Surface Reconstruction from
Image Sequences

Texture and Apparent Contour Constraints

Geoffrey CROSS

Visual Geometry Group
Department of Engineering Science
University of Oxford

Trinity Term, 2000
Acknowledgements

As with most research, there are naturally numerous people to thank for their help, support and friendship throughout my stay in Oxford. I shall attempt to mention a cross-section of them here, but I can only hope not to forget too many!

Obviously, Prof. Andrew Zisserman should get recognition for his enthusiastic and unfailing support and supervision throughout the course of this research. I don't believe any of this work would have been possible without Andrew's faith in the seemingly impossible and I just hope this thesis does his ideas justice.

In the lab, both Fred and Dave get a special mention (as they saw this thesis through to the bitter end): Fred for never answering any question directly whilst at the same time over-flowing with useful information; Dave for his constant enthusiasm towards work and life in general. Despite them having deserted me before the end, Antonio, Phil and Big Dave also deserve special thanks for creating a great working and living environment. Andrew Fitzgibbon doesn't only deserve the customary thank you for code and technical help, but also for much fun outside the lab.

Outside working hours, many people in Holywell Manor deserve some sort of recognition. To name a few, I'd like to thank (in no particular order) Matthew, Nick, Alana, Clyde, Libbie and Beth for all being great friends. Holywell Manor means a lot to me, and they were all part of it. For three of my years in Holywell, Prof. Henry McQuay was Praefectus and having had much direct contact with him (as MCR President and Assistant Praefectus) I think he deserves some credit for the happiness I now look back on. And Helen: thank you for the support, love and faith; and thank you for being part of my life.

More directly related to the production of this thesis, Helen and Fred ran all over Oxford trying to get this thesis printed, bound and submitted on time, Josephine helped proofread and Matthew finally took the finished work-of-art to the binders: thank you. Financially, this work was supported by the EPSRC.

Finally, I would like to thank my examiners, Dr. Ian Reid and Dr. Andrew Stoddart, for the time they dedicated to reading this work and the interesting discussions which then followed.
Abstract

This thesis considers the general problem of surface reconstruction from a sequence of images, with particular emphasis on the combined use of various sources of information about the surface from image data. It is shown that both texture information over two or more views and silhouette information can be used to reconstruct the surface.

First, the problem of reconstructing a viewed quadric surface is considered. By placing a strong prior on the reconstructed surface, a reliable and accurate set of algorithms is presented. In particular, the advantages of quadric surface reconstruction in the dual-space are explained. It is shown that a quadric surface can be reconstructed from 9 points correspondences, and from 3 silhouettes. It is shown that a degenerate quadric can be reconstructed from 8 point correspondences or 3 silhouettes. Throughout, statistically robust methods for the reconstructions are proposed.

Second, the problem of reconstructing a general surface of unknown topology is approached. It is seen that silhouette and texture information naturally complement each other, and an efficient algorithm based on the space carving approach is presented. It is shown that a general surface, of arbitrary topology with concavities can be reconstructed.

Finally, a level-set approach is proposed, minimising both silhouette reprojection and texture correlation errors. A silhouette reprojection error metric is presented which can then be expressed as a surface evolution problem.

It is assumed that the camera motion is not known a priori, but algorithms for computing the camera projection matrices are proposed. Example results are given throughout the thesis.
Contents

1 Introduction and Background .............................................. 1
   1.1 Surface Reconstruction ........................................... 1
   1.2 Applications ..................................................... 1
   1.3 Literature ....................................................... 2
   1.4 Background Geometry ........................................... 4
      1.4.1 Notation ..................................................... 4
      1.4.2 Camera model ............................................... 4
      1.4.3 Epipolar geometry ......................................... 5
   1.5 Background on Conics and Quadrics ............................ 6
      1.5.1 Conics ....................................................... 6
      1.5.2 Quadrics .................................................... 10
   1.6 Overview ....................................................... 14

2 Reconstructing Quadric Surfaces — Theory .......................... 16
   2.1 Introduction ..................................................... 16
      2.1.1 Previous work and extensions ............................ 18
   2.2 Geometry of Imaged Quadrics .................................... 19
      2.2.1 Single view ................................................ 19
      2.2.2 Quadric apparent contour ................................ 20
      2.2.3 Quadric induced transfer ................................ 23
   2.3 Dual-Space Geometry ............................................ 25
2.3.1 Simple geometric constructions .......................... 26
2.3.2 Single view ............................................. 27
2.3.3 Two views .............................................. 28
2.3.4 Algebraic duality ......................................... 29
2.4 Quadric Reconstruction .................................... 32
   2.4.1 Reconstruction from one view .......................... 33
   2.4.2 Reconstruction from two views ........................ 35
   2.4.3 Reconstruction from three views ........................ 37
   2.4.4 Degeneracy constraint .................................. 39
2.5 Conic Reconstruction ..................................... 40
   2.5.1 Conic representation in dual-space ..................... 42
   2.5.2 Conic reconstruction in dual-space ..................... 42
   2.5.3 Dual-space cone to real-space conic ................... 43
2.6 Conclusions ................................................. 44

3 Reconstructing Quadric Surfaces — Implementation .... 45
   3.1 General Quadric Reconstruction .......................... 46
      3.1.1 Reconstruction from points ........................ 46
      3.1.2 Reconstruction from silhouettes ..................... 47
      3.1.3 Reconstruction from points and silhouettes .......... 52
   3.2 Degenerate Quadric Reconstruction ....................... 53
      3.2.1 Reconstruction from points ........................ 53
      3.2.2 Reconstruction from silhouettes ..................... 54
   3.3 Optimal Quadric Reconstruction .......................... 55
      3.3.1 Error model ......................................... 55
      3.3.2 Point constraint cost function ....................... 55
      3.3.3 Silhouette cost function ............................ 56
      3.3.4 Combined point and silhouette cost function ........ 57
      3.3.5 Implementation ...................................... 57
4 Reconstructing General Surfaces — Theory

4.1 Reconstruction from silhouettes
   4.1.1 General
   4.1.2 Silhouette cone
   4.1.3 Reconstructing from silhouettes
   4.1.4 Epipolar tangency
   4.1.5 Epipolar curves
   4.1.6 Epipolar nets
   4.1.7 The visual hull
   4.1.8 Limitations of the visual hull

4.2 Reconstruction from texture
   4.2.1 Triangulation
   4.2.2 Texture-based matching

4.3 Conclusions

5 Reconstructing General Surfaces — Implementation

5.1 Surface Representation
   5.1.1 Parameterised surfaces
   5.1.2 Volumetric surface representation
   5.1.3 Voxmap representation for reconstruction

5.2 Surface Reconstruction From Silhouettes
   5.2.1 Building a voxmap from silhouettes
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2 Improved efficiency algorithm</td>
<td>97</td>
</tr>
<tr>
<td>5.2.3 Results</td>
<td>102</td>
</tr>
<tr>
<td>5.3 Refining Reconstruction using Image Textures</td>
<td>103</td>
</tr>
<tr>
<td>5.3.1 Space carving—past and future</td>
<td>103</td>
</tr>
<tr>
<td>5.3.2 Surface directed initialization</td>
<td>104</td>
</tr>
<tr>
<td>5.3.3 Improved photo-consistency test</td>
<td>105</td>
</tr>
<tr>
<td>5.3.4 Indicator function</td>
<td>107</td>
</tr>
<tr>
<td>5.4 Gallery of results</td>
<td>108</td>
</tr>
<tr>
<td>5.5 Summary</td>
<td>108</td>
</tr>
<tr>
<td>6 Optimal Surface Reconstruction</td>
<td>120</td>
</tr>
<tr>
<td>6.1 Overview</td>
<td>121</td>
</tr>
<tr>
<td>6.1.1 Silhouette errors</td>
<td>121</td>
</tr>
<tr>
<td>6.1.2 Texture errors</td>
<td>121</td>
</tr>
<tr>
<td>6.1.3 Summary</td>
<td>122</td>
</tr>
<tr>
<td>6.2 Optimal Surface Reconstruction From Texture Information</td>
<td>122</td>
</tr>
<tr>
<td>6.3 Optimal Surface Reconstruction From Silhouettes</td>
<td>123</td>
</tr>
<tr>
<td>6.3.1 Silhouette reprojection errors</td>
<td>123</td>
</tr>
<tr>
<td>6.3.2 Evolution equation</td>
<td>124</td>
</tr>
<tr>
<td>6.3.3 Evolution from silhouettes and texture</td>
<td>125</td>
</tr>
<tr>
<td>6.4 Level-Set Representation of a Surface</td>
<td>125</td>
</tr>
<tr>
<td>6.4.1 Level-set embedding</td>
<td>126</td>
</tr>
<tr>
<td>6.4.2 Geometric properties of $S$</td>
<td>127</td>
</tr>
<tr>
<td>6.4.3 Level-set evolution</td>
<td>127</td>
</tr>
<tr>
<td>6.4.4 Advantages of the level-set representation</td>
<td>128</td>
</tr>
<tr>
<td>6.4.5 Discrete level-set formulation</td>
<td>128</td>
</tr>
<tr>
<td>6.5 Examples</td>
<td>129</td>
</tr>
<tr>
<td>6.6 Future Work and Conclusions</td>
<td>129</td>
</tr>
<tr>
<td>7 Recovering Camera Motion</td>
<td>133</td>
</tr>
</tbody>
</table>
7.1 Camera Motion and Calibration from Turntable Sequences 133
  7.1.1 Geometry of turntable sequences 134
  7.1.2 Projective ambiguity 135
  7.1.3 Reducing projective ambiguity 136
7.2 Camera Motion from Epipolar Tangencies 142
  7.2.1 Introduction 142
  7.2.2 Parallax geometry for smooth surfaces 144
  7.2.3 Computing the epipolar geometry 146
  7.2.4 Computing the trifocal geometry 150
  7.2.5 Computation of camera matrices for multiple views 153
  7.2.6 Conclusions and Extensions 156

8 Conclusions and Further Work 159
  8.1 Summary and Novel Contributions 159
  8.2 Future Work 162
    8.2.1 Quadric reconstruction 162
    8.2.2 General surface reconstruction 162

Bibliography 164

A Matrix Adjoint 178
Chapter 1

Introduction and Background

1.1 Surface Reconstruction

A fundamental problem in computer vision is the reconstruction of a 3-D object from an image sequence. The general problem considered in this thesis is: what can be inferred about an object from a number of images of the surface taken from arbitrary viewpoints?

Whilst considerable previous effort has been put into resolving this problem, it will be seen in section 1.3 that most of the current techniques only work accurately and reliably under controlled conditions. In contrast, this thesis considers the more difficult problem of reconstructing a surface given:

- the camera motion is unknown and recovered automatically from the image information;

- no assumptions are made about the existence or density of surface features such as edges, points, textures or even colour;

- no a priori information is known about the surface geometry or topology. This is contrasted with the case where a strong prior is place on the surface.

There are two main areas of research in this thesis. First, given a set of images of an object for reconstruction, how should the surface be represented? Second, how can the image data be used most efficiently to recover this surface representation?

1.2 Applications

The applications of surface reconstruction are clear and vast, with just three discussed here:
• **Virtual reality.** Figure 1.1 shows two image sequences, and a virtual reality world built from these images. The advantages of populating virtual reality worlds with models built automatically from images of real objects are clear;

• **Virtual museums.** Figure 1.2 depicts an image sequence of an ancient vase. It is increasingly attractive to be able to broadcast such works of art to the general public electronically;

• **Tele/Web-marketing.** The enormous potential of tele/web-marketing has only recently become apparent, and would clearly be enhanced by providing the consumer with a more realistic view of the merchandise.

### 1.3 Literature

The following section gives a high level review of previous attempts at surface reconstruction. It should be noted that a more complete summary is given in the relevant places in the following chapters throughout this thesis.

There has been much previous work on the general problem of reconstructing an unknown surface from a sequence of images of the surface. However, with a few exceptions which are revisited in chapters 4 and 5, much of the work has concentrated on either reconstructing a limited set of surfaces, or reconstructing a general surface from a well-defined and known camera motion.

Whilst most other areas of surface reconstruction are covered later in this thesis, it is appropriate to take a brief tour the group of algorithms refered to as dense stereo reconstruction. Dating back to the works of Baker and Binford [2] and Grimson [48], the dense stereo algorithm attempts to generate a dense depth map from a set of images. Ideally, for each pixel in the image (or even at a sub-pixel resolution), a depth is computed by triangulating from a matched point (correlating textures) in a second image. More efficient algorithms have been suggested: see Ohta and Kanade [85] and Cox *et al.* [27], for example.

There are two main limitations of currently available dense stereo techniques:
Figure 1.1: Two image sequences of two toy dinosaurs. Whilst the sequences are unrelated (taken at different times, different locations and with different cameras), a model of each dinosaur can be built, and a virtual world generated in which the models are manipulated and can interact with each other.
• Dense stereo is rather limited by the need to build a depth map in one image. Recent work [103] has shown that it is possible to paste together a number of reconstructions, but a “central” image must still be chosen. It is difficult to reconstruct an object which is circumnavigated, for example;

• Reconstruction of surfaces with sparse texture poses further problems. In these regions, the depth can not be accurately computed, and the reconstruction fails. Typically, a smoothness constraint is therefore imposed on the depth map [27, 65] and regions of low texture are interpolated. However, using this approach means that discontinuities and sharp edges on the surface are not accurately represented.

1.4 Background Geometry

1.4.1 Notation

This thesis employs a standard notation which is consistent throughout the work:

• Vectors are denoted by upper-case bold symbols (e.g. \( \mathbf{v} \)), and matrices by type-face symbols (e.g. \( \mathbf{M} \));

• 3-D points are denoted by upper-case bold symbols (e.g. \( \mathbf{X} \));

• 2-D points and image points are denoted by lower-case bold symbols (e.g. \( \mathbf{x} \));

• Quadrics, conics and other surfaces or curves are represented by the caligraphy symbols (e.g. \( \mathbb{Q} \) and \( \mathbb{S} \)).

1.4.2 Camera model

The most general linear camera model is known as a central projection camera (pinhole camera). A 3-D point in space is projected onto the image plane by means of straight visual rays, all passing through the focal point or camera center. The corresponding image point is given by the intersection of the image plane and the visual ray.
Algebraically, the 3-D point, \((X, Y, Z)\), is projected onto the image point \((x, y)\) by means of the homogenous equation

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
= P
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

where the equality is only up to scale, and the \(3 \times 4\) projection matrix represents the camera properties.

### 1.4.3 Epipolar geometry

Epipolar geometry is used extensively throughout this thesis, and a set of standard results are summarised here. For more detail, the reader is referred to [55] which also summarises in more detail the principles of epipolar geometry.

**Epipolar transfer**

Consider two cameras viewing a single 3-D point, \(X\). Figures 1.3 and 1.4 show that there is a plane, \(\pi\) which passes through the two camera centres, \(C\) and \(C'\), and \(X\). The plane intersects with the image planes in a pair of corresponding epipolar lines, \(I\) and \(I'\). The images of \(X\), \(x\) and \(x'\), must lie on the two epipolar lines \(I\) and \(I'\) respectively.

Thus, given \(x\) (the image of \(X\) in one image), and knowing \(C\) and \(C'\), the plane \(\pi\) can be found. From \(\pi\), the second epipolar line \(I'\) can also be found, and \(x'\) must lie on \(I'\). Summarising: *knowing the geometry of two cameras (i.e. the fundamental matrix, or the two camera projection matrices) and having an image of a 3-D point in one image, the search for the corresponding image point in the second image is reduced from a 2-D search to a 1-D search along the corresponding epipolar line.*

**Fundamental matrix**

The relationship between two images of a single 3-D point is best explained by the fundamental matrix equation

\[
x' F x = 0 ,
\]

(1.1)
where $\mathbf{F}$ is a $3 \times 3$ rank 2 matrix.

Summarising from [55, Table 8.1], the following results are presented here for completeness

- The epipolar lines are given by $l = \mathbf{F} \mathbf{x}$ and $l' = \mathbf{F}^\top \mathbf{x}'$;

- The epipoles are given by $\mathbf{Fe} = \mathbf{0}$ and $\mathbf{F}^\top \mathbf{e}' = \mathbf{0}$.

- The epipolar lines are transferred between the images by [75]: $l' = (\mathbf{e} \times \mathbf{F}^\top)^\top l$ where $\mathbf{e} \times \mathbf{x}$ denotes the cross-matrix$^1$ of $\mathbf{e}$.

### 1.5 Background on Conics and Quadrics

The following section provides some basic background to the geometry of conics in 2-D and quadric surfaces in 3-D. Much use will be made of this material in chapter 2 where the theory will be developed to encompass the imaging of quadric surfaces and the reconstruction of quadrics from multiple views.

#### 1.5.1 Conics

**Definition**

A conic is a family of planar curves generated by taking any cross-section of a 3D infinite cone. Examples of (proper) conics include parabolas, hyperbolas and ellipses (see figure 1.5). Degenerate conics will be defined and considered later.

Any point, $(x, y)$, on a conic satisfies the equation

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0 \ ,$$

where $a$, $b$, $c$, $d$, $e$ and $f$ are constant parameters. “Homogenizing” this by the replacements: $x \mapsto x_1/x_3$ and $y \mapsto x_2/x_3$ gives

$$ax_1^2 + 2bx_1x_2 + cx_2^2 + 2dx_1x_3 + 2ex_2x_3 + fx_3^2 = 0 \ ,$$

$^1$The cross-matrix of a vector $\mathbf{v} = (v_1, v_2, v_3)^\top$ is given by

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

such that the cross-product $\mathbf{v} \times \mathbf{x} = [\mathbf{v}]_\times \mathbf{x}$ for any vector, $\mathbf{x}$. 
Figure 1.2: An image sequence of a museum piece from the Ashmolean Museum, Oxford. A model of the artwork can be reconstructed and broadcast to a much wider audience than would normally have access to the museum, at an extremely low cost.

Figure 1.3: **Point correspondence geometry.** (a) The two cameras are indicated by their centres $C$ and $C'$ and image planes. The camera centres, 3-space point $X$, and its images $x$ and $x'$ lie in a common plane $\pi$. (b) An image point $x$ back-projects to a ray in 3-space defined by the first camera centre, $C$, and $x$. This ray is imaged as a line $l'$ in the second view. The 3-space point $X$ which projects to $x$ must lie on this ray, so the image of $X$ in the second view must lie on $l'$.

Figure 1.4: **Epipolar geometry.** (a) The camera baseline intersects each image plane at the epipoles $e$ and $e'$. Any plane $\pi$ containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines $l$ and $l'$. (b) As the position of the 3D point $X$ varies, the epipolar planes “rotate” about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.
Figure 1.5: **Examples of proper conics.** (a) Circle; (b) Ellipse; (c) Hyperbola and (d) Parabola.

or in matrix form

\[ \mathbf{x}^\top \mathbf{C} \mathbf{x} = 0 \; , \]

where the conic coefficient matrix, \( \mathbf{C} \), is given by

\[
\mathbf{C} = \begin{bmatrix}
a & b & d \\
b & c & e \\
d & e & f
\end{bmatrix} .
\]

Only the ratios of these matrix elements are important, since multiplying \( \mathbf{C} \) by a non-zero scalar does not affect equation (1.2) above. Thus, and noting that \( \mathbf{C} \) is symmetric, it can be seen that a conic has 5 degrees of freedom.

**Projective transformation**

Consider a 2-D projective transformation, \( \mathbf{H} \), such that \( \mathbf{x}' = \mathbf{Hx} \). Assuming the transformation to be non-singular, and substituting \( \mathbf{x} = \mathbf{H}^{-1} \mathbf{x}' \) into equation (1.2) gives

\[ \mathbf{x}'^\top \mathbf{H}^{-\top} \mathbf{C} \mathbf{H}^{-1} \mathbf{x}' = 0 \; . \]

As the matrix \( \mathbf{H}^{-\top} \mathbf{C} \mathbf{H}^{-1} \) is symmetric (and noting the similarity with equation (1.2)) equation (1.3) defines a new conic, \( \mathbf{C}' = \mathbf{H}^{-\top} \mathbf{C} \mathbf{H}^{-1} \). In general, a conic remains a conic under a projective transformation.

**Tangent line to conic**

The line \( \mathbf{l} \) tangent to the conic \( \mathbf{C} \) at a point \( \mathbf{x} \) (on the conic) is given by

\[ \mathbf{l} = \mathbf{C} \mathbf{x} \; . \]
This can be seen by observing that the line \( l \) passes through \( x \) since \( \mathbf{1}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{C} \mathbf{x} = 0 \) if \( x \) lies on the conic. Further, it is necessary to show that, as \( l \) is to be tangent to \( \mathcal{C} \), the line does not touch the conic at any other point. Proof by contradiction follows: if \( l \) meets \( \mathcal{C} \) at a second distinct point \( y \), then \( y^\top \mathcal{C} y = 0 \). In order to satisfy this equation, it follows that \((x + \alpha y)^\top \mathcal{C}(x + \alpha y) = 0\) for all \( \alpha \) which means that either \( x \) and \( y \) are coincident or the whole line \( l \) lies on \( \mathcal{C} \) [55, Result 1.7, page 10]. The latter case being possible only for degenerate conics.

**Conic envelope**

As there exists a tangent line for all points on a conic, it seems plausible that the conic could itself be defined by the tangents lines. Further, this follows from a standard result in projective geometry [55, Result 1.6, section 1]: *To any theorem of [2-dimensional] projective geometry there corresponds a dual theorem, which may be derived by interchanging the roles of points and lines in the original theorem.*

For a non-degenerate conic, equation (1.4) can be inverted to find the point at which the line \( l \) is tangent to the conic \( \mathcal{C}: \mathbf{x} = \mathbf{C}^{-1} \mathbf{l} \). Substituting into equation (1.2) gives

\[
(\mathbf{C}^{-1} \mathbf{l})^\top \mathcal{C} (\mathbf{C}^{-1} \mathbf{l}) = \mathbf{1}^\top \mathbf{C}^{-1} \mathbf{l} = 0
\]

as \( \mathcal{C} \) (and hence \( \mathbf{C}^{-1} \)) is symmetric. The “dual conic” \( \mathbf{C}^{-1} \) is known as a conic envelope, and as it is directly related to the conic \( \mathcal{C} \), it has five degrees of freedom (six parameters up to scale).

A more general proof (applicable to both rank 2 and 3 conics) can be found in [100], where it is shown that more generally

\[
\mathbf{1}^\top \mathcal{C}^* \mathbf{l} = 0
\]

where \( \mathcal{C}^* \) denotes the adjoint of \( \mathcal{C} \) (see appendix A). For a non-singular symmetric matrix, \( \mathcal{C}^* = \mathcal{C}^{-1} \) (up to scale).

**Degenerate conics**

Consider the conic \( \mathbf{x}^\top \mathcal{C} \mathbf{x} = 0 \) where the \( 3 \times 3 \) symmetric matrix \( \mathcal{C} \) is singular. Such conics are termed “degenerate” and it is often necessary to consider them as special cases (see
If the rank of $\mathcal{C}$ is 2, it has a one dimensional null-space, and thus there exists a single (homogeneous) point, $\mathbf{n}$, for which $\mathcal{C}\mathbf{n} = \mathbf{0}$. At this point, the tangent to the conic, as given by equation (1.4), is undefined. Such a conic is a pair of distinct lines, with $\mathbf{n}$ being the point of intersection of the lines (noting that $\mathbf{n}$ is not finite in the case of two parallel lines). Clearly, a rank 2 degenerate conic has $2 \times 2 = 5 - 1 = 4$ degrees of freedom (corresponding to 2 degrees of freedom for each line or five degrees of freedom minus the singularity constraint—a cubic constraint on the elements of $\mathcal{C}$).

The degenerate conic

$$
\mathcal{C} = \mathbf{l}\mathbf{m}^\top + \mathbf{m}\mathbf{l}^\top ,
$$

is composed of the two lines, $\mathbf{l}$ and $\mathbf{m}$. This can be seen by observing that all points on $\mathbf{l}^\top \mathbf{x} = 0$ are on the conic: $\mathbf{x}^\top \mathcal{C} \mathbf{x} = (\mathbf{x}^\top \mathbf{l})(\mathbf{m}^\top \mathbf{x}) + (\mathbf{x}^\top \mathbf{m})(\mathbf{l}^\top \mathbf{x}) = 0$. All points on $\mathbf{m}^\top \mathbf{x} = 0$ are also on the conic. The matrix $\mathcal{C}$ is symmetric and rank 2 (unless $\mathbf{l}$ and $\mathbf{m}$ are coincident in which case $\mathcal{C}$ has rank 1) [55, Example 1.8, page 11].

If the rank of $\mathcal{C}$ is 1, there is a two dimensional null-space: a (homogeneous) line. Such a conic is a pair of coincident lines. Following a similar argument to above, a rank 1 degenerate conic has 2 degrees of freedom.

**Envelopes of degenerate conics**

Equation (1.5) defines the conic envelope, $\mathcal{C}^*$, for a conic $\mathcal{C}$ which may not have full rank. Appendix A proves that the adjoint of a $3 \times 3$ matrix is non-zero if the rank of the matrix is greater than one. The consequence of this result is that the envelope of a rank 2 conic (pair of distinct lines) is defined, whereas the envelope of a rank 1 conic (pair of coincident lines) is not defined [100, chapter V, §7].

**1.5.2 Quadrics**

**Definition**

A 3-D point, $(x, y, z)$ on the surface of a quadric satisfies the equation

$$
a x^2 + 2bxy + cy^2 + 2dxyz + 2eyz + f z^2 + 2gx + 2hy + 2iz + j = 0 ,
$$
or in homogeneous coordinates \((x, y, z) \mapsto (x_1/x_4, x_2/x_4, x_3/x_4)\)

\[a x_1^2 + 2b x_1 x_2 + c x_2^2 + 2d x_1 x_3 + 2e x_2 x_3 + f x_3^2 + 2g x_1 x_4 + 2h x_2 x_4 + 2k x_3 x_4 + j x_4^2 = 0\,.

The equivalent matrix definition for a quadric surface is as follows:

\[X^\top Q X = 0\,.
\] (1.7)

with the quadric parameter matrix

\[Q = \begin{bmatrix} a & b & d & g \\ b & c & e & h \\ d & e & f & i \\ g & h & i & j \end{bmatrix}.
\]

In future analysis, it will be useful to expand equation (1.7) as

\[
\begin{bmatrix} X_3^\top & X \end{bmatrix} \begin{bmatrix} Q_{33} & q \\ q^\top & q \end{bmatrix} \begin{bmatrix} X_3 \\ X \end{bmatrix} = 0,
\] (1.8)

where \(Q_{33}\) is a symmetric \(3 \times 3\) matrix, \(q\) a 3-vector and \(q\) a scalar.

A quadric surface has nine degrees of freedom: a symmetric \(4 \times 4\) matrix having 10 distinct elements whilst only their ratios are significant in equation (1.7).

Examples of proper quadric surfaces include spheres, ellipsoids, hyperboloids and paraboloids (see figures 1.6 and 1.7).

**Degenerate quadrics**

As in the case of a conic, a degenerate quadric is characterized by the singularity of the parameter matrix, \(Q\). A cylinder, cone, or pair of planes (distinct or coincident) are examples of degenerate quadrics. Degeneracy is invariant to a projective transformation (see later on *quadric signatures*). Figure 1.8 depicts some example degenerate quadric surfaces.

A degenerate quadric has a maximum of \(9 - 1 = 8\) degrees of freedom (the singularity constraint provides a single quartic constraint on the elements of \(Q\)).

**Projective transformation**

Under a 3-D projective transformation \(H\), the quadric surface \(Q\) becomes \(Q' = H^{-\top} Q H^{-1}\) (the proof being almost identical to that of the conic under a 2-D projective transformation).
Quadric signatures

Since $Q$ is real and symmetric, it has real eigenvalues and an orthonormal basis of eigenvectors [46, Theorem 8.1.1, page 410]. Hence it can be written as $Q = V^T S V$ where $V$ is a real orthogonal matrix, and $S$ is a diagonal matrix. Appropriate scaling and reordering of the rows of $V$ permit this to be rewritten as $Q = V^T D V$ where $D$ is now diagonal with entries equal to $+1,-1$ or $0$ (in that order). The “signature” of the $D$, denoted $\sigma(D)$, is defined to be the number of $+1$ entries minus the number of $-1$ entries, and $\sigma(Q) = \sigma(D)$. It is shown in [55, Section 2.2.4] that the signature is unchanged under a projective transformation, whilst the rank is unchanged under a non-singular projective transformation [100, chapter V, §4].

Table 1.1 summarises the different quadric surfaces according to signature, and comparing with the equivalent table for the conics (table 1.2), the following note is made: whilst all proper conics are projectively equivalent, all proper quadrics (e.g. hyperboloid of one sheet and an ellipsoid) are not projectively equivalent. Figures 1.6, 1.7 and 1.8 give example quadrics.

<table>
<thead>
<tr>
<th>Rank</th>
<th>$\sigma(Q)$</th>
<th>$D$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>(1,1,1,1)</td>
<td>No real points</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(1,1,-1,-1)</td>
<td>Ellipsoid, hyperboloid of two sheets, paraboloid</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(1,-1,-1,-1)</td>
<td>Hyperboloid of one sheet</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(1,1,1,0)</td>
<td>Single real point</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>(1,1,-1,0)</td>
<td>Cone, cylinder</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(1,0,0,0)</td>
<td>Single line</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(1,-1,0,0)</td>
<td>Two distinct planes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,0,0,0)</td>
<td>Repeated planes</td>
</tr>
</tbody>
</table>

Table 1.1: Summary of the different categories of non-degenerate and degenerate quadric surfaces with associated rank and signature $\sigma(Q)$. Quadrics with different signatures are not projectively equivalent (see text).

Quadric envelope

The plane $\Pi$ tangent to the quadric $Q$ at $X$ is given by

$$\Pi = QX.$$  (1.9)
Figure 1.6: Example quadric surfaces (all projectively equivalent): (a) Sphere; (b) Ellipsoid; (c) Hyperboloid of 2 sheets; (d) Paraboloid.

Figure 1.7: Two examples of a hyperboloid of 1 sheet. Note that the hyperboloid of 1 sheet is not projectively equivalent to the quadrics in figure 1.6.

Figure 1.8: Example degenerate quadrics: (a) Cylinder and (b) cone (parameter matrix $Q$ has rank 3); (c) Pair of planes (rank 2).
Table 1.2: Summary of the different categories of non-degenerate and degenerate conics with associated rank and signature $\sigma(C)$.

The proof is similar to that given in section 1.5.1 above for the tangent line to a conic.

The quadric envelope, or “dual quadric” is similarly given by

$$\Pi^\top Q^* \Pi = 0,$$

(1.10)

with reference to the discussion of equation (1.5) above.

**Envelopes of degenerate quadrics**

As before, and with reference to appendix A, it should be noted that the envelope of a rank 1 or rank 2 quadric is undefined, as $Q^*$ collapses to zero. The envelope of a rank 3 quadric (i.e. a cylinder or a cone) is defined (with the same note on point singularities as above) by equation (1.10) [100, chapter XI, §1].

**1.6 Overview**

**Chapter 1** gives a brief literature survey of previous work on surface reconstruction. A basic background knowledge of both epipolar geometry and quadric surfaces is necessary for later chapters, and it is summarized in this chapter.

**Chapters 2 and 3** describe first the theory and then the implementation of a complete quadric surface reconstruction algorithm, from both point correspondences over two or more images and image silhouettes in a number of images.

**Chapters 4, 5 and 6** describe reconstruction of a general surface. The theory of reconstruction from image silhouettes and stereo information is given in chapter 4, whilst an algorithm to exploit these sources of shape information is described in chapter 5. Whilst
the algorithms in chapter 5 place a particular emphasis on efficiency, chapter 6 describes the “optimal” reconstruction of a surface: that is, the best reconstruction given a set of images of a surface.

Finally, chapter 7 outlines two novel approaches to computing the camera motion from a set of image sequences, which is assumed known in the reconstruction chapters. First, it is shown that robust and accurate projection matrices can be computed automatically from a sequence of images of an object on a turn-table; second the problem of computing camera motion for a sequence of an untextured object is considered.
Chapter 2

Reconstructing Quadric Surfaces — Theory

2.1 Introduction

Planar scene reconstruction has been studied in depth in many different works. Examples include, but are not limited to [6, 111, 121]. Quadric surface reconstruction is the natural extension from this work. For example, many fundamental principles of imaging and reconstructing planes are closely linked with the theory laid out in this chapter:

- Both planes and quadric surfaces provide a convenient binary partition of space;
- A transfer map (homography) is defined between two images of a plane: a similar concept (to be termed quadric induced transfer) is defined between two images of a quadric surface;
- A plane can be reconstructed from image correspondences over two or more views of the plane: it will be shown that a quadric surface can be reconstructed in a similar fashion;
- The “vanishing line” of a plane is in some way analogous to the “silhouette” of an imaged quadric surface: both provide important geometric cues about the underlying 3-D structure of the surface.

Being more general than a plane (the quadric surface has finite curvature), whilst maintaining desirable properties such as smoothness and compactness, a quadric surface has 9 degrees of freedom (a plane has 3).
As a precursor to showing how a quadric surface can be reconstructed from image information alone (chapter 3), it is necessary to extend the basic geometry of conics and quadrics summarised in section 1.5. Imaging a plane induces a well-understood homography between the plane in space and the image: imaging a quadric surface is not as simple. Once this has been studied in more detail, it is possible to use the relationships between a quadric and its image in order to reconstruct the quadric.

Section 2.2 shows that a single image point back-projects to a ray in space intersecting a quadric in zero, one or two points: hence a relationship between a point on the quadric surface and its image is developed (section 2.2.1). It then follows that the “silhouette” or outline of a quadric surface in an image is also related to the parameters of the quadric surface (section 2.2.2). Finally, the principle of quadric induced transfer is introduced: there is an (algebraic) relationship between any two images of a single quadric surface (section 2.2.3).

Whilst it is not difficult to impose point constraints on a quadric surface in real-space, imposing tangency constraints, as generated by the silhouette, is more easily investigated in the dual-space. Section 2.3 introduces this new approach. Sections 2.3.2 and 2.3.3 describe the dual-space equivalent of viewing a quadric surface from one and two views respectively. Section 2.3.4 then shows how the dual-space is not only useful as a visualization tool, but also provides a simple algebraic relationship between the dual quadric and its dual image outline.

With these preliminaries out of the way, the framework is then in place to discuss the reconstruction of a quadric surface from a single view (section 2.4.1), two views (section 2.4.2), and three or more (section 2.4.3) views. It has been shown that point constraints are best applied in real-space, whilst tangency constraints from image outlines are best applied in dual-space. It is seen that points impose one constraint on a quadric surface, whilst outlines impose 5 constraints on the dual quadric. However, it is also seen that two image outlines only impose 8 constraints, as two of the constraints are not independent.

Finally, section 2.5 shows that whilst a plane conic cannot be represented in real-space, in the dual-space it is represented by a degenerate quadric and reconstruction in the dual-
space proceeds in a similar way to the reconstruction of a degenerate quadric in real-space. Section 2.5.3 then shows how the parameters of the plane conic can be found from its dual-space representation.

2.1.1 Previous work and extensions

The earliest references to reconstruction of quadric surfaces are that of Karl et al. [64] and Ma and Li [78]. The existence of a linear relationship between a quadric envelope and its silhouette in an image is expressed in both papers, albeit not in the general form, or standard notation of this chapter. [79] correctly points out that a quadric surface cannot be completely reconstructed from two image silhouettes but fails to explain why other than noticing the degeneracy of the algebraic solution. Further, no attention is paid to the exact ambiguities in this reconstruction. No attempt is made to include degenerate quadrics in the solutions proposed.

Shashua and Toelg [106] also consider the problem of reconstructing a quadric surface, but this time attempt to mix the constraints from both point correspondences and silhouettes. Unfortunately, the approach chosen is almost impossible to generalize beyond the proposed algorithms for reconstructing a surface from a single silhouette and 4 point correspondences.

Finally, Kahl and Heyden [62] notice the degeneracy of two equations in the reconstruction of a general quadric from two silhouettes, but do not provide more than a note to this effect with a suggestion to an algebraic proof.

In contrast, this chapter provides a complete and general approach to quadric surface reconstruction:

- The principle of quadric induced transfer is introduced analogous to a plane induced homography between two images;

- By considering reconstruction in the dual-space, a more general and geometrically complete solution is found. For example, the reconstruction of a plane conic does not require any further specialization as is shown in section 2.5;
• Special care is taken in establishing the ambiguities of each reconstruction. These ambiguities are explained geometrically;

• The concept of a degeneracy constraint (that is, assume the imaged surface is a degenerate quadric and hence impose this constraint on the reconstruction) is introduced and in chapter 3 solutions will be proposed. Once again, care is taken to explain all ambiguities geometrically.

Many of these results are summarized in [30].

2.2 Geometry of Imaged Quadrics

2.2.1 Single view

Consider an image of a quadric surface. That is, any (finite) homogeneous point on the surface, \( \mathbf{X} \), projects into the image as \( \mathbf{x} = P \mathbf{X} \) where \( P \) represents the \( 3 \times 4 \) projection matrix for the camera. This projection equation can be expanded, representing the projection matrix as \( P = [\mathbf{A} \ a] \) with \( \mathbf{A} \) and \( \mathbf{a} \) being a \( 3 \times 3 \) matrix and a \( 3 \)-vector respectively, as:

\[
\begin{bmatrix}
\mathbf{A} & \mathbf{a}
\end{bmatrix}
\begin{bmatrix}
\mathbf{X}_3 \\
1
\end{bmatrix} = \lambda \mathbf{x} .
\]

The scalar, \( \lambda \), is an arbitrary scale factor.

It follows that back-projecting a ray from the image point \( \mathbf{x} \) gives

\[
\mathbf{X}_3 = \lambda \mathbf{A}^{-1} \mathbf{x} - \mathbf{A}^{-1} \mathbf{a} .
\] (2.1)

Substituting equation (2.1) into the quadric equation (1.8) gives a quadratic equation in \( \lambda \). To simplify the notation, the substitution \( \mathbf{c} = -\mathbf{A}^{-1} \mathbf{a} \) is made (\( \mathbf{c} \) representing the camera centre):

\[
\left( \mathbf{x}^\top \mathbf{A}^{-\top} \mathbf{Q}_{33} \mathbf{A}^{-1} \mathbf{x} \right) \lambda^2 + 2 \left( \mathbf{x}^\top \mathbf{A}^{-\top} \mathbf{Q}_{33} \mathbf{c} + \mathbf{x}^\top \mathbf{A}^{-\top} \mathbf{q} \right) \lambda + \left( \mathbf{c}^\top \mathbf{Q}_{33} \mathbf{c} + 2 \mathbf{c}^\top \mathbf{q} + q \right) = 0 .
\] (2.2)

Hence there are zero, one or two real solutions for \( \lambda \), which, substituting into equation (2.1), gives \( \mathbf{X} \) (the point(s) on \( \mathcal{Q} \) projecting to \( \mathbf{x} \)). The number of solutions for \( \lambda \) has the follow geometric interpretation:
• If there are two real solutions for $\lambda$, there must be two corresponding points, $X_1$ and $X_2$, from equation (2.1). Hence the ray hits the quadric surface at two points and the image point $x$ lies within the image of the quadric (see figure 2.1);

• If there are no real solutions for $\lambda$, the ray defined by equation (2.1) does not intersect with the quadric $Q$ at any real point. Thus, the point $x$ does not lie within the image of the quadric. This situation is depicted in figure 2.2. As $\lambda$ is complex, the point $X$ will be correspondingly complex and have no geometric meaning;

• Finally, it is possible that there is just one real solution (two equal roots) to equation (2.2). In this case, there is just one point of contact between the ray and the quadric surface: the ray “touches” the surface, and the point $X$ lies on the contour generator (see figure 2.3). Hence, the point $x$ lies on the apparent contour or silhouette of the surface in the image. This will be considered in more detail in the next section.

Degeneracies

It should be noted that equation (2.2) makes no assumptions about the rank of $Q$, and therefore holds for for all quadric surfaces.

If the quadric passes through the camera centre $c$, the final term, $c^T q_{ss} c + 2c^T q + q$, disappears, and $\lambda = 0$ is a solution for all $x$, as expected.

2.2.2 Quadric apparent contour

3-D to 2-D

A point on the apparent contour of $Q$ is characterized by the fact that the back-projected ray touches the surface at a single point. In this situation, equation (2.2) gives two coincident solutions for $\lambda$.

It follows that the two solutions are given by

$$
\lambda_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},
$$

(2.3)
Figure 2.1: An image point (green) back-projects to a ray in space (blue). If the image point lies within the image of the quadric surface, the ray intersects the surface at two points (red).

Figure 2.2: If the image point (green) does not lie within the image of the quadric surface, the back-projected ray (blue) does not intersect with the surface.

Figure 2.3: If the image point (green) lies on the apparent contour of the quadric surface, the back-projected ray (blue) touches the surface at a single point on the contour generator.
2 Reconstructing Quadric Surfaces — Theory

with

\[ a = x^\top A^{-T} Q_{33} A^{-1} x \]
\[ b = 2 \left( x^\top A^{-T} Q_{33} c + x^\top A^{-T} q \right) \]
\[ c = c^\top Q_{33} c + 2c^\top q + q \]

The two solutions are coincident if, and only if, the discriminant, \( b^2 - 4ac \), is zero:

\[ x^\top (A^{-T} \left[ Q_{33} c c^\top Q_{33} + 2q c^\top Q_{33} + q q^\top - (c^\top Q_{33} c + 2c^\top q + q) Q_{33} \right] A^{-1}) x = 0 . \tag{2.4} \]

With reference to equation (1.2), it can be seen that equation (2.4) defines a conic in the image such that the image point \( x \) obeys \( x^\top C x = 0 \) where

\[ C = A^{-T} \left[ Q_{33} c c^\top Q_{33} + 2q c^\top Q_{33} + q q^\top - (c^\top Q_{33} c + 2c^\top q + q) Q_{33} \right] A^{-1} . \]

As an aside, it is satisfying to note that this conic can be written as \( C = A^{-T} C' A^{-1} \) where

\[ C' = Q_{33} c c^\top Q_{33} + 2q c^\top Q_{33} + q q^\top - (c^\top Q_{33} c + 2c^\top q + q) Q_{33} . \]

\( C' \) is also a conic which is only dependent on the camera centre and \( Q \), whilst \( C \) is a 2-D projective transformation (homography) of \( C' \). This is an example of the more general rule that a change of the orientation and other parameters of the camera (here specified by \( A \)) whilst maintaining the camera centre fixed, induces a general homography in the image.

Summarizing this result: the apparent contour for a quadric surface is a conic in the image plane.

Special case

It is often possible to choose the projective frame for a reconstruction, and it is common [7, 77] to choose one camera matrix as \( P = [I \ 0] \) (thus \( A = I \) and \( c = 0 \)): equation (2.4) reduces to

\[ x^\top \left( q q^\top - q Q_{33} \right) x = 0 , \]

or

\[ C = q q^\top - q Q_{33} . \tag{2.5} \]

This special case will be considered further in section 2.4.1.
2-D to 3-D

Consider the conic \( C \) as the apparent contour of an unknown quadric. Each point \( x \) on the conic back-projects to a ray in space defined by \( x = PX \). Substituting into the conic equation (1.2) gives

\[
X^TP^TCPX = 0.
\]

Clearly, \( P^TCP \) is a \( 4 \times 4 \) symmetric matrix of maximum rank 3 (as \( P \) has rank 3), and by comparison with equation (1.7), these back-projected rays define a quadric

\[
Q_o = P^TCP.
\]  

(2.6)

\( Q_o \) is a cone with a singularity at the camera centre (as \( Pc = 0 \) and touches the generating quadric along the contour generator (see figure 2.4).

Degeneracy

Once again, no assumptions are made here about the rank (degeneracy) of \( C \). If \( C \) is rank deficient, the quadric \( Q_o \) is correspondingly rank deficient. That is, a full rank conic back-projects to a rank 3 quadric (a cone, with a point singularity at \( c \)); a rank 2 conic (pair of lines, crossing at the image point \( x_o \)) back-projects to a rank 2 quadric (a pair of planes, with a line singularity along the ray through \( c \) and \( x_o \)).

Special case

Continuing the special case of \( P = [I \ 0] \) from above, equation (2.6) gives

\[
Q_o = \begin{bmatrix} qq^T & qQ_{33}^T & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.
\]

2.2.3 Quadric induced transfer

It is well known that two views of a planar surface are related by a homography. This relationship between the images has been used in super-resolution \([20, 21]\) and matching \([88]\).
Figure 2.4: The apparent contour back-projects to a “cone” in space, passing through the camera centre, and tangent to the generating quadric along the contour generator.

This section demonstrates that a similar relationship between two images of a quadric surface is available. Once again, this can be used in super-resolution, the generation of new views [129] and matching of features.

Consider now two images of a quadric surface. A point in the first image, \( \mathbf{x} \), back-projects to a ray in 3D. This ray, in turn, intersects the quadric surface at two points (which may be real, coincident, or complex as shown previously). These two points, \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \), are then imaged in the second image as \( \mathbf{x}'_1 \) and \( \mathbf{x}'_2 \). This mapping, from \( \mathbf{x} \) to \( \mathbf{x}'_1 \) and \( \mathbf{x}'_2 \), is termed \textit{quadric induced transfer} (see figure 2.5).

As above, for the case where \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \) are coincident, \( \mathbf{x} \) lies on the apparent contour of the surface in the first image. Note that \( \mathbf{x}'_1 = \mathbf{x}'_2 \) does not, in general, lie on the silhouette in the second image.

Further, whilst geometrically the point \( \mathbf{x} \) transfers to two points in a second view, in practice this is not the case. As one of \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \) will obscure the other (assuming the quadric to be opaque\(^1\)), only one is visible in the first view, and it only makes sense to chose this 3-D point to be transfered to the second view. See section 3.4 for further consideration of this problem where examples are given.

\(^1\)This is termed the \textit{opacity constraint} in [106].
2 Reconstructing Quadric Surfaces — Theory

Figure 2.5: **Quadric induced transfer.** A point in one image back-projects to a ray in space. This in turn intersects the quadric in two points, which project as two points in the second image.

2.3 Dual-Space Geometry

Considering point quadrics has provided some useful results. However, equation (2.4), relating the conic silhouette to the generating quadric surface, is complicated and cumbersome to use in practice (each element of $C$ is a quadratic in the elements of $Q$). Shashua and Toelg [106] have shown how one such equation can be used to provide constraints from an image conic on a quadric in space, but it is difficult (and not solved) to extend this approach further: for example, it is seemingly not possible to use the same approach to reconstruct a quadric from two or more image silhouettes.

Generally equation (1.7) is used to define a quadric surface, relating a set of 3-D points to the quadric parameters. Therefore, given a point on the surface of an unknown quadric, it is not difficult to express this as an algebraic constraint on the quadric (section 2.4.2). However, it is not as easy to express the relationship between a conic silhouette and the quadric parameters. The silhouette back-projects to a cone which is tangent to the quadric surface: it is therefore clear that the silhouette provides a set of tangency constraints on the quadric.
<table>
<thead>
<tr>
<th>Real-space</th>
<th>Dual-space</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadric, $Q$</td>
<td>quadric, $Q^*$</td>
</tr>
<tr>
<td>line</td>
<td>line</td>
</tr>
<tr>
<td>point</td>
<td>plane</td>
</tr>
<tr>
<td>plane</td>
<td>point</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of various primitives and their tangent plane dual-space representation.

Fortunately, the tangent plane dual space [16, 17, 18] provides a convenient way of visualising complex real-space geometry constructions, and has particularly useful properties when applied to projective camera cases due to the projective invariance of tangency.

In particular, the back-projected outline of a surface in a perspective view is equivalent (or dual to) a plane section of the dual of that surface, where the sectioning plane is the plane dual to the camera centre [16, 17]. This result will be developed in sections 2.3.1 and 2.3.2.

Section 2.3.3 will then describe the dual-space equivalent of two cameras viewing a single quadric surface, with particular reference to epipolar tangencies.

Finally, section 2.3.4 provides a (linear) algebraic relationship between an image silhouette and the quadric surface in dual-space.

### 2.3.1 Simple geometric constructions

The following relationships between real-space and dual-space are necessary in the discussions of the subsequent sections. Table 2.1 summarises the transformation from real to dual-space.

**Dual plane**

Consider a plane, $\Pi$, in 3-D as a surface. Naturally, at any point, $X$, on the plane (such that $\Pi^T X = 0$), the surface tangent plane is given by $\Pi$. Thus, the dual to the plane $\Pi$ is given by $\Pi$ itself, and can be visualized as the point $\Pi$ in dual space.
2 Reconstructing Quadric Surfaces — Theory

Dual line

A similar approach can be taken for a 3-D line, \( \mathbf{l} \). There is clearly a one parameter family of planes which are all tangent to the line at any point on \( \mathbf{l} \). Therefore, the real-space line translates to a line in dual-space. Indeed, given two arbitrary planes, \( \Pi_1 \) and \( \Pi_2 \), which intersect in \( \mathbf{l} \), the family of tangent planes to \( \mathbf{l} \) is given by \( \Pi_\lambda = \Pi_1 + \lambda \Pi_2 \). Each of these planes is visualized as a point in dual-space, and therefore the dual to \( \Pi_\lambda \) is the line passing through the duals to \( \Pi_1 \) and \( \Pi_2 \).

Dual point

If a 3-D point, \( \mathbf{X} \), is defined as the intersection of 3 planes, \( \Pi_1, \Pi_2 \) and \( \Pi_3 \), the dual to \( \mathbf{X} \) is the plane passing through the duals of these three planes (points). The proof is trivial, and omitted as it is very similar to that given above for a dual line.

Dual quadric

It has already been seen (section 1.5.2) that the dual to the quadric \( Q \) is \( Q^* \) if \( Q \) is full rank.

If \( Q \) is rank 3 (that is, a cone or cylinder), the dual is a plane conic (this is seen by construction in section 2.3.2 below).

If \( Q \) is rank 1 or 2, the quadric surface is made up of a pair of coincident or distinct planes: the dual is correspondingly a pair of coincident or distinct points.

These duals are summarized in table 2.2.

2.3.2 Single view

With the previous section in mind, it is now possible to visualise figure 2.4 (a quadric viewed by a single camera) in dual-space. The viewed quadric, \( Q \), transforms to another quadric, \( Q^* \), and the camera centre transforms to a plane in dual-space.

More interesting, however, is the dual to the back-projected apparent contour (a cone, \( Q_o \), in real-space). The tangent plane at any point on this cone must pass through the camera centre (the apex of the cone): in dual-space, this means that the dual to \( Q_o \) must lie on the plane dual to the camera centre. Further, the planes tangent to \( Q_o \) are also tangent
to $Q$ and so the dual to $Q_o$ also lies entirely on $Q^*$.

The dual to the back-projected apparent contour (a rank 3 quadric) is therefore the intersection of the dual quadric and the dual camera centre: a plane conic (see figure 2.6).

Figure 2.6: The dual-space equivalent of a single camera viewing a quadric surface (figure 2.4). The dual quadric intersects with the dual camera centre (a plane) to form a conic (red) which is dual to the back-projected silhouette (a cone).

### 2.3.3 Two views

Figure 2.7 shows a quadric surface imaged from two different viewpoints. The silhouette in each image is back-projected as two cones, each tangent to the original quadric. The dual-space equivalent is shown in figure 2.8.

Once again, the cones back-projected from the image silhouettes become plane conics in dual-space. The line of intersection of the two dual image centres is dual to the base-line (i.e. the line joining the two image centres in real-space): it was seen in section 2.3.1 that the dual to the base-line, passing through both camera centres, lies at the intersection of the planes dual to each camera centre.

In dual-space, if both plane conics intersect with the dual base-line, it can be seen from figure 2.9 that the conics both share a pair of points. These points are particularly signifi-
cant, as it means that the dual-space conics are not necessarily independent.

These two points lie on the intersections of the dual conics (see figure 2.9). As they also lie on the dual quadric and the dual camera centres, they correspond to real-space planes which are tangent to the viewed quadric and pass through both camera centres: they correspond to the epipolar tangencies (see section 4.1.4 and figure 2.10). As a line can only meet a conic in zero, one or two points, this also proves that there can only ever be a maximum of two epipolar tangents given two views of a quadric surface.

2.3.4 Algebraic duality

It was shown in section 2.2.2 that the outline of an imaged quadric, \( Q \), is a conic, \( C \). However, the relationship between the conic outline and quadric, expressed in equation (2.4), is complicated, and in this section we will show that there is a straightforward linear relationship between the dual conic outline and the dual quadric [124].

A conic, \( C \), can be represented either by the locus of (homogeneous) points, \( x \), on the curve,

\[
x^\top C x = 0,
\]
only by its envelope of lines (see section 1.5.2), \( I \),

\[
I^\top C^* I = 0.
\]

Back-projecting the lines to planes, using the camera projection matrix, gives \( \Pi = P^\top I \). The planes are tangent to the point quadric and so satisfy equation (1.10),

\[
I^\top P Q^* P^\top I = 0.
\]

Since this is true for all \( I \), it follows that

\[
C^* = P Q^* P^\top.
\] (2.7)

Equation (2.7) expresses a simple relationship between a dual quadric and its dual conic outline, using the projection matrix to parameterise the camera position and orientation [124].
Figure 2.7: A quadric surface imaged from two viewpoints. The image apparent contour back-projects to two cones.

Figure 2.8: The dual-space equivalent to figure 2.7. The planes represent the duals to the two camera centres. The line of intersection of these two planes represents the line joining the camera centres in real-space (the base-line).
Figure 2.9: A quadric surface imaged from two viewpoints in dual-space. The plane conics (red) represent the back-projected image silhouettes and the dual points (green) represent a pair of real-space planes which are tangent to the quadric whilst passing through the two camera centres (forming epipolar tangencies). The dual quadric is removed (right) to aid visualization.

Figure 2.10: Imaged from two viewpoints, there are two planes which touch the surface of a quadric and pass through both camera centres. These planes form epipolar tangencies in the image. The planes are dual to the points in dual-space depicted in figure 2.9.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Real-space Description</th>
<th>Dual-space Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Proper quadric, $Q$</td>
<td>Dual quadric, $Q^*$</td>
</tr>
<tr>
<td>3</td>
<td>Proper quadric cone, $Q$</td>
<td>Plane conic</td>
</tr>
<tr>
<td></td>
<td>Plane conic</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Pair of distinct planes, $Q$</td>
<td>Pair of distinct points</td>
</tr>
<tr>
<td>1</td>
<td>Repeated planes, $Q$</td>
<td>Pair of repeated points</td>
</tr>
</tbody>
</table>

Table 2.2: Summary of proper and degenerate quadric surfaces and their tangent plane dual-space representation.

**Degeneracy**

It is important to note that in contrast to equation (2.4), equation (2.7) only holds for quadrics of rank 4 (i.e. proper quadrics such as ellipsoids and hyperboloids). If the rank of $Q$ is less than 3, appendix A shows that $Q^*$ is zero, and hence equation (2.7) collapses. If the rank is 3, the quadric envelope is not so easily defined.

Further, it can be seen from equation (2.7) that a full rank quadric (having a full rank envelope) projects to a full rank conic (also having a full rank envelope). In reverse, this shows that a degenerate quadric projects to give a degenerate conic silhouette.

Table 2.2 summarises the transformations between real and dual space for various proper and degenerate quadrics.

### 2.4 Quadric Reconstruction

The following section concentrates on reconstructing a quadric surface from image information alone. That is, given a set of image constraints, the quadric parameter matrix, $Q$, is computed. These image constraints fall into two categories: point constraints and tangency constraints.

**Point constraints.** Two image points, corresponding to a single 3-D point, $X$, can be back-projected to a pair of rays intersecting in $X$. If this point is known to lie on $Q$, it is clear that this pair of point correspondences provide a constraint on $Q$.

It will be seen (section 2.4.2) that a single point constraint imposes one linear constraint
on $Q$, and hence 9 constraints are needed to unambiguously reconstruct the quadric.

**Silhouette constraints.** The (conic) silhouette of a quadric surface back-projects to a cone in space. This cone is tangent to the quadric along the contour generator, and also imposes constraints on $Q$ (although it has been seen that these constraints are more simple if imposed on the dual-space quadric $Q^*$).

Section 2.4.1 shows that a single silhouette, together with 4 point correspondences can reconstruct the quadric surface up to a two-fold ambiguity. Thus, the silhouette is equivalent to 5 point constraints. It is then shown (section 2.4.2) that by performing the reconstruction in the dual-space, two silhouettes impose 8 constraints on the dual quadric (giving a single parameter family of solutions). Finally, section 2.4.3 demonstrates that a third silhouette provides one final constraint (the number of constraints is now nine) on the dual quadric, permitting it to be fully reconstructed.

It should be noted that all implementational details are left to chapter 3.

### 2.4.1 Reconstruction from one view

The obvious first question in reconstruction is: what constraints does a single view of quadric impose on the parameter matrix?

**Point constraint**

With a single view of a quadric surface, and no further *a priori* information, very little information can be gained from single images of quadric surface points. Equation (2.2), for example, requires that $\lambda$ (i.e. the “depth” of the 3-D point $X$ from the image point $x$) is known.

If, however, the depth of nine image points on the quadric surface is known, then $Q$ can be found.
Silhouette constraint

On the other hand, both equations (2.4) and (2.7) relate the quadric parameters to the silhouette in a single image, assuming the camera projection matrix, P, is known. In the case of equation (2.4) the conic is related directly to Q, whilst in the case of equation (2.7), the dual conic is related to the dual quadric Q*.

The two approaches are considered separately and contrasted.

Silhouette constraint on quadric  As pointed out in the special case above (section 2.2.2), by appropriate choice of projective frame, it is possible to transform the projection matrix of one camera to the form P = [I 0]. Hence, equation (2.4) simplifies to equation (2.5). Making the substitution [106]

$$q_{33} = \frac{qq^\top - C}{q}$$

into equation (1.8) gives

$$\begin{bmatrix} X^\top_3 & X \end{bmatrix} \begin{bmatrix} qq^\top & qq^\top - C & q^2 q \end{bmatrix} \begin{bmatrix} X_3 \\ X \end{bmatrix} = 0,$$

which expands to

$$\left( X^\top_3 q + qX \right)^2 = X^\top_3 C X_3,$$

or, if X is finite and, from equation (2.1), $X = \lambda x$,

$$\left( \lambda x^\top q + q \right) = \pm \sqrt{\lambda x^\top C x}.$$

This is a single equation, with the four (five up to scale) unknown quadric parameters q and q. Four such equations (with four different pairs $(x, \lambda)$, or X from equation (2.1)) would give a solution for Q. Note that there is an inherent two-fold ambiguity $(q, q) \leftrightarrow (-q, -q)$ in equation (2.8) and any reconstruction would therefore have a two-fold ambiguity (more on this topic later).

This solution is overly complicated and not easy to extend: hence the application of the dual-space reconstruction. More significantly, it is very difficult to remove the dependence on point constraints, $(x, \lambda)$, which requires depth information to be known.
Silhouette constraint of dual quadric   In contrast to the previous section, the reconstruction in this section is applied in the dual-space. It has been seen that fitting a quadric to a back-projected image conic (a cone) in real-space is equivalent to fitting a (dual) quadric to a plane conic in dual-space (figure 2.6).

Equation (2.7) can be rewritten as

\[
\begin{bmatrix}
C^*_{11} \\
C^*_{12} \\
C^*_{13} \\
C^*_{22} \\
C^*_{23} \\
C^*_{33}
\end{bmatrix} = P_{6 \times 10} \begin{bmatrix}
Q^*_{11} \\
Q^*_{12} \\
Q^*_{13} \\
Q^*_{14} \\
Q^*_{22} \\
Q^*_{23} \\
Q^*_{24} \\
Q^*_{33} \\
Q^*_{34} \\
Q^*_{44}
\end{bmatrix}, \tag{2.9}
\]

where \(C^*_{ij}\) and \(Q^*_{ij}\) are the \((i, j)\)th elements of \(C^*\) and \(Q^*\) respectively, and \(P_{6 \times 10}\) is a \(6 \times 10\) matrix as a quadratic function of the elements of \(P\):

\[
P_{6 \times 10}^T = \begin{bmatrix}
P_{11}P_{11} & P_{11}P_{12} & P_{11}P_{13} & P_{11}P_{14} & P_{11}P_{15} & P_{11}P_{16} & P_{11}P_{17} & P_{11}P_{18} & P_{11}P_{19} \\
P_{12}P_{12} & P_{12}P_{13} & P_{12}P_{14} & P_{12}P_{15} & P_{12}P_{16} & P_{12}P_{17} & P_{12}P_{18} & P_{12}P_{19} \\
P_{13}P_{13} & P_{13}P_{14} & P_{13}P_{15} & P_{13}P_{16} & P_{13}P_{17} & P_{13}P_{18} & P_{13}P_{19} \\
P_{14}P_{14} & P_{14}P_{15} & P_{14}P_{16} & P_{14}P_{17} & P_{14}P_{18} & P_{14}P_{19} \\
P_{15}P_{15} & P_{15}P_{16} & P_{15}P_{17} & P_{15}P_{18} & P_{15}P_{19} \\
P_{16}P_{16} & P_{16}P_{17} & P_{16}P_{18} & P_{16}P_{19} \\
P_{17}P_{17} & P_{17}P_{18} & P_{17}P_{19} \\
P_{18}P_{18} & P_{18}P_{19} \\
P_{19}P_{19}
\end{bmatrix}.
\]

The matrix equation (2.9) provides six linear equations relating the elements of \(C^*\) to the elements \(Q^*\). The rows of \(P_{6 \times 10}\) are linearly independent (and thus \(P_{6 \times 10}\) has rank 6) whilst the (homogeneous) equation is only correct up to scale: given \(C^*\), equation (2.9) imposes five independent linear constraints on \(Q^*\). The solution to equation (2.9) is therefore a 4-parameter family of dual quadrics.

### 2.4.2 Reconstruction from two views

#### Point constraint

Consider two images, \(x_1\) and \(x_2\), of a point, \(X\), on the surface of a quadric. Two rays in the form of equation (2.1) can be back-projected and intersected to reconstruct \(X\). For a given
point on the locus of a quadric surface, \( \mathbf{X} = (x, y, z, 1)^\top \), equation (1.7) can be rewritten in the form

\[
\mathcal{X} \mathbf{Q}_v = 0 ,
\]

(2.10)

where

\[
\mathcal{X} = \begin{bmatrix}
x^2 & 2xy & 2xz & 2x & y^2 & 2yz & 2y & z^2 & 2z & 1
\end{bmatrix} ,
\]

and \( \mathbf{Q}_v \) is the unknown \( \mathbf{Q} \) in vector form:

\[
\mathbf{Q}_v^{\top} = \begin{bmatrix}
Q_{11} & Q_{12} & Q_{13} & Q_{14} & Q_{22} & Q_{23} & Q_{24} & Q_{33} & Q_{34} & Q_{44}
\end{bmatrix} .
\]

Solving equation (2.10) gives a 9-parameter family as the null-space of \( \mathcal{X} \). Nine such equations (for 3-D points in general position) can be solved simultaneously to give a single solution for \( \mathbf{Q}_v \) (and hence \( \mathbf{Q} \)) as the null-space of a general \( 9 \times 10 \) matrix (a stacked set of equations in the form of equation 2.10).

**Silhouette constraint**

The previous section showed how a single image silhouette can be used to impose constraints on the reconstruction of a point quadric. However, this approach (as suggested by Shashua and Toelg [106]) is very difficult to extend to multiple images.

In contrast, the linear relationship between the (dual) image silhouette and the dual quadric is naturally extended to two or more images.

Given a single image silhouette, equation (2.9) has been shown to impose 5 constraints on the dual quadric (leaving a 4-parameter family of solutions). A second silhouette, from a different viewpoint, would also impose 5 constraints on the dual quadric. It might be expected that the 10 constraints would fully constraint the dual quadric (having 9 degrees of freedom): in practice, 2 of these constraints are not independent, leaving 8 constraints (and a 1-parameter family of dual quadrics).

Figure 2.9 depicts this reconstruction scenario in the dual-space: reconstructing a quadric given two silhouettes from different viewpoints is equivalent (in dual-space) to fitting a
Reconstructing Quadric Surfaces — Theory

quadric to two plane conics. If the conics did not intersect, they would impose a full $5 \times 2 = 10$ constraints on the quadric, and the reconstruction would be over-constrained.

However, if one of the conics intersects the dual base-line in two points, the second conic is also constrained to passing through these two points: in dual-space, this is the equivalent to the epipolar tangency constraint (see section 2.3.3). Both conics therefore only impose a total of $(5 \times 2) - 2 = 8$ constraints on the quadric surface.

If the conics intersect at a single point (there exists only a single epipolar tangent plane to the surface), then the two dual-space conics must pass through this point, and be tangent to the dual base-line: once again, imposing two constraints on the two conics and hence leaving only 8 degrees of freedom in total. In this case, the dual-space quadric is tangent to the dual base-line—the real-space reconstructed quadric passes through the base-line.

In summary, two image silhouettes of a quadric surface impose:

- in general 10 constraints on a quadric surface;

- in practice, only 8 constraints are independent if there exists one or two epipolar tangencies between the two views of the quadric surface.

2.4.3 Reconstruction from three views

Silhouette constraint

Adding a third silhouette view of the quadric surface adds another set of constraints in the form of equation (2.7). In the dual-space, this is equivalent to fitting a (dual) quadric to three conics. Each conic imposes 5 constraints on the quadric reconstruction, giving a total of 15 constraints.

However, there are two epipolar tangents between images 1 and 2 (as explained above), images 1 and 3, and images 2 and 3: a total of 6 epipolar tangents. Thus in the general case, three silhouettes only impose $15 - 6 = 9$ constraints on the quadric reconstruction (it is fully constrained, but not over-constrained).

Consider a fourth image silhouette. Each of the other three images provide two epipolar tangents on this new silhouette, and hence the silhouette is fully constrained from the
<table>
<thead>
<tr>
<th>Image feature</th>
<th>Number of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point correspondence pair</td>
<td>1</td>
</tr>
<tr>
<td>Outline in 1 image</td>
<td>5</td>
</tr>
<tr>
<td>Outline in 2 images</td>
<td>8</td>
</tr>
<tr>
<td>Outline in 3 images</td>
<td>9</td>
</tr>
<tr>
<td>Outline in (n &gt; 3) images</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 2.3: **Image-based quadric reconstruction constraints.** The number of constraints imposed on a quadric from different image features.

epipolar tangencies alone. Thus, a fourth image silhouette provides no further information about a quadric surface.

**Linear motion** The case of linear camera motion is special as can be seen in the dual-space (figure 2.11). All three camera centres lie on the same base-line, and therefore the third camera centre will also lie on an epipolar tangent plane between images 1 and 2. There will only be 2 independent epipolar tangents over the three images, and hence a total of \(15 - 2 = 13\) constraints on the quadric surface: the quadric surface is fully constrained.

![Linear camera motion](image.png)

Figure 2.11: **Linear camera motion.** If the camera motion is linear, there are only two epipolar tangent planes (green points in this dual-space representation) and hence three image silhouettes impose a total of 13 constraints (see text).
2.4.4 Degeneracy constraint

Degenerate quadrics (cylinders in particular) are very common in the man-made world, and it is quite often that a viewed quadric is known to be degenerate a priori. Enforcing this degeneracy requirement permits the quadric to be reconstructed both robustly, and from less image information.

As seen in section 1.5.2, a degenerate quadric is characterized by the singularity of the parameter matrix $Q$. A degenerate quadric has 8 degrees of freedom. Therefore a degenerate quadric can be completely reconstructed from 8 point correspondences.

Two image silhouettes (being degenerate image conics as shown in section 2.3.4) each impose 4 constraints on the quadric, whilst sharing one epipolar tangent (see below) and hence impose $2 \times 4 - 1 = 7$ constraints on the quadric surface (a one parameter family of degenerate quadrics). Accordingly, three image silhouettes impose 9 constraints and completely (over-) constraint the degenerate quadrics.

Reconstruction from point constraints

Reconstruction from point constraints needs little further introduction than the algorithm and implementation details given in section 3.2.1. Eight points give a one-parameter family of quadrics, imposing the constraints as in equation (2.10) and more completely in section 3.1.1. Imposing the degeneracy removes this ambiguity to give a finite number of solutions.

Reconstruction from silhouettes

Reconstructing a degenerate quadric from image silhouettes is more subtle as the surface cannot be represented as a quadric in dual-space. However, the quadric can be represented as a plane conic (see table 2.2).

The silhouette of a degenerate quadric is a degenerate conic: a pair of lines (distinct or coincident). The back-projected conic gives a pair of planes passing through the camera centre (see figure 2.12): a pair of points in dual-space, lying on the plane dual to the camera centre.
2 Reconstructing Quadric Surfaces — Theory

As the image conics are degenerate, there is a point singularity in each image (the point of intersection of the pair of lines). These singularities back-project to rays intersecting in the quadric singularity (the conic apex). For each camera pair, a plane, containing these two rays, is formed by this “epipolar tangency” (not strictly an epipolar tangent, as the quadric surface tangent is not defined at the singularity), passing through the two camera centres and the cone apex. In the dual-space, the two rays are visualized as lines passing through the dual back-projected planes. The epipolar tangent plane is represented as a point, at the intersection of these two dual lines. Figure 2.13 shows this construction for two images.

As a result, for each camera pair, one constraint is lost on the two image conics: two image conics impose only \(2 \times 4 - 1 = 7\) constraints on the degenerate quadric. Three image conics impose \(3 \times 4 - 2 = 10\) constraints on the degenerate quadric.

2.5 Conic Reconstruction

Conics are widely accepted as one of the most fundamental image features together with points and line segments. For this reason, conic reconstruction has certainly received previous interest [90, 91, 95].

In this section, it is shown that whilst previous quadric reconstruction techniques are not able to generalise to include plane conic reconstruction (and hence the need for specialised papers such as those cited above), the approach outlined in section 2.3 permits conic reconstruction to be considered simply as a special case of quadric surface reconstruction. This is possible by noting that plane conics cannot be represented as a point quadric, but can be represented as a dual quadric.

There is therefore little difference between reconstructing a full rank quadric and a plane conic in dual-space. This section briefly outlines the geometry of conic reconstruction. It will be assumed that no information about the conic is known: both the plane (3 parameters) and the conic (5 parameters) are unknown leaving a full 8 degrees of freedom for a general plane conic.
2 Reconstructing Quadric Surfaces — Theory

Figure 2.12: **Degenerate quadric reconstruction.** The silhouette of a degenerate quadric is a pair of lines in each image. These back-project to planes in 3-D, tangent to the quadric.

Figure 2.13: **Dual-space degenerate quadric reconstruction.** Dual-space representation of figure 2.12. The degenerate quadric to be reconstructed is shown in red, whilst the silhouettes are represented by dual-space points (green). The lines (yellow) are dual to the rays back-projected from the silhouette singularities, intersection at the point (blue) dual to the epipolar plane passing through the quadric singularity (see text).
2.5.1 Conic representation in dual-space

Consider a plane conic, \( C_p \) (note that \( C_p \) in this case, does not represent a parameter matrix and is only used for convenience), as the intersection of a 3-D plane, \( \gamma \), and a cone, \( Q_c \). It has already been seen that the dual to \( Q_c \) is a plane conic (the plane, \( \gamma \), being dual to the apex of the cone); the dual to \( \gamma \) is a point. The dual to \( C_p \) is therefore given as the cone passing through the (conic) dual to \( Q_c \) and the (point) dual to \( \gamma \).

2.5.2 Conic reconstruction in dual-space

Reconstruction of the conic proceeds in the usual manner of section 2.4.2. Two “silhouettes” (in this case, the term “outline” is more appropriate) back-project to two cones in real-space (conics in dual-space). The reconstructed dual-space cone, \( Q_c^* \), passes through each of these conics (see figure 2.14). As seen above, two back-projected silhouettes provide 8 constraints on a quadric and hence fully constrain the cone in dual-space (being a degenerate quadric with 8 degrees of freedom).

![Dual-space conic reconstruction](image)

Figure 2.14: **Dual-space conic reconstruction.** In the dual space, a plane conic is represented by a cone. The conics (*red*) represent the outline of the conic back-projected from two views. Two views are enough to fully constraint the conic, and reconstruct it up to a 2-fold ambiguity (see section 3.5).
2.5.3 Dual-space cone to real-space conic

It has been shown that in the dual-space, the reconstructed cone has its apex at the dual point corresponding to \( \gamma \) in real-space. Hence, \( \gamma \) is found as the null-space of \( Q_c^* \).

Extracting an equation for the plane conic is not as trivial. In order to define the conic, it is necessary to choose a coordinate system on the plane, \( \gamma \). [55, Section 2.2.1, page 48] shows that a coordinate system on the plane can be defined by

\[
X = Mx .
\] (2.11)

This transforms a 2-D point, \( x \), on \( \gamma \) to the 3-D point \( X \). \( M \) is a \( 4 \times 3 \) matrix spanning the null-space of \( \gamma^T \). In this coordinate system, the conic is defined by \( x^T C x = 0 \) or as an envelope of lines

\[
I^T C^* I = 0 .
\] (2.12)

The dual-space cone is also defined by the envelope of tangent planes

\[
\Pi^T Q_c^* \Pi = 0 .
\] (2.13)

For a point, \( X \), on \( \Pi \): \( \Pi^T X = 0 \). The line of intersection, \( I \), of \( \gamma \) (the fixed support plane) and \( \Pi \) (as it varies), in the coordinate system of equation (2.11), follows by trivial substitution:

\[
\Pi^T Mx = 0 .
\]

From \( I^T x = 0 \), it is clear that \( I = M^T \Pi \).

The conic envelope is defined both in the plane by equation (2.12) and in dual-space by equation (2.13). Substituting for \( I \) in equation (2.12) gives

\[
\Pi^T M C^* M^T \Pi = 0 ,
\]

and hence

\[
Q_c^* = M C^* M^T .
\]

\( Q_c^* \) is the reconstructed dual-space cone, and known: \( C \) can be found given the left-inverse for \( M \) (the pseudo-inverse is used as \( M \) is a \( 4 \times 3 \) matrix):

\[
C^* = M^{-1} Q_c^* M^{-T} .
\]
2.6 Conclusions

This chapter has outlined the basic geometry of imaged quadric surfaces. The implementational details have been left for the next chapter, and examples of all the methods proposed are given.

In particular, it has been shown that reconstruction in the tangent dual-space permits the conic silhouette of a quadric to be used as a set of constraints on the parameters of the quadric. Other constraints considered are points and degeneracy.

Table 2.3 summarises the important results of the chapter, and most significantly it should be noted that, in general, whilst a single conic silhouette imposes 5 constraints on the quadric, a second silhouette only imposes 8 constraints: two constraints are not independent due to the existence of two epipolar tangents. Further, assuming the quadric to be degenerate reduces the reconstruction to just 8 degrees of freedom.

Finally, whilst previous reconstruction techniques (based on reconstruction in real-space) cannot be generalised to the reconstruction of a plane conic (indeed, in real-space a plane conic can not be represented as a quadric surface), the dual-space approach has been shown to naturally solve this problem.
Chapter 3

Reconstructing Quadric Surfaces — Implementation

Chapter 2 motivated and introduced the concept of quadric surface reconstruction. In particular, it was seen that a quadric surface can be reconstructed both from a set of image silhouettes or from point correspondences over two or more views.

However, little attention was paid to the important implementational details of surface reconstruction.

Section 3.1 considers the implementation of general quadric reconstruction from either points (section 3.1.1) or from image silhouettes (section 3.1.2). The combination of both point and silhouette constraints is studied in section 3.1.3. In contrast to chapter 2, specific consideration is given to the exact ambiguities of the reconstruction. Reconstruction from image silhouettes requires the silhouettes to correspond (i.e. consistent with the epipolar geometry): an optimal approach to conic correction is presented in section 3.1.2.

Section 3.2 then specializes the reconstruction problem, and considers the reconstruction of degenerate quadrics. Sections 3.2.1 and 3.2.2 discuss the reconstruction of degenerate quadrics from point constraints and silhouette constraints respectively.

Whilst sections 3.1 and 3.2 give closed-form solutions for the quadric surface reconstruction, they are not statistically optimal. Section 3.3 assumes all image data to be noisy, and shows how the closed-form solutions can be used as starting points in a more general non-linear, but “optimal”, reconstruction algorithm. Results of the reconstructions are given throughout.

Section 3.4 shows that quadric induced transfer can be used to register, to sub-pixel accuracy, multiple images of a quadric surface.
Finally, it is shown in section 3.5 that a conic can be accurately reconstructed from 2 views, and an example is given.

Once again, throughout the chapter, it is assumed that the epipolar geometry, and camera projection matrices are known.

### 3.1 General Quadric Reconstruction

#### 3.1.1 Reconstruction from points

Equation 2.10 shows how a point in 3-D imposes a single linear constraint on the quadric parameter matrix $Q$. Given $n$ (where $n \geq 9$) such equations, with $n$ 3-D points on the quadric surface, a unique solution for $Q$ can be found.

For each 3-D point, $X_i = (x_i, y_i, z_i, 1)\top$, the row-vector $X_i$ is built as

$$X_i = \begin{bmatrix} x_i^2 & 2x_iy_i & 2x_iz_i & 2x_i & y_i^2 & 2y_iz_i & 2y_i & z_i^2 & 2z_i & 1 \end{bmatrix}. $$

The quadric parameter vector, $Q_v$, is then the null-space of the stacked $n \times 10$ matrix

$$M = \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_n \end{bmatrix}. $$

In general (see below) for $n = 9$, there will be a single solution for $Q_v$ (up to scale, but noting that the matrix $Q$ is also only defined up to scale): $M$ is a $9 \times 10$ matrix. For $n > 9$, $M$ is a $n \times 10$ matrix and is hence full rank (i.e. rank 10): in this case, the least-squares null-vector is chosen\(^1\).

It is assumed that the 3-D points, $X_i$, are “in general position”. If not (for example, if any three points lie on a line), the corresponding row-vectors, $X_i$, will not be linearly independent, and $M$ will drop rank accordingly. In the absence of noise, and for $n > 9$, this is the natural case: all $X_i$ lie on a quadric, and hence $M$ will be of rank 9.

\(^1\)The vector corresponding to the smallest singular value.
3.1.2 Reconstruction from silhouettes

It has been shown (section 2.4.3) that a quadric surface can be unambiguously reconstructed from three silhouettes under general motion\(^2\). Section 3.1.3 will consider the problem of reconstructing a quadric from one or two silhouettes.

The reconstruction of a quadric surface from three image silhouettes is best performed in the dual-space, as equation (2.7) provides a linear relationship between this dual-space quadric and the silhouette. A more general solution, permitting quadric surface reconstruction from \(n\) silhouettes (where \(n \geq 3\)) is provided, noting that if \(n > 3\) under general motion, and for all \(n\) under linear motion, the solution is over-constrained.

The image silhouette for the \(i\)th image, with camera projection matrix, \(P_i\), is represented as a conic, \(C_i\) (see section 2.2.2). Each silhouette then imposes a constraint in the form of equation (2.9):

\[
C_{i,v}^* = P_i \cdot 6 \times 10 Q_v^* ,
\]

where the \(v\) subscript represents the vectorized forms of \(C_i^*\) and \(Q^*\), and \(P_i \cdot 6 \times 10\) is defined in equation (2.9) as a matrix made up of quadratic functions of the elements of \(P_i\). In order to combine \(n\) such equations, it is necessary to introduce a scalar, \(\alpha_i\), for each equation as the equality is only up to an unknown scale. The equation set (3.2) then can be rewritten as

\[
\begin{bmatrix}
P_{1,6 \times 10} & C_{1,v}^* & 0 & \cdots & 0 \\
P_{2,6 \times 10} & 0 & C_{2,v}^* & 0 \\
\vdots & 0 & 0 & \ddots & \vdots \\
P_{n,6 \times 10} & 0 & 0 & \cdots & C_{n,v}^*
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_n
\end{bmatrix} = 0 .
\]

For \(n \geq 3\), \(Q_v^*\) is computed from the least-squares null-vector from the singular value decomposition of the left-hand side.

**Conic correspondence**

It is important to note that equation (3.3) is only valid if the set of image conics correspond exactly. In other words, they must be consistent with the epipolar geometry. For example,

\(^2\)In this case, *general motion* only requires that no two camera centers are coincident: linear motion is permitted; rotation about the camera centre is not permitted.
consider two views of a quadric surface; given the conic in the first image, the conic in the second image is constrained by the epipolar tangents from this first conic: it has only $5 - 2 = 3$ degrees of freedom.

In general, the image silhouettes are found by fitting conics to image data (edgels), and do not necessarily correspond (due to image noise—see section 3.3). An algorithm to “optimally” correct a set of inconsistent conics is presented here. Figures 3.1 and 3.2 show examples of the image geometry.

The objective is to find the best epipolar consistent conics, $C_i$ (for $i = 1...m$), in each image, fitted to the image data $s_{i,j}$ (where $j$ indexes the image points in the $i$th image image):

$$
\min_{C_k} \sum_{i=0}^{m} \sum_{j} D(C_i, s_{i,j}) ,
$$

(3.4)

for all $k$, subject to the conics $C_k$ remaining consistent with the epipolar geometry. The minimization in equation (3.4) assumes that all image conics are consistent, whilst optimizing the fit to the image data, $s_{i,j}$: this is difficult to implement in this form, but a more manageable form is now derived.

Each epipolar tangent imposes one (tangency) constraint on the corresponding conics, and there are in general $2(\sum_{i=0}^{m} i)$ epipolar tangents, given $m$ images. Each epipolar tangent can therefore be parameterised by a single value if the epipoles are known and fixed (for instance, the angle of the epipolar line in one image; the angle of the corresponding epipolar line in the second image being given by the epipolar geometry). The minimization equation (3.4) can therefore, without loss of generality, be written as

$$
\min_{v} \sum_{i=0}^{m} \sum_{j} D(C'_i(v, s_{i,j}), s_{i,j}) ,
$$

(3.5)

where the epipolar tangent parameters are represented by $v$, and the conic $C'_i(v, s_{i,j})$ is fitted first to the epipolar tangent lines, and then to the image points $s_{i,j}$. This is measuring the same quantity (i.e. the accuracy of fit of the conics to the image data) as equation (3.4), whilst internally ensuring that the conics are always consistent with the epipolar geometry.

The specific cases of 2- and 3-view conic correction are outlined with reference to equation (3.5), and in particular, with reference to the computation of $C'_i(v, s_{i,j})$: 
Figure 3.1: Top: A pair of corresponding conics. The corresponding epipolar lines (red) are given by the epipolar tangencies in 3-D and must be tangent to the conic in the image. Bottom: A pair of conics which do not correspond. The tangent epipolar lines in the first image (red) are not tangent to the conic in the second image. Similarly, the tangent epipolar lines in the second image (blue) are not tangent to the conic in the first image.

Figure 3.2: Three corresponding conics. The corresponding epipolar lines between the images 1 and 2 (red), 1 and 3 (blue) and 2 and 3 (green) are all tangent to the image conics.
• **Three images.** There are 6 epipolar tangents in total between two views of a quadric surface (see figure 3.2); in each image there are 4 epipolar tangent lines to which the conic must be tangent. The procedure for computing \( \mathcal{C}_i'(v, s_{i,j}) \) is as follows:

- Given \( v \), and hence the four epipolar tangent lines in the \( i \)th image, fit a dual conic to these lines, giving a pencil of dual conics: \( \mathcal{C}_i^* = \alpha_1 \mathcal{C}_1^* + \alpha_2 \mathcal{C}_2^*; \)
- Solve for \( \alpha \) and \( \beta \) by finding the best-fit conic to the image data, \( s_{i,j} \):

\[
\min_{\alpha_1, \alpha_2} \sum_j D(\mathcal{C}_i, s_{i,j}) .
\]

A Levenberg-Marquardt minimization is used;

- Compute \( \mathcal{C}_i = (\mathcal{C}_i^*)^* \).

• **Two images.** There are 2 epipolar tangents in total, and hence two in each view. Computation of \( \mathcal{C}_i'(v, s_{i,j}) \) is similar to the three view case:

- Given \( v \), fit a dual conic to the two epipolar lines, giving a 3-parameter family of dual conics: \( \mathcal{C}_i^* = \alpha_1 \mathcal{C}_1^* + \alpha_2 \mathcal{C}_2^* + \alpha_3 \mathcal{C}_3^* + \alpha_4 \mathcal{C}_4^*; \)
- Solve for \( \alpha_1, \alpha_2, \alpha_3 \) and \( \alpha_4 \) as above for the 3-view case;
- Compute \( \mathcal{C}_i = \mathcal{C}_i^* \).

**Conic correction algorithm**

This optimal conic correction algorithm is summarised as follows:

• Fit (using a Bookstein conic fitting algorithm [13]) conics to the image data, \( s_{i,j} \), in each image;

- Use these conics as a starting position for a Levenberg-Marquardt minimization over \( v \) of equation (3.5). At each step of the optimisation, \( v \) gives \( \mathcal{C}_i'(v, s_{i,j}); \)

- Given \( v \), compute the final conics, \( \mathcal{C}_i = \mathcal{C}_i'(v, s_{i,j}) \), as above for each image.

A synthetic example of the algorithm is given in figure 3.3.
Figure 3.3: (a) Three corresponding conics (green). The epipolar tangencies between images 1 and 2 (magenta), images 1 and 3 (cyan) and images 2 and 3 (black) are shown. (b) Assuming the conics to be fitted to a set of image features (blue) with unknown gaussian noise (in this case as much as \( \sigma = 16 \) pixels to exaggerate the errors) gives a set of conics (red) which no longer correspond: this can be seen from the lack of correspondence between the tangent lines (these are no longer epipolar tangencies). (c) Minimising the geometric distance between the image features and the set of conics whilst imposing the epipolar tangency constraints gives a new set of corresponding conics (yellow) which are close to the original conics (green).
3.1.3 Reconstruction from points and silhouettes

1 silhouette and 4 or more points

It was seen in section 2.4.1 that a single conic silhouette imposes 5 constraints on a general quadric surface. Thus a further 4 constraints are needed in general to fully reconstruct the quadric surface.

Transforming the system into a projective frame such that one of the projection matrices can be expressed as $P_1 = [I \ 0]$ allows the results of section (2.8) to be applied: this permits a quadric to be reconstructed given a silhouette and 4 or more points on the quadric surface (once again, computed by back-projecting point correspondences over 2 or more views). It was seen that one silhouette and 4 points gives a reconstruction up to a two-fold ambiguity. A further point reduces this ambiguity.

2 silhouettes and 1 point

Two image silhouettes back-project to two cones in space. Each cone imposes 5 tangency constraints on the quadric surface, equivalent to 5 point constraints in the dual-space. However, it has been shown (section 2.3) that two of these constraints are not independent and thus two silhouettes impose only 8 constraints on a quadric surface.

Once again, reconstruction in the dual-space permits a linear relationship to be written between the two (dual) silhouettes, $C_1^*$ and $C_2^*$, and the dual-space quadric $Q^*$. Following a similar argument to that given in section 2.4.1 above, two equations of the form of equation (2.9) can be written as

$$
\begin{bmatrix}
P_{1,6x10} & C_{1,v}^* & 0 \\
P_{2,6x10} & 0 & C_{2,v}^*
\end{bmatrix}
\begin{bmatrix}
Q_v^* \\
\alpha_1 \\
\alpha_2
\end{bmatrix} = 0.
$$

This time, as there are only 8 independent constraints, the null-space gives a one parameter family of solutions for $Q_v^*$ and hence the dual-quadric can be written as:

$$
Q^* = \beta Q_1^* + \gamma Q_2^*
$$

where $Q_1^*$ and $Q_2^*$ span this null-space, and $\beta$ and $\gamma$ are arbitrary scalars. Assuming $Q^*$ to be full rank (degenerate quadric reconstruction is considered in section 3.2), the adjoint of
both sides can be taken to give

\[ Q = \beta^3 Q_1 + \beta^2 \gamma Q_2 + \beta \gamma^2 Q_3 + \gamma^3 Q_4 \]  

(3.6)

where \( Q_1, Q_2, Q_3 \) and \( Q_4 \) are constant parameter 4 \times 4 matrices dependent on \( Q_1^* \) and \( Q_2^* \) alone. As \( Q \) is only defined up to scale, this is a cubic equation in either \( \beta \) or \( \gamma \).

Imposing a point constraint by substituting equation (3.6) into equation (1.7) gives a simple cubic equation in the form,

\[ \beta^3 x^\top Q_1 x + \beta^2 \gamma x^\top Q_2 x + \beta \gamma^2 x^\top Q_3 x + \gamma^3 x^\top Q_4 x = 0 \]

in turn giving three solutions for \((\beta, \gamma)\) and hence \(Q\). However, of these three solutions, two are equal up to scale: the solutions \((\beta, \gamma)\) and \((-\beta, -\gamma)\) result in the same reconstructed quadric. There are thus only two independent solutions.

A further point constraint removes this ambiguity.

### 3.2 Degenerate Quadric Reconstruction

As seen in section 1.5.2, a degenerate quadric only has 8 degrees of freedom. Further, it is often the case that a viewed quadric is known to be degenerate (either a priori, or due to the degeneracy of the silhouette as noted in section 2.3.4). It will be assumed that a rank 3 quadric is being reconstructed: rank 1 or 2 quadrics are of little interest as they can be represented as 1 or 2 planes respectively.

Imposing this constraint is therefore useful, as it shows that a degenerate quadric can be reconstructed from 8 points constraints up to a two-fold ambiguity (section 3.2.1) and from three image silhouettes (section 3.2.2).

#### 3.2.1 Reconstruction from points

Given eight points in general position on a quadric surface, these eight constraints can be written in the form of equation (3.1). The null-space of \( \mathcal{M} \) is gives a one parameter family (a “line”) of solutions for \( Q_v \) and hence \( Q \) which can be written as

\[ Q = \beta Q_1 + \gamma Q_2 \]
Imposing the degeneracy constraint by forcing the singularity of \( Q \) gives a quartic equation in \((\beta, \gamma)\) (once again, \( Q \) is only defined up to an arbitrary scale factor) with coefficients \( q_1, q_2, q_3, q_4 \) and \( q_5 \) dependent on the elements of \( Q_1 \) and \( Q_2 \):

\[
q_1\beta^4 + q_2\beta^3\gamma + q_3\beta^2\gamma^2 + q_4\beta\gamma^3 + q_5\gamma^4 = 0 .
\] (3.7)

There are a maximum of four solutions to equation (3.7), in turn giving two solutions for \( Q \). A further point constraint can be used to remove this reconstruction ambiguity.

In summary, 8 point constraints permit a degenerate quadric surface to be reconstructed up to a two-fold ambiguity. A total of 9 point constraints is necessary to unambiguously reconstruct a degenerate quadric surface.

### 3.2.2 Reconstruction from silhouettes

Section 2.4.4 showed that fitting a degenerate quadric to a number of image silhouettes is equivalent to fitting a plane conic to a number of points (each silhouette providing two points). However, it was shown that whilst the first silhouette provides a full 4 constraints (in dual-space, this was seen to be two points in a plane), the second silhouette only provided 3 constraints (one general point, with the second point constrained to lie on a line and hence having only one degree of freedom). A third silhouette then provides 2 further constraints: 9 in total and enough to fully constrain the quadric surface.

A degenerate quadric surface is represented by a plane conic in the dual-space, and cannot therefore be represented by a dual-space quadric parameter matrix, \( Q^* \). As seen in section 2.4.4, degenerate quadric reconstruction is equivalent, in dual-space, to fitting a conic to a set of points.

Each image silhouette, \( C_i \) in the \( i \)th image, back-projects as a pair of planes, \( \Pi_{i,1} \) and \( \Pi_{i,2} \). These planes give the points, \( \Pi_{i,1} \) and \( \Pi_{i,2} \) in dual-space. The conic (dual to the degenerate quadric) lies in a plane dual to the apex (singularity) of the degenerate quadric: this is reconstructed by intersecting all the planes \( \Pi_{i,j} \). The conic can therefore be reconstructed by defining a coordinate system on the dual-space plane, and fitting a conic to the dual-space points (using a Bookstein conic fitter, for example). The quadric parameter ma-
3 Reconstructing Quadric Surfaces — Implementation

A tris is given by the exact reverse of section 2.5.3 where a dual-space quadric is transformed into a real-space planar conic.

3.3 Optimal Quadric Reconstruction

3.3.1 Error model

Whilst this chapter assumes the matching phase of the algorithms to be perfect, it will not be assumed that localizing corners in the image is perfect.

Corners are detected at 2-D intensity discontinuities in the image using an algorithm such as that proposed by Harris [49]. There are two main sources of error in the localization of corners: spatial sampling noise and intensity quantization noise. Other errors such as lens distortion (beyond quadratic radial lens distortion [33] which can be removed), interlacing and non-uniform blurring all add localization noise to the detected corners.

It is assumed, therefore, that the localization of an image point, x, is subject to an unknown uniform gaussian error\(^3\) (typically errors of 0.5 to 1 pixels are found). The result of this “noise” on the algorithm outlined above is critical: in practice, the back-projected rays from each of the corresponding points over a number of images do not intersect in a single point, or do not intersect at all.

3.3.2 Point constraint cost function

Consider reconstructing a quadric surface from a set of noisy point correspondences over a 2 or more views. Each correspondence set back-projects to a single point in 3-D, which in turn lies on the surface of the quadric.

A single, unknown, 3-D point, X, is imaged as \(x_i = p_iX\) in the \(i\)th of \(m\) images. The optimal reconstruction of X is given by minimizing the reprojection errors:

\[
\min_X \sum_{i=1}^{m} D(x_i, p_iX),
\]

where \(D(x_i, p_iX)\) is the euclidean distance from the imaged point \(x_i\) to the reprojected point \(p_iX\).

\(^3\)This is justified by the application of the Central Limit Theorem: it is common to assume the sum of a number of unknown and uncorrelated noise signals to be close to gaussian noise. This noise model is demonstrated to be empirically valid by Torr et al. [123].
Reconstructing a quadric surface from point correspondences follows a similar approach: minimization of the reprojection errors. This time, the unknown parameters are the \( n \) 3-D points \( \mathbf{X}_j \) (where \( j = 1, \ldots, n \)) and the quadric surface, \( \mathcal{Q} \). Clearly, \( \mathbf{X}_j \) must lie on the quadric surface. The optimal reconstruction is given by the solution of

\[
\min_{\mathcal{Q}, \mathbf{X}_j} \sum_{j=1}^{n} \sum_{i=1}^{m} D(\mathbf{x}_{i,j}, P_i \mathbf{X}_j),
\]

subject to \( \mathbf{X}_j^T \mathcal{Q} \mathbf{X}_j = 0 \). The image point \( \mathbf{x}_{i,j} \) is the projection of the point \( \mathbf{X}_i \) into the \( j \)-th image.

**Implementation.** In practice, equation (3.8) is difficult to implement. Therefore a slightly less optimal minimization is proposed.

Consider two views, \( \mathbf{x}_{j,a} \) and \( \mathbf{x}_{j,b} \) (in images \( a \) and \( b \)), of a 3-D point, \( \mathbf{X}_j \), on \( \mathcal{Q} \). A quadric-induced mapping function can be defined such that, given a perfect reconstruction and noiseless data, \( T_{a,b}(\mathbf{x}_{j,a}) = \mathbf{x}_{j,b} \). In practice, as \( \mathbf{x}_{j,a} \) and \( \mathbf{x}_{j,b} \) are noisy, the quadric is reconstructed by minimizing the errors in this transfer over each correspondence and each image:

\[
\min_{\mathcal{Q}} \sum_{j=1}^{n} \sum_{a=1}^{m} \sum_{b=1, b \neq a}^{m} D(\mathbf{x}_{j,b}, T_{a,b}(\mathbf{x}_{j,a})).
\]

### 3.3.3 Silhouette cost function

Consider \( m \) silhouettes of a quadric surface from different viewpoints. The silhouettes are detected by edge-detecting the image [19]: in the \( i \)-th image they are represented by the set of points, \( s_{i,k} \) (where \( k \) indexes the points in each image). This image data is assumed to be noisy.

The optimal reconstructed quadric minimises the distance between its reprojected silhouette in each image, and the measured image silhouettes:

\[
\min_{\mathcal{Q}} \sum_{i=1}^{m} \sum_{k} D(s_{i,k}, \mathcal{C}_i),
\]

where \( \mathcal{C}_i \) is the silhouette of \( \mathcal{Q} \) in the \( i \)-th image, and \( D(s_{i,k}, \mathcal{C}_i) \) is euclidean distance between the conic \( \mathcal{C}_i \) and the image point \( s_{i,k} \).
3.3.4 Combined point and silhouette cost function

Reconstructing a quadric from both point and and silhouette constraints is now simply a matter of combining the minimizations of equations (3.9) and (3.10):

\[
\min_{\mathcal{Q}} \sum_{j} \sum_{a} \sum_{b} D(x_{j,b}, T_{a,b}(x_{j,b})) + \sum_{i} \sum_{k} D(s_{i,k}, \mathcal{C}_{i}) ,
\]

(3.11)

As these errors are all taken within the image, and it can be assumed that the localisation of points within the image is constant, no scaling is necessary between these two cost functions.

3.3.5 Implementation

This leads to a general non-linear minimization of equation 3.11, to reconstruct the quadric surface.

General quadric reconstruction

In the case of an arbitrary quadric reconstruction, the surface has 9 degrees of freedom. The following algorithm reconstructs the quadric parameters:

- Obtain a closed-form linear solution to the surface parameters. Depending on the number and type of image information available, section 3.1 outlines how the constraints can be used to compute an estimate of the quadric parameters;

- Optimizing over all 9 parameters of \( \mathcal{Q} \) (\( \mathcal{Q} \) has 10 elements, but is only defined up to an arbitrary scale factor: one of the elements is fixed), apply a Levenberg-Marquardt minimization with equation (3.11) giving the residual.

Degenerate quadric reconstruction

If the viewed quadric surface is known to be degenerate, only 8 parameters are required. The constraint \( |\mathcal{Q}| = 0 \) provides the final parameter:

- Obtain a closed-form linear solution with reference to section 3.2;
• Again, equation (3.11) is minimized, this time with only eight elements of $Q$ variable: one element is fixed as the scale of $Q$ is arbitrary, and the final element is found from $|Q| = 0$ (strictly, this gives 2 independent solutions for $Q$ as the solutions of a quartic equation; at each step of the algorithm, the “best” (minimum cost) of the 2 solutions is taken).

### 3.3.6 Results

This section presents some examples of quadric reconstruction. It is seen that sub-pixel reprojection errors are typical, whilst the silhouette constraints naturally complement the surface constraints available from points correspondences.

**Reconstruction from point correspondences**

Figure 3.4 depicts a synthetic example of quadric reconstruction form point correspondences alone. About 50 points, sampled randomly, are taken on the surface of a unit sphere. They are then imaged in two $500 \times 500$ pixel images, and gaussian noise ($\sigma = 8$ pixels) is added to the correspondences. A linear quadric fit gives image 1 to image 2 transfer errors of about 4 pixels, whilst a non-linear quadric fit gives errors of 2 pixels. It can also be seen that the reprojected silhouettes are much improved despite this information not being used in the reconstruction. Note that the gaussian noise in this case is significantly larger than that ever expect in an image (typically, $\sigma = 0.5$ pixels).

Figure 3.5 shows a real quadric surface reconstructed from 50 point correspondences. Image 1 to image 2 transfer error falls from 0.97 pixels for a linear quadric fit, to 0.8 pixels for a non-linear quadric fit. The reprojected silhouette (again, not used in the reconstruction) has errors of about 5 pixels: this is a larger error than expected, but in this particular example most of the point constraints are towards the centre of the image making the quadric fit rather badly conditioned (the points are close to planar).

Figure 3.6 shows two views of a degenerate quadric surface, which is reconstructed from about 80 point correspondences. The transfer error is 0.8 pixels for a linear fit, and 0.5 pixels for a non-linear fit.
Figure 3.4: Two views of a synthetic sphere (top). Approximately 50 points correspondences (blue) are chosen at random and noise ($\sigma = 8$ pixels) is added. The silhouette reprojected from the reconstruction is superimposed (bottom): the linear reconstruction silhouette is shown in red, and the non-linear reconstruction silhouette is shown in yellow. The exact silhouette is shown in green.

Reconstruction from silhouettes

Figures 3.7 and 3.9 demonstrate quadric surface reconstruction from silhouettes alone (3 images in each case). As expected, the reprojected silhouette is very close to the measured image silhouette: averaging about 0.5–0.8 pixels error. Point transfer close to the silhouette is good (about 0.5 pixel error), whilst point transfer far from the silhouette increases to about 1–1.5 pixels error. This is as expect, as the surface is not as well constrained far from the silhouette. Figure 3.8 shows that transfer in a further image (not used in the reconstruction) gives approximately 1.5 pixel transfer error, and a reprojected silhouette error of about 0.9 pixels.
Figure 3.5: Two views of a ping-pong ball. Point correspondences (blue) are used to reconstruct the ball, and the silhouettes from a linear (red) reconstruction and a non-linear (yellow) reconstruction are shown.
Figure 3.6: Two views of a champagne bottle, reconstructed as a cylinder. Point correspondences are shown in blue, which are used to reconstruct the cylinder. The reprojected silhouettes from linear (red) and a non-linear (yellow) reconstructions are shown.
Figure 3.7: Three images of a ping-pong ball. The silhouettes (green) are detected with an edge-detector (edgels shown in blue). The epipolar tangents are shown between images 1 and 2 (magenta), 2 and 3 (black), and 1 and 3 (cyan). An example of the reconstructed quadric transfer is shown (bottom image triplet).

Figure 3.8: A new image of the ping-pong ball reconstructed from the sequence shown in figure 3.7. The transfer example from figure 3.7 (bottom) and reprojected silhouettes are shown.
Reconstruction from points and silhouettes

Figures 3.10 demonstrates the result of reconstructing two quadrics from both silhouettes and points. Transfer errors of about 0.5–0.8 pixels with reprojected silhouette errors of about 0.5 pixels are achieved. Figures 3.11 and 3.12 shows how a VRML model can be generated from the reconstruction, and used to create new views of the surface.

3.4 Quadric Induced Transfer

It was seen in section 2.2.3 that a quadric induces a transfer between points in a first image and points in the second image. This is very similar to the homography between two images of a plane (indeed, in the case of a degenerate rank 1 quadric, being a pair of repeated
Figure 3.10: Two cooling-towers from the Didcot power-station, reconstructed from both point and silhouette constraints. The left tower is reconstructed from two silhouettes and seven points, the right tower is reconstructed from two silhouettes and a single point. An example of transfer is shown in each case.
planes, the *quadric induced transfer* reduces to a simple homography transfer).

It is therefore possible, given one view (with known camera projection matrix) of a known quadric (reconstructed by any one of the algorithms discussed in sections 3.1 and 3.2 above) to generate any other view of the quadric, given a new projection matrix:

- For each point in the second image, \( x_2 \), back-project a ray and intersect with the quadric using equation (2.2);

- Each intersection in space (in general, there are two intersections but one can be ignored due to the *opacity constraint* as discussed in the next section), \( X \), then projects into the first image to give \( x_1 \);

- The image intensity in image 1, at \( x_1 \), is then transfered to the second image at \( x_2 \). Bilinear or bi-cubic interpolation is used for non-integer \( x_1 \).
3.4.1 Opacity constraint

It was seen in section 2.2.3 that a single point on one image transfers to zero, one or two points in a second view. In the case of a one point to two points transfer, it is necessary to “choose” one of these transferred points.

Shashua et al. [106] suggested the introduction of an opacity constraint when viewing a quadric surface. Consider a ray back-projected from image 1 hitting the quadric surface at two points, \( X_1 \) and \( X_2 \). If the quadric is assumed to be opaque, one of these points will occlude the other, and hence only one will really be visible in image 1. Thus, only this point can sensibly be transferred to image 2, and a one-to-one mapping is then defined.

One subtlety is worth mentioning: as a reconstruction from uncalibrated cameras is only possible up to a general projective transformation of space, it is necessary to consider if the opacity constraint is invariant to such transformations. Define \( c_1 \) as the camera...
centre for the first image, and noting that ordering is preserved under a projective transformation, it is necessary to consider the three possible situations:

- If the points \( c_1, X_1 \) and \( X_2 \) lie along the back-projected ray in this order, clearly \( X_2 \) is occluded in the image 1 by \( X_1 \) and is hence not visible;

- Similarly, if the points lie in the order \( c_1, X_2, X_1 \), the point \( X_2 \) is visible in image 1;

- However, if the points lie in the order \( X_1, c_1, X_2 \), there are parts of the quadric surface on either side of the \( c_1 \). As the “direction” of the ray is not known, it is not possible to determine which of the points is visible, and which lies “behind” the image plane. It can, however, be assumed that all points on the quadric surface lie on the same side of the image plane: given at least one point on the quadric surface (from a pair of point correspondences, for example), the orientation of the camera with reference to the surface can be found, and either \( X_1 \) or \( X_2 \) can be eliminated.

### 3.4.2 Example

Figure 3.13 shows an example of quadric-induced transfer. It is seen that sub-pixel registration is possible, and could be used in algorithms such as mosaicing [116, 117, 20] or super-resolution [59, 60, 20, 133].

### 3.5 Conic Reconstruction

Section 2.5 outlines the reconstruction of a plane conic viewed in two or more views. Reconstruction takes place in the dual space.

Two image conics impose 8 constraints on a dual quadric, and it has been shown that a plane conic in real-space is represented by a cone (rank 3 quadric) in dual-space: the conic can be completely reconstructed from two images.
3.13: Two images of a quadric surface (top) are used to reconstruct \( Q \), and hence define a quadric-induced transfer between the two images. The first image can then be mapped into the second image (bottom), for example. The bottom-right image demonstrates the accurate registration: the edges from image 2 are super-imposed (blue) onto the transferred image to demonstrate sub-pixel registration.

3.5.1 Reconstruction from 2 views

Reconstructing a plane conic from its image in two views is equivalent to reconstructing a dual-space cone from two conics (see figure 2.14).

Two conics impose a total of 8 constraints on the cone, and hence equation (3.3) provides a total of 8 independent equations. The solution is therefore given by a family of dual-space quadrics

\[
Q^* = \beta Q_1^* + \gamma Q_2^* .
\]

In a similar approach to section 3.2.1, as the dual quadric is known to be rank 3, clearly
\(|Q^*| = 0\), giving a quartic equation in the unknown scalars \(\beta\) and \(\gamma\):
\[
q_1^*\beta^4 + q_2^*\beta^3\gamma + q_3^*\beta^2\gamma^2 + q_4^*\beta\gamma^3 + q_5^*\gamma^4 = 0 ,
\]
where \(q_1^*, q_2^*, q_3^*, q_4^*, q_5^*\) are constant coefficients as functions of \(Q_1^*\) and \(Q_2^*\). As before, both \((\beta, \gamma)\) and \((-\beta, -\gamma)\) are solutions, and of the four solutions to this quartic, only three are independent. Further there is a degenerate rank 2 solution to this quartic which can also be discounted: the quadric corresponding to a pair of planes (dual to the two camera centres).

A conic from two views can therefore be reconstructed up to a 2-fold ambiguity. A further view removes this ambiguity. Figure 3.14 shows an example conic reconstruction.

**Final note.** It is satisfying to note that this algorithm is capable of reconstructing any conic, whilst previous algorithms [91] have traditionally failed to reconstruct a conic which intersects with the base-line.

### 3.6 Summary

This chapter has presented both closed-form and optimal algorithms for the reconstruction of

- General quadrics from point and/or silhouettes constraints:
  - 9 point constraints give a complete reconstruction;
  - 3 silhouettes give a complete reconstruction;
  - 2 silhouettes and 1 point constraint gives a 2-fold ambiguity in reconstruction;
- Degenerate quadrics from point and/or silhouette constraints:
  - 8 points allow a reconstruction up to a 2-fold ambiguity;
  - 3 silhouettes give a complete reconstruction;
- Planar conics;
  - 2 views gives a reconstruction up to a 2-fold ambiguity;
– 3 views gives a complete reconstruction.

In all cases, the reconstructions have been shown to be of sub-pixel accuracy when reprojected into the original views.

Further, an algorithm for correcting a set of corresponding conics (that is, either multiple views of a planar conic, or a number of silhouettes of a general quadric) has also been outlined.

Finally, results of registering views with quadric-induced transfer are given, and it is seen that, again, sub-pixel accuracy is achieved.
Figure 3.14: Three views of a conic are sufficient to fully reconstruct the conic. The reconstructed conic can then be reprojected into a fourth view (red).
Chapter 4

Reconstructing General Surfaces — Theory

This chapter considers the problem of applying image data in a structured manner with the goal of reconstructing a viewed surface. Whilst other sources of information about the surface structure might be available (for instance shape from shaping in a single view), it is clear from figure 4.1 that both silhouettes and texture or points matched over multiple views are a rich source of data. The theory behind these information sources is considered in detail, whilst an efficient implementation is presented in chapter 5.

A novel two-tiered approach is adopted:

- Silhouettes provide a rich source of information about the viewed surface structure, and efficient algorithms to exploit this information are available. However, reconstructing from silhouettes alone is not sufficient: surface concavities, for example, are not accurately reconstructed;

- Imaged surface texture provides further information about the structure, but is traditionally difficult to exploit, as dense matching algorithms are both slow and unreliable. It is shown that given the information available from silhouettes, surface texture can be exploited much more effectively as an additional source of surface information.

These geometric constraints available are complementary, so that the deficiencies of one source can be overcome by the strengths of the other.

Section 4.1 rehearses and extends the theory of reconstructing a surface from the silhouette. It is shown in section 4.1.3 that a set of silhouettes can be back-projected to reconstruct the visual hull: in some cases a good approximation to the viewed surface. Sec-
4 Reconstructing General Surfaces — Theory

tion 4.1.4 then considers in detail the geometry of epipolar tangents, showing that whilst, in general, silhouettes only provide a tangency constraint on the surface, at *frontier points* this tangency constraint is upgraded to a point constraint. The concept of the *epipolar net* is proposed (section 4.1.6) as the set of all point constraints provided by a sequence of silhouettes of a surface.

The limitations of reconstructing a surface from silhouettes alone are considered in section 4.1.8. An alternative source of information, which will be seen to complete the information from silhouettes is that available from stereo triangulation of matched points or texture between two or more views (section 4.2). In order to prepare the ground for an elegant algorithm which will use this texture information, section 4.2.2 introduces a *surface-induced transfer* matching criterion for texture matches viewed over multiple images: this is possible, as it is assumed that a local approximate to the surface is available from information such as the image silhouettes.

As in chapter 3, it is assumed that multiple views of a surface are available, and that the camera projection matrix for each image is known.

![Image of silhouettes and texture matches]

**Figure 4.1:** Six images (top image set) from a sequence of a toy dinosaur. The figure demonstrates that the apparent contours (middle image set) are a rich source of information for reconstruction. Corner matches (lower image set) also provide further information on the surface structure.
4.1 Reconstruction from silhouettes

4.1.1 General

Consider a surface, $S$, viewed in an image with camera projection matrix $P$ and camera centre $C = \text{null}(P)$. A point, $X$ on $S$ projects to $x = PX$ and a 3-D ray passes through $X$ and $C$ (intersecting the image plane at $x$).

If this ray touches—or is tangent to—the surface at $X$, then $X$ lies on the \textit{contour generator}. Further, $x$, in the image, lies on the \textit{apparent contour} (see figure 4.2). If the ray does not intersect $S$ at any other point, then $x$ lies on the silhouette. The contour generator forms a curve (or set of unconnected curves), $\Gamma$, on $S$. Any point on the surface projects \textit{inside} the silhouette in the image.

At any image point, $x$, on the silhouette, a tangent line to the silhouette can be defined. This line back-projects to a plane in 3-D, which is also tangent to $S$ at $X$. In reverse, the tangent plane to $S$ at any point on the contour generator must pass through the camera centre, and project to an image line which is tangent to the image silhouette.

Figure 4.2: The apparent contour is the image of the contour generator. The surface tangent plane at any point on the contour generator passes through the camera centre $C$. 
4.1.2 Silhouette cone

Any image point on the silhouette back-projects to a ray in space passing through the camera centre, \( C \), and touching \( S \) on the contour generator. This set of rays form a “cone” in space.

It was seen that any point on the surface projects inside the image silhouette: it follows that any point in the surface lies inside the silhouette cone. Further, any volume, \( V \), entirely enclosed by \( S \), projects inside the image silhouette.

4.1.3 Reconstructing from silhouettes

The silhouette cone encloses and is tangent to the surface along the contour generator. Thus is provides important cues which can be used in reconstructing \( S \).

Consider two silhouette cones back-projected from two different silhouettes of \( S \), and a general 3-D point, \( X \), in space. If \( X \) lies outside either one of the cones, from section 4.1.2, it is clear that \( X \) lies outside \( S \). Similarly, if \( X \) projects outside the image silhouette in both images, \( X \) lies outside \( S \). Note that the converse is not necessarily true: if \( X \) projects inside both silhouettes (and hence lies inside both back-projected cones), it does not necessarily lie inside \( S \) (this will be reconsidered in section 4.1.8).

The intersection of all back-projected silhouette cones is termed the visual hull [71].

Historical Summary. As pointed out in [113], the first reconstruction algorithm to exploit this volumetric approach was proposed by Baugnart [5], although widely overlooked in the computer vision literature.

It was suggested in Barrow and Tenenbaum [3] that the surface orientation along the apparent contour could be computed directly from image data. Koenderink [66] related the curvature of an apparent contour to the intrinsic curvature of the surface (Gaussian curvature). Further work on determining local surface parameters from the apparent contour, under general camera motion, is that of both Blake and Cipolla [12, 24] and Vaillant and Faugeras [127]. Cipolla and Blake used their work to demonstrate obstacle avoidance [23].

Giblin and Weiss [44] and Connolly and Stenstrom [26] both discussed reconstruction
of the visual hull by back-projecting a polygonal representation of the image silhouette represented.

4.1.4 Epipolar tangency

Consider two views of a general surface, $S$. Each image silhouette is the projection of a contour generator, a curve on $S$. If the two contour generators intersect at one of more points (termed frontier points in [23]), an epipolar tangency [87, 92] is formed (see figure 4.3).

A frontier point $X_f$ projects into each of the two images as $x_{f,1}$ and $x_{f,2}$. As $X_f$ lies on the contour generator for each image, the image points $x_{f,1}$ and $x_{f,2}$ must also lie on the silhouettes in their respective images. Defining the plane, $\Pi$, tangent to $S$ at $X_f$, it follows from section 4.1.3 that this plane passes through both camera centers. Therefore $\Pi$ projects into the images as lines, $I_1$ and $I_2$. Both $I_1$ and $I_2$ pass through the epipoles in either image and are tangent to the silhouette.

The following surface constraint follows: given the silhouettes, $O_1$ and $O_2$, of a surface in two views, and the epipoles, $e_1$ and $e_2$ in their respective images, if there exists an epipolar line, tangent to $O_1$ (touching the silhouette at $t_1$), then there must exist a corresponding epipolar line in the second image also tangent to $O_2$ (at $t_2$). The points $t_1$ and $t_2$ correspond to the same 3-D point $T$ which lies on the surface.

Whilst it has been seen that in general a silhouette provides only a tangency constraint on the surface for reconstruction, it has now been shown the an epipolar tangency also provides a point constraint.

4.1.5 Epipolar curves

Epipolar tangency is a property of two views only, and no direct extension can be made in the case of three of more views of a surface unless the camera motion lies entirely on a plane tangent to the surface, in which case all the cameras share the same frontier point. Such motion is unlikely and not considered here.

Consider a moving camera, with camera centre $C(t)$ at time $t$, viewing a surface. A frontier point, $T(t, t + \Delta t)$, given two views at times $t$ and $t + \Delta t$, lies on the surface tangent
Figure 4.3: A frontier point is given by locating the pair of epipolar lines tangent to the silhouette, and back-projecting the tangent points. The surface tangent plane at the epipolar tangent point, $T$, passes through the two camera centres.

plane, $\Pi(t, t+\Delta t)$, passing through $C(t)$ and $C(t+\Delta t)$. Letting $\Delta t \to 0$, the epipolar tangent plane, $\Pi(t)$, passes through both $C(t)$ and $C(t + dt) = C(t) + dC$, and is tangent to $S$. The frontier point, $T(t)$, is now only a function of $t$ and defines an epipolar curve on $S$ [45].

Clearly, all points on the epipolar curve are frontier points, given the complete image sequence, and can hence be reconstructed from the silhouettes alone.

### 4.1.6 Epipolar nets

Whilst the epipolar curve is of particular interest, the complete significance is not clear until the following extension is made.

A frontier point is found given any two images. The epipolar curve assumed these images to be a neighboring pair in the sequence of images, but this assumption is restrictive. It is relaxed by considering the frontier point, $T(t_1, t_2)$, generated by two images at times $t_1$ and $t_2$: no assumptions are made about $t_1$ and $t_2$. The epipolar curve is then given, as above, by $t_2 = t_1 + dt$.

Of interest is the set of frontier points generated by holding $t_1$ constant (note that $T(t_1, t_2) = T(t_2, t_1)$ and hence holding $t_1$ constant is equivalent to holding $t_2$ constant). Choosing a reference image by setting $t_1 = t_r$ gives a set of frontier points $T_{t_r}(t)$. Once again, $T_{t_r}$
Figure 4.4: **Epipolar curve.** A sequence of images of a surface gives a set of frontier points on the surface: an epipolar curve (green). The first and last image planes and the contour generators for six images out of the sequence are shown (red for the first image fading through to blue for the last image). The camera centres for all six images are seen in black.

is only a function of $t$, and hence defines another curve on $S$. The set of all such curves defines the **epipolar net** (see figures 4.5 and 4.6).

Although it is common to consider a subset of images from a sequence taken by moving a camera as suggested in section 4.1.5, it is important to note that in practice only a finite number of images are available. Thus, the epipolar net reduces from the set of curves given by $T_t(t)$ to the set of discrete points given by $T_m(n)$ where $m$ and $n$ are indices into the image sequence. Given a discrete number of images, **the epipolar net is the set of all frontier points generated from every epipolar tangency between every pair of images in the sequence.** The epipolar net therefore describes all the information available from epipolar tangencies given an image sequence (see figure 4.7 for example).

### 4.1.7 The visual hull

The entire visual net (that is, each frontier point) lies on the surface of the visual hull. The proof is trivial: from section 4.1.4, a frontier point must lie on the contour generators for at least two images, and as the surface of the visual hull is coincident with each contour
Figure 4.5: A set of frontier points (green) on a surface viewed from six different viewpoints. Only the frontier points generated between the first and each of the other five other images are shown for clarity. Note that the frontier points lie at the intersections of the contour generators.

Figure 4.6: Epipolar net. Given a sequence of images of a surface, the epipolar net (green) encodes all the information available from epipolar tangencies.
Figure 4.7: Six from a sequence of 20 images of a gourd (top), with the corresponding reconstructed model (bottom). All the frontier points (the epipolar net) are superimposed in red. Note the density of frontier points, and therefore point constraints available from the image silhouettes alone.

generator (section 4.1.3) it follows that the frontier point must lie on the visual hull.

The following result is therefore paramount: the visual hull encodes all information available from the image silhouettes. That is, both tangency to $S$ along the contour generators and the point constraint at frontier points is encoded.

### 4.1.8 Limitations of the visual hull

It was noted qualitatively by Martin and Aggarwal [80] and Srinivasan et al. [108] that the visual hull was in some cases not an accurate or complete representation of the surface. However it was Laurentini [71] who first considered in detail the implications of reconstructing a surface from silhouettes alone. In particular, Laurentini made the following observations:

- Concavities are never “seen” if only the silhouettes of a surface are considered. There-
fore the visual hull does not represent these areas, and in general only convex objects\textsuperscript{1} are accurately reconstructed by from image silhouettes.

- \textit{The visual hull depends not only on the object itself, but also on the region allowed to the viewpoint} [71].

\textbf{Concavities}

It is well known that the silhouette does not provide information about concavities. This is clearly seen in figure 4.8, for example.

![Figure 4.8: A surface (red) is viewed by five cameras (blue), and the visual hull is shown in green. Note that the concavity is never “seen” on any of the silhouettes and hence is not be reconstructed from silhouette information alone: the visual hull smoothes over this region.](image)

\textbf{Number and position of viewpoints}

A further limitation of the visual hull is its dependence on both the number of viewpoints and the position of the viewpoints (figures 4.9 and 4.10). The visual hull only provides information about the surface along the contour generators, which depends not only the surface shape, but also on the position of the viewpoint relative to the surface. Naturally, given more viewpoints, the visual hull provides a better representation of the surface.

\textsuperscript{1}Although this is true in 2-D, the 3-D situation is a little more subtle. For example, a hyperbolic shape can be reconstruction from the silhouettes alone: the correct formulation is given in [25], and is not expanded here.
Figure 4.9: A surface (red) viewed by just two cameras, with the visual hull shown in green. Although the surface is close to convex (in contrast to the surface viewed in figure 4.8), from two viewpoints the visual hull is clearly not close to the viewed surface.

Figure 4.10: The visual hull from the dinosaur sequence in figure 4.1, reconstructed from (left to right) 2, 4 and all 36 silhouettes. Note the natural increase in accuracy as the number of silhouettes increases.

4.2 Reconstruction from texture

It has been shown that the apparent contour does not provide any information about concavities of the viewed surface. In order to accurately reconstruct such surface regions, it is necessary to make use of other information such as texture.

Dense stereo algorithms (for example, but not limited to [65]) make good use of texture information, but to date have only been extended from two views to many by merging surface patches — an approach that does not generalize well to circumnavigation of the object as is the case here (see chapter 7).

It is also possible to obtain surface shape information from features such as corners (see [49]) or edges (see [19]) in the image. However, such data is sparse and generally a
parameterized surface model must subsequently be fitted (for example, see [48]) in order to obtain a dense reconstruction.

### 4.2.1 Triangulation

Consider two or more views of a surface point: clearly, by back-projecting from each view, the 3-D point can be reconstructed (see figure 4.11).

Stereo information like this is very important as it does not suffer from either of the limitations of the visual hull as provided by silhouette information alone: if any surface point is visible in two or more images, it can be reconstructed by triangulating from the images:

- Concavities can now be reconstructed correctly;
- The resulting reconstruction is not dependent on the number or locations of the viewpoints as long as the point is not occluded in one of the views.

![Triangulation Diagram](image.png)

Figure 4.11: The rays from two views of a surface point can be back-projected, and the surface point reconstructed from this triangulation. Any viewed point on the surface can be reconstructed in this manner.
4.2.2 Texture-based matching

To exploit the advantages of reconstructing surface points by triangulating from two or more views, it is necessary to be able to match a pair of points between two or more images. Once a pair of points have been matched, they can be back-projected and the rays intersected to give a 3-D reconstruction.

**Point intensity matching**

As an example, one matching criterion simply matches the image intensities between the two hypothesized correspondences. That is, given an image point in each of two images, \( x_1 \) and \( x_2 \), with corresponding image intensity values given by \( I_1(x_1) \) and \( I_2(x_2) \), the matching score is typically defined as

\[
s = (I_1(x_1) - I_2(x_2))^2 .
\]

Equation 4.1

A low score indicates a good match.

In most practical cases, this matching score is insufficient. Image noise (on \( I_1(x_1) \) and \( I_2(x_2) \)) affects \( s \), as does noise localization noise on \( x_1 \) and \( x_2 \). Further, \( s \) is not robust to lighting changes between images, which generally result in an unknown affine transformation in intensities over the entire (or part of the) image.

**Intensity correlation**

A more common matching score in current use is a texture correlation approach (figure 4.12). This time, the score \( s \), lying between -1 (for a good match) and 1 (for a bad match), considers the intensities over a pre-defined \( p \times p \) patch (typically 5-7 pixels) in the image: the cross-correlation \( <I_1, I_2> \) is defined (the summation taken over the whole texture patch) as

\[
<I_1, I_2> = \frac{1}{(2p+1)^2} \sum_{m=-p}^{(p,p)} (I_1(x_1 + m) - \bar{I}_1(x_1)) (I_2(x_2 + m) - \bar{I}_2(x_2)) ,
\]

Equation 4.2

where the mean intensity \( \bar{I}_k(x_k) \) is given by

\[
\bar{I}_k(x_k) = \frac{1}{(2p+1)^2} \sum_{m=-p}^{(p,p)} I_k(x_k + m) .
\]

Equation 4.3
The correlation score is then given by

\[ s = -\frac{\langle I_1, I_2 \rangle}{\langle I_1, I_1 \rangle^{1/2} \langle I_2, I_2 \rangle^{1/2}}. \]  

(4.4)

This correlation score is a vast improvement over that given by equation (4.1). It is robust to small errors in \( x_1 \) and \( x_2 \) as the signal is locally correlated, and image noise is averaged over the texture patch. Further, a normalized correlation such as this is invariant to lighting changes which induce an overall affine transformation on the image intensities.

However, equation (4.4) fails if the image pair viewpoints are widely separated [88, 89] (figure 4.14) or have distinctly different orientations (figures 4.13). Inherently, the correlation assumes that if \( (x_1, x_2) \) forms a corresponding pair between two images (and hence \( I_1(x_1) \approx I_2(x_2) \) modulo the noise discussion in the previous section), then for a “small” image vector, \( v \), the pair \( (x_1 + v, x_2 + v) \) also back-project to the same 3-D point.

**Surface-induced transfer**

Consider two views of a known surface, \( S \). Further, assume that the camera projection matrices, \( P_1 \) and \( P_2 \), are known. A surface-induced mapping,

\[ x_2 = M(x_1) \]

can be defined, mapping an image point, \( x_1 \), in the first image to the corresponding point, \( x_2 \), in the second image. The mapping \( M \) is clearly a function of \( S \), \( P_1 \) and \( P_2 \).

The quadric induced transfer discussed in section 2.2.3 is an example of this mapping, as is the homography induced between two images of a plane.

The cross-correlation function of equation (4.2) can now be redefined (see figure 4.15):

\[ \langle I_1, I_2 \rangle = \frac{1}{(2p + 1)^2} \sum_{m=-(p,p)}^{(p,p)} (I_1(x_1 + m) - \bar{I}_1(x_1)) (I_2(M(x_1 + m)) - \bar{I}_2(x_2)) \]  

(4.5)

with

\[ \bar{I}_2(x_2) = \frac{1}{(2p + 1)^2} \sum_{m=-(p,p)}^{(p,p)} I_2(M(x_1 + m)) \] .

As the point pair \( (x_1 + v, M(x_1 + v)) \) are (from the definition of \( M \)) guaranteed to correspond to the same 3-D point, equation (4.5) now ensures that the intensity comparisons are only made between corresponding image pixels.
Figure 4.12: **Texture Correlation.** A common matching test involves correlating the intensities between two images over a fixed-size patch.

Figure 4.13: If the views are at very different orientations, equation (4.4) fails to provide an accurate correlation score.

Figure 4.14: In similar way to figure 4.13, views taken from widely separated viewpoints are also difficult to match without further information about the scene geometry.
Equation (4.5) is therefore robust under widely separated views, arbitrary deformations of one of both images and changes of lighting.

Figure 4.15: **Surface-induced transfer.** Given two views of a surface, a mapping can be defined between points in the first image and points in the second image (*left*). If the assumption about the surface structure is incorrect (*right*), the mapping function itself will also be incorrect.

**Practical Issues.** In practice, \( P_1 \) and \( P_2 \) can be assumed to be known, whilst \( S \) is not known (indeed, this is the goal!). However, often an approximation to \( S \) can be found:

- In the absence of all other information, the visual hull generated from many views has been seen (section 4.1) to approximate convex surfaces well. In particular, the visual hull is coincident with and tangent to \( S \) along the contour generators: in these regions, the visual hull is a good approximation to \( S \), and the mapping function \( M \) can be locally approximated;

- Often it is possible to approximate \( S \) only locally. Matched corner features or edges, for instance, give a sparse reconstruction and a parameterised surface (planar, quadratic...) can be fitted to locally estimate \( M \);

- Finally, if no information about the surface shape is known, \( M \) can be estimated as a similarity transform (as implicitly assumed by equation 4.2), whilst still encoding information about the image orientations.
In all cases, $M$ is unlikely to be accurate over the entire surface, but on a local scale it is clear that the matching function of equation (4.2) is the worst-case situation (it assumes the surface to be fronto-parallel at all points), when no a priori information about scene or cameras is available.

### 4.3 Conclusions

Much consideration has been given in this chapter to the use of both silhouettes and imaged surface texture as information about the shape of a viewed surface.

It has been seen that the silhouettes from a sequence of views of a surface can be used to reconstruct the visual hull: this surface entirely encloses, and is tangent along the contour generators to, the viewed surface. The geometry of epipolar tangencies, allowing frontier points on the viewed surface to be reconstructed from silhouettes alone, was considered in detail: the set of all such point constraints was introduced as the *epipolar net*.

Naturally complementing the information available from image silhouettes is that available from imaged surface textures, and it was seen that by matching and triangulating textures in two or more images, information about surface regions such as concavities (which can not be reconstructed from silhouettes alone) can be found. Finally, it was shown that, by locally estimating the surface structure (from the visual hull, for example) as a starting point, a robust correlation-based matching function can be defined.
Chapter 5

Reconstructing General Surfaces — Implementation

This chapter outlines a robust and efficient algorithm to reconstruct a general surface from a sequence of images. First, it is shown that the silhouettes can be used to reconstruct the visual hull, and this is then used as a starting point for a space carving algorithm which then reconstructs areas of the surface which are not accurately represented by the visual hull.

As with any surface reconstruction algorithm, the choice of surface representation is paramount. Section 5.1 gives a broad overview of the choices of surface representation. Broadly, the representations fit into two categories: parameterised surfaces (section 5.1.1), and volumetric surface representations (section 5.1.2).

The reconstruction of the visual hull is then considered in section 5.2. Representing the surface as a voxelmap, an efficient algorithm to reconstruct the visual hull is outlined in section 5.2.1, and efficiency improvements on this algorithm are suggested in section 5.2.2. Section 5.2.3 presents some results and demonstrates the robustness of the algorithm.

Section 5.3 shows how the surface can be refined using information from surface texture over a number of views. The space carving algorithm is outlined in section 5.3.1. It is suggested that the algorithm can be improved with three advances:

- Initialising the space-carving from the visual hull, and not from an arbitrary surface enclosing the viewed object (section 5.3.2);

- Implementing the surface-induced transfer photo-consistency test proposed in chapter 4 (section 5.3.3);
Selective choice of application of the photo-consistency test to voxels in the model (section 5.3.4).

A gallery of results (section 5.4) shows some of the models generated using this algorithm, which is summarised in section 5.5.

Throughout the chapter, it is assumed that the projection matrices for a sequence of views of an object are known. These are obtained automatically from image features (chapter 7).

### 5.1 Surface Representation

Of primary concern in implementing any surface reconstruction algorithm is the choice of underlying surface representation. Stoddart et al. [110] sets out a list of characteristics ideally satisfied by the choice of representation which are summarized here:

- Ability to efficiently represent curved smooth surfaces, whilst still permitting discontinuities ($G^0$—a “step”—or $G^1$, such as a “roof edge”);

- Adaptable, and not necessary uniform “level of detail”. The surface complexity might be increased in areas of interest or complex structure;

- Representation of an arbitrary topology surface must be possible (for example, a sphere has a different topology to a torus).

This section considers commonly available surface representations and their ability to fulfill these requirements.

Broadly, general surface representations fall into two categories:

- Parameterised surfaces, such as piecewise planar (polygonal meshes) or spline patches, are considered in section 5.1.1;

- Volumetric methods, such as voxmaps and sampled level-sets, are considered in section 5.1.2.
5 Reconstructing General Surfaces — Implementation

It is seen that whilst voxmaps do not provide curvature information directly, they are both efficient to compute, and deal particularly well with arbitrary topology surfaces. Further, enhancements such as oriented particles or level-sets are closely related and do provide higher order surface information.

5.1.1 Parameterised surfaces

Piecewise planar meshes

Possibly the most common surface representation is the polygonal mesh. Examples include the traditional triangular mesh and simplex meshes [32]. The surface is assumed to be piecewise planar, and is generally stored as a set of “faces”, delineated by a joining mesh of edges. This is the most general primitive provided by most graphics standards (OpenGL, VRML and RenderMan\(^1\) for example).

Clearly, such a representation places no constraints on the topology of the surface and the level of detail can be adapted by changing the number and size of faces in the mesh.

However, with a finite number of faces, a polygonal mesh can not represent a general smooth surface. Whilst this is generally considered surmountable as the number of faces required for a visually good representation is well within the capabilities of modern workstations, the main difficulty lies in manipulating a polygonal mesh. Vast areas of literature have been dedicated to the problem of smoothing, refining, removing knots from and reparameterising triangular or polygonal meshes.

**Winged-Edge Model.** The winged-edge model was developed by Baumgart [5] as a representation for piecewise planar meshes. The model stores a polygonal mesh as a linked set of edges, vertices and faces: some redundancy is deliberately introduced in order to allow the mesh to be more efficiently manipulated. [5] presents a comprehensive set of algorithms to apply operations such as intersection and subdivision on a polygonal mesh.

\(^1\)http://www.pixar.com
Spline patches

Spline representations of surfaces were suggested by Bartels et al. [4]. A general surface is broken into patches, and locally parameterised as a quadratic or cubic patch (rarely is the added complexity of higher order polynomials necessary). General deformable surfaces with fixed topology are presented by Terzopoulos et al. [120, 119].

In many cases, such a representation is more efficient than a piecewise planar approach. The size of the patches can be adjusted to give a required level of detail.

Representing arbitrary topology surfaces with spline patches is not as easy. Many spline networks, such as tensor product B-splines parameterise the surface as a rectangular grid and hence surfaces such as a sphere can not be represented without a very non-uniform distribution of control points.

NURBS. Non-uniform rational B-splines are the CAD industry standard parametric surface. They are particularly interesting in projective reconstruction as they are well-behaved under a general projective transformation. Further, they are able to represent any topology. However, in common with almost all other parametric surfaces, the topology must be specified a priori, and is difficult to dynamically change.

GBBS. The generalised bi-quadratic B-spline is a more recent surface representation [73] made from B-spline patches. Providing geometric continuity [35], GBBSs can represent an arbitrary topology surface.

Whilst this approach does satisfy the requirements suggested in the introduction to this section, and despite recent advances in reducing the computation weight of GBBS [96, 110, 109] (under the name SLIME), they still remain an inefficient surface representation at high resolution. Further, unlike NURBS, GBBS do not behave well under projective transformations: in general, a reparameterisation is necessary as with other spline patch networks.
5.1.2 **Volumetric surface representation**

**Voxmap**

A voxmap provides a binary partition of space. A region of space is subdivided at a given resolution, and each voxel is labelled with one of two values: generally, these are termed “inside” or “outside” referring to whether the voxel lies entirely enclosed by or entirely outside the represented shape\(^2\). See figure 5.1.

The advantage of such a representation is three-fold:

- A voxmap can represent an arbitrary topology surface, and topology changes are handled with no further cost;

- Whilst “smoothness” is not necessarily defined (the curvature is unknown), a local parameterisation can be computed as necessary by fitting a quadratic or cubic patch as discussed in section 5.1.1:

- Voxmap generation is very easy to parallelize.

---

**Figure 5.1:** **Colored voxmap.** A region of space enclosing a surface (*red*) is subdivided into voxels (*left*). The voxels are colored according to their position in relation to the surface (*right*): *black* voxels lie entirely inside the surface, *white* voxels lie entirely outside the surface and *grey* voxels lie partially inside and partially outside the surface.

---

\(^2\)As the sampling resolution is finite, voxels which span the surface, and hence lie partially “inside” and partially “outside”, have a special status which will be reconsidered in the following sections.
Octree

An octree representation of space can be considered as an efficient storage of a voxelmap [61, 81]: regions of identically marked voxels are grouped together (figure 5.2). In practice, an octree is a tree-structured representation: a low-resolution voxelmap is generated, where each voxel is either labelled, or contains a pointer to a set of higher resolution voxels (4 in 2-D, 8 in 3-D).

Figure 5.2: **Octree.** The octree representation of a surface. The space is sampled at a low resolution, and voxels of interest (i.e. those close to the surface) are subdivided until the required resolution is obtained.

Oriented particles

A development on the traditional voxelmap is the idea of an oriented particle approach to surface representation [115]. In this case, the surface is parameterised by a discrete set of points defining a normal and a curvature as well as location. Local interaction between particles ensures smoothness as well as providing a local triangulation for visualization purposes. Changes in topology are only constrained by these local interactions.

Unfortunately, some of the disadvantages of a simple triangular mesh are also inherent in the oriented particles approach: without careful attention to the interactions between particles (imposing a smoothness constraint, for example), “knots” can form as a particle is moved. Imposing constraints such as these make topology changes difficult to handle.
Discrete sampled level-set

A level-set defines a 3-D surface, $\mathcal{S}$, as the zero-set of a function, $\Phi(x, y, z)$. That is, a point in space, $X = (x, y, z)^T$, lying on $\mathcal{S}$, satisfies $\Phi(X) = 0$. It is common to define $\Phi$ as the signed distance to $\mathcal{S}$. Sethian [101] and chapter 6 give a more complete discussion of level-sets.

Sampling $\Phi$ at a set of discrete points gives a representation of the level-set, and it has been shown [101] that primitives such as the surface normal and local curvature can be estimated.

5.1.3 Voxmap representation for reconstruction

The representation chosen in this chapter is a voxel (specifically an octree) representation. It has been seen that an octree is efficient to compute whilst having the ability to represent arbitrary topology surfaces with no extra computational cost.

Further it will be seen in the following sections that a voxel representation of the visual hull can be efficiently computed from the visual hull. This then leads naturally on to the space carving algorithm proposed in section 5.3.

The lack of no higher order surface information such as normals or curvature will be considered in section 5.3.3 where a local quadratic patch will be fitted to the surface of the voxmap in order to provide such information.

5.2 Surface Reconstruction From Silhouettes

Having chosen to represent the surface as an octree, the next step is to reconstruct the visual hull from image silhouettes.

This section proposes an efficient algorithm to compute the voxmap representation of the visual hull given a set of image silhouettes of a surface.

5.2.1 Building a voxmap from silhouettes

A point in space, $X$, lies inside the visual hull of a shape if its projection into each image lies inside the respective image silhouette (see section 4.1.3). Similarly, a voxel, $V$, lies entirely
inside the visual hull if its projection into each image also lies entirely inside the image silhouette. Or more precisely, a voxel, $V$, lies entirely within the visual hull of a shape, $S$, if the silhouette of $V$ in each image lies entirely inside the silhouette of $S$ in this image. The reverse follows: if $V$ projects outside one or more silhouettes of $S$, then $V$ lies outside the visual hull.

**Visual hull surface voxels.** A special case is needed for voxels which span the surface of the visual hull. Clearly, such voxels will project inside the silhouette in some images, but will also span the silhouette in some images. The defining characteristic, however, is that they do not project outside the silhouette in any image.

**Voxmap terminology.** For brevity, the terminology of Szeliski [115] is adopted: a voxel lying inside the visual hull is termed “black”, a voxel lying outside the visual hull is termed “white”, and a voxel spanning the surface of the visual hull is termed “grey”.

**Outlined labelling algorithm.** An algorithm is require which will label each voxel in a voxmap as lying either inside or outside the visual hull from the image silhouettes. There are two choices available:

- The voxel labelling is performed in 3-D: the silhouettes are each back-projected as a “cone”, and each voxel is tested to see if it lies inside each of these cones;
- The voxel labelling is performed in the images: each voxel is projected into each image to determine if lies inside the visual hull.

It is generally more efficient to perform 2-D boolean operations on shapes, and thus the second approach is generally chosen.

As suggested by Szeliski [115], the following algorithm labels each voxel in a voxmap and hence computes the voxel representation of the visual hull. Given the silhouette of a surface in $m$ images, and subdividing the space into $n$ voxels:
• For each voxel, \( V_i \) with \( i = 1..n \):
  
  - For each image, \( j \) with \( j = 1..n \):
    
    * Project \( V_i \) into image \( j \);
    
    * If the silhouette of \( V_i \) lies outside the silhouette of \( S \) in image \( j \), then mark \( V_i \) as *white* and exit loop;
    
    * If the silhouette of \( V_i \) spans silhouette of \( S \) in image \( j \), mark as *grey*;
  
  - If \( V_i \) unlabelled, label as *black*;

• The visual hull given by all voxels marked as *black*. The voxels on the surface of the visual hull are marked as *grey*.

See figure 5.3. Figure 5.1 shows a 2-D example of the output of this algorithm.

Aggarwal *et al.* [80, 22] first proposed reconstructing the visual hull as a voxmap from silhouettes in three views. Veenstra and Ahuja [128] extended this to a full sequence. However, it was not until Szeliski [115] that this more efficient algorithm was proposed: previously, a separate voxmap was generated from each image, and the subsequent voxmaps were then intersected.

### 5.2.2 Improved efficiency algorithm

**Voxel projection test**

The voxmap generating algorithm requires each voxel in space to be successively projected into each image and its position relative to the image silhouette tested (assigned a “color”). This test is performed \( m \times n \) times and is the most expensive part of the algorithm: it naturally makes sense that an efficient test is derived.

It is assumed that a binary silhouette image is available for each image in the sequence: each pixel in the image is marked as black if it lies outside the silhouette, and white if it lies inside the silhouette. Various methods of providing this segmentation are available, and in this case chroma-keying (also known as blue-screening) is used. Figure 5.4 gives an example sequence with the resulting binary silhouette images. This pre-processing can be
achieved in close to real-time and is a fixed computation cost for a given sequence.

**Complete Voxel Test**  The outline of a voxel, projected into an image, is a 6-gon (strictly, it can have from 4 to 6 sides). Given the binary silhouette images as described above, a voxel can therefore be labelled as follows:

- Project the voxel into the image: this involves projecting each of the 8 vertices of the voxel;
- Compute the outline polygon of the projected voxel by taking the convex hull;
- Iterate over each pixel inside the projected voxel in the silhouette image: if every pixel is white, the voxel lies entirely outside the silhouette; if every pixel is black, the voxel lies inside the silhouette.

This algorithm is depicted in figure 5.5.

However this algorithm is inefficient. Computing the convex hull of the projected vertices requires a fixed cost over-head for each voxel, whilst scanning each voxel inside the convex hull is also slow. Whilst this test is guaranteed to never mis-label a voxel, a hierarchy of more efficient tests is now presented.

The first test is efficient, as only the projected vertices of the voxel are tested. However it will be seen that it is possible for this algorithm to mis-label a voxel (e.g. label the voxel as lying inside the visual hull when it actually lies on the surface of the visual hull).

Therefore a second test is required for some voxels. This time, only the distance from the centroid of the voxel to the image silhouette is considered (assuming the voxel to be approximated by a bounding circle in the image). Once again, however, it is possible that some voxels will be mis-labelled. In this case, it is necessary to resort to the expensive, but complete test proposed in this section.

**Vertex-Based Voxel Test.**  The first test in this hierarchy of voxel labelling algorithms simply tests the positions of each of the vertices of the voxel, relative to the image silhouette (see figure 5.6):
Figure 5.3: **Voxmap from Silhouettes.** A region of space, enclosing the viewed object, is sampled and each voxel projected into each image. If the projected voxel lies outside the silhouette, the voxel itself must lie outside the visual hull and is labelled “white”. If the projected voxel lies inside the silhouette, it is labelled as lying inside the visual hull (“white”). Voxels lying on the surface of the visual hull are marked as “grey”.

Figure 5.4: Six images from a sequence of 36 images of a cup, with their associated binary silhouette images.

Figure 5.5: Each pixel inside a projected voxel is sampled: *blue* pixels lie inside the image silhouette (*red*), and *green* pixels lie outside.
5 Reconstructing General Surfaces — Implementation

- Project each voxel vertex into the image;

- If the vertices are all inside, or all outside, the silhouette, (tentatively—see below) mark the voxels as black, or white respectively;

- Otherwise, mark the voxel as grey.

Figure 5.7 shows the two cases in which this test does not provide the correct voxel labelling: if all the vertices lie inside or outside the silhouette, it is not possible to correctly label the voxel from this information alone. In these two cases, the second test in the proposed hierarchy of voxel labelling algorithms is required.

Figure 5.6: The projected vertices of a voxel give an idea of the position of the voxel in relation to the image silhouette. Vertices lying outside the silhouette are marked green, and vertices inside the silhouette are marked blue.

Figure 5.7: The two ambiguous cases when determining the relative position of a projected voxel to the image silhouette from the positions of the projected voxel vertices alone. If all the vertices lie either outside (green) or inside (blue) the silhouette, this is not enough information to label the voxel as either lying completely inside or completely outside the silhouette.
**Distance-Based Voxel Test.** A “bounding box”, or in this case, a “bounding circle”, is computed around the projected voxel in the image. If, for each point in the image plane, the minimum signed distance to the silhouette is known, an efficient voxel test simply requires this distance to be compared with the radius of the bounding circle (see figure 5.9).

- Project each of the voxel vertices into the image;
- Compute bounding circle, defined by a center point, \( c \), lying at the projection of the centroid of the voxel and a radius, \( r \);
- Look up distance, \( d \), at \( c \) on the distance map:
  - If \( |d| > r \) and \( d < 0 \) then voxel lies entirely inside silhouette;
  - If \( |d| > r \) and \( d > 0 \) then voxel lies entirely outside silhouette
  - If \( |d| < r \), the voxel is labelled (tentatively—see below) labelled as grey.

Efficient algorithms for computing, or approximating, \( |d| \) for each point in the silhouette image are available: the chess-board distance can be computed from Rosenfeld and Kak [93], whilst a more accurate distance is given from the Borgefors algorithm [14]. Both algorithms require one or two passes through the image: a fixed computational cost which need only be payed once for each image in the sequence, and the results may be cached (see figure 5.8).

Again, the algorithm is not guaranteed to correctly label any voxel: as the voxel is approximated by a circle in the image, in figure 5.8(b) the voxel is labeled as grey despite the fact that it does not actually span the surface of the visual hull. Therefore in all cases where the voxel is labelled grey, the fall-back test of scanning the interior of the visual hull must be performed.

**Summary.** It is therefore proposed that the following hierarchy of tests, from most efficient to least efficient, be performed on each voxel in order to label it relative to the silhouette in one image:
• Perform the vertex-based test. If the voxel is marked as grey, then exit;
• Perform the distance-based test. If the voxel is marked as black or white, then exit;
• Perform the complete test and exit.

5.2.3 Results

General

The results of this algorithm are presented in figures 5.10, 5.11 and 5.12. In each case, it can be seen that a good representation of the surface is generated from then silhouettes alone.

In order to texture-map the surface, it is necessary to convert the voxmap to a triangular mesh: the well-known marching cubes [74] or marching triangles [56] algorithms show how this can be performed. Figure 5.12 demonstrates these results.

Robustness

Naturally, the algorithm presented above requires the silhouette to be accurately segmented from the image. In many cases, this is very efficient using chroma-keying techniques such as blue-screening: often the silhouette is located to within a pixel accuracy. However, this section considers the impact of inaccurate silhouette segmentation, and shows that the algorithm is robust even under very noisy conditions.

Observe, from the algorithm outlined in section 5.2.1 above, that a voxel is labelled as white, or outside, if it lies outside the silhouette in any one image. It is therefore important that the segmented silhouette gives an “outer-bounds” for the actual silhouette: if a projected voxel is found to lie outside the segmented silhouette, whilst actually lying inside the actual silhouette, it will be erroneously labelled white.

On the other hand, if a voxel appears to lie inside the silhouette (due to errors in the segmentation) in one image, but these errors are small enough, or random enough, and it lies outside the silhouette in a second image, the voxel will be correctly labelled.

Figure 5.13 depicts a set of voxmaps of the cup sequence under various levels of added synthetic noise. The silhouette is correctly segmented (as used in figure 5.10), but noise is
then added to the silhouette images\(^3\). It is seen that up to about 800 5-pixel-radius “specks”
can be added to the image as simulated noise before the voxelmap deviates significantly
from the noise-free voxelmap. A real example is also shown: the vase in figure 5.14 is against
a white background, and it was found to be difficult to segment automatically (particularly
around the base of the vase): despite this, the visual hull is clearly well reconstructed.

5.3 Refining Reconstruction using Image Textures

It was shown in section 4.1.8 that the visual hull does not fully describe a surface with
local concavities, and is also dependent on the positions of the viewpoints and number
of available views. In order to refine the visual hull, surface features (such as texture) are
used.

The following sections describe an implementation of the space carving algorithm [69]
which is used to exploit image surface features and hence refine the models generated from
silhouettes alone.

5.3.1 Space carving—past and future

Space carving [69, 99] provides a successive algorithm for removing voxels in a voxel
map with the aim of creating a 3D shape which reproduces the input images.

In brief, the idea of space carving is to project each voxel on the surface of a voxelmap into
a set of images. The projected voxel defines a correspondence between image points, and
the intensity at the corresponding points is evaluated to determine if it is phot-consistent
(see the following sections), ensuring the voxel is only compared with pixel values in im-
ages in which it is not occluded:

- A voxel which is not photo-consistent with the images is removed (carved) from the
  occupancy space;

- A photo-consistent voxel is coloured from the images with the corresponding pixel
colour.

\(^3\)In practice, it is often seen that blue-screening errors appear as random patches of incorrectly segmented
data. The noise added is of this form: black marks are added at random throughout the image to simulate
blue-screening errors.
Voxels are “uncovered” as surface voxels are removed, and the algorithm repeats until convergence. The final result is a set of voxels which closely reproduces the original images.

**Extensions.** There are 3 main drawbacks to the implementation described in [70]:

- **Voxmap initialisation.** Initialising the algorithm requires a voxelated region of space to be chosen. As voxels are only ever removed from the voxmap, it is assumed that the initialisation volume completely encloses the viewed surface. Obviously, the number of iterations of the algorithm (and hence the time to convergence) depends on the choice of starting volume: a smaller starting volume will converge faster. This problem is considered in section 5.3.2;

- **Photo-consistency test.** The problem of defining a robust, but efficient photo-consistency test is treated in [69]. A significant extension on the most recent algorithm is presented in section 5.3.3.

- **Repetition efficiency.** An iteration of the space carving algorithm projects each surface voxel into every image. Towards convergence of the algorithm, few voxels are removed on each iteration, and it becomes apparent that the same set of surface voxels are being repeatedly tested for photo-consistency. This is an obvious inefficiency which is considered in section 5.3.4.

### 5.3.2 Surface directed initialization

Each iteration of the space carving algorithm removes (or carves) voxels from a voxmap, and converges towards a voxel representation of the viewed surface. Therefore, there is only one requirement of the chosen starting point for the algorithm: *the voxmap at the beginning of the first iteration must completely enclose the viewed surface.* As with most iterative processes, ideally, the starting point should be as close to the final solution as possible, and the consequences of choosing a starting surface which is too large are two-fold:

- Each iteration only considers the voxels on the surface of the voxmap, and is therefore
limited to selectively (based on photo-consistency) carving these voxels. The first consequence of choosing a starting voxelmap which is too large is clear: the larger the starting voxelmap, the more iterations before the algorithm converges;

- The second consequence is less obvious, but considered in detail in [70]: the risk of converging to a local minimum. A local minimum, in this case, is a surface representation which is (depending on the reliability of the photo-consistency test), completely consistent with the input images, whilst not necessary even close to the viewed surface (see figure 5.15).

The visual hull is a good starting point for the space carving algorithm. Clearly it fulfills the requirement that it completely encloses the viewed surface, and in many cases (especially given a long image sequence circumnavigating the surface) it is a good representation of the surface. Further, it has been seen that a voxel representation of the visual hull can be efficiently constructed from the image silhouettes.

5.3.3 Improved photo-consistency test

In [68], the following simple photo-consistency test is proposed: project the centre of each voxel into each image and compare the intensities of the corresponding pixels. If the intensities differ by less than a threshold then the voxel is considered photo-consistent. Clearly, this test is insufficient in the presence of image noise:

- Consider one pixel in the image which is corrupted by noise and hence the intensity is random. As the noisy image is not consistent with the 3-D geometry, the photo-consistency constraint will fail for any point on a ray back-projected from the erroneous image point;

- The test is also prone to errors from the spatial sampling noise inherent in the voxelization of space. As the 3-D points must all lie on a grid, no surface can perfectly reconstruct the input images, and hence no surface is completely photo-consistent with the images.
A “shift transform” is introduced in [69] to solve the second of these problems: the corresponding pixels are compared with the intensities in a fixed-size patch around the projected voxel centre. However, it is noted that a bias in the reconstructed model is introduced: whilst points are less likely to be labelled as non-photo-consistent due to image noise, some points will be erroneously labelled as photo-consistent and hence the resulting model will be larger and smoother than the correct reconstruction.

An improved photo-consistency test follows. Considering a surface voxel, a parameterised surface patch is fitted locally. This enables a mapping to be defined between any two images of this voxel, and hence the surface-induced transfer test proposed in section 4.2.2 can be used.

Photo-consistency constraint

This surface-induced mapping improves the photo-consistency constraint in two fundamental ways. First, as the surface is locally parameterised, the problems of sampling space are no longer significant. A pixel in one image is mapped to a point in the second image with sub-pixel accuracy. Second, as a cross-correlation score can now be applied, the photo-consistency test is tolerant to image noise, and invariant to lighting changes (which are characterised by an affine change in the intensity over the entire image).

In order to define a surface-induced mapping between the images, it is necessary to fit a local surface patch. Two natural choices of parameterized surface are available: fitting a plane locally to the surface of the voxmap induces a homography between any two images; fitting a quadratic patch to the surface induces also induces an algebraic mapping (see section 2.2.3) between the images.

Algorithm. The algorithm is as follows:

- For each surface voxel, \( \mathcal{V} \), make a set of 3-D points containing
  - The centroid of \( \mathcal{V} \);
  - The centroids of each the 26 neighbours of \( \mathcal{V} \) if they are themselves on the surface of the voxmap;
• Fit a quadric surface to this point set;

The quadric-induced mapping then maps points between any two images of this region of the surface, and a sub-pixel cross-correlation score (see equation (4.5)) can be applied to classify the voxel.

5.3.4 Indicator function

Figures 5.16 and 5.17 demonstrate a further problem with the traditional space carving algorithm. Although the first few iterations of the algorithm remove large numbers of voxels, as the algorithm progresses, much of the surface is found to be photo-consistent and is not removed. The result is that each photo-consistent voxel is checked during each iteration of the algorithm.

Further, as the space carving is initialised from the visual hull, regions which are accurately modelled by the visual hull will already be consistent with the images and no further carving is necessary (this is clearly seen in figure 5.18, where convex regions of the skull are correctly modeled by the visual hull).

An improved algorithm maintains a cache of “consistent” voxels, which ensures that a voxel is only checked once. Thus, each surface voxel is passed through an indicator function, by applying the photo-consistency test, to determine regions of the surface which must be carved away:

• Apply photo-consistency test to all surface voxels and mark those that are inconsistent;

• Remove all inconsistent voxels, and repeat test for the newly revealed voxels.

Regions which are photo-consistent from the beginning, or become consistent after a few iterations of the algorithm, are therefore only tested a few times, rather than on every iteration.

Two implementational points should be noted. Firstly, as the photo-consistency test proposed above locally fits a surface patch to the voxel map, removing a single voxel would have an effect on the fitted patch of all neighbouring voxel: the solution is to re-apply the
consistency test to all neighbouring voxels on removal of a voxel. Secondly, removal of a voxel might have an effect on the occlusion of voxels on a completely different area of the surface: this problem is solved by testing all surface voxels from time to time (generally every 5 iterations of the algorithm).

5.4 Gallery of results

The following figures are presented as results of the models generated using the algorithms described in this section. The voxmap is represented as a triangulated mesh, smoothed and texture-mapped:

- The marching triangles algorithm is used to convert a voxmap to a set of triangles in space;

- A simple gaussian-smoothing algorithm for visualisation purposes:
  - A plane is fitted to each vertex, \( V \), and its closest neighbours;
  - \( V \) projected onto the plane.

- The model is texture-mapped from the input images:
  - For each face in the mesh, the most fronto-parallel image in the sequence is chosen;
  - The texture from this image is projected onto the face.

Figure 5.18 depicts the various stages of the algorithm: the visual hull is computed from the image silhouettes, and then space carving is used to reconstruct the concavities. Figures 5.19, 5.20, 5.21 and 5.22 demonstrate further examples.

5.5 Summary

A general reconstruction algorithm, capable of reconstructing arbitrary unknown surfaces from images sequences has been presented in this chapter. It has been seen that the visual hull can be efficiently reconstructed from the silhouettes, and this is a good starting point
for the space carving algorithm which can then recover regions of the surface (such as concavities) which are not accurately modeled by the visual hull.

In summary, the algorithm proceeds as follows:

- Compute projection matrices from image data (see chapter 7);
- Compute an octree representation of the visual hull using the image silhouettes;
- Compute a complete voxel representation of the surface using the space carving algorithm, with an improved photo-consistency constraint.
Figure 5.8: An example image (left), the extracted silhouette image (center), and a distance map (right) where contour lines indicate a constant distance to the silhouette. The silhouette and distance map are computed in constant time from the input image.

Figure 5.9: A more efficient inclusion test for a voxel than that depicted in figure 5.5. A bounding circle (yellow) is defined around the projected voxel, and knowing the signed distance (negative distances for points inside the silhouette) from the center of this circle (blue) to the image silhouette (purple) gives a quick test to determine if the voxel lies inside, outside or spanning the silhouette. (a) The voxel is correctly labelled as lying outside the visual hull. (b) The voxel is incorrectly labelled as spanning the surface for the visual hull.
Figure 5.10: The voxmap representation at three different resolutions of the visual hull for the cup sequence of figure 5.4. From left to right, the octree is shown at a maximum resolution of $32^3$, $64^3$ and $256^3$ voxels.

Figure 5.11: Six from a sequence of 36 images of a toy dinosaur mounted on a turntable. The visual hull is reconstructed from the silhouettes and shown up to a voxmap resolution of $256^3$. 
Figure 5.12: Six from a sequence of 30 images of a gourd. The visual hull is shown before (left) and after texture-mapping (right). The “body” of the gourd, is reconstructed up to a resolution of $256^3$ voxels, whilst the stalk is reconstructed as a separate $128^3$ voxmap.
Figure 5.13: Voxmap generation from silhouettes under four different levels of added silhouette segmentation noise. From top to bottom, 0, 200, 400, 800 and 1200 “specks” of noise are added at random to each of the 36 silhouette images. On the right, the resulting voxmap is shown: red voxels are erroneously labelled.
Figure 5.14: Six images from a sequence of 120 images of a vase from the Beazley Archive (Ashmolean Museum, Oxford). The silhouette is badly segmented (note around the base in the first and fourth image, for example), but the visual hull is accurately represented by the voxmap.

Figure 5.15: Two sets of surfaces which project to give the same images. Both surface configurations are photo-consistent with the input images, but differ significantly.
Figure 5.16: The skull model after 0, 3, 6, 9, 12 and 15 iterations of the space algorithm. Notice that only areas around the eye sockets and nose are carved, whilst regions around the forehead, which are accurately represented by the visual hull, do not change much.

Figure 5.17: Number of voxels remaining in the voxmap as the space carving algorithm is applied to the skull sequence of figure 5.16. Once again, notice that most of the carving is done over the first few iterations.
Figure 5.18: (a) A voxelated model of the visual hull of a skull. It can be seen that the visual hull “smoothes over” concavities such as the nose and eye sockets. (b) A photometric consistency function is applied to the surface of the visual hull, and regions with high score are shown in red and those with low score are shown in black. A high score indicates that the reprojection of the surface is consistent with the input images. It can be seen that the eyes and nose regions have been marked as “incorrect”. (c) The final model after applying the space carving algorithm. After space carving, areas such as the eye socket and nose region are correctly reconstructed. A simple mesh smoothing algorithm has been applied to the surface to highlight these areas, but the model is stored as $128^3$ voxels. (d) A textured-mapped model using intensities from the original images.
Figure 5.19: A reconstructed texture-mapped model of a dinosaur with six images out of the full sequence of 36. The model is also shown in figure 1.1. The model is generated from a $256^3$ voxmap.

Figure 5.20: The vase from figure 5.14 reconstructed and texture-mapped. All 120 images are used in the reconstruction, and a the voxmap is generated at a resolution of $512^3$. 
Figure 5.21: Six images from a sequence of 36 of a mug, and the reconstructed model. A voxmap resolution of $256^3$ is used.
Figure 5.22: Six images from a sequence of 72 of a model head. The model, from a $256^3$ voxmap, is shown. The vertical lines on the head are due to changes in contrast between the images used for texturing, and not artifacts in the model.
Chapter 6

Optimal Surface Reconstruction

Section 5 presented a robust and efficient algorithm for reconstructing a general surface from both silhouette and texture information. The algorithm payed particular attention to the different constraints provided by image silhouettes and surface texture, and showed how the complementary sources of information can be used to efficiently generate a “photo-consistent” model, given a sequence of views of an object.

However, whilst the space carving algorithm is, broadly speaking, providing the vision community with some of the best models (at least subjectively) from general image sequences, it suffers from one significant drawback: the necessity to represent the underlying surface as a voxmap. The obvious result is that the quality of any reconstructed model is strictly limited by the resolution of the voxmap; the computation time of the algorithm being itself linked to the chosen voxmap resolution.

This chapter considers the problem of “optimal” reconstruction of a viewed surface. That is, reconstruction with the aim of minimising reprojection errors.

Section 6.1 introduces the concept of optimal surface reconstruction, and an image silhouette error measure is introduced (section 6.1.1). This is contrast with currently available texture error measures (section 6.1.2).

It is then shown in section 6.2 that the minimisation of this new silhouette error metric can be re-expressed as a surface evolution. The evolution is demonstrated by embedding a surface as a level-set (section 6.4).
6.1 Overview

Consider a set of images of a surface. The optimal (in a statistical sense) reconstruction of this surface, is the surface which, projected into each image, reproduces the input images exactly. More precisely, the optimal surface reconstruction minimises the difference between the projected and measured image data, according to the noise model. The problem lies in defining a robust error measure: that is, a measure of the “difference” between an input image and an image generated by reprojecting the hypothesized model.

Two such error measures have been considered in chapter 4: silhouette and texture information. Their applicability to an optimal surface reconstruction algorithm are summarized here.

6.1.1 Silhouette errors

An optimal reconstruction of an imaged surface can be projected into each image to generate a silhouette. Comparing this to the actual measured silhouette in the image gives the required error measure. That is,

$$E_s = \sum_{i=0}^{N} D \left( \mathcal{T}_i, \mathcal{T}'_i \right),$$

where $\mathcal{T}_i$ and $\mathcal{T}'_i$ are the measured and reprojected silhouettes respectively in the $i$th image, and the summation is taken over each of $N$ images. The function $D \left( \mathcal{T}_i, \mathcal{T}'_i \right)$ defines a “distance” between the two silhouettes $\mathcal{T}_i$ and $\mathcal{T}'_i$. Clearly, if the reconstructed surface is optimal, then $\mathcal{T}_i = \mathcal{T}'_i$, and $E_s = 0$. It should be noted that in practice, due to image noise, this is never achieved exactly.

6.1.2 Texture errors

Texture information can be treated in a similar fashion. Consider a point, $\mathbf{X}$, on a reconstructed surface, $\mathcal{S}$. Further, consider two views, $i$ and $j$, of the surface in which $\mathbf{X}$ is not occluded. In these views $\mathbf{X}$ projects as $\mathbf{x}_i$ and $\mathbf{x}_j$.

If the reconstruction, $\mathcal{S}$, is perfect, then $\mathbf{x}_i$ and $\mathbf{x}_j$ will be a corresponding pair of points in the two images, and their intensity values, $I_i(\mathbf{x}_i)$ and $I_j(\mathbf{x}_j)$, will match. Again, as the
images are noise, this is never achieved exactly, and an error metric should be defined and minimised.

Further, the surface induces a mapping between a point in the first image and a point in the second (see section 4.2.2) such that $M_{ij}(x_i) = x_j$. Taking a step, $v$, from $x_i$ in the first image and mapping to the second image gives a new point $x'_j = M_{ij}(x_i + v)$: the points $x_i + v$ and $x'_j$ should also have corresponding intensity values. Indeed, the intensity values should correspond over a $p \times q$ patch, and can be compared with a standard cross-correlation approach. This can be thought of as estimating both surface shape (here, $X$) as well as the surface albedo or BRDF.

### 6.1.3 Summary

It has been suggested (chapter 4) that texture information and silhouettes are complementary sources of information and should be used together. For example, if a texture-less area of a surface lies on one of the contour generators, it can be reconstructed from the silhouette information. In contrast, textured areas of a surface which do not lie on the visual hull, but are visible in two images can be reconstructed from the texture information.

Combining these two complementary sources of information about a reconstructed surface gives a global cost-function which, when minimised, provides an optimal reconstruction of the viewed surface:

$$E = \alpha E_s + \beta E_t,$$

where $\alpha$ and $\beta$ are weightings reflecting the statistical variance of the two measurements.

The optimal surface reconstruction, $S_o$, is given by

$$\min_{S_o} \alpha E_s + \beta E_t.$$

### 6.2 Optimal Surface Reconstruction From Texture Information

Practically speaking, minimising a texture-based cost function (such as equation (4.5)) directly is not possible. Parameterising and regularising the reconstructed surface is one
problem: typically a surface is represented by in the order of $10^3$ parameters and a non-linear minimisation is almost inconceivable.

However, recent work of Faugeras and Keriven [37, 38] has shown that a minimisation of equation (4.5) can be rewritten in the form of a PDE. It then becomes a simple process of iterating from a starting point towards the optimal reconstruction.

### 6.3 Optimal Surface Reconstruction From Silhouettes

In order to minimise the silhouette reprojection errors, it is necessary to define a suitable error metric.

#### 6.3.1 Silhouette reprojection errors

Consider an image sequence of images of a surface. The projection matrix for each image is known. Given a 3-D point in space, $\mathbf{X}$, the following error measure,

$$e_i(\mathbf{X}) = D(\mathbf{x}_i, \mathcal{T}_i),$$

returns the distance between the projection of this point in the $i$th image, $\mathbf{x}_i = P_i \mathbf{X}$, and the silhouette, $\mathcal{T}_i$, in this image. The function $e_i(\mathbf{X})$ is defined such that $|e_i(\mathbf{X})|$ is the minimum euclidean distance from the projection of $\mathbf{X}$ to $\mathcal{T}_i$, and $e_i(\mathbf{X})$ is negative if $\mathbf{x}_i$ lies inside and positive if it lies outside the image silhouette: it is a signed distance function.

A general point in space either lies inside, on the surface, or outside the visual hull, $\mathcal{V}$:

- If $\mathbf{X}$ lies inside the visual hull, by definition it projects inside the image silhouette in every image. Therefore $e_i(\mathbf{X} \in \mathcal{V}) < 0$ for all $i$.

- If $\mathbf{X}$ lies outside the visual hull, it must project outside the image silhouette in at least one image: $e_i(\mathbf{X} \notin \mathcal{V}) > 0$ for some $i$.

- If $\mathbf{X}$ lies on the surface of the visual hull, it projects onto the silhouette in one (or more) images, and inside the silhouette in all other images: $e_i(\mathbf{X}) = 0$ for one or more $i$, and $e_i(\mathbf{X}) < 0$ otherwise \(^1\).

\(^1\)If $e_i(\mathbf{X}) = 0$ for two or more images, $\mathbf{X}$ is a frontier point, as it lies on the contour generator in at least two images (see section 4.1.4)
The aim is to define an overall error metric which reconstructs the visual hull. That is, all points, \( \mathbf{X} \), should lie on the visual hull. Combining the information from all images, the following cost function is defined:

\[
E(\mathbf{X}) = \max_i e_i(\mathbf{X}) .
\] (6.1)

Ignoring the sign of \( E(\mathbf{X}) \) for the moment, the following cases can be examined:

- For a point outside the visual hull, \( E(\mathbf{X}) \) returns the largest image distance between any silhouette and the corresponding projection of \( \mathbf{X} \). This can be visualised as the minimum distance (in an image) that \( \mathbf{X} \) would have to move in order for it to lie on the visual hull—the (image) distance of \( \mathbf{X} \) from the visual hull;

- For a point inside the visual hull, \( E(\mathbf{X}) \) returns the smallest image distance between any silhouette and the corresponding projection of \( \mathbf{X} \). This can be visualised as the maximum distance (in an image) that \( \mathbf{X} \) can be moved for it to remain inside the visual hull—the (image) distance of \( \mathbf{X} \) from the visual hull;

- Finally, \( E(\mathbf{X}) = 0 \) for any point on the visual hull.

Thus, in each case, \( E(\mathbf{X}) \) gives the image distance between any point \( \mathbf{X} \) and the visual hull.

### 6.3.2 Evolution equation

In general, minimising equation (6.1) over a complete surface is difficult, as the number of unknown parameters is very large. However, it is possible to re-express this minimisation as an evolution equation and this can be used as the basis for a level-set formulation later in the chapter.

It is clear that given the silhouette constraint alone, the evolution of a surface should proceed as follows: any point inside the visual hull should be pulled “outwards”, whilst any point outside the visual hull should be pushed “inwards”. Thus, the surface would converge towards the surface of the visual hull. This evolution is described by

\[
\frac{d\mathbf{X}}{dt} = -E(\mathbf{X}).\mathbf{N},
\] (6.2)

where \( \mathbf{N} \) is the outward-facing surface normal at \( \mathbf{X} \) and \( E(\mathbf{X}) \) is given by equation (6.1).
6.3.3 Evolution from silhouettes and texture

Consider a surface with both concave and convex regions.

In the convex regions, the visual hull accurately represents the surface, and hence the cost function of equation (6.1) is small. The surface evolution (equation (6.2)) does not move these regions.

In a concave region, the visual hull does not accurately represent the underlying surface, and the cost of equation (6.1) is non-zero (negative, as the points lie inside the visual hull). The evolution of equation (6.2) would (incorrectly) tend to pull these regions outwards towards the visual hull. In the absence of further reconstruction constraints, this is the best solution. However, if stereo data (such as a texture correlation cost) is available, a better solution would to allow this to direct the evolution in concave regions, as the silhouette provides no information about the surface in this case.

An improvement on the cost function of equation (6.1) is therefore given by

\[ C_{\text{silhouette}}(X) = \max (E(X), 0), \]

with the following evolution equation:

\[ \frac{dX}{dt} = -C_{\text{silhouette}}(X)N, \]

Thus, for any point outside the visual hull, \( C_{\text{silhouette}}(X) \) is positive, increasing with projected distance from the silhouettes (and hence distance from the visual hull): the points are pulled inwards towards the visual hull. However, for any point inside the visual hull, \( C_{\text{silhouette}}(X) = 0 \), and therefore the point does not move under the influence of the image silhouette constraints.

6.4 Level-Set Representation of a Surface

Equation (6.1) provides a representation of the visual hull as a function of any point in 3-D space. The zeroth level-set of this function gives a representation for the visual hull, and thus an algorithm such as Marching Cubes [74] or Marching Triangles [56] could be used to generate a smooth triangulated representation of the visual hull.
However, a similar cost function for any point in space based on the information provided by correlated imaged surface textures is not available, as the correlation requires local surface normals and (in some cases) curvatures to be known (see section 4.2): these can not be estimated without a local parameterisation of the surface. Thus, it was shown by Faugeras and Keriven [38], that by representing the surface as a level-set, second-order surface primitives such as these can be reliably extracted if the surface is represented as a level-set.

It was seen in section 5.1 that a level-set representation of a surface has a two main advantages:

- Arbitrary topology surfaces can be represented accurately;
- Surface parameters, such as the normal and curvature, are implicitly encoded.

### 6.4.1 Level-set embedding

Consider a closed curve in 2-D or a surface in 3-D, $S$. A level-set representation of $S$ embeds an $N$-dimensional surface as the $d$th level-set of an $N + 1$-dimensional function, $\Phi$.

That is, given any point, $\mathbf{x} = (x, y)$ on a 2-D curve, the function $\Phi(x, y)$ is defined such that $\Phi(\mathbf{x}) = d$. Similarly, in 3-D, $\Phi(x, y, z)$ is defined such that, for any point $\mathbf{X} = (x, y, z)$ on $S$,

$$
\Phi(\mathbf{X} \in S) = d \ .
$$

(6.4)

Typically, for convenience, $d = 0$: $S$ is given by the zeroth level-set of $\Phi$.

Generating $\Phi(x, y, z)$ is problem-dependent, but a common choice is to define

$$
\Phi(\mathbf{X}) = D(\mathbf{X}, S) \ ,
$$

(6.5)

where $D(\mathbf{X}, S)$ is the signed distance from $\mathbf{X}$ to $S$. This distance function is defined such that $D(\mathbf{X}, S)$ is negative for points inside $S$ and positive for points outside $S$. Either the euclidean or an algebraic distance can be used.
6.4.2 Geometric properties of $S$

It follows from the definition of $\Phi$ in equation (6.4) that the unit normal vector, $\mathbf{N}$, to $S$ at any point on $S$ is given by

$$
\mathbf{N} = \frac{\nabla \Phi}{|\nabla \Phi|} .
$$

The surface curvature of $S$ can also be found from the divergence of the unit normal vector:

$$
\kappa = \nabla \cdot \mathbf{N} = \frac{\Phi_{xx} \Phi_y^2 - 2\Phi_x \Phi_y \Phi_{xy} + \Phi_{yy} \Phi_x^2}{(\Phi_x^2 + \Phi_y^2)^{3/2}} ,
$$
in the case of a 2-D level-set.

6.4.3 Level-set evolution

Consider now the problem of embedding an evolving surface, $S(t)$, changing with time. This is of particular interest as it permits a surface to propagate under the influence of an external force such as, in this case, image data.

Rewriting equation (6.4), $\Phi(\mathbf{X}, t)$ is now defined such that at any time, $t$, the underlying time varying surface $S(t)$ is given by the $d$th level-set of $\Phi$:

$$
\Phi(\mathbf{X} \in S(t), t) = d .
$$

(6.6)

It is assumed, imposing no restrictions on the surface propagation, that under the influence of a set of both internal (functions of properties such as curvature and position) and external forces, a point, $\mathbf{X}$, on $S$ propagates with speed $F$ along the outward-facing surface normal, $\mathbf{N}$: that is, $\mathbf{X}_t = F\mathbf{N}$. Differentiating equation (6.6) gives

$$
\frac{\partial \Phi}{\partial t} + \nabla \Phi \cdot \mathbf{X}_t = 0 ,
$$

and hence

$$
\frac{\partial \Phi}{\partial t} = -F|\nabla \Phi| .
$$

(6.7)
This is the propagating level-set equation of Osher and Sethian [86, 101, 102].

Note that in general, computation of \( F \) for points other than those on the \( d \)th level-set (i.e. on the underlying surface) has no geometric meaning. However, as equation (6.7) is defined for any point on the \( d \)th level-set, the propagation is valid however \( F \) is defined. It is often convenient, but not essential, to define \( F \) such that the distance characteristics of points off the \( d \)th level-set of equation (6.5) is maintained for all \( t \): this is discussed in detail by Gomes and Faugeras [47], and the specifics are omitted.

### 6.4.4 Advantages of the level-set representation

Figures 6.1 and 6.2 clearly demonstrate the advantage of representing a surface as a level-set. As the curvature and surface normal is encoded even in a discrete sampled level-set, the model is smoother and it requires a lower resolution sampling to generate a model of similar quality.

### 6.4.5 Discrete level-set formulation

In practice, a level-set is sampled on a regular-space grid of points. That is, the level-set \( \Phi(x, y, z) \) is represented by the discrete function \( \Phi_d(i, j, k) = \Phi(x_0 + i\Delta x, y_0 + j\Delta y, z_0 + k\Delta z) \), where \( (x_0, y_0, z_0) \) is a fixed point, and \( \Delta x, \Delta y \) and \( \Delta z \) defined the constant sample resolution. The parameters \( i, j \) and \( k \) are integers.

Much detail is available in [101] regarding the robust computation of the surface normal, \( \mathbf{N} \), and the surface curvature, \( \kappa \), from a discrete sampled level-set. It is shown that central-weighted differentials are less likely to propagate errors. Thus, the surface normal, for example, is given by

\[
\mathbf{N} = \frac{1}{2} \begin{bmatrix}
\Phi_d(i + 1, j, k) - \Phi_d(i - 1, j, k) \\
\Phi_d(i, j + 1, k) - \Phi_d(i, j - 1, k) \\
\Phi_d(i, j, k + 1) - \Phi_d(i, j, k - 1)
\end{bmatrix}.
\]

Similarly, an expression for \( \kappa \) can be derived.
6 Optimal Surface Reconstruction

Figure 6.1: Three representations of a curve (green) sampled on a $20^2$ grid. The left image represents the curve as a voxelmap (black). Centre: the voxelmap is “triangulated” using a 2-D marching cubes algorithm [74] and is shown in (red). Representing the curve as a level-set, sampled again on a $20^2$ grid gives a much better representation (right).

Figure 6.2: A sphere represented as a voxelmap (left), a triangulated voxelmap (centre) and a level-set (right). In each case, the sphere is sampled on a $40^3$ grid. The triangulated voxelmap is generated using the marching cubes algorithm.

6.5 Examples

Figures 6.3 and 6.4 shows a surface propagating under this silhouette-based evolution equation. It can be seen that the surface rapidly converges to the visual hull. In general, about 50-60 iterations are required, taking about 4-5 minutes to converge on a typical workstation, sampling the level-set on a $100^3$ grid.

6.6 Future Work and Conclusions

This chapter has presented a surface evolution equation which minimises the reprojection error given silhouette information alone. It should be noted that a similar texture correlation minimisation is also available from the work of Faugeras and Keriven [38]. The combination of these two cost functions remains for future work, but is outlined here.
Figure 6.3: The reconstructed surface of a toy dinosaur as it propagates under the influence of the silhouette constraints. The level-set is sampled at a resolution of $100^3$, and converges after 50 iterations. Note the improvement in quality of the final model as compared with that represented as a voxmap in figure 5.11.
Figure 6.4: The reconstructed surface of a cup as it propagates under the influence of the silhouette constraints. The level-set is sampled at a resolution of $100^3$ and converges after 60 iterations. Note the change in topology as the handle is reconstructed. Compare with the voxmap equivalent in figure 5.10.
Equation (6.2) provides a silhouette information evolution equation for any point, \( \mathbf{X} \), on the surface

\[
\frac{d\mathbf{X}}{dt} = -C_{silhouette}(\mathbf{X}).\mathbf{N},
\]

where \( \mathbf{N} \) is the surface normal at \( \mathbf{X} \). Similarly, Faugeras and Keriven [38] present a corresponding evolution equation from texture information alone in the form

\[
\frac{d\mathbf{X}}{dt} = -C_{texture}(\mathbf{X}).\mathbf{N}.
\]

A natural combination of these two approaches might be considered:

\[
\frac{d\mathbf{X}}{dt} = -C'(\mathbf{X}).\mathbf{N}
\]

where

\[
C' = \alpha C_{silhouette} + \beta C_{texture},
\]

with the scalars \( \alpha \) and \( \beta \).

Alternatively, an approach similar to chapter 5 might be adopted: the silhouette information might be allowed to evolve the surface to convergence as the visual hull. From this point, the texture correlation constraint (which is slower to compute as less robust) might be allowed to evolve the surface further.

Either approach has significant advantages over the reconstruction from one information source alone:

- convergence is accelerated;
- convergence to a local minimum is less likely.

In both cases, it is possible to reconstruct an optimal surface from the available image information.
Chapter 7
Recovering Camera Motion

Throughout chapters 2–6, it has been assumed that the projection matrices for the image sequences under consideration are known (and hence both the camera motion and internal calibration have been found). Previous reconstruction systems have similarly relied on knowing these parameters apriori, but in most cases the systems are either pre-calibrated with known camera motion and calibration [84], or (more often) carefully constructed artificial cues (such as a Tsai grid [126]) are inserted into the scene in order to permit pseudo-automatic calibration [15, 112].

The general case of computing camera motion and calibration from point correspondences over multiple views is considered initially in [36] and more recently in [51]. In general it has been shown that each camera is specified by 11 parameters [55, Section 6.1, page 167], and hence over m views there are 11m degrees of freedom. A simplification generally made, and assumed here, is that the scene is static, and the camera internal parameters (zoom, focus, aspect ratio..) are unchanging over the sequence. Thus the system is reduced to 6m + 5 degrees of freedom, noting that the camera defined by a full 5 degrees of freedom [55, Section 5.1, page 143]. Automated systems for considering such image sequences are proposed in [8, 50, 82, 121].

7.1 Camera Motion and Calibration from Turntable Sequences

This section looks at the geometry of recovering these parameters if the motion over the sequence is constrained to single-axis rotation (turn-table sequences): it is shown that such an assumption reduces the ambiguity to m + 8 (naturally, this is a significant advantage over the unconstrained solution). Some of the results are summarized from [42],
which itself is built upon incomplete formulations of reconstruction from turn-table sequences [63, 114, 118], whilst extensions are proposed to reduce the inherent ambiguities in the reconstruction by making sensible assumptions about the camera internal parameters.

### 7.1.1 Geometry of turntable sequences

Having the freedom to choose the world coordinate system ensures that the first camera projection matrix can be expressed as

$$ P_0 = H [I \mid t] , $$

where $H$ is a $3 \times 3$ homography representing the world camera internal parameters and rotation about the camera centre, and $t = (t, 0, 0) \top$. Clearly, the camera is centered on the point $t$. A rotation of the camera by $\theta$ about the $z$-axis is achieved by post-multiplying $P_0$ by

$$ \begin{bmatrix} R_2(\theta) & 0 \\ 0 & 1 \end{bmatrix} , $$

giving

$$ P_{\theta} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 & t \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \quad (7.1) $$

The homography $H$ is made up of the column vectors $h_i, i = 1...3$.

It is important to note the following:

- the homography $H$ is unchanged through the sequence;
- the radius, $t$, is also unchanged through the sequence, and can be chosen arbitrarily as the reconstruction is subject to at least a scaling of space (see below);
- hence $H$ has 8 degrees of freedom (9 homogeneous elements) and a sequence of $m$ images as $m + 8$ degrees of freedom.
7.1.2 Projective ambiguity

It is shown in [36, 54] that reconstruction from images, having no further information about the calibration of the cameras, is only possible up to an unknown projective transformation of 3-space. This is clear from observing that under any \(4 \times 4\) invertible transformation \(T\), a point maps from \(X\) to \(TX\) whilst camera projection matrices map from \(P\) to \(PT^{-1}\). Thus, the projected image point \(x = PX = PT^{-1}TX\) does not change for a projective transformation of 3-space.

The special case of single axis rotation is clearly a sub-set of the case of general motion. In practise, it can be shown [42, Section 2.4] that a single-axis rotation is only ambiguous up to a euclidean transformation (which may be ignored) and a projective transformation given by

\[
T^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \alpha & 0 \\
0 & 0 & \beta & 1
\end{bmatrix},
\]

where \(\alpha\) and \(\beta\) are arbitrary scalars.

The family of projection matrices is therefore given by

\[
P_\theta' = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{bmatrix}
\cos \theta & \sin \theta & 0 & t \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \alpha & 0 \\
0 & 0 & \beta & 1
\end{bmatrix}. \quad (7.2)
\]

Under this projective ambiguity, points, given by \(X\), transform according to \(X' = TX\), or

\[
X' = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1/\alpha & 0 \\
0 & 0 & -\beta/\alpha & 1
\end{bmatrix} X,
\]

which shows that the ambiguity is expressed as a 1-D projective transformation in the \(z\)-direction (i.e. the axis of rotation). There is no ambiguity in the angles of rotation, \(\theta\). Figure 7.1 depicts examples of this ambiguity.
Figure 7.1: **Projective ambiguity:** With no apriori information about the internal parameters of the camera or about the external geometry of the scene, there is a 1-D projective transformation in the $z$-direction. Different choices of $\alpha$ and $\beta$ in equation (7.2) give very different reconstructions, all reprojecting into the original images correctly.

### 7.1.3 Reducing projective ambiguity

Equation (7.2) can be rewritten in the following form:

$$P'_\theta = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & \beta t \\ 0 & 1 & 0 \\ 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 & t \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} .$$  \hspace{1cm} (7.3)

This is useful, as it demonstrates that only the calibration homography, $H'$, depends on the chosen projective frame. Indeed, the family of homographies is extracted from equation (7.3) permitting $P'_\theta$ to be re-expressed in the form of equation (7.1), with

$$H' = \begin{bmatrix} h'_1 & h'_2 & h'_3 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & \beta t h_3 + \alpha h_1 \end{bmatrix} .$$  \hspace{1cm} (7.4)

Equations (7.4) and (7.3) suggest the following points:

- The image-point $h_1 = h'_1$ is the image of the origin, $(0, 0, 0)^\top$, in 3-D, and this does not depend on the chosen projective frame;

- The image-point $h'_3$ is the image of the homogeneous point $(0, 0, 1, 0)^\top$ in the respective projective frame: the vanishing point of the $z$-axis;

- The final column of $H'$ is dependent on the chosen projective frame under the constraint that $h'_3$ must lie along the line $h_1$ to $h_3$.

Summarizing, in order to reduce the ambiguity from a 1-D projective transformation along the axis of rotation, the last column of $H$ must be chosen according to some further apriori information—either known geometry constraints in the scene, or known internal parameters of the camera.
It is clear from equation (7.4), and the notes that follow it, that the projective ambiguity is strongly tied to the vanishing point of the z-axis, which, if known, significantly simplifies the problem.

The following section demonstrates how this assumption can be used, but demonstrates that a further constraint is required in order to reduce the ambiguity completely. Example results are given.

**Fronto-parallel constraint**

One possible, and sensible, assumption is that the vanishing point of the z-axis is at, or close to, infinity. This is equivalent to assuming that the image plane is parallel to the axis of rotation. In practise this is a very common situation, particularly in turn-table sequences as it ensures that most of the surface of any object on the turn-table is seen. Reconstruction close to \( \mathbf{h}_3 \) is likely to be unstable as the visual flow is at a minimum, and thus it is desirable for \( \mathbf{h}_3 \) to be as far outside the image as possible.

In homogeneous coordinates let \( \mathbf{h}_3' = (x, y, 0)^\top \). Define the line, \( \mathbf{l}_s = \mathbf{h}_1 \times \mathbf{h}_3 \), the line joining \( \mathbf{h}_1 \) and \( \mathbf{h}_3 \), and the image of the rotation axis (z-axis). As \( \mathbf{h}_3' \) must lie on \( \mathbf{l}_s = (l_{sx}, l_{sy}, l_{sz})^\top \), it follows that

\[
\mathbf{h}_3' = \gamma \begin{bmatrix}
-l_{sy} \\
l_{sx} \\
0
\end{bmatrix} = \gamma \mathbf{v} ,
\]

where \( \gamma \) is an unknown scale-factor, and hence equation (7.4) becomes

\[
\mathbf{H}' = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \gamma \mathbf{v} \end{bmatrix} .
\]  

**Degeneracy** As an aside, it is worth considering the possible degeneracies of this equation. If \( \mathbf{h}_1 \) lies at infinity then it is clear that \( \mathbf{H}' \) will become singular under this constraint. This situation is only possible if the camera plane lies perpendicular to the axis of rotation which is in direct conflict with the assumption made at the beginning of this section.

As expected, we have reduce the ambiguity from a 2-parameter family to a single parameter family. A further constraint is needed to reduce the ambiguity further. There are
no further sensible external constraints that can be applied in the general case, but internal camera parameter constraints are now considered.

**Internal camera constraints**

Knowledge of the internal camera parameters reduces the ambiguity in the reconstruction. Often constraints such as known aspect ratio, zero skew, or known (e.g. centre of image) principal point are used. The following section outlines how these contraints can be expressed in practise.

**Decomposing the projection matrix** The projection matrix, $P'_\theta$, can be written, from equation (7.3) as

$$P'_\theta = H'R_z = KTR_z$$

where $K$ is termed the camera calibration matrix \([55]\), an upper-triangular $3 \times 3$ matrix parameterising all the internal camera properties such as focal length, aspect ratio and principle point, $T$ is a $3 \times 3$ rotation matrix (orthonormality ensuring that $T^{-1} = T^\top$) parameterising the camera orientation with respect to the coordinate system defined in section 7.1.1, and

$$R_z = \begin{bmatrix}
\cos \theta & \sin \theta & 0 & t \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}.$$  

This decomposition is derived from the QR decomposition.

The components of $K$ are as follows:

$$K = \begin{bmatrix}
a_x & s & c_x \\
0 & a_y & c_y \\
0 & 0 & 1
\end{bmatrix} \tag{7.6}$$

where $a_x/a_y$ is the aspect ratio, $(c_x, c_y)$ is the principal point, and $s$ is the “skew”.

**Image of the absolute conic** The image of the absolute conic ([40, 39] and summarized in [55, Section 2.6, page 63 and section 7.5, page 198]), $\omega$, is a useful construction as it can be shown that $\omega = (PP^\top)^{-1} = (KK^\top)^{-1}$. 


From equation (7.3), it follows that

\[ \omega = \left( H' H'^\top \right)^{-1} = H'^{-\top} H'^{-1}. \]  

(7.7)

Writing the homography as \( H' = (h'_1, h'_2, h'_3) \), the inverse can be written in terms of the column vectors:

\[ H'^{-1} = \frac{1}{|H'|^2} \left[ h'_2 \times h'_3 \quad h'_3 \times h'_1 \quad h'_1 \times h'_2 \right]^\top. \]

Hence equation (7.7) can be expressed in terms of the image points, \( h'_1, h'_2 \) and \( h'_3 \):

\[ \omega = \left( h'_2 \times h'_3 \right) \left( h'_2 \times h'_3 \right)^\top + \left( h'_3 \times h'_1 \right) \left( h'_3 \times h'_1 \right)^\top + \left( h'_1 \times h'_2 \right) \left( h'_1 \times h'_2 \right)^\top. \]

Substituting \( h'_1 = h_1, h'_2 = h_2 \) and \( h'_3 = \beta h_3 + \alpha h_1 \) from equation (7.4) gives a biquadratic expression for \( \omega \) in terms of the unknowns, \( \alpha \) and \( \beta \):

\[
\omega = - [h_1]_x \left( \beta^2 t^2 h_3 h_3^\top + \alpha \beta t \left( h_3 h_1^\top + h_1 h_3^\top \right) + \alpha^2 h_1 h_1^\top \right) [h_1]_x \\
- [h_2]_x \left( \beta^2 t^2 h_3 h_3^\top + \alpha \beta t \left( h_3 h_1^\top + h_1 h_3^\top \right) + \alpha^2 h_1 h_1^\top \right) [h_2]_x \\
- [h_1]_x h_2 h_2^\top [h_1]_x .
\]  

(7.8)

The image of the absolute conic has five degrees of freedom (3 × 3 symmetric matrix up to scale), and thus it represents five equations relating the elements of \( \omega \) to the \( \alpha \) and \( \beta \). Two such equations, and therefore two constraints on the elements of \( \omega \) are sufficient to resolve the projective ambiguity in the reconstruction: the solution is given as the intersection of two conics (zero to four solutions), by rewriting equation (7.8) in the form:

\[ [\alpha \quad \beta t \quad 1] C_{ij} \begin{bmatrix} \alpha \\ \beta t \\ 1 \end{bmatrix} = 0, \]

where \( C_{ij} \) is a function of \( h_1, h_2 \) and \( h_3 \) as well as the \((i,j)\)th element of \( \omega \).

If, however, the fronto-parallel constraint of equation (7.5) is used to make the substitution \( h'_3 = \gamma v \), equation (7.4) now reduces to:

\[ \omega = \gamma^2 \left[ (h_2 \times v) (h_2 \times v)^\top + (v \times h_1) (v \times h_1)^\top \right] + (h_1 \times h_2) (h_1 \times h_2)^\top. \]

This now requires only one constraint to be placed on the matrix \( \omega \).
Equation (7.6) permits the image of the absolute conic, $\omega = (KK^\top)^{-1}$, to be written in terms of the individual camera parameters:

$$
\omega = \begin{bmatrix}
    a_y^2 & -a_y s & a_y(c_y s - a_y c_x) \\
    -a_y s & a^2 + s^2 & -s(c_y s - a_y c_x) - a_x^2 c_y \\
    a_y(c_y s - a_y c_x) & -s(c_y s - a_y c_x) - a_x^2 c_y & (c_y s - a_y c_x)^2 + a_x^2 c_y^2 + a_x^2 a_y^2
\end{bmatrix} .
$$  

(7.9)

**Imposing square pixels constraint**

In practice, the aspect ratio of a modern CCD array is generally close to unity\(^1\). Making this assumption, or that $a_x = a_y = a$ in equation (7.6), simplifies equation (7.9):

$$
\omega = \begin{bmatrix}
    a^2 & -a s & a(c_y s - a_x c_x) \\
    -a s & a^2 + s^2 & -s(c_y s - a_x c_x) - a_x^2 c_y \\
    a(c_y s - a_x c_x) & -s(c_y s - a_x c_x) - a_x^2 c_y & (c_y s - a_x c_x)^2 + a_x^2 c_y^2 + a_x^2 a_y^2
\end{bmatrix} .
$$  

(7.10)

**Imposing small skew constraint**

It is very rare that the skew, $s$, is non-zero\(^2\), and in general $a_x^2 \gg s^2$. The implication being that $a_x^2 + s^2 \approx a_x^2$ for most cameras:

$$
\omega = \begin{bmatrix}
    a_y^2 & -a_y s & a_y(c_y s - a_y c_x) \\
    -a_y s & a_x^2 & -s(c_y s - a_y c_x) - a_x^2 c_y \\
    a_y(c_y s - a_y c_x) & -s(c_y s - a_y c_x) - a_x^2 c_y & (c_y s - a_y c_x)^2 + a_x^2 c_y^2 + a_x^2 a_y^2
\end{bmatrix} .
$$  

(7.11)

**Imposing zero skew constraint**

Finally, by assuming the skew to be zero:

$$
\omega = \begin{bmatrix}
    a_y^2 & 0 & -a_y^2 c_x \\
    0 & a_x^2 & -a_x^2 c_y \\
    -a_y^2 c_x & -a_x^2 c_y & a_x^2 c_x^2 + a_x^2 c_y^2 + a_x^2 a_y^2
\end{bmatrix} .
$$  

(7.12)

**Examples**

The following three sections use the different constraints described above in order to calibrate the sequence of figure 7.2.

\(^1\) [55, Section 2.5.2, page 203] gives an example of a measured camera calibration matrix for a CCD camera where the aspect ratio, $a_x/a_y$ is seen to be 1.0096.

\(^2\) In the example of [55, Section 2.5.2, page 203], the ratio $a_x^2/s^2$ is in the order of $10^5$.  


Figure 7.2: **Cup sequence**: Four images from an image sequence (250 frames) of a cup on a turn table; used as an example sequence in the discussion on calibrating single-axis rotation sequences. Notice that the image plane is almost parallel to the axis of rotation: a useful cue in reducing the ambiguity in the reconstruction.

**Square pixels and zero skew**

Applying both the known aspect ratio and zero skew constraints is sufficient to reduce the reconstruction ambiguity. The following camera parameter matrix is thus obtained:

\[
K = \begin{bmatrix}
1236.7 & 0 & 308.3 \\
0 & 1236.7 & 408.6 \\
0 & 0 & 1
\end{bmatrix}.
\]

Further, the vanishing point of the \(z\)-axis, given by \(h_3 = (44.7, 6196.7)\) is found to be well outside the image as expected (the rotation axis is close to parallel to the image plane).

**Fronto-parallel, square pixels and small skew**

If the fronto-parallel constraint (i.e. the assumption that the image plane is parallel to the rotation axis), the ambiguity is reduced to a single parameter family of solutions. Imposing the assumption that the aspect ratio is known (one), and making the loose assumption that the skew is “small” (see above) results in the following camera parameter matrix:

\[
K = \begin{bmatrix}
1264.3 & 4.3 & 320.3 \\
0 & 1264.3 & 144.9 \\
0 & 0 & 1
\end{bmatrix}.
\]

It is clear that the small skew assumption is fair: in this case, \(a_x^2/s \approx 8 \times 10^4\).

**Fronto-parallel and zero skew**

Finally, imposing a zero skew constraint gives

\[
K = \begin{bmatrix}
1264.4 & 0 & 320.3 \\
0 & 1314.5 & 144.9 \\
0 & 0 & 1
\end{bmatrix},
\]
where it can be seen that although no constraint is imposed on the aspect ratio, it is close to one as is expected. It is worth noting, however, that if the rotation axis is either close to vertical, or close to horizontal, the zero skew constraint is not sufficient to fully constraint the calibration. In general, the zero skew constraint is not well-behaved [132].

Figure 7.3 shows the reconstructed model, imposing the fronto-parallel, zero skew and square pixel constraints.

### 7.2 Camera Motion from Epipolar Tangencies

#### 7.2.1 Introduction

The computation of epipolar geometry and trifocal geometry from image point correspondences is now well established [52, 122, 131]. The case of scenes containing smooth curves and objects, where point correspondences are not available, is less thoroughly investigated although iterative search based algorithms are available for two views of smooth objects [23] (reviewed below) and for multiple views of a space curve [11] or surface [1]. However, as yet there are no closed form solutions in the case of smooth surfaces.
7 Recovering Camera Motion

This section investigates scenes containing smooth objects \textit{and} a plane which may be used to register the views. The plane plus \textit{points} configuration has received significant attention in the past, not least because it arises frequently in everyday scenes. The image motion is decomposed into a planar homographic transfer plus a residual image parallax vector \cite{57, 67, 98, 105}. This decomposition has the advantage that it partially factors out dependence on the camera relative rotation and internal parameters. In this paper we show that similar advantages apply in the plane plus smooth object case. Furthermore there are additional advantages in that the multiple view geometry can be computed for smooth objects without entailing an iterative search.

For the plane plus smooth object configuration new solutions and uniqueness results are derived for the computation of the fundamental matrix (section 7.2.3), the trifocal tensor \cite{53, 104, 107} (section 7.2.4) and cameras for multiple (i.e. more than 3) views (section 7.2.5).

\textbf{Planar parallax}

The underlying parallax geometry is shown in figure 7.4. The world plane induces a homography \( H \) between the two images, so that the image of points on the plane are mapped as \( x' = Hx \), where \( x \) and \( x' \) are the image points in the first and second views respectively. The homography can be determined from a minimum of four correspondences over the two views of points (or lines) on the plane \cite{83}. Then the fundamental matrix for the views is given by \cite{76}

\[
F = H^{-\top} [e]_\times = [e']_\times H
\]

where \( e' \) is the epipole in the second image and \( [e']_\times \) is the \( 3 \times 3 \) skew matrix for which \( [e']_\times x = e' \times x \). Thus, in order to determine the fundamental matrix it is only necessary to determine the position of the epipole, i.e. only two parameters need be specified.

The parallax vector in the second view is the vector joining the image of a world point \( x' \) with the transferred image of that point from the first view \( \hat{x}' = Hx \). The line joining \( x' \) and \( \hat{x}' \) contains the epipole \( e' \), so two such lines are sufficient to compute its position, and the
Figure 7.4: Plane induced parallax. The ray through $X$ intersects the plane $\pi$ at the point $X_\pi$. The images of $X$ and $X_\pi$ are coincident points at $x$ in the first view. In the second view the images are the points $x'$ and $\tilde{x}' = Hx$ respectively. These points are not coincident (unless $X$ is on $\pi$), but both are on the epipolar line $l'$ of $x$.

full epipolar geometry follows from (7.13). The magnitude of the parallax vector is related to the distance of the world point and cameras from the world plane [57, 67, 98].

7.2.2 Parallax geometry for smooth surfaces

As illustrated in figure 7.5, at points at which the epipolar plane is tangent to a surface, the occluding contour is tangent to the corresponding epipolar lines. Conversely, corresponding image epipolar tangent points backproject to a 3D point on the surface (at the point at which the epipolar plane is tangent to the surface). Thus, image epipolar tangent points are equivalent to a point correspondence and may be used to compute the epipolar geometry. A similar result holds in the case of epipolar tangents to an imaged space curve.

Porrill and Pollard [87] used epipolar tangent points to refine the epipolar geometry estimated from point correspondences. Cipolla et al. [23] then estimated epipolar geometry from epipolar tangencies alone. However, there are two problems in this case: first, the outline geometry must be sufficiently rich to provide at least 7 epipolar tangents; second,
Figure 7.5: Upper: epipolar tangency. The epipolar plane \( OO'X \) is tangent to the surface at \( X \). The imaged outline is tangent to the epipolar lines at \( x \) and \( x' \) in the two views. The dashed curves on the surface are the contour generators. Lower: the first view is mapped to the second using the homography induced by the plane \( \pi \). The mapped point \( \tilde{x}' = Hx \). Imagine that the first camera is a light source and casts a shadow of the surface onto the plane, then the mapped outline is the image of this shadow boundary. The epipolar line is tangent to both the original and mapped outlines.

and more seriously, the solution involves a search over the 7 parameters of the fundamental matrix. This is a far worse situation than that of point correspondences where 7 correspondences directly determine either 1 or 3 solutions for the fundamental matrix, with no search involved.

If the outline from the first view is mapped to the second using the homography induced by the plane then, as illustrated in figure 7.5, an epipolar line which is tangent to
the outline in the second view will also be tangent to the mapped outline. For brevity such lines will be referred to as bitangents, but it should be understood that the lines are only tangent to each contour once. Note that under the homography, the epipole $e$ is mapped to $e'$, and corresponding epipolar lines are mapped to each other. A real example showing the mapped outline and epipolar bitangents is given in figure 7.9. The situation is then completely equivalent to the point parallax case, with two bitangents (each the equivalent of a point correspondence) uniquely determining the epipole and thence the complete epipolar geometry using (7.13). Note that all epipolar tangents generate bitangents, but not all bitangents are epipolar tangents (just as not all putative point correspondences are correct).

To summarize: Given the homography induced by the plane and two (epipolar) bitangents, then the epipolar geometry is uniquely determined.

This result applies for a general motion of the camera including a change in the internal parameters. If the camera undergoes the particular motion of a pure translation (no rotation) with fixed internal parameters then the views may be registered by an identity homography (i.e. simply superimposed) as this homography corresponds to registering using the plane at infinity for this special motion. The bitangent result for this case was used in [97].

![Figure 7.6: Three images from different viewpoints. The surface of the jug is featureless other than a black mark that has been added to visually assess the computed geometry.](image)

### 7.2.3 Computing the epipolar geometry

In this section, it is seen that the bitangent constraint described in section 7.2.2 can be used to automatically compute the fundamental matrix between two views.
Figure 7.7: *Upper:* The first two images from figure 7.6 registered to the plane. *Lower:* The automatically detected bitangents are superimposed. In this case, 34 bitangents are detected.

At least two epipolar tangents are required. For any finite smooth object, which does not intersect the baseline, there are at least two epipolar planes which are tangent to the surface (‘below’ and ‘above’). There are thus at least two bitangents, provided the epipolar lines corresponding to these planes lie within the finite image. However, for anything other than the most simple objects, it is normally the case that there are many epipolar tangents throughout the image.
Figure 7.8: Many of the bitangents from figure 7.7 do not correspond to valid epipolar tangents from the curvature sign constraint (see text) and can be removed. This leaves 17 of the original 34 bitangents.

**Automatic computation of the epipole**

It has been shown (section 7.2.2 above) that certain of the bitangents between corresponding curves in a pair of registered images contain the epipole. However, as not all bitangents contain the epipole, an algorithm is required to reject irrelevant bitangents and determine a best estimate of the epipole from those remaining.

One constraint that is satisfied by epipolar bitangents, but not by all bitangents, is that the corresponding outline curves have a consistent sign of curvature at epipolar tangents. This follows because the surface is on one side of the epipolar plane at surface tangent points (here sign includes zero in the case of tangency at outline inflections). Of course, the photometric constraint used to disambiguate interest point matches (based on the similarity of intensity neighbourhoods) may also be employed here.

A robust algorithm based on RANSAC [41] is then used to vote for the most likely epipole. The steps are: (1) Randomly select and intersect a pair of bitangents to produce a hypothesized epipole position; (2) Measure the support for this putative epipole position by the number of other lines through this epipole which are within a threshold distance of being bitangent; (3) Choose another pair of bitangents and repeat. The selected epipole is the one with maximum support. Figures 7.7 – 7.9 illustrate the steps of this algorithm on a real
Figure 7.9: *Upper:* The epipole bitangent lines computed automatically from those of figure 7.8. The epipole lies on the intersection of these 5 bitangents. *Lower:* The accuracy of the computed epipolar geometry is demonstrated for a point on the jug. The residual error is sub-pixel.

example. In the implementation the bitangents are computed as described in [94]. It is generally found that provided there are 4 or more epipolar bitangents available the epipole is correctly determined.

The algorithm described above is able to compute the epipolar geometry using bitangents alone. However, it is often the case that a smooth surface contains surface markings which provide point correspondences. These point correspondences generate parallax vectors as described in section 7.2.1. Such point correspondences may be used simultaneously with bitangents to compute the epipolar geometry.

It is also worth noting that geometrically the epipolar tangents are necessarily in areas of the image where the surface is tangent to rays through the camera centre. Markings coinciding with these surface tangent points will therefore not be visible in the image, so
epipolar tangents are a complementary source of correspondences to texture markings.

7.2.4 Computing the trifocal geometry

In this section a new minimal solution for computing the trifocal tensor is developed. Furthermore, the solution is suitable for use with epipolar tangencies. Previously it has been shown [28, 58] that the trifocal tensor is uniquely determined from the homography induced by a plane together with the correspondence of two points off the plane over all three views. However, this solution is not suitable in the case of epipolar tangencies because epipolar tangent points are a property of two views and for general motion the surface epipolar tangent points differ between the view pairs. Epipolar tangent points are common to three views for the special motion where the camera centres are collinear (so that epipolar planes are common to view pairs), or where the trifocal plane (the plane defined by the three camera centres) is tangent to the surface. It is shown here that the trifocal tensor may be determined uniquely from the plane homography together with two epipolar tangents between pairs of views.

The solution is obtained by constructing the three camera matrices (up to a common projective transformation of 3-space) and then computing the trifocal tensor from these matrices. The trifocal tensor is invariant to the 3-space projective ambiguity of the camera matrices and is computed from them as described by Hartley [53]. The views are numbered 1–3, with $e_{ij}$ denoting the epipole in view $i$ which is the image of the $j$th camera centre. It is supposed that the homographies induced by the world plane between the first and second view, $H_{12}$, and first and third views, $H_{13}$, are known. If the second view is mapped onto the first by $H_{12}^{-1}$ and the third onto the first by $H_{13}^{-1}$, then the computation of the trifocal tensor reduces to a particularly simple form. The $3 \times 4$ camera matrices of the three views may be chosen as [28]

$$p^1 = [I \mid 0], \quad p^2 = [I \mid e_{12}], \quad p^3 = [I \mid \lambda e_{13}]$$

up to a homography of 3-space, where $\lambda$ is a scalar. As shown in section 7.2.2 the epipole $e_{12}$ may then be computed from two epipolar tangents between views 1 and 2. Similarly, $e_{13}$ may be computed from two epipolar tangents between views 1 and 3. It only then remains
to determine $\lambda$ (note that the relative scaling of the identity matrix and epipoles within the $P$ matrices is fixed).

The scalar $\lambda$ is computed from a consistency relationship on the epipoles over the three views. For simplicity assume the third camera centre is finite. Then it may be written as $C^3 = (-\lambda e_{13}^\top, 1)^\top$ (since $P^3C^3 = 0$). Then

$$e_{23} = P^2C^3 = -\lambda e_{13} + e_{12}$$ (7.14)

The epipole $e_{23}$ may be computed from two epipole tangents between views 2 and 3, and thence $\lambda$ determined from the two inhomogeneous constraints of (7.14). However, provided the epipoles are not coincident, a single epipolar tangent line, $l_{23}$, between views 2 and 3 determines $\lambda$ because the epipole $e_{23}$ lies on this line, and so $l_{23}^\top e_{23} = 0$ gives a linear equation in $\lambda$ from (7.14). The epipoles in the registered views are coincident if, and only if, the camera centres are collinear.

To summarize, we have shown the following result in the case of general motion over three views: *Given the homography induced by the plane and two (epipolar) bitangents between two of the pairs of views (say views 1 & 2, and 1 & 3), and one epipolar bitangent between the other pair (say views 2 & 3), then the trifocal geometry is uniquely determined.*

**Trifocal tensor computation algorithm**

In summary, the trifocal tensor may be computed by the following four steps

1. **Register the views:** Compute the homography induced by the world plane between views 1 and 2, and transfer the outline from view 2 into view 1; similarly, compute the homography induced by the world plane between views 1 and 3, and transfer the outline from view 3 into view 1. From here on all computations are on outlines in the registered view.

2. **Compute epipolar bitangents:** Two bitangents are required between two of the outlines (between 1 and 2, and 1 and 3 say) and one, $l_{23}$, between the others (2 and 3 say).
3. **Determine the epipoles and scalar** $\lambda$: The epipole $e_{12}$ is the intersection of the two bitangents to outlines 1 and 2, and $e_{13}$ is the intersection of the two bitangents to outlines 1 and 3. The scalar $\lambda$ is determined as $\lambda = (l_{23}^T e_{12})/(l_{23}^T e_{13})$.

4. **Compute the trifocal tensor for the registered views from the camera matrices:**
   From $P^1 = [I \mid 0], P^2 = [I \mid e_{12}], P^3 = [I \mid \lambda e_{13}]$ determine the tensor as described in [53].

**Automatic computation of the trifocal tensor**

The solution described above has very undemanding requirements. There will always be at least one bitangent between outlines (at the imaged “top” of the object, if it is resting on the plane), and the solution only requires one more bitangent between two of the three view pairs. An automatic algorithm, based on the RANSAC paradigm, proceeds in much the same way as that for computing the fundamental matrix. Again epipolar tangents and point correspondences (off the plane) may be combined in the same algorithm. An example is shown in figures 7.10–7.12.

![Image of registered images](image_url)

Figure 7.10: The three images of figure 7.6 registered to the plane.
Figure 7.11: The bitangents (blue) between images 1 and 2 define the epipole $e_{12}$, whilst the bitangents (red) between images 2 and 3 define the epipole $e_{23}$. The remaining epipole, $e_{13} = e_{31}$, lies on the line joining $e_{12}$ and $e_{23}$.

Figure 7.12: The trifocal tensor permits a single point in image 1 (shown here by a circle) to be transferred to an epipolar line in image 2. If the correspondence along the epipolar line in image 2 is known, a single point is determined in image 3. The black mark visible in figure 7.6 lies at the centre of the superimposed circle.

7.2.5 Computation of camera matrices for multiple views

In this section we describe how the 2 and 3 view results of the previous section may be extended to multiple views. The objective is to obtain a projectively consistent camera matrix $P^i$ for each view of the sequence, such that if a point $X_j$ is visible in several views then its image is given by $x^i_j = P^i X_j$. We wish, of course, to make best use of the scene constraint that a plane is available.

Previous methods of computing consistent cameras for an uncalibrated image sequence,
for example [6, 43, 72], have applied to general 3D scenes, and this has entailed extra difficulty. In our current case the homographies which are readily available throughout the sequence provide a firm basis for the camera computation.

Suppose we have obtained homographies $H^i$ relative to a reference image, how does this constrain $P^i$? A homography has 8 independent elements, and so only imposes 8 constraints on the 11 independent elements of the camera matrix. If the world coordinates are chosen such that the reference plane has $z = 0$, then points on that plane $X = (x, y, 0, 1)^\top$ are mapped by the projection matrix for a particular image as:

$$
x = PX = [p_1 \ p_2 \ p_3 \ p_4] (x, y, 0, 1)^\top
= [p_1 \ p_2 \ p_4] (x, y, 1)^\top
$$

Thus the 3 independent parameters of the third column of $P$ are undetermined. This ambiguity corresponds to the position of the camera centre. However, if the internal parameter matrix $\kappa$ is known, then this ambiguity is reduced from a 3 parameter family to a finite number of solutions. Furthermore, the internal parameters can be computed from $H^i$ if we assume that (some of) the camera parameters remain fixed throughout the sequence [125]. We now have a complete path to determining a metric set of camera matrices:

1. Compute the homographies $H^i$ for the sequence.

2. Determine the internal calibration matrix $\kappa$.

3. Determine initial estimates of the camera matrices $P^i$.

4. Use the pairwise epipolar tangent points to refine the estimated camera matrices $P^i$.

These steps are now described in more detail.

**Computation of homographies**

The reference plane homographies are computed in a standard manner [20]. Point and line features are extracted in each image and the inter-image homographies are robustly estimated using RANSAC. Because the reference plane is generally visible in all views, each
Figure 7.13: Example sequence 1. Although the object is resting on a calibration tile, the camera calibration is **not** determined from this.

Figure 7.14: Example sequence 2. *Reclining figure*, Henry Moore. These works are reproduced by permission of the Henry Moore Foundation and are not to be reproduced or altered without permission.

Homography can then be directly related to the reference image, with a concomitant improvement in accuracy. The homographies are bundled adjusted to be projectively consistent throughout the sequence.

**Determinant of camera calibration**

Given the set of homographies, the camera calibration can be extracted as described by Triggs [125]. Briefly, each of the recovered homographies are of the form

\[
H^i = K \begin{bmatrix} r^i_1 & r^i_2 & t^i \end{bmatrix} H^0
\]  
(7.15)

where $K$ is the matrix of internal camera parameters, and $H^0$ is the unknown projective transformation between the reference image and Euclidean coordinates on the plane $z = 0$. Writing

\[
K^{-1}H^i(H^0)^{-1} = \begin{bmatrix} r^i_1 & r^i_2 & t^i \end{bmatrix}
\]  
(7.16)

and requiring that $r^i_1$ and $r^i_2$ are proportional to the first two columns of a rotation matrix yields two constraints on the unknowns $K$ and $H^0$. Minimizing the reprojection error
implied by imposing these constraints gives estimates of the unknown parameters.

**Computation of projection matrices**

The above procedure returns the internal camera parameters, and the first two columns of
the rotation matrix for each camera. It is then a trivial matter to construct the third column
as \( r_3 = r_1 \times r_2 \) and generate the full projection matrix for each view \( P^i = K [R^i \mid t^i] \).

These projection matrices provide estimates of the epipoles that are sufficiently accu-
rate to usefully prune the putative matches in the RANSAC for bitangents—only bitangents
that pass within a threshold distance of the approximate epipoles are considered.

**Refinement of projection matrices**

The projection matrices produced by the above procedure are very accurate for points on
the reference plane, but accuracy decreases sharply off the plane. Happily however, the
pairwise epipolar tangencies typically provide corresponding points which are far from
the plane.

The two sources of information are combined by entering the epipolar tangencies as
2-view point correspondences in a Euclidean bundle adjustment for the \( P^i \). This has the
effect of stabilizing the camera computation, leading to very accurate results. Cameras
computed for the example sequence of Figure 7.14 are shown in Figure 7.15.

**7.2.6 Conclusions and Extensions**

It has been demonstrated that given a registering homography from a plane and an occlud-
ing contour with sufficiently rich geometry (to support at least 2 epipolar tangents) the 2
and 3 view relations can be computed from this information alone, without requiring an
iterative search. In practical circumstances the visual information is generally not so im-
poverished and both interest points and bitangents are available on the object. Both may
be used interchangeably in the multiple view estimation algorithms.

This is a very pragmatic approach since planes are often available in the scene—indeed
the plane need not be an actual plane, for example given sufficient (4 or more) vanishing
points and their correspondences the plane at infinity may be used.
A natural and interesting extension for this work would allow multiple sequences to be registered—again using the plane as the main registration device. In this manner, sequences captured at a set of azimuths whilst circumnavigating an object may be fairly effortlessly combined. Much like peeling an apple in reverse.
Figure 7.16: Volumetric occupancy grids, computed from silhouettes tracked in the original images, with camera positions computed as described in §7.2.5. This “visual hull” penetrates many, but not all, concavities.

Figure 7.17: Texture-mapped surface models, viewed from positions not in the original sequences. The black areas are not textured as they were not seen in any image.
Chapter 8

Conclusions and Further Work

8.1 Summary and Novel Contributions

Chapter 2 presented the theory of reconstructing quadric surfaces from both point constraints and the constraints provided from the silhouette of the quadric from a number of viewpoints. By using the dual-space as a tool, a complete and consistent framework was presented which allowed the silhouette constraints to be considered as point constraints in the dual-space. The impact of epipolar tangencies on quadric surface reconstruction was explained both in real-space and in the dual-space. Finally, it was shown that the framework allowed degenerate quadrics and planar conics to be reconstructed with little specific effort.

Chapter 3 followed the theory of chapter 2 with a set of algorithms to reconstruct a quadric from either point correspondences over a number of images, or silhouettes. Much care was taken to provide an “optimal” surface reconstruction from noisy image data. In particular, it was seen that the silhouettes of a quadric over two or more images must correspond (that is, they must be consistent with the epipolar geometry) and, due to image noise, they must be “corrected” before they can be used as a constraint on the viewed quadric: an optimal and robust conic correction algorithm was proposed.

Having reconstructed the quadric surface, models and new views of the surface can be generated. It has been seen in chapter 3 that sub-pixel accuracy is achieved, with quadric-induced transfer providing registration between two images. Results were given throughout the chapter.

Chapter 4 considered the general problem of reconstructing an arbitrary surface viewed from a number of viewpoints. It has been seen that the silhouettes of a surface provide
both tangency constraints along the contour generators and point contraints at the frontier points: the concept of an epipolar net was proposed as the set of all point constraints provided by a sequence of silhouettes. Further, it was shown that the visual hull encodes all the information provided by the silhouettes. Surface textures also provided important information about a viewed surface, and it has been proposed that this information is complementary to the information provided by the silhouettes. Thus a complete reconstruction algorithm should make good use of both sources of image data.

As for quadric surface reconstruction, chapter 5 provided a set of algorithms to accurately reconstruct a surface from both silhouettes and stereo triangulation. First, a very efficient voxel algorithm was presented, using information from the image silhouettes to reconstruct the visual hull. This was then used to bootstrap a space-carving algorithm, which accurately reconstructs concave regions which are not reconstructed from the silhouettes. An improved photo-consistency test was presented, based on texture correlation over a number of images. Again, examples of the algorithm are presented.

Chapter 6 considered the problem of reconstructing an optimal surface from the available image data. A cost function is presented which minimises the reprojection errors of the silhouette of a surface. This is then combined with a similar cost function minimising the texture correlation over a number of images. The minimisation of the cost functions is expressed as a surface evolution, and the surface encoded as a level-set, providing an optimal surface reconstruction algorithm given both silhouette and texture information.

Finally, chapter 7 presented two methods for recovering the camera projection matrices from an image sequence. First, it was shown that both the camera motion and calibration can be robustly computed from a general sequence of images of an object on a turn-table. Second, it was shown that the camera motion can be found for sequences of texture-less objects: most previously available algorithms required point correspondences between pairs of images to recover the epipolar geometry.
Summarising these novel contributions for completeness gives:

- **Chapter 2: Reconstructing Quadric Surfaces — Theory**
  - Quadric reconstruction from dual-space geometry;
  - Degenerate quadric and planar conic reconstruction;

- **Chapter 3: Reconstructing Quadric Surfaces — Implementation**
  - Image silhouette correction from epipolar geometry constraints;
  - Optimal quadric reconstruction from noisy image data;
  - Sub-pixel quadric-induced transfer registration between two or more images;

- **Chapter 4: Reconstructing General Surfaces — Theory**
  - Introduction of the *epipolar net*;
  - Visual hull encoding all information available from silhouette;
  - Texture and silhouette information complementary;

- **Chapter 5: Reconstructing General Surfaces — Implementation**
  - Two-tier approach: compute visual hull, then use texture to refine;
  - Surface-induced transfer as photo-consistency constraint;
  - Selective voxel testing in space carving;

- **Chapter 6: Optimal Surface Reconstruction**
  - Optimal silhouette information cost function;
  - Surface propagation from silhouette information;
  - Level-set formulation of silhouette induced reconstruction;

- **Chapter 7: Recovering Camera Motion**
  - Automatic camera calibration from turn-table sequences;
  - Robust automatic computation of epipoles from silhouettes alone.
8.2 Future Work

Whilst the results of the algorithms provided in this thesis are satisfying, there are naturally areas of research that still remain open. This section presents some of these.

8.2.1 Quadric reconstruction

It has been seen that a quadric surface can be accurately reconstructed from both silhouette and point constraints. In particular, it was seen that by considering the reconstruction in the dual-space, “tangency constraints”, as provided by the image silhouettes, become “point constraints” on the dual quadric.

One natural extension is the reconstruction of higher order surfaces, using the same framework. For instance, twisted cubics are a class of curves which might be easily reconstructed using this approach as they have easily definable mathematical properties (such as local tangency).

8.2.2 General surface reconstruction

The algorithm for general surface reconstruction presented in chapter 5 would also provide many avenues for further work.

Currently, the image silhouettes are recovered by careful use of chroma-keying techniques. For many indoor scenes, this is easy, albeit rather artificial; for outdoor scenes this is often difficult if not impossible (see figure 7.14 for example). An area for future investigation would be the automated tracking of the image silhouette over a full sequence. Previous work such as that of Berger [10] has touched on this area, but it is believed that further investigation would be interesting. Once a partial model of the surface has been built, from a number of images for example, it is quite likely that this can be used to “predict” the image silhouette for further images. The prediction could then be refined using spline-based techniques. The results of figure 5.13 suggest that slight imperfections in the segmentation of the silhouette are not critical over a long image sequence, and so it is very likely that small tracking errors would be insignificant.

Whilst both texture and silhouette information have been considered in detail in this
thesis, little note has been made of the other sources of information about surface shape. Shading is one such source which has been investigated extensively in the literature [34, 9, 130]. Other sources might include a priori information about the surface material properties. An improved photo-consistency test including these information sources would provide more robust recovery of surfaces with sparse texture.
Bibliography


Appendix A

Matrix Adjoint

The adjoint of a matrix, $\mathbf{M}$, is defined as $\mathbf{M}^* = (\text{cofactors}(\mathbf{M})^\top)$. As a consequence, $\mathbf{M}^{-1} = \mathbf{M}^* / |\mathbf{M}|$ where $|\mathbf{M}|$ is the determinant. Further, $(\mathbf{M}^*)^* = \mathbf{M}$ up to scale.

It is interesting to note that the adjoint of a singular $n \times n$ matrix of rank $r$ is non-zero if $r > (n - 2)$. A proof follows:

Express a real $n \times n$ matrix as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$  \hspace{1cm} (A.1)

using a singular value decomposition, such that $\mathbf{U}$ and $\mathbf{V}$ are $n \times n$ orthonormal rotation matrices and $\mathbf{\Sigma}$ is an $n \times n$ matrix with positive real diagonal elements in decreasing order. The adjoint its definition above as

$$\mathbf{M}^* = \mathbf{V}^\top \mathbf{\Sigma}^* \mathbf{U}^* .$$

Both $\mathbf{U}$ and $\mathbf{V}$ are non-singular and hence have non-zero adjoints, and therefore $\mathbf{M}^*$ is only non-zero if $\mathbf{\Sigma}^*$ is also non-zero. The proof is completed by observing the structure of $\mathbf{\Sigma}$ for $\mathbf{M}$ of rank $r$:

$$\mathbf{\Sigma}^* = \text{cofactors} \begin{bmatrix} \mathbf{\Sigma}_{r \times r} & 0_{r \times (n-r)} \\ 0_{(n-r) \times r} & 0_{(n-r) \times (n-r)} \end{bmatrix} .$$

By taking cofactors on the right-hand side, it is clear that if $r \leq (n - 2)$, the adjoint of $\mathbf{\Sigma}^*$ is zero.