Where to Look Next Using a Bayes Net: Incorporating Geometric Relations [⋆]

Raymond D. Rimey and Christopher M. Brown

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

Abstract.
A task-oriented system is one that performs the minimum effort necessary to solve a specified task. Depending on the task, the system decides which information to gather, which operators to use at which resolution, and where to apply them. We have been developing the basic framework of a task-oriented computer vision system, called TEA, that uses Bayes nets and a maximum expected utility decision rule. In this paper we present a method for incorporating geometric relations into a Bayes net, and then show how relational knowledge and evidence enables a task-oriented system to restrict visual processing to particular areas of a scene by making camera movements and by only processing a portion of the data in an image.

1 Introduction

An important component in an active vision system is a spatially-varying sensor that can be pointed in space (using a pan-tilt platform) to selectively view a scene. Thus the system can not view the entire scene at once. We assume the sensor provides a peripheral image that is a low-resolution image of the entire field of view from one camera angle, and a fovea that is a small high-resolution image that can be selectively moved within the field of view. Spatially-varying sensors can be constructed in many ways: using special sensor array chips, two cameras with different focal lengths, or programmed in software.

The main reason for using a pointable spatially-varying sensor is the computational advantage it affords. Only a portion of the scene is imaged (and analyzed) at a time and even then only a portion of the potential image data is used. However, in exchange for this advantage a new problem is introduced: deciding where to point the camera (or fovea) and also what visual operations to run.

Our approach to this problem uses Bayes nets and a maximum expected utility decision rule. Bayes nets encode prior knowledge and incorporate visual evidence as it is gathered. The decision rule chooses where to point the camera (or fovea) and what visual operators to run.

This paper presents *expected area* nets, a method for incorporating geometric relations into a Bayes net, and shows how they can be used to restrict visual processing to particular areas of a scene. Section 2 summarizes our overall system, called TEA-1, and Section 3 presents the *expected area* net in detail. Section 4 explains how TEA-1 uses the *expected area* net: 1) to move cameras, 2) to create and use masks that process only a portion of an image, and 3) to make decisions while considering relational and location information. Experimental results are presented in Section 5. Section 6 contains some concluding remarks.

[⋆] This material is based upon work supported by the National Science Foundation under Grants numbered IRI-8920771 and IRI-8903582. The Government has certain rights in this material.

2 TEA-1: A Framework for Studying Task-Oriented Vision

This section summarizes the TEA-1 system, our second implementation of TEA, a general framework of a task-oriented computer vision system. The reader is refered to [10] for a detailed description of TEA-1. Earlier work involving TEA-0 and TEA-1 appears in [8, 9].

Main Control Loop. In TEA, a task is to answer a question about the scene: Where is the butter? Is this breakfast, lunch, dinner, or dessert? We are particularly interested in more qualitative tasks: Is this an informal or fancy meal? How far has the eating progressed? (Our example domain is table settings.) The TEA system gathers evidence visually and incorporates it into a Bayes net until the question can be answered to a desired degree of confidence. TEA runs by iteratively selecting the evidence gathering action that maximizes an expected utility criterion involving the cost of the action and its benefits of increased certainties in the net: 1) List all the executable actions. 2) Select the action with highest expected utility. 3) Execute that action. 4) Attach the resulting evidence to the Bayes net and propagate its influence. 5) Repeat, until the task is solved.

Bayes Nets. 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 (*i.e.* 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. Several references provide good introductions to the Bayes net model and associated algorithms, *e.g.* [2, 5, 7].

Composite Bayes Net. TEA-1 uses a composite net, a method for structuring knowledge into several separate Bayes nets [10]. A `PART-OF` net models subpart relationships between objects and whether an object is present in the scene or not. An *expected area* net models geometric relations between objects and the location of each object. Section 3 presents the *expected area* net in detail. Associated with each object is an `IS-A` tree, a taxonomic hierarchy modeling one random variable that has many mutually exclusive values [7]. Task specific knowledge is contained in a *task* net. There is one *task* net for each task, for example "Is this a fancy meal?", that TEA-1 can solve. 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 through a mechanism called packages, which updates values in evidence nodes in the *task* net using copies of belief values in the other nets.

Actions. TEA-1 uses the following description of an action:

- *Precondition.* The precondition must be satisfied before the action can be executed. 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.
- *Function.* A function is called to execute the action. All actions 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.
- *Adding evidence.* An action may add evidence to several nets and may do so in several ways (see [7]): 1) A chance node can be changed to a dummy node, representing virtual or judgemental evidence bearing on its parent node. 2) A chance node can

be instantiated to a specific value. Object locations get instantiated in the *expected area* net. 3) Evidence weight can be added to an `IS-A` type of net.

Each kind of object usually has several actions associated with it. TEA-1 currently has 20 actions related to 7 objects. For example, the actions related to plates are: The `per-detect-template-plate` action moves the camera to a specified position and uses a model grayscale template to detect the presence and location of a plate in the peripheral image. `Per-detect-hough-plate` uses a Hough transform for plate-sized circles for the same purpose. `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` moves the fovea (but not the camera) to a specified location and uses a color histogram to classify the area as paper or ceramic.

Calculating an Action's Utility. 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, how useful it is for toward the task, is based on Shannon's measure of average mutual information, $V(\alpha) = I(T, e_\alpha)$, 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. An action's cost is its execution time. The exact forms of the cost and utility functions depend on the *expected area* net and will be given in Section 4.

An important feature of the TEA-1 design is that a different *task* net is 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.

3 An Expected Area (Object Location) Bayes Net

Geometric relations between objects are modeled by an *expected area* net. The *expected area* net and `PART-OF` net have the same structure: A node in the `PART-OF` net identifies a particular object within the sub-part structure of the scene, and the corresponding node in the *expected area* net identifies the area in the scene in which that object is expected to be located. Fig. 1 shows the structure of one example of an *expected area* net.

In TEA-1 we assume a fixed camera origin. The location of an object in the scene is specified by the two camera angles, $\theta = (\phi_{pan}, \phi_{tilt})$, that would cause the object to be centered in the visual field. The height and width of an object's image is also specified using camera angles.

Thus a node in the *expected area* net represents a 2-D discrete random variable, θ. $BEL(\theta)$ is a function on a discrete 2-D grid, with a high value corresponding to a scene location at which the object is expected with high probability. Fig. 2(a)-(b) shows two examples of expected areas. Note that these distributions are for the location of the *center* of the object, and *not* areas of the scene that may contain *any part* of the object. Each node also contains values for the height and width of the object. Initially these are expected values, but once an object is located by a visual action the detected height and width are stored instead. The height and width are not used in belief calculation directly, but will be used to calculate conditional probabilities on the links (see below).

A root node R of an *expected area* net has an *a priori* probability, $P(\theta_R)$, which we assume is given. A link from node A to node B has an associated conditional probability, $P(\theta_B \mid \theta_A)$. Given a reasonable discretization, say as a 32x32 grid, each conditional probability table has just over a million entries. Such tables are unreasonable to specify

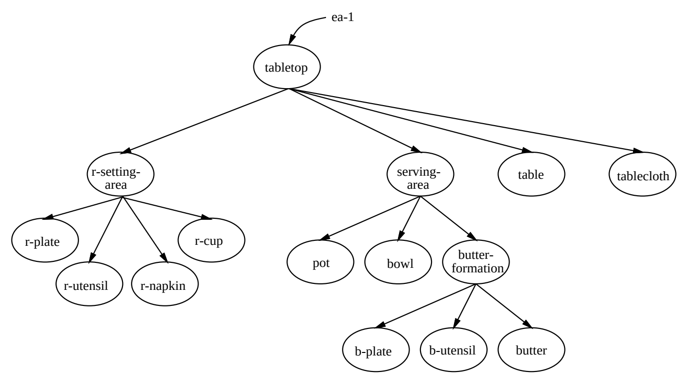

Fig. 1. The structure of an *expected area* net. The corresponding PART-OF net is similar.

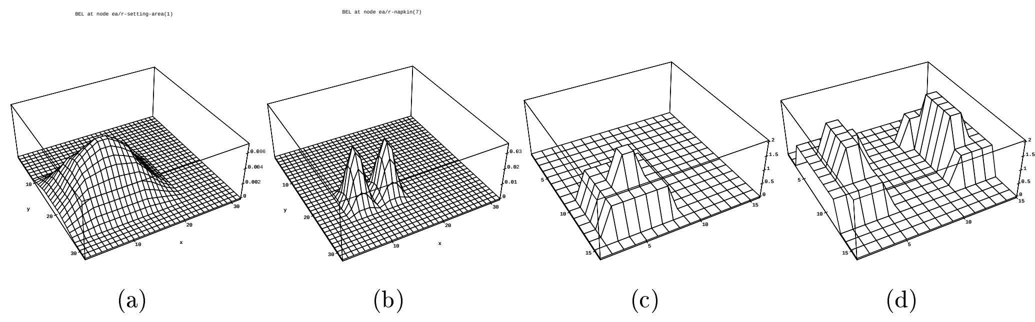

Fig. 2. The expected area (a) for *setting-area* (a place setting area) before the location of any other object is determined, and (b) for *napkin* after the location of the tabletop and plate have been determined. The relation maps (c) for *setting-area* given *tabletop*, and (d) for *napkin* given *setting-area*.

and cause the calculation of new belief values to be very slow. Next we present a way to limit this problem.

Relation Maps Simplify Specification of Probabilities. We make the following observations about the table of $P(\theta_B \mid \theta_A)$ values: 1) The table is highly repetitious. Specifically, ignoring edge effects, for every location of object A the distributions are all the same if they are considered *relative* to the given location of object A. 2) Belief calculations can be sped up by detecting terms that will have zero value. Therefore, rather than calculate all values of the distribution, we should use a function to calculate selective values. 3) The distribution depends on the size of object A. We assume the expected height and width of object A's image are known, but whenever an action provides direct observations of the object's dimensions those values should be used instead.

Our solution is to compute values of the conditional probabilities using a special simplified distribution called a relation map. A relation map assumes that object A has unity dimensions and is located at the origin. The relation map is scaled and shifted appropriately to obtain values of the conditional probability. This calculation is performed by a function that can be called to calculate select values of the conditional probability. Note that the spatial resolution of the relation map grid can be less than that of the expected area grid. Fig. 2(c)-(d) shows two examples of relation maps that were used in the calculation of the two expected areas shown in Fig. 2(a)-(b).

Given an expected area grid that is 32x32 (NxN) and a relation map grid that is 16x16 (MxM), all the values for one link's conditional probability table can be obtained by specifying only 256 (M^2) values. The brute force approach would require that 1048576 (N^4) values be specified.

Speeding up Calculation of Belief Values. When the set of expected locations for an object covers a relatively small area of the entire scene, the table of $P(\theta_B \mid \theta_A)$ values contains a large number of essentially zero values that can be used to speed up the belief propagation computation. The equations for belief propagation (and our notation) can be found in [7]. We do not give all the equations here for lack of space. The calculation of new $BEL(x)$ values for node X, with parent node U, contains two key equations: $\pi(x) = \sum_j P(x \mid u_j)\pi_X(u_j)$ and $\lambda_X(u) = \sum_i P(x_i \mid u)\lambda(x_i)$. These summations involve considerable time since x and u both denote a 2-D array (grid) of variables. Time can be saved in the first equation by not summing a term (which is an array) when it is multiplied by an essentially zero value. Specifically, for all j where $\pi_X(u_j)$ is essentially zero, we do not add the $P(x \mid u_j)\pi_X(u_j)$ term (an array) into the summation. Similar savings can be obtained in the second equation. For any given value of i, the $P(x_i \mid u)\lambda(x_i)$ term (an array) contains essentially zero values everywhere except for a few places (a small window in the array). We locate that window and only perform the sum for values inside the window.

Combining Location Information. The expected area for node B is actually calculated not from a single node like node A, but by combining "messages" about expected areas sent to it from its parent and all its children. This combination is performed within the calculation of $BEL(B)$. Generally, it is useful to characterize relations as "must-be", "must-not-be" and "could-be". Combination of two "must-be" maps would then be by intersection, and in general map combination would proceed by the obvious set-theoretic operations corresponding to the inclusive or exclusive semantics of the relation. In TEA-1, however, all the relations are "could-be", and the maps are essentially unioned by the belief calculation.

4 Using expected areas

Moving cameras. Actions in TEA-1 that must move the camera to the expected location of a specific (expected) object, say X, will move the camera to the center of mass of the expected area for object X. (This happens even if the expected area, when thresholded to a given confidence level, is larger than the camera's field of view. That case could be handled by making several camera movements to cover the expected area.)

Processing Only a Portion of an Image. Every action related to a specific object X processes only the portion of the image that is covered by the expected area of object X, when thresholded to a given confidence level. Let $l \in (0, 1)$ be the confidence level, which usually will be chosen close to 1 (typically 0.9). Let G_X^l be the smallest subset of all the grid points G_X for node X (that corresponds with object X) in the *expected area* net, such that their probabilities add up to l. G_X^l is the portion of the scene that should be analyzed by the action. Each action in TEA-1 creates a mask that corresponds to the portion of the current image data (*i.e.* after a camera movement) that overlaps G_X^l, and processes only the image pixels that are covered by that mask.

Deciding with Expected Areas. TEA-1's utility function for an action has the following features: 1) Costs are proportional to the amount of image data processed.

2) It deals with peripheral actions that detect an object but don't otherwise generate information for the task. 3) It considers that an action may have the impact of making the expected areas of other objects smaller. Recall that the utility of an action α is fundamentally modeled as a ratio of value $V(\alpha)$ (average mutual information) and cost $C(\alpha)$ as explained near the end of Section 2.

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 $C(\alpha) = r^l_A C_0(\alpha)$, 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 and the area of the entire scene. So the value of r^l_X is the size of the subset G^l_X divided by the size of the set G_X. $r^l_X = 1$ means that object X could be located anywhere in the entire scene. Over time, as other objects in the scene are located and as more and tighter relations are established, the value of r^l_X approaches zero. (Soon we will use a more accurate cost function that has an additional term for the cost of moving the camera or fovea, $C(\alpha) = C_{move}(\alpha) + r^l_A C_0(\alpha)$.)

TEA-1 uses the following "lookahead" utility function $U(\alpha)$ for action α.

$$U(\alpha) = \frac{V(\alpha) + V(\beta)}{C(\alpha) + C(\beta)} + H \sum_{X \in Net} \varDelta U(X) \qquad (1)$$

where

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

$$\varDelta U(X) = \max_{\gamma \in Actions(X)} [\frac{V(\gamma)}{s^l_X C_0(\gamma)} - \frac{V(\gamma)}{r^l_X C_0(\gamma)}]$$

The first term in equation (1) accounts for the future value of establishing the location of an object. $Pre(\alpha)$ 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 in equation (1) accounts for the impact of making expected areas smaller so that future actions will have lower costs. s^l_X is like r^l_X except it assumes that the location of action α's associated object is known. $H \in (0, 1)$ is a gain factor that specifies how much to weigh the second term relative to the first term. See [7] and [10] for more details about I and U respectively.

5 Experimental Results

A Basic Run of the System. The task of deciding whether a dinner table is set for a fancy meal or for an informal meal was encoded in a *task* net, and TEA-1 was presented the scene shown in Fig. 3, which shows a "fancy" meal. The sequence of actions executed by TEA-1 is summarized by the table in Fig. 3. The *a priori* belief of the table setting being fancy is 0.590, compared with 0.410 that it is informal. As the system executed actions to gather specific information about the scene, the belief that the setting is a fancy one approaches 0.974. The graphics on the right of the figure illustrate the sequence of camera movements executed by the system. Fig. 4 illustrates the execution of a few actions in the sequence, showing each action's results after any camera (or fovea) movement has been made and the expected area mask has been applied.

time	$U(\alpha)$	α, an action	$BEL(i)$
0		*a priori*	0.410
1	10.0	`table`	0.400
2	10.5	`per-detect-hough-cup`	0.263
3	42.8	`per-classify-cup`	0.343
4	11.3	`per-detect-hough-plate`	0.340
5	11.9	`per-classify-plate`	0.041
6	20.9	`per-detect-utensil`	0.041
7	58.8	`per-classify-utensil`	0.033
8	4.3	`per-detect-napkin`	0.026
9	3.3	`fov-classify-cup`	0.026
10	2.4	`fov-classify-plate`	0.026
11	1.7	`per-detect-hough-bowl`	0.026
12	0.6	`per-detect-butter`	0.026
13	0.4	`fov-verify-butter`	0.026

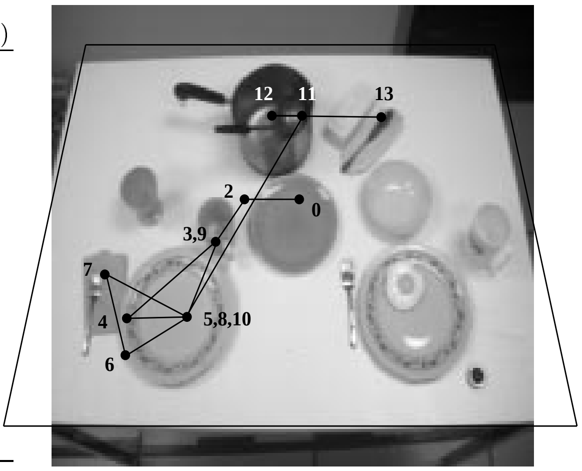

Fig. 3. The sequence of actions selected and executed by TEA-1 is shown in the table at left. Each line corresponds to one cycle in the main control loop. The belief values listed are those after incorporating the results from each action. The $BEL(i)$ column shows $BEL(informal)$, and $BEL(formal) = 1 - BEL(i)$. The path drawn on the wide-angle picture of the table scene at the right illustrates the camera movements made in the action sequence.

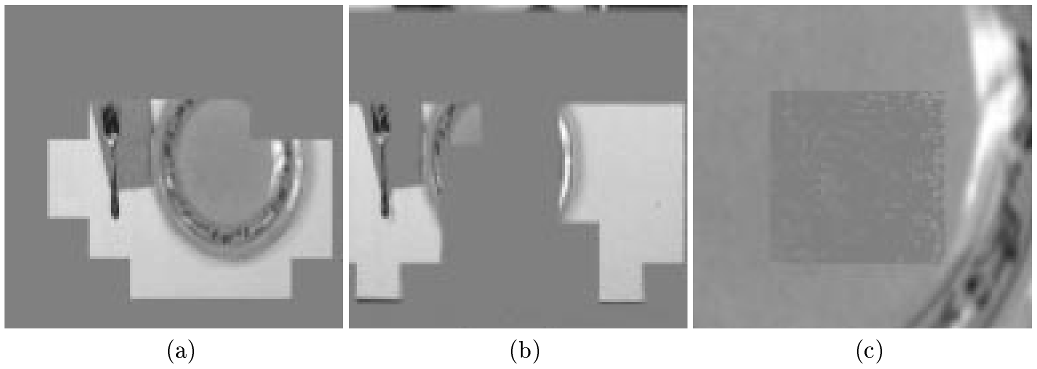

Fig. 4. Processing performed by individual actions. Image pixels outside the expected area mask are shown as gray values. (a) Results from the per-detect-hough-plate action executed at time step 4. (b) Results from the per-detect-napkin action executed at time step 8. The mask prevents the red napkin from being confused with the pink creamer container just above the plate. (c) Results from the fov-classify-plate action executed at time step 10. A zoomed display of the fovea centered on the plate is shown. Note: Fig. 5(b) shows results from the per-detect-hough-cup action executed at time step 2.

Expected Areas Shrink Over Time. As more objects are located via actions, the expected areas for the remaining objects (not yet located by actions) get narrower. Assume that TEA-1 has located (in order) the tabletop, then the plate, and finally the napkin. Fig. 5 shows how the cup's expected area gets narrower and how the `per-detect-hough-cup` action would *hypothetically* perform after each additional object is located. Parts (a) and (e) show the situation before any other objects have been located. The expected area is rather large, much larger than the field of view. The camera movement, made to the center of the cup's expected area, is much higher than the true location of the cup, and the action mistakenly detects the creamer container as the cup. The situation improves once the tabletop is located, as shown in parts (b) and (f). The expected area is (almost) small enough to fit in the field of view and its center corresponds better with the cup's actual location. A small portion of the image is masked out by the expected area, and the cup is correctly detected, but this is just lucky since the creamer and many other objects are still in the unmasked area. Parts (c) and (g) show the situation after the plate has been located. The cup's expected area is much smaller. Finally, in parts (d) and (h), once the napkin has been located, the cup's expected area is small enough that the action is very likely to detect the cup correctly.

6 Concluding Remarks

Several people are investigating the use of Bayes nets and influence diagrams in sensing problems. The most relevant work comes from two groups: Levitt's group was the first to apply Bayes nets to computer vision [1, 6]. Dean's group is studying applications in sensor based mobile robot control, using a special kind of influence diagram called a temporal belief network (TBN) [3, 4]. More recently, they have used sensor data to maintain an occupancy grid, which in turn affects link probabilities in the TBN.

The current TEA-1 system design, incorporating *expected area* nets, provides a framework that enables the system to make decisions about moving a camera around and about selectively gathering information. Thus we can begin using TEA-1 to study questions regarding task-oriented vision [8, 9, 10].

Deciding where to move a camera (or fovea) is an interesting problem. TEA-1 does the simplest thing possible by moving to the center of the expected area of one object. If several objects of interest should fall in the field of view, then it may for example be better to move the camera to the center of that set of objects. In our experiments to date, TEA-1 has relied mainly on camera movements to get the first piece of information about an object, while fovea movements are mostly used for verification. This behavior is determined by the costs and other parameters associated with actions. Another interesting problem is to consider the tradeoffs between a camera and a fovea movement. A camera movement is expensive and an action following one processes a completely new area of the scene, which means there is risk of not finding anything, but if something is found it will likely have large impact for the task. Alternatively, a fovea movement is cheap but produces image data near an area already analyzed, so there is a good chance of finding some new information, but it will tend to have a small impact on the task.

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. E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.
3. T. Dean, T. Camus, and J. Kirman. Sequential decision making for active perception. In *Proceedings: DARPA Image Understanding Workshop*, pages 889–894, 1990.
4. T. L. Dean and M. P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.
5. M. Henrion, J. S. Breese, and E. J. Horvitz. Decision analysis and expert systems. *AI Magazine*, 12(4):64–91, 1991.
6. 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.
7. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

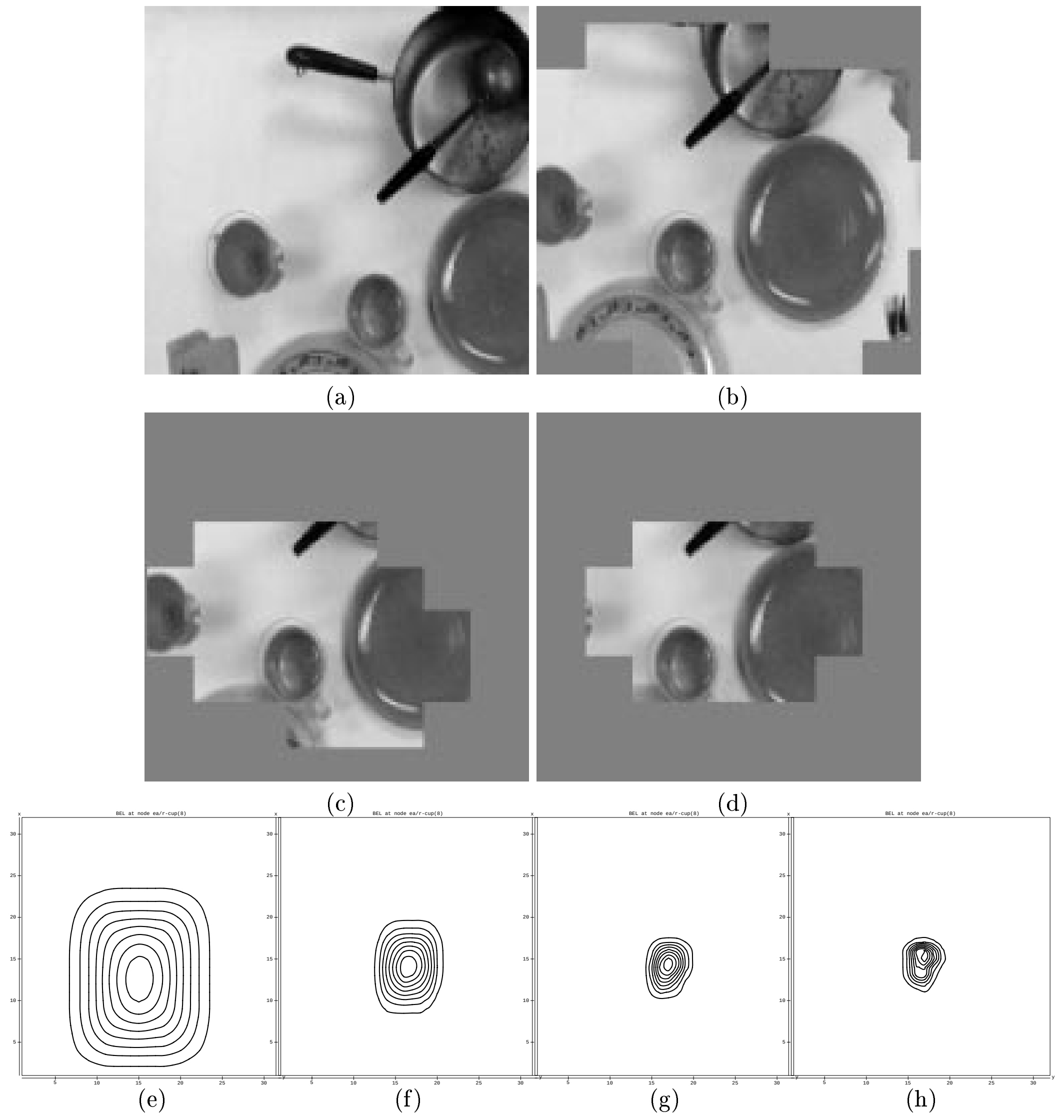

Fig. 5. Performance of a cup detection action as the cup's expected area narrows over time: (a) before any objects have been located, (b) after the tabletop has been located, (c) after the tabletop and plate have been located, (d) after the tabletop, plate and napkin have been located. The cup's expected areas in (a)-(d) are plotted separately in (e)-(h). These plots must be rotated 90 degrees clockwise to match the images in (a)-(d).

8. R. D. Rimey. Where to look next using a Bayes net: An overview. In *Proceedings: DARPA Image Understanding Workshop*, pages 927–932, 1992.
9. R. D. Rimey and C. M. Brown. Task-oriented vision with multiple Bayes nets. Technical Report 398, Department of Computer Science, University of Rochester, November 1991.
10. 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.

This article was processed using the LaTeX macro package with ECCV92 style