Towards 3D Object Model Acquisition and Recognition using 3D Affine Invariants *

Sven Vinther and Roberto Cipolla
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ, England

July 1993

Abstract

We evaluate the power of 3D affine invariants in an object recognition scheme. These invariants are actively calculated by the real-time tracking of 2D image features (*corners*) over an image sequence. This is done optimally by using a Kalman filter. Object information is located in a hash table where it is stored and retrieved using the invariants as stable indices. Recognition takes place when significant evidence for a particular shape has been found from the table. Preliminary results with real data are presented, and some of the noise problems arising due to the weak perspective approximation and corner localisation errors are discussed.

1 Introduction

Three dimensional object recognition is a rapidly expanding field within computer vision. Central to object recognition systems are questions of model acquisition and representation, feature extraction and matching. All of these issues are closely related, as the type of feature extraction performed will depend directly on the model adopted, and will also affect the matching technique selected. Extensive surveys describing methods of object representation and matching have been made by Chin and Dyer [1], Besl and Jain [2].

A number of working recognition schemes have been proposed for simple 3D objects. Lowe [3] presents a 3D object recognition system which uses a single image, *perceptual groupings* and viewpoint consistency constraints to detect 3D objects from 2D data. Thompson and Mundy [4] use *vertex-pairs* to derive the affine transformation between a 3D polyhedral object model and its projection into the image viewplane.

A more common approach is the use of 3D data extracted from range data or multiple views. The approach by Grimson and Perez [5] operates by examining all hypotheses between segmented range data and model surfaces, and efficiently discarding inconsistent ones by using local constraints. This is achieved by a depth first search of their *interpretation tree*, which at each level attempts to match an image feature to all model features. The pruning of subtrees after a failed match and the use of a search cutoff keeps search time to acceptable levels. *MARVIN* [6] is a system developed at Sheffield University, which recovers 3D scene descriptions using calibrated stereo views and polygonal approximation. These descriptions are then matched against 3D database models using a method similar to the interpretation tree. Structure from motion techniques allow an alternative to stereo imagery [7]. These involve rebuilding the 3D structure of a shape based on a stream of data from a sequence of images, in which either camera or object is moving.

An alternative approach is *geometric invariants* [8]. Several successful 2D object recognition

*Paper to appear in the 4th British Machine Vision Conference 1993 systems have been implemented using invariants [9]. In this paper we evaluate the power of 3D affine invariants in an object recognition scheme.

2 Theoretical Framework

The image of a 3D object will appear different depending on viewpoint because its 2D projection suffers from distortions, self-occlusion and a loss of depth. Solid object descriptors such as edge length, surface area and angles between lines which are invariant in 3D space, are no longer so in the 2D image. However a closer examination of the effects of projective transformations reveals that it is possible to find properties of combinations of features which remain *invariant* in different views. For example the *cross-ratio* of four points on a line remain invariant to general projection, and simpler invariants can be found for simpler transformations. A more detailed overview of this field of research and its applications can be found in [8].

2.1 Weak Perspective

The full perspective transformation of world points to image points (u, v) can be represented by (1). It combines the effects of aligning a general world frame to a camera-centered frame, and then projecting the points using the *pinhole camera* model. (Further details can be found in Appendix A.)

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1}$$

A simplification of this transformation is possible under *weak* perspective [10] (orthographic projection plus a scale factor). Weak perspective is a good approximation of full perspective when the distance from object to camera, Z, is much greater than the extent of the object ΔZ. Under these conditions scaling factor $s = t_{31}X_w + t_{32}Y_w + t_{33}Z_w + t_{34}$, becomes approximately constant $s = t_{34}$ in effect linearising the transformation.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ 0 & 0 & 0 & t_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2}$$

There are just eight degrees of freedom for this weak perspective transformation so only four points are required to evaluate the transformation, and no camera calibration is required.

2.2 Affine Invariants from Multiple Views

The linearity of weak perspective allows us to recover invariant 3D object structure from two or more images [8, 11, 12]. Given five 3D points $\mathbf{X_i}, i \in \{0, .., 4\}$ it is possible to calculate 3 invariants [8]. This can be shown by considering a basis $\mathbf{E_i} = \mathbf{X_i} - \mathbf{X_0}, i \in \{1, 2, 3\}$ in 3D space. Any 5th point in this basis can be expressed as a linear combination of $\mathbf{E_i}$'s (3) (see Figure 1).

$$\mathbf{X_4} = \mathbf{X_0} + \alpha\mathbf{E_1} + \beta\mathbf{E_2} + \gamma\mathbf{E_3} \tag{3}$$

The (α, β, γ) are the coordinates of the point $\mathbf{X_4}$ in this basis, which are related to the 3-dimensional shape of the object under 3D affine transformations, and are invariant in 3D space. Due to the linear nature of weak perspective (α, β, γ) will remain viewpoint invariant under this

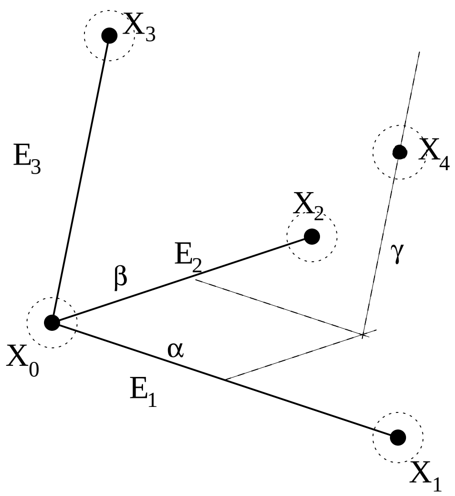


Figure 1: Affine Basis. Each point is surrounded by an error ellipse to show uncertainty in location due to noise and poor localisation

transformation. We will refer them as 3D *affine* invariants of our shape. We rewrite (3) in matrix form, and combine it with (2).

$$\begin{bmatrix} su_4 \\ sv_4 \\ s \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ 0 & 0 & 0 & t_{34} \end{bmatrix} \begin{bmatrix} E_{1x} & E_{2x} & E_{3x} & X_{0w} \\ E_{1y} & E_{2y} & E_{3y} & Y_{0w} \\ E_{1z} & E_{2z} & E_{3z} & Z_{0w} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ 1 \end{bmatrix} \tag{4}$$

Which we can rewrite more simply as:

$$\begin{bmatrix} u_4 - u_0 \\ v_4 - v_0 \end{bmatrix} = \begin{bmatrix} e_{1u} & e_{2u} & e_{3u} \\ e_{1v} & e_{2v} & e_{3v} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \tag{5}$$

where $\mathbf{u_0}, \mathbf{u_4}$, are the projected basis origin and point respectively,$\mathbf{e_1} \ldots \mathbf{e_3}$ are the projections of our basis vectors, all of which are obtainable from image measurements. This provides us with two equations and three unknowns, the problem has become underdetermined, so a single 2D image does not allow the recovery of the invariants. A second view with known point correspondences to the first view will however give an overdetermined set of equations solvable by standard least square minimisation methods.

Solving Overdetermined Equations

Our previous equation (5) can be rewritten in slightly different matrix form to include the effects of noisy measurements and the extra views:

$$\begin{bmatrix} u_4 - u_0 \\ v_4 - v_0 \\ u'_4 - u'_0 \\ v'_4 - v'_0 \end{bmatrix} = \begin{bmatrix} e_{1u} & e_{2u} & e_{3u} \\ e_{1v} & e_{2v} & e_{3v} \\ e'_{1u} & e'_{2u} & e'_{3u} \\ e'_{1v} & e'_{2v} & e'_{3v} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} \vdots \\ \epsilon_i \\ \vdots \end{bmatrix} \tag{6}$$

$$\underline{u} = \mathbf{A}\underline{\alpha} + \underline{\eta} \tag{7}$$

$\mathbf{e}'$ is the projection of the basis in our second view and $\underline{\eta}$ is the noise in the measured points. Integrating more views into this equation is done trivially by each time adding two more rows to the matrix and can be used to give better conditioned equations. The least squares solution $\hat{\underline{\alpha}}$ can then be found using standard methods.

$$\hat{\underline{\alpha}} = (A^T A)^{-1} A^T \underline{u} \tag{8}$$

The solution minimises the residual sum of squares $(\underline{u} - \mathbf{A}\underline{\alpha})^T(\underline{u} - \mathbf{A}\underline{\alpha}))$. Singular value decomposition can also be used to find this solution, with the added advantage that it can check whether or not the equations are well-conditioned. The *variance - covariance* matrix associated with $\hat{\underline{\alpha}}$ is $V(\hat{\underline{\alpha}}) = \sigma^2(B^T B)^{-1}$. We are assuming mean noise $E(\underline{\eta}) = 0$, and noise variance $V(\underline{\eta}) = \sigma^2 I$ (*ie* disturbance terms are uncorrelated with one another.)

3 Object Model Acquisition and Recognition

The object recognition scheme is based on representing an object using these 3D affine invariants. This imposes a slight limitation on the method in that we only consider image data viewed under weak perspective.

Feature extraction as with most recognition schemes forms the first step, in this case we wish to reliably detect corners in the image which relate to the projections of vertices on an object. Multiple views of the object with known point correspondences are then used to calculate the object invariants. In order to avoid the point correspondence and calibration problems of multiple views the vertices are tracked through a sequence of images. A Kalman Filter is then used to optimally estimate the invariants from the stream of data.

The next task is to find invariants from the image data which match those from a model, by having identical values and being associated with the same basis. Instead of calculating the invariants associated with a model at recognition time, they are precomputed and data linked to them is stored off-line, speeding up the final recognition process at the cost of extra storage space.

The geometric hashing method proposed by Wolfson et al [13, 9], with an expanded index to reflect the 3D (as opposed to 2D) object database, is used to implement this data storage. The invariants form a fixed hash table index which points at a *bin* in the table, where we store information about the features and object used to compute those invariants. When attempting to match our image data with a model we extract the contents of the indexed bin, thereby discovering all possible object and pose correspondences. By using several invariants, and hence the contents of several bins, we are able to prune the possible object matches down by finding multiple evidence for particular bases. These can then be examined more closely by trying to match projected model edges with those present in the image.

3.1 Feature detection

In order to simplify object recognition it is often useful to preprocess images, extracting features within the object, which help to give a simplified but distinctive description of it. The features may include texture, straight lines, ellipses, points or higher level combinations of these such as parallel lines, or surfaces. Corners are useful for describing artificial objects and are of special interest due to their accurate localisation. Sensitivity to noise, speed and selectivity are all issues which need to be considered when attempting to implement a corner detector. A number of methods for detecting corners within shapes include template matching, second-order derivative schemes [14, 15], autocorrelation [16], median based methods [17] and the interesting point operator [18].

The corner detector implemented in the following experiments is particularly suited to detecting corners which are projections of vertices. It locates corners by first finding regions where several lines are present and then pinpointing where those lines intersect. Since vertices are the features used in our object description it is helpful to the overall recognition speed, if the corner detector doesn't pinpoint too many other types of corner. The procedure is tolerant to noise and reasonably fast (Figure 2). (For more details see Appendix B.)

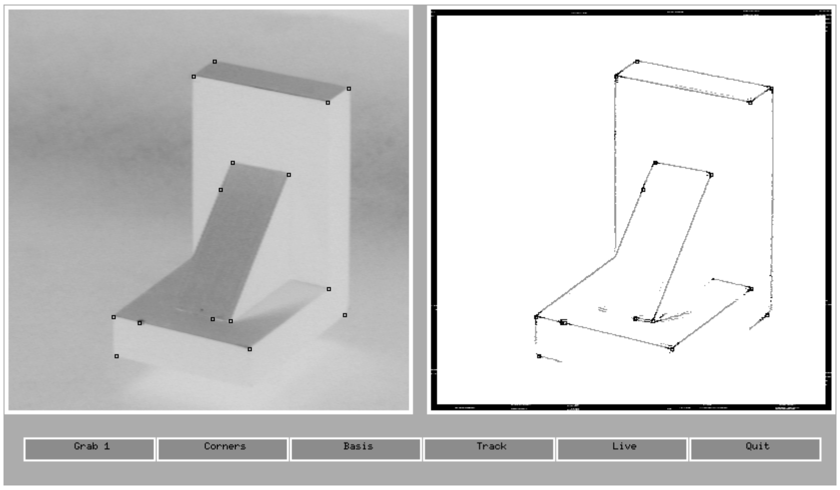


Figure 2: An image of a block with detected corner points superimposed. The image is in reverse video to highlight the corners. There are a couple of noisy points due to dents on the block and other effects such as shadows. Also other points are not detected because our edge threshold was set too high. There is a compromise here in trying to keep background noise to a minimum and detecting all points.

3.2 Feature Tracking

A way of avoiding the correspondence problem in multiple images is to exploit temporal continuity in a monocular sequence of frames. We track all the corner points initially detected in the image as the object moves in the scene. Since the corners will only shift slightly between consecutive frames, the approach taken is to perform local corner detection centered on the old corner point locations, finding new corner locations in the neighbourhood. The algorithm is shown in Figure 3. A further improvement is to perform cross-correlation between a template of the old corner and new corner in cases where we need to disambiguate between several possible maxima in a local region. However it was found that the extra robustness gained by doing this was outweighed by the extra computational cost. Figure 5 shows an example of a test object where corners have been detected and then tracked.

3.3 Optimal Estimation of α, β, γ and their Uncertainty

The most important stage in the recognition scheme is the accurate estimation of the 3D affine invariants. However there is some uncertainty in any estimate of α, β and γ due to errors introduced by the weak perspective approximation and errors in corner localisation. By using the redundancy in a stream of data from multiple views, and ensuring adequate object rotation between the first and final images it is possible to improve the robustness of the resulting estimate over one calculated from a single stereo image pair. A Kalman filter is used to integrate data from multiple views, and optimally estimate α, β, γ (Figure 4 and Appendix C).

Our Kalman filter is in fact very simple since the output α, β, γ is *invariant*. The Kalman filter will also return the uncertainty of the α, β, γ estimates in terms of estimate variances. When this uncertainty falls to within acceptable limits we can stop tracking the corners and proceed with the actual object recognition.

Results

Figure 6 shows the results of calculating four sets of invariants using a Kalman Filter (using the track data seen in figure 5). All the graphs have an initial transient stage due to the initialisation of

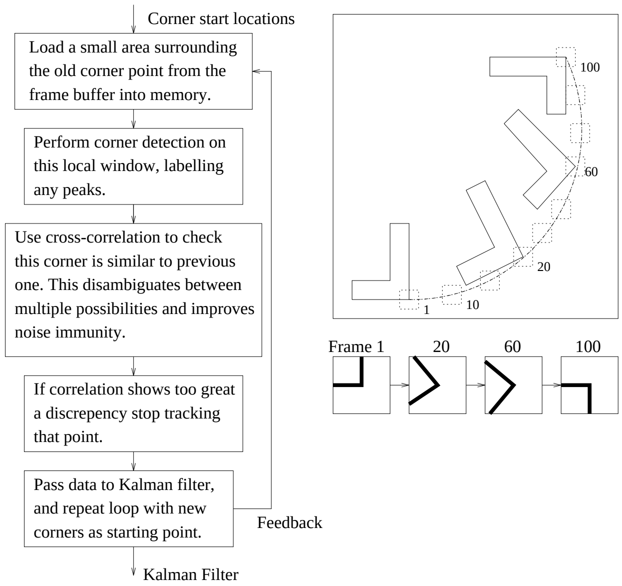


Figure 3: Tracking Routine

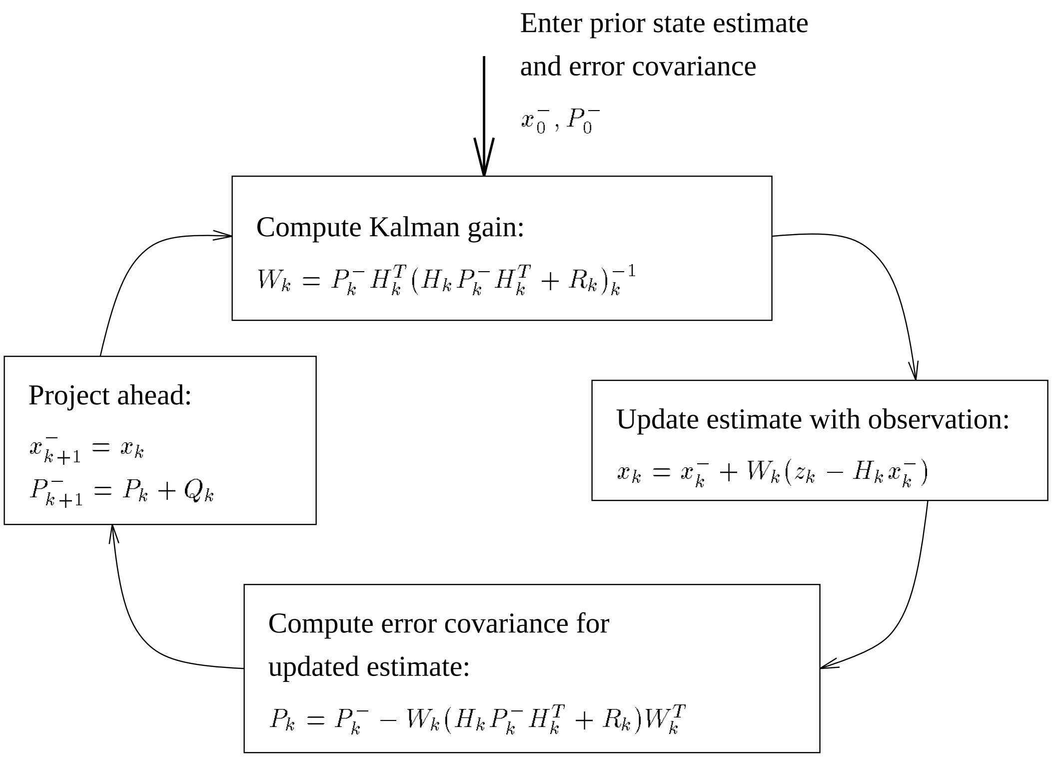


Figure 4: Kalman filter loop

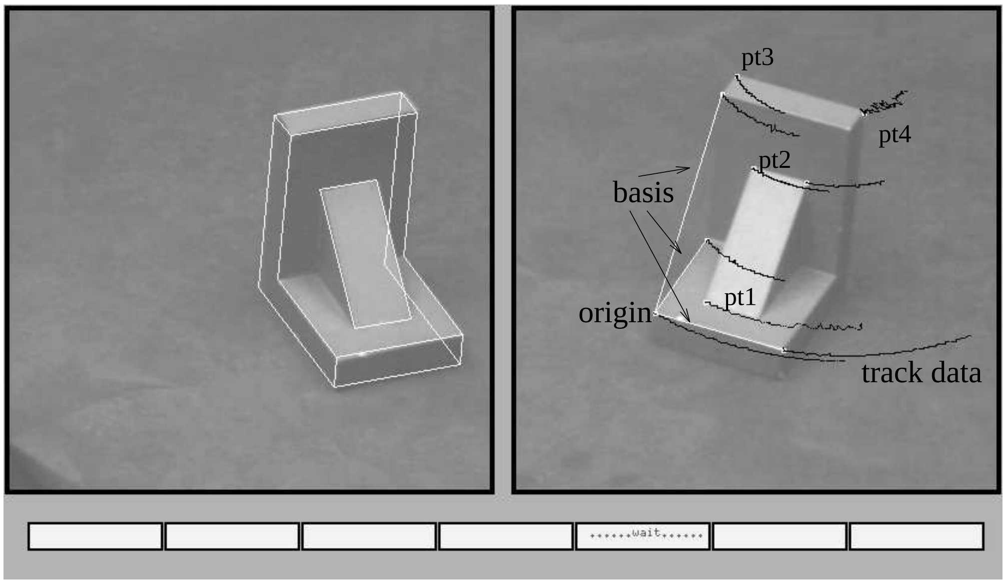


Figure 5: Several points were tracked on our object as it rotated. This is shown as black arcs on our image.

Invariant	Real data	Perfect Result	Invariant	Real data	Perf. Res.
pt1 α	0.28	0.28	pt2 α	0.29	0.28
β	0.29	0.29	β	0.41	0.43
γ	0.01	0.00	γ	0.57	0.57
pt3 α	0.01	0.00	pt4 α	0.98	1.00
β	0.23	0.27	β	0.30	0.27
γ	1.01	1.00	γ	0.98	1.00

Table 1: Example of Invariants output by the Kalman filter

the filter with zeros, however this soon settles down as more track data is incorporated, improving the estimate of the invariants. Points 1 and 4 have the noisier tracks, and this is reflected in the graphs, where the invariant estimate has a greater uncertainty taking longer to settle. The final output is compared with what we expected from perfect data in table 1.

The 3D basis geometry will also affect the sensitivity of α, β, γ to noise. If the basis is formed from points which are nearly coplanar the resulting equations (6) which are to be solved will become ill-conditioned. It may be necessary to try several different bases in order to find one which gives well-conditioned equations. This could also be implemented by running several Kalman filters on the track data in parallel, each using a different set of points as the basis and selecting the invariants from the one with least uncertainty.

3.4 Geometric Hashing for data storage

The geometric hashing approach to model-based object recognition was proposed by Lamdan, Schwartz and Wolfson [13, 9]. It relies on computing object related invariants from our features, which are then used as a simple and stable index $H(\alpha, \beta, \gamma)$ into a hash table. Each indexed location in the hash table will store information about the object used to create the invariant, the points used to form the basis and the point referenced in that basis (9).

$$H(\alpha, \beta, \gamma) = (Object,\ Basis,\ Point) \tag{9}$$

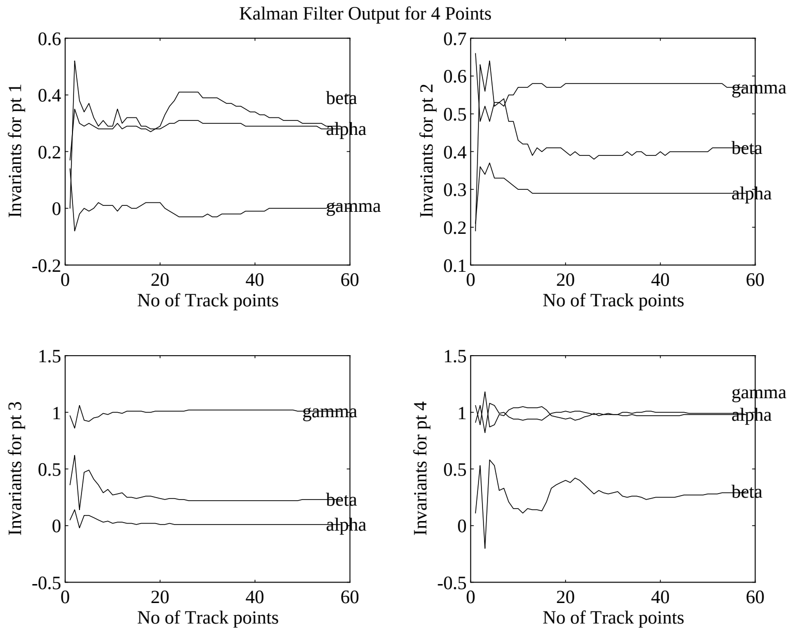


Figure 6: Invariants output by the Kalman filter

At the recognition phase we then index into the hash table, and retrieve all possible objects and their bases, using a voting scheme to decide on their likelihood and a final matching process to select the best fitting object.

Initially a large amount of pre-computation is required on the model library to create the hash table, but since this can be done off-line it will not affect recognition speed. Invariants are calculated for all permutations of basis and points and data stored in the hash table. Since 3D data is available in our model we can calculate our invariants directly using (10) [cf (3)].

$$\begin{pmatrix} E_{1x} & E_{2x} & E_{3x} \\ E_{1y} & E_{2y} & E_{3y} \\ E_{1z} & E_{2z} & E_{3z} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} X_4 - X_0 \\ Y_4 - Y_0 \\ Z_4 - Z_0 \end{pmatrix} \tag{10}$$

An obvious problem when creating the hash table is the large quantity of data present. If we

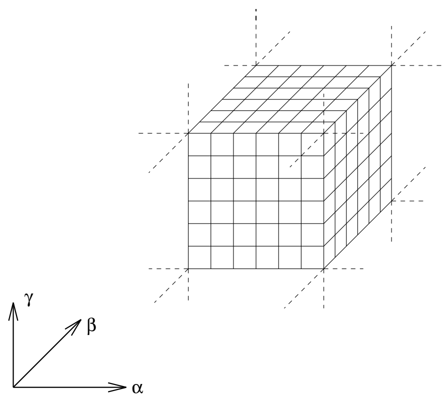


Figure 7: Hash Table: In each bin indexed by $H(\alpha, \beta, \gamma)$ we store related object information.(Which object, What Basis and Which Point).

have an object with N_v vertices, there are $N_b = N_v!/(N_v-4)!$ possible bases and $N_{invs} = N_b(N_v-4)$ associated invariant sets. This exponentially growing problem needs to be addressed, by pruning away invariant sets. A number of choices are available:

- Four coplanar points cannot be used to form an adequate basis.
- Ill-conditioned bases will be susceptible to noise and should not be included.
- Use characteristic views of an object [19] to decide which points will be available from a common view, hence removing points never present simultaneously.
- Eliminate any points the corner detector is poor at localising in the real images if this is known.
- An important decrease in data can also be achieved by noting that we do not need to store object information for all invariants calculated from permutations of the same basis. The effect is simply to permutate our invariants, so by arranging (α, β, γ) in order of size we just need to store information at one location in the hash table. This gives a six-fold reduction in data.

Taking this a step further we note that for a set of four points there are 4! ways of forming a basis. However any invariants calculated in one of these bases is related to all the others (Appendix D), so it is possible to find a global invariant for all of these. The drawback with this is we increase the amount of computation that needs to be done at recognition time.

One issue not addressed so far is the effect of poor corner localisation on our invariants (α, β, γ). A study performed by Grimson et al. [20] on 2D affine data shows that if we assume our corner points lie within some error disc, radius ϵ (see figure1), the region of uncertainty associated with our affine coordinates is an ellipse whose area, centre and principle axes depend on ϵ and the invariants. This would become an ellipsoid in our 3D case. The area of this ellipse increases $\propto (1 + |\alpha| + |\beta|)^2$, so larger affine coordinates will have larger errors. This matches well with the worst case study done in [9] which state results from numerical analysis.

$$(A + \delta A).(\underline{\alpha} + \delta \underline{\alpha}) = (\underline{x} + \delta \underline{x}) \tag{11}$$

Given we can estimate δA and $\delta \underline{x}$ then $\delta \underline{\alpha} \leq (1 + x) c_k \epsilon$ (where $x = |\alpha| + |\beta| + |\gamma|$ is the sum of the absolute values of our invariants, and $c_k = |c_{k1}| + |c_{k2}| + |c_{k3}|$, is the sum of the absolute values of the k'th row in the matrix $C = A^{-1}$).

As indicated in the error analysis the error increases proportionally with α, β, γ, so we therefore linearly increase the size of the *bins* in our hash table for larger invariants. Bases in our images which are reasonably short are also more affected by poor corner localisation.

3.5 The Recognition Process

The actual recognition process is now outlined, and some results are given.

1. Extract corner points from the initial image, these points are then tracked as the object moves.
2. Select four of these points to form the basis, and calculate invariants for all the other points. A good selection of basis will provide a better estimate of the invariants. If the recognition process is a failure a different basis can be tried. Kalman filtering ensures optimal estimation of the 3D invariants.
3. For each invariant set index into the hash table. For every (Object , Basis) grouping which appears in the associated bin register a vote in a recognition tree (figure 8). The recognition tree is shown as a quick way of registering votes for a particular (object, basis) match, which

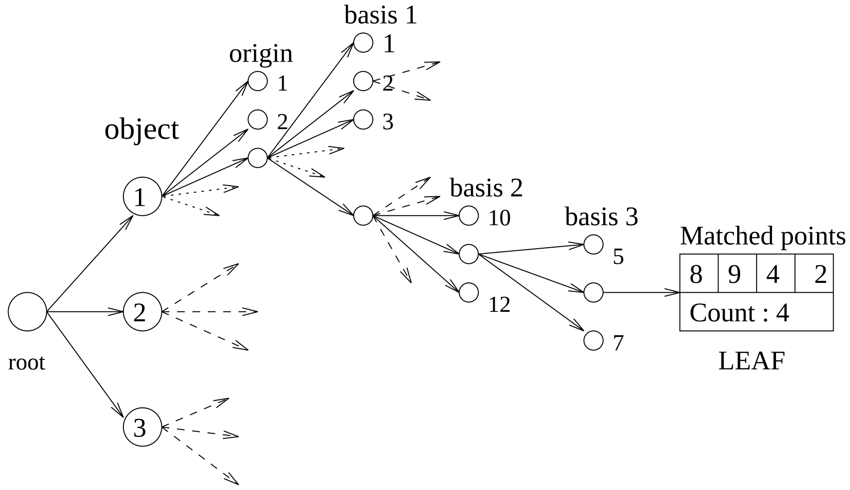


Figure 8: Recognition Tree: For each entry in a bin we register a vote in our tree.

is faster than keeping and updating a list of possible matches. Use the object number, origin point and then consecutive basis points to select which branch to follow. At the final leaf register a vote for this grouping by incrementing a counter. The point match for this invariant set is also stored here. In order to account for uncertainty in the 3D invariant estimate, examine not just the indexed bin but also local surrounding ones.

4. If a grouping (Object , Basis) scores a large number of votes, then this object is possibly present in the scene.

5. Compute the transformation (2) between matched model and real data points. A minimum of four points are required to calculate this transformation. At least five point matches (usually more) are available, so least squares minimisation is used to find a solution. Verify that the least squares solution to the transformation projects all model points satisfactorily to their matched image points.

6. Transform the edges of the model according to this transformation. Verify how close the projected edges lie to scene edges, if there is good correlation then the object has been recognised. Models that have been successfully projected onto the image are shown in figures 5 and 9. Failure to match the edges requires the process to be restarted with a different basis.

4 Summary and Conclusions

An object recognition scheme has been proposed and implemented which is successful at identifying simple 3D objects. Central to the system is an object representation that is viewpoint invariant under the weak perspective assumption, allowing fast matching with a model database. The representation is inherently robust to partial occlusion, since the invariants are obtained from groups of just five corner points. By tracking more features we provide enough information to determine a small number of possible object matches, which can then be evaluated by matching image edges with projected model edges. Certain issues remain which include increasing the speed and reliability of the tracking system to ensure our 3D invariants are as accurate as possible. We also wish to further examine the problem of exponential growth of data in the hash table, which occurs with increasing numbers of features on an object.

The aim of our future work is test the performance of this system with more complex objects and in cluttered scenes. We will be evaluating the discriminating power and computational efficiency of the method under such conditions.

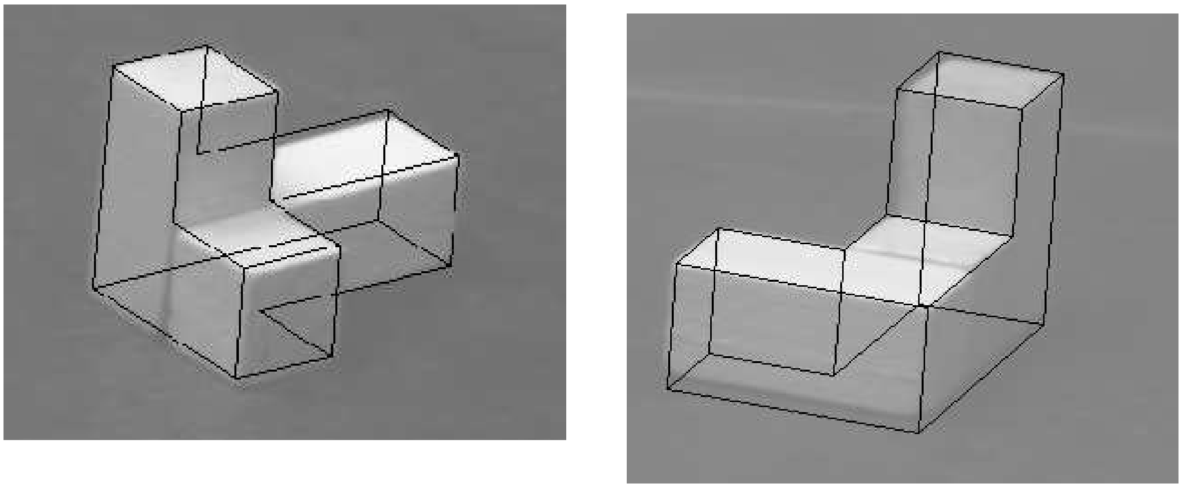


Figure 9: Recognised model is projected onto various shapes.

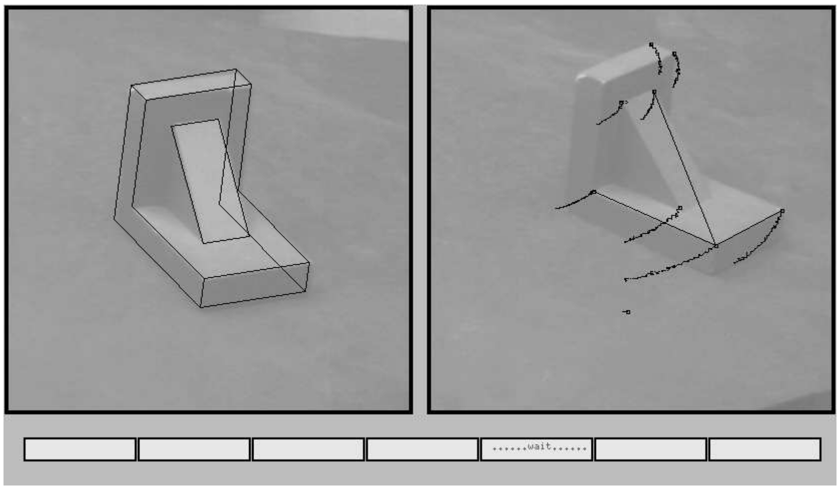


Figure 10: Another track and projection

References

[1] R.T.Chin and C.R.Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[2] P.J.Besl and R.C.Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[3] D.G.Lowe. The viewpoint consistency constraint. *Int. Journal of Computer Vision*, 1:57–72, 1987.

[4] D.W. Thompson and J.L. Mundy. Three-dimensional model matching from an unconstrained viewpoint. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 208–220, 1987.

[5] W.E.L.Grimson and T.Lozano-Perez. Localising Overlapping Parts by Searching the Interpretation Tree. *IEEE Trans. Pattern Analysis and Machine Intell.*, 9(4):469–482, 1987.

[6] M.Rygol, S.Pollard, and C.Brown. A multiprocessor 3D vision system for pick and place. In *Proc. 1st British Machine Vision Conference*, 1990.

[7] C Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. Journal of Computer Vision*, 9(2):137–154, 1992.

[8] J.L. Mundy and A.Zissermann editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[9] Y.Lamdan, J.T.Schwartz, and H.J.Wolfson. Affine invariant model-based object recognition. *IEEE Trans. on Robotics and Automation*, 6(5):578–589, 1990.

[10] L.G.Roberts. Machine perception of three - dimensional solids. In J.T. Tippet, editor, *Optical and Electro-optical Information Processing*. MIT Press, 1965.

[11] J.J.Koenderink and A.J.van Doorn. Affine structure from motion. *J. Opt. Soc. America*, 8(2):377–385, 1991.

[12] E.B.Barrett et al. In J.L. Mundy and A.Zissermann, editors, *Geometric Invariance in Computer Vision*, chapter 14, pages 277–292. MIT Press, 1992.

[13] H.J.Wolfson and Y.Lamdan. In J.L. Mundy and A.Zissermann, editors, *Geometric Invariance in Computer Vision*, chapter 17, pages 335–353. MIT Press, 1992.

[14] L.Kitchen and A.Rosenfeld. Gray-level corner detection. *Pattern Recog. Letters Vol.1*, pages 95–102, 1982.

[15] L.Dreschler and H.-H.Nagel. Volumetric model and 3d-trajectory of a moving car derived from monocular tv-frame sequences of a street scene. *Proc. Int. Joint Conf. on Artif. Intell.*, pages 692–697, 1981.

[16] C.G. Harris. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 16. MIT Press, 1992.

[17] K.Paler et al. Local ordered grey levels as an aid to corner detection. *Pattern Recognition Vol.17 No.5*, pages 535–543, 1984.

[18] H.Moravec. *Robot Rover Visual Navigation*. U.M.I. Research Press, 1981.

[19] H.Plantinga and C.R.Dyer. Visibility, occlusion and the aspect graph. *Int. Journal of Computer Vision*, 5(2):355–39, 1990.

[20] D.P.Huttenlocher W.E.Grimson and D.Jacobs. A study of affine matching with bounded sensor error. In *2nd European Conf. on Computer Vision*, 1992.

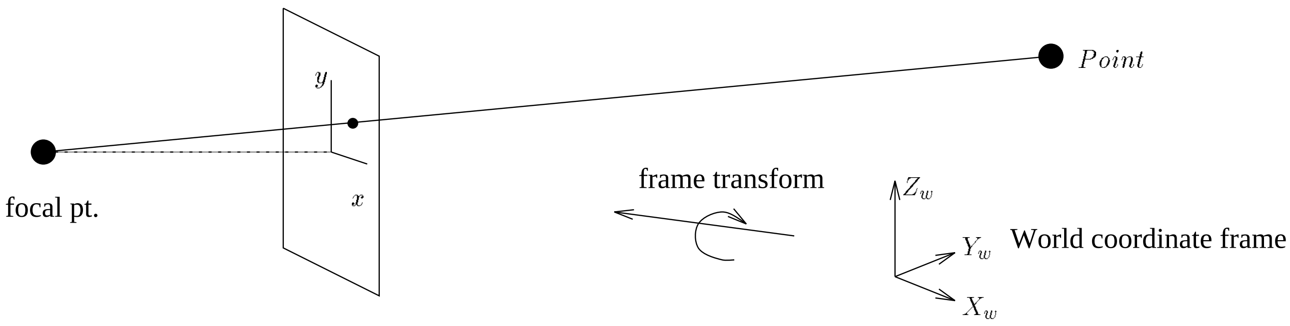


Figure 11: Perspective and Weak Perspective Projection

A Camera Projections

A.1 Rigid motion

If we define an object by the 3-dimensional coordinates of its vertices, and allow this object to be rotated and translated in space the transformation of our vertices is given by (12):

$$\underline{X}' = R\underline{X} + \underline{T} \tag{12}$$

where $\underline{X}'$ represents the new coordinates of a vertex, $\mathbf{R}$ is a 3x3 rotation matrix and $\underline{T}$ is a translation vector. This transformation can be used to convert vertex coordinates in the world frame to ones in the camera frame. By using homogeneous coordinates we can write this in a simple matrix form (13). This matrix will be referred to as $\mathbf{T_{ext}}$.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{13}$$

A.2 Pinhole camera model

In the pinhole camera model, 3-dimensional points project to image points in such a way that lines joining them all pass through a single point, known as the *focal point* (Figure 11). This transformation is known as *perspective* projection. We can express the image coordinates (x, y) of an object in terms of the camera focal length and the objects coordinates in a camera reference frame:

$$x = \frac{fX}{Z} \qquad\qquad y = \frac{fY}{Z} \tag{14}$$

We can convert (x, y) into pixel values (u, v) with a simple linear transformation, which scales the axes and shifts the origin.

$$u = K_x x + u_0 \tag{15}$$
$$v = K_y y + v_0 \tag{16}$$

All of these *intrinsic* camera parameters can be expressed in a single equation (17), which transforms points in a camera reference frame to pixel coordinates in an image. This matrix will be referred to as $\mathbf{T_{int}}$. (Note that T_{int} could also incorporate a more complex set of intrinsic parameters.)

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} K_x & 0 & \frac{u_0}{f} & 0 \\ 0 & K_y & \frac{v_0}{f} & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{17}$$

A.3 General Projection

General projection of the world points to image pixels can be expressed as $\mathbf{T} = \mathbf{T_{int}}\mathbf{T_{ext}}$, and is represented below. The matrix combines the effects of world - camera alignment and pinhole projection.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{18}$$

B Corner Detection

The method implemented in the following experiments is particularly suited to detecting object vertices. It locates corners at points where several lines intersect, by finding the gradient direction of points in the local area and forming a *gradient direction histogram* with this information. Peaks in the direction histogram indicate the presence of lines in a neighbourhood. Line intersections are found using a simple weighting algorithm. The procedure is tolerant to noise and reasonably fast, and is outlined in more detail below.

Edge detection

Use Prewitt or Sobel operators to calculate g_x and g_y, the intensities in the x and y directions for the image at each point. From these the gradient magnitudes ($0 \le g_m \le 255$) and directions ($-\pi/2 \le g_d \le \pi/2$) can be calculated.

$$g_m = (g_x^2 + g_y^2)^{\frac{1}{2}} \tag{19}$$

$$g_d = tan^{-1}(\frac{g_y}{g_x}) \tag{20}$$

An edge point $\mathbf{e}$ is defined as a point with ($g_m > \sigma_{threshold}$). Thin all the edges by finding their maxima perpendicular to the edge direction, and discard all other edge points.

The Histogram

For each edge point $\mathbf{e}$ left after thinning construct a gradient direction histogram:

- First define a window W(x,y) around the point $\mathbf{e}$.
- For each point in W(x,y) if $g_m(x,y) > \sigma_{threshold}$ enter the quantised value of $g_d(x,y)$ into the histogram.
- Each significant peak in the histogram represents a line. Where several lines (several peaks) are present there is a possibility these lines intersect thus forming a corner.
- The histogram wraps around, *ie* the final column is close in terms of direction to the first.
- In order to reduce effects from noisy points we only consider those edge points in W(x,y) with a direction found to be a peak in the histogram (*ie* points on an edge).

If there are 2 or more peaks in a histogram we can mark $\mathbf{e}$ as a possible corner point $\mathbf{c}$. The result of processing the whole image in this way is to give clusters of $\mathbf{c}$ points around each real corner in our image. An example of such a histogram is given in figure 13. The next task is to find the exact position of the line intersection.

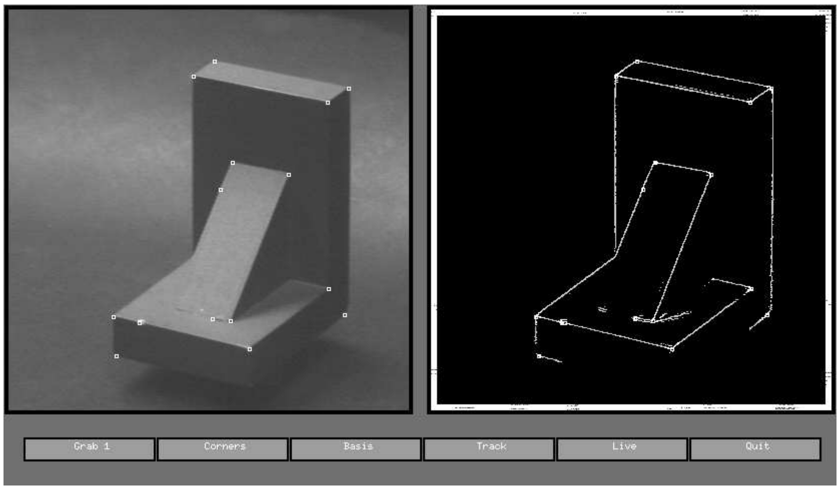
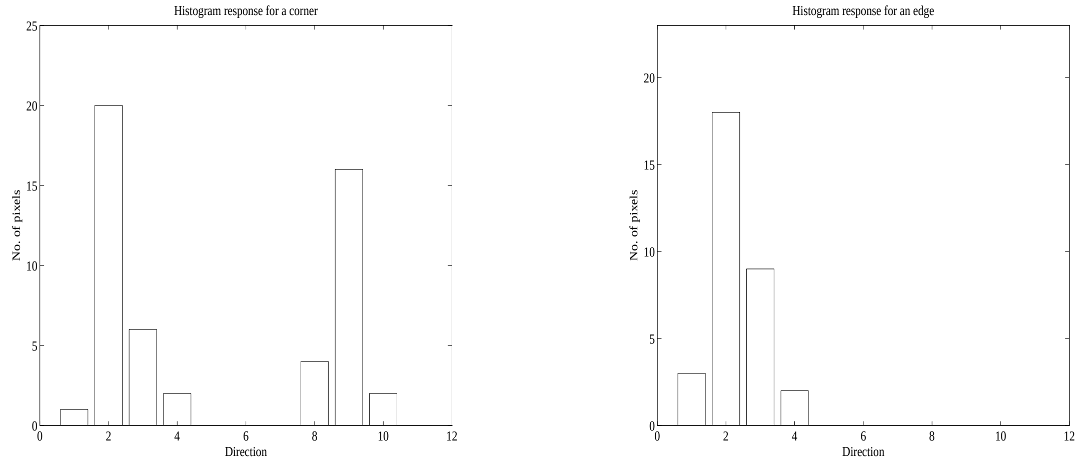


Figure 12: Corners detected on a wooden block



$-\pi/2$ (a) $\pi/2$ $-\pi/2$ (b) $\pi/2$

Figure 13: (a) The histogram for a point near a corner. (b) The histogram for a point on an edge.

Localisation

A candidate corner point **c** needs to lie at the intersection of local lines to be selected as the localised corner point. The perpendicular distance (21) from each local edge point to **c** is calculated.

$$dist_{x,y}(i,j) = |(x-i)cos(g_d(i,j)) + (y-j)sin(g_d(i,j))| \tag{21}$$

If the mean of these is small then **c** lies at the intersection of the lines. The candidate corner with the smallest mean in a local area is selected as being the localised corner point.

This algorithm was found to work well, especially for sharp corners. However if two corners lie close together or there is an excessive amount of noise, localisation will be poor as with most other types of corner detector. In Figure 12 an example of the corner detector results can be seen. Some dents on the object have also been detected.

C The Kalman Filter

The Kalman filter is a recursive linear estimator which successively calculates a minimum variance estimate for a state, on the basis of periodic observations that are linearly related to this state. The basic algorithm consists of a continuous "prediction - observation - update" cycle, where the error between prediction and observation is used to revise the old state estimate x_k^-.[1]

Prediction - Observation - Update

The difference between the observation z_k and the predicted observation $H_k x_k^-$ is termed the innovation v_k:

$$v_k = z_k - H_k x_k^- \tag{22}$$

The innovation measures the deviation between filter estimates and observation sequence. It is used to build a model of the observations and to validate new measurements, to see whether they should be incorporated in the updated filter. We also compute the variance associated with the innovation:

$$S_k = E[v_k v_k^T] = H_k P_k^- H_k^T + R_k \tag{23}$$

where R_k is the variance of the observation noise. We can now calculate the Kalman gain, W_k, which allows us to weight the observation and prediction according to the confidence we have in their respective values.

$$W_k = P_k^- H_k^T S_k^{-1} \tag{24}$$

We now use this gain matrix to update our state and covariance using the following equations:

$$x_k = x_k^- + W_k v_k \tag{25}$$

$$P_k = P_k^- - W_k S_k W_k^T \tag{26}$$

These values are then used to make our next prediction.

$$x_{k+1}^- = x_k \tag{27}$$

$$P_{k+1}^- = P_k + Q_k \tag{28}$$

where Q_k is our process noise which we assume to be 0.

The innovation can also be used to define a "normalised" measure of the difference between observation and prediction, termed the validation gate D_k. Fixed values of D_k define ellipsoids of constant probability, and we can therefore use a threshold value of D_k to validate our observations.

$$D_k = v_k S_k^{-1} v_k^T \tag{29}$$

Initialisation

The initialisation of the filter is done by calculating a batch least squares solution for a few of the initial observations. This will return both a value for x_0 and P_0 indicating an initial estimate of the affine coordinates and a covariance matrix. In fact we may want to increase P_0 just to ensure that it is more significant than the process noise by a factor of 5-10. It is important that R_k, our observation noise, is set correctly since it governs the confidence we have in our observations. Ideally experimental analysis should be used to ascertain its value.

[1] A "-" superscript denotes the old variable value.

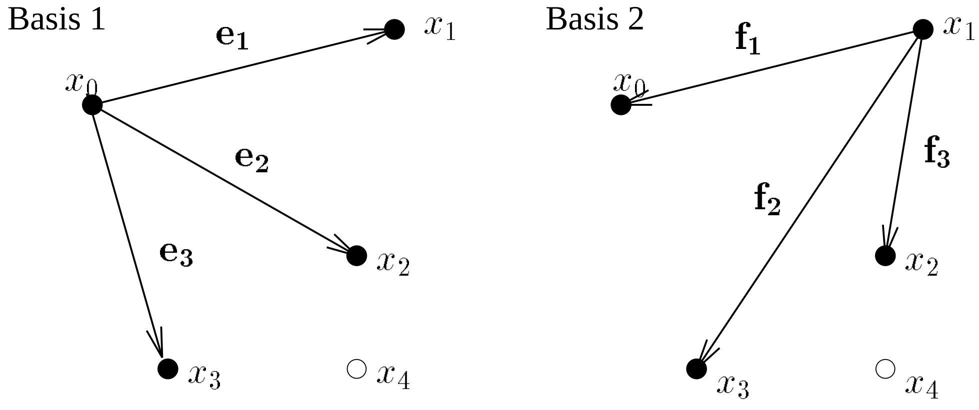


Figure 14: Four points can form different 3D bases.

D Vector Geometry

Given four points $x_0 \ldots x_3$, it is possible to form a 3D basis in several different ways. Figure 14 shows two such bases, with different origins, but formed from the same points. These two bases are related, and this appendix shows the link between invariants in the different bases.

It is possible to write vectors $\mathbf{f_1} \ldots \mathbf{f_3}$ in basis 2 in terms of vectors $\mathbf{e_1} \ldots \mathbf{e_3}$ of basis 1.

$$\begin{aligned} \mathbf{f_1} &= -\mathbf{e_1} && (30) \\ \mathbf{f_2} &= \mathbf{e_3} - \mathbf{e_1} && (31) \\ \mathbf{f_3} &= \mathbf{e_2} - \mathbf{e_1} && (32) \end{aligned}$$

Any point x_4 in these bases can be written as:

$$x_4 = x_0 + \alpha \mathbf{e_1} + \beta \mathbf{e_2} + \gamma \mathbf{e_3} \tag{33}$$

or

$$x_4 = x_1 + \alpha' \mathbf{f_1} + \beta' \mathbf{f_2} + \gamma' \mathbf{f_3} \tag{34}$$

in basis 1 and 2 respectively. Using the fact that $\mathbf{e_1} = x_1 - x_0$, and equating (33) and (34):

$$\alpha \mathbf{e_1} + \beta \mathbf{e_2} + \gamma \mathbf{e_3} = \mathbf{e_1} + \alpha' \mathbf{f_1} + \beta' \mathbf{f_2} + \gamma' \mathbf{f_3} \tag{35}$$

substituting for $\mathbf{f_1} \ldots \mathbf{f_3}$:

$$\alpha \mathbf{e_1} + \beta \mathbf{e_2} + \gamma \mathbf{e_3} = \mathbf{e_1} - \alpha' \mathbf{e_1} + \beta' (\mathbf{e_3} - \mathbf{e_1}) + \gamma' (\mathbf{e_2} - \mathbf{e_1}) \tag{36}$$

We can now find how invariants in the two bases match.

$$\begin{aligned} \alpha &= 1 - \alpha' - \beta' - \gamma' && (37) \\ \beta &= \gamma' && (38) \\ \gamma &= \beta' && (39) \end{aligned}$$

Similar results can be calculated for all other bases made from these points. It turns out that $(\alpha + \beta + \gamma), (1 - \alpha), (1 - \beta)$ and $(1 - \gamma)$ are four values that are general invariants of any basis formed using $x_0 \ldots x_3$, the only problem being these could be returned in any order, depending on basis. The L_2-norm of these terms would however be completely invariant.

$$Inv_{L2} = sqrt((\alpha + \beta + \gamma)^2 + (1 - \alpha)^2 + (1 - \beta)^2 + (1 - \gamma)^2) \tag{40}$$