Adapting Object Recognition Across Domains: A Demonstration

Bruce A. Draper[1], Ulrike Ahlrichs[2], Dietrich Paulus[2]

[1] Department of Computer Science
Colorado State University
Fort Collins, CO 80523, USA
draper@cs.colostate.edu
[2] Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg
91058 Erlangen, Germany
ahlrichs,paulus@informatik.uni-erlangen.de

Abstract. High-level vision systems use object, scene or domain specific knowledge to interpret images. Unfortunately, this knowledge has to be acquired for every domain. This makes it difficult to port systems from one domain to another, and therefore to compare them. Recently, the authors of the ADORE system have claimed that object recognition can be modeled as a Markov decision process, and that domain-specific control strategies can be inferred automatically from training data. In this paper we demonstrate the generality of this approach by porting ADORE to a new domain, where it controls an object recognition system that previously relied on a semantic network.

1 Introduction

High-level vision systems can be defined as those that use object, scene or domain knowledge to control the recognition process. By this definition, the first working high-level vision system may have been Nagao and Matsuyama's aerial image analysis system [1]. Since then, high level vision systems have been built using production systems [2-4], blackboard systems [5, 6], semantic networks [7-9] and Bayesian networks [10, 11], not to mention hybrid combinations of these techniques above, e.g. [12, 13]. Although much of the work on high-level vision originated in the 1980's, research continues today (e.g. [9, 14]); see [15] for a recent review.

There are good intellectual reasons to pursue high-level vision. There are currently no general-purpose object recognition techniques, only techniques that can recognize limited classes of objects in restricted domains. It is therefore natural to

speculate that a general-purpose vision system might be built by selecting among or combining multiple techniques. Moreover, there is no reason to believe that any single technique should work for all objects – some objects are defined by their shapes, while others are defined by their colors, subparts, textures or contexts. This suggests that object knowledge should help in selecting the most efficient and robust method for interpreting a scene. Finally, there is psychological evidence for the use of object-specific *signal features* to categorize objects (see, for example, [16] pg. 114).

Unfortunately, high-level vision systems have proved to be problematic. They are difficult and time-consuming to build, and are often brittle once built. (See [17, 18] for discussions of knowledge engineering and vision.) Worse still, it is difficult to conclude anything from the behavior of these prototype systems, since changing the knowledge base can alter almost any success or failure. Of particular importance to this paper, the dependence of high-level systems on domain-specific databases makes it difficult to port them from one domain to another, since the databases have to be re-engineered for every task. This limits their utility, since no high-level system is really general. Moreover, since they cannot be ported without fundamentally changing their behavior, direct comparisons are impossible, undermining the improvements that are supposed to come from competition and refinement.

In the last several years, a new class of high-level vision systems have emerged that avoid many of the problems above, while still using object, scene and domain information to direct processing. These systems use machine learning techniques to acquire information from training images. This simplifies system construction, and makes it possible to port them from domain to domain. Examples of these systems include [19-22].

For the last several years, the first author has advocated the use of Markov models for high-level computer vision, with reinforcement learning as the training method [18, 23]. As a prototype, ADORE (for *Ad*aptive *O*bject *Re*cognition) was built and trained to find buildings in aerial imagery [19]. In this paper we demonstrate that ADORE really can be ported from one domain to another, by training it to recognize objects in a new domain (office supplies), using images and routines developed at another university (Erlangen-Nürnberg). The system was ported by two people in the span of one week, and required no significant changes in the underlying learning algorithms or control systems. The routines of the Erlangen System are also used in an object recognition system on the same task domain [24].

2 ADORE

The adaptive object recognition (ADORE) project at Colorado State University approaches object recognition as a supervised learning task. Developers train ADORE to recognize specific objects by providing training images and training signals, where a training signal provides rewards based on how closely the output

matches the desired output for a training image. ADORE learns control strategies that maximize the expected value of the reward signal, given a library of visual procedures. These control strategies can then be used to hypothesize new object instances in novel images. Thus ADORE is a method for learning and applying high-level visual control strategies, as depicted in Figure 1.

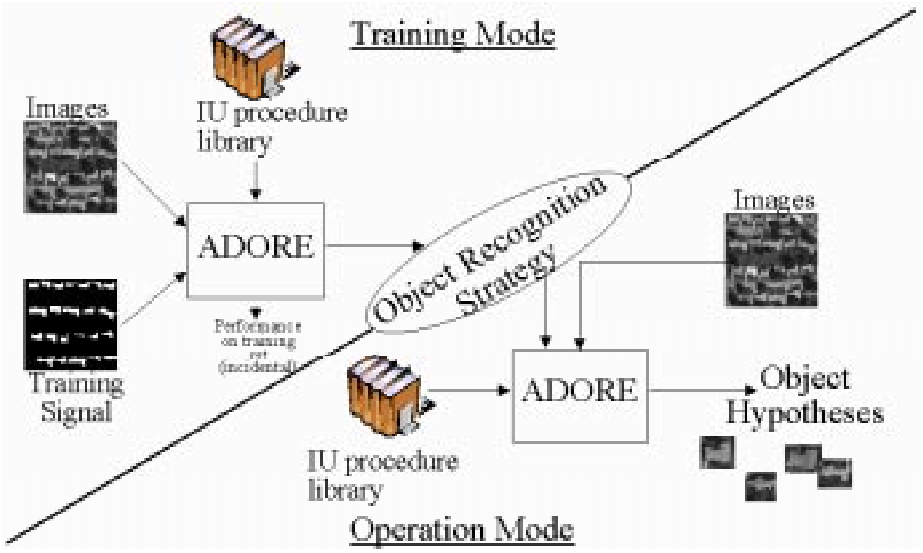

Fig. 1. Overview of ADORE. The system learns an object-specific recognition strategy that controls vision procedures by comparing training images to an ideal training signal. Once learned, these recognition strategies are then used to find object instances in novel test images.

ADORE learns control strategies by modeling object recognition as a Markov decision process. In general, a Markov decision problem is defined in terms of a set of states and a set of actions; the goal is to find a control policy that maps states onto actions so as to maximize the expected total reward. In the case of ADORE, 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, such as correlation or line extraction. Actions change the state of the system by producing new data tokens from old data tokens. A control strategy (or 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. (See [19] for a software-level description of ADORE.) It should be noted that ADORE controls vision procedures, not physical sensors, so the strategies learned by ADORE are not active vision strategies. Instead, they are knowledge-directed strategies that use learned information to direct the recognition process.

The control strategies learned by ADORE are dynamic. In other words, ADORE does not choose a fixed sequence of actions (vision procedures) to apply to all images in a domain. Instead, it learns to select the next vision procedure based on attributes is measures of the previous result. In the past, this capability was used mostly to recover from unreliable actions. For example, if a segmentation routine returned too many regions, the control strategy could respond by invoking a region merging procedure. In this paper, however, it is used to select among different object labels.

One of the strengths of ADORE is that it is a very general mechanism for learning visual control strategies, and is not limited to one style of visual processing or another. In [19], the objects to be recognized were rigid geometric shapes, and the routines in the procedure library grouped regions, points and lines into geometric structures. In [25], the procedure library contained focus of attention routines and preprocessing routines for a principal components analysis (PCA) recognition system. In this paper, the procedure library contains more traditional pattern recognition routines that segment images and classify regions based on size, color and texture (see Section 4).

3 ANIMALS

The ANIMALS[1] system (shown in Figure 2) was developed at the University of Erlangen-Nürnberg as a prototype for systems combining active vision with knowledge-based object recognition [9]. In particular, the goal in ANIMALS was to combine data-driven and knowledge-based techniques so as to enable goal-directed exploration under the guidance of an explicit knowledge base. As a result, ANIMALS is a complex, multi-part system. It includes a bottom-up subsystem that uses sensor movements along a rail to compute 3D depth maps of scenes that are registered to color images. It has an active vision component that responds to cues (typically colors) in the image data by panning and zooming the camera to produce high-resolution images at known scales (based on the depth data) centered on potential objects in the scene. Finally, it has a knowledge-based object recognition component that uses a semantic network to interpret and label high-resolution images. These three components are combined in a loop, so that the system can get a low-resolution but 3D view of a large scene, select promising locations, and then iteratively zoom in and interpret each region of interest until a target object is found.

Of interest to this paper is the knowledge-based object recognition component of ANIMALS. The goal of this system is to determine whether an image contains an instance of one of a set of target objects; in this paper, a hole punch, tape dispenser or glue stick. Because of its role within the larger ANIMALS system, the object recognition subsystem is always given a high-resolution image that is centered on a potential target. However, since color is not a perfect cue, some of these images are centered on other similarly colored objects found in office scenes, such as books or

[1] ANIMALS is an acronym for “An Image Analysis System”.

staplers. The accuracy of the depth computation is sufficient to determine the required focal length of the active camera such that close-up views of objects can be captured in which the objects have approximately the same size. Using scale invariant features for object recognition, these images can be classified reliably. However, the images differ in scaling up to 20%.

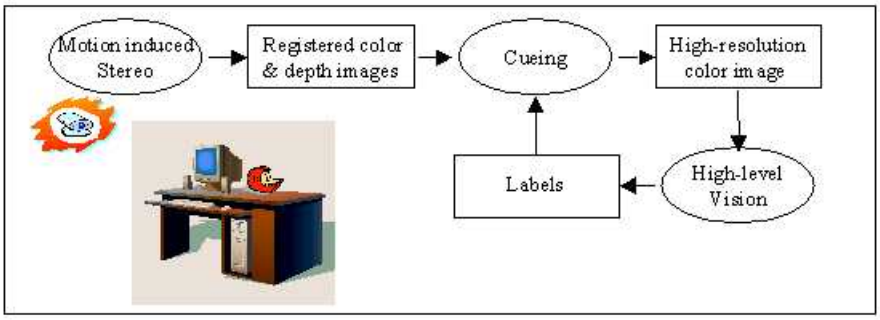

Fig. 2. The ANIMALS active object recognition system [9]

As described in [9], one of the available object recognition components of ANIMALS is a knowledge based system for recognizing 3D objects in 2D images. It begins by segmenting the image into regions, and then measures properties of those regions such as size and color to find the most likely object label. A semantic network relates objects to each other, and provides the basis for an A* search that matches features to object types [24].

4 Porting ADORE to ANIMALS

The goal of this exercise is to test the claim that ADORE is a general-purpose, high-level vision system that can be ported from domain to domain. In principle, all ADORE needs is a library of vision routines, a data set and a training signal; it will learn the best strategy possible for that library and task. This has never been demonstrated, however, by actually porting it to another domain. (Nor, to our knowledge, has it been demonstrated for any other high-level vision system.) In this paper, we port ADORE to the University of Erlangen-Nürnberg and apply it to high-resolution images captured by the active imaging component of ANIMALS. The task assigned to ADORE is to apply ANIMAL's segmentation and recognition components to compute an optimal object recognition strategy.

4.1 The Vision Procedure Library

The vision procedures for this exercise were extracted from the ANIMALS object recognition system. Each procedure consumed and/or produces one of four data representations: *image, segmentation, feature vector* or *label*. The system is given attributes it can measure about segmentations, feature vectors and labels. These attributes form the basis for its decisions about what procedure to execute at each choice point.

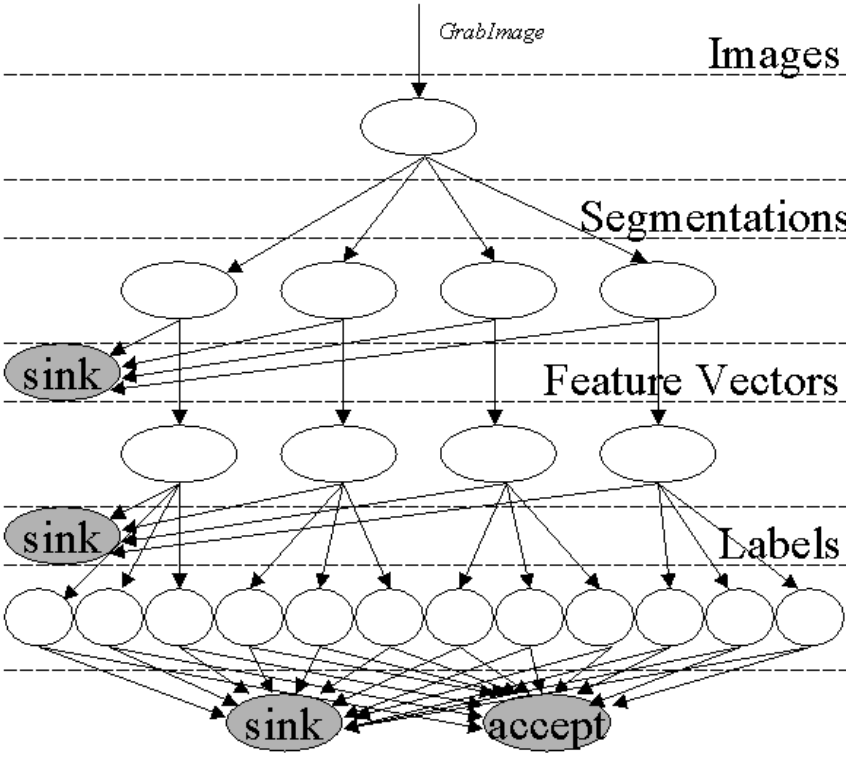

Fig. 3 The Visual Procedure Library depicted as a tree. Arrows represent procedures, while ovals represent data tokens. The individual procedures and representations are described in the text.

There are a total of twelve vision procedures. One of the procedures is *grabimage*, which is used to read in an image and start the process. Three of the procedures are *reject* procedures used to stop processing and signal to the user that that image con-

tains no known object[2]. Four of the procedures are segmentation procedures resulting in four segmentation results (see Fig. 3), although there are only two underlying color region segmentation algorithms. The two color segmentation algorithms are applied that were programmed to have uniform interfaces. One is a split and merge algorithm extended to work on color similarity measures. The other is the so-called CSC segmentation algorithm [26]. There are a total of four segmentation options in the procedure library because both segmentation algorithms are parameterized, and it is the goal of this experiment to determine the best sets of parameters. We therefore included two parameterizations of each algorithm in the database, giving ADORE four methods of segmenting images.

Once an image is segmented, there is only one procedure for reducing a segmentation to a feature vector, although ADORE also has the option of rejecting the segmentation if it does not contain a region that might be the target object. Given a feature vector, ADORE can hypothesize one of three labels (hole punch, tape dispenser, glue stick) using ANIMAL's Bayesian inference system. The attribute that ADORE can access for labels is the label probability. The difference is that where ANIMAL used a semantic net to guide which objects should be matched to the region data, ADORE selects which label to match based on a function it has learned over the space of feature vectors. Finally, ADORE must either accept or reject the label it creates.

Figure 3 shows the vision procedure library depicted as a tree. The arrows in the figure represent vision procedures, while the ovals are data tokens and therefore decision points. Every path through the tree in Figure 3 represents a valid sequence of vision procedures, and it is ADORE's task to dynamically choose paths based on attributes of the data.

4.2 The Training Data and Training Signal

We saved 57 images that had been acquired by the active imaging component of ANIMALS as a data set for ADORE. To avoid testing on the training data, we split the data set into ten groups of approximately six images each. We then trained ADORE on nine image sets and tested it on the tenth, using a cross-validation testing methodology[3]. Figure 4 shows some of the images. As in previous papers, we tested all possible sequence of actions on every test image (according to Figure 3) in order to build a database that allowed us to efficiently simulate running the system thousands of times; as a result, the training times for the experiments reported here were on the order of one-half hour each.

For a training signal, we manually labeled each image by object type. During training, when ADORE accepted an object label it received a reward of 0.9 if the label was correct, and 0.1 if the label was erroneous. If ADORE correctly con-

---

[2] Since every procedure in ADORE can only be applied to one datatype, it takes three different procedures to reject segmentations, feature vectors and labels.

[3] Due to time limitations, only 36 of the 57 images (six of ten groups) were used as test images.

cluded that the image did not contain a target object, it received a reward of 0.2, in order to bias it toward selecting object labels.

Fig. 4. Three images captured by the active imaging component of ANIMALS. These three images show the tape dispenser, hole punch and glue stick, respectively.

5 Experimental Results

The most important result of this demonstration is not the output of ADORE on any given image, but rather the fact that ADORE could be ported to a new domain with a new vision procedure library in two people-weeks (one calendar week). Moreover, porting it did not require any significant changes to the learning algorithms or control mechanisms of ADORE. It did require writing a new library file to describe the new vision procedures to ADORE, and it took some effort to extract the relevant vision algorithms from ANIMALS and convert them into stand-alone programs. This process was greatly helped by the fact that ANIMALS was developed using the ζππος (HIPPOS [27]) system. Most of the porting effort was spent writing shell scripts to support the cross-validation training and testing protocols, which have never been automated in ADORE.

The purpose of the cross-validation study, however, was to see if ADORE performed well in this new domain. Table 1 gives a summary of the results in the form of a confusion table. Overall, ADORE correctly identified 28 out of 36 images. It had the most trouble with the glue stick, which it identified correctly only once in four images. Most likely this is because long and narrow regions can also be produced by oversegmenting the hole punch or other objects such as the spines of books. The errors in Table 1 might be reduced by adding better attributes to describe segmentations, or by introducing new procedures to verify hypothesized labels.

Table 1. Confusion table for ADORE applied to the office supply domain describes in [9]

	Hole Punch	Tape Dispenser	Glue Stick	Other
Hole Punch	13	0	0	0
Tape Dispenser	0	7	1	3
Glue Stick	0	0	1	0
Other	2	0	2	7

A better way to judge ADORE's performance is in terms of rewards. The goal of ADORE's learning algorithm is to maximize the total expected reward. In this domain, the maximum possible reward is 0.9 for any image that contains a hole punch, tape dispenser or glue stick, and 0.2 for any other image. For the 36 images tested, the maximum possible total reward was 25.4.[4] The reward received by ADORE was 21.1, or 83% of maximum.

The results above (including Table 1) were generated by running ADORE in a traditional Markov process mode. In particular, it is not possible to "undo" an action in a traditional Markov process. The control policy selects the action with the best expected total future reward, but the actions are probabilistic, and in some instances the results of an action are worse than expected. In this case, a Markov process must carry on from the new, undesired state because it cannot backtrack. This traditional model was developed for the control of physical devices, where it is not possible to go back in time and undo the effects of previous actions. Object recognition, however, is a computational process. As long as the system has memory, it can always return to a previous state. For example, if a vision procedure oversegments an image, it is not necessary to proceed with the overly fractured segmentation; the system can go back to the original image and select another segmentation routine instead. In other words, backtracking is possible.

[4] Note that the maximum possible reward for a Markov system with this action & state space – i.e. the optimal control policy -- is unknown, but must be less than or equal to this number.

Table 2. Confusion table for ADORE with backtracking applied to the office supply domain.

	Hole Punch	Tape Dispenser	Glue Stick	Other
Hole Punch	13	0	0	0
Tape Dispenser	0	7	0	7
Glue Stick	0	0	4	1
Other	2	0	0	2

If we let ADORE backtrack while interpreting test images, it does more work in the sense that it executes more procedures. However, its performance improves. Table 2 shows the confusion table with backtracking. Overall, it interprets 26 of 36 images correctly – two less than without backtracking. It does a better job of maximizing its reward function, however. The total reward with backtracking improves to 23.0, compared to 21.1 without backtracking. This brings ADORE to within 91% of the maximum possible reward. The reward increases because the reward function is biased to prefer mislabeling an image that contains no object over failing to label an image that contains an object. With backtracking enabled, ADORE segments images multiple times if the first segmentation does not include a viable target region. In this way, it tries repeatedly to generate plausible labels. This maximizes the reward function by correctly labeling 26 of the 28 images that contain objects[5], as opposed to 21 of 28 without backtracking. The penalty is that it only correctly labels 2 of 10 images without target objects, but this is a good trade-off in terms of the reward function. Of course, ADORE could be retrained with other reward functions.

6 Conclusions

A major problem with traditional high-level vision systems is that they cannot be ported from domain to domain without manually re-engineering their knowledge

[5] One of the two mislabeled hole punch images is an imaging failure, in the sense that only part of the punch is in the field of view.

bases. In theory, newer systems that use machine learning techniques to infer their knowledge bases overcome this problem. However, their ability to be ported has never (to our knowledge) been tested in practice. This paper reports on an experiment in which ADORE was ported from one domain to another (and one university to another) by two people in a single week, and successfully learned to recognize objects in a new domain. We find it significant that ADORE was able to adapt to a new domain and procedure library so quickly, and also that the routines of ANIMALS allowed for such an export to another system.

7 Acknowledgements

This work was partially supported by DFG Ni 191/12 and SFB 603.

8 References

1. Nagao, M. and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*. 1980, New York: Plenum Press.
2. Ohta, Y., *A Region-oriented Image-analysis System by Computer*, Ph.D. thesis, 1980, Kyoto University: Kyoto, Japan.
3. McKeown, D.M., W.A. Harvey, and J. McDermott, *Rule-Based Interpretation of Aerial imagery*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1985. **7**(5): p. 570-585.
4. Clement, V. and M.Thonnat, *A Knowledge-Based Approach to Integration of Image Processing.* Computer Vision, Graphics and Image Processing, 1993. **57**(2): p. 166-184.
5. Draper, B.A., *et al.*, *The Schema System.* International Journal of Computer Vision, 1989. **2**(2): p. 209-250.
6. Andress, K.M. and A.C. Kak, *Evidence Accumulation & Flow of Control in a Hierarchical Spatial Reasoning System.* AI Magazine, 1988. **9**(2): p. 75-94.
7. Freuder, E.C. *A Computer System for Visual Recognition using Active Knowledge Sources.* in *International Joint Conference on Artificial Intelligence*. 1977. Cambridge, MA.
8. Niemann, H., *et al.*, *Ernest: A Semantic Network System for Pattern Understanding.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990. **12**(9): p. 883-905.
9. Paulus, D., *et al. Active Knowledge-Based Scene Analysis.* in *International Conference on Vision Systems*. 1999. Las Palmas de Gran Canaria, Spain: Springer-Verlag. p. 180-199.

10. Rimey, R.D. and C.M. Brown, *Control of Selective Perception using Bayes Nets and Decision Theory*. International Journal of Computer Vision, 1994. **12**: p. 173-207.
11. Mann, W.B. and T.O. Binford. *SUCCESSOR: Interpretation Overview and Constraint System*. in *Image Understanding Workshop*. 1996. Palm Springs, CA: Morgan Kaufman.
12. Hwang, V.S.-S., L.S. Davis, and T. Matsuyama, *Hypothesis Integration in Image Understanding Systems.* Computer Vision, Graphics and Image Processing, 1986. **36**(2): p. 321-371.
13. Stilla, U., E. Michaelson, and K. Lütjen, *Structural 3D-Analysis of Aerial Images with a Blackboard-based Production System*, in *Automatic Extraction of Man-made Objects from Aerial and Space Images*, A. Gruen, O. Kuebler, and P. Agouris, Editors. 1995, Birkhäuser: Basel. p. 53-62.
14. Thonnat, M., S. Moisan, and M. Crubézy. *Experience in Integrating Image Processing Programs*. in *International Conference on Vision Systems*. 1999. Las Palmas de Gran Canaria, Spain: Springer. p. 200-215.
15. Crevier, D. and R. LePage, *Knowledge-based Image Understanding Systems.* Computer Vision and Image Understanding, 1997. **67**(2): p. 161-185.
16. Kosslyn, S.M., *Image and Brain: The Resolution of the Imagery Debate*. 1994, Cambridge, MA: MIT Press. 516.
17. Draper, B.A. and A.R. Hanson, *An Example of Learning in Knowledge Directed Vision*, in *Theory and Applications of Image Analysis*, P. Johansen and S. Olsen, Editors. 1992, World Scientific: Singapore. p. 237-252.
18. Draper, B.A., A.R. Hanson, and E.M. Riseman, *Knowledge-Directed Vision: Control, Learning and Integration.* Proceedings of the IEEE, 1996. **84**(11): p. 1625-1637.
19. Draper, B.A., J. Bins, and K. Baek, *ADORE: Adaptive Object Recognition.* Videre, 2000. **1**(4): p. 86-99.
20. Peng, J. and B. Bhanu, *Closed-Loop Object Recognition Using Reinforcement Learning.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998. **20**(2): p. 139-154.
21. Au, W. and B. Roberts. *Adaptive Configuration and Control in an ATR System*. in *Image Understanding Workshop*. 1996. Palm Springs, CA: Morgan Kaufman. p. 667-676
22. Maloof, M.A., *et al. Learning to Detect Rooftops in Aerial Images*. in *Image Understanding Workshop*. 1997. New Orleans: Morgan Kaufman. p. 835-846
23. Draper, B.A., *Learning Control Strategies for Object Recognition*, in *Symbolic Visual Learning*, K. Ikeuchi and M. Veloso, Editors. 1997, Oxford University Press: New York. p. 49-76.
24. Ahlrichs, U., D. Paulus, and H. Neiman. *Integrating Aspects of Active Vision into a Knowledge-Based System*. in *International Conference on Pattern Recognition*. 2000. Barcelona.

25. Draper, B.A. and K. Baek. *Unsupervised Learning of Biologically Plausible Object Recognition Strategies,*. in *IEEE International Workshop on Biologically Motivated Computer Vision*. 2000. Seoul: IEEE CS Press.
26. Priese, L. and V. Rehrmann. *On Hierarchical Color Segmentation and Applications*. in *IEEE Conference on Computer Vision and Pattern Recognition*. 1993.
27. Paulus, D. and J. Hornegger, *Pattern Recognition of Images and Speech in C++*, in *Advanced Studies in Computer Science*. 1997, Vieweg: Braunschweig.