Reconstruction of Spheres using Occluding Contours from Stereo Images

Sudanthi N.R. Wijewickrema, Andrew P. Papliński
Clayton School of Information Technology,
Monash University,
VIC 3800, Australia
{snw, app}@csse.monash.edu.au

Charles E. Esson
Colour Vision Systems Pty Ltd,
11, Park Street, Bacchus Marsh,
VIC 3340, Australia
charleses@cvs.com.au

Abstract

This paper discusses an efficient way of reconstructing spheres from the occluding contours of two views. It takes into consideration, the geometric properties of the system and uses frontier points as additional data. This algorithm is especially useful for applications where the objects to be modelled are known to resemble spheres and where speed is of importance. It is developed for fruit sorting applications where the processing has to be done in real-time.

1. Introduction

Modelling of objects as geometric primitives has gained much importance in recent years in applications where the true shape may not be needed but the size and approximate shape of the object is required. An example of such a process is quadric reconstruction, which is a well researched area (e.g., Kang et al. [4], Cross [2] etc.).

Although spheres are a special case of quadrics and could be reconstructed using the same algorithms, it may be a waste of computational power in cases where the objects are known to be spherical and an approximation as a sphere would be sufficient. For example, in fruit sorting, if the fruit to be sorted are spherical (e.g., oranges), we can assume that such a model would be appropriate.

In Wijewickrema et al. [8], it is shown experimentally that a general ellipsoid reconstruction algorithm can successfully be used to model spheres where the axes of the resulting ellipsoid become very close in size. But the calculations required are unnecessarily complex for a sphere which only has 4 constraints as opposed to the 9 constraints of an ellipsoid. Hence, more specialized methods for spherical models would be advantageous in some situations.

The most common way to link projections to their quadrics is by using dual space geometry (Hartley and Zisserman [3]). Quadric reconstruction in dual space is discussed in works such as Cross [2] and Kang et al. [4].

2. Review - Quadrics and Projections

A quadric is defined as a surface in 3d, and can be represented by a $4 \times 4$ symmetric matrix $Q$. For a point $X$ on the surface, represented by a homogeneous coordinate vector $X = [x \ y \ z \ 1]^T$, the following condition is satisfied.

$$X^T Q X = 0$$ (1)

For a sphere $Q_s$, the same condition holds: i.e., $X^T Q_s X = 0$. If $c$ is the center of the sphere, and $r^2 = c^T c - \gamma$ is the square of the radius, $Q_s$ is given as:

$$Q_s = \begin{bmatrix} I & -c^T \\ -c & \gamma \end{bmatrix}$$ (2)

The projection of a quadric (occluding contour) is the projection of its contour generator which is the conic formed by the tangent rays travelling from the camera center. The occluding contour is a conic, represented by a $3 \times 3$ symmetric matrix ($C$) in homogeneous coordinates. For a point $X$ on the conic, the relationship in eqn (3) is satisfied.

$$X^T C X = 0$$ (3)

3. Proposed Reconstruction Algorithm

First, to obtain the occluding contours, the projected images have to be fitted with conics. For this, any suitable
conic fitting algorithm could be used. We use the ellipse specific algorithm discussed in Wijewickrema and Papliński [7] as it is simple and robust against outliers.

In real applications, the fitted conics may not adhere to epipolar tangency constraints (Cross [2]) due to errors in fitting and the fact that the objects may not be ideal quadrics. Hence, a conic correction algorithm has to be applied on the fitted conics before the reconstruction. For this, we use the method of adjustment introduced in Wijewickrema et al. [8] which uses frontier points.

Then, using the adjusted conics and frontier points calculated in the previous step, we generate the 4 parameters that define the sphere as given in eqn (2). The following subsections discuss this in more detail.

3.1. Frontier Points

The concept of ‘frontier points’ was first introduced in Rieger [6], and later interpreted in Porril and Pollard [5] as the fixed point on the surface of the quadric, corresponding to the intersection of two corresponding contour generators. This is shown in figure 1. Further, they lie on the epipolar planes shared by the two tangent cones.

This relationship was used in Wijewickrema et al. [8] to calculate the frontier points using two occluding contours, and to adjust the fitted conics to adhere to epipolar tangency constraints. We use this algorithm for the adjustment, and use the conics such adjusted, in the reconstruction.

3.2. Reconstruction of the sphere

The contour generator of a sphere is essentially a circle, as any plane through the sphere would result in a circle. The projection of this (the occluding contour), is a conic \( C \), which is a circle or an ellipse. Since we know the occluding contours of the sphere (in this case the adjusted ellipses) and the camera projection matrices, we can calculate the tangent cones. The \( 3 \times 4 \) camera projection matrix \( P \), includes the intrinsic and extrinsic parameters of the camera (Hartley and Zisserman [3]), and gives the projection \( x \), of a point in 3d \( X \), as follows:

\[
x = PX
\]

By solving eqns (3) and (4), we get the tangent cone \( Q_c \), as the back-projection of the occluding contour, \( C \) (Hartley and Zisserman [3]).

\[
Q_c = P^T C P
\]

At any point on the contour generator, the sphere and the cone are tangent to each other. Hence, a normal drawn at such a point is common to both surfaces and goes through the center of the sphere. Let us consider any plane passing through the line connecting the camera center and the center of the sphere. Figure 2 illustrates this relationship.

\( A \) and \( B \) are points on the contour generator. \( AC \) and \( BC \) are tangent to the sphere, and \( OA \) and \( OB \) are normal to the tangents. This forms identical triangles \( OAC \) and \( OBC \) making \( OC \) intersect \( AB \) at right angles. Hence, the tangent cone is right circular with \( OC \) as its axis. That is, the axis of the tangent cone goes through the center of the sphere.

So, by determining the tangent cone (and therefore the axis) of one view, we narrow down the location of the center to a line. A second view and a corresponding tangent cone uniquely determine the center. Let such a tangent cone be defined by \( Q_c \) as follows:

\[
Q_c = \begin{bmatrix}
\hat{Q}_c & \alpha \\
\alpha^T & b
\end{bmatrix}
\]

To calculate the axis of \( Q_c \), we transform it to a coordinate system where its apex lies at the origin and the axis is aligned with one of the coordinate axes. Such a cone is given in eqn (7). Note that \( \hat{A} \) is a diagonal matrix.
\begin{equation}
A = \begin{bmatrix}
\tilde{A} & 0 \\
0 & 0 \\
\end{bmatrix}
\end{equation}

Let, The transformation that converts the cone \( Q_c \) to this new coordinate system be \( g \). This is a rigid body transformation consisting of a rotation \( R \) and a translation \( T \).

\begin{equation}
g = \begin{bmatrix}
R & T \\
0 & 1 \\
\end{bmatrix}
\end{equation}

The rigid body motion of a point is \( X = g\tilde{X} \) where \( \tilde{X} \) and \( X \) are the point coordinates in the first and second coordinate systems respectively. Substitution of this in eqn (1), results in the quadric transformation given in eqn (9).

\[
A = \begin{bmatrix}
R^T & 0 \\
T^T & 1 \\
\end{bmatrix}
\begin{bmatrix}
\tilde{Q}_c & \mathbf{a} \\
\mathbf{a}^T & b \\
\end{bmatrix}
\begin{bmatrix}
R & T \\
0 & 1 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
R^T\tilde{Q}_cR & R^T(\tilde{Q}_cT + \mathbf{a}) \\
(T^T\tilde{Q}_c + \mathbf{a}^T)R & T^T\tilde{Q}_cT + 2\mathbf{a}^Tb + b \\
\end{bmatrix}
\]

(9)

By comparing the first elements of \( A \) in eqns (7) and (9), we get eqn (10) which gives an eigen decomposition, since \( R^T = R^{-1} \). Hence, the matrix of eigenvalues of \( \tilde{Q}_c \) gives \( \tilde{A} \) which fully defines the cone in the new coordinate system and the eigenvectors give the eigen decomposition.

\[
\tilde{A} = R^T\tilde{Q}_cR \implies \tilde{Q}_c = R\tilde{A}R^T
\]

(10)

Similarly, the comparison of the 2\( ^{nd} \) or 3\( ^{rd} \) element, results in the following value for the translation vector, \( T \).

\[
\tilde{Q}_cT + \mathbf{a} = 0 \implies T = -\tilde{Q}_c^{-1}\mathbf{a}
\]

(11)

Finally, the comparison of the last element gives a relationship between the elements of \( \tilde{Q}_c \) that has to be satisfied.

\[
T^T\tilde{Q}_cT + 2\mathbf{a}^TT + b = 0 \implies b = \mathbf{a}^T\tilde{Q}_c^{-1}\mathbf{a}
\]

(12)

To make sure that this relationship holds, we have to first divide the matrix \( \tilde{Q}_c \) by the constant \( \mathbf{a}^T\tilde{Q}_c^{-1}\mathbf{a} \) to form a new matrix (which also defines the same cone as multiplication or division of the quadric matrix by a scalar does not alter its properties), before the eigen decomposition.

Then, from eqn (8) we get the transformation \( g \). Using \( g \), we transform the axis of \( A \) (which is aligned with a coordinate axis) to get the axis of \( \tilde{Q}_c \). By calculating the axes of the two cones, we uniquely determine the center \( \mathbf{c} \) as their intersection point. A point on the surface can then be used to determine the remaining parameter.

The best candidate for the point on the surface is one of the frontier points (say \( X_f \)) which was already calculated using only the two occluding contours. It satisfies \( X_f^T\tilde{Q}_cX_f = 0 \). Since \( \mathbf{c} \) is known, this gives the value of \( \gamma \) as shown in eqn (2), from which we can find the radius.

4 Example Application

The proposed algorithm is aimed at a specific real-life application: namely fruit sorting. Here we model approximately spherical fruit (for examples oranges) in 3d to gain information about their size. The calculations have to be done in real-time and hence, the simplicity of the algorithm described here is ideal for this process.

The fruit are placed on a conveyor and rest on four rollers each. There are two cameras mounted on either side of the conveyor and are synchronized to capture images at the same instance as the fruit travel along. The next step is to fit ellipses to the images of fruit and to adjust them to fit epipolar tangency constraints. An example is shown in figure 3.

The proposed method is then used to find the radius and center of the approximated sphere. Such a reconstructed fruit with the rollers it rests on, is shown in figure 4.

5 Experimental Results

As it is not possible to measure the approximate radius of a fruit accurately, we used actual spherical objects in the experiments. These spheres were placed on the conveyor instead of the fruit and images were captured by stereo cameras as they travelled forward. The radius of each object was measured and was used in the calculation of error.

For the analysis of error, we use two measures: re-projection error and radius error. The former gives a measure of the error involved in the conic fitting and adjustment while the latter gives a measure of the error in the size of the reconstructed sphere. This forms a more complete view of the errors involved in the whole reconstruction process.

To calculate the re-projection error, we define the algebraic distance of a point with respect to a conic. If the conic \( C \) satisfies eqn (3) for a point on it, the algebraic distance of any point \( x_i \) (on, inside or outside \( C \)) is: \( d_i = x_i^TCx_i \).

Then, we project the reconstructed sphere on the image
Figure 4. Reconstructed Sphere

plane to get the conic $C_i$, and calculate the algebraic distance of each edge point. The re-projection error in pixels is defined in eqn (13), where $n$ is the number of edge points.

$$e_{rep} = \frac{1}{n} \sum_{i=1}^{n} d_i^2$$

(13)

The average distance from any edge point to the reprojected conic is determined in pixels using the above equation. We calculate the re-projection errors for both the left and right images and then determine the average to get a better idea of the error involved.

The radius error is calculated as the percentage error between the radius of the actual and reconstructed spheres. The results for some of the reconstructed spheres are summarized in table 1.

The average re-projection and radius errors were found to be $3.542 \times 10^{-3}$ pixels per edge point and 1.8% respectively. A radius error of less than 5% is considered to be acceptable for fruit grading and hence the method is very suitable for this application. Further it was found that when compared to the ellipsoid modelling algorithm introduced in Wijewickrema et al.[8], this method was 30% faster when applied to the same known spherical objects.

6 Conclusion

We discussed an algorithm to be used in practical applications (for example, size sorting of spherical fruit) to fit spheres to objects using their occluding contours. The advantage of this is that the calculations are simple and there is no need to go into the dual space for the reconstruction. The results obtained are accurate and fast, and hence is suitable for applications that require high speed. The results were generated with respect to a system designed for fruit sorting but can be used in any application that requires modelling of spheres using stereo views. The primary limitation of this method is that the objects to be modelled should be approximately spherical to give accurate results.

Future work involves the incorporation of rotation and translation, selection of areas best covered by each camera and positional view, and the determination of their projections. The aim is to extract optimal areas from a set of images taken at regular intervals, so that they can be used for blemish detection and colour sorting of fruit.

References


<table>
<thead>
<tr>
<th>Sphere</th>
<th>Re-projection Error (pixels ×10^{-3})</th>
<th>Radius Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere 1</td>
<td>3.956</td>
<td>1.4455</td>
</tr>
<tr>
<td>Sphere 2</td>
<td>2.689</td>
<td>2.2763</td>
</tr>
<tr>
<td>Sphere 3</td>
<td>3.908</td>
<td>1.0958</td>
</tr>
<tr>
<td>Sphere 4</td>
<td>2.282</td>
<td>2.3288</td>
</tr>
<tr>
<td>Average</td>
<td>3.542</td>
<td>1.8121</td>
</tr>
</tbody>
</table>

Table 1. Error Analysis