ADORE: Adaptive Object Recognition

Bruce A. Draper, Jose Bins, Kyungim Baek

Department of Computer Science
Colorado State University
Fort Collins, CO, USA. 80523
(draper|bins|baek)@cs.colostate.edu

Abstract. Many modern computer vision systems are built by chaining together standard vision procedures, often in graphical programming environments such as Khoros, CVIPtools or IUE. Typically, these procedures are selected and sequenced by an *ad-hoc* combination of programmer's intuition and trial-and-error. This paper presents a theoretically sound method for constructing object recognition strategies by casting object recognition as a Markov Decision Problem (MDP). The result is a system called ADORE (*Ad*aptive *O*bject *Re*cognition) that automatically learns object recognition control policies from training data. Experimental results are presented in which ADORE is trained to recognize five types of houses in aerial images, and where its performance can be (and is) compared to optimal.

1 Introduction

As the field of computer vision matures, fewer and fewer vision systems are built "from scratch". Instead, computer vision systems are built by chaining together standard vision procedures, including (but by no means limited to): image smoothing and enhancement, edge and corner extraction, region segmentation, straight line and curve extraction, grouping, symbolic model matching (including Hausdorf matching, key feature matching, and heuristic search), appearance matching, pose determination, and depth from stereo, motion or focus. Separately, each of these procedures addresses part of the computer vision problem. Sequenced together, they form end-to-end vision systems that perform specific tasks.

Computer vision software environments help users build end-to-end systems by providing procedure libraries and graphical user interfaces for sequencing them together. In Khoros, for example, programmers build applications by selecting procedures (called "glyphs") from menus and graphically connecting the output of one procedure to the input of another [27]. CVIPtools is a similar software environment intended primarily for academic use [33]. The Image Understanding Environment (IUE) is an object-oriented software developer's environment, but it also includes a GUI for sequencing procedures [23]. (The IUE is still under development at the time of writing, but a preliminary version is freely available at http://www.aai.com/AAI/IUE/IUE.html.)

Software tools such as Khoros, CVIPtools and IUE make it easier for programmers to form and test sequences of vision procedures. Unfortunately, they do not help programmers with the underlying problems of how to select procedures for a specific task, or how to compare one control strategy to another. Programmers are left to choose vision procedures based on intuition, and to refine sequences of procedures by trial and error.

The goal of the Adaptive Object Recognition (ADORE) project is to provide a theoretically sound mechanism for dynamically selecting vision procedures for specific tasks based on the current state of the interpretation. In the future, we hope to build systems that can adapt to any recognition task by dynamically selecting actions from among dozens (if not hundreds) of vision procedures. At the moment, however, this ambitious goal exceeds our grasp. This paper describes an initial prototype of ADORE that learns to find houses in aerial images using a library of ten vision procedures. While this system is clearly short of our ultimate goal, it is an example of an end-to-end system that adapts by dynamically controlling vision procedures. This paper describes two experiments with the prototype version of ADORE – one where ADORE succeeds in finding a nearly optimal recognition strategy, and one where it is less successful.

2 Examples of Static and Dynamic Control

Before describing ADORE in detail, let us first illustrate the problem it is supposed to solve. Figure 1 shows a nadir-view aerial image. The task is to find instances of specific styles of houses, such as the duplex in Figure 2. To accomplish this task, ADORE is given access to ten vision procedures and a template of the duplex. (Descriptions of all ten procedures can be found in Section 5.2.) ADORE is also given training images and training signals that give the position and orientation of each duplex. ADORE's role is to dynamically select and execute procedures so as to produce duplex (and only duplex) hypotheses.

ADORE finds duplexes by learning a control strategy that selects one vision procedure at each processing step. For example, ADORE can begin be selecting one of three procedures for producing regions of interest (ROIs) from images: a rotation-free correlation procedure [28], a statistical distribution test [25], and a probing routine. All three can be used to generate duplex hypotheses, but ADORE learns from the training data that for this task – where the duplexes are almost identical to each other, and lighting differences and perspective effects are minimal – pixel-level correlation outperforms the other two procedures. ADORE therefore learns a recognition strategy that begins with correlation.

The next step is more complex. Figure 3 shows three ROIs produced by correlation. The ROI on the left of Figure 3 matches the position and orientation of a duplex very well. In fact, none of the procedures in ADORE's procedure library can improve this hypothesis, so the best strategy for ADORE is to accept it. The ROI on the right in Figure 3, on the other hand, does not correspond to any duplex. The best strategy here is to reject it.

Fig. 1: A nadir-view image of a residential section of Ft. Hood, TX

The ROI in the middle of Figure 3, on the other hand, is more interesting. This ROI roughly matches a duplex, but the ROI is below and slightly rotated from the true position of the duplex, probably because the duplex is partially occluded in the image. In this case, the best strategy is to refine the hypothesis by resegmenting the image chip [10] and then applying a Generalized Hough Transform [5] to align the template with the extracted region boundary. Figure 4 shows the resulting hypothesis after these two procedures are applied.

Fig. 2. A Duplex

Examples like the one in Figure 3 demonstrate the importance of *dynamic control*. In all three cases, the first procedure was the same: correlation. The choice of the next procedure, however, depends on the quality of the data (in this case, ROI) produced by the previous step. In general, control strategies should choose procedures based not only on static properties of the object class and image domain, but also on properties of the data produced by previous procedures.

3 Related Work

Long before the appearance of software support tools like Khoros, researchers argued for specialized recognition strategies built from reusable low-level components. As far back as the 1970s, Arbib argued from psychological evidence for specialized visual "schemas" built from reusable procedures [4]. In the 1980's, Ullman

developed a similar theory, in which primitive "visual routines" are combined to form specialized recognition strategies [32]. Later, Aloimonos [2] and Ikeuchi & Hebert [16] argued for specialized recognition strategies made from primitive vision operations in the context of visual robotics.

Fig. 3. A demonstration of dynamic control: the three ROIs above were all created by rotation-free correlation, yet the best strategy for refining these hypotheses depends on the quality of the ROIs, not the history of how they were created. In this case, the best strategy is to 1) accept the ROI on the left, 2) reject the ROI to the right, and 3) refine the ROI in the middle through segmentation [10] followed by the Generalized Hough Transform [5].

In practice, researchers have been building systems with special-purpose recognition strategies for twenty years. In the late '70s and early '80s, researchers built AI-style production and blackboard systems that selected and sequenced vision procedures to achieve specific tasks. Nagao & Matsuyama's production system for aerial image interpretation [24] was one of the first, and lead to several long-term development efforts, including SPAM [22], VISIONS/SCHEMA [11], SIGMA [15], PSEIKI [3] and OCAPI [9]. More recently, other researchers [8,17,19] have applied AI-style planning technology to infer control decisions from databases describing the task and the available procedures.

Fig. 4. The middle ROI from Figure 3 after refinement via segmentation [10] and the Generalized Hough Transform [5].

Unfortunately, knowledge-based systems are often *ad-hoc*. Researchers formulate rules for selecting procedures based on their intuition, and refine these rules through trial and error. (See [12] for a description of the knowledge engineering process in object recognition) As a result, there is no reason to believe that the control policies

emerging from these heuristics are optimal or even good, nor is there any way to directly compare systems or to evaluate control policies.

Recently, researchers have tried to put the control of object recognition on a stronger theoretical foundation using Bayes nets (e.g. TEA1 [29] and SUCCESSOR [21]). Unfortunately, the design of Bayes nets can itself become an *ad-hoc* knowledge engineering process. Other researchers try to eliminate the knowledge acquisition bottleneck by learning control policies from examples. Researchers at Honeywell use genetic algorithms to learn target recognition strategies [1], while reinforcement learning has been used by Draper to learn sequences of procedures [13] and by Peng & Bhanu to learn parameters for vision procedures [26]. Maloof et. al. train classifiers to accept or reject data instances between steps of a static sequence of procedures [20].

4 Object Recognition as a Supervised Learning Task

The goal of the adaptive object recognition (ADORE) project is to avoid knowledge engineering by casting object recognition as a supervised learning task. Users train ADORE by providing training images and training signals, where a training signal gives the desired output for a training image. ADORE learns control strategies that dynamically select vision procedures in order to recreate the training signal as closely as possible. This control strategy can then be used to hypothesize new object instances in novel images.

To learn control strategies, ADORE models object recognition as a Markov decision problem. The state of the system is determined by data tokens produced by vision procedures. For example, the state of the system might be a region of interest (ROI), a set of 2D line segments, or a 2D contour. The actions are vision procedures that change the state of the system by producing new data tokens from the current data. A control policy is a function that maps states onto actions. In the context of ADORE, control policies map data tokens onto vision procedures, thereby selecting the next action in the recognition process.

At the software system level, ADORE is most easily thought of as two distinct components: a run-time execution monitor that applies vision procedures to data, and an off-line learning system. The connection between these two systems are the control policies that are developed by the learning system and applied by the execution monitor.

4.1 The Execution Monitor

The run-time execution monitor is a three-step loop that implements dynamic control policies. On each cycle, the execution monitor:

1. Measures properties of the current data token, producing a feature vector. The length and contents of the feature vector depend on the type of the data token; for example features measured of an image (average intensity, entropy, etc.) are different from features measured of a contour (length, curvature, contrast).

2. The control policy is a function that maps feature vectors onto vision procedures (see Section 4.2 below).
3. The selected procedure is applied to the current data token, thereby producing new data tokens.

The loop begins when an image is presented to the system as the first data token; the monitor then executes the loop above until a vision procedure fails to return any data, at which point the recognition process stops.

Of course, this simple description glosses over some important details. First, there are two special procedures – called *accept* and *reject* – that return no data, and therefore provide a stopping point. These are the procedures that interact with the user; the accept procedure signals that an object instance has been found, while the reject procedure indicates that the current data is a false hypothesis and should be abandoned. Second, the execution monitor also measures the run-time of each procedure, to be used in task statements (i.e. reward functions – see Section 4.2 below) that involve tradeoffs between time and accuracy. Finally, many vision procedures return multiple outputs. For example, the peak detection procedure (see Section 5.1) may detect several peaks corresponding to possible hypotheses. Similarly, many other detection, matching and grouping routine return multiple hypotheses. When this happens, we assume that the outputs do not overlap and consider each to be a separate hypothesis, forking as many new instances of the execution monitor as are needed to match the number of hypotheses.

In terms of software, the execution monitor is independent of the vision procedure library. Each vision procedure is an independent unix executable; a library file tells the execution monitor the number and type of input arguments for each procedure, the number and type of output arguments, and the unix pathname. The design goal is to allow vision procedures to be added or removed from the system by simply editing the library file. Similarly, the execution monitor is independent of particular data representations, since all data tokens are kept in files. For each data type, the library file tells the execution monitor 1) the name of the data type (so the monitor can match data tokens with arguments to vision procedures); 2) the length of the feature vector; and 3) the path of the unix executable for measuring features. Thus new data types, like new vision procedures, can easily be added to the system.

4.2 Control Policies

Control strategies are represented by *policies* that select vision procedures based on feature vectors. Since good control strategies depend on the target object class and image domain, a different strategy is learned for every object recognition task.

To learn theoretically justifiable control strategies, object recognition is modeled as a Markov Decision Problem (MDP). Although a general introduction to MDPs is beyond the scope of this article, they are structurally similar to finite state machines. The system begins in some a s_1 and applies an action a_1, thereby creating a transition to a new state s_2. This process repeats itself, creating a series of states and actions, s_1, a_1, s_2, a_2, s_3,…. Unlike a finite state machine, however, the state transitions in an MDP are probabilistic; when an action a_i is applied in state s_n, the resulting state is selected

from a probability distribution associated with the state/action pair $<s_n,a_i>$. Because every state/action pair has a different probability distribution, the system has to select which action to apply at each state. This selection is made by a *control policy*, which is a mapping of states onto actions. Finally, every state transition has a *reward* (a.k.a. *penalty*) associated with it. The goal in a Markov decision problem is to find the control policy that maximizes the expected reward over time.

In ADORE, object recognition is cast as a Markov decision problem by equating actions with computer vision procedures (e.g. edge detection, region segmentation, grouping). These procedures produce and consume intermediate data tokens such as images, regions and line groups. The state of the Markov process is determined by a feature vector that describes the current data token. Vision procedures are probabilistic because even though we know the type of data they produce – for example, edge detection procedures create edges -- we do not know in advance what feature values that resulting data will have.

The reward/penalty function used for object recognition is task-specific. If the goal is to optimize recognition regardless of cost then the reward associated with every procedure other than *accept* is zero. When the system invokes *accept* it signals that it has found an instance of the object class, and it is rewarded or penalized according to how well that hypothesis matches the training signal. (The error function used to compare hypotheses to the training signal is also task-specific.) If the goal is to optimize a cost/quality tradeoff, every procedure other than *accept* and *reject* is penalized according to its runtime.

In this framework, a control policy is a function that maps feature vectors onto actions. (This mapping is limited by the practical constraint that every vision procedure can be applied to only one type of data). The control policy is built up from a set of Q-functions, one for every vision procedure. In Markov control theory, Q(s,a) is the function that predicts the expected reward over time that follows from applying action *a* in state *s*. For example, in ADORE the off-line learning system trains a Q-function to predict the reward that results from segmenting ROIs (in the context of the current task), based on the features of the ROI. It also trains Q-functions for predicting the rewards that follow image correlation, curve matching, and every other procedure in the procedure library. The control policy evaluates these Q-functions on the current data and selects the procedure with the highest Q-value.

It is important to note that the Q-function predicts the total future reward that follows a procedure, not just the immediate reward. As described above, in most object recognition tasks the system does not get a positive reward until the final step when it accepts a (hopefully correct) hypothesis. As a result, Q-functions predict the quality of the hypothesis that eventually follows a procedure, even if it takes several additional steps to form that hypothesis.

4.3 Off-Line Learning

The control and artificial intelligence literatures contain many techniques for learning optimal Q-functions for control problems with discrete state spaces. If the transition probabilities associated with the actions are known (a so-called *process model*), dynamic programming will estimate Q-values and produce an optimal control policy.

In the absence of a process model, reinforcement learning (most notably the temporal difference [30] and Q-learning [34] algorithms) have been shown to converge to optimal policies in a finite number of steps.

Unfortunately, the object recognition problem as defined here depends on a continuous state space of feature vectors. Tesauro [31] and Zhang & Dietterich [35] have shown empirically that neural nets can approximate Q-functions for continuous feature spaces within a reinforcement learning system and still produce good control policies. Unfortunately, their method required hundreds of thousands of training cycles to converge. ADORE has a sequence of continuous feature spaces, one for each data representation (images, ROIs, contours, etc.) and would require getting a sequence of neural nets to converge on a single control policy. Although theoretically possible, we have not yet succeeded in making this work.

Instead, we train Q-functions by optimistically assuming that the best control policy always selects the action that yields the highest possible future reward for every data token. Strictly speaking, this assumption is not always true: a control policy maps points in feature space onto actions, and it is possible for two different tokens to have the same feature measurements and yet have different "optimal" actions. Nonetheless, the optimistic assumption is approximately true, and it breaks the dependence between Q-functions, allowing each neural net to be trained separately.

In particular, we approximate Q-functions by training backpropagation neural networks. The training samples are data tokens extracted from the training images. We apply all possible sequences of procedures to every training sample, in order to determine which procedure yields the maximum reward. A neural network Q-function is trained for every vision procedure using the data features as input and the maximum reward as the output. In this way, the neural net learns to approximate the future reward from an action under the optimistic control assumption. (Complicating the picture somewhat, we "bag" the neural nets to reduce variance; see [14].)

5 Experiments

5.1 Task #1: Duplex Recognition

To test ADORE in a tightly controlled domain, we trained it to recognize houses in aerial images like the one in Figure 1. In the first experiment, the goal is to find duplexes of the type shown in Figure 2. The training signal is a bitmap that shows the position and orientation of the duplexes in the training images; Figure 5 shows the training signal matching the image shown in Figure 1. The reward function is the size of the pixel-wise intersection of the hypothesis and the training signal, divided by the size of the union. This evaluation function ranges from one (perfect overlap) to zero (no overlap).

Fig. 5. The duplex training signal for the image shown in Figure 1.

5.2 The Vision Procedure Library

The vision procedure library contains ten 2D vision procedures, as depicted in Figure 6. Three of the procedures produce likelihood images (with orientation information) from intensity images and a template[1]. The rotation-free correlation procedure [28] correlates the template at each position in the image by first rotating the template until the direction of the edge closest to the center of the template corresponds to the edge direction at the center of the image window. The TAStat procedure is a modification of the algorithm in [25]. For every image window it also rotates a mask of the object until it aligns with the local edge data, and then measures the difference between the intensity distributions of the pixels inside and outside of the mask. The greater the difference between the intensity distributions, the more likely the mask matches an object at that location and orientation in the image. Finally, the probing procedure also uses edge information to rotate the template for each image window, and then samples pairs of pixels in the image window, looking for edges that match the location of edges in the template.

[1] In all of our experiments, we assume that a template of the object is available.

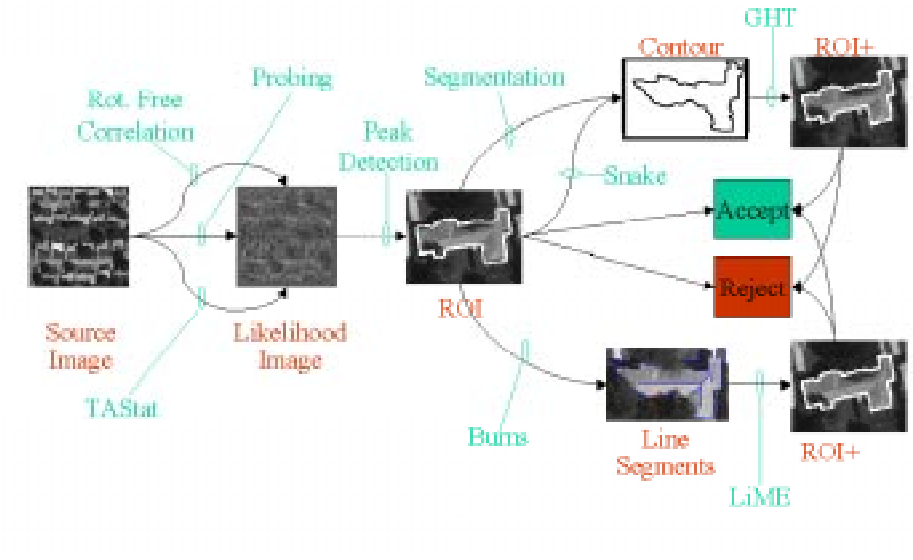


Fig. 6. A visual depiction of ADORE's vision procedure library. Note that the peak detection procedure produces approximately twenty ROIs each time it is called.

Regions of interest (ROIs) are chips from the original image that are hypothesized to correspond to object instances; each ROI also has a mask that details the hypothesized position and orientation of the object. ROIs can be extracted from likelihood images using a peak detection procedure, which finds the top N peaks in a likelihood image. For these experiments, the peak detection procedure was parameterized to extract twenty peaks from each likelihood image.

Five procedures can be applied to any ROI. Two of these actions are the special actions mentioned in Section 4.1, accept and reject. The other three options are: 1) an active contour procedure [18] that modifies the outline of an ROI mask until the contour lies along edges in the original image; 2) a segmentation procedure [10] that extracts the boundary of a new region (as a 2D contour) within the image chip; or 3) a straight line extraction procedure [7].

A Generalized Hough Transform procedure [5] matches 2D image contours to the contour of a template, thus creating a new ROI. A symbolic line matching procedure (LiME; [6]) finds the rotation, translation and scale that maps template (model) lines onto image lines, again producing an ROI. It should be noted that LiME transforms hypotheses in scale as well as rotation and translation, which puts it at a disadvantage in this fixed-scale domain.

5.3 Duplex Results

To test the system's ability to learn duplex recognition strategies, we performed N-fold cross-validation on the set of eight Fort Hood images. In other words, we divided the data into seven training images and one test image, trained ADORE on seven training images, and evaluated the resulting strategy on the test image. We repeated this process eight times, each time using a different image as the test image. All the results presented this paper are from evaluations of test images.

Figure 7 shows the results of two tests, with the ROIs extracted by ADORE outlined in white on top of the test image. As a crude measure of success, ADORE found 21 out of 22 duplexes, while producing 6 false positives. The only duplex not found by ADORE can be seen in the image on the right of Figure 7– it is the duplex that is half off the bottom right-hand corner of the image. Every duplex that lies completely inside an image was recognized. (The right side of Figure 7 also shows one false positive.)

Fig. 7. Duplexes extracted from two images. In the image on the left, all three duplexes were found. On the right image, a false positive appears on the upper right side. The half-visible duplex to the bottom right is the only false negative encountered during testing.

It would be incomplete to just analyzing ADORE in terms of false positives and false negatives, however. Much of the benefit of ADORE's dynamic strategies lies in their ability to refine imperfect hypotheses, not just make yes/no decisions. ADORE maximizes its reward function by creating the best hypotheses possible, given the procedure library. Table 1 gives a quantitative measure of ADORE's success. The left most entry in Table 1 gives the average reward across all 22 positive duplex instances from the optimal strategy, where the optimal strategy is determined by testing all sequences of procedures and taking the best result. The second entry gives the average reward generated by the strategy learned by ADORE. As further points of comparison, we implemented four static control strategies. The third entry in Table 1 gives the average reward for duplex instances if no further processing is applied

after correlation and peak detection. The fourth entry gives the average reward if every duplex ROI is segmented and then repositioned by matching the region boundary to the duplex template boundary via a Generalized Hough Transform. The fifth entry gives the average reward if the active contour (i.e. snake) procedure is applied to every ROI, followed once again by the Generalized Hough Transform. Finally, the sixth entry is the average reward if straight lines are extracted from ROIs, and then the LiME geometric model matcher is applied to determine the position and rotation of the duplex.

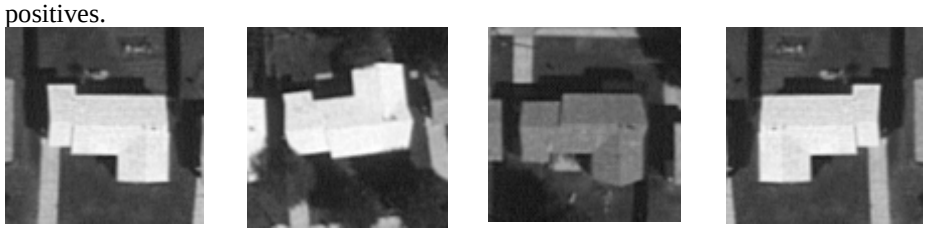
	Optimal Policy	ADORE Policy	Accept or Reject	Segment	Active Contours	Line Extract
Avg Reward	0.8991	0.8803	0.7893	0.8653	0.7775	0.1807

Table 1. Comparison between the optimal policy, the policy learned by ADORE, and the four best static policies.

Two conclusions can be drawn from Table 1. First, the strategy learned by ADORE for this (admittedly simple) task is within about 98% of optimal. Second, the dynamic strategy learned by ADORE, although not perfect, is better than any fixed sequence of actions. As a result, our intuitions from Section 2 are validated: we achieve better performance with dynamic, rather than static, control strategies.

5.3 Task #2: Finding Smaller Houses

Having succeeded in finding a good strategy for recognizing duplexes, we changed the training signals and templates to recognize the four other styles of houses shown in Figure 8. The same procedure library and token features were used for these houses as for the duplex. After training, ADORE identified 18 of 19 instances of the house style *A* but generated 22 false positives. Combining the results from house styles A through D, ADORE found 47 out of 61 instances, while generating 85 false positives.



(A) (B) (C) (D)

Fig. 8. Templates of four other styles of houses. Note that the boundaries of the templates are similar to each other

On the one hand, these results were encouraging. We had used the same procedure library and features to find five different classes of objects, by training five different control policies. On the other hand, these results were clearly not as good as the previous results with recognizing duplexes had been.

Why were these problems harder than finding duplexes? The most critical control decision (given our limited procedure library) occurs after ROIs are extracted from likelihood images. At this point, ADORE has a choice of five procedures: segmentation, active contours, line extraction, accept or reject. Of these five actions, line extraction is never optimal, but the other four actions are all optimal for some choice of ROI.

The control policy must select optimal actions based on ROI feature vectors. By inspecting the weights of the neural net Q-functions trained for duplex recognition, we discovered that two of the eleven ROI features dominated the control decision. One feature was the average edge strength along the boundary of the ROI. The second was the percent of pixels outside the mask that match the average intensity value of pixels under the mask. (We informally refer to these as "missing" pixels, since their intensities suggest that they were accidentally left out of the hypothesis.)

Based mostly on these two features, ADORE learned a strategy that worked well for duplexes. If we interpret the behavior of the Q-functions in terms of these two features, the duplex recognition strategy can be described as follows (see Figure 9): ROIs with very high boundary edge strength and very few "missing" pixels should be accepted as is. (The points in Figure 9 correspond to training samples, coded in terms of the optimal action for each ROI. Points marked with an open circle correspond to ROIs that receive approximately the same reward whether segmented or accepted as is.) If there is less edge strength but relatively few pixels are "missing", then the ROI should be segmented to adjust the boundary. Although many false hypotheses are segmented according to this rule, there is another control decision after segmentation where false hypotheses can be rejected and true ones accepted if the adjusted boundary still has low edge strength. There is also one small region in feature space where the active contour procedure is optimal. This is harder to explain and may result from the training set being a little too small, or it may be a decision triggered by some combination of the other nine features. Finally, if the missing pixel count is high or the edge strength is low, the ROI should be rejected. The solid boundaries in Figure 9 are our hand-drawn interpretation of the control policy's decision boundaries.

When we look at the other house recognition tasks, however, we find that the same features do not discriminate as well. If you overlay the four templates shown in Figure 8, it turns out that most of the boundaries are aligned. As a result, if an ROI for style A is incorrectly placed over an instance of styles B, C or D, the average edge strength is still very high. (The same is true for ROIs of style B, C and D). As a result, the edge strength feature does not discriminate between these styles of houses. Since every hypothesis must identify the *type* of house, ADORE has a hard time learning to distinguish true hypotheses from false ones, resulting in (gracefully) degraded performance. In effect, the difference *in feature space* between one style and another is too small to support a more reliably strategy. The policies learned by ADORE make the most of the feature set it is given and identify most instances

correctly, but to improve performance on this task will require new and better features.

6 Conclusion

We have built a prototype adaptive object recognition system capable of learning object-specific recognition strategies, given a procedure library and features that describe intermediate hypotheses. When the intermediate-level features are descriptive enough to support intelligent control decisions, the result is a near-optimal object recognition system. When the intermediate-level features are less descriptive, ADORE still learns the best control policy relative to these (weak) features, but the resulting performance is naturally degraded.

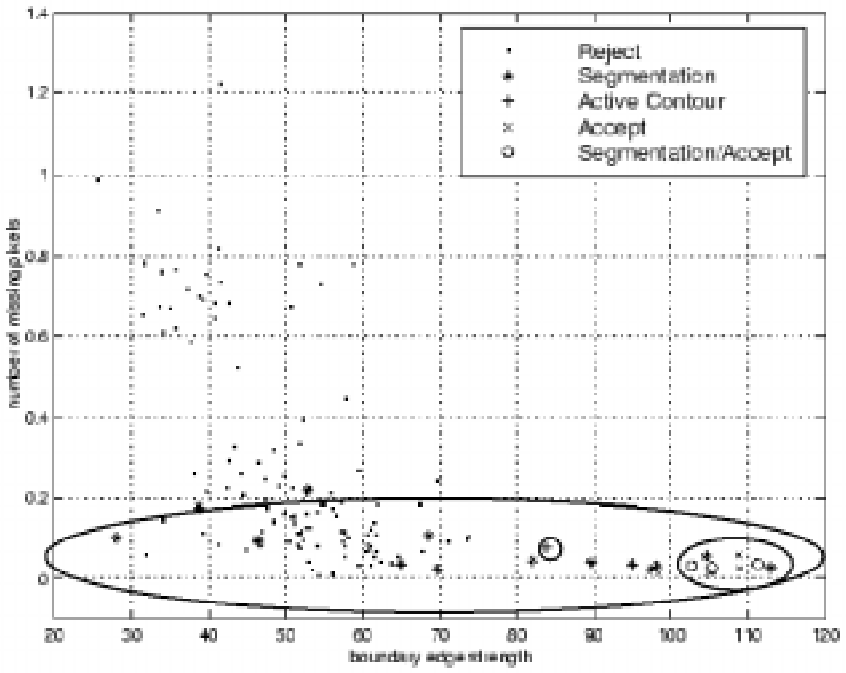


Fig. 9. ROIs plotted in two dimensions of the eleven dimensional feature space. The shape of the points indicates the optimal control decision for each ROI. (Open circles represent ROIs that receive roughly the same reward whether they are accepted as is or refined through segmentation.) The ellipses correspond to our interpretation of the decision boundaries learned by the neural networks.

References

[1] W. Au and B. Roberts. “Adaptive Configuration and Control in an ATR System,” *IUW*, pp. 667-676, 1996.

[2] J. Aloimonos. “Purposive and Qualitative Active Vision”, *IUW*, pp 816-828, Sept. 1990.

[3] K. Andress and A. Kak. “Evidence Accumulation and Flow of Control in a Hierarchical Spatial Reasoning System,” *AI Magazine*, 9(2):75-94, 1988.

[4] M. Arbib. *The Metaphorical Brain: An Introduction to Cybernetics as Artificial Intelligence and Brain Theory*. Wiley Interscience, New York, 1972

[5] D. Ballard. “Generalizing the Hough Transform to Detect Arbitrary Shapes,” *PR*, 13(2):11-122, 1981.

[6] R. Beveridge. *LiME Users Guide*. Technical report 97-22, Colorado State University Computer Science Department, 1997.

[7] B. Burns, A. Hanson and E. Riseman. “Extracting Straight Lines,” *PAMI* 8(4):425-455, 1986.

[8] S. Chien, H. Mortensen, C. Ying and S. Hsiao. “Integrated Planning for Automated Image Processing,” *Integrated Planning Applications*, AAAI Spring Symposium Series, March 1995, pp 26-35.

[9] V. Clement and M. Thonnat. “A Knowledge-Based Approach to Integration of Image Processing Procedures,” *CGVIP*, 57(2):166-184, 1993.

[10] D. Comaniciu and P. Meer. “Robust Analysis of Feature Space: Color Image Segmentation,” *CVPR*, pp.750-755, 1997.

[11] B. Draper, R. Collins, J. Brolio, A. Hanson and E. Riseman. “The Schema System,” *IJCV*, 2(2):209-250, 1989.

[12] B. Draper and A. Hanson. “An Example of Learning in Knowledge Directed Vision,” *Theory and Applications of Image Analysis*, World Scientific, Singapore, 1992. pp.237-252.

[13] B. Draper. “Modelling Object Recognition as a Markov Decision Process,” *ICPR*, D95-99, 1996.

[14] B. Draper and K. Baek. “Bagging in Computer Vision,” *CVPR*, pp. 144-149, 1998.

[15] V. Hwang, L. Davis and T. Matsuyama. “Hypothesis Integration in Image Understanding Systems,” *CGVIP*, 36(2):321-371, 1986.

[16] K. Ikeuchi and M. Hebert. “Task Oriented Vision,” *IUW*, pp. 497-507, 1990.

[17] X. Jiang and H. Bunke. “Vision planner for an intelligent multisensory vision system,” *Automatic Object Recognition IV*, pp. 226-237, 1994.

[18] M. Kass, A. Witken and D. Terzopoulis. “Snakes: Active Contour Models,” *IJCV* 1(4):321-331, 1988.

[19] A. Lansky, et al. “The COLLAGE/KHOROS Link: Planning for Image Processing Tasks,” *Integrated Planning Applications*, AAAI Spring Symposium Series, 1995,pp 67-76.

[20] M. Maloof, P. Langley, S. Sage, T. Binford. “Learning to Detect Rooftops in Aerial Images,” *IUW*, 835-846, 1997.

[21] W. Mann and T. Binford. “SUCCESSOR: Interpretation Overview and Constraint System,” *IUW*, 1505-1518, 1996.

[22] D. McKeown, W. Harvey and J. McDermott. “Rule-Based Interpretation of Aerial Imagery,” *PAMI*, 7(5);570-585, 1985.

[23] J. Mundy. “The Image Understanding Environment Program,” *IEEE Expert*, 10(6):64-73, 1995.

[24] M. Nagao and T. Matsuyama. *A Structural Analysis of Complex Aerial Photographs*. N.Y.: Plenum Press, 1980.

[25] D. Nguyen. *An Iterative Technique for Target Detection and Segmentation in IR Imaging Systems*, Technical Report, Center for Night Vision and Electro-Optics, 1990.

[26] J. Peng and B. Bhanu. “Closed-Loop Object Recognition using Reinforcement Learning,” *PAMI* 20(2):139-154, 1998.

[27] J. Rasure and S. Kubica. “The KHOROS Application Development Environment,” In *Experimental Environments for Computer Vision*, World Scientific, New Jersey, 1994.
[28] S. Ravela, B. Draper, J. Lim and R. Weiss. “Tracking Object Motion Across Aspect Changes for Augmented Reality,” *IUW*, pp. 1345-1352, 1996.
[29] R. Rimey and C. Brown. “Control of Selective Perception using Bayes Nets and Decision Theory,” *IJCV*, 12(2):173-207.
[30] R. Sutton. “Learning to Predict by the Methods of Temporal Differences,” *ML*, 3(9):9-44, 1988.
[31] G. Tesauro. “Temporal Difference Learning and TD-Gammon,” *CACM*, 38(3):58-68, 1995.
[32] S. Ullman. “Visual Routines,” *Cognition*, 18:97-156, 1984.
[33] S. Umbaugh. *Computer Vision and Image Processing: A Practical Approach using CVIPtools*, Prentice Hall, New Jersey, 1998.
[34] C. Watkins. *Learning from Delayed rewards*, Ph.D. thesis, Cambridge University, 1989.
[35] W. Zhang and T. Dietterich. “A Reinforcement Learning Approach to Job-Shop Scheduling,” *IJCAI*, 1995.