Using Projective Invariants for Constant Time Library Indexing in Model Based Vision

C.A. Rothwell, A. Zisserman, D.A.Forsyth and J.L. Mundy*
Robotics Research Group
Department of Engineering Science
University of Oxford
Oxford OX1 3PJ

Abstract

Projectively invariant shape descriptors allow fast indexing into model libraries, because recognition proceeds without reference to object pose. This paper describes progress in building a large model based vision system which uses many projectively invariant descriptors. We give a brief account of these descriptors and then describe the recognition system, giving examples of the invariant techniques working on real images. We demonstrate the ease of model acquisition in our system, where models are generated directly from images. We demonstrate fast recognition without determining object pose or camera parameters.

1 Introduction.

We use *projective invariants* to produce a model based recognition system which references models in time constant with respect to the size of the model base. Typical model based recognition systems, based on pose determination (such as the work of Lowe [7]), have recognition time linear in the size of the model base. In such systems, recognition proceeds by extracting a collection of features from the image, hypothesising a correspondence between these features and an appropriate set of model features, and determining the position and orientation (pose) of the model from this hypothesis. The hypothesis is verified by using the computed pose to predict further image features, and confirming that some or all of these features are found as predicted. There are two problems with this approach: firstly, to identify all model instances in the image exhaustively each model must be tested; secondly, there are many possible matches between image and model features, and each match is expensive to verify. Consequently, on a serial machine, recognition time is linear in the number of models, and the constant of proportionality is large. As each individual match is expensive the constant of proportionality is large and so linear time systems are slow.

There have been a few attempts to produce recognition in sub-linear time, for example [4], but again the constants of proportionality are high. If real time recognition is to be performed for large model bases, the processing time must be independent of the size of the model base. A few authors have achieved this [5, 6, 9, 15]. These systems all use projective invariants, which can directly identify a model instance from image features. Projective invariants are properties of plane shapes which are unaffected by perspective imaging. The values of invariants measured in the image are the same as the values measured on the model for corresponding features. This means that invariant values measured in the image index a model immediately. Verification proceeds as for [7]; however, fewer

*CAR acknowledges the support of General Electric. AZ acknowledges the support of the United Kingdom Science and Engineering Research Council. DAF acknowledges the support of Magdalen College, Oxford. JLM acknowledges the support of the General Electric Coolidge Fellowship. The General Electric Corporate Research and Development Laboratory is supported in part by the following: DARPA contract DACA-76-86-C-007, AFOSR contract F49620-89-C-003.

BMVC 1991 doi:10.5244/C.5.9

hypotheses need to be verified because the object has been identified. Thus, inappropriate matches (between image features and either the wrong features on the right model, or features on the wrong model) are avoided.

Our technique relies on building a list of invariants for each model, these are referred to as *feature vectors*, which consist of all the possible invariants for an object. Each component of the vector is a potential recognition cue. For most objects this vector will be large and so only a few features need to be visible to enable recognition. This promotes stability to occlusion and noise. Much of our previous work has concentrated on using the two invariants of pairs of coplanar conics. We have successfully built and tested a simple recognition system using pairs of conics. Once recognised, an object's pose can be quickly determined [11]. Apart from pairs of coplanar conics many other plane projective invariants exist, examples are given in section 2.

This paper describes the use of these invariants within a model based recognition system. First we state the form of each invariant, we then describe model acquisition and recognition. We then show recognition hypothesis verification. Examples are based entirely on real objects and images.

2 Planar Projective Invariants.

Example 1: The Cross-Ratio.

The cross-ratio for four collinear points $\mathbf{x_i}$, $i \in \{1,..,4\}$ along a line with linear parameterisation θ_i is [13]:

$$\{\mathbf{x_1},\mathbf{x_2},\mathbf{x_3},\mathbf{x_4}\} = \left(\frac{\theta_1-\theta_3}{\theta_1-\theta_4}\right) \Big/ \left(\frac{\theta_2-\theta_3}{\theta_2-\theta_4}\right) \tag{1}$$

Four concurrent lines can also be used to form a cross-ratio. This follows directly for the duality of lines and points in projective space.

Example 2: Five Coplanar Points.

Given five coplanar points $\mathbf{x_i}$, $i \in \{1,..,5\}$, two projective invariants are defined:

$$I_1 = \frac{|M_{431}||M_{521}|}{|M_{421}||M_{531}|} \qquad I_2 = \frac{|M_{421}||M_{532}|}{|M_{432}||M_{521}|} \tag{2}$$

where $M_{ijk} = (\mathbf{x_i},\mathbf{x_j},\mathbf{x_k})$ and $|M|$ is the determinant of M. Should any triple of points become collinear the first invariant is undefined. An alternative definition appears in [8]. Again, we may use lines instead of points to form the invariants.

Example 3: A Conic and Two Points.

A conic, C, and two points not lying on the conic define a single invariant given by [5]:

$$I = \frac{(\mathbf{x_1}^{\mathrm{T}} C \mathbf{x_2})^2}{(\mathbf{x_1}^{\mathrm{T}} C \mathbf{x_1})(\mathbf{x_2}^{\mathrm{T}} C \mathbf{x_2})} \tag{3}$$

where C is the matrix of the conic: a conic takes the form $ax^2+bxy+cy^2+dx+ey+f=0$ which can be expressed as the quadratic form:

$$[x,y,1]\begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0, \tag{4}$$

or as $\mathbf{x}^{\mathrm{T}} C \mathbf{x} = 0$. The envelope of the conic and two lines is the dual system. The envelope of a conic C is the conic C^{-1}, and so two lines and a conic yield an invariant as well.

Example 4: A Pair of Conics.

Using the notation of a conic matrix C_i in example 3 above, two projectively invariant measures can be formed for a pair of conics normalised so that $|C_i| = 1$. These are:

$$I_1 = Trace[C_1^{-1}C_2] \quad \text{and} \quad I_2 = Trace[C_2^{-1}C_1] \tag{5}$$

3 The Model Based System

3.1 Overview.

Before recognition is possible the models must be acquired and added to the library. A model consists of a set of lines and conics, the invariants formed by them, and a model name. All the model data are extracted from unoccluded views of the object, which make acquisition easy. The recognition process is then broken down into three stages:

1. **Feature extraction.** The conics and lines needed to form the invariants we use are extracted from image edge data.
2. **Hypothesis generation.** The invariants for groups of features are computed. We index the measured invariants against invariant values in the library using a hash table, and if they match, produce a recognition hypothesis.
3. **Hypothesis verification.** When a potential match is found we confirm it by projecting edge data from an acquisition image to the test scene. Should the projected and scene edge data be sufficiently close the match is confirmed.

The curve segmentation we use and the above processes are described in detail in the rest of this section.

3.2 Curve Segmentation.

The aim of the segmentation methods we use is to extract straight lines, conics, and higher order curves from edge features so that we can form invariants. Points are hard to locate accurately from edge data, so we locate them by intersecting lines or conics. So far we have used only lines and conics because the points found by their intersection do not yield any functionally independent invariants.

Continuous edge curves, or edge curves with single pixel breaks, are taken directly from Canny output [2]. For each curve, the curvature (κ) and its derivative ($\dot{\kappa}$) are determined as functions of arc length. The curvature representation is not projectively invariant, but because we are interested in finding high curvature points the approximation is good enough. Both κ and $\dot{\kappa}$ are measured by differentiating the slope of a local linear fit to the edge data and smoothing with a 1-D Gaussian. This is done with a discrete convolution kernel. Points on the curve where both κ and $\dot{\kappa}$ exceed thresholds are marked [10], and the curve broken into separate line segments at these points. We have found this provides effective segmentation. If the line segments are long enough we attempt to classify them by:

1. Fitting a straight line to the curve by orthogonal regression. If the fit is good enough it is accepted, else move on to (2). Points within one standard deviation (of the smoothing Gaussian) of the ends of the segment are discarded before fitting as these are corrupted during smoothing.
2. Fit a conic to the curve by minimising the algebraic distance [1]. Accept the fit if the algebraic distance is small enough, else go to (3).
3. If the curve is not a straight line or a conic store the edge data for later use. Such curves may not be algebraic, but they do possess invariants [12, 14].

The straight line and conic fitting procedures are both in closed form (by eigenvalue decomposition [1]), and so the computational cost of fitting is low. An example of line and conic fitting to image edge data is shown in figure 2. Occlusion and noise can cause extra segmentation of image curves and so pairs of lines or conics may represent the same line or conic. This will only happen to curves adjacent in edge point lists. Therefore we test whether consecutive lines have the same or nearly the same equations. This is done by testing whether the endpoints of one line segment lie close to the the other line. If two lines are similar we refit a single line to the two original sets of edge data. We do a similar test on conics. For a review of the method see [11]. We have found that these simple heuristics produce good results.

The segmentation produces a disjoint set of lines and conics. We then group the straight lines into graphs. If the endpoints of two lines are within a few pixels of each other the lines are marked as being connected. 'T' junctions are also formed if a line endpoint lies close to another line but does not form an 'L' junction with it. These are used during recognition.

All lines extracted from a single edgel list are then grouped together into a simple chain. The chains consist of lines separated by conics, and non-conic or non-linear edge points, such as corners, etc. We group image features to cut down the matching combinatorics. For example to determine all the five line invariants for an image containing n lines we need to compute $O(n^5)$ invariants[1] This is too many and we lose the benefit of having linear library indexing[2]. Therefore we group the lines into chains using the connectivity of the edges and compute invariants only for consecutive lines in a chain. This is $O(n)$. Hence we have potentially fast recognition. We do, however, ignore many potential matches because of poor segmentation, or because other objects occlude the line chains and so break them into distinct pieces. At present no grouping is done between different chains, and so no invariants can be computed for segments in the separated groups. Because we are still able to compute a large number of invariants from broken edge chain the chances of recognition remain high.

3.3 Model Acquisition and Library Formation.

The invariants we use are for planar objects, or for objects with plane faces. A novel aspect of this work is the ease of model acquisition. We extract all possible invariants from an unoccluded view of the isolated object. These invariants are entered into a feature vector, defined as the set of model invariants. We measure another feature vector from a different viewpoint. We then intersect the two sets represented by the feature vectors, and form the object feature vector from all invariants which are measured in both views. The invariants in the object feature vector are included in the library (described below).

If a measurement changes between two different view-points we know that the features used to compute the measurement are not coplanar, or that the features are produced by variable segmentation, and so the measurement is not useful. If the measurement is constant it is an invariant because invariants are constant for all views. All the constant measures are the model descriptors, and are stored. The edge data forming all the coplanar features for an object are also stored for use in the hypothesis verification stage.

Invariant model libraries are very simple. The library we use is split up into different sub-libraries, one for each type of invariant. Each sub-library then has a list of each of the invariant values tagged with an object name, and is structured as a hash table. The hash table must be large (in terms of memory size) to provide sufficient distinguishing

[1]Only $O(n)$ of the invariants are functionally independent.

[2]Compared to pose methods that only match three points, which is $O(n^3)$. Including the model matching, for λ models, gives $O(\lambda n^3)$, which is better than exhaustive invariant matching for $\lambda < n^2$. We expect the number of image features n to be large.

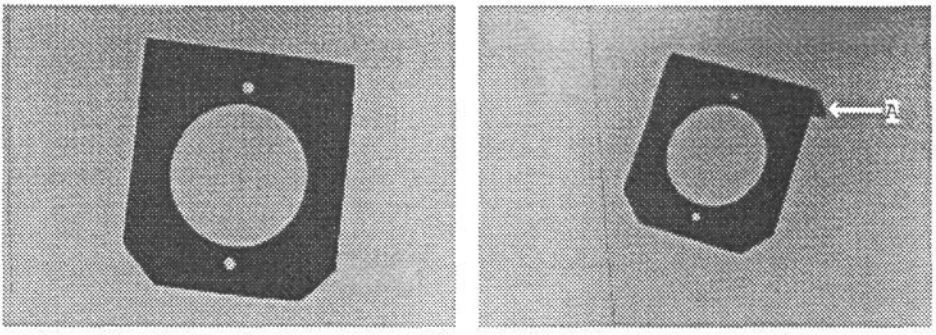

Figure 1: The left image shows one of the calibration scenes for the a right-angled bracket. A different view of the bracket is shown in the right image. Feature 'A' produces an extra line in the edge chain, and so the measured invariants are different from the first calibration image. We are still able to find two pairs of matching line invariants as well as four conic/line-pair invariants between the two images. All the invariant values for the bracket are computed from the fitted forms for the two scenes, and compared to each other All invariants that stay approximately constant are entered into the feature vector for the object.

power between the invariants assigned to the models. For smaller buckets there is more distinguishing power, and so a lower likelihood of collision, but more buckets are needed. So long as the machine's virtual memory is local the buckets can still be accessed quickly when there is fine distinguishing power. This makes model indexing very fast. When an invariant is measured in a scene we choose the relevant library, depending on the invariant type, and hash in to find the corresponding object.

As an example we show two views of a right angle bracket in figure 1. From each image we measure all possible invariants. The five line invariants are formed by taking all possible sequences of five lines around the line chains, and the conic/line-pair invariant are formed from the conic fit to the central hole in the bracket and all pairs of consecutive lines in the chain. The measured invariants are shown in tables 1 and 2. All invariants that match to within a small tolerance (3%), are entered into the model library. The invariant vectors for each image are different because of feature 'A' shown in figure 1. We note that the conic/line-pair invariants tend to be less stable than the five line invariants.

3.4 Hypothesis Generation.

Once an object's invariants have been stored in the database it can be recognised from a cluttered scene without any knowledge of pose or camera calibration. For each feature group in a test scene we form a feature vector. These vectors will be large, containing a number of invariant values corresponding to actual object features as well as invariants derived from other objects in the scene but not in the library.

At this stage we make use of the 'T' junction labelling performed during the grouping phase. The 'T' junctions provide a heuristic which cuts down search. Two lines forming a 'T' junction often arise from occlusion, and thus are not part of the same plane object boundary. Therefore if we find two consecutive lines in a chain connected by a 'T' junction we do not compute any invariants using both of the lines.

Each measured invariant is then matched via the hash table to an object name in the model library if the invariant value (within some error bound) is stored in the library.

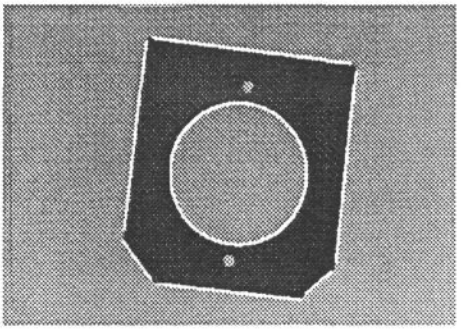

Figure 2: This image shows the lines and conic fitted to the bracket for the first calibration scene. The edge curves are broken at points of high κ and $\dot{\kappa}$. Note that the endpoints of the lines are not connected. This is because the smoothing kernel corrupts points towards the ends of the lines, and so we ignore these points.

Image	Five Line Invariant Values					
	1	2	3	4	5	6
1	0.553,2.118	0.679,3.572	-	-	-	0.844,1.229
2	-	-	0.527,-41.855	1.013,0.984	0.539,3.477	0.839,1.239
deviation %	-	-	-	-	-	0.3,0.4

Image	Five Line Invariant Values				
	7	8	9	10	11
1	0.451,5.200	0.837,1.412	0.684,1.880	-	-
2	0.458,5.040	-	-	0.896,1.245	0.658,1.513
deviation %	0.8,1.6	-	-	-	-

Table 1: These two tables show all the five line invariants extracted from the two calibration images. Each image row shows the invariant feature vector for the corresponding image. The object feature vector is given by the intersection of these two rows. The feature vectors for each image differ because of the difference in the apparent outline of the object in the two images. The % deviation between matching invariants is shown in the last row of each table.

Image	Conic/Line-Pair Invariant Value							
	1	2	3	4	5	6	7	8
1	2.493	2.470	-	-	1.331	1.311	1.321	1.336
2	2.249	-	1.401	1.186	1.294	1.316	1.288	1.293
deviation %	9.8	-	-	-	2.3	0.4	2.5	3.2

Table 2: This table shows all the conic/line-pair invariants extracted from the calibration images. The invariants are ordered around the extracted edge chain which helps match the invariants from different scenes. The deviations show that the conic/line-pair invariants are less stable than the five line invariants. The first invariants shown, which have differences of 9.8% to the mean, would not be accepted as the error is too large.

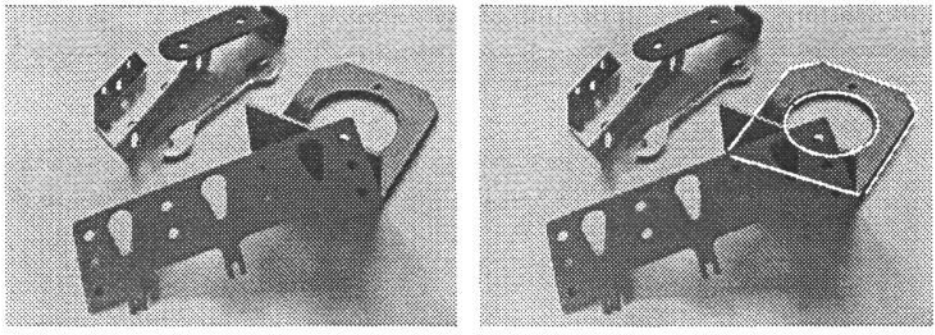

Figure 3: The left image shows the bracket occluded in a scene by objects not in the library. On the right the edge data from the first calibration scene are shown projected onto the test scene using the model to image transformation hypothesised by the match. The close match between the projected data (shown in white), and the scene edges shows that the recognition hypothesis is valid.

invariant	library	scene	error %
five line	(0.8415,1.2340)	(0.813,1.289)	(3.4,4.5)
conic/line	1.3125	1.262	3.7
conic/line	1.3135	1.321	0.6
conic/line	1.3045	1.336	2.4

Table 3: This table shows all the invariants from the scene which are formed by features actually corresponding to bracket features. The second column shows the library values and the third column scene values.

Should an object name be matched a sufficient number of times within a vector a recognition hypothesis is formed. We can then verify the hypothesis by projecting model edge data which are taken from the calibration scene onto the test scene and checking that there is sufficient correlation. This verification technique is similar to that used in [3], and is explained below.

As an example of object recognition we demonstrate recognising a bracket in a scene with occlusion and clutter caused by other objects (figure 3). All possible five line invariants and conic/line-pair invariants were measured and the matching invariants are given in table 3. From a scene such as this a large number of possible invariants can be derived. We found that only one five line invariant matched any of the five line invariants of the bracket, and this was a correct match. The clutter does not produce false positives for this invariant. A number of conic/line-pair invariants were measured in the scene which match the invariants of the bracket, but are not formed by bracket features. Hence the conic/line-pair invariant is less likely to produce a unique match. The false positives can be overcome during the hypothesis verification stage. Alternatively, if different measured invariants have common model and image features we may merge the individual hypotheses to form *joint hypotheses*. The invariants may not necessarily be compatible and so a list of joint hypotheses is formed which contains both the individual and the merged hypotheses. The list is ordered so that the joint hypotheses covering the most invariants are verified before those with fewer invariants.

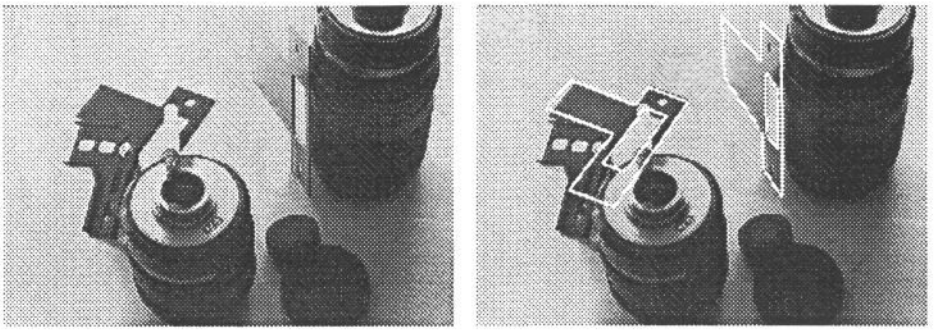

Figure 4: Two different objects (lock striker-plates) are recognised in this image even with substantial occlusion and significant perspective distortion.

3.5 Joint Hypothesis Verification.

Once the invariants have been extracted from an image and a potential match to a model found, we must perform verification to remove all false negatives. Verification is relatively expensive, but using invariants as recognition cues critically reduces the number of hypothesis tests that have to be made.

Verification does *not* involve determining object pose:

1. We determine the projective transformation between the calibration and test scenes using the lines that form the matching invariants (we will have at least five if we use the invariants of five lines)[3]. The transformation can be determined uniquely if the joint hypothesis maps three or more model lines into the image. If there are only two lines, as for a single conic/line-pair hypothesis, there is a four fold ambiguity in the transformation. This highlights the benefit gained from using the joint hypotheses to increase the number of model-scene feature matches.
2. We project the stored edge data from the calibration scene onto the test scene.
3. If the projected data and the scene data are close in the image (within 5 pixels) and of the right orientation (within 10^o) confirm the hypothesis. All edge data used to determine the invariants in the test scene should correspond to the projected edge data, though occlusion means that the converse is not necessarily true. The proportion of correspondence required for a match to be confirmed depends on the application. This technique is the same as that used in [3].

An example is shown in figure 3. The edge data from the first calibration image (figure 1) has been projected onto the test image and is shown in white. The close correspondence between this data and the test edge data (over 50% within 5 pixels and 10^o) confirms the recognition hypothesis.

A further recognition example is shown in image 4. Two different objects are found in the image by the recogniser even though one of the objects is substantially occluded and the other has undergone significant perspective distortion.

[3]We need a minimum of four lines or points to determine the projective transformation linking two representations. The method is given in [11]. We may also determine the projective transformation between a conic and a pair of lines, and their corresponding projection in a different frame.

4 Conclusions.

This paper describes the initial phases in the development of a large model based vision system which uses projectively invariant descriptors. The methods used are shown to be workable, and by combining many different types of invariant we can measure large feature vectors for objects and so tolerate occlusion and image noise. The problems of extracting invariants from feature groups and of the use of invariants to produce recognition hypotheses has also been discussed. Future work on the project will include:

1. Determining stability criteria so that we know how sensitive each type of invariant is to imaging noise.
2. Determining error bounds for measured invariants.
3. Integrating invariants of higher order planar curves and parametric curved surfaces.
4. Including distinguished plane and differential invariants as recognition cues. One such invariant, reported in [12], has been tested and has been shown to provide a strong recognition cue.

The model base currently consists of twenty-one models, six of which are similar to the objects displayed in this paper, and the rest are the labels shown in [5]. So far, the actual time spent in the hypothesis generation phase is negligible, most of the processing time being for feature detection. Even with twenty-one models the model base is too small to draw any firm conclusions on the overall speed of indexing for a large model base, but future work should reveal whether constant time library indexing is really possible.

References

[1] Bookstein, F. "Fitting Conic Sections to Scattered Data." *CVGIP*-9, p.56-71, 1979.

[2] Canny J.F. "Finding Edges and Lines in Images," TR 720, MIT AI Lab, 1983.

[3] Connolly, C.I., Mundy, J.L,. Stenstrom, J.R. and Thompson, D.W. "Matching From 3-D Range Models into 2-D Intensity Scenes," Proceedings ICCV1, p.65-72, 1987.

[4] Ettinger, G.J. "Large Hierarchical Object Recognition Using Libraries of Parameterized Model Subparts," Proceedings CVPR, p.32-41, 1988.

[5] Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. "Invariant Descriptors for 3-D Object Recognition and Pose," To appear *PAMI*, 1991.

[6] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR, p.335-344, 1988.

[7] Lowe, D.G. *Perceptual Organization and Visual Recognition*, Klewer Academic Publishers, 1985.

[8] Mohr, R. and Morin, L. "Relative Positioning from Geometric Invariants," Proceedings CVPR, p.139-144, 1991.

[9] Nielsen, L. and Sparr, G. "Projective Area-Invariants as an Extension of the Cross-Ratio," Proceedings First DARPA-ESPRIT Workshop on Invariance, p.455-480, March 1991.

[10] Pridmore, A.P., Porrill, J. and Mayhew, J.E.W. 'Segmentation and Description of Binocularly Viewed Contours," *IVC*-5, No. 2, p.132-138, 1987.

[11] Rothwell, C.A., Zisserman, A.P., Marinos, C.I., Forsyth, D.A. and Mundy, J.L. "Relative Motion and Pose From Arbitrary Plane Curves," *IVC* in press, 1991.

[12] Rothwell, C.A. "Model Based Computer Vision Using Projective Invariants", First Year Report, Department of Engineering Science, Oxford University, Oxford, 1991.

[13] Semple, J.G. and Kneebone, G.T. *Algebraic Projective Geometry*, Oxford University Press, 1952.

[14] Van Gool, L. Kempenaers, P. and Oosterlinck, A. "Recognition and Semi-Differential Invariants," Proceedings CVPR, p.454-460, 1991.

[15] Wayner, P.C. "Efficiently Using Invariant Theory for Model-based Matching," Proceedings CVPR, p.473-478 ,1991.