}

// the beta angles are the angles at which the basis vector lies, in
// relation to the object coordinate system
beta1 = atan(b_vector[1]/b_vector[0]);
beta2 = atan(b_vector[2]/(sqrt((b_vector[0]*b_vector[0]) +
         (b_vector[1]*b_vector[1]))));

// calculates the rotation matrix that will transform points from
// image 1 into a basis vector system
CEpipolarDlg::CalcRotMatrix(orien1[0][1], orien1[1][1], orien1[2][1], beta1,
                                        beta2, A);
focal = orien1[3][0];
```

```
// sets a second x coordinate on the epipolar line, half the image
// size from the user shosen point
if( xa < 0)
        xb = xa + (sizeTotal.cx * 0.5);
else
        xb = xa - (sizeTotal.cx * 0.5);

// calculates the y coordinate that corresponds to the x coordinate
// set above
yb = ((A[1][1]*A[2][0]*xb - A[1][1]*A[2][2]*focal - A[2][1]*A[1][0]*xb +
        A[2][1]*A[1][2]*focal)*ya + A[2][2]*focal*A[1][0]*(xb-xa) +
        A[2][0]*focal*A[1][2]*(xa-xb))/(-A[2][1]*A[1][0]*xa +
        A[1][2]*focal*A[2][1] + A[2][0]*xa*A[1][1] - A[2][2]*focal*A[1][1]);

// calculates the parameters of the epipolar line on image 1
m1 = (yb-ya)/(xb-xa);
b1 = (ytrans - ya) - (m1*(xa + xtrans));

// calculates the y coordinate of the user selected point on the
// normalized image
va = -focal*(A[1][0]*xa + A[1][1]*ya - A[1][2]*focal)/(A[2][0]*xa + A[2][1]*ya
                - A[2][2]*focal);

// sets the second y coordinate on the normalized image to be the
//same and the first, the x coordinates are arbitrarily chosen
ua = -0.25*sizeTotal.cx;
ub = 0.25*sizeTotal.cx;
vb = va;

// calculates the rotation matrix that will transform points from
// the basis vector system into the system of image 2
CEpipolarDlg::CalcRotMatrix(orien2[0][1], orien2[1][1], orien2[2][1], beta1,
                                        beta2, A);
focal = orien2[3][0];

// calculates the two points on the image 2 epipolar line from the
// normalized image points
xa = -focal*(A[0][0]*ua + A[1][0]*va - A[2][0]*focal)/(A[0][2]*ua + A[1][2]*va
                - A[2][2]*focal);
ya = -focal*(A[0][1]*ua + A[1][1]*va - A[2][1]*focal)/(A[0][2]*ua + A[1][2]*va
                - A[2][2]*focal);
xb = -focal*(A[0][0]*ub + A[1][0]*vb - A[2][0]*focal)/(A[0][2]*ub + A[1][2]*vb
                - A[2][2]*focal);
yb = -focal*(A[0][1]*ub + A[1][1]*vb - A[2][1]*focal)/(A[0][2]*ub + A[1][2]*vb
                - A[2][2]*focal);
        // puts the calculated coordinates back in the raster image
        // coordinate system
        if((orien2[4][0] == -1)||(orien2[4][1] == -1)){
                xa = xa + 0.5*sizeTotal.cx;
                ya = 0.5*sizeTotal.cy - ya;
                xb = xb + 0.5*sizeTotal.cx;
                yb = 0.5*sizeTotal.cy - yb;
        }
        else{
                xa = xa + orien2[4][0];
                ya = orien2[4][1] - ya;
                xb = xb + orien2[4][0];
                yb = orien2[4][1] - yb;
        }

        // sets the distortion parameters for the next routine
        DistortionFile = DistortionFile2;
        xpp = orien2[4][0];
        ypp = orien2[4][1];

        // calculates the epipolar line in the raster system
        m2 = (yb-ya)/(xb-xa);
        b2 = ya - (m2*xa);

        m_EpipLine = true;

}

/**********************************************************************
  LOCATETARG : PREFORMS CROSS CORELLATION ANALYSIS

                range: pDown - user located target centre on image 1
                       m1 - slope of epipolar line on image 1
                       b1 - intercept of epipolar line on image 1
                       m2 - slope of epipolar line on image 2
                       b2 - intercept of epipolar line on image 2
                       xpp - x coordinate of principal point
                       ypp - y coordinate of principal point
                       DistortionFile - path to the text file containing lens distortion
                                        information
                       target - estimate of target location in the second image
                RETURNS A POINTER TO THE VIEW

  Performs cross correlation analysis, along the given epipolar lines, for the purpose of
  localizing the target in image 2 that corresponds to the user chosen target in image 1.
  This routine uses the bi-linear interpolator to interpolate between grey values, after the
```

search window has been oriented to lie along the epipolar line.

```
*************************************************************************/
CView* CTargetView::LocateTarg(CPoint pDown, double m1, double b1, double m2,
                               double b2, double xpp, double ypp, Cstring
                               DistortionFile, CPoint &target){
        CPoint TargCenter;
        CSize TargSize;
        double **TempImg;
        double **TestImg;
        int counter=0, i=0, j=0, Option;
        double sumx2,sumy2,sumxy;
        double sxy=0,sx2=0,sy2=0,ro=0,largestro=0;
        double averagex, averagey;
        double **TargCanidates;
        double trial[3][3];
        int counter2=0,largestnum=0;
        double boxCentrex, boxCentrey, angle, xi, yi;
        int xint,yint;
        double g1,g2,g3,g4;
        double xtemp,ytemp;
        CDC *pDC;
        CEpipolarDlg pEpip;
        CPoint line, point;
        int numpts,low,high;
        CString TempStr;
        double **distortion;
        double raddist, correct;

        pDC = GetWindowDC();

        // call SizeTarg to get dimensions of template target
        SizeTarg(pDC, pDown, 128,TargCenter,TargSize);

        // set the dimensions of the template target's window
        box_type pBox;
        pBox.left = TargCenter.x - 1.9*TargSize.cx;
        pBox.right = TargCenter.x + 1.9*TargSize.cx;
        if((pBox.right - pBox.left) % 2 == 0)
                pBox.right = pBox.right + 1;

        pBox.top = TargCenter.y - TargSize.cy-3;
        pBox.bottom = TargCenter.y + TargSize.cy+3;
        if((pBox.top - pBox.bottom) % 2 == 0)
                pBox.bottom = pBox.bottom + 1;
```

```
int rectdx, rectdy;

rectdx = pBox.right - pBox.left;
rectdy = pBox.bottom - pBox.top + 1;
int dimensions = rectdx * rectdy;

// set up two image matricies
TempImg = AllocMatrix(rectdy,rectdx);
TestImg = AllocMatrix(rectdy,rectdx);

// determine the centre of window which was created
boxCentrex = (rectdx - 1)/2;
boxCentrey = (rectdy - 1)/2;

// determine the angle that the epipolar line makes
// with the x-axis in image 1
angle = atan(m1);
counter = 0;
averagex = 0;
for(i = 0; i < rectdy ; i++)
{
        for(j = 0; j < rectdx; j++)
        {
                // rotate the template window around it's centre
                // until it lies along the epipolar line
                xi = cos(angle)*(j - boxCentrex) - sin(angle)*(i - boxCentrey);
                yi = sin(angle)*(j - boxCentrex) + cos(angle)*(i - boxCentrey);

                // determine the interger values of the pixel coordinates
                // which are contained in the rotated window
                xint = (int)(xi + TargCenter.x);
                yint = (int)(yi + TargCenter.y);

                // read in the pixels surrounding the rotated pixel
                g1 = RGB_GETBLUE(pDC->GetPixel(xint –
                     GetScrollPosition().x, yint - GetScrollPosition().y));
                g2 = RGB_GETBLUE(pDC->GetPixel(xint+1 –
                     GetScrollPosition().x, yint - GetScrollPosition().y));
                g3 = RGB_GETBLUE(pDC->GetPixel(xint –
                     GetScrollPosition().x, yint+1 - GetScrollPosition().y));
                g4 = RGB_GETBLUE(pDC->GetPixel(xint+1 –
                     GetScrollPosition().x, yint+1 - GetScrollPosition().y));

                // interpolate to find the rotated pixel's grey value
                TempImg[i][j]=Interpolate(xi,yi,g1,g2,g3,g4);
                // used to calculate the average of all grey values on image 1
                averagex = averagex + TempImg[i][j];

                counter++;
        }
}

// the average of the grey value's in the template window
averagex = averagex/(counter+1);
ReleaseDC(pDC);

// call switch window to change the programs focus to
// the second image
CView *pView = SwitchWindow();

if(pView != NULL){
        pDC = pView->GetParentFrame()->GetActiveView()->GetWindowDC();
        CIndmetDoc* pDoc = (CIndmetDoc*) pView->GetParentFrame()->
                                GetActiveDocument();
        ASSERT_VALID(pDoc);
        CDib* pDib = &(pDoc->m_dibTarget);
        CSize sizeTotal;
        sizeTotal = pDib->GetDimensions();

        CString title;
        // stores the previous title of image 2
        title = pView->GetParentFrame()->GetActiveDocument()->GetTitle();
        // sets the title of image 2 to "Scanning..."
        pView->GetParentFrame()->GetActiveDocument()->
        SetTitle("Scanning...");
        //DrawLine(pDC, 0, 0, 50, 50, RGB(255,0,255));

        // decision structure to deterimine which side of the image frame
        // the epipolar line begins on, and to set uo the first search
        // window accordingly
        if(b2 < 0){
                line.x = (int)(0.5*rectdy - b2)/m2;
                line.y = (int)0.5*rectdy;
                Option = 1;
        }
        else if(sizeTotal.cy >= b2 && b2 >= 0){
                line.x = (int)0.5*rectdx;
                line.y = (int)m2*0.5*rectdx + b2;
                Option = 2;
        }
        else {
```

```
        line.x = (int) (sizeTotal.cy - 0.5*rectdy -b2)/m2;
        line.y = (int) sizeTotal.cy - 0.5*rectdy;
        Option = 3;
}

// creation of a matrix in which the top 5 highest correlation
// values and thier points can be stored
TargCanidates = AllocMatrix(6,3);
for(i=0;i<3;i++){
        for(j=0;j<3;j++)
                //Initialization of parameter used for creating target
                //candidate list
                trial[i][j] = 0;
}
for(i=0;i<6;i++){
        for(j=0;j<3;j++)
                //Initialization for the target canidate list
                TargCanidates[i][j] = 0;
}

bool checkline = true;
int scrollposx, scrollposy;
FILE *dis;

// if the principal point information was not entered, then
// set the origion of the coordinate system to the center
// of the image
if((xpp == -1)||(ypp == -1)){
        xpp = 0.5*sizeTotal.cx;
        ypp = 0.5*sizeTotal.cy;
}

// if a lens distortion file was entered, read in the
// lens distortion data
if(DistortionFile != "N/A"){
                dis = fopen(TrimFilePath(DistortionFile),"r");
                if(dis == NULL )
                        TRACE("Error: Unable to open
                         %s",TrimFilePath(DistortionFile));
                else{
                        fscanf(dis,"%s",TempStr);
                        fscanf(dis,"%s",TempStr);
                        fscanf(dis,"%s",TempStr);
                        fscanf(dis,"%d",&numpts);
                        low = numpts - 2;
                        high = numpts - 1;
                fscanf(dis,"%s",TempStr);
                fscanf(dis,"%s",TempStr);
                fscanf(dis,"%s",TempStr);
                fscanf(dis,"%s",TempStr);
                fscanf(dis,"%s",TempStr);
                fscanf(dis,"%s",TempStr);

                distortion = AllocMatrix(numpts,2);
                for(i=0;i<numpts;i++){
                        fscanf(dis,"%lf",&xtemp);
                        fscanf(dis,"%lf",&ytemp);
                        distortion[i][0] = xtemp;
                        distortion[i][1] = ytemp;
                }
        }
        fclose(dis);
}

do{
        if(DistortionFile != "N/A"){
                // set the coordinate system to the principal point
                xtemp = line.x - xpp;
                ytemp = ypp - line.y;

                // compute the points radial distance from the
                // principal point
                raddist = sqrt((xtemp*xtemp)+(ytemp*ytemp));

                // search the list of radial distortions until
                // raddist has been bracketed
                for(i=0;i<numpts-1;i++){
                        if((raddist >= distortion[i][0])&&(raddist <
                           distortion[i+1][0])){
                                low = i;
                                high = i + 1;
                                break;
                        }
                }

                // calculate lens distortion correction, through interpolation
                correct = distortion[low][1] + ((distortion[high][1] –
                           distortion[low][1])/(distortion[high][0] –
                           distortion[low][0]))*(raddist - distortion[low][0]);

                // apply lens distortion correction
        xtemp = xtemp + correct*xtemp/raddist;
        ytemp = ytemp + correct*ytemp/raddist;

        // change coordinate system back to raster grid
        point.x = xtemp + xpp;
        point.y = ypp - ytemp;
}
else{
        point.x = line.x;
        point.y = line.y;
}

// set up search window dimensions, the same
// as template window dimensions
pBox.left = point.x - 0.5*rectdx;
pBox.right= point.x + 0.5*rectdx;
pBox.top = point.y - 0.5*rectdy;
pBox.bottom= point.y + 0.5*rectdy;

// scroll the view so the current point is
// center screen
SetScrollPos(pView,point);

counter = 0;
averagey = 0;

// calculate angle that epipolar line makes
// with x-axis in image 2
angle = atan(m2);

for(i = 0; i < rectdy ; i++){
        for(j = 0; j < rectdx; j++){

                // determine the scroll bar positions
                scrollposx = pView->GetParentFrame()->
                      GetActiveView()->GetScrollPos(SB_HORZ);
                scrollposy = pView->GetParentFrame()->
                      GetActiveView()->GetScrollPos(SB_VERT);

                // rotate the search window so that it lies along the
                // epipolar line
                xi = cos(angle)*(j - 0.5*rectdx) - sin(angle)*(i –
                     0.5*rectdy);
                yi = sin(angle)*(j - 0.5*rectdx) + cos(angle)*(i –
                     0.5*rectdy);
                // determine the interger values of the pixel
                // coordinates which are contained in the rotated
                // window
                xint = (int)(xi + point.x);
                yint = (int)(yi + point.y);

                // read in the pixels surrounding the rotated pixel
                g1 = RGB_GETBLUE(pDC->GetPixel(xint –
                     scrollposx, yint - scrollposy));
                g2 = RGB_GETBLUE(pDC->GetPixel(xint+1 –
                     scrollposx, yint - scrollposy));
                g3 = RGB_GETBLUE(pDC->GetPixel(xint –
                     scrollposx, yint+1 - scrollposy));
                g4 = RGB_GETBLUE(pDC->GetPixel(xint+1 –
                     scrollposx, yint+1 - scrollposy));

                // interpolate to find the rotated pixel's grey value
                TestImg[i][j]=Interpolate(xi,yi,g1,g2,g3,g4);

                // used to calculate the average of all grey values on
                // image 2
                averagey = averagey + TestImg[i][j];

                counter++;
        }
}

// the average of the grey value's in the template window
averagey = averagey/(counter+1);

sumx2 = 0;
sumy2 = 0;
sumxy = 0;
counter = 0;
for(i = 0; i < rectdy; i++){
        for(j = 0; j < rectdx; j++){
                sumxy = sumxy + (TempImg[i][j] –
                        averagex)*(TestImg[i][j] - averagey);
                sumx2 = sumx2 + (TempImg[i][j] –
                        averagex)*(TempImg[i][j] - averagex);
                sumy2 = sumy2 + (TestImg[i][j] –
                        averagey)*(TestImg[i][j] - averagey);
                counter++;
        }
}
```

```
// the calculated covariance value between template
// and search windows
sxy = sumxy/counter;

// the variance of the grey values in the template image
sx2 = sumx2/counter;

// the variance of the grey values in the search image
sy2 = sumy2/counter;

// if either of the variances are close to zero,
// skip this point
if( sx2<=0.000001 || sy2<=0.000001 ){
        line.x = line.x + 1;
        line.y = m2*(line.x) + b2;
        continue;
}

// the correlation value between the two windows
ro = sxy/(sqrt(sx2)*sqrt(sy2));

// the largest correlation value to date is stored here
if(ro > largestro){
        largestro = ro;
        target.x = point.x;
        target.y = point.y;
}

// trial is used to locate local maxima in correlation values, which
// are likely to occur when targets are crossed
trial[0][0] = trial[1][0];
trial[0][1] = trial[1][1];
trial[0][2] = trial[1][2];

trial[1][0] = trial[2][0];
trial[1][1] = trial[2][1];
trial[1][2] = trial[2][2];

trial[2][0] = ro;
trial[2][1] = point.x;
trial[2][2] = point.y;

// store the top 5 correlation values on the
// caniadate list
if(trial[1][0] > trial[2][0] && trial[1][0]>trial[0][0]){
        mbsort(TargCanidates,6);
```

```
                        if(counter2<5 )
                                counter2++;
                        else
                                // i.e. 6 - 1 since the TargCanidates Vector has an
                                // initial index of zero
                                counter2 = 5;

                        TargCanidates[counter2][0] = trial[1][0];
                        TargCanidates[counter2][1] = trial[1][1];
                        TargCanidates[counter2][2] = trial[1][2];
                }

                // decision structure that uses the previously determined
                // side of the frame upon which the epipolar line starts, to
                // set the exit conditions that stop correlation iterations
                // b < 0
                if (Option == 1 && line.y >= (sizeTotal.cy - rectdy)){
                        checkline = false;
                        TRACE("#1 = %d \n",rectdy);
                }
                //  0 <= b <= size
                else if (Option == 2 && line.x >= (sizeTotal.cx - rectdx)){

                        checkline = false;
                        TRACE("#2 = %d \n",(sizeTotal.cx - rectdx));
                }
                // b2 > size
                else if (Option == 3 && line.y <= rectdy){
                        checkline = false;
                        TRACE("#3 = %d \n",(sizeTotal.cy - rectdy));
                }

                // advances the focus one pixel along the epipolar line
                line.x = line.x + 1;
                line.y = m2*(line.x) + b2;

        }while(checkline == true);

        // restores the title on image 2
        pView->GetParentFrame()->GetActiveDocument()->SetTitle(title);
        ReleaseDC(pDC);

        if(DistortionFile != "N/A")
                FreeMatrix(distortion,numpts);
}
FreeMatrix(TempImg,rectdy);
```

```
        FreeMatrix(TestImg,rectdy);
        FreeMatrix(TargCanidates,6);

        TRACE("The Largest Correlation Value Is %lf @ ( %d, %d )
        \n",largestro,target.x,target.y);

        return pView;
}

/*********************************************************************
MATCHTARG : PREFORMS TEMPLATE LEAST SQUARES TARGET MATCHING

                range: **TempImg - pointer to the array containing
                                   the template image grey values
                       row - number of rows in TempImg
                       col - number of columns in TempImg
                      *pView - pointer to the view containing the
                               search window
                       target - estimate of the target center in
                                pView's window


 Performs template least squares target matching on a given target in a given view, given
 a template for comparison. This routine calls on the services of an interpolator sub-
 routine, that uses bi-linear interpolation to resample the search window. It also uses a
 separate least squares module to perform the adjustment. This routine also uses initial
 assumptions of Tx = 0, Ty = 0, Sx = 1, Sy = 1, Rx = 0 and Ry = 0 in it's operation. It
 exits when Tx < 0.01, Ty < 0.01, Sx < 0.0001, Sy < 0.0001, Rx < 0.0001 and
 Ry < 0.0001
*********************************************************************/
void CTargetView::MatchTarg(double **TempImg,int row,int col,CView *pView,
CPoint target){
        double **TestImg;
        int     scrollposx,     scrollposy, message=0;
        int i,j,counter,iter, check=0, r2, c2;
        double Tx,Ty,Sx,Sy,Rx,Ry;
        double g1,g2,g3,g4,g5;
        int xi,yi;
        double xc,yc,rms;
        double x1,x2,x3,x4,x5;
        double y1,y2,y3,y4,y5;
        double **a;
        double *l;
        double *x;
        double sumsq;
        CDC *pDC;
```

```
pDC = pView->GetParentFrame()->GetActiveView()->GetWindowDC();

// scroll the view so that the highest correlated
// target is placed in the center
SetScrollPos(pView,target);

CIndmetDoc* pDoc = (CIndmetDoc*) pView->GetParentFrame()->
                        GetActiveDocument();
ASSERT_VALID(pDoc);
CDib* pDib = &(pDoc->m_dibTarget);
CSize sizeTotal;
sizeTotal = pDib->GetDimensions();

//setting the dimensions of the search window
box_type pBox;
pBox.left = target.x - 0.5*col;
pBox.right= target.x + 0.5*col;

pBox.top = target.y - 0.5*row;
pBox.bottom= target.y + 0.5*row;

TestImg = AllocMatrix(row,col);
a = AllocMatrix((row-2)*(col-2),6);
l = (double *) malloc((row-2)*(col-2)*sizeof(double));
x = (double *) malloc(6*sizeof(double));

// storing grey values from search window
for(i = 0; i < row; i++){
        for(j = 0; j < col; j++){
                scrollposx = pView->GetParentFrame()->GetActiveView()->
                                GetScrollPos(SB_HORZ);
                scrollposy = pView->GetParentFrame()->GetActiveView()->
                                GetScrollPos(SB_VERT);

                TestImg[i][j] = RGB_GETBLUE(pDC->GetPixel(pBox.left + j –
                                scrollposx, pBox.top + i - scrollposy));
        }
}

// Inital assumptions for adjustment
Tx = 0;
Ty = 0;
Sx = 1;
Sy = 1;
Rx = 0;
Ry = 0;
```

```
// parameters to transform the coordinate system to the
// search windows centre
r2 = (row - 1)/2;
c2 = (col - 1)/2;

iter = 0;
check = 0;
do{
        counter = 0;
        sumsq = 0;
        for(i = 1; i < (row - 1); i++){
                for(j = 1; j < (col - 1); j++){
                        // interpolate grey value for point under observation
                        x1 = Tx + Sx * (j-c2) + Rx * (i-r2);
                        y1 = Ty + Ry * (j-c2) + Sy * (i-r2);
                        x1 = x1 + c2;
                        y1 = y1 + r2;
                        xi = (int)(x1);
                        yi = (int)(y1);
                        if((xi>0 && xi<(col-1)) && (yi>0 && yi<(row-1)))
                                g1=Interpolate(x1,y1,TestImg[yi][xi],
                                    TestImg[yi][xi+1], TestImg[yi+1][xi],
                                    TestImg[yi+1][xi+1]);
                        else
                                continue;

                        // interpolate grey value for point above
                        // the point under observation
                        x2 = Tx + Sx * (j-1-c2) + Rx * (i-r2);
                        y2 = Ty + Ry * (j-1-c2) + Sy * (i-r2);
                        x2 = x2 + c2;
                        y2 = y2 + r2;
                        xi = (int)(x2);
                        yi = (int)(y2);
                        if((xi>0 && xi<(col-1)) && (yi>0 && yi<(row-1)))
                                g2=Interpolate(x2,y2,TestImg[yi][xi],
                                TestImg[yi][xi+1], TestImg[yi+1][xi],
                                TestImg[yi+1][xi+1]);
                        else
                                continue;

                        // interpolate grey value for point to the left
                        // of the point under observation
                        x3 = Tx + Sx * (j-c2) + Rx * (i-1-r2);
                        y3 = Ty + Ry * (j-c2) + Sy * (i-1-r2);
```

```
x3 = x3 + c2;
y3 = y3 + r2;
xi = (int)(x3);
yi = (int)(y3);
if((xi>0 && xi<(col-1)) && (yi>0 && yi<(row-1)))
        g3=Interpolate(x3,y3,TestImg[yi][xi],
            TestImg[yi][xi+1],TestImg[yi+1][xi],
            TestImg[yi+1][xi+1]);
else
        continue;

// interpolate grey value for point below
// the point under observation
x4 = Tx + Sx * (j+1-c2) + Rx * (i-r2);
y4 = Ty + Ry * (j+1-c2) + Sy * (i-r2);
x4 = x4 + c2;
y4 = y4 + r2;
xi = (int)(x4);
yi = (int)(y4);
if((xi>0 && xi<(col-1)) && (yi>0 && yi<(row-1)))
        g4=Interpolate(x4,y4,TestImg[yi][xi],
            TestImg[yi][xi+1],TestImg[yi+1][xi],
            TestImg[yi+1][xi+1]);
else
        continue;

// interpolate grey value for point to the right
// of the point under observation
x5 = Tx + Sx * (j-c2) + Rx * (i+1-r2);
y5 = Ty + Ry * (j-c2) + Sy * (i+1-r2);
x5 = x5 + c2;
y5 = y5 + r2;
xi = (int)(x5);
yi = (int)(y5);
if((xi>0 && xi<(col-1)) && (yi>0 && yi<(row-1)))
        g5=Interpolate(x5,y5,TestImg[yi][xi],
            TestImg[yi][xi+1],TestImg[yi+1][xi],
            TestImg[yi+1][xi+1]);
else
        continue;

// terms of the design matrix, using point derivatives
// to solve the equations
a[counter][0] = (g4 - g2)/2;
a[counter][1] = (g5 - g3)/2;
a[counter][2] = ((g4 - g2)/2)*(x1-c2);
```

```
                        a[counter][3] = ((g5 - g3)/2)*(y1-r2);
                        a[counter][4] = ((g4 - g2)/2)*(y1-r2);
                        a[counter][5] = ((g5 - g3)/2)*(x1-c2);

                        // error vector
                        l[counter] = TempImg[i][j] - g1;
                        // sum of squares of error vector, used for
                        // RMS determination
                        sumsq = sumsq + l[counter]*l[counter];

                        counter++;
                }
        }
        if(counter > 0)
                message=template_matching(a, counter, 6, l, x);
        else{
                AfxMessageBox("Template Matching Failure: Interpolation
                                Failed");
                return;
        }

        if(message != 0){
                AfxMessageBox("Template Matching Failed: Unable To Compute
                                Matrix Inverse");
                return;
        }

        // adjustment of initial assumptions
        Tx = Tx + x[0];
        Ty = Ty + x[1];
        Sx = Sx + x[2];
        Sy = Sy + x[3];
        Rx = Rx + x[4];
        Ry = Ry + x[5];

        // check to determine if the adjustment has converged
        if(fabs(x[0])<0.01 && fabs(x[1])<0.01 && fabs(x[2])<0.0001 &&
          fabs(x[3])<0.0001 && fabs(x[4])<0.0001 && fabs(x[5])<0.0001)
                check = 1;
        else{
                // if no convergence after 500 iterations, exit
                if(iter >= 500)
                        check = 1;
                iter++;
        }
}while(check != 1);
```

```
if(iter >= 500)
        AfxMessageBox("Template Matching Failure: Failed To Converge");
else{
        // set dialog box parameters to final solution
        xc = Tx + target.x;
        yc = Ty + target.y;
        rms = sqrt(sumsq)/counter;

        CInsertDialog pDlg;

        // set up the dialog
        CString pTemp;
        pTemp.Format("Point %d", pDoc->GetCount());
        pDlg.m_sLabel = pTemp;

        pTemp.Format("( %.3f , %.3f )", xc, yc);
        pDlg.m_sPoint = pTemp;

        pTemp.Format("( %.3f , %.3f )", rms, rms);
        pDlg.m_cRms = pTemp;
        pDlg.m_sType = "Template Least Squares Matching";

        if(pDlg.DoModal() == IDOK){
                pDoc->Insert(pDlg.m_sLabel, GetDocument()->m_sImage, xc, yc,
                                rms, pDlg.m_sType);

                if(pView->GetParentFrame()->
                   IsKindOf(RUNTIME_CLASS(CBothFrame))){
                        CBothFrame* pFrame = (CBothFrame*)pView->
                                                GetParentFrame();
                        if(pFrame == NULL){
                             TRACE("no frame in
                                       CTargetView::MatchTarg()\n");

                            AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
                            return;
                        }
                        CPointView* pPoint = (CPointView*)pFrame->
                                                m_wndSplitter.GetPane(0,1);

                        // remove origin element
                        if(pPoint->GetListCtrl().GetItemText(0,6) == "0")
                                pPoint->GetListCtrl().DeleteAllItems();

                        CString pTemp1;
```

```
                                        pPoint->GetListCtrl().InsertItem(pPoint->
                                        GetListCtrl().GetItemCount(), pDlg.m_sLabel);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1,1, pDoc->m_sImage);
                                        pTemp1.Format("%.3f", xc);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1,2, pTemp1);
                                        pTemp1.Format("%.3f", yc);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1, 3, pTemp1);
                                        pTemp1.Format("%.3f", rms);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1, 4, pTemp1);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1, 5, pTemp1);
                                        pPoint->GetListCtrl().SetItemText(pPoint->
                                        GetListCtrl().GetItemCount()-1, 6, pDlg.m_sType);
                                        pPoint->GetListCtrl().Update(0);
                                }
                        }
                }

        ReleaseDC(pDC);
        FreeMatrix(TestImg,row);
        FreeMatrix(a,(row-2)*(col-2));
        free(l);
        free(x);
}

/*****************************************************************
 TEMPLATE_MATCHING : PERFORMS A LEAST SQAURE ADJUSTMENT

        range:  a - the design matrix
                rowsa - the number of rows in a
                colsa - the number of columns in a
                b - the misclosure vector
                x - the vector of estimates

        RETURN: 0 - success
                1 - not enough memory
                2 - cholesky failed

 This routine preforms a least squares matching algorithm designed to solve the matrix
 equation Ax = B.
*****************************************************************/
int template_matching(double **a,int rowsa, int colsa, double *b, double *x){
```

```
double **at;
double **ata;
double **ata2;
double **I;
double *atb;
int i,j,message;

if( (at=alloc_matrix(colsa,rowsa)) == NULL ){
        strcpy( ellstr,"Error: Not enough memory");
        return(1);
}
if( (ata=alloc_matrix(colsa,colsa)) == NULL ){
        strcpy( ellstr,"Error: Not enough memory");
        return(1);
}

if( (atb = (double *) malloc(colsa*sizeof(double))) == NULL ){
        strcpy( ellstr,"Error: Not enough memory");
        return(1);
}

for(i=0;i<colsa;i++){
        for(j=0;j<rowsa;j++){
                // create the transpose of matrix a
                at[i][j] = a[j][i];
        }
}

// multiplies the transpose of the design matrix by
// the design matrix
mult_matrices( at, colsa, rowsa,a, rowsa, colsa, ata);

// multiplies the transpose of the design matrix by
// the misclosure vector
matvect_mult( at, colsa, rowsa, b, rowsa, atb );

// calculates the choleski matrix inverse of the matrix ata
message = dcholeski( ata, colsa );
if(message != 0){
        strcpy( ellstr,"Error: Cholesky Inverse Failed");
        return(2);
}

// multiplies the inverse of ata with the matrix atb
// to result in a the vector of estimates
matvect_mult( ata, colsa, colsa,atb, colsa, x );
```

```
        free_matrix( at, colsa );
        free_matrix( ata, colsa );
        free( atb );
        return(0);
}
```

Control Functions

void CTargetView::OnLButtonDown(UINT nFlags, CPoint point) - This function is used to define the actions taken by the program when the user clicks on a point within a document. This is true for all subpixel target center locators whether they are automatic or not. Only the section that deals specifically with automatic target matching has been commented.

void CIndmetApp::OnFileOpenTargetmatching() - This function is used to open and set up input for automatic target matching to begin. It opens two document images, calls the orientation input dialog box and passes this information into the file epipolardlg.cpp

void CEpipolarDlg::CreateEpipolarInput(double orien[5][2], CSize Size, CString DistortionFile, int ImageNo) - This Function is designed to operate in conjuction with CalcEpipolarLine(). It collects the orientation matrices and the image sizes. It then squirts them to CreateEpipolarLine() through use of the global variables sizeTotal, orien1[4][2] and orien2[4][2].

Calculative Functions

void CEpipolarDlg::CalcRotMatrix(double Omega, double Phi, double Kappa, double beta1, double beta2, double A[3][3]) - This function calculates the rotation matrix that will transform image coordinates into a Base Vector Parallel System. It operates in two parts, first setting up a rotation matrix to rotate object system coordinates into a base vector parallel system. It then sets up a standard rotation matrix that converts the image system to the object system. These two matrices are then multiplied and the result is output.

int dcholeski(double **a, int nsize) - This routines finds the inverse of a symmetric matrix by Choleski algorithm, this is fast but it requires 2 extra normal matricies to be allocated! If the matrix is less than 3x3, simpler methods than choleski are used.

void CEpipolarDlg::CorrectLensDistortion(CString File, double &x, double &y, int ApplyCorrection) - This function corrects or adds lens distortion to the point entered. If ApplyCorrection is 0, then the distortion is added to the point, if it is 1 then the distortion is corrected

void matvect_mult(double **a_mat, int arow, int acol, double *b_vect, int brow, double *c_vect) - This routine multiplies a matrix by a vector and returns the results into the result vector c_vect.

void mult_matrices(double **a_mat, int arow, int acol, double **b_mat, int brow, int bcol, double **c_mat) - This routine multiplies two matricies of appropriate size and returns a pointer to a third matrix which contains the answer.

double CTargetView::Interpolate(double x, double y, double g1, double g2, double g3, double g4) - Uses bi-linear interpolation to determine grey values for non-integer coordinates. Is used to perform image resampling for the MatchTarg sub-routine.

int CTargetView::SizeTarg(CDC *pDC, CPoint pDown,int cutoffbv, CPoint &TargCenter, CSize &TargSize) - Used to estimate the target center and the size of the target, for which the entered coordinates correspond. To make these estimates, the routine examines and counts pixels, which extend out in the four cardinal directions, until a contrast edge is encountered. The routine averages these results and continues until the average center changes by 1 pixel or less.

Graphical And User Interface Functions

void CTargetView::DrawLine(CDC *DC, long left, long top, long right, long bottom, COLORREF color) - Draws a line across the document image in any 256 color that is desired. It uses pixel coordinates of where the line is to start and end. This was written incase the epipolar line needed to be drawn on the image.

void CTargetView::SetScrollPos(CPoint pDown) - This is an overloaded function that repositions the scrolling window so that the x and y coordinates entered will appear in the centre of the window. This version automatically scrolls the first, default, window that was opened.

void COrientationDlg::OnFind() - This function opens the standard open file dialog box so that the user can search for the correct lens distortion file for the open image.

void COrientationDlg::OnLoad() - This function opens and gathers data from preformed data orientation text files. It then prints this data into the orientation dialog box's appropriate text boxes. The files were created using the OnSave function.

void COrientationDlg::OnOK() - This function gathers data entered into the orientation dialog box's text boxes. It then uses the C++ functions necessary to convert that data from string data into real numbers.

void COrientationDlg::OnSave() - This function saves data entered into the orientation dialog's text boxes. It reads the information that is entered and saves it in a user defined text file.

void CTargetView::SetScrollPos(CView *pView, CPoint pDown) - This is an overloaded function that repositions the scrolling window so that the x and y coordinates entered will appear in the centre of the window. This version allows the user to indicate which window is to be scrolled when multiple windows exist.

CView* CTargetView::SwitchWindow() - Redirects the program focus to the next document window in the z-order list.

Miscellaneous Functions

double** CTargetView::AllocMatrix(int rows, int cols) - AllocMatrix first allocates an array of pointers (one for each row), then for each row pointer it allocates an array of double values (cols).

void CTargetView::FreeMatrix(double **m, int rows) - Frees a matrix that was allocated with AllocMatrix

void CTargetView::mbsort(double **list, int numpts) - Performs a modified bubble sort on an integer list of numbers, from smallest to largest. At the same time, it rearranges a vector of coordinates associated with the list

CString CTargetView::TrimFilePath(CString FilePath) - Removes the associated c:\...\...\ from the file name so that fopen can be used to open the file, given that the file path has already been stored

Appendix B

Image Information For Digital Images Used In Program Testing

All the information in this section was provided by the Applanix Corporation and was used in the testing of the automatic target matching program.

Table 1: Camera Calibrated Parameters of DSS Serial 0005 (Applanix Corp., 2002.)

Parameter	Value	Accuracy
f (mm)	55.048	0.009 mm
Pixel Non-squareness	1.0	0.0000001
Xpp (pixels) +	2025.34	0.5 pixel
Ypp (pixels) +	2032.94	0.5 pixel
Xpp^C (mm) ++	-0.127	0.0045
Ypp^C (mm) ++	0.118	0.0045

+ Xpp and Ypp are measured from image upper left corner, (image size 4079 x 4092) see Figure 1
++ Xpp^C and Ypp^C are measured from image centre (pixel size = 9 microns) see Figure 1

Table 2: Radial Lens Distortion Table of DSS Serial 0005 (Applanix Corp., 2002.)

Radial Distance (mm)	Radial Distortion (Pixel)	Radial Distortion (microns)
2	-0.02	-0.18
4	-0.16	-1.41
6	-0.53	-4.73
8	-1.24	-11.18
10	-2.41	-21.73
12	-4.15	-37.31
14	-6.53	-58.75
16	-9.64	-86.72
18	-13.53	-121.74
20	-18.22	-164.01
22	-23.71	-213.40
24	-29.92	-269.29
26	-36.72	-330.46

Table 3: Camera Station Coordinates And Orientations (Applanix Corp., 2002.)

PHOTO ID	X (m)	Y (m)	Z (m)	OMEGA (deg)	PHI (deg)	KAPPA (deg)
A_H_0	630233.168	4857734.161	158.808	96.50644	24.70634	181.96748
C_H_0	630229.855	4857737.831	158.798	107.17451	68.68136	195.90932
C_L_0	630229.829	4857737.813	157.767	123.80958	64.12096	211.13974

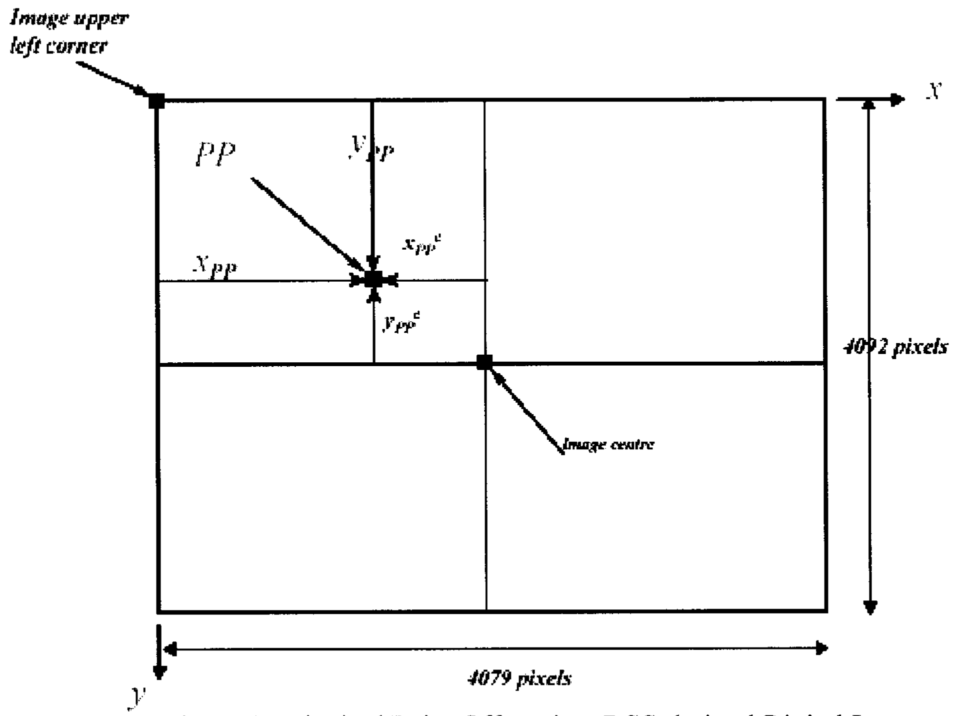


Figure 1: Principal Point Offsets in a DSS-derived Digital Image (Applanix Corp., 2002.)

Table 4: Ground Control Coordinates For Image Targets (Applanix Corp., 2002.)

Ground Control Point ID	X (m)	Y (m)	Z (m)
101	630234.66790	4857739.16881	158.88692
102	630234.66827	4857739.16887	158.07409
103	630234.66608	4857739.17058	157.06540
104	630234.77126	4857738.86436	158.86917
105	630234.77501	4857738.86495	158.08065
106	630234.77658	4857738.86617	157.05676
107	630235.20899	4857737.57277	158.68335
108	630235.21078	4857737.57503	158.05442
109	630235.21578	4857737.57391	157.05445
110	630235.28484	4857737.34110	158.88386
111	630235.28739	4857737.34196	158.13346
112	630235.29512	4857737.34252	157.06467
113	630235.40356	4857739.41705	158.89790
114	630235.40907	4857739.41768	158.13425
115	630235.41924	4857739.42316	157.05943
116	630235.85972	4857738.72338	158.88702
117	630235.86093	4857738.72401	157.96054
118	630235.86599	4857738.72621	157.05982
119	630235.96745	4857738.41292	158.88032
120	630235.96975	4857738.41729	158.07638
121	630235.97516	4857738.42371	157.06087
122	630236.02869	4857737.59014	158.87351
123	630236.02760	4857737.59354	158.15902

Ground Control Point ID	X (m)	Y (m)	Z (m)
124	630236.03432	4857737.59467	157.05591
125	630236.15587	4857739.71139	158.72338
126	630236.16073	4857739.71335	158.02745
127	630236.16325	4857739.71555	157.25418
128	630236.78742	4857737.84756	158.87833
129	630236.78833	4857737.84951	158.15513
130	630236.79419	4857737.85028	157.06192
131	630236.51928	4857739.83272	158.89295
132	630236.52461	4857739.83528	158.12270
133	630236.51957	4857739.83690	157.07364
134	630236.74516	4857739.19503	158.08343
135	630236.74530	4857739.19612	157.06305
136	630236.93918	4857738.61756	158.06396
137	630236.93857	4857738.62144	157.06811
138	630237.15191	4857737.96766	158.90208
139	630237.15532	4857737.97132	158.08798
140	630237.15943	4857737.97417	157.06530
201	630234.64470	4857739.17176	161.89525
202	630234.65131	4857739.17332	161.07718
203	630234.66436	4857739.17842	160.08213
204	630234.74223	4857738.86568	161.89277
205	630234.75782	4857738.87073	160.99254
206	630234.76684	4857738.87377	160.07486
207	630235.18382	4857737.57239	161.68838
208	630235.19194	4857737.57175	161.11311
209	630235.19812	4857737.57840	160.08711
210	630235.26070	4857737.34345	161.89919
211	630235.26490	4857737.34679	161.18486
212	630235.27536	4857737.34976	160.06673
213	630235.38386	4857739.41802	161.90930
214	630235.39197	4857739.42175	161.14420
215	630235.40182	4857739.42717	160.07047
216	630235.84059	4857738.72386	161.89907
217	630235.84573	4857738.72434	161.00023
218	630235.85411	4857738.72894	160.09124
219	630235.93876	4857738.41978	161.89737
220	630235.94641	4857738.42314	161.10855
221	630235.95246	4857738.42600	160.07379
222	630236.00293	4857737.59288	161.88833
223	630236.00552	4857737.59377	161.18546
224	630236.01632	4857737.59838	160.07438
225	630236.13705	4857739.71219	161.78275
226	630236.14659	4857739.71533	161.13412
227	630236.15275	4857739.71844	160.24826
228	630236.76533	4857737.85014	161.90860
229	630236.77025	4857737.85238	161.19397

Ground Control Point ID	X (m)	Y (m)	Z (m)
230	630236.77815	4857737.85325	160.07269
231	630236.49833	4857739.83493	161.89397
232	630236.51269	4857739.83821	161.10314
233	630236.51690	4857739.84127	160.07814
234	630236.72684	4857739.19369	161.09137
235	630236.73303	4857739.19470	160.07833
236	630236.91884	4857738.61967	161.07259
237	630236.92482	4857738.62121	160.07662
238	630237.12676	4857737.97111	161.90729
239	630237.13383	4857737.97416	161.18361
240	630237.13855	4857737.97458	160.07676
301	630233.70212	4857742.00989	161.89572
302	630233.70986	4857742.01195	161.13321
303	630233.71771	4857742.01693	160.06829
304	630233.80532	4857741.70499	161.88548
305	630233.81361	4857741.71083	161.12355
306	630233.82105	4857741.71090	160.07001
307	630234.21806	4857740.48466	161.69813
308	630234.22425	4857740.48744	161.05416
309	630234.25318	4857740.41445	160.06872
310	630234.31646	4857740.17936	161.90098
311	630234.32351	4857740.18223	161.17585
312	630234.33129	4857740.18469	160.06722
313	630234.44162	4857742.26248	161.89776
314	630234.44977	4857742.26425	161.06755
315	630234.45656	4857742.26631	160.07334
316	630234.89271	4857741.56709	161.87977
317	630234.90203	4857741.56778	161.02673
318	630234.90503	4857741.56971	160.07018
319	630234.99548	4857741.26268	161.87063
320	630235.00105	4857741.26465	161.14063
321	630235.01073	4857741.26629	160.07887
322	630235.05870	4857740.43456	161.90446
323	630235.06362	4857740.43401	161.16818
324	630235.06805	4857740.43884	160.07398
325	630235.21759	4857742.48661	161.90071
326	630235.22518	4857742.48795	161.16304
327	630235.23052	4857742.48870	160.37156
328	630235.82033	4857740.69626	161.90567
329	630235.82303	4857740.69569	161.14849
330	630235.83203	4857740.69952	160.08184
331	630235.57819	4857742.60993	161.90389
332	630235.58560	4857742.61160	161.10650
333	630235.58940	4857742.61202	160.08550
334	630235.80509	4857741.96667	161.14465
335	630235.81185	4857741.96999	160.08529

Ground Control Point ID	X (m)	Y (m)	Z (m)
336	630235.97431	4857741.46464	161.15681
337	630235.98018	4857741.46492	160.08948
338	630236.17612	4857740.82157	161.91479
339	630236.18676	4857740.82045	161.02586
340	630236.19114	4857740.82311	160.10289
401	630233.71433	4857742.00996	158.87809
402	630233.71760	4857742.01005	158.16652
403	630233.72261	4857742.02066	157.04542
404	630233.82181	4857741.71139	158.87658
405	630233.82309	4857741.71403	158.13036
406	630233.82872	4857741.71767	157.06029
407	630234.23147	4857740.48840	158.70689
408	630234.23595	4857740.49057	158.05180
409	630234.24006	4857740.49872	157.06546
410	630234.33111	4857740.18300	158.89266
411	630234.33594	4857740.18756	158.14005
412	630234.34072	4857740.19252	157.05576
413	630234.44958	4857742.26277	158.88377
414	630234.45245	4857742.26080	158.10988
415	630234.46742	4857742.27068	157.06245
416	630234.90782	4857741.57042	158.88716
417	630234.91244	4857741.57293	158.17186
418	630234.91923	4857741.57611	157.07008
419	630235.01260	4857741.26546	158.88584
420	630235.01466	4857741.26665	158.12668
421	630235.02007	4857741.27063	157.06432
422	630235.07425	4857740.43794	158.89478
423	630235.07479	4857740.43904	158.19290
424	630235.08130	4857740.44201	157.06721
425	630235.23216	4857742.48887	158.87924
426	630235.23359	4857742.48846	158.05344
427	630235.24313	4857742.49305	157.07090
428	630235.82784	4857740.69333	158.89145
429	630235.84197	4857740.69966	158.12004
430	630235.84369	4857740.69903	157.07174
431	630235.59277	4857742.61233	158.87712
432	630235.59665	4857742.61097	158.07089
433	630235.60454	4857742.61549	157.07897
434	630235.81481	4857741.96546	158.07124
435	630235.79853	4857742.04412	157.06586
436	630235.98683	4857741.46685	158.00361
437	630235.99089	4857741.46784	157.07407
438	630236.18915	4857740.81891	158.89645
439	630236.19466	4857740.81984	158.06086
440	630236.20173	4857740.81910	157.07618

Appendix C

Digital Images Used In Program Testing

The Images presented in this section are low-resolution images of the actual images used in program testing. The high-resolution images can be found on the CD accompanying this thesis.

Figure 1: Image A_H_O (Applanix Corp., 2002.)

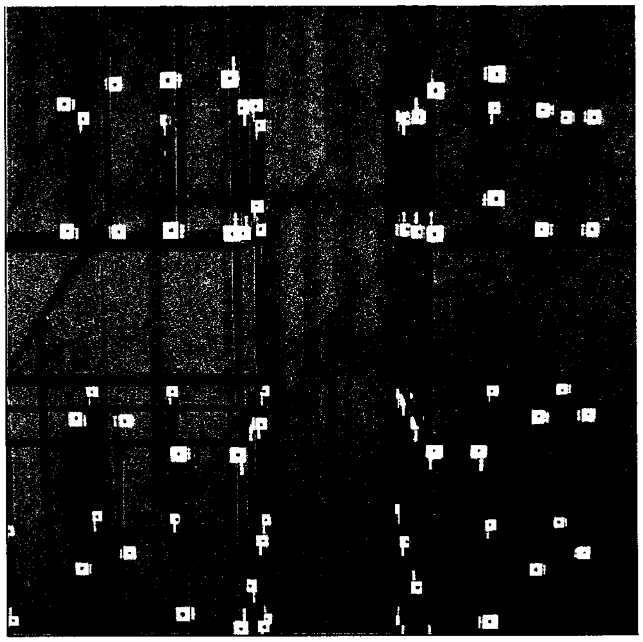


Figure 2: Image C_H_O (Applanix Corp., 2002.)

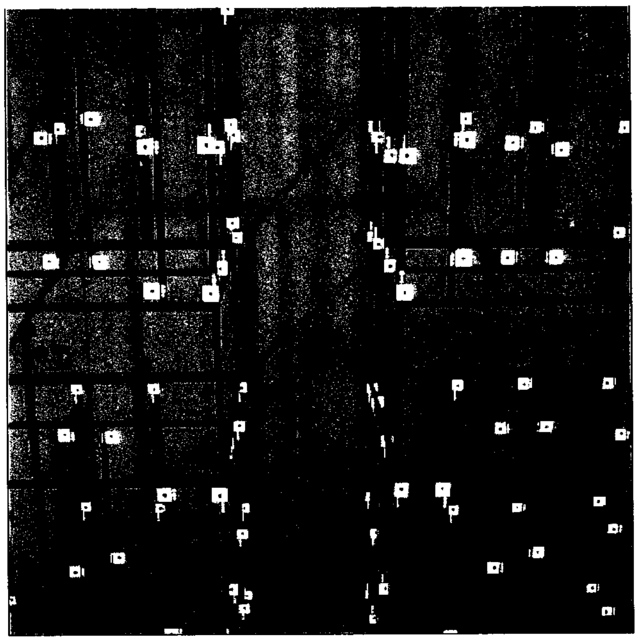


Figure 3: Image C_L_O (Applanix Corp., 2002.)

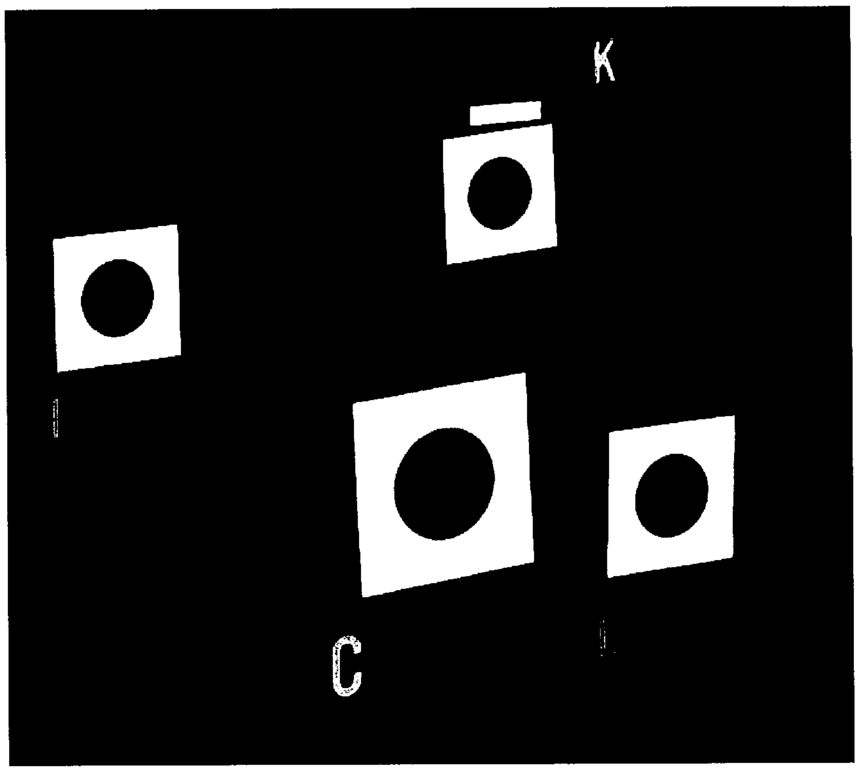


Figure 4: Image Used To Test Least Squares Matching Subroutine (Cosandier and Chapman, 1992.)