EXTRACTION OF SURFACE TEXTURE DATA FROM LOW QUALITY PHOTOGRAPHS TO AID THE CONSTRUCTION OF VIRTUAL REALITY MODELS OF ARCHAEOLOGICAL SITES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF SCIENCE
AT THE UNIVERSITY OF CAPE TOWN
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

By
John G. Williams

June 2001

Supervised by
Edwin H. Blake

Abstract

A tool has been designed and implemented to use information extracted from photographs captured using uncalibrated cameras (so-called casual photographs) to fill the occlusions which occur in three-dimensional models of photogrammetrically captured sites. Capturing the geometry of archaeological sites by photogrammetric means is relatively expensive and, because of the layouts typical of such sites, usually results in a degree of occlusion. Occlusions are filled by extracting texture and calculating hidden geometry from casual photographs with the support of three-dimensional geometric data gleaned from the photogrammetric survey.

The essential philosophy underlying the tool is to segment each occlusion into surfaces which may be approximated using curves and then use known geometry in the region of the occlusion to calculate the most probable locations of the junctions of such surface segments. The tool is primarily a combination of existing techniques for pre-filtering and calibrating the casual photograph, boundary detection and ultimately texture adjustment. The technique implemented for calculating the locations of occluded corners using minimisation of least square errors is new.

The tool has been applied to occlusions of the various configurations that are expected to be typical of archaeological sites and has been found to deal well with such features and to provide accurate patches from typical data sets. It is also shown that the three-dimensional geometric model is clearly improved by the filling-in of the occlusion.

Acknowledgements

I would like to thank Prof. Edwin Blake, my supervisor, for providing direction and ideas when necessary and for the thankless task of working though numerous drafts of this dissertation. I would also like to thank Prof. Heinz Ruther for suggesting this topic of work and for helpful input along the way; and Simon Hull who provided the datum virtual reality models.
All registered trademarks mentioned in this document are property of their respective owners.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

Due to the widespread availability of computers capable of navigating three-dimensional graphical models there has been a proliferation of three-dimensional textured models of architectural subjects and in particular archaeological sites ([3], [35], [57], [69]) amongst many others. The standardisation of the language most commonly used for constructing virtual reality models (VRML[61]) has accelerated this trend.

In order to construct three-dimensional textured models sites must be surveyed using various techniques. A commonly used technique is photogrammetry, which relies on the matching of corresponding features in a pair of images to calculate the geometry of the scene.

The process of capturing accurate geometric and texture data of large archaeological sites using photogrammetry is expensive and time consuming. This is because the equipment used to capture and process data is expensive and must be set-up with great care in order to ensure accuracy. A further difficulty is introduced by the fact that archaeological sites include features such as columns, niches, irregularly shaped walls and other irregular surfaces which interfere with the image capturing process. A consequence of these factors is that photogrammetric surveys of archaeological sites always include regions which are occluded[1] because of the trade-off between the coverage (that is the number of image capture points and directions) and the cost of that coverage.

Effectively, an occlusion is a part of the site about which only the location of its edge is known. Three-dimensional textured models, such as virtual reality models, constructed using data sets which include occlusions thus have gaps scattered around within them. These gaps reduce the visual and immersive impact of such models.

On the other hand it is relatively inexpensive to collect a far more comprehensive set of ordinary photographs of a site. Conceptually it appears reasonable that information from such images could be used to fill the gaps in the three-dimensional site model.

This work has two aims: firstly to *determine whether it is possible* to fill the occlusions in the three-dimensional model of a site with data extracted from such, so-called, casual photographs.

The second aim of this work, in addition to establishing the *feasibility* of using casual photographs to fill occlusions, is to provide a *tool* to use for filling occlusions. This tool could be used by model builders when presented with a scenario of incomplete model supplemented with casual photographs.

A further application of any such tool would be for constructing models of now partially damaged sites for which photographs are in existence, for example buildings damaged during wars during the past 150 years. Essentially the damaged or missing parts of the site are analogous to occlusions.

[1] it will be shown (see §5.3.4) that, because calculation of geometry by photogrammetric methods is reliant on the visibility of corresponding features in a pair of images captured from different locations, that occlusions in the three-dimensional model have an extent equivalent to the *union* of the occlusions from both images.

1.1 Outline of main arguments

It is proposed that the feasibility (of filling occlusions from casual photographs) be established by demonstration. That is, by building (or finding) a tool and using it to fill occlusions having a set of characteristics likely to be encountered in the field. Thus the first objective of this work (determining feasibility) is a result of the second (providing a tool).

Any tool to be used for filling occlusions in a three-dimensional model should, besides having the ability to extract image texture data, be capable of extracting geometric data from images. Several techniques and tools currently exist to extract three-dimensional geometric data from ordinary photographs but they generally rely on assumptions of regularity present in architecture such as flat surfaces ([20], [25]), orthogonal planes ([6], [7], [63]) and other features [73] which are not applicable to typical archaeological scenes. The tool constructed is novel in that it is suitable for irregular shapes typically present in archaeological scene rather than being based on assumptions of regularity.

Furthermore, it is very important to note that it is possible to be misled by inaccurate but convincing three-dimensional models (discussed in detail in §2.1.3). Therefore, any tool built for scientific research and information dissemination should make the *greatest reliance on the data available* and the *smallest reliance on assumptions regarding* site geometry.

In order to build such a tool it is expected to be necessary, amongst other things, to provide functionality to: calibrate the casual photograph, segment the casual photograph along discontinuities in surfaces; warp patches of surface texture from their shape in the casual photograph to their shape in the model; provide some means of estimating the geometry within the occlusions; and allow the adjustment of patch texture data to match the surrounding photogrammetrically captured texture better. The body of this dissertation details how this required functionality has been provided; including discussion on the refinement and detailed specification of requirements, solution design, implementation and evaluation.

1.2 Important novel aspects

The exploitation of *calibration* of the casual images is an important and novel feature of the tool implemented because it leverages the fact that casual photographs will include many features which have been accurately surveyed. Furthermore, the calibration parameters extracted above are used to *project the boundaries of the occluded area* (in the geometry) onto the casual photograph. This boundary forms the point of integration between the photogrammetrically gathered texture and geometry and that obtained from the casual photograph. The concept of projecting the boundaries of occlusions onto images as a means of segmentation of the image and providing the boundary for integration of texture types (casual photograph with photogrammetric) is also novel aspect of this work.

The *least mean squares algorithm for the computation of the hidden geometry* is believed to be another novel aspect of this work. Briefly, the algorithm computes the positions of hidden corners by performing a least squares error fit for the three-dimensional location of each edge pixel under the constraints of i) the surfaces being approximated by a polynomial and ii) the location in the scene of the said pixel lying on a three-dimensional line (pixel-ray) consistent with its position on the casual photograph. The algorithm implemented makes the greatest use of information available (namely surrounding geometry, edge shape in the casual photograph and user segmentation of the image) and the least use of assumptions (the surface is assumed to be bi-cubic, which is in itself not a very constraining). Furthermore, being designed for regions which include edges and corners it addresses the sorts of problems typical in occlusions.

1.3 Outline of dissertation

The body of this dissertation describes the tool in terms of its underlying technologies, implementation, use and evaluation; and is divided into chapters as follows:

Chapter 1.**Introduction** - the current chapter.

Chapter 2.**Background** - A brief exploration of the state of three-dimensional modelling of archaeology and some examples of its use (documenting existing structures, recreating structures from archaeological evidence and risks inherent in three-dimensional computer modelling of archaeological sites); a review of the various technologies which need to be combined to produce a tool which can extract data from casual photographs (edge detection, geometry extraction, calibration etc); and a comparison of the techniques available for presenting three-dimensional archaeological models. The discussion of the constituent technologies is not restricted to those actually used but also describes related approaches which were shown to be less well suited.

Chapter 3.**Underlying algorithms** - A detailed description of the algorithms which have been implemented as part of the tool, namely casual photograph calibration, image filtering, boundary detection, geometry calculation (two methods which proved useful and two which did not) and texture adjustment.

Chapter 4.**Description of the tool built** - describes the design and use of the tool built; it is introduced with a description of the requirements to be met; the design criteria applied and followed a detailed description of the functionality provided. In addition, the use of the tool is described by way of a comprehensive example.

Chapter 5.**Evaluation** - The evaluation is divided into two major parts, namely the evaluation of the constituent technologies and then the evaluation of the tool as a unit. A description and motivation of evaluation test cases are included.

Chapter 6.**Conclusions,** recommendations for casual photograph capture and recommendations for further work.

Chapter 2 Background

The present research partially addresses the task of recreating three-dimensional models of structures from archaeological evidence by allowing the use of photographs to fill-in parts of the site which are occluded. In order to provide a tool to fulfil such a function it is necessary to:

1. establish what types and formats of three-dimensional models are most widely used; for what purposes they are used; and how they are used. Areas of importance are the techniques, format and problems of scene data capture as they relate to model construction; and accuracy and presentation technologies of models. This understanding is needed in order to establish a definition of the tool functionality and data input and output formats.
2. research the techniques and algorithms which could be combined to create such a tool.
3. study other tools in this field to discover whether one with equivalent functionality exists.

This chapter is divided firstly into a brief exploration of the *state of three-dimensional modelling of archaeology* and some examples of its use for documenting existing structures and recreating structures from archaeological evidence (where much of the original structure has been lost). Evidence of the widespread existence and use of three-dimensional computer models of archaeological sites predicates the value of any tool to enhance or make such models more complete. Also explored briefly in this context is the interpretation of evidence and the assumptions made when building up a model from incomplete evidence and the impact this has on the usefulness of the model for research and teaching.

Secondly, the chapter contains a review of the various technologies which need to be combined to produce a tool which can extract data from casual photographs. A study of existing technologies is necessary in order to select and incorporate those techniques which are likely to be best suited to the task at hand. The discussion of the constituent technologies is not restricted to those actually used but also describes related approaches which were shown on closer inspection to be less well suited.

The chapter concludes with a comparison of the techniques available for presenting three-dimensional archaeological models. A comparison of techniques and formats available and to what extent they are in use is necessary in order to determine how the enhance three-dimensional models produced should be formatted to render them most useful to the widest audience.

2.1 Three-dimensional modelling of archaeological structures

In the early part of the last century Robert L. Mond and Ernest J.H. Mackay (as described in [50]) conducted a comprehensive photographic survey of the Theban tombs at Sheikh Abd el-Qurna (about 3300 black and white photographs of 18 tombs). They took a large number of overlapping close-range images using a camera rig; these photographs were printed and stuck together to form an image of an entire wall. The image of each wall was then pasted onto its corresponding wall in a three-dimensional physical model of the site. In providing a textured three-dimensional model of an archaeological site the work of Mond and Mackay can be regarded as the forerunner of virtual reality modelling in the field of archaeology.

Archaeology is an observational and often a particularly visual science which typically addresses large and complex data sets. As was attempted by the work of Mond and Mackay the study of site configuration and the deciphering of evidence are made more workable through graphical representation.

Computer modelling in general, or the construction of a simplified representation of real world objects or systems, is attractive to archaeologists as a tool for the description and analysis of the large amounts of data associated with three-dimensional reality. For example, solid modelling has been used on the temple of Roman Baths at Bath [57]. The combination of data from two-dimensional drawings into the computer model provides researchers (and to a degree non-specialists) with a sense how the original constructors utilised spatial relationships to emphasise the symbolic relationships between various parts of the temple. Furthermore, because archaeological excavation destroys the very historic contexts being investigated every available means to document and record and disseminate excavations should be utilised.

In the early days such computer models, however, were typically technical showcases sponsored by large computer firms and were often very expensive to create, computationally intensive, very restricted in terms of computer platform and of little archaeological use; most significantly there was no means of interaction with the user. Examples of such models are the Winchester Graphics System [58] and the Hadrianic Baths at Leptis Magna [56].

These problems have to a significant extent been addressed by the application of virtual reality technology and the availability of powerful relatively low-cost PCs (as will be discussed in §2.3.1).

2.1.1 Documenting Existing Structures

A vast number of existing archaeological sites have been surveyed and modelled as a means of studying and recording the structures in their current form. Some interesting examples are:

- Winchester Old Minster [58] which appears to be the earliest example of a three-dimensional textured model of an archaeological site.
- The caves at Lascaux, France which contain pre-historic rock paintings [18]
- Bahla Fortress in the Sultanate of Oman [33]
- Siwa Oasis in Egypt [33]
- Imperial burial mound Yongding-Ling in China [33]
- Castle of Nerajia on the island of Kos [38]
- Castle of the City of Chios [38]
- Monastery of the Nea Moni on the island of Chios [38]
- Hal Saflieni Hypogeum, Malta [12]
- Egyptian tomb of Sen-nedjem was developed for installation in the Egyptian Gallery of the Kelvingrove Museum and Art Gallery, Glasgow [72]

The existence of such a large number of three-dimensional computer models of such a variety of archaeological subjects is evidence of the perceived usefulness of such models and provides motivation for producing a tool which can be used to improve or complete them.

2.1.2 Recreating Structures from Archaeological Evidence

Three-dimensional textured models have been built by researchers based on archaeological evidence rather than purely from measurements of sites as they currently. These models are thus an extrapolation of what such a site may have been like; or more usually provide a broader physical context containing at its core those parts of the site which are extant. Such models are then used as a substitute for to an actual site for further research [13], [12], particularly to enhance collaboration [76]; and for education [72].

Some examples of recreated structures (rather than pure survey-based models) are:

- Roman baths at Bath [57]
- Stonehenge [69]
- Roman Chester [3]
- Maya sites [35]

2.1.3 Risks inherent in three-dimensional computer modelling of archaeological sites.

There has been much discussion of the value and relevance of three-dimensional archaeological modelling. The central question (as expressed by Sims [65]) being whether they are "a helpful research tool, or just pretty pictures".

It is possible to be misled by inaccurate but convincing models. The accuracy of any model is a product of the data captured during site survey and any assumptions made to simplify and complete model construction. An often quoted example relates to the columns of the Parthenon in Athens which are tapered so as to create the illusion of exaggerated height; this effect would be lost were the columns assumed to be cylindrical. It is recognised that making such assumptions based on current knowledge is risky.

Furthermore, models can be used to fool observers that conjecture and hypotheses are proven facts. Virtual reality models, because they contain information in a form which requires little interpretation (relative to images, CAD models or text data for example) and because they are "complete" have a far greater believability; with potentially positive or negative effects. This believability has a pernicious effect on both researchers and laypersons which is exacerbated by the often uncritical acceptance of the outputs of computer technology, especially that proclaimed as high technology.

In the case of a non-specialist, a model may be accepted as the truth rather than simply a single scholar's interpretation of collected evidence. In the case of specialists this problem is well described by the words of Martin Emele [24]:

> 'The more advanced the level of technology used for the reconstruction, the greater the belief in its authenticity. Surprisingly, this applies to both the audience and the creators. Paradoxically, even scientists still fall for the magic of a near-perfect visualisation of images of the past and are susceptible to the belief in the seeming infallibility of the objective computer.'

In summary it should be noted that special care should be taken that models are accurate and to draw to the users attention those elements of a model which are based on interpretation or assumption rather than measurement.

The purpose of the tool produced by this project is to (as far as possible) use information in the form of images rather than make assumptions, that is to make the model more complete based on evidence rather than assumptions.

2.2 Constituent Technologies

Discussion of the technologies used is divided into edge detection, camera calibration, geometry extraction and texture synthesis. It is important to note at this stage that the primary objective of this work is to provide a tool. As such emphasis was given, in the selection of suitable constituents, to the stability and maturity of the technologies involved; under the condition that such technologies would at least be fit for purpose. Furthermore, unless absolutely necessitated but the lack of suitable techniques, it is *not our intention to make a contribution in these areas.*

2.2.1 Edge detection

The field of computational edge detection can be traced back to the work of Hueckel[37], Sobel[66] and Beaudet[5] who defined so-called edge operators. Edge operators are essentially formulae based on models of edge profiles in terms of the values of image pixels (typically including partial derivatives of grey-scale image intensity.) The image pixels are convolved[2] with the operators (expressed as two-dimensional arrays) in order to highlight edge segments.

Marr and Hildreth [47] proposed an image filter based on theories of "human early vision". Their filter is defined as the Laplacian of the two-dimensional Gaussian distribution; and edges are found by convolving the pixel intensity data with the filter and marking where the resulting value is near zero. Conceptually the basis of this approach is firstly that the Gaussian is an efficient filter of small-scale texture. Secondly the Laplacian is the sum of partial second-derivative of the filtered pixel intensity data and thus has zero value at points where the first derivative is maximum (or minimum), that is edges.

The next significant development of this approach is the work of Canny [10] which combines Gaussian smoothing with simple vertical and horizontal difference operators. The difference operators are summed and positions where the gradient is a local maximum along gradient direction are found. A thresholding operation is performed to find those locations where the gradient is above some limit.

Furthermore, Canny produced a set of criteria which define an optimal edge detector, namely:
- low probability of error (failing to mark real edge points or vice versa)
- good edge localisation: points marked as edges should be as close as possible to the true edge
- single response per edge

A further development based on the work of Canny is the edge operator of Deriche [21] which is shown by him to optimally satisfy the Canny criteria.

A totally different approach is that proposed by Perona and Malik [54] which utilises the concept of anisotropic[3] diffusion. Here pixel intensity is allowed to diffuse over time, with the amount of diffusion being inversely related to the magnitude of the local intensity gradient. This process sharpens boundaries separating uniform regions within an image but is computationally expensive (involves the solution of partial differential equations).

Subsequent approaches to edge detection have utilised techniques as diverse as simulated annealing [71], Kalman filtering [19], wavelets [43] and genetic algorithms [9], [11].

All the methods described thus far suffer from the problems of unwanted and broken edges because conceptually they indicate locations on the image with the highest level of edge-ness, that is local changes in pixel intensity. Also they do not lend themselves to seeding (entering

[2] calculating new pixel values by multiplying the pixel intensities in the region of the output pixel by a matrix of weights to be applied to local pixels. Convolution is described more fully in §3.2.

[3] A physical property which has significantly different values in different directions. For example capillary motion of water in wood.

points through which the edges being detected must pass). To circumvent this, deformable whole boundary methods have been developed. A feature of these methods is the use of a curve which deforms due to gradient features of an image (typically near an object boundary). For the present discussion we will refer this category of methods as snakes although they are also known as active contours or live-wires.

A snake (first described by Kass *et al* [42]) is a deformable curve modified by internal "energy" resulting from the geometry of the curve (the distance between points, the curvature, etc.) and image energy due primarily to the intensity gradient. There are several optimisations to the snake approach, including the original variational approach Kass *et al*, dynamic programming by Amini *et al* [2] and the greedy algorithm by Williams and Shah [76].

A relevant development of the snake model is the so-called "live-wire" of Mortensen and Barratt [52]; presumably so called because the edge/snake moves from the start to the end seed points along the path of least resistance analogous with the behaviour of electric current. Their use of dynamic programming removes he need to enter manually a rough approximation of the edge to be located. It allows the entry of start and end points only (this is particularly useful to the current application because we may not have geometry information for any other points - as will be discussed later).

Model-based approaches (detecting features based on a priori knowledge of parameterisable shapes which are likely to be encountered) such as those described in Worring et al [77] and Staib and Duncan [67] were not seriously considered. It seemed highly unlikely that regular target shapes would be encountered within the occluded areas of archaeological sites.

It is interesting to note the opinion of Rothwell et al [60] that "In our view, the edge detection line of development is now mature." Furthermore, as will be seen later we require a seedable and adjustable but fairly simple scheme (because we are expect to be detecting fairly straight edges between known points). Thus we selected as a basis of our edge detection function the formulation of Mortensen and Barratt previously described.

2.2.2 *Camera calibration*

In order to deduce image information from three-dimensional scene geometry (or vice versa), one must determine the parameters that relate the position of a point in a scene to its position in the image. This is known as camera calibration.

The parameters relating image to scene geometry may be divided into intrinsic and extrinsic parameters. Intrinsic parameters are those which describe the configuration of the camera (such as the focal length) and the extrinsic parameters describe the pose of the camera (location and orientation). Although generally called camera calibration, the process of calculating both intrinsic and extrinsic parameters is in fact image calibration. For consistency with the generally used terminology we will use the term camera calibration.

The so-called pinhole camera (assumes a perspective projection of the scene onto the image plane without lens or image plane distortion) can be calibrated using six or more non-co-planar correspondences between a single image and subject three-dimensional geometry (described in [27]. Calibration is based on so-called projective geometry: extension of Euclidean geometry which allows points and lines at infinity to be treated in the same way as whose in finite space.

A major improvement of this approach is the "perspective projection model" of Tsai [75] which introduces a first-order radial lens distortion coefficient and allows calibration based on a set of co-planar points. The radial distortion coefficient addresses distortion of images captured with cameras having short focal lengths (most commonly seen as the bending of straight lines).

Tsai uses distinctive grid as a target for calibration. The grid is made up of dark squares on a white background, the corners of the squares being used at features for calibration. Calibration can be performed using a set of co-planar points by using a pair of images with known camera displacement.

Maybank and Faugeras [48] introduced the concept of camera *self-calibration* which exploits the fact that the geometry is rigid and assumes that the scene does not change over time. Here the camera is calibrated from a sequence of images without knowledge of the scene geometry or the camera position *but* the sequence of images *must be taken from the same place* (that is the camera motion is constrained to rotation).

A more intuitive (but computationally expensive) approach is that of Hartley [31] which calculates the calibration matrix iteratively by finding a set of three-dimensional points consistent with the image points in the set of images. Conceptually the problem is solved by finding the only set of intrinsic calibration parameters which would allow the same feature on different images have the same three-dimensional location (which it must unless the scene has changed between images). The Hartley model allows unconstrained motion of the camera.

Pollefeys [55] goes further by using that fact that pixels are square and introducing a post-processing phase which finds so-called dense correspondences. This phase finds dense correspondences (that is at the pixel level) based on a) the gross feature correspondences and b) epipolarity constraints (which are in turn a function of the extrinsic calibration parameters). This algorithm of Pollefeys removes a further constraint; namely the focal length of the camera (one of the intrinsic parameters) is allowed to vary.

The self-calibration approach is particularly attractive in an application which includes geometry recovery because the positions of corresponding points are calculated as a by-product of camera calibration. (The work of Pollefeys described above could just as easily be discussed under the later category of *geometry recovery*)

Becker and Bove [7] take a very different approach in that they exploit the fact that the locations of the vanishing points in a perspective view contain information about the intrinsic and extrinsic camera parameters. Furthermore, they can calculate the radial and de-centering lens distortion by minimising the vanishing point dispersion. Their algorithm, however, requires images to contain three sets of parallel lines which are mutually orthogonal. Although such lines will very commonly be found in architectural/urban scenes, it seems fairly unlikely that such lines will occur in archaeological sites.

Stereo camera calibration is excluded from discussion here on grounds of relevance because, for the purposes of this project, our source images are monocular.

On the basis of certain of our requirements, namely that:

1. we have unsequenced images
2. geometric regularities (parallel lines etc) are unlikely to be present in the scenes and images

it was decided to implement the pinhole camera model. It was further felt that should this method prove inadequate due to radial distortion effects then the algorithm of Tsai [75] would be implemented.

2.2.3 *Geometry recovery*

There are a variety of different approaches to the recovery of geometry from images. These range from those which are:

> *model based*, that is based on the expectation that the scene contains objects of known shape, to those which are based on
>
> *geometric optics*, that is calculating positions of features in scenes from their positions in two or more images (even very many images as in the case of a video stream).

The underlying concept of all *model-based*[4] approaches is the *à priori* knowledge of what is to be found in the scene, for example straight lines, planes or symmetry. Scenes containing urban environments, architecture or other man-made objects are particularly suited to a model-based approach.

The model based paradigm was introduced by Roberts [59] who stressed the importance of the perspective transformation that underlies our perception of the three-dimensional world, and exploited the properties of this projection to recognise objects in images.

Kanade [41] introduced the concept of skewed-symmetry which is based on the fact that many objects are bilaterally symmetrical (both those occurring in nature and man-made). These features, when arbitrarily orientated, are projected onto the image as skewed-symmetries. Skewed-symmetry can be exploited by geometry recovery algorithms as a test of whether segments could be the projection of features that are bilaterally symmetric in three-space. It is recognised that although prevalent in the natural world in particular, bilateral symmetry and its side effect skewed-symmetry is untypical of scenes which depict archaeology.

In the author's view the most influential work in the field of model-based geometry construction is the FAÇADE system of Debevec, Taylor and Malik [20]. Their system allows he user to identify manually the corresponding structures in a pair of images of a scene and fits pre-defined parameterised models to these structures. They have substituted the goal of finding correspondences between points to the simpler and more robust problem of finding correspondences between parameterised volumes in the scene.

Recent developments in the field of model-based geometry extraction are those due to Shum et al [63] (generation of three-dimensional models from a panoramic mosaic by exploiting architectural features such as parallel lines and planes) and Becker [7],[6] (which in essence exploits the presence of orthogonal sets of parallel lines, such as those found on the edges of cuboids, to calculate vanishing points and then uses this information to calibrate the camera).

Coorg and Teller [16] use an algorithm which uses a very large number of so-called pose-annotated (location and orientation known) images of an urban environment. They extract vertical facades, as they are common in urban scenes. Pose information enables their algorithm to pre-select the much smaller set of images that is relevant to any 3-D region of interest. Texture data is then synthesised by combining the large set of images which include each facade together using a weighting factor based on the relative orientations of image and facade (very oblique views have a low weight).

[4] not to be confused with the model-based approach to edge detection (see §2.2.1) which assumes a model for the local profile of an edge; the model-based approach to geometry recovery relies on assumptions about the larger-scale shape of edge curves.

However, in the words of Coorg and Teller "Clearly our algorithm has many significant limitations, for example.... facade relief". This is an inherent weakness of any model-only based approach – the detailed geometry (that is the underlying shapes) is assumed and only parameters or scales are calculated. Great care should thus be taken, when using model based approaches in the area of archaeological modelling, so as not to generate inaccuracies introduced by assuming underlying geometry and also so as not to lose archaeologically important relief information.

There exist a variety of approaches to geometry reconstruction which are not model-based but rather *geometric-optics-based* and rely on multiple images. These approaches can be categorised further into *stereopsis, parallax* and *video stream* based.

The *stereo* approach could be described as the classical geometry recovery method (where two images of the geometry are available) and can be traced in the computing field to the algorithm of Longuet-Higgins [46]. Features are identified in the one image and searched for in a second image along, or near to, the so-called epipolar lines. An epipolar line is that line (actually a curve) on an image along which a point in an image taken from a different location must lie. Once the feature has been found in the second image its three-dimensional location can be calculated by triangulation.

The *plane+parallax* method of Kumar et al [44] is superficially similar to the stereo approach; the difference being that residual parallax constraints are used rather than epipolarity constraints to simplify the search for correspondences. The parallax approach assumes a dominant three-dimensional structure (for example a few intersecting planes) in order to calculate the likely position of corresponding points. Residual parallax is the calculated difference in the location of points (between images captured from different places) actually on the surface of the dominant three-dimensional structure: points near-to the dominant three-dimensional structure would be expected to be translated by a similar amount. It should be noted that although this approach is model-based in that it assumes an underlying model: the model only reduces the search for corresponding points and does not limit the detailed geometry extracted.

A great deal of work has been done recently in the field of geometry extraction from **video** *streams*. This field is very closely linked to (now inseparable from) developments in self-calibration (described earlier) resulting in some overlap in the present discussion. The most well known is the so-called factorisation approach of Tomasi and Kanade [74] (developed further by Morita and Kanade [51]). Here a matrix which describes the two-dimensional positions of each feature in each frame is built up. The position of each feature is relative to the centroid of the set of feature positions, that is the two-dimensional translation is *factored* out. This matrix is the product of a rotation matrix per frame and a three-dimensional shape matrix. These matrices are separated out via a technique they describe as singular value decomposition.

Beardsley et al [4] describe a method for the fully automatic construction of graphical models of scenes when the input is a sequence of closely spaced two-dimensional images (ie video sequence). They develop on the concept of the epipolarity constraint between two images (the geometric constraint on the position of a feature in a second image based on its position in the first). They introduce a so-called tri-focal tensor (analogous to the fundamental matrix for a pair of images) which relates the location of a feature in a third image to its position in two other images. The tri-focal tensor is given an initial value based on a few known correspondences and then allowed to converge on a more complete solution by including all correspondences which are consistent with it (within a predefined disparity).

Fitzgibbon and Zisserman [28] have a similar approach but find they fundamental matrix by searching for sets of co-planar points. They choose as example subjects either architectural scenes (house, castle etc) or an isolated non-polyhedral object (dinosaur on a turntable). These two very different types of subjects are selected to show the broad applicability of their approach. However, the actual method used in these two cases differs significantly: in the case of the architectural scenes planes are extracted from the 3D data using the RANSAC technique: random three-point subsets (co-planar by definition) of the data are selected and the number of 3D points which are near to the location of each plane are counted. The plane with the greatest count is stored, and the data points which were consistent with it removed from further processing. Repeating this process extracts the largest planes from the dataset. This approach is clearly model-based (in that they are searching for and expecting to find planes).

The approach taken for the non-polyhedral dinosaur subject differs in that geometry is extracted by "segmenting the background and intersecting the cones formed by the occluding contours". Although Fitzgibbon and Zisserman state, "For the non-polyhedral objects, the surface extraction problem is more difficult, mainly due to the *sparsity of the data."* (my emphasis) it appears to the present author that the problem is simply due to their model's lack of scalability from subjects comprised of a few planes to those comprised of many conical sections.

On the basis of certain of our requirements, namely that:

1. we have single (rather than multiple images) images
2. geometric regularities (symmetry, planes, parallel lines, orthogonal sets of lines, vertical facades) are unlikely to be present in the scenes and images of most archaeological sites

it appears as if no suitable techniques or algorithms exist for our application (see Table 1). It was thus decided that a technique should be developed which could be used with single images of scenes which do not contain regular features but takes into account known geometry in the region of occlusions.

Technique	**Requires multiple images**	**Assumes geometric regularities**
skewed-symmetry (Kanade [41])		Yes
FAÇADE system (Debevec, Taylor and Malik [20]).		Yes
parallel lines and planes (Shum et al [63])		Yes
orthogonal sets of parallel lines (Becker [7],[6])		Yes
vertical facades (Coorg and Teller [16])		Yes
classical stereo (Longuet-Higgins [46])	Yes	
plane+parallax (Kumar et al [44])	Yes	
factorisation of video streams (Tomasi and Kanade [74], Morita and Kanade [51].	Yes	
tri-focal tensor (Beardsley et al [4])	Yes	
sets of co-planar points (Fitzgibbon and Zisserman [28])	Yes	Yes

Table 1 Geometry recovery techniques reviewed

Existing software tools for extracting geometric data from images

Two commercially available products provide functionality to extract geometry from photographs. *PhotoModeler* [25]uses manually entered correspondences between sets of two or more photographs to calculate the relative geometry of the points. It is then possible to define planar triangles between these points and drape texture from the source photographs over them. Although there appear to be similarities between the Photomodeler product and this project they differ significantly as follows:

1. PhotoModeler requires multiple images of any region of the site (we are addressing the problem where only a single image exists).
2. PhotoModeler assumes planar surfaces which while being appropriate for architectural scenes are not expected to be applicable to archaeological scenes.
3. There is no means to include accurate measurements of surrounding geometry, such as those available from photogrammetric survey of the site.

In summary PhotoModeler is designed primarily for man-made objects with surfaces that can be described well by large triangular elements.

In a similar vein is the tool *3D-Builder* [73] which combines information from a large number of photographs of scenes containing complex shapes. 3D-Builder does appear to allow the extraction of geometry from a single photo by exploiting two features of modern man made objects which, however, are seldom present in archaeological subjects - parallel and perpendicular lines and large planar surfaces. This product is not applicable to the current problem for largely the same reasons as those sited for PhotoModeler above.

2.2.4 *Texture and structure synthesis*

An alternative to using texture from images (to fill gaps due to occlusions) is to synthesise texture. Texture can be synthesised via a range of different techniques such as using statistics of pixels and wavelet coefficients [34], properties of joint wavelet coefficients [64], multi-scale sampling [22]. A texture synthesis approach has been avoided in the current project on the basis of its inherent lack of accuracy and that we in fact do have image data. However, in the absence of image data it may be a desirable route: given the assumption that realistic (but unreal) texture is a better than no texture at all.

An extension to texture synthesis is *structure*[5] *synthesis*: using large-scale spatial domain information when filling in holes in an image. Work has been done in this field by Bertalmio et al [8] (based on the image noise removal by using frequency and spatial domain information approach of Hirani and Totsuka [36]). The underlying principle of this work is to shrink gaps in an image by growing the isophotes (lines on the image where the intensity of light is the same) arriving at the gap boundary in a smooth way, without crossings and without changing the angle of arrival. Intuitively (and from the careful selection of images by Bertalmio et al) it is expected that such an approach is very sensitive to the size and shape of gaps to be filled: narrow gaps (such as those due to scratches, folds, print or writing) being easier to fill than gaps having more balanced aspect ratios. The sorts of occlusions which occur due to photogrammetric survey typically have a range of aspect ratios; thus excluding the technique of Bertalmio et al from further consideration.

[5] The use of the term structure here relates to large-scale spatial domain information as represented in the image (as opposed to structure in the three-dimensional geometric sense). It is accepted that the distinction between texture and structure is a matter of scale. For example, the arrangement of bricks in a wall would be considered structure in an image which includes a few bricks and texture in an image which includes several city blocks.

2.3 Presentation

Various competing approaches exist for the graphical presentation of real world scenes.

The most common approach for rendering three-dimensional scenes is **simplified-geometry** based and functions by generating surfaces consisting of polygons supplemented with texture draping to add detail; the polygons being calculated from the underlying geometry measurements and inferences from the three-dimensional scene; and the texture providing further details (some small-scale geometric information is included in the texture).

Because of the widespread adoption of the geometry-based approach there exist well-developed underlying technologies in both software and hardware (eg OpenGL). VRML is geometry-based and is described more fully in the next sub-section (*Virtual Reality and VRML*).

A weakness of the simplified-geometry based approach described above is that texture mapping is not always an effective substitute for mapping three-dimensional details to a surface[6]. When observed from an oblique angle the flatness of the surface is noticeable (or if seen by a moving observer the absence of parallax reveals the flatness of the surface). A recent development which addresses this problem is the work of Oliveira et al [53] which introduces an efficient algorithm whereby the texture of the surface is pre-warped in accordance with its relief. The algorithm is efficient because the parallax effects due to the height field are factored out in a way which allows their calculation in one dimension at a time; the in-built rendering capabilities of the platform are then used to process the pre-warped texture map further. However, the underlying assumption of their approach is that surface may be represented as planar with a much smaller-scale height field. This assumption may be applicable to models of architecture but is not valid for the types of sites, namely archaeological, which are expected to be addressed using the tool being implemented.

The purpose of computer graphics is to create a realistic visual representation of a scene. In geometry-based rendering it is assumed that the interface between computer vision (that is scene capturing) and computer graphics is data in the form of geometry and texture: in other words the scene is reduced to a geometric model with surface textures which is then used to render a representation of the scene.

An entirely different approach is that adopted by *image-based rendering*. Here the steps of a) transforming captured images into geometry (edge/boundary detection, calibration, geometry recovery etcetera) and then b) creating renderings from them are avoided. In principle, image-based rendering avoids the problem of extracting geometry by simply using the input images to render the output graphics. There are two basic approaches to image-based rendering, namely *interpolation* between images and the use of *light-fields*.

In the *interpolation* approach new images are generated for new viewpoints using interpolation between images [14] or, in order to create a virtual reality effect, between "panoramic cylinders" [15] and logically "panoramic bubbles" [78].

The underlying concept of the *light-field* approach is to represent the scene as an n-dimensional scalar field which describes the flow of light in all places in all directions. The light field has been developed by various researchers: the 5D plenoptic model of Adelson and

[6] The decision as to what scale of geometric detail is better represented as polygons versus texture (based on balancing fidelity, data volumes and rendering performance) is the subject of an entire branch of computer graphics research and is not discussed here.

Bergen [1] (with further development by McMillan and Bishop [49]) and the most recently defined 4D Lumigraph of Gortler et al [29].

By using images such systems provide visually realistic results, while avoiding the task of geometric reconstruction. As would be expected, however, image-based rendering systems are weak in the areas of navigation, collision detection and simulation of different lighting conditions; and most significant to our application: geometric accuracy.

The most important reason that the image-based rendering approach is not suitable to the current task is that the overwhelming majority of the geometry is well known from the photogrammetric survey of the site. The purpose of this work is to fill in gaps due to parts of the site occluded in the photogrammetric images.

2.3.1 Virtual Reality and VRML

For the present discussion virtual reality is defined as a self-directed computer-generated experience which gives the illusion of participating in a synthetic three-dimensional environment. The concept of virtual reality based on computers can be traced back to the writings in of Ivan Sutherland in 1965 [70]. The earliest form of virtual reality (the use of stereographic head-mounted displays within a computer-generated room) has been around since 1970 [45]

Developments in the field continued at institutions with access to the large amount of computing power required to make the generation of useful synthetic environments practical. During the 1980s, the drop in the cost of computers and the development in technology allowed virtual reality to become more generally used. However, the use of virtual reality at that time was still restricted to powerful special purpose machines.

More recently further advances have made it possible to run virtual reality models on PCs (albeit with problems of latency and slow frame rates which are the focus of a range of efforts within the industry). This broader availability of virtual reality technology and the existence of the Internet as a means of access to data have in turn brought about a requirement for standard ways to author, store and publish virtual reality model data.

This need has been addressed by defining a language called Virtual Reality Modelling Language (VRML) for capturing and disseminating virtual world data which is used in much the same way as HTML (Hyper Text Mark-up Language) can be used to transfer documents over the Internet.

VRML, first known as Virtual Reality Mark-up Language, was conceived in the spring of 1994 and discussed at the historical First International Conference on World-Wide-Web held at CERN, Geneva [61]. The name was subsequently changed to Virtual Reality Modelling Language to emphasise that the language was primarily concerned with three dimensions.

VRML is based on the Inventor file format [62] from Silicon Graphics, which includes ways to specify geometry, lighting, materials, 3D user interface widgets, and viewers. VRML has been enhanced (at version 2.0, also known as VRML97) to allow the specification of interaction and animation. VRML97 has been adopted as the web standard for three-dimensional environments by the International Standards Organisation [40].

For the reasons listed above and in the light of the prominence of the VRML format in the field of geometric modelling it felt important that any output from the tool should be available in this format.

2.4 Summary

Our investigation of the relevant literature has yielded the following:

- Geometric, and in particular virtual reality modelling of archaeological sites is a growing field (see §2.1.1). There are, however, inherent risks relating to the interpretation of evidence and presentation via such technologies (see §2.1.3).

- The formulation of Mortensen and Barratt [52] appears to be the most suitable to our boundary detection requirements (see §2.2.1).

- The pinhole camera calibration model as described in [27] seems a good basis for our calibration requirements (see §2.2.2). The algorithm of Tsai [75] is available as an alternative should radial distortion errors prove to be a problem.

- No existing technique appears to satisfy our requirements with respect to geometry recovery, implying that this need will require satisfaction by the development of an algorithm (see §2.2.3). Similarly, existing software for the extraction of geometry is not suitable.

- Texture synthesis is probably not suitable to the problem under consideration (see §2.2.4).

- An option must be built-in to store output in VRML format (see §2.3.1).

Chapter 3 Underlying algorithms

In essence the tool has been built in order to extract patches of surface texture from casual photographs and apply them to regions of three-dimensional models which are occluded in photogrammetric surveys of archaeological sites. In addition the geometry within the occluded regions is estimated by using information from the casual photograph along with some assumptions about surfaces in the region of the occlusion.

In order to achieve this it is necessary amongst other things to provide functionality to: calibrate the casual photograph using data from parts of the three-dimensional model which are not occluded; segment the casual photograph along discontinuities in surfaces; warp patches of surface texture from their shape in the casual photograph to their shape in the model; provide some means of estimating the geometry within the occlusions; and allow the adjustment of patch texture data to match the surrounding photogrammetrically captured texture better. Algorithms have been devised, adapted or implemented to satisfy these functional requirements.

The tool built is based on the implementation of algorithms to address the problem as described above along with supporting functionality such as the ability to read and write files etc. This chapter describes the following algorithms upon which the tool is based:

1. *casual photograph calibration* - It is possible to use data based on corresponding features between the geometry of a scene and a photograph thereof to calculate a set of parameters. These parameters describe the camera and its location and orientation within the coordinate system of the scene and can be used to calculate the position of all features within the scene. The process of calculating such parameters is known as camera calibration.
2. *image filtering* - The quality of casual photographs is expected to vary widely. It is thus necessary to enhance the images using filters in order to improve in particular the detection of boundaries.
3. *boundary detection* - Regions of casual photographs are used to provide texture patches to occluded parts of the scene. Furthermore, the boundaries of these regions will correspond to geometric model and scene features such as corners to which they must be made to adhere. Accurate detection of boundaries is thus essential to satisfactory functioning of the tool.
4. *geometry calculation* - Many occlusions include corners: both concave and convex, and of two or more mostly flat surfaces. Using boundary information from casual photographs along with assumptions about the surfaces within these boundaries allows the estimation of the geometry within occluded regions. The two algorithms used *are planar rough-triangles* (see §3.4) and *minimisation of least squares error* (see §3.5). Two algorithms which on close inspection were found to be intrinsically flawed are described in Appendix B .
5. *texture adjustment* - Image data from casual photographs is used to fill-in occluded regions of the model. However, due to differences in lighting particularly due in the case of archaeological sites to the position and presence of the sun, there are differences in intensity, contrast and colour balance between images. It is thus necessary to adjust the patches extracted from casual photographs so that they more closely match the photogrammetrically captured images which comprise the bulk of the model's texture.

This chapter describes a number of underlying algorithms which are largely unrelated outside the context of the tool constructed and are thus treated independently. Implementation of the above algorithms along with their integration into a tool is discussed separately in Chapter Chapter 3.

3.1 Camera calibration

In order to project the boundaries of occlusions onto casual photographs the photographs must be calibrated. As would be expected these projected occlusion boundaries form many of the boundaries of patches extracted from casual photographs.

Camera calibration is defined here as the process of calculating parameters which describe both a) the camera and b) its position within the coordinate system of the geometry photographed. The calibration algorithm used is described as the pinhole camera model [27].

Essentially a pinhole camera (one with no radial distortion of the lens/s) can be described mathematically using a matrix to relate the positions of objects in space to their representation on the image plane (or a printed image — here the parameters include the effect of the image printing process). This relationship can be expressed as follows[7]:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f/p & 0 & 0 \\ 0 & f/p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} u_c \\ v_c \\ 0 \end{bmatrix} Z \qquad (1)$$

where

the object is at (X, Y, Z),
s is a scale factor which can be eliminated from the equations
f is the focal length,
u_c and v_c define the point on the image plane through which the Z-axis passes (optical centre)
p is the pixel dimension measured in units of the three-dimensional world containing the scene and is used to normalise the image to units of pixel size
and the image of the object appears at pixel (u, v).

In the above formulation the origin of the X, Y and Z-axes are at the camera position. In order to generalise the matrix which relates the photographic image to the geometry it is necessary to include the position of the camera relative to a world coordinate system. This is achieved by introducing the camera's rotation relative to the axes (a 3-by-3 array) and its displacement relative to the origin of these axes (a three-dimensional vector).

[7] In the case where the optical centre is at the origin (ie u_c and v_c are zero), the pixel width and heights are equal to the unit distance in the three-dimensional coordinate system, the value of f substituted with d and s substituted with λd and the equation $\lambda w = Z$ added as a row to introduce a third dimension to the left hand side; this formula reduces to the simple perspective projection:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda w \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Inclusion of the abovementioned modifies the calibration matrix to the form:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u r_1 + u_c r_3 & \alpha_u t_x + u_c t_z \\ \alpha_v r_2 + v_c r_3 & \alpha_v t_y + v_c t_z \\ r_3 & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2)$$

where

$\alpha_u = f$ / pixel width,

$\alpha_v = f$ / pixel height,

r_1, r_2 and r_3 are the row vectors of the camera rotation matrix

and (t_x, t_y, t_z) defines the displacement of the camera from the origin of the geometric coordinate system.

For the purpose of the current exercise all that is required is the calibration matrix; in other words we are not interested in separating out the camera position, focal length and so forth but only in the calibration matrix as a whole (in order to project lines within the geometry onto the casual photographs). Therefore we can treat the matrix simply as a set of homogeneous parameters of the following form:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (3)$$

We thus have a matrix with eleven unknowns (q_{11} though q_{33}). In order to solve for the set of q parameters a system of 11 simultaneous equations is required. This can be achieved by relating six geometry-to-photograph points (an equation in u and v per point). Actually, using six points provides us with an over-constrained set of equations (12 equations with 11 unknowns) which may be solved using a pseudo inverse.

In summary this algorithm allows the calculation of the parameters relating the positions of features in the three-dimensional scene to their corresponding locations in the image and requires a minimum of six corresponding points to function. Furthermore, due to the nature of a minimisation of least squares calculation the robustness of the results improves with an increasing number of correspondences (as long as additional points have a similar magnitude of error as the minimum set of six).

3.2 Filtering algorithms

Filtering of the image data captured from casual photographs was expected to be necessary because of the likely presence of noise in such data (for example scratches and edges of shadows which could be the source of spurious edges). Furthermore, features of fine-grained texture such as cracks on and joins between stones (which none the less fall within an inter-edge region) should be removed.

Filters used need to be both user adjustable (so that they can be tuned for image sets) and be targeted at noise and fine-grained texture without removing image data necessary for locating edges. Various filtering algorithms have been selected for inclusion in the tool as it was expected that this would improve the detection of edges.

The image filtering techniques selected are:

Box filter. Each filtered pixel is simply the mean value of a set of pixels at and near to that locality, the exact set size being selected by the user (degree or scale of the filter). The effect of this conceptually simple filter is to "blur" the image and thus remove fine-grained noise and texture (ie. high spatial frequencies).

Gaussian smoothing. It is well established that convolution of the image with a Gaussian mask will remove small-scale texture and noise as effectively as possible for a given spatial extent in the image. Gaussian smoothing is a special kind low pass filter, similar to the box filter described above except that output pixel values are calculated from local input pixel values according to a weighting factor. These weights can be represented as a `bell-shaped' hump or Gaussian curve with the peak localised at the position of the output pixel. In 2-D, an isotropic (i.e. circularly symmetric) Gaussian has the form:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where x and y define the position of the source pixels relative to the output pixel and σ is the degree of the filter, that is a large value of σ results in larger scale smoothing.

Difference of Gaussian. In the field of computer and biological vision there is considerable evidence to support the theory that edges of objects can be made more prominent by the application of a so-called difference of Gaussian filter (Adelson and Burt 83). Like the filters described above output pixel values are calculated as a weighted sum of input pixels. The weighting function in this case is the difference between a pair of Gaussian curves with differing σ value. The difference of Gaussian function has a "Mexican hat" shape. This filter sharpens and image by subtracting a fraction of nearby pixel values (conceptually the blurred bit) from the input pixel at that location. The degree (or scale) of the filter and the ratio of degrees or the pair of Gaussian curves as adjustable in order to allow optimisation for differing image sets.

The spatial scale of all filters is user adjustable as is the so-called sigma-ratio of the difference of Gaussian filter (default value of 1.6 which approximates a Laplacian of a Gaussian).

3.3 Boundary detection

Occluded regions of a three-dimensional scene will typically be made up of separate surfaces, for example a hidden corner would be make up of two (or three) principal surfaces. These surfaces will appear in a casual photograph as regions. In order to apply surface texture from a region in a casual photograph to a surface in a three-dimensional model (and for use in calculating hidden geometry) it is necessary to detect the boundaries of such regions in the casual photograph. Furthermore, the detection of boundaries should be based on user supplied seed points, which correspond to the significant features in the scene.

The boundary detection functionality implemented is based on the algorithm of Mortensen and Barrett 95 [52] known as Live-Wire. Live-Wire is a variation of the active contour (also known as snakes) methods of boundary detection.

This algorithm calculates a path starting at a seed point and ending at another. The route of the path should be boundary-like (ie continuous, fairly smooth) and follow the boundary of the region (ie localise on image intensity characteristics typical of an edge).

In order to achieve this a "cost" is calculated between each pixel (starting with the costs to those pixels surrounding the start seed pixel). The sequence for calculating the costs-to of the remaining pixels is determined by recursively executing the following steps:

1. the lowest cost-to pixel of all calculated pixels is then selected and the costs to its neighbours calculated
2. these neighbouring pixels are inserted to the list of calculated pixels in accordance with their costs-to

In this way the costs from the start pixel to pixels in an expanding region localised around the start pixel are calculated.

A subtlety of the technique is that the selection of lowest cost-to pixel (for recursive calculation of the costs to its neighbour) is from the set of *all* pixels that have been reached so far. This means that the calculation of pixels focuses in areas which have the lowest cost. When the next pixel selected in this way for calculation is the end pixel then the lowest cost path has been found. The search can be visualised much like an ink blot spreading out from the start pixel along the lowest cost paths quickest and terminating once the end pixel is reached.

The algorithm of Mortensen and Barratt 95 defines the cost of a path as the sum of the costs at and between pixels (local costs - *l(p,q)*) and is comprised of the components: *Laplacian Zero-Crossing* (f_Z) , *Gradient Magnitude* (f_G) and *Gradient Direction* (f_D) as follows:

$$l(p,q) = \omega_Z f_Z(q) + \omega_D f_D(p,q) + \omega_G f_G(q) \tag{1}$$

Laplacian Zero-Crossing. Convolution of the image with a Laplacian kernel approximates to the second derivative of the image[8]. Zero-crossings, ie where adjacent pixels have differently signed values of the Laplacian, correspond to points which have the highest (or lowest) gradients. Points with high gradients fall near to or on edges (Marr and Hildreth 80) and are thus allocated a low cost. As the zero-crossings occur between pixels (with extremely rare exceptions) the low cost is assigned only to the single pixel which has the Laplacian closest to zero. That is:

$$f_z(q) = \begin{cases} 0 & \textit{if } I_L(q) \text{ and } I_l(\textit{adjacent } q) \text{ have different signs and } I_L(q) \text{ nearer to 0} \\ 1 & \text{all other situations} \end{cases} \tag{2}$$

where $I_L(q)$ is the Laplacian calculated at pixel q.

Gradient Magnitude. The Laplacian zero-crossing above, however, does not differentiate between points of high and low gradient. The gradient magnitude is derived from the magnitude of a gradient vector which has as x and y components the partial derivatives of the pixel value in the x and y directions respectively. In order to generate a low gradient magnitude cost at pixels where the gradient vector magnitude is greatest the following transformation is performed. First the gradient vector

[8] use of the difference of Gaussian filter described previously in §filter with a sigma ratio of 1.6 is a close approximation to a Laplacian.

magnitude is normalised relative to the highest gradient vector magnitude in the entire image, which ensures that only values between zero and one are produced and that image contrast is factored out. Secondly the reciprocal value of the normalised gradient vector magnitude is used as the gradient magnitude cost. That is:

$$f_G = 1 - \frac{G}{\max(G)} \tag{3}$$

where the gradient $G = \sqrt{I_x^2 + I_y^2}$ and

I_x and I_y are the partial derivatives of the image in the x and y directions respectively.

Gradient Direction. This cost is dependent on the lack of smoothness or straightness and operates by assigning a high cost to sharp changes in edge direction. A gradient vector, defined as the direction in which the gradient is highest, may be calculated as the vector which has as x and y components the partial derivatives of the pixel values in x and y directions respectively. A low gradient direction cost is assigned to the edge between a pair of adjacent pixels where their gradient vectors are close to perpendicular to the edge joining them. That is:

$$f_D(p,q) = \frac{1}{\Pi}\left(\cos\left[d_p(p,q)\right]^{-1} + \cos\left[d_q(p,q)\right]^{-1}\right) \tag{4}$$

where $d_p(p,q) = D(p) \bullet L(p,q)$,

$d_q(p,q) = L(p,q) \bullet D(q)$,

$$L(p,q) = \begin{cases} (q-p) & \text{if } D(p)\bullet(q-p) \geq 0 \\ (p-q) & \text{if } D(p)\bullet(q-p) < 0 \end{cases}$$

and *D(p)* is the unit vector perpendicular to the gradient at *p* and *(p-q)* is the unit vector[9] from *q* to *p*.

The Laplacian zero-crossing and gradient magnitude can be viewed as the external costs of the boundary, attracting it into regions of low cost. The gradient direction is mostly an internal cost which is kept low by keeping the edge straight and perpendicular to the gradient of pixel intensity.

The abovementioned three cost components are combined as a weighted sum.

The lowest cost path is found recursively by selecting the pixel with the lowest cost-to (starting at the *start* pixel) and calculating the costs to its adjacent pixels. These new pixels and their costs-to are added to the list of calculated pixels from which the new lowest cost-to pixel is then selected to start the next recursion. When the *end* pixel is selected for calculation the lowest cost has been found.

[9] the original formulation as published in Mortensen and Barratt 95 did not mention that in the calculation of *L(p,q)* that the vectors *(q-p)* and *(p-q)* linking adjacent pixels should be normalised; otherwise diagonal paths between pixels (being $\sqrt{2}-1$ longer) are unintentionally penalised by being given a higher cost.

The Live-Wire approach was selected as the preferred method for finding boundaries because it has the following characteristics particularly suited to the problem under consideration:

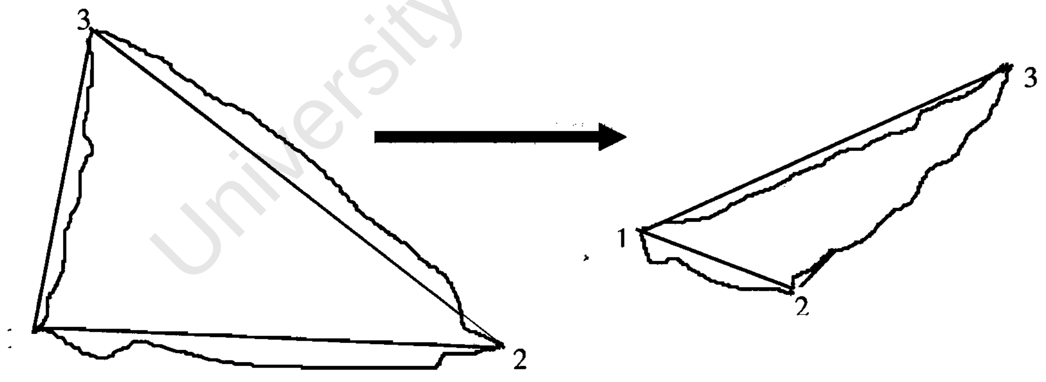
> The *start and end points can be selected* and only the best path between them found. This requirement excludes edge detection techniques which simply highlight all "edgy" regions, for example Canny edge detection (Canny 86).
>
> The approach is particularly appealing because it *combines edge cost terms* in a way that allows the relative importance of each of the terms to be set by the user, namely by using adjustable weighting factors. This is expected to be useful when finding boundaries in images with a wide range of region and edge characteristics.

3.4 Draping image data over planar triangles

Initially it was believed reasonable that the problem under consideration could be simplified by assuming that the three-dimensional regions to be filled-in from casual photographs are made up of quasi-triangular planes[10]. Triangles were selected because they are the only polygons which are always planar. As will be shown in §5.2.3 this approach is not suitable in all cases, particularly for highly curved surfaces; it does however provide an effective solution for situations where the surfaces are near to planar.

This algorithm enables the calculation of the positions in three-dimensions of the edges of the planar regions based on:

1. the shape of the rough edges of the corresponding image triangle and
2. the difference in the underlying shapes of the three-dimensional and image triangles (as defined be the configuration of their vertices - see Figure 1)



a) image rough triangle b) geometry rough triangle

Figure 1 showing the resulting rough triangle in b) which results from the rough image triangle in a) and the relative positions of the triangle vertices in the image a) and the geometry b)

Effectively a planar surface is being generated in the three-dimensional model upon which to warp the corresponding triangular image data.

The location of an edge point in three-dimensions can be calculated from the corresponding edge-pixel in the image using a transformation based on the relationship between the

[10] quasi-triangular because the edges of the triangles rather than being straight are deformed in accordance with edges detected on casual photographs

configurations of the underlying image- and three-dimensional triangles. At a high level this transformation functions by calculating the position of each edge pixel relative to a new coordinate system the axes corresponding to two edges of the underlying image triangle (the (*a*,*b*) coordinate system). The location in three-dimensions is then calculated using the location of the pixel in (*a*,*b*) coordinates but using the corresponding edges of the underlying three-dimensional triangle as the *a* and *b* axes.

The transformation is derived by relating a) the position of each boundary pixel in two-dimensions to the positions of the vertex pixels in two-dimensions, b) the assumption that rough triangles are planar and c) the relative positions of the corresponding vertices in three-dimensions; it is possible to calculate the locations in three-dimensions of the pixel. The algorithm is as follows (see Appendix for detailed derivation):

First calculate the position of the edge pixel in coordinate system (*a*, *b*) defined by two of the edges of the triangle on the photograph as shown in Figure 2.

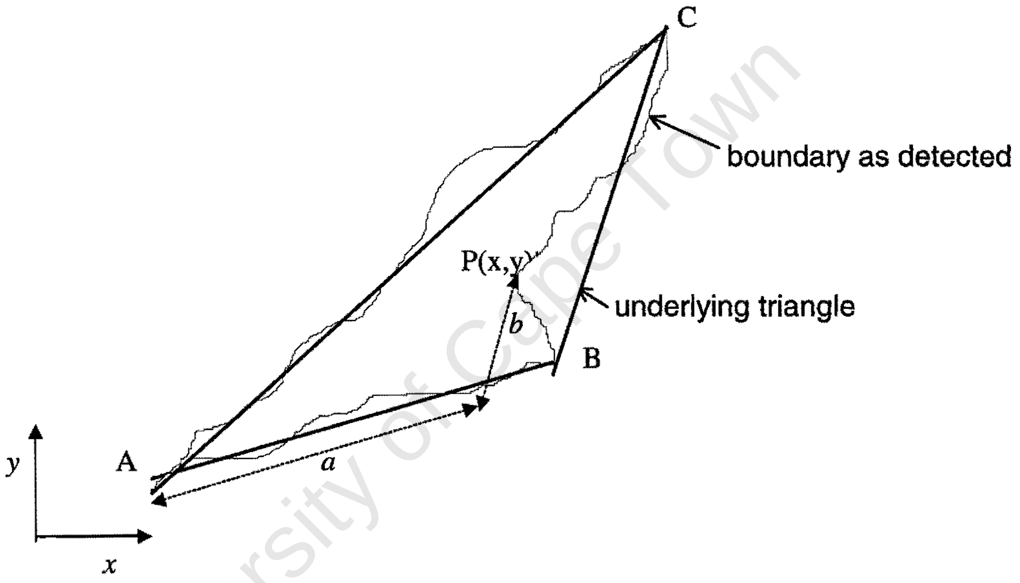


Figure 2 showing the position of a boundary pixel P(*x*,*y*) in terms of the new coordinate system *(a,b)* based on the edges of a triangle ABC. The origin of the (a.b) coordinate system lies at point A and the directions of the two axes a and b lie in parallel with the edges AB and BC respectively of the underlying triangle

The pixel at x, y lies at the following position in the *(a,b)* coordinate system:

$$a = \frac{(x - x_A)/(x_C - x_B) - (y - y_A)/(y_C - y_B)}{(x_B - x_A)/(x_C - x_B) - (y_B - y_A)/(y_C - y_B)} \tag{1}$$

$$b = \frac{(x - x_A)}{(x_C - x_B)} - \frac{(x_B - x_A)\left((x_B - x_A)/(x_C - x_B) - (y_B - y_A)/(y_C - y_B)\right)}{(x_C - x_B)\left((x - x_A)/(x_C - x_B) - (y - y_A)/(y_C - y_B)\right)} \tag{2}$$

Next the *a* and *b* coordinates for each pixel in two-dimensions can be transformed into three-dimensions using the corresponding triangle edges in three-dimensions as shown in Figure 3.

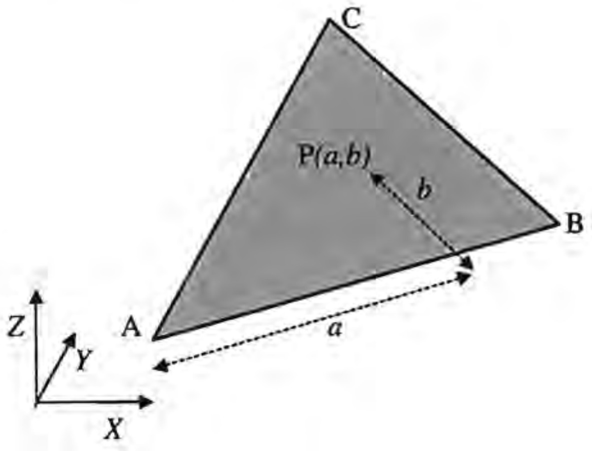


Figure 3 showing the position of a point P*(a,b)* on the plane passing through the points ABC in an XYZ coordinate system

In three-dimensions the (a,b) coordinate system is defined as to correspond to the edges of the underlying triangle. In order to calculate the location in (X,Y,Z) coordinates it is necessary to apply the a and b coordinate values to the vectors making up the edges of the triangle as follows (see Figure 3):

$$P(X,Y,Z) = A + aV_{AB} + bV_{BC} \tag{3}$$

where

A is a vertex of the triangle in three-dimensions (and the origin of the (a,b) coordinate system)

V_{AB} is the vector from A to B (and the a axis in the (a,b) coordinate system),

V_{BC} is the vector from B to C (and the b axis in the(a,b) coordinate system).

In summary the above algorithm allows the calculation of a three-dimensional rough triangle based on the edges of an image rough triangle and the relative configurations of vertices of the image and three-dimensional triangles.

3.5 Estimating the geometry of occluded corners by minimisation of least squares error

In some cases significant discontinuities, such as corners, lie in occluded regions of the site. In such cases the simplifying assumption of a largely flat surface between the edges of the occluded region is clearly inappropriate.

It therefore becomes necessary to estimate the position of the corners from whatever information is available. Various algorithms were derived and implemented in order to solve the problem of estimating occluded geometry. Of these only new algorithm based on minimising least squares error (described immediately below) proved useful. Two other algorithms, namely transformation of coordinate system (see §1) and reverse parallax (see §2) were explored and to some extent implemented but gave unsatisfactory results.

Briefly the *minimisation of least squares error* method calculates the positions of discontinuities in three dimensions by finding surfaces which lie between them which i) are most consistent with the shape (in the casual photograph) of the discontinuity and ii) are most consistent with the known geometry in the non-occluded part of the scene.

The following is known or may be assumed in order to estimate the position of the occluded corner (see Figure 4):

- The positions in two-dimensions of the pixels which make up that corner on the casual photograph are known (as found by the edge detector)
- The positions in three-dimensions of the *occlusion boundary*[11] within which the corner falls are known from the photogrammetric survey.
- The positions of the end points of the corner, namely the top and bottom of walls are approximately known (from the rough survey or existing plan of the site)
- It may be assumed that the geometry of the regions between corners may be approximated by smooth surfaces, specifically that they can be approximated by a polynomial of some degree.

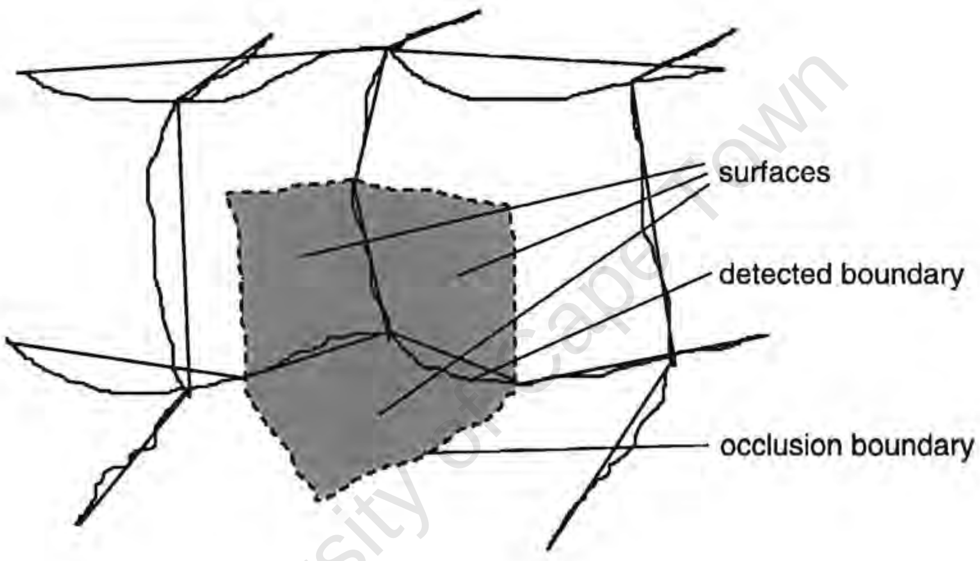


Figure 4 shows an occluded area (shaded), occlusion boundary and detected boundary

A cubic curve has been used to approximate the surface, as this is the lowest order polynomial which allows the surface to have both convex and concave elements. (A parabolic curve would allow a concave or convex surface but not both within the same surface. This is the same motivation for the use of cubic-splines for approximating curves and surfaces)

At a high level the algorithm to compute the locations in three-dimensions of the pixels making up the occluded corner is as follows:

1. The casual photograph is "calibrated" using the six points corresponding to the vertices of the pair of adjacent quadrilaterals which are joined along the occluded corner. These calibration points correspond to the end points of the occluded corner in question and the end points lying on the occlusion boundary (see Figure 5). Calibration is done using the pinhole camera model described in §3.1.

11 the boundary between a) those regions sufficiently visible to the photogrammetric equipment to allow their position to be determined accurately and b) the occluded region. This may not be the actual edge of an occlusion but could be the bounds of accuracy of the photogrammetric technique, typically due to the orientation of the surface relative to the camera.

2. These calibration parameters thus extracted is used to calculate a *pixel ray* for each pixel along the corner. The so-called extrinsic parameters of location and orientation of the camera are used to calculate the origin (camera position) and direction (dependent on pixel location in the image) of the pixel ray. By definition the pixel ray is the line in three-dimensions on which the scene feature represented by a pixel must fall.
3. The actual locations in three-dimensions of the points corresponding to corner pixels in the image must lie at the intersection of a) the pixel ray and b) the pair of surfaces adjacent to this corner (see Figure 6). The problem of finding the location of the edge may thus be treated as an error minimisation, where the error is the sum of the distances along each pixel ray between adjacent surfaces. The error is due to the fact that the surfaces are smooth; being approximated using polynomials rather than having the having the irregular shape of real surfaces. The parameters of the polynomial for each surface are calculated using Newton iteration by minimising the sum of least squared distance between the intersection of the pixel ray and the adjacent surfaces.

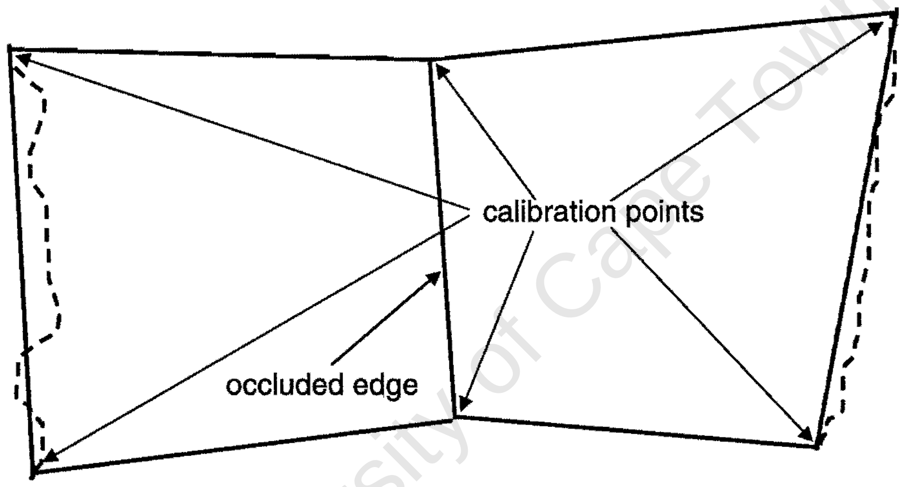


Figure 5 showing the calibration points located in the corners of the quadrilateral grid made up of the two quadrilaterals adjacent to the occluded corner. The occlusion boundary is shown as a dashed line.

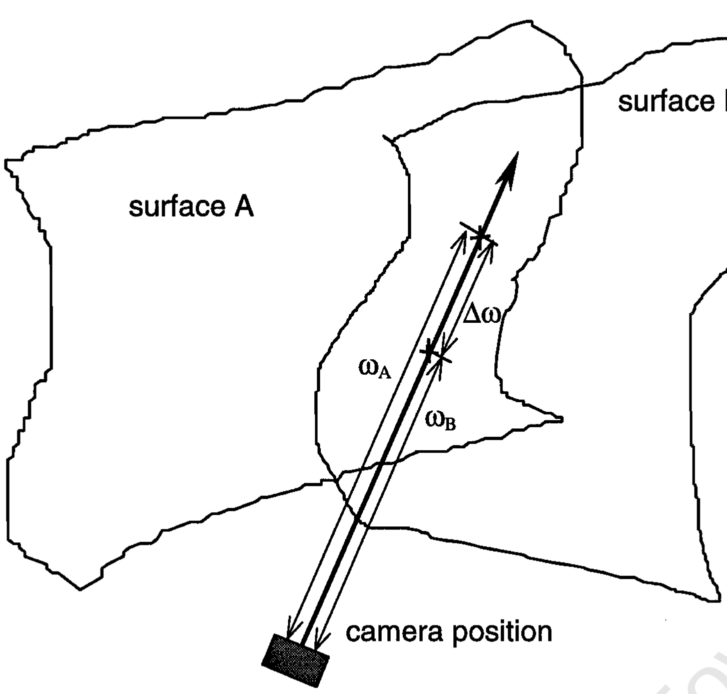


Figure 6 showing the intersection of pixel ray originating at the camera position and adjacent surfaces. The distance between the points of intersection with each of the surfaces is $\Delta\omega$.

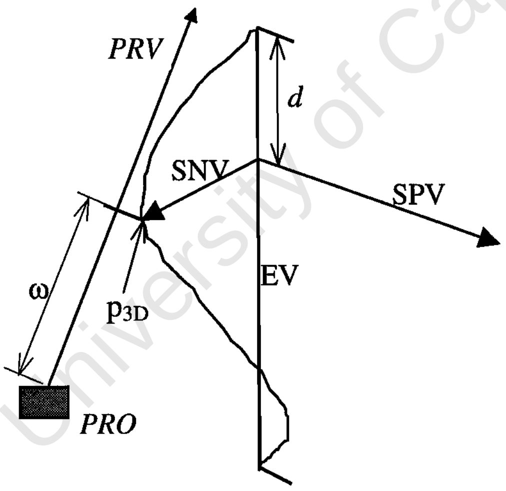


Figure 7 showing a pixel ray intersecting with the cubic surface at distance w. Also shown are the edge vector (EV), the surface parallel vector (SPV) and the surface normal vector (SNV).

The location of surface in three-dimensions of the point corresponding to each pixel in the image is calculated according to the following system of equations, the derivation of which may be found in Appendix B . The formulation is intentionally kept in polynomial form, mostly because this allows us to the flexibility to change the equation representing the surface curvature (from cubic to quadratic for example). It also separates out those terms which are pre-calculated (per edge and per pixel) from those which change with every iteration.

The location of p_{3D} , the point on the surface in three-dimensions corresponding to an image pixel is (see Figure 7)

$$p_{3D} = PRO + \omega\, PRV \tag{1}$$

where

PRO is the pixel ray origin, that is the camera position
PRV is the pixel ray vector, a unit vector originating at the camera position in the direction of the feature corresponding to the pixel and
ω is the distance along the pixel ray vector to the surface.

The intersection of the pixel ray and polynomial surface (see Figure 7) occurs at a distance ω from the pixel ray origin along the pixel ray vector, that is from the camera position, where

$$\omega = \frac{-(P^0_{edge}(d)^0_{xz} + P_{surface}(d)P^1_{edge}(d)_{xz}) - (P^0_{edge}(d)^0_{xy} + P_{surface}(d)P^1_{edge}(d)_{xy})d}{PRV_y P_{parallel\,vector}(d)_x + PRV_x P_{parallel\,vector}(d)_y} \tag{2}$$

The polynomial to solve for d is

$$P^0_{pixel}(d) + P_{surface}(d)P^1_{pixel}(d) = 0 \tag{3}$$

which is of degree five; or two more than the degree of the surface approximating polynomial $P_{surface}(d)$ because the degree of both the $P_{pixel}(d)$ polynomials is two (as is shown below).

$P\ (d)$ are all polynomials in d and are calculated as follows:

$$P^0_{edge}(d)_{xz} = P_{parallel}(d)_z - P_{parallel}(d)_y \tag{4}$$

where

$$P_{parallel}(d)_z = (Cpv^0_y + Cpv^1_y d)(x_{edge\,start} + x_{edge\,vector}d - x_{pixel\,ray\,origin}) \tag{5}$$

and

$$P_{parallel}(d)_y = (Cpv^0_x + Cpv^1_x d)(y_{edge\,start} + y_{edge\,vector}d - y_{pixel\,ray\,origin}) \tag{6}$$

$$P^1_{edge}(d)_{xz} = P_{parallel}(d)_{xz} - P_{parallel}(d)_{xy}, \tag{7}$$

where

$$P_{parallel}(d)_{xz} = (Csn^0_x + Csn^1_x d)(Cpv^0_y + Cpv^1_y d), \tag{8}$$

and

$$P_{parallel}(d)_{xy} = (Csn^0_y + Csn^1_y d)(Cpv^0_x + Cpv^1_x d), \tag{9}$$

The coefficients $Cpv^{n}_{x,y\,or\,z}$ and $Csn^{n}_{x,y\,or\,z}$ describe the surface of the underlying quadrilateral in terms of its parallel vector (pv) and surface normal (sn) respectively. The value of n represents the power of the d term of the polynomial to which the coefficient applies. Thus Cpv^0 applies to the constant (ie d^0) term and Cpv^0 applies to the d term.

Cpv^0 is the vector from the start of one edge to the start of the edge on the opposite side of the quadrilateral, that is the vector which corresponds to the top edge; and Cpv^1 is the vector across the bottom of the quadrilateral less the one across the top. Thus the sum of Cpv^0 and $Cpv^1 d$ where d is equal to 1.0 (the bottom end of the edge) is the bottom edge vector.

Similarly Csn^0 corresponds to the vector perpendicular to both the edge vector and the surface parallel vector and is calculated using the cross product of these vectors. Csn^1 has a value so that the sum of Csn^0 and $Csn^1 d$ is the surface normal vector at the bottom of the edge.

$$P_{surface}(d) = a(d^3 - d) + b(d^2 - d) \tag{10}$$

is the polynomial describing the surface where a and b are the coeffients which define the cubic curve passing through the start- and end-points of the edge.

PRV_x and PRV_y are the x and y components respectively of the normalised pixel ray vector as calculated previously.

The polynomials $P_{parallel}(d)$, which encapsulate the surface orientation of the underlying quadrilateral are calculated as follows:

$$P_{parallel}(d)_x = (Cpv^0_x + Cpv^1_x d) \tag{11}$$

and

$$P_{parallel}(d)_y = (Cpv^0_y + Cpv^1_y d) \tag{12}$$

The polynomials $P_{pixel}(d)$, which are pixel location dependent, are calculated as follows:

$$P^0_{pixel}(d) = PRV^2_x P^0_{edge}(d)_{x^2} + PRV_x PRV_y P^0_{edge}(d)_{xy} + PRV_x PRV_z P^0_{edge}(d)_{xz} \tag{13}$$

where

$$P^0_{edge}(d)_{x^2} = P_{parallel}(d)_{yz^2} - P_{parallel}(d)_{y^2z}, \tag{14}$$

$$P_{parallel}(d)_{xy^2} = (Csn^0_z + Csn^1_z d)(Cpv^0_y + Cpv^1_y d),$$

$$P_{parallel}(d)_{y^2z} = (Cpv^0_y + Cpv^1_y d)(z_{edge\,start} + z_{edge\,vector} d - z_{pixel\,ray\,origin}),$$

$$P^0_{edge}(d)_{xy} = P_{parallel}(d)_{z^2} - P_{parallel}(d)_{y^2}, \qquad (15)$$
$$P_{parallel}(d)_{z^2} = (Cpv^0_z + Cpv^1_z d)(x_{edge\,start} + x_{edge\,vector}d - x_{pixel\,ray\,origin}),$$
$$P_{parallel}(d)_{y^2} = (Cpv^0_x + Cpv^1_x d)(z_{edge\,start} + z_{edge\,vector}d - z_{pixel\,ray\,origin})$$

$$P^0_{edge}(d)_{xz} = P_{parallel}(d)_z - P_{parallel}(d)_y, \qquad (16)$$
$$P_{parallel}(d)_z = (Cpv^0_y + Cpv^1_y d)(x_{edge\,start} + x_{edge\,vector}d - x_{pixel\,ray\,origin}),$$
$$P_{parallel}(d)_y = (Cpv^0_x + Cpv^1_x d)(y_{edge\,start} + y_{edge\,vector}d - y_{pixel\,ray\,origin}),$$

and

$$P^1_{pixel}(d) = PRV^2_x P^1_{edge}(d)_{x^2} + PRV_x PRV_y P^1_{edge}(d)_{xy} + PRV_x PRV_z P^1_{edge}(d)_{xz} \qquad (17)$$

$$P^1_{edge}(d)_{x^2} = P_{parallel}(d)_{xyz^2} - P_{parallel}(d)_{xy^2z}, \qquad (18)$$
$$P_{parallel}(d)_{xyz^2} = (Csn^0_y + Csn^1_y d)(Cpv^0_z + Cpv^1_z d)$$
$$P_{parallel}(d)_{xy^2z} = (Csn_z + Csn^1_z d)(Cpv^0_y + Cpv^1_y d)$$

$$P^1_{edge}(d)_{xy} = P_{parallel}(d)_{xz^2} - P_{parallel}(d)_{xy^2}, \qquad (19)$$
$$P_{parallel}(d)_{xz^2} = (Csn^0_x + Csn^1_x d)(Cpv^0_z + Cpv^1_z d),$$
$$P_{parallel}(d)_{xy^2} = (Csn^0_z + Csn^1_z d)(Cpv^0_x + Cpv^1_x d)$$

$$P^1_{edge}(d)_{xz} = P_{parallel}(d)_{xz} - P_{parallel}(d)_{xy}, \qquad (20)$$
$$P_{parallel}(d)_{xz} = (Csn^0_x + Csn^1_x d)(Cpv^0_y + Cpv^1_y d),$$
$$P_{parallel}(d)_{xy} = (Csn^0_y + Csn^1_y d)(Cpv^0_x + Cpv^1_x d).$$

The coefficients $Cpv^n_{x,y\,or\,z}$ and $Csn^n_{x,y\,or\,z}$ and the formula for $P_{surface}(d)$ have been explained in the discussion of the formula for ω above.

3.5.1 *Relative influence of boundaries*

The shape of each surface is calculated by minimising the sum of least squared errors of the points lying along their connections with adjacent surfaces (these connections corresponding to the inter-segment boundaries in the casual photograph). There are, however, two very different types of connections: those lying *within* the occluded region and those on the *boundary* of this region. The difference between them being that in the latter case the true position is known from photogrammetric survey whereas in the former only the position of the end-points is known.

It would thus seem reasonable that the calculation of a surface which has a *within* connection (ie location unknown) at one end and an occlusion boundary (ie location known) at the other should assign greater significance to errors at the known end. Conceptually this is like making the bond between the surface being calculated and known geometry stronger the bond with an unknown surface. It is thus possible for the user of the tool to adjust the relative stickiness of known boundaries.

3.5.2 Closing the gaps between curved surface segments

As in the case of planar triangle segmentation described in §4.8, gaps occur between adjacent surfaces. The errors (distance between the intersection of the pixel ray and the adjacent surfaces) are, however, much smaller than in the case of the planar triangle because the surface has been allowed to deform precisely in order to minimise these gaps (or at least the sum of their squares).

In a fashion similar to that used for planar triangles the gap has been closed by calculating a single point from the coordinates of the two calculated positions of the image pixel (that is due to its location on each surface). However, rather than simply being the mid-point the location is calculated as a weighted average of the two locations; the weights being inversely proportional to the widths of the surface in question. Thus if the surface was narrow it would be distorted by a relatively small amount in closing the gap.

It is recognised, as was the case when closing gaps between planar rough-triangles, that this mechanism for calculating a new location as an average of the locations on adjacent segments is inexact. However, in this case the fact that surfaces are not planar but have been allowed to deform implies firstly that the gap being closed is small; and secondly that as this deformation is to a nearly correct location that any error generated is small.

3.6 Texture adjustment

Due to differences in lighting conditions between casual photographs and photogrammetrically captured images it is necessary to adjust casual photographs in order that they match the rest of the model better (particularly along the edges of patches). Typically it is necessary to adjust intensity, contrast, colour saturation and colour balance. Algorithms have been utilised for this purpose based on adjusting the values of each pixel as follows:

> *Intensity* - the intensity of each pixel is adjusted by separately increasing (or decreasing) the red, green and blue (RGB) sub-values by an equivalent proportion of their initial values. For example a pixel with initial RGB values of 32, 64 and 128 will after an intensity increase of 50% have RGB values of 48, 96 and 192.
>
> *Contrast* - a grey-scale value is calculated for each pixel along with an average value for the entire image. The RGB sub-values are then adjusted so as to increase (or decrease) the difference between the grey-scale value of that pixel and the average by the same proportion for all pixels.
>
> *Colour saturation* - once again a grey-scale value of each pixel is calculated. Adjustment is achieved by changing the difference in value between each RGB component and the grey-scale value (effectively the average of the three RGB sub-values) of that pixel.
>
> *Colour balance* - each of the RGB sub-values per pixel are adjusted independently

3.7 Geometry calculation algorithms which proved deficient

During the course of designing and implementing the tool various algorithms for calculating geometry in occluded regions were devised and found to be flawed or incomplete for a variety of reasons. Descriptions of the most pertinent of these techniques are included in order to alert researchers as to the weaknesses in the approaches and as additional motivation for the paths chosen. All the techniques relate to the single most difficult aspect of this project, namely the calculation of the shape in three-dimensions of the segments making up occluded regions. The techniques which were considered (and in some cases implemented) but found to be flawed are:

Segmenting the occluded region into planar rough-triangles[12]. The complexity of the tool can be reduced by assuming that the triangle-based segments that make up the occluded area are planar. This assumption obviates the need to estimate the bi-cubic surface geometry, a significant aspect of the task at hand. The location of each plane in three-dimensions can be calculated simply from the locations of the user entered vertices. Then the assumption that the segment is planar permits a fairly simple calculation of the three-dimensional location of each pixel along the edges of the segment. A more detailed description of the approach has been provided in §3.4.

The weakness of this approach was found to be the fact that gaps appeared between segments whenever the segments were non-planar (practically always). Each boundary on the casual photograph is an edge of a *pair* of adjacent segments; the calculated position of the same pixel along the boundary of one of the segments will differ from that calculated for the other segment (due to the different orientation of the underlying plane). This difference in calculated position for the same source pixel results in gaps or overlaps between adjacent segments in the three-dimensional view (see Figure 22 on p52).

It is possible to close these gaps by re-calculating a single position for each edge pixel based on the two positions as calculated using the assumption of planar segments, for example the mid-point or the two positions. This gives visually acceptable results (that is no gaps) but places the edge at an arbitrary "depth"; although the location of the pixel in three-dimensions must lie on the line passing through the two planar calculated positions, it could be anywhere on this line (we are assuming that it lies at the mid-point between the two planes).

Reverse parallax. This approach develops on the planar-rough-triangles technique described above. Briefly, it exploits the fact that *four* different three-dimensional positions can be calculated for each image pixel (based on four different underlying planar triangles – two triangles can be generated for both of the quadrilaterals adjacent to an edge by using a different diagonal to make-up the triangle). The correct position in three-dimensions is at the intersection of the two lines which correspond to a pair of planar calculated points each. A more detailed description of the approach is included in Appendix B .

Briefly, the weakness of this approach was found to be that it is based on mutually exclusive assumptions, namely that the rough-triangles are planar (to generate the four source points) and that they are not (so that the points don't always coincide).

[12] The segments are said to be triangle-based because each corresponds to a set of three vertices as entered by the user. The edges between these vertices, however, are not straight as would be the case in a regular triangle but are the "rough" boundaries as detected in the casual photograph.

Transformation of the coordinate system. In order to provide an explicit linear set of equations defining the locations of the edge points which could be solved relatively easily, a simplifying transformation of the coordinate system was considered. Essentially the three-dimensional positions of all points lying on edges and the pixel-ray (the line in three-dimensions along which the point must lie) are transformed into a two-dimensional coordinate system relative to the underlying orientation of the edge and adjacent sides. A set of simultaneous linear equations, having the distance along the pixel-ray from the viewer of each point as the unknown, is generated and solved. A more detailed description of the approach is included in Appendix B .

As described more fully in Appendix B (Transformation of coordinate system) it became clear that the reduction in dimension upon which this technique is based assumes that the edges are in the same orientation as the image plane. That is the entire edge is at the same depth in the image, which is patently false.

Chapter 4 Design of occlusion-filling tool and example of its use

This chapter describes the design and of the tool built together with an example of its use; it is introduced with a description of the requirements to be met, the design criteria applied and followed a detailed decomposition of the functionality provided.

As previously described the problem being addressed is the filling-in of three-dimensional models of archaeological sites where data is missing due to occlusions in the images captured in photogrammetric surveys. Such occlusion is always present to some degree depending on the complexity of site topography (more complexity implies more occlusion) and the number of photogrammetric images captured (more images from different, well selected positions results in less occlusion). The data to be used for filling-in must be extracted from so-called casual photographs of the site.

4.1 Requirements and Design Criteria

In order to address this problem adequately the tool should satisfy the following requirements:

a) Provide results that are accurate in terms of geometry and texture. As discussed previously the use of three-dimensional models for scientific research and teaching requires a high level of accuracy.
b) Be efficient and practical to use. The primary motivation for the creation of such a tool is to eliminate the cost of a more complete photogrammetric survey of the site. The use of this tool must be materially cheaper than carrying out a more comprehensive photogrammetric survey; particularly as the tool's output will be inherently less accurate than that of photogrammetry.
c) Make the *greatest* use of information available, namely photogrammetrically captured geometry and texture along with casual photographs and the *least* use of assumptions about the site. Compared with architectural model building, for example, where assumptions of rectilinearity and of planar surfaces may be made with justification, the geometry of archaeological sites is seldom regular.
d) Be generally applicable within the domain of archaeology. The tool should not be limited to particular types of archaeological sites (or site), particular photogrammetric technologies or types of casual photographs.
e) Deal well with the sorts of problems due to occlusions. Occluded regions are typically due to site features such as niches and the projection of columns and walls onto other parts of the site. They are thus expected to have irregular boundaries, include edges and corners and have fairly balanced extent in all directions. For example, the simpler problem of filling-in the long and narrow regions due to scratches on images is not addressed here.
f) Usable by all members of the archaeological modelling community, namely photogrammetrists and archaeologists, who are currently able to use existing tools for creating geometry from sets of images.
g) Usable on a wide range of computer platforms. The tool should not be limited to a single or specialist group of computer platforms. A description of and motivation for the computer platform which satisfies this requirement is included in §4.4.

In addition to the requirements, a list of criteria to be taken into account during the design process was specified. The difference between requirements and design criteria being that:

requirements must be satisfied; whilst design criteria simply inform decisions made during design. The following criteria were placed on the design:

a) The tool should be complete, that is should carry out all the steps in the process which transforms digital image and geometry into a virtual reality model.
b) Use existing and widely accepted standards, technology, terminology in order to make the tool available to a broad range of users and applicable to most geometric and image data sources.
c) *Wherever available* use reasonably mature technologies, that is technologies which have previously been applied effectively by researchers other than the originators. The primary purpose of this project is to provide a tool based on the combination of existing technologies and not to test emerging techniques. It was thought prudent to use technologies which had at least been implemented successfully by disinterested third parties
d) The eventual usability of the tool should be considered and general usability based principles such as affordances, feedback, user interface standards etc applied.
e) The tool should be general in the sense that it should not be restricted to a specific site or set of casual images.
f) Implementable in about six man-months. Due to the fact that this work is being done as part of an MSc by coursework and dissertation the tool had to be implementable reasonably quickly. This emphasises the need to use mature technologies.

4.2 Functionality

In essence the tool built estimates geometry from casual photographs supplemented by known geometry in the photographed scene; and provides texture extracted from the casual to be draped over the geometry generated. Correspondences between features in the photograph on the one hand and known geometry on the other are used to generate information to supplement that obtained using photogrammetric techniques. The information gathered is used to fill in those parts of a virtual reality model of the site which cannot cost effectively be captured by traditional means.

At a very high-level the functions of the tool may be divided into the following steps (see Figure 8 for process flow diagram):

1. Load the input data (that is the known geometry and casual image data),
2. Calibrate[13] the casual photograph,
3. Segment those parts of the casual photograph which correspond to occlusions in accordance with their internal features (such as edges and corners),
4. Compute the geometry within the occlusions,
5. Drape casual photograph based texture data over the geometry thus calculated
6. View and adjust the casual photograph texture data to match surrounding texture data and
7. Save the results

[13] Calibration is defined as the calculation of parameters which describe the camera and its position in the scene. These parameters can then be used to relate the positions of features in the scene to their representation in an image captured by the camera.(See §3.1)

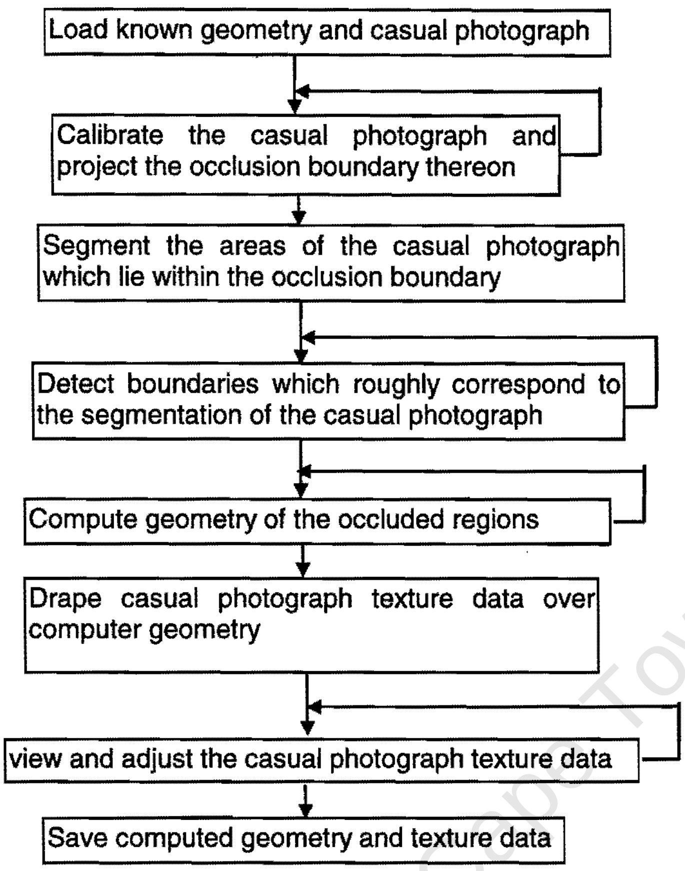

Load known geometry and casual photograph
Calibrate the casual photograph and project the occlusion boundary thereon
Segment the areas of the casual photograph which lie within the occlusion boundary
Detect boundaries which roughly correspond to the segmentation of the casual photograph
Compute geometry of the occluded regions
Drape casual photograph texture data over computer geometry
view and adjust the casual photograph texture data
Save computed geometry and texture data

Figure 8. Process flow diagram showing the use of the tool implemented at a high level.

At a more detailed level the abovementioned functionality is provided as follows:

1. *Load the input data.* Facility to load and view both the casual photograph held in either a lossless image file format (TIFF, GIF & PNG) or lossy file format (JPG) and the three-dimensional geometry from a CAD data file (AutoCAD .DXF). The availability of this broad range of widely used (and *de facto* standard) formats renders the tool useful to a large community of users (arguably all) in photogrammetry and archaeological modelling.
2. *Calibrate the casual photograph* using the classical pinhole camera model using at least six user-selected correspondences between scene and photograph. The exploitation of calibration of the casual images is an important and novel feature of the tool implemented because it leverages the fact that casual photographs will include many features which have been accurately surveyed. The inclusion of a calibration function allows the extraction of geometric information from the vast majority of casual photographs; excluded is the use of photographs taken using a short focal length due to radial lens distortion. A detailed description of the calibration technique implemented is included in section §3.1.
3. *Segment the casual photograph* to provide the building blocks from which the geometry is calculated and corresponding patches of surface texture extracted (see Figure 12 in §4.3.4). This functionality is comprised of the following:
   a) The calibration parameters extracted above are used to project the boundaries of the occluded area (in the geometry) onto the casual photograph. This boundary forms the point of integration between the photogrammetrically gathered texture and geometry and that obtained from the casual photograph. The concept of projecting the boundaries of occlusions onto images as a means of segmentation of the image and providing the boundary for

integration of texture types (casual photograph with photogrammetric) is a novel aspect of this work.

b) Allow the user to indicate corresponding points (seed points) on the casual photograph and the three-dimensional geometry. The seed points are typically positioned at the ends of discontinuities in the surfaces (usually corners). The line segments between seed-points are the user's approximation of the location of significant discontinuities in the geometry (typically corners where planes intersect). The *sequence* in which the seed points are selected is used further to generate a grid (of triangles or quadrilaterals which are made visible to the user). This grid segments the area being filled into approximately planar regions.

c) A suite of user selectable/combinable filters to remove, in particular, noise from the photographic image in order to improve the edge detection phase. The primary motive for providing these filters is to allow the use of a broad range of casual photograph image qualities. These images may include scratches, the edges of shadows and fine-scale texture which should be removed in order to improve the boundary detection phase. Furthermore, the filters need to be user adjustable to allow them to be specifically tuned to different image sets. The filters provided are the well-known low bandpass, Gaussian, difference of Gaussian and their implementation is described more fully in §3.2.

d) A user adjustable boundary detector to find the actual paths of discontinuities between seed points. The algorithm implemented is based on the so-called active contour or snake model; here the boundary is found by minimising external energies (such as image intensity gradient) and internal energies (such as curvature). The algorithm was selected because it permits the user to adjust the relative importance of the energy terms (which in turn makes it usable for a wide variety of site and casual photograph types) and it allows the user to set boundary start and end points. A detailed description of the boundary detection technique implemented is included in §3.3.

4. *Compute the geometry*. The three-dimensional shapes of the approximately planar segments of the occluded geometry are calculated. Briefly, this function is achieved by performing a least squares error fit for the three-dimensional location of each edge pixel under the constraints of i) the surfaces being approximately bi-cubic and ii) the location in the scene of the said pixel lying on a three-dimensional line (pixel-ray) consistent with its position on the casual photograph. This functionality (which approximates the shape of surfaces adjacent to each edge by minimising the sum of least squared differences between the intersection of each pixel-ray which makes up an edge and the pair of surfaces adjacent to that edge) is another entirely novel aspect of this project. The algorithm implemented makes the greatest use of information available (namely surrounding geometry, edge shape in the casual photograph and user segmentation of the image) and the least use of assumptions (the surface is assumed to be bi-cubic, which is in itself not a very constraining). Furthermore, being designed for regions which include edges and corners it addresses the sorts of problems typical in occlusions. A more detailed description of the geometry calculating technique implemented is included in section §4.3.7 .

5. *Drape texture data*. Functionality to drape the image data for each region (as delimited by edges detected on the photograph) onto its corresponding patch of bi-cubic geometry as calculated above. This function is designed to use the underlying three-dimensional modelling facilities provided by the programming language.

6. *View and save results*

a) A facility to view the resulting image data draped over the geometry (in 3D from user selected positions and distances) in order to make a subjective judgement as to the realness/accuracy of the result.

b) A means to store the output so that it can be reused in the course of a larger model-building process; or explored with one of many widely available virtual reality viewers (VRML format, the generally accepted file storage standard for virtual reality modelling). More details are provided in §4.10.

In addition to providing *functionality* which satisfies the requirements and complies with the design criteria (as described above), the tool implemented also addresses these requirements and criteria in a more general way. For example the selection of Java and Java3D as programming language allows the use a wide range of computer platforms (see §4.4). IN order to enhance usability features have been included such as visual feedback and the reliance on affordances (sliders, buttons, scroll-bars and the like). Wherever available, mature technologies have been relied upon (calibration, filtering, boundary detection).

The following section shows the functionality of the tool by proceeding though a worked example of its use. This is followed by a detailed description of the functionality as summarised in this section.

4.3 Use

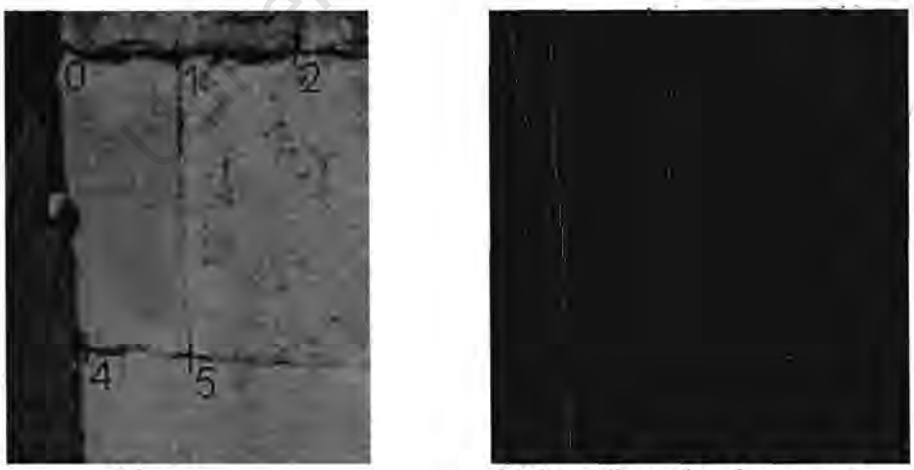
The tool is used in accordance with the steps shown in Figure 8 and supporting text. Each of the steps listed there is accomplished as follows:

4.3.1 Loading the casual photograph and three-dimensional geometry

The casual photograph data and three-dimensional geometry data are loaded using a dialogue boxes based on the Java Utilities' FileChooser available under the standard File item on the menubar. Similarly, the save function is available.

4.3.2 Select corresponding features for calibration

To allow calibration of the casual photograph a minimum of six corresponding features are marked [14] on the geometry and image panels (see Figure 9). Calibration points are blue coloured cross-hairs.



a) image b) three-dimensional geometry

Figure 9 showing the calibration points created by the user as blue cross-hairs on the casual photograph (Figure 9a) and the three-dimensional geometry (Figure 9b). Corresponding points are given the same number.

[14] Calibration points are manipulated using the mouse together with the CTRL key.

4.3.3 Calibrate the casual photograph and project the occlusion boundary

The casual photograph is then calibrated based on the set of corresponding calibration points (see Figure 10). The calibration parameters calculated are then used to project the three-dimensional calibration points onto the casual photograph (shown as the blue circles in Figure 11). The degree to which these circles are centred on the image calibration points (blue cross-hairs in Figure 10) is in an indication of the accuracy of the calibration. Errors are introduced into the calibration calculation by the inaccurate localisation of calibration points.

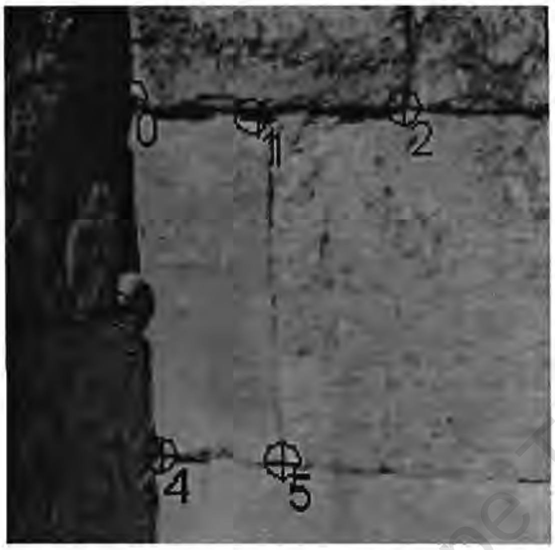
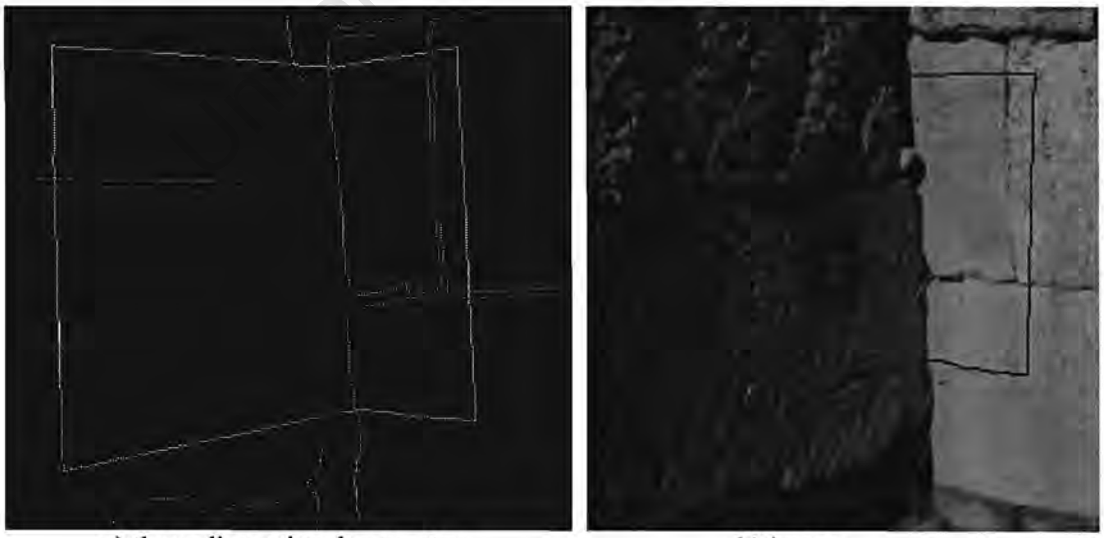


Figure 10 showing the calibration points (blue cross-hairs) each surrounded by the projection of the corresponding point from the three-dimensional geometry (blue circle). The deviation between the centres of the projected circles and the calibration points is an indication of the amount of error in the calibration parameters and serves as visual feedback as to the efficacy of the calibration.

Next the occlusion boundary is projected onto the casual photograph (as shown in Figure 11). The source data from which the occlusion boundary is projected is a nominated set of line segments in the three-dimensional geometry data file (a layer in the AutoCAD .DXF file) which correspond to the edges of what has been captured photogrammetrically.

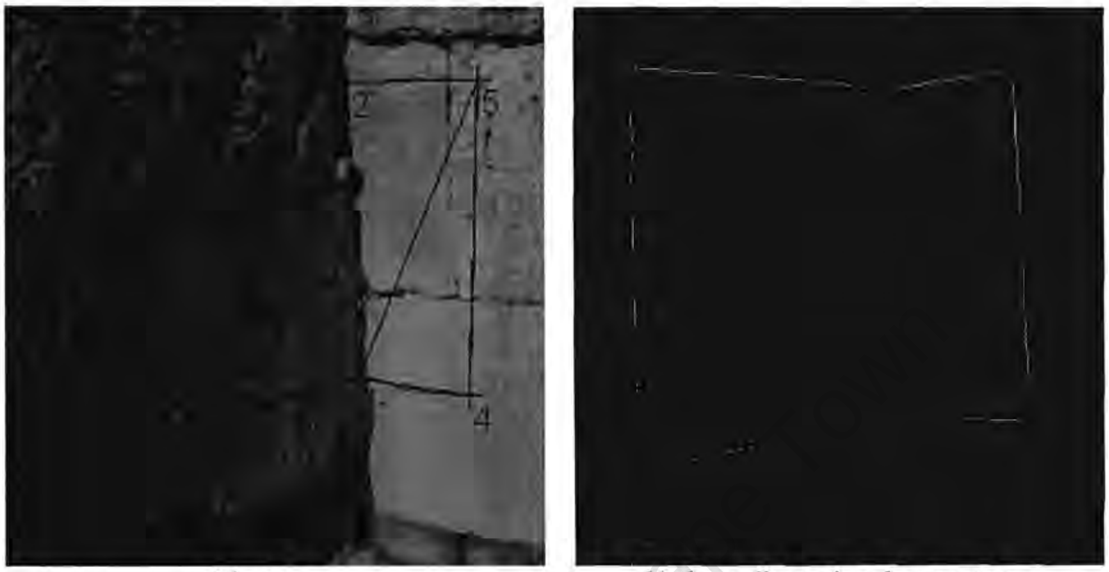


a) three-dimensional geometry b) image

Figure 11 which shows a) the occlusion boundary superimposed onto the three-dimensional geometry (the lime green rectangular shape) and b) the projection of this boundary onto the image (the blue rectangular shape).

4.3.4 Mark correspondences between image and geometry

Corresponding features are marked on the geometry and image panels using a double-click of the mouse. A grid of triangles is generated between correspondences according their selection sequence (Figure 12). The lines between the corresponding points are added automatically by the application. Double-clicking on or near an existing point will *select* that point (rather than create a new point) for inclusion in a new triangle in the grid. Seed points and the lines between them are coloured red.



a) image b) three-dimensional geometry

Figure 12 which shows the boundary seed points created by the user as red cross-hairs on a) the casual photograph and b) the geometry; along with the grid of triangles made up from the lines between seed points.

4.3.5 Filter image data

A control panel is available to set the filtration and edging parameters; and to trigger the calibration, boundary detection and draping functions. The filter parameters and trigger button fill the left half of the control panel (see Figure 13).

The filter parameters which can be set using the control panel are (see §3.2 for a more complete definition of the filter parameters):

- box filter degree (effectively the spatial extent of the filter: for example degree 5 includes a single layer of surrounding pixels in a calculation of filtered pixel value, degree 10 two layers etc)
- Gaussian sigma is the spatial extent of the Gaussian filter
- DoG sigma is the spatial extent of the difference of Gaussian filter
- DoG sigma ratio is used to change "Mexican hat" profile difference of Gaussian filter (a value of 160 best approximates a Laplacian)

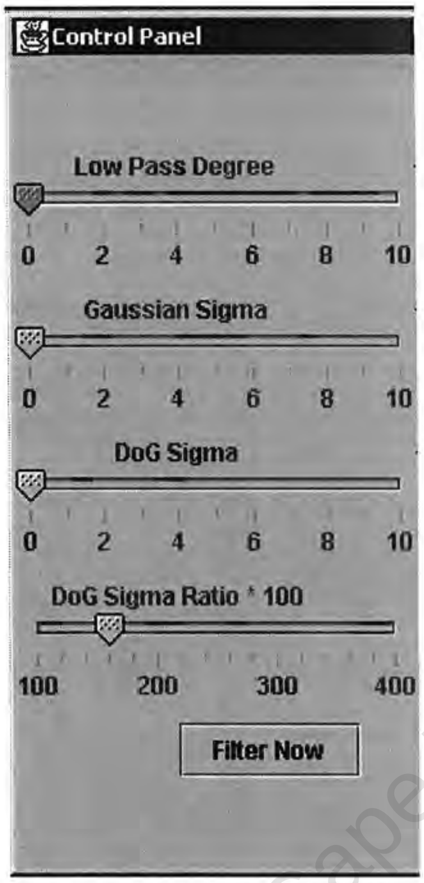
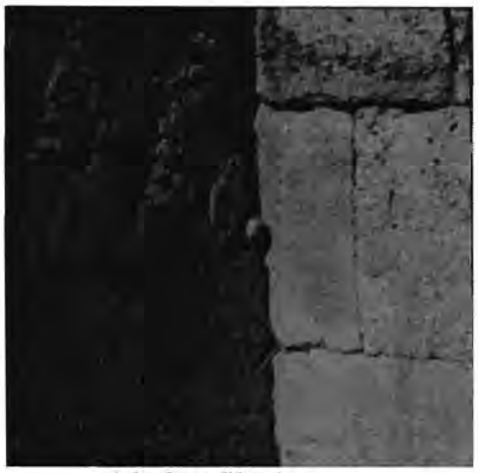
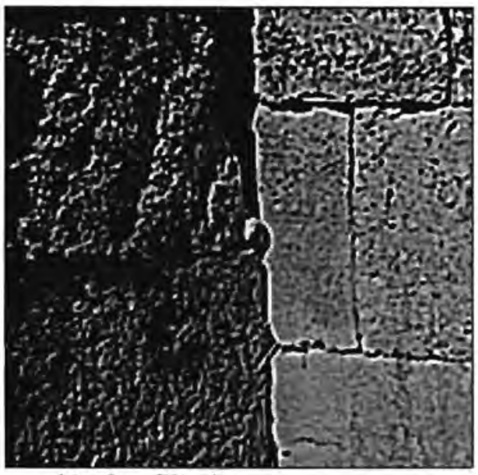


Figure 13 showing that portion of the control panel (the left half) which is used for setting the filtration parameters and triggering the filtering function; the right half of the control (shown in Figure 15) is used to trigger the calibration, boundary detection and draping functions. The filter parameters which can be set are box filter degree, Gaussian sigma, difference of Gaussian sigma (effectively the spatial extent of the filter of each of these thee filter types) and the difference of Gaussian sigma ratio which tunes the difference of Gaussian filter.

The image data is filtered in order to reduce noise and fine-grained texture (see Figure 14)



a) before filtering

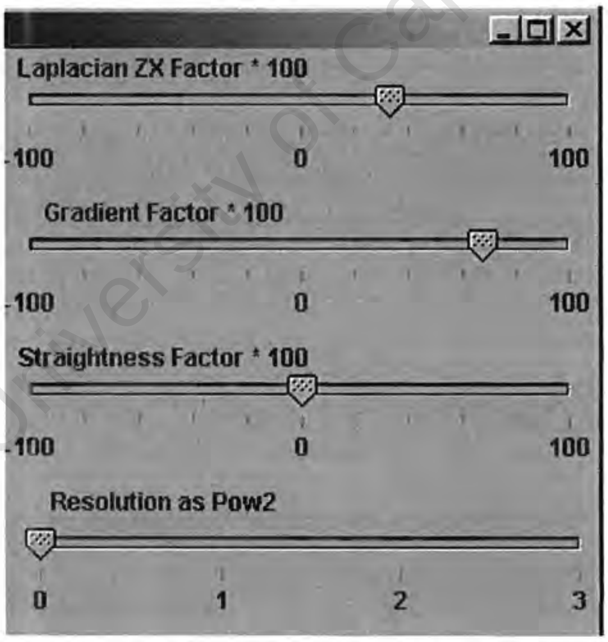


b) after filtering

Figure 14 showing an image before (a) and after (b) difference of Gaussian filtering. Note how the edges have been sharpened.

4.3.6 Detect boundaries

The boundary detection parameters are adjusted (if necessary) using the right half of the control panel (Figure 15). The boundaries are then detected and indicated on the casual photograph in red (see Figure 16)



Laplacian ZX Factor * 100

Gradient Factor * 100

Straightness Factor * 100

Resolution as Pow2

Figure 15. Controls for boundary detection which are included as the right half of the Control Panel (the left half is used for setting filter parameters and is shown in Figure 13). The first three sliders control the weighting factors for edge cost aggregation (namely $\omega_{Laplacian}$, $\omega_{gradient}$ and $\omega_{straightness}$) and the fourth slider controls the resolution of the edge calculation on order to speed up boundary detection on high-resolution casual photographs.

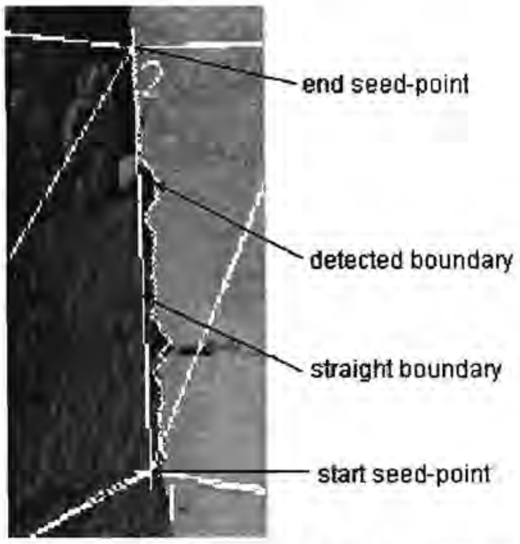


Figure 16 showing the boundaries as detected between seed-points 1 and 2 along with the straight boundaries connecting seed-points.

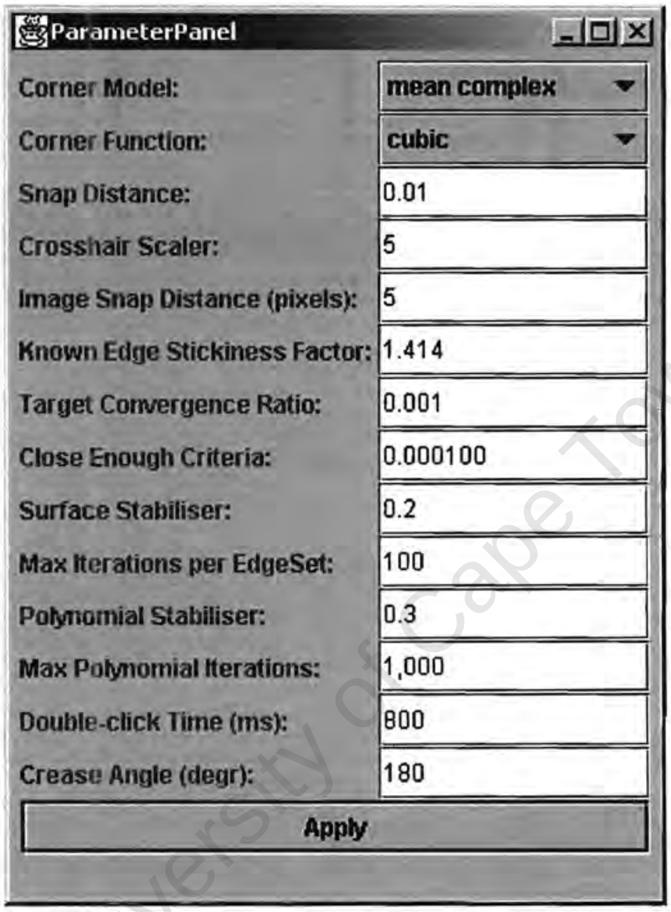
At this point seed points can be moved, filter or boundary detection parameters changed in order to find better boundary with which to proceed to the geometry calculation phase.

4.3.7 Calculate geometry

Geometry calculation is controlled by parameters which can be set using the parameter panel (see Figure 17). Parameters applicable to geometry calculation are as follows:

> Corner Model selects whether the surfaces within the occlusion should be modelled as planar or complex (see corner function) and whether the edges of the patches shown be left separate (as calculated by minimising least squares error) or closed up by calculating a mean value (see §3.5)
>
> Corner Function (applicable only if a *complex* corner model is selected) is used to select whether the surfaces are to be modelled as quadratic or cubic.
>
> Known Edge Stickiness Factor is the weight that the least mean squares error along edges with known geometry should have relative to hidden edges.
>
> Target Convergence Ratio is used to set the point at which the geometry calculation ceases. Once the least mean squares error drops by less that this proportion the calculation is deemed to be complete.
>
> Close Enough Criteria is the permissible error in the calculation of the value of d in equation (see §4.9.1) for each pixel along an edge.
>
> Surface Stabiliser is a value between 0.0 and 1.0 which attenuates the amount by which a coefficient it changed per iteration, the value α in equation 1 in §4.9.1 .
>
> Max Iterations per Edge Set is the maximum number of iterations allows for calculating the hidden occlusion geometry.
>
> Polynomial Stabiliser functions in the same way as the surface stabiliser but for the solution of the polynomial in d (see equation 3 in §3.5).

Max Polynomial Iterations is the maximum number of iterations allowed for each solution of the polynomial in d (see equation 3 in §3.5).



Corner Model:	mean complex
Corner Function:	cubic
Snap Distance:	0.01
Crosshair Scaler:	5
Image Snap Distance (pixels):	5
Known Edge Stickiness Factor:	1.414
Target Convergence Ratio:	0.001
Close Enough Criteria:	0.000100
Surface Stabiliser:	0.2
Max Iterations per EdgeSet:	100
Polynomial Stabiliser:	0.3
Max Polynomial Iterations:	1,000
Double-click Time (ms):	800
Crease Angle (degr):	180

Figure 17 Parameter panel which is used to set the parameters governing the calculation of geometry hidden within occlusion. Parameters applicable to geometry calculation are corner model, corner function, known edge stickiness factor, target convergence ratio, close enough criteria, surface stabiliser, max iterations per edge set, polynomial stabiliser and max polynomial iterations.

4.3.8 Drape the image data over calculated geometry

Draping is initiated using the "Drape Now" button on the control panel. Draping consists of calculating a) the geometry and b) the texture mapping of occluded regions and then inserting these results to the three-dimensional geometry shown in the geometry panel (as shown Figure 18).

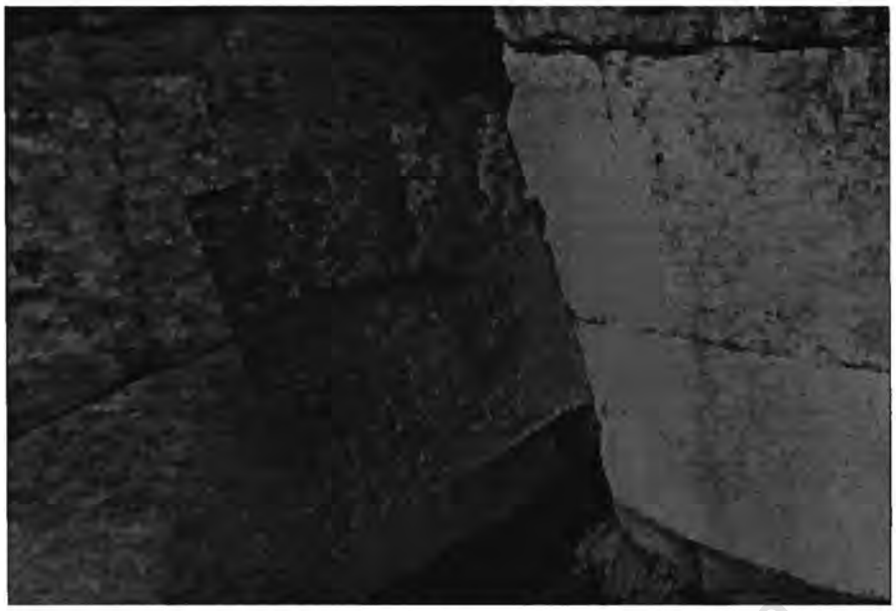
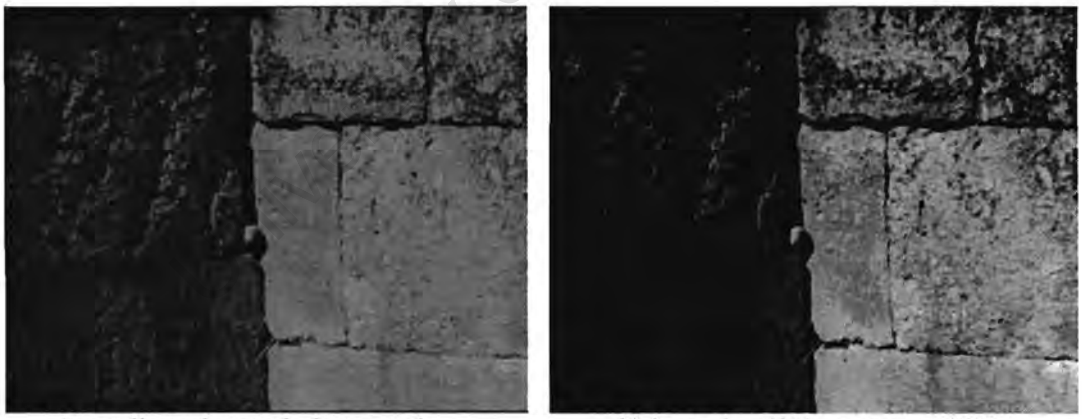


Figure 18 showing texture extracted from a casual photograph draped over a synthetic occlusion in the model.

4.3.9 View and adjust patch texture

In order to allow better matches between occlusion patches and surrounding texture a texture adjuster is available. Figure 19 shows the effect of texture adjustment on a casual photograph. Texture adjustment is controlled using the Texture adjuster control panel (see Figure 20) which allows the adjustment of the image intensity, contrast, colour saturation and colour balance.



a) unadjusted casual photograph b) intensity 50%, contrast 200%

Figure 19 shows the difference between a portion of a) an unadjusted casual photograph and b) the same image with intensity halved and contrast doubled.

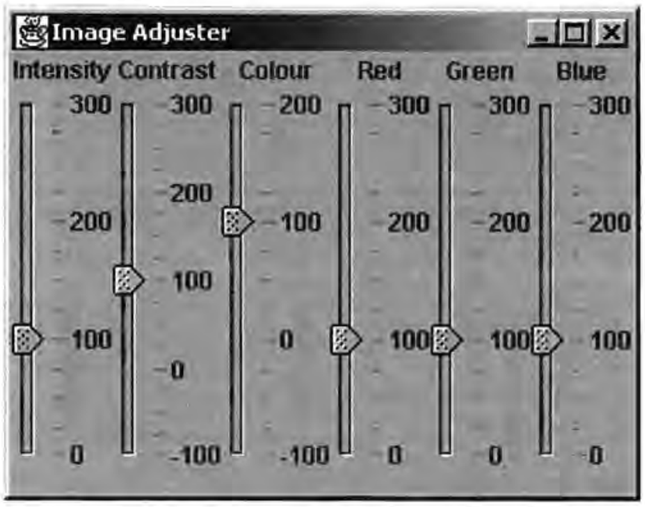


Figure 20. Texture adjuster control panel which is used to adjust the image intensity, contrast, colour saturation and colour balance.

4.3.10 Parameter panel

In addition to its use in setting geometry calculation parameters the Parameter Panel (see Figure 21) can also be used for setting the following general parameters at any time during the application use:

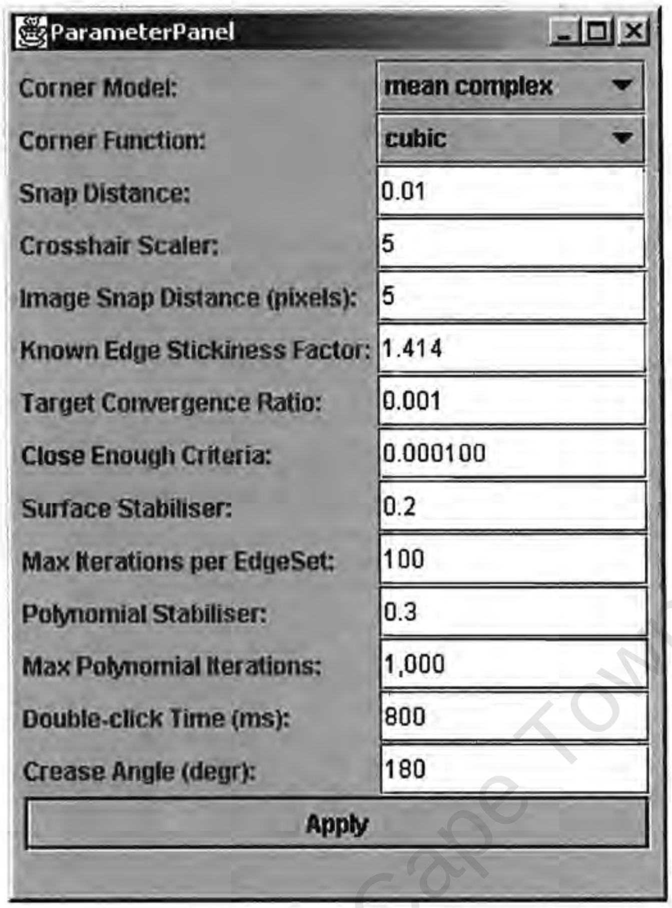
Snap Distance is the distance in units of the geometric model that a mouse click will snap onto a feature.

Cross-Hair Scaler is the size of the cross-hair generated in units of the snap distance

Image Snap Distance is the distance in pixels that a mouse click will snap onto an existing feature marker.

Double-Click Time is the length of time in milliseconds below which two mouse clicks will be recognised as a double-click.

Crease Angle is the angle above which any vertex on a calculated surface will be allowed to crease rather than be smoothed.



Corner Model:	mean complex
Corner Function:	cubic
Snap Distance:	0.01
Crosshair Scaler:	5
Image Snap Distance (pixels):	5
Known Edge Stickiness Factor:	1.414
Target Convergence Ratio:	0.001
Close Enough Criteria:	0.000100
Surface Stabiliser:	0.2
Max Iterations per EdgeSet:	100
Polynomial Stabiliser:	0.3
Max Polynomial Iterations:	1,000
Double-click Time (ms):	800
Crease Angle (degr):	180

Figure 21 Parameter panel which besides being used to set the parameters governing the calculation of geometry can be used to set other general application parameters. These additional parameters are the snap distance, cross-hair scaler, image snap distance, double-click time and crease angle.

4.4 Platform

The selection of a appropriate computing platform (hardware, operating system and programming language are relevant in this case) is important so as not to limit the tool's availability and lifespan while at the same time utilising in-built functions to solve the problem at hand.

It was decided that the tool should be written in Java wherever practicable. Java was selected for the following reasons:

- Portability - code written in Java and "compiled" to Java byte-code will run on any platform which has a version of the Java Virtual Machine available, that is practically all.
- Availability of a comprehensive suite of functions to facilitate the production of the user interface in particular (after all the purpose of this work was not to produce an original type of user interface from the ground up). The functions are accessible to the programmer via a well-documented API.
- Suitability for use over the WWW/Internet - because Java enforces the production of secure and modular code; and can be run from a web browser over the Internet. The educational and collaborative nature of VR modelling of archaeological sites is enhanced by keeping tools web-enabled.
- Availability of built-in functionality for three-dimensional modelling (Java3D) and for the conversion of input and output image formats (Java Advanced Imaging)

- Software costs. Currently Java and the extensions required to build (compilers and APIs) and use this tool developed are free of charge.

Elements of code which were found to be computationally expensive were ported to C[15]. In order to preserve portability, however, a Java version of all C functionality has been preserved (by implication such functionality is very slow under Java). The Java classes have been coded in such a way that the absence of a C version is dealt with automatically, that is invisibly to the user.

The selection of Java obviates any need to select the operating system or hardware components of the computer platform. It suffices to say that the application will only operate in environments which support the Java Virtual Machine and the Java3D and Java Advanced Imaging extensions.

4.5 Calibration and Occlusion Boundaries

An important and novel feature of the tool implemented is that it exploits two unique characteristics of a data set gathered from photogrammetric survey supplemented by casual photographs, namely:

a) Casual photographs will typically include many features which have been accurately surveyed.
b) The edge of the region which we are trying to extract image data for is by definition the occlusion boundary. Furthermore, we know the geometry of the occlusion boundary as it appears in the photogrammetric survey data set (only just).

The first characteristic can be exploited to "calibrate" the casual photograph. Calibration is defined here as the process of calculating parameters which describe the camera and its position which can then be used to calculate the position on the image that any scene feature would take. A full description of the camera calibration algorithm is included in §3.1.

A feature of the calibration algorithm used is that it requires at least six correspondences between three-dimensional scene features and two-dimensional image locations; with increasing robustness of result with increasing number of points. In addition the set of equations cannot be solved if all the points are co-planar (a very remote possibility in archaeological images but possible a problem with architectural images). It was thus decided that the only limit placed on the selection of corresponding points by the user would be a minimum of six, that is the number required as a minimum to solve the set of equations.

Once the image is calibrated any geometric data can be projected onto the casual photograph. In particular the points making up the occlusion boundary (available for the reason described in the second characteristic above) can be projected and later used along with edges detected by other means to bound regions.

4.6 Filtering algorithms implemented

As previously discussed (see §3.2) filtering of the image data captured from casual photographs was expected to be necessary to improve the detection of boundaries.

Filters used need to be both user adjustable (so that they can be tuned for image sets) and be targeted at noise and fine-grained texture without removing image data necessary for locating

[15] the Intelligent Scissors edge detection algorithm was found to be about 25 times faster in C than in Java (all else being equal)

edges. The following filtering algorithms have been implemented (see §3.2): box filter, Gaussian smoothing, and difference of Gaussian.

It is possible to use the above filtering techniques in combination or separately. It is expected that the filters will normally be used separately because it makes little sense for example to combine a Gaussian filter with a difference of Gaussian filter. However, the facility to combine the filters has not been restricted as images may exist which will benefit from combined filtering and the user can select whether or not to actually combine filters. The filtering parameters of spatial extent and, in the case of the difference of Gaussian filter, sigma ratio can be adjusted using sliders.

4.7 Boundary detection implementation

In order to extract patches of surface texture from a casual photograph it is necessary to detect the boundaries of such patches. In addition the shapes of these patches is an input to the calculation of hidden geometry. The detection of boundaries must be user guided and be applicable to pre-filtered image data.
The boundary detection algorithm implemented is based on a technique known as Live-Wire (Mortensen and Barrett 95) and is described in §3.3. This algorithm requires the following as input:

1. post-filter image pixel intensity data
2. a start and end seed point for each boundary to be detected
3. weighting factors for the three cost components (Laplacian zero-crossing, gradient magnitude gradient direction)

Start and end seed points are entered by double-clicking on an image location. The implementation the weighting factors may be set using sliders which have as default values those suggested by Mortensen and Barratt 95 namely:

Laplacian Zero-Crossing	0.43
Gradient Magnitude	0.43
Gradient Direction	0.14

Effectively the cost of a path is the sum of pixel-pair costs for all the pixel pairs along the length of said path. It is possible to calculate the pixel-pair cost in advance of calculating the lowest cost paths (ie eagerly). However, this is wasteful because it implies calculating costs in regions of the image remote from the seed points and thus unlikely to be on the lowest cost path. Therefore calculation of the cost terms is performed in a lazy mode (that is whenever the pixel is adjacent to the end of the of the lowest cost path). This results in the pixel-pair costs of many of the pixel-pairs in the vicinity of the lowest cost path being re-calculated several times.

The lowest cost path is found (via a directed graph search using dynamic programming) by recursive execution of the following steps:

> A pixel (initially the start pixel) is selected for calculation (of cost to adjacent pixel) based on it being the pixel having the lowest cost path to it from the start pixel. The process of selecting the next pixel for calculation is optimised by storing calculated pixels in a cost-to-pixel sequenced list.
>
> The cost to adjacent pixels is calculated as the weighted sum of cost terms described above and these pixels and their costs are stored in the cost-to-pixel sequenced list (which eliminates the search for the next lowest-cost pixel).

When the end pixel is selected for calculation the lowest cost has by definition been found.

As expected, boundary detection calculations were found to be computationally expensive. In order to ameliorate this it was decided a) to port the code into C and b) allow the reduction of image resolution (less pixels between the start and end of a boundary implies a shorter path and therefore less calculation). It was found that the C version was typically 25 times faster than the original Java version when finding the same boundaries.

4.8 Draping image data over planar triangles

Initially it was believed reasonable that the problem under consideration could be simplified by assuming that the regions to be filled-in from casual photographs are made up of a set of planar segments. As will be shown in §5.2.3 evaluation this approach is not suitable in all cases, particularly for highly curved surfaces; it does however provide an effective solution for situations where the surfaces are near to planar.

Because a triangle is the only polygon which is always planar it was decided to allow the user to segment the occluded regions into what are believed to be planar triangles. The tool would then find boundaries on the image corresponding roughly to the edges of the triangles (rough triangles) and then a) project these rough triangles onto the geometry and b) drape/warp the image data onto the triangles.

The process per triangle is as follows:

1. The three corresponding vertices of the triangle are selected on both the casual photograph (which we will call the image) and the three-dimensional representation of the scene (which we will call the geometry).
2. Lowest cost boundaries are detected between the vertices on the images. This provides a list of pixels which make up each boundary.
3. An algorithm for calculating rough triangles (see §3.4) in the geometry based on the above data is applied.

The data thus calculated (effectively a canvas of the correct shape upon which to drape image data) is passed to the texture draping facilities available in Java3D as an array of three-dimensional edge point coordinates with corresponding image positions. The draping and necessary warping is performed by Java3D.

4.8.1 Connecting adjacent triangles

As discussed previously each occluded area is segmented into a number of planar rough-triangles. When these planes, as calculated in the previous section, are rendered in three-dimensions it is found that *gaps* exist between adjacent triangles (see Figure 22). The gaps or cracks differ in size depending on the following:

a) the degree to which the boundaries of the underlying geometry are in fact non-planar.
b) the difference in orientation between the adjacent planes and
c) the differences in amount of warp in adjacent triangles. As discussed in § 3.4 the amount of warp is a function of the difference in shape between the triangle in the image and the triangle in the three-dimensional model. The gap is a result of the fact that the same edge pixel in the image is being projected into three-dimensions by differing transformations due to adjacent triangles.

In essence these gaps are due to the fact that each edge pixel in the image lies on the edge of two adjacent triangles in three-dimensions. Because the positions of these two points are the result of two differing transformations the will most probably have different coordinates.

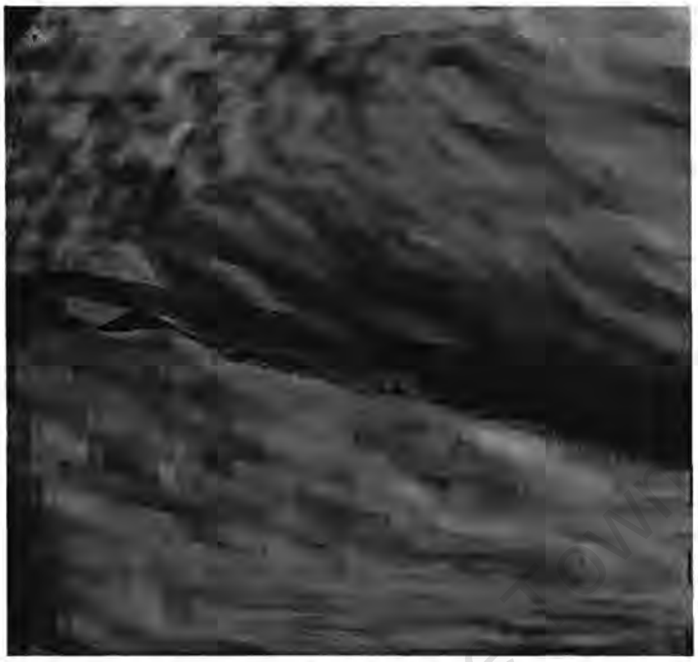


Figure 22 showing a "crack" between two surfaces due to the assumption that each surface is planar.

The only means of closing such a gap is to calculate a single value for each pair of edge locations calculated for each pixel along the edge; the simplest approach is to calculate a common point based on the mid point between the two locations in three-dimensions of the same boundary pixel. The result of this is that the rough triangles in the geometry are now slightly non-planar but the crack artefact has been removed. This gives visually acceptable results (that is no gaps) but places the edge at an arbitrary point (in this case mid-way) between the planar-calculated pixel locations.

Clearly, the errors introduced by the assumption of planar rough-triangles increase with an increase in curvature of the surface. Conversely, because many surfaces modelled will be close to planar and recognising the simplifications allowed by the assumption of planar surfaces, this means of calculation has been made available as an option to users of the tool.

4.9 Minimising least mean square error to estimating the geometry of occluded corners

Typically the regions lying within occlusions cannot be assumed to be planar, that is they contain significant discontinuities. A geometry estimation algorithm based on the minimisation of least squares error has been devised (see §3.5) to address this type of scenario.

At a high level the algorithm is implemented as shown in Figure 23.

```
repeat
    for values of coefficient near the current value
        for each edge ∈ edges around an within an occlusion
            for each pixel ∈ pixels along an edge
                repeat
                    calculate $d$ as root of polynomial in $d$
                until change in $d$ is sufficiently small
                calculate $\omega$
            calculate $\Sigma$ least mean square ($\Delta\omega$ per pixel)
        calculate error at coefficient value = $\Sigma$ ($\Sigma$ least mean square ($\Delta\omega$ per pixel))
        calculate change in error per change in coefficient value
    calculate new coefficient values
until convergence is complete (difference coefficient value between iterations is sufficiently small)
```

Figure 23 High-level structure of least mean squares geometry estimation implementation

The algorithm implemented is iterative with the least mean squares error (LMS) being calculated with each iteration. A solution is deemed to have been found when the percentage difference between the LMS of successive iterations is less than a user-definable value, that is a solution is found when the difference between iterations is sufficiently small.

The aggregate LMS is a weighted sum of the LMS per edge. The relative weighting of known edges (ie edges along the occlusion boundary where the actual geometry is known) can be set by the user. This makes it possible to make estimated surfaces adhere more closely to known edges that occluded edges can be visualised as a "known edge stickiness factor".

The initial values of the surface polynomial coefficients are set so as to result in planar surfaces. Conveniently the corresponds to surface polynomial coefficient values of zero

The LMS per edge is the sum of the squares of the distance between where a pixel ray intersects adjacent surfaces for each pixel along the boundary between the corresponding surfaces in the casual photograph. The shape of the surfaces and thus the distances between them along each pixel ray is determined by the coefficients of the surface polynomial. The degree of the surface polynomial (that is quadratic or cubic) can be set by the user.

New surface coefficient values are calculated for each coefficient with each iteration as follows:

1. the LMS are calculated for a slightly higher and slightly lower value of the current coefficient while keeping all other coefficients unchanged in order to obtain numerically the partial derivative of the coefficient relative to the LMS - $\partial\, polynomial\ coefficient / \partial\, LMS$ - that is the rate of change of coefficient with change in LMS.
2. This value along with the absolute LMS for the current value of the coefficient is used to calculate a new value of the coefficient as follows:

$$polynomial\ coefficient_{new} = polynomial\ coefficient_{old} + \alpha\left(\partial\, polynomial\ coefficient / \partial\, LMS \right) \tag{1}$$

where α is a user adjustable stabilisation factor to prevent the new coefficient value overshooting the minimum LMS value. This is particularly important because each coefficient for both adjacent surfaces in being adjusted independently.

4.9.1 Calculation of d and ω

As is described in § 3.5, finding the location of the intersection of a pixel ray with a surface requires the solution of a polynomial in d (the proportion of the total distance down the edge of the pixel-ray / surface intersection).

The value of *d* is calculated using Newton Iteration (similarly to the values of the surface polynomial coefficients). The Newton Iteration is stabilised by setting a factor (between 0.0 and 1.0) which is applied to the Δ calculated at each step.

4.10 Formats for data input and output

Support for a range of data file formats, for both the input of raw image and geometry data and the storage of results, has been provided.

4.10.1 Input data

Data required for this task consists of casual photographs stored in digital form; and site geometry stored as AutoCAD files.

A large range of image formats has been provided (simply for convenience as facilities to convert between the different standards are widely available). In order to preserve the accuracy of the images it is suggested that the lossless image formats are used. Facilities have been implemented (using the Java Advanced Imaging API) to read in casual photographs held in the following range of formats:

- Tagged Image Format file (.TIFF) - lossless
- CompuServe Graphics Interchange Format (.GIF)- lossless
- Joint Photographic Experts Group (.JPG) - lossy
- Microsoft Windows bitmap image file (.BMP) - lossless
- Portable Network Graphics (.PNG)- lossless

Geometry is read using a facility created to interpret the widely used AutoCAD Drawing Exchange File (.DXF) format.

Geometry is read using a facility created to interpret the widely used AutoCAD Drawing Exchange File (.DXF) format.

4.10.2 Output data

The tool generates output data in various forms, namely image, three-dimensional geometry and virtual reality model data.

Rendered image data of the three-dimensional scene (analogous to a photo) is output to provide evidence of the efficacy to the tool and support the documentation of a modelling project and is available in the GIF and JPG formats above.

Virtual reality output data is stored in the widely accepted Virtual Reality Modelling Language format (VRML - formerly Virtual Reality Mark-up Language). Data in VRML format lends itself to editing and, in particular, combination with other VRML files. This would be useful when combining the output geometry and texture data from this tool with incomplete models based on photogrammetry.

Chapter 5 Evaluation

The aim of this research project is to build a tool which can be used to extract geometric and more importantly texture data from casual photographs for inclusion in virtual reality models of archaeological sites. The tool should satisfy the following functional requirements (summarised from §4.1)

a) Provide results which are accurate in terms of geometry and texture.
b) Be efficient and practical to use.
c) Deal well with the sorts of problems due to occlusions.

The satisfaction of the remaining requirements such as the ability to use various image formats or platform independence are discussed in §Chapter 1. Evaluation takes place on two levels:

1. direct evaluation of constituent technologies which make up the tool (see §5.2 Evaluation of constituent technologies). Constituent technologies are evaluated individually because this indicates their usefulness and contribution to the tool and the types of scenaria (combination of casual photograph and occlusion geometry) to which they are best applied. The constituent technologies evaluated are filters and boundary detection (§5.2.1), calibration (§5.2.2), geometry calculation (§5.2.3) and texture adjustment (§5.2.4).

2. evaluation of the integrated tool as a unit (see §5.3 Evaluation of the tool). The evaluation of the tool concentrates on its completeness when being applied to a range of likely occlusion configurations and casual photograph types rather than upon the relative capabilities of sub-components.

Quantification of the benefits accruing from the tools implemented is very difficult for the following reasons:

- The proportion of each site which is occluded from photogrammetric survey is dependent on its topology (across a wide range of scales), the number of image pairs captured and the positioning of the cameras. It is not possible in any useful way to generalise about the amount of occlusion in the case of archaeological sites; but it suffices to say that with the theoretical exception of very simple polyhedron based sites (or those with an infinite number of survey images) that *some occlusion will exist.*
- It is difficult to quantify the link between completeness of a model and its level of "reality", immersiveness etc. Clearly a model with missing bits is less able to provide a sense of reality.

Although quantification of the benefits is not possible it is reasonable to assert that on a *qualitative* level that the filling of occlusions (which are inevitably present) makes the model more complete and thus more useful.

Therefore, the approach of this evaluation is to compare the results of using this tool with models captured entirely using photogrammetric techniques, that is, we use the results captured using photogrammetric modelling as the correct or datum model with which we compare the tool generated output. This is achieved by artificially "occluding" non-occluded parts of a three-dimensional model; filling the "occluded" region using the tool; and then comparing the resulting three-dimensional model with the non-occluded version. The only way to obtain a model which is precisely the same as a complete photogrammetric model except that it has some occlusion is to artificially create an occlusion on the complete model by blanking some of the surface out.

The test cases of flat surface, curved surface, broad- and narrow corner have been selected in order to provide the best coverage the likely configurations of archaeological sites at an elemental level (see §5.1).

5.1 Evaluation test cases

Test cases have been selected in order to achieve comprehensive coverage of the types of features likely to be encountered in occluded regions of archaeological sites. Four different geometrical scene configurations are used, namely:

- flat surface containing an occlusion (shown in Figure 24)
- curved surface containing an occlusion (shown in Figure 25)
- broad occlusion over a corner (shown in Figure 26)
- and narrow occlusion over a corner (shown in Figure 27)

It is reasoned that this is good coverage, firstly because the entire range of surface curvature (from flat to a corner) is addressed in incremental steps and, secondly that the features typically hidden (read occluded) in niches are corners.

All the test cases are from a site which is made up of blocks. This provides the evaluator with a set of grooves[16] (between the blocks) which can be used to judge the degree of match-up between patches and occluded geometry.

In all the test cases Graphic Image Format (GIF) image files have been used for the following reasons:

1. The Virtual Reality Modelling Language (VRML) Standard [40] only requires that browsers support GIF and Joint Photographic Experts Group (JPEG) images.
2. GIF pixels are presented "as is" [75] rather than in the approximated so-called continuous-tone form of JPEGs [74].
3. Associated with the above reason, from a very close range separate pixels are visible in GIF images which allow a better comparison between images.

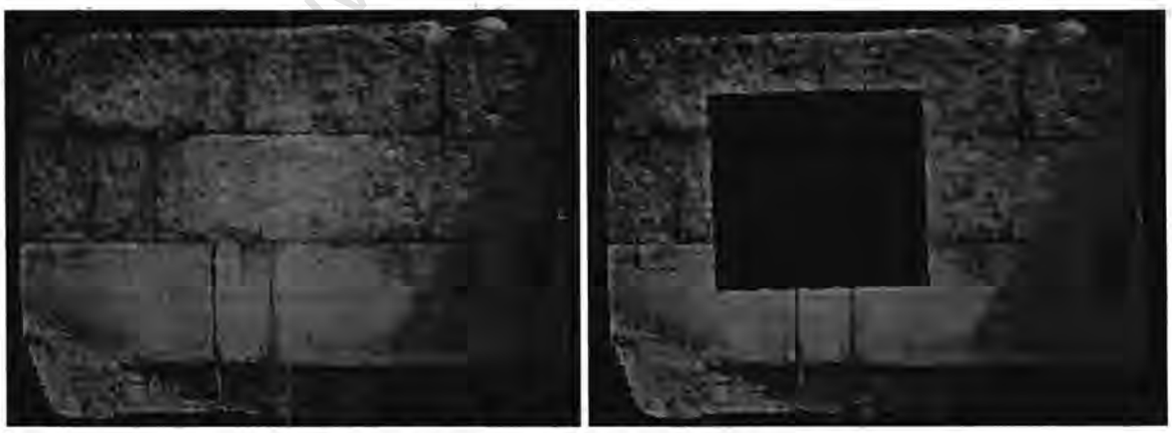


Figure 24 showing a *flat* wall complete and with an occlusion

[16] The term groove is used to describe the mostly linear spaces between blocks in order to differentiate them from boundaries (detected on casual photographs), corners (where walls meet) and particularly cracks (gaps between quadrilateral segments of patches as shown for example in **Figure 45**d)

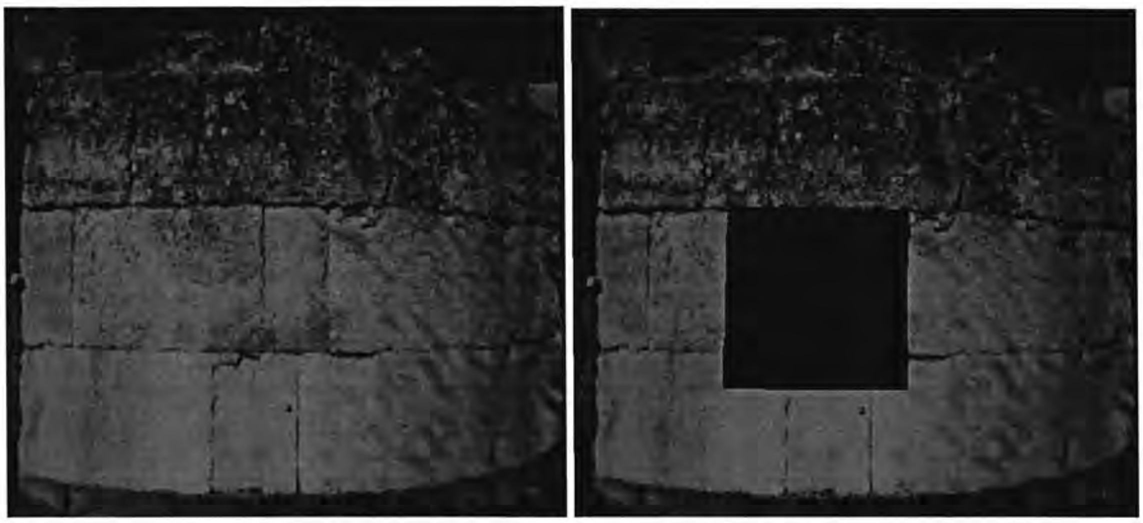


Figure 25 showing a *curved* wall complete and with an occlusion

For each of the above types of geometry and occlusion we will show a set of images of the three-dimensional model from various vantage points. Each image set is made up of the same view of a) the photogrammetrically captured model and b) the model with "occlusion" filled using the tool. As will become apparent it is necessary to view the fixed models from various vantage points in order to provide in two-dimensional form material which is better viewed as a three-dimensional model.

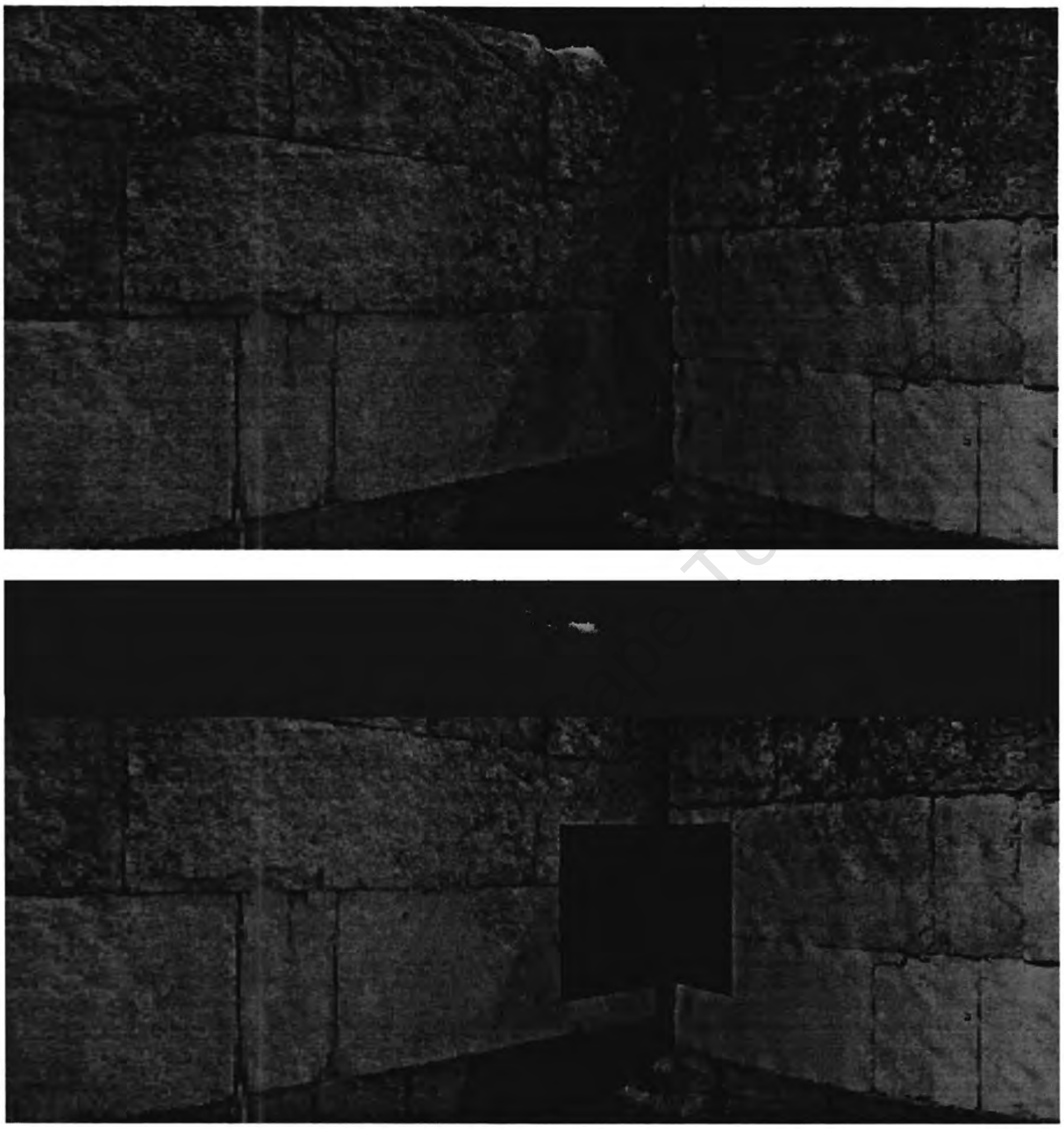


Figure 26 showing the *corner* between a curved and flat wall complete (top) and with a *broad* occlusion (bottom)

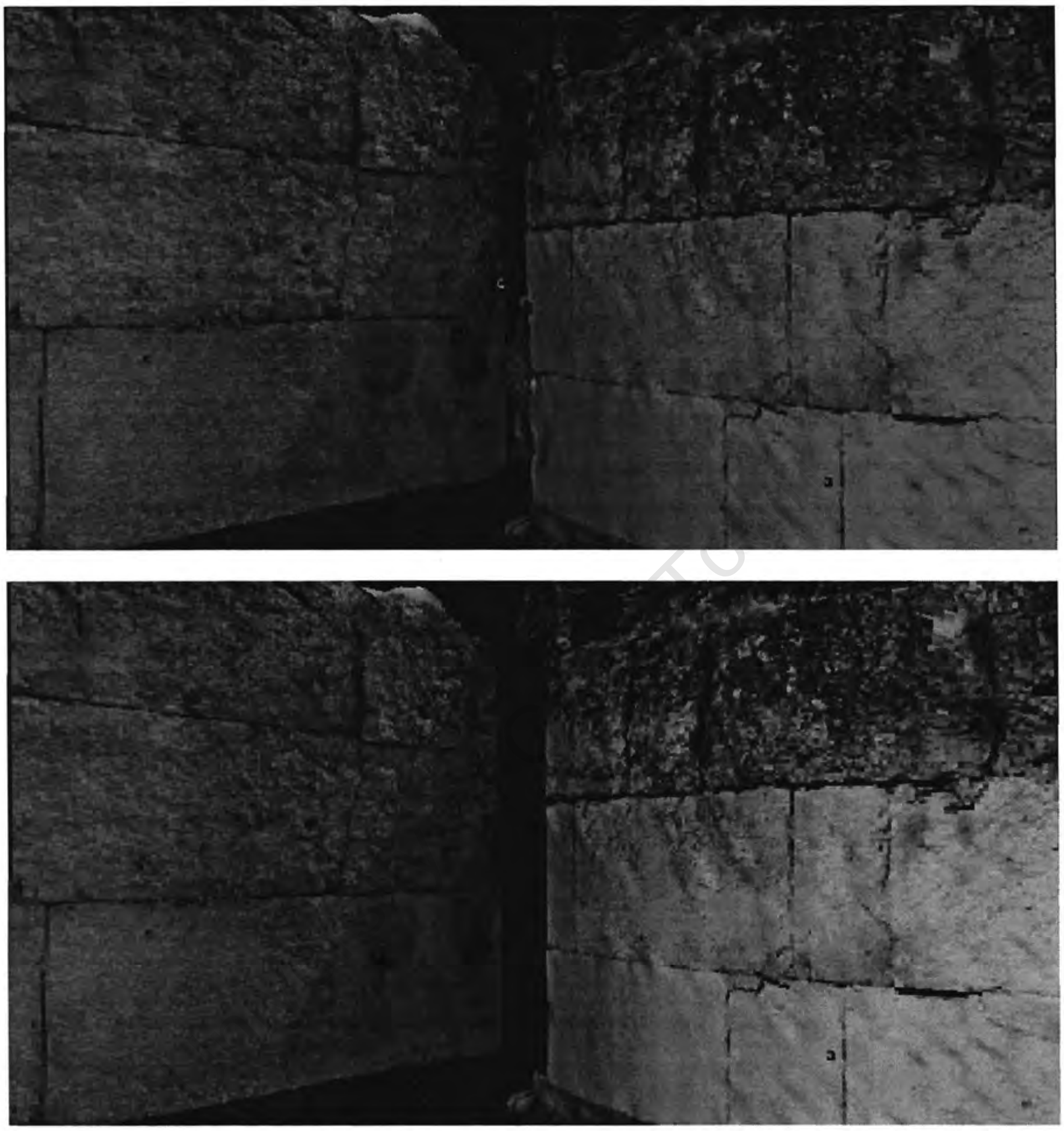


Figure 27 showing the *corner* between a flat and curved wall complete (top) with a very *narrow* occlusion (bottom). As can be seen in the non-occluded view (top) much of the detail in this corner is missing. Here it is hoped to use the tool to actually fill this missing geometry and texture information from a casual photograph.

A selection of relevant elements of the test cases described above are used for the evaluation of constituent technologies (§5.2) and the complete test cases are used for the evaluation of the tool in its entirety (§5.3).

5.2 Evaluation of constituent technologies

In addition to the indirect evaluation of constituent technologies (eg boundary detection and texture adjustment) which is implicit in the evaluation of the complete tool (see §5.3); this section describes the individual evaluation of the constituent technologies of filters and boundary detection (§5.2.1), calibration (§5.2.2), geometry calculation (§5.2.3) and texture adjustment (§5.2.4). Evaluation of the constituent technologies is important because it indicates a) their usefulness and contribution to the tool and b) the types of scenaria (combination of casual photograph and occlusion geometry) to which they are best applied.

5.2.1 Filters and boundary detection

Accurate localisation of boundaries is essential to:

1. the correct delineation of texture patches. The tool functions by extracting patches of texture (corresponding to surfaces within occluded regions of the three-dimensional model) from casual photographs. The usefulness of these patches is thus dependent on the accuracy of their boundaries.
2. the subsequent calculation of occluded geometry. An important input to the calculation of occluded geometry is the positions of pixels corresponding to surface discontinuities (see §3.5). The correct calculation of this geometry is thus dependent on the accuracy of boundary detection.

It is also important that the boundary detection algorithm as implemented is practical to use in terms of its performance, that is, the amount of time it takes to find a boundary. Times of the order of a second to localise a typical boundary would be practical but times of the order of minutes or more would make the tool very impractical to use.

Evaluation is visual and qualitative; achieved by observing the degree to which known discontinuities visible on the casual photograph are in fact detected as boundaries. Figure 28 indicates the position of the image used for evaluating the boundary detector and filters. This region was selected for boundary detection evaluation because it includes edge elements of differing orientation, thickness, smoothness and visual prominence

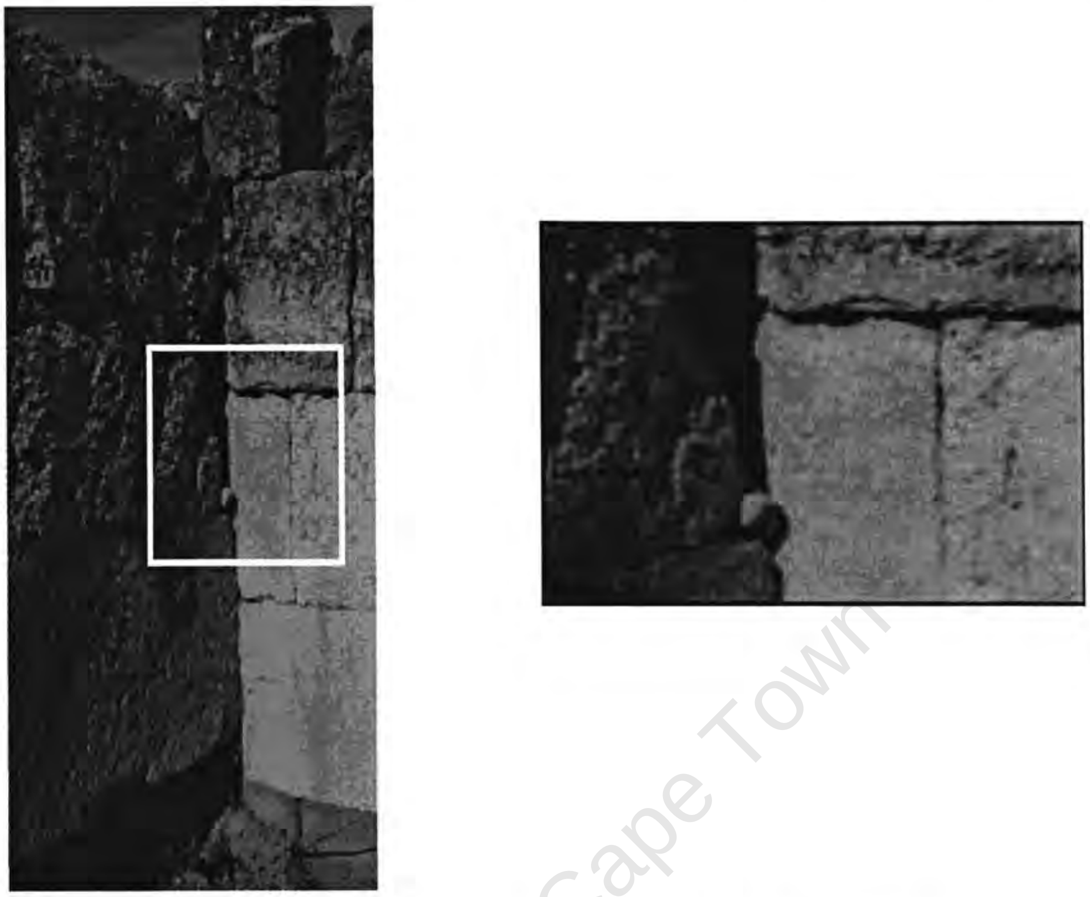


Figure 28 indicating the section of casual photographs used in evaluating the edge detectors and filters

The boundary detection algorithm implemented (see §3.3) localises boundaries by finding the lowest-cost path between a start- and end pixel. The relative weightings of the cost terms of *Laplacian Zero-Crossing (f_Z)*, *Gradient Magnitude (f_G)* and *Gradient Direction (f_D)* are user adjustable and may be used to tune the algorithm to a set of images. First we evaluate the effect of changing the three cost weighting factors on the localisation of boundaries independently of the use of filters (that is, without pre-filtering of the casual photograph).

The following is a series of images showing an edge detected using different weightings for the cost terms of f_Z , f_G and f_D: without applying any filters.

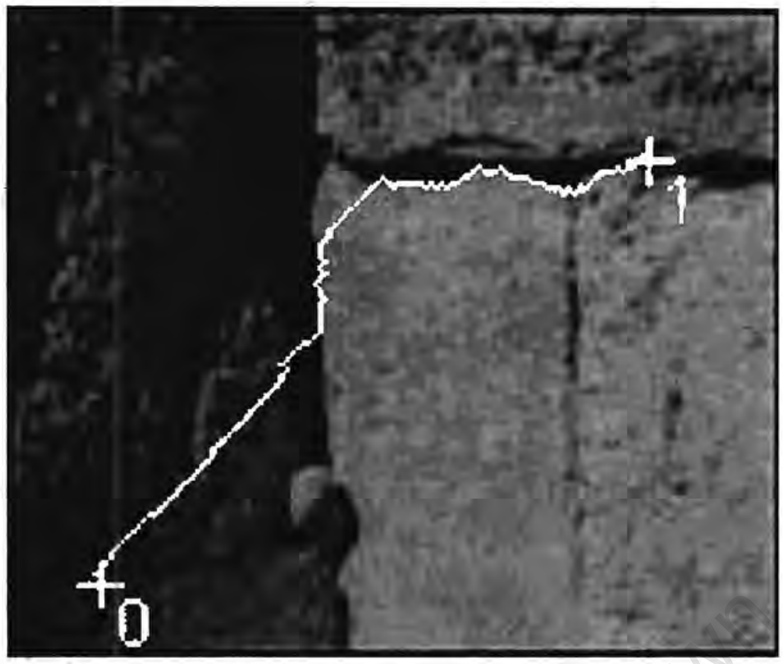
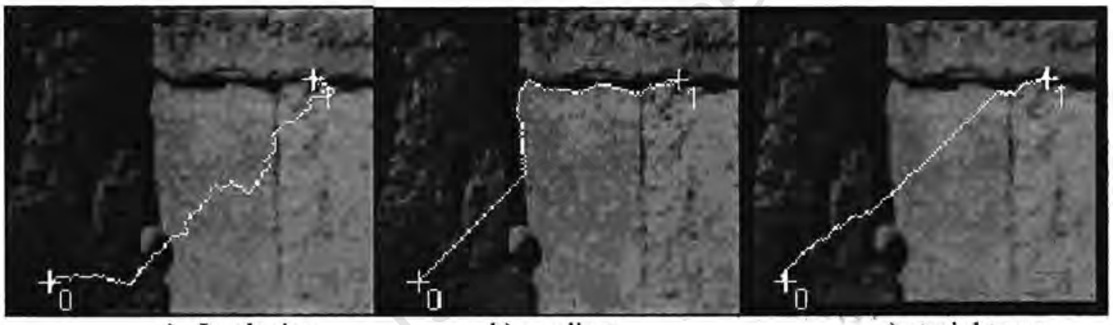


Figure 29 showing the edges detected without filtering and using the weighting values as suggested by Mortensen and Barratt [52] of 43%, 43% and 14%. It can be seen that the edge has been poorly localised with these weighting factors.



a) Laplacian b) gradient c) straightness

Figure 30 shows a sequence of edges detected by setting only one of the weighting factors. The figures show the effect of a pure Laplacian detector (a), a pure gradient detector (b) and a pure straightness detector (c).

It can be clearly seen from Figure 30b above that the gradient is the most important for the accurate localisation of edges. From Figure 30a it is apparent that Laplacian zero crossings occur at many locations on the image. The edge thus detected is the shortest path with some small deviations due to zero crossings occurring near to this path.

It was found that the best edge localisation occurred with the following factors: Laplacian 33%, gradient 67% and straightness 0% (shown in Figure 31). Furthermore, it can be seen that the boundary corresponding to the corner (the vertical portion) has been well localised. The time taken to detect this boundary is a faction of a second.

An interesting fact is that at no time did the use of the straightness factor improve the detection of edges. In the author's view this term is redundant as the effect of non-straightness is accounted for by the fact that more pixels are traversed and thus greater "costs" incurred when a very non-straight route is followed between any pair of pixels.

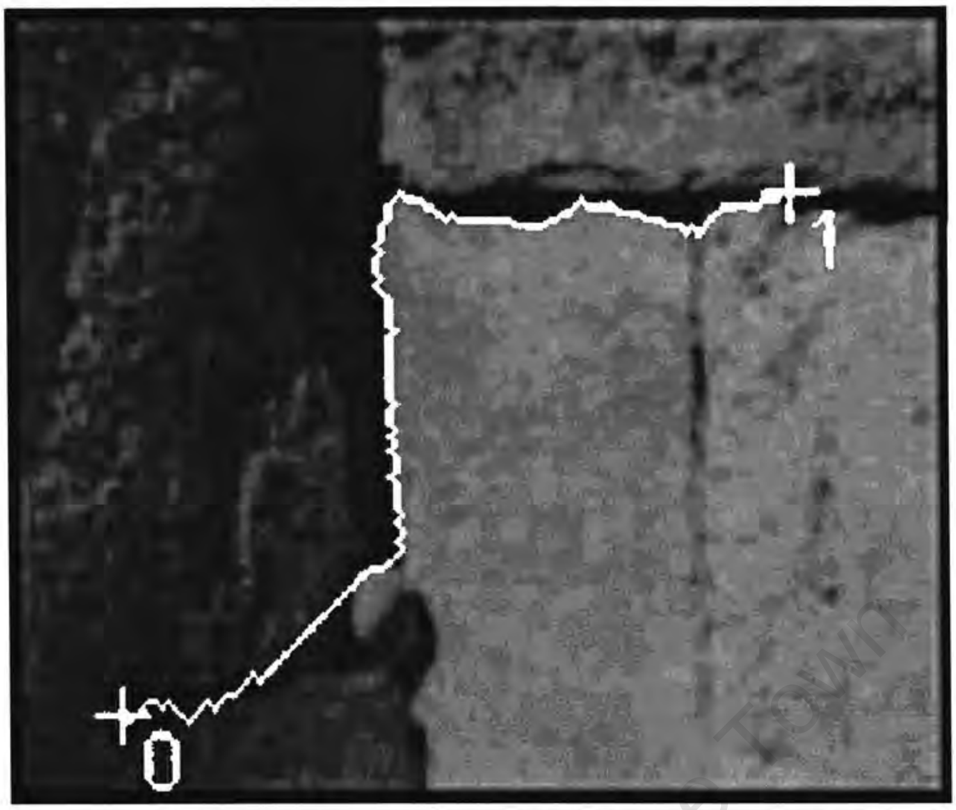


Figure 31: Edge detected with parameter values of: Laplacian 33%, gradient 67% and straightness 0%

The effect of filters

Various filters have been applied to see whether this results in better edge detection. The next series of figures shows the application of various filters together with constant values of detection parameters as determined above.

The degree of the filter refers to the spatial extent of the filter in terms of the number of raw pixels used in the calculation of a filtered pixel value. A filter of degree one nominally takes into account a single layer of adjacent pixels; degree two uses two layers etc. The precise number of pixels used and their relative weightings depends on the type of filter.

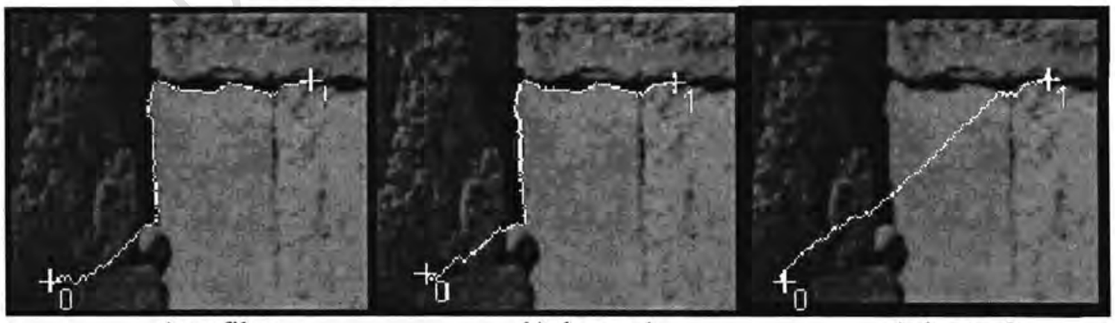


Figure 32 shows the use of a box filter of various degrees: a) no filter, b) degree 1 and c) degree 2.

As can be seen in Figure 32 there is no benefit in using a box filter. In fact the filter of degree one reduces the accuracy of edge detection slightly and a filter on degree two results in the edge being lost altogether.

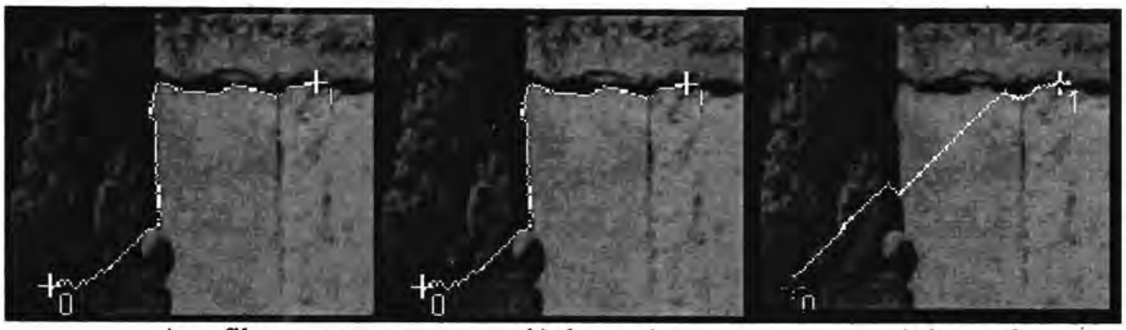


Figure 33 shows the use of a Gaussian filter of various degrees: a) no filter, b) degree 1 and c) degree 2.

As can be seen in Figure 33 there appears to be a small benefit in using a Gaussian filter. The filter of degree one makes no difference and a filter on degree two results in the edge being lost altogether.

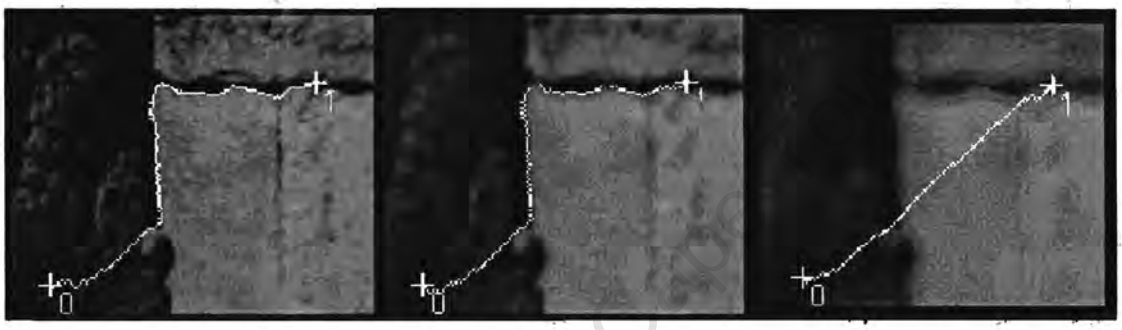


Figure 34 shows the use of a difference of Gaussian filter of various degrees: a) no filter, b) degree 1 and c) degree 2.

As is the case with a Gaussian filter, it can be seen in Figure 34 that there appears to be a small benefit in using a difference of Gaussian filter. The filter of degree one results in a marginal improvement but a filter of degree two results in the edge being lost altogether.

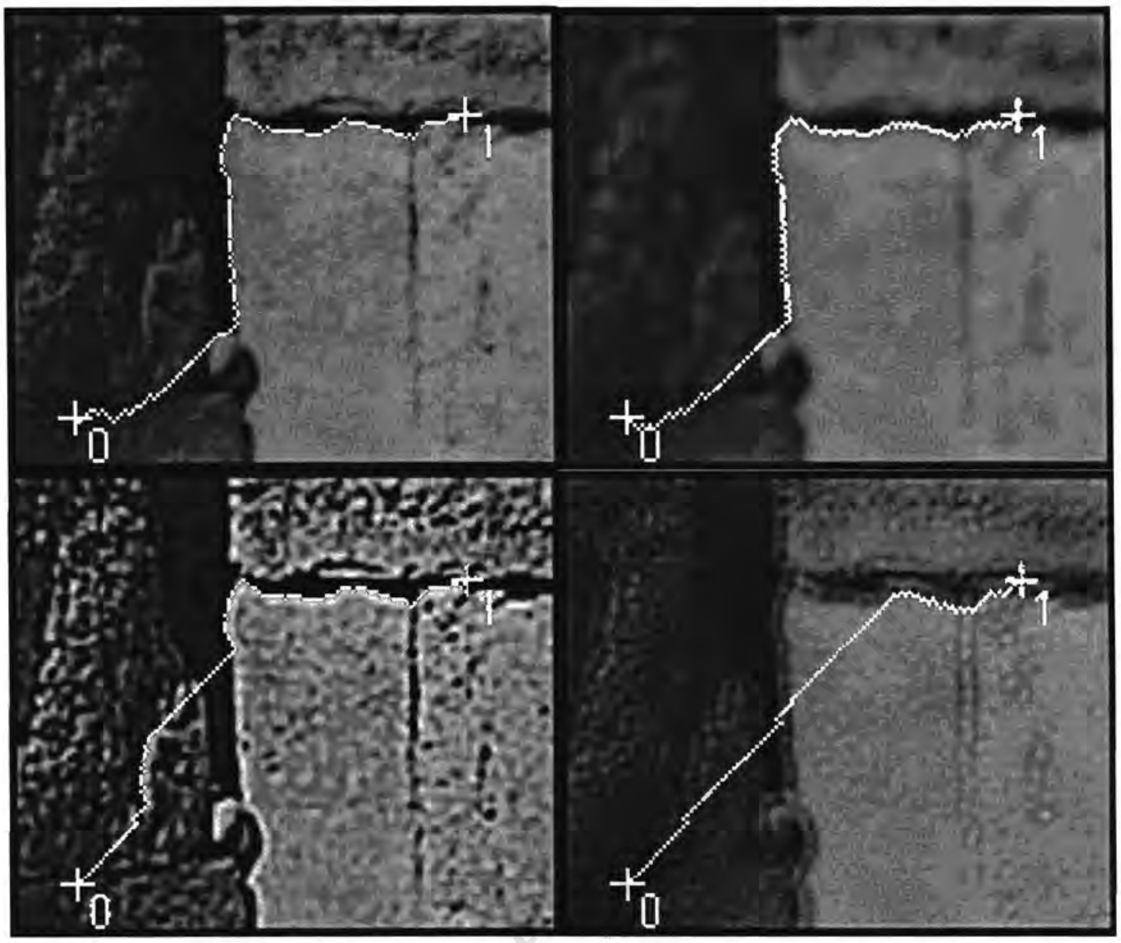
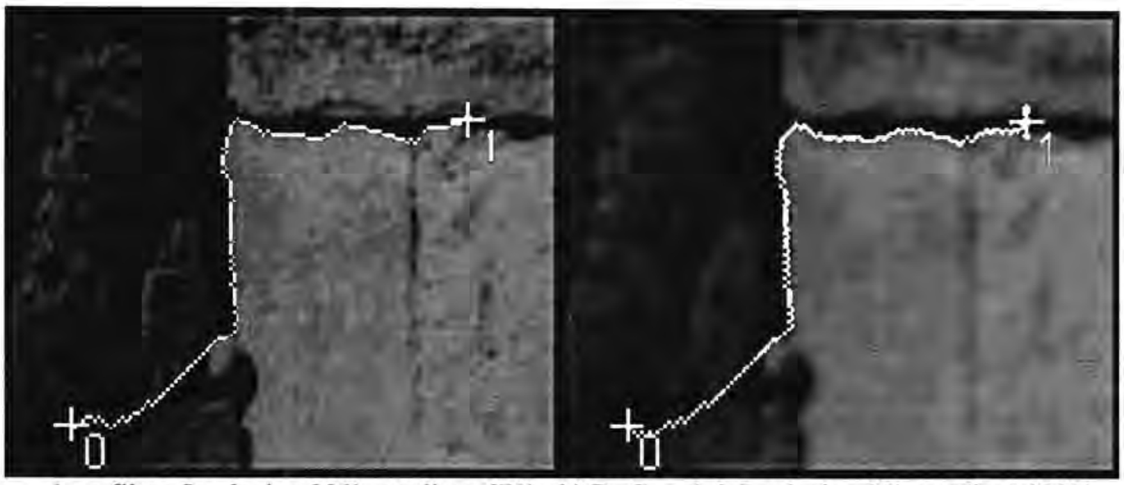


Figure 35 showing the effect of difference of Gaussian filters with various values of sigma ratio. Starting at top left and proceeding clockwise the images show no filter then filters of degree one having sigma ratios of 1.6, 2.4 and 3.2.

It can be seen from Figure 35 the best sigma ratio is a value of 1.6. A value of 2.4 does bring to prominence all edges in the image (as can be seen from the deviation of the edge located about a third of the way along). It was found, however, that in the data set used that these more subtle edges are not those being sought. Using a sigma ratio of 2.4 had the effect of obliterating most of the edge.

As can be seen form the preceding set of figures, filtering did not improve the detection of edges in the data set used for evaluating them. In fact with the exception of a difference of Gaussian filter of degree one (shown in Figure 34b) which showed no effect, the filters had a deleterious effect on edge detection.

From a theoretical perspective is interesting to note the near equivalence of a difference of Gaussian filter with a sigma ratio of 1.6 with a Laplacian. In Figure 36 it can be seen that almost precisely the same edge is detected if the Laplacian component of the edge detector is substituted by a difference of Gaussian filter with sigma ratio of 1.6.



a) no filter, Laplacian 33%, gradient 67% b) DoG σ 1.6, Laplacian 0%, gradient 100%

Figure 36 showing edges detected with a) no filter and parameters of Laplacian = 33% and gradient = 67%; and b) a difference of Gaussian filter having sigma ratio of 1.6 with parameters of Laplacian = 0% and gradient = 100%

In summary it can be said that the boundary detection algorithm provided accurate boundary localisation within usable turn-around-times. Pre-filtering was found to provide no improvement with the images used.

5.2.2 Calibration

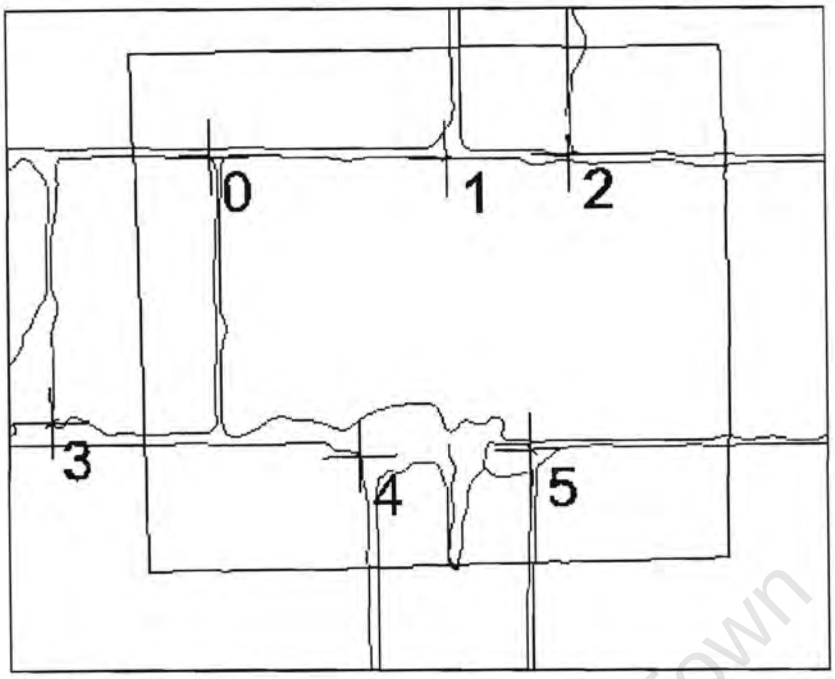
The underlying calibration functionality (see §3.1) is important to the tool in two ways:

1. it is used to project the occlusion boundary onto the casual photograph. The points making up the occlusion boundary are projected using the parameters calculated by the calibration process and used along with edges detected by other means to bound regions.
2. it is used to calculate *pixel-rays* which are in turn used by the geometry-calculating algorithm (see §3.5).

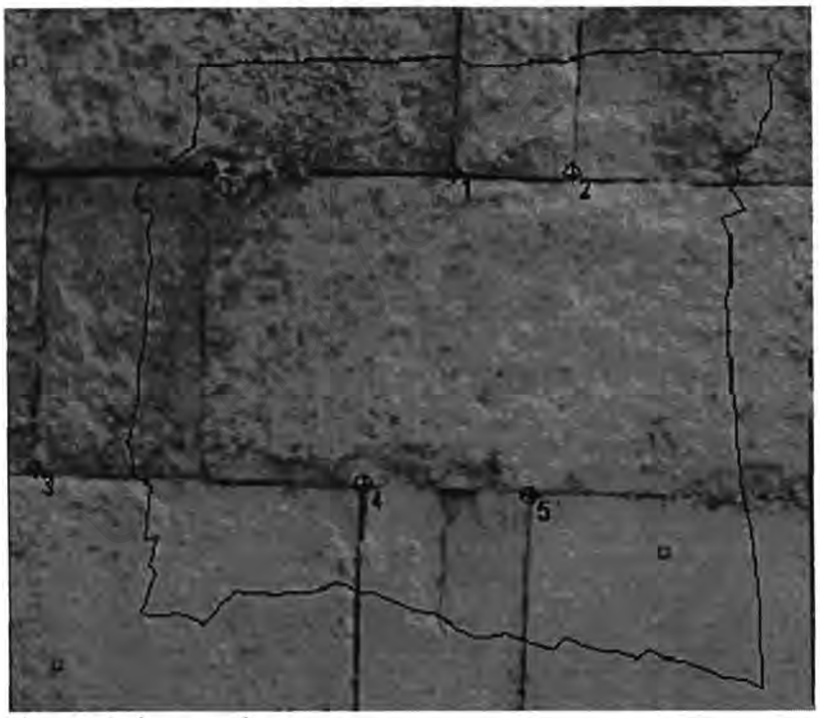
It in known that the accuracy and robustness of a calibration is dependent on the configuration of the three-dimensional geometry; in particular the degree to which the calibration points fall in the same plane (a matrix inversion singularity occurs if all the points fall in exactly the same plane - see §3.1). For this reason the evaluation of the calibration algorithm is based on a flat (that is, nearly planar) *and* a curved surface.

Flat surface

Figure 37 shows the results of a calibration and the projection of a rectangular occlusion boundary using points on a flat wall.



a) geometry



b) casual photograph

Figure 37 flat wall calibration showing a) the numbered calibration points (cross-hairs) and rectangular occlusion boundary on the geometry and b) the numbered calibration points (cross-hairs) and their calculated position by least squares calibration (circles) and the projection of a rectangular occlusion boundary.

It can be seen from Figure 37b that although a calibration can be calculated for the almost planar set of points; evidenced by the fact that the image calibration points (cross-hairs) and corresponding points calculated from the geometry (circles on the photograph) coincide very well; this solution is unstable (due to the fact the calibration points are nearly co-planar), as can be seen from the projection of the occlusion boundary shown in Figure 37b. When this occlusion boundary is used as the basis for extracting casual photograph texture the results are unsatisfactory. This can be seen from the misalignment and distortion of the lines in Figure 38, problems are encountered when clipping from casual photographs of planar subjects because of the difficulty in obtaining a robust[17] calibration using such subjects.

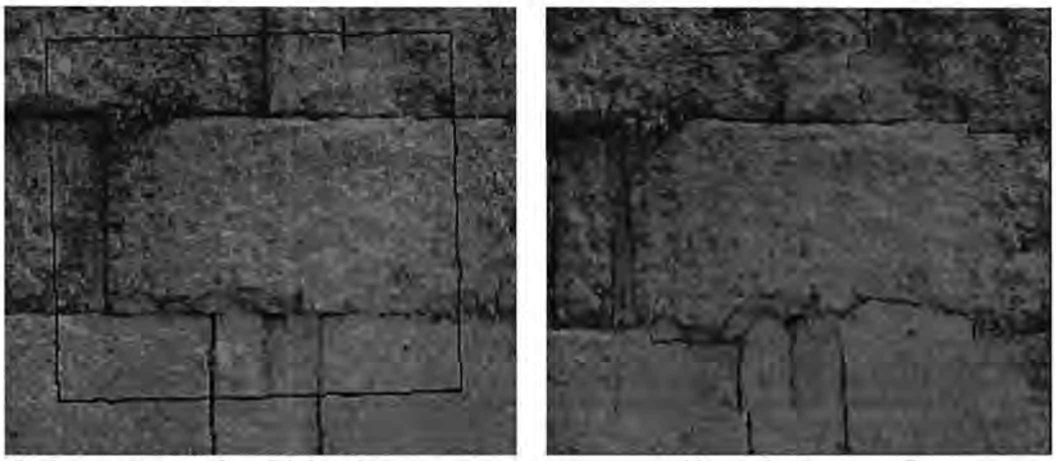
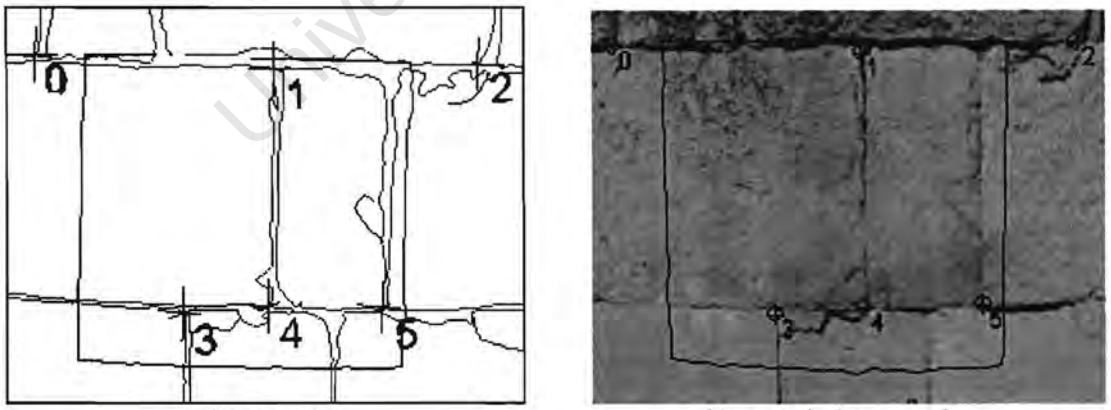

a) photogrammetric with border b) patched

Figure 38 showing the fix applied to the flat wall using the occlusion boundary, the edge locations of which has been calculated from the calibration of the casual photograph.

Curved surface

The curved surface is far more amenable to calibration because such a surface is, by definition not planar, which makes the selection of non-co-planar points inevitable. (It is possible, but difficult to select points on a curved surface which are co-planar). Figure 39 shows the calibration points and resultant projected occlusion boundary on the casual photograph. The fact that the projection of the rectangular occlusion boundary generates a rectangle on the casual photograph is evidence of the robustness of the calibration.

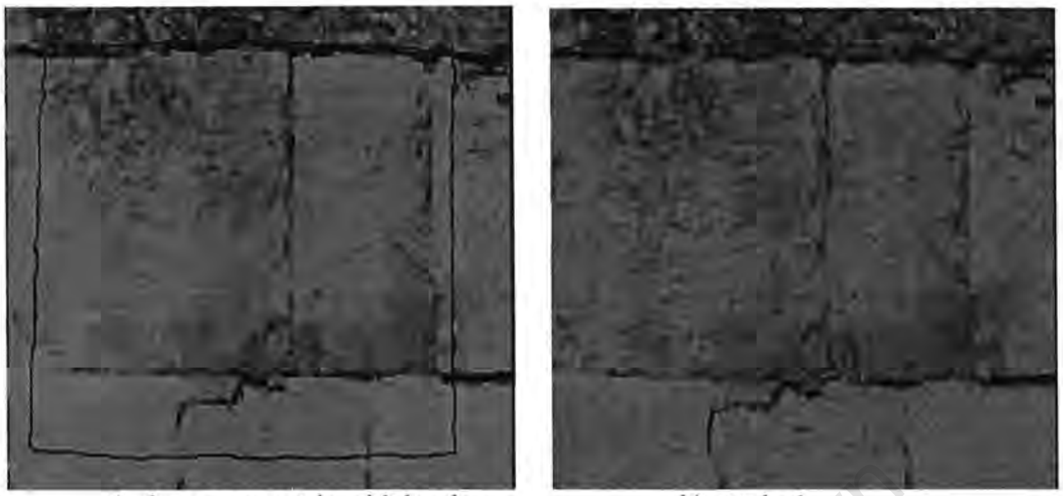


a) geometry b) casual photograph

Figure 39 curved wall calibration showing a) the numbered calibration points (cross-hairs) and rectangular occlusion boundary on the geometry and b) the numbered calibration points (cross-hairs) and their calculated position by least squares calibration (circles) and the projection of a rectangular occlusion boundary.

[17] The turn robust is used here in the mathematical sense related to the concept of stability, namely that a small error in input points will result in a small error in calibration.

It can be clearly seen by comparing the shape of the occlusion boundary in Figure 39 above with Figure 37b that the curved surface is more amenable to calibration. As a result a better patch is obtained (as shown in Figure 40)



a) photogrammetric with border b) patched

Figure 40 showing the fix applied to the curved wall using the occlusion boundary, the location of which has been calculated from the calibration of the casual photograph. The good groove alignment and lack of distortion is very apparent in b)

As expected calibration using a set of points from a planar surface provides unacceptable results. However, when the set of points in non-planar then satisfactory calibration, occlusion projection and patch extraction can be achieved, thus calibration provides a robust and practical basis for clipping from the casual photograph.

5.2.3 *Geometry calculation*

A variety of different techniques have been used to address the problem of calculating (or inferring) the geometry in and around occlusions. This subsection compares the following algorithms in order to determine their usefulness and suitability to different occlusion configurations (the test configurations were described previously in §5.1). Briefly these algorithms may be classified as:

- those for integrating the patch and the remainder of the model (that is, the photogrammetrically captured part).
    - planar triangles (see §3.4)
    - occlusion boundary geometry bounded. The known position of the occlusion boundary in three-dimensions is used to edge the texture patches.

These are evaluated in the next subsection - *Planar triangles vs occlusion boundary edged patches.*

- those for calculating corners within occlusions.
    - ignoring the corners, that is, applying a flat patch over a corner
    - assuming the surfaces which meet at these corners are essentially flat (either with a straight corner or with a corner based on a detected boundary).
    - assuming the surfaces may be approximated by a smooth cubic or quadratic polynomial (see §3.5). Here the corner can be closed or

allowed to retain a small crack-like gap caused by the least mean squares nature of this algorithm (see §3.5.2).

These are evaluated in the three subsections *Comparison of Corner Models* using a *pseudo-corner* and *broad-* and *narrow occlusions*, respectively, as test cases below.

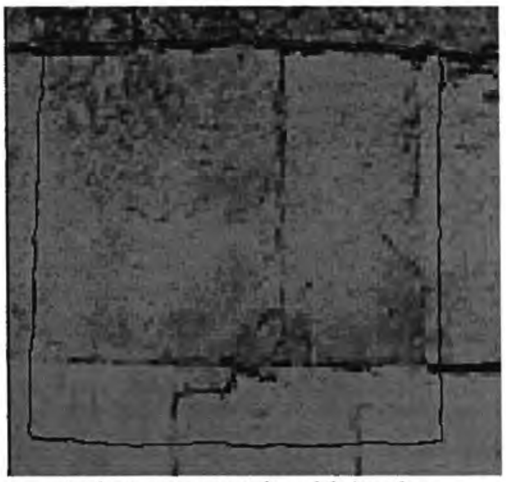
Planar triangles vs occlusion boundary edged patches

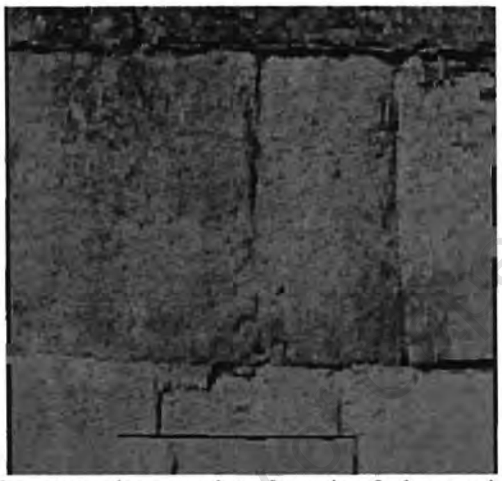
The following series of figures illustrates the use of a pair of *alternative techniques* for calculating the geometry of the occluded region. In both cases the projected occlusion boundary is used to locate that part of the casual photograph which corresponds with the occluded region. However, in the first case (see Figure 41b, Figure 42b and Figure 43b) this region is assumed to consist of a pair of *planar triangles* (the corners of which correspond to the corners of the rectangular occlusion). In the second (see Figure 41c, Figure 42c and Figure 43c) *the geometry of the occlusion boundary is used to provide edges* to the patch.

In all cases it can be seen that the use of the occlusion boundary to provide edges (the Figure c in each case) is the better technique. The following features are of particular significance:

- the gap and misalignment of lines at the bottom edge in Figure 41b relative to the seamless fit in Figure 41c.
- from a comparison of figures Figure 42b and c, which is a close-up view of the top edge of the occlusion, no material difference in fit is illustrated
- the gap and lack of relief in the close-up oblique view in Figure 43b relative to that in Figure 43c.

The difference in pixel sizes, apparent particularly in figures Figure 42 and Figure 43, should be ignored as they only become apparent at very close ranges. In the normal course of events virtual reality models will not be viewed from such ranges.


a) photogrammetric with border


b) patch assumed to consist of a pair of planar triangles

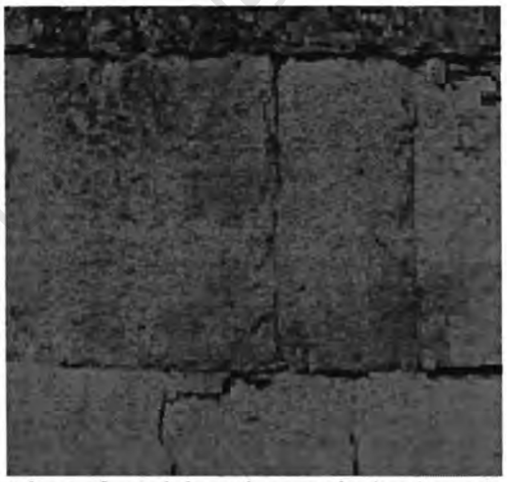
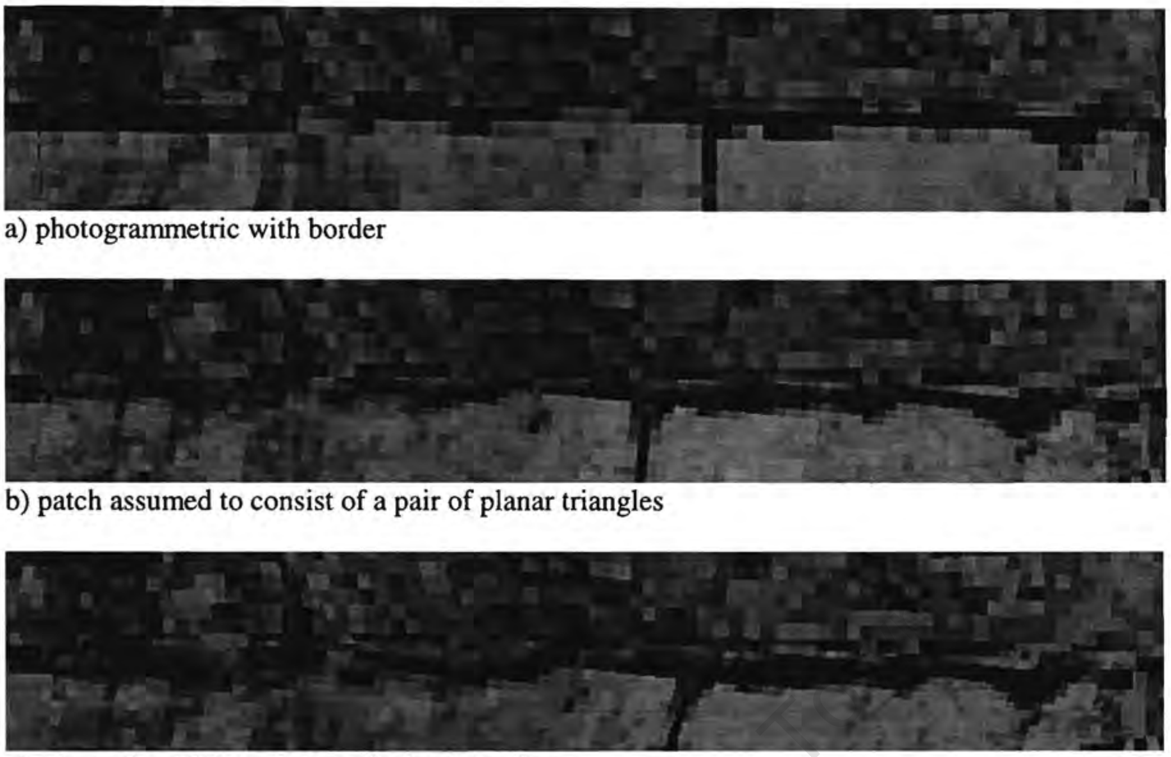

c) edges of patch based on occlusion boundary

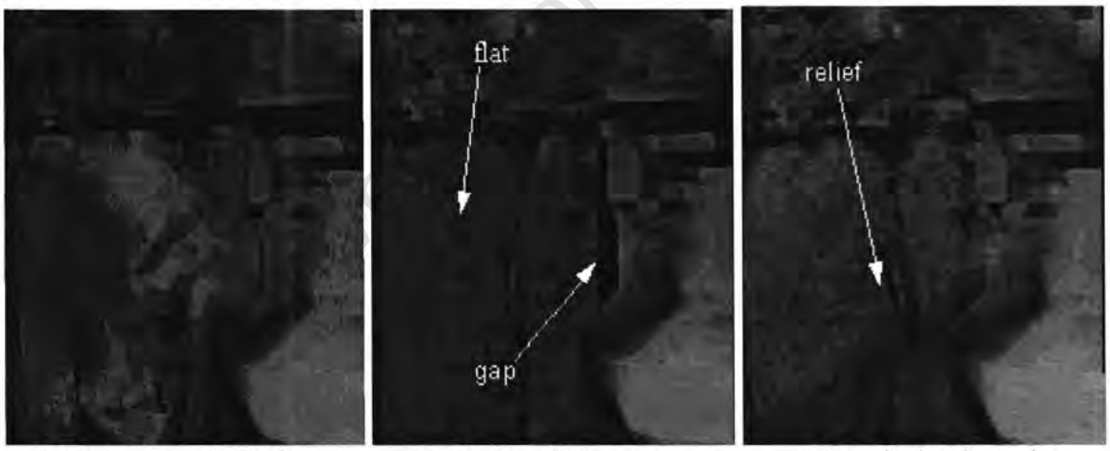
Figure 41 showing a front view of the curved wall in a) original photogrammetrically captured form, b) patched using a pair of planar triangles and c) patched using occlusion boundary edge geometry. Note the gap and misalignment of lines at the bottom edge of the patch in b).


a) photogrammetric with border

b) patch assumed to consist of a pair of planar triangles

c) edges of patch based on occlusion boundary

Figure 42 showing a close up view of the top edge of the occlusion on the curved wall in a) original photogrammetrically captured form, b) patched using a pair of planar triangles and c) patched using occlusion boundary edge geometry. The fact that these images are practically identical is evidence of the degree to which the planar triangle (b) and occlusion boundary edge (c) approaches provide satisfactory patches.



a) photogrammetrically b) pair of triangles c) occlusion boundary

Figure 43 showing a close up view of the top right corner of the occlusion on the curved wall in a) original photogrammetrically captured form, b) patched using a pair of planar triangles and c) patched using occlusion boundary edge geometry. Note the gap in b) in about the centre of the figure and the lack of relief.

From the previous sequence of figures it is clear that the results obtained using the occlusion boundary to provide the geometry of the patch are far superior to those using planar triangles.

Comparison of Corner Models using a pseudo-corner

Thus far we have only considered the case where the patch is made up of a single "quadrilateral" made up of either a pair of triangles or the projected rectangular occlusion boundary. A means to partially evaluate the algorithm for calculating occluded corners would be to introduce a pseudo-corner in a region where none exists and see whether or not a corner like feature is generated (none should be generated). A pseudo-corner can be introduced by dividing a flat occluded region into a pair of quadrilaterals; the edge between them being the "corner" to be calculated.

In the case of a cylindrical surface a pseudo-corner in the same alignment as the curvature should have the same curvature as the surface in general. In the following sequence of figures the results of using various corner models are compared with: each another, the photogrammetrically based model and the results of the single quadrilateral approach.

The approaches which are contrasted here are:

- a single quadrilateral, that is without a pseudo-crack along the inter-block groove. This is included here as it results in a patch which although curved along the occlusion boundary is flat near the centre (that is in the region of the inter-block groove). See Figure 45 b.
- cubic polynomial surfaces with the crack between the edges of the patches closed by calculating a mean position based on separate surfaces. See Figure 45 b.
- cubic polynomial surfaces without the closing the crack. See Figure 45 c.
- quadratic polynomial surfaces with the crack between the edges of the patches closed by calculating a mean position based on separate surfaces. See Figure 45 d.
- quadratic polynomial surfaces without the closing the crack. See Figure 45 e.

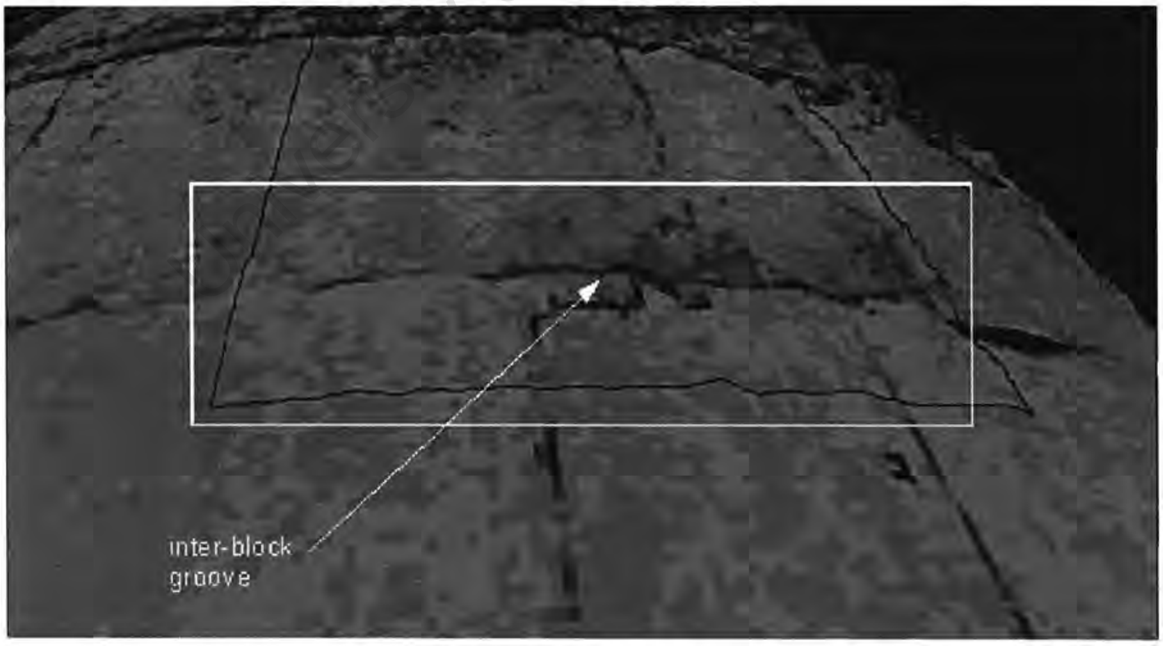
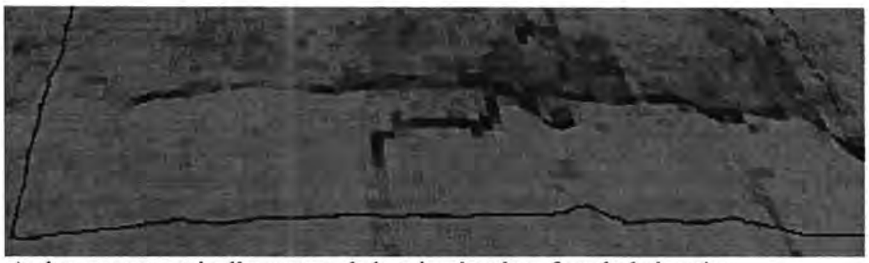
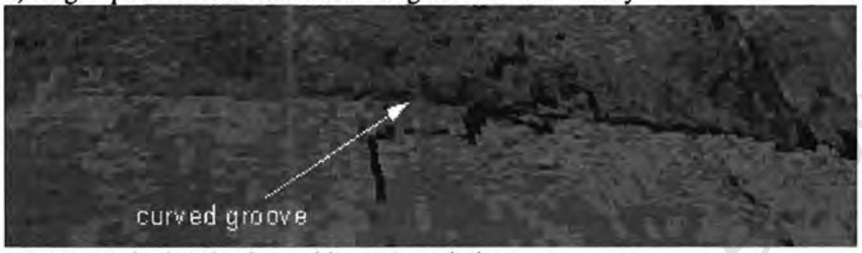


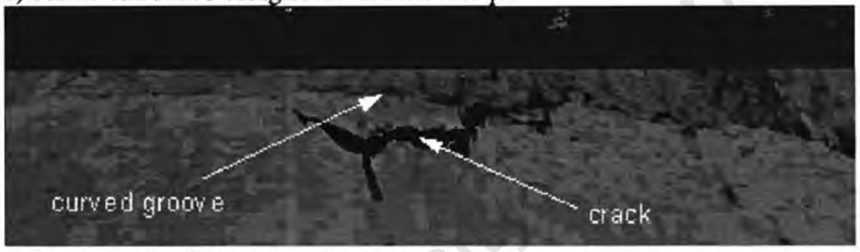
Figure 44 showing the location on the curved surface with which the following sequence of close-up images shown in Figure 45 correspond.


a) photogrammetrically captured showing border of occluded region

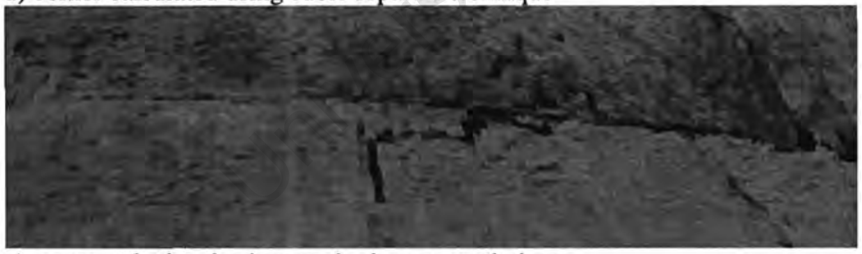
straight groove
b) single quadrilateral calculated using occlusion boundary



c) corner calculated using cubic mean technique



d) corner calculated using cubic separate technique


e) corner calculated using quadratic mean technique

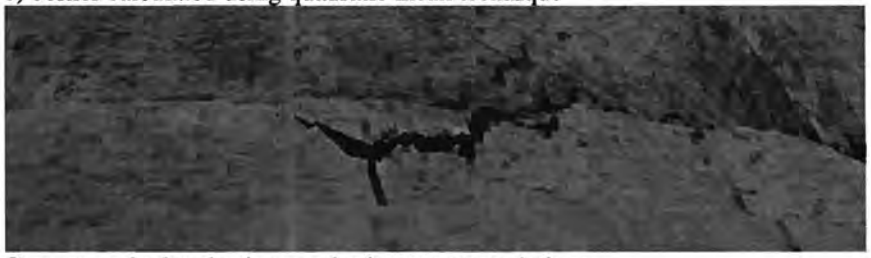

f) corner calculated using quadratic separate technique

Figure 45 shows a comparison of different techniques for calculating the curvature of the surface corresponding with the line across the centre of the image.

By comparing Figure 45a with Figure 45b it can be seen, as would be expected, that the curvature of the surface is lost when using a single quadrilateral. The use of a single quadrilateral is equivalent to using a straight line (that is a polynomial of degree one) to connect the ends; clearly this results in a flat surface.

Figure 45c and d show the effect of using cubic surfaces (respectively without or with the resulting gap at the join being closed be calculating a new join at the average position of the two separate edges). It is clearly visible that the curved surface is largely restored by the use of cubic surfaces, that is that *the geometry which is occluded has been restored by the corner-calculating algorithm.*

Figure 45e and f show the effect of using quadratic surfaces (once again with and without gap in join.). There is no discernible difference in the case of what is approximately a round surface in the effect of using cubic or quadratic approximations.

Closing the gap between the surfaces, as shown in Figure 45d and f provides a visually more realistic result without compromising accuracy (cf. Figure 45a).

An interesting aspect is that in both the cubic and quadratic cases there exists a degree of flattening; this is in the author's view due to the fact that a least mean squares technique is used to "fit" the surface. A least mean squares curve fit is an inherently flattening effect (see Figure 46).

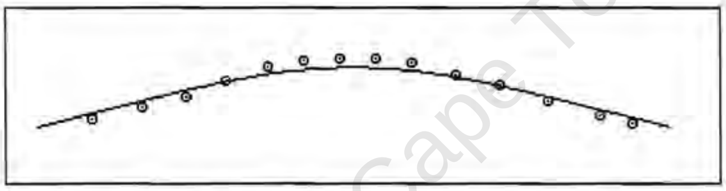
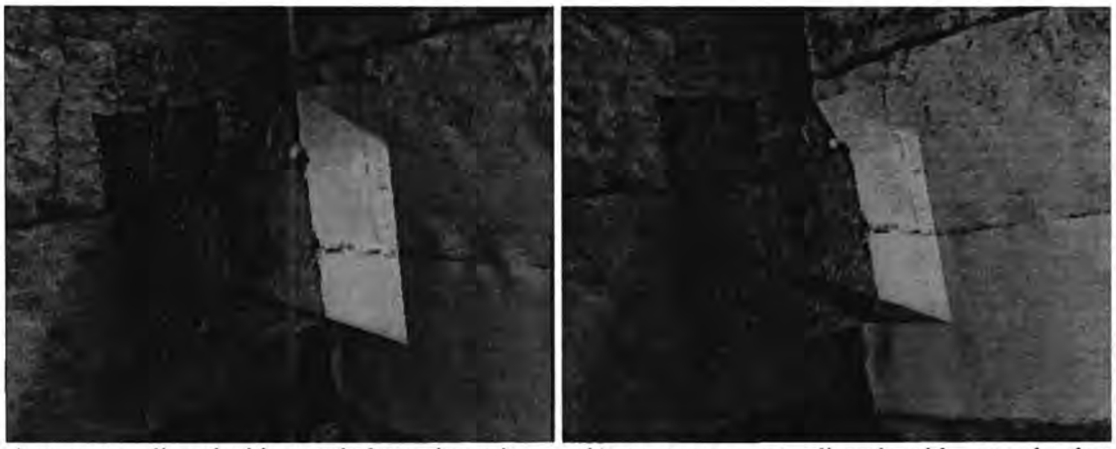


Figure 46 showing the inherently flattening nature of a least means squares curve fit.

Comparison of Corner Models applied to a broad occlusion

In the cases treated so far (flat and curved wall) the occluded region is reasonably continuous. In the following two cases (the broad and narrow corner) the occluded regions contain a discontinuity in the form of a corner which will be used to test the corner-calculating algorithm further. Figure 47 shows what results if the presence of the corner is ignored (naturally the edges of the occluded region follow the shape of the corner). Clearly the results generated without taking the corner into account are unsatisfactory.



a) geometry aligned with casual photo viewpoint geometry

b) geometry not aligned with casual photo

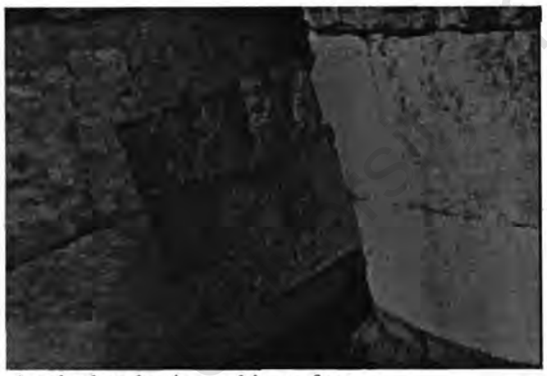
Figure 47 shows the filling of an occlusion using a single "quadrilateral" corresponding with the occlusion boundary. The image in the left shows the geometric model orientated at approximately the same position as the casual photograph. The right image, which has another orientation, exposes the problems incurred with ignoring the geometry where a corner exists.

a) location of area being studied

b) photogrammetrically captured showing border

c) calculated using flat surfaces with straight corner[18]

d) calculated using flat surfaces with detected corner


e) calculated using cubic surfaces closed

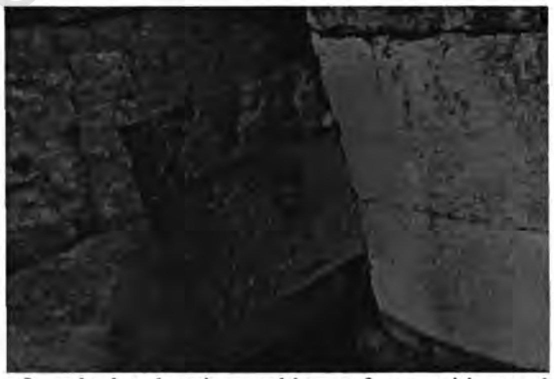

f) calculated using cubic surfaces with crack

Figure 48. A medium range view of the broad corner showing the use of various techniques for calculating the geometry of an occluded corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site; images b) through f) show the results of using different techniques for calculating the corner geometry.

Figure 48 illustrates, firstly that the corner calculating can be used to good effect; but secondly that at this scale there is no distinguishable difference between the results using a variety of different geometry calculating techniques. In order to try to establish which technique provides the best results it is necessary to take a closer look at the filled corner. Figure 49 shows a very close-range view of the same corner. It can been seen that the

[18] a straight corner refers to the join between two largely flat surfaces. The term straight is used to indicate that this join occurs along the straight line between the join's ends, that is the straightness is orthogonal to the corner.

assumption of flat surfaces with a straight corner (Figure 49c) provides that best result; marginally better than Figure 49d, which is based on a detected edge.

It is clearly shown from the spurious deviation of the corner in Figure 49e in particular that the assumption of a cubic surface is inappropriate. The implication of this is that where the surfaces are in fact essentially flat, errors can be introduced by the imposition of a cubic surface.

a) location of area being studied

b) photogrammetrically captured showing border

c) calculated using flat surfaces with straight corner

d) calculated using flat surfaces with detected corner

e) calculated using cubic surfaces

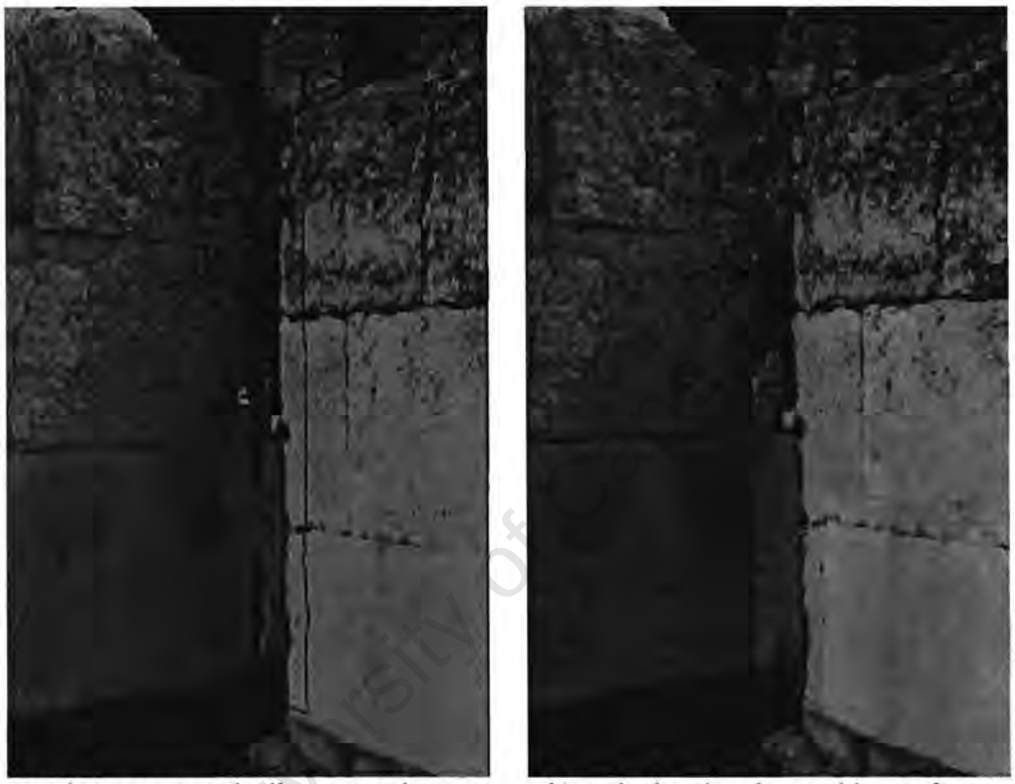
f) calculated using cubic surfaces with crack closed

Figure 49 A very close-range view of the broad corner showing the use of various techniques for calculating the geometry of an occluded corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site; images b) through f) show the results of using different techniques for calculating the corner geometry.

Comparison of Corner Models applied to a narrow occlusion

Similarly to the evaluation of corner models based on a broad occlusion contained in the previous section, the corner models are evaluated based on a narrow occlusion in this subsection. A narrow occlusion is used because the surrounding non-occluded geometry is expected to have a different (greater) effect on the geometry calculated.

It can be seen in Figure 50 that a narrow occlusion containing a corner can be filled. As was the case with the broad corner (see Figure 48) little difference can be noticed from a medium range in the results generated by the various corner-calculating algorithms.



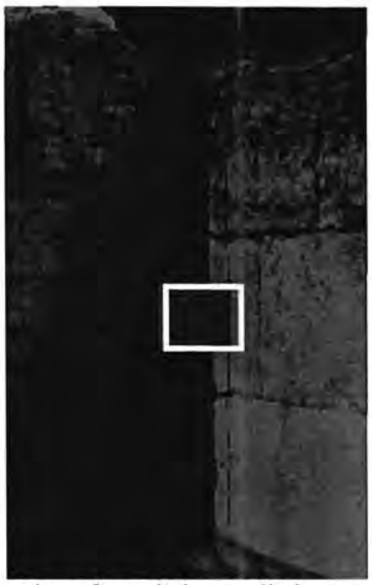
b) photogrammetrically captured showing border

b) calculated using cubic surfaces, detected corner without closing gap

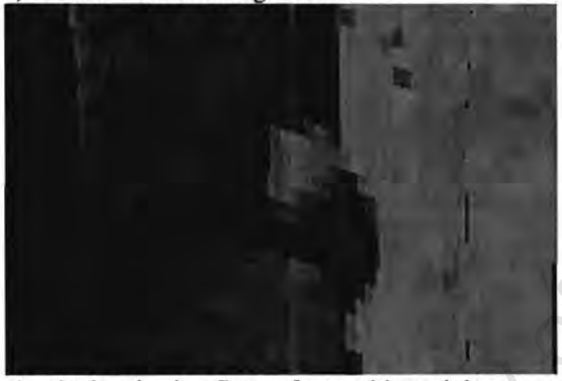
Figure 50 showing the use of the tool to fill an occlusion. It can be seen within the occlusion boundary in a) that some actual occlusion exists in the photogrammetrically captured model.

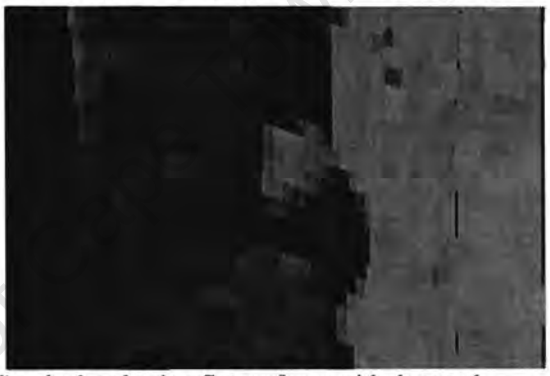
Once again, in order to try to establish which algorithm provides the best results it is necessary to take a closer look at the filled corner. Figure 51 shows a very close-range view of the same corner. In this case it is practically impossible to perceive any differences between the uses of the alternative algorithms.

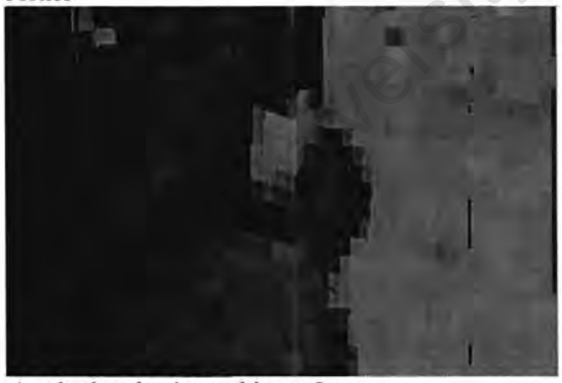
The presence of the "crack" in the centre of Figure 51e (due to the fact that the pair of cubic surfaces used to approximate the adjacent sides of the corner do not meet precisely) is evidence that these surfaces are better treated as planar. However, the result due to the cubic assumption but with the "crack" closed by calculating an average value (see Figure 51f) is almost indiscernible from the results calculated using planar surfaces.


a) location of area being studied

b) photogrammetrically captured showing border


c) calculated using flat surfaces with straight corner


d) calculated using flat surfaces with detected corner


e) calculated using cubic surfaces closed

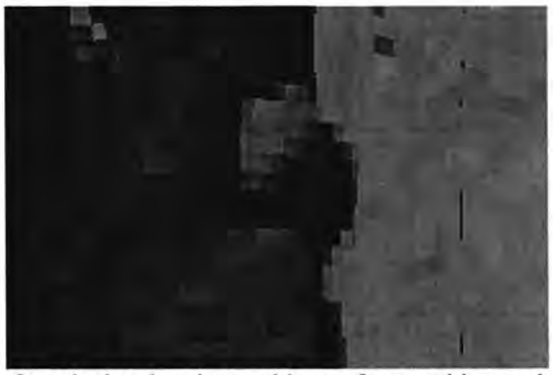

f) calculated using cubic surfaces with crack

Figure 51 A very close-range view of the narrow corner showing the use of various techniques for calculating the geometry of an occluded corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site; images b) through f) show the results of using different techniques for calculating the corner geometry.

It can thus be seen that the algorithms provide satisfactory results when correctly applied. For example when the surfaces are flat the assumption of planar surfaces provides good results but the assumption of a cubic surface introduces errors and vice versa.

5.2.4 Texture adjustment

Most of the evaluation images shown thus far have included a degree of image intensity and/or contrast adjustment; Figure 52a shows the effect of not applying any adjustment to the image. It is clear that the texture adjusting facility is important to the provision of convincing occlusion patches.

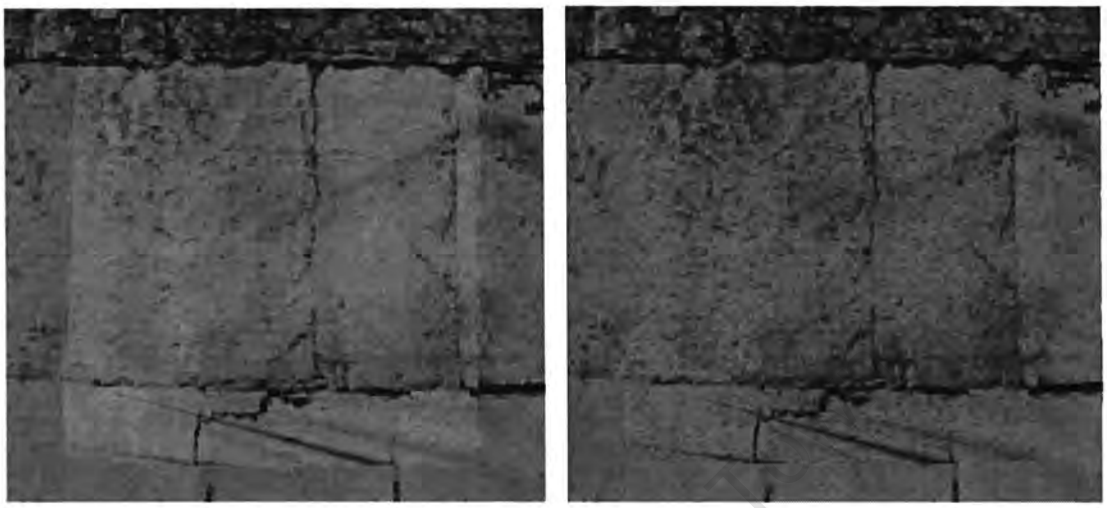


Figure 52 shows the use of the texture adjuster. The borders of the patch are far more visible with and unadjusted patch (a) than when the images is adjusted (b) by applying a 10% reduction in intensity.

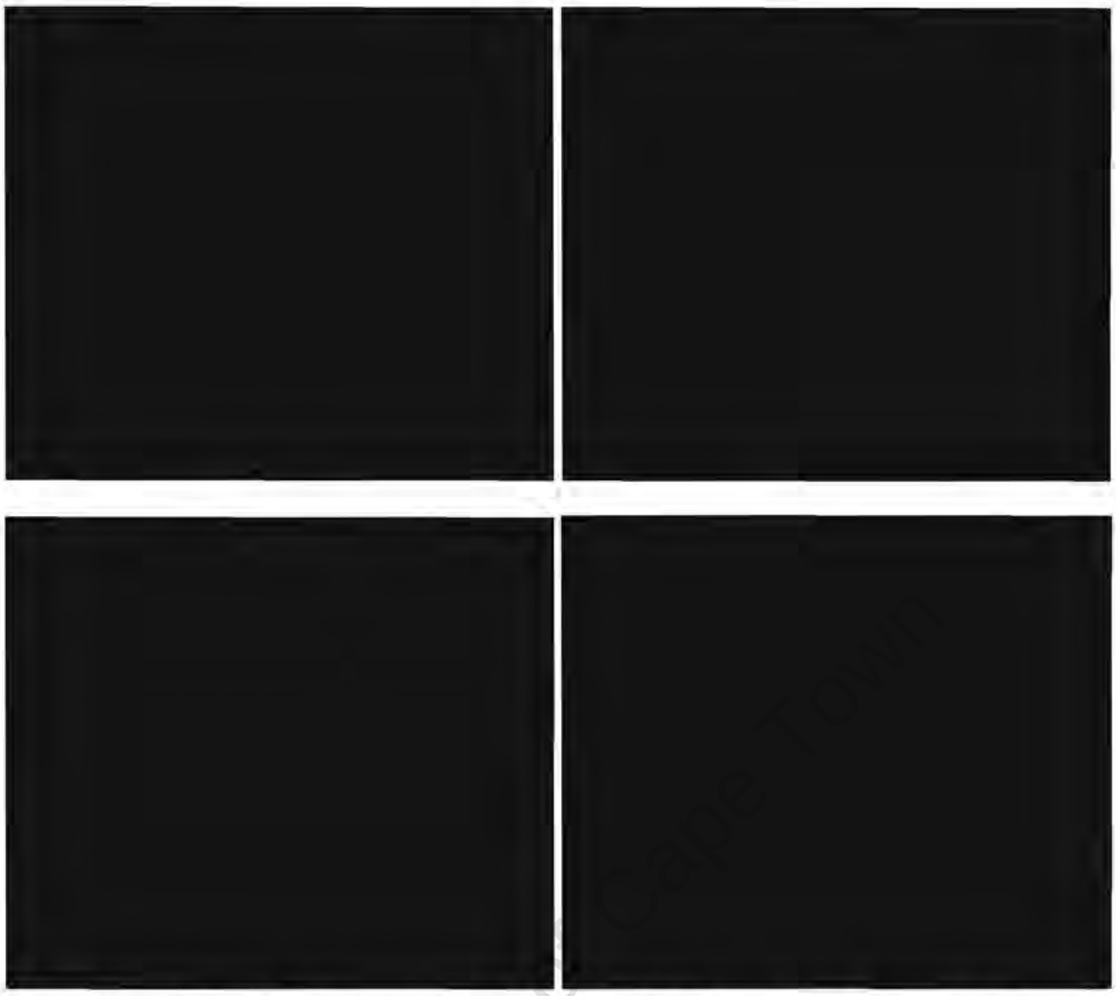


Figure 53 showing the effect of various texture adjustments. Clockwise from the top left are unadjusted; contrast reduced by 40% and intensity increased by 20%; contrast reduced by 40% and intensity increased by 30%; contrast reduced by 40% and intensity increased by 40%;

Texture adjustment is more difficult when the orientations of faces in the image have great variation and are photographed in direct sunlight. As can be seen in Figure 53 either the one or the other surface can be made to match at the cost of the level of match of the other. Also texture is lost in the adjustment process.

The primary weakness of the texture adjustment functionality provided is the fact that the adjustment is applied uniformly to the entire image. A better, but potentially far more difficult method to implement would be to apply some type of texture matching along the edges (that is the regions where casual photograph pixels and photogrammetric pixels lie next to one another). In addition it would be useful to enhance specifically certain features (typically grooves in the evaluation data set) to match the same features outside of the image.
Such a high degree of image manipulation effectively reduces the utilisation of casual photograph data to the extent that it is used *only* for geometry calculation, with the texture provision aspect being replaced by what is effectively texture synthesis based on surrounding photogrammetric data. Due mainly to the overriding requirement for accuracy in models to be used for research, the use of casual photographs is preferred rather than artificial texture synthesis. However, it is recognised that such an approach may provide more satisfactory results from a visual point of view.

5.2.5 Summary of constituent technology evaluation

The results of the evaluation of constituent technologies are summarised in Table 2 below.

Technology	Results of evaluation
filters (**§5.2.1**)	All filters types (namely box filter, Gaussian, difference of Gaussian) function as expected. Although the filters are functional they do not improve the detection of boundaries in the sample set of images; in fact the box filter and Gaussian filters reduce the effectiveness of boundary detection.
boundary detection (**§5.2.1**)	The boundary detection algorithm implemented, based on "Live Wire" of Mortensen and Barratt [52] provides accurate boundary localisation within usable turn-around-times. The "straightness" cost-term of the Live Wire algorithm appears to be redundant for the image set used.
calibration (**§5.2.2**)	Satisfactory calibration and occlusion boundary projection is obtained for sets of non-planar points using the pinhole camera model [27]. Calibration provides a robust and practical basis for clipping from the casual photograph.
geometry calculation (**§5.2.3**)	All algorithms (namely planar triangles, flat with occlusion boundary edges, cubic- and quadratic polynomial surfaces) provide satisfactory results when applied appropriately. That is, problems arise, for example if a cubic surface is fitted to what is in reality a flat surface, etc. The set of algorithms implemented is believed to be complete in the sense that the range of occlusion configurations from flat, though curved to material discontinuities (corners) is covered.
texture adjustment (**§5.2.4**)	Functional but of limited use. This is because the technique implemented applies a similar adjustment in intensity, contract and colour to all pixels. In order to be more useful the technique should match edges of patch with surrounding texture.

Table 2 Summary of the evaluation of constituent technologies.

5.3 Evaluation of the tool

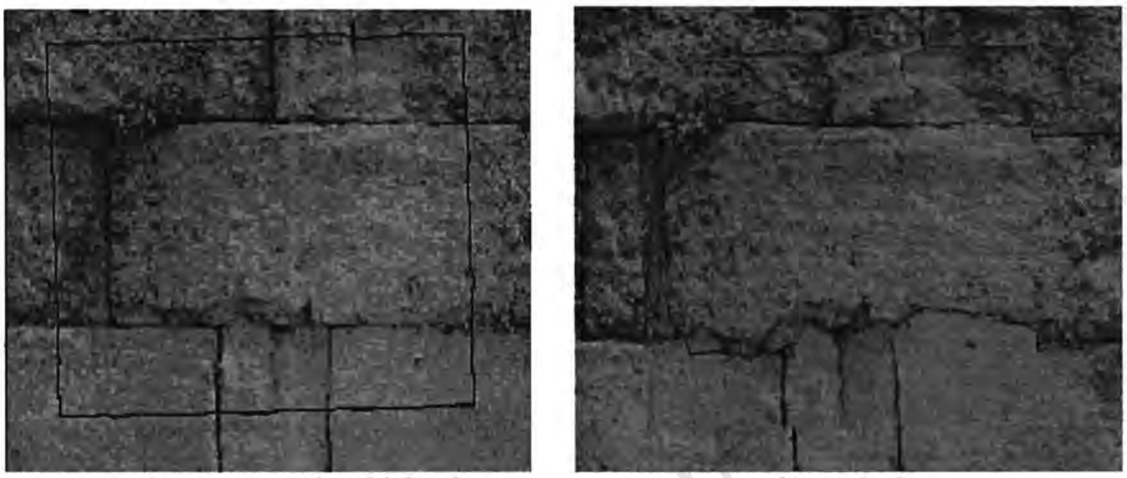
In addition to the separate evaluation of underlying technologies it is necessary to evaluate the tool, which is an integration of these technologies, as a whole. The evaluation of the tool concentrates on its completeness when being applied to a range of likely occlusion configurations and casual photograph types rather than upon the relative capabilities of sub-components.

The test cases used are the flat surface, curved surface, broad- and narrow corner (as described in §5.1) because they best cover the set of elemental geometries likely to be encountered in archaeological sites.

In addition, the effect of using different casual photographs is evaluated in order to establish to what extent our results are dependent on the location of the camera (see §5.3.5) and the resolution of casual photographs (see §5.3.6).

5.3.1 *Flat surface*

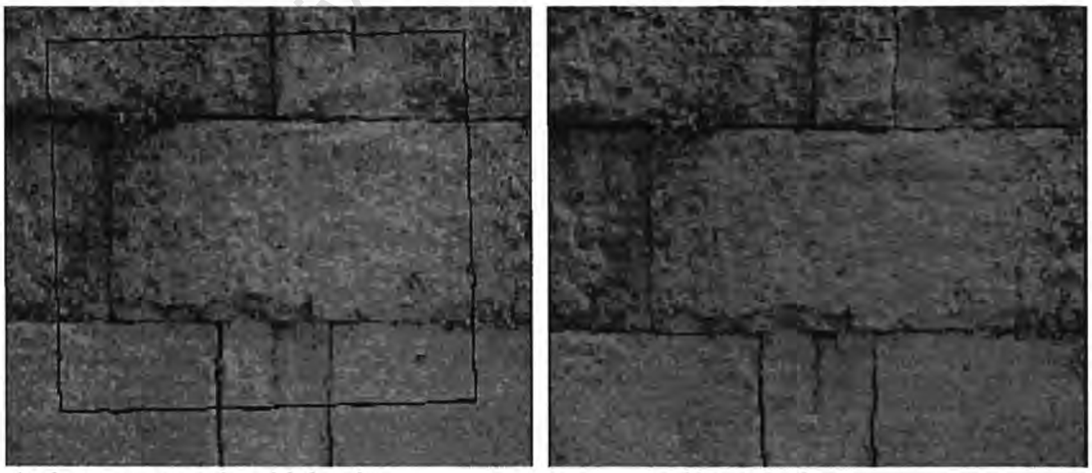
As has been demonstrated in §5.2.2 , which describes the problems of calibration based on planar features, a calibration/occlusion boundary based approach is not always suitable. Figure 54 shows the poor results of using a projected occlusion boundary to extract a texture patch in the case of a flat surface.



a) photogrammetric with border b) patched

Figure 54 showing the fix applied to the flat wall using the occlusion boundary, the edge locations of which have been calculated from the calibration of the casual photograph.

An alternative approach available within the tool is to manually mark the corners of the occluded region. If great care is taken in marking corresponding points a good fit can be obtained. The process of tuning the corresponding points is facilitated by the tool because points can be moved and the fit viewed until a good fit is obtained. Figure 55 shows the fit obtained by this approach (the geometry of the occluded region is assumed to consist of a pair of triangles).



a) photogrammetric with border b) patched

Figure 55 showing the fix applied to the flat wall using a pair of triangles the corners of which have been manually located on the casual photograph.

Although the fix shown in Figure 55 is sufficiently accurate at that scale, on closer and oblique inspection (Figure 56) it can be seen that the edges do not match up because the patch is flat whereas the wall is rough. Alignment of the inter-block grooves is satisfactory, albeit also suffering from minor problems due to the lack of three-dimensionality of the patch.

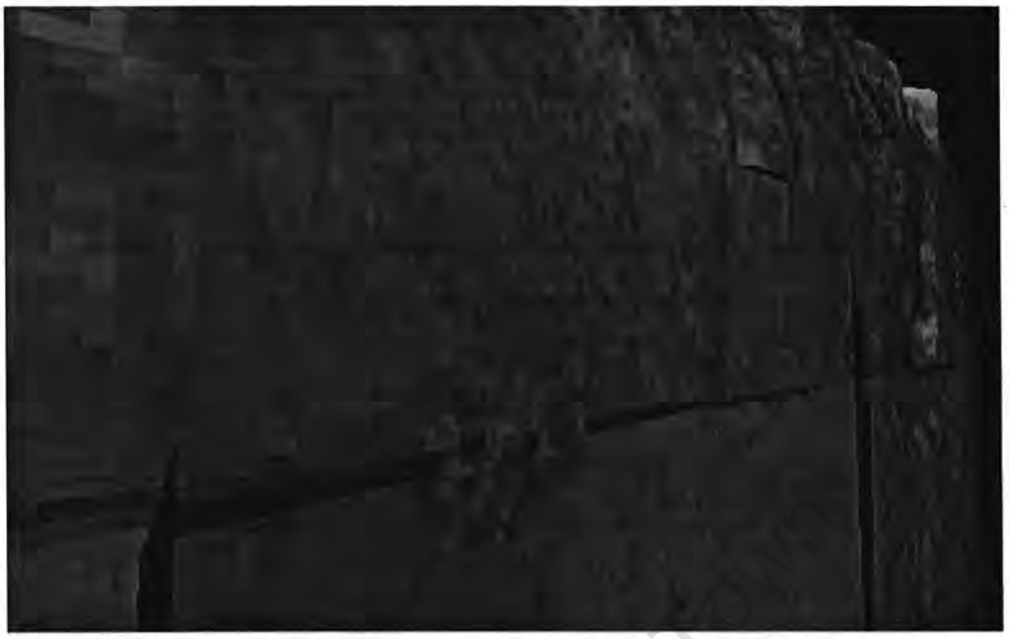
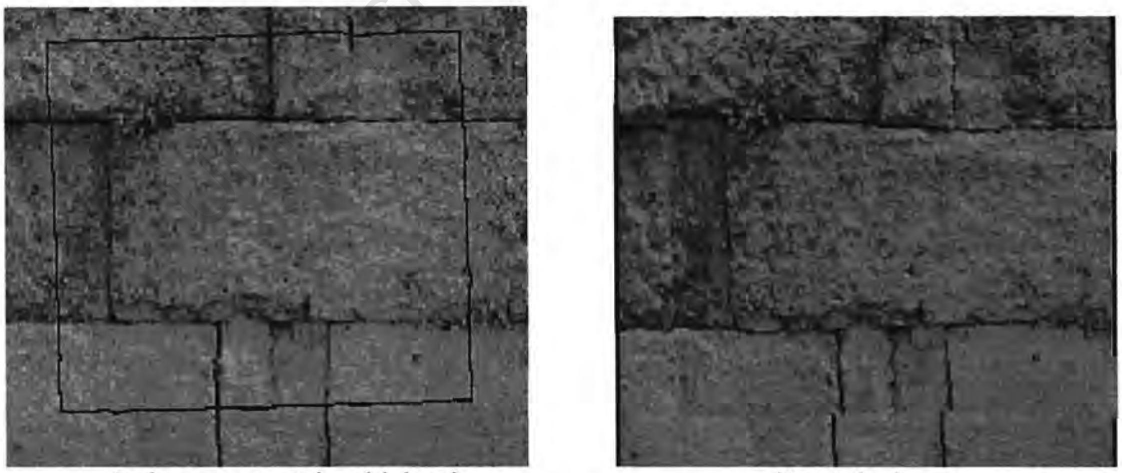


Figure 56 shows the edges do not meet when filling with a pair of planar triangles due to the fact that the underlying geometry is only approximately planar. Note the gaps between the planar patch and underlying geometry along the near and far sides of the patch.

Great care does, however, have to be taken when indicating correspondences between geometry and casual photograph. Figure 57 shows the difficulty of clipping the correct patch (without the aid of calibration and occlusion boundary projection). Here an initial estimate (rather than finely tuned by observation of the results) of corner positions has been used. It is clear from Figure 57 that the lines are both out of alignment and distorted (cf. Figure 55b).



a) photogrammetric with border b) patched

Figure 57 showing the fix applied to the flat wall using a pair of triangles the corners of which have been manually located on the casual photograph. In this case an initial estimate of corner positions has been used rather than the finely tuned positions used in Figure 55 (in order to demonstrate the sensitivity of patch quality to correspondence localisation by the user).

In summary, although the use of calibration is not suitable for largely planar subjects, the tool does provide another means to address such subjects, that is by the use of planar triangles and manual identification of corner points, which if used with care is effective. A minor weakness of this approach is the lack of match-up of the patch edges with the surrounding geometry due to the patch edges being flat. This is viewed as a minor weakness because the lack of match-up is inversely proportional to the flatness; and this approach is only necessary when the surfaces are flat or very nearly so.

5.3.2 Curved surface

The curved surface is far more amenable to calibration because it makes non-co-planar points available (see §5.2.2) which results in a better patch being obtained (as shown in Figure 58). The filling of occlusions based on curved surfaces does not use any complex hidden geometry calculation.

All that is required from the user in this case is the identification of six or more calibration points and four or more correspondences falling on the occlusion boundary.

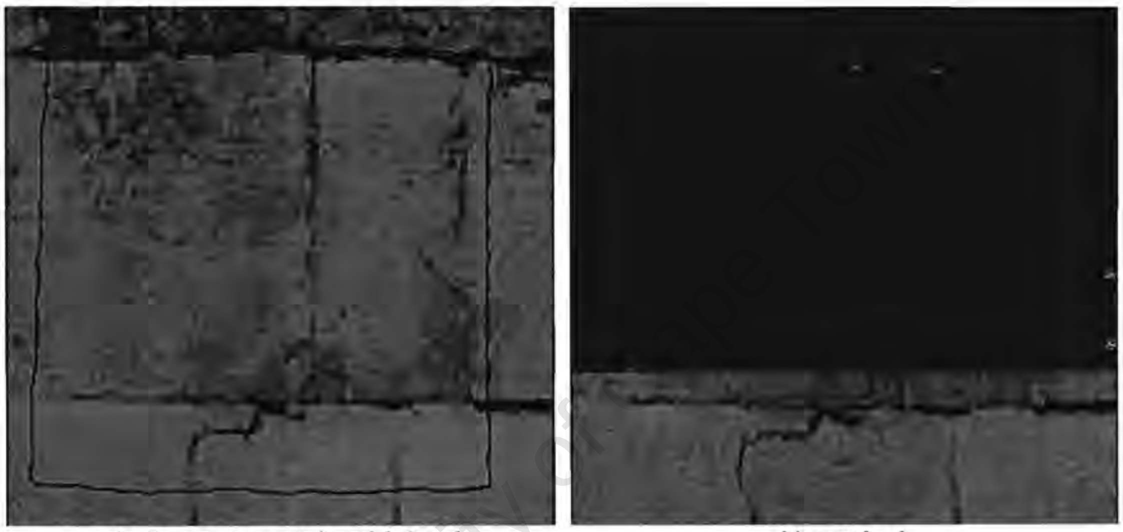


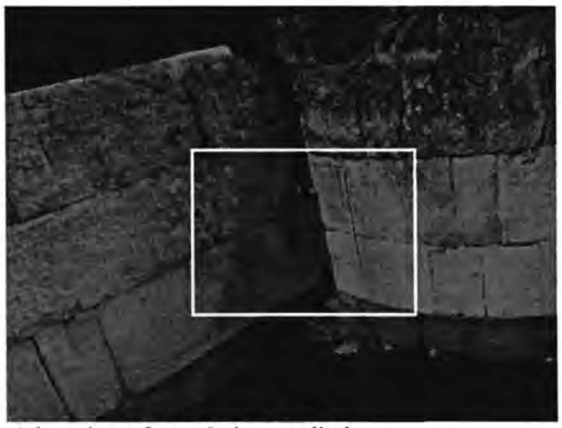
Figure 58 showing the fix applied to the curved wall using the occlusion boundary, the location of which has been calculated from the calibration of the casual photograph. The good groove alignment and lack of distortion is very apparent in b)

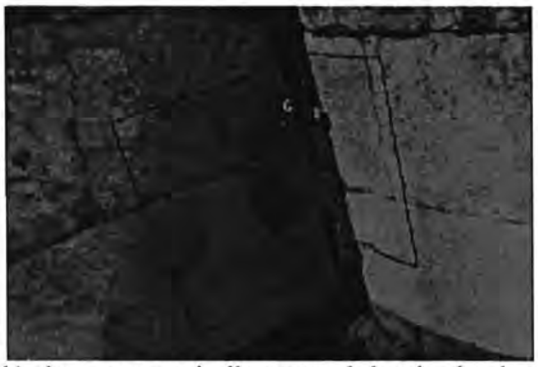
Thus in the case of a curved surface, the tool provides an accurate patch with a very modest amount of user input.

5.3.3 Broad corner

Here we evaluate the tools ability to deal with a corner which lies within a large occlusion. The significance of a large occlusion is that the occluded corner is surrounded by surfaces which are also largely occluded. This means that there is little information to guide the calculation of geometry in the proximity of the corner.

Figure 59 illustrates that the tool can be used to good effect to fill the occlusion in the case of a corner within a broad occlusion.


a) location of area being studied


b) photogrammetrically captured showing border

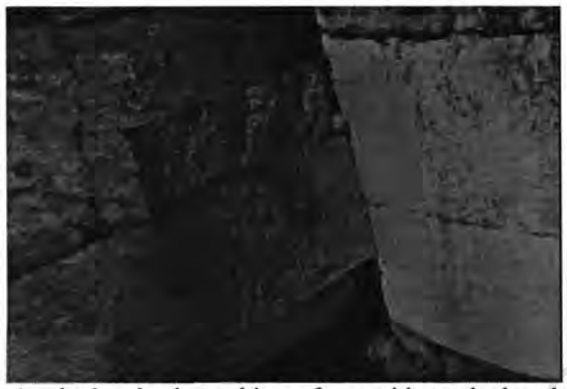
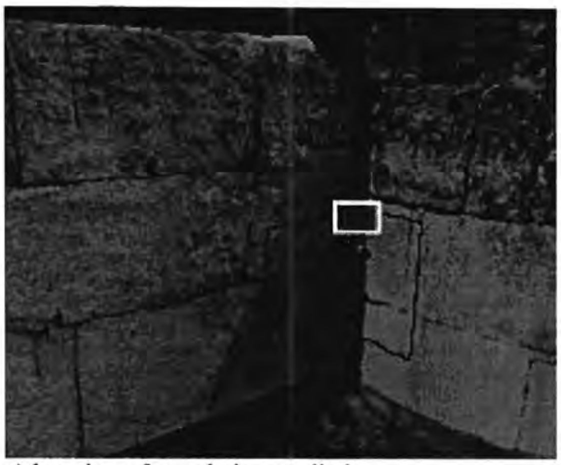
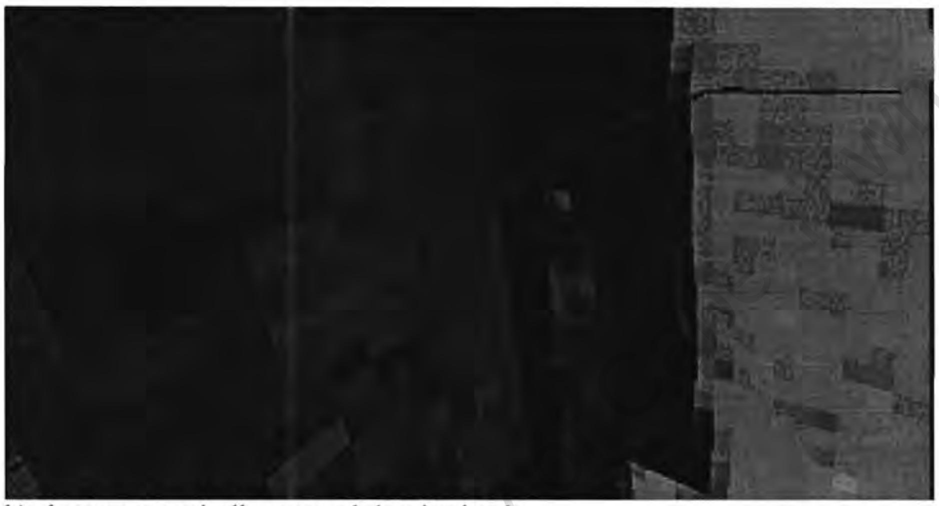

c) calculated using cubic surfaces with crack closed

Figure 59. A medium range view of the broad corner showing the use of various techniques for calculating the geometry of an occluded corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site; c) shows the results of using the tool to calculate the corner geometry.

On closer inspection (Figure 60 is a very close-range view of the same corner) it can be seen that the results are not perfectly accurate. The errors are however, only perceivable at a very small scale.



a) location of area being studied



b) photogrammetrically captured showing border

c) filled using the tool

Figure 60 A very close-range view of the broad corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site, b) the reference photogrammetrically captured corner and c) shows the corner as filled using the tool.

5.3.4 Narrow corner

In the case of the narrow corner, the reference photogrammetrically captured model is actually incomplete for the first time in the series of evaluation models. This is due to the fact that much of the narrow corner is obscured in the photogrammetric image set (due to the curved wall obscuring the flat wall in one of the stereo pair of images of the flat wall - see Figure 61). This scenario, which one would expect to occur fairly commonly where pairs of images of a subject captured from different locations are required, allows the demonstration of the tool for its intended use.

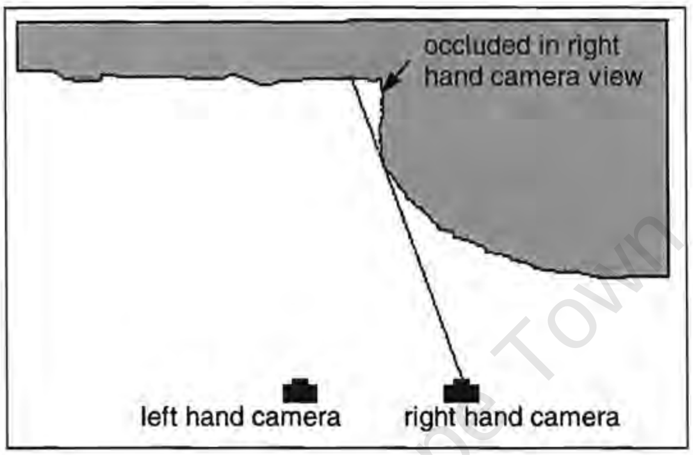
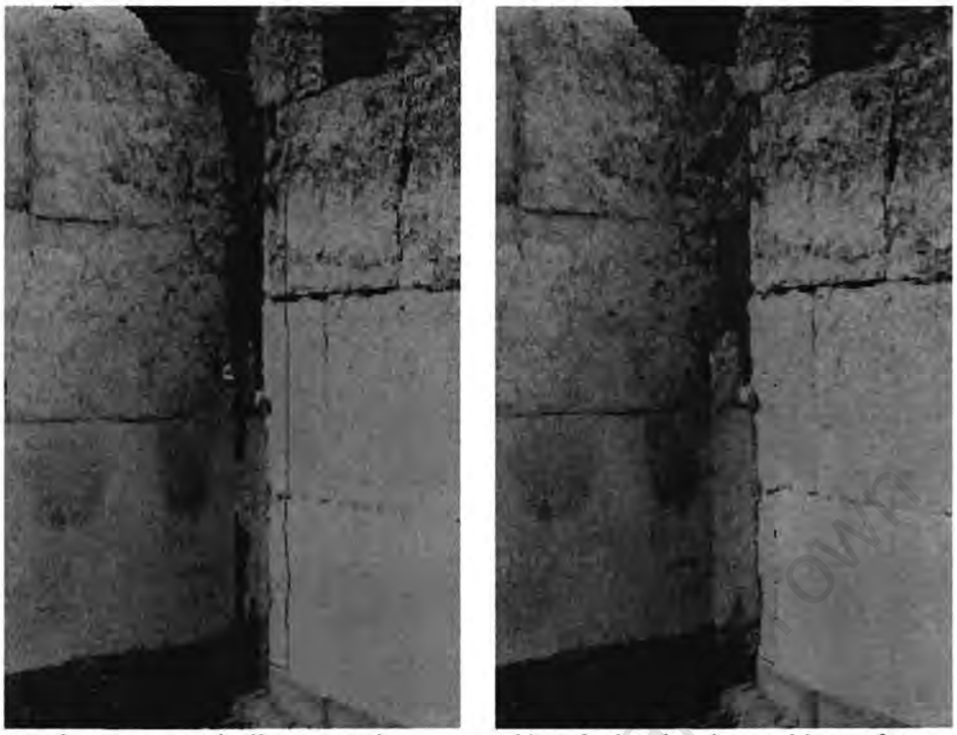


Figure 61 showing a top view of a typical corner between two walls. Because the corner is occluded in one of the camera's views the geometry for this region cannot be extracted by photogrammetric means.

Conversely to the previous section, here we are evaluating the tools ability to deal with a corner which lies within a narrow occlusion. The significance of a narrow (as opposed to large) occlusion is that there is abundant information to guide the calculation of geometry in the proximity of the corner.

It can be seen in Figure 62 that the tool can be used to fill a narrow occlusion containing a corner. It is very apparent that the model is improved by filling the occlusion.

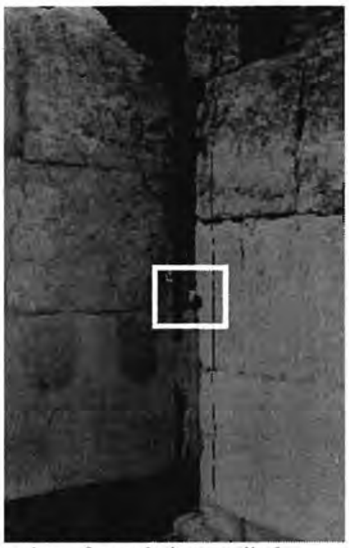


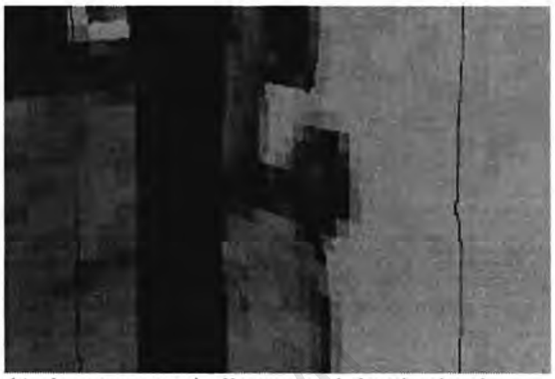
b) photogrammetrically captured showing border

b) calculated using cubic surfaces, detected corner with gap closed

Figure 62 showing the use of the tool to fill an occlusion. It can be seen within the occlusion boundary in a) that some actual occlusion exists in the photogrammetrically captured model. The corner as filled using the tool is shown in b). The fill used here has been extracted from the left hand camera image.

On very close inspection (see Figure 63) it can be seen that even at a very small scale the results are remarkably accurate (cf. the broad corner in Figure 60 which is less accurate). This is due to the proximity of known geometry which the basis for the calculation of the corner geometry.


a) location of area being studied


b) photogrammetrically captured showing border

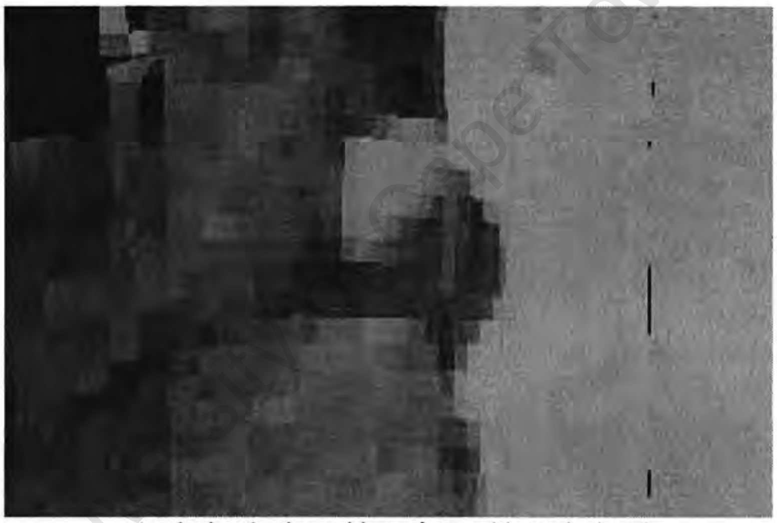
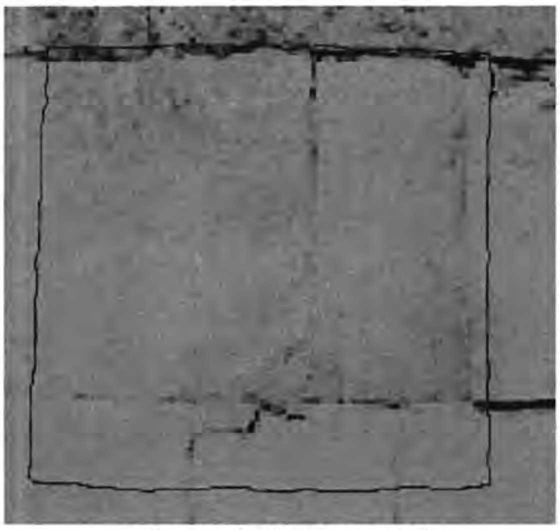

c) calculated using cubic surfaces with crack closed

Figure 63 A very close-range view of the narrow corner showing the use of various techniques for calculating the geometry of an occluded corner. Image a) indicates the position of the region being studied within the surrounding photogrammetrically captured site; c) shows the results of using the tool to calculate the corner geometry.

5.3.5 *Casual photographs from different positions*

The results of the tool must not be overly sensitive to the position from which the casual photograph is taken. In this section the tool is used to fill the same occlusion from a pair of casual photographs which differ only in the positioning of the camera. This is done in order to establish the sensitivity of results to the position of the camera.


orthogonal photogrammetric

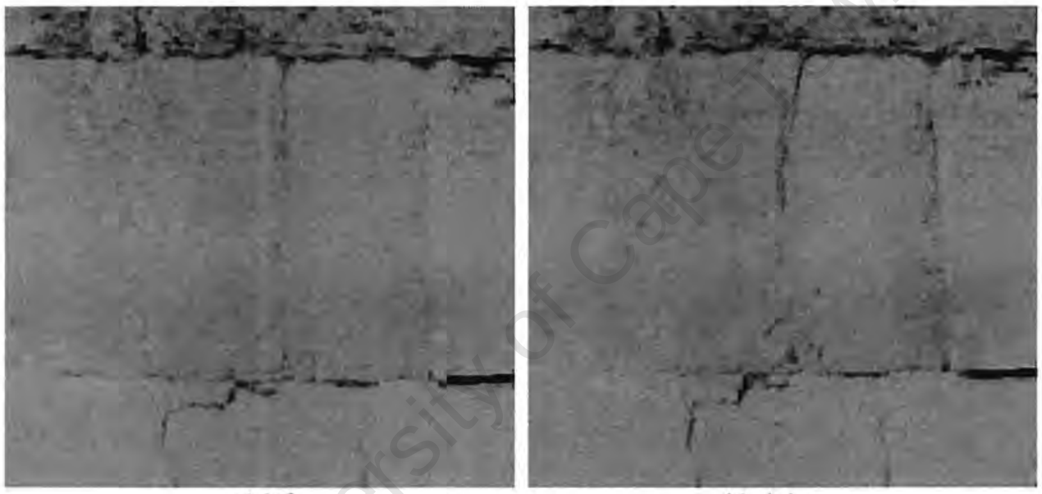

a) left b) right

Figure 64 showing the effect of using casual photographs taken from different positions.

It can be seen from Figure 64a that the occlusion patch calculated using the casual photograph taken from the left position gives better results than that captured from the right position (see Figure 64b). This is due to the left image being more orthogonal (in other words facing more squarely) to the wall resulting in less distortion due to the fact that the wall is not flat. Wherever possible casual photographs should be captured as orthogonally as possible.

5.3.6 Low resolution images

Casual photograph resolution is expected to be a very significant factor in the quality of tool output. Beyond the obvious reduction in patch texture detail due to low-resolution images; the selection of correspondences for calibration (and this occlusion boundary projection) and the selection of correspondences for geometry calculation are likely to be negatively effected with a commensurate effect on the accuracy of patch calculation.

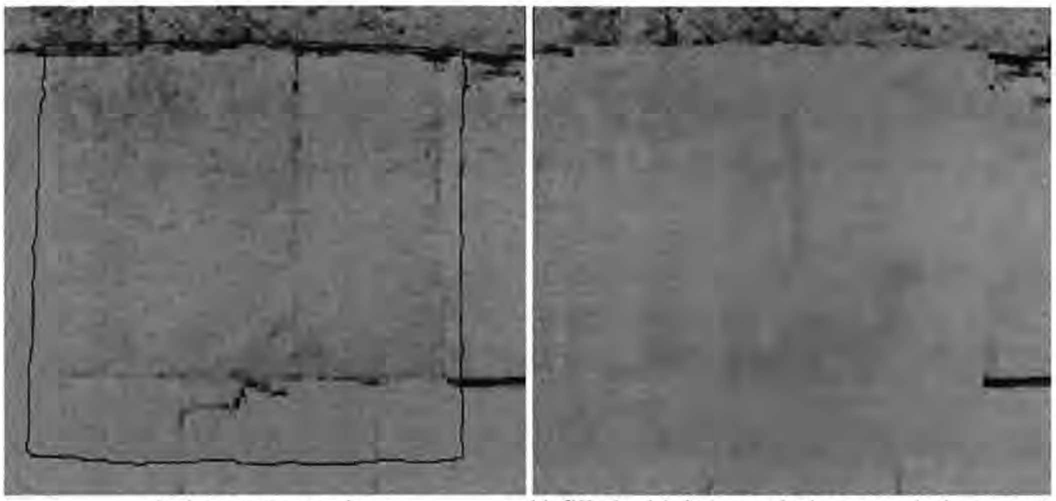


Figure 65 showing the use of a low-resolution (192 X 128) image. Compare with Figure 64 which is based on an 852 X 604 pixel image.

As expected, when the casual photograph is of very low resolution it provides a very poor basis for calculating a patch (see Figure 65).

5.3.7 Summary of tool evaluation

The results of the evaluation of the tool are summarised in **Table 2** below.

Aspect	Results of evaluation	Recommendations
occlusion configuration (§5.3.1 , §5.3.2 , §5.3.3 , §5.3.4)	Can deal effectively with the entire range of likely occlusion configurations, that is the tool can provide accurate patches with a modest amount of user input. This has been demonstrated using flat- and curved surfaces and broad and narrow occlusions over corners. Furthermore, by comparing patched with un-occluded geometries at very close ranges it has been shown qualitatively that the results are accurate. In the case of the narrow corner, where the datum photogrammetric model is not complete, it is very apparent that the geometric model is improved by the filling-in of the occlusion.	The tool should be utilised to enhance three-dimensional models of archaeological sites by filling occlusions due to the techniques used to capture the scene geometry and texture.
	However, the tool deals less well with scenes having entirely planar geometry (for example a scene which	Casual images should, wherever feasible, include some non-co-planar

Aspect	**Results of evaluation**	**Recommendations**
	contains only a single nearly flat wall with an occlusion on the wall) due to the difficulty of calibrating an image of such a scene. This implies that the user must manually mark the location of features near the occlusion boundary on the casual photograph in order to extract texture patch. The accuracy of the patch in this case is thus dependent on the amount of time and effort expended by the user.	features to allow robust calibration.
position of casual photograph (§5.3.5)	Results are moderately sensitive to the position from which the casual photograph is taken. The more orthogonal the camera axis is relative to the surface the more accurate the results.	Casual photographs should, wherever convenient, be captured square-on to surfaces containing occlusions. This may imply a pair of photographs is required to capture an occluded corner for example.
quality/resolution of casual photograph (§5.3.6)	Poor results if the casual photograph is of very low resolution.	Casual photographs must be of as high a resolution as possible. Where only relatively low-resolution images are available it is preferable to use texture synthesis to fill the occlusion.

Table 3 Summary of the evaluation of the tool.

6 Conclusions

This dissertation has presented a tool for extracting texture data from casual photographs for filling-in occlusions in photogrammetrically captured models of archaeological sites.

The tool is complete in that it contains all the functionality needed in the patch creation process: from importation of images in a wide range of formats and geometry in the well-known (and open) Drawing Exchange Format; to the generation of VRML Standard formatted output of patch geometry and texture.

The tool built is based on the implementation of algorithms as follows: (along with supporting functionality such as the ability to read and write files etc)

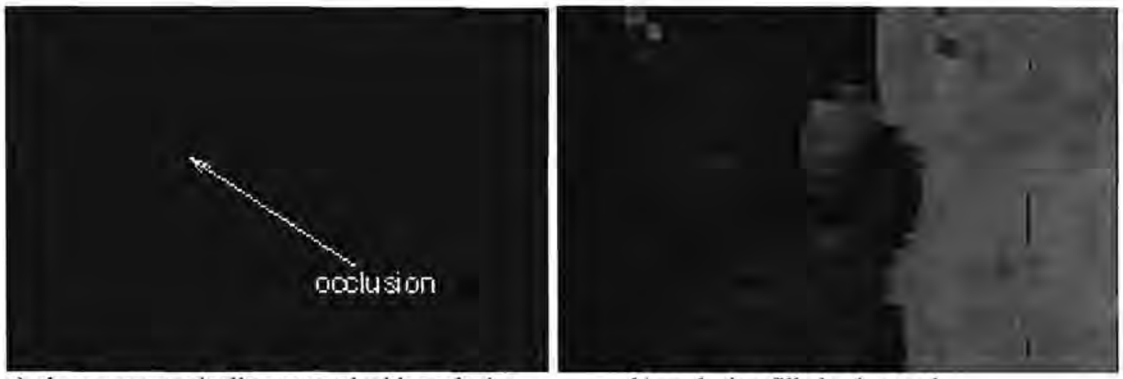
1. *casual photograph calibration* - This is the calculation of the parameters which define, amongst others, the camera position and orientation which can be used to relate the locations of image pixels to locations in the three-dimensional scene.
2. *image filtering* - images are filtered in order to improve the detection of boundaries.
3. *boundary detection* - Regions of casual photographs are used to provide texture patches to occluded parts of the scene. The boundaries of these regions need to be detected.
4. *geometry calculation* - Many occlusions include discontinuities, that is they are not simply flat or curved surfaces. Algorithms have been developed and implemented to estimate such discontinuous geometry within occluded regions.
5. *texture adjustment* - A facility to adjust the patches of texture extracted from casual photographs so that they more closely match the photogrammetrically captured images.

Furthermore the tool has been found to be effective, to produce results which are sufficiently accurate (see §5.3.2 , §5.3.3 and §5.3.4) and to deal well with the types of occlusions which are likely in photogrammetric surveys of archaeological sites (see §5.1).

The feasibility of using casual photographs to fill occlusions in three-dimensional models has thus been demonstrated. The tool provides accurate patches from typical data sets (occluded photogrammetrically captured model and casual photograph set). The three-dimensional geometric model is clearly improved by the filling-in of the occlusion, as is demonstrated in the case where the underlying photogrammetric model is not complete (see Figure 66).

There is no specific limit on the size of occlusion that may be filled. Size is expected to be limited by the size of largely planar elements in actual geometry.

The sample occlusions used to evaluation were typically patched in about 30 seconds once the user become familiar with the tool.



a) photogrammetrically captured with occlusion b) occlusion filled using tool

Figure 66 showing the effect, from very close range, of filling an occlusion over a corner, in this case a the narrow corner (see §5.3.4 for a more detailed discussion)

The tool has been applied to occlusions of various configurations. In addition to dealing well with the narrow occlusion over a corner, shown above, it has been found to deal well with flat surfaces (see §5.3.1), curved surfaces (see §5.3.2) and broad corners (see §5.3.3).

The tool deals less well with scenes having entirely planar geometry (for example a scene which contains only a single nearly flat wall with an occlusion on the wall) due to the difficulty of calibrating an image of such a scene (see §5.2.2). This implies that the user must mark the location of features near the occlusion boundary on the casual photograph in order to extract a texture patch corresponding to an occlusion. It is felt, however, that this configuration is unlikely to be encountered often because occlusions are expected to be rare in scenes containing large flat surfaces.

It was found that low resolution images were not useful for filling occlusions (see §5.3.6) not only because they proved poor texture for patches but also because they provide insufficient information for calibration and boundary detection and thus for geometry calculation.

The position from which the casual photograph is taken has a moderate influence on resultant patches (see §5.3.5). The more orthogonal the camera axis is relative to the surface the more accurate the results.

The various technologies which comprise the tool have been evaluated in isolation with the following results:

- image filters (§5.2.1) - functional but do not improve the detection of boundaries in the sample set of images. All filters types (namely box filter, Gaussian, difference of Gaussian) function as expected.
- boundary detection (§5.2.1) - implementation of "Live Wire" algorithm of Mortensen and Barratt [52] provides accurate boundary localisation within usable turn-around-times.
- calibration (§5.2.2) - implementation of the pinhole camera model [27] provides a robust and practical basis for clipping from the casual photograph.
- geometry calculation (§5.2.3) - All algorithms provide satisfactory results when applied appropriately. The set of algorithms implemented is believed to be complete.
- texture adjustment (§5.2.4) - Functional but of limited use. This is because the techniques implemented apply a similar adjustment to all pixels.

Direct output to VRML Standard format provides the results of the tool in a form usable by a wide community of users.

Although the tool is primarily based on the implementation of existing algorithms, the following novel aspects are worth noting:

- The tool is suitable for *irregular shapes* typically present in archaeological scenes rather than being based on assumptions of regularity.
- *Calibration* of the casual images because this exploits the fact that casual photographs will include many features which have been accurately surveyed (see §3.1).
- *Projecting the boundaries of occlusions* onto images as a means of segmentation of the image and providing the boundary for integration of texture types, namely casual photograph and photogrammetrically captured (see §4.3.3).
- The *least mean squares algorithm for the computation of the hidden geometry* because this algorithm implemented makes the greatest use of information available (namely surrounding geometry, edge shape in the casual photograph and user segmentation of the image) and the least use of assumptions (see §4.3.7).

6.1 Recommendations for casual photograph capture

The underlying motivation for the current work is to allow the creation of more complete models based photogrammetry supplemented by casual photographs. In order to improve the resulting three-dimensional models the following factors should be taken into account when capturing casual photographs:

Casual images should, wherever feasible, include some *non-co-planar features* to allow robust calibration.

Direct sunlight and spectral highlights should be avoided because the result in images beyond the effective range of the texture adjuster. This results in patches which either retain visible edges (because they have not been adjusted sufficiently to match the surrounding texture) or exhibit a visible loss of image data (because by matching the edges the contrast and/or intensity of the image has been changed too much).

Casual photographs must be of the *highest resolution* possible.

Casual photographs should, wherever convenient, be captured *square-on to surfaces* containing occlusions. This may imply that a pair of photographs is required to capture an occluded corner for example.

6.2 Recommendations for further work

The tool currently *only addresses casual photographs which include some non-planarity*. To be more specific; the tool will deal well with any image which includes at least a reference point which is not co-planar with the others, or with a set of two or more planes. A limitation, however, manifests itself when all points in the image are co-planar, such as is the case when *only part of a flat wall* appears in the image *in the absence of any other features*.

The tool could be enhanced by adding facilities to match corresponding linear features (such as cracks) on the casual photograph with those in the visual geometry and then use these matched features as the boundary of the occlusion for clipping purposes.

Implementing such functionality is thought to be beyond the scope of this project because:

> the authors wanted to concentrate on the more typical non-planar features,
>
> images which only include features which are co-planar are likely to be rare,
>
> the techniques which underlie such functionality are varied, complex and are an entirely different means of finding correspondences between image and geometry (cf. occlusion boundary projection which is the basis of integration of occluded and photogrammetric in this work),
>
> most importantly considering the low frequency with which such images are likely to be encountered, require relatively extensive effort to implement and
>
> manual matching of features is available as an option to address these cases.

The *texture adjuster* could be improved by implementing an algorithm which functions by matching the edge textures while applying as little as possible adjustment to the centre of a patch. This task has not been tackled as part of the current project because it is fairly isolated in terms of both its integration with the rest of the tool (feasibly one could be found or built and plugged-in) and in terms of its underlying algorithms.

Bibliography

[1] Adelson, E.H., and Bergen, J.R. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, Landy and Movs - hon., Eds. MIT Press, Cambridge, Massachusetts, Ch. 1, 1991

[2] Amini, A.A., Weymouth, T.E., and Jain, R.C. Using Dynamic Programming for Solving Variational Problems in Vision. IEEE Trans. PAMI, 12, no. 2, 855-866, 1990

[3] Baum, J., Robinson, D., Carrington, P., Mason, D. & Strickland, T., The Chester Project: Reconstructing Roman Chester http://dialspace.dial.pipex.com/julianbaum/tcp.shtml as viewed on 8 February, 2000., 1998

[4] Beardsley, P., Torr, P. and Zisserman, A. , "3D Model Acquisition from Extended Image Sequences", In Proc. ECCV, LNCS 1064/1065, pages 683--695. Springer-Verlag, 1996.

[5] Beaudet, P.R. "Rotationally Invariant Image Operators," Proceedings ICPR, p.579-583, 1979.

[6] Becker S.C.. *Vision-assisted modelling from model based video representations*. PhD thesis, Massachusetts Institute of Technology, February 1997.

[7] Becker, S. and Bove, V. M. J.. Semi-automatic 3D model extraction from uncalibrated 2D camera views. In *SPIE Visual Data Exploration and Analysis II*, volume 2410, pages 447-461, February 1995.

[8] Bertalmio, M., Sapiro, G., Caselles, V. and Ballester, C., Image Inpainting, in Proc. SIGGRAPH 2000, 2000

[9] Bhandarkar, S.M. , Zhang, Y. and Potter, W.D. An edge detection technique using genetic algorithm-based optimization. Pattern Recognition, 27, pp1159-1180, 1994.

[10] Canny F. J.. A computational approach to edge detection. IEEE Trans PAMI, 8(6), pp679-698, 1986

[11] Caponetti, L., Abbattista, N. and Carapella, G., A genetic approach to edge detection. In Proc. IEEE Int. Conf. Image Processing, volume I, pp318-322, 1994.

[12] Chalmers, A, Stoddart, S, Belcher, M & Day, M (1996), An Interactive Photo-Realistic Visualisation System for Archaeological Sites, http://www.cs.bris.ac.uk/Research/Digitalmedia/insite.html as viewed on 19 June 2001

[13] Chalmers, A., Stoddart, S., Tidmus, J. and Miles, R. INSITE: an Interactive Visualisation System for archaeological sites, Proceedings of the 22nd Computer Applications in Archaeology conference held at Glasgow University, Glasgow, 1994

[14] Chen and L. Williams. View Interpolation for Image Synthesis. SIGGRAPH'93, 1993.

[15] Chen, E., Quicktime VR - An image-based approach to virtual environment navigation. In Proc. SIGGRAPH 95, pages 29-38, 1995.

[16] Coorg, S. and Teller, S., Extracting textured vertical facades from controlled close-range imagery. In *Computer Vision and Pattern Recognition*, pages 625-632, Fort Collins, Colorado, 1999.

[17] Counsell, J. and Brkljac, N. A Role for VRML as a Multimedia Backbone in Interpreting Cultural Heritage Sites. Proceedings of the 1999 International Conference on Information Visualization, 1999

[18] Daap, http://www.daap.uc.edu/CERHAS/benb/lascaux.htm college of design, architecture, art and planning at the University of Cincinnati

[19] Dattatreya, G.R. and Kanal, L.N., Detection and smoothing of edge contours in images by one-dimensional Kalman techniques. IEEE Trans. Systems, Man and Cybernetics, 20, pp159-165, 1990.

[20] Debevec, P.E., Taylor, C.J. and Malik, J., Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. SIGGRAPH 96, p149-165, 1996

[21] Deriche, R. ''Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector," IJCV, Vol. 1, p.167-187, 1987.

[22] Efros and T. Leung, "Texture synthesis by non-parametric sampling," *Proc. IEEE International Conference Computer Vision*, 1033-1038, Corfu, Greece, September 1999.

[23] Eiteljorg, H, "The Compelling Computer Image - a double-edged sword" Internet Archaeology 8, 2000 (http://intarch.ac.uk/journal/issue8/eiteljorg_toc.html)

[24] Emele, M. Archaeological Simulation between Linear Media and a Virtual Museum (http://www.uni-kiel.de/cinarchea/sympose/sympo981.htm) 1998 as viewed on 15/03/2001

[25] EOS Corp, PhotoModeler, http://www.photomodeler.com/ as viewed on 19 June 2001.

[26] Faugeras, O., D., Luong, Q.-T., Maybank, S.,J., Camera Self-Calibration: Theory and Experiments, ECCV'92, Lecture notes in Computer Science, Vol 588. Ed. G. Sandini, Springer-Verlag, 1992, pp. 321-334.

[27] Faugeras, O., Three-Dimensional Computer Vision: A Geometric Viewpoint. M.I.T. Press, 1993

[28] Fitzgibbon, A. W. and Zisserman, A., Automatic 3D model acquisition and generation of new images from video sequences. In Proceedings of European Signal Processing Conference (EUSIPCO '98), Rhodes, Greece, pages 1261-1269, 1998.

[29] Gortler, J., R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in Proc. SIGGRAPH'96, pp. 43-54, 1996

[30] Graphics Interchange Format (Version 89a), CompuServe, Inc., Columbus, Ohio, 1990

[31] Hartley, R., Euclidian reconstruction from uncalibrated views, In Applications of Invariance in Computer Vision, LNCS 825, Springer-Verlag, 1994

[32] Hartley, R., Self-calibration from multiple views with a rotating camera, Proc. European Conference on Computer Vision, LNCS 800/801. Springer-Verlag, 1994

[33] Heckes, J. & Hornschuch, A. (1997), Digital photogrammetry and image processing for the documentation of monuments and sites, International Archives of Photogrammetry and Remote Sensing, Vol XXXII, Part 5C1B, pp 108-112.

[34] Heeger and J. Bergen. *Pyramid based texture analysis/synthesis*. Computer Graphics, pp. 229-238, SIGGRAPH95, 1995.

[35] Heine, E. (1999), Interdisciplinary Association for Maya Studies, Erwin Heine (ed.), Online, Available http://www.cis.tu-graz.ac.at/IAM 8 February, 2000.

[36] Hirani and T. Totsuka. *Combining Frequency and spatial domain information for fast interactive image noise removal.* Computer Graphics, pp. 269-276, SIGGRAPH 96, 1996.

[37] Hueckel, M.H. ``A Local Visual Operator which Recognizes Edges and Lines," Journal of the Association for Computing Machinery, Vol. 20, No. 4 pp 634-647, 1971.

[38] Ioannidis, C., Potsiou, C. & Badekas, J., The use of contemporary photogrammetric procedures for the recording and documentation of large monuments and their graphic

representation, International Archives of Photogrammetry and Remote Sensing, Vol XXXII, Part 5C1B, pp 131-140. 1997

[39] ISO DIS 10918-1. Digital Compression and Coding of Continuous-tone Still Images (JPEG), CCITT Recommendation T.81.

[40] ISO/IEC 14772-1:1997, the Virtual Reality Modelling Language (VRML)

[41] Kanade, T., "Recovery of the three-dimensional shape of an object from a single view", J. of Artificial Intelligence, vol. 17, pp. 409-460, 1981.

[42] Kass, M., A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," International Journal of Computer Vision 1 , pp. 321-331. 1988

[43] Kim, H.J., Kim, W., and Li, C.C., A performance study of two wavelet-based edge detectors. In Proc. 11th Int. Conf. Pattern Recognition, volume C, pages 302--306, 1992.

[44] Kumar, R., Anandan, P. and Hanna, K., ``Shape Recovery from Multiple Views: A Parallax Based Approach," Arpa IUW, Monterey, CA, pp. 947--955. 1994

[45] Lastra, A., Introduction to Virtual Reality – Hardware, http://www.siggraph.org/conferences/siggraph96/core/conference/courses/14.htm as viewed on 19 June 2001., 1996

[46] Longuet-Higgins, H., C., A computer algorithm for reconstructing a scene from two projections, Nature, vol. 293, Sept 1982.

[47] Marr, D.C. and Hildreth, E., Theory of edge detection, Proc. Royal Society, London vol B 207, pp 187-217, 1980

[48] Maybank S and Faugeras O, A theory of self-calibration of a moving camera, Int Journal of Computer Vision, Vol 8, 123-151, 1992

[49] McMillan, L., and Bishop, G. Plenoptic modeling: An image-based rendering system. In *Computer Graphics, Annual Conference Series, 1995*, pp. 39–46. 1995

[50] Mitchell, W. The Mond Photographs of the Tomb of Menna, http://www.doc.mmu.ac.uk/RESEARCH/virtual-museum/Menna/mond.html as viewed on 12 June 2001, 1998

[51] Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. In ARPA Image Understanding Workshop, Monterrey, pages 1177-1188, 1994.

[52] Mortensen, E.N. and Barrett, W.A., Intelligent scissors for image composition. In SIGGRAPH'95, 1995

[53] Oliveira, M.M., Bishop, G. and McAllister, D., Relief Texture Mapping, from SIGGRAPH 2000, 2000

[54] Perona, P. and J. Malik. Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12:629--639, 1990.

[55] Pollefeys M., Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences. PhD thesis, Katholieke Universiteit Leuven, Belgium, 1999. (as described at http://www.esat.kuleuven.ac.be/~pollefey/reconstruction/node10.html)

[56] Rattenbury, W. The Hadrianic Baths at Leptis Magna - A Reconstruction of the Buildings, http://archpropplan.auckland.ac.nz/virtualtour/hadrians_bath as viewed on 14 March 2001, 1991

[57] Reilly, P., Three-dimensional modelling and primary archaeological data, Archaeology and the Information Age: a Global Perspective, Routledge, London, pp 147-173. 1992

[58] Reilly, P., Data Visualisation in Archaeology, IBM Systems Journal, vol. 28, no. 4, pp. 569–579, 1989

[59] Roberts, L.G. ``Machine Perception of Three-Dimensional Solids," Optical and Electro-optical Information Processing, Tippett, et al. editors, MIT Press, p.159-197, 1965

[60] Rothwell, C., Mundy, J., Hoffman, W. and Nguyen, V.-D. "Driving Vision by Topology" Technical Report No. 2444, INRIA, 1994.

[61] Rzepa, H., Report on: The First International Conference on World-Wide-Web, http://www.ch.ic.ac.uk./talks/www94_report.html) as viewed on 19 June 2001. 1994

[62] SGI Corp., OpenGL Inventor - Summary of The Inventor Mentor, http://www.sgi.com/software/inventor/vrml/VRMLDesign.html as viewed on 19 June 2001

[63] Shum, H.-Y., Han, M. and Szeliski, R.. Interactive construction of 3D models from panoramic mosaic. In *Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998.

[64] Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. 5th IEEE Int'l Conf. on Image Processing, Chicago, IL. Oct 4-7, 1998.

[65] Sims, D., Archaeological models: pretty pictures or research tools?, IEEE Computer Graphics and Applications, Potel, M. (ed.), January/February 1997, pp. 13-1, 1997.

[66] Sobel , I. Neighbourhood coding of binary images for fast contour following and general binary array processing. Comp. Graphics and Image Processing, vol 8, pp 127-135, 1978.

[67] Staib L.H. and J.S. Duncan. Boundary finding with parametrically deformable models. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 14, pp 161-175, 1992.

[68] Stamos, Ioannis and Allen, Peter K., 3D Model Construction Using Range and Image Data (Submitted to CVPR 2000) Department of Computer Science, Columbia University, 2000

[69] Superscape, - Superscape/Intel Stonehenge visualisation, http://www.connectedpc.com/cpc/ecs/stonehenge 1997

[70] Sutherland I, The Ultimate Display, Proceedings of IFIPS Congress 1965, New York, Vol. 2, pp. 506-508. 1965

[71] Tan, H.L., Gelfand, S.B. and Delp, E.J., A cost minimization approach to edge detection using simulated annealing. IEEE Trans. Pattern Analysis and Machine Intelligence, 14, pp 3-18, 1992.

[72] Terras, M., "A Virtual Tomb for Kelvingrove: Virtual Reality, Archaeology and Education" - Internet Archaeology 7, 1999 (http://intarch.ac.uk/journal/issue7/terras_toc.html) as viewed on 10 Nov 2000, 1999.

[73] Three D Construction Corp., 3DConstruction Software, http://www.3dconstruction.com/gallery.htm as viewed on 14 Aug 2000

[74] Tomasi, C. and Kanade, T., ``Shape and Motion from Image Streams under Orthography: a Factorization Method," International Journal of Computer Vision, 9, 2, pp. 137-154, 1992.

[75] Tsai R., "An Efficient and Accurate Camera Calibration Technique for 3-D Machine Vision", in Proc. IEEE Conf. Comput. Vision and Pattern Recognition, pp. 364-374, 1988.

[76] Williams D. and Shah M., "A fast algorithm for active contour and curvature estimation," CVGIP: Image Understanding, vol. 55, pp. 14-26, January 1992

[77] Worring, M., Smeulders, A.W.M., Staib, L.H. and Duncan, J.S., Parametrized feasible boundaries in gradient vector fields. Information Processing in Medical Imaging, pp 48--61, 1993

[78] Xiong, Y. and Turkowski, K. Creating image-based VR using a self-calibrating fisheye lens. In Conference on Computer Vision and Pattern Recognition, pages 237--243, San Juan, Puerto Rico, June 1997

Appendix A Planar triangle formula derivation

This appendix provides the derivation of the transformation for calculating the three-dimensional position for each edge pixel. The transformation functions by calculating the position of a pixel relative to a new coordinate system based on axes corresponding to two edges of the underlying image triangle (the *(a,b)* coordinate system). The location in three-dimensions is then calculated using the location of the pixel in *(a,b)* coordinates but using the corresponding edges of the underlying three-dimensional triangle as the *a* and *b* axes.

First calculate the position of the edge pixel in coordinate system *(a, b)* defined by two of the edges of the triangle on the photograph as shown in Figure 2.

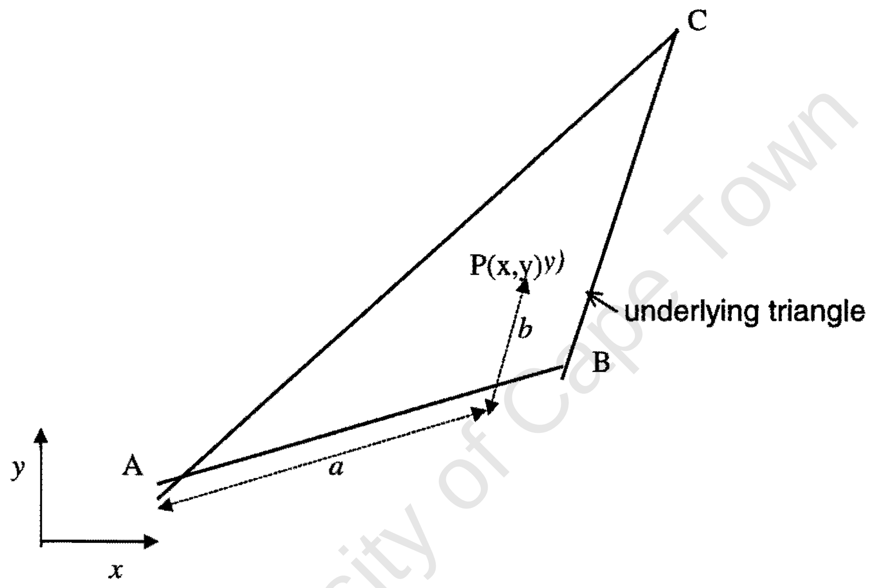


Figure 67 showing the position of a boundary pixel P*(x,y)* in terms of the new coordinate system *(a,b)* based on the edges of a triangle ABC. The origin of the (a.b) coordinate system lies at point A and the directions of the two axes a and b lie in parallel with the edges AB and BC respectively of the underlying triangle

The pixel at x, y lies at the following position:

$$P(x,y) = A + aV_{AB} + bV_{BC} \tag{1}$$

where
V_{AB} is the vector from point A to B.
V_{BC} is the vector from point A to B.

If we express A, V_{AB} and V_{BC} in terms of their *(x,y)* coordinates:

$A = (x_A, y_A)$,

$V_{AB} = (x_B - x_A, y_B - y_A)$ and

$V_{BC} = (x_C - x_B, y_C - y_B)$

Substituting these into (1) yields

$$P(x,y) = (x_A, y_a) + a(x_B - x_A, y_B - y_A) + b(x_C - x_B, y_C - y_B)$$

or with x and y coordinates separated:

$$x = x_A + a(x_B - x_A) + b(x_C - x_B)$$

$$y = y_A + a(y_B - y_A) + b(y_C - y_B)$$

Isolating b in both of these equations and equating the result gives

$$b = \frac{((x - x_A) - a(x_B - x_A))}{(x_C - x_B)} \tag{2}$$

$$b = \frac{((y - y_A) - a(y_B - y_A))}{(y_C - y_B)}$$

Equating these and isolating *a* gives

$$a = \frac{(x - x_A)/(x_C - x_B) - (y - y_A)/(y_C - y_B)}{(x_B - x_A)/(x_C - x_B) - (y_B - y_A)/(y_C - y_B)} \tag{3}$$

Substituting his into (2) gives

$$b = \frac{(x - x_A)}{(x_C - x_B)} - \frac{(x_B - x_A)((x_B - x_A)/(x_C - x_B) - (y_B - y_A)/(y_C - y_B))}{(x_C - x_B)((x - x_A)/(x_C - x_B) - (y - y_A)/(y_C - y_B))} \tag{4}$$

Appendix B Corner location equation derivation

In order to calculate the location of the intersection of a pixel ray (a three-dimensional vector corresponding to an image pixel which is calculated using camera calibration parameters) and a smooth surface defined by a polynomial, the following must be done:
calculate the pixel ray origin and direction
calculate the surface normal and parallel vectors for that edge (these are used as the axes for the surface displacement due to the polynomial)
calculate the surface displacement due to the polynomial near that edge
calculation of the intersection of the polynomial surface by each pixel ray

Furthermore the equations must be rendered in a form amenable to efficient calculation, for example, constant and variable terms need to be separated out.

1. Calculating the pixel ray origin and direction

The first step is to calculate the pixel ray origin for the image and then the pixel ray direction vector for each pixel.

The general calibration matrix in homogenous coordinates for a pinhole camera (see Equationf 3 on p19) is as follows:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where

the object is at (X, Y, Z) in the three-dimensional scene,
s is a scale factor,
the values q_{mn} comprise the calibration matrix $\tilde{Q}$
and the image of the object appears at pixel (u, v).

We reduce the matrix $\tilde{Q}$ in (1) to one which is square as follows:

$$\begin{bmatrix} su - q_{14} \\ sv - q_{24} \\ s-1 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2}$$

or

$$\begin{bmatrix} su - q_{14} \\ sv - q_{24} \\ s-1 \end{bmatrix} = \tilde{Q}' \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

Inverting $\tilde{Q}'$ to find $\tilde{P}$ where

$$\tilde{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

changes (3) to the form:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} su - q_{14} \\ sv - q_{24} \\ s - 1 \end{bmatrix} \tag{4}$$

The origin of all pixel rays (the PRO) in an image is the point where the value of $s = 0$ which from (4) will lie at:

$$PRO = \left(-\left(p_{11}q_{14} + p_{12}q_{24} + p_{13}\right), -\left(p_{21}q_{14} + p_{22}q_{24} + p_{23}\right), -\left(p_{31}q_{14} + p_{32}q_{24} + p_{33}\right)\right) \tag{5}$$

and the line along which the feature which generates the pixel in the image must lie (the PRV) is:

$$PRV = \left(p_{11}u + p_{12_v} + p_{13}, p_{21}u + p_{22_v} + p_{23}, p_{31}u + p_{32_v} + p_{33}\right) \tag{6}$$

The pixel ray is defined as

$$pixelray = PRO + \omega\, PRV \tag{7}$$

where ω is the distance along the pixel ray.

2. The surface normal and parallel vector along an edge

The "smooth" face of each element of surface is defined by a polynomial displaced by a distance normal to the quadrilateral defined by the four corners of the surface element. The quadrilateral made up by joining these points is called the underlying quadrilateral.

The vector normal to the underlying quadrilateral at any point along an edge (see Figure 68) can be calculated via a simple linear interpolation from the normals at the start and end of that edge. Similarly the vectors parallel to the underlying quadrilateral can be calculated via linear interpolation.

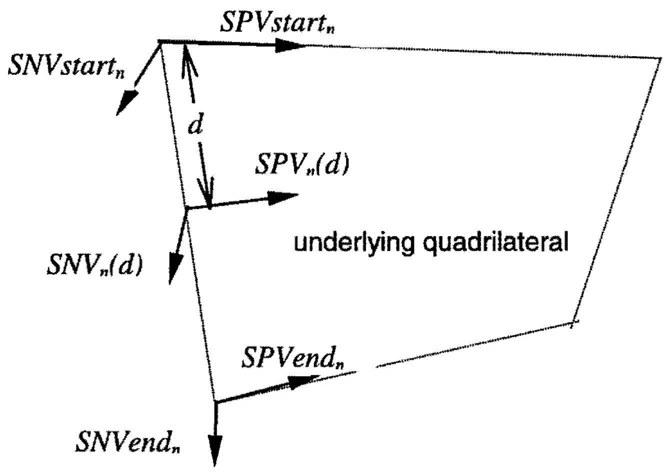


Figure 68 showing the surface normal (SNV) and surface parallel vectors (SPV) for an edge of the underlying quadrilateral

The value of the surface normal vector (*SNV*) at normalised distance *d* down edge *n* is:

$$SNV_n(d) = (1-d)SNVstart_n + d\,SNVend_n$$

or

$$SNV_n(d) = Csn_n^0 + d\,Csn_n^1 \tag{8}$$

where

Csn_n^0 is $SNVstart_n$ and

$Csn_n^1 = SNVend_n - SNVstart_n$

Similarly in the case of the surface parallel vector (*SPV*) at normalised distance *d* down edge *n*:

$$SPV_n(d) = Cpv_n^0 + d\,Cpv_n^1 \tag{9}$$

where

Cpv_n^0 is $SPVstart_n$ and

$Cpv_n^1 = SPVend_n - SPVstart_n$

3. Location of the polynomial surface near an edge

The surface defined by the polynomial is located at (see Figure 69):

Surface = position of edge start
+ distance down edge
+ displacement normal to underlying quadrilateral due to polynomial
+ displacement parallel to underlying quadrilateral (10)

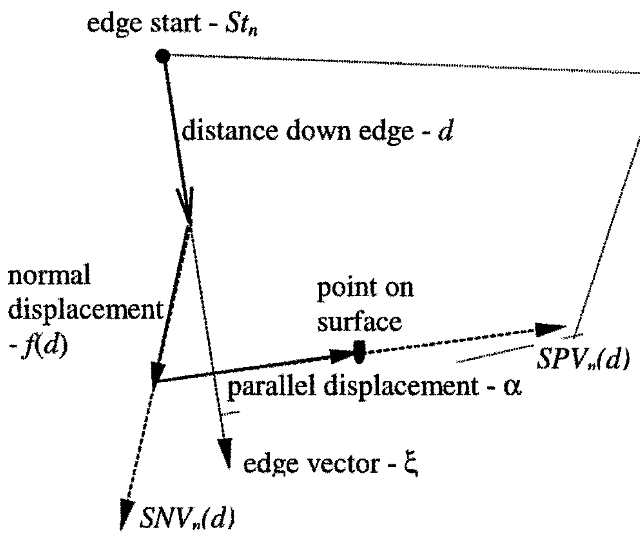


Figure 69 showing the components of displacement from the start of an edge to the position of an arbitrary point on the polynomial surface.

The components of (10) can be expressed as:ξ

$$Surface = St_n + d\,\xi_n + f(d)\,SNV_n(d) + \alpha\,SPV_n(d) \tag{11}$$

where

St_n is the position of the start of edge *n,*
d is the distance along edge *n*,
ξ_n is the edge vector running from start to end of edge *n*,
$f(d)$ is the polynomial defining the smooth surface,
α is the displacement parallel to the underlying quadrilateral,
$SNV_n(d)$ and $SPV_n(d)$ are the surface normal and parallel vectors as calculated in equations (8) and (9) respectively.

4. Intersection of pixel ray vector with smooth surface

Intersection of the pixel ray with the surface occurs at the point where equations (11) and (7) are equal, that is:

$$PRO_n + \omega\,PRV_n = St_n + d\,\xi_n + f(d)\,SNV_n(d) + \alpha\,SPV_n(d) \tag{12}$$

The above equation has (for a defined polynomial) three unknowns, namely ω the distance along the pixel ray where it intersects the surface, d the distance down the edge of said intersection and α the displacement parallel to the underlying quadrilateral of the intersection). It can thus be solved for a point in three dimensions as follows:

Equation (12) expressed as an equation in each dimension is:

$$PRO_n.x + \omega\,PRV_n.x = St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x + \alpha\,SPV_n(d).x \tag{13}$$

$$PRO_n.y + \omega\, PRV_n.y = St_n.y + d\,\xi_n.y + f(d)\,SNV_n(d).y + \alpha\, SPV_n(d).y \quad (14)$$

and

$$PRO_n.z + \omega\, PRV_n.z = St_n.z + d\,\xi_n.z + f(d)\,SNV_n(d).z + \alpha\, SPV_n(d).z \quad (15)$$

Manipulating (13) and (14) respectively to isolate ω results in:

$$\omega = \frac{St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x + \alpha\, SPV_n(d).x - PRO_n.x}{PRV_n.x} \quad (16)$$

and

$$\omega = \frac{St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x + \alpha\, SPV_n(d).x - PRO_n.x}{PRV_n.x} \quad (17)$$

Combining (16)and (17) and isolating α results in:

$$\alpha = \frac{\dfrac{St_n.y + d\,\xi_n.y + f(d)\,SNV_n(d).y - PRO_n.y}{PRV_n.y} - \dfrac{St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x - PRO_n.x}{PRV_n.x}}{\left(\dfrac{SPV_n(d).x}{PRV_n.x} - \dfrac{SPV_n(d).y}{PRV_n.y}\right)}$$

(18)

Substituting for α from (18) into (16) yields

$$\omega = \frac{St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x + \left(\dfrac{\dfrac{St_n.y + d\,\xi_n.y + f(d)\,SNV_n(d).y - PRO_n.y}{PRV_n.y} - \dfrac{St_n.x + d\,\xi_n.x + f(d)\,SNV_n(d).x - PRO_n.x}{PRV_n.x}}{\left(\dfrac{SPV_n(d).x}{PRV_n.x} - \dfrac{SPV_n(d).y}{PRV_n.y}\right)} \right) SPV_n(d).x - PRO_n}{PRV_n.x}$$

(19)

Substituting for α from (18) and ω from (19) into (15) yields

$$PRV_n.z\left(\frac{St_n.x+d\,\xi_n.x+f(d)SNV_n(d).x+\left(\frac{\frac{St_n.y+d\,\xi_n.y+f(d)SNV_n(d).y-PRO_n.y}{PRV_n.y}-\frac{St_n.x+d\,\xi_n.x+f(d)SNV_n(d).x-PRO_n.x}{PRV_n.x}}{(\frac{SPV_n(d).x}{PRV_n.x}-\frac{SPV_n(d).y}{PRV_n.y})}\right)SPV_n(d).x-PRO_n.x}{PRV_n.x}\right)$$

$$+PRO_n.z=St_n.z+d\,\xi_n.z+f(d)SNV_n(d).z$$

$$+SPV_n(d).z\left(\frac{\frac{St_n.y+d\,\xi_n.y+f(d)SNV_n(d).y-PRO_n.y}{PRV_n.y}-\frac{St_n.x+d\,\xi_n.x+f(d)SNV_n(d).x-PRO_n.x}{PRV_n.x}}{(\frac{SPV_n(d).x}{PRV_n.x}-\frac{SPV_n(d).y}{PRV_n.y})}\right) \tag{20}$$

Multiply both sides by $(\frac{SPV_n(d).x}{PRV_n.x}-\frac{SPV_n(d).y}{PRV_n.y})$ and equate to zero

$$(\frac{SPV_n(d).x}{PRV_n.x}-\frac{SPV_n(d).y}{PRV_n.y})(St_n.z+d\,\xi_n.z+f(d)SNV_n(d).z-PRO_n.z$$

$$-\frac{PRV_n.zSt_n.x}{PRV_n.x}-\frac{PRV_n.zd\,\xi_n.x}{PRV_n.x}-\frac{PRV_n.zf(d)SNV_n(d).x}{PRV_n.x}+\frac{PRV_n.zPRO_n.x}{PRV_n.x})$$

$$+SPV_n.z\left(\frac{St_n.y+d\,\xi_n.y+f(d)SNV_n(d).y-PRO_n.y}{PRV_n.y}-\frac{St_n.x+d\,\xi_n.x+f(d)SNV_n(d).x-PRO_n.x}{PRV_n.x}\right.$$

$$-PRV_n.z\left(\frac{SPV_n.x}{PRV_n.x}\right)\left(\frac{St_n.y+d\xi_n.y+f(d)SNV_n(d).y-PRO_n.y}{PRV_n.y}-\frac{St_n.x+d\xi_n.x+f(d)SNV_n(d).x-PRO_n.x}{PRV_n.x}\right)$$

$$=0$$

Multiply by $(PRV_n.x)^2 PRV_n.y$ to remove all components of PRV_n from denominators in order to prevent division by zero in cases where pixel rays have no *x* or *y* component.

$$\begin{aligned}
&\left(PRV_n.y\,SPV_n(d).x - PRV_n.x\,SPV_n(d).y\right)\\[6pt]
&\left(PRV_n.x\,St_n.z + PRV_n.x\,d\,\xi_n.z - PRV_n.x\,PRO_n.z + PRV_n.x\,f(d)\,SNV_n(d).z\right)\\
&-\left(PRV_n.y\,SPV_n(d).x - PRV_n.x\,SPV_n(d).y\right)\\[6pt]
&\left(PRV_n.z\,St_n.x + PRV_n.z\,d\,\xi_n.x - PRV_n.z\,PRO_n.x + PRV_n.z\,f(d)\,SNV_n(d).x\right)\\
&+\left(PRV_n.x\,SPV_n(d).z - PRV_n.z\,SPV_n(d).x\right)\\[6pt]
&\left(PRV_n.x\,St_n.y + PRV_n.x\,d\,\xi_n.y - PRV_n.x\,PRO_n.y + PRV_n.x\,f(d)\,SNV_n(d).y\right)\\
&+\left(PRV_n.z\,SPV_n(d).x - PRV_n.x\,SPV_n(d).z\right)\\[6pt]
&\left(PRV_n.y\,St_n.x + PRV_n.y\,d\,\xi_n.x - PRV_n.y\,PRO_n.x + PRV_n.y\,f(d)\,SNV_n(d).x\right)\\
&=0
\end{aligned}$$

Substituting $SNV_n(d)$ and $SPV_n(d)$ using (8) and (9) respectively yields

$$\begin{aligned}
&\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\,(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.x\,St_n.z + PRV_n.x\,d\,\xi_n.z - PRV_n.x\,PRO_n.z + PRV_n.x\,f(d)\,(Csn_n^0.z + d\,Csn_n^1.z)\right)\\
&-\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\,(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.z\,St_n.x + PRV_n.z\,d\,\xi_n.x - PRV_n.z\,PRO_n.x + PRV_n.z\,f(d)\,(Csn_n^0.x + d\,Csn_n^1.x)\right)\\
&+\left(PRV_n.x\,(Cpv_n^0.z + d\,Cpv_n^1.z) - PRV_n.z\,(Cpv_n^0.x + d\,Cpv_n^1.x)\right)\\[6pt]
&\left(PRV_n.x\,St_n.y + PRV_n.x\,d\,\xi_n.y - PRV_n.x\,PRO_n.y + PRV_n.x\,f(d)\,(Csn_n^0.y + d\,Csn_n^1.y)\right)\\
&+\left(PRV_n.z\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\,(Cpv_n^0.z + d\,Cpv_n^1.z)\right)\\[6pt]
&\left(PRV_n.y\,St_n.x + PRV_n.y\,d\,\xi_n.x - PRV_n.y\,PRO_n.x + PRV_n.y\,f(d)\,(Csn_n^0.x + d\,Csn_n^1.x).\right)\\
&=0
\end{aligned}$$

and separating out the terms which are dependent on the surface polynomial $f(d)$ yields

$$
\begin{aligned}
&\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.x\,St_n.z + PRV_n.x\,d\,\xi_n.z - PRV_n.x\,PRO_n.z\right)\\
&-\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.z\,St_n.x + PRV_n.z\,d\,\xi_n.x - PRV_n.z\,PRO_n.x\right)\\
&+\left(PRV_n.x\;(Cpv_n^0.z + d\,Cpv_n^1.z) - PRV_n.z\;(Cpv_n^0.x + d\,Cpv_n^1.x)\right)\\
&\qquad\left(PRV_n.x\,St_n.y + PRV_n.x\,d\,\xi_n.y - PRV_n.x\,PRO_n.y\right)\\
&+\left(PRV_n.z\;(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.z + d\,Cpv_n^1.z)\right)\\
&\qquad\left(PRV_n.y\,St_n.x + PRV_n.y\,d\,\xi_n.x - PRV_n.y\,PRO_n.x\right)\\
&+(\\
&\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.x\,(Csn_n^0.z + d\,Csn_n^1.z)\right)\\
&-\left(PRV_n.y\,(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.y + d\,Cpv_n^1.y)\right)\\
&\qquad\left(PRV_n.z\,(Csn_n^0.x + d\,Csn_n^1.x)\right)\\
&+\left(PRV_n.x\;(Cpv_n^0.z + d\,Cpv_n^1.z) - PRV_n.z\;(Cpv_n^0.x + d\,Cpv_n^1.x)\right)\\
&\qquad\left(PRV_n.x\;(Csn_n^0.y + d\,Csn_n^1.y)\right)\\
&+\left(PRV_n.z\;(Cpv_n^0.x + d\,Cpv_n^1.x) - PRV_n.x\;(Cpv_n^0.z + d\,Cpv_n^1.z)\right)\\
&\qquad\left(PRV_n.y\;(Csn_n^0.x + d\,Csn_n^1.x).\right)\\
&\hspace{30em})f(d)\\
&= 0
\end{aligned}
$$

Finally we combine terms to provide the form of this equation amenable to codification and efficient calculation (see §4.9.1).

$$P_{pixel}^0(d) + P_{surface}(d)P_{pixel}^1(d) = 0 \tag{21}$$

where

$$P_{surface}(d) = f\left(d\right)$$

$$
\begin{aligned}
&P^{0}_{pixel}(d) = \\
&\left(PRV_n.y\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.y + d\,Cpv^1_n.y)\right) \\
&\qquad\qquad\qquad \left(PRV_n.x\,St_n.z + PRV_n.x\,d\,\xi_n.z - PRV_n.x\,PRO_n.z\right) \\
&-\left(PRV_n.y\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.y + d\,Cpv^1_n.y)\right) \\
&\qquad\qquad\qquad \left(PRV_n.z\,St_n.x + PRV_n.z\,d\,\xi_n.x - PRV_n.z\,PRO_n.x\right) \\
&+\left(PRV_n.x\,\,(Cpv^0_n.z + d\,Cpv^1_n.z) - PRV_n.z\,\,(Cpv^0_n.x + d\,Cpv^1_n.x)\right) \\
&\qquad\qquad\qquad \left(PRV_n.x\,St_n.y + PRV_n.x\,d\,\xi_n.y - PRV_n.x\,PRO_n.y\right) \\
&+\left(PRV_n.z\,\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.z + d\,Cpv^1_n.z)\right) \\
&\qquad\qquad\qquad \left(PRV_n.y\,St_n.x + PRV_n.y\,d\,\xi_n.x - PRV_n.y\,PRO_n.x\right)
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
&P^{1}_{pixel}(d) = \\
&PRV_n.x\,\left(PRV_n.y\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.y + d\,Cpv^1_n.y)\right)(Csn^0_n.z + d\,Csn^1_n.z) \\
&-PRV_n.z\left(PRV_n.y\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.y + d\,Cpv^1_n.y)\right)(Csn^0_n.x + d\,Csn^1_n.x) \\
&+PRV_n.x\,\,\left(PRV_n.x\,\,(Cpv^0_n.z + d\,Cpv^1_n.z) - PRV_n.z\,\,(Cpv^0_n.x + d\,Cpv^1_n.x)\right)(Csn^0_n.y + d\,Csn^1_n.y) \\
&+PRV_n.y\,\,\left(PRV_n.z\,\,(Cpv^0_n.x + d\,Cpv^1_n.x) - PRV_n.x\,\,(Cpv^0_n.z + d\,Cpv^1_n.z)\right)(Csn^0_n.x + d\,Csn^1_n.x)
\end{aligned}
\tag{23}
$$

If we now factorise out the *PRV* components

$$
P^{0}_{pixel}(d) = PRV_x^2 P^{0}_{edge}(d)_{x^2} + PRV_x PRV_y P^{0}_{edge}(d)_{xy} + PRV_x PRV_z P^{0}_{edge}(d)_{xz} \tag{24}
$$

where

$$
\begin{aligned}
&P^{0}_{edge}(d)_{x^2} = P_{parallel}(d)_{yz^2} - P_{parallel}(d)_{y^2z}\,, \\
&\quad P_{parallel}(d)_{xy^2} = (Csn^0_z + Csn^1_z d)(Cpv^0_y + Cpv^1_y d)\,, \\
&\quad P_{parallel}(d)_{y^2z} = (Cpv^0_y + Cpv^1_y d)(\,St_n.z + \xi_n.z d - PRO_n.z)\,,
\end{aligned}
$$

$$
\begin{aligned}
&P^{0}_{edge}(d)_{xy} = P_{parallel}(d)_{z^2} - P_{parallel}(d)_{y^2}\,, \\
&\quad P_{parallel}(d)_{z^2} = (Cpv^0_z + Cpv^1_z d)(St_n.x + \xi_n.x d - PRO_n.x)\,, \\
&\quad P_{parallel}(d)_{y^2} = (Cpv^0_x + Cpv^1_x d)(St_n.z + \xi_n.z d - PRO_n.z)
\end{aligned}
$$

$$
\begin{aligned}
&P^{0}_{edge}(d)_{xz} = P_{parallel}(d)_{z} - P_{parallel}(d)_{y}\,, \\
&\quad P_{parallel}(d)_{z} = (Cpv^0_y + Cpv^1_y d)(St_n.x + \xi_n.x d - PRO_n.x)\,, \\
&\quad P_{parallel}(d)_{y} = (Cpv^0_x + Cpv^1_x d)(St_n.y + \xi_n.y d - PRO_n.y)\,,
\end{aligned}
$$

and

$$P_{pixel}^{1}(d) = PRV_x^2 P_{edge}^{1}(d)_{x^2} + PRV_x PRV_y P_{edge}^{1}(d)_{xy} + PRV_x PRV_z P_{edge}^{1}(d)_{xz} \tag{25}$$

$$P_{edge}^{1}(d)_{x^2} = P_{parallel}(d)_{xyz^2} - P_{parallel}(d)_{xy^2z},$$
$$P_{parallel}(d)_{xyz^2} = (Csn_y^0 + Csn_y^1 d)(Cpv_z^0 + Cpv_z^1 d)$$
$$P_{parallel}(d)_{xy^2z} = (Csn_z + Csn_z^1 d)(Cpv_y^0 + Cpv_y^1 d)$$

$$P_{edge}^{1}(d)_{xy} = P_{parallel}(d)_{xz^2} - P_{parallel}(d)_{xy^2},$$
$$P_{parallel}(d)_{xz^2} = (Csn_x^0 + Csn_x^1 d)(Cpv_z^0 + Cpv_z^1 d),$$
$$P_{parallel}(d)_{xy^2} = (Csn_z^0 + Csn_z^1 d)(Cpv_x^0 + Cpv_x^1 d)$$

$$P_{edge}^{1}(d)_{xz} = P_{parallel}(d)_{xz} - P_{parallel}(d)_{xy},$$
$$P_{parallel}(d)_{xz} = (Csn_x^0 + Csn_x^1 d)(Cpv_y^0 + Cpv_y^1 d),$$
$$P_{parallel}(d)_{xy} = (Csn_y^0 + Csn_y^1 d)(Cpv_x^0 + Cpv_x^1 d)$$

Appendix C Incorrect geometry calculation algorithms

1. Transformation of coordinate system

As is the case for the reverse parallax method described in the previous section, this approach initially appeared to be a suitable way to calculate geometry within occluded regions. It too turned out to be flawed; on this occasion while generating test data set. Similarly it is believed useful to describe this approach.

Once again, the attraction of this approach is that it provides an explicit solution for the location of the edges and it avoids the implicit calibration of the image.

The central concept of this approach is to transform the three-dimensional coordinate system into a two-dimensional one based on the orientation of the edge, the pixel and the underlying quadrilaterals of the adjacent sides. The approach is based on the assumption that the sides have surface characteristics, for example if a side is convex at one end is more likely to be convex at the other than not.

If we imagine a roughly horizontal slice through an adjacent pair of roughly vertical faces (see Figure 70). The slice is made at the same proportional distance d as the pixel is down the edge in two-dimensions (see Figure 71)

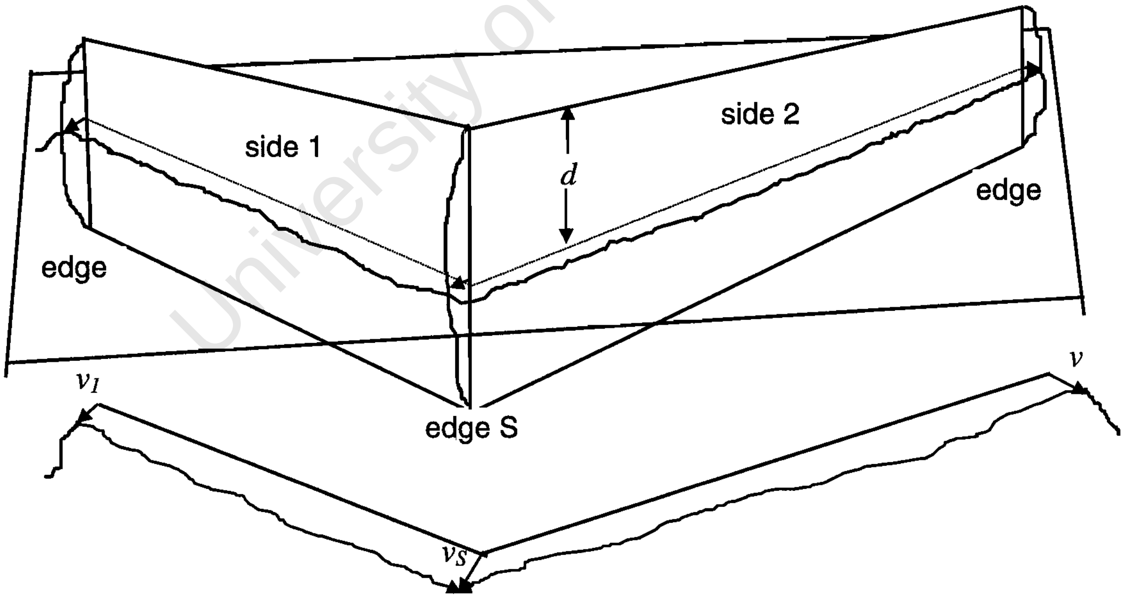


Figure 70 showing a slice through the two sides at distance d from the top. The displacements v_n are the distances (in the plane of the slice) that the actual surfaces are from their underlying quadrilaterals

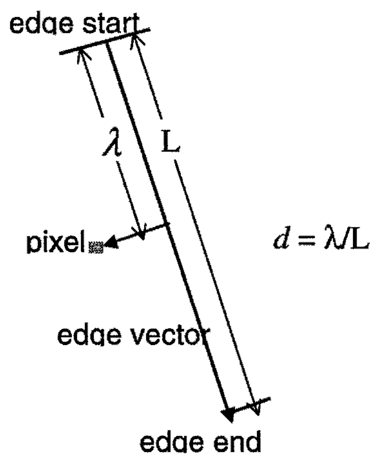


Figure 71 showing the calculation of d (the proportional distance of the slice down the edge) from the positions of the pixel and edge vector in the photograph.

The new coordinate system (u,v) has its origin at the position where the edge cuts the slice and v-axis parallel with the pixel ray (see Figure 72). The u-axis is orthogonal to the v-axis.

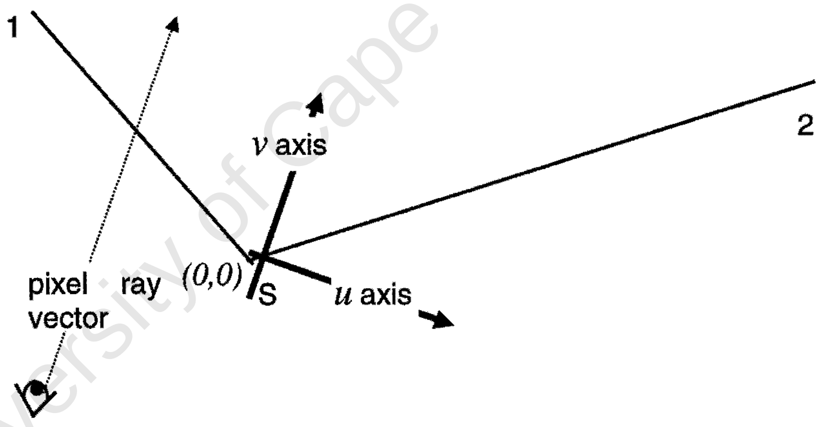


Figure 72 showing the orientation of the new coordinate system (u,v) with origin at the position of the current edge (S) and orientation in accordance with the pixel ray vector.

The pixel ray vector is calculated using triangulation based on the positions of the edge and the two adjacent edges in the photograph and geometry (see Figure 73).

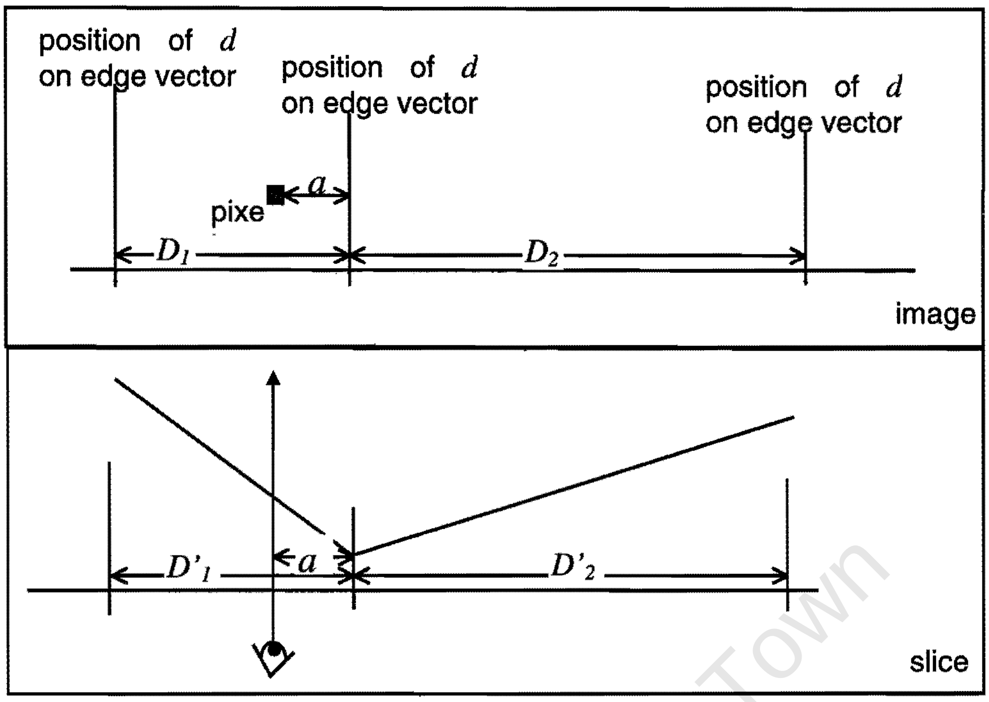

position of *d* on edge vector
position of *d* on edge vector
position of *d* on edge vector
pixe
a
D₁
D₂
image
D'₁
a
D'₂
slice

Figure 73 showing the distances between the current edge (S), the adjacent edges (1 and 2) and the location of the pixel in the image. The orientation of the pixel ray is calculated by rotating the slice until the ratio $\frac{D_1'}{D_2'} = \frac{D_1}{D_2}$, and its position is calculated from the ratio $\frac{a'}{D_1'} = \frac{a}{D_1}$.

It is then a matter of calculating the intersection of the pixel ray with the surfaces adjacent to that edge; and then estimating the actual position as a weighted average of surface length (see Figure 74).

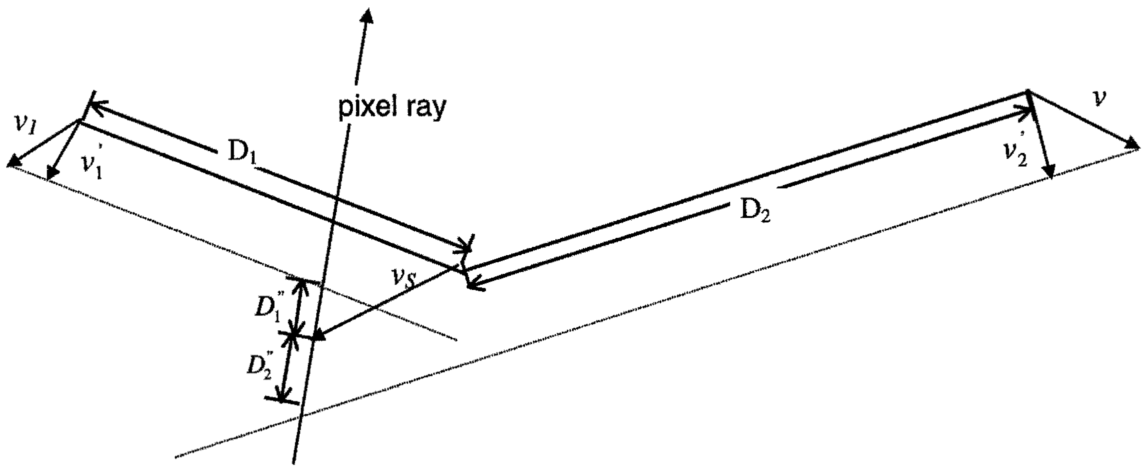


Figure 74 which shows the displacement vector (V_S) for the current corner. This vector is calculated by dividing the distance between the points of intersection (of the pixel ray and the surface offsets) into the same ratio D_1''/D_2'' as the ratio of the lengths of the adjacent sides D_1/D_2. The surface offsets are parallel to the lines joining the current edge to it's adjacent edges but offset by the component of the respective surface displacement vector normal to the side.

It is possible to set up a system of linear simultaneous equations to describe the relationship between all the unknown edge displacements v_i. These can be solved by the inversion of this simple albeit large matrix.

The weakness of this approach is that it does not take into account the fact that *the edge in the scene photographed is seldom in the same plane as the image plane*; the approach assumes that the distance of the two-dimensional slice down the edge (that is d - in Figure 71) is always calculable from the position of the pixel relative to the edge axis in the photograph. Thus any protruberences along the edge are assumed only to have an effect "out of the page" whereas the would actually move the pixel down (in the case of a protuberance in a forward tilting edge - as shown in Figure 75 or up (in the case of a backward-tilting edge).

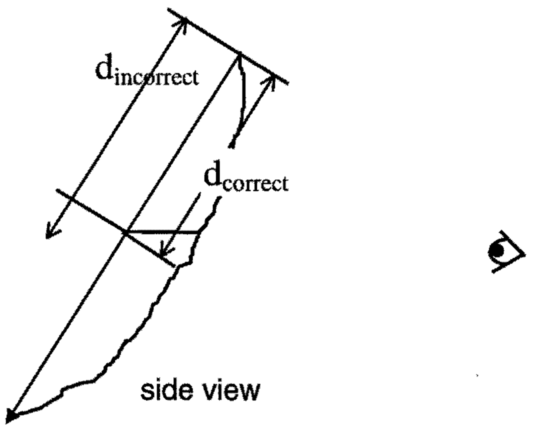


Figure 75 shows the error caused ($d_{incorrect}$ - $d_{correct}$) because the above described algorithm assumes that surfaces will be vertical.

2. Reverse parallax

This approach initially appeared to be an elegant way to calculate geometry within occluded regions. During implementation, however, it turned out to be flawed. It is believed useful to describe this approach, which proved unsuccessful although initially appearing to be reasonable, firstly as a guide to later researchers and secondly as evidence of the thoroughness with which the options for calculating occluded geometry have been addressed.

The attraction of the reverse parallax approach is that it provides an explicit solution for the location of the edges (that is one which does not require any numerical/iterative methods which can be computationally expensive); and it avoids the implicit calibration of the image whilst calculating surfaces.

Briefly this approach develops on the concept of the position of a pixel on the plane defined by a triangle in three-dimensions discussed in §4.3.8 , namely that it is possible to calculate a unique three-dimensional location for a point by assuming it to be in the same plane as a pre-defined triangle and having a known location relative to the triangle vertices in two dimensions. Furthermore, any pixel lying on an edge shared between two quadrilaterals can be calculated to have four separate positions (one position for each of the four planar triangles the edge forms part of - each quadrilateral consists of a pair of triangles as shown in Figure 76).

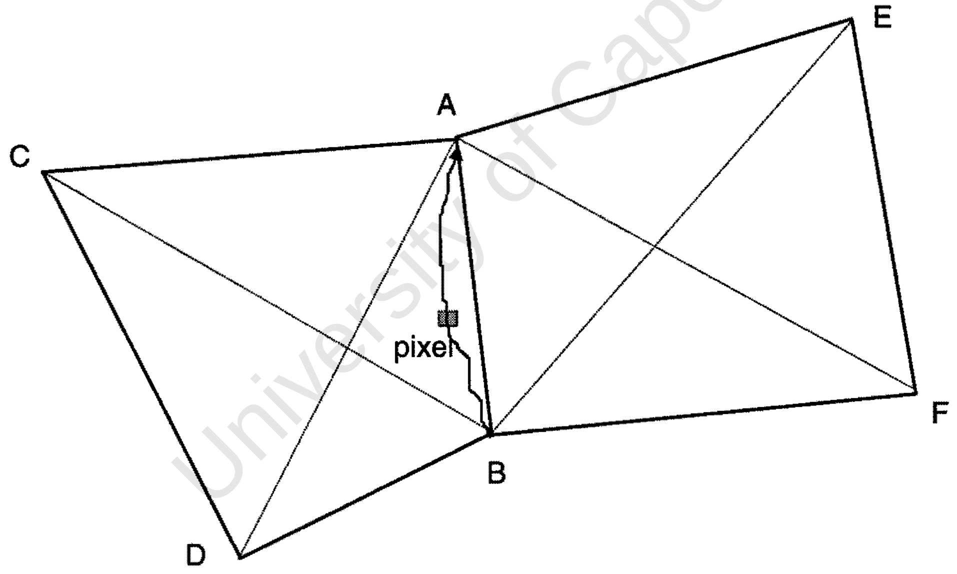


Figure 76 showing a pixel, which is, near the shared edge AB and is thus near the edges of each of the four triangles: ABC, ABD, ABE, and ABF.

Four different positions can be calculated for any pixel based on its relationship to the vertices of each of the triangles.

Considering each pair of points separately, it would thus seem reasonable that the true position of the pixel in space must be somewhere on the line which passes through them (Figure 77). Furthermore, as we have two pairs of points we can calculate two lines (and the correct point in three-dimensions must lie on both of the lines).

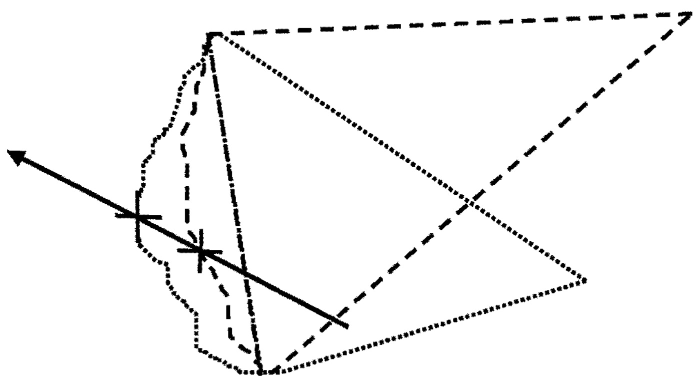


Figure 77 showing a pair of triangles which are not co-planar, each having a different locus of projected edge pixels. The projection of a single pixel according to its position relative to the vertices of each triangle is shown by the crosses. The position of the feature which appears as this pixel is expected to lie on the line through the pair of crosses.

Thus it seems possible to calculate the true position of the point from the intersection of the lines as shown in Figure 78.

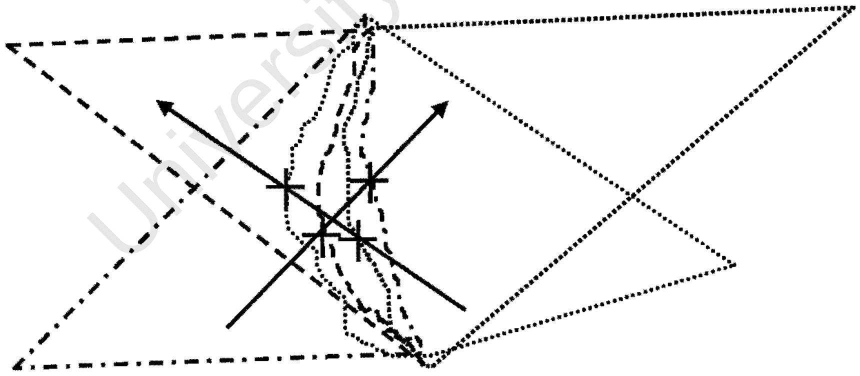


Figure 78 showing the pair of lines (the vectors) due to the same pixel being transformed into a three-dimensional position using the vertices of four triangles. The intersection of these two lines was expected to be the true position of the feature which appears as this pixel

The problem with this approach was recognised while trying to produce a formulation for the intersection of the two lines, particularly in the situation where one of the quadrilaterals is planar. This results in the two triangles being co-planar, which in turn results in the two pixel positions generated from them being at a single point.

Calculating a unique intersection of a line and *any line passing through a point* is not possible. This problem is exacerbated when both the adjacent quadrilaterals are planar (the unique intersection of any lines passing through a pair of points) but brings into focus the weakness in the argument.

The only situation which would allow the calculation of a unique point of intersection would be when the two points happen to coincide; and the point would then lie at this point of coincidence. The only time that this could occur is when the point is co-planar with both quadrilaterals (which are also planar).

In summary, this method contains a flaw in the form of the following contradiction:

> To generate lines each quadrilateral must *not* be co-planar
>
> but
>
> under certain circumstances the only way to calculate a unique point of intersection of these lines is if they *are* co-planar

Appendix D User Manual

This manual describes, in sufficient detail for actual users, how to use the tool (a brief description of the use of the tool for non users is included in §4.3 in the body of this dissertation).

The different functions may be classified according to either the menu hierarchy (that is a functional grouping - see §1) or according to the order in which the core functions are used (that is a procedural grouping - see §2). The remainder of this manual is made up of detailed descriptions of the use of each function. To simplify location of functions each starts on a new page, a table of contents is included on the following page and an index follows the body of the manual.

User Manual Table of Contents

1. Index of functions

Functionality can be accessed using the window menu (see Table 4) or the Control Panel (see Table 5).

Menubar	Submenu	Function	Refer to
File (Alt-F)	Open Image	Load a casual photograph image file.	§3 / p128
	Open Geometry	Load a three-dimensional geometry file describing the site with occlusions.	§4 / p129
	Open Texture	Load a previously calculated set of texture patches.	§5 / p130
	Save Geometry	Save three-dimensional site geometry to a local (.gmy) format	§6 / p131
	Save Texture	Save calculated texture patches to a local (.ttr) format.	§7 / p132
	Save VRML	Save patched site geometry to a VRML format file.	§8 / p133
	Save Geometry as Image File	Save a snapshot of the patched geometry in the current pose to an image file format.	§9 / p134
	Save adjusted image	Save an adjusted image to an image file format for use as a patch.	§10 / p135
	Print Image	Print image using operating system drivers	
Options (Alt-O)	Control Panel	Open an instance of the control panel.	Table 5
	Texture adjust Panel	Open an instance of the Texture adjust Panel.	§18 / p146
	Parameter Panel	Open an instance of the Parameter Panel.	§19 / p147
	Scale to fill	Change the scale of the image to fit the image display panel.	
Help	Manual	Link to an HTML version of this document	
	About	Version information	

Table 4 Functions arranged as per menu hierarchy

Button or slider	Function	Refer to
Filter sliders (LHS)	Set the image filter parameters	§14 / p140
Filter Now	Trigger filtering of the image	§14 / p140
Edging sliders (RHS)	Set the boundary calculation sliders	§15 / p142
Calibrate	Trigger image calibration	§12 / p137
Project Occlusion Boundary	Trigger projection of the occlusion boundary onto the image	§12 / p137
Find Edges Now	Find edges between seed points on the image	§15 / p142
Drape Now	Calculate hidden geometry (if applicable) and drape texture patch data onto geometry.	§17 / p145

Table 5 Functions accessible from the Control Panel

2. Sequence of use

The tool built is used according to the following sequence of steps:

1. *load casual photo* of relevant part of site. The casual photograph is the source of texture data to fill-in occluded regions.(see §3 / p128)
2. *load three-dimensional geometry* of the relevant part of the site. This geometry will be used to for casual photograph calibration, it indicates the occluded region and provides geometric data around the occluded region.(see §4 / p129)
3. *select corresponding features for calibration* which can be used to calibrate the casual photograph image are selected.(see §11 / p136)
4. *calibrate* the casual photograph image and *project the occlusion boundary* in order to show on the acsual photograph where the occlusion in the three-dimensional geometry occurs. The projected occlusion boundary also serves as an outer boundary of the area of casual photograph texture to use when filling-in the occlusion.(see §12 / p137)
5. *mark correspondences between image and geometry* which occur either on or near discontinuities or the occlusion boundary are selected (seed points). These correspondences segment the area within the occlusion into quadrilaterals. Each quadrilateral will serve as a patch when filling-in the occlusion.(see §13 / p139)
6. *filter image data* (optional) in order to make the detection of boundaries in the next step more robust.(see §14 / p140)
7. *detect boundaries* between seed points. These boundaries will serve as the edges of patches extracted from the casual photograph and provide an input to the geometry calculation process. (see §15 / p140)
8. *calculate geometry* with occluded regions in order to make the model more complete.(see §16/ p143)
9. *drape* the image data extracted over the corresponding geometry.(see §17 / p145)
10. *view and adjust* casual photograph image intensity, contrast and colour data so as to achieve a more reaslistic match with the surrounding photogrammetrically captured texture data. (see §18 / p146)
11. *save* results. (see §7 / p132 for Saving texture patches to local (.ttr) format, §8 / p133 for Saving patched three-dimensional model to VRML format, §9 / p134 for Saving snapshot of geometry to image file, §10 / p135 for Saving adjusted image)

Note that although the above sequence of steps must be followed as a minimum it is possible to repeat sub-sequences to improve results, for example steps 6 and 7 may be repeated until satisfactory edges are found.

3. Loading the casual photograph

Casual photograph data is the source from which patches texture data (and geometry information) will be extracted in order to fill-in occlusions. The image data may be in any of the following formats:

Microsoft Windows bitmap image file (.bmp)
CompuServe Graphics Interchange Format (.gif)
Joint Photographic Experts Group (.jpeg or .jpg)
Portable Network Graphics (.png)
Tagged Image Format file (.tif or .tiff)

The casual photograph data is loaded using a dialogue box as shown in Figure 79 accessible as menu item Open Image under the standard File item on the menubar.

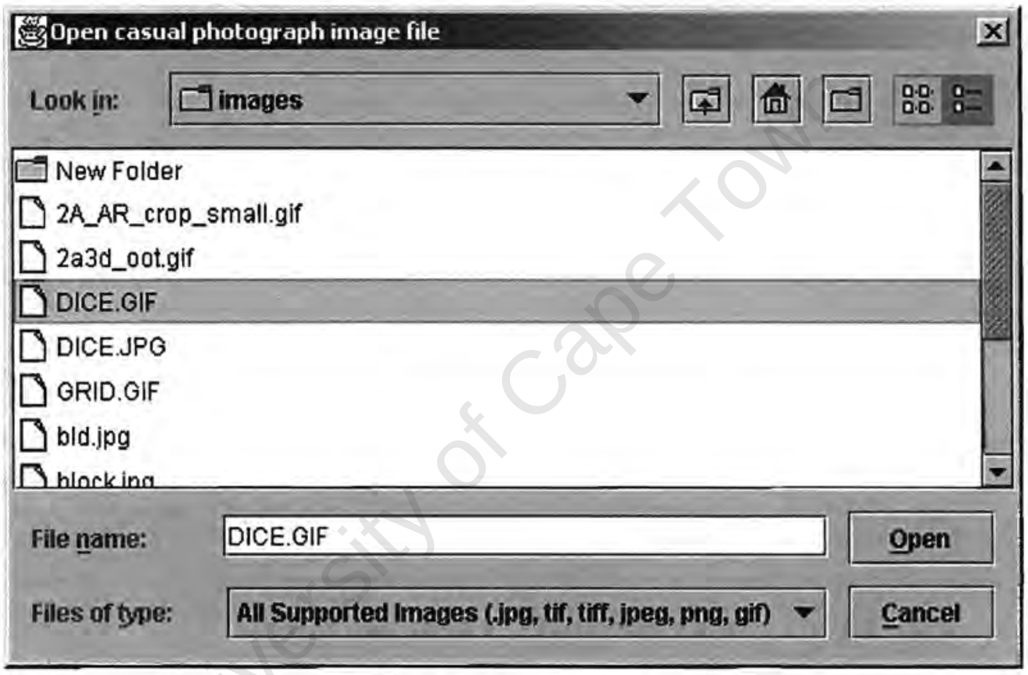


Figure 79 Image file opening dialog.

The name of the file containing the casual photograph data must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of image file types may be changed using the Files of type pull-down list.

4. Loading three-dimensional geometry

Three-dimensional geometry data is a definition of the site geometry and will include some occlusions. This data may in either the open AutoCAD-based Drawing Exchange Format (.dxf) or a simpler but less general format in which geometry generated by the tool can be saved (.gmy).

The known three-dimensional geometry is loaded using a dialogue box as shown in Figure 80 accessible as menu item Open Geometry under the standard File item on the menubar.

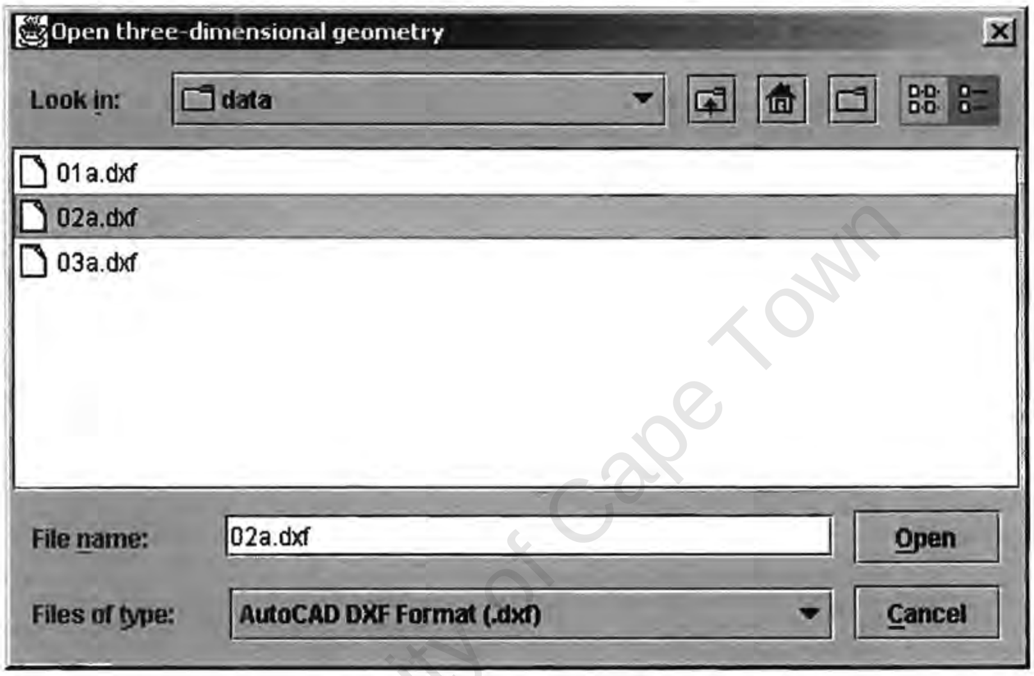


Figure 80 Geometry file opening dialog.

The name of the file containing the known scene geometry must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of geometry file types may be changed using the Files of type pull-down list.

5. Loading texture data

Previously calculated texture patches are defined as three-dimensional geometry and corresponding image texture coordinate data. This data is formatted as a list of three-dimensional geometric coordinates which define the edge of each patch together with and a link to an image file and corresponding image coordinates.

Previously calculated texture data is loaded using a dialogue box as shown in Figure 81 accessible as menu item Open Texture under the standard File item on the menubar.

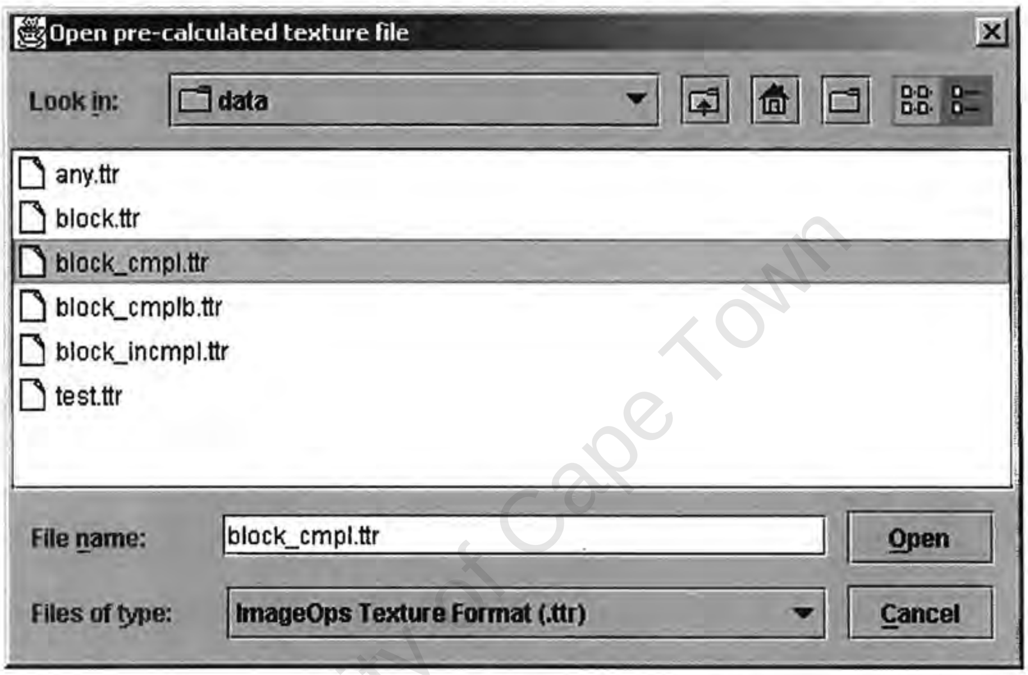


Figure 81 Pre-calculated texture data-opening dialog.

The name of the file containing the pre-calculated texture data must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of texture data file types may be changed using the Files of type pull-down list.

6. Saving three-dimensional geometry to local (.gmy) format

Three-dimensional geometry data is a definition of the site. This data will be saved in a local .gmy format, that is can only be read by the tool.

The three-dimensional geometry is saved using a dialogue box as shown in Figure 82 accessible as menu item Save Geometry under the standard File item on the menubar.

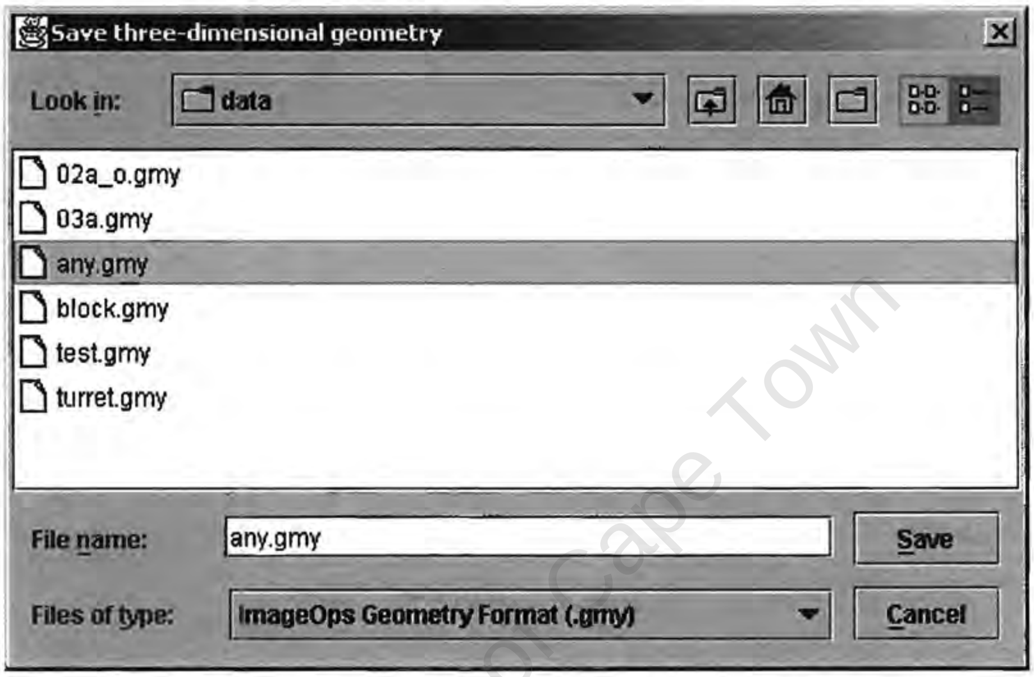


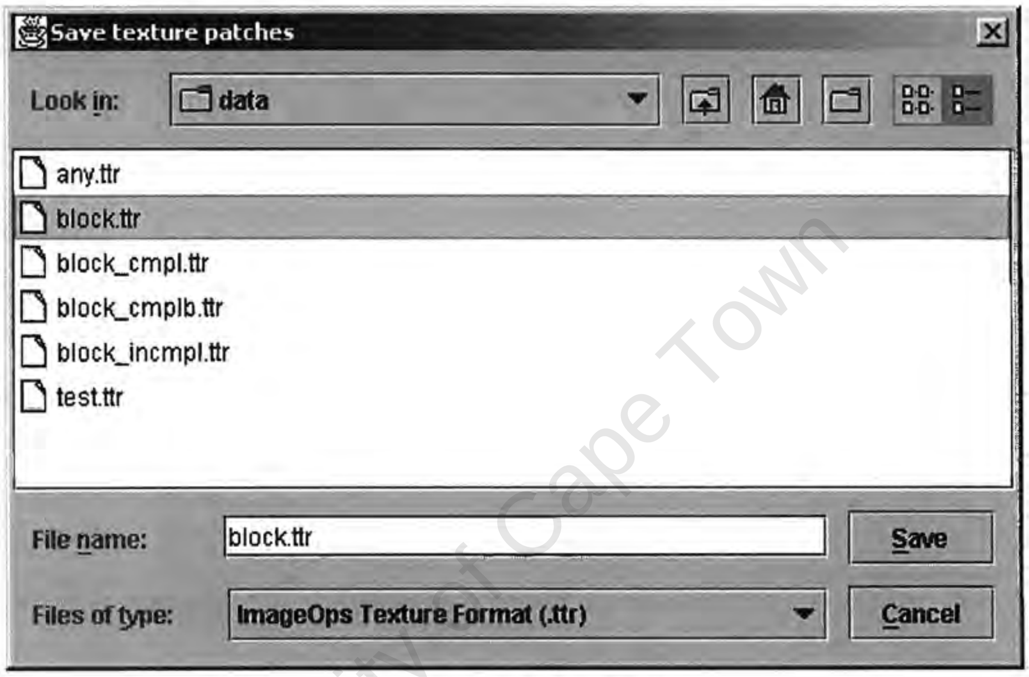
Figure 82 Geometry save dialog.

The name of the file containing the pre-calculated texture data must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of local geometry data file types may be changed using the Files of type pull-down list.

7. Saving texture patches to local (.ttr) format

Texture patches are defined as three-dimensional geometry and corresponding image texture coordinate data. This data is formatted as a list of three-dimensional geometric coordinates which define the edge of each patch together with and a link to an image file and corresponding image coordinates.

The texture patches saved using a dialogue box as shown in Figure 83 accessible as menu item Save Texture under the standard File item on the menubar.



any.ttr
block.ttr
block_cmpl.ttr
block_cmplb.ttr
block_incmpl.ttr
test.ttr

File name: block.ttr

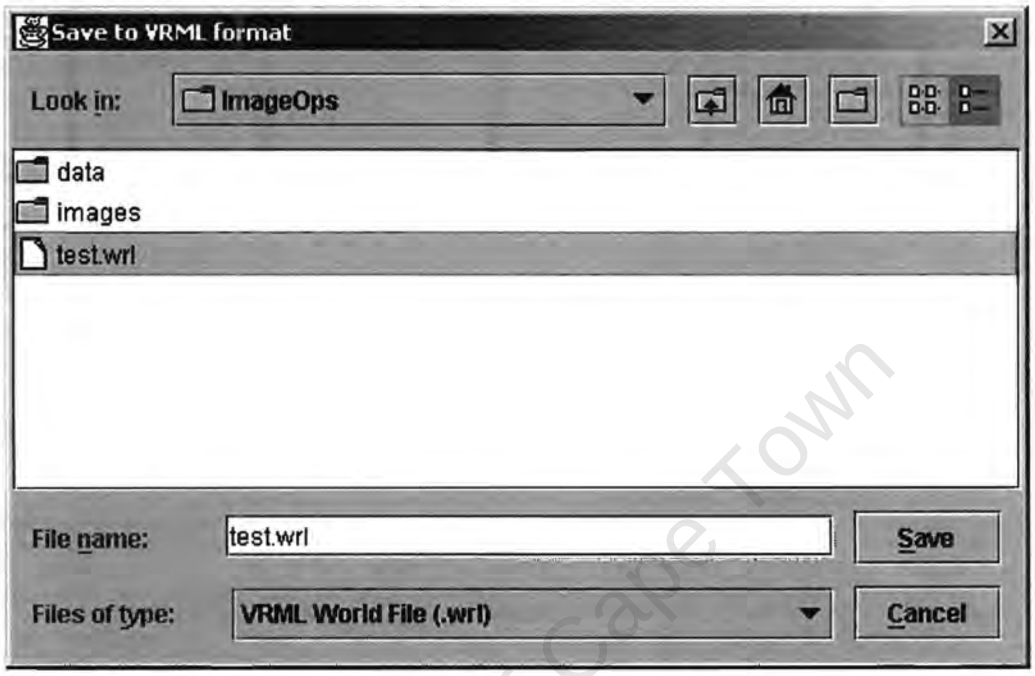
Files of type: ImageOps Texture Format (.ttr)

Figure 83 Texture patch save dialog.

The name of the file to containing the texture patches must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of texture data file types may be changed using the Files of type pull-down list.

8. Saving patched three-dimensional model to VRML format

VRML 2.0 compatible files containing the patched model can be created for viewing by standard browser plug-ins or other tools. They are saved using a dialogue box as shown in Figure 84 accessible as menu item Save VRML under the standard File item on the menubar.



File name: test.wrl

Files of type: VRML World File (.wrl)

Figure 84 VRML model save dialog

The name of the file to contain the VRML data must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of VRML file types may be changed using the Files of type pull-down list.

9. Saving snapshot of geometry to image file

Images showing the geometry in the current pose can be created by using a dialogue box as shown in Figure 85 accessible as menu item Save Geometry as Image File under the standard File item on the menubar.

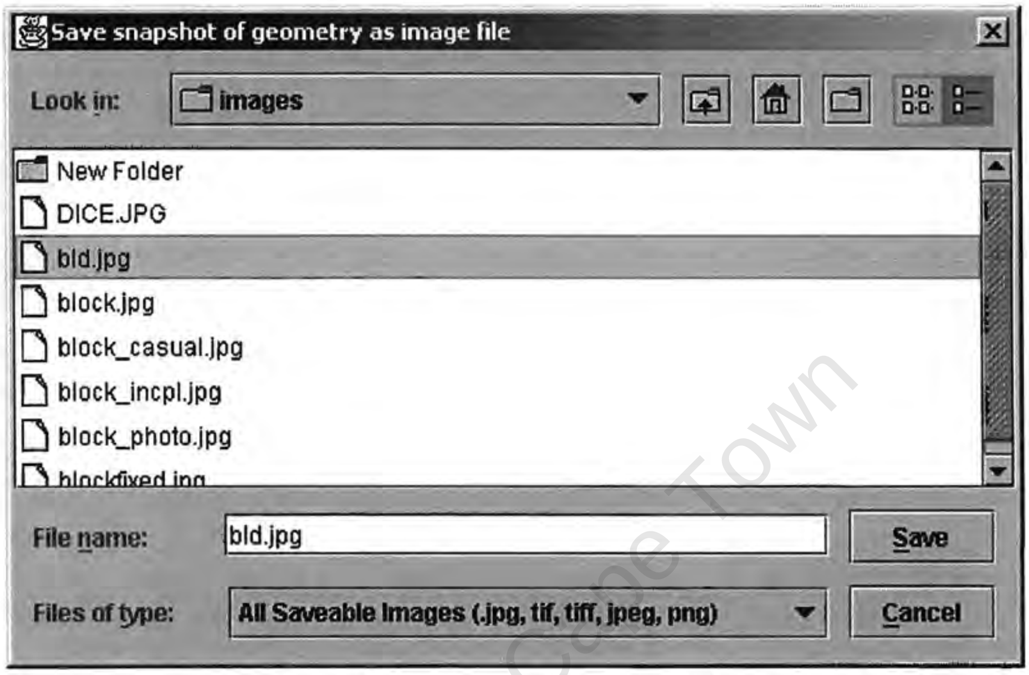


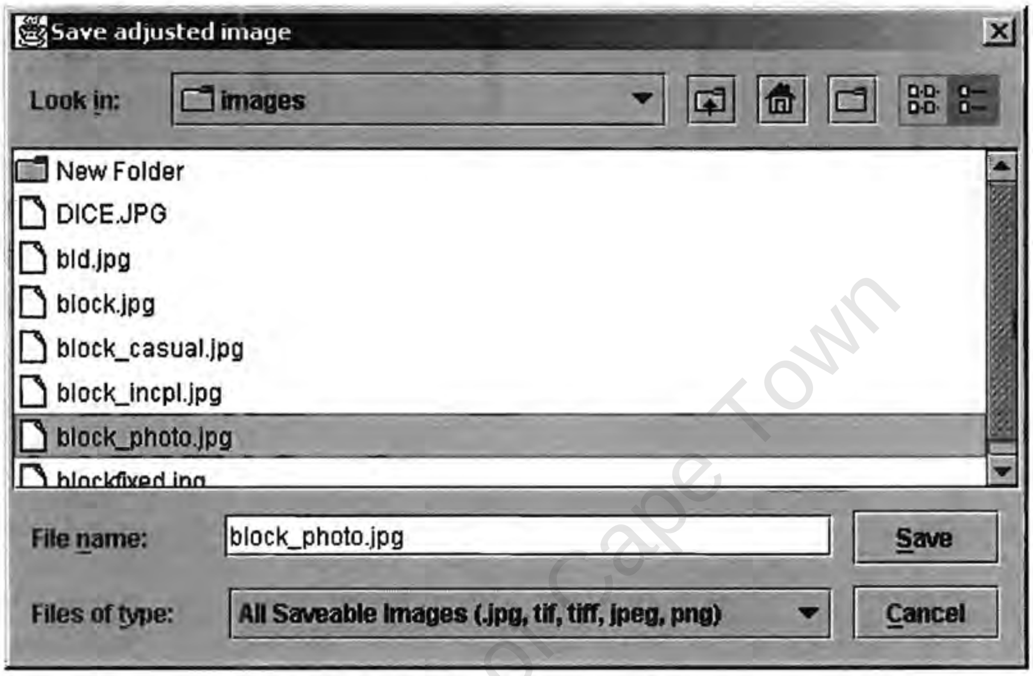
Figure 85 Geometry snapshot image save dialog.

The name of the file to contain the snapshot image of the geometry must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of image file types may be changed using the Files of type pull-down list.

10. Saving adjusted image to file

An image, once adjusted can be saved to JPEG, TIFF or PNG format for later use as part of patches.

The image is saved using a dialogue box as shown in Figure 86 accessible as menu item Save Adjusted Image under the standard File item on the menubar.



File name: block_photo.jpg

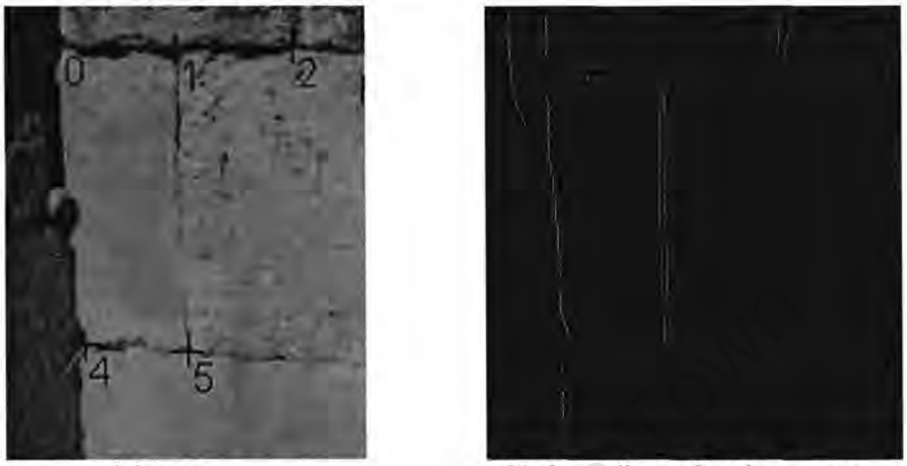
Files of type: All Saveable Images (.jpg, tif, tiff, jpeg, png)

Figure 86 Adjusted Image save dialog.

The name of the file to contain adjusted image data must be entered in the field File name. The directory containing the file may be changed using the Look in pull-down list and the pre-filtering of image file types may be changed using the Files of type pull-down list.

11. Select corresponding features for calibration

To allow calibration of the casual photograph a minimum of six corresponding features are marked on the geometry and image panels (see Figure 87). Calibration points are manipulated using the mouse together with the CTRL key; double-click adds a point, shift-double-click deletes a point and points can be dragged by holding down the left mouse button whilst moving the cursor. Calibration points are blue coloured cross-hairs.



a) image b) three-dimensional geometry

Figure 87 showing the calibration points created by the user as blue cross-hairs on the casual photograph (Figure 87a) and the three-dimensional geometry (Figure 87b). Corresponding points are given the same number.

12. Calibrate the casual photograph and project the occlusion boundary

The casual photograph is then calibrated based on the set of corresponding calibration points (see Figure 88). Calibration is triggered by the Calibrate button on the Control Panel.

The calibration parameters calculated are then used to project the three-dimensional calibration points onto the casual photograph (shown as the blue circles in Figure 88). The degree to which these circles are centred on the image calibration points (blue cross-hairs in Figure 87) is in an indication of the accuracy of the calibration. Errors are introduced into the calibration calculation by the inaccurate localisation of calibration points.

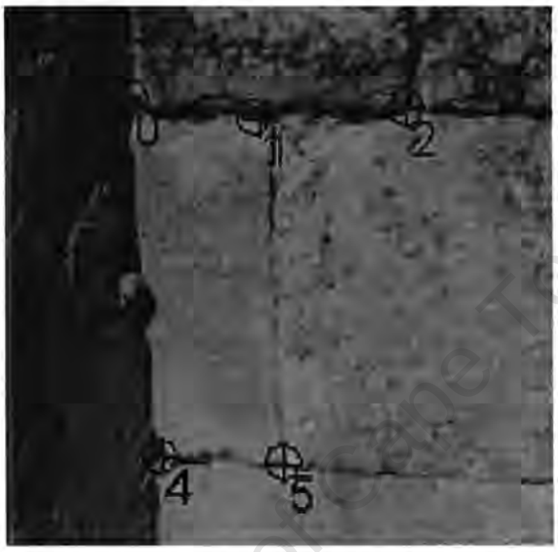
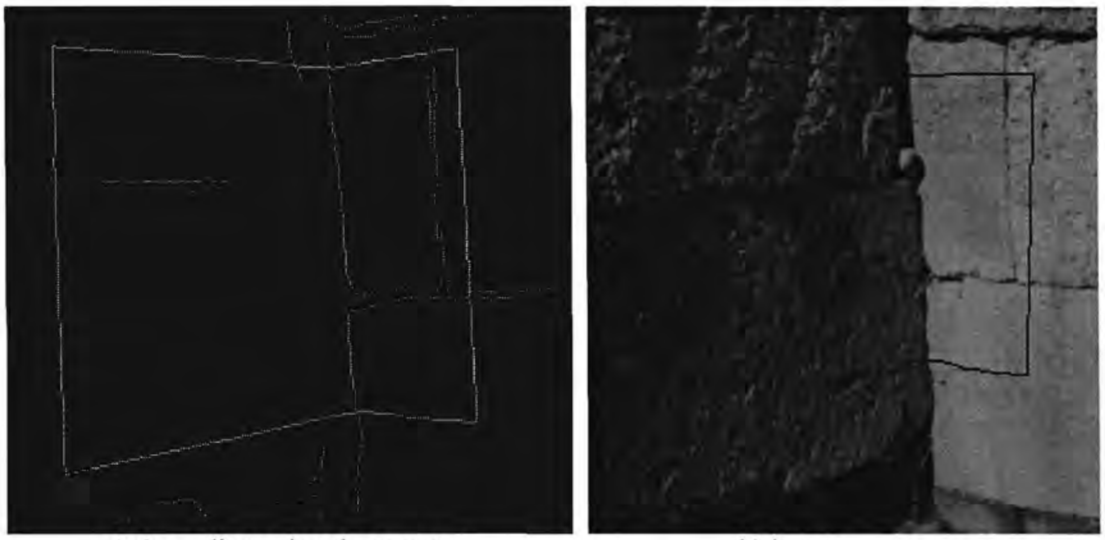


Figure 88 showing the calibration points (cross-hairs) each surrounded by the projection of the corresponding point from the three-dimensional geometry (circles). The deviation between the centres of the projected circles and the calibration points is an indication of the amount of error in the calibration parameters and serves as visual feedback as to the efficacy of the calibration.

Next the occlusion boundary is projected onto the casual photograph (as shown in Figure 89). Occlusion boundary projection is triggered by the Project Occlusion Boundary button on the Control Panel.

The source data from which the occlusion boundary is projected is a nominated set of line segments in the three-dimensional geometry data file (a layer in the AutoCAD .DXF file) which correspond to the edges of what has been captured photogrammetrically.



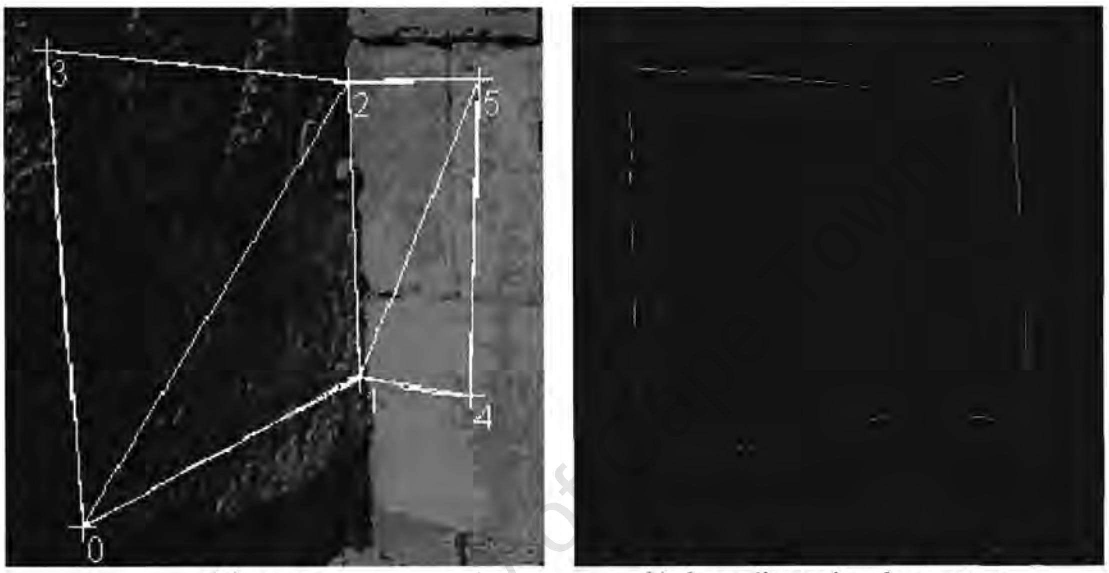
a) three-dimensional geometry b) image

Figure 89 which shows a) the occlusion boundary superimposed onto the three-dimensional geometry (the lime green rectangular shape) and b) the projection of this boundary onto the image (the blue rectangular shape).

13. Mark correspondences between image and geometry

Corresponding features are marked on the geometry and image panels using the mouse; double-click adds a point, shift-double-click deletes a point and points can be dragged by holding down the left mouse button whilst moving the cursor.

A grid of triangles is automatically generated between correspondences according their selection sequence (Figure 90). The lines between the corresponding points are added automatically by the application. Double-clicking on or near an existing point will *select* that point (rather than create a new point) for inclusion in a new triangle in the grid. Seed points and the lines between them are coloured red.



a) image b) three-dimensional geometry

Figure 90 which shows the boundary seed points created by the user as cross-hairs on a) the casual photograph and b) the geometry; along with the grid of triangles made up from the lines between seed points.

14. Filter image data

A control panel is available to set the filtration and edging parameters; and to trigger the calibration, boundary detection and draping functions (discussed previously). The filter parameters and trigger button fill the left half of the control panel (see Figure 91).

The filter parameters, which can be set using the control panel, are:

- box filter degree (effectively the spatial extent of the filter: for example degree 5 includes a single layer of surrounding pixels in a calculation of filtered pixel value, degree 10 two layers etc)
- Gaussian sigma is the spatial extent of the Gaussian filter
- DoG sigma is the spatial extent of the difference of Gaussian filter
- DoG sigma ratio is used to change "Mexican hat" profile difference of Gaussian filter (a value of 160 best approximates a Laplacian)

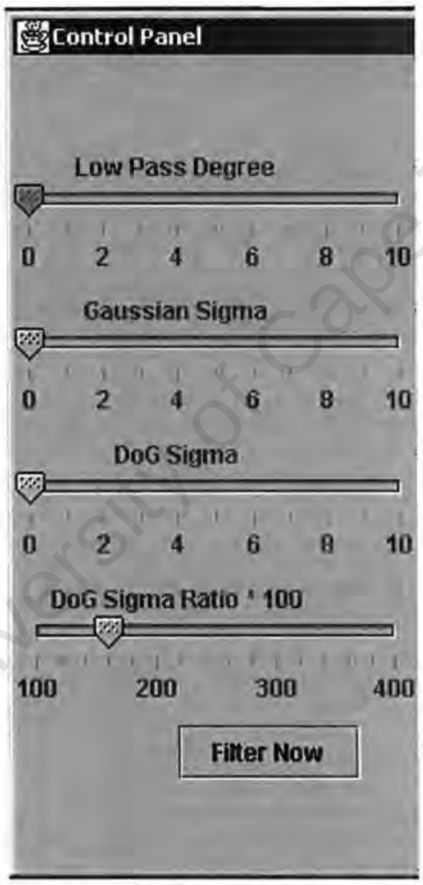
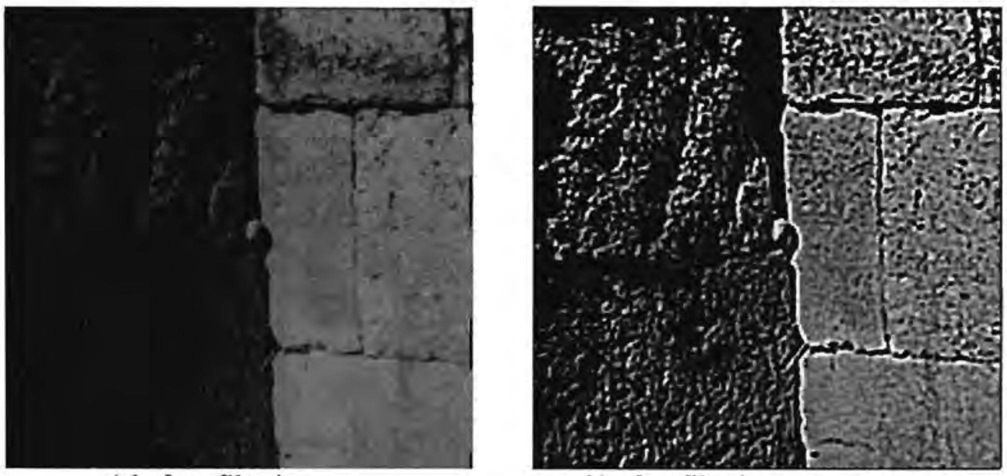


Figure 91 showing that portion of the control panel (the left half) which is used for setting the filtration parameters and triggering the filtering function; the right half of the control (shown in Figure 93) is used to trigger the calibration, boundary detection and draping functions. The filter parameters, which can be set, are box filter degree, Gaussian sigma, difference of Gaussian sigma (effectively the spatial extent of the filter of each of these thee filter types) and the difference of Gaussian sigma ratio, which tunes the difference of Gaussian filter.

The image data is filtered in order to reduce noise and fine-grained texture (see Figure 92)



a) before filtering b) after filtering

Figure 92 showing an image before (a) and after (b) difference of Gaussian filtering. Note how the edges have been sharpened.

15. Detect boundaries

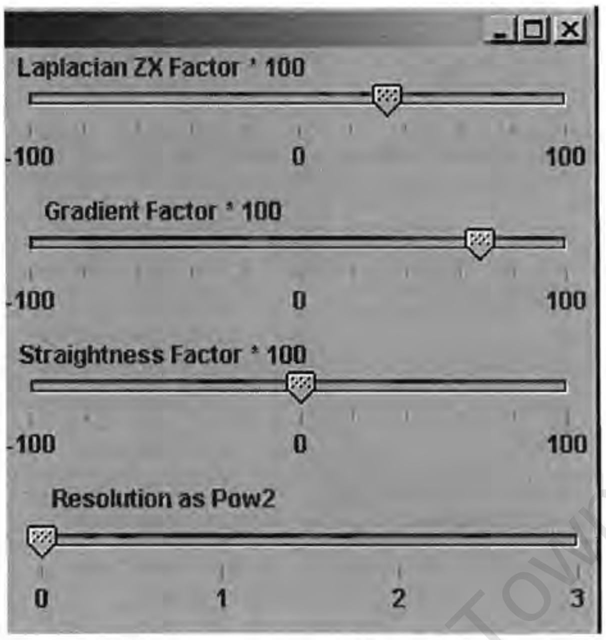


Figure 93. Controls for boundary detection which are included as the right half of the Control Panel (the left half is used for setting filter parameters and is shown in Figure 91). The first three sliders control the weighting factors for edge cost aggregation (namely $\omega_{Laplacian}$, $\omega_{gradient}$ and $\omega_{straightness}$) and the fourth slider controls the resolution of the edge calculation on order to speed up boundary detection on high-resolution casual photographs.

The boundary detection parameters are adjusted (if necessary) using the right half of the control panel (Figure 93). The boundaries are then detected and indicated on the casual photograph in red (see Figure 94)

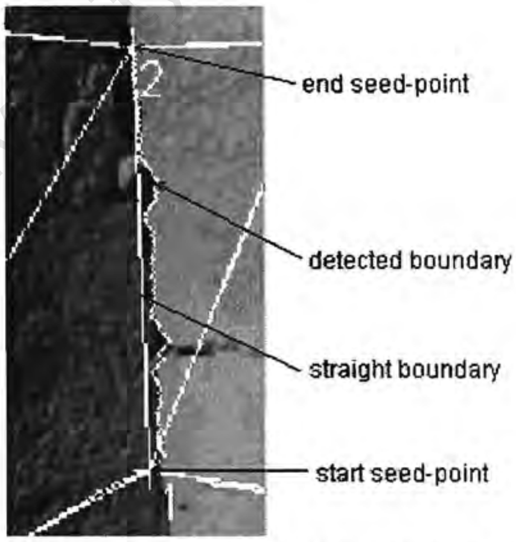


Figure 94 showing the boundaries as detected between seed-points 1 and 2 along with the straight boundaries connecting seed-points.

At this point seed points can be moved, filter or boundary detection parameters changed in order to find better boundary with which to proceed to the geometry calculation phase.

16. Calculate geometry

Once the image has been segmented (and boundaries detected when applicable) the calculation of hidden surface and corner geometry can triggered using the Drape Now button on the Control Panel

Geometry calculation is controlled by parameters which can be set using the parameter panel (see Figure 95). Parameters applicable to geometry calculation are as follows:

Corner Model selects whether the surfaces within the occlusion should be modelled as planar or complex (see Corner Function below) and whether the edges of the patches shown be left separate (as calculated by minimising least squares error) or closed up by calculating a mean value.

Corner Function (applicable only if a *complex* corner model is selected) is used to select whether the surfaces are to be modelled as quadratic or cubic.

Known Edge Stickiness Factor is the weight that the least mean squares error along edges with known geometry should have relative to hidden edges.

Target Convergence Ratio is used to set the point at which the geometry calculation ceases. Once the least mean squares error drops by less that this proportion the calculation is deemed to be complete.

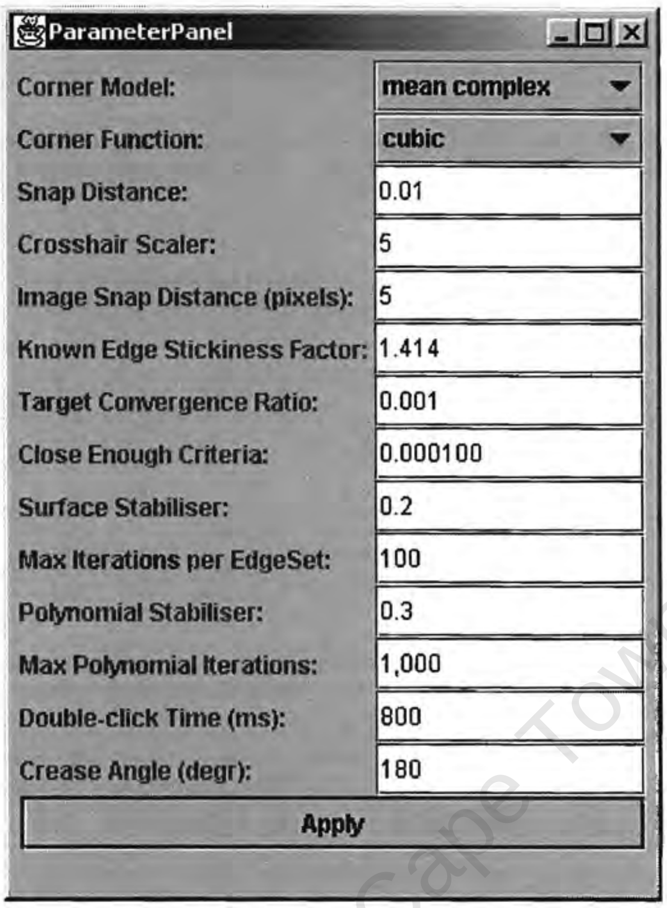
Close Enough Criteria is the permissible error in the calculation of d (the proportional distance of the pixel down its edge) for each pixel along an edge.

Surface Stabiliser is a value between 0.0 and 1.0 which attenuates the amount by which a coefficient it changed per iteration.

Max Iterations per Edge Set is the maximum number of iterations allows for calculating the hidden occlusion geometry.

Polynomial Stabiliser functions in the same way as the surface stabiliser but for the solution of the polynomial in d.

Max Polynomial Iterations is the maximum number of iterations allowed for each solution of the polynomial in d.



Corner Model:	mean complex
Corner Function:	cubic
Snap Distance:	0.01
Crosshair Scaler:	5
Image Snap Distance (pixels):	5
Known Edge Stickiness Factor:	1.414
Target Convergence Ratio:	0.001
Close Enough Criteria:	0.000100
Surface Stabiliser:	0.2
Max Iterations per EdgeSet:	100
Polynomial Stabiliser:	0.3
Max Polynomial Iterations:	1,000
Double-click Time (ms):	800
Crease Angle (degr):	180

Figure 95 Parameter panel, which is used to set the parameters governing the calculation of geometry hidden within occlusion. Parameters applicable to geometry calculation are corner model, corner function, known edge stickiness factor, target convergence ratio, close enough criteria, surface stabiliser, max iterations per edge set, polynomial stabiliser and max polynomial iterations.

17. Drape the image data over calculated geometry

Draping is initiated using the "Drape Now" button on the control panel. Draping consists of calculating a) the geometry and b) the texture mapping of occluded regions and then inserting these results to the three-dimensional geometry shown in the geometry panel (as shown Figure 96).

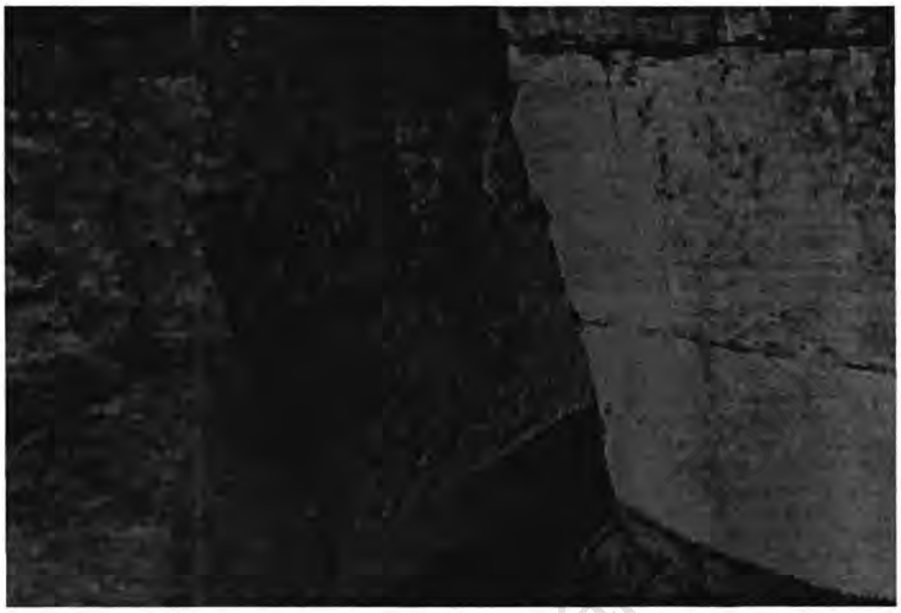
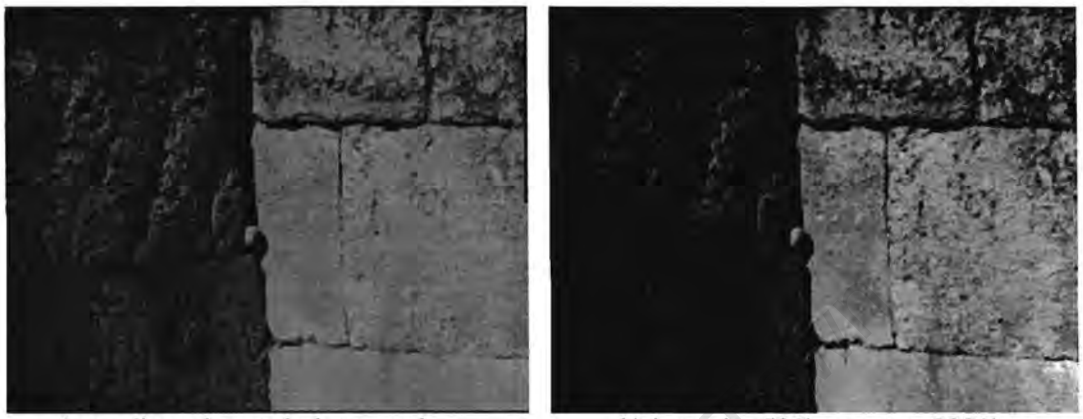


Figure 96 showing texture extracted from a casual photograph draped over a synthetic occlusion in the model.

18. View and adjust patch texture

In order to allow better matches between occlusion patches and surrounding texture a texture adjuster is available. Figure 97 shows the effect of texture adjustment on a casual photograph. Texture adjustment is controlled using the Texture adjuster control panel (see Figure 98), which allows the adjustment of the image intensity, contrast, colour saturation and colour balance.



a) unadjusted casual photograph b) intensity 50%, contrast 200%

Figure 97 shows the difference between a portion of a) an unadjusted casual photograph and b) the same image with intensity halved and contrast doubled.

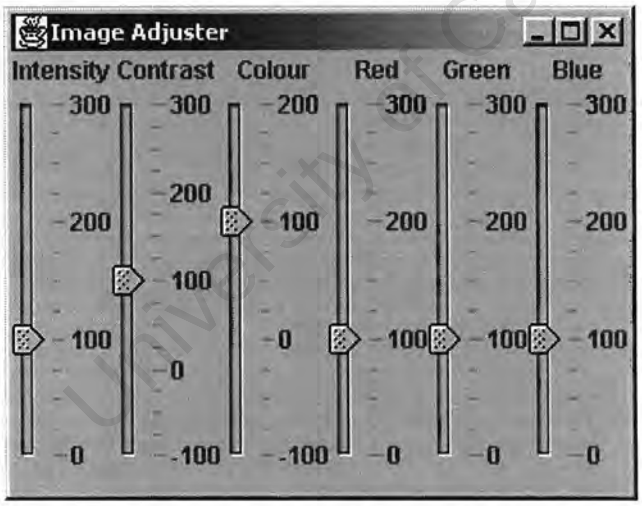
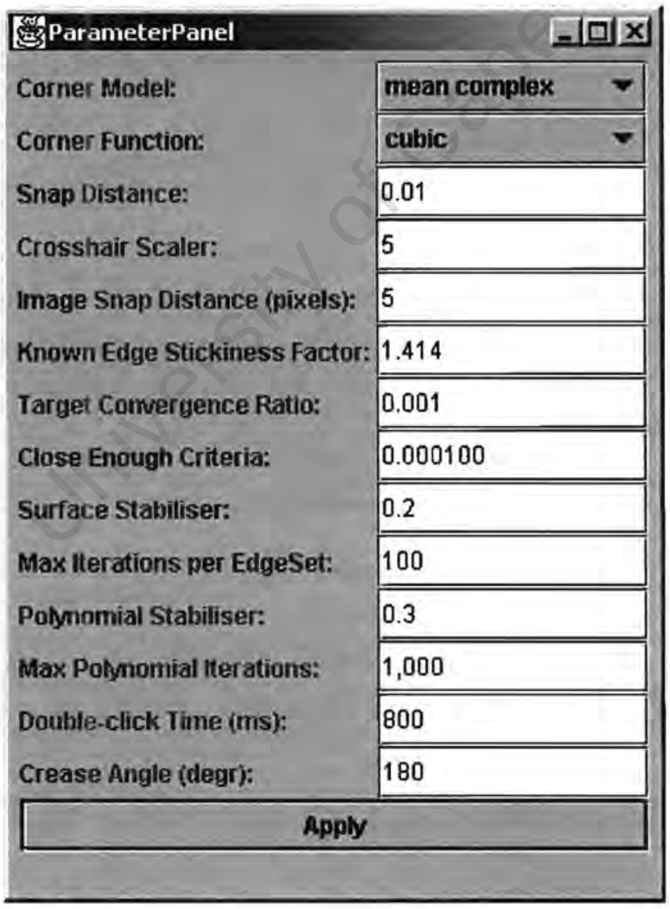


Figure 98. Texture adjuster control panel, which is used to adjust the image intensity, contrast, colour saturation and colour balance.

19. Parameter panel

In addition to its use in setting geometry calculation parameters the Parameter Panel (see Figure 99) can also be used for setting the following general parameters at any time during the application use:

> Snap Distance is the distance in units of the geometric model that a mouse click will snap onto a feature.
>
> Cross-Hair Scaler is the size of the cross-hair generated in units of the snap distance
>
> Image Snap Distance is the distance in pixels that a mouse click will snap onto an existing feature marker.
>
> Double-Click Time is the length of time in milliseconds below which two mouse clicks will be recognised as a double-click.
>
> Crease Angle is the angle above which any vertex on a calculated surface will be allowed to crease rather than be smoothed.



Corner Model:	mean complex
Corner Function:	cubic
Snap Distance:	0.01
Crosshair Scaler:	5
Image Snap Distance (pixels):	5
Known Edge Stickiness Factor:	1.414
Target Convergence Ratio:	0.001
Close Enough Criteria:	0.000100
Surface Stabiliser:	0.2
Max Iterations per EdgeSet:	100
Polynomial Stabiliser:	0.3
Max Polynomial Iterations:	1,000
Double-click Time (ms):	800
Crease Angle (degr):	180

Figure 99 Parameter panel which besides being used to set the parameters governing the calculation of geometry can be used to set other general application parameters. These additional parameters are the snap distance, cross-hair scaler, image snap distance, double-click time and crease angle.

20. Index