Task-Specific Utility in a General Bayes Net Vision System

Raymond D. Rimey Christopher M. Brown*

Computer Science Department
The University of Rochester
Rochester, New York 14627

Abstract

TEA is a task-oriented computer vision system that uses Bayes nets and a maximum expected-utility decision rule to choose a sequence of task-dependent and opportunistic visual operations on the basis of their cost and (present and future) benefit. We discuss technical problems regarding utilities, present TEA-1's utility function (which approximates a two-step lookahead), and compare it to various simpler utility functions in experiments with real and simulated scenes.

1 Introduction

The natural association between the utility function formalism from decision theory and computer vision applications was pointed out long ago [2, 8, 9], but was neglected until the general resurgence of interest in stochastic modeling, *e.g.* [13, 16].

A task-oriented system should perform its task with the minimum necessary effort. In computer vision, such a system must decide which information to gather, which operators to use at which resolution, and where to apply them. TEA is a task-oriented system that uses Bayes nets and a maximum expected-utility decision rule. Knowledge about the scene domain and about a specific visual task is represented in the Bayes nets. Selective viewing of a scene is achieved by using both a pointable camera and a small movable high-resolution window (called a fovea) in the camera field of view. The decision of where to point a camera (or a fovea) and what vision modules to run is made using a value/cost utility measure, where value is based on average mutual information measured between nodes in the Bayes net that correspond to the expected results of an action and to the task goal.

TEA's domain is table settings. Each task can be specified by asking a question about the scene. We are particularly interested in more qualitative tasks and those that require a variety of scene information to solve, for example: Is this breakfast, lunch, dinner, or dessert? Is this an informal or fancy meal? How far has the eating progressed?

In the sequel we discuss some issues in designing a utility function, present TEA-1's utility function, and demonstrate its use in real and simulated scenes. The TEA-1 framework and the utility function approach work together to produce task-specific utilities and behavior.

2 Some Issues Concerning Sequences of Observation Actions

TEA executes a sequence of visual actions until enough information has been gathered and integrated to solve the task. Thus, we have the following decision problem: *decide on the "best" sequence of observation actions.* If the goal of the task is captured by a utility on a (perhaps partial) world state corresponding to a leaf in a decision tree, and if all paths in the decision tree are enumerated, then classic decision theory can provide the solution. However, real problem domains are fantastically complex, making it infeasible to enumerate all paths in the decision tree. In some applications it may be acceptable to use a small set of "strategies", a subset of all paths, which can be scored using decision theory techniques to pick the best one. An alternative approach is a *myopic decision policy*, which decides only on the very next action to execute, considering the consequences of that action and possible subsequent actions only up to a finite horizon of steps. Then, the *horizon problem* is to decide how many actions to look ahead while still getting useful evidence from the actions.

A one-step myopic policy does not appear sufficient for most computer vision applications. For example, although an action that only detects an object may have little direct impact on the goal of a task, it enables subsequent actions to determine properties of

*This material is based upon work supported by the National Science Foundation under Grants numbered IRI-8920771 and IRI-8903582 and DARPA Contract MDA972-92-J-1012. The Government has certain rights in this material.

that object, and these properties may have very significant impact on the goal. Thus for practical decision making we need a utility function (1) that has a limited horizon or small-step lookahead and (2) that considers interactions between actions and their long term effects.

Decision theory is commonly used to solve the problem of deciding *what evidence to get next*. But in computer vision the system must also decide *where to look* for evidence. The problem is acute if the system has separate actions for camera and fovea movements. Visual actions have the *precondition* that their object be in the field of view, so the viewpoint-changing action can inherit the utility of the visual action. Also, the expected location of an object may depend on geometric constraints involving several objects; thus preconditions for efficiently locating that object may span a sequence of several actions. Again, a camera movement should inherit the value of placing several potentially interesting objects in the field of view at once. In an *information subgoal* several actions act together (or at least are executed at nearly the same time) to gather information for deciding on a specific subgoal. The *how to look* problem involves the choice of visual operators. Computer vision modules are robust and reliable only when applied in very specific (geometric or semantic) contexts. For example, a context may be an area in the scene defined by geometric relations with objects in the scene. Several preceding actions may be involved in establishing such a context.

Thus, efficiencies in selectively analyzing a scene arise from interactions in *sequences* of observation actions. One of our goals in this work is to push more lookahead into the utility function so it can deal with such interactions.

3 TEA-1

The TEA system [14, 15] gathers evidence visually and incorporates it into a Bayes net until the task, specified as a question, is answered to a desired degree of confidence. TEA-1 runs myopically by iteratively selecting the evidence gathering action that maximizes the expected utility criterion: 1) List all the executable actions. 2) Select the action with highest expected utility. 3) Execute that action. 4) Add the resulting evidence to the Bayes net and propagate its influence. 5) Repeat, until the task is solved.

Nodes in a Bayes net represent random variables with (usually) a discrete set of values (*e.g.* a *utensil* node could have values *(knife, fork, spoon)*). Links in the net represent (via tables) conditional probabilities that a node has a particular value given that an adjacent node has a particular value. Belief in the values for node X is defined as $BEL(x) = P(x \mid \mathbf{e})$, where $\mathbf{e}$ is the combination of all evidence present in the net. Evidence, produced by running a visual action, directly supports the possible values of a particular node (variable) in the net. There exist a number of evidence propagation algorithms, which recompute belief values for all nodes given one new piece of evidence [3, 7, 10, 13, 16].

TEA-1 uses a composite net, a method for structuring domain-specific knowledge into several separate Bayes nets [14]. A `PART-OF` net models subpart relationships between objects and whether an object is present in the scene or not. Associated with each object is an `IS-A` tree, a taxonomic hierarchy modeling one random variable that has many mutually exclusive values [4, 13]. An *expected-area* net models geometric relations between objects and the location of each object [15]. Task-specific knowledge is contained in a *task* net. There is one task net for each task that TEA-1 can solve (e.g. "Is this a fancy meal?"). Each of the separate nets in the composite net, except the task net, maintains its BEL values independently of the other nets. Evidence in the other nets affects the task net by updating its evidence nodes using BEL values in the other nets.

All actions in TEA-1 are constructed from one or more low-level vision modules, process either foveal image or peripheral image data, and may first move the camera or fovea. The interface between TEA-1 and an action involves a precondition, a function that executes the action, and rules for adding evidence. There are four types of precondition: that a particular node in the expected-area net be instantiated, that it not be instantiated, that it be instantiated and within the field of view for the current camera position, and the empty precondition. An action may add evidence to several nets and may do so in several ways [13].

Each kind of object usually has several actions associated with it. TEA-1 currently has 20 actions related to 7 objects. For example, the `per-detect-hough-plate` action moves the camera to a specified position (determined by the expected area net) and uses a Hough transform for plate-sized circles to detect the presence and location of a plate in a low-resolution image. `Per-classify-plate` moves the camera to a specified position, centers a window in the peripheral image there, and uses a color histogram to classify that area as paper or ceramic. `Fov-classify-plate` is similar but moves the fovea and uses high-resolution image data.

4 Utility Functions with Lookahead

4.1 TEA-1's Utility Function

TEA-1's utility function for an action has the following features: 1) An action's value is determined relative to the needs of the current task. 2) An action's cost is proportional to the amount of image data processed. 3) The utility accounts for peripheral actions that detect an object but don't otherwise generate information for the task. 4) It also accounts for the impact of making expected areas smaller so that future actions will have lower costs.

The utility $U(\alpha)$ of an action α is fundamentally modeled as $U(\alpha) = V(\alpha)/C(\alpha)$, a ratio of value $V(\alpha)$ and cost $C(\alpha)$. The value of an action is based on Shannon's measure of average mutual information [13], $V(\alpha) = I(T, e_\alpha)$, where T is the variable representing the goal of the task and e_α is the combination of all the evidence added to the composite net by action α [14].

An action α related to a specific object X has a cost proportional to the amount of image data that it processes. Thus TEA-1 defines the cost as $C(\alpha) = r^l_X C_0(\alpha)$, where $C_0(\alpha)$ is the execution time of action α if it processed a hypothetical image covering the entire scene. r^l_X is the ratio of the expected area for object X (within a confidence level l, we use $l = 0.9$) and the area of the entire scene. Over time, geometric constraints between located objects drive r^l_X toward zero.

TEA-1 uses a utility function $U(\alpha)$ for action α that incorporates two kinds of lookahead:

$$U(\alpha) = U_1(\alpha) + HU_2(EA_\alpha) \quad (1)$$

$H \in (0, 1)$ is a gain factor that specifies how much to weigh the second term relative to the first term. Currently we set $H = 1$. The first term, $U_1(\alpha)$, accounts for the future value of establishing the location of an object:

$$U_1(\alpha) = \frac{V(\alpha) + V(\beta)}{C(\alpha) + C(\beta)}$$

where

$$\beta = argmax_{\gamma \in Pre(\alpha)} \frac{V(\gamma)}{C(\gamma)}.$$

$Pre(\alpha)$ is the set of actions γ such that EITHER γ has a precondition satisfied by executing action α OR γ is already executable and $V(\gamma)/C(\gamma) < V(\alpha)/C(\alpha)$.

The second term, $U_2(EA_\alpha)$, accounts for the impact of making expected areas smaller so that future actions will have lower costs:

$$U_2(EA_\alpha) = \sum_{X \in EANet - EA_\alpha} \Delta(X)$$

where EA_α is the expected area node for the object associated with action α and

$$\begin{aligned} \Delta(X) &= \max_{\gamma \in Actions(X)} [\frac{V(\gamma)}{s^l_X C_0(\gamma)} - \frac{V(\gamma)}{r^l_X C_0(\gamma)}] \\ &= [\frac{r^l_X}{s^l_X} - 1] \max_{\gamma \in Actions(X)} [\frac{V(\gamma)}{r^l_X C_0(\gamma)}]. \end{aligned}$$

s^l_X is like r^l_X except it assumes that the location of action α's associated object is known.

Once the idea of an enhanced utility function is established, other improvements readily come to mind. An important one, now missing, is the cost of deploying the physical sensor. Depending on the speed of the active vision effectors, pointing the camera can be quick and cheap or slow and expensive. As the cost of moving the camera goes up, strategies of analyzing more and more objects in the current image will automatically result. It makes sense to add a term derived from the expected location net to predict the value of detecting new objects peripherally when the gaze direction changes. To some extent the well-known "cost of planning" can be addressed by estimating the cost of propagating information through the nets.

4.2 Experiments

Experiments show the relative performance of different utility functions in TEA's domain. A simulator was used for these experiments to increase the volume of data. The behavior of the simulator replicates the behavior we have observed in our long experience with lab experiments. The location and properties of each object in the simulated scene are randomly selected according to a configuration file specific to the kind of scene (*e.g.* a "fancy" table setting). An action designed to detect objects of type A will, with specific probabilities, either: (1) not detect an object of type A, (2) detect an object of type A if there are any in the field of view, (3) incorrectly say some other object in the field of view is of type A, or (4) detect "noise" as an object of type A. In each case, values for evidence reports, object size and location are picked according to probability distributions supplied to the simulator. An action designed to classify an object of type A randomly selects evidence report values according to the true type of the object (and whether it is in the field of view).

In this experiment we remove (ablate) various terms from the utility function and examine the system's performance, as measured by plotting the value of $BEL_k(notfancy)$, from the root node of the task tree after executing the k^{th} action, against T_k, the cumulative cost of executing the first through k^{th} actions. The task is to decide if the table setting is "fancy"

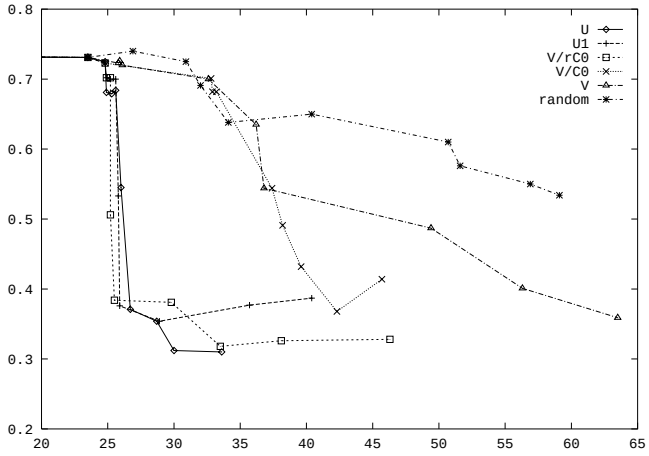

Figure 1: Utility ablation results for a "fancy" scene. There is one curve showing $BEL_k(notfancy)$ versus cumulative cost T_k, both averaged over 10 simulated trials, for each ablated utility function. After the *a priori* belief at $k = 0$, the next data point at $k = 1$ and $T_k = 23.5$ is the table-locating action which is always executed first.

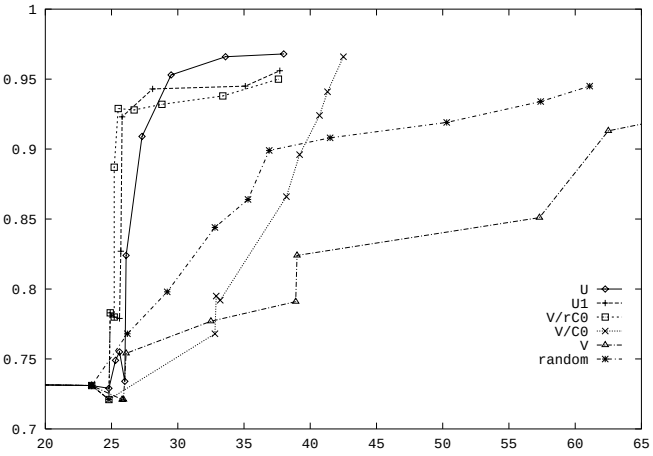

Figure 2: Utility ablation results for a "notfancy" scene.

or "notfancy", so if the scene is actually "fancy" then the value of $BEL(notfancy)$ should progress from its *a priori* value to a lower value. Here, better performance means that $BEL(notfancy)$ gets lower faster. For each degraded version of the utility function, the system was run on 10 fancy scenes and 10 notfancy scenes. Each curve in Figures 1 and 2 shows the average values of $BEL_k(notfancy)$ and T_k for the first 10 actions over the 10 trials when each degraded utility function was used. The five degraded utility functions were: (a) a random number (star points in the graph); (b) value only $-V(\alpha)$ (triangles); (c) value over operation cost $-V(\alpha)/C_0(\alpha)$ (X's); (d) value over operation cost including location constraints $-V(\alpha)/r_X^l C_0(\alpha)$ (boxes); (e) lookahead to include enabled utility $-U_1(\alpha)$, the first term in equation 1 (plusses); and (f) lookahead also to include location constraints $-U(\alpha)$ (diamonds), the full utility function as given by equation 1.

The curves for (a) thru (d) show that incorporating the respective terms into the utility function can markedly improve performance, *i.e.* that value of information [13], value/cost [2], and dynamic cost updating in r [15], are all very useful. Curves for (e) and (f) (plusses and diamonds) show why lookahead can be beneficial and why it can cause problems. One would expect what we had previously observed and what these simulated results verify: Lookahead tends to delay the sharp drop (or rise) in the belief curve, but the resulting change in belief tends to be more dramatic. On the other hand, at some point setting up for an action more valuable than the current best one can lead the system astray or in the wrong direction rather than simply doing the currently best action *now*. This problem seems more pronounced after the most useful actions have already been executed or if the scene only has a few objects of interest in it. Thus as we move towards more complex and realistic scenes and tasks the benefit of (indeed, need for) lookahead should become more pronounced.

We are interested in the interaction of visual actions, and Figure 3 illustrates the interactions. The execution of an action can have three short term effects on the choice of the next action to execute: it can affect $V(\alpha)$, r_X^l and preconditions. TEA-1 chooses the next action to execute by sorting the action utilities into decreasing order and executing the action with the largest utility, the first in the list. Let $R_k(\alpha)$ be the position of action α in the sorted list at time step k. $R_k(\alpha) = -1$ means action α has been executed, and $R_k(\alpha) = 21$ means the precondition for action α is not satisfied. Figure 3 shows $R_k(\alpha)$ for every action for the first 10 time steps in one run of the system (on a fancy scene and using the full utility function).

Crossing lines show dynamic changes in the rank order of actions. The effect of changing preconditions is clearly depicted by lines going to/from the 21 level. For example, a cup detection action executed at $k = 2$ negates the precondition on an alternative action to

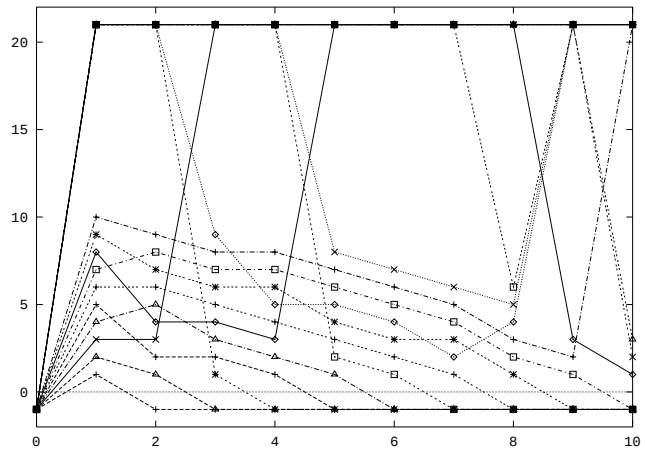

Figure 3: Rank of actions for each time step through one run of the system. Rank 21 means precondition not satisfied, rank -1 means action has been executed.

detect cups and enables two actions (foveal and peripheral) that classify cups. Moving the camera to the butter's expected area in the scene at $k = 8$ negates several foveal actions.

Lines otherwise cross in the plot (other than those going to/from level 21) when evidence from one action effects the $V(\alpha)$ and r_X^l values of other actions. The $V(\alpha)$ values tend not to change radically in this task and domain. Most of the crossing is due to radical changes in the r_X^l values.

5 Task Directed Behavior

5.1 The Task Net

An important feature of the TEA-1 design is that a different task net must be plugged into the composite net for each task the system is able to solve. The calculation of an action's value depends on the task net. Thus the action utilities directly reflect the information needs of the specific task, and produce a pattern of camera and fovea movements and visual operations that is unique to the task. Serendipitously acquired information is also incorporated, so the system is automatically opportunistic.

5.2 Experiments

The task is again to decide whether a table setting is fancy or notfancy, but this experiment used scenes in the lab. Figure 4 shows the utilities of the actions that initially have valid preconditions. Figures 5 and 6 show fancy and notfancy settings respectively. The sequence of actions executed by TEA-1 is summarized by the tables in the figures. In both cases, as the system executes actions to gather specific information about the scene, the belief in the goal (fancy or notfancy) approaches the correct value. The graphics superimposed on the images show the sequence of camera movements executed by the system.

$U(\alpha)$	α, an action
10.000	`table`
6.970	`per_detect_hough_cup`
6.927	`per_detect_hough_plate`
6.715	`per_detect_template_plate`
6.507	`per_detect_template_cup`
6.309	`per_detect_napkin`
6.210	`per_detect_utensil`
1.756	`per_detect_butter`
1.126	`per_detect_hough_bowl`
0.988	`per_detect_template_bowl`

Figure 4: Initial action utilities, at $k = 0$, for the experimental runs shown in Figures 5 and 6.

6 Summary

A similar approach (Bayes nets and expected utility) has been reported by a few groups, the most prominent work being [1, 11, 12] and [5, 6, 7] for vision and mobile robot applications respectively. TEA-1 is different because it deals with camera and fovea movements (via the expected-area net), and thus provides a solution to the where to look next problem. TEA-1 also distinguishes between vision modules that operate either on foveal or on peripheral image data. TEA-1's utility function contains several new ideas: Costs are proportional to the portion of the image that a vision module processes. The utility function tries to look a step ahead to account for various future consequences of an action's execution. Finally, TEA-1 causes utilities to be relative to the information requirements of a specific visual task (via the task net).

Acknowledgements

Peter von Kaenel helped build the simulator and worked on several of the vision modules and visual actions in the lab version.

References

[1] J. M. Agosta. The structure of Bayes networks for visual recognition. In *Uncertainty in AI 4*, pages 397–405. North-Holland, 1990.

[2] R. C. Bolles. Verification vision for programmable assembly. In *Proceedings: International Joint Conference on Artificial Intelligence*, pages 569–575, 1977.

k	$U(\alpha)$	α, an action	$BEL(nf)$
0		*a priori*	0.734
1	10.000	`table`	0.734
2	7.434	`per_detect_hough_cup`	0.714
3	13.010	`per_classify_cup`	0.552
4	5.409	`per_detect_hough_plate`	0.551
5	2.048	`per_detect_utensil`	0.551
6	19.049	`per_classify_utensil`	0.567
7	1.251	`per_classify_plate`	0.241
8	1.687	`per_detect_napkin`	0.116
9	1.866	`fov_classify_cup`	0.112
10	1.420	`per_detect_butter`	0.043

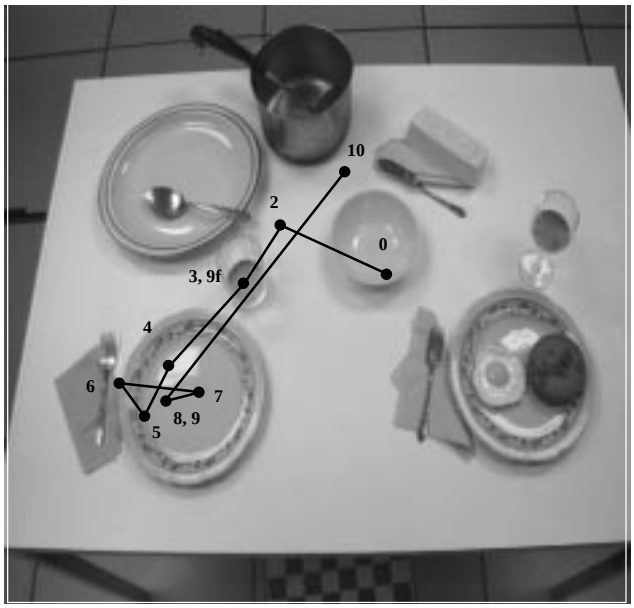

Figure 5: This is a "fancy" scene. The sequence of actions selected and executed by TEA-1 is shown in the table at the top. Each line corresponds to one cycle in the main control loop. The belief values listed are those after incorporating the results from each action. $BEL(nf)$ is an abbreviation for $BEL(notfancy)$. The path drawn on the wide-angle picture of the table scene at the bottom illustrates the camera movements made in the action sequence. The action at time step $k = 9$ moves the fovea to the position labeled "9f", which is in the far upper-right of the camera's field of view, while the camera remains pointing at position "9".

k	$U(\alpha)$	α, an action	$BEL(nf)$
0		*a priori*	0.734
1	10.000	`table`	0.734
2	7.434	`per_detect_hough_cup`	0.714
3	12.666	`per_classify_cup`	0.730
4	4.300	`per_detect_utensil`	0.730
5	10.146	`per_classify_utensil`	0.742
6	4.626	`per_detect_hough_plate`	0.742
7	2.514	`per_classify_plate`	0.930
8	2.342	`per_detect_napkin`	0.957
9	2.098	`fov_classify_cup`	0.958
10	1.792	`per_detect_butter`	0.983

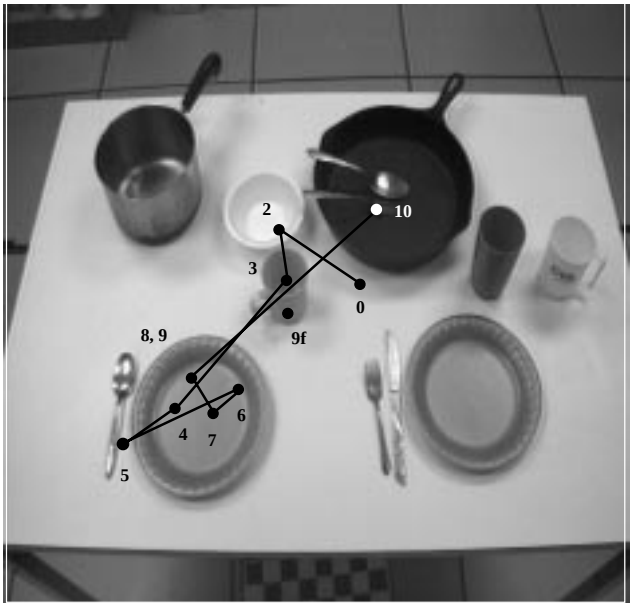

Figure 6: Same as Figure 5, but this is a "notfancy" scene.

[3] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[4] P. B. Chou and C. M. Brown. The theory and practice of Bayesian image labeling. *International Journal of Computer Vision*, 4(3):185–210, 1990.

[5] T. Dean, T. Camus, and J. Kirman. Sequential decision making for active perception. In *Proceedings: DARPA Image Understanding Workshop*, pages 889–894, 1990.

[6] T. Dean and J. Kirman. Representation issues in Bayesian decision theory for planning and active perception. In *Proceedings: DARPA Image Understanding Workshop*, pages 763–768, 1992.

[7] T. L. Dean and M. P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[8] J. Feldman and R. Sproull. Decision theory and artificial intelligence II: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[9] T. Garvey. Perceptual strategies for purposive vision. Technical Report 117, SRI AI Center, 1976.

[10] M. Henrion, J. S. Breese, and E. J. Horvitz. Decision analysis and expert systems. *AI Magazine*, 12(4):64–91, 1991.

[11] T. Levitt, T. Binford, G. Ettinger, and P. Gelband. Probability-based control for computer vision. In *Proceedings: DARPA Image Understanding Workshop*, pages 355–369, 1989.

[12] W. B. Mann and T. O. Binford. An example of 3D interpretation of images using Bayesian networks. In *Proceedings: DARPA Image Understanding Workshop*, pages 793–801, 1992.

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

[14] R. D. Rimey and C. M. Brown. Task-oriented vision with multiple Bayes nets. In A. Blake and A. Yuille, editors, *Active Vision*, pages 217–236. MIT Press, 1992.

[15] R. D. Rimey and C. M. Brown. Where to look next using a Bayes net: Incorporating geometric relations. In *Proceedings: European Conference on Computer Vision*, pages 542–550, 1992.

[16] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.