IRLM: AN IMAGING SYSTEM FOR OBJECT RECOGNITION, LOCALIZATION, AND MOTION RECOVERY

Wu, J.J AND Caelli, T.M.

Department of Electrical Engineering
University of Alberta
Edmonton, AB, T6G 2G7, Canada

Abstract

In this paper, a system (called IRLM) is presented for object recognition, localization, and motion recovery using grey level images. The simplicity of IRLM is achieved by the mathematical formulation of perspective projections of features. The generic features used here are conic sections which are projected into easily detectable conic sections in images, and the matching process is tuned to conform to specific model constraints and projection equations.

To locate the object and recover its motion, coarse-to-fine algorithms (including closed-form and iterative solutions) are used. The recovered location and motion are used to predict other features' visibilities and attributes of the image. The predicted attributes are then used to direct the interpretation process and to verify the consistency of interpretation.

During interpretation, either top-down or bottom-up matching processes are used, depending on how much information is gathered at the time by the system. This allows IRLM to search the most needed information in identifying and locating the object in any given frame.

1. Introduction: Overview of IRLM

In this paper we present an efficient approach to object recognition and motion recovery based upon the projected properties of features and transformations of rigidly moving object features. In our implementation, object features such as conic sections are used to construct our models and other features that have known projection properties. The analysis of such feature transformations enable us to recover objects' identities and transfunctional states. The recovered motion states are then used to verify object recognition since features on a same rigid object should undergo the same transformation and it is this consistency algorithm and the hypothesis-and-test components which form the basis of the IRLM algorithm. Here the inputs of IRLM are intensity images, in contrast to range data used in most recent research (Faugarus, 1986; Pentland, 1986; Bolles & Horaud, 1984; Grimson & Thomas, 1984, 1985). Our IRLM has 4 basic dynamically interesting components:

1.1 *Feature Finder*.

The feature finder extracts conic sections from edge images and computes their attributes. Detected features of the same kind are organized according to their attributes for each retrieval.

1.2 *The Locator*.

The locator takes the matches of the model and image features as input and calculates the orientation and position of the object in space regardless of whether the matches are consistent or not.

1.3 *Motion Recoverer*.

The motion recoverer estimates the motion parameters so as to minimize the predicted and measured attributes of features on images.

1.4 *Recognizer*.

Recognition is defined by setting up correspondences between image and model features under specific constraints, determined by the shape properties and projection rules. Such constraints are useful for reducing the search space for the correspondence process.

Whenever enough information is gathered (e.g., three noncollinear line correspondences, or one ellipse correspondence), the recognizer calls the locator to determine the object's orientation and position. The recognizer then uses the obtained orientation and position to predict features in the image, and then the measured and predicted features. If the hypothesized matches are consistent with existing matches, they are assimilated into the set of consistent matches. The recognizer then calls the locator again to calculate the new location. This process is repeated until the identity of the object is uniquely determined.

2. Salient Computational Features of each Process

2.1 *The Feature Finder*.

The feature finder extracts conic sections from edge images. Edges are thinned to a single pixel in width, then grouped and fitted into different types of conic equations. After the feature type is classified, its measurements, such as slope and intercept of line, centre of ellipse, etc., are computed.

2.2 *The Locator*.

The locator consists of various locating modules for different shape interpretations. At the early stage, closed-form solutions are used because they require little computational effort. For example, an ellipse match gives two sets of numerical solutions, as summarized in the Appendix. As a second example, three line matches give, at most, four sets of possible solutions (see Wu, 1988, for details). On the other hand, at the final stage, a more accurate extended Kalman Filter (EKF) is used to report the location of the recognized object.

In the following discussion regarding location parameters, we use **O** for the orientation matrix, **p** for the position vector, and **q** for a quaterion expression of the orientation matrix.

2.3 *The Motion Rcoverer*.

The motion recoverer uses the set of consistent matches at each frame to estimate the motion in space between frames. This is implemented by using an EKF algorithm in which models of motion and perspective projection are constructed beforehand. The motion recoverer calculates the motion such that the differences between predicted and measured attributes of featues in the image are minimized (Wu, Rink, Caelli & Gourishankar, 1988).

2.4 *The Recognizer*.

Correspondences between Detected Features and Model Features. Object recognition is accomplished by determining whether a collection of detected features in the image belongs to a certain object. Due to occlusion or overlapping, detected features belonging to a same object may not be connected. Instead, they can be separated by other objects' imaged features. A set of probably disjoint features in the image is recognized as images of a certain object if a set of consistent correspondences between these detected features and the object's model features exist. There are two ways the recognizer can set up correspondences and check consistencies. One is to pair model features to a given detected image (Bottom-up). The other is to pair detected image features to a given model feature (Top-down).

Pairing Model Features to a Given Detected Feature in the Image. Suppose there are N object models, each of which has M features, and there are L features detected in the image. The aim of matching is to label each detected feature as one of the model features. This is equivalent to search a L-level tree, in which each level has NM nodes. If a brute-force search is used, $(NM)^L$ pairs of possible correspondences have to be tried in the worst case.

In order to reduce the search effort, projection rules and transformation constraints are used to prune the search tree and order the matching sequence. The projection rules are the relations given in Table 1. These rules are the "node" consistent constraints which say that a feasible model feature (a node) to be paired to to the given imaged feature is the one that can be projected to the same type of feature as the given imaged feature. For example, if a full ellipse is detected in the image, it must originate from a model ellipse; but if a partial ellipse is detected, it could correspond to a parabola or an ellipse. As a

second example, when a line is detected in the image, it is most likely from a model line. Although an ellipse, a hyperbola, or a parabola can also be projected onto a line, it is a very rare event. The probability of a detected feature originating from a certain model feature is used to order the sequence of nodes to be matched.

Table 1
Projection of Conic Sections

Model	Point	Line	Ellipse	Hyperbola	Parabola
Image	Point	Line* Point*	Ellipse Line*	Hyperbola line*	Ellipse Hyperbola Parabola Line*

Note. The object containing the conic sections is assumed to be in front of the camera. The image primitives marked "*" are the degenerate cases which seldom occur.

The second type of constraint states that if there is a set of correspondences between detected features and an object's model features, then the solution of the object's orientation and position must be consistent among all of the correspondences. Such kind of transformation constraints is called "arc' consistent constraints because they provide the restriction among feasible nodes at different levels in the search tree. The transformation constraints are used to prune the tree and perform the verification of correspondences.

Possible transformation constraints are inequality equations describing depth, viewing direction, viewing window, and attributes of matched features such as slope of line and centre of ellipse, etc.

The transformation constraints, when the orientation matrix is expressed in terms of quaternions, are quadratic or linear inequality equations of quaternion **q** and position **p**. A closed-form solution for propagating quadrantic constraints is derived by first transforming quadratic constraints into linear constraints and then propogating the linear constraints over **q** and **p**.

In using these two types of constraints, the bottom-up matching process is divided into two stages. First, selecting a detected feature which has rich information in identifying and locating its originating object. Second, matching feasible model features to the given detected feature. The feasible model features are those features that satisfy the projection rules and transformation constraints. The feasible model features are ordered in sequences and tested according to that order.

Selecting a Detected Feature to be Matched. Selection of which detected feature to be matched is an important task in the correspondence process. A good selection can reduce substantial matching effort. For instance, a correspondence between detected and model ellipses (or hyperbolas) reduces the sets of solutions of **O** and **p** to only two possibilities (see Appendix for the algorithm). Thus, an additional correspondence can provide the exact set of solutions of **O** and **p**. As a second example, two intersecting lines not only are easily detectable but also provide additional measurements of the intersection point to their own measurements. Finally, the fact that collinearity, parallelism, and proximity are preserved in most instances of projection (Lowe, 1987) can also be used to select imaged features in bottom-up matching.

Finding feasible Model Features. Although the use of projection rules in finding and ordering feasible model feaures is straightforward, the application of transformation constraints require some more explanation. Suppose more than one (detected) feature is labelled as a model feature of an object. Algebraic constraints from these correspondences on quaternion **q** and position **p** of the object are propagated to obtain refined intervals for components in **q** and **p**. Suppose now a new detected feature is given and to be labelled as one of the model features of the object. A feasible model feature has to abide by the constraints from the intervals for **q** and **p**. Therefore, a model feature is feasible if the new intervals for **q** and **p**, after propagating the algebraic constraints of the tentative match over the existing intervals, are not empty. The experimental match between a candidate model feature and the given detected feature is called a tentative match: it is used to test if the candidate is feasible in labelling the given detected feature.

Pairing Detected Features to a Given Model Feature. Although the bottom-up matching is sufficient in establishing correspondences between detected and model features, given appropriate projection rules and algebraic constraints, it is not efficient in establishing significant correspondences. For instance, the locator mentioned above required at least three correspondences between imaged lines and noncollinear model

lines. If we have established two correspondences, it is more efficient to use a top-down approach by finding a model line that is not collinear to the existing two, than to use a bottom-up approach. In fact, the top-down matching is used to add the most needed information to the aquired information from existing correspondences. In the top-down correspondence, the problem is equivalent to labeling a given model feature as one of the detected features. The time complexity, in the worst case, is L^{NM}. Like bottom-up correspondence, there are two steps in top-down matching.

Selecting a Model Feature. The prefered model features are those that are non-collinear to previously matched model features, that have intersection with previously matched model features, that have distinct information or contribution, and that require little matching effort.

Finding Feasible Detected Features. The projection rules and transformation constraints used in the top-down matching are the inverse of those in the bottom-up approach. Again, the projection rules are symbolic and trivial. In using transformation constraints, expressions which correspond to measurements in the image are evaluated over the existing intervals for **q** and **p** to have the expressions' upper bound and lower bound. The feasible detected features are those that have measurements within their corresponding expressions' upper and lower bounds.

Search strategy and time complexity of the recognizer

Backtracking. After using projection rules and transformation constraints to prune subtrees and to order the candidate subtrees, the recognizer uses backtracking search to facilitate a set of consistent correspondences between detected and model features. The backtracking is carried out in depth-first fashion. Backtracking is invoked when an inconsistency occurs. An inconsistency occurs if the correspondences provide an inconsistent solution for **q** and **p**. Once an inconsistency is detected, the recognizer backs up to the current node's direct parent node (multipe level backtracking is possible when perceptual groupings such as parallelism, collinearity, and proximity are used) in the search tree , etc. If subtrees of the node are all tried without success, the recognizer backs up one more level. This is repeated until a set of consistent correspondences is found or all nodes in the tree have been searched. All possible interpretations can be found if the termination is deferred until all nodes are searched. Clearly, the best correspondence is obtained by comparing all possible interpretations.

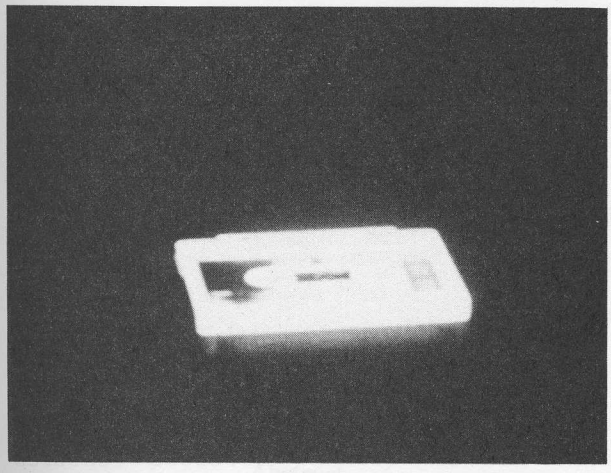
Shadowing can present particular problems, in particular, producing a feature which may not have a model counterpart. To account for this situation, a null node is added to each level of the tree in bottom-up matching. If a given detected feature has no counterpart in the model, it is labeled as the null node. Similarly, because of occlusion, a given model feature may not have corresponding detected features. Therefore, null nodes are also used in top-down pairing (see Grimson & Thomas, 1985).

Time Complexity. Since the locator needs three non-collinear line correspondences to locate an object, that the significant detected features are matched first, and since top-down matching is used to pick up the feature mostly needed, the search rarely goes down to the fifth level of search tree. If projection rules and transformation constraints successfully prune each level to N_{pruned} nodes, the time complexity is then $(N_{pruned})^5$. It is believed that N_{pruned} is much smaller than NM.

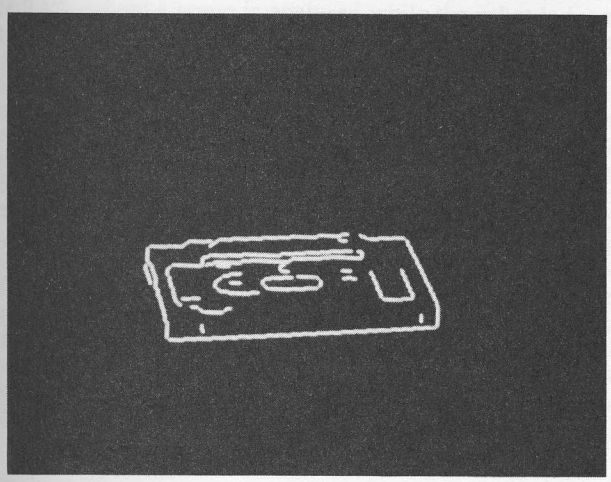
Multiple instances of the same object model in the image. In the case of multiple instances of the same object model in the image, a search tree is created for each instance of the object. For example, suppose we have two detected image features belonging to two different instances of a same object model. After we model the first image feature as one node in the first instance's search tree, the second image feature, if selected to be matched, will be rejected in the second level of the first instance's search tree and will be labelled as a node in the first level of the second instance's search tree.

3. Experimental Results

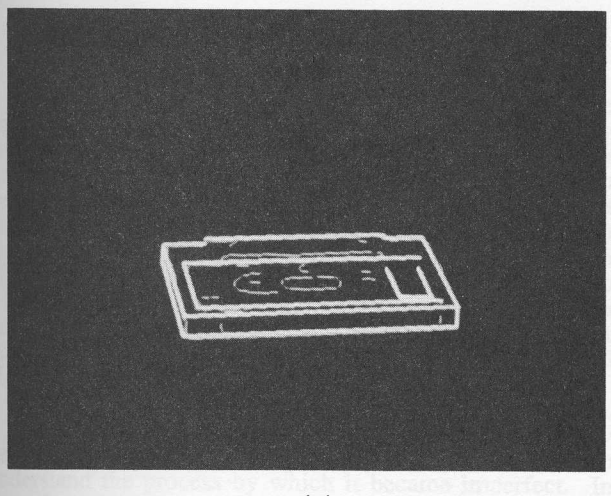
We have implemented these processes using only line features. Figure 1 shows an example of an interpretation. First, the intensity image (Fig. 2(a)) is convolved with a Sobel operator to obtain an edge image. Edge images are then traced and broken into line segments (Fig. 1(b)) using a subdivision algorithm. The line segments are then matched to a wireframe object model. Figure 1(c) shows the superimposition of the wireframe model and the image line segments after recognition and localization are successfully completed. All the processes of this implementation were done by using an IBM AT with an ITI Series 100 board. The interpretation process, in most cases, takes less than 60 sec.



(a)



(b)



(c)

Figure 1. (a)Intensity image; (b) line segments; (c) Superimposition of the Projected wireframe model on image line segments after successful interpretation.

4. Discussion.

The ability to determine object location during interpretation not only constrains the search space for a set of consistent matches but also tells the machine where to look for what kind of information. The closed-form solutions of using the minimum number of correspondences (e.g., three lines or one ellipse) to located the object enable us to determine the location parameters whenever a set of hypothesized matches is established and to test the validity of the hypothesis immediately with little computational effort.

Clearly, our emphasis on recovering object location does not suggest that symbolic relations among object parts is not important. It appears to us that a good use of both location information and symbolic relations is a promising direction of research for model-based vision.

Acknowledgments

This project was supported by Grants #A2568 and #5629 from the Natural Science and Engineering Research Council of Canada.

Appendix
Locating an ellipse in space

Let the model ellipse be the intersection of $x^2/\alpha^2+y^2/\beta^2=1$ and $z=0$, and the orientation and position of the model relative to the camera coordinates be **O** and **p**. Now, if the corresponding image ellipse is $[XY1]M[XY1]^T=0$, then we have the following procedure to calculate the unknown parameters **O** and **p**.

1. Calculate eigenvalues of M as λ_1, λ_2 and λ_3, and the corresponding eigenvectors of M as e_1, e_2, and e_3.
2. Let $u=e_1 \times e_2$, $v=e_3$, and $w=(e_2.e_3)e_1 - (e_2.e_1)e_3$.
3. If the third element of **u** is greater than zero, let **u**=-**u**.
4. If **u**.**v**>0, let **v**=-**v**.
5. Let $w=w/|w|$.
6. Let $\rho=(\beta/|wxu|)(-\lambda_2/\lambda_3)$.
7. $p=\rho u$ and $O=[w,vxw,v]$ or $[-w,-vxw,v]$.

References

Bolles, R.C. and Horaud, P. (1984). 3DPO: A Three Dimensional Part Orientation System. The International Journal of Robotics Research, 5(3).

Faugeras, O. D., and Hebert, M. (1986) The Representation, Recognition, and Locating of 3-D Objects. The International Journal of Robotics Research, 5(3), 27-52.

Grimson, W.E.L. and Thomas, L.-P. (1984). Model-based Recognition and Localization from sparse range or Tactile data. In A. Rosenfeld (ED.), Techniques for 3-D Machine Perception (pp.113-148). Elsevier Science Publishers B.V. (North-Holland), 1986.

Grimson, W.E.L. and Thomas, L.-P. (1985). Recognition and Localization of overlapping parts from sparse range. MIT A.I. Memo 841.

Lowe, D.G. (1987). Three-Dimensional Object Recognition from Single Two-Dimensional Images. Artificial Intelligence 31. 355-395.

Pentland, A.P. (1986). Perceptual Organization and the Representation of Natural Form. Artificial Intelligence 28, 293-331.

Wu, J. (1988). IRLM: An imaging system for object recognition, localization and motion recovery. Ph.D. dissertation, Department of Electrical Engineering, University of Alberta.

Wu, J., Rink, R., Caelli, T.M., & Gourishankar, V. (1988). Recovery of 3D locations and motions of rigid objects through Camera Imaging (Extended Kalman filter approach), International Journal of Computer Vision, (in press).