[Journal Name], [Volumn Number], 1–22 ([Volumn Year])

Application of Lie Algebras to Visual Servoing

TOM DRUMMOND AND ROBERTO CIPOLLA
Department of Engineering, University of Cambridge,
Trumpington St, Cambridge, CB2 1PZ
fax: +44 1223 332662
email: twd20@eng.cam.ac.uk, cipolla@eng.cam.ac.uk

;

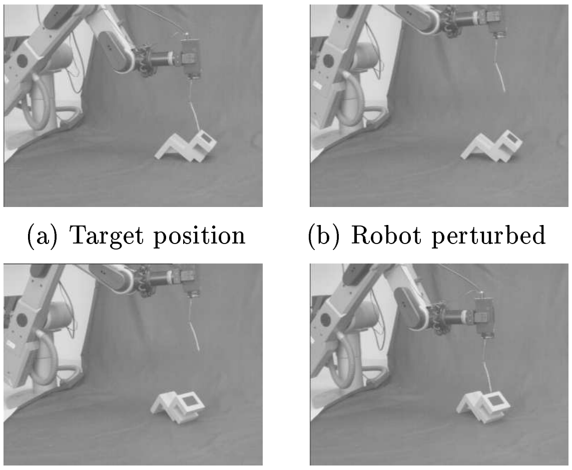
Abstract. A novel approach to visual servoing is presented, which takes advantage of the structure of the Lie algebra of affine transformations. The aim of this project is to use feedback from a visual sensor to guide a robot arm to a target position. The target position is learned using the principle of 'teaching by showing' in which the supervisor places the robot in the correct target position and the system captures the necessary information to be able to return to that position. The sensor is placed in the end effector of the robot, the 'camera-in-hand' approach, and thus provides direct feedback of the robot motion relative to the target scene via observed transformations of the scene. These scene transformations are obtained by measuring the affine deformations of a target planar contour (under the weak perspective assumption), captured by use of an active contour, or snake. Deformations of the snake are constrained using the Lie groups of affine and projective transformations. Properties of the Lie algebra of affine transformations are exploited to provide a novel method for integrating observed deformations of the target contour. These can be compensated with appropriate robot motion using a non-linear control structure. The local differential representation of contour deformations is extended to allow accurate integration of an extended series of small perturbations. This differs from existing approaches by virtue of the properties of the Lie algebra representation which implicitly embeds knowledge of the three-dimensional world within a two-dimensional image-based system. These techniques have been implemented using a video camera to control a 5 DoF robot arm. Experiments with this implementation are presented, together with a discussion of the results.

Keywords: visual servoing, active contours, affine geometry, lie groups, lie algebras

1. Introduction

The use of real-time video information for robotic guidance is increasingly becoming a more attractive proposition and is the subject of much research (Hager and Hutchinson 1996). Recent advances in the power and availability of image processing capabilities have made possible the tracking of complex features, such as surface contours (Kass, Witkin and Terzopoulous 1988), at frame or field rate on standard workstations. This has enabled visual servoing of sufficient accuracy that many useful tasks may now be accomplished.

Here we present an approach which takes advantage of the structure of the Lie algebra of affine transformations to provide an accurate, efficient and stable servoing system. A technical innovation is introduced which exploits the properties of Lie algebras to provide a method for integrating group transformations in a consis-



(a) Target position (b) Robot perturbed

(c) Part perturbed (d) Robot returns

Fig. 1. Problem statement

tent manner. The use of Lie algebras provides a substantial advantage over traditional image-based visual servoing techniques due to the manner in which the structure of the two-dimensional affine transformation group is used to implicitly embed three-dimensional knowledge within an image-based system.

The applications which motivate this work are tasks such as welding of automotive or ship parts. These tasks are characterised by the need to accurately place a tool onto a workpiece which may be inaccurately located in relation to the robot. The use of vision to assist in solving these tasks is becoming increasingly attractive because of the capability to respond accurately and rapidly to errors in part placement that are difficult to detect by other means.

The problem considered in this paper is to learn a target position and return to it after both the robot and the part have been perturbed as illustrated in Figure 1. Figure 1a shows the robot placed in the target position by the supervisor. Figure 1b shows the robot perturbed from the target position and Figure 1c shows the part perturbed also. Finally the robot returns to the target position relative to the perturbed part as seen in Figure 1d.

The remainder of this section reviews a number of issues in the design of visual servoing systems and outlines the approach taken in this paper. Section 2 introduces Lie transformation groups and shows how they may be used to constrain active contour models, or snakes. Section 3 introduces Lie algebras and derives a key result which allows group transformations to be integrated in a consistent manner. Section 4 describes the non-linear control system used to bind observed deformations to corrective robot motion. Section 5 ties this to a geometric description and shows Lyapunov stability before going on to give a simulated example which illustrates the benefits of the approach. Section 6 describes degeneracies within the system and shows how these can be overcome with the use of multiple contours. Finally, section 7 describes our implementation and presents results from a series of experiments.

1.1. *Background*

There are a number of important issues which must be addressed in the construction of a visually guided robotic system. The approach proposed here, makes use of image-based visual servoing (Sanderson, Weiss and Neumann 1987, Espiau, Chaumette and Rives 1992), in which the control loop is closed in the two-dimensional image domain rather than in the three-dimensional workspace. This is by contrast to systems which explicitly use the three-dimensional nature of the world (Basri, Rivlin and Shimshoni 1998, Wilson, Williams Hulls and Bell 1996) (often referred to as position-based visual servoing systems).

Three-dimensional visual servoing is typically achieved either by building some reconstruction of the world in a workspace co-ordinate system (Wilson et al. 1996), or by computing homographies and using knowledge of the epipolar geometry to servo the robot (Basri et al. 1998).

The two-dimensional approach works in the image domain, and computes error measurements directly, such as the (x,y) image location of target points (Espiau et al. 1992, Colombo and Allotta 1999). The system then attempts to move so as to minimise this observed error typically using a Jacobian which relates robot motions to changes in observables.

Hybrid approaches have also been attempted, for example (Malis, Chaumette and Boudet 1999) in which homographies are used to compute camera rotation in three dimensions, while translation is computed directly from observed errors in point correspondences in the 2D image.

One aim of this work is the tracking of moving targets. It is therefore not possible to assume that all observed change is due to motion of the robot. Systems which make this assumption can constrain the problem and use techniques such as continuous update or maintenance of the Jacobian (Cross and Cipolla 1996). Dynamic visual servoing on the other hand, must deal with moving targets and must implement some kind of closed loop control strategy (Corke and Goode 1996).

There are two natural locations in which to mount a visual sensor (Hutchinson, Hager and Corke 1996), namely static with respect to the world, and static relative to the robot. Our system uses the latter approach with a single camera mounted in the robot's end effector. This constrains the visual servoing problem since the sensor directly measures the relationship between the robot and workpiece. In this case the transformation between hand and eye must be computed for the three-dimensional approach (Horaud and Dornika 1995), although uncalibrated image-based techniques have also been used (Yoshimi and Allen 1995). There is the additional benefit of scaling; that as the robot approaches the target position, the size of features on the workpiece grows in the image plane. This enables greater accuracy to be obtained in positioning, but in general provides less control over lighting conditions and the use of multiple cameras than the alternative. The choice between one or more cameras has a significant impact on the design of a servoing system. The use of multiple cameras encourages the use of a reconstructive three-dimensional approach, although divergent stereo has also been used (Santos-Victor, Sandini, Curotto and Garibaldi 1995). Other schemes such as a camera free to rotate about the scene independently of the robot end effector have also been used (Kinoshita 1998) or mounted on a second robot arm (Bard, Laugier, Milési-Bellier, Troccaz, Triggs and Vercelli 1995).

This work uses the principle of teaching by showing, in which the supervisor shows the system the correct location by placing the robot in the desired relative pose to the target. The system then learns this pose by storing sufficient reference information, captured whilst in that pose, to characterise the three-dimensional relationship between the end effector and the workpiece. This is accomplished by observing one or more contours located on the surface of the workpiece. By recording the view of these contours from the correct target position, any observed deviations from this view can be detected and corrected for by appropriate robot motion.

Contours form a particularly useful feature to track for visual servoing for a number of reasons. Firstly, advances have been made in the robust tracking of contours such that they can be tracked reliably amongst clutter (Isard and Blake 1996). Secondly, an accurate measure of image motion can be obtained from contours due to the large number of measurements that can be made (usually one per control point on the contour), or by integration of normal velocities around the contour (Cipolla and Blake 1997). Finally, a planar contour is constrained to undergo affine deformation under the weak perspective assumption, and projective deformations under strong perspective. The affine deformation group, GA(2), has six dimensions and thus theoretically (and, as will be shown, practically) provides enough information to guide a robot through space. These motions can be described by the group of rigid three-dimensional motions, SE(3), also having six dimensions. The use of the affine/projective constraint separates this method from previous work using deformable models such as (Couvignon, Papanikolopoulos, Sullivan and Khosla 1996) which relies on a more traditional dynamical model of active contours. The affine transformation group of planar contour deformations has also been used previously for visual servoing (Cross and Cipolla 1996, Colombo and Allotta 1999) and this approach is distinguished by the way that the large scale structure of the group is used to provide consistent control. Other features have also been used for visual servoing such as corners (Basri et al. 1998), or contrived features such as discs (which act as points) (Espiau et al. 1992). An eigen-image approach has also been used, in which the features are the full set of pixel values within a region, projected onto a principal component subspace (Nayar, Nene and Murase 1996).

The robot control system is based on a Jacobian between the robot motions and the generators of the group of deformations of the contour. Proper-

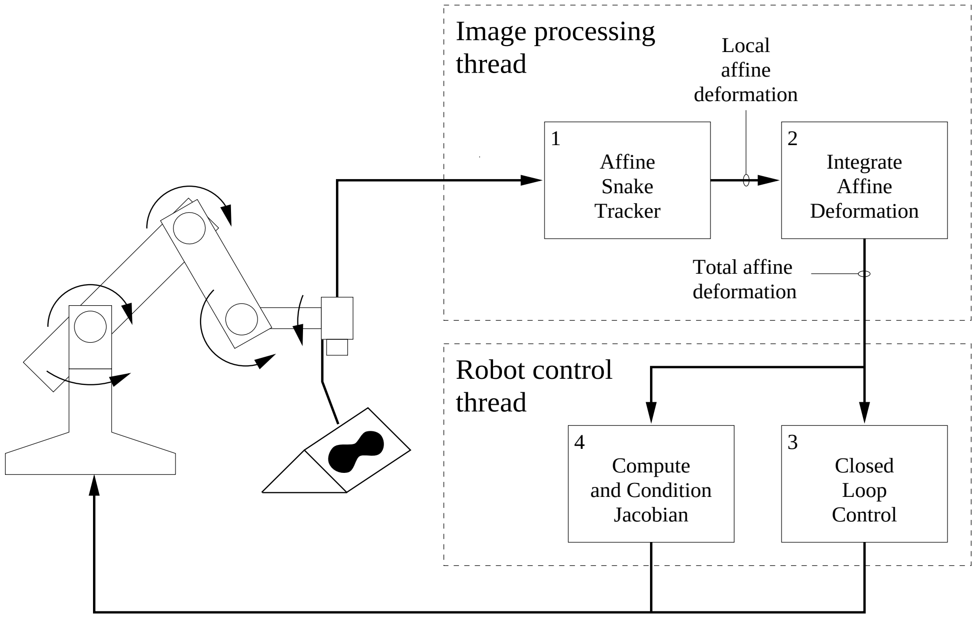


Fig. 2. System architecture

ties of the Lie algebra of this group are exploited to provide a consistent representation for integrating general affine (or other group) deformations to the contour. This approach allows a single Jacobian, computed once near the target location, to be used across a large range of perturbations.

1.2. System Overview

The system runs on a workstation which receives a live video feed from the robot camera and communicates directly with the robot controller. As illustrated in Figure 2, it comprises two separate threads which operate concurrently.

The first one is the image processing thread which performs all computations that must occur at video frame rate. In particular, there are two major modules that operate within this thread, namely the affine snake which tracks a contour on the workpiece and an integrator that computes the total observed affine deformation.

The second one is the robot control thread which operates on a much longer cycle time ranging from 0.5 – 1.5 seconds. This thread takes integrated errors from the contour tracker and computes desired robot motions to compensate. The Jacobian computation module which calibrates motions against observed deformations also resides within this thread. Figure 2 also shows the relationship between these modules, indicating the flow of information through the system, described briefly here.

1) The live video feed is delivered to the affine snake tracker module (described in more detail in Section 2) which computes a series of local transformations which describe the deformation of the contour of interest. Velocities in transformation space are computed and maintained in order to assist in tracking rapid motions.

2) The local transformations computed by the affine snake are integrated using knowledge of the Lie algebra of affine transformations to obtain an accurate measure of the the total transformation describing the current position of the contour (see Section 3).

3) The integral of affine transformation is passed to the robot control module which uses an affine-to-robot Jacobian, together with a non-linear control law described in Section 4 to compute robot motions.

4) The robot control thread also contains a calibration module which computes and conditions the Jacobian by correlating trial motions of the robot with integrated deformations of the contour of interest.

1.3. Lie Groups and Lie Algebras

The techniques presented in this paper make extensive use of concepts of Lie groups and their associated algebras in order to construct a consistent representation for the hand-eye co-ordination problem. This paper is particularly concerned with the use of Lie groups for tracking surface contours on the workpiece and the application of Lie algebras of these groups to integrate these changes in a consistent manner that can be used for robotic control. Concepts from Lie groups and algebras have previously been used to represent robot control structures (Park, Barrow and Ploen 1995, Murray and Sastry 1994). Lie groups and algebras are reviewed below in Sections 2 and 3.

2. Review of Lie Groups and Affine Snakes

A Lie group is a group which locally has the topology of $\mathbb{R}^n$ everywhere (a more precise definition may be found in (Varadarajan 1974) or (Sattinger and Weaver 1986), together with a more complete discussion of Lie groups and algebras). Lie groups provide a useful way of describing the transformations that a system can undergo and this section will show how they can be used in a generic way to constrain and assist a tracking mechanism by means of the vector fields that they generate in the image. There are a number of groups which are interesting for the purposes of this work and which can be defined by their action on $\mathbb{R}^m$ ($\mathbb{R}^2$ or $\mathbb{R}^3$ in this context). The groups SE(3) (Euclidean transformations in three dimensions with determinant 1), GA(2) (general affine transformations in two dimensions) and P(2) (projective transformations in two dimensions) are of particular relevance.

2.1. SE(3), GA(2) and P(2)

Each of these groups defines an allowable range of transformations on $\mathbb{R}^2$ or $\mathbb{R}^3$, with the identity element of the group performing the trivial transformation in which each point is mapped to itself. The dimension n of each group corresponds to the number of independent ways that it can make a small (infinitesimal) transformation. For SE(3) and GA(2) this is 6, whereas P(2) has 8 dimensions.

SE(3) is the group of rigid transformations in $\mathbb{R}^3$. This group is well understood in robotics since it describes the full range of possible motions of a rigid body, such as the end effector, in three-dimensional space. The six independent modes of transformation are typically described as:

1. x translation
2. y translation
3. z translation
4. Rotation about the x axis
5. Rotation about the y axis
6. Rotation about the z axis

These transformations are often referred to as screws (Murray and Sastry 1994), since the general transformation involves rotation about an axis in space, together with translation along it. The derivatives of screws, often referred to as twists correspond to vectors in the Lie algebra of SE(3) (discussed in more detail in Section 5). The transformations can be represented by matrices acting on homogeneous co-ordinates in three dimensions. Matrices corresponding to pure transformations in each of these six modes of deformation, parameterised by α are:

$$
\begin{aligned}
M_1 &= \begin{pmatrix} 1 & 0 & 0 & \alpha \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
M_2 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \alpha \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
M_3 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \alpha \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
M_4 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
M_5 &= \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
M_6 &= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{1}
$$

GA(2) is the group of all affine transformations on two-dimensional space. The six dimensions of the group are commonly broken down as follows:

1. x translation
2. y translation
3. Rotation about the origin
4. Dilation about the origin
5. Shear (squash y, stretch x)
6. Shear at 45 degrees to (5)

These transformations are typically represented by matrices in homogeneous co-ordinates. Those giving pure transformations in each of these six modes of deformation, parameterised by α are:

$$\begin{aligned} M_1 &= \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ M_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix} \\ M_3 &= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ M_4 &= \begin{pmatrix} \mathrm{e}^{\alpha} & 0 & 0 \\ 0 & \mathrm{e}^{\alpha} & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ M_5 &= \begin{pmatrix} \mathrm{e}^{\alpha} & 0 & 0 \\ 0 & \mathrm{e}^{-\alpha} & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ M_6 &= \begin{pmatrix} \cosh\alpha & \sinh\alpha & 0 \\ \sinh\alpha & \cosh\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{2}$$

GA(2) describes the transformations that the image of a planar object can undergo when viewed under weak perspective from a camera moving in three dimensions(Koenderink and van Doorn 1975). This group has three interesting subgroups, they are the translation group (transformations 1 and 2), the Euclidean group (1,2 and 3) and the similarity group (1,2,3 and 4).

P(2) is the group of all transformations on two-dimensional projective space. This is the group of all 3×3 matrix transformations on a two-dimensional point in homogeneous co-ordinates, with the convention that the third value in the co-ordinate is always scaled back to 1. This convention implies that scaled versions of the same matrix produce identical transformations, thus there are $9-1=8$ dimensions to this group. These 8 dimensions contain the 6 dimensions of GA(2) as a subgroup and include two additional dimensions which produce warping in the image and correspond to the movement of the horizon or vanishing line of the planar object being viewed. P(2) describes the transformations of a planar image under strong perspective. The additional dimensions correspond to the matrices:

$$M_7 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & 0 & 1 \end{pmatrix}, M_8 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \alpha & 1 \end{pmatrix} \tag{3}$$

2.2. Vector Fields and Lie Derivatives

The matrices in Equation (2) each describe a continuous one-dimensional family of transformations on $\mathbb{R}^2$, parameterised by α. Thus for each matrix, for each α, a point (x, y) is mapped to some point (x', y'). Setting α to zero generates the identity transformation: $(x', y') = (x, y)$. Differentiating with respect to α and evaluating at $\alpha = 0$ creates a vector field:

$$\tilde{L}_i = \left.\frac{\mathrm{d}M_i(\alpha)\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}}{\mathrm{d}\alpha}\right|_{\alpha=0} \qquad (1 \le i \le 6) \tag{4}$$

Since differentiation is linear, writing

$$G_i = \left.\frac{\mathrm{d}M_i(\alpha)}{\mathrm{d}\alpha}\right|_{\alpha=0} \quad \text{gives:} \quad \tilde{L}_i = G_i \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{5}$$

The matrices G_i are referred to as generators of the Lie group and form a basis for the Lie algebra discussed in more detail in Section 3. For GA(2), the generators are:

$$\begin{aligned} G_1 &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & G_2 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ G_3 &= \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & G_4 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ G_5 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} & G_6 &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{6}$$

Since the bottom row of each of these matrices is (0 0 0), the last component of $\tilde{L}_i$ is equal to zero. Writing L_i for the first two components of $\tilde{L}_i$ gives vector fields in the image plane. These vector fields are used to compute the affine transformation which describes the deformation of a contour in the robot's view. This is achieved through the use of affine snakes. The vector fields generated by the matrices in (6) are:

$$\begin{aligned} L_1 &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} & L_2 &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} & L_3 &= \begin{pmatrix} -y \\ x \end{pmatrix} \\ L_4 &= \begin{pmatrix} x \\ y \end{pmatrix} & L_5 &= \begin{pmatrix} x \\ -y \end{pmatrix} & L_6 &= \begin{pmatrix} y \\ x \end{pmatrix} \end{aligned} \tag{7}$$

Thus the vector fields for affine transformations are linear functions of position. In order to extend to full projective deformation, two quadratic vector fields are needed in addition. These are discussed in more detail in Section 2.4.

Algorithm 1: Transformation Group Lock

for each node, ξ, in snake **do**
  Compute unit normal to snake, $\hat{n}^\xi$
  Compute d^ξ = distance to edge along $\hat{n}^\xi$
end for
for $i, j = 1$ to num-generators **do**

$$O_i = \sum_\xi d^\xi (L_i^\xi \cdot \hat{n}^\xi)$$
$$M_{ij} = \sum_\xi (L_i^\xi \cdot \hat{n}^\xi)(L_j^\xi \cdot \hat{n}^\xi)$$

end for
Compute $E = M^{-1}O$ using SVD

2.3. Affine Snakes

In the system presented here, active contours (snakes) are used to track contours on the workpiece. These snakes are closed polygons with between 16 and 1024 vertices and are initialised by hand either inside or outside the contour of interest. The snake then expands out (or contracts) until strong edges are located in the image (see Figure 3). Once the snake has locked on to these edges, it becomes constrained to only undergo deformations within some transformation group of interest. Currently, the system supports five such groups (on $\mathbb{R}^2$). These are the translation group, the Euclidean group, the similarity group, the general affine group and the projective group (each of these being a subgroup of those that follow it).

This is achieved using Algorithm 1 which operates by measuring d^ξ, the distance to the contour in the image along the normal to the snake tangent at node ξ. Each such measurement provides a single estimated constraint on the deformation of the snake. Thus the measurement space has one degree of freedom per node on the snake. The measurement is then projected down onto the subspace defined by the transformation group of interest using singular value decomposition to produce a least squares fit. This computes the amount of each mode of deformation, E_j, to minimise

$$\sum_\xi \left(d^\xi - \textstyle\sum_j E_j (L_j^\xi \cdot \hat{n}^\xi)\right)^2 \tag{8}$$

Algorithm 2: Runge-Kutta Update

for each node, ξ, in snake **do**
  Let $\binom{x}{y}$ = position of ξth node in snake
  Compute:
    L_j = vector field of jth generator at $\binom{x}{y}$
    $\bar{C}_1 = \sum_j U_j L_j$
    L'_j = vector field at $\binom{x}{y} + \frac{1}{2}\bar{C}_1$
    $\bar{C}_2 = \sum_j U_j L'_j$
    L''_j = vector field at $\binom{x}{y} + \frac{1}{2}\bar{C}_2$
    $\bar{C}_3 = \sum_j U_j L''_j$
    L'''_j = vector field at $\binom{x}{y} + \bar{C}_3$
    $\bar{C}_4 = \sum_j U_j L'''_j$
    $\bar{C} = \frac{1}{6}(\bar{C}_1 + 2\bar{C}_2 + 2\bar{C}_3 + \bar{C}_4)$
  Let position of node $\xi = \binom{x}{y} + \bar{C}$
end for

which is the squared geometric distance between the observed contour and the transformed snake, integrated (summed) around the snake. The transformation group subspace is identified by computing $L_j^\xi \cdot \hat{n}^\xi$, the image motion created by generator j of the transformation group at node ξ of the snake, in a direction normal to the tangent of the snake at ξ. Taking ξ as the dimension index, these form a set of vectors identified by j and thus span a subspace with the same dimensionality as the transformation group. These vectors then provide the basis for the least squares linear regression procedure. Both the general affine deformation group and the projective group (for the projective compensation mechanism described

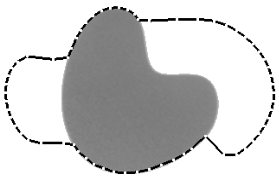


Fig. 3. Snake snapping on to contour after external initialisation

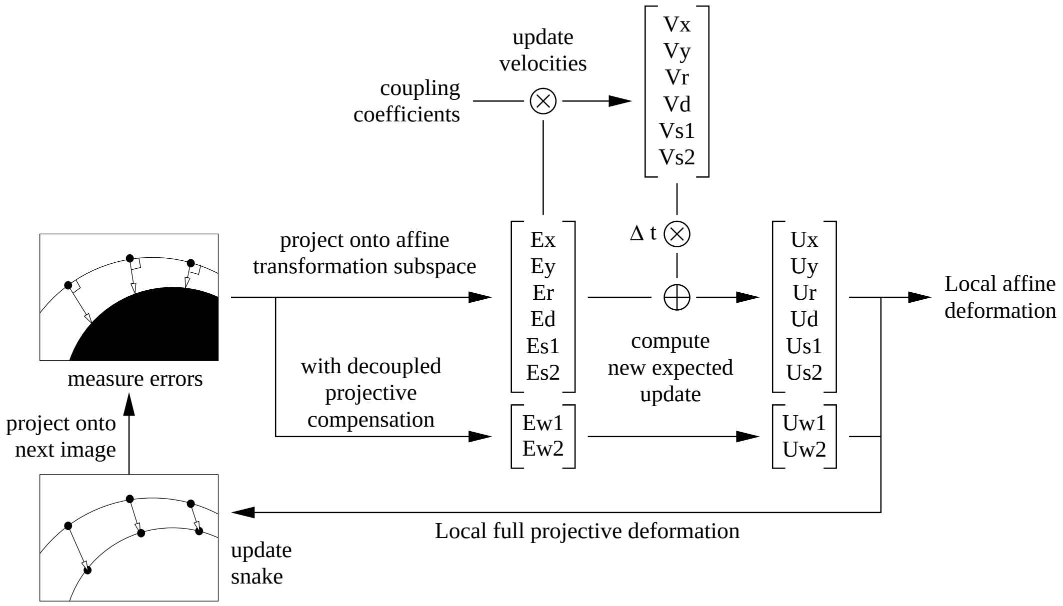


Fig. 4. Affine snake system

in Section 2.4) have been implemented using this procedure.

The snake tracks the contour using an estimate of the velocity of deformation of the contour in general affine transformation space, V. This is updated using the measured deviation between the prediction and observed contour, E, to provide a reliable new estimate of transformation velocity (see Figure 4). The velocity estimate is then combined with the observed deviation to give U which is used to update the contour to the predicted position at the next time step:

$$U_i^{t_2} = E_i^{t_1} + (t_2 - t_1)V_i^{t_1} \tag{9}$$

$$V_i^{t_2} = V_i^{t_1} + \alpha_i E_i^{t_2}/(t_2 - t_1) \tag{10}$$

where α_i are coupling constants chosen so as to damp oscillatory behaviour in the snake and $t_2 - t_1$ is the time elapsed in frames since the previous observation. This allows the system to cope with missing or intermittent video frames with graceful rather than catastrophic failure.

At each time step, the contour is updated using the update vector U_i from Equation (9) and the fourth order Runge-Kutta algorithm described in Algorithm 2. The new affine transformation errors E_i are then computed by Algorithm 1 and finally Equation (10) computes the new affine transformation velocity V_i.

2.4. Projective Compensation

The weak perspective assumption used to compute the affine deformation of the contour does not entirely hold in the situation presented here. The full space of deformations of a planar contour is properly described by the two-dimensional projective deformation group, outlined in Section 2.1. This means that if only affine deformations are used to track the contour, the tracker cannot deform to properly match the shape of the observed contour. This gives rise to tracking failures such as the example shown in Figure 5.

To manage this kind of problem, the tracking module incorporates a projective compensation mechanism which extends the range of deformations to the full projective group in such a way that the affine component of the observed deformation is essentially left intact. This is important because the projective distortion modes are not consistent across a large domain of observation of the contour, for example when the contour is viewed from a distance. This means that any control mechanism which incorporates this information may make large moves in an attempt to correct perceived projective distortions when this is inappropriate. In order to achieve this independence between the affine and projective modes, it is important to decouple the vector fields for

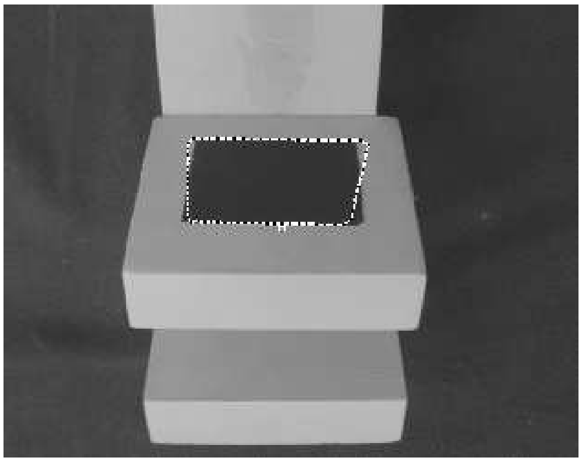


Fig. 5. Tracking failure in affine snake. The snake has tracked the contour as the block has been turned. Because it has been constrained to only undergo affine transformations, the snake no longer fits the contour properly due to the strong perspective deformation that is observed.

the projective warps from those of affine deformation. This can be largely achieved by computing the warp vector fields relative to the centroid of the snake. The two additional vector fields which provide this compensation are:

$$L_7 = \begin{pmatrix} x'y' \\ y'^2 \end{pmatrix},\ L_8 = \begin{pmatrix} x'^2 \\ x'y' \end{pmatrix} \qquad (11)$$

where x' and y' are relative to the centroid.

This approach works since by computing the projective transformations relative to the centroid, they operate independently of the mean motion of the snake. The remaining interactions between the affine and projective modes are due to the integral of the product of the quadratic projective term with the remaining linear component of the affine term. Since $\bar{x}' = \bar{y}' = 0$, this term should typically be small (for example, consider $\oint x'^2 y'$). In practise the disturbances to the affine approximation are sufficiently small that projective tracking can be achieved without interfering with the affine control process.

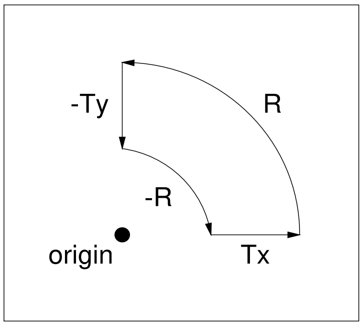
It would be possible to use this 8-dimensional measurement to drive the control system, however this is not advantageous for two reasons. Firstly, there measurement errors for the projective modes are much larger because the image motions they induce are comparatively small. Secondly, these measurements are not consistent over a large range of configurations (since the amount of projective deformation observed due to camera motion falls off much more rapidly with distance than the observed affine deformation).

2.5. Choice of Metric

The vector fields in Equations (7) and (11) can be scaled linearly without in principle affecting the solution for E_j in Algorithm 1. However, the choice of scale does affect the numerical stability of the algorithm, so the question arises as to how a suitable choice of basis fields, or metric, can be made. For semi-simple groups (see (Varadarajan 1974, Sattinger and Weaver 1986) for details of this), there is a natural choice of metric, the Cartan killing form. GA(2) is not semi-simple (since it contains an Abelian ideal, the two-dimensional translation group T(2)). In this case the natural metric is restricted to the quotient group GA(2)/T(2) and a separate metric must be chosen for the two dimensions of T(2). This means that the scale of the translational vector fields may be freely determined. The numerical stability can be improved by choosing them so that the condition number of the matrix M_{jk} in Algorithm 1 is minimised (see Section 6 - Intrinsic Degeneracy - for a description of the problems that can occur if this matrix is ill conditioned). In practise this means that L_1 and L_2 are scaled up so that they have a similar typical magnitude to vectors L_3 to L_8.

3. Review of Lie Algebras and Affine Integration

There is a natural representation for affine transformations in terms of matrices in homogeneous co-ordinates, such as those shown in Equation (2). However, an alternative representation is to use a co-ordinate system to represent small transformations near the identity. In this co-ordinate system, the axes correspond to the different modes of deformation and affine transformations are specified as a weighted sum of the group generators added to the identity. This leads naturally to a local vector space representation for infinitesimal transformations, in which an affine transformation matrix, A can be obtained from a vector, $\mathcal{A}$ by the expo-



$$\begin{bmatrix}\alpha\\0\\0\\0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\0\\\pi/2\\0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\-\alpha\\0\\0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\0\\-\pi/2\\0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\0\\0\end{bmatrix} \,! \;\; (\text{not} \begin{bmatrix}\alpha\\-\alpha\\0\\0\\0\\0\end{bmatrix})$$

Fig. 6. Adding vectors in a Lie algebra

nential map:

$$A = e^{\sum_i \mathcal{A}_i G_i} \tag{12}$$

where

$$e^X = I + X + \frac{1}{2}X^2 + \frac{1}{6}X^3 + \ldots$$

For small $\mathcal{A}_i$ this can be approximated by the linear term:

$$A = I + \sum_i \mathcal{A}_i G_i \qquad (\mathcal{A}_i \ll 1) \tag{13}$$

Due to the linearity, the G_i form a basis for a vector space, known as a Lie algebra. Formally, a Lie algebra is a vector space together with a bilinear anti-symmetric operator, the Lie bracket, satisfying the Jacobi identity:

$$[\mathcal{A}, [\mathcal{B}, \mathcal{C}]] + [\mathcal{B}, [\mathcal{C}, \mathcal{A}]] + [\mathcal{C}, [\mathcal{A}, \mathcal{B}]] = 0 \tag{14}$$

Where a Lie algebra is obtained from a group in the manner identified above, the Lie bracket is defined by the commutator of the generators:

$$[G_i, G_j] = G_i G_j - G_j G_i \tag{15}$$

so

$$\mathcal{C} = [\mathcal{A}, \mathcal{B}] \tag{16}$$

is given by

$$\sum_k \mathcal{C}_k G_k = \sum_{i,j} \mathcal{A}_i \mathcal{B}_j (G_i G_j - G_j G_i) \tag{17}$$

That the algebra is closed as a vector space with this definition follows from the fact that the G_i are the generators of a group. For GA(2) the commutation relations of $[G_i, \mathbf{G}_j]$ are shown in the following table:

	$\mathbf{G}_1$	$\mathbf{G}_2$	$\mathbf{G}_3$	$\mathbf{G}_4$	$\mathbf{G}_5$	$\mathbf{G}_6$
G_1	0	0	$-G_2$	$-G_1$	$-G_1$	$-G_2$
G_2	0	0	G_1	$-G_2$	G_2	$-G_1$
G_3	G_2	$-G_1$	0	0	$2G_6$	$-2G_5$
G_4	G_1	G_2	0	0	$2G_5$	0
G_5	G_1	$-G_2$	$-2G_6$	$-2G_5$	0	$-2G_3$
G_6	G_2	G_1	$2G_5$	0	$2G_3$	0

The infinitesimal representation of the Lie algebra can be extended by considering the exponential map for non-infinitesimal transformations. This defines a mapping from the Lie algebra onto the group, thus providing a convenient way of representing affine transformations as vectors which, in this scenario, can be used to drive the robot control system. Because higher order terms are incorporated into the vector space by this method, it is no longer possible to naïvely add vectors together to obtain a vector representing the composite transformation. This difficulty is illustrated in Figure 6.

A new addition law must be found which preserves the non-commutativity of matrix multiplication so that the sum of two vectors is the vector representing the true product of the transformations. Thus

$$\mathcal{A} = \mathcal{B} \oplus \mathcal{C}$$

implies

$$e^{\sum_i \mathcal{A}_i G_i} = e^{\sum_i \mathcal{B}_i G_i} e^{\sum_i \mathcal{C}_i G_i}$$

writing $\bar{A} = \sum_i \mathcal{A}_i G_i$, ($\bar{B}$,$\bar{C}$ similarly) gives

$$I + \bar{A} + \tfrac{1}{2}\bar{A}^2 + \dots = (I + \bar{B} + \tfrac{1}{2}\bar{B}^2 + \dots)(I + \bar{C} + \tfrac{1}{2}\bar{C}^2 + \dots)$$

this is solved by setting

$$\begin{aligned} \bar{A} &= \bar{B} + \bar{C} \\ &+ \tfrac{1}{2}[\bar{B}, \bar{C}] \\ &+ \tfrac{1}{12}[\bar{B} - \bar{C}, [\bar{B}, \bar{C}]] \\ &+ O(\bar{B}, \bar{C})^4 \end{aligned} \tag{18}$$

(the derivation of this is given in Appendix B). This expression applies to the matrices $\bar{A}$, $\bar{B}$ and $\bar{C}$. It also therefore holds in the vector space of the Lie algebra replacing the matrix commutator with the Lie bracket. This is an important result because the correction terms provide a method of consistently adding together vectors which represent group transformations such that the result is correct in the exponential map.

This can then be used to integrate a series of affine deformations, so that the integral faithfully represents the total deformation. Equation (18) is used to add the vector representing the local inter-frame deformation B (computed by the affine snake module) to that representing the current value of the integral C. Figure 7 illustrates the importance of including these correction terms in the integral. The light contour shows the integral synthesised separately from a reference copy of the snake, stored at the beginning of the experiment. A series of moves were applied to the contour, each move being a large motion in some mode of deformation. These were chosen so that the sequence of deformations would not commute. The system without the correction terms diverged rapidly (Figure 7(b)) while the system with the corrections incorporated maintained a good track even over an extended sequence (Figure 7(a)).

One of the key advantages of this representation (as opposed to parameterising in terms of the elements of the matrix in homogeneous co-ordinates, for example) is that straight lines through the origin of the vector space are geodesics in the manifold of the Lie group. It is this property that allows a Jacobian computed in one place to operate correctly across a large range of perturbations. A further benefit is that velocities and observed errors can be combined correctly to enhance the tracking capabilities of the real time affine snake.

The simulated example in Section 5.2 illustrates the benefits that stem from applying geodesic transformation trajectories by comparison to the more common method of computing linear image trajectories or parameterising in terms of the affine transformation matrix elements.

4. Robot Control

The robot control thread takes the integral of the affine transformation in the form of a 6-vector, A and uses an affine-to-motion Jacobian (J^{-1}, the matrix of partial derivatives of robot motion with respect to affine image transformations) to generate robot motions.

4.1. *Computing the Jacobian*

The affine-to-motion Jacobian is computed from a series of trial robot motions performed in the vicinity of the target location. The Jacobian must be learned since the system has no prior knowledge

(a) With Correction Terms (after 60 moves) (b) No Correction (after 3 moves)

Fig. 7. Effect of correction terms on integral

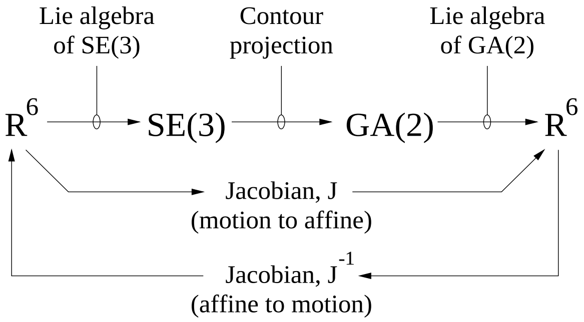


Fig. 8. Computing the control Jacobian

of the relative position of the plane of the contour with respect to the camera.

Each of the six possible robot motions produces in a vector of integrated deformation, resulting in a 6×6 motion-to-affine Jacobian, J. This is inverted using SVD to compute the affine-to-motion Jacobian, J^{-1} (see Figure 8). In the case of servoing from two contours, as discussed in Section 6.1, this becomes the 6×12 pseudo-inverse.

4.2. Control System

This inverse Jacobian can then be used to compute compensatory motion from the observed integral of affine deformation. The naïve control law is then simply:

$$\Delta R = -J^{-1} A \qquad (19)$$

where J^{-1} is the affine-to-motion Jacobian and A is the integrated total affine transformation giving ΔR as a 6-vector describing the desired change in robot co-ordinates. This differs from the traditional control law since A is computed using the affine integration technique described in Section 3. In the traditional approach, if M is the matrix describing the affine transformation from the target view to the current observation, then A is defined by $M = I + \sum_i A_i G_i$. This approach replaces this with $M = \exp(\sum_i A_i G_i)$. It can be seen that when the current observation is near the target view, then the two approaches converge since the exponential map may be approximated by the linear term. However, when the transformation M is large, the two control laws differ significantly. This difference is illustrated with a simulated example presented in Section 5.2.

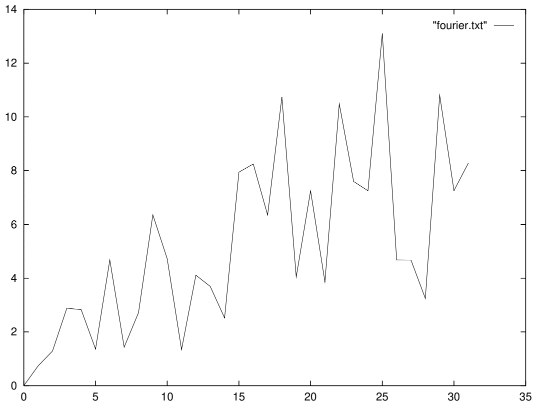


Fig. 9. Plot of power spectrum of motions on axis 4

In practise, the simple unit-gain approach is unstable over long distances due to errors in measurement and also over short distances due to inaccuracies in the robot controller. The former problem has been solved by limiting the motion to twice the trial motion used for calibration (this also acts as a safety mechanism preventing potentially dangerous large robot motions).

The second problem which leads to oscillatory behaviour about the target position proved more difficult to deal with. In an attempt to model the system as a second order differential equation, the Fourier transforms of robot motion values captured during oscillations were computed. These transforms showed a power spectrum approximately linearly proportional to frequency (i.e. the derivative of a white noise spectrum) (see Figure 9). This suggests that each attempt to reposition the robot results in a random error with low correlation from one time step to the next. This problem has been ameliorated by employing a nonlinear control law in which the gain is lowered as the target position is approached.

In order to determine the roll-off in the gain control, a series of experiments were performed in which the system was operated with a set gain between 0 and 1. The standard deviation of oscillations was plotted against gain (see Figure 10). A quadratic was fitted (by hand) and the gain was selected to be a linear function of the estimated distance to target. The gain was chosen so that the standard deviation of oscillations observed at that gain divided by the gain was half the estimated distance to the target. This resulted in the gain curve shown in Figure 11.

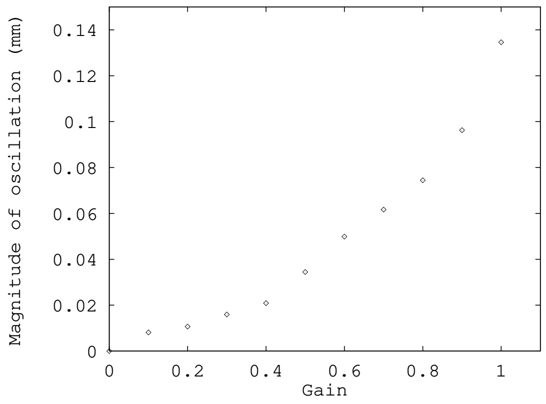


Fig. 10. Plot of oscillations against gain

5. Geometric Interpretation

In this section, the relationship between infinitesimal motions in SE(3) to infinitesimal transformations in GA(2) is discussed. This is used to demonstrate the benefits yielded by the use of the Lie algebra of affine transformations in terms of providing a robust Jacobian. Infinitesimal SE(3) motions are generated by the following matrices (using homogeneous co-ordinates), which are the derivatives of the matrices, M_i with respect to α in (1) evaluated at $\alpha = 0$.

$$E_1 = \begin{bmatrix} 0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix}, E_2 = \begin{bmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0 \end{bmatrix}, E_3 = \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0 \end{bmatrix},$$
$$E_4 = \begin{bmatrix} 0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0 \end{bmatrix}, E_5 = \begin{bmatrix} 0&0&1&0\\0&0&0&0\\-1&0&0&0\\0&0&0&0 \end{bmatrix}, E_6 = \begin{bmatrix} 0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \qquad (20)$$

Applying one of these generators to the homogeneous co-ordinate vector representing a point in three-dimensional space yields the Lie derivative of the co-ordinates with respect to the generated motion. For example the Lie derivative (motion) of a point, $\mathbf{p}$, under Y axis rotation (generated by E_5) is given by:

$$\mathcal{L}_{E_5}\mathbf{p} = \begin{pmatrix} X' \\ Y' \\ Z' \\ 0 \end{pmatrix} = E_5\mathbf{p} = \begin{pmatrix} Z \\ 0 \\ -X \\ 0 \end{pmatrix} \qquad (21)$$

The X,Y,Z co-ordinate system is chosen such that its origin is at the optic centre of the camera, with the Z axis along the optical axis. This means (given a normalised camera) that $\mathbf{p}$ projects onto the camera plane at $\mathrm{p} = \binom{X/Z}{Y/Z}$. The Lie derivatives of p with respect to the E_i can now be computed since:

$$\mathrm{p}' = \begin{pmatrix} X'/Z - XZ'/Z^2 \\ Y'/Z - YZ'/Z^2 \end{pmatrix} \qquad (22)$$

Without loss of generality the the surface normal of the contour is considered to be in the Y-Z plane (see Figure 12).

If the curve is specified in local co-ordinates as $\binom{u}{v}$ then the mapping into the three-dimensional co-ordinate system and into the image is:

$$\begin{pmatrix} u \\ v \end{pmatrix} \rightarrow \begin{pmatrix} u \\ v\cos\theta \\ d + v\sin\theta \end{pmatrix} \rightarrow \begin{pmatrix} \frac{u}{d+v\sin\theta} \\ \frac{v\cos\theta}{d+v\sin\theta} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad (23)$$

Only the affine part of observed transformations is computed here. This approximation requires

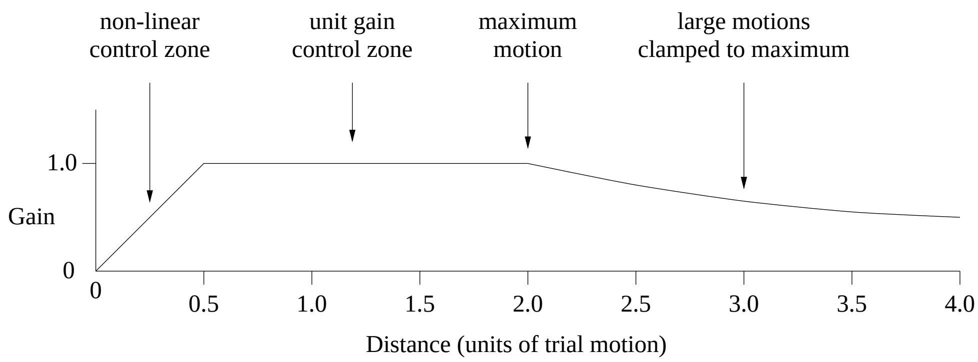


Fig. 11. Gain curve for non-linear control system

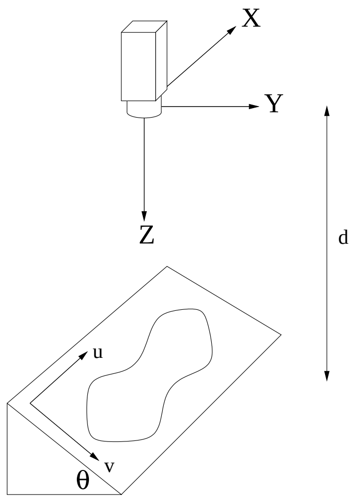


Fig. 12. Geometry of planar contour

that $u, v \ll d$. Using this, the Lie derivatives of $\binom{x}{y}$ can be computed for each generator of motion of the camera in SE(3) and the affine transformation component of the observed contour represented in terms of the vector fields L_i:

X translation (E_1):

$$\begin{aligned}
&& \mathcal{L}_{E_1}\mathbf{p} &= \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_1}x &= 1/Z = \frac{1}{d + v\sin\theta} \\
&& &= \frac{1}{d}(1 - y\tan\theta) \\
\text{and} && \mathcal{L}_{E_1}y &= 0 \\
\text{so} && \mathcal{L}_{E_1}\mathrm{p} &= \frac{1}{d}L_1 + \frac{\tan\theta}{2d}(L_3 - L_6)
\end{aligned} \tag{24}$$

Y translation (E_2):

$$\begin{aligned}
&& \mathcal{L}_{E_2}\mathbf{p} &= \begin{pmatrix} 0\\1\\0\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_2}x &= 0 \\
\text{and} && \mathcal{L}_{E_2}y &= 1/Z = \frac{1}{d}(1 - y\tan\theta) \\
\text{so} && \mathcal{L}_{E_2} &= \frac{1}{d}L_2 + \frac{\tan\theta}{2d}(L_5 - L_4)
\end{aligned} \tag{25}$$

Z translation (E_3):

$$\begin{aligned}
&& \mathcal{L}_{E_3}\mathbf{p} &= \begin{pmatrix} 0\\0\\1\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_3}x &= -X/Z^2 = -x/Z \approx -x/d \\
\text{and} && \mathcal{L}_{E_3}y &= -Y/Z^2 = -y/Z \approx -y/d \\
\text{so} && \mathcal{L}_{E_3}\mathrm{p} &\approx -\frac{1}{d}L_4
\end{aligned} \tag{26}$$

Rotation about X axis (E_4):

$$\begin{aligned}
&& \mathcal{L}_{E_4}\mathbf{p} &= \begin{pmatrix} 0\\-Z\\Y\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_4}x &= -XY/Z^2 = -xy \\
\text{and} && \mathcal{L}_{E_4}y &= -1 - Y^2/Z^2 = -1 - y^2 \\
\text{so} && \mathcal{L}_{E_4}\mathrm{p} &\approx -L_2
\end{aligned} \tag{27}$$

Rotation about Y axis (E_5):

$$\begin{aligned}
&& \mathcal{L}_{E_5}\mathbf{p} &= \begin{pmatrix} Z\\0\\-X\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_5}x &= 1 + X^2/Z^2 = 1 + x^2 \\
\text{and} && \mathcal{L}_{E_5}y &= YX/Z^2 = xy \\
\text{so} && \mathcal{L}_{E_5}\mathrm{p} &\approx L_1
\end{aligned} \tag{28}$$

Rotation about Z axis (E_6):

$$\begin{aligned}
&& \mathcal{L}_{E_6}\mathbf{p} &= \begin{pmatrix} -Y\\X\\0\\0 \end{pmatrix} \\
\text{so} && \mathcal{L}_{E_6}x &= -Y/Z = -y \\
\text{and} && \mathcal{L}_{E_6}y &= X/Z = x \\
\text{so} && \mathcal{L}_{E_6}\mathrm{p} &= L_3
\end{aligned} \tag{29}$$

These equations form the Euclidean-to-affine connection of the system. That is, they form a map between the tangent space (at the identity) of SE(3), for which the E_i form a basis, to the tangent space of GA(2), giving the motion to affine Jacobian. This can then be inverted to give the affine to motion Jacobian as:

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & \frac{-2d}{\tan\theta} \\
0 & 0 & 0 & 0 & \frac{2d}{\tan\theta} & 0 \\
0 & 0 & 0 & -d & -d & 0 \\
0 & -1 & 0 & 0 & \frac{2}{\tan\theta} & 0 \\
1 & 0 & 0 & 0 & 0 & \frac{2}{\tan\theta} \\
0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix} \tag{30}$$

This mapping is in general dependent on the d, θ configuration, although Equations (27–29) are independent, reflecting the well known fact that motion due to camera rotation is independent of the scene geometry. Thus, for these components, the Jacobian will be valid across the entire configuration space of the contour. Equation (26), however, depends on the distance, d, to the contour and Equations (24) and (25) also depend on the angle, θ, of the inclination of the plane of the contour. Despite these dependencies, however, the Jacobian is still stable across a large range of the configuration space. This is due to the fact that the change in the value of the θ, d dependencies over a large range of configurations is small enough that the learned Jacobian can still operate reasonably effectively. In practise, the Jacobian provides stable behaviour for $d > 0.7 \times d_{\text{target}}$, and for $20^o < \theta < 70^o$.

5.1. *Lyapunov Stability*

In order to show Lyapunov stability, it is necessary to examine the relationship between the Jacobian computed at the target and the Jacobian as it would be computed at the current position. The positive definite energy function used for this example is simply the norm of the vector describing the total observed deformation.

$$V(A) = \|A\| \tag{31}$$

Then if J is the Jacobian obtained from calibration in the target position and J' is the Jacobian that would be computed in the current position, the actual motion is in the direction $-J_{ij}A_j$, whilst $\|A\|$ is minimised in the direction $-J'_{ij}A_j$. Thus:

$$\Delta\|A\| < 0 \Leftarrow (J_{ij}A_j).(J'_{ik}A_k) > 0 \tag{32}$$

Note that in Equations (24–29) the only varying coefficients are powers of d and $\tan\theta$. Since $d > 0$ (the target is in front of the camera), the only coefficient that can change sign is $\tan\theta$ (which changes sign when θ does). This corresponds to the two-fold ambiguity of the weak perspective assumption. Thus the system is locally $(0 < \theta < 90^o)$ asymptotically stable since $\|A\| = 0$ is unique. Clearly this first order demonstration of Lyapunov stability only applies to the ‘look and move’ style of implementation used here. In the case of a dynamic system, a second order energy function must be used together with appropriate manipulation of the non-linear gain shown in Figure 11.

5.2. *A Simulated Example*

In order to compare the use of the Lie algebra for representing transformations to the more traditional approach which uses the entries in the transformation matrix as a co-ordinate system, a simulated example was constructed.

In the traditional approach, the aim is to drive the affine transformation matrix to the identity along a path that is linear in the matrix entries. Thus if the observed transformation corresponds to the matrix M, then the transformation error is $A' = M - I$ and the path back to the identity (parameterised by t is:

$$M(t) = I + (1-t)A' \tag{33}$$

This corresponds to the path in which each point travels to its desired position along a linear image trajectory, as in (Colombo and Allotta 1999). As will be shown, this approach can lead to some unexpected robot behaviour.

By contrast, the approach presented here aims to drive the affine transformation matrix to the identity along a group geodesic. Thus if M is represented as $M = \exp(\sum_i A_i G_i)$ then the path is given by:

$$M(t) = \exp((1-t)\textstyle\sum_i A_i G_i) \tag{34}$$

In this example, two contours were synthesised (shown in Figure 13) and both idealised trajectories computed. Figure 14 shows a comparison between the two approaches. Note that the scale of the contour changes in the linear approach.

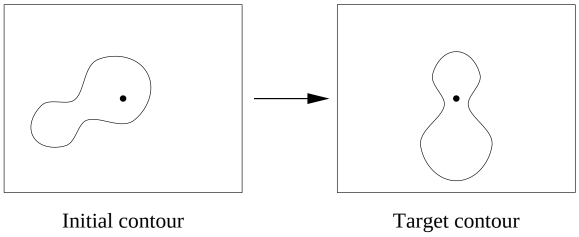


Fig. 13. Simulated contours

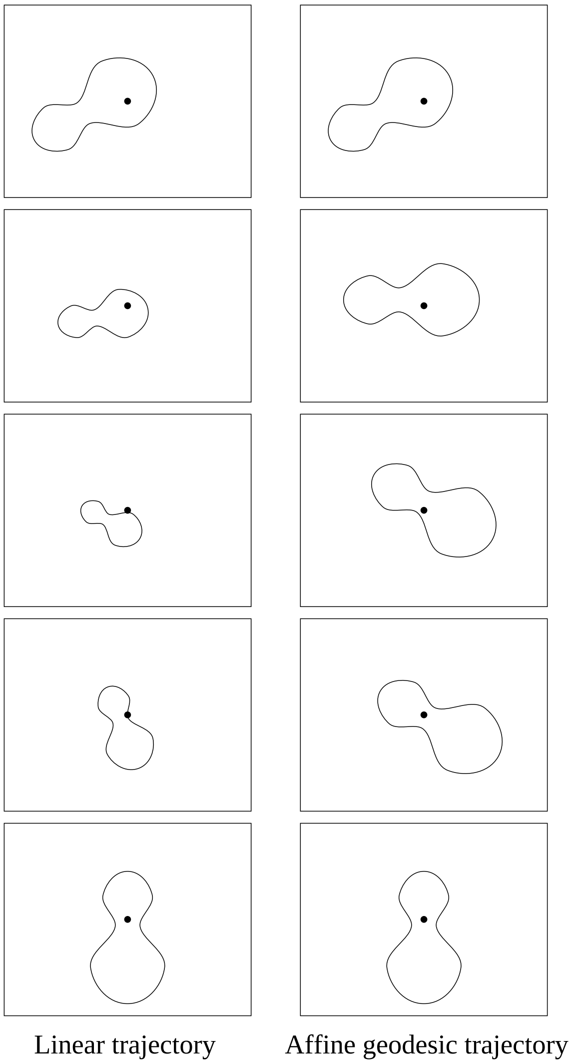


Fig. 14. Simulated image trajectories

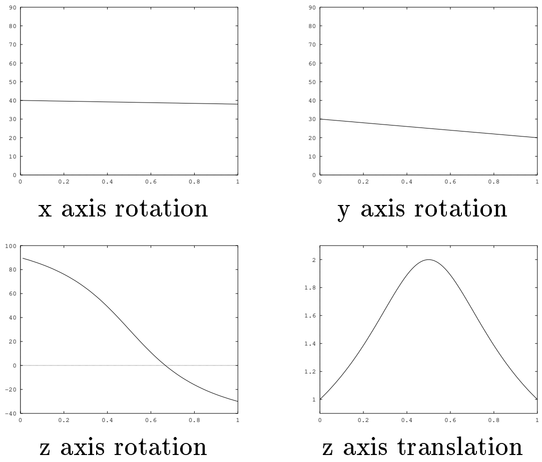


Fig. 15. Robot trajectories for linear case

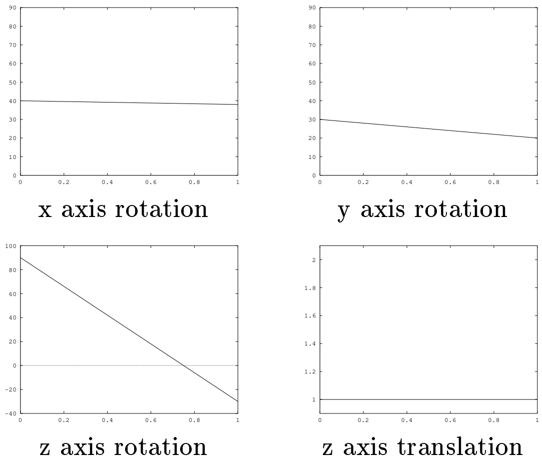


Fig. 16. Robot trajectories for geodesic case

In the affine geodesic case, the trajectory consists solely of translations and rotations, whereas in the linear case, there is a substantial component of dilation (first negative, and then positive). When these image trajectories are used to compute simulated camera (and therefore robot) trajectories in SE(3), the affine geodesic case corresponds to rotation of the camera about a fixed axis at constant speed. By contrast, the linear image trajectory corresponds to a substantial z translation, first away, and then towards the image, returning to the original z position, together with camera rotations about an axis that varies with time. These robot trajectories corresponding to linear contour interpolation are shown in Figure 15 to compare with those for those computed from the affine geodesic interpolation which are shown in Figure 16.

6. Degeneracies and Condition Numbers

There are two significant types of degeneracy suffered by this approach affine visual servoing. These are:

Intrinsic degeneracy in which the inherent shape of the contour creates an ambiguity as in the case of a circular contour, for which rotation about the contour centre is not observable. When this occurs, the rank of the matrix used to identify the affine transformation (matrix M_{jk} in Algorithm 1) drops from 6 to 5. In general this problem occurs when the contour is a conic section. In the situation where most of the contour is conic (for example a straight line, in the case where the contour is a long thin box), the snake can only use the control points on the remainder of the contour to resolve the ambiguous motion and hence this situation is also unstable. Examples of contours and their intrinsic condition numbers are given in Section 7.1.

Extrinsic degeneracy which occurs when the contour is fronto-parallel to the camera in the target position. In this event, the computed Jacobian suffers a two-fold degeneracy and its rank drops from 6 to 4. This happens because the two shear modes of affine deformation become inaccessible simultaneously. Consequently there are only four remaining degrees of freedom in the observation to control a six degree of freedom robot, causing the control strategy to become ill-posed. The relationship between the angle of inclination of the contour and the condition number of the Jacobian is shown experimentally in Section 7.2.

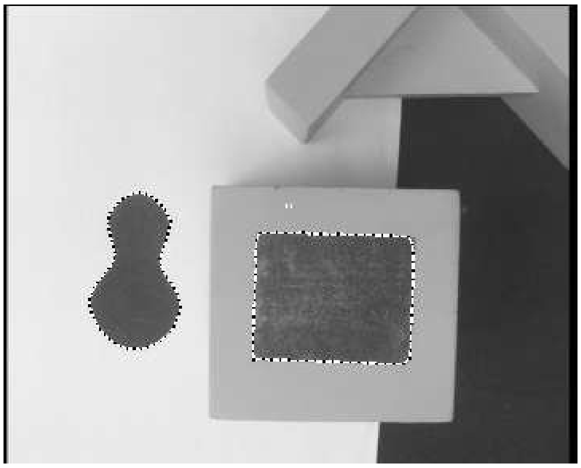


Fig. 17. Use of two degenerate contours for servoing

Both modes of degeneracy can be detected by monitoring the condition numbers of matrices in the system. When a matrix is rank deficient, its condition number rises to infinity. Numerically, this is computed by calculating the ratio of the largest and smallest diagonal values in the singular value decomposition. If this ratio rises above some threshold, the matrix is considered to be rank deficient. The dimensionality of the deficiency is the number of diagonal values that are less than the largest value divided by the threshold. The thresholds determined experimentally from stability trials were 2000 for the affine transformation matrix (indicating intrinsic degeneracy) and 100 for the Jacobian (indicating extrinsic degeneracy).

6.1. Multiple Contours

These problems can be overcome by the use of multiple contours. Servoing can be performed from the use of two separate contours, each of which individually suffers extrinsic degeneracy (where both are parallel to the image plane and thus the Jacobian for each contour has rank 4), provided that they lie at different depths (are not co-planar). An example is shown in Figure 17. By concatenating the vectors for the two individual contours, a 12-vector of observed deformations is formed. This results in a 6×12 Jacobian of robot-to-affine motions. Again, by using SVD, the pseudo-inverse of this matrix, a 12×6 affine-to-robot Jacobian can be computed. In the case shown in Figure 17, this Jacobian has full rank and thus the servoing problem is well conditioned.

Additional contours are also useful, even when the original contour is well conditioned. Typically the condition number of the Jacobian dropped (from about 30 to 20) and qualitative evaluation showed that the system was stable over a greater range of perturbations when used in this configuration since it was necessary for both contours to approach singularities before the system became unstable.

7. Implementation and Results

This approach has been implemented in the lab using a SCORBOT ER VII 5 DoF robot arm with a monochrome video camera (which can be seen in Figure 22) connected to an SGI O2 workstation. The workstation is able to track up to 256 snake nodes at video frame rate and control the robot with a cycle time of 0.5 – 1.5 seconds. The principal cause of this long cycle time in the robot control thread is the slow communications with the robot controller.

7.1. *Choice of contour*

The system has been tested with a number of contours, a selection of which is shown in Figure 18. The condition numbers of the transformation matrices under affine and projective locking was computed experimentally for each of these contours in order to examine intrinsic degeneracy. The results of this are shown in Table 1. Contours 1–9 represent an evolving series. Contour 1 has the least intrinsic degeneracy, being the least elliptical, while contour 8 is sufficiently close to elliptical that the tracking system has substantial difficulties in tracking it stably. Contours 10 and 11 are essentially triangular. This situation is projectively degenerate, since all motion of a triangle in a plane can be represented by general affine transformations. Consequently, the condition numbers for projective tracking are very large, while the affine condition number is certainly small enough for stable tracking. Contour 12 is highly non-degenerate and has the smallest condition numbers.

Table 1. Contour condition numbers

contour no.	Affine	Projective
1	127	145
2	149	182
3	190	212
4	232	255
5	340	378
6	535	615
7	1110	1280
8	1880	2250
9	1160	1550
10	333	20500
11	530	8500
12	38	80

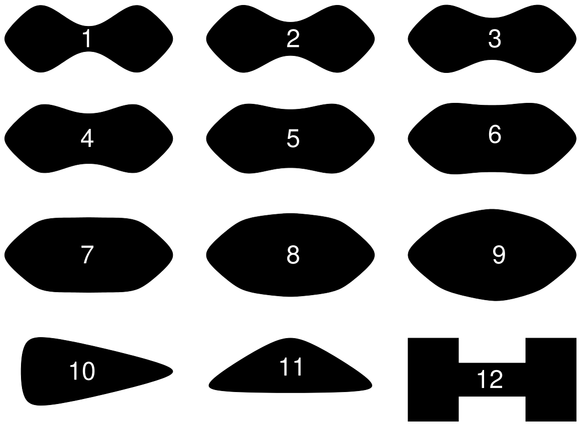


Fig. 18. Contours tested

7.2. *Angle of inclination*

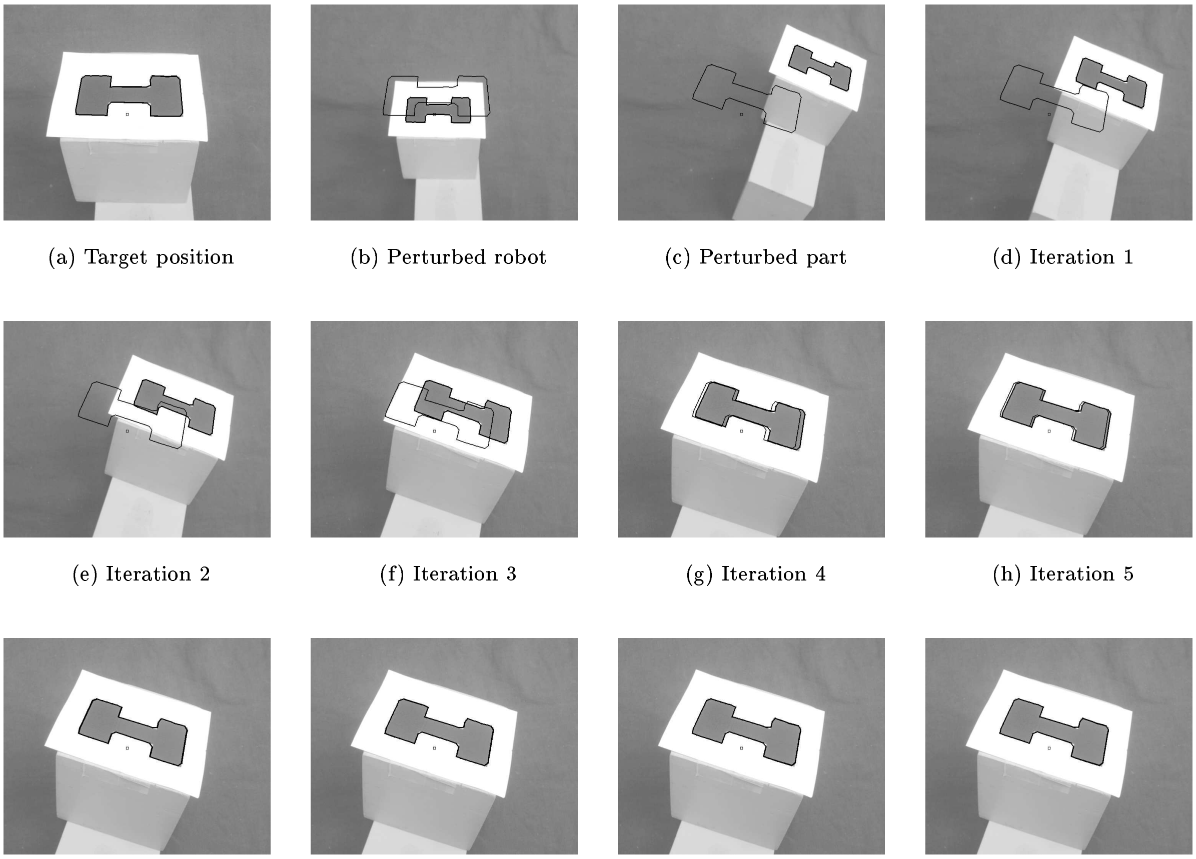
Contour 12 was selected and the angle, θ of the inclination of the plane of the contour (as shown in Figure 12) was varied. The Jacobian was computed from trial motions in each configuration and the condition number computed in order to examine extrinsic degeneracy. The results of this are shown in Table 2.

The configuration with $\theta = 0$ is theoretically degenerate and should have an infinite condition number. In practise this does not occur since there is measurement noise in computing the affine deformation of the contour and the robot trial motions are not infinitesimal, and thus induce a small amount of affine shear.

Although it was possible to perform visual servoing with $\theta = 10^o$, in practise, reliable performance was only obtained when the condition number was less than 50 (i.e. $\theta > 20^o$).

Table 2. Jacobian condition numbers

θ (degrees)	Condition no
0	336.5
5	103.9
10	77.7
20	41.1
30	30.4
40	23.7
50	20.4
60	18.8
70	18.0
80	broke



(a) Target position (b) Perturbed robot (c) Perturbed part (d) Iteration 1

(e) Iteration 2 (f) Iteration 3 (g) Iteration 4 (h) Iteration 5

(i) Iteration 6 (j) Iteration 7 (k) Iteration 8 (l) Iteration 9

Fig. 19. Sequence for returning to target position

7.3. Tool placement experiments

A set of experiments were conducted in which the aim was to accurately place a simulated tool onto a workpiece. The length of the tool (and thus the viewing distance when at the target) was 200mm. The reference inclination of the plane of the contour for these experiments was chosen to be $\theta = 40^o$. Because the robot is over-constrained, the sixth degree of freedom was synthesised as rotation about the optical axis of the camera. In order to maintain consistent placement of the tool, this was computed as rotation about the location of the tool tip in the image. This location must be identified by the operator.

7.3.1. Range of perturbation An experiment was conducted to test the range of possible perturbations of the workpiece from target position. The experiment was conducted by moving the robot back away from the target position to a starting position. The part was then perturbed and the robot asked to servo back to the target location. A typical run is shown in Figure 19 which shows the computer view of the sequence with reference and tracking snakes superimposed on the video images. 19a shows the contour as seen from the target position, in which the system initialises the reference snake. 19b shows the robot perturbed to the starting position. In 19c, the part is perturbed by a translation and rotation. The component of rotation about the optical axis of the camera corresponds to the missing degree of freedom of the robot. Consequently, this rotation is handled by also rotating the target reference contour about a pre-specified point in the image plane. 19d–l show

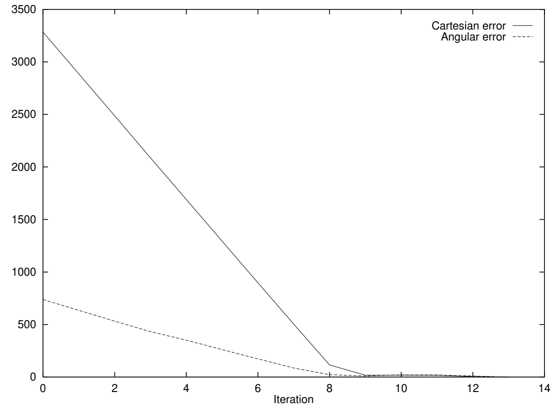


Fig. 20. Typical convergence. The angular (rotation) error is in units of 0.01^o while the cartesian (translation) error is in units of 0.01 mm

the sequence of iterations as the robot returns to the target position. Typical convergence performance for this experiment is shown in Figure 20.

This experiment showed that the translational perturbation is limited by the requirement to keep the contour within the image boundary, resulting in a maximum of approximately 200mm translational perturbation in x, y and z. The maximum rotational perturbation of the angle of inclination of the plane was limited to 20^o about the reference angle of $\theta = 40^o$ with a limit of 40^o about the horizontal axis with a reference orientation of 0^o (see Figure 21).

Servoing from reference angles other than $\theta = 40^o$ has also been tested, with the system successfully responding to perturbations of the workpiece with $10^o \leq \theta \leq 70^o$. At the extreme ends of this range, the range of permissible perturbations is limited since the perturbed part must have $0^o < \theta < 80^o$. If $\theta < 0^o$ then the computed Jacobian is incorrect (since the part has passed through the singularity) and if $\theta > 80^0$ then the contour becomes intrinsically degenerate and is subject to frequent catastrophic tracking failure.

7.3.2. Precision The accuracy of placement of the robot under visual servoing was also tested. The workpiece was left fixed, and the robot asked to servo back to the target position from a series of random starting positions. The accuracy of positioning (1 standard deviation) of the camera in this experiment was ± 0.65mm in x and y and 0.3mm in z with 0.15^o in both pitch and roll. The maximum error measured at the tool tip over a series of runs was 1mm, with almost all errors being less than 0.5mm. This is of higher accuracy than the camera positioning due to correlations in the position and rotation errors.

The convergence rate was also computed, with the mean time to convergence being 2.5 cycles after the robot has reached the non-linear control zone shown in Figure 11.

7.3.3. Closed loop tracking A final experiment was concerned with closed loop tracking and aimed to test the range of acceptable perturbations that can be tracked gradually under closed loop, as shown in Figure 22. In this mode substantial perturbations extending to approximately double those shown in Figure 21 were successfully tracked.

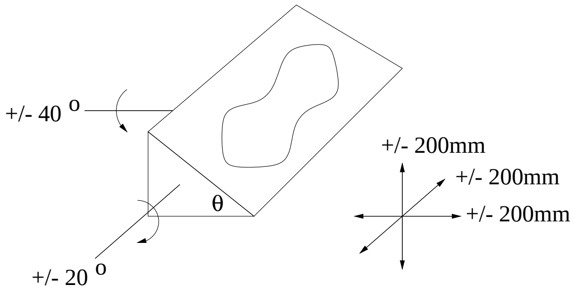


Fig. 21. Range of stable perturbations

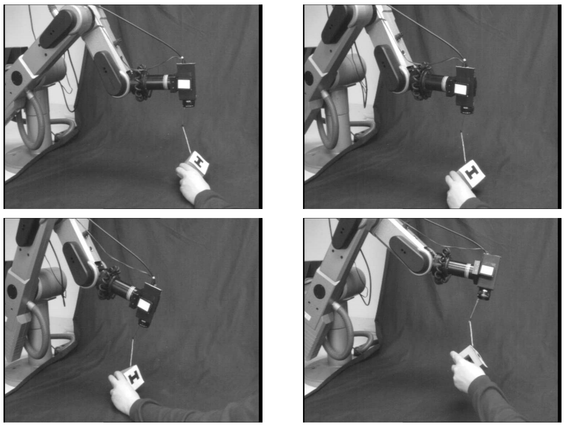


Fig. 22. Closed loop tracking

8. Conclusions

This paper has shown how the Lie algebra of affine transformations can be used for visual servoing. This is achieved by means of a novel technique which allows group transformations to be accurately integrated in the Lie algebra representation. This is combined with a Jacobian based non-linear control system to obtain high accuracy visual servoing. This system has been implemented and experiments show an accuracy of < 1mm error in position and 0.15^o error in rotation. The use of multiple contours to overcome degeneracies has also been presented.

9. Acknowledgements

This work was supported by an EC (ESPRIT) grant no. LTR26247 (VIGOR) and by an EPSRC grant no K84202. The robot was donated by the Olivetti Research Laboratory, Cambridge.

Appendix A: Definitions

An **n-dimensional manifold** is a topological space such that about every point, there is an open neighbourhood which is topologically isomorphic to $\mathbb{R}^n$.

A **Lie group** is a group and a topology which form a manifold.

The **Lie algebra** of a Lie group is the tangent space of the group manifold at the identity. The **generators** of the group are vectors in this space and form a basis for it. The **Lie bracket** is an anti-symmetric bi-linear form which is obtained from the commutators of the group generators.

An **ideal** of a Lie group is a subgroup which is mapped into itself by conjugation with any member of the group.

An **abelian group** is one in which all pairs of elements commute.

A **semi-simple** Lie group is one with no abelian ideals.

Appendix B: Derivation of Equation (18)

The formula (18) is derived by repeatedly refining the estimate of A to higher orders by expanding both sides as a Taylor series. Expanding

$$e^A = e^B e^C$$

to first order gives:

$$\begin{aligned} e^A &= I + A \\ e^B e^C &= (I+B)(I+C) = I + B + C \end{aligned}$$

So, to first order:

$$A = B + C \tag{1}$$

Substituting (1) back in to the Taylor series for e^A and expanding to second order gives:

$$\begin{aligned} e^A &= I + (B+C) + \tfrac{1}{2}(B^2 + BC + CB + C^2) \\ e^B e^C &= I + B + C + \tfrac{1}{2}(B^2 + 2BC + C^2) \\ &= e^A + \tfrac{1}{2}(BC - CB) \\ &= e^A + \tfrac{1}{2}[B, C] \end{aligned}$$

So, to second order:

$$A = B + C + \tfrac{1}{2}[B, C] \tag{2}$$

Substituting (2) into e^A and expanding to third order gives:

$$\begin{aligned} e^A = I &+ B + C + \tfrac{1}{2}(B^2 + 2BC + C^2) \\ &+ \tfrac{1}{4}((B+C)[B,C] + [B,C](B+C)) \\ &+ \tfrac{1}{6}(B^3 + B^2C + BCB + BC^2 \\ &\quad + CB^2 + CBC + C^2B + C^3) \end{aligned}$$

$$\begin{aligned} e^B e^C &= I + B + C + \tfrac{1}{2}(B^2 + 2BC + C^2) \\ &\quad + \tfrac{1}{6}(B^3 + 3B^2C + 3BC^2 + C^3) \\ &= e^A - \tfrac{1}{4}((B+C)[B,C] + [B,C](B+C)) \\ &\quad + \tfrac{1}{6}(2B^2C + 2BC^2 \\ &\qquad - BCB - CB^2 - CBC - C^2B) \\ &= e^A - \tfrac{1}{4}((B+C)[B,C] + [B,C](B+C)) \\ &\quad + \tfrac{1}{6}((2B+C)[B,C] + [B,C](B+2C)) \\ &= e^A + \tfrac{1}{12}((B-C)[B,C] + [B,C](C-B)) \\ &= e^A + \tfrac{1}{12}([(B-C),[B,C]]) \end{aligned}$$

So, to third order:

$$A = B + C + \tfrac{1}{2}[B, C] + \tfrac{1}{12}([(B-C),[B,C]] \tag{3}$$

In practise this third order approximation is sufficient for experimental purposes.

References

Bard, C., Laugier, C., Milési-Bellier, C., Troccaz, J., Triggs, B. and Vercelli, G.: 1995, Achieving dextrous grasping by integrating planning and vision-based sensing, *International Journal of Robotics Research* **14**(5), 445–464.

Basri, R., Rivlin, E. and Shimshoni, I.: 1998, Visual homing: Surfing on the epipoles, *Proceedings of International Conference on Computer Vision (ICCV '98)*, pp. 863–869.

Cipolla, R. and Blake, A.: 1997, Image divergence and deformation from closed curves, *International Journal of Robotics Research* **16**(1), 77–96.

Colombo, C. and Allotta, B.: 1999, Image-based robot task planning and control using a compact visual representation, *IEEE T-Systems, Man and Cybernetics (B)* **29**(1), 92–100.

Corke, P. and Goode, M.: 1996, Dynamic effects in visual closed-loop systems, *IEEE T-Robotics and Automation* **12**(5), 671–683.

Couvignon, P., Papanikolopoulos, N., Sullivan, M. and Khosla, P.: 1996, The use of active deformable models in model-based robotic visual servoing, *Journal of Intelligent and Robotic Systems* **17**(2), 195–221.

Cross, G. and Cipolla, R.: 1996, Affine visual servoing, *Proceedings of British Machine Vision Conference (BMVC '96)*, Vol. 2, Edinburgh, pp. 425–434.

Espiau, B., Chaumette, F. and Rives, P.: 1992, A new approach to visual servoing in robotics, *IEEE T-Robotics and Automation* **8**(3), 313–326.

Hager, G. and Hutchinson, S.: 1996, Introduction to special section on vision-based control of robotic manipulators, *IEEE T-Robotics and Automation* **12**(5), 649–650.

Horaud, R. and Dornika, F.: 1995, Hand-eye calibration, *International Journal of Robotics Research* **14**(3), 195–210.

Hutchinson, S., Hager, G. and Corke, P.: 1996, A tutorial on visual servo control, *IEEE T-Robotics and Automation* **12**(5), 651–670.

Isard, M. and Blake, A.: 1996, Visual tracking by stoachastic propagation of conditional density, *Proceedings of the 4th European Conference on Computer Vision*, pp. 343–356.

Kass, M., Witkin, A. and Terzopoulous, D.: 1988, Snakes: Active contour models, *International Journal of Computer Vision* **1**(4), 321–331.

Kinoshita, K.: 1998, Robotic control with partial visual information, *Proceedings of International Conference on Computer Vision (ICCV '98)*, pp. 883–888.

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

Malis, E., Chaumette, F. and Boudet, S.: 1999, 2D$\frac{1}{2}$ visual servoing, *IEEE T-Robotics and Automation.* (to appear).

Murray, R. and Sastry, S.: 1994, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, London.

Nayar, S., Nene, S. and Murase, H.: 1996, Subspace methods for robot vision, *IEEE T-Robotics and Automation* **12**(5), 750–758.

Park, F., Barrow, J. and Ploen, S.: 1995, A Lie group formulation of robot dynamics, *International Journal of Robotics Research* **14**(6), 609–618.

Sanderson, A., Weiss, L. and Neumann, C.: 1987, Dynamic sensor based control of robots with visual feedback, *IEEE Journal of Robotics and Automation* **3**, 404–417.

Santos-Victor, J., Sandini, G., Curotto, F. and Garibaldi, S.: 1995, Divergent stereo in autonomous navigation: From bees to robots, *International Journal of Computer Vision* **14**, 159–177.

Sattinger, D. and Weaver, O.: 1986, *Lie groups and algebras with applications to physics, geometry, and mechanics*, number 61 in *Applied Mathematical Sciences*, Springer-Verlag.

Varadarajan, V.: 1974, *Lie Groups, Lie Algebras and Their Representations*, number 102 in *Graduate Texts in Mathematics*, Springer-Verlag.

Wilson, W., Williams Hulls, C. and Bell, G.: 1996, Relative end-effector control using cartesian position based visual servoing, *IEEE T-Robotics and Automation* **12**(5), 684–696.

Yoshimi, B. and Allen, P.: 1995, Alignment using an uncalibrated camera system, *IEEE T-Robotics and Automation* **11**(4), 516–521.