MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

Affine Contour-Based Visual Servoing

G.M.T. Cross* and R. Cipolla†
Department of Engineering
University of Cambridge

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 (see figure 1). All the calibrations required are performed within the algorithm, and the system is assumed to work using an uncalibrated camera.

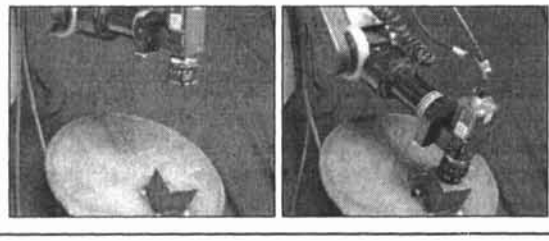

Figure 1: This figure depicts an example of the task to be performed by a visual servoing system. The camera in the first frame has detected the contour on the surface of the metal plate, and using visual cues, has "landed" on the surface in the second frame.

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 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 extracting the depth information from an image, and using this information to navigate within the scene.

By inducing relative motion between a viewer and the scene, the image plane 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 the image velocity field have been attempted, but the approximation of *affine* transformations has been shown to have many advantages [3, 5].

2 Background

It has been shown that for a small field of view, and smooth change of viewpoint the image velocity field is well approximated by a linear transformation [7] consisting of translations, rotations, isotropic expansions, and pure shears. For the purposes of this paper, we will be constantly tracking the deformation of a shape in the image, and the camera is fixated on the centroid of this shape. As a result, there are no translational components in the image velocity field.

The image velocity field, $\mathbf{v}(x, y)$, in the image can therefore be expressed as a simple 2-D affine transformation (see figure 2):

$$\mathbf{v}(x,y) = \begin{pmatrix} u \\ v \end{pmatrix} \approx \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad (1)$$

where (x, y) is the image position, and (u_x, u_y, v_x, v_y) are the first order partial derivatives (with respect to the subscript) of the image velocity field.

3 Theory

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 in

*Address: Robotics Research Group, Department of Engineering Science, Oxford, OX1 3PJ, United Kingdom. Email: 92gmtc@eng.cam.ac.uk

†Address: Department of Engineering, Cambridge, CB2 1PZ, United Kingdom. Email: cipolla@eng.cam.ac.uk

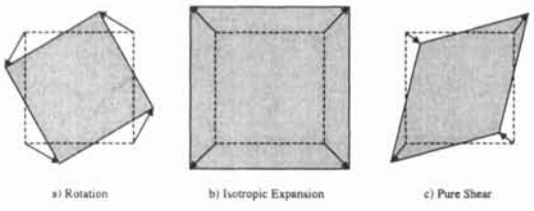

Figure 2: An affine transformation is a linear combination of a) rotations (defined by one angle in 2D), b) isotropic expansions (defined by one scale factor), and c) shears (defined by an axis, and a scale factor).

the scene, the coordinate system is reduced to four dimensions. For the purposes of this paper, we shall define the coordinate system as in figure 3.

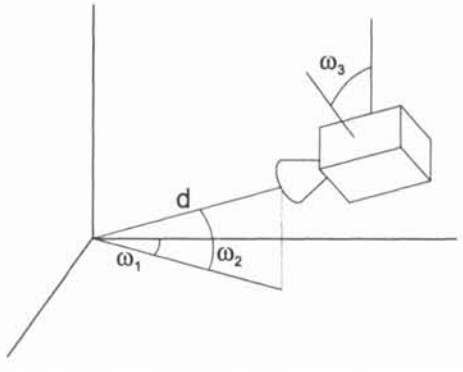

Figure 3: The camera coordinate system used in section 3.1 for a camera fixated on a point in the scene specifies three angle rotations, ω_1, ω_2 and ω_3, and a depth scale factor, d.

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 in the scene ensures that the velocity field does not contain any translational component (as in equation (1)), and therefore has four degrees of freedom which correspond to the four degrees of freedom of the viewing camera.

3.2 Transformation Field Calibration

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. As an example, the effect of "rolling" about the ω_3-axis is a rotation in the image plane, whilst a change of relative depth, d, induces an isotropic expansion.

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 (1): $\mathbf{T}_{\omega_1}$, $\mathbf{T}_{\omega_2}$, $\mathbf{T}_{\omega_3}$, 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) under 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, $\mathcal{F}$, where

$$\mathcal{F}(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d) = \left(\mathbf{T}_{\omega_1}\right)^{\Delta\omega_1}\left(\mathbf{T}_{\omega_2}\right)^{\Delta\omega_2}\left(\mathbf{T}_{\omega_3}\right)^{\Delta\omega_3}\left(\mathbf{T}_d\right)^{\Delta d} \quad (2)$$

3.3 Visual Feedback

Within the range of validity of equation (2), 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 interpolated by solving

$$\mathbf{S} = \mathcal{F}(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d) \ . \quad (3)$$

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 over image sequences. It is necessary to obtain at least two correspondences, and therefore this method requires two independent

closed B-spline snakes [4] 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] shows that this method should not be accurate, as the snake has no way of extracting the component of velocity tangential to the curve. However the errors can be minimized 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 according to the restrictions of affine transformations [6] 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)

$$\mathbf{y}_n = \begin{pmatrix} y_{n,1} \\ y_{n,2} \end{pmatrix} \quad \text{and} \quad \mathbf{y}'_n = \begin{pmatrix} y'_{n,1} \\ y'_{n,2} \end{pmatrix} \tag{4}$$

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 minimize

$$\sum_{n=1}^{m} |\mathbf{y}_n - \mathbf{S}\mathbf{y}'_n|^2 \quad . \tag{5}$$

3.5 Approximating the Position Error

In section 3.3 we introduced the relationship between the overall observed transformation, $\mathbf{S}$, and the displacements, $\Delta\omega_1$, $\Delta\omega_2$, $\Delta\omega_3$ and Δd. Equation (5) 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:

$$\min_{\mathbf{q}} \sum_{n=1}^{m} \left| \mathbf{y}_n - \left(\mathbf{T}_{\omega_1}\right)^{\Delta\omega_1} \left(\mathbf{T}_{\omega_2}\right)^{\Delta\omega_2} \left(\mathbf{T}_{\omega_3}\right)^{\Delta\omega_3} \left(\mathbf{T}_{d}\right)^{\Delta d} \mathbf{y}'_n \right|^2 \tag{6}$$

where $\mathbf{q} = (\Delta\omega_1, \Delta\omega_2, \Delta\omega_3, \Delta d)$.

The calibration transformations can be found from a least squares fit to equation (5).

Techniques for solving this optimization are available, and both Hooke and Jeeves (see [2] for a description of this optimization method) and the standard non-linear least squares optimization methods have been shown to be suitable. It should be noted that equation (2) is only valid for small values of $\Delta\omega_1$, $\Delta\omega_2$, $\Delta\omega_3$ and Δd which reduces the search region significantly.

3.6 Implementation

3.6.1 Algorithm

The experiments were performed using an 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.

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. 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 centered in the image field.

4. Perform three[1] calibrating motions in each of the three indepedant dimensions and measure the resulting image plane transformation of the B-spline snake for each of these motions.

5. The solution to equation (6) is found using the Hooke and Jeeves search method. The variables should be constrained to ensure they remain within the validity of equation (2). This part of the algorithm is the most computationally expensive, but in practice, the optimization converged within 1 to 2 seconds.

6. Use the motion vector obtained from the previous step to move the camera towards the target. Due to the approximations made in this algorithm, the motion vector is not exact, but it does ensure that the new camera position is closer to the target position. Repeating the algorithm allows the target position to be approached iteratively.

[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.

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 within 5mm of the target position (table 1 and figure 3.6.2).

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.

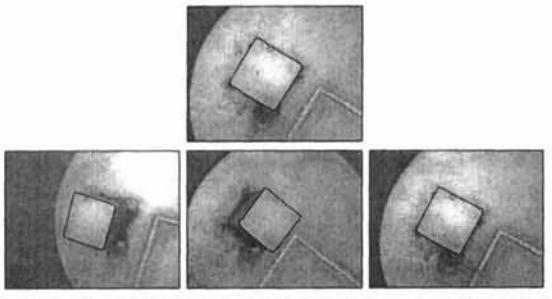

Figure 4: The first image (*top*) depicts a target image, and the following three images depict the viewer image before the first iteration (*bottom left*), after the first iteration (*bottom centre*), and after the third iteration (*bottom right*). Clearly, the final image exhibits almost no transformations from the initial target image.

4 Conclusion

Visual feedback provides a 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).

Acknowledgments

We would like to warmly thank the Toshiba Corporation for their financial support. The robot used during testing and implementation was financed by Olivetti.

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.

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

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

[5] R. Cipolla and A. Blake. Image divergence and deformation from closed curves. *International Journal of Robotics Research (to appear)*, 1997.

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

[7] 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.