Task-Oriented Vision with Multiple Bayes Nets

Raymond D. Rimey
rimey@cs.rochester.edu
(716) 275-1448

Christopher M. Brown
brown@cs.rochester.edu
(716) 275-7852

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

Technical Report 398
November 1991

Abstract

We present the basic framework of a task-oriented computer vision system, called TEA, that uses Bayes nets and a maximum expected utility decision rule. Knowledge about the scene and about the nature of the specific task given to the system are represented in Bayes nets. We introduce a new kind of Bayes net, called an expected area net, that models both geometric relations between objects and the areas in the scene where objects are expected to be located. The decision of what areas of the scene to run a vision module on can be made using this relational knowledge. The decision of what vision modules to run is made using a value/cost utility measure, where value is based on mutual information measured between nodes in the Bayes net that correspond to actions and to the goal of the task, and cost is proportional to the fraction (using the expected area net) of the image that a vision module processes. We present a method for combining several Bayes nets that represent different types of scene and task knowledge into a single composite network. Composite nets support more complex visual tasks, and enable the calculation of action utilities relative to the information requirements of a specific task. TEA models camera movements and distinguishes between vision modules that operate either on foveal or on peripheral image data. Experimental results are presented from the TEA-0 system, our initial implementation of the general TEA framework.

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.
1 Introduction

A general computer vision system should be task oriented, not task-specific. A task-oriented system must determine the minimum information necessary to solve a given task. Determining what that information is and how to obtain it involves a number of decisions. The system must decide what vision modules to run, what areas of the scene to run the vision modules on, what resolution of image data to use, how long to allow a vision module to run on a given area of the scene, etc.

The "where to look next?" problem is now one of the recognized central problems for a behaving vision system that must allocate scarce resources (a camera, computing cycles, time) to a visual problem. A generalized version of this problem is "where and how to look next?", which is just the task-oriented vision problem. Task-oriented vision will endow a system with a flexible control structure for allocating and using only the resources necessary to answer the visual question of the moment to the desired level of detail or certainty. Under this control, vision modules of known characteristics are sequentially brought to bear on selective areas of the scene. Their choice should depend on the results of previously executed modules. Each module produces a partial representation of the (minimal) information needed to answer the question, and this evidence is combined to produce a partial answer. The process continues until the partial answer is certain enough. Table 1 summarizes the key differences between the standard passive (or Marr) vision paradigm and the task-oriented approach we propose.

A task-oriented vision capability can act as the cognitive executive for active vision. If an active computational agent is subject to an information load that can overwhelm its resources, the executive can allow it to ignore irrelevant stimuli, choose its tasks wisely, survive, and achieve its goals. Alternatively, task-oriented vision can simply save time and effort in doing visual jobs that in humans require attentional shifts, such as radiograph and CAT scan interpretation, photo interpretation, traffic monitoring, etc. Last, the task-oriented approach should make for more dependable vision performance built from more general (less domain-specific) vision tools. The claim is that current vision modules, even relatively simple ones, can become useful and robust when they are carefully applied in a specific context.

In what follows we present the basic framework of a task-oriented computer vision system, called TEA, that uses Bayes nets and a maximum expected utility decision rule.

<table>
<thead>
<tr>
<th>Passive vision</th>
<th>Task-oriented vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>use all vision modules</td>
<td>use only some vision modules</td>
</tr>
<tr>
<td>process entire image</td>
<td>process areas of the image</td>
</tr>
<tr>
<td>maximal detail</td>
<td>sufficient detail</td>
</tr>
<tr>
<td>extract representation first</td>
<td>ask question first</td>
</tr>
<tr>
<td>answer question from representation data</td>
<td>answer question from scene data</td>
</tr>
<tr>
<td>unlimited resources</td>
<td>resource limitations</td>
</tr>
</tbody>
</table>

Table 1: Key differences between passive vision and task-oriented vision.
Knowledge about the scene and about the nature of the specific task given to the system are represented in Bayes nets. We introduce a new kind of Bayes net, called an expected area net, that models both geometric relations between objects and the areas in the scene where objects are expected to be located. The decision of what areas of the scene to run a vision module on can be made using this relational knowledge. The decision of what vision modules to run is made using a value/cost utility measure, where value is based on mutual information measured between nodes in the Bayes net that correspond to actions and to the goal of the task, and cost is proportional to the fraction (using the expected area net) of the image that a vision module processes. We present a method for combining several Bayes nets that represent different types of scene and task knowledge into a single composite network. Composite nets support more complex visual tasks, and enable the calculation of action utilities relative to the information requirements of a specific task. TEA models camera movements and distinguishes between vision modules that operate either on foveal or on peripheral image data. Experimental results are presented from the TEA-0 system, our initial implementation of the general TEA framework.

Our approach has as background a large amount of research into visual attention [Ullman, 1984; Poggio and Hurlbert, 1985; Posner and Presti, 1987; Humphreys and Bruce, 1989; Schwartz, 1990], classical work in eye movements [Yarbus, 1967; Stark and Ellis, 1981], and recent advances in active vision [Bajcsy, 1988; Brown, 1988; Burt, 1988; Aloimonos, 1990; Ballard, 1991], including camera movements and foveal - peripheral sensors [Browse and Rodrigues, 1988; Burt, 1988; Clark and Ferrier, 1988; Yeshurun and Schwartz, 1989; Bolle et al., 1990]. Specifically, our tools are decision theory, utility theory, and Bayesian probabilistic models [Garvey, 1976; Bolles, 1977; Feldman and Sproull, 1977; Durrant-Whyte, 1988; Hutchinson and Kak, 1989; Chou and Brown, 1990; Shafer and Pearl, 1990; Hager, 1990; Safranek et al., 1990]. Two recent key developments are Bayes nets [Pearl, 1988], and influence diagrams [Shachter, 1986]. Applications using these new techniques are beginning to appear. The first large experimental system that applied Bayes nets to computer vision is by Levitt [Levitt et al., 1989]. The formulation of that system using influence diagram techniques is discussed in [Agosta, 1990]. A sensor/control problem involving a real milling machine is solved using influence diagram techniques in [Agogino and Ramamurthi, 1990]. A special kind of influence diagram, called a temporal belief network, is discussed in [Dean et al., 1990; Dean and Wellman, 1991], and is being studied for an application in sensor based mobile robot control.

2 Preliminaries

The TEA system runs by iteratively selecting the evidence gathering action that maximizes an expected utility criterion:

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.

The following sections expand on aspects of the above algorithm. Section 3 introduces Bayes nets, the idea of a composite network, and then the four kinds of nets (including the expected area net) that we use to build a composite net. Section 4 describes visual actions and how they are integrated into a Bayes net. Section 5 shows how to calculate the expected utility of an action using a composite net. Preliminary experiments are presented in Section 6, and the last section is a wrap up.

But first, in this section, we present the application domain that we are currently using. The approach is applicable whenever the scenes obey regularities or have structure that can be captured in the semantic data structures supporting Bayesian inference. One example is biomedical images that reflect known relationships and properties of anatomy. Another is aerial views of certain cultural areas such as industrial sites or airports. There is nothing inherently 2-D in the method. With the TEA system we currently use table settings. Figure 1 shows a typical table top scene.

We assume a spatially-varying sensor, which makes the “where to look next” question even more central. A spatially variant sensory device can be created in several ways. Anthropomorphic VLSI sensors are being constructed (e.g. [Rojer and Schwartz, 1990; Tistarelli and Sandini, 1990]). Software and hardware resolution pyramids are a classic technique (e.g. [Burt, 1988]). Another choice is simply to use two cameras with two different focal lengths. Finally, a simple electronic window could be used. In the TEA system the peripheral image is a low-resolution image of the entire field of view from one camera angle, and the fovea is a small high-resolution image (i.e. window) that can be selectively moved within the field of view.
We assume the system cannot view the entire scene at once. Often a camera movement must be made to an area of the scene that has not been viewed before. The target location of such a camera movement must be determined via relations with other portions of the scene for which image data is (or previously has been) available. Following a camera movement the fovea is centered in the field of view, but afterwards the system can move the fovea within the field of view. The target location for a fovea movement is always within the field of view so it can be determined either from peripheral image data or by relations with other portions of the scene.

Our goal is to support many different visual tasks efficiently. Each task can be specified by asking a question about the scene: Where is the butter? Is this breakfast, lunch, dinner, or dessert? Is this an informal or fancy meal? How far has the eating progressed? Is this table messy? We are particularly interested in more qualitative tasks.

3 A Composite Bayes Net

This section introduces the basic Bayes Net model using a simple "monolithic" network. Next we present each of four specific kinds of networks, including the expected area net. Last, we combine several of these networks to create a "composite" network, and we present an algorithm for computing beliefs in a composite network.

3.1 Basic Bayes Net

A Bayes net is a way of representing the joint distribution of a set of variables in a way that is especially useful for knowledge representation. For example, Figure 2 shows a highly simplified Bayes net that describes a place setting for a meal. Nodes in the net represent variables. Here, nodes drawn with solid lines denote parts of a place setting. The variable setting has four possible values (breakfast, lunch, dinner, dessert) that denote the respective types of meals. A plate can be either paper or ceramic. Links in the net represent conditional probabilities, for example the link from setting to napkin represents \( P(\text{napkin} \mid \text{setting}) \), which says whether a napkin is expected at each of the possible meals. The net as a whole represents the joint distribution of the variables, in this case, \( P(\text{setting}, \text{napkin}, \text{plate}, \text{utensil}) = P(\text{napkin} \mid \text{setting})P(\text{plate} \mid \text{setting})P(\text{utensil} \mid \text{setting})P(\text{setting}) \). Consult the references for a complete introduction to Bayes Nets, for example [Pearl, 1988; Henrion, 1990; Neapolitan, 1990; Shafer and Pearl, 1990; Dean and Wellman, 1991; Peot and Shachter, 1991].

The Bayes net formalism also includes a form of inference. Each node in the Bayes net has an associated quantity called "belief", which says how likely each of the node's value are, given all the evidence collected so far. Formally, belief in the values for node \( X \) is defined as \( \text{BEL}(x) = P(x \mid e) \), where \( e \) is the combination of all evidence present in the net. The evidence consists of "evidence reports", which are produced by evidence gathering actions. An action produces evidence that directly supports (affects) the possible values of a particular node (i.e. variable) in the net, and the evidence report is attached directly to that node. Specifically an evidence report contains a score for each possible value of the Bayes net node it is attached to. (The next section will present details about actions,
(a) The Bayes net structure.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>setting</td>
<td>breakfast, lunch, dinner, dessert</td>
</tr>
<tr>
<td>napkin</td>
<td>present, notPresent</td>
</tr>
<tr>
<td>plate</td>
<td>paper, ceramic</td>
</tr>
<tr>
<td>utensil</td>
<td>fork, knife, spoon</td>
</tr>
</tbody>
</table>

(b) Possible values for variables in the Bayes net.

(c) Some link probabilities in the Bayes net.
Not shown: $P(\text{napkin} \mid \text{setting})$, $P(\text{utensil} \mid \text{setting})$, $P(\text{cup} \mid \text{setting})$, $P(\text{bowl} \mid \text{setting})$.

Figure 2: Very simple example of a “monolithic” Bayes net that describes a place setting for a meal.
evidence reports, and action nodes in a Bayes net.) A fundamental problem is to propagate the effect of an evidence report to all other nodes in the net, i.e. to compute new belief values for each node in the net.

A variety of solutions have been developed for incorporating a single piece of evidence into the net and for propagating its effect to all other nodes in the net [Pearl, 1988; Henrion, 1990; Peot and Shachter, 1991]. An elegant solution exists when the network is a tree [Pearl, 1988]. Variations on this solution have been used for polytrees and for general networks [Pearl, 1988]. Other methods for general trees also exist [Lauritzen and Spiegelhalter, 1988; Jensen et al., 1990]. A new result, the revised polytree algorithm [Peot and Shachter, 1991], generally subsumes all the above results. Stochastic simulation approaches are also being developed (see [Henrion, 1990; Peot and Shachter, 1991]).

3.2 Why Use a Composite Net?

A “monolithic” network is a single network that encodes the knowledge required to solve a task. In such a network, different nodes may denote either different kinds of knowledge or knowledge about different things.

Figure 2 is a simple example of a monolithic network. It contains three distinct kinds of nodes, for example: setting denotes a property of the place setting, utensil denotes the kind of utensil, and napkin denotes whether or not a napkin is present. As a result, the conditional probabilities are between variables denoting different kinds of knowledge.

Monolithic networks are acceptable for simple applications, but more complex applications can have very complex knowledge structures that are difficult to maintain in a single monolithic net. A “composite” network organizes nodes according to the characteristics of the knowledge that each node denotes. Each piece is a separate Bayes net, propagating evidence only within itself. For example, our composite network contains pieces for: the sub-part structure of objects in the scene, the expected locations of objects in the scene, and a hierarchy of classifications for each object. These pieces constitute several different kinds of knowledge, and knowledge about several different things in the scene. All of the networks are combined by a single, special network called the task net, which isolates all the task specific knowledge in the system. Although our composite network is composed of only trees, the motivation of creating a composite network is general and applies to non-tree nets as well.

The following sections describe the four kinds of network. Previous vision systems have used PART-OF nets [Levitt et al., 1989] and IS-A trees [Levitt et al., 1989; Chou and Brown, 1990], but the expected area net and task net presented here are new, as is the composite net. The last section describes how we combine the separate networks, and compute belief values in the composite network.

3.3 A PART-OF Net

A PART-OF net, an example of one is shown in Figure 3, models the physical structure of the scene. We assume the scene and all the objects in the scene can be modeled as a hierarchy of parts. All nodes in this net have the same set of possible values: present and
notPresent. The conditional probability on each network link indicates the likelihood that a subpart exists. Otherwise this network follows the model of the basic Bayes net presented in Section 3.1.

### 3.4 An Expected Area (Object Location) Net

An expected area is an area in the scene in which a particular object is expected to be located. Expected areas and geometric relations between objects are modeled by the expected area net. This is a special version of the basic Bayes net presented in Section 3.1.

An expected area net is used here in combination with a PART-OF net. The two networks have the same structure, for example Figure 3. 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. The location of an object in the scene is specified by the two camera angles that will cause the object to be fixated. We currently discretize these angles using a resolution equal to that of the peripheral image. The range of angles corresponds with the range of camera movements that are physically possible.

Each node in the expected area net represents a 2D discrete random variable, the pair of camera angles that defines the location of an object. Picture the probability distribution of this variable (actually, its BEL) as a grid. An area with high values in the grid corresponds to an area in the scene in which the object is expected, within some confidence level. Knowing the location of an object corresponds to a grid that contains a single impulse. The resolution of the grid is low, but locations with much higher resolution can be recovered from the grid: Say the grid contains four non-zero values arranged in a square, then the location is calculated by interpolating between these four values.
Figure 4: Relation maps defining the expected area for a plate and a utensil relative to a setting-area.

Each link in the *expected area* net has an associated conditional probability, \( P(\text{location} | \text{parentLocation}) \). It is unrealistic to enumerate this conditional probability, but specific values of the conditional probability (as required by the belief propagation algorithms) can be reasonably computed using a function. The function is the same for all the links, but it uses a "relation map" that is different for each link. (The root node of a tree has only an *a priori* probability, which we assume is given.)

A relational map maps out the area in which an object, say A, is expected to be located, given the location of another object, say B. Currently, the values in our maps are either zero or a constant, but a map generally may contain a distribution of values. A relation map is always relative to a B object of "unit" dimension, so when a map is used it must be scaled according to the dimensions of object B. The dimensions of object B are known if the object has previously been detected by a visual action. Otherwise the expected dimensions of object B must be used. Once a relation map has been scaled it is convolved with the expected area of object B, thus producing the new expected area of object A.\(^1\) For example, the relation maps defining the expected area for a plate and a utensil relative to a setting-area are illustrated in Figure 4.

The usefulness of the expected area of a node \( X \) can be measured by the fraction \( r_X^l \) of the scene area that it covers. Let \( l \in (0, 1) \) be a confidence level, which usually will be chosen close to 1, like 0.9. Let \( G_X^l \) be the smallest subset of all the grid points \( G_X \) for node \( X \), such that their probabilities add up to \( l \). The value of \( r_X^l \) is the size of the subset \( G_X^l \).

\(^1\) The expected area for node A is actually calculated not from a single node like node B, 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 \( BEL(A) \). In general it will be 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 the relations are "could-be", and the maps are essentially unioned by the belief calculation. The belief calculation could presumably be modified to perform other types of combinations.
divided by the size of the set $G_X$. $r_X^X = 1$ means that node $X$'s object 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 $r_X^X$ will approach zero. The cost of a visual operation is often proportional to the area over which it must be applied. If a visual action associated with node $X$ only processes image data in the expected area, then the cost of executing that action must be scaled by the factor $r_X^X$.

The topic of modeling relations in a Bayes net has been mentioned in the papers from Dean's and Levitt's groups. We have formalized a version of this problem, presented a solution technique, and applied it within the TEA system to limit the areas in a scene that a vision module processes.

### 3.5 An IS-A Tree

A taxonomic hierarchy models one random variable that can have many mutually-exclusive values, where the possible values can be organized into subsets forming a tree hierarchy. The full hierarchy of subsets can be represented as a tree structured graph.

For example, consider the following set of mutually exclusive values for the random variable named object: \{plate-paper, plate-real, fork, knife, spoon, napkin, cup-win, cup-mug, cup-cocktail, cup-paper, pot, bowl-yellow, bowl-black, butter, table, tablecloth\}. A hierarchy would include subsets of these values, for example one subset is \{fork, knife, spoon\}, which we will call the utensil subset. The full hierarchy of subsets is shown as a tree in Figure 5.

A special version of a Bayes net has been developed to incorporate the mutual-exclusivity constraint [Pearl, 1988; Chou and Brown, 1990]. Each leaf node $h_i$ in the tree has a belief $BEL(h_i)$. Since an intermediate node represents the disjunction of all its children, its $BEL$ value is the sum of its children's $BEL$ values. Evidence $e$ that supports a subset $S$ of the leaf nodes is added to the network by weighting the leaves' $BEL$ values by a likelihood ratio. The new $BEL$ values of the leaves are computed as

$$BEL'(h_i) = KW_iBEL(h_i)$$

where $K$ is a normalizing factor so the new $BEL$'s sum to one, and $W_i$ is a weight

$$W_i = \begin{cases} \lambda & h_i \in S \\ 1 & \text{otherwise} \end{cases}$$

that is based on the likelihood ratio of the evidence

$$\lambda = \frac{P(e \mid S)}{P(e \mid \neg S)}.$$

Evidence that supports the set $S$ will have $\lambda$ greater than 1 and thus will increase $BEL$ for the members of $S$ proportional to the strength of the evidence. The updated $BEL$ values can be calculated using a global algorithm, as the above equations suggest, or using a propagation-based algorithm [Pearl, 1988].

We use the above kind of Bayes net to model an abstraction hierarchy, called an IS-A tree, for each instance of an object in the scene. Figure 5 is an example.
Figure 5: An IS-A Bayes net (tree).
3.6 The Task Net

One of our scientific goals is to make a tight formal and practical coupling between “task specific knowledge” and visual actions. Task specific knowledge is contained in a task net (for example, Figure 6), and is thus distinguished from other types of knowledge. The task network is a basic Bayes net of the type presented in Section 3.1.

One feature of task knowledge is that subtask nodes could be shared by several tasks. Questions such as “Is this a fancy meal?” may be answered using a range of image clues. Some simple tasks, such as “Where is the butter?” do not require a task net since they only involve one particular node in a net.

3.7 Calculating $BEL$ through a Composite Network

The organization of the composite network that we are using is shown in Figure 7. It consists of four separate net structures: a PART-OF net, an expected area net, IS-A trees, and a task net. An IS-A tree is associated with each node in the PART-OF net that corresponds with an object.

The task net serves to combine the $BEL$ values contained in all the other networks, in a manner that reflects the knowledge and information needed to solve the specific task for which the task net was created. Here we describe how information can be transferred from the other nets into the task net, so that $BEL$ values can be calculated in the task net. This calculation will be used in a later section to calculate a utility for an evidence gathering action in one of the other networks.

$BEL$ values are calculated in the composite net as follows: (1) Propagate belief in each of the networks, except the task net. In other words, each of the separate networks in the
composite net, except the task net, maintains its BEL values independently of the other networks. (2) Construct packages of BEL values from the other networks for transfer to the task net. A package is treated as an evidence report that is attached to a node in the task net. A package serves to transfer belief from one knowledge domain to another. (3) Propagate belief in the task network.

The construction of a package for transporting BEL values is somewhat domain specific, in particular it depends on the values of the node in the task net that the package is attached to. We use one general kind of package that combines belief about the presence of an object in the scene with belief about the detailed classification of the object. Let the object's node in the PART-OF net have the following belief values.

\[
\begin{align*}
BEL(\text{present}) &= \alpha \\
BEL(\text{notPresent}) &= 1 - \alpha
\end{align*}
\]

And, in the object's IS-A tree, let the subset of desired classifications for the object have the following belief values.

\[
\begin{align*}
BEL(\text{class}_1) &= \beta_1 \\
BEL(\text{class}_2) &= \beta_2 \\
... \\
BEL(\text{class}_d) &= \beta_d
\end{align*}
\]

Assuming that a node in the task net needs information about this object, a package is
constructed that contains the following values.

\[
\begin{pmatrix}
\alpha \beta_1 \\
\alpha \beta_2 \\
\vdots \\
\alpha \beta_d \\
1 - \alpha \sum_d \beta_i
\end{pmatrix}
\]

This package is attached as an evidence report to the corresponding node in the task net, whose variable in this case would have the following possible values: class1, class2, ... classd, and notPresentOrOther.

For example, if the object was that corresponding to the \( r-\text{utensil} \) node in the \( \text{PART-OF} \) net, the subset of desired classifications for that object is \( \{\text{fork, knife, spoon}\} \), and the package is: \( (\alpha \beta_{\text{fork}}, \alpha \beta_{\text{knife}}, \alpha \beta_{\text{spoon}}, 1 - \alpha(\beta_{\text{fork}} \beta_{\text{knife}} \beta_{\text{spoon}})) \). This package is attached to the \( \text{utensil} \text{-} \text{type} \) node in the \( \text{task} \) net. The possible values of that node are: \( \text{fork, knife, spoon, and notPresentOrOther} \).

4 Adding Actions to a Bayes Net

This section introduces a set of visual actions that are constructed from several low-level vision modules. Next we describe how visual actions are integrated with a basic Bayes net via special nodes called action nodes. We will see, in the next section, how belief calculations are combined with action nodes to calculate a utility measure (for each action) that captures the value and cost of executing an action while taking all current evidence into account.

4.1 Vision Modules and Actions

All actions in the system are constructed from one or more low-level vision modules. In TEA, each module can operate on either a foveal image or a peripheral image. We do not emphasize special-purpose visual modules that are guaranteed to perform well in a specific application. Rather we desire a framework in which any module with quantifiable performance can be incorporated.

Examples of some low-level vision modules are:

- **Color histogram matching.** A color histogram of an example of the object is matched against a new input image by computing the percent of model histogram values that can be located in the input image histogram. The location of the object in the image is determined by “backprojecting” the largest histogram bins to the locations in the image where they came from.

- **Grayscale template matching.** Scan a model template, a window of Sobel edge magnitude data, over the input (Sobel edge magnitude) image and compute at each scan location the mean squared error (mse) between the template and the image data (under it). The minimum value of the mse over all scan locations measures how well the template matches against the input image. Multiple instances of an object can be detected by picking all locations within a few percent of the minimum mse value.
Table 2: Summary of some examples of visual actions and action nodes.

- **Hough transform for circles.** A standard Hough transform detects circles in a specified range of radius.

- **Object models using straight lines and arcs.** Simple object models are matched to straight line segments and curved arcs extracted from the image.

One or more vision modules may be used in a visual action. Each kind of object will usually have several actions associated with it. As an example, the Bayes net in Figure 2, used in one of our experiments with TEA-0, contains 11 visual actions related to 5 objects. We are currently constructing the TEA-1 system, which uses the composite network in Figure 7. That system will have approximately 25 actions when complete. Some examples of visual actions are summarized in Table 2. Following is a more detailed summary of the actions related to plates:

- **per-detect-plate.** Use a model grayscale template to detect the presence of a plate in the peripheral image. The resulting evidence report contains scores for the following possible values: `notPresent` and `present`. The location of the detected plate is saved.

- **per-hough-plate.** Use a Hough transform for plate-sized circles to detect a plate in the peripheral image. The resulting evidence report contains scores for the following possible values: `notPresent` and `present`. The location of the detected plate is saved.

- **per-class-plate.** The plate location must have been determined previously. Use a color histogram to classify the plate as paper (blue) or ceramic (green), using a window centered on the plate in the peripheral image. The resulting evidence report contains scores for the following possible values: `paper` and `ceramic`.

- **fov-class-plate.** Move the fovea to the plate and proceed as for per-class-plate but with foveal data.

Complex actions like per-detect-plate above might be decomposed into simpler actions (e.g. detect X using peripheral image, move peripheral window, classify X using peripheral window data, move fovea, classify X using foveal data, move camera).
Some actions have preconditions that must be satisfied before they can be executed. For example, an action can require that the location of its object be known (or be an impulse in the expected area map). per-class-plate is like this since it requires the location of the plate (e.g. from per-detect-plate or per-hough-plate). An action can require that the object’s expected area (within a given confidence level) be small enough to fit a particular size window in the peripheral image or to fit the fovea. Other preconditions are also possible.

4.2 Action Nodes in a Basic Bayes Net

Every node in a Bayes net that has an associated evidence gathering action will have an action node connected to it. An action node that is connected to a Bayes net node \( X \) represents a random variable that is a visual action’s “evidence report”. The evidence report contains a score for each possible value of the Bayes net node \( X \). The conditional probability on the link connecting the action node with its parent describes the expected performance of the action.

The evidence propagation algorithm of the Bayes net makes an action’s evidence report affect the \( \text{BEL} \) values of the parent node \( X \) and of all the other nodes in the net. Before the action is executed, the action node is a “chance” node like most of the nodes in the net, and the node contains the \( \text{BEL} \) of the evidence report. After an action is (successfully) executed the action node is changed to be an “instantiated” node (see [Pearl, 1988]) and set to the value of the evidence report.

As an example, Figure 8 shows our basic Bayes net (from Figure 2) with action nodes drawn in using dotted lines. The per-class-utensil action might generate the following evidence report \((4.9,1.4,2.7)\), which contains the scores for each of the objects, \((\text{fork, knife, spoon})\). The impact of this evidence on the parent (and other nodes in the net) is dependent on the action node’s conditional probability, \( P(x_{\text{evidence}}|x_{\text{true}}) \), which has the following values.

<table>
<thead>
<tr>
<th>true (parent)</th>
<th>evidence</th>
<th>fork</th>
<th>knife</th>
<th>spoon</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{\text{true}} )</td>
<td>( x_{\text{evidence}} )</td>
<td>0.600</td>
<td>0.100</td>
<td>0.250</td>
</tr>
<tr>
<td>fork</td>
<td>0.150</td>
<td>0.800</td>
<td>0.050</td>
<td></td>
</tr>
<tr>
<td>knife</td>
<td>0.250</td>
<td>0.100</td>
<td>0.700</td>
<td></td>
</tr>
</tbody>
</table>

Among other things, the above table says that if the object is truly a fork, the action gets it wrong 40 percent of the time, saying the object is a spoon or knife in 25 and 15 percent of the cases.

4.3 Using Actions with a Composite Net

An action in a system that uses a composite net generally can post evidence reports to several different nets. Below we summarize how evidence and action nodes are handled in each kind of net. All the nets do it similarly, like it is described above for the basic Bayes net, with only minor variations. The exception is the IS-A net, which is different.
PART-OF Net: Evidence is handled as in the basic Bayes net. All evidence reports are of the same form, containing scores for the possible values: not Present and present.

Expected Area Net: Evidence is handled as in the basic Bayes net. The conditional probability of an action node may be specified using a parameterized 2D distribution, such as a bivariate Gaussian distribution. Vision modules are often very good at estimating the location of an object so this distribution may be very close to an impulse.

IS-A Tree: Action nodes cannot be used in an IS-A tree. Instead, evidence is injected into the tree via the weight $W_t$ associated with each leaf node $h_i$ (recall Section 3.5). If it is necessary to estimate the expected impact of an evidence report, it is always possible to use some kind of expected value for the $\lambda$ values used to compute $W_i$.

Task Net: Evidence is handled as in the basic Bayes net. But actions generally should not post evidence directly to the task tree, since such actions would be task-specific, something we want to avoid.

Examples: The per-detect-utensil action posts evidence both to the r-utensil node in the PART-OF net and to the object's corresponding IS-A net, supporting equally the nodes in the subset named utensil. The per-class-utensil action posts evidence only to the object's corresponding IS-A net, supporting the nodes under utensil to varying degrees. The fov-class-utensil action is similar but it posts different evidence than per-class-utensil (supposedly better, which would be reflected by the likelihood ratios).
5 Calculating an Action’s Utility

This section develops a utility function, which measures the expected usefulness of an action if it were to be executed. We introduce the basic form of the utility function, a value divided by a cost, and how the value and cost are calculated. Next the calculation of an action’s value is extended to the more complicated case of a composite Bayes net. Last we extend the utility function to look one step into the future.

5.1 A Basic Utility Function

Let the action node $\alpha$ have the node $A$ as its parent. The utility $U(\alpha)$ of an action $\alpha$ is of the form

$$U(\alpha) = \frac{V(\alpha)}{C(\alpha)}.$$ 

$C(\alpha)$ is the cost of executing the action:

$$C(\alpha) = r_A^l C_0(\alpha).$$

$C_0(\alpha)$ is the execution time of action $\alpha$ on the entire peripheral image (or the foveal image, if $\alpha$ is a foveal action). $r_A^l$ is the fraction of the image covered by the expected area of the object associated with node $A$, when a confidence level of $l$ is used (recall Section 3.4). Before any actions have been executed, no objects have been located, and so all $r_A^l$ values are close to 1.0. Over time, as other objects in the scene are located and as more and tighter relations are established, the value of $r_A^l$ will approach zero.

$V(\alpha)$ is meant to be the value of the action, how useful it is for achieving the task’s goal. All actions in a computer vision system are information gathering actions, so we based our value function on a fundamental measure of information, Shannon’s measure of average mutual information (see, e.g. [Proakis, 1983; Pearl, 1988]):

$$I(x, y) = \log_2 \frac{P(x | y)}{P(x)}.$$ 

$I$ describes the information content about the event $x$ that is provided by the occurrence of the event $y$ (and vice versa). If $X$ and $Y$ are independent then $I = 0$. If $X$ and $Y$ are fully dependent then

$$I(x, y) = I(x) = \log \frac{1}{P(x)} = -\log P(x).$$

Called self information, this is (by definition) the information of the event $x$. Note that higher probability events provide less information.

The average mutual information between two random variables is

$$I(X, Y) = \sum_x \sum_y P(x, y) I(x, y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}.$$ 

The value of $I$ is always $\geq 0$. The average mutual information can also be written as

$$I(X, Y) = H(X) - H(X | Y)$$
where

\[ H(X | Y) = \sum_y P(y) H(X | y) \]

\[ H(X | y) = -\sum_x P(x | y) \log P(x | y) \]

and

\[ H(X) = -\sum_x P(x) \log P(x). \]

\( H(X) \) is the average self information and is also called the entropy of \( X \). \( H \) measures the cost of removing uncertainty. \( H(X) \) is a larger cost since it assumes nothing about \( Y \). Knowledge of \( Y \) will usually remove uncertainty, making \( H(X | Y) \) smaller than \( H(X) \).

When \( X \) and \( Y \) are nodes in a (single) Bayes net, the probabilities can be replaced by belief:

\[ I(X, Y) = \sum_x \sum_y B E L(x, y) \log \frac{B E L(x, y)}{B E L(x) B E L(y)} \]

where

\[ B E L(x, y) = B E L(x | y) B E L(y). \]

The values of \( B E L(x) \) and \( B E L(y) \) are respectively available at nodes \( X \) and \( Y \) in the Bayes net. The values of \( B E L(x | y) \) can be calculated by temporarily instantiating node \( Y \) to each of its values, propagating beliefs, and taking the resulting value of \( B E L(x) \) as \( B E L(x | y) \) [Pearl, 1988].

So, if a system uses a single monolithic Bayes net, the value of an action \( \alpha \) is:

\[ V(\alpha) = I(target, \alpha), \]

where \( \alpha \) refers to the action node for action \( \alpha \), and the task’s goal is represented by the node \( target \). For example, in Figure 8 the task’s goal is \( setting \) and one of the action nodes is \( per\text{-}detect\text{-}plate \).

5.2 Calculating an Action’s Value using a Composite Net

When a composite net is used there are two differences in how a quantity like \( I(target, \alpha) \) is calculated. First, the action \( \alpha \) may post several evidence reports, one to each net. Second, the \( target \) node and the action’s evidence reports are normally in different Bayes nets. The calculation is similar to that above.

Recall that until this point, an action has always generated exactly one evidence report, so our notational convention has been that action \( \alpha \) has an action node named \( \alpha \) which represents an evidence report that we also call \( \alpha \). Now, let action \( \alpha \) post several pieces of evidence \( \alpha_1, \alpha_2, \ldots, \alpha_g \) to the composite net, each component network receiving at most one piece of evidence. Let \( \hat{\alpha} \) be the cross product of all the possible combinations of the separate evidence reports. The value of action \( \alpha \) is calculated as

\[ V(\alpha) = I(target, \hat{\alpha}), \]
where \(I(\text{target}, \hat{a})\) is calculated using
\[
I(X, \hat{Y}) = \sum_x \sum_{\hat{y}} \text{BEL}(x, \hat{y}) \log \frac{\text{BEL}(x, \hat{y})}{\text{BEL}(x) \text{BEL}(\hat{y})}
\]
and
\[
\text{BEL}(x, \hat{y}) = \text{BEL}(x | \hat{y}) \text{BEL}(\hat{y}).
\]

The values of \(\text{BEL}(x | \hat{y})\) can be calculated by temporarily instantiating the evidence reports to the values in each possible cross product \(\hat{y}\), propagating beliefs, and taking the resulting value of \(\text{BEL}(x)\) as \(\text{BEL}(x | \hat{y})\). Note that beliefs are transferred from the other nets to the task net via evidence packages, as described in Section 3.7.

The value of \(\text{BEL}(x)\) is available at node \(X\) in the Bayes net, but calculating a value for \(\text{BEL}(\hat{y})\) poses a problem. If the components of the \(\hat{y}\) cross product are assumed independent, \(\text{BEL}(\hat{y})\) can be calculated as \(\text{BEL}(\hat{y}) = \prod_y \text{BEL}(y)\). While convenient, this conflicts with the assumption, in the calculation of \(\text{BEL}(x | \hat{y})\), that the components of \(\hat{y}\) are partially dependent. Alternatively, a consistent independence assumption about the components of the cross product can be maintained by computing \(I(X, Y)\) independently for each component of the cross product, and summing the results, \(I(X, \hat{Y}) = \sum_y I(X, Y)\). We currently use the former method.

5.3 Accounting for Future Value

It is important to "look ahead" at the future impact of executing an action. Therefore we use the following "lookahead" utility function.

\[
U(\alpha) = \frac{V(\alpha) + V(\beta)}{C(\alpha) + C(\beta)} + H \sum_{X \in \text{Net}} \Delta U(X)
\]

where

\[
\beta = \arg \max_{\gamma \in \text{LocPre}(A)} \frac{V(\gamma)}{C(\gamma)}
\]

Recall that action node \(\alpha\) has the node \(A\) as its parent.

The first term in equation (1) accounts for the future value of establishing the location of an object. Action \(\alpha\) might detect and locate an object, but not provide any information \((I = 0)\) toward the task node in the Bayes net, however it does locate the object and thereby satisfy the preconditions of other actions that in turn will provide information useful for accomplishing the task. The interpretation of the first term in equation (1) is as follows. \(\text{LocPre}(A)\) is the set of actions associated with node \(A\) that have a precondition satisfied by executing action \(\alpha\). Let \(\beta\) be the "best" of the actions in that set. The new utility of action \(\alpha\) is an average over both \(\alpha\) and \(\beta\), more specifically an average of the value and cost of the two actions \(\alpha\) and \(\beta\).

Action \(\alpha\) may decrease the size of the expected area of other objects. Smaller expected areas will make it easier in the future to execute actions for those objects. The second term
in equation (1) accounts for the future impact of making these expected areas smaller. Each node in the network contributes a term \( \Delta U(X) \) to the utility:

\[
\Delta U(X) = \max_{\gamma \in \text{Actions}(X)} \left[ \frac{V(\gamma)}{r_X C_0(\gamma)} - \frac{V(\gamma)}{s_X C_0(\gamma)} \right]
\]

\[
= \left[ \frac{r_X}{s_X} - 1 \right] \max_{\gamma \in \text{Actions}(X)} U(\gamma)
\]

This term is the increase in utility of the best action of node \( X \). Recall that \( r_X \) is the fraction of the entire scene covered by the expected area of node \( X \)'s object, when a confidence level of \( l \) is used. \( s_X \) is like \( r_X \) except it assumes that the location of node \( A \)'s object is known. It is computed in a manner similar to \( BEL(x \mid y) \) above: The expected area node corresponding to node \( A \) is temporarily instantiated as a point at the center of its expected area mass, beliefs are propagated in the expected area net, and the resulting value of \( r_X \) is taken as \( s_X \).

\( H \in (0,1) \) is a gain factor that specifies how much to weigh the second term in equation (1) relative to the first term.

Equation (1), where the action values are calculated using a composite net, is the utility function actually used in the TEA system.

6 Experimental Results

6.1 The TEA-O System and Experiments

This section describes TEA-O, an initial implementation that follows the technical approach outlined above but is simplified in many ways. TEA-O demonstrates the basic approach of using Bayes nets, the basic idea of relation maps and expected areas, and the lookahead utility function.

TEA-O works in a simplified domain, a single place setting. The entire scene can be viewed in one image so the system does not use camera movements. The task is further simplified by assuming that the scene could contain only a single instance of each possible object. Figure 9 shows a typical scene.

TEA-O uses the basic decision loop presented in Section 2, uses a single monolithic Bayes net tree, and solves tasks that can be posed via the root node of a monolithic tree.

The system uses visual actions like those presented in Section 4 with one exception. The per-hough-plate and per-class-plate actions are combined into a single action called per-plate. Bowl and cup actions are similarly combined into the actions per-bowl and per-cup. Preconditions are handled in an ad hoc way: while the preconditions of an action are not satisfied the cost of that action is set to a very large value. A foveal action requires a fixation location and therefore has a precondition that one of the peripheral actions has successfully been run. The set of action nodes attached to a given Bayes net node has an associated precondition function that is stored at the Bayes net node.

Relation maps and expected areas are maintained in an ad hoc way in each node of the monolithic Bayes net, rather than in a separate expected area net. Each node in the net is
allowed to have a relation map (only) with any of its siblings. If several siblings suggest expected areas for one node, all the expected areas are unioned together. Expected areas can not be propagated to other nodes.

TEA-0 uses the utility function in equation (1), where action values are calculated for a monolithic net (as in Section 5.1) and the simplified version of expected areas is used.

**Experiment**

The problem is to decide whether a place setting is set for breakfast, lunch, dinner or dessert. A relationship between the possible objects and the type of meal was contrived and encoded as the Bayes net model shown in Figure 8. The Bayes net contains only one relation map that defines the expected area for a utensil relative to a plate. The scene shown in Figure 9 is used in this example. The goal is to obtain high values for $BEL(\text{setting})$. The values of $BEL(\text{setting})$ before any actions have been executed are:

<table>
<thead>
<tr>
<th>TASK BELIEF</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>blast</td>
<td>0.100</td>
</tr>
<tr>
<td>lunch</td>
<td>0.300</td>
</tr>
<tr>
<td>dinner</td>
<td>0.500</td>
</tr>
<tr>
<td>dessert</td>
<td>0.100</td>
</tr>
</tbody>
</table>

The execution of the system is sketched in the following (edited) sequence of tables. In each row, the ACTION UTILITY table lists the available actions and their utility values. The action at the top of this table is then executed, producing the evidence report shown in
the EVIDENCE table. This evidence is propagated through the entire network, producing the new values of BEL(setting) shown in the TASK BELIEF table.

Finding and classifying the plate causes only a small (immediate) change in the task belief:

<table>
<thead>
<tr>
<th>ACTION UTILITY</th>
<th>EVIDENCE</th>
<th>TASK BELIEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>per-plate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>per-detect-napkin</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>per-detect-utensil</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>per-cup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>per-bowl</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Knowing the location of the plate gives an $r_{utensil}$ value of 0.270, reducing the cost of the utensil detection action. $U(\text{per-detect-utensil})$ is increased since $C(\alpha)$ in equation (1) changes from $1.000C_0(\alpha)$ to $0.270C_0(\alpha)$, in addition $V(\beta)$ is affected by the evidence from the per-plate action.

Note that the list of actions (below) now includes fov-class-plate since its precondition of knowing the plate’s location is now satisfied, however its utility is very small. The next action, checking if a napkin is present in the scene, has a large effect of the task belief:

<table>
<thead>
<tr>
<th>ACTION UTILITY</th>
<th>EVIDENCE</th>
<th>TASK BELIEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>per-detect-napkin</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>per-detect-utensil</td>
<td>0.01329</td>
<td></td>
</tr>
<tr>
<td>per-cup</td>
<td>0.012329</td>
<td></td>
</tr>
<tr>
<td>per-bowl</td>
<td>0.000544</td>
<td></td>
</tr>
<tr>
<td>fov-class-plate</td>
<td>0.164644</td>
<td></td>
</tr>
</tbody>
</table>

The system continues, first finding the utensil without actually classifying it (not changing the task belief). Next, executing per-class-utensil and then per-cup do effect the task belief. However the utensil, actually a fork, is scored first as a knife and only second as a fork by the per-class-utensil action (using low resolution peripheral image data). Next, the utensil is re-examined using better quality foveal image data:

<table>
<thead>
<tr>
<th>ACTION UTILITY</th>
<th>EVIDENCE</th>
<th>TASK BELIEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>fov-class-utensil</td>
<td>0.003264</td>
<td></td>
</tr>
<tr>
<td>per-bowl</td>
<td>0.000762</td>
<td></td>
</tr>
<tr>
<td>fov-detect-napkin</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>fov-class-plate</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>fov-class-cup</td>
<td>0.000000</td>
<td></td>
</tr>
</tbody>
</table>

The task beliefs are further helped by finding what type of bowl is in the scene, but after that the remaining actions produce very small changes in the task belief. The final task belief values correctly indicate that the place setting is most likely set for a dinner meal.

6.2 The TEA-1 System and Experiments

We are currently implementing TEA-1, a complete version of the system presented in this paper. We plan to solve two tasks initially. The “Find the butter” task requires both high belief in the butter object’s existence and a small expected area for the butter object’s
location. The "Is this a fancy meal?" task requires high belief in the root of the task tree shown in Figure 6.

7 Wrap Up

This paper has presented TEA, a task-oriented computer vision system that uses Bayes nets and a maximum expected utility decision rule. It shows how to combine several Bayes nets into a composite net that permits the utility of visual actions to be calculated relative to a specific task. One component of the composite net is an expected area net, which is a new method for using geometric relations with a Bayes net.

The utility of a visual action incorporates many features: a mutual information measure between nodes in the Bayes nets, execution cost proportional to the expected area of an object in the scene, and the expected future benefits that are enabled by the action.

The combination of the composite net and the utility function makes the system task-oriented, enabling TEA to decide where to look and what visual action to use to gather information sufficient to solve the task. TEA's solution uses expected area's to decide about camera and fovea movements, and to limit the portions of an image that a vision module processes.

We are pursuing two main streams of work. One stream develops the TEA systems, a progression of systems that support increasingly sophisticated task-oriented vision by providing solutions to the "where to look next" problem. The second stream of work uses and extends the TEA framework to explore broader and more advanced issues in task-oriented vision, which we call the "how to look" problem: foveal - peripheral vision algorithms, qualitative visual tasks, limited-context vision algorithms that gain in robustness or accuracy by being applied in well-understood circumstances, and incremental visual actions whose results monotonically improve as more time is spent on them. Our idea of a true task-oriented vision system will be achieved by bringing together solutions to the "where to look next" and the "how to look" problems.

7.1 "Where to Look Next"

TEA-1. TEA-1 is the system described in this paper. Its main features are the expected area net, the addition of multiple interacting trees, which permit the system to solve more complex tasks, and projecting utility calculations through the task tree. TEA-1 will work with scenes large enough to require physical camera movements and thus expected areas for unseen areas of space. We are currently implementing TEA-1. We are also implementing more sophisticated (but still simple) object recognizers, using relational structures of line segments for models.

TEA-2. TEA-0 and TEA-1 are "myopic", making decisions by only looking one step ahead. The anticipatory utility function is an improvement, trying to pack look-ahead into the utility of a single action. Ultimately our problem involves full-scale planning, in which sequences of actions are evaluated as to their expected utility. We intend to develop a simple planning system (for computer vision) using Bayes nets. We do not propose "planning
research" *per se*, but rather shall likely use some STRIPS-like planning algorithm. The idea is to substitute a search in action space rather than to try to pack all the intelligence into a (quasi-static) utility function.

7.2 "How to Look"

*Limited-Context Vision Algorithms.* One claim of this work is that vision algorithms can be more robust and reliable if they are known to work in a limited context. For example, TEA can use simple color histograms for object identification only because it has foveated a small area of the image previously. Similarly restricting input to a small volume of space means geometric hashing can work more reliably. We want to explore limited context effects that arise naturally in task-oriented vision when the vision problem is known to be simplified or better specified than normal (by camera actions, foveal processing, and generally by satisfaction of preconditions).

*Incremental actions.* We want to investigate vision modules that can run for different periods of time, improving their results the longer they run (e.g. some scale space algorithms, multi-feature classifiers, and anytime algorithms [Dean and Wellman, 1991]). Such vision actions are generalizations of TEA’s peripheral-foveal actions which produce a peripheral result at one cost and follow it up with a foveal action for a further cost. An evidence/time function can quantify the incremental benefit of such an action. New control strategies should then emerge, such as running a set of incremental actions cyclically to attain the maximum evidence per unit time from the set.

7.3 Task-Oriented Vision

Our idea of a true task-oriented vision system will be achieved by bringing together solutions to the "where to look next" and the "how to look" problems.

*Multiple Tasks.* We plan to solve multiple tasks in any given domain using the same set of visual actions. This exercise will test the generality of our knowledge representations and visual actions and probably encourage us to extend and modify both. Also we expect to encounter interesting new problems for visual actions used in answering qualitative questions such as “Is this desk messy?”.

*Multiple Domains.* We believe that a task-oriented vision system should be verified using more than one domain. We shall seek out other domains. A possible domain is model trains to be monitored on a more or less complex system of tracks. Another is monitoring or searching the laboratory space in 3-D and performing head movements as well as camera movements, and ultimately dynamic scenes. Medical images are another possibility emphasizing reliability as opposed to active vision. Expanding the domains will doubtless mean that visual actions need to be re-engineered and improved to apply more generally. Difficulties in encoding or coping with new domains will motivate extensions and modifications to our formalisms. New domains may necessitate the use of more complex knowledge representations, in particular non-tree Bayes nets.
Acknowledgements

Peter von Kaenel worked on the Hough transform for circles and is currently building the modules that match models using straight lines and circular arcs. He also built several of the visual actions.
References


