Active Object Recognition

David Wilkes and John K. Tsotsos
Department of Computer Science
University of Toronto
Toronto, Canada, M5S 1A4
E-mail: wilkes@vis.toronto.edu

Abstract

The concept of active object recognition is introduced and a proposal for its solution is described. The camera is mounted on the end of a robot arm on a mobile base. The system exploits the mobility of the camera by using low-level image data to drive the camera to a standard viewpoint with respect to an unknown object. From such a viewpoint, the object recognition task is reduced to a two-dimensional pattern recognition problem. The system uses an efficient tree-based, probabilistic indexing scheme to find the model object that is likely to have generated the observed data, and for line tracking uses a modification of the token-based tracking scheme of Crowley et al. [6]. We have successfully tested the system on a set of 8 origami objects. Given sufficiently accurate low-level data, recognition time is expected to grow only logarithmically with the number of objects stored.

1 Introduction

Bajscy [1] introduced the concept of active perception, that the sensor's state parameters may be purposefully changed according to sensing strategies. We propose to apply this concept to the task of object recognition. Recognition is the task of assigning names, given image data collected during some period of time, to objects that have appeared in the data.

The difficulty of the recognition task comes largely from the many sources of variation in the appearance of objects in image data. Among the sources of variation are changes in illumination, changes in imaging geometry, and changes in the relative positions of other objects surrounding the object. Other objects affect the degree of occlusion of the object of interest, and also its distinguishability from the background.

Active object recognition deals with these difficulties by dividing the recognition problem into two subtasks, with the claim that each of the subtasks is simpler than the original task.

The first subtask deals with changes in illumination and imaging geometry by moving the camera and primary light source to a position called a *standard view* of the unknown object. The motion to a standard view is driven by low-level image data. In our current implementation, a standard view is defined to be a position at which the lengths of two non-parallel object lines are maximized, and the longer of the lines has a specified length in the image. Moving to such a viewpoint has much in common with robotic docking maneuvers that maintain a certain bearing with respect to an object, based on low-level image data (Wunsche, as reported by Dickmanns and Graefe [7]). Also related is the interesting work in 2D-data-driven control by Zheng et al. [15].

Given a successful strategy for moving the camera to a standard view, we are left with a two-dimensional recognition problem, with remaining variation in object appearance due to two main factors.

The first factor is the relative position of other objects, including the background. This affects the detectability of various object features, causing certain object features to be missed, or grossly distorted (as is the case with a partially occluded line). Features that in fact belong to other objects may be erroneously grouped with features of the desired object, resulting in the apparent "invention" of new features. In our current experimentation we have attempted to minimize the complexity of the setting of the objects, in order to demonstrate the principles behind the system in a reasonable amount of time.

The second factor is error in the acquisition of a standard view. This can be due to error in both camera and light source positioning, and also the presence of secondary, uncontrolled light sources. Positioning errors result primarily in minor perturbations of feature parameter values. Lighting variations can result in missing features.

In order to deal with these remaining sources of variation, we present a method of storing and probabilistically retrieving objects based on *noisy sets* of features. By a noisy set, we mean a set that is subject to two types of variation. The first type, *membership variation*, is the addition or deletion of some elements. The second type of variation, *element variation*, is the perturbation of parameters of a particular element.

Lamdan and Wolfson [11] have shown how to express geometric properties of objects in object-based coordinate systems, which are invariant to changes in the relationship between the object in question and its environment. Thus, we may express properties of our object set as vectors (in a parameter space) that have

0-8186-2855-3/92 $3.00 ©1992 IEEE

such invariance.

For example, in our current implementation, we encode tha positions of line segments in the image as four-tuples giving the position (x, y) of the centre of each segment, and length and orientation (dx, dy) of each segment. These are expressed in coordinate systems defined with the x-axis interval [-1, 1] coincident with one of the lines used to define the current standard view. There are two such systems, depending on which direction is defined as the positive y-axis.

We store each object as the set of four-dimensional feature vectors corresponding to the lines visible from the current standard view. For simplicity, we consider different standard views of the same object to be different objects. Empirically, two to six standard views per object have been necessary.

The problem of storing and retrieving sets of object features has been addressed by many authors ([11], [13], [2], [5], [4], [3]). We have restricted ourselves to relatively uncluttered scenes, to avoid the complexity difficulties encountered when grouping features belonging to a single object from a large set of visible features [9]. Our retrieval method is a probabilistic algorithm based on a pruned search of a conventional data structure for the model that gives the highest probability of occurrence for the observed data. It provides for a significant reduction in retrieval cost over other methods, while still providing robustness to missing or extra features.

In the next section of the paper we give an overview of our system. Section 3 describes the algorithms used to move to a standard view. Section 4 describes the indexing algorithms. Section 5 presents our experimental results to date, and the final section summarizes the paper and describes areas of ongoing work.

2 Overview

We will describe our approach at two levels of abstraction. The first corresponds to the ideas behind the approach and the second corresponds to the actual implementation.

Figure 1 illustrates the higher-level view. Starting with image data, a low level image processing module extracts some sort of multi-parameter features, chosen to be appropriate for the application domain. A set of conditions on the features is chosen that uniquely defines a single position of the camera with respect to the object. The system drives either the camera or the object or both to positions that achieve this viewpoint for the camera. For small, movable objects, it might be appropriate to move the object rather than the camera. From the standard position, the object is recognized by matching the feature set extracted from the image at this position with stored feature sets, using some indexing scheme.

Figure 2 illustrates the implementation. Our low

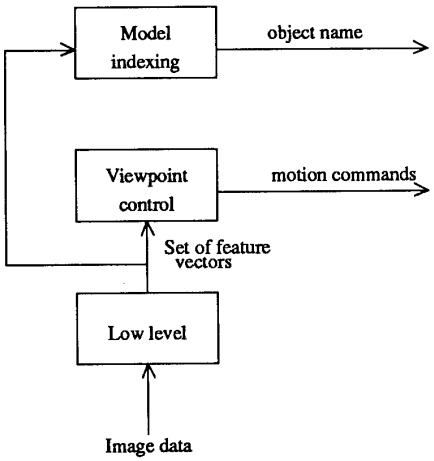


Figure 1: High-level view of the system

level image processing module extracts four-parameter line segments from the image data, expressed in coordinates with respect to the most prominent line in the image. We choose to define a standard view as a position at which each of two prominent, non-parallel line segments has maximal length, over all possible viewpoints at a given distance. The desired distance is defined as that at which the longer line segment has a certain specified length in the image (e.g. $\frac{\pi}{6}$ radians). The standard view is achieved by moving the camera on the end of a robot arm on a mobile platform. From a standard viewing position, lookup of the extracted line segments is done in our index tree, to find a matching stored object.

The low level module of our system is not a new research contribution, but we will briefly describe it for completeness. Figure 3 illustrates stages in the line segment extraction. We begin by extracting pixels that are of high rank in their neighbourhoods, in order of response magnitude to any of several oriented edge operators. The extracted pixels are then fit with "active" line segments whose parameter values optimize a goodness of fit measure.

As the camera is moved, frames are only grabbed once for every four centimetres of motion. This strategy requires robust tracking of object primitives. Crowley et al. [6] proposed a tracking scheme that maintains a model of each primitive that exists over multiple images, as long as sufficient support for the model exists in the image set. We have adopted such a scheme, but with primitives described in primitive-centred coordinates. Each token in the model describes the corresponding primitive by its context, that is, the set of surrounding primitives in a coordinate system described by the current primitive.

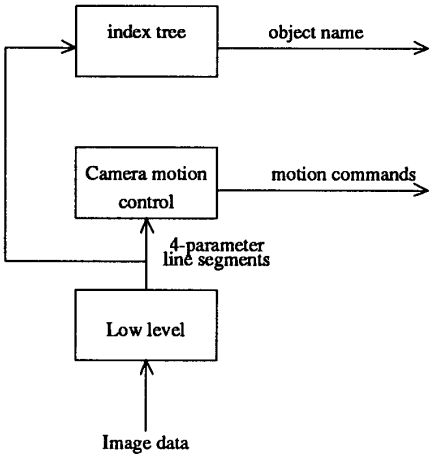


Figure 2: The implementation

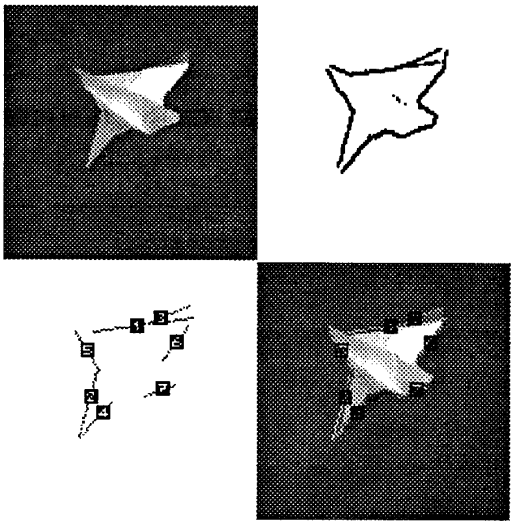


Figure 3: Line segment extraction. Upper left: An input image. Upper right: Pixels with locally maximal response to one of several oriented edge detectors. Lower left: Segments fit to upper right image. Lower right: segments superimposed on input.

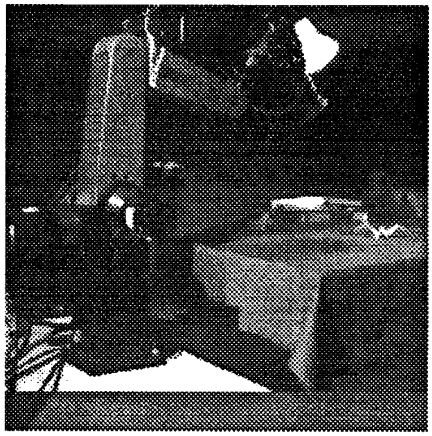


Figure 4: The robot and camera

3 Motion to a standard view

Figure 4 shows our CRSPlus five-degree-of-freedom robot arm mounted on a TRS Labmate base. The arm has a revolute waist, prismatic shoulder, elbow and wrist, with a second revolute joint just before the gripper. We mount a camera and light in the gripper. By mounting the principal source of illumination with the camera, shadow lines are minimized. By solving the inverse kinematics of the arm, we are able to specify the 5 joint angles necessary to bring the camera to a specified position in the base coordinates of the robot, with a specified aiming direction. Note that in order to achieve a particular position and aiming direction, the camera may take on an arbitrary rotation about the optical axis of the lens. The base is used to recentre the arm in its workspace when a move of the arm would take it outside of its workspace, under program control.

Figure 5 gives a high level statement of the camera motion strategy. A prominent line in the image is chosen to be maximized. The current implementation simply uses line length as the measure of prominence. Alternative measures could be based on the degree of support for the line. Since our system is initially very conservative about accepting support points for the line segments, length serves well as a measure of the likelihood that the line will persist. We call the chosen line L_0. First, L_0 is centred in the image, then the camera is moved towards or away from the line to bring it to a manageable initial length. All subsequent motions keep the camera roughly on the surface of a sphere centred on L_0, at a fixed distance from its centre. Motion steps parallel to the image of L_0 are interleaved with steps that recentre L_0 and return the camera to the sphere surface. Up to sixteen samples of the line length at equally spaced angles on the sphere

1. Choose prominent line segment in image as L_0
2. Centre L_0 in image
3. Move towards or away from centre of L_0 to give it a standard length
4. Choose a direction of motion parallel to L_0 that increases its length
5. Collect 16 L_0 length samples by repeating:
   a. Move parallel to image of L_0
   b. Recentre L_0 in image
   c. Move foreward to maintain constant distance from centre of L_0
6. Fit quadratic to length samples to find position from which L_0 has maximum length
7. Move to position of maximum length
8. Choose prominent line segment in image that is not parallel to L_1 and which has an endpoint close to an endpoint of L_0
9. Choose a direction of motion perpendicular to L_0 that increases the length of L_1
10. Collect 16 L_1 length samples by repeating:
    a. Move perpendicular to image of L_0
    b. Recentre L_0 in image
    c. Move forward to maintain constant distance from centre of L_0
11. Fit quadratic to length samples to find position from which L_1 has maximum length
12. Move to position of maximum length

Figure 5: Camera motion strategy

are fit with a parabola to determine the position at which the line has maximal length. The variation in length is actually described by the cosine of the tilt angle, but a parabola provides an adequate approximation. The locus of points at which the chosen line has maximal length for a given viewing distance is the intersection of the plane that bisects the line and is perpendicular to the line with the sphere at the given distance. Thus, we may maintain the maximal length of L_0 by moving perpendicular to its image. The next stage of the camera motion thus interleaves such motions with the recentering steps and sphere-rejoining steps in order to get length samples for a second line, L_1. L_1 is chosen based on two criteria. It must not be parallel to L_0, and it should have an endpoint close to one belonging to L_0, to increase the likelihood that the two lines belong to the same object. A position on the sphere at which L_1 has maximal length is found using another fit of a parabola to length measurements. The final result is a camera position on the sphere jointly maximizing the lengths of L_0 and L_1.

There are a number of potential sources of difficulty for the camera motion to a standard view. The camera motion strategy may occasionally fail due to a missed correspondence from one frame to the next, due to a reduction in the detectability of one of L_0 or L_1. In such a case, the system currently halts. Ideally, it would simply restart the search with a new prominent line. ALso, care must be taken to make sure that the motion will not attempt to drive the camera through any solid obstacles. Our solution to this problem is to place study objects near the edge of a tabletop, so that the arm is free to move below table height if necessary. Solutions that detected the collision, and then did something sensible, would be preferable.

4 Indexing algorithms

The indexing algorithm used in our recognition system is part of an ongoing investigation into methods for efficient retrieval of noisy sets of feature vectors. The recognition experiments described in the next section use few enough objects that it would suffice to compare the set of feature vectors extracted during recognition with each stored model set, without attempting to reduce the number of models examined. In an effort to ensure that the system is scalable, however, it is worthwhile to explore ways of limiting the number of models, that need to be compared with the data, by some means.

Our general approach to indexing is to replace the set of feature vectors in the data with an *indicator vector* of some sort, that encodes the feature vectors present in the data using one or more elements of a single vector.

The next step is to store indicator vectors for each model feature vector set in a conventional search tree of some sort.

In order to find the model indicator vector that best fits the image data, a probabilistically pruned, depth-first search is conducted in the tree. Each node visited in the search represents a hypothesis that one of the models below the node gave rise to the observed data, given some error process. If $\mathbf{v}$ is the query indicator vector extracted from the image data, then the algorithm seeks to find the stored vector $\mathbf{u}$ that maximizes the probability $p(\mathbf{v}|\mathbf{u})$ of occurrence of the observed data, given a certain error process and the current hypothesized model set. The search is pruned wherever a node at depth k is encountered such that $p(\mathbf{v}|\mathbf{u})$ is negligable, based on the first k elements of $\mathbf{v}$ and possible $\mathbf{u}$'s with the first k elements implied by the path to the node.

More detailed description and analysis of the indexing scheme will appear in a forthcoming paper.

Figure 6 shows the particular version of this approach that is applied in our active vision system. In order to map sets of feature vectors to single indicator vectors, we discretize our four-dimensional feature space coarsely, giving each cell in the space a sequence number using an ordinary array storage mapping [8]. The cell size is chosen according to the magnitude of

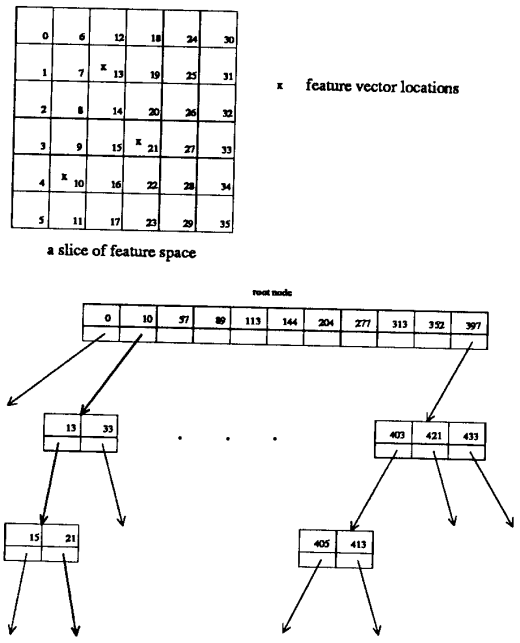


Figure 6: The index

the expected measurement error, so that feature vectors extracted from a given object have a reasonable probability (but not certainty) of falling within the same cell in repeated trials.

A binary indicator vector is constructed from a set of feature vectors by using one element in the indicator vector for each cell in the feature space. The element corresponding to a cell contains a one if the cell contains a feature vector, and a zero otherwise. This process maps all variation in a feature vector into an increased probability of apparent membership variation. That is, of invention of a one in the indicator vector where there had been a zero, and the miss of a one where a one had been.

Indicator vectors are extracted and stored for each likely standard view, for each object in the training set. A positional tree [8] is used, that stores each vector by listing (in numerical order) the positions at which ones occur in the vector. This tree is appropriate because the vectors are sparse. If the vectors had roughly equal numbers of zeros and ones, a *binary trie* [8] would be appropriate.

In order to evaluate this approach on a problem with a large number of models, a body of synthetic data was constructed. This consisted of 10000 randomly-constructed 4096-bit indicator vectors. These vectors had an average of 16 one bits in a field of zeros, representing 16 feature vectors for each model.

Simulated query vectors were generated by perturbing randomly chosen stored vectors at certain bit-invention and bit-miss rates. At error rates that caused an average of four spurious ones and four missed ones in the query vector, the correct stored vector was retrieved roughly 97 percent of the time, with no vector returned in the remaining trials. This required that an average of 24688 nodes per query be examined, or roughly 15 percent of the tree.

5 Experimental results

To date, our experiments have focussed on a set of eight origami objects, as shown in figure 7. For each object, about four feature sets are stored, using different standard views and/or coordinate bases. In a recognition trial, each object in the set is presented to the system twice, at a randomly chosen stable orientation, in an uncluttered setting. Below we describe results from a recognition trial that is representative of our current typical performance. Good overall performance of the system requires reasonable performance from both the view-finding module and the indexing module. Although both are still in the early stages of experimentation, they have performed reasonably well.

With an index tree containing 34 feature sets, with a total of 270 features, and a coarse division of ten cells along each of the four dimensions, we have 270 10000-bit indicator vectors stored.

The view-finding module found a standard view to within about 10 degrees of arc on fourteen of sixteen trials. In one of the trials, the problem was that the edge chosen as the most prominent became partially occluded by another edge of the same object. In another of the trials, the low-level system lost track of a key edge partway through execution. Figure 8 shows the initial and final camera positions in one trial, and a plot of the camera motions as seen from above.

The indexing module also performed well. In spite of the fact that the models tend to get flattened in storage, there was correct retrieval in all but two cases. In both of these cases, the coordinate basis used to retrieve the object was not one for which a feature set had been stored.

6 Conclusions and further work

We have presented an approach to object recognition that simplifies the 3D recognition problem by solving two easier problems. Tests so far have been successful as a proof of concept. There are still some sources of error that we believe could be reduced or eliminated. Also, we would like to perform trials and enhancement in a more cluttered environment, and with larger object sets. More interesting object sets

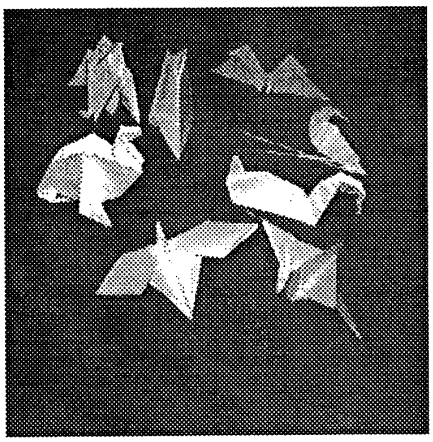
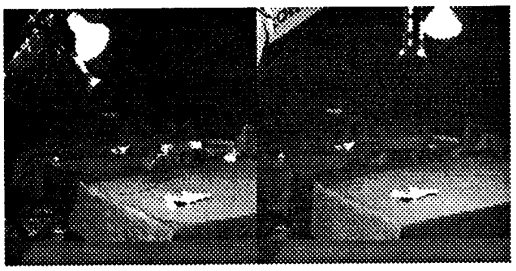


Figure 7: The object set.



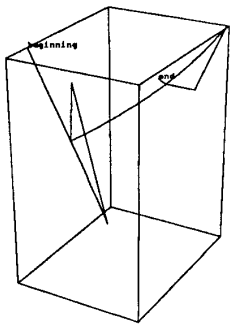


Figure 8: Initial and final camera positions; camera motion

could be attempted with a more sophisticated low-level system. Additional analysis and larger-scale simulation tests of the indexing algorithms are under way.

Acknowledgements The authors wish to thank the Natural Sciences and Engineering Research Council of Canada, the Information Technology Research Centre, and the Canadian Institute for Advanced Research for financial support. We would also like to thank the two anonymous reviewers for their comments.

References

[1] Bajcsy, R., "Active Perception vs Passive Perception," *Proc. IEEE Workshop on Computer Vision: Representation and Control*, Bellaire, Michigan, 1985.

[2] Bolle, R.M., Califano, A., Kjeldson, R., Mohan, R., "Active 3D Object Models," *Proc. Third International Conference on Computer Vision*, IEEE Computer Society Press, Washington, 1990.

[3] Breuel, T.M., "Model Based Recognition using Pruned Correspondence Search," *Proc. Computer Vision and Pattern Recognition '91*, IEEE Computer Society Press, Washington, 1991.

[4] Califano, A., and Mohan, R., "Multidimensional Indexing for Recognizing Visual Shapes," *Proc. Computer Vision and Pattern Recognition '91*, IEEE Computer Society Press, Washington, 1991.

[5] Clemens, D.T. and Jacobs, D.W., "Model Group Indexing for Recognition," *Proc. Computer Vision and Pattern Recognition '91*, IEEE Computer Society Press, Washington, 1991.

[6] Crowley, J.L., Stelmaszyk, P. and Discours, C., "Measuring Image Flow by Tracking Edge-Lines," *Proc. Second International Conference on Computer Vision*, IEEE Computer Society Press, Washignton, 1988.

[7] Dickmanns, E.D., and Graefe, V., "Dynamic Monocular Machine Vision and Applications of Dynamic Monocular Machine Vision," *Tech. Report UniBwM/LRT/WE 13/FB/88-3*, Institut fur Meβtechnik, Universitat der Bundeswehr Munchen, July 1988.

[8] Gotlieb, C.C., and Gotlieb, L.R., *Data Types and Structures* Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[9] Grimson, W.E.L., and Huttenlocher, D.P., "On the Sensitivity of Geometric Hashing," *Proc. Third International Conference on Computer Vision*, IEEE Computer Society Press, Washington, 1990.

[10] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active Contour Models," *Proc. First International Conference on Computer Vision*, IEEE Computer Society Press, Washington, 1987.

[11] Lamdan, Y., and Wolfson, H.J., "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second International Conference on Computer Vision*, IEEE Computer Society Press, Washington 1988.

[12] Marr, D., "A Theory of Cerebellar Cortex," *Journal of Physiology*, Vol. 202, pp.437-470, 1969.

[13] Stein, F., and Medioni, G., "Efficient Two Dimensional Object Recognition," *Proc. Tenth International Conf. on Pattern Recognition* IEEE Computer Society Press, Washington, 1990.

[14] Tsotsos, J.K., "A 'Complexity Level' Analysis of Vision," *Proc. First International Conference on Computer Vision*, IEEE Computer Society Press, Washington, 1987.

[15] Zheng, J.Y., Chen, Q., Kishino, F., and Tsuji, S., "Active Camera Controlling for Manipulation," *Proc. Computer Vision and Pattern Recognition '91*, IEEE Computer Society Press, Washington, 1991.