ELSEVIER Artificial Intelligence 78 (1995) 397–430

Artificial Intelligence

Control of perceptual attention in robot driving

Douglas A. Reece[a,*], Steve A. Shafer[b]

[a]*Institute for Simulation and Training, University of Central Florida, 3280 Progress Drive, Orlando, FL 32826-0544, USA*

[b]*Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Received March 1993; revised July 1994

Abstract

Computer vision research aimed at performing general scene understanding has proven to be conceptually difficult and computationally complex. *Active vision* is a promising approach to solving this problem. Active vision systems use optimized sensor settings, reduced fields of view, and relatively simple algorithms to efficiently extract specific information from a scene. This approach is only appropriate in the context of a *task* that motivates the selection of the information to extract. While there has been a fair amount of research that describes the extraction processes, there has been little work that investigates how active vision could be used for a realistic task in a dynamic domain. We are studying such a task: driving an autonomous vehicle in traffic.

In this paper we present a method for controlling visual attention as part of the reasoning process for driving, and analyze the efficiency gained in doing so. We first describe a model of driving and the driving environment, and estimate the complexity of performing the required sensing with a general driving-scene understanding system. We then introduce three programs that use increasingly sophisticated perceptual control techniques to select perceptual actions. The first program, called Ulysses-1, uses *perceptual routines*, which use known reference objects to guide the search for new objects. The second program, Ulysses-2, creates an inference tree to infer the effect of uncertain input data on action choices, and searches this tree to decide which data to sense. Finally, Ulysses-3 uses domain knowledge to reason about how dynamic objects will move or change over time; objects that do not move enough to affect the robot's decisions are not selected as perceptual targets. For each technique we have run experiments in simulation to measure the cost savings realized by using selective perception. We estimate that the techniques included in Ulysses-3 reduce the computational cost of perception by 9 to 12 orders of magnitude when compared to a general perception system.

* Corresponding author. E-mail: dreece@ist.ucf.edu.

1. Introduction

1.1. *The need to control perceptual attention*

Early mobile robot research concentrated on making a robot solve problems. These problems were expressed in symbolic form, having already been abstracted away from the real world. Computer vision research has attempted to match this view of a robot by developing general systems that can build complete symbolic descriptions of arbitrary scenes. This vision task has proven to be extremely difficult to accomplish, and when it is possible it is often at great computational cost. More recently, *active vision* has been seen as a more feasible way to provide robots with the information they need to solve problems. Active vision systems do not completely characterize the entire scene, but instead optimize sensor parameters, the field of view, and algorithms to extract specific information from the scene. Such vision systems depend on an external task to determine what information is needed.

We believe that active perception is essential for a mobile robot performing a complex task in a dynamic environment. In addition to choosing actions, a controller for such a robot must select perceptual actions to get information needed to perform the task. Thus perception and action selection should be indistinguishable parts of the same control function. In our research we have developed a robot control program that integrates perception and action selection. The program, called Ulysses, drives a simulated robot in traffic—a task that is fairly complex, in an environment that is dynamic and visually complex. Driving knowledge is encoded in such a way that Ulysses can reason about it to select an efficient set of perceptual actions. Ulysses considers how traffic objects could affect its choice of actions, requests appropriate sensory data, and commands maneuvers. Ulysses currently operates in a world provided by a detailed traffic simulator we developed called PHAROS; PHAROS simulates the perceptual actions and carries out the maneuvers. PHAROS also controls the other vehicles in the environment.

The central themes of this paper are illustrated by the following example. Consider the driving scene shown in Fig. 1. It shows a robot driving through a scene containing vehicles in various locations and poses. To recognize an arbitrary vehicle, a perception system would have to consider variations in vehicle shape, color, and illumination, as well as anomalies due to surface markings, reflections, transparent surfaces, occlusions, etc. The vehicles are different distances and directions from the robot, and so appear in different locations in the robot's image of the scene. A perception system attempting to find *all* vehicles in the scene would have to search all possible locations, ranges, and poses. To interpret the scene completely, the perception system would also have to locate and identify all other traffic objects, with similar variations. The combinatorics of all of these factors together make such *exhaustive* perception far too expensive for a robot driver to do in a dynamic situation. Even humans cannot do this.

Fortunately, it is not necessary for a robot driver to update its world model completely. Although traffic objects may be found in many locations throughout a

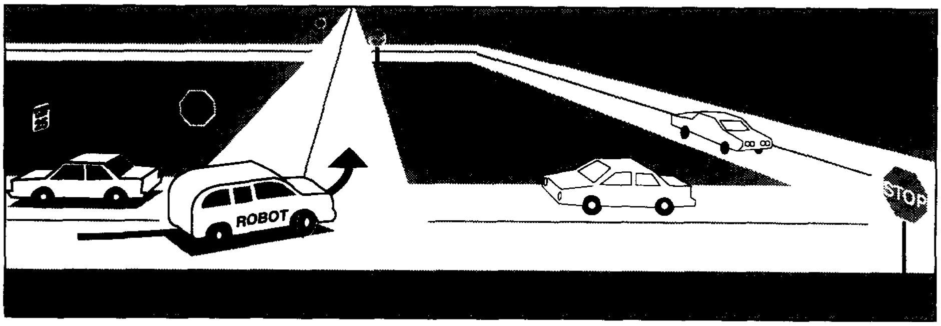

Fig. 1. A driving scene.

scene, it is wasteful to look for all of them because they are not all important to the robot. As the robot considers the various physical limitations, traffic rules, etc. that affect its choice of actions, it can look for relevant objects in the appropriate areas. Fig. 2 illustrates this limited visual search.

1.2. Driving and sensing with Ulysses

The Ulysses driving program represents driving knowledge well enough to allow Ulysses to understand many complex traffic situations and select safe actions. Driving knowledge also allows Ulysses to reason about perception and select perceptual actions. We present three specific methods for controlling perception. These methods have been implemented incrementally in three programs: Ulysses-1, -2 and -3.

(1) The **Ulysses** driving model encodes driving rules as a set of constraints and preferences. Constraints limit the robot's choice of actions to safe maneuvers. The preferences allow Ulysses to choose among safe maneuvers to further some objective. The constraints and preferences are represented explicitly as an inference tree with sense inputs on the bottom and an

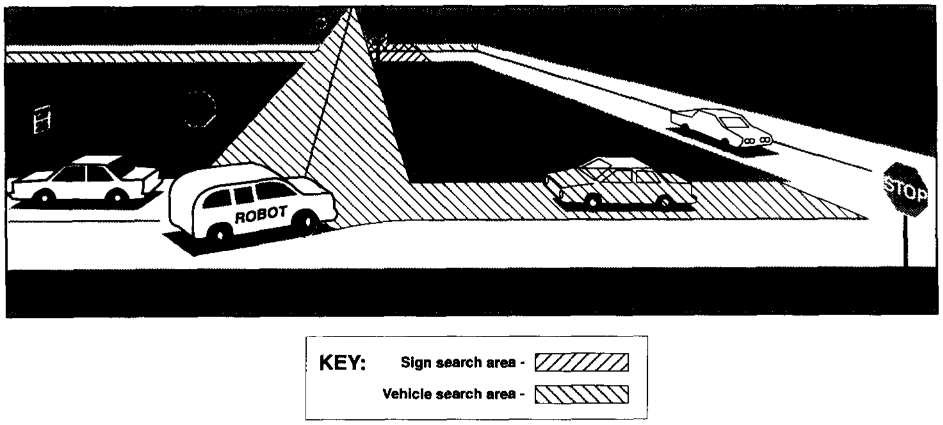

Fig. 2. Selective visual search.

action output at the top. The tree acts as a constraint network through which uncertain input values propagate to yield ranges of possible actions.

(2) **Ulysses-1** looks only in the appropriate parts of the scene for traffic objects instead of looking everywhere. Ulysses-1 uses task-dependent perceptual operators called *perceptual routines* to narrow its search area. The routines make use of the fact that the meaning of traffic objects depends on their spatial relations to other objects (e.g., the closest car *ahead in this lane*). The routines search for objects only near specific reference objects.

(3) **Ulysses-2** constrains search areas and also tries to find the minimum set of objects necessary to determine the correct action. Although Ulysses-1 uses knowledge of the traffic environment to constrain where it looks for objects, it does not consider the current situation to determine which objects are really important. Ulysses-2 reasons using the inference tree to determine the most critical input, and calls the appropriate perceptual routine to get the required data. The new information propagates up the tree to constrain possible actions. This selection and sensing process is repeated until there is no uncertainty about what action should be taken.

(4) **Ulysses-3** uses the same mechanism used in Ulysses-2 but also maintains a persistent world model. Facts in the model are time-stamped, and Ulysses-3 uses domain knowledge to reason about how object characteristics change over time. The inference tree reasoning process automatically determines when changing objects must be sensed again to reduce uncertainty.

In this paper we first present an estimate of the complexity of perception for driving to demonstrate why it is necessary for a driving robot to actively focus its perceptual attention. We then describe the encoding of the driving task and the three perceptual reasoning techniques used in Ulysses. The impact of this reasoning is estimated experimentally by running Ulysses in various traffic situations and measuring the simulated cost of its perceptual actions.

1.3. Related work

1.3.1. Active vision

The attention control that Ulysses performs is one aspect of *active* computer vision. Active vision comprises at least two levels of attention control, however. At a high level, active vision involves deciding which parts of the scene are most important for the robot to be looking at given its task. This is the level that this paper addresses. Our system considers where to look from a given viewpoint, rather than where to move a viewpoint. At a lower level, active vision concerns gaze stabilization, pointing control for object tracking, verging stereo, camera–hand coordination, automatic focus, camera motion and localized optical flow, etc. [5, 7]. Ulysses doesn't consider any of these issues, but does borrow the concept of perceptual routines from the lower-level *visual routines* of Ullman [65]. We will discuss routines more in Section 4.

The need to focus perceptual attention has been addressed in a basic way in many systems. Various planners have incorporated sensing operations into plans to find specific objects so that uncertainty could be reduced or operator preconditions verified [14, 25]. These systems focus attention statically, and are not suitable for a dynamic domain like driving. Hayes-Roth's patient monitoring system [38] reasons about computational resource limitations and adjusts *filters* to reduce the data flow from the sensors. While they are effective in applications with simple sensory processing requirements, filters are too simplistic for robotics applications.

A number of robot systems have used task knowledge to limit visual searches to small portions of the scene. Mobile robots have used these small "windows" so that they have less data to process when looking for signs or other objects [4, 36, 44, 63]. These systems require only minimal intelligent perception control though, since their tasks require that the robot always look for the same object in about the same place—for example, monitoring a lane line or watching for speed limit signs. Burt [15] also advocates using small windows, but with the ability to move around the image to probe interesting areas. However, Burt did not present specific methods for choosing where the windows should be placed.

Ulysses' use of reference objects to identify and limit search regions dynamically draws on the techniques used by other researchers for efficient object detection [13, 31, 49, 60]. In these systems a domain-specific model describes spatial relationships between objects. After a hypothesis has been formed from information detected in one part of the image, the spatial relationships can indicate an effective place to look to confirm or refute the hypothesis. Garvey called this two-phase search *indirect search*. Wixson [67] performed an analysis of the cost of searching for one object with indirect search as compared with direct search; he found indirect search to be more efficient in this case by a factor of 2-8.

As a vision system, Ulysses is most similar to intelligent agent control systems that use task-directed sensing. For example, Firby's RAP (for "Reactive Action Package") system [29] controls both perception and action for a robot. However, while this system may theoretically allow a designer to program complex visual search strategies, the reactive nature of RAPs seems better suited for simple sensing actions. Firby's examples illustrate sensing operations that verify preconditions of actions—e.g., finding an object about to be manipulated. The RAP system and Ulysses-3 both use a time-stamped data base and simple models of growing uncertainty to help select perceptual actions. The model in the RAP system is simpler, though; it uses only the age of information instead of modeling the related uncertainty in position, velocity, etc. Chapman's Sonja system [18] also reactively generates sensory actions. In contrast, Ulysses *deliberates* to try to find the optimum set of perceptual actions at each moment. We feel that this deliberation is a desirable step for automatically finding good sensing policies in complex situations, and that eventually the policies should be compiled into reactive rules.

Ulysses' use of deliberation to select perceptual actions is close in spirit to the body of work on decision-theoretic sensing (for example, [20, 31, 47, 59]). These systems deliberate in order to find the most *cost-effective* perceptual actions—actions that increase confidence in a hypothesis the most for the least cost. This approach contrasts with the deterministic selection approach of RAPs, Sonja, and Ulysses. While a probabilistic approach is attractive for a real-world problem, the hard constraints in our driving model suggested that a probabilistic decision tree would collapse to a deterministic one anyway. This is because we assume that the cost of having an accident is infinite. In addition, the probabilistic approach becomes computationally expensive in a complex domain like driving.

1.3.2. Driving

The driving task has been characterized as having three levels: strategic, tactical and operational [53]. These levels are illustrated in Table 1. Each level in general provides commands to the level below; the strategic level provides a route plan, a driving style, etc. to the tactical level, which in turn selects control behaviors for the operational level. While sophisticated planning programs and control systems have made substantial progress in automating the strategic and operational levels respectively, neither has addressed active perception. The strategic level is mostly symbolic so does not require high-bandwidth input from sensors, while the operational level does not require complex task planning and can rely on fixed sensing operations.

Ulysses addresses the tactical level. Models of tactical driving have come mostly from the domain of human driving. Michon identified three types of tactical driving models: task analysis, information flow control, and motivational [53].

Task analysis models (e.g., [50]) divide driving into many subtasks and describe each. Unfortunately, a description of component tasks alone is not sufficient for building a driving model. This is because a task list does not address the dynamic relations between tasks. The list does not specify how the driver chooses a task in a given situation, or whether one task can interrupt another, or how two tasks might be performed simultaneously (especially if they require conflicting actions).

Information flow control models are computer simulations of driving behavior. These simulations, such as SIMRO [19]. TEXAS [33], and NETSIM [28, 68], are capable of dynamically interleaving driving subtasks as required by the situation. However, these simulations are incomplete as driving models. For example, they

Table 1
Characteristics of three levels of driving

	Examples	Characteristics	Existing model
Strategic (high level)	Planning a route; Estimating trip time	Static; abstract	Planning programs
Tactical (mid level)	Determining right of way; Passing another car	Dynamic; physical	Human driving models
Operational (low level)	Tracking a lane; Following a car	Feedback control	Robot control systems

typically allow cars to change lanes instantly without having to drive across lane lines. These simulations do not represent other physical information such as road geometry, accurate vehicle location, or the location and appearance of traffic control devices (TCDs). Finally, these simulations do not address perception and do not contain knowledge of how to interpret traffic conditions from *observable* objects.

Motivational models are theories of human cognitive activity during driving. These models generally describe mental states such as "intentions", "expectancy", "perceived risk", "target level of risk", "need to hurry" or "distractions" (e.g., the models reviewed by van der Molen [66]). These states are combined with perceptions in various ways to produce actions. However, motivational models are typically not computational—they do not concretely show how to represent driving knowledge, how to perceive traffic situations, or how to process information to choose actions. Some researchers are now building computational models of human driving (for example, Aasman [2]). Ulysses is essentially a cognitive process model, but not modeled on human cognition.

2. The computational cost of general perception

As part of our study of selective perception, we have analyzed the cost of sensing the driving environment. This analysis serves two purposes: it illustrates why selective perception is essential in this domain, and it provides a basis for evaluating the selective perception techniques we introduce. While intuitive, qualitative arguments for selective perception indicate some benefits, the full impact of this approach is much more apparent when at least the order of magnitude of cost is known.

For the purposes of this analysis, we have hypothesized a generic scene interpretation system that uses standard image processing and interpretation techniques. Although there are now many vision systems that can find roads and cars directly in front of or behind a robot, and signs along side the robot's road, no one has yet built a complete system that can find all the objects needed for situation analysis (for example, [16, 34, 51, 69]. The road-finding work is mostly directed at lane-keeping control; vehicle detection is used for car following and collision avoidance, and does not look for vehicles approaching intersections on side roads.

We have analyzed the processing steps for exhaustively complete perception and developed a complexity expression that is polynomial in the number of pixels in the image. We estimated the coefficients of the polynomial from characteristics of the scene and of the processing algorithms. The number of pixels was estimated by analyzing the field of view and resolution necessary to see and recognize objects at the required ranges. The result is a numeric estimate of the number of operations necessary to perform the sensing. A detailed explanation of our cost model is contained in [56].

2.1. The environment

A general perception system has to find all traffic objects of potential interest to the planner. For the Ulysses driving model, these objects include roads regions, road markings, vehicles, traffic signs, and traffic signals. Road markings include lane lines, stop lines, arrows and letters. Vehicle velocity estimates are required. Traffic signs and signals must be detected even when they are turned away from the line of sight by about 45°. The robot must sometimes recognize Stop and Yield signs *from the back* at intersection [58]. The size of the object features determines the resolution needed to detect them, and the number of variations determines the number of models that must be matched against scene regions. We took the object characteristics from a standard highway design manual [27].

Since it is not practical to consider perceiving the entire world, we arbitrarily set a range limit on the sensor system. In some driving situations it may be desirable to see long distances ahead; for example, signals are supposed to be visible from 218 m away on a road with 100 kph traffic [27, p. 4B-11], and the sight distance needed for passing is given as 305 m at this speed [1, p. 147]. While Ulysses is capable of driving on simulated highways, for this work we have concentrated on arterial urban streets where the environment is more diverse. Since streets have lower speeds than highways, we have chosen 150 m as the perceptual range limit.

Our analysis of perceptual cost considers factors not explicitly represented in the environment modeled by PHAROS. The environment is assumed to have shadows, trees, clouds, occluding objects, textures, and reflections. These factors prevent us from using cheap recognition algorithms based on single, uniform features. For example, although standard traffic sign colors will be useful for segmenting out sign regions, there may be other signs or other objects that have the same colors. Additional information will be needed to distinguish the traffic signs from the other regions in the image.

2.2. Object recognition procedures

Our generic perception system has three main steps: feature extraction, model matching, and secondary matching.

Feature extraction includes image transformation, image segmentation, and region property computation. Image transformation includes steps such as transforming color space, detecting edges, converting range data to local coordinates, or making correlations for optical flow. Transforming the image and computing region properties (e.g. texture, moments, average color, etc.) both require performing operations on each pixel in the image. Segmenting the transformed image to find regions not only involves operations on each pixel, but comparisons between pixels and regions and between regions when assigning pixels to regions. Based on experiments we have performed with images of outdoor scenes, we believe that the number of regions is generally proportional to the number of pixels. Thus the cost of segmentation has a component that depends on the square of the number of pixels. We estimated the cost coefficients of these feature extraction algorithms from our experiences at CMU in perception for vehicle navigation and an examination of the literature [11, 22, 26, 30, 37, 39, 43].

The *model matching* step involves matching S scene features to M object model features [10]. The complexity is M^S in the worst case, but can be reduced substantially by introducing heuristics such as constraints and object hierarchies [26, 35]. The complexity of matching with heuristics is not fixed because it depends on how well the heuristics work in a given domain. We assumed that heuristics would prune the search tree strongly so that after two model features were matched only one scene feature would match each remaining model feature. In this case the complexity is only proportional to S^3, which is roughly proportional to the number of pixels cubed.

We assume a *secondary matching* step to find details on traffic objects that have been recognized in the primary model matching step. Two-step matching avoids having to examine the entire scene at high resolution. The cost of secondary matching is still comparable to primary matching, however, because the reduction in scene area is almost countered by the increase in resolution. Surface markings in particular require very high resolution to detect at maximum range.

The resulting complexity expression for general driving perception is approximately

$$10^4 p + p^2 + 10^{-5} p^3 , \tag{1}$$

where p is the number of pixels in the image.

2.3. Pixel count

The number of pixels in the processed image is basically determined by the required field of view and the required resolution. We assume that a general driving perception system would have to see 360° around the robot, but only 45° up and down. The angle subtended by a pixel must be no larger than the smallest required scene feature; we estimate that this will be a 30 cm patch of road. Such a patch subtends only $(1.5 \times 10^{-3})°$ at 150 m range when viewed from an assumed sensor height of 2 m. We further assume that the perception system will take advantage of the fact that objects will be in a vertical range of about 0 to 7 m high. Thus when the sensors are pointing down at 45°, they are looking at markings that are very close (and therefore big). The system can thus use larger pixels at high and low sensor angles. The net result is that approximately 8.1×10^7 pixels are required to analyze the whole scene. Using this pixel count in Eq. (1) above, the total cost to analyze a single image is approximately 9×10^{18} operations.

Ulysses is assumed to process a new image every 100 milliseconds; so Ulysses would require about 10^{20} operations per second if it used naive, exhaustive perception. A "fast" computer that can perform a billion operations per second would thus be almost 11 *orders of magnitude* too slow to analyze the scene. Even if these estimates are off by several orders of magnitude, it is clear that a general approach to perception is intractable and that the driving robot must sharply limit its perceptual focus of attention.

3. A model of robot driving

We are studying selective perception in the context of a real-world task: driving. Actually, we are developing selective perception *together with* a robot driving program because for this sort of real-world activity, perception is intimately linked to the task. This section describes our driving program, Ulysses [58]. The discussion here is devoted mostly to the task of selecting maneuvers; we describe perception control in later sections and refer to the system as Ulysses-1, -2, or -3. We begin by describing how some of the driving knowledge is encoded in Ulysses. Next we present the architecture of Ulysses. Finally, we describe PHAROS, the simulator used to develop and test Ulysses.

3.1. Tactical driving knowledge

Tactical driving is a complex task that requires the robot to consider roads, surface markings, signs, signals, and other vehicles in various locations. The meaning of these objects varies depending on where they are relative to the robot and each other. Some of this complexity is illustrated in Fig. 3. In this figure, the robot is required to yield by the nearby sign, but it must make a judgment about the distance, speed, and likely constraints on the vehicle to the right to decide whether it (the robot) should stop. In addition, if the robot does not stop, it must consider its intended maneuver, the signal and the approaching car at the next intersection.

In our system, Ulysses controls robot maneuvers by repeatedly selecting an acceleration and a lane command for the robot. Ulysses' driving knowledge is encoded in constraints and preferences for each of these commands.

Constraints generally encode traffic and safety rules that limit the robot's actions. For example, acceleration is constrained so that the robot speed stays below the speed limit, the robot does not enter an intersection when it does not have right of way, it can stop before the end of the known (detected) road, and it can slow down without hitting the car in front should that car brake. Lane choice is constrained by traffic in adjacent lanes, and the compatibility of the lanes' allowed turn maneuvers with the robot's intended maneuver. These constraints are combined by taking their logical intersection.

Preferences select among the remaining allowed actions to further goals such as "go as fast as possible". Of the allowed acceleration values remaining, Ulysses prefers the highest one. For lanes, traffic flow speeds in different lanes and intended turns at distant intersections determine preferences. Ulysses uses the relevant preference with the highest priority to make a final selection. This constraint- and preference-based knowledge representation scheme was inspired

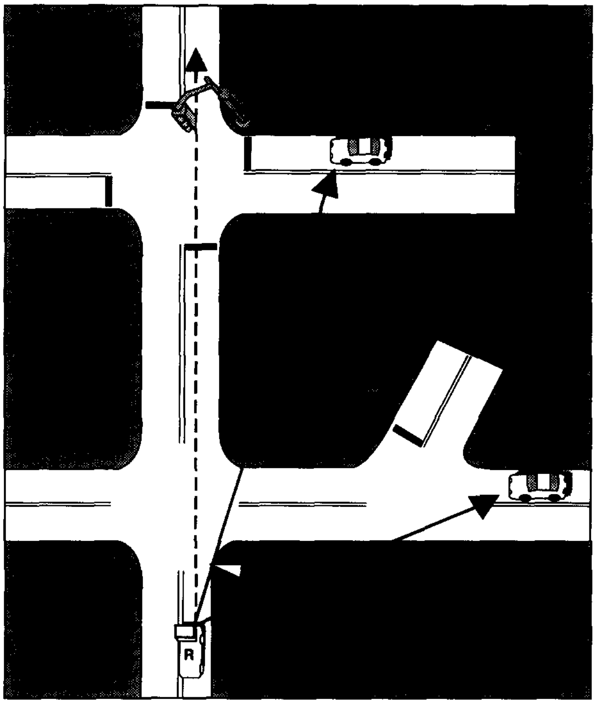

Fig. 3. Example of a tactical driving task. A robot driver (R) is approaching a crossroad.

by similar action selection mechanisms in other systems such as Soar [45] and Prodigy [17].

Constraints operationalize not only traffic laws but "self-preservation" goals as well—staying on the road and avoiding collisions. These goals introduce basic acceleration constraints to avoid loss of control. For example, the robot's speed is limited on curves so that the lateral acceleration is within the limits of the vehicle. This sort of constraint could be implemented at the operational level rather than requiring the tactical level to deliberate about it. However, safe driving often requires recognizing that conditions will change. Ulysses can do this by looking ahead for features. This prediction is not within the scope of operational control, especially if conditions are detected indirectly (e.g. by reading warning signs).

The perception system may not be able to detect objects reliably beyond some range. Ulysses deals with this uncertainty by making two assumptions: first, that there is a known range within which perception is certain; and second, that objects and conditions that trigger constraints are always present just beyond this range. The latter assumption is the worst case and results in conservative decisions. For example, Ulysses always assumes that the road ends just beyond road-detection range.

While the Ulysses driving model produces fairly good driving behavior, it still suffers from several limitations when compared to the general capabilities of humans.

- *Range of objects.* Ulysses only understands a finite set of objects. It would be unable to figure out the meaning of, say, a new traffic signal that used text or a new icon to indicate its meaning. Ulysses treats unknown objects (including bicyclists and pedestrians, currently) as obstacles to be avoided.

- *Structural abstractions*, Ulysses makes heavy use of the lane structure in roads. It would be unable to understand an open parking lot or take a shortcut across traffic lanes. It also makes use of TCD conventions, and would miss unconventionally located signs and markings (as would many humans). For example, Ulysses does not look on the left side of the road for signs.
- *Perception*. Ulysses assumes perfect perception, within the limits mentioned above. Allowing increased uncertainty requires addressing the idea of risk in driving. Section 7 discusses this further.
- *Fixed parameters*. Ulysses uses many fixed parameters: the braking capability of the vehicle, the reaction time margin to allow other drivers, etc. A more sophisticated model would allow these to change as environmental and traffic conditions changed.
- *Reasonable driver behavior*. While Ulysses watches for other drivers making sudden lane changes and running red lights, it nevertheless assumes mostly reasonable driver behavior. It is always possible for the driver of an oncoming car to suddenly swerve a few feet and cause a head-on collision. Since there is no way to accommodate such behavior, Ulysses ignores the possibility.

3.2. Driving program architecture

Fig. 4 shows the basic structure of Ulysses and how it interfaces to the strategic and operational levels. The strategic level provides a route plan, while the operational level implements commands. In the current implementation routes

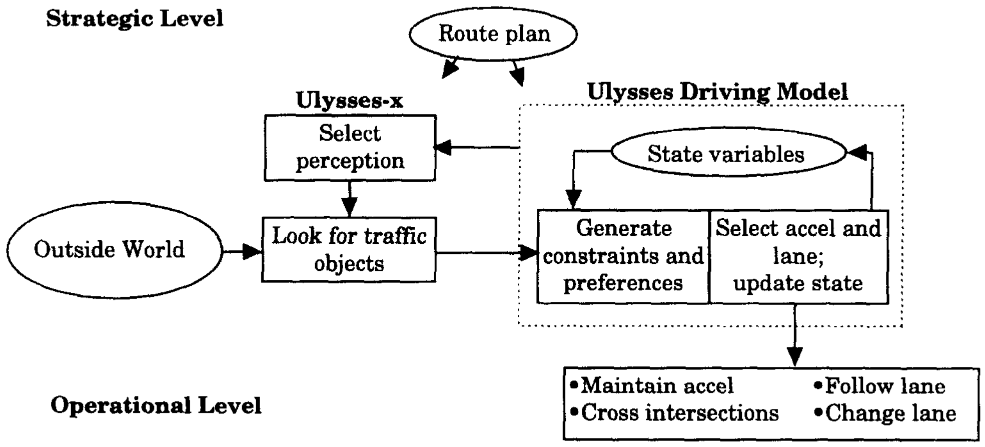

Fig. 4. Architecture of Ulysses and interface to other driving levels.

are created by a human operator and given directly to Ulysses. The plan is assumed to be based on an accurate map of the street network, so the desired maneuver at every intersection is known ahead of time. The operational level is currently assumed to be able to maintain a specified acceleration, follow a lane, drive to a lane across an intersection, and change lanes smoothly. This three-level architecture is analogous to robot control designs proposed by many other researchers to connect symbolic planning and numerical control functions (for example [6, 8, 32, 48, 52]).

The tactical driving program takes information about traffic objects and selects actions using constraints and preferences (see below). In Ulysses-1, -2, and -3, information about what has been seen so far is used to help select additional sensing actions. The route plan is used both to select maneuvers at intersections and to guide the selection of relevant traffic objects. The tactical level uses a simple model of operational level capabilities—maximum acceleration at different speeds, maximum braking rate, time to complete a lane change, etc.—so that it can produce commands that the operational level can execute.

For this research the tactical level of the robot is assumed to execute a simple "sense-plan-act" cycle. At the beginning of the cycle, the robot senses the world; next, it chooses an action; and finally, it executes the action. Ulysses maintains only a few bits of state between cycles, getting fresh information each cycle by sensing. The cycle period is 100 milliseconds. We assume that the system can look for whatever it wants and finish deliberating within the 100 milliseconds. Such an ideal system can respond perfectly to changing situations in this domain—both because it will quickly notice anything that changes and will determine the best possible reaction (that it knows). At the same time, this system is very simple and allows us to concentrate on the issues of selective perception.

An important question about such a simple, ideal system is how it relates to real systems. Can a robot really deliberate enough in 100 milliseconds to drive competently? AI planning research has dealt with deliberation complexity in a variety of ways—guaranteeing that action selection takes only bounded time (e.g. [42]), using algorithms that provide a coarse result at first and incrementally improve (e.g. [24]), reasoning about time required for planning (e.g. [40]), or speeding up performance over time (e.g. [12]). Robot builders, on the other hand, have often found that perceptual processing requires far more computation than "planning" on real mobile robots [61]. We have found that Ulysses' decisions can easily be made in real time. Therefore in this research we assume that tactical driving decisions *can* be made in a short period without metareasoning about planning time, etc.

3.3. Simulation environment

In order to develop Ulysses, we have built a detailed, microscopic traffic simulator called PHAROS (for Public Highway And ROad Simulator) [57]. PHAROS provided the usual benefits of simulation for experimentation: convenience, controllability, repeatability, and safety. It also allowed us to explore robot driving in parallel with efforts by others to develop operational control and perception for driving (see for example [64]). Another important function of PHAROS was to define the model of the physical, observable environment. Developing such a model is a necessary step in modeling the overall driving task. The environmental model determines what objects are important, how the world is abstracted, and what information can be obtained via perception. PHAROS uses a fairly detailed description of the street environment, including geometric information indicating the shape of streets, the appearance of road markings, the location of signs, and the configuration of traffic signals at intersections.

Vehicles in PHAROS (which we call "zombies") basically use the Ulysses driving model. There are two key differences, however. First, the zombies do not use perception, and in fact can look directly in the database to find information that would otherwise have to be inferred. Second, the zombies are intended to behave as humans would, so reaction delays are incorporated into their behavior.

PHAROS has a network-based interface so that a robot driving program can control one vehicle in the simulation. The interface includes both perceptual queries and lane and acceleration commands. Ulysses therefore has a limited access to information in the environmental data base and cannot "cheat" to get unrealistic information. PHAROS generates an animated, bird's eye view display of the vehicles. This graphical output provides us with our primary means of evaluating the behavior of both zombies and the robot.

4. Ulysses-1: perceptual routines

This section describes Ulysses-1, the simplest version of Ulysses. Ulysses-1 incorporates a mechanism—perceptual routines—for the reasoning module to request specific perceptual actions to sense important objects. These routines are task-specific to allow the perception module to search the scene more efficiently. Below we explain what perceptual routines are and show how routines reduced perceptual cost in simulated driving situations.

4.1. What are perceptual routines?

Tactical driving requires information about spatial relations between objects. For example, a driver might want to know if there is a car approaching a downstream intersection from the right. There are at least two ways for a robot to find the objects in the desired relations. One way would be to detect all cars, and then test each car to see if it was on the road to the right at the intersection. "The road to the right" would itself be found by finding all roads and checking to see if they connected to the robot's road at the intersection in question. This method would be very difficult because there may be many cars and roads in the robot's field of view, and each one requires significant computation to find and test.

Ulysses-1 (and the later implementations of Ulysses as well) uses a different technique to detect specific objects in specific relations to one another. The

Table 2
Perceptual routines in Ulysses-1

Find-current-lane	Find-path-in-intersection
Track-lane	Find-signal
Find-next-car-in-lane	Distance-between-marks
Find-intersection-approach-roads	Mark-adjacent-lane
Find-crossing-cars	Find-next-sign
Profile-road	Find-next-car-in-intersection
Find-next-lane-marking	Find-back-facing-signs

perception subsystem uses the reference object and a relation to focus its search for a new object. In the example above the robot would use a routine perceptual action to look *along the corridor ahead* to the intersection, *look to the right of the intersection* to find the approach road, and then *look along the road* for a car. The car detection process is thus limited to a specific portion of the field of view, determined by the location of the road. This perception process is inherently sequential; the focus of attention moves across the image from or along a reference object until a target is located. Ulysses-1 uses about a dozen of these routines, as shown in Table 2. Most of the routines *mark* objects and locations when they finish, so Ulysses-1 can continue finding objects later. For example, the Track-lane routine stops scanning when an intersection is encountered (indicated by the change in lane lines in the well-marked PHAROS world), but marks the intersection so that Find-path-in-intersection, Find-signal, etc. can find objects relative to that intersection.

Our perceptual routines are related to the visual routines proposed by Ullman to explain human vision [65]. These visual[1] routines are composed of operations such as shifting processing focus, finding unique locations in the image, tracing the boundary of a contour, filling a region to a boundary, and marking points. The routines are invoked from the top-down when needed in different tasks. Agre and Chapman used visual routines in their video game-playing systems, Pengi [3] and Sonja [18]. In these systems the visual routines find objects with certain spatial relationships, for example "the block that the block I just kicked will collide with". The routines in Ulysses-1 assume more domain-dependent processes are available to the routines than is the case in Pengi; for example, the perceptual routines must understand how to look for a sign rather than just any object.

The routines listed in Table 2 are specialized for the driving task. Perception systems for other tasks would clearly have to use routines specialized for those tasks. This idea of task-specific vision is similar to the task-oriented vision described by Ikeuchi and Hebert [41]. Although it may seem that driving decisions could be made with simpler, more general sensing functions, this is not generally true. For example, a low-level operator to "detect objects approaching

[1] We use the term "perceptual routines" instead of "visual routines" in our work to emphasize that they may include higher levels than just low-level vision, and that they may include other sensing technologies.

from the right" could not determine which road the cars were on or whether they had the right of way.

4.2. The cost of using routines

The perceptual routines are subject to the same environmental conditions as a general, naive perception system, so must pay the same feature extraction costs. However, the use of routines can reduce perception costs in several ways:

- *Reduced search area.* The azimuth and elevation angles swept by the routine are much smaller than the area searched by the naive perception system.
- *Range-limited resolution.* The maximum depth reached in the routine's search area limits the resolution required.
- *Object-limited resolution.* The routines look for only one type of object at a time; resolution is determined by the size of that type, not by the smallest of all types.
- *Limited features.* Since the routines look for only one type of object at a time, they may be able to use a small set of features that are specific to that object type. Only these features need to be extracted from the image.

The effectiveness of these reductions depends on the situation; for example, if the robot used several routines to search the same area for different objects, it would not get the benefit of limited features or object-limited resolution.

We can estimate the cost of perceptual routines by developing an expression that is dependent on the number of pixels, as we did in the previous section. We use the same general assumptions about feature extraction algorithms, sensor placement, traffic object characteristics, pixel sizes at various ranges, etc. that we used for the general perception system. However, since each routine looks for specific traffic objects, each uses different values for some parameters in the cost equation. For example, routines that search only for features on the road use smaller coefficients for the p and p^2 terms to reflect the reduced complexity of understanding two-dimensional features in a plane. Further details of the cost expression for individual routines are given in [56].

Once we have an expression for the cost of each routine, the number of pixels processed can be plugged in to compute cost. However, since the number of pixels depends on the area of the scene actually scanned, the cost of the routine depends on the particular situation. The perceptual interface in PHAROS is able to determine the extent of a routine's scan area and estimate the number of pixels required to examine it; thus we can measure the dynamic cost of routines in simulated driving situations.

4.3. A driving scenario

We can now show how the estimated perceptual cost varies continuously during driving. We have created a situation in which the robot is driving on a four-lane artery and must turn left on a side road. The scenario requires the robot to

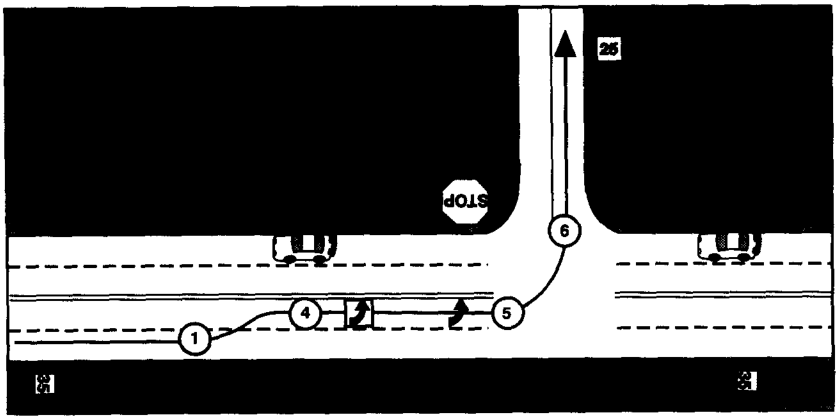

Fig. 5. Left side road scenario (not to scale).

change lanes, wait for oncoming traffic to pass, and then make the left turn. Fig. 5 shows the situation and the path of the robot.

Fig. 6 is a graph of the estimated perceptual cost over time for the left turn

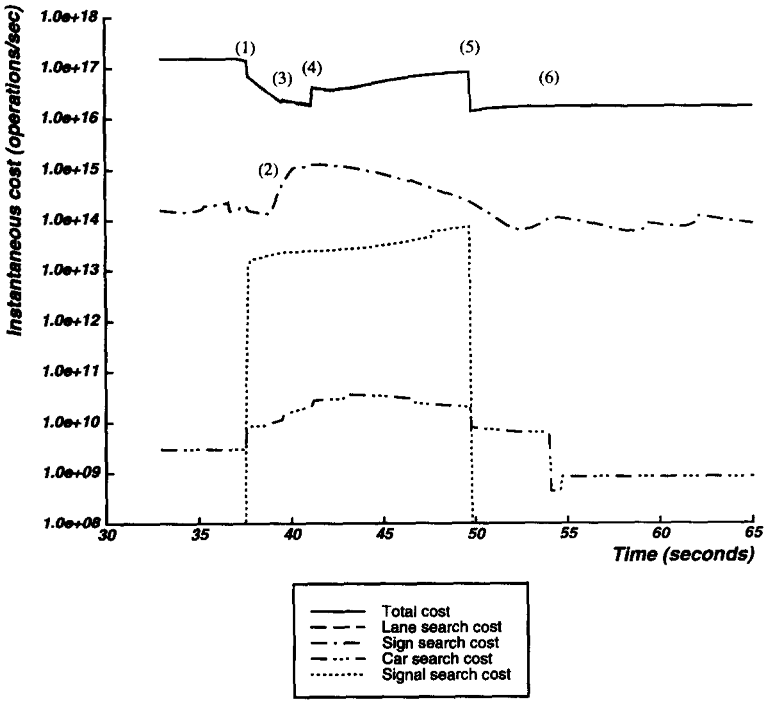

Fig. 6. Perceptual costs during scenario. Numbers correspond to figure above, except: (2) sensors reach robot's destination street on far side of intersection; (3) sensors reach all intersection approach roads. Total cost is nearly the same as lane search cost.

scenario. The numbers in this figure correspond to reference points in Fig. 5. The cost of perception is dominated by the lane search cost, which in the figure is indistinguishable from the "Total cost" curve. Searching the lane is the most costly type of activity because foreshortened lane and stop lines are the smallest objects in the environment. Since Ulysses-1 sometimes has to search out to its sensor range limits, the cost of finding lanes can be very high.

The perceptual cost before point (1) is approximately 1.5×10^{17} operations per second; this represents the cost of a four-lane highway without intersections. At point (1), the sensors detect the intersection and trigger additional searches for signals, approach roads, and blocking cars. The figure shows that the signal and car search cost jumps up here. Ulysses-1 also determines that the robot cannot turn left from the right lane and begins a lane change. During this time, the robot stops searching the adjacent lane for traffic, so the cost of searching lanes drops sharply. At point (2), the sensors detect the robot's corridor on the far side of the intersection and begin searching for signs and cars on that road. At (3) the sensors can detect all approach roads, so the robot begins to search these roads for lanes, cars, and backward-facing Stop and Yield signs. After the lane change is complete at (4) the robot again watches both lanes of the road and the lane search cost increases accordingly. At point (5) the robot enters the intersection and ceases its search for signals, approaching cars, and other objects that would affect its right-of-way decision. At (6) the robot no longer needs to look for unexpected cross traffic in the intersection either.

In addition to this scenario, we have created several other driving situations and used Ulysses-1 to drive a robot through them [56]. They include the following:

- A four-way intersection with no traffic control (unordered intersection). This illustrates visual search for different right-of-way logic.
- A four-lane highway with no intersections. This scenario removes the complication of intersections, but includes car following and passing actions.
- An intersection of four-lane roads controlled by traffic lights. This scenario includes many signs and markings and more cars than the other scenarios. It also illustrates an intersection with completely different traffic control.
- A set of closely spaced intersections of two-lane roads. This scenario includes Stop and Yield signs for the robot, and requires consideration of two downstream intersections at once. There is also an intersection on an approach road to complicate the search for conflicting cars.

4.4. Summary

The Ulysses-1 driving system introduces a language for controlling perception. This language is a collection of perceptual routines, each of which performs a specific, limited perceptual action. Routines do not search the entire scene for their target objects, but instead search in specific places relative to other objects. The routines are task-specific, because the reasoning component depends on the semantics of this relative search to get the right objects in the scene. The reasoning component of the system can thus avoid doing the spatial reasoning by pushing it off on the perception component; the routines do the spatial reasoning implicitly in their search of the physical world.

The perceptual routines significantly reduce the cost of perception. Experiments in simulated driving situations indicate that routines require about 10^{17} operations per second, as compared with about 10^{20} operations per second necessary for general, bottom-up perception. Thus even before Ulysses reasons about what the most critical objects are, perceptual cost can be reduced by three orders of magnitude by using perceptual routines.

5. Ulysses-2: ignoring redundant constraints

The previous section described how perceptual routines can sharply limit where the perception system has to look for specific objects in the scene. However, Ulysses-1 still asks the perception system to look for everything that could possibly generate a constraint or preference. In the second implementation of the driving system, Ulysses-2, the reasoning component requests only enough objects to determine a unique action for the robot. Ulysses-2 takes advantage of the specificity of perceptual routines to find one object at a time. Perceptual routines are called sequentially in an efficient order.

The main idea of Ulysses-2 can be illustrated with an example. Consider the robot driver approaching the intersection in Fig. 7. The robot has not looked for objects in the regions indicated with question marks—cars in the lane in front of it and on the side streets, and speed limit signs along the side of the road. Each of these could generate constraints on the robot's acceleration. In the worst case, a "Speed Limit 15" sign would force the robot to drive at 15 mph starting at the beginning of the unknown region. For cars, the worst case would be a car stopped in the road in front of the robot, or committed to enter the intersection from the side road. A car stopped in front of the robot at the beginning of the unknown area would impose the hardest constraint on the robot (i.e. force it to decelerate the most). Intuitively, the robot should look there first.

Continuing with the example, suppose that the robot looked and found a car in its lane stopped at the intersection. This is far enough away that the robot might be forced to look for a Speed Limit sign anyway; slowing to 15 mph nearby might require more deceleration than slowing to a stop near the intersection. However, since the robot has to stop before the intersection anyway, it does not yet have to look for cars on the cross streets. This section explains how Ulysses-2 implements the above reasoning process.

5.1. The corridor

Ulysses-1 generates constraints from objects around a *corridor*—the intended path down roads and through intersections. In Ulysses-2 the concept of a corridor is made more explicit in a network of frames [54]. Each section of road and intersection is a frame with slots for signs, markings, cars, signals, adjacent road

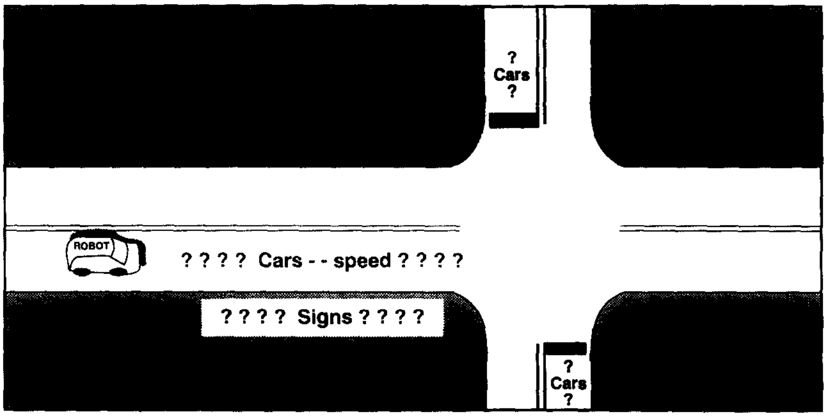

Fig. 7. This robot approaching the intersection has not yet looked for cars or signs in the regions marked by "?".

sections and intersections, etc. Many slots point to lists to allow variable numbers of these objects. The slots correspond to traffic objects that generate constraints and preferences; they are connected to the inputs of the inference trees described below.

At the beginning of every decision cycle, Ulysses-2 starts with frames with empty slots. The perceptual selection process requests that various slots be filled in order to determine the effects of the corresponding constraints. When such a request is made, a demon in the slot requests an appropriate perceptual routine to look for an object. The results of the routine update slots and sometimes cause the corridor frame network to be expanded as new road sections are discovered.

5.2. *The inference tree*

Ulysses-2 represents driving knowledge explicitly so that it can reason about how sensory data affects the choice of actions. The knowledge is encoded in two inference trees that transform inputs (sensed data) to outputs (acceleration and lane choice). This transformation is a complex function that comprises many component functions—hence, a tree of connected function *nodes*. Ulysses' inference trees include nodes that implement arithmetic operations, logical relations, set functions, and production rules. In general such functions would allow any action selection mechanism to be implemented.

Nodes can represent uncertainty in their output values. For continuous functions, uncertainty is represented as an interval of values; for nodes with discrete (symbolic) functions, uncertainty is represented with a set of symbols. In order to help keep the choice of perceptual actions unambiguous, and to aid in the efficient representation of the sets, we used ordered sets of symbols in the nodes. For example, it is possible to order traffic control indications as follows: Green, Yield, Stop, Red. We have defined appropriate uncertain-value propagation mechanisms for all of the continuous and discrete functions.

The nodes at the input of the trees use default uncertainty bounds for their input values when the attached corridor frame slots are empty. The default bounds are worst case, so that the sensed values will fall between the bounds. These defaults are domain-dependent.

Each node has an update demon attached to it which executes when the node value is updated. These demons can add new nodes to the tree to accommodate new parts of the corridor structure that are added when more of the environment is sensed.

The inference trees can be viewed as one-way constraint propagation networks, with bounds on the inputs being propagated to the output through the node functions. The inputs are sensory data, so the inference trees in effect translate a range of possible world states into a range of robot actions. The goal of Ulysses-2 is to find the minimum set of objects to sense that will reduce the uncertainty in the output to one unique action.

5.3. The search algorithm

Ulysses-2 begins each decision cycle with the worst-case, default input values propagated through the tree to the output. The output value thus presents a wide range of possible actions. As long as the output does not specify a unique action, Ulysses-2 repeatedly searches the tree to find another perceptual routine to run to get more information. The algorithm is as follows:

```
procedure ChooseAction (tree) {
  while tree output is not a single value {
        next-routine⇐Evaluate (tree);
        Execute (next-routine);
        PropagateNewInputValues (tree);
  }
}
```

where the next perceptual routine is chosen in a recursive tree evaluation:

```
function Evaluate (node) {
  if node is an input node
    return (demon for corridor slot connected to this node);
  else {
    child-node⇐ChooseCriticalInput(node);
    return (Evaluate(child-node));
  }
}
```

The key part of the recursive tree evaluation is the selection of the critical node input. Ulysses-2 looks for an input that dominates all of the others in determining the value of the node. For example, of the inputs to a node that computes the Maximum function, the input with the highest upper bound alone determines the upper bound of the Max node. Since this input must be evaluated before the value of the Max node can be completely determined, that input is selected for evaluation first. Not all node functions have such a dominating input; for these, Ulysses-2 selects an input according to a default ordering (which is chosen to give good performance for this task). Ulysses-2 can nevertheless find good sequences of perceptual actions because the nodes at the top of its trees do have dominating inputs. This fact is due to the encoding of knowledge Ulysses uses, with ordered preferences used to select actions.

5.4. *Experimental results*

The Ulysses-2 system was run on the same scenarios used to test Ulysses-1. The intent was for Ulysses-2 to make the same decisions (for acceleration and lane choice) as Ulysses-1. In all of the scenarios, Ulysses-2 indeed produced the exact same acceleration and lane commands at exactly the same time as Ulysses-1. And as the following results show, Ulysses-2 makes fewer perceptual requests.

Fig. 8 provides a comparison between the perceptual costs in Ulysses-1 and in Ulysses-2 in the left-side road scenario shown in Fig. 5. Initially Ulysses-2 is cheaper because it does not needlessly examine the adjacent lane for traffic, signs, etc. The information Ulysses-2 collects early in the decision cycle selects the current lane as the preferred one, and the adjacent lane is ignored. When the intersection is detected, the cost decreases steadily because the robot is getting closer to the intersection and looking at less and less road in front of it. Ulysses-2 does not yet look at or beyond the intersection because it would not make any difference in the robot's actions. At around $t = 47$, the approaching car enters the intersection. Since the intersection is searched before the approach roads, finding a car here allows Ulysses-2 to ignore the approach roads; hence the cost is much lower than for Ulysses-1. At $t = 50$ Ulysses has determined that the approaches are clear and enters the intersection; perceptual cost drops dramatically because the approach roads are no longer being scanned.

5.5. *Related work*

Ulysses-2 makes use of bounds propagation and tree search to select sensing functions. While both of these mechanisms have roots in previous work, Ulysses-2 extends them and combines them in a novel way. The propagation of numerical intervals through functions has been explored extensively in "constraint propagation" systems [23]. For example, as part of a system for arithmetic reasoning, Simmons [62] implemented the same arithmetic interval propagation functions

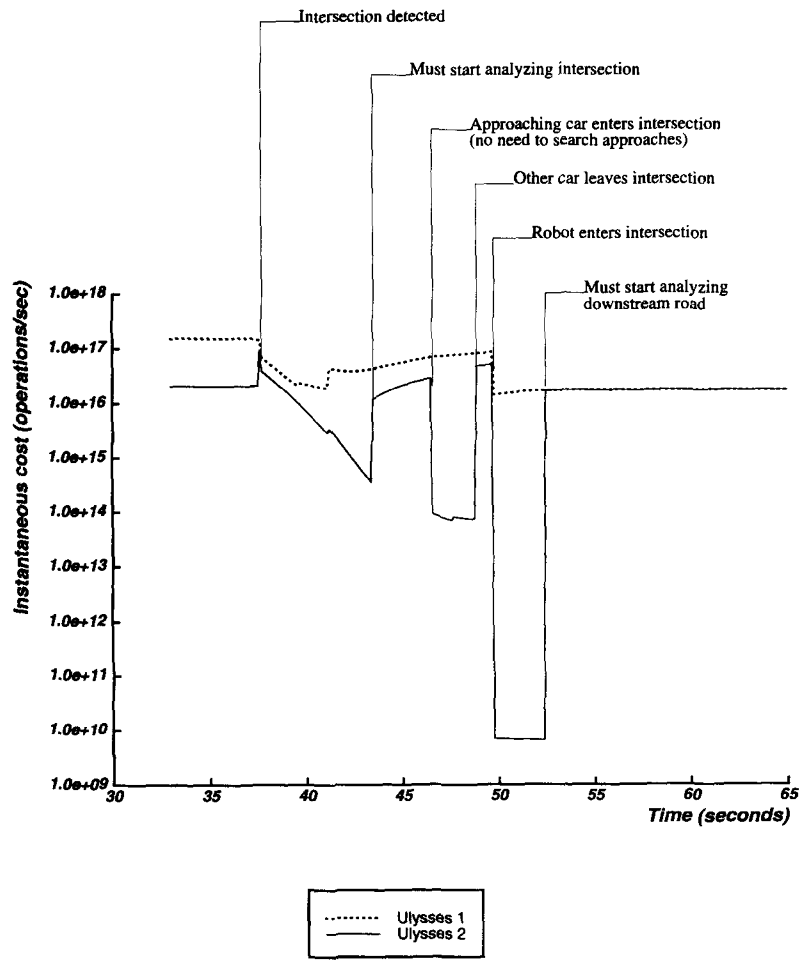

Fig. 8. Total perceptual cost of Ulysses-2 compared to Ulysses-1 for left-side road scenario.

used in Ulysses-2. He demonstrated that unusual numerical functions could be handled with the same basic ideas. Ulysses-2 also goes beyond the basic arithmetic functions, and furthermore includes Boolean operators, predicate functions, conditional expressions, etc.

The tree search algorithm can be thought of as an extension of branch-and-bound (BB) search [46]. BB represents the uncertainty of unexplored branches with intervals and looks for dominating inputs. While BB trees have only Max (or only Min) nodes in them, B* [9] can search a game tree with both Min and Max nodes. Ulysses-2 used Min and Max nodes, and extends the concept to other arithmetic and symbolic functions as well.

5.6. Summary

Ulysses-2 implements the Ulysses driving model with explicit data structures—a network of frames to model the world and an inference tree to encode the driving knowledge. Ulysses-2 selects perceptual actions by repeatedly evaluating the tree to determine the next most critical fact to sense. The most critical input is often uniquely determined by the extreme inputs to functions such as Max and Min. This evaluation is very effective because at the top of the inference trees, the driving model always includes some prioritizing knowledge—expressed in, e.g., Min and Max functions—to rank possible actions. Thus tree branches can be pruned near the top, thereby eliminating a lot of potential sensing. We conjecture that many task descriptions share this characteristic and would thus benefit similarly from this search technique.

Fig. 8 shows the effectiveness of Ulysses-2 in the left-side road scenario. In situations in which there is little choice of action, and few constraints on the robot (such as the two-lane highway beyond the intersection), Ulysses-2 shows no improvement over Ulysses-1. When there is a choice of lanes, Ulysses-2 offers about an order of magnitude of improvement because it prunes away the unnecessary search of the adjacent lane. Ulysses-2 makes its biggest impact when there are several factors that could significantly constrain the robot's action. In all scenarios that contained an intersection, Ulysses-2 was at times able to find a constraining condition quickly and stop sensing. This resulted in a cost savings of from one to *six orders of magnitude* over Ulysses-1.

6. Ulysses-3: modeling world dynamics

The first two versions of the driving program reduce the perceptual load on the robot when it looks at the world each decision cycle. However, both Ulysses-1 and Ulysses-2 throw away all information between cycles. The third implementation of the driving program, Ulysses-3, takes advantage of coherence in the world over time to further reduce the information it needs to sense. Knowledge about how each object can change is stored in the inference tee with the input node for that object, and Ulysses-3 automatically determines when the robot needs to sense that object again.

6.1. Algorithm modifications

Ulysses-3 uses almost the same data structures and search algorithms as Ulysses-2. However, the slots in the corridor frames do not start the decision cycle empty. Instead, they retain their values from the previous cycle. A time stamp is added to indicate their ages. The nodes at the inputs to the inference tree do not simply use default values (or uncertainty intervals) at the beginning of the cycle; now, an initialization program in the input node examines the associated slot value and its age and computes new uncertainty bounds. As in Ulysses-2, these bounds represent the worst case.

The initialization programs encode knowledge about domain dynamics. Some slots' bounds can be fixed with high confidence over time, while others may change predictably or unpredictably between cycles. For example, the *type* of a sign does not change over time. The *range* to the sign decreases predictably according to the motion of the robot. The range to the closest car ahead of the robot is unpredictable; it is presumably possible for a car to change lanes and "cut off" the robot at any time. The driving model cannot yet predict when such a lane change could occur, so Ulysses-3 must find the lead car from scratch every decision cycle.

6.2. Experimental results

Ulysses-3 was used to drive the simulated robot through the same scenarios that were used for Ulysses-1 and Ulysses-2. Ulysses-3 produced the exact same acceleration and lane commands as the earlier versions. Fig. 9 shows the perceptual cost during the left-side road scenario and compares it with the costs using Ulysses-2. In general, Ulysses-3 reduces perceptual cost significantly.

Before the intersection is detected at $t = 37.5$, the Ulysses-3 search cost is usually between 10^{11} and 10^{12}. This is much less than for Ulysses-2 (10^{16} or so) because Ulysses-3 is looking only for new pieces of road that it could not see in the last decision cycle (i.e., a piece of road at the sensor range limits). The spikes in cost occur when Ulysses-3 makes a search along the road ahead for the next sign. If this next sign is far ahead, then Ulysses-3 is able to wait some time before extending its sign horizon again.

When the intersection is detected there is one decision cycle of high cost as Ulysses-3 considers changing lanes. Cost drops during the lane change because Ulysses-3 temporarily stops checking for an intersection in the distance. There is another spike at about $t = 41.5$ when the robot finishes the lane change as Ulysses-3 makes sure there is no lane farther to the left. Otherwise, Ulysses-3 does not have to look for lanes or signs or signals again until the intersection becomes important at $t = 43$. At this point there is a sharp increase in cost because Ulysses-3 looks for many objects while it analyzes the intersection. However, Ulysses-3 looks only *once* for signs and signals at the intersection. After the spike, the cost drops again for several seconds. This indicates that although Ulysses-3 has found the approaching car, it does not keep looking at the car because it knows the car cannot yet be at the intersection. Ulysses-3 does not look for the car again until about $t = 46$.

When the car finally clears the intersection, Ulysses-3 makes one last check for approaching traffic before deciding to go through the intersection. This check is reflected in the spike in cost at $t = 48.5$. When the robot enters the downstream lane, search costs have a similar character to those on the approach road.

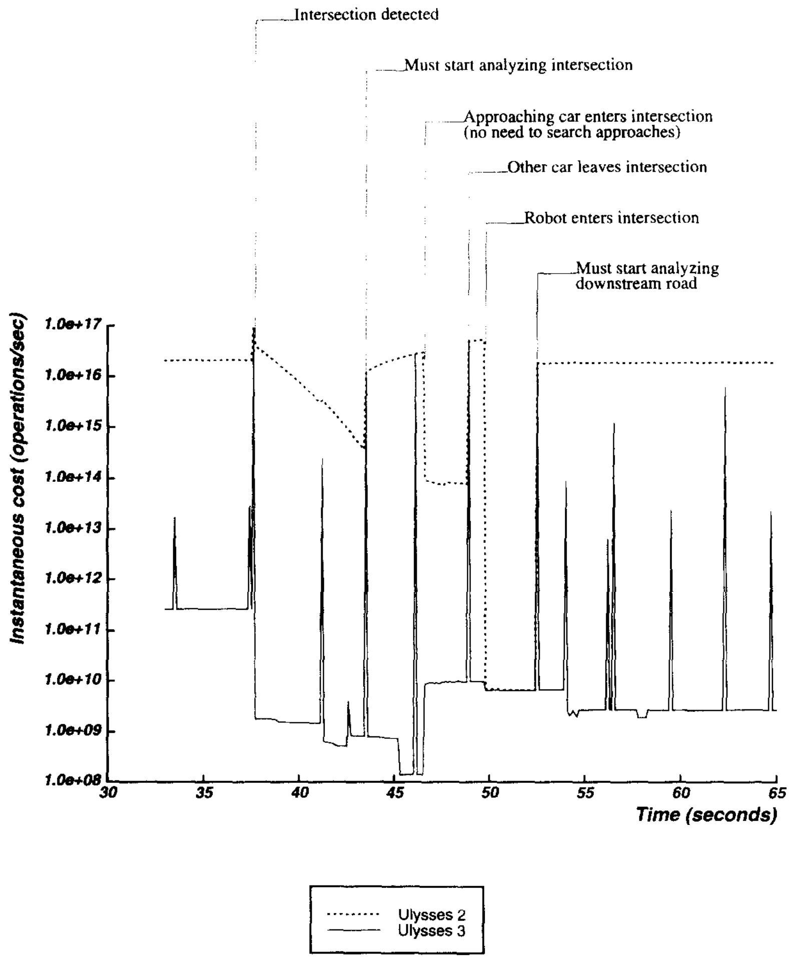

Fig. 9. Total perceptual cost of Ulysses-3 compared to Ulysses-2 for the left-side road scenario.

However, the cost on the two-lane road is cheaper than on the four-lane road because the robot does not have to consider changing lanes and therefore does not continually search for downstream intersections. The spikes occur when Ulysses-3 extends its road or sign horizon in jumps.

The impact of explicitly modeling world dynamics is quite significant. The experimental results show that most of the time, the cost for Ulysses-3 is between 5 and 7 orders of magnitude less than for Ulysses-2. This savings is primarily the result of remembering static objects—roads and signs. Ulysses-3 also performs much better than Ulysses-2 when constraints come from somewhat predictable objects such as cross traffic. Typically, Ulysses-3 has cost spikes at intervals of several seconds that reach Ulysses-2 levels; these spikes occur when the program must analyze a new situation or when the uncertainty about a critical object has grown too large. The next section discusses possible ways to reduce the magnitude of these spikes.

7. Discussion

7.1. Net effect of selective perception

Fig. 10 illustrates the cost of perception in the left-side road scenario for naive, general perception and for all of the Ulysses implementations. On the right side of the graph, the average cost rate over the whole scenario (i.e., about 32 seconds) is given for the three implementations. The minimum cost, incurred just to identify the road in front of the robot, is also indicated in the figure. Most of the time, Ulysses-3 requires about seven orders of magnitude less computation than Ulysses-1, and about 10 orders of magnitude less than an unguided general perception system. When the cost of the "spikes" of Ulysses-3 is averaged over the whole scenario, Ulysses-3 is still two orders of magnitude cheaper than Ulysses-1, and over five orders of magnitude cheaper than general perception.

The approximate capabilities of some computers is also indicated in the figure. We have placed the computers a little low in the plot because the perceptual cost estimates did not really address data transfer costs, which would occupy some computer power. The "fast" computer is a hypothetical one delivering 10^9 operations per second. The approximation for the Navlab with a Warp systolic array processor was taken from Clune et al. [21]. Although the "fast" computer is almost fast enough to keep up with the requirements of Ulysses-3 most of the time, when the spikes are averaged in the cost is still too high by over five orders of magnitude. This discrepancy suggests that additional techniques should be found to reduce the cost of the spikes.

7.2. Reducing peak costs

A comparison of the instantaneous and average costs for Ulysses-3 in Fig. 10 indicates that most of the cost is in the spikes. Furthermore, the peaks of the spikes in Ulysses-3 reach the cost levels of Ulysses-2. Thus if the robot were required to have the capacity to handle the maximum instantaneous load, it would need the same resources as the Ulysses-2 system; nothing would be gained in Ulysses-3. However, it would not be difficult, using the inference trees, to amortize the spikes over time and limit peak cost. For example, suppose that the available computation in each decision cycle were limited. The limit might represent processing limits or mechanical limits—pan rate, zoom rate, etc. The action selection process would stop when it hit the limit, even if not complete. Since the sense nodes are always initialized to worst-case values, and sensing only

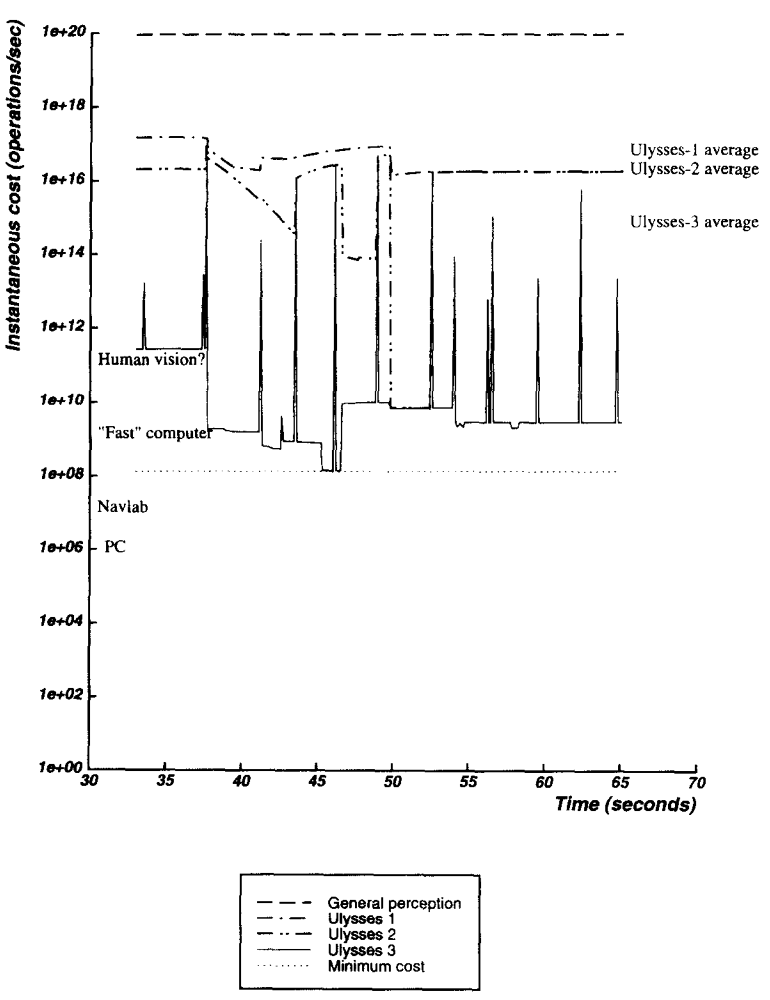

Fig. 10. Perceptual cost for all implementations of Ulysses in the left-side road scenario. Dotted line indicates the estimated cost for finding the lane in front of the robot. This operation is not under control of Ulysses-2 or -3; thus it represents the minimum cost these implementations can achieve.

improves the performance of the task, stopping before completion leaves pessimistic (safe) constraints in the tree. Ulysses-3$^+$ would select the available action with the lowest currently possible preference priority (i.e., the current default preference). For acceleration this would be the highest upper value allowed by the pessimistic constraints; for lane changing the result would often, but not always, be to stay in the same lane. The effect would be to slow the robot down until it could see all that it needed to in order to proceed safely.

Ulysses-3, and the extension just described, performs "lazy perception" to save computation—i.e., it only looks for things when absolutely necessary. Another way to reduce peak costs would be to use excess capacity to look for things before it is strictly necessary. Advance estimates could avoid the need to slow the robot while the perception system is catching up. Such predictive perception could be guided with the inference tree by projecting the value of uncertain sense nodes into the future and finding out which ones would impact the value of the top node.

Peak costs, and in some cases continuous costs, could also be reduced by using a tracking perceptual routine that simply watches one object continuously (between decision cycles). Tacking objects would be relatively inexpensive, since it involves no reasoning about spatial relations between traffic objects, no sequential scanning, and no complex model matching. By tracking a car approaching an intersection, for example, Ulysses-3^+ would not have to search for the car again when pessimistic position estimates placed it near the intersection. Instead, Ulysses-3^+ could wait until the vehicle had actually reached the intersection and then look for more cars. Similarly, at signalized intersections Ulysses-3^+ would not have to find and interpret signals every decision cycle; a tracking routine could easily fix itself on the appropriate signal head.

7.3. Balancing correctness, cost, and performance

The height of the sensing peaks in Fig. 10 could also be reduced by reducing the "correctness" instead of performance. Correct driving, according to Ulysses, means always considering all constraints in the model. Ulysses is intended to generate safe actions for the robot when all constraints are applied. If, because of resource limits, *occlusion*, or other *sensor limitations*, Ulysses-3 could not determine some fact, Ulysses-3 would slow the robot—degrade its performance—rather than ignore a constraint. For example, if there was limited sight distance at an intersection and the robot could not see up a side road, Ulysses-3 would always assume that there could be a car on the side road and prepare the robot to stop. This conservatism could get extreme; if Ulysses included the notion of children spontaneously running into the street, Ulysses-3 would force the robot to creep by all parked cars just in case a child was hiding behind one.

It is interesting to note that humans often do not reduce their performance in similar conditions. Humans clearly do not look at everything of importance in the driving environment, because they have collisions with objects that they could have avoided if they had anticipated them. There are two observations we can make about human visual search and Ulysses. For example, humans have a limited field of view. The location of "human vision" in Fig. 10 (estimated from [55]) also suggests that humans do much less processing than the Ulysses model requires. Human peripheral vision helps to cover some situations by flagging moving and colorful objects for attention, but it is doubtful that it can cover all of the Ulysses constraints thoroughly. Second, humans make assumptions about the likelihood of various traffic situations, and selectively ignore possibilities if they seem unlikely. Thus humans can choose to ignore the possibility of children hiding behind parked cars, ignore the possibility of cars on minor side roads, assume that the car they are tailgating won't stop suddenly, overdrive their headlights, etc. Of course, human expectations are occasionally violated and they have accidents.

If a driving robot is ever to achieve human-level performance—i.e., drive at similar speeds in similar situations—it will also have to accept the risks that humans accept. This risk tradeoff could possibly be incorporated into a driving program using decision theory. Robot speed (or lack thereof) and accidents would contribute to cost, and domain knowledge or learned experience would provide probability estimates for observable conditions. In each decision cycle, the driving program would have to incrementally select perceptual actions to provide the most information given the time available, and then select the motor actions that were expected to be the most cost-effective. Unfortunately, there are several difficulties with a decision-theoretic approach. First, with many constraints and many sensing options, the complexity of finding the best next action could be overwhelming. Second, probability distributions for various events and conditions could be very difficult to estimate. And finally, it is not clear what cost to assign an accident, especially one that damages property other than the robot. The driving system described in this paper effectively assigned infinite cost to accidents. This last question has philosophical ramifications that go beyond the technical problems.

8. Conclusions

This paper addresses the problem of efficiently controlling perception for a complex task. We have developed a driving control system that defines useful perceptual operations and a mechanism for integrating perception control with situation assessment and action selection. The system was implemented in three stages. Ulysses-1 defined the perceptual operations and provided the basic connection between reasoning and perception that allows reasoning to request sensing actions. These operations, called perceptual routines, allow the reasoning component to request specific types of objects that are semantically important for driving decisions. At the same time they guide the perception component in its physical search for these objects. Ulysses-2 reasons about worst-case bounds on unknown conditions in the world. Ulysses-2 looks for objects in the order of their potential impact on robot actions, and stops sensing when further information cannot affect the choice of action. There are three main components of the perceptual reasoning mechanism: a frame-like data structure for encoding structure in the environment; a constraint-propagating inference tree to encode the action selection knowledge; and a search algorithm for efficiently selecting critical tree leaves to evaluate (objects to sense). Finally, Ulysses-3 takes advantage of coherence in the domain over time. Information is retained between decision cycles, and domain dynamics are encoded in update routines which increase the uncertainty in facts as they get older.

The effect of these perceptual control mechanisms is dramatic. The three versions of Ulysses were used to drive a simulated robot through several different situations. Ulysses-1 immediately dropped the estimated cost of perception by three to four orders of magnitude compared with bottom-up perception. The impact of Ulysses-2 varied more with the moment; in situations near intersections where there was often an obvious constraint on robot actions, the estimated perceptual cost dropped an additional three to six orders of magnitude. Ulysses-3 had an even bigger impact. Most of the time the estimated cost for Ulysses-3 was five to seven orders of magnitude below that for Ulysses-2, with momentary jumps every couple of seconds as the system looked around to reduce the growing uncertainty. On average, Ulysses-3 required about five orders of magnitude fewer arithmetic operations per second than did naive, bottom-up perception.

Our eventual goal is to drive a real robot. Further reductions in perceptual cost will be necessary to achieve this goal. As discussed in the previous section, the mechanism of Ulysses can be extended to limit the peak costs of perception and significantly reduce the average cost. Additional perceptual operations, such as tracking, have the potential to reduce cost even further. The results of using these perceptual attention control mechanisms give us confidence that the perception problem for driving is tractable and perhaps within the capability of computing systems of the near future.

References

[1] AASHTO, A policy on geometric design of highways and streets, American Association of State Highway and Transportation Officials, Washington, DC (1984).

[2] J. Aasman, Implementations of car-driver behaviour and psychological risk models, in: J.A. Rothengatter and R.A. de Bruin, eds., *Road User Behaviour: Theory and Practice* (Van Gorcum, Assen, 1988) 106–118.

[3] P.E. Agre and D. Chapman, Pengi: an implementation of a theory of activity, in: *Proceedings AAAI-87*, Seattle, WA (1987) 268–272.

[4] H. Akatsuka and I. Shinichiro, Road signposts recognition system, in: *Proceedings SAE International Congress* (SAE, Detroit, MI, 1987).

[5] J. Aloimonos, I. Weiss and A. Bandyopadhyay, Active vision, *Int. J. Comput. Vision* **1** (4) (1988) 333–356.

[6] R. Arkin, Motor schema based navigation for a mobile robot, in: *IEEE International Conference on Robotics and Automation*, Raleigh, NC (1987).

[7] D.H. Ballard, Animate vision, *Artif. Intell.* **48** (1991) 57–86.

[8] W. Becket and N. Badler, Integrated behavioral agent architecture, in: *Proceedings Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL (1993).

[9] H. Berliner, The B* tree search algorithm: a best-first proof procedure, *Artif. Intell.* **12** (1979) 23–40.

[10] P.J. Besl and R.C. Jain, Three-dimensional object recognition, *ACM Comput. Surv.* **17** (1) (1985).

[11] T.O. Binford, Survey of model-based image analysis systems, *Int. J. Rob. Res.* **1** (1) (1982).

[12] J. Blythe and T.M. Mitchell, On becoming reactive, in: *Proceedings 6th International Workshop on Machine Learning*, Ithaca, NY (1989).

[13] R. Bolles, Verification vision for programmable assembly, in: *Proceedings IJCAI-77*, Cambridge, MA (1977).

[14] R.A. Brooks, Symbolic error analysis and robot planning, AI Memo 685, MIT, Cambridge, MA (1982).
[15] P. Burt, 'Smart sensing' in machine vision, in: H. Freeman, ed., *Machine Vision: Algorithms, Architectures, and Systems*, Perspectives in Computing **20** (Academic Press, San Diego, CA, 1988) 1–30.
[16] M. Campani et al., Visual routines for outdoor navigation, in: *Proceedings Intelligent Vehicles 93 Symposium* (1993).
[17] J. Carbonell, C. Knoblock and S. Minton, Prodigy: an integrated architecture for planning and learning, in: K. VanLehn, ed., *Architectures for Intelligence* (Lawrence Erlbaum, Hillsdale, NJ, 1991) Chapter 9.
[18] D. Chapman, *Vision, Instruction, and Action* (MIT Press, Cambridge, MA, 1991).
[19] H.C. Chin, SIMRO: a model to simulate traffic at roundabouts, *Traffic Eng. Control* **26** (3) (1985) 109–113.
[20] L. Chrisman and R. Simmons, Sensible planning: focusing perceptual attention, in: *Proceedings AAAI-91*, Anaheim, CA (1991).
[21] E. Clune, J. Crisman, G. Klinker and J. Webb, Implementation and performance of a complex vision system on a systolic array machine, *Future Gen. Comput. Syst.* **4** (1988) 15–29.
[22] J. Crisman and C. Thorpe, Color vision for road following, in: *Proceedings of SPIE Conference on Mobile Robots* (1988).
[23] E. Davis, Constraint propagation with intervals, *Artif. Intell.* **32** (1987) 281–331.
[24] T. Dean and M. Boddy, An analysis of time-dependent planning, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 49–54.
[25] R. Doyle, D. Atkinson and R. Doshi, Generating perception requests and expectations to verify the execution of plans, in: *Proceedings AAAI-86*, Philadelphia, PA (1986).
[26] G.J. Ettinger, Large hierarchical object recognition using libraries of parameterized sub-parts, in: *Proceedings Conference on Computer Vision and Pattern Recognition*, Ann Arbor, MI (1988) 32–41.
[27] FHWA, *Manual on Uniform Traffic Control Devices* (Federal Highway Administration, U.S. Department of Transportation, Washington, DC, 1978).
[28] FHWA, *Traffic Network Analysis with NETSIM: A User Guide* (Federal Highway Administration, Washington, DC, 1980).
[29] R.J. Firby, Task directed sensing, in: *Sensor Fusion II: Human and Machine Strategies* (SPIE, 1989) 480–489.
[30] T. Fujimori and T. Kanade, Knowledge-based interpretation of outdoor road scenes, in: C. Thorpe and T. Kanade, eds., *1987 Year End Report for Road Following at Carnegie Mellon*, CMU-RI-TR-88-4, Carnegie Mellon University, Pittsburgh, PA (1988) 45–96.
[31] T. Garvey, Perceptual strategies for purposive vision, Technical Note 117, SRI International, Menlo Park, CA (1976).
[32] E. Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, in: *Proceedings AAAI-92*, San Jose, CA (1992).
[33] D.R.P. Gibson, Available computer models for traffic operations analysis, in: *The Application of Traffic Simulation Models*, TRB Special Report 194, National Academy of Sciences (1981) 12–22.
[34] V. Graefe, Vision for intelligent road vehicles, in: *Proceedings of Intelligent Vehicles 93 Symposium* (1993).
[35] W.E.L. Grimson, The combinatorics of object recognition in cluttered environments using constrained search, *Artif. Intell.* **44** (1990) 121–165.
[36] N.C. Griswold and B. Bergenback, Stop sign recognition for autonomous land vehicles *(ALVs)* using morphological filters and log-conformal mapping, Texas A & M University, College Station, TX (1989).
[37] A. Hanson and E. Riseman, Segmentation of natural scenes, in: A. Hanson and E. Riseman, eds., *Computer Vision Systems* (Academic Press, New York, 1978) 129–163.
[38] B. Hayes-Roth, Making intelligent systems adaptive, in: K. Van Lehn, ed., *Architectures for Intelligence* (Lawrence Erlbaum, Hillsdale, NJ, 1991).

[39] M. Hebert and T. Kanade, 3-D vision for outdoor navigation by an autonomous vehicle, in: C. Thorpe and T. Kanade, ed., *1987 Year End Report for Road Following at Carnegie Mellon*, CMU-RI-TR-88-4, Carnegie Mellon University, Pittsburgh, PA (1988) 29–41.

[40] E.J. Horvitz, Reasoning about beliefs and actions under computational resource constraints, in: L.N. Kanal, T.S. Levitt and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* (Elsevier Science, Amsterdam, 1989) 301–324.

[41] K. Ikeuchi and M. Hebert, Task oriented vision, in: *Proceedings DARPA 1990 Image Understanding Workshop* (1990) 497–507.

[42] L.P. Kaelbling, Gals as parallel program specifications, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 60–65.

[43] K. Kluge and H. Kuga, Car recognition for the CMU Navlab, in: C. Thorpe and T. Kanade, eds., *1987 Year End Report for Road Following at Carnegie Mellon*, CMU-RI-TR-88-4, Carnegie Mellon University, Pittsburgh, PA (1988) 88–115.

[44] K. Kluge and C. Thorpe, Explicit models for road following, in: *Proceedings IEEE Conference on Robotics and Automation*, Scottsdale, AZ (1989).

[45] J.E. Laird, A. Newell and P.S. Rosenbloom, Soar: an architecture for general intelligence, *Artif. Intell.* **33** (1987) 1–64.

[46] E. Lawler and D. Wood, Branch-and-bounds methods: a survey, *Oper. Res.* **14** (1966) 699–719.

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

[48] J. Malec and M. Morin, A pre-intelligent driver information unit, in: *Proceedings Intelligent Vehicles 93 Symposium* (1993).

[49] D.M. McKeown Jr and J.L. Denlinger, Cooperative methods for road tracking in aerial imagery, in: *Proceedings 1988 DARPA IUS Workshop* (1988) 327–341.

[50] J. McKnight and B. Adams, Driver education and task analysis, Volume I: task descriptions, Final Report, Department of Transportation, National Highway Safety Bureau, Washington, DC (1970).

[51] B. Mertsching et al., Interpretation of traffic scenes using a hierarchical data structure, in: *Proceedings Intelligent Vehicles 93 Symposium* (1993).

[52] E. Mettala, The OSD tactical unmanned ground vehicle program, in: *Proceedings 1992 DARPA Image Understanding Workshop* (1992).

[53] J.A. Michon, A critical view of driver behavior models: what do we know, what should we do? in: L. Evans and R. Schwing, eds., *Human Behavior and Traffic Safety* (Plenum, New York, 1985).

[54] M. Minsky, A framework for representing knowledge, in: P. Winston, ed., *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975).

[55] H.P. Moravec, Locomotion, vision and intelligence, in: M. Brady and R. Paul, eds., *Robotics Research* (MIT Press, Cambridge, MA, 1984) 215–224.

[56] D.A. Reece, Selective perception for robot driving, Technical Report CMU-CS-92-139, Carnegie Mellon University, Pittsburgh, PA (1992).

[57] D.A. Reece and S.A. Shafer, An overview of the Pharos traffic simulator, in: J.A. Rothengatter and R.A. de Bruin, eds., *Road User Behaviour: Theory and Practice* (Van Gorcum, Assen, 1988).

[58] D.A. Reece and S.A. Shafer, A computational model of driving for autonomous vehicles, *Transportation Res. A* **27A** (1) (1993) 23–50.

[59] R. Rimey and C. Brown, Control of selective perception using Bayes nets and decision theory, *Int. J. Comput. Vision* (submitted).

[60] D. Russell, Constraint networks: modeling and inferring object locations by constraints, Technical Report 38, University of Rochester, Computer Science Department, Rochester, NY (1978).

[61] S.A. Shafer, A. Stenz and C.E. Thorpe, An architecture for sensor fusion in a mobile robot, Technical Report CMU-RI-TR-86-9, Carnegie Mellon University, Pittsburgh, PA (1986).

[62] R. Simmons, 'Commonsense' arithmetic reasoning, in: *Proceedings AAAI-86*, Philadelphia, PA (1986).

[63] U. Solder and V. Graefe, Object detection in real time, in: *Proceedings SPIE Symposium on Advances in Intelligent Systems* (1990) 121–165.
[64] C. Thorpe, M. Hebert, T. Kanade and S. Shafer, Toward autonomous driving: the CMU Navlab, Part I: perception, *IEEE Expert* (August 1991) 31–42.
[65] S. Ullman, Visual routines, *Cognition* **18** (1984) 97–160.
[66] H.H. van der Molen and A.M.T. Botticher, Risk models for traffic participants: a concerted effort for theoretical operationalizations, in: J.A. Rothengatter and R.A. de Bruin, *Road Users and Traffic Safety* (Van Gorcum, Assen/Maastricht, Netherlands, 1987) 61–82.
[67] L. Wixson, Exploiting world structure to efficiently search for objects, Technical Report 434, University of Rochester, Computer Science Department, Rochester, NY (1992).
[68] S.-Y. Wong, TRAF-NETSIM: how it works, what it does, *ITE J.* **60** (4) (1990) 22–27.
[69] X. Yu et al., Road tracking, lane segmentation and obstacle recognition by mathematical morphology, in: *Proceedings Intelligent Vehicles 93 Symposium* (1993).