Affine Visual Servoing

Geoffrey M.T. Cross and Roberto Cipolla
Department of Engineering, University of Cambridge,
Cambridge, CB2 1PZ.
{92gmtc,cipolla}@eng.cam.ac.uk

Abstract

Small movements of a viewer relative to the surrounding scene induce deformations in the shape and detail of the projected image. This paper will consider the problem of using these deformations to provide visual feedback on the current position of the viewer relative to the scene.

The implementation calculates the transformations that occur due to small movements around the current position. If a "target" transformation is specified, the equivalent motion can be interpolated. As a result, it is possible to position the viewer relying solely on visual feedback. All the calibrations required are performed within the algorithm, and the system is assumed to work using an uncalibrated camera.

1 Introduction

Viewing a three dimensional world projected onto a two dimensional plane causes the image produced by a conventional camera to be both ambiguous and difficult to interpret automatically. The ability to navigate around obstacles with the information from a single viewpoint is one that most living beings can develop, but is difficult to implement into an artificial system. In essence, the problem is one of learning the structure of a scene and then being able to maneuver within the scene to a predefined position exploiting visual cues only.

By inducing relative motion between the viewer and the scene, the image field is augmented to a velocity field, and the details of these deformations have been shown to encode further information about the structure of the scene which can be useful for visual navigation. Many representations of these image velocity fields have been attempted and an economical approach is to decompose the transformations into the first order differential invariants of the image velocity field [4, 11]—the curl, divergence and deformation components.

This paper will demonstrate a method of using these invariants to provide an accurate estimate of the relative distance to a surface by simply tracking a contour on the surface in the image plane during deliberate movements of the viewer relative to the scene.

Building on this technique the paper will then demonstrate a robust method of returning to a predefined target position in the scene. This is achieved by quantifying the transformation that has occurred due to a disturbance from this target position, and moving in such a way as to reverse this transformation. Such tasks could be used for many different problems, but notably in all visual navigation tasks.

2 Relative Depth from Image Divergence

Contours on a distant surface are naturally seen to deform as the viewer moves relative to the scene. Furthermore, these deformations have been shown to encode both the motion of the viewer and the structure (depth and orientation) of the surface. If the viewer motion is known, as is the case in most *active vision* applications [3, 4] the structure of the surface contour can be extracted.

2.1 Review

For the following analysis the image velocity field is decomposed into its first differential invariants [11], namely the curl (vorticity), divergence (dilation) and deformation (pure shear) components.

For a sufficiently small field of view and smooth change in view point, the differential invariants depend only on the viewer motion (translational velocity, $\mathbf{U}$, and rotational velocity, $\mathbf{\Omega}$), the surface depth, Z, and the relation between the viewing direction (unit vector along the optical axis, $\mathbf{Q}$) and the surface normal. A change of coordinate system would not have any effect on the results to be derived below [11, 9, 4].

Defining the vector quantities $\mathbf{A}$ and $\mathbf{F}$ as

$$\mathbf{A} = \frac{\mathbf{U} - (\mathbf{U} \cdot \mathbf{Q})\mathbf{Q}}{Z} \tag{1}$$

$$\mathbf{F} = \frac{f \nabla Z}{Z} \tag{2}$$

where f is the focal length of the camera and lens, it is possible to obtain a simple and useful relationship between the ego-motion of the viewer, and the divergence in the image velocity field [10]:

$$\mathrm{div}\mathbf{v} = \frac{2\mathbf{U} \cdot \mathbf{Q}}{Z} + \mathbf{F} \cdot \mathbf{A} \ . \tag{3}$$

2.2 Relation between Divergence and Time to Contact

The use of equation (3) in the field of visual servoing (and many other uncalibrated visual navigation tasks) stems from the special case of motion by the viewer along the optical axis, towards the surface. In this case, the ray direction, $\mathbf{Q}$, is parallel to the translation direction, $\mathbf{U}$. By ensuring that the length of $\mathbf{Q}$ is unity, equation (3) reduces to the form

$$\mathrm{div}\mathbf{v} = \frac{2\,|\mathbf{U}|}{Z} = \frac{2}{t_c} \tag{4}$$

where t_c is the time to contact[1].

2.3 Estimating the Image Divergence

It has been shown [4] that a robust calculation of the differential invariants can be made from the area moments of closed contours within the image plane. If a well-defined closed contour, at the centre of the image, on the surface of interest is

[1] The time duration before the viewer and the surface collide if the motion is held constant [8].

tracked as the viewer proceeds along the optical axis (as described above), the contour will appear to undergo an isotropic expansion. By estimating the divergence from the first moment of area of this closed contour, point and line correspondence calculations are avoided. This results in a simple contour integration to evaluate the divergence, which has the effect of averaging any image noise over a large area of the image and leads to reliable estimates. It can be shown that

$$\frac{da(t)}{dt} = \int\int_{a(t)} \nabla \cdot \mathbf{v} dx dy \approx a(t) \mathrm{div} \mathbf{v} \tag{5}$$

where $a(t)$ is the area enclosed by the contour.

Substituting equation (3) into (5) and solving the linear differential equation for the case where the viewer velocity, $\mathbf{U}$, is a constant (and therefore the time to contact decreases linearly) leads to the required solution (see also figure 2)

$$a(t) = a(0) \left[\frac{t_c(0)}{t_c(t)} \right]^2 . \tag{6}$$

2.4 Implementation

2.4.1 Algorithm

The following algorithm is adopted to provide a robust measurement of the time to contact to a surface (and hence a measurement of the distance to this surface).

The experiments were performed using a uncalibrated CCD camera held in the grippers of the manipulating arm of a Scorbot ER-7 robot arm. All processing is performed in real-time on a single Sun SPARCstation 20.

1. Initialise a closed B-spline snake around a contour on the surface [5]. The snake is allowed to “lock on” to the contour by deforming it according to the local intensity gradient. A region of search is specified, and this is reduced as the snake moves closer to the contour. In practice, 20 to 40 control points were used depending on the complexity of the contour (gradient discontinuities in the B-spline can be modelled by placing two or more control points at the same position).

2. The contour can now be tracked as the viewer moves, and a refinement is to constrain the deformations of the snake to affine transformations [7]—pure rotations, isotropic expansions, and shears about an arbitrary axis (see figure 1). A result of this assumption is that the computational load of the contour tracking is significantly reduced, and the snake is more robust to the distractions from background scenery and contours.

3. The viewer is deliberately moved towards to surface along the optical axis, at a constant velocity. The area enclosed by the B-spline snake is recorded at regular intervals by integrating around the contour. This is efficiently calculated as a function of the B-spline snake control point positions[4].

4. By linearising equation (6) and fitting the area measurements to a straight line via a least-squares technique, noise due to quantisation can be reduced.

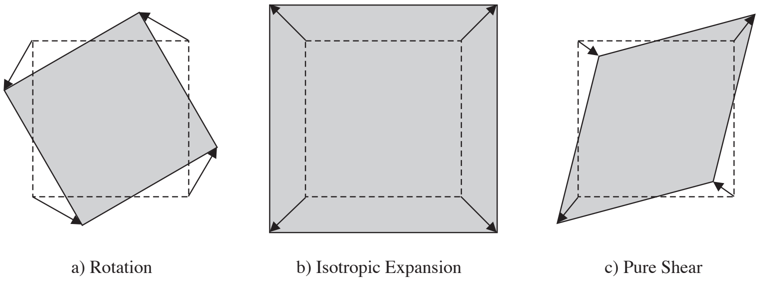

Figure 1: An affine transformation is a linear combination of a) rotations (defined by one angle in 2D), b) isotropic expansions (defined by one scale factor), and c) shears (defined by an axis, and a scale factor).

2.4.2 Results

Following the algorithm above, the camera was started at a distance of approximately 50cm from the surface. The time to contact was estimated to 1 part in 100 by taking 20 area measurements during the motion towards the surface (see figure 3).

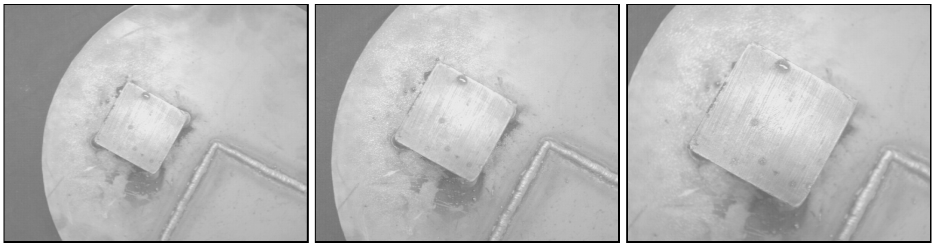

Figure 2: This is a sequence through a CCD camera as the viewer is deliberately moved towards a surface contour. As the sequence progresses, the apparent area enclosed by the contour is seen to increase and this provides enough information to extract the time to contact or relative depth of the surface (see also figure3).

3 Visual Servoing

The problem with the approach described above for determining the distance of a surface from a viewer is that it relies on the accuracy of only one estimate of the time to contact and no further feedback can be provided beyond this point. The orientation of the surface has also been ignored, but a similar approach can be considered with viewer motion perpendicular to the optical axis rather than parallel to it [3] (the measurements are very susceptible to quantisation and inaccuracies in the assumed viewer motion).

In this section, we will describe a method of closing the control loop, and therefore providing feedback information on the quality of the measurements. The

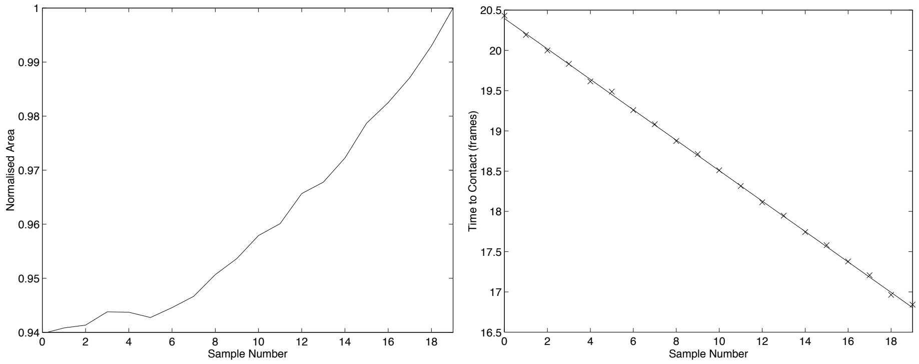

Figure 3: The first graph depicts the normalised area of the closed contour as it gets closer to the viewer at a constant rate. By linearising equation (6) and re-plotting the data, we have demonstrated how accurately the theory supports the experimental results.

result of this construction is that much more coarse estimates can be made, and the errors reduced iteratively.

3.1 Image Field Transformations

A camera in free-space has six degrees of freedom, but if we place the constraint that the optical axis should be fixated towards a fixed position on the surface, the coordinate system is reduced to four dimensions. For the purposes of this paper, we shall define the coordinate system as in figure 4.

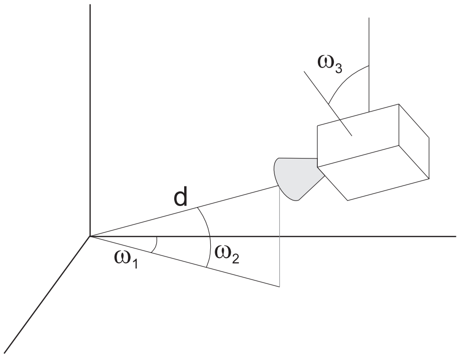

Figure 4: The camera coordinate system used in section 3.1 for a camera fixated on a point in the scene specifies three angle rotations, ω_1, ω_2 and ω_3, and a depth scale factor, d.

Any motion of the camera in one of these four independent directions induces deformations of the image, and therefore an image velocity field can be constructed. The requirement that the camera remains fixated on a point on the surface ensures that the velocity field does not contain any translational component, and can be written at a point (x, y) in the image as:

$$\begin{pmatrix} u \\ v \end{pmatrix} \approx \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{7}$$

where (u_x, u_y, v_x, v_y) are the first order partial derivatives of the velocity with respect to the subscript.

This transformation clearly has four degrees of freedom which correspond to the four degrees of freedom of the viewing camera.

3.2 Calibrating the Transformation Field

Following on from the previous section, the camera can make small movements in each of its four axes and record the transformations that take place in the image field. In general, the effect of "rolling" about the ω_3-axis is a rotation in the image plane, and a change of relative depth, d, induces an isotropic expansion (section 2).

Each of these perturbations will induce an image plane transformation, and four such transformations complete the parameterisation. These transformations can be expressed as 2-D matrices as in equation (7): $\mathbf{T}_{\omega_1}$, $\mathbf{T}_{\omega_2}$, $\mathbf{T}_{\omega_3}$, and $\mathbf{T}_d$ where the subscript references the axis in which the motion was made. As the movements of the camera are independent, it is clear that these matrices are non-singular (moving the camera back inverses the transformation) and associative (the order of the movements is not important, as the axes are independent) in multiplication.

If we now assume that the transformation field is linear for small perturbations of the camera, it can be inferred that, for example, three movements each causing a transformation $\mathbf{T}$ will result in an overall transformation $\mathbf{T}^3$. Further, a transformation in the ω_1 direction followed by a transformation in the ω_2 direction will induce an overall transformation of $\mathbf{T}_{\omega_1}\mathbf{T}_{\omega_2}$. A general movement of the camera given by the four dimensional vector, $(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d)$, will induce a transformation, F, where

$$\mathrm{F}(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d) = (\mathbf{T}_{\omega_1})^{\Delta\omega_1} (\mathbf{T}_{\omega_2})^{\Delta\omega_2} (\mathbf{T}_{\omega_3})^{\Delta\omega_3} (\mathbf{T}_d)^{\Delta d} \ . \tag{8}$$

3.3 Visual Feedback

Within the range of validity of equation (8), it should be possible to identify the motion in the four dimensional space that would have induced any affine transformation in the image plane. This is the basic requirement of a visual servoing system.

If the transformation observed is represented by matrix $\mathbf{S}$, the motion can be found by solving

$$\mathbf{S} = \mathrm{F}(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d) \ . \tag{9}$$

Unfortunately there does not appear to be an analytical solution to this problem, and therefore the solution must be found by an iterative solution, which in itself requires an "error function" to be defined (see section 3.5).

3.4 Computing the Transformations

The results obtained with this algorithm depend on being able to estimate the linear transformation between two images. This in turn requires correspondences to be tracked as the images change. Two methods are now described:

- The centroid of a closed contour is invariant as the image deforms according to the affine constraints. Therefore each closed contour that can be tracked in the image provides one correspondence between images. It is necessary to obtain at least two correspondences, and therefore this method requires two independent closed B-spline snakes to follow two contours on the surface of interest. In practice, most real surfaces have many suitable contours to track, which makes this method viable, but computationally expensive.
- A large number of matches can be obtained by considering the control points of one B-spline snake to provide correspondences between the images. The Aperture Problem [1] states that this method should not work correctly, as the snake has no way of extracting the component of tangential velocity to the curve. However the errors can be minimised by ensuring that the second differential of the contour position is as large as possible (i.e. that the curve does not have long smooth regions), and in practice the method has been shown to work well. Constraining the snake to deform affinely [7] also helps solve this problem.

Both these methods provide two sets of position vectors (one for the starting image, and one for the image after the camera has been moved)

$$\mathbf{y}_n = \begin{pmatrix} y_{n,1} \\ y_{n,2} \end{pmatrix} \text{ and } \mathbf{y}'_n = \begin{pmatrix} y'_{n,1} \\ y'_{n,2} \end{pmatrix} \tag{10}$$

where $n = 1 \ldots m$ and m is the number of correspondences found. The transformation $\mathbf{S}$ is required which will transform the first image onto the second with as little error as possible. We wish, therefore, to find the transformation, $\mathbf{S}$, which will minimise

$$\sum_{n=1}^{m} |\mathbf{y}_n - \mathbf{S}\mathbf{y}'_n|^2 \quad . \tag{11}$$

3.5 Approximating the Position Error

In section 3.3 we introduced the relationship between the overall observed transformation, $\mathbf{S}$, and the displacements, x, y, z and Θ. Equation (11) provides a good cost function, and by adjusting the variables until a minimum value is identified, very good estimates of the overall displacements are found. The problem can be expressed as:

$$\min_{x,y,z,\Theta} \sum_{n=1}^{m} \left| \mathbf{y}_n - (\mathbf{T}_x)^x (\mathbf{T}_y)^y (\mathbf{T}_z)^z (\mathbf{T}_\Theta)^\Theta \mathbf{y}'_n \right|^2 \quad . \tag{12}$$

The calibration transformations can be found from a least squares fit to equation (11).

Techniques for solving this optimisation are available, and both Hooke and Jeeves (see [2] for a description of this optimisation method) and the standard non-linear least squares optimisation methods have been shown to be suitable. It should be noted that equation (8) is only valid for small values of x, y, z and Θ which reduces the search region significantly.

3.6 Implementation

3.6.1 Algorithm

The following algorithm outlines one implementation of this method of visual servoing:

1. Initialise a closed B-spline snake around a contour feature on the surface of interest, and start to track the contour as described in section 2.4.1. While continually tracking the snake, the camera should be moved to the target position.

2. Record the target image as a set of points on the image defined by the position of the control points of the B-spline snake. A large number of points ensures that small errors due to the Aperture Problem will not be significant later in the experiment, and 20 points were used in our implementation.

3. Perturb the camera position to a new position, ensuring that the contour is continually tracked and centred in the image field.

4. Perform three[2] calibrating motions in each of the three dimensions and measure the resulting transformation of the B-spline snake for each of these motions.

5. The solution to equation (12) is found using the Hooke and Jeeves search method. The variables should be constrained to ensure they remain within the validity of equation (8). This part of the algorithm is the most computationally expensive, but in practice, the optimisation converged within 1 to 2 seconds.

6. Use the motion vector obtained from the previous step to move the camera towards the target, and repeat from step 4.

3.6.2 Results

Starting from a position about 50cm from the target position, the camera position converged very quickly towards to target. It was found that 3 iterations were necessary to place the camera to within 5mm of the target position (table 1 and figure 3.6.2).

[2] As described in section 2.4.1, the robot arm used for these experiments only had 5 degrees of freedom, and therefore the camera position has only 3 degrees of freedom if it is fixated on a point in the scene.

Iteration	*Vector to target*			*Distance to target (in cm)*
	x	y	z	
0	25.2	-10.0	-22.0	34.5
1	17.3	-8.7	-15.3	24.7
2	10.1	-3.0	3.0	11.0
3	1.8	0.0	1.0	2.1
4	0.1	0.2	0.8	0.8
5	0.1	-0.1	-0.4	0.4
6	0.0	0.0	0.3	0.3

Table 1: An example of the convergence rate for the affine visual servoing algorithm. The camera is started at a distance of 34.5cm from a target position, and over 3 iterations, it is maneuvered to within 1cm of the target. Further iterations oscillate within 1cm of the target.

4 Conclusions

The image divergence can be accurately extracted from a closed contour image sequence. It provides an excellent algorithm for estimating the time to contact between a surface and a viewer simply by tracking a contour on the as the camera moves towards the surface. A similar technique can be used to estimate the orientation of the surface, by moving the camera in a plane perpendicular to the optical axis. However the deformations are small and are not accurately estimated by the divergence alone.

Visual feedback provides a more robust method of estimating the camera position relative to a "target" position. The technique can be used on an uncalibrated system, and provides excellent results. Currently the calibration matrices must be obtained by deliberately moving the camera at the start of each iteration of the feedback loop. However future work will involve inferring these transformations from the data gathered during the previous iterations—calibrating moves will only be made if necessary (due to the lack of data from previous moves).

5 Acknowledgements

We gratefully acknowledge the support of Dr S. Takayanagi and M. Kimura of the Toshiba Corporation of Japan. The robot was donated by Olivetti Research Laboratory in Cambridge.

References

[1] F. Bergholm. Motion from flow along contours: a note on robustness and ambiguous case. In *International Journal of Computer Vision*, pages 3:396–415, 1989.

[2] G.S.G. Beveridge and R.S. Schechter. *Optimization: Theory and Practise.* McGraw-Hill, 1990.

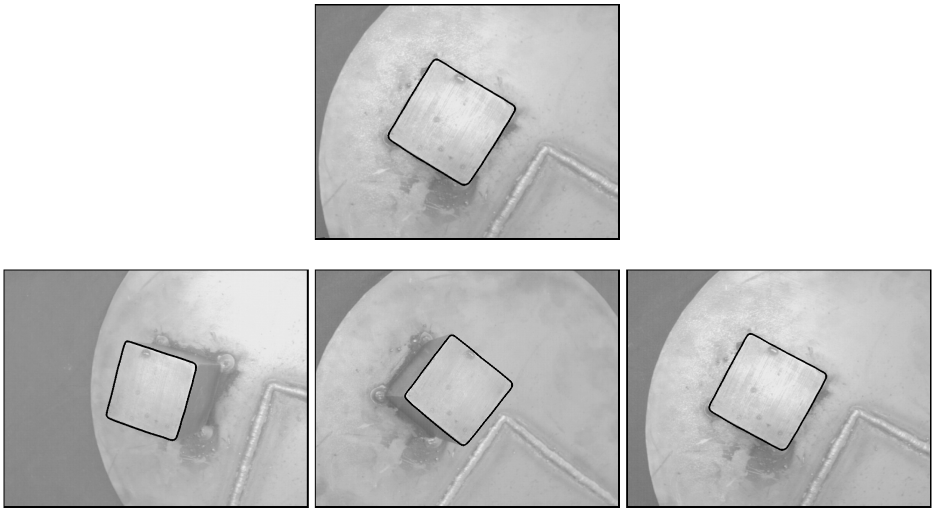

Figure 5: The first image depicts a target image, and the following three images depict the viewer image before the first iteration, after the first iteration, and after the third iteration. Clearly, the final image exhibits almost no transformations from the initial target image.

[3] R. Cipolla. *Active Visual Inference of Surface Shape.* Lecture Notes in Computer Science 1016 Springer-Verlag, 1995.

[4] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In G. Sandini, editor, *2nd European Conference on Computer Vision*, pages 187–202. Springer-Verlag, 1992.

[5] R. Cipolla and A. Blake. Surface shape from the deformation of apparaent contours. In *International Journal of Computer Vision*, pages 9(2):83–112, 1992.

[6] R. Cipolla and A. Blake. Image divergence and deformation from closed curves. To appear in *Int. Journal of Robotics Research*, 1996.

[7] N.J. Hollinghurst and R. Cipolla. Uncalibrated stereo and hand-eye coordination. *Image and Vision Computing*, pages 12(3):187–192, 1994.

[8] F. Hoyle. *The Black Cloud.* Heinemann, London, 1957.

[9] K. Kanatani. Structure and motion from optical flow under orthographic projection. In *Computer Vision, Graphics and Image Processing*, pages 35:181–199, 1986.

[10] J.J. Koenderink. Optic flow. In *Vision Research*, pages 26(1):161–179, 1986.

[11] J.J. Koenderink and A.J. Van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. In *Optica Acta*, pages 22(9):773–791, 1975.