Control of Selective Perception Using Bayes Nets and Decision Theory

Raymond D. Rimey and Christopher M. Brown
The University of Rochester, Computer Science Department, Rochester, NY 14627

Abstract

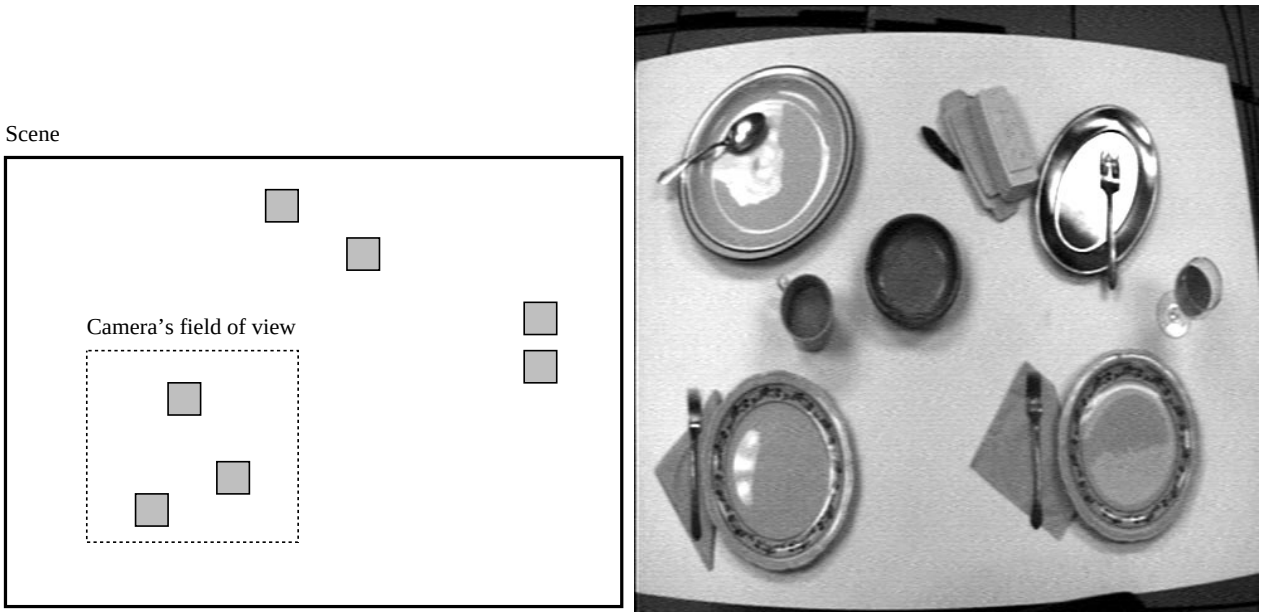
A selective vision system sequentially collects evidence to answer a specific question with a desired level of confidence. Efficiency comes from processing the scene only where necessary, to the level of detail necessary, and with only the necessary operators. Knowledge representation and sequential decision-making are central issues for selective vision, which takes advantage of prior knowledge of a domain's abstract and geometrical structure (*e.g.* "part-of" and "adjacent" relationships), and also uses information from a scene instance gathered during analysis. The TEA-1 selective vision system uses Bayes nets for representation, benefit-cost analysis for control of visual and non-visual actions, and its data structures and decision-making algorithms provide a general, reusable framework. TEA-1 solves the T-world problem, an abstraction of a large set of scene domains and tasks. Some factors that affect the success of selective perception are analyzed by using TEA-1 to solve ensembles of randomly produced, simulated T-world problems. Experimental results with a real-world T-world problem, dinner table scenes, are also presented.

Keywords: active vision, Bayesian belief network, decision theory, influence diagram, selective perception.

1 Introduction

1.1 Selective Perception

An *active* vision system explicitly controls its vision sensor; this can lead to robust low-level algorithms, but also calls for resource-allocation decisions (*e.g.* where to point the sensor): the high-level control system that allocates limited resources becomes a fundamental issue. One constraint on resource allocation is that of *purposiveness*; the system can attempt specific finite tasks rather than general, open-ended tasks like "reconstruct the 3-D world from this image sequence". The task defines the goal that the high-level control system, within the constraints of the limited resources, is trying to achieve by processing the image data. Purposive vision allows a scene to be *selectively* processed, *i.e.* processed only where necessary, to the level of detail necessary, and with only the necessary operators. Selective scene processing relies on prior knowledge, which allows inferences about scene content. By definition, selective vision takes place in a semantic and geometric context that limits



(a) (b)

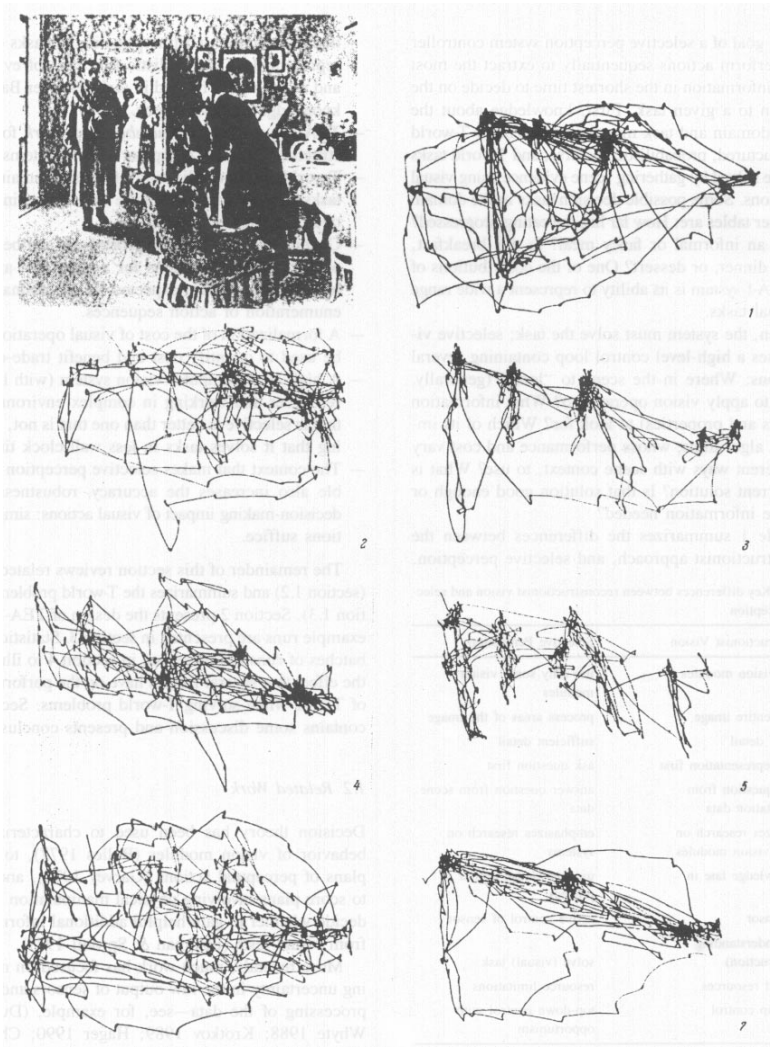
Figure 1: Example scenes from two T-world problems. (a) A simulated, abstract, T-world scene. (b) A dinner table scene, a real-world instance of a T-world problem. In both examples, the camera's field of view is significantly smaller than the scene.

the possibilities of image content, and the context allows hypothesis-driven *sufficing vision*: vision algorithms that verify likely hypotheses can be simpler, faster, and more robust than algorithms with no semantic context.

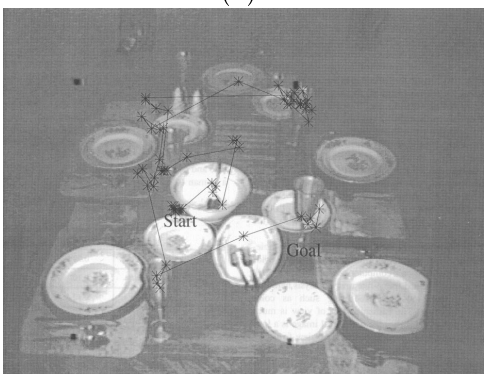
TEA-1 is a software framework that allows design of purposive, selective vision systems. It runs on real and simulated scenes, and uses Bayesian networks and decision theory (using quantified measures of both benefits and costs) to control active and sufficing vision algorithms. TEA-1's domain is T-world (section 1.3), a formalization of a large set of image analysis problems. Figure 1 shows both an abstract T-world scene and a real-world one, a dinner table scene.

Part of the inspiration for designing selective perception systems is that humans seem to exhibit similar characteristics. For example, figure 2 depicts some of the eye movements of a human subject instructed to solve several different tasks. The subject foveates only a fraction of the scene and in each case the objects selected for fixation are ones relevant to solving the given task. Further, "irrelevant" features are often not remembered: the subject does not seem to be analyzing the whole scene.

The goal of a selective perception system controller is to perform actions sequentially to extract the most useful information in the shortest time to decide on the solution to a given task. First, knowledge about the visual domain and task must be represented. T-world is a structured, probabilistic universe, and T-world tasks must be solved by gathering scene evidence using visual operations. Some possible decision tasks in the domain of dinner tables are: How far has the eating progressed? Is this an informal or fancy meal? Is this



(a)



(b)

Figure 2: (a) Seven records of eye movements by the same subject from the classic work in [Yarbus, 1967]: 1) Free examination of the picture. Before subsequent recording sessions the subject was asked to: 2) estimate the material circumstances of the family in the picture; 3) give the ages of the people; 4) surmise what the family had been doing before the arrival of the "unexpected visitor"; 5) remember the clothes work by the people; 6) remember the position of the people and objects in the room; 7) estimate how long the "unexpected visitor" had been away from the family. (b) Recorded eye movements from a subject instructed to determine how many people are drinking at the dinner table.

Reconstructionist vision	**Selective perception**
use all vision modules	use only some vision modules
process entire image	process areas of the image
maximal detail	sufficient detail
extract representation first	ask question first
answer question from representation data	answer question from scene data
emphasizes research on isolated vision modules	emphasizes research on systems
use knowledge late in process	use knowledge earlier in process
static sensor	active control of sensor
image understanding (reconstruction)	solve (visual) task
unlimited resources	resource limitations
bottom-up control	top-down control with opportunism

Table 1: Key differences between reconstructionist vision and selective perception.

breakfast, lunch, dinner, or dessert? One of the contributions of the TEA-1 system is its ability to represent a wide range of visual tasks.

Then, the system must solve the task; selective vision uses a high-level control loop containing several decisions: Where in the scene to "look" (generally, where to apply vision operations)? What information (objects and properties) to look for? Which of its imperfect algorithms, whose performance and cost vary in different ways with scene context, to use? What is the current solution? Is that solution good enough or is more information needed?

Table 1 summarizes the differences between the reconstructionist approach, and selective perception.

We make the following claims for this work.

- Bayes nets and decision theoretic techniques provide a sound formal basis for representation and control in a selective perception system. In particular, complex, abstract, qualitative tasks can be represented as probabilistic functions of evidence and so are easily framed in *task nets* over Bayesian knowledge representations.
- TEA-1 is a *general, reusable framework* for constructing selective computer vision systems.
- T-world is a formalization of visual domains and tasks for studying some of the basic issues in selective computer perception.
- General decision-making procedures can be based on "goodness" functions for actions that approximate the maximum expected utility attainable by enumeration of action sequences.
- A formalization of the cost of visual operations can be used to quantify cost and benefit tradeoffs.
- A high-level computer vision system (with limited resources and working in complex environments) that is selective is better than one that is not, meaning that it solves tasks in less wall-clock time.

- The context that makes selective perception possible also increases the accuracy, robustness, and decision-making impact of visual actions: simple actions suffice.

The remainder of this section reviews related work (section 1.2) and summarizes the T-world problem (section 1.3). Section 2 presents the design of TEA-1. Two example runs are presented in section 3. Statistics over batches of runs are presented in section 4 to illustrate the effect that several factors have on the performance of TEA-1 when solving T-world problems. Section 4 contains some discussion and presents conclusions.

1.2 Related Work

Decision theory has been used to characterize the behavior of vision modules [Bolles, 1977], to score plans of perceptual actions [Garvey, 1976], and both to score plans involving physical manipulation and to decide whether to gain helpful additional information from visual tests [Feldman and Sproull, 1977].

More recently, much work has focussed on modeling uncertainty in the raw output of sensors and early processing of the data (*e.g.* [Durrant-Whyte, 1988; Krotkov, 1989; Hager, 1990; Chou and Brown, 1990; Wu and Cameron, 1990; Chen and Mulgaonkar, 1992]).

At a higher level, uncertain reasoning has been receiving increasing attention in AI research [Shafer and Pearl, 1990; Dean and Wellman, 1991]. Two recent key developments are Bayes nets [Pearl, 1986] and influence diagrams [Shachter, 1986]. The first large experimental system that used Bayes nets for computer vision was [Levitt, 1986], and that work is continuing, *e.g.* [Agosta, 1991; Mann and Binford, 1992]. Bayes nets have also been applied to mobile robots [Dean and Wellman, 1991; Dean *et al.*, 1990]. Other vision applications include [Jensen *et al.*, 1992; Sarkar and Boyer, 1993]. Dempster-Shafer methods of evidence combination are used along with non-probabilistic geometric models to perform model-based object recognition and localization in [Hutchinson and Kak, 1989].

A foveal/peripheral sensor was used with three controllers in [Bolle *et al.*, 1990] to determine the location of the next fixation. Similar objectives were accomplished with resolution pyramid hardware [Burt, 1988], where foveation was implemented as coarse to fine search through the pyramid. The problem of choosing sensor parameters, given precise geometric and sensor models, to satisfy specified feature detectability constraints is studied in [Tarabanis *et al.*, 1991], and several sensing parameters affecting the efficiency of an object search task are analyzed in [Wixson and Ballard, 1993].

1.3 T-world

Selective vision is applicable in any domain where sufficient scene structure exists that it can be exploited to make vision easier. Example domains are anatomy, road traffic, table settings, aerial views of airports or industrial sites. T-world (figure 1) is an abstraction of vision tasks in which the domain has adequate structure to support selective vision. A detailed formalization of T-world is given in [Rimey, 1993]. Briefly, a T-world scene is a 2-D layout of recursively defined spatial groups of objects, and objects have location, type, and properties. Scene domains are formalized as probabilistic rules governing the spatial

grouping, location, type and properties of objects in a scene. T-world has abstractions of peripheral and foveal sensing, sensor positioning, and visual actions with preconditions and costs. Actions detect objects and obtain values of object properties. The result of an action is probabilistically defined as a function of the object's properties, the field of view, and various parameters like image resolution. A T-world task is to determine the value of a task variable, which is a (probabilistic) function of a subset of the number, type, location and properties of objects in the scene.

In the laboratory [Ballard and Brown, 1992], table setting scenes are viewed from overhead with a color camera mounted on a computer-controlled pan and tilt platform. Pipelined frame-rate image analysis hardware does certain low-level processing such as color histogramming. The camera's field of view is much smaller than the scene; The peripheral image is a low-resolution image of the entire field of view from one camera angle (implemented by subsampling within the frame buffer), and the fovea is a small high-resolution subimage that can be selectively moved within the field of view.

A T-world simulator can generate static or dynamic scenes, and on top of the world state simulator is an action simulator library, which simulates the visual actions available. TEA-1 programs can transparently run with the simulator or laboratory providing input and accepting output. The simulator allows the development of TEA-1 representations and algorithms for domains (*e.g.* dynamic domains) for which visual operators are not yet developed [von Kaenel *et al.*, 1993], and rapid quantitative experiments to characterize the effects of control algorithm choice or other system design parameters.

2 The TEA-1 System

2.1 Knowledge Representation Using Bayes Nets

Knowledge is represented in TEA-1 using Bayes nets. This section introduces Bayes nets and then presents the two different knowledge representation structures that TEA-1 can use: *composite nets* and *two-nets*.

A Bayes net represents the joint probability distribution of a set of variables in a way that is especially useful for knowledge representation and plausible inference [Pearl, 1988; Henrion, 1990; Shafer and Pearl, 1990; Dean and Wellman, 1991; Peot and Shachter, 1991]. Nodes in the net represent variables with (usually) a discrete set of labels, *e.g.* a *cup* node could have labels $(wine, cocktail, mug, paper)$. Directed links in the net represent (via tables) conditional probabilities that a node has a particular label given that successor nodes have particular labels (*e.g.* see Table 2). The graph structure and conditional probability tables are supplied by a human (the knowledge engineer).

The Bayes net formalism also includes a form of inference. 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 can be added to a net in several ways, summarized in section 2.7. The inference mechanism can propagate the effect of evidence and recompute belief values for all other nodes. An elegant propagation solution exists when the net is a tree (TEA-1 uses this algorithm), and variations on this solution have been used for polytrees and for general nets

		meal	
		notfancy	*fancy*
cup	*wine*	0.20	0.60
	cocktail	0.10	0.25
	mug	0.40	0.10
	paper	0.30	0.05

Table 2: An example of a conditional probability, $P(cup \mid meal)$.

[Pearl, 1988]. Other methods for general nets also exist [Lauritzen and Spiegelhalter, 1988; Peot and Shachter, 1991]. Stochastic simulation approaches are also being developed (see [Henrion, 1990; Peot and Shachter, 1991]).

Good introductions to Bayes nets are located in [Pearl, 1986; Pearl, 1988; Dean and Wellman, 1991; Charniak, 1991].

2.2 The Composite Net

A composite net uses four kinds of knowledge structured into separate nets: The PART-OF net and IS-A tree [Chou and Brown, 1990] are standard, and these kinds of Bayes nets where used in [Levitt *et al.*, 1989]. The expected area net and task net presented here are new, as is the composite net that results from linking the four.

The PART-OF net models the physical structure of the scene. All nodes in this net have the same set of possible labels: *present* and *notPresent*. The conditional probability on each network link indicates the likelihood that a subpart exists. Figure 3 shows an example PART-OF net for the table setting domain.

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.

A fundamental requirement for a computer vision system is the ability to model and reason about geometric relations and locations of objects in a scene. This issue has hardly been addressed in previous work using Bayes nets. In TEA-1 we assume a known fixed camera origin. Since the scene is 2D, we can use a global 2D coordinate system. The location of an object in the scene is specified by the location of the object's center in global 2D coordinates, $p = (p_x, p_y)$. A simple equation relates (p_x, p_y) with the two camera angles, $(\phi_{pan}, \phi_{tilt})$, that would cause the object to be centered in the visual field.

Thus a node in the expected area net represents a 2-D discrete random variable, p. $BEL(p) = P(p \mid \mathbf{e})$ is a function on a discrete 2-D grid (usually 32×32), with a high value corresponding to a scene location at which the object is expected with high probability. The $BEL(p)$ distribution for an object will sometimes be called the *expected area* of that object. Figure 4(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.

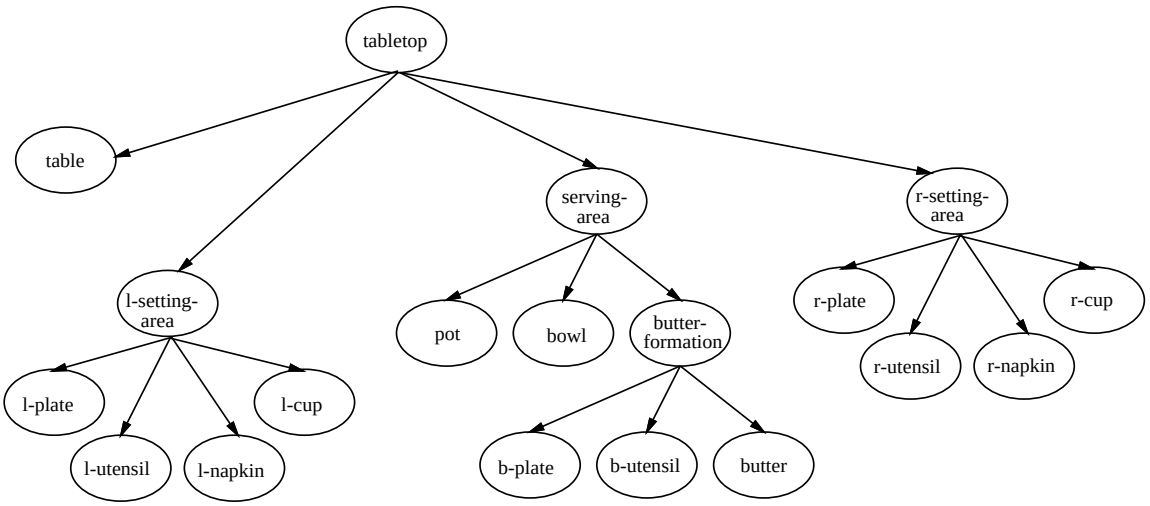


Figure 3: The core portion of a PART-OF Bayes net. Additional nodes (not shown) represent outputs of the visual actions. For example, in the full net the *l-plate* node has a child named `per-detect-plate`. The associated expected area net has the same structure as the core of the PART-OF net.

A root node R of an expected area net has an *a priori* probability, $P(p_R)$, which specifies the absolute location of the object in the scene corresponding to node R. A link from node A to node B has an associated conditional probability, $P(p_B \mid p_A)$, which specifies the relative geometric relation between the two objects corresponding to nodes A and B. Given a reasonable discretization of each p, say as a 32×32 grid, each conditional probability table has just over a million entries. Such tables are impractical. TEA-1 uses a simplified distribution called a *relation map*. A relation map $R_{B|A}(x, y)$ is associated with each link in the expected area net, specifies the (distribution for the) location of object B relative to the size and location of object A, and is supplied by the knowledge engineer. 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 $P(p_B \mid p_A) = f(p_B; R_{B|A}, p_A, h_A, w_A)$. The values for the height h_A and width w_A of an object A are stored in the data structure for object A's node in the expected area net. These values are simply stored in the node, formal BEL values are not maintained for them. A priori supplied values are used for the height and width initially, but once an object is located using image data the observed values are stored instead.

Figure 4(c)-(d) shows two examples of relation maps that were used in the calculation of the two expected areas shown in figure 4(a)-(b). The spatial resolution of the relation map grid can be less than that of the expected area grid. The small size and large number of zeroes in the expected area grids speed up the belief propagation computation [Rimey and Brown, 1992]. The calculation can be sped up in other ways too, for example BEL values for a node need not be recalculated if the messages sent to a node are unchanged from the previous ones, and the global belief propagation calculation can stop once no node's state

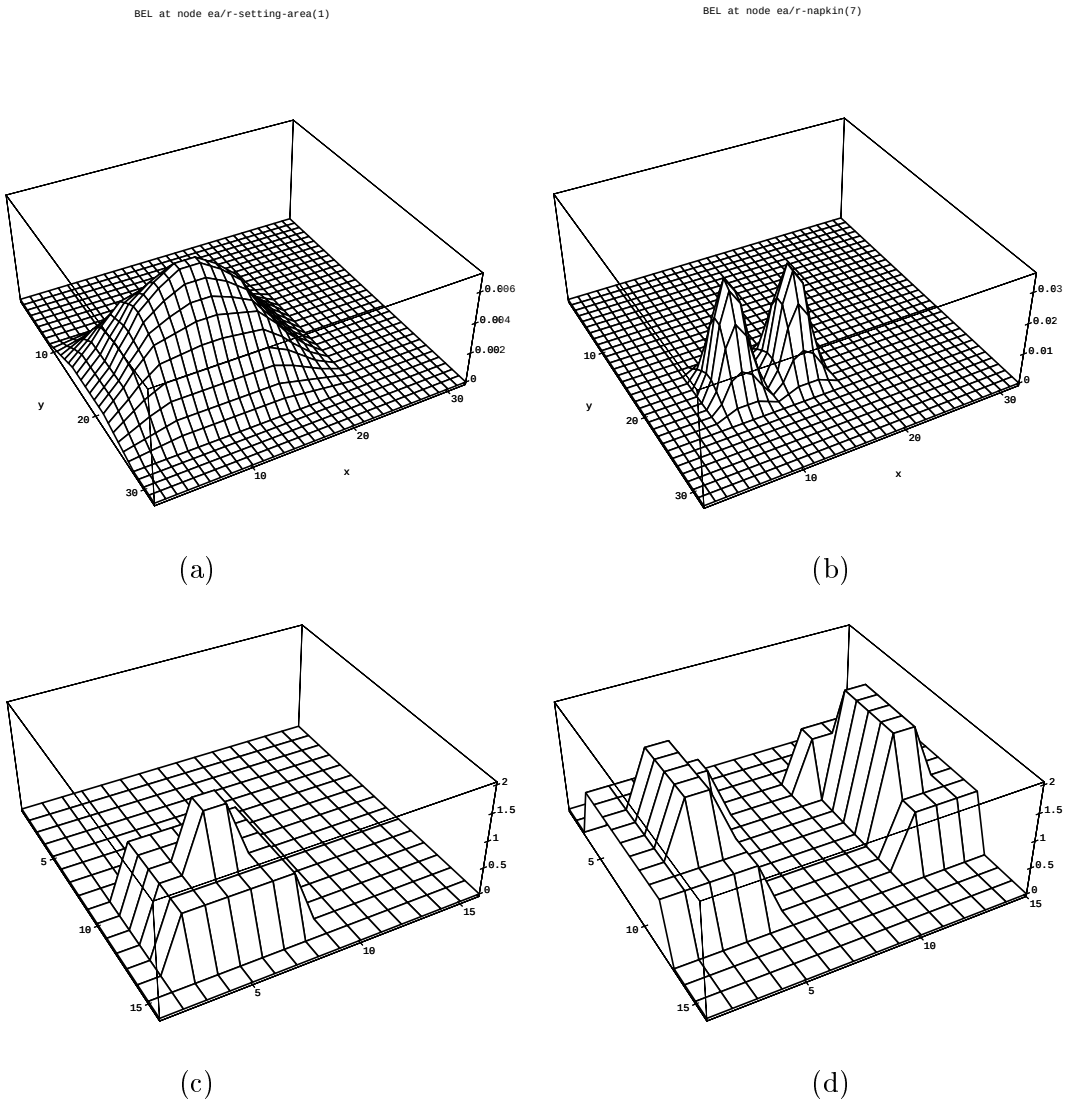


Figure 4: (a) $BEL(p_{\textit{l-setting-area}})$, the expected area for *l-setting-area* (a place setting area) before the location of any other object is determined. (b) $BEL(p_{\textit{l-napkin}})$, the expected area for *l-napkin* after the location of the tabletop and plate have been determined. (c) $R_{\textit{l-setting-area}|\textit{tabletop}}$, the relation map for *l-setting-area* given *tabletop*. (d) $R_{\textit{l-napkin}|\textit{l-setting-area}}$, the relation map for *l-napkin* given *l-setting-area*. The expected area grid is 32×32, and the relation map grid is 16×16.

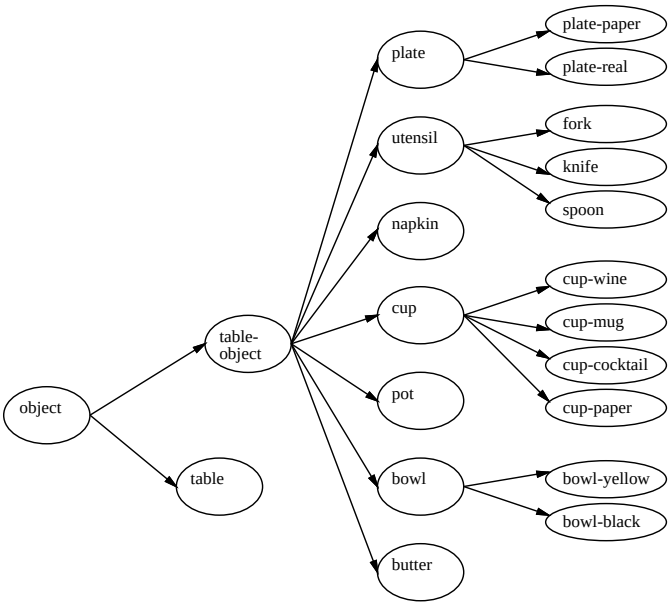


Figure 5: An IS-A Bayes tree.

changes.

The IS-A tree is a taxonomic hierarchy representing mutually-exclusive subset relationships in the domain [Pearl, 1988]. The classification of each object in the scene is represented with an IS-A tree, meaning there is an IS-A tree for each node in the PART-OF net. Figure 5 shows the IS-A tree for the table setting example.

Task-specific and domain-specific knowledge are distinguished in a composite net. Domain knowledge is represented by the PART-OF, expected area, and IS-A nets. The knowledge specifically needed to solve a given task is encoded in a *task net*. Solving the class of T-world problems defined in section 1.3 requires knowledge about what objects and object property values are expected for each possible outcome of the task variable. For example, figure 6 shows a task net that encodes the knowledge needed to decide whether a table is set for a fancy or informal meal. Specifically, in this example a fancy scene is more likely to contain ceramic plates (instead of paper), fork type utensils, a napkin, wine or cocktail glasses, and butter on a butter plate. The objects and property values in a notfancy scene are simply the opposite.

A task net encodes what subset of scene information would be useful for solving the task, but says nothing about *how* to obtain that information. A different task net is plugged into the composite net for each task the system is to solve, and a general decision making framework (section 2.10), in combination with the Bayes nets, produces a pattern of camera and fovea movements and visual operations that is unique to the task and even task instance.

Evidence propagates in all the nets, except the IS-A tree, as described in [Pearl, 1988]. For the IS-A tree, a special version of a Bayes net incorporates the mutual-exclusivity constraint [Pearl, 1988; Chou and Brown, 1990].

BEL values are calculated in the composite net as follows: (1) Propagate belief in

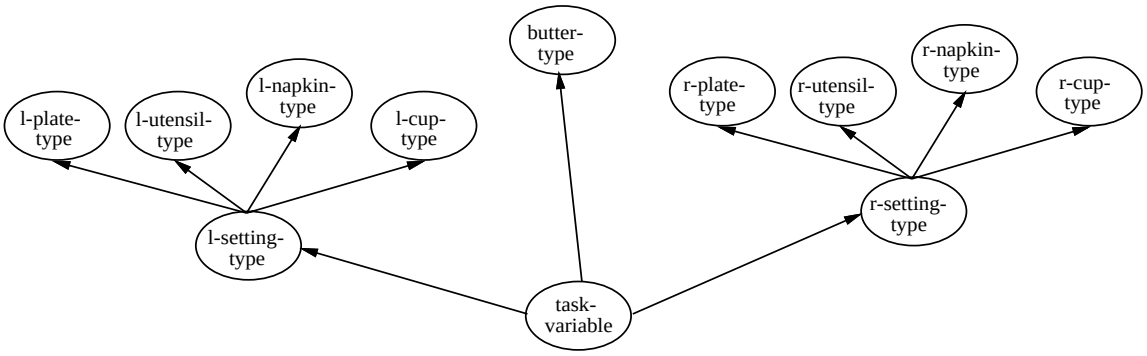


Figure 6: The core of a task Bayes net. This net encodes the knowledge needed to decide whether a table is set for a fancy or informal meal. A leaf node in the full net may have a child node (not shown) that represents a package value.

each of the nets, except the task net. That is, each of the separate nets in the composite net, except the task net, maintains its BEL values independently of the other nets. (2) Construct *packages* of BEL values from the other nets for transfer to the task net. A package is treated as an evidence report that is attached to a node in the task net. (3) Propagate belief in the task net.

TEA-1 currently uses one general type of package, which is somewhat specific to T-world domains and problems. The package 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: $BEL(present) = \alpha$ and $BEL(notPresent) = 1 - \alpha$. And, in the object's IS-A tree, let the subset of desired classifications for the object have the following belief values: $BEL(\omega_i) = \beta_i$, for each class ω_i in the subset. Assuming that a node in the task net needs information about this object, the package contains the following values: $(\alpha\beta_1, \alpha\beta_2, ...\alpha\beta_d, 1 - \alpha \sum_{i=1}^{d} \beta_i)$. 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 labels: ω_1, ω_2, ... ω_d, and $notPresentOrOther$. For example, if the object was that corresponding to the *l-utensil* node in the PART-OF net, and the subset of desired classifications from the IS-A net for that object is $\{fork, knife, spoon\}$, then the package is: $(\alpha\beta_{fork}, \alpha\beta_{knife}, \alpha\beta_{spoon}, 1 - \alpha(\beta_{fork} + \beta_{knife} + \beta_{spoon}))$. The package values are used as an evidence report that is added to the *l-utensil-type* node in the task net using a "dummy" evidence node [Pearl, 1988]. The *l-utensil-type* node in the task net has the following labels: $fork$, $knife$, $spoon$, and $notPresentOrOther$.

2.3 The Two-Net

A *two-net* contains an expected area net, identical to that in the composite net, and a *monolithic net*, which represents all the other knowledge. A two-net represents the same type of knowledge as the composite net, but in a simpler and far less general form. While the composite net contains some *ad hoc* features, particularly the use of packages, the two-net has none, which is desirable for certain experimental analysis. Figure 7 shows an example

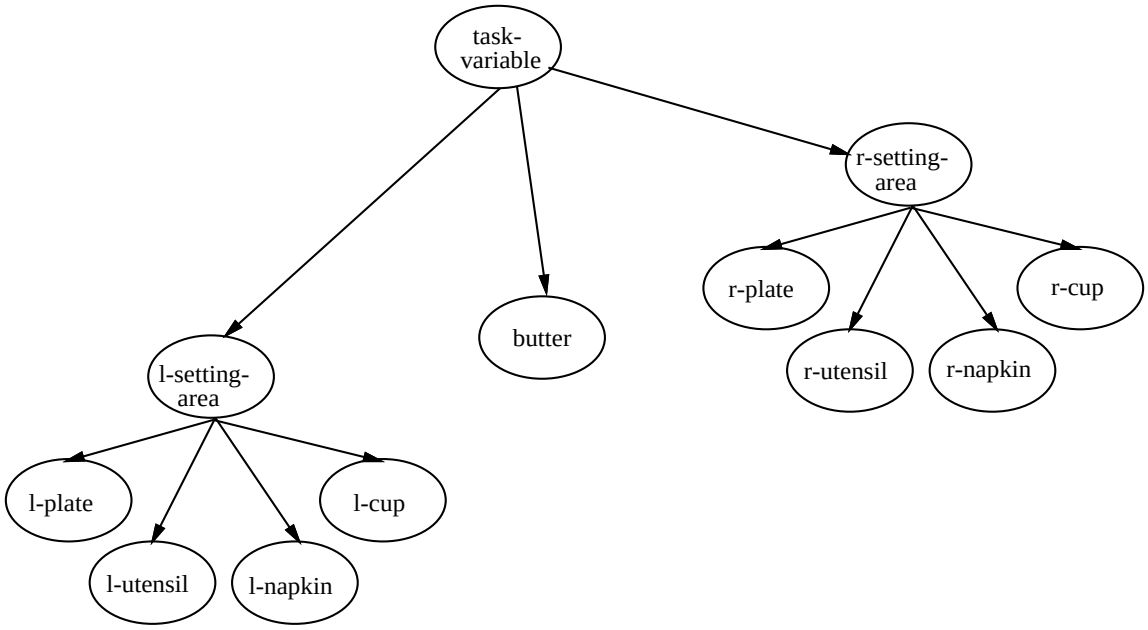


Figure 7: The core of a monolithic net for the "fancy/notfancy" task in the table setting domain. Additional nodes (not shown) represent outputs of the visual actions.

of a monolithic net for the "fancy/notfancy" task in the table setting domain.

2.4 Actions

TEA-1 contains two types of action. A *visual action* processes image data to produce evidence. Figure 8 shows a schema for a visual action. A schema contains four major parts, described below: a function that executes the action, a precondition, rules for adding the action's results as evidence to the composite net (or two-net), and models for the cost and performance of the action. All the examples in this chapter will assume a composite net, rather than a two-net.

TEA-1 can use any vision algorithm/module that can be encapsulated in TEA-1's action schema. Most importantly, the cost and performance models in TEA-1's action scheme must be reasonable models of the action's behavior.

A *camera movement action* moves the center of the camera's view to a specified position $p = (p_x, p_y)$ in the scene. It has no precondition and does not produce any evidence. There is a model for the cost of moving the camera. A camera movement always moves exactly to the specified position, so there is no need for a performance model.

TEA-1's costs are measured in time, and the cost of electro-mechanical actions such as camera movements is likely to be higher than purely electronic ones, such as a visual action or moving an electronic fovea (like ours) around in an image. Thus when action sequences are considered, minimizing camera motions should be a natural consequence of a good

action-name : `per-detect-plate`
action-function : per-detect-plate
input parameters: $BEL(X), h_X, w_X$
return value: double results[MAXRESULTS]
global read access: composite net, image data, p_{camera}
precondition :
not-instantiated net 1 node 5, and
in-FOV net 1 node 5, and
status-0
evidence adding rules :
R1: use results[7-8] as dummy-evidence to net 0 node 5,
R2: use results[1-4] as instantiate-ea-evidence to net 1 node 5,
R3: use results[8] as isa-evidence to net 2 node 4
cost model :
$C_0 = 15$,
$C_1 = 10$
performance model :

R1: $P(x_{action} \mid x_{true}) =$

		x_{true}	
		$notPresent$	$present$
x_{action}	$notPresent$	0.9	0.1
	$present$	0.1	0.9

R2: none used

R3: $\lambda(x_{true}) =$

	x_{true}	
	$notPresent$	$present$
λ	0.5	2.0

Figure 8: An action schema.

strategy. This is the main reason for separating camera motions from visual (mechanical from electronic) actions.

2.5 Action Function

A visual action in TEA-1 either tries to detect and locate a specified type of object in the image data, or it tries to ascertain the value of a specified property of an object already located in the image data. Since an action, denoted as α, tries to either detect or classify a specific object, denoted X, we will often refer to the action α as being *associated with* object X. A function is called to execute a visual action. All actions are constructed from one or more low-level vision modules, process either foveal image or peripheral image data, and may process only a portion of the image data. If an action processes a foveal image then it may first move the fovea.

There may be any number of visual actions. Each kind of object usually has several actions associated with it. In the table setting domain, TEA-1 currently has 21 visual actions related to seven kinds of scene objects, as well as a camera movement action. Since one kind of object may appear in a scene many times, the number of *instances* of visual actions may be much larger. Currently, there are 34 instances of actions in the fancy/notfancy demonstration system. Some examples of visual actions related to plates are: `Per-detect-template-plate`, which uses a model grayscale template to detect the presence and location of a plate in the peripheral image. `Per-detect-plate` uses a Hough transform for plate-sized circles for the same purpose. `Per-classify-plate` moves a window to the location of a plate in the peripheral image and uses a color histogram to classify that area as paper/plastic (blue) or ceramic (brown). `Fov-classify-plate` moves the fovea to a specified location and uses a color histogram to classify the area as paper/plastic or ceramic.

The input to an action function is the current camera position p_{camera}, the expected area data $BEL(X)$ for object X, the (possible expected) height h_X and width w_X of object X, and of course the raw image data. p_{camera} is needed to compute the expected area mask (see below) from $BEL(X)$. The result output by an action function is a vector of numbers containing (a) an execution status value (results[0]), (b) the location and size of the object (results[1-4]), and (c) a vector of scores (results[7-(MAXRESULTS-1)]). The meaning of the scores is different for each kind of action. For an object detection action, the score vector gives the likelihood that the object is present in the image or not. For a classification action, the score vector gives the likelihood of each possible classification of the object (or generally the possible values of a specified property). The evidence adding rules, to be described shortly, specify how the outputs from an action are added as evidence to the composite net.

Visual actions can be applied to image subsets: if the expected area is narrower than the camera's field of view then it can be used to create a mask, called an *expected area mask*, and only those pixels passed by the mask are processed by the action.

Processing only part of the image data is important because: 1) The probability of an action producing the desired result is higher when less image data is processed (assuming the probability of the object being in that data is held constant) – this is the sufficing vision

argument. 2) The cost (*i.e.* execution time) of the action is smaller, again since less image data is processed.

Let $l \in (0,1)$ be a 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 in the expected area for object X, associated with action α, such that their probabilities add up to l. The points in G_X^l, mapped from expected area to image coordinates, specify the mask.

As more objects are located via actions, the expected areas for related objects (not yet located by actions) get narrower. In the table setting domain, assume that TEA-1 has located (in order) the tabletop, then the plate, and finally the napkin. Figure 9 shows how the cup's expected area gets narrower and how the `per-detect-cup` action would perform after each additional object is located.

2.6 Action Precondition

Before an action may be executed its precondition must be satisfied. The precondition may be any conjunction of the following five types of condition: a specified node in the expected area net must be instantiated (meaning the location of the object is known), or it must not be instantiated (location not known), the centroid of a specified expected area (*i.e.* the expected or known location of a specific object) must be within the field of view for the current camera position, the status value of the action must be zero (meaning the action has never been run before), and the empty precondition. For example, three conditions must be true before the `per-detect-plate` action can be executed: the plate's location must not already be known, the expected location of the plate must be within the field of view for the current camera position, and the action must not have been executed previously.

2.7 Evidence Adding Rules

An action generally posts evidence to several different nets. Each instance of an action contains a set of rules for adding evidence to the composite net. Each rule specifies a portion of the action's vector of output values, and one of three methods for adding that portion of the output as evidence to a specific place in the composite net. The three methods in TEA-1 for adding evidence are called dummy-evidence, instantiate-ea-evidence and isa-evidence.

- dummy-evidence: The *dummy-evidence* method converts a specified node into a *dummy* node, setting the dummy node's values to the given evidence values. A dummy node Y represents judgemental evidence about the value of its parent node X. Specifically a dummy node represents and stores the values of $P(observation \mid X)$. (Alternatively the dummy node could be *attached to* the specified node, thus filtering the evidence through the conditional probability between the specified node and it's parent.)

- instantiate-ea-evidence: When a specific value of a random variable is observed to be true, the node in a Bayes net representing that random variable is *instantiated* to that value. The belief is 1.0 for that value and 0.0 for the other values. The *instantiate-ea-evidence* method is given the observed location and dimensions of an object, and

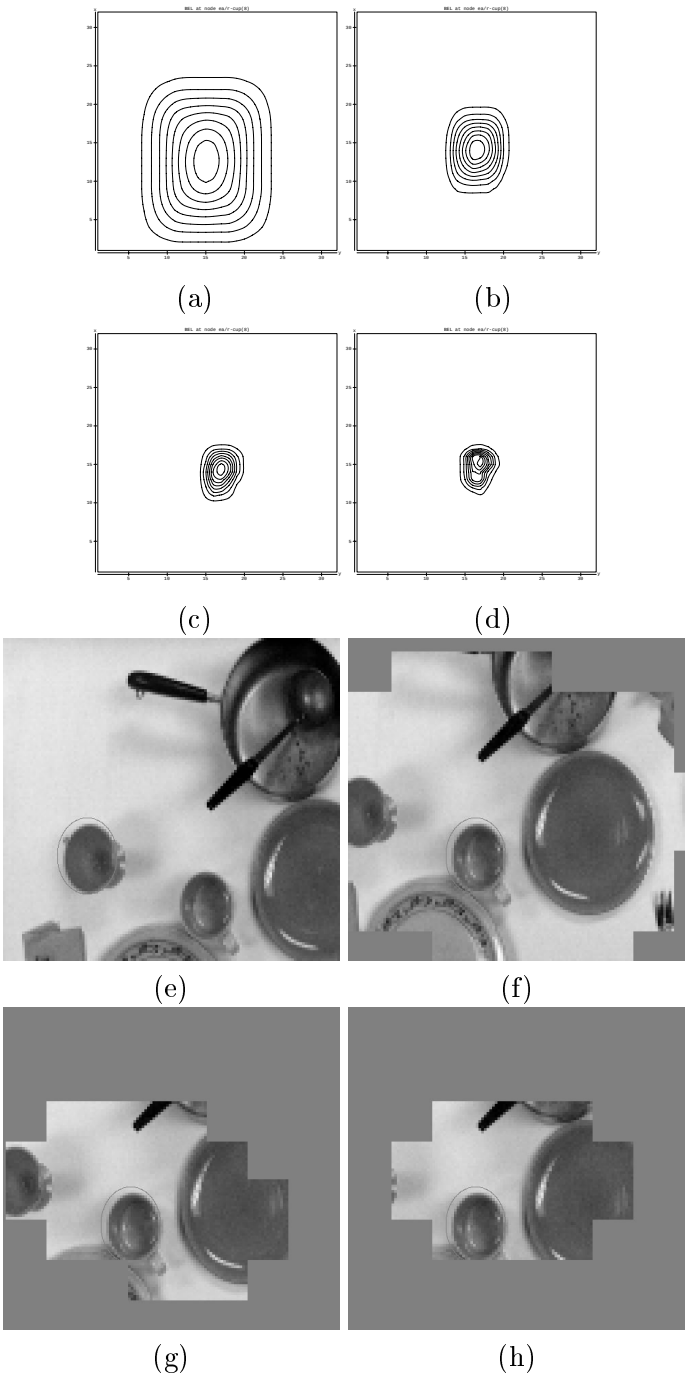


Figure 9: (a)-(d) How the cup's expected area changes over time as more objects are located via actions, and (e)-(h) how the performance of a cup detection action is affected by the changing expected area: (a) and (e) before any objects have been located (the creamer container is mistakenly detected as the cup), (b) and (f) after the tabletop has been located (the cup is correctly detected but this was lucky), (c) and (g) after the tabletop and plate have been located (the cup is correctly detected), (d) and (h) after the tabletop, plate and napkin have been located (the cup is correctly detected).

instantiates a specified node in the expected area net to the location, and then also stores the height and width dimensions of the object in that node.

- isa-evidence: Evidence is added to a specified subset S of nodes in an IS-A net as described in [Pearl, 1988; Chou and Brown, 1990]: The values output by the action, and added to the IS-A net, are treated as the likelihood ratio, $\lambda = \frac{P(e|S)}{P(e|\neg S)}$, of the evidence e observed by the action.

See [Pearl, 1988] for a detailed explanation of the different types of nodes and evidence that may be used in a Bayes net.

2.8 Modeling Cost of an Action

The unit of measurement for action cost is proportional to the real-world running time of the respective action in the lab. The coefficients in the equations below for each action are estimated by timing key parts of the code that implements that action.

A camera movement action α has cost

$$C(\alpha) = \begin{cases} C_0^{camera} + C_1^{camera} d & \text{if } d \neq 0 \\ 0 & \text{if } d = 0 \end{cases} \quad (1)$$

where d is the distance from the current camera position to the new position.

A detection type action α has cost

$$C(\alpha) = C_0^{\alpha} + C_1^{\alpha} r_X^l. \quad (2)$$

A detection action uses an expected area mask, processing only the pixels that are passed by the mask, so the cost model has a term that is proportional to the fraction of pixels processed. Let G_X^l be the smallest subset of all the grid points G_X in the expected area for object X, associated with action α, such that their probabilities add up to l. The value of r_X^l is the size of the subset of G_X^l that fills the field of view divided by the size of the subset of G_X that fills the field of view. $r_X^l = 1$ means that object X could be located anywhere in the entire field of view. Over time, as other objects in the scene are located and as more and tighter relations are established, the value of r_X^l approaches zero.

A classification type action α has cost

$$C(\alpha) = C_0^{\alpha} + C_1^{\alpha} n_X \quad (3)$$

where n_X is the number of possible classifications for the object X that the action α tries to classify.

2.9 Modeling Performance of an Action

The performance model for an action specifies the expected consequences of the action for each evidence adding rule. There are three types of rule, each with its own general form of model.

- dummy-evidence: The performance model is a table of $P(x_{action} \mid x_{true})$ values, where x_{action} is the evidence added by the rule, and x_{true} is the corresponding true state of the world. When the random variables x_{action} and x_{true} are represented by nodes in a Bayes net, this probability is associated with the link connecting those two nodes.

- isa-evidence: The performance model is simply the expected value of the likelihood ratios (the λ values) that the action outputs. There is one set of λ values for each true state of the world.

- instantiated-ea-evidence. This method of adding evidence instantiates one value of a specified node in the expected area net, meaning the action is assumed always to be perfect and no performance model is needed. (Alternatively a form of dummy evidence could be used here. Then the performance model could specify a probability distribution for the location of the object as determined by the action, given the true location of the object).

2.10 Decision Making

TEA-1 is to find the "best" sequence of actions to produce evidence supporting a solution to a specified task. The sequence of actions with the shortest total execution time is the best sequence. The best sequence generally will change after the results from each executed action are available.

TEA-1 uses the following high-level control loop:

1. Based on the current evidence, decide what the solution to the task is.

2. Decide whether to gather more evidence or to stop (and therefore accept the solution to the task from step 1 as the final solution).

3. Decide what action to execute next. This may be either a camera movement action or a visual action.

4. Execute the action, incorporate the resulting evidence into the knowledge representation, update the BEL values in the Bayes nets, and go to step 1.

The decision about the solution to the task is denoted as d. In the table setting example, the choices for d are d_0 and d_1, corresponding to $notfancy$ and $fancy$, but in general there may be any number of choices. The optimal decision d^* about the solution to the task is given by

$$d^* = \arg\max_i EV(d_i) \tag{4}$$

where

$$EV(d_i) = \sum_{j=0}^{1} V(d_i, t_j) P(t_j) \tag{5}$$

The answer to the task, *i.e.* the true state of the world, is modelled by the variable t with a known associated distribution, $P(t)$. In the table setting example, the possible values

of t are t_0 and t_1, which respectively denote the states $notfancy$ and $fancy$. In general t may have any number of possible values. $V(d_i, t_j)$ is a payoff function. In the table setting example we use

$$V(d_i, t_j) = \begin{cases} 1000 & \text{if } i = j \\ -1000 & \text{if } i \neq j. \end{cases} \tag{6}$$

The units of payoff are seconds. Thus if $P(d_0, t_0) = P(d_1, t_1) = 1000$, a correct decision about the task is worth 1000 seconds of effort. The value associated with the optimal decision is

$$EV_0 = \max_i EV(d_i) = EV(d^*). \tag{7}$$

The variable t can represent an entire composite net (or two-net). Specifically, a composite net (or two-net) can be reduced to a single chance node, called t, using influence diagram manipulations. As a result, the $P(t_j)$ term in equation (5) can be replaced by $BEL(t_j)$, and the reduction accounts for all evidence added to the knowledge representation since $BEL(t) = P(t \mid \mathbf{e})$.

TEA-1 uses the following rule for deciding whether to gather more evidence or to stop:

> If $EV(d^*) > T$ and this has been true after the previous K actions, then stop, else gather more evidence.

T and K are constants, where $\min_{i,j} V(d_i, t_j) \leq T \leq \max_{i,j} V(d_i, t_j)$ and $K \geq 0$. (Typical values are $T = 600$ and $K = 2$.) Larger values of T force the system to gather more evidence. Small values of K are useful in practice because of the variation and uncertainty in real scenes and images.

Deciding what action to execute next is a complicated decision. The value of a visual action's result is calculated using a standard concept from decision theory called the *expected value of sample information (EVSI)* (See *e.g.* [Clemen, 1991; Dean and Wellman, 1991; Pearl, 1988]). The basic idea is that the EVSI for an action's result is the difference between the expected value of the task decision before and after the action is assumed to have been executed.

The expected value of the task decision given a piece of evidence e, which has n_e possible values, is

$$EV_e = \sum_{k=0}^{n_e} [\max_i P(d_i, e_k)] P(e_k) \tag{8}$$

where

$$P(d_i, e_k) = \sum_{j=0}^{1} V(d_i, t_j) P(t_j \mid e_k). \tag{9}$$

As before, a composite net (or two-net), can be collapsed into the variable t, with the effect of replacing $P(e_k)$ and $P(t_j \mid e_k)$ in the above equations with $BEL(e_k)$ and $BEL(t_j) \mid_{e=e_k}$. $BEL(t_j) \mid_{e=e_k}$ is the value of $BEL(t_j)$ after node e has been instantiated to e_k.

The expected value of sample information for the evidence e is

$$EVSI(e) = EV_e - EV_0. \tag{10}$$

Deciding what action to execute next requires that sequences of future actions be considered, but it is infeasible to enumerate all sequences of actions. The practical alternatives fall into three broad categories:

- Fixed set of explicit strategies: In some applications it may be acceptable to list each action sequence explicitly in a small fixed subset of all possible sequences. Each sequence can be scored using decision theory techniques to pick the best one [Dean and Wellman, 1991; Dean *et al.*, 1990]. Most vision problems are too complex for this simplistic approach.

- Goodness function: A *myopic decision policy* 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. A one-step myopic policy does not appear sufficient for most computer vision applications: For example, every table setting has plates, so plate detection itself gives no new information; however it enables plate material classification, fork-counting, *etc.* which could have high impact. A *goodness function* is used to score each possible choice of action; it should capture all the important things to reason about in action sequences, and is usually based on approximations (since the exact situations are too complex to model exactly). The goodness function can sometimes simulate search.

- Search: Enumerate and evaluate all action sequences. The AI literature contains a variety of techniques for approaching search problems, but it is unclear how useful long plans actually are in the vision context and core planning problems (subgoal interaction) are less relevant.

Although all the approaches consider (more or less explicit) sequences of actions, TEA-1 assumes the cost of planning is small and the likelihood of surprise is large, so only the first action in the best sequence is executed; the rest of the sequence is forgotten, and the decision-making cycle starts over again.

We have investigated both goodness functions and brute-force action sequence enumeration. The tradeoff between the two methods is similar to that between static evaluation function sophistication and search depth in a state-space problem solver.

Early versions of TEA-1 (described in [Rimey and Brown, 1992; Rimey, 1993]) characterized the value of a visual action's result by Shannon's measure of average mutual information. The construction of goodness functions based on average mutual information is similar to the construction using EVSI, but this measure has several disadvantages [Pearl, 1988]. In particular, average mutual information is a log measure, which is difficult to interpret and difficult to calibrate with cost, a linear measure.

2.11 A Goodness Function Approach to Control

When using the goodness function method, TEA-1 separates the decision of what action to execute next into two sequential decisions: First, whether to move the camera or not, and if so then where to move it. Second, what visual action to execute.

TEA-1 models the value $V(\alpha)$ of an action α as

$$V(\alpha) = [EV_e h(EA_\alpha \mid p_{camera}) + EV_0(1 - h(EA_\alpha \mid p_{camera}))] - EV_0 \quad (11)$$
$$= h(EA_\alpha \mid p_{camera}) EVSI(e_\alpha). \quad (12)$$

EA_α is the node in the expected area net for the object associated with action α. $h(EA_\alpha \mid p_{camera})$ is the probability that the object is in the current field of view, called FOV, a rectangular area (in the expected area coordinate system) that is centered at the current camera position p_{camera}, and is given by

$$h(EA_\alpha \mid p_{camera}) = \sum_{(i,j) \in FOV} BEL(EA_\alpha \text{ at } (i,j)). \quad (13)$$

e_α is the evidence that results from action α.

The value $V(\alpha)$ and cost $C(\alpha)$ of a visual action α are the core elements in a goodness function. Value and cost are measured in the same units, seconds of execution time. The tradeoff between value and cost is captured by subtracting the two. For example, a very simple goodness function for a visual action is

$$G'(\alpha) = V(\alpha) - C(\alpha) \quad (14)$$
$$= h(EA_\alpha \mid p_{camera}) EVSI(e_\alpha) - C(\alpha). \quad (15)$$

Features of this goodness function are that an action's value is determined relative to the needs of the current task, and an action's cost is proportional to the amount of image data processed.

2.11.1 Goodness Function for a Visual Action

TEA-1's goodness function is more complex, accounting for peripheral actions that detect an object but do not otherwise generate information for the task, and also accounting for the impact of making expected areas smaller (which makes other actions have lower costs and better performance). The goodness function $G(\alpha)$ of a visual action α is

$$G(\alpha) = G_1(\alpha) + HG_2(EA_\alpha). \quad (16)$$

$H \in (0,1)$ 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, $G_1(\alpha)$, accounts for the future value of establishing the location of an object:

$$G_1(\alpha) = [V(\alpha) - C(\alpha)] + \max_{\beta \in PreExe(\alpha)} [V(\beta) - C(\beta)]. \quad (17)$$

$PreExe(\alpha) = \{\beta \mid$ EITHER action β has a precondition satisfied by executing action α OR action β is already executable and $[V(\beta) - C(\beta)] < [V(\alpha) - C(\alpha)]\}$. The second term, $G_2(EA_\alpha)$, accounts for the impact of making expected areas smaller so that future actions will have higher probability of success and lower costs:

$$G_2(EA_\alpha) = \frac{1}{\parallel EANet - EA_\alpha \parallel} \sum_{X \in EANet - EA_\alpha} \Delta(X \mid EA_\alpha) \quad (18)$$

where EA_α is the expected area node for the object associated with action α, $EANet$ is the set of nodes in the expected area net, and

$$\Delta(X \mid EA_\alpha) \quad = \quad \max_{\gamma \in PreAction(X)} [G_1(\gamma \mid EA_\alpha) - G_1(\gamma)].$$

$\Delta(X \mid EA_\alpha)$ is the best increase in goodness for an action that affects X. $PreAction(X) = \{\gamma \mid$ action γ is associated with object X and has a currently valid precondition $\}$. $G_1(\gamma \mid EA_\alpha)$ is computed using equation (17), while assuming the location of object EA_α is at the centroid of its expected area, meaning that the expected area node EA_α is temporarily instantiated to its centroid.

TEA-1 decides what visual action α^* to execute next using the following decision rule

$$\alpha^* = \arg \max_{\alpha \in Pre} G(\alpha) \tag{19}$$

where Pre is the set of all actions with true precondition.

2.11.2 Goodness Function for a Camera Movement Action

TEA-1 decides whether to move the camera, and if so then where, using the following decision rule:

$$p^* = \arg \max_{p \in L} G(p). \tag{20}$$

The set L contains the locations to which a camera movement should be considered and p_{camera} (the current camera position). The locations in L are normally the centroids of the expected areas in the expected area net. Both objects and abstract groups have expected areas. The choice of p_{camera} corresponds to the choice of not moving the camera.

The basic idea of $G(p)$ is that the goodness function associated with moving the camera to location p is the sum of the goodness associated with all the actions that could be executed if the camera was centered at location p minus the cost of moving the camera to p. The goodness function $G(p)$ for moving to location p is

$$G(p) = TDS(p) - C(p) \tag{21}$$

where

$$TDS(p) = \sum_{\gamma \in PreCamera(p)} G(\gamma \mid p_{camera} = p). \tag{22}$$

$PreCamera(p) = \{\gamma \mid$ action γ has valid precondition, assuming the current camera position is $p\}$. $G(\gamma \mid p_{camera} = p)$ is the goodness function for a visual action from equation (16), given that the camera is at location p. $C(p)$ is the cost of moving the camera to position p and is given by equation (1). Note that $C(p_{camera}) = 0$.

An alternative to the above function for $G(p)$ is to sort the terms inside the summation into decreasing order (let this order be indexed by k) and then multiply each term by a discounting term such as β^k ($\beta \in (0, 1)$, typically chosen to be 0.9). The resulting goodness function places more emphasis on the first few actions that would likely be executed at the location.

The set L could be chosen to be the set of all grid points in the expected area coordinate system. In this case the $TDS(p)$ function is like a top-down version of a "salience map" [Clark and Ferrier, 1988; Rimey and Brown, 1991; Elfes, 1992] that covers the entire expected area grid. This map is roughly like a sum of weighted expected area maps, one for each object that has an associated action whose preconditions are met. The weight of the map for an object is roughly determined by the goodnesses of the actions associated with that object. The resulting surface, with height defined over each expected area grid square, is some measure of the promise each location holds for visual exploitation, and the maximum value of the surface is the best candidate for the next camera pointing action (before camera movement costs are considered).

2.12 A State-Space Search Approach to Control

When using the state-space search approach to control TEA-1 decides on the best sequence of actions to execute using the following decision rule

$$S^* = \arg \max_{S \in Pre(H)} G(S) \tag{23}$$

where $S = (\alpha_{i_1}, \alpha_{i_2}, ..., \alpha_{i_H})$ is a sequence of visual and camera movement actions. The next action to execute, α^*, is the first action $\alpha^*_{i_1}$ in the sequence S^*. The set $Pre(H)$ contains all action sequences of length H, except those sequences prohibited by preconditions. A camera movement action can move the camera to one of a finite number of locations in the scene, defined as the set L and usually chosen to be the centroids of the expected areas in the expected area net.

The goodness function $G(S)$ for a sequence S of actions is

$$G(S) = \sum_{k=1}^{H} \beta^k [h(EA_{\alpha_{i_k}} \mid \alpha_{i_{k-1}}, ..., \alpha_{i_1}, p_{camera}) EVSI(e_{\alpha_{i_k}}) - C(\alpha_{i_k} \mid \alpha_{i_{k-1}}, ..., \alpha_{i_1}, p_{camera})] \tag{24}$$

$\beta \in (0, 1)$ is a user specified value close to 1.0 (we use 0.9) that causes the best sequencing of a set to be selected, specifically it places the "better" actions nearer the start of the sequence. $h(EA_{\alpha_{i_k}} \mid \alpha_{i_{k-1}}, ..., \alpha_{i_1}, p_{camera})$ and $C(\alpha_{i_k} \mid \alpha_{i_{k-1}}, ..., \alpha_{i_1}, p_{camera})$ are similar to equations (13) and (1)-(3) but also account for the fact that the actions α_{i_1} through $\alpha_{i_{k-1}}$ may move the camera and instantiate nodes in the expected area net. An efficient way to calculate $G(S)$ for all the action sequences is via a depth-first search of the space of all action sequences, while simulating state changes caused by the actions α_{i_1} through $\alpha_{i_{k-1}}$. Equation (24) is similar to value functions used by others, *e.g.* [Dean *et al.*, 1990]. The definition of the expected value of sample information can be extended to action sequences, and requires enumerating all possible sequences up to a given length, but approximations are possible [Heckerman *et al.*, 1993].

3 Experimental Results

Two example runs are presented below to illustrate the basics of how TEA-1 works.

k	t	task decision	camera movement decision	visual action decision
1	0	notfancy	move to *l-cup* at (17.29,11.54)	`per-detect-cup`
2	37	notfancy	don't move	`per-classify-cup`
3	62	notfancy	move to *l-utensil* at (22.29, 8.83)	`per-detect-napkin`
4	94	notfancy	don't move	`per-detect-plate`
5	108	notfancy	don't move	`per-classify-plate`
6	123	notfancy	move to *l-utensil* at (22.73, 9.41)	`per-detect-utensil`
7	167	notfancy	don't move	`per-classify-utensil`
8	202	fancy	move to *b-plate* at (12.32,15.86)	`per-detect-butter`
9	237	fancy	move to *r-cup* at (17.44,22.96)	`per-detect-cup`
10	275	fancy	move to *r-cup* at (16.29,25.38)	`per-classify-cup`
11	321	fancy	move to *r-setting-area* at (20.40,23.65)	`per-detect-napkin`
12	352	fancy	-	-

Table 3: Summary of decisions made during the example run on a fancy scene using the goodness function method. Each line of the table corresponds with one cycle of the main decision loop (indexed by k).

3.1 Goodness Function Method — Fancy Scene

The domain is dinner table settings, and the task is to decide whether a table is set for a fancy meal or for an informal (also called notfancy) meal. The domain and task nets, actions, decision rules, payoff function, constants, *etc.* are those presented as examples in section 2, although for technical reasons the $G_2(EA_\alpha)$ term was omitted from the goodness function for a visual action in equation (16). A run for a fancy meal is described here. For more details and an example of an informal meal, see [Rimey, 1993].

All the objects in the scene are ones expected to appear in a fancy meal, except a cup on the left side, which is a coffee mug. A wine glass or cocktail glass are more likely to appear in a fancy meal. The field of view of the camera is considerably smaller than the scene. A wide angle view of the scene is shown in Figure 1(b). The decisions made by TEA-1 are summarized in Table 3 and the camera movements made during the run are illustrated in Figure 10. The the expected values associated with the possible task decisions are plotted in figure 11.

After executing 11 visual actions and 7 camera movement actions over a total of 352 seconds, the system decided to stop and decided that the scene is a fancy one. Figures 12 and 13 illustrate the execution of several actions executed during this run. The camera is positioned initially at the approximate center of the table, as shown in Figure 12(a). The first action executed is a camera movement. Next, actions to detect and classify a cup are executed. The circle drawn in (b) indicates where a cup was detected. The cup was determined correctly to be a coffee mug. Since a mug is more indicative of an informal meal, the value of $BEL(notfancy)$ increased. The expected areas for all the objects were initially quite large and therefore the expected area mask used by the cup detection action passed the entire image. Locating the cup significantly decreases the width of the expected areas for several objects. After moving the camera again, a napkin detection action is executed.

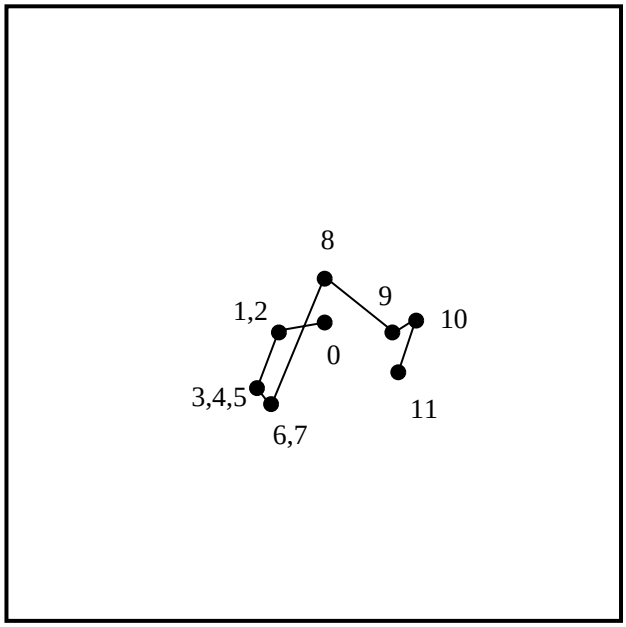


Figure 10: The position of the camera during the example run on a fancy scene using the goodness function method. The values of k, the index in the main decision loop, when the camera is at each location are also shown.

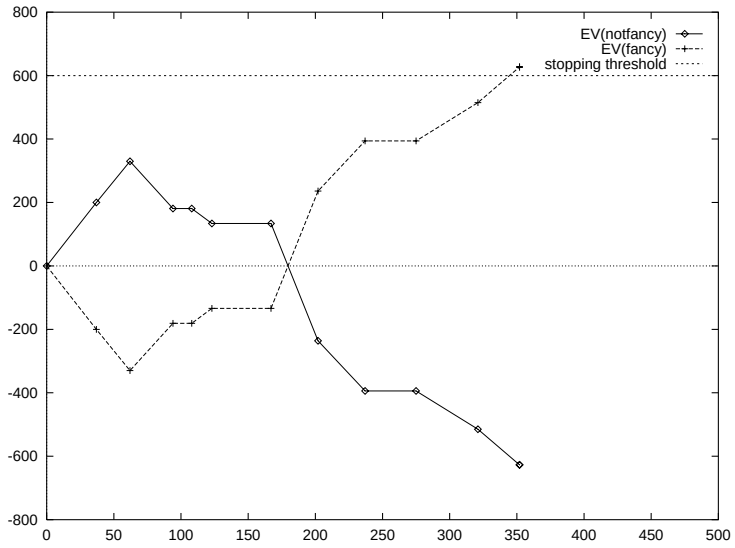


Figure 11: The expected values associated with the possible task decisions, $EV(notfancy)$ and $EV(fancy)$, over time t during the example run on a fancy scene using the goodness function method.

The large gray area in Figure 12(c) depicts the expected area mask used by the napkin detection action. Pixels covered by the gray area are not processed. Next, a plate detection action is executed, as shown in Figure 12(d). Knowing the location of both the cup and the napkin significantly constrains the location of the plate, and the expected area mask here is quite small. After examining some objects in the place setting on the left-hand side of the table, the system analyzes some objects in the central serving area and then in the place setting on the right. There are no direct geometric relations between the objects expected in those areas and the objects located so far, so the expected areas for the objects in those areas are still relatively large. The expected area masks pass the entire image for the one object (butter) the system looks for in the serving area, and for the first object (cup) the system looks for in the right-hand setting area, as shown in figure 13. All the actions executed in this example run produced essentially correct results.

Figure 14 shows the value over time t of the goodness function $G(p)$ for moving the camera to location p. The situation between $t = 94$ ($k = 4$) and $t = 202$ ($k = 8$) is particularly interesting because it illustrates the tradeoff between (a) staying at the current position and executing visual actions or (b) moving the camera to a new location. The value $G(STAY)$ is larger than all the other G values at $t = 108$ ($k = 5$) and $t = 123$ ($k = 6$), so the camera is not moved, but at $t = 167$ ($k = 7$) the value of $G(STAY)$ is just slightly less than $G(l\text{-}utensil)$ and therefore the camera is moved. The value of $G(STAY)$ is again maximum at the next movement decision so the camera is not moved.

3.2 State-Space Search Method — Fancy Scene

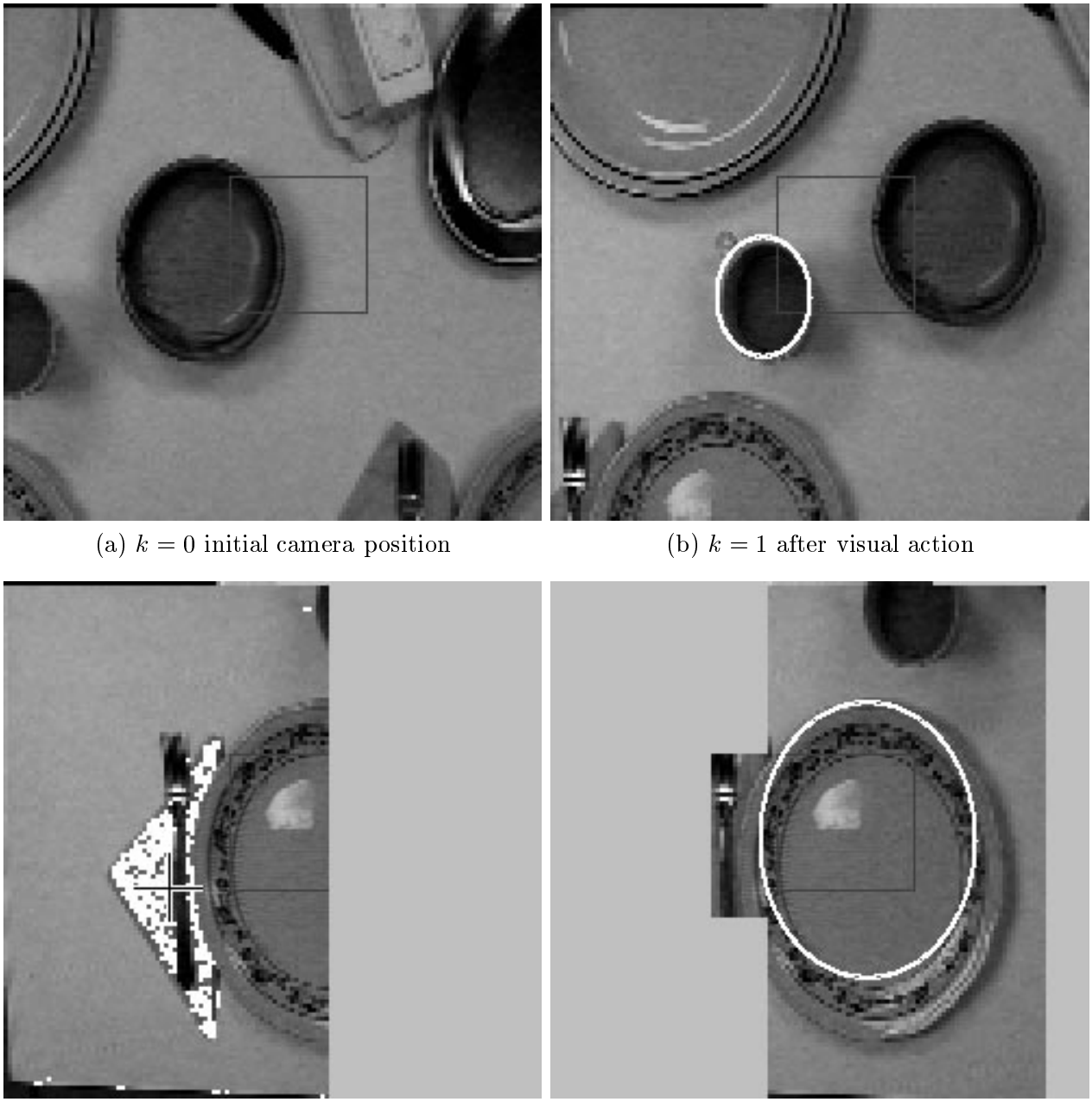
The state-space search control method is illustrated here using an example similar to that in section 3.1. However, it cannot be compared closely with the previous run because of differences in the scenes and action behaviors.

The T-world simulator was used for this example. TEA-1 was presented the scene shown in figure 15, which shows a fancy meal. The large square in the figure depicts the field of view of the simulated camera used by TEA-1.

TEA-1 was directed to consider all sequences S of $H = 3$ actions. The decisions made by TEA-1 are summarized in Table 4. Each line in the table lists the best sequence $S = (\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3})$ found during one iteration of the main decision loop. Only the first action α_{i_1} in the best sequence is executed, and a new best sequence is determined in the next iteration. The camera movements made during the run are illustrated in Figure 16. The expected values associated with the possible task decisions are plotted in Figure 17. After executing 10 visual actions and 4 camera movement actions over a total of 233 seconds, the system decided to stop and decided that the scene is a fancy one.

4 Experimental Analysis of Factors Affecting Performance

We investigated the effect that several key factors (different sorts of scene structure, system parameters, and choice of goodness function) have on the performance of TEA-1 when solving T-world problems. A simulator generated T-world domains, scenes, and tasks, and



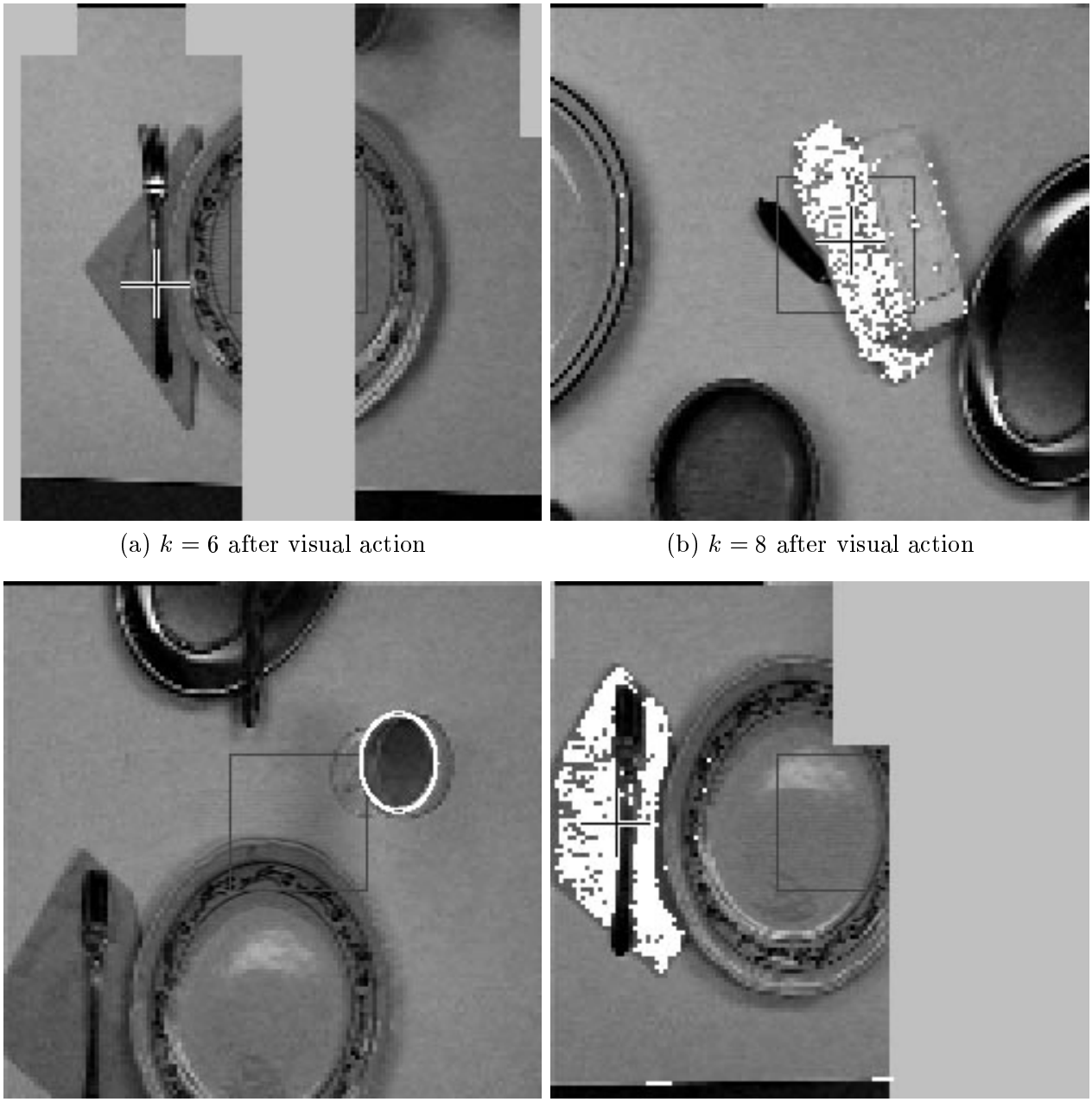
(a) $k = 0$ initial camera position

(b) $k = 1$ after visual action

(c) $k = 3$ after visual action

(d) $k = 4$ after visual action

Figure 12: Graphical display of results from actions during the example run on a fancy scene using the goodness function method.



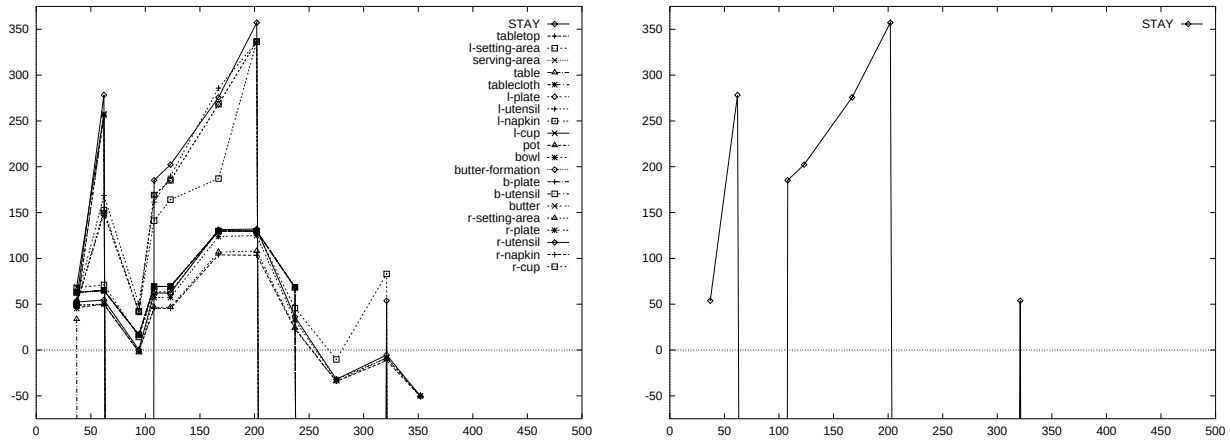
(a) $k = 6$ after visual action

(b) $k = 8$ after visual action

(c) $k = 9$ after visual action

(d) $k = 11$ after visual action

Figure 13: Graphical display of results from actions during the example run on a fancy scene using the goodness function method.



(a) (b)

Figure 14: (a) The value over time t of the goodness function $G(p)$ for moving the camera to location p during the example run on a fancy scene using the goodness function method. The locations considered are the centroids of the expected areas in the expected area net. When the system does not expect to have any actions to execute at position p then $G(p)$ is set to a large negative value. The value of $G(STAY)$, the goodness associated with not moving the camera, is included in (a) and plotted separately in (b).

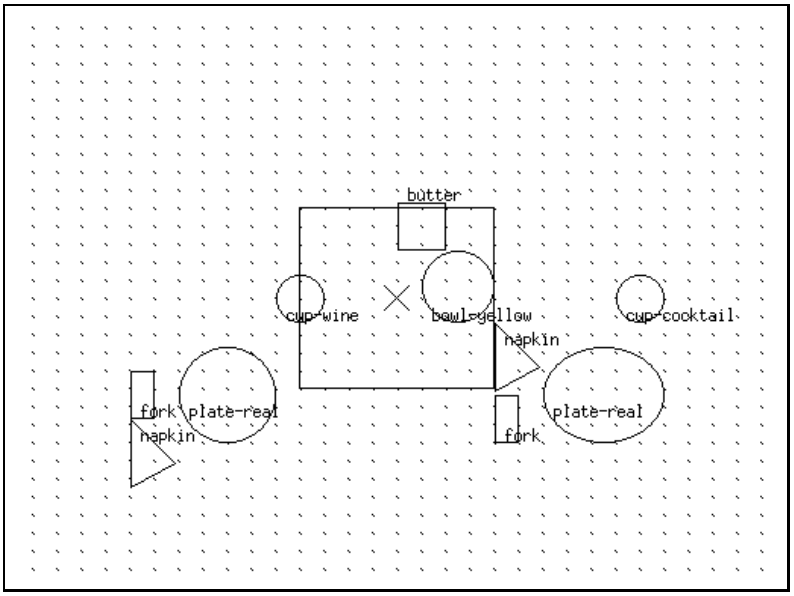


Figure 15: The simulated scene used in the example run on a fancy scene using the state-space search method. The large square shows the camera field of view.

k	t	task decision	α_{i_1}	α_{i_2}	α_{i_3}
1	0	notfancy	move to *r-utensil*	`per-detect-cup`	`per-detect-napkin`
2	23	notfancy	`per-detect-plate`	`per-detect-napkin`	`per-classify-plate`
3	38	notfancy	`per-detect-napkin`	`per-classify-plate`	move to *r-napkin*
4	46	notfancy	`per-classify-plate`	move to *l-napkin*	`per-detect-napkin`
5	61	fancy	move to *r-cup*	`per-detect-cup`	`per-classify-cup`
6	84	fancy	`per-detect-cup`	`per-classify-cup`	`per-detect-utensil`
7	89	fancy	`per-classify-cup`	move to *l-napkin*	`per-detect-napkin`
8	114	fancy	move to *l-napkin*	`per-detect-plate`	`per-detect-napkin`
9	142	fancy	`per-detect-plate`	`per-detect-napkin`	`per-classify-plate`
10	157	fancy	`per-detect-napkin`	`per-classify-plate`	`per-detect-cup`
11	167	fancy	`per-classify-plate`	`per-detect-cup`	`per-classify-cup`
12	182	fancy	move to *l-cup*	`per-detect-cup`	`per-classify-cup`
13	203	fancy	`per-detect-cup`	`per-classify-cup`	`per-detect-utensil`
14	208	fancy	`per-classify-cup`	move to *b-plate*	`per-detect-butter`
15	233	fancy	-	-	-

Table 4: Summary of decisions made during the example run on a fancy scene using the state-space search method. Each line of the table corresponds with one cycle of the main decision loop (indexed by k). Sequences of length $H = 3$ were evaluated. The best action sequence is $S = (\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3})$.

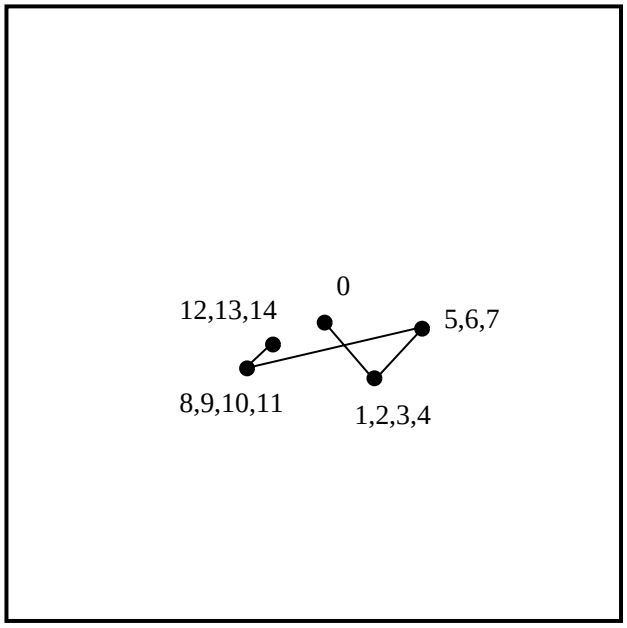


Figure 16: The position of the camera during the example run on a fancy scene using the state-space search method. The values of k, the index in the main decision loop, when the camera is at each location are also shown.

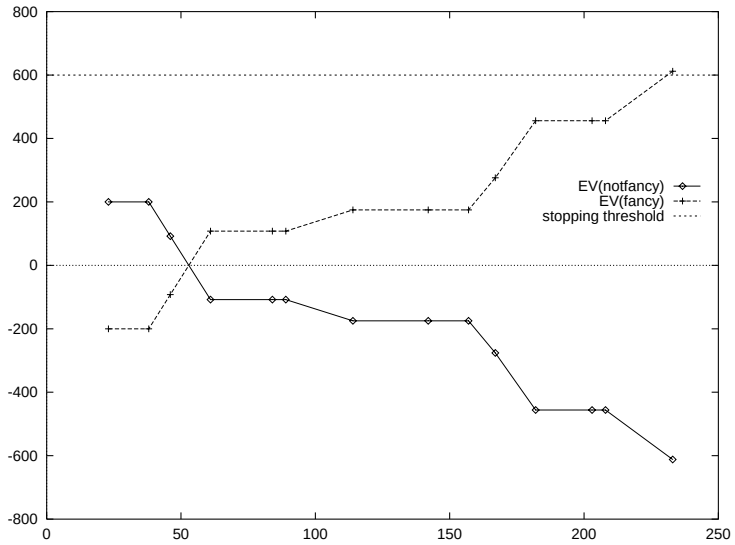


Figure 17: The expected values associated with the possible task decisions, $EV(notfancy)$ and $EV(fancy)$, over time t during the example run on a fancy scene using the state-space search method.

Number of objects in scene: 16
Grouping structure: 4 groups (no subgroups), 4 objects in each group
Average spatial dimensions of a group: 30% of scene dimensions
Spatial dimensions of the camera's field of view: 25% of scene dimensions
Cost of camera movements: $C_0^{camera} = 20$, $C_1^{camera} = 20$
Impact of an object's classification: uniformly chosen from the set (A, C, E, G)

Table 5: Key aspects of the default rules that define T-world problems used in the experimental analysis.

then simulated actions in the generated scenes. The same program created TEA-1's Bayes nets and action schemas for use with the generated scenes. As each key factor was varied, TEA-1's average solution time on ensembles of 10 generated scenes and tasks was recorded. Table 5 shows the default values used for key factors.

4.1 Scene Structure

Selective perception exploits scene structure. Some scene-structure geometric factors are: number of groups and number of subgroup levels that objects are organized into, spatial extent of groups, average number of geometric relations connecting to high impact objects, shape (and type) of geometric relation distributions between objects.

Figures 18 and 19 show how the performance of TEA-1 varies with group structure. In figure 18, sixteen objects are organized into N groups containing 16/N objects each. The average spatial dimensions of a group was set here to 20% of the scene dimensions, rather than the default of 30%, because 30% was too large to allow $N = 8$ groups to fit in the

scene. Each bar in (a) depicts the total amount of time taken while executing actions until a final decision was made about the solution to a task, and each sub-bar depicts the total amount of time spent executing camera movements. Thus the distance above the sub-bar is the time spent executing visual actions. Each bar in (b) depicts the total number of actions executed, and each sub-bar shows the number of camera movement actions executed. The chart in (c) shows the percentage of the scene viewed by the camera's field of view. Solution time increases when there are more groups because more camera movements are needed to move around among the larger number of groups, and because there are fewer constraining relations between objects in any one group (more relations would make it easier to find other objects in the same group). Figure 19 shows similar data for the spatial extent of groups (specified as a percentage of scene width), where solution time increases for groups with larger spatial extent, because more camera movements are needed to locate objects and because relations between objects in any one group are less constraining.

Another factor in T-world problems is whether all properties have the same impact on the task, meaning how much effect knowing the value of a specified property has on the value of the task variable. Impact is determined by the magnitude of the conditional probabilities in the task net. Figure 20 shows how the performance of TEA-1 varies when the average impact that an object has on the task is varied. The first four bars depict the results when all objects have approximately the same average impact, where the average impact increases from left to right. Stronger impacts yield significantly shorter solution times. Objects in realistic scenes have a wide variety of impacts. The rightmost bar shows the result when the impacts of the objects in a scene are an equal mix of the impacts used for the previous four bars. The solution time here is quite fast because the system performs an optimization that takes into account the expected impacts of (and costs of obtaining) object properties. (The values shown for the impact labelled A are extrapolated estimates, since TEA-1 could not reach a final decision at the required confidence level before running out of objects in the scene.)

4.2 System Parameters

The system parameters category includes: performance model of a visual action (a table of probabilities), relative costs of visual and non-visual actions, size of the camera's field of view, size of the fovea, relative speed of computation (in the multiprocessor version of TEA-1). Varying system parameters can change the best sequence of actions to execute. For example, making camera movements more expensive means that more time is spent analyzing more of the things visible at each camera fixation (figure 21).

4.3 Control Method

Figure 22 shows how the performance of TEA-1 varies when TEA-1 uses progressively simpler goodness functions for a visual action. The T-world problems were generated from the set of default rules. Each bar corresponds with a progressively simpler goodness function for a visual action, specifically: G0 denotes the full goodness function $G(\alpha)$ as given by equation (16), which incorporates two kinds of lookahead. In G1, the $G_2(EA_\alpha)$ lookahead

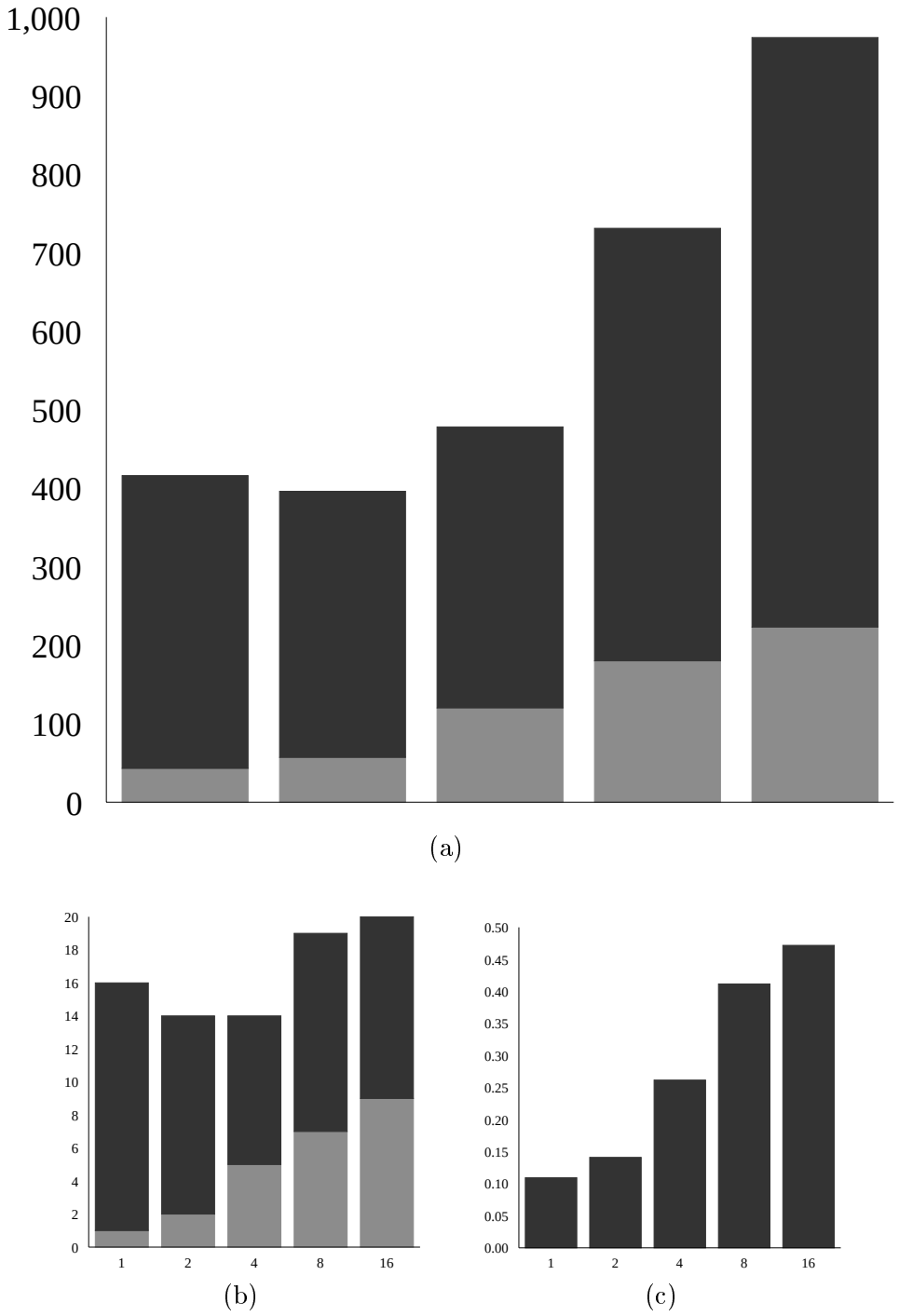


Figure 18: Performance when the number of groups is varied. (a) The solution time (the sub-bar is camera movement time). (b) The number of actions executed (the sub-bar is number of camera movements). (c) The percentage of the scene viewed by the camera.

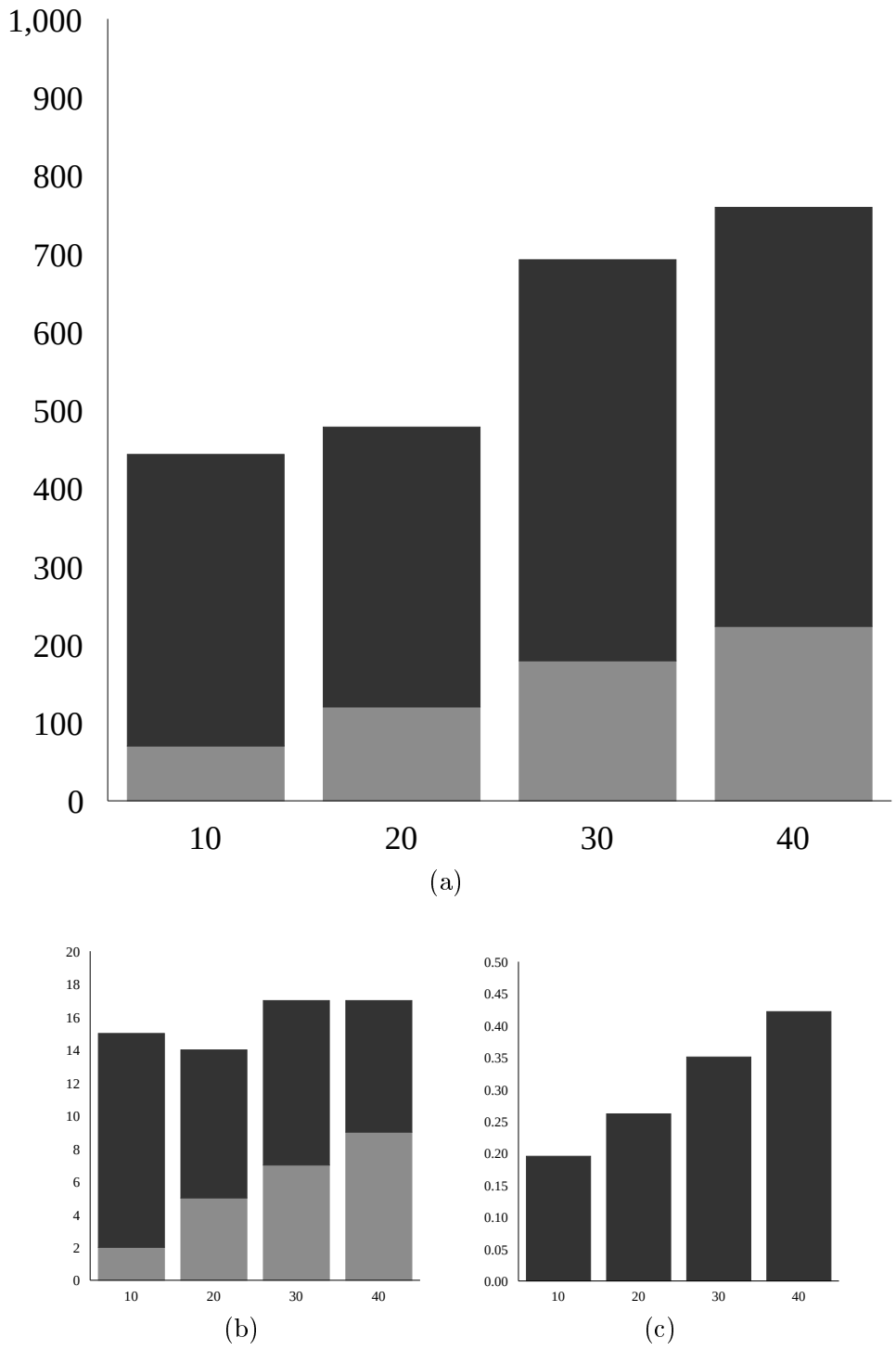


Figure 19: Performance when the average spatial extent of groups is varied. (a) The solution time (the sub-bar is camera movement time). (b) The number of actions executed (the sub-bar is number of camera movements). (c) The percentage of the scene viewed by the camera.

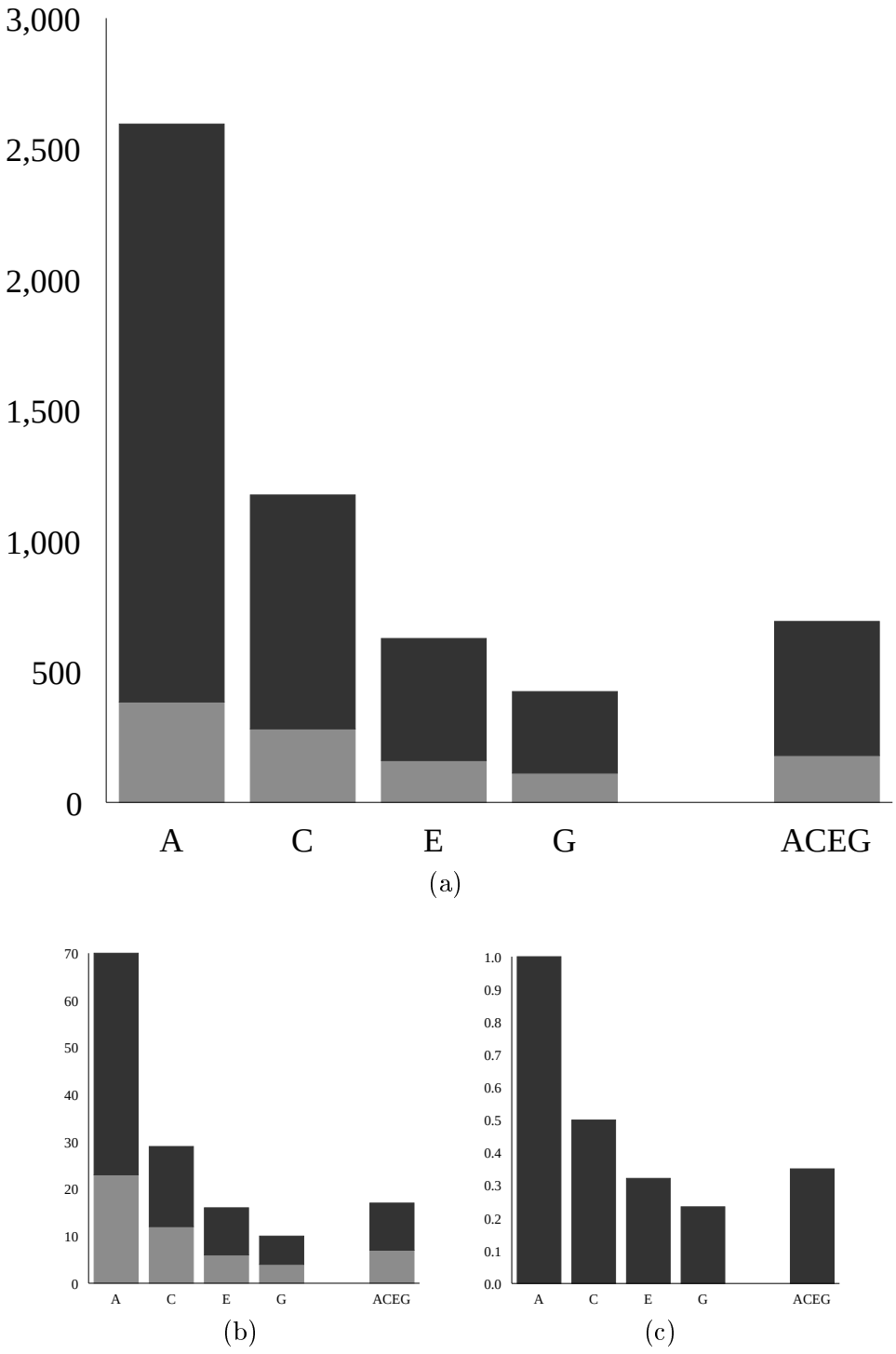


Figure 20: Performance when the average impact of an object's classification is varied. (a) The solution time (the sub-bar is camera movement time). (b) The number of actions executed (the sub-bar is number of camera movements). (c) The percentage of the scene viewed by the camera.

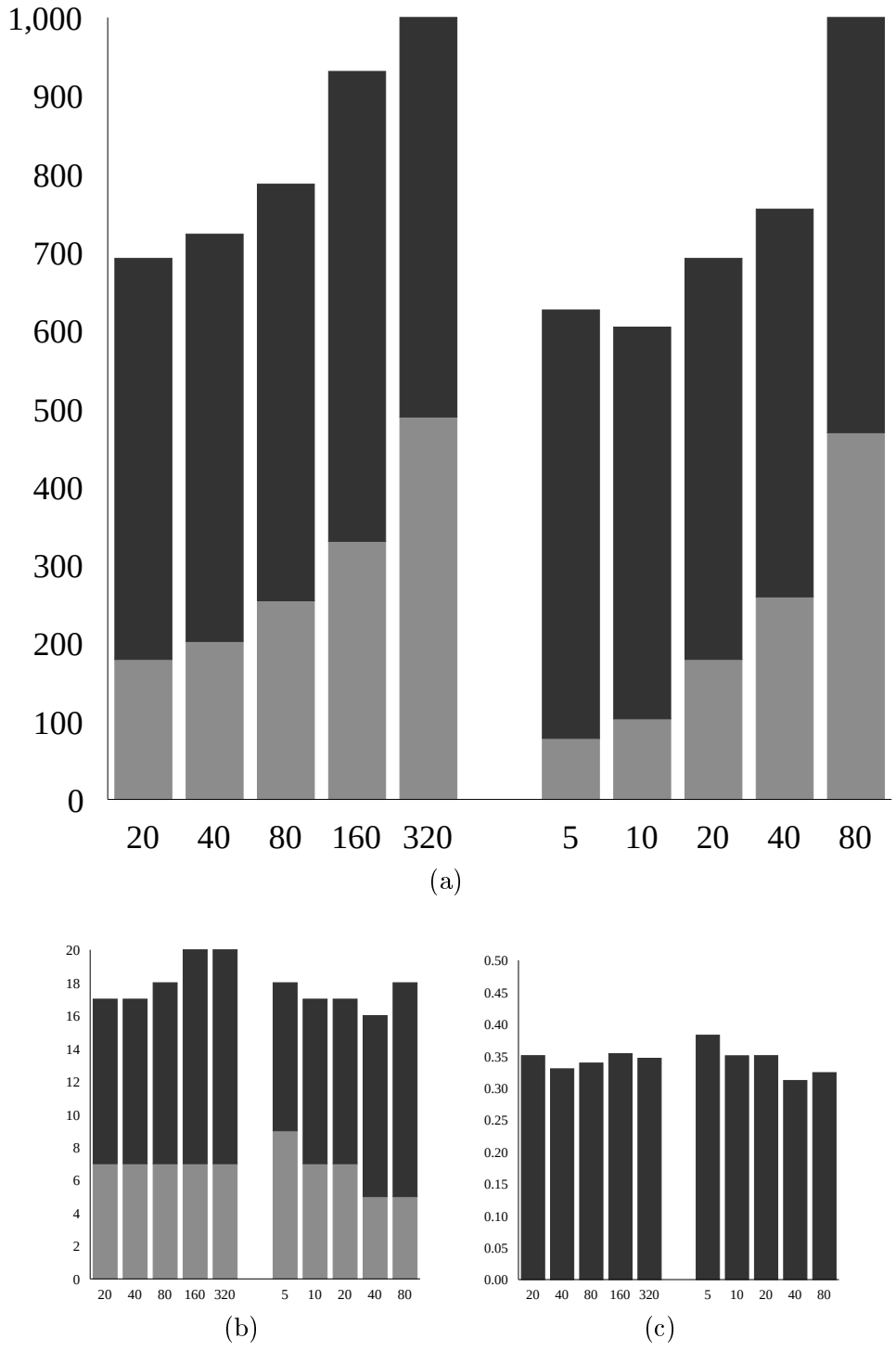


Figure 21: Performance when the cost of camera movement is varied. (a) The solution time (the sub-bar is camera movement time). (b) The number of actions executed (the sub-bar is number of camera movements). (c) The percentage of the scene viewed by the camera. C_1^{camera}, the cost parameter proportional to the distance moved, is varied inside the left-hand group of bars. C_0^{camera}, the overhead cost parameter, is varied inside the right-hand group.

term is set to a constant average value, so the goodness function no longer captures the future benefit of establishing geometric relations. In G3, the other lookahead term is also set to a constant, so the goodness function is essentially only $V(\alpha) - C(\alpha)$. In G5, the cost term is similarly removed, leaving essentially $V(\alpha)$. For G9 the goodness function was set equal to the cost $C(\alpha)$ times a constant factor (5). Finally, in G8 a random number, uniformly chosen between 0 and 200, was used for the value of the goodness function of each visual action during each cycle in the main control loop. The results generally show better performance as more terms are incorporated into the goodness function, demonstrating that these terms are all important for good performance.

5 Discussion and Conclusions

5.1 Knowledge Representation Using Bayes Nets

Tree-structured Bayes nets. TEA-1's basic framework can use general Bayesian networks. At this research stage, a system using trees sacrifices little in terms of principles demonstrated while gaining significantly in simplicity (the progression from trees to more general graphs is typical of Bayes net research). Understanding has now progressed to the point that any future work with TEA-like systems should probably use general networks. General-purpose software systems have recently become available that contain both the basic Bayes net algorithms (usually several versions, and for general networks) and graphical interfaces for creating, editing and examining networks.

Speed of belief propagation. Belief propagation for TEA-1's expected area net is slow, requiring a few seconds on a SPARCstation 10 for an expected area net with 20 nodes. Because each node represents a 2-D discrete random variable, typically with $32 \times 32 = 1024$ possible values, calculations based on the potentially 1024×1024-sized conditional probability matrix take significant time. The current design for expected area nets trades off speed in order to provide a completely general representation for geometric relations. Alternatively, geometric relations could be described using a 2-D parameterized Gaussian distribution. While a single Gaussian is a limited form of expected area, its belief calculation is very fast [Pearl, 1988]. Further, Gaussian mixtures could be used with only small combinatorial problems.

Belief propagation in a general net is an NP-hard problem [Cooper, 1990], but current belief propagation algorithms [Lauritzen and Spiegelhalter, 1988; Anderson *et al.*, 1989; Jensen *et al.*, 1990] are very fast in practice. Exact belief calculations for very large nets will always be slow. Approximate, Monte-Carlo style algorithms (see *e.g.* [Henrion, 1990; Peot and Shachter, 1991]) have promise of offering fast solutions even on large nets.

Bayes nets and explicit probabilities. A common criticism of Bayes nets is that many probabilities are required; see [Pearl, 1988] for several rejoinders to this point. The values required for all experiments with TEA-1 were specified by a human familiar with the application domain and tasks. They are approximate values only. Statistical trials could be used to provide values for the probabilities describing the performance of individual actions. Experience with TEA-1 suggests that the general behavior of the system is relatively

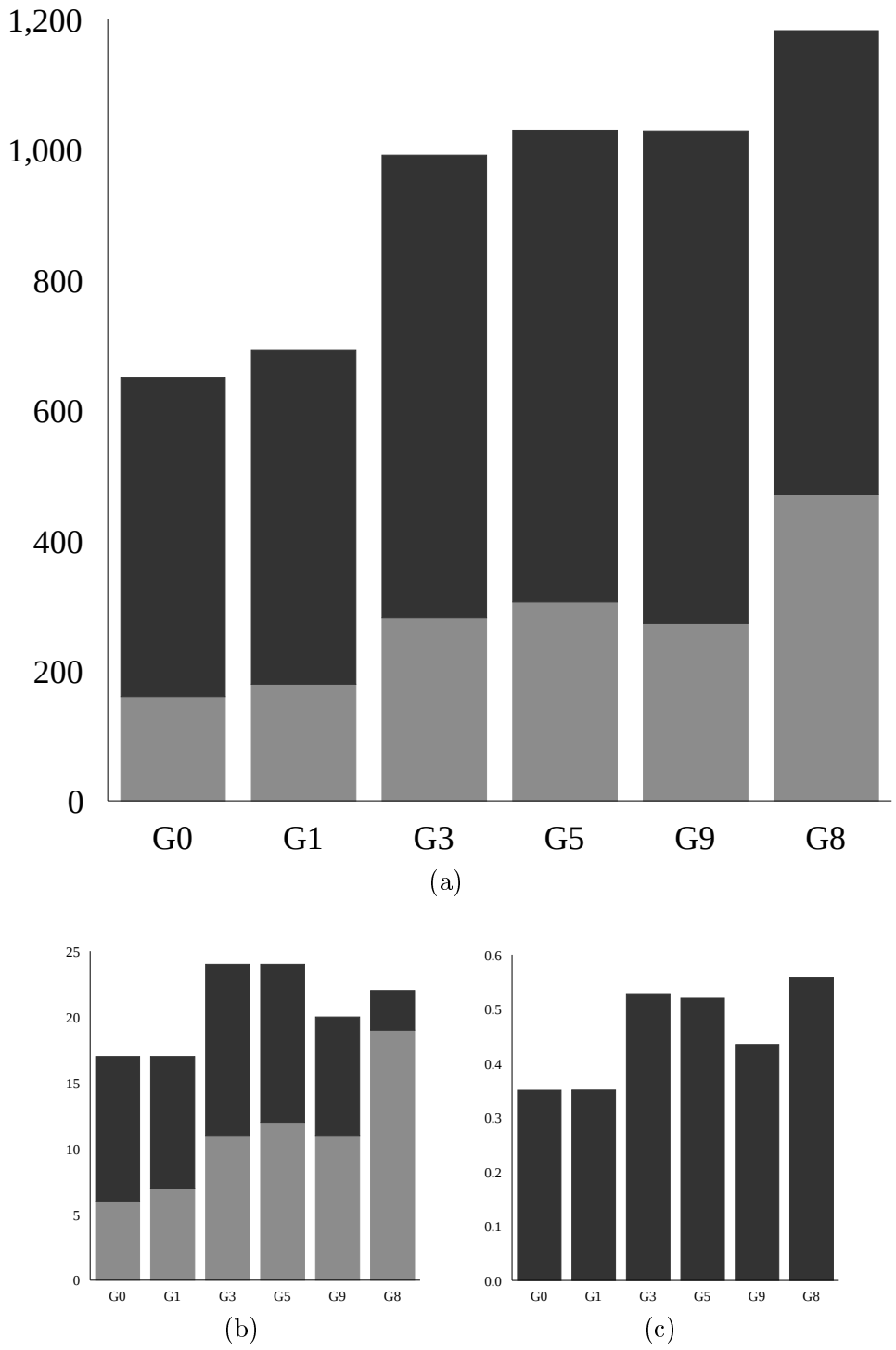


Figure 22: Performance when the goodness function for a visual action is ablated. (a) The solution time (the sub-bar is camera movement time). (b) The number of actions executed (the sub-bar is number of camera movements). (c) The percentage of the scene viewed by the camera.

insensitive to variations in the values of the supplied probabilities. It appears that larger nets will be even less sensitive.

Knowledge engineering is hard. Knowledge engineering was not the focus of this work, and TEA-1's representations could be extended in many ways:

- A general way to represent object properties (shape, dynamic properties, height, width, and other properties) is needed.
- Expected area nets are currently two-dimensional and do not model the results of object or camera rotations, scaling (in general), and perspective.
- The expected area for a node is calculated by combining "messages" about expected areas from its parent and all its children. 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.

5.2 Actions

Uncertainty in object location. TEA-1 needs a mechanism to deal with expected areas that are larger than the camera's field of view. Currently, if TEA-1 fails to detect an object from one camera position it will never again try to locate that object. TEA-1 doesn't deal with bimodal expected areas very well. For example, it could make a camera movement to the center space between the two modes.

Action engineering. The existing action schema in TEA-1 can accommodate a large variety of actions, but the schema could usefully be extended. As a start, the cost and performance models in TEA-1 should be extended to be functions of the resolution of the image data processed, the fraction of the field of view processed, and the amount of time spent processing image data. Movement of the fovea was performed by visual actions in an earlier version of TEA-1, and any new design should probably have separate fovea movement commands. Sensor platform movement actions (beyond pointing) and active vision algorithms (*i.e.* actions where processing of image data is tightly coupled with sensor movement) could also be added, but would probably require a significantly new design. Since future systems may represent object properties in the Bayes nets in a more general way, the way in which an action accesses information in the nets may need to be redesigned.

5.3 Decision Making

Classical decision theory and computer vision applications. 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 goodness 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. 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. A taxonomy or formal description of such interactions would be useful.

Control knowledge. Bayes nets represent domain knowledge, and goodness functions represent control knowledge in TEA-1. Influence diagrams capture only a simple form of control knowledge. In the context of a selective vision system like TEA-1 the field has very little idea what the control knowledge should be, and even fewer useful ideas about how to represent it. More complex goodness functions, with general re-usable structure, are needed.

Cost of planning. Performance is measured in this work by the total run time of the camera movement and visual actions executed by TEA-1 during a run. Making decisions and incorporating the results of actions (mainly belief propagation) also consume time during a run. A problem is that the time required for each of these three different things (decide, execute, incorporate) can essentially be scaled by arbitrary factors by using parallelism, better algorithms, *etc.*, so combining them into a grand total time would have little meaning.

5.4 Overall System

Does the system scale? The TEA-1 framework scales well to deal with more objects, more actions, and more knowledge.

Bayes nets are inherently highly structured, making it easy to build up large nets, and the conditional probabilities are easy to provide. The knowledge engineering process is being helped by the recent development of sophisticated software tools with powerful user interfaces. The TEA-1 framework of modular Bayes nets and the separation of domain from task knowledge also eases knowledge engineering, and TEA-1's action schema makes it easy to add actions. Domain knowledge can continue to accumulate between applications, each of which requires a new task net.

The task net guarantees that the execution time, performance of individual vision modules, and the overall performance are largely independent of how many objects there are in a scene. The system will never try to analyze all the objects in a scene. It always looks at select objects, in select areas of the scene, using visual actions that are sufficient, nothing more.

The major weaknesses to scaling are: 1) Belief propagation can take a long time with very large nets, and theoretically takes exponential time in the worst case, although in practice the algorithms run in useable times. 2) The goodness functions are currently

evaluated for every possible action and every possible movement target. Future work on the issue of control knowledge will address this issue.

However, the sheer size of a practical knowledge base is daunting. The work reported here avoids the knowledge engineering issues in favor of the decision and control issues. Future work will continue to focus on the control problem, but now that we have a system in place we can also begin to address the problem of what knowledge the system really needs. All these issues are *system-level* issues and must be studied in the context of complete systems. The increasing number of recent developments regarding the problem of learning Bayes net structures and probabilities from large data bases of examples may play a key role here.

Restrictive assumptions. The largest constraint in the current TEA-1 system is that the scene must contain predictable structure. For example, the absolute location of objects in the scene must be relatively close to the system's represented knowledge, *i.e.* currently within the width of the field of view. Relative geometric relations between objects serve to compensate for shifted objects, but doing so requires that the system can locate *some* objects in order to use those geometric relations.

5.5 Conclusions

General framework. The decision theoretic approach to computer vision is a promising research area. It provides a sound formal basis for designing and evaluating vision systems. We presented the TEA-1 system, an example of a general, reusable framework for a selective vision system using Bayes nets and decision theoretic techniques. TEA-1 addresses high-level computer vision, and is one of the first to consider how to control an actively pointable sensor and to emphasize purposive and sufficing control. Since TEA-1 is a fully implemented system, we have been able to perform extensive experiments in simulation and in the lab using complete runs on a large number of scenes.

T-world. The T-world problem includes a large class of vision problems, and is an adequate problem for studying some of the basic issues in selective computer perception. In this work, the key factors that make the selective perception approach appealing are explored by analyzing how each factor affects TEA-1's overall performance when solving a set of automatically generated (in simulation) T-world domains and tasks.

Task representation. TEA-1 represents tasks using Bayes nets, called task nets, and the TEA-1 approach can solve a variety of visual tasks (certainly ones in T-world). Task nets are easy to create and designing them is relatively independent from designing the domain knowledge. The system scales gracefully since the domain knowledge can be reused and incrementally enlarged for each application. Task-oriented behavior emerges from the combination of the representation and general decision making procedures.

Decision making. There are many approaches to control in a selective perception system ranging from brute-force and heuristic search through hand-crafted goodness functions to a formal planning system. This work presents and evaluates several goodness functions for T-world problems, and compares these to brute-force search solutions. TEA-1 represents and and reasons about the cost (execution time) of vision modules.

Selective vs. non-selective perception. This work presents an empirical analysis supporting the selective perception approach to computer vision. Control of selective perception is a kind of optimization problem, and we have presented experimental evaluation of some factors involved in that optimization process. Other support includes the theoretical analyses provided in [Tsotsos, 1989; Wixson and Ballard, 1993], and the empirical analysis for driving tasks in [Reece and Shafer, 1992].

Sufficing Vision. Limiting the semantic and geometric context of the vision problem makes it easier. Since the job of selective perception is to capitalize on domain and problem-instance knowledge, it can obtain reliable results from simple, robust vision operators.

Acknowledgements

Peter von Kaenel helped build the T-world simulator, worked on several of the vision modules and visual actions in the lab version, and implemented and performed experiments with *d*TEA-1, which extends TEA-1 to a simple dynamic domain, model trains. Tim Becker parallelized the belief propagation algorithm, helped modularize the T-world/TEA-1 system, and created the multiprocessor version. Martin Jagersand obtained the eye movement recording in Figure 2(b), using software developed by Jeff Pelz. We thank Lambert Wixson, Jim Müller, Dana Ballard, Randal Nelson, and the referees for their helpful comments on this work and drafts of this article.

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.

References

[Agosta, 1991] J. M. Agosta, *Probabilistic Recognition Networks, An Application of Influence Diagrams to Visual Recognition*, PhD thesis, Stanford University, 1991.

[Anderson *et al.*, 1989] S. K. Anderson, K. G. Olesen, F. V. Jensen, and F. Jensen, "HUGIN - A Shell for Building Belief Universes for Expert Systems," In *Proceedings: International Joint Conference on Artificial Intelligence*, pages 1080–1085, 1989.

[Ballard and Brown, 1992] D. H. Ballard and C. M. Brown, "Principles of Animate Vision," *Computer Vision, Graphics, and Image Processing: Image Understanding*, 56(1):3–21, 1992.

[Bolle *et al.*, 1990] R. M. Bolle, A. Califano, and R. Kjeldsen, "Data and Model Driven Foveation," In *Proceedings: IEEE International Conference on Pattern Recognition*, pages 1–7, 1990.

[Bolles, 1977] R. C. Bolles, "Verification Vision for Programmable Assembly," In *Proceedings: International Joint Conference on Artificial Intelligence*, pages 569–575, 1977.

[Burt, 1988] P. J. Burt, "Smart Sensing within a Pyramid Vision Machine," *IEEE Proceedings*, 76(8):1006–1015, 1988.

[Charniak, 1991] E. Charniak, "Bayesian Networks Without Tears," *AI Magazine*, 12(4):50–63, 1991.

[Chen and Mulgaonkar, 1992] C. Chen and P. G. Mulgaonkar, "Automatic Vision Programming," *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(2):170–183, 1992.

[Chou and Brown, 1990] 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.

[Clark and Ferrier, 1988] J. J. Clark and N. J. Ferrier, "Modal Control of an Attentive Vision System," In *Proceedings: International Conference on Computer Vision*, pages 514–523, 1988.

[Clemen, 1991] R. T. Clemen, *Making Hard Decisions: An Introduction to Decision Analysis*, PWS-Kent Publishing Company, 1991.

[Cooper, 1990] G. Cooper, "The Computational Complexity of Probabilistic Inference Using Belief Networks," *Artificial Intelligence*, 42(2-3):393–405, 1990.

[Dean *et al.*, 1990] T. Dean, T. Camus, and J. Kirman, "Sequential Decision Making for Active Perception," In *Proceedings: DARPA Image Understanding Workshop*, pages 889–894, 1990.

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

[Durrant-Whyte, 1988] H. F. Durrant-Whyte, *Integration, Coordination, and Control of Multi-Sensor Robot Systems*, Kluwer Academic, 1988.

[Elfes, 1992] A. Elfes, "Dynamic Control of Robot Perception Using Multi-Property Inference Grids," In *Proceedings: IEEE International Conference on Robotics and Automation*, pages 2561–2567, 1992.

[Feldman and Sproull, 1977] J. Feldman and R. Sproull, "Decision Theory and Artificial Intelligence II: The Hungry Monkey," *Cognitive Science*, 1:158–192, 1977.

[Garvey, 1976] T. Garvey, "Perceptual Strategies for Purposive Vision," Technical Report 117, SRI AI Center, 1976.

[Hager, 1990] G. D. Hager, *Task-Directed Sensor Fusion and Planning: A Computational Approach*, Kluwer Academic, 1990.

[Heckerman *et al.*, 1993] D. Heckerman, E. Horvitz, and B. Middleton, "An Approximate Nonmyopic Computation for Value of Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):292–298, 1993.

[Henrion, 1990] M. Henrion, "An Introduction to Algorithms for Inference in Belief Nets," In *Uncertainty in AI 5*, pages 129–138. North-Holland, 1990.

[Hutchinson and Kak, 1989] S. A. Hutchinson and A. C. Kak, "Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities," *IEEE Journal of Robotics and Automation*, 5(6):765–783, 1989.

[Jensen *et al.*, 1992] F. V. Jensen, H. I. Christensen, and J. Nielsen, "Bayesian Methods for Interpretation and Control in Multi-Agent Vision Systems," In *Proceedings: Applications of AI X: Machine Vision and Robotics*, 1992.

[Jensen *et al.*, 1990] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen, "Bayesian Updating in Recursive Graphical Models by Local Computations," *Computational Statistics Quarterly*, 4:269–282, 1990.

[Krotkov, 1989] E. P. Krotkov, *Active Computer Vision by Cooperative Focus and Stereo*, Springer-Verlag, 1989.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems," *Journal of the Royal Statistical Society B*, 50(2):157–224, 1988.

[Levitt *et al.*, 1989] 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.

[Levitt, 1986] T. S. Levitt, "Model-Based Probabilistic Inference in Hierarchical Hypothesis Spaces," In *Uncertainty in AI*, pages 347–356. North-Holland, 1986.

[Mann and Binford, 1992] 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.

[Pearl, 1986] J. Pearl, "Fusion, Propagation, and Structuring in Bayesian Networks," *Artificial Intelligence*, 29(3):241–288, 1986.

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

[Peot and Shachter, 1991] M. A. Peot and R. D. Shachter, "Fusion and Propagation with Multiple Observations in Belief Networks," *Artificial Intelligence*, 48(3):299–318, 1991.

[Reece and Shafer, 1992] D. A. Reece and S. A. Shafer, "Planning for Perception in Robot Driving," In *Proceedings: DARPA Image Understanding Workshop*, pages 953–960, 1992.

[Rimey, 1993] R. D. Rimey, *Control of Selective Perception Using Bayes Nets and Decision Theory*, PhD thesis, Department of Computer Science, University of Rochester, 1993, (Also available as Technical Report 468, Department of Computer Science, University of Rochester, December 1993).

[Rimey and Brown, 1991] R. D. Rimey and C. M. Brown, "Controlling Eye Movements with Hidden Markov Models," *International Journal of Computer Vision*, 7(1):47–65, 1991.

[Rimey and Brown, 1992] 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.

[Sarkar and Boyer, 1993] S. Sarkar and K. L. Boyer, "Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):256–272, 1993.

[Shachter, 1986] R. D. Shachter, "Evaluating Influence Diagrams," *Operations Research*, 34(6):871–882, 1986.

[Shafer and Pearl, 1990] G. Shafer and J. Pearl, editors, *Readings in Uncertain Reasoning*, Morgan Kaufmann, 1990.

[Tarabanis *et al.*, 1991] K. Tarabanis, R. Y. Tsai, and P. K. Allen, "Automated Sensor Planning for Robotic Vision Tasks," In *Proceedings: IEEE International Conference on Robotics and Automation*, pages 76–82, 1991.

[Tsotsos, 1989] J. Tsotsos, "The Complexity of Perceptual Search Tasks," In *Proceedings: International Joint Conference on Artificial Intelligence*, pages 1571–1577, 1989.

[von Kaenel *et al.*, 1993] P. A. von Kaenel, C. M. Brown, and R. D. Rimey, "Goal-Oriented Dynamic Vision," Technical Report 466, Department of Computer Science, University of Rochester, August 1993.

[Wixson and Ballard, 1993] L. Wixson and D. Ballard, "Using Intermediate Objects to Improve the Efficiency of Visual Search," *International Journal of Computer Vision*, 1993, This issue.

[Wu and Cameron, 1990] H. L. Wu and A. Cameron, "A Bayesian Decision Theoretic Approach for Adaptive Goal-Directed Sensing," In *Proceedings: International Conference on Computer Vision*, pages 563–567, 1990.

[Yarbus, 1967] A. Yarbus, *Eye Movements and Vision*, Plenum Press, 1967.