Using Active Vision to Simplify Perception for Robot Driving

Douglas A. Reece and Steven Shafer

November 1991

CMU-CS-91-199

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

A shorter version of this report appears in the 1991 AAAI Fall Symposium on Sensory Aspects of Robotic Intelligence, and in the January 1992 DARPA Image Understanding Workshop.

Abstract

As mobile robots attempt more difficult tasks in more complex environments, they are faced with combinatorially harder perceptual problems. In fact, computation costs for perception can easily dominate the costs for planning in a mobile robot. Existing perception systems on mobile robots are potentially many orders of magnitude too slow for real-world domains. In this paper we show active vision at the system level can make perception more tractable. We describe how our planning system for a complex domain, tactical driving, makes specific perceptual requests to find objects of interest. The perception system then scans the scene using routines to search for these objects in limited areas. This selective vision is based on an understanding and analysis of the driving task. We illustrate the effectiveness of request-driven routines by comparing the computational cost of general scene analysis with that of selective vision in simulated driving situations.

This research was sponsored by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597 and Contract DACA76-85-C-0003.

Keywords: Vision and scene understanding, robotics, simulation and modeling, active vision, traffic engineering

1. Introduction

Today's mobile robots can drive down hallways and roads and across fields without getting stuck or colliding with obstacles. As these robots attempt more challenging tasks, they will need better perception and planning systems. In the past, planning and perception have been independent components of the robot reasoning system. Figure 1-1 shows that in traditional systems the planning component works on a symbolic world model, which it assumes the perception component keeps up to date. The perception component must therefore find and understand *everything* of potential interest to the planner. This arrangement cannot be used in robots in the real world because scenes are too complex and change too fast for unguided, exhaustive interpretation.

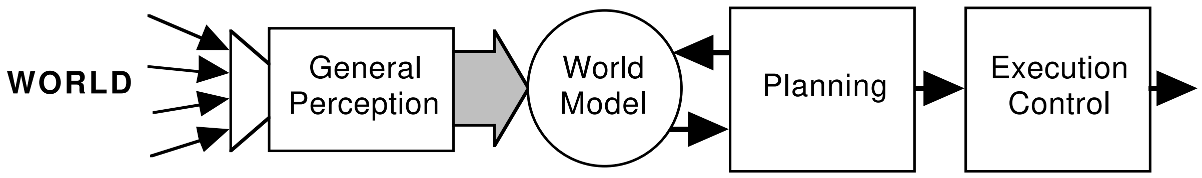


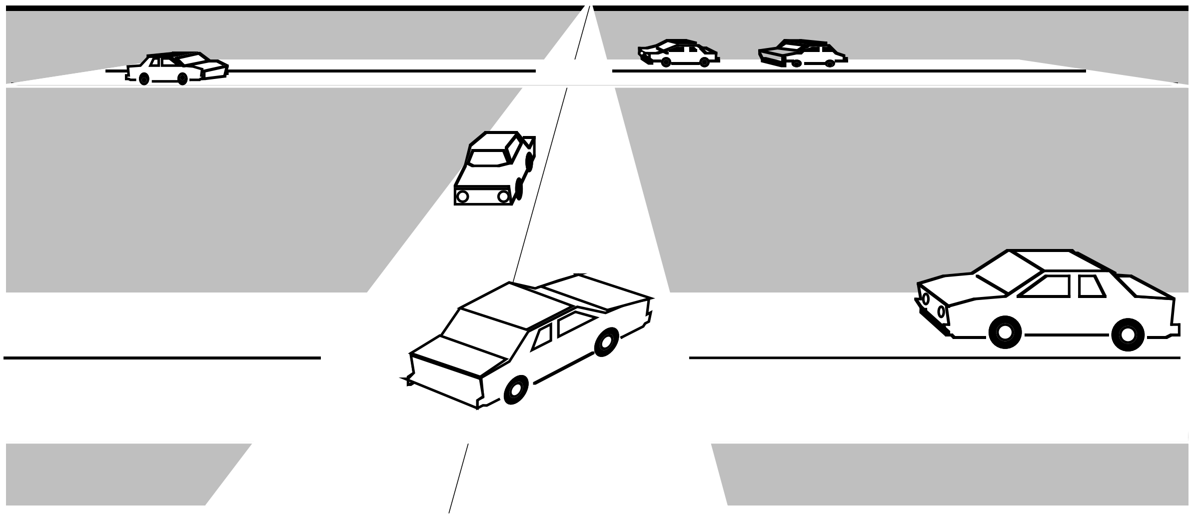
Figure 1-1: A traditional robot control system.

Consider the driving scene shown in Figure 1-2a. It contains several vehicles. The vehicles are different distances and directions from the observer, and so appear in different locations in the image. A general perception system attempting to find all vehicles in the scene would have to search all possible locations, ranges, and poses, as we sketch in Figure 1-2b. For each of these possibilities, the perception system would also have to consider variations in vehicle shape, color, and illumination. The driving scene changes from moment to moment, with vehicles moving, disappearing, reappearing, and becoming partly occluded. The combinatorics of perception in the real world are enormous. Figure 1-3a illustrates that even if traffic objects can be found in various poses throughout a scene, it is wasteful to look for them because they are not all important to the robot. We propose to reduce perceptual complexity by using system-level active vision—using the planner to limit for what a robot has to look, and where it has to look. Figure 1-4 shows the resulting architecture. A robot driver can then use its perception effectively, as shown in Figure 1-3b.

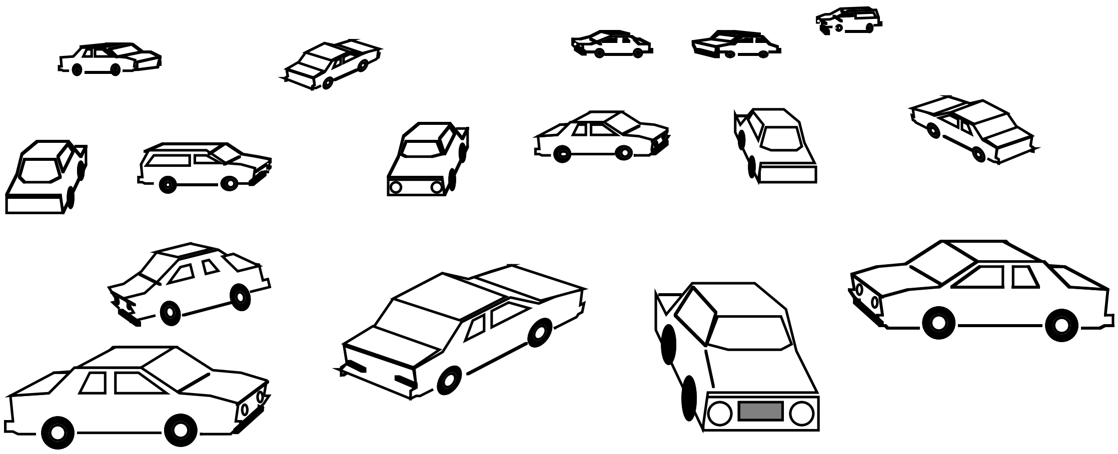
In the next section we explain what characteristics of real world scenes make them so difficult to interpret. We introduce the domain of driving in traffic, and describe how we define it using an environment simulator and computational model. We then estimate the computational cost of general perception in the driving domain. Such an estimate demonstrates the futility of a naive approach to perception for a real problem, and provides a basis for evaluating our active vision techniques. Section 3 describes two such techniques. The first requires the planner to *request* information about the world, so perception only has to look for currently relevant objects. The second technique is the use of *perceptual routines* that look (only) for specific objects in specific areas of the scene. [1]

---

[1]This paper emphasizes demand-driven perception and routines, but does not address other reductions in search made possible by modeling the world better. Our current work addresses those issues.



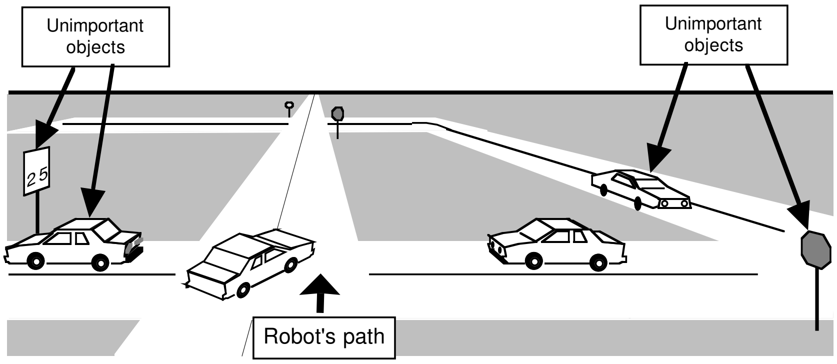
a. A driving scene with vehicles.



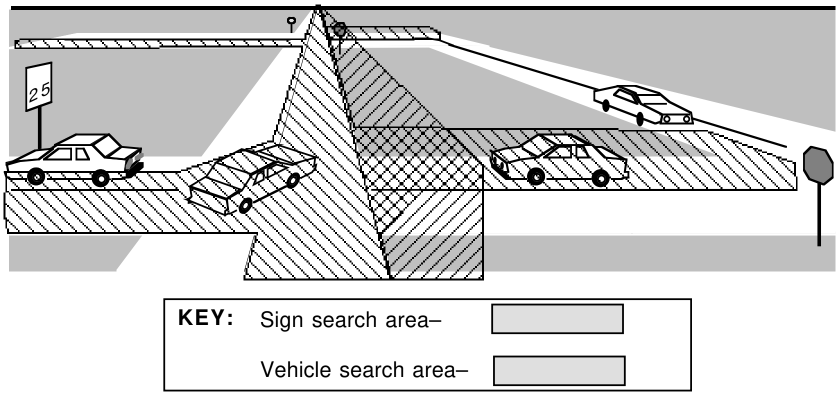
b. What a naive robot must look for.

Figure 1-2: The combinatorics of naive perception.

Section 4 illustrates how the selective perception techniques work for driving. We discuss how we have implemented our driving model in a computer program so that we can study perceptual issues in simulation. Next we demonstrate how perceptual routines are used in a driving situation. Finally, the section shows the results of simulations in several driving situations and compares the computational cost to the naive approach. We conclude in Section 5 that general perception is intractable and that selective vision is necessary for a task such as driving.



a. Robot’s view of traffic objects.



b. Desirable visual search constraints.

Figure 1-3: Not all traffic objects are relevant to the robot.

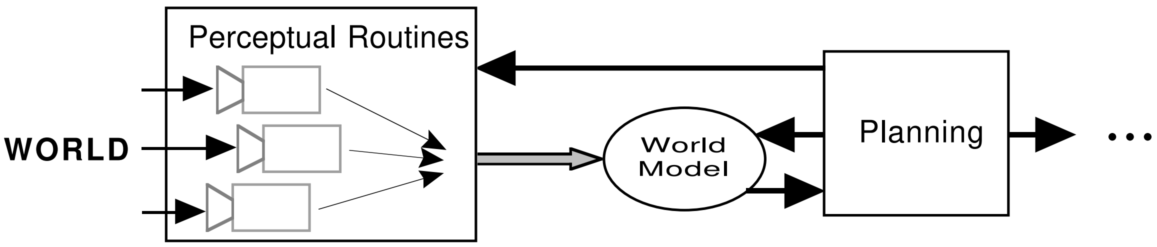


Figure 1-4: An active-vision robot control system.

2. Driving with General Perception

In this section we discuss the complexity of perception for driving using a naive, bottom-up approach. For this approach the perception system must analyze the entire scene around the robot and find all the traffic objects that the planner might need. First we explain why this approach is in general inadequate in a complex, dynamic domain. Next we describe the driving task and analyze the characteristics that affect perception. Finally, we use this analysis to estimate the computational cost of general perception for driving.

2.1. Why Perception is Hard in Complex, Dynamic Domains

Robots in some domains can use minimal vision systems. Some robots can make extensive use of an internal world model to predict and plan all their actions from an initial state. In fact, if the initial world state is pre-compiled, no perception is necessary at all. In other domains the environment is much more complex, but the task requires only simple perception. Monitoring scalar values, navigating with proximity sensors and tracking blobs with coarse imaging sensors are examples.

Other domains require perception of objects, but the objects and backgrounds are still simple. The environment doesn't change except for robot actions. Decades of computer vision research have shown that vision is very difficult even with these simplifying conditions. Recently several reports have estimated the complexity in such domains. Template matching is polynomially complex in the size of the template (model) and image [31]. Matching processed image features to model features is in general exponentially complex [15], but in some cases polynomial in the number of scene and model features [20]. Interpreting all features in a scene together requires searching the space of all possible interpretations, which is exponentially large:

$$\textit{Number of interpretations} = (\textit{Number of region classes})^{\textit{Number of regions}}$$

Still, in a static environment, it is not unreasonable to take a long time to look at the world.

Robot perception is even harder in complex, dynamic domains. This difficulty is due to several domain characteristics:

- The world state is not known in advance and cannot be predicted. Therefore the robot must observe the environment continuously.
- Objects play different roles. For example, they are not all just obstacles.
- The appearance of objects changes due to many factors, including location in the field of view, range, size, orientation, shape, color, marking, occlusion, illumination, reflections, dirt, haze, etc. The appearance can vary as much as the product of all these factors.
- The environment is cluttered and distracting. It contains many background features that can be confused with the features of important objects.
- Domain dynamics place time constraints on perceptual computations.

The robot must search through a lot of data—from a rich sense such as vision—to discern objects in such a confounding environment. This is why perception is so computationally expensive. If the computational cost is many orders of magnitude too high for the time and resources available, then we can say that perception is *effectively intractable.*

Various techniques have been used to reduce the computation time for object recognition. These include using easy-to-find object features; using distinguished features that quickly discriminate between objects; using geometric or other constraints between features; using coarse input data (followed by fine data in small areas); and computing features in parallel. Without these techniques, even relatively simple vision tasks would be impractical. To date, these approaches have been demonstrated only for simple tasks in complex environments (for example, obstacle avoidance and road following), or for complex tasks in controlled environments (for example, bin picking). We believe that a powerful perception system that works as quickly and effectively as the human system will use all of these methods. However, general perception is still inadequate in complex, dynamic domains.

2.2. The Perceptual Cost of Driving

Driving in traffic is an example of a complex task in a complex, dynamic environment. This paper addresses *tactical* driving [27], which is the selection of speed and steering maneuvers. Tactical driving requires features of both real-time servo-control systems and symbolic reasoning systems. Decisions must be made dynamically in response to changing traffic situations. Tactical driving also involves reasoning about road configurations and traffic control devices to figure out right-of-way puzzles, as illustrated in Figure 2-1. The environment is visually complex because of the many objects and types of objects—cars, roads, markings, signs, signals, etc. These objects vary in appearance under different conditions. Previous work in autonomous road following and car and sign recognition [3, 10, 11, 16, 22, 25] has shown that perceiving individual traffic objects in constrained situations is computationally expensive; finding all objects in all situations in real time is a daunting task.

No one has actually built a traffic scene interpretation system, so we cannot determine the exact complexity of the problem. Two opposing factors make analysis of this unsolved problem especially difficult. First, it is possible that further research and experimentation will yield a very simple and inexpensive method of interpreting some aspect of the scene. On the other hand, real-world computer perception problems tend to be much more difficult than originally expected. For example, the NAVLAB project here at Carnegie Mellon investigated road following for several years. The simple road-recognition techniques that were attempted first often failed when conditions weren't ideal.

In the remainder of this section we examine the perception problem for driving. We explain how we define the environment and the task, and what assumptions we make about sensors and sensor data processing. This allows us to estimate the computational cost of using general perception for driving. Later, we use the same definitions and assumptions as we explore an alternate approach to perception.

The environment. Driving is a complex human activity that is difficult to describe precisely. However, in order program an autonomous robot to perform such an activity, we must be able to describe exactly what it is the robot has to do. First of all, we must describe the important aspects of the environment. In our research, we did this by developing a model of the driving environment and implementing it in a microscopic traffic simulator called PHAROS [28]. PHAROS contains detailed representations of roads, lanes, intersections, signs, signals, markings, and cars. Although PHAROS provides a rich setting for driving experiments, it also makes important abstractions that simplify and limit the driving problem for our work. For example, all roads are structured with lanes; also,

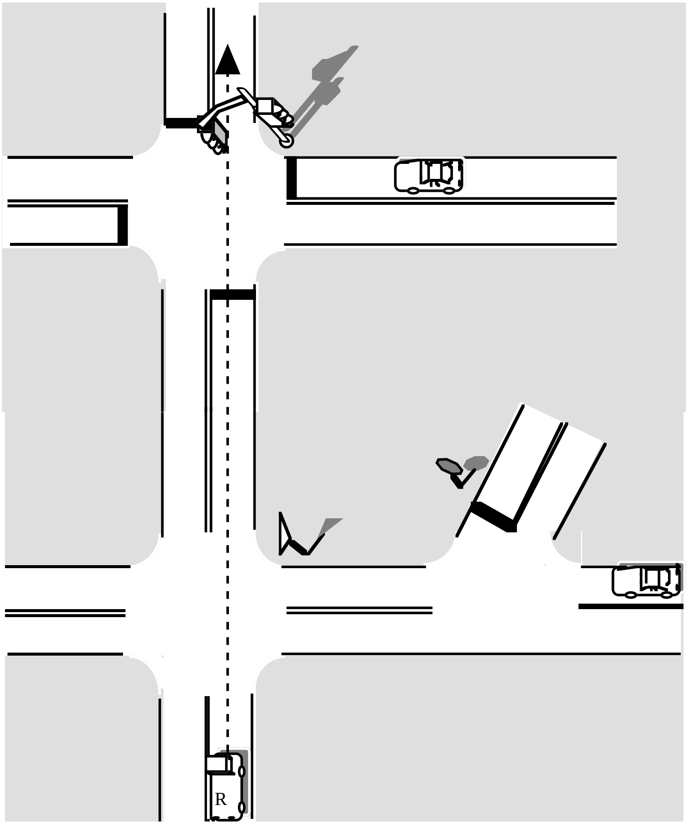


Figure 2-1: To select a speed and maneuver, the robot must determine right-of-way by considering road configurations, vehicle locations, and traffic control devices.

there are no pedestrians or bicylists in PHAROS.

We assume that the environment has shadows, trees, clouds, occluding objects, textures, and reflections. These complicating factors prevent us from using cheap recognition algorithms based on single, uniform features. For example, although sign colors will be useful for segmenting out sign regions, there may be other objects that also have these colors. Additional information will be needed to distinguish the signs from the other regions in the image.

The task. Once we have created a representation of the world, we must specify what it means to drive. Although driving laws are published in books [24], and the driving task has been thoroughly analyzed in isolated situations [26], there are no driving descriptions that actually specify what actions to take at any time. We have developed a computational model of tactical driving called Ulysses [29] that does encode what action to take in any situation in the PHAROS world. Ulysses is a sophisticated model that incorporates knowledge of speed limits, car following, lane changing, traffic control devices, right of way rules, and simple vehicle dynamics. We use a programmed implementation of Ulysses to drive one vehicle in the PHAROS world. Ulysses is our definition of the driving task for this research.

A general perception system has to find all traffic objects of potential interest to the driving planner. In our driving model, these objects include

- Road regions.
- Road markings, including lane lines and any other markings that would indicate turn lanes, for example.
- Vehicles. Vehicles may be turned at various angles. Velocity estimates are required.
- Traffic signs. We require the the robot to recognize signs that are facing up to 45 degrees away from the line of sight. Signs may be up to 7m above the roadway. We consider only a limited set of regulatory and warning signs in this task. The robot must sometimes recognize Stop and Yield signs from the back at intersections [29].
- Traffic signals. We require the the robot to recognize signals that are facing up to 45 degrees away from the line of sight. Signals may be about 6m above the roadway.

These objects are characterized in more detail in Appendix I.

Since it is not practical to consider perceiving the entire world, we arbitrarily set a range limit on the system. In some driving situations it may be desirable to see long distances ahead; for example, signals are supposed to be visible from 218m away on a road with 100kph traffic [12, pg. 4B-11], and the sight distance needed for passing is given as 305m at this speed [1, pg. 147]. While Ulysses is capable of driving on simulated highways, for this work we have concentrated on arterial urban streets where the visual environment is more diverse. Since streets have lower speeds than highways, we have chosen 150m as the perceptual range limit.

Sensors. Since the environment is so varied in appearance, we assume that analysis must be based on several types of sensor data. We assume the robot has cameras and laser rangefinders whose images can be registered. This combination is complementary in that a rangefinder is almost immune to the illumination changes across an object which confuse color based segmentation. A camera, on the other hand, can discern markings and other important regions on uniform surfaces like signs [18]. Since the robot must find objects in all directions, we assume it has sensors pointing in several directions. We thus avoid issues such as sensor aiming time.

We assume the sensors also have the range and resolution to see objects at 150m. The resolution needed at this range is determined by the smallest object. Since flat, horizontal objects like road markings are turned away from the line of sight and greatly foreshortened, they appear to be the smallest objects. Figure 2-2 shows that to cover a foreshortened 10cm-wide lane line with 2 pixels at 150m, a camera mounted 2m above the ground would have to have a resolution of 2.5×10^{-4} degrees per pixel. Appendix I discusses the resolution requirements of different traffic objects in more detail.

Data processing. Interpreting general traffic scenes will be very difficult. We would expect a general perception system to extract many features from the image, including regions, boundaries, lines, corners, etc. The complexity of the environment would in general prevent perception from using single, uniform features to uniquely distinguish objects. Extraction would use intensity, color, optical flow, range, and reflectance data. Features must be grown, characterized, and merged. Scene features would be matched against features of traffic object models to identify traffic objects. This matching would be done in two stages; for example, sign surfaces will first be located before the sign message is examined at higher resolution. Our analysis is explained in detail in Appendix I.

Based on work at CMU on robot driving, and on an examintation of the literature, we estimate

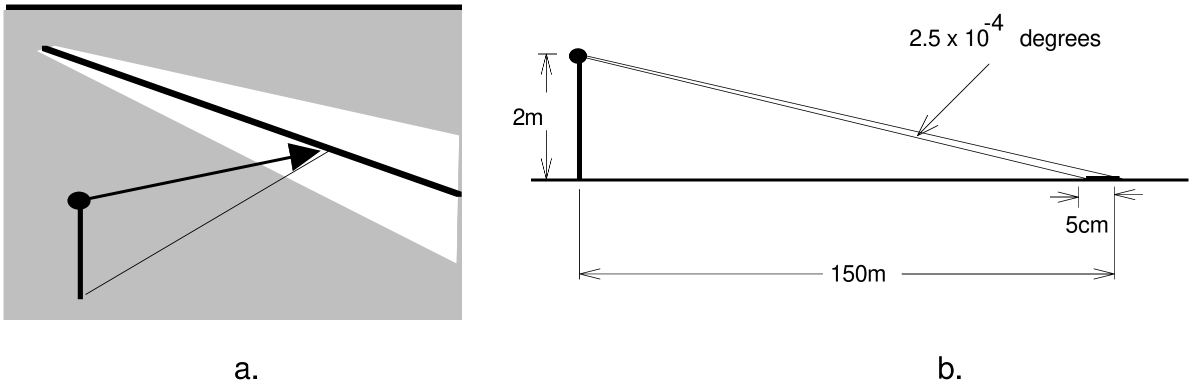


Figure 2-2: Minimum pixel size: a. Elevated sensor looking at transverse lane line. b. Sensor angular resolution required to resolve a 5cm patch at a range of 150m. Sensors are mounted on a 2m-high platform.

that the cost of naive perception in this domain is

$$Cost \approx 1.1{\times}10^4 P + 1.1\,P^2 + 1.7{\times}10^{-5} P^3 \text{ operations} \qquad (1)$$

where P is the number of pixels in the image. The unit cost is an arithmetic operation on a data value or pixel. The linear term reflects the computations to find colors, edges, optical flow, etc. at each pixel; the squared term comes from pixel clustering and comparing pairs of features to each other; and the cubic term is our estimate of the cost of matching, using constraints to prune the search space. Traffic objects are small enough to require about 1cm resolution, which translates to about $8{\times}10^7$ pixels. The net cost is then

$$\begin{aligned} Cost &\approx 8.9{\times}10^{11} + 7.2{\times}10^{15} + 9.0{\times}10^{18} \\ &\approx 9.0{\times}10^{18} \text{ operations.} \end{aligned}$$

This means that a computer that performs 10^8 operations in the 100ms Ulysses decision cycle would be almost 11 *orders of magnitude* too slow to analyze the scene. Even if our estimates are off by several orders of magnitude, it is clear that a general approach to perception is intractable.

3. A Model of Active Vision for Driving

Our system uses two forms of active vision to control visual search. The first involves using knowledge in the planner to direct perception, and the second involves perceptual routines. This section briefly discusses active vision in general and then describes the two forms we use in more detail.

3.1. Characteristics of Active Vision

"Active vision" [4, 5] refers to several related techniques in which the robot moves or points its (limited) perceptual resources to different places for different tasks. A camera can be moved—i.e., the point of view translated—to look at a different aspect of an object [19], to establish optical flow constraints or determine shape and depth more easily. This is especially effective if the camera is rotated at the same time to remain fixated on an object [5]. Cameras can be pointed without translating to track objects and stabilize the image around the object [8]. If the camera boresight represents a limited area of high-resolution vision (i.e., a fovea), then the robot can limit its perception processing costs by pointing the camera only at interesting objects.

A variation of this latter technique is to consider an image of the entire scene taken at one moment in time and use high level knowledge to limit which regions of the image are processed—in essence, defining where a camera with a limited field of view would point. We call this use of high-level knowledge "system-level" active vision to distinguish it from active vision for feature extraction. This method has been used in several mobile robot systems [3, 16, 23, 30]. However, these systems have used processing "windows" that are specific to single objects in known locations. Burt *et al* have proposed more general "focal probes" [9], but have not described how they would be created and located for a particular task. Other planning systems specify what objects to look for as a task executed [13], but they do not define any windows to limit visual search or otherwise simplify the recognition process. Our approach both specifies target objects and localizes visual search.

3.2. Using Task Knowledge to Constrain Search

We limit visual search in our driving system by having the planner direct the actions of perception. Perception is demand-driven rather than forward-flowing. The driving planner uses task-specific knowledge to determine what objects are relevant to the current driving situation.

Since perception only acts upon the request of the planner, Ulysses starts its decision process with almost no information. At the beginning of each decision cycle, perception finds one key object—the road in front of the robot. Ulysses then applies appropriate driving knowledge to interpret the situation and generate action constraints. This knowledge generally involves other objects related to the key object. For example, Ulysses looks for cars on and signs next to the road in front of the robot. Thus the program generates demands for perception system to look for these cars and signs. When objects are found—especially objects indicating an intersection—additional rules may become relevant and more objects may need to be found. Ulysses' demands thus incrementally determine for *what* the perception system must look in the scene.

Ulysses also controls visual search by constraining *where* the perception system has to look in the image. As Figure 1-3 shows, not all traffic objects are important to the robot; only the ones with specific relations to the robot are important. For example, the Stop sign in the middle of the figure affects the robot because it is *along the robot's intended path.* The Stop sign to the far right is not. Ulysses can thus specify where to look for objects—not directly in terms of azimuth and elevation ranges, but in terms of previously observed objects. Ulysses conveys this information by making perceptual requests in the form of perceptual routines.

3.3. Perceptual Routines

Perceptual routines are sequences of primitive image processing operations. Ullman describes how routines may be used in the human vision system to determine object properties and spatial relations [32]. These visual [2] 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

[2] 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.

boundary, and marking points for later reference. While the input to such routines is an image processed from the bottom-up, the routines themselves are invoked from the top-down when needed in different tasks. For example, the routines can be used to determine if two points lie within the same contour. Agre and Chapman used visual routines in their video game-playing system, Pengi [2]. These routines used the primitive actions described by Ullman. The Pengi planner executed visual routines just like any other actions in order to get information about the world state. Since the planner used rules that always related objects to one another or the agent, the visual routines were able to find important objects directly by computing the appropriate spatial relationships. For example, a visual routine could find "the block that the block I just kicked will collide with."

The perception system for Ulysses was inspired by Pengi. As we described above, the rules for driving also have spatial relations incorporated in them. Ulysses uses routines to directly find roads related to the robot, cars related to the roads, signals related to particular intersections, etc. Figure 3-1 lists the routines currently available to Ulysses. The routines in Ulysses assume more domain-dependent processes are available to the routines than is the case in Pengi; for example, the routines must understand how to trace roads instead of just contours, and how to stop scanning when a sign is recognized. Most of the routines in the figure mark objects and locations when they finish, so Ulysses can continue finding objects later. For example, track-lane stops 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. The next section provides a detailed example of how Ulysses uses routines.

Find current lane	Find next car in lane
Mark adjacent lane	Find crossing cars
Track lane	Find next sign
Profile road	Find next overhead sign
Find intersection roads	Find back-facing signs
Find path in intersection	Find signal
Find next lane marking	Marker distance

Figure 3-1: Perceptual routines for Ulysses.

Agre and Chapman point out two advantages of routines in terms of knowledge representation. First, only relevant objects in the world need to be represented explicitly in the agent's internal model. Pengi, for example, does not have to model all of the blocks on the video screen. Second, when the planner needs to know if there is an object with a certain relation to a reference object, it does not need to check all visible objects to find out.

These representational advantages have perceptual analogs which are illustrated better by

Ulysses because it addresses perception more realistically. First, not only do perceptual routines avoid having to represent all world objects internally, they avoid having to *look* for all the objects. In Pengi perception was abstract and essentially cost-free; for a real driving robot, looking for things is extremely expensive and any reductions in sensing requirements are important. Second, checking the relationship between two objects is more complicated than, say, looking at a value in a property list. Computing arbitrary geometric relations may be difficult, and it is much better to do it once and find the appropriate object directly. For example, finding a car on a road could involve making a region-containment test on every visible car; it would be better to scan the road region until the car was found.

The routines listed in Figure 3-1 are clearly specialized for the driving task. They are designed to allow the domain-specific knowledge in the planner to be used for controlling perception. A system that uses such routines, as illustrated in Figure 1-4, would use different routines for different tasks. Our perception philosophy thus agrees with the task-oriented vision ideas of Ikeuchi and Hebert [21].

4. Driving with Perceptual Routines

4.1. How Routines Are Used

We begin this section by illustrating how Ulysses uses perceptual routines in a simulated driving scenario. In our driving model the role that traffic objects play depends on their relation to the corridor ahead of the robot. The corridor is the series of lanes and intersections downstream of the robot that follows the robot's intended route. In a given situation, the robot first finds the pieces of the corridor ahead. The knowledge about roads encoded in Ulysses then indicates, for each corridor piece, for what and where the robot should look. Perceptual routines are the language Ulysses uses to describe these perceptual actions.

The situation, shown in Figure 4-1, is that the robot is approaching a left-side road and must turn left. Traffic can appear on any road. There are a few speed limit signs, a route sign, a STOP sign, a "left turn only" sign overhead and a "left turn only" arrow on the pavement. The robot's road has four lanes, while the side road has only two.

In the following series of figures we show the perceptual activity during one decision cycle of the robot. We pick up the robot after it has changed lanes and is approaching the intersection (Figure 4-2). There is a car approaching from the through direction, one in adjacent lane behind, and one in the oncoming lanes (having turned from the side road).

The first thing the robot does is find the lane immediately in front of it. It then tracks the lane forward (stopping at the intersection) to obtain the first section of the corridor. The robot then looks for a cars in the lane and signs along the side of the road. Figure 4-3 shows the areas scanned by these four routines. The lane tracking routine leaves a marker at the entrance to the intersection.

Figure 4-4 depicts the process of determining the channelization of the robot's lane (i.e., whether it can turn left from the lane). Ulysses scans the width of the road—profiles it—at the intersection (at the marker left earlier) to determine the position of the robot's lane at the intersection. The robot

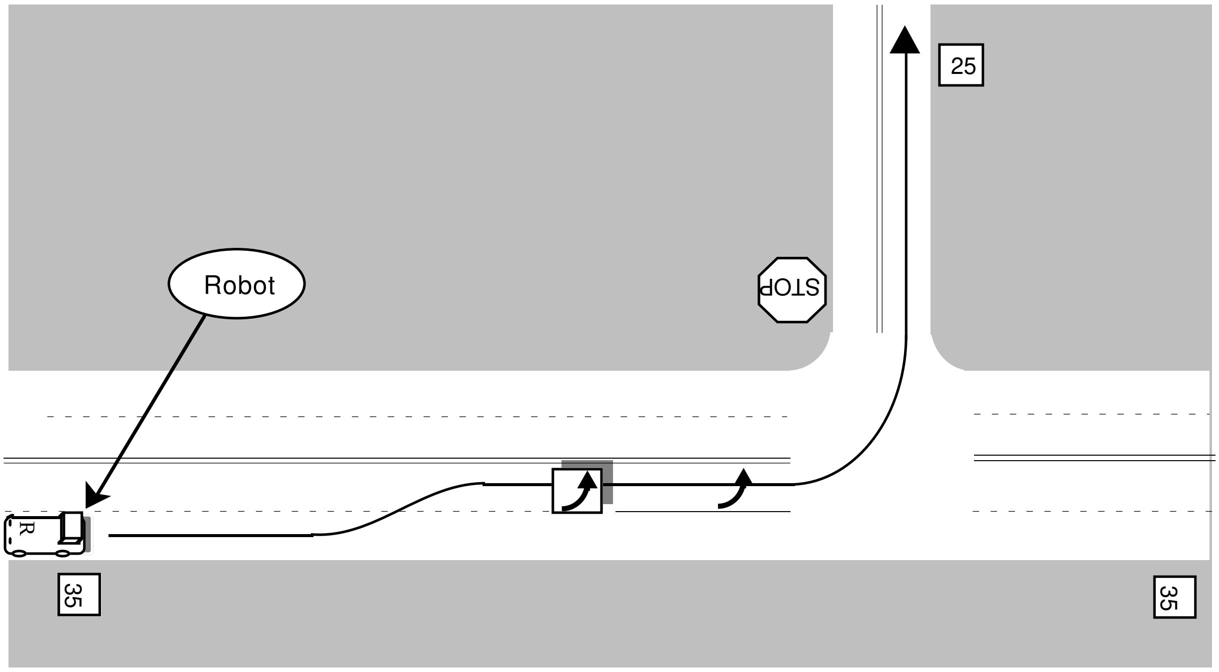


Figure 4-1: Left side road scenario (Not to scale).

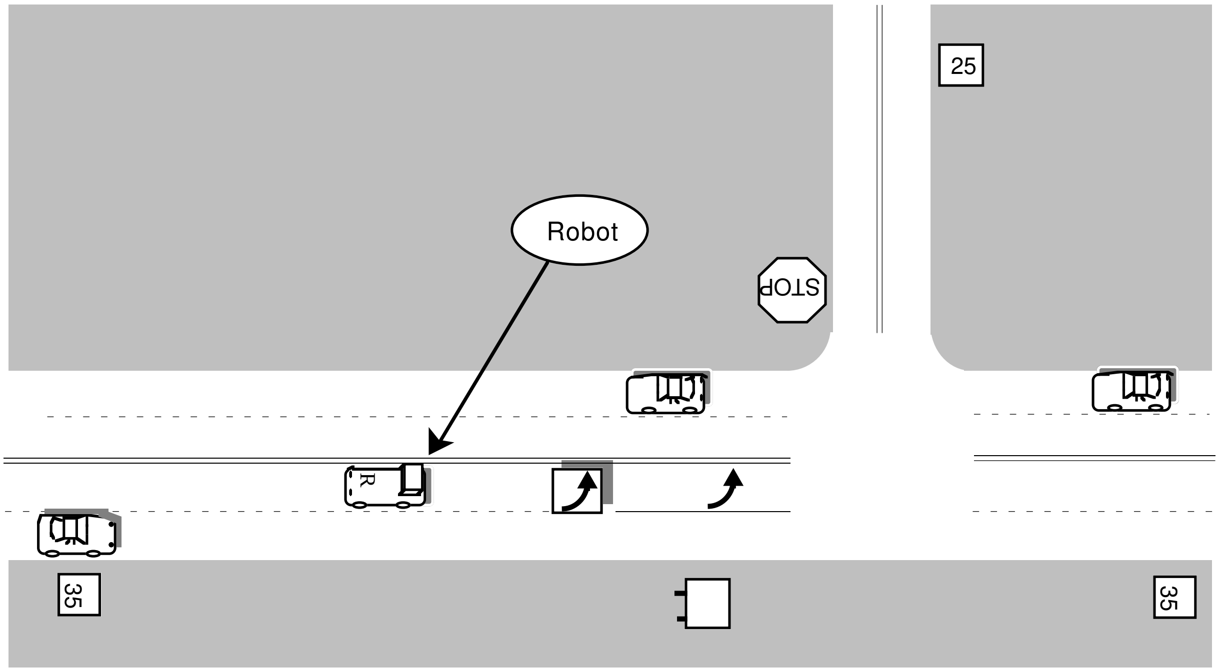


Figure 4-2: Partway through the left side road scenario.

also looks for signs over the lane and markings in the lane to indicate turn restrictions. These scans are repeated until the routines have reached the end of the lane. Markers indicate where one leaves

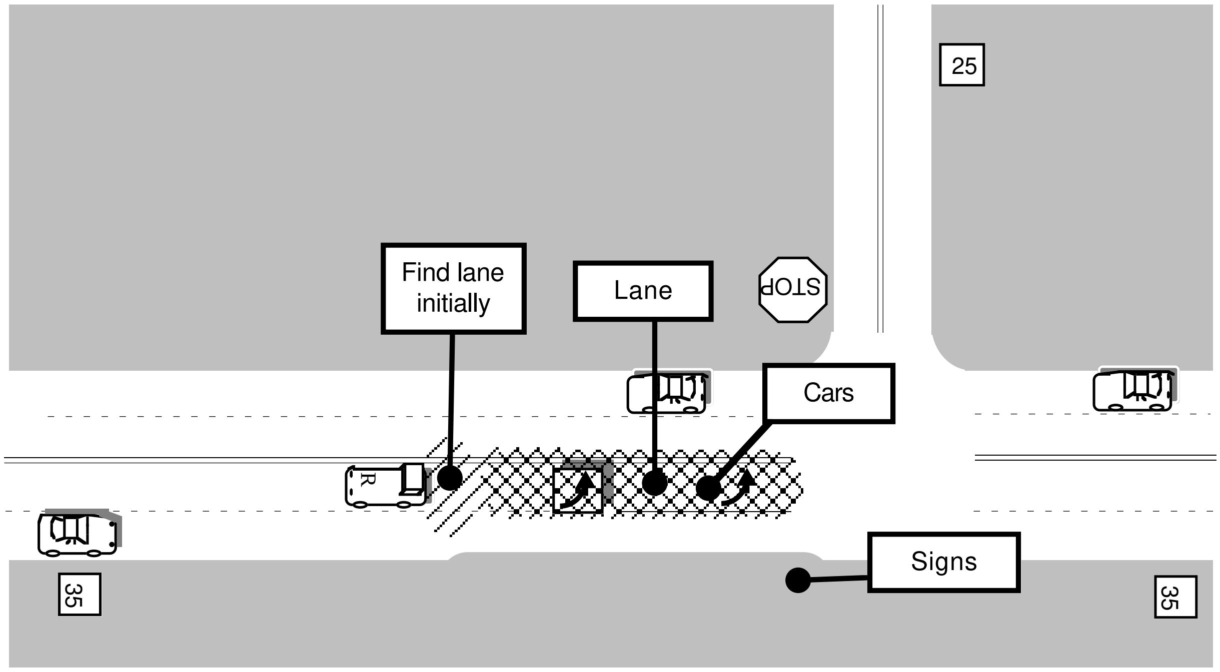


Figure 4-3: The robot scans for the lane, cars, and signs ahead.

off (at an observed sign or marking) and the next one starts.

In Figure 4-5 Ulysses is finding the next part of the corridor and analyzing the intersection itself. This requires finding a path in the intersection, looking for lead cars on this path, looking for crossing or obstructing cars, looking for traffic signals around the intersection, and identifying all approach roads. Information Ulysses already has allows it to determine whether there is a traffic control sign facing the robot, and how many lanes the robot's road has. Lane counts help determine which road is bigger and are part of the right-of-way determination process.

Figure 4-6 shows areas scanned to check for approaching traffic on each road, and for STOP or YIELD signs facing this traffic. Ulysses first finds the lanes on the approach roads, and then looks for the closest car to the intersection in each lane.

In Figure 4-7 the robot is looking beyond the intersection to its downstream road. As with the first piece of corridor, Ulysses first finds the lane, then looks for cars and signs. In this case one sign is found.

At this point Ulysses has examined everything necessary to constrain the robot's speed. The remaining steps are performed to select a lane. Ulysses considers lane selection even as the robot approaches the intersection because in some situations it may be desirable or necessary to change lanes to move around a slow car. Figure 4-8 shows that Ulysses finds the adjacent lane (to the right only, in this case) and analyzes it much as the robot's lane. Ulysses tracks the lane ahead to the

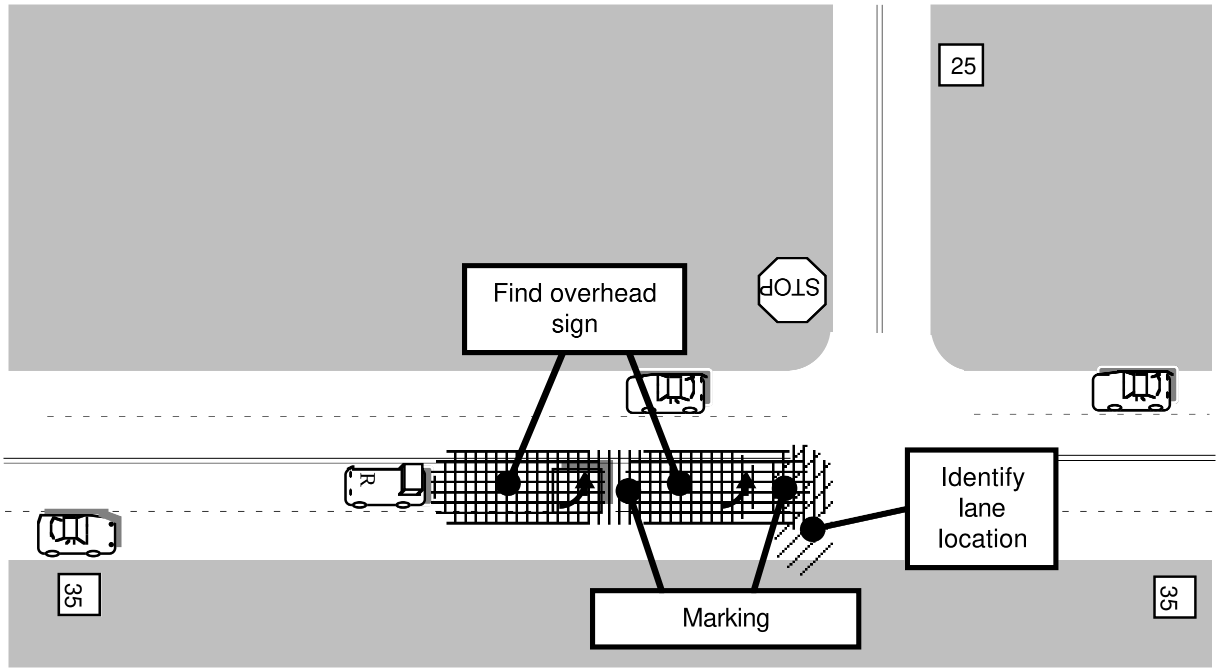


Figure 4-4: The robot checks lane position and looks for channelizing signs and marks.

intersection, and looks for cars, overhead signs (signs to the right have already been found), markings, and lane position at the intersection. The robot also looks in the lane behind to find the car behind.

4.2. Implementing Ulysses

Directing Perception. The preceeding example shows how Ulysses uses knowledge of roads and intersections to find objects relevant to each piece of the corridor. The implementation of Ulysses reported here uses only the corridor and the routines to focus perceptual attention. The robot looks for the corridor as far away as possible, and then looks for all the related objects. There is much potential to incorporate knowledge about static and dynamic objects, critical horizons, etc. in Ulysses; our current research addresses some of these issues. In this section we will show, however, that even just the use of the corridor and selective routines can reduce the perceptual cost of the task by several orders of magnitude.

Execution Architecture. Ulysses has a simple architecture that makes it almost completely reactive. In other words, at each decision time—every 100ms—Ulysses makes a fresh analysis of the situation without using any information from the past (except for a few bits of state [3]). Ulysses thus does not actually make and execute plans, but continuously decides what the robot should do *now*. We are assuming for now that any number of perceptual routines can run in one decision cycle. This scheme allows Ulysses to respond quickly to any new situation.

[3]Also, to obtain velocity estimates for objects, previous images must be retained so that optical flow may be computed

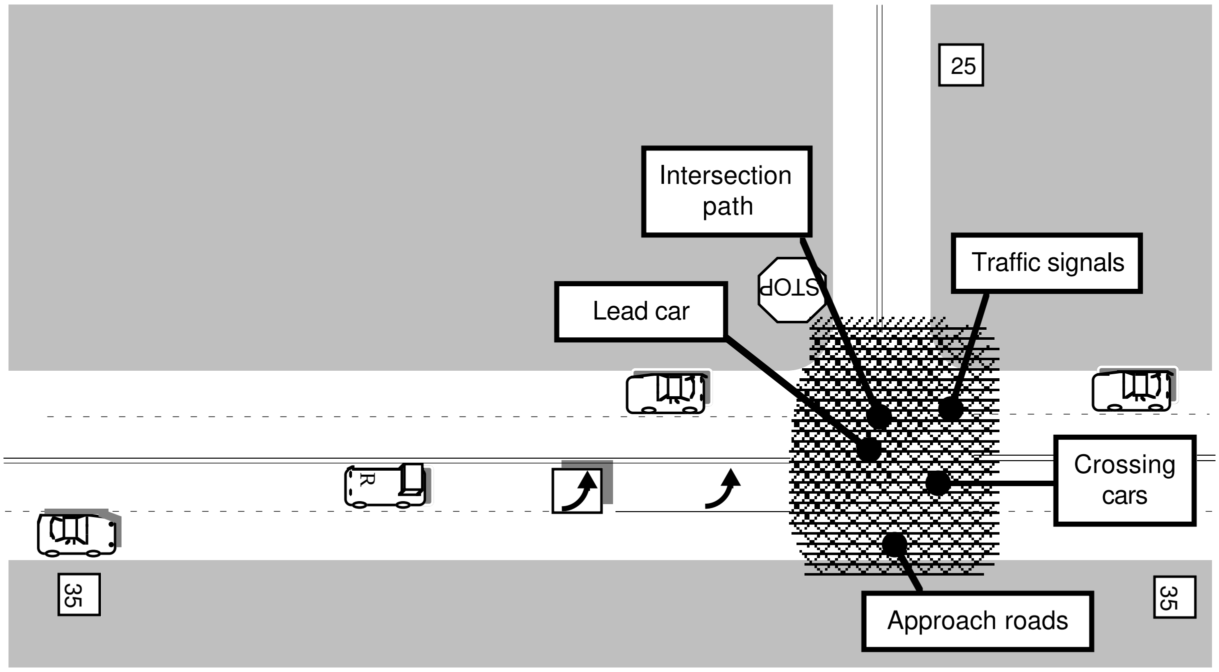


Figure 4-5: The robot looks for a path in the intersection, and then cars, signals, and approach roads.

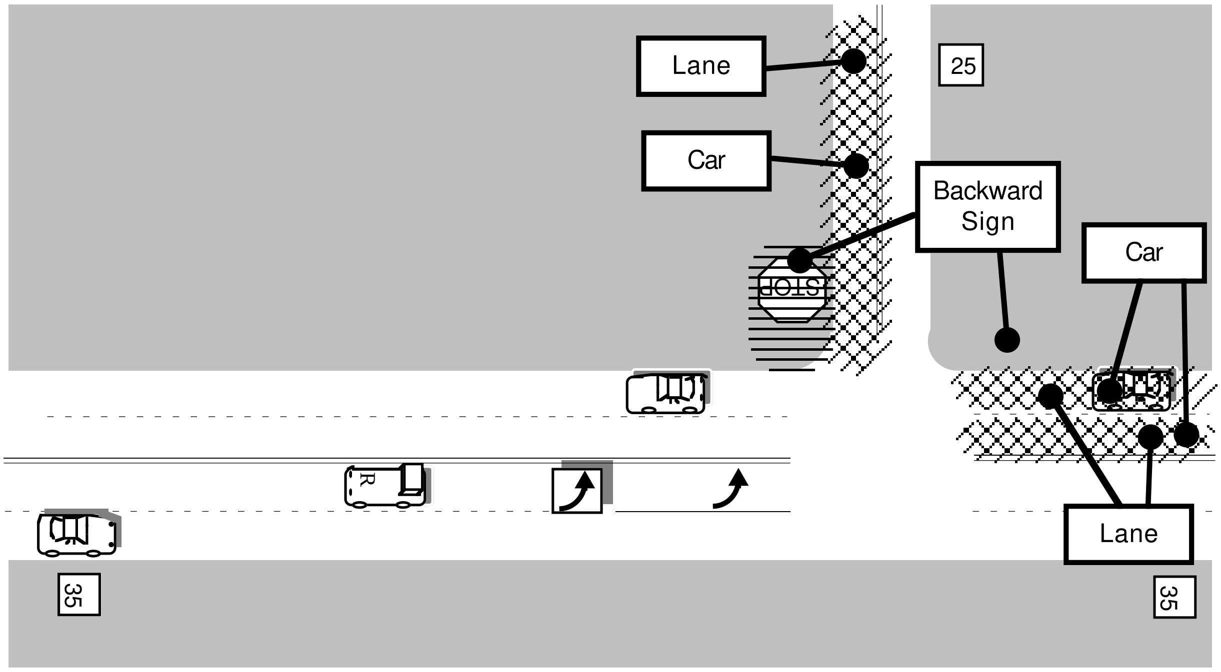


Figure 4-6: The robot checks the intersection approaches for traffic and signs.

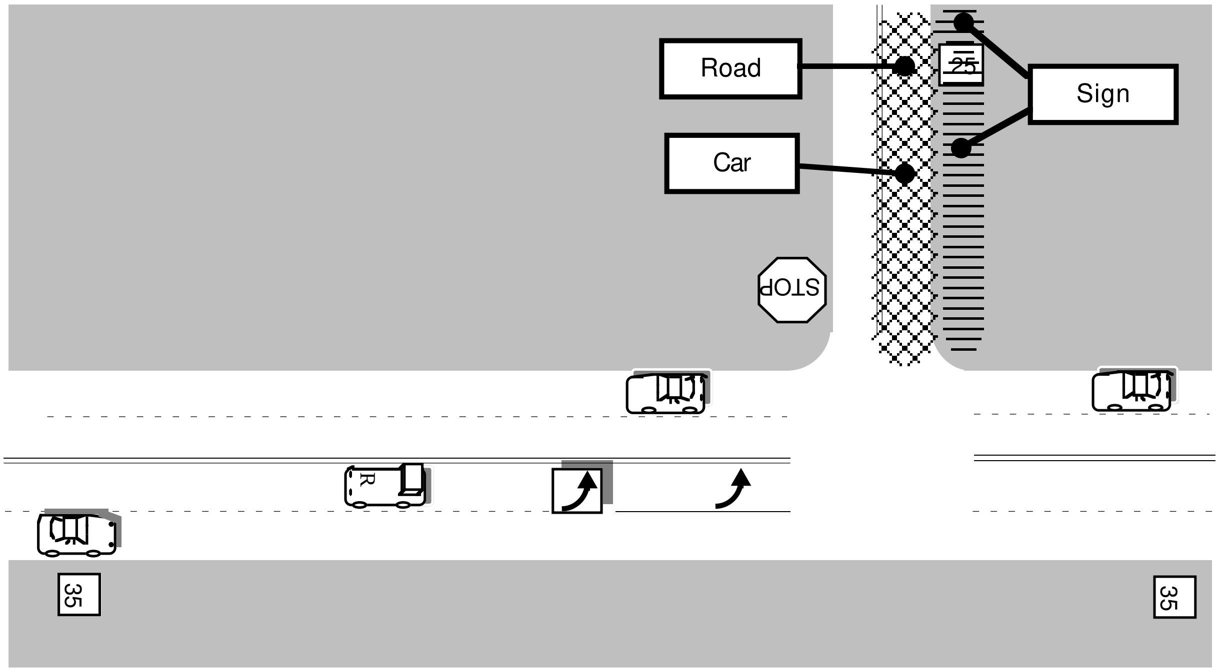


Figure 4-7: The robot looks for a lane, cars, and signs downstream of the intesection.

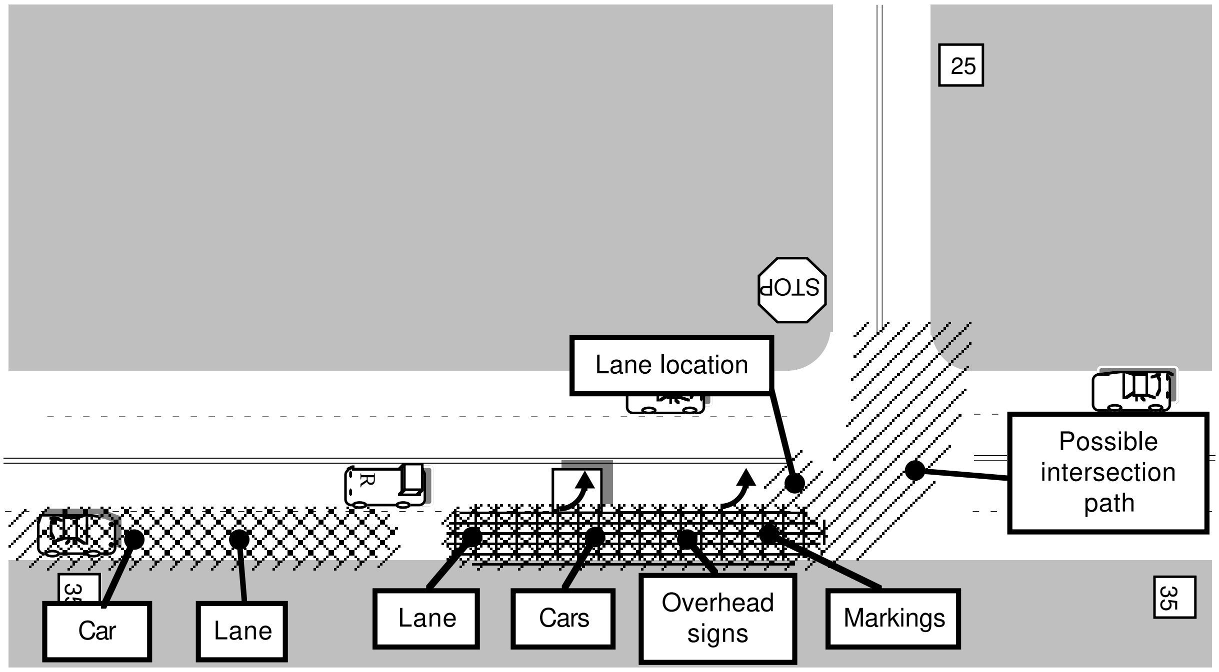


Figure 4-8: The robot looks forward and backward for constraints affecting the adjacent lane, and traffic in that lane.

An alternative to a reactive system is one that builds an extensive internal model of the world. When such a system requires information to reason about the situation, it may then get it quickly from the model rather than from sensors. However, in a dynamic domain the world may invalidate the model quickly so the system must update some parts anyway. We have chosen the reactive architecture for now in order to highlight visual search problems.

The reactive system may also seem unrealistic because it requires the perception system to do all of its work in a short time interval. However, time can be traded off against the number of cameras and computational resources, so we can still use this model by interpreting cost in terms of *resources* rather than *time*.

Simulated Perception. Most of the cars in PHAROS (which we call "zombies") do not perform any perception or scene interpretation tasks. However, the car controlled by Ulysses gets all of its information about the world through a restricted interface to PHAROS. Figure 4-9 shows this relationship. PHAROS simulates the execution of perceptual routines and passes symbolic information back to Ulysses. By monitoring the interface between the two programs, we can determine exactly what information Ulysses requires at any time. In addition, the interface automatically calculates the cost of perception for each decision cycle, using the estimates described below.

4.3. The Cost of Using Routines

The perceptual routines are subject to the same environmental conditions as the naive perception system, so must pay the same feature extraction costs in general. However, 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 simpler techniques to extract a more limited variety of features 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.

In some cases recognition algorithms used by the routines could be simpler than those used in the naive system. For example, when looking for a car on a road ahead, it is only necessary to detect a large solid blob. The routine is looking only on a road, so any blob there may reasonably be assumed to be a car. Fewer features are needed to recognize blobs than are needed to recognize cars. However, for the purposes of this paper we assume that the recognition method is the same for routines as it is for the naive system.

Although the PHAROS world is artificial and devoid of real-world features, perception costs for Ulysses must be estimated for the same complex environment that we assumed for the naive system.

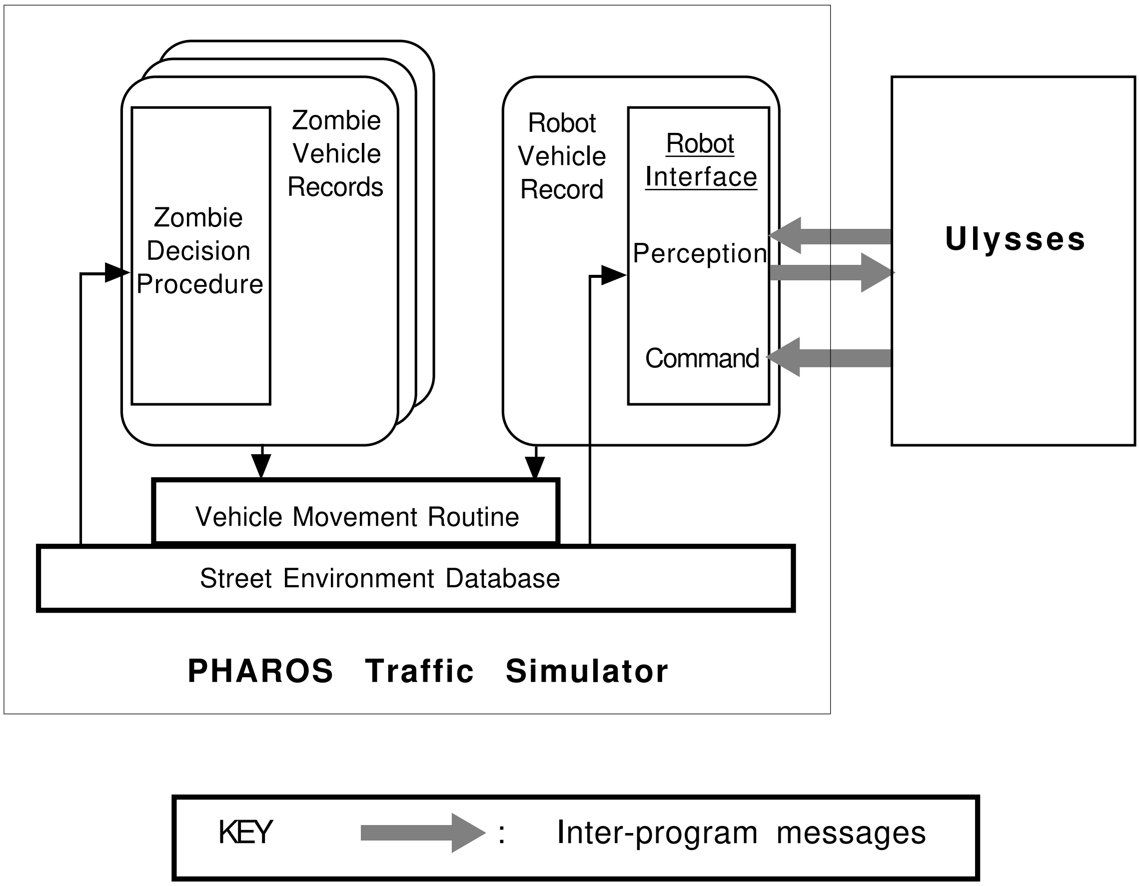


Figure 4-9: The Ulysses driving program controls one car in the PHAROS traffic simulator, and communicates via simulated perception.

Therefore we use essentially the same equation we used for the naive system (Equation 1). However, we use a feature size particular to the routine (from Table I-1), and the particular area searched by the routine (as illustrated by Figures 4-3 through 4-8 above) to calculate the number of pixels. Resolution is also allowed to decrease as the search area gets closer to the robot (as it is in the naive case). Appendix II explains the cost calculations in detail. Appendix III shows the cost of the individual perceptual actions described in Figures 4-3 through 4-8 above.

4.4. The perceptual cost of different driving situations

We have created several driving situations using PHAROS and used Ulysses to drive a robot through them. By tracking the perceptual costs over time, we can see how the cost of routines compares to the cost of naive perception, and what situations cost the most with the given routines. The scenarios we use are a four-way intersection with no traffic control (unordered intersection), a four-lane highway with no intersections, the left-side-road scenario described above, an intersection of four lane roads controlled with traffic lights, and a set of closely space intersections of two-lane roads. In addition, the approach roads in these scenarios illustrate the cost of driving on two- and four- lane roads without intersections.

4.4.1. Unordered Intersection

Figure 4-10 shows the unordered intersection scenario. The robot approaches the intersection and must yield to a car on the left before proceeeding with its left turn. The car on the right is expected to yield to the robot. There are several miscellaneous signs along the roads. The circled numbers are reference points that correspond to the labeled points in the following figure.

Figure 4-11 is a graph of the perceptual cost in operations during the scenario. These values were computed as described in Appendix II by the interface to PHAROS. The cost of perception is dominated by the lane search cost, which in the figure is indistinguishable from the "Total cost" plot. This relation is also true in all other scenarios. Searching the lane is the most costly type of activity because foreshortened lane and stop lines are the smallest objects in the environment (see Appendix I). Since this version of Ulysses does not use any knowledge about event horizons, but naively searches to its sensor range limits, the cost of finding lanes can be very high.

The perceptual cost before point (1) in Figure 4-11 is approximately 2×10^{15} operations; this represents the cost of a simple two-lane highway without intersections. After point (1), the robot has detected the intersection and begins to search for signals, roads, cars, and additional signs according to the corridor model. At point (2), the sensors can reach the far side of the intersection and detect the approach roads, so these roads are scanned for lanes, cars, and backward-facing stop and yield signs. In addition, the robot begins looking at the corridor downstream of the intersection. At point (3) the robot enters the intersection and ceases its search for signals, approaching cars, and other objects that would affect its right-of-way decision. Just before entering the intersection, the perceptual cost reaches its peak at about 7.7×10^{15} operations. This is over three orders of magnitude less than the cost using general perception (about 9×10^{18} operations). At (4) the robot no longer needs to look for unexpected cross traffic in the intersection either. After (4), the robot returns to a two-lane highway situation.

4.4.2. Four-lane Highway

Figure 4-12 is the four-lane highway scenario. In this case the robot has to pass a slower car using the left lane, and then return to the right lane. There are no other traffic objects except for various signs along the side of the road.

Figure 4-13 shows the estimated costs of using perceptual routines in this situation. Before point (1) and after point (5), the robot is simply driving a four-lane highway with no traffic or intersections. The cost of this task is about 1.5×10^{16} operations—a factor of about 600 smaller than the cost using general perception. At points (2) and (4), the robot commits to a lane change and stops looking behind itself in the adjacent lane for traffic. This causes a sharp decrease in the lane-search and total cost.

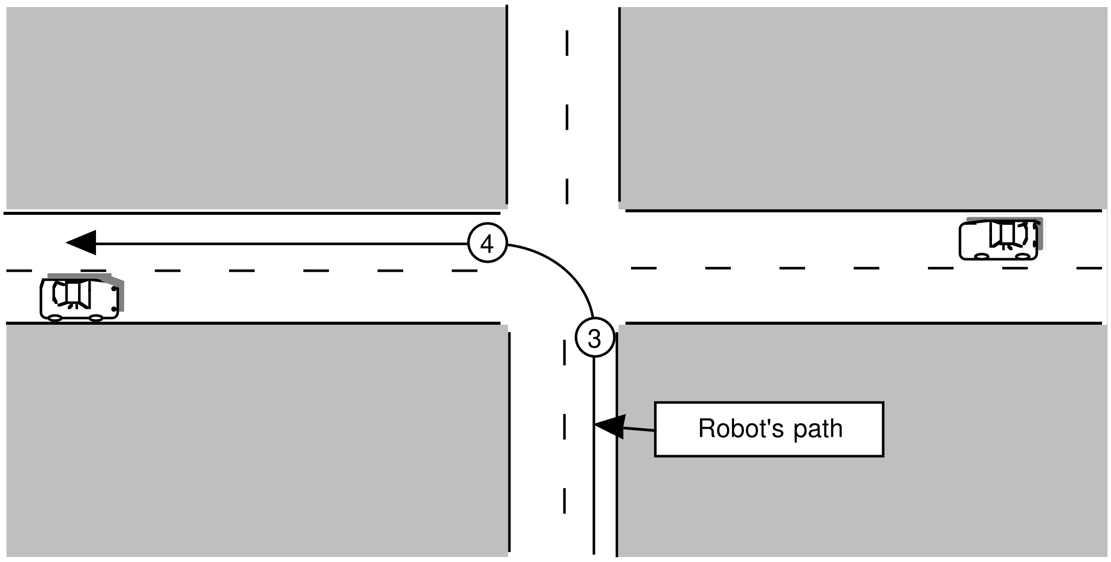


Figure 4-10: Unordered intersection scenario (not to scale).

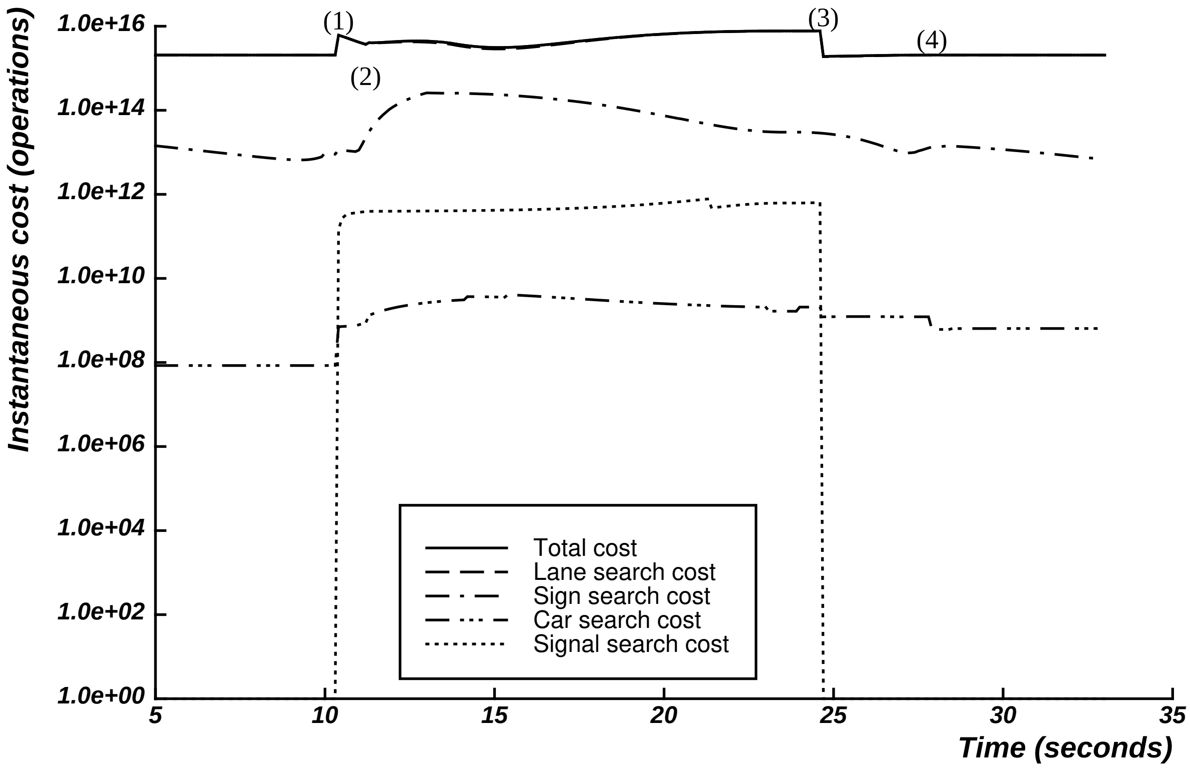


Figure 4-11: Perceptual costs during scenario. Notes correspond to figure above, except: (1) sensors reach intersection; (2) sensors reach all intersection approach roads. Total cost is nearly the same as lane search cost.

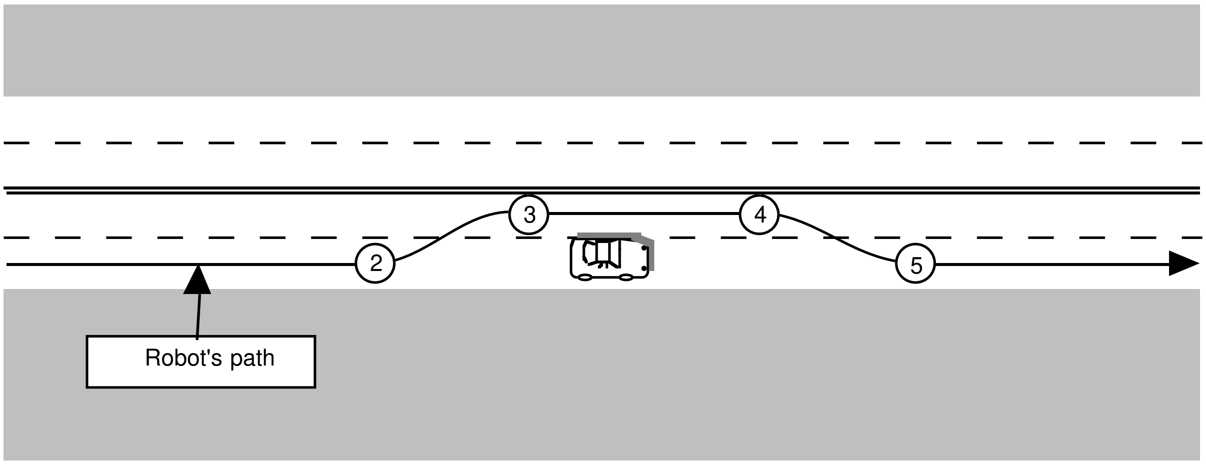


Figure 4-12: 4-lane passing scenario (not to scale).

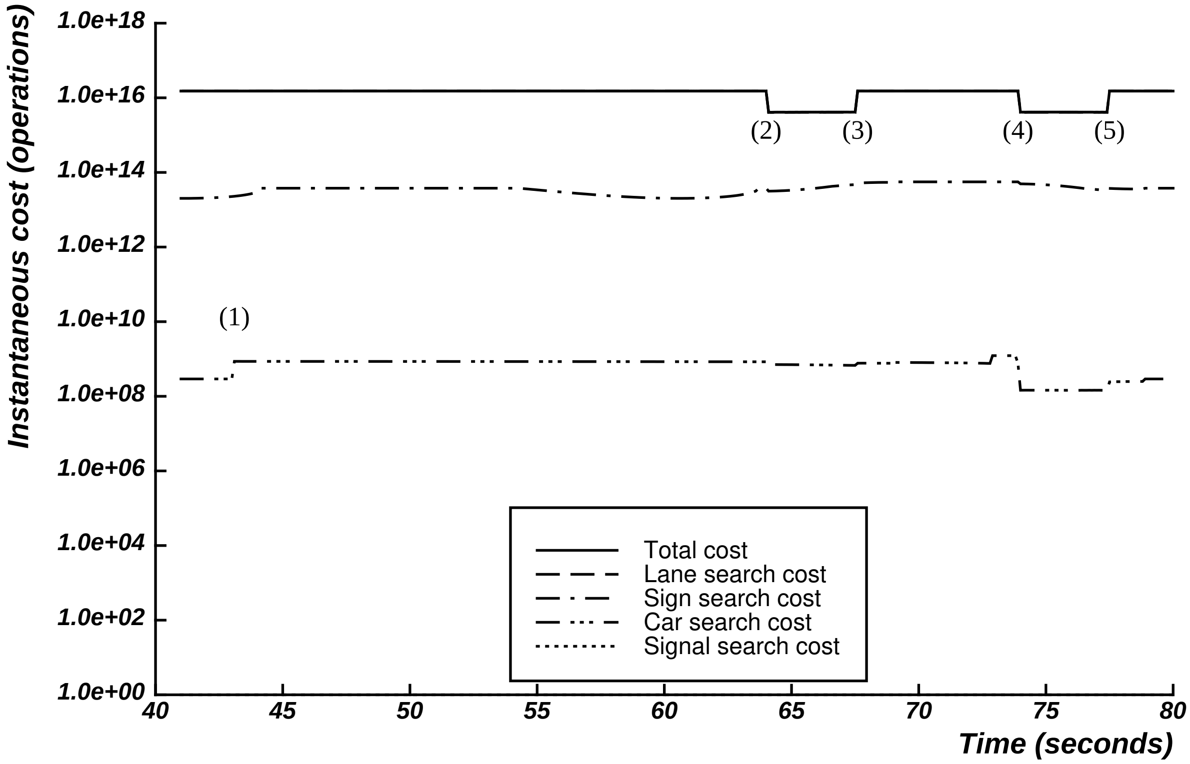


Figure 4-13: Perceptual costs during scenario. Notes correspond to figure above, except: (1) sensors reach lead car. Total cost is nearly the same as lane search cost.

4.4.3. Left Side Road

Figure 4-14 illustrates the left side road scenario introduced earlier; Figure 4-15 is the graph of perceptual cost over time. This graph combines some of the same events of the previous two scenarios. At point (1), the sensors detect the intersection and trigger additional searches for signals and cars. In addition, the robot determines that it cannot turn left from the right lane and begins a lane change. As with the previous scenario, this action causes lane search costs to decrease. At point (2), the sensors detect the robot's corridor on the far side of the intersection and begin searching for signs, cars, etc. on that road. At (3) the sensors can see all approach roads, so the robot begins searching their lanes for approaching cars. After the lane change is complete at (4) the scenario is approximately the same as the unordered intersection scenario. Appendix III shows the costs of each perceptual action at this point. Note that the cost before (1) is that of a four-lane road, and it drops after (6) to that of the two-lane road.

4.4.4. Intersection With Traffic Lights

Figure 4-16 shows the traffic light scenario. This scenario includes more traffic objects than previous scenarios, including signals and more markings and signs. Not shown in the figure are cars that approach from all directions during the scenario. The robot enters in the right lane and is required to make two lane changes before entering the intersection. The left-most signal head facing the robot shows a left-turn arrow after the robot has waited at point (6) for a few seconds. After the robot crosses the intersection, it again moves to the right-most lane.

Figure 4-17 shows the estimated perceptual costs for this scenario. The graph's features reflect sensor detection events, lane changes, and intersection entries as described for previous scenarios. The costs are not significantly affected by the additional objects in the environment; for example, signal search costs are almost the same in this scenario as in the left side road scenario. The cost of searching the intersection for signal heads apparently dominates the cost of finding lens symbols on the heads that are found.

4.4.5. Multiple Intersections

The final scenario includes multiple intersections. Figure 4-18 shows that the robot must traverse two closely spaced intersections, and scan across an intersection on an approach road. At the first intersection the robot has a Yield sign, but determines that the vehicle approaching from the right is stopped and will not conflict with the robot. The robot therefore determines that it can proceed freely through the first intersection. However, the Stop sign at the second intersection forces the robot to slow down even before it reaches the first intersection. At the second intersection, the robot waits for crossing traffic to clear before proceeding. Figure 4-19 shows the estimated cost for this scenario.

5. Conclusions

We have described how tactical driving is a complex task in a complex environment. While clever bottom-up image processing techniques will be necessary for a robot to handle the perceptual load in this domain, they will not be sufficient. A robot must also use task knowledge to constrain the perception problem. We have designed a driving model and used it to control perception for driving—to specify what the robot should look for and where it should look for it. With these constraints, the perception task is simpler and and can be performed more quickly by special

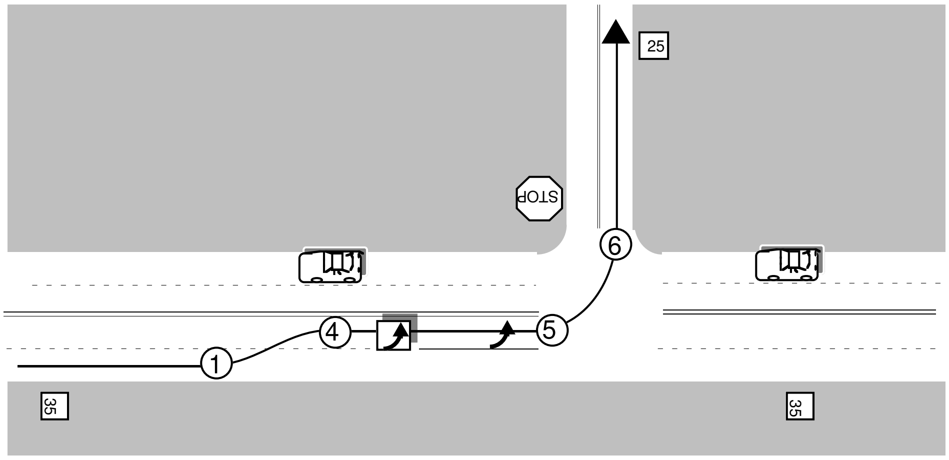


Figure 4-14: Left side road scenario (not to scale).

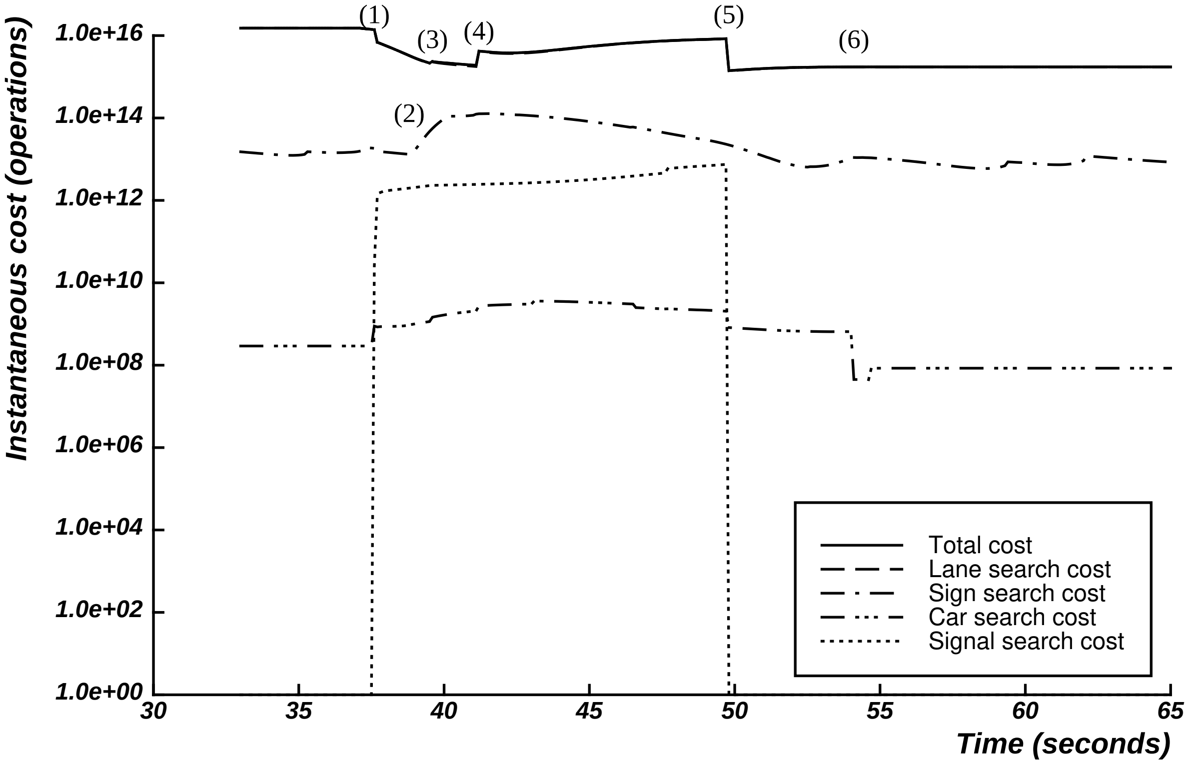


Figure 4-15: Perceptual costs during scenario. Notes correspond to figure above, except: (1) sensors reach intersection; (2) sensors reach robot's street on far side of intersection; (3) sensors reach all intersection approach roads. Total cost is nearly the same as lane search cost.

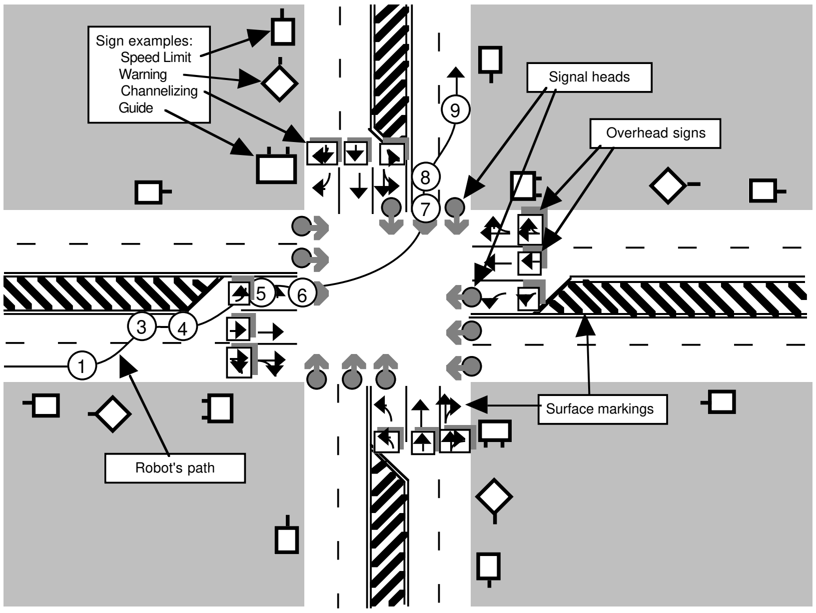


Figure 4-16: Traffic light scenario (not to scale).

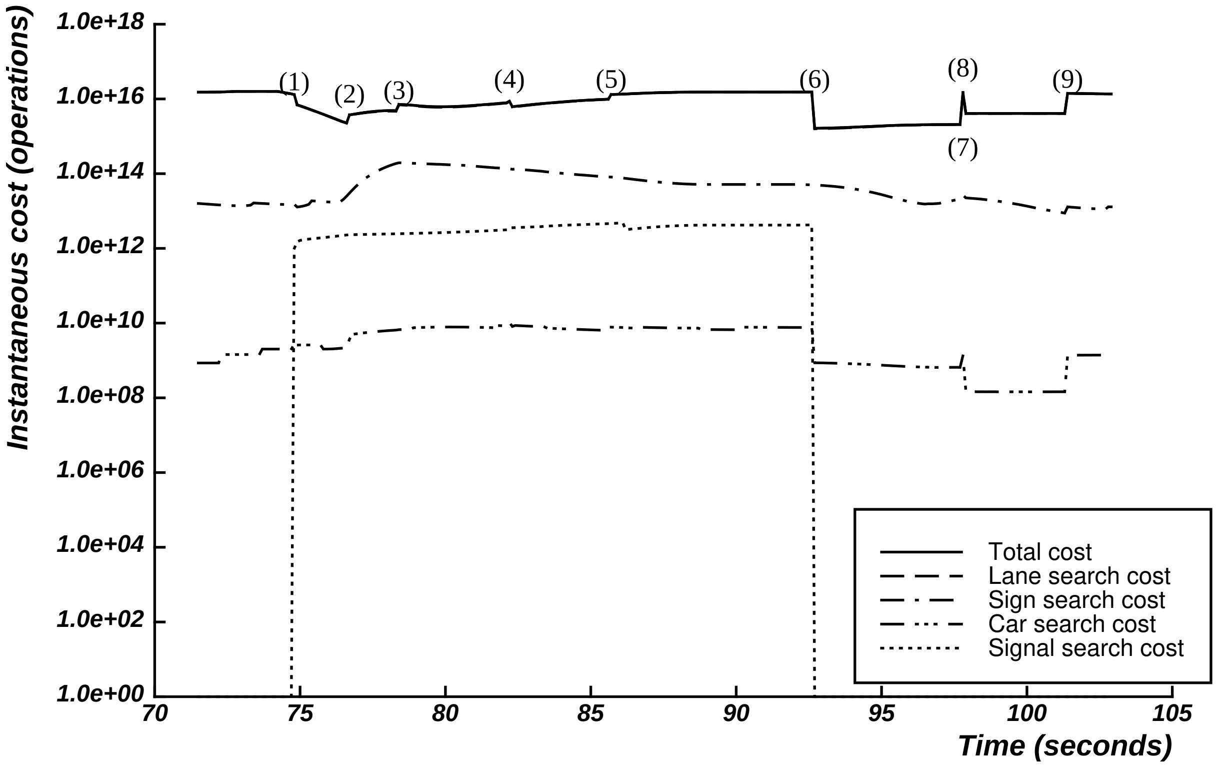


Figure 4-17: Perceptual costs during scenario. Notes correspond to figure above, except: (1) sensors reach intersection; (2) sensors reach all intersection approach roads. Total cost is nearly the same as lane search cost.

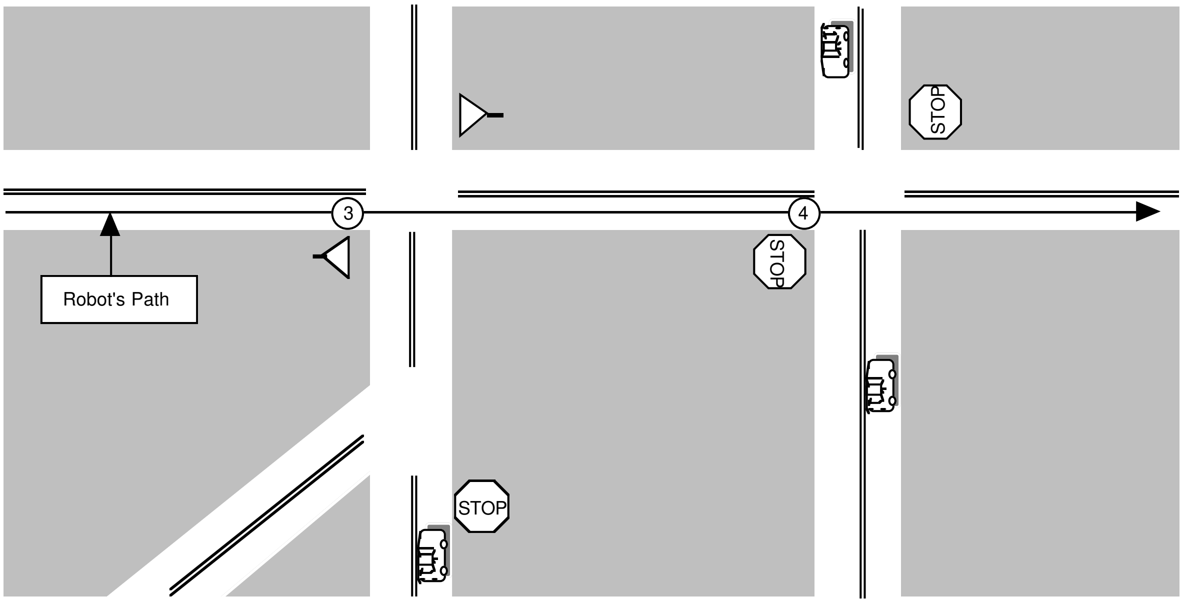


Figure 4-18: Multiple intersection scenario (not to scale).

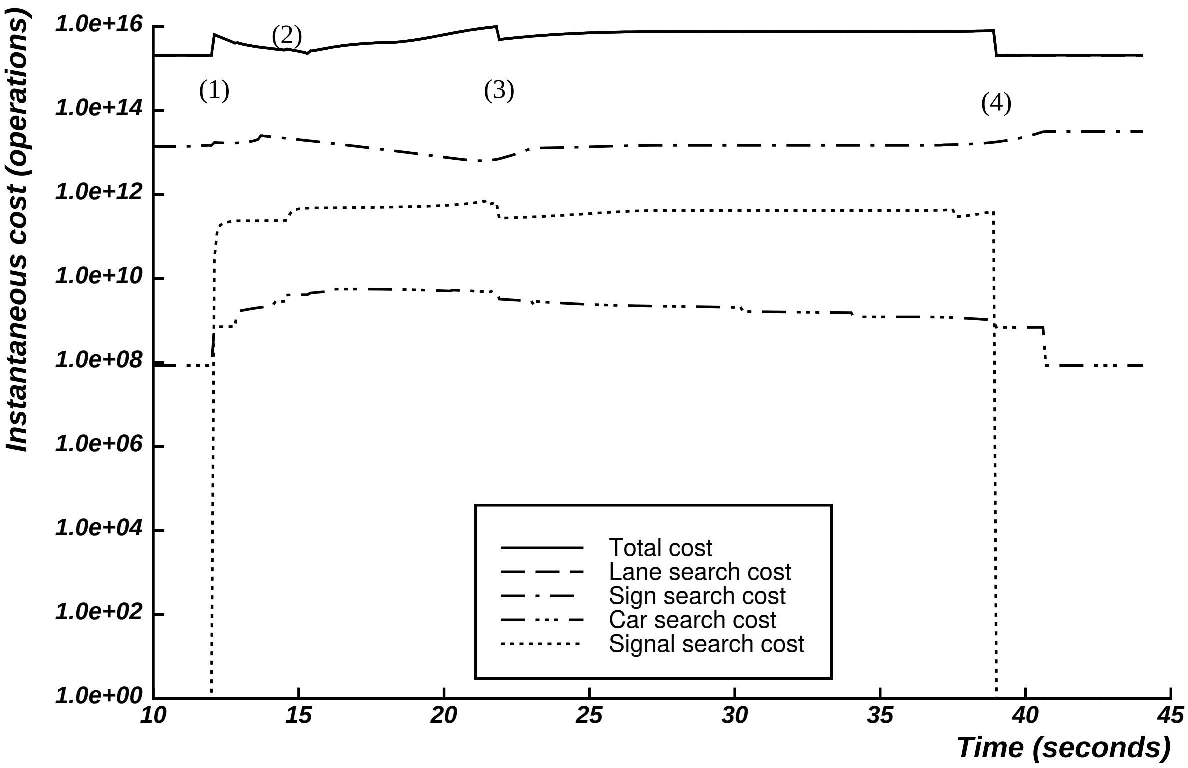


Figure 4-19: Perceptual costs during scenario. Notes correspond to figure above, except: (1) sensors reach first intersection; (2) sensors reach second intersection. Total cost is nearly the same as lane search cost.

purpose routines. A general analysis of the complexity of naive perception and simulation of driving using routines shows that routines reduce compuational cost by several orders of magnitude.

We are continuing to develop our driving model and the perceptual routines for driving. For example, there may be alternate routines or new ways of inferring traffic conditions that are simpler but do not compromise the information needs of the driving model. We are also exploring additional ways of reducing perceptual needs, such as using models of the coherence of the world over time. This research will help to discover ways that a robot can perform interesting tasks in perceptually difficult environments.

Appendices

I. Computation Cost for the General Perception Model

In this appendix we explain our estimates of the cost of general perception. Our unit of cost is an arithmetic operation on one data value or pixel. First we describe the important characteristics of the traffic objects, and then we describe the procedures used to identify them.

Traffic Object Characteristics

Minimum size and number of shape variations are the most important object characteristics for our cost calculations. For signs and signals, we also need to consider the variation in patterns within the object. Figure I-1 shows the dimensions of some traffic objects. These dimensions determine the highest angular resolution needed in an image, so we have pessimistically shown the smallest objects. For example, 20cm is the smaller of two standard sizes for traffic signal lenses.

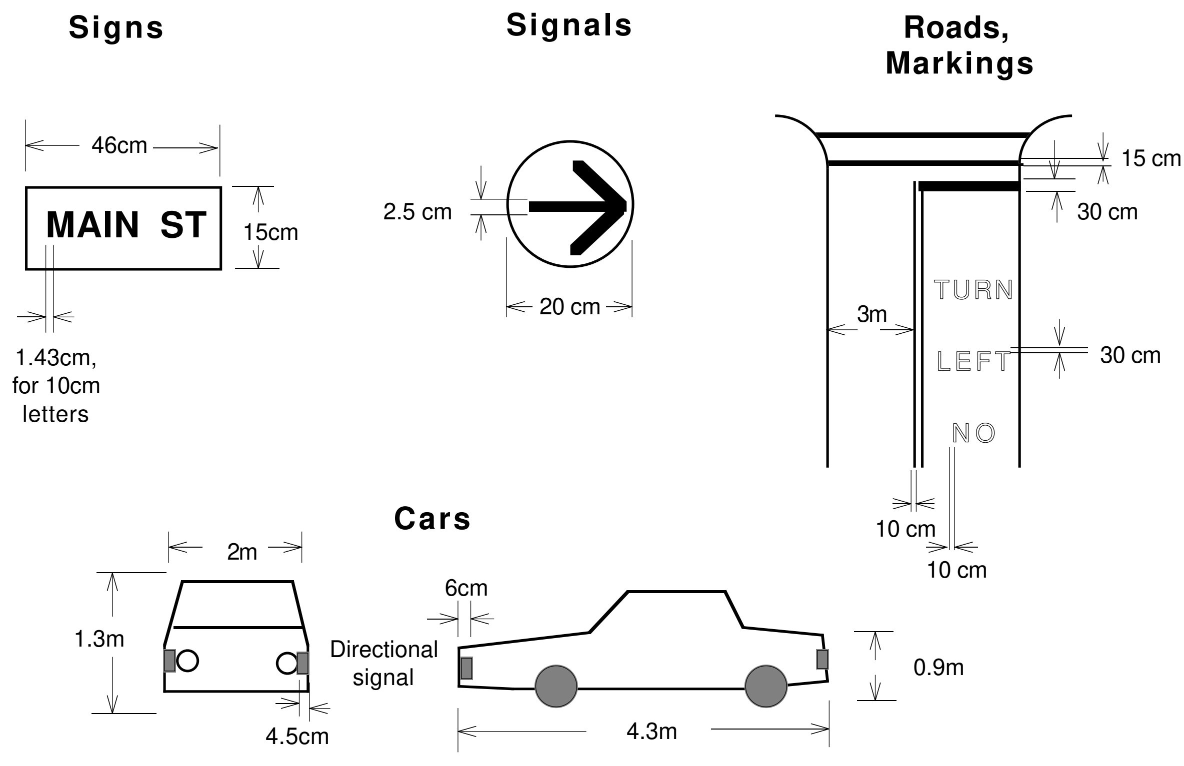


Figure I-1: Dimensions of traffic objects.

We assume that to recognize two-dimensional objects reliably, there must be 10 pixels across the object in the image. Line objects require only 2 pixels across their width to be detected reliably. Table I-1 lists the resulting pixel resolution we require for each object. The table also gives the number of shape variations for each object. For signs, signals and markings, these are taken from lists in the MUTCD [12]. Car variations do not really have to be modeled in detail, because any object that is moving on the road can be considered a car; however, the naive perception system looks everywhere in the scene for vehicles, so must have detailed enough models to avoid falsely

Object Type		Model Variations	Minimum resolution
Road		1	30cm
	Road marking:	Lane lines- 6 (dashes, colors) Crosswalk lines Stop lines Symbols and letters- 42 Total: 50	5cm
Signs		Octagon, triangle; Diamond- 4 (colors), Rectangle- 2(colors) Total: 8	1.5cm
	Sign contents:	STOP, YIELD- 1 Warning- 100 Construction- 60 Regulatory- 50 Route- 40 Guide- 40 Street- 36 Average: 50	0.7cm
Signals		Lens configurations- 10	2cm
	Signal symbols:	Circle, arrows- 5	1cm
Cars		15	10cm
	Turn signals:	1	3cm

Table I-1: Approximate number of model variations and resolution requirements for traffic objects. Indented object types are secondary and are considered only after the primary objects are found.

recognizing every large object.

Recognition Procedures

We assume that the general procedure for recognition will be to extract features from the image and match them to object models. Since the perception system has no guidance, it essentially has to look for all objects everywhere. We allow a few reasonable modifications, however. First, the perception system will only look for roads and road markings at negative elevation angles (i.e., below the horizon). Second, only road regions will be searched for markings. These improvements significantly reduce the pixel resolution needed in the rest of the image, since small, foreshortened road markings are the smallest objects in the scene. Third, the messages on signs and signals (e.g. letters and arrows) will only be examined after sign and signal objects have been identified. This optimization avoids having to look over the entire scene with enough resolution to read the lettering on distant signs.

The recognition procedure is thus as follows:

1. Feature extraction
2. Model matching
3. Secondary matching (detail of signs and signals)
4. Searching for road markings

The cost for recognition is the sum of the costs of these activities.

Feature Extraction. Since no one has yet built a traffic scene interpretation system, and since the environment is so complex, we do not yet know which low-level features will be necessary to recognize objects. Therefore we assume that a very wide variety of low level processing techniques must be used.

There are three general parts to feature extraction: image transformation, image segmentation, and property computation. The cost of feature extraction is the sum of the costs of these parts.

- *Image transformation.* This step includes all processes that calculate iconic features (a value at each pixel). This includes transforming color space, finding edge strength and direction, converting range data to world coordinates and local normals, transforming range and surface normals to alternate coordinates, finding gradients, and making correlations for optical flow. Each operation involves performing about the same computation on each pixel in the image. We estimate approximately $3{\times}10^3$ operations per pixel; thus

  $$\textit{Cost of image transformation} \approx 3{\times}10^3 P$$

  where P is the number of pixels in the image.

- *Image segmentation.* After finding iconic features, they must be grouped into semantic features such as regions, boundaries, corners, lines, etc. This process includes linking edges into lines, clustering image pixels into regions by position, color, depth, reflectance, motion, surface orientation, etc., and splitting and merging regions and lines. These steps also require some global evaluation of the quality of the segmentation using scene knowledge [7, 10, 18]. Segmentation includes three types of computations:

  1. Operations on each pixel ($\textit{Cost} = aP$)
  2. Comparisons between pixels and regions to test membership ($\textit{Cost} = bP \times \textit{Number of regions}$)
  3. Pairwise comparisons between regions to perform global segmentation evaluations ($\textit{Cost} = c \times (\textit{Number of regions})^2$)

  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, $\textit{Number of regions} = \alpha P$. This is because at higher resolutions, when there are many small pixels, more regions are visible. We can thus write an expression for total segmentation cost as

  $$\textit{Cost of segmentation} = a\,P + b\,\alpha P^2 + c\,(\alpha P)^2.$$

  From experiences at CMU in perception for vehicle navigation and examination of the literature [10, 11, 14, 17, 18, 22], we estimate that $\alpha \approx 10^{-2}$, $a \approx 3\times10^{3}$, $b \approx 10^{2}$, and $c \approx 10^{2}$, so the total cost for segmentation is

$$\textit{Cost of segmentation} \approx 3\times10^{3} P + P^{2} + \varepsilon.$$

- *Property computation.* After features are extracted, many geometrical, topological and physical properties of the features will be computed to provide further constraints for matching. These properties could include orientation, area, perimeter, moments, texture, bounding box, surface description, average color, shape description, etc. (e.g., [14]). Adaptive feature models—such as color classes, for example [10]—may also be updated. The cost of these steps will be roughly proportional to the number of pixels, because many of these computations require the examination of every pixel. We estimate that

$$\textit{Cost of property computation} \approx 10^{3} P.$$

Adding these together, our estimate for the total cost of feature extraction is

$$\textit{Cost of feature extraction} \approx 10^{4} P + P^{2} \tag{2}$$

where P is the number of pixels.

Model matching. Many reports have described the basic process of matching S scene features to M model features (for example, the survey by Besl and Jain [6]). A brute-force search through all possible pairings of features is exponentially complex—M^{S} in the worst case. There are many ways to reduce matching complexity using constraints, feature hierarchies, etc. (e.g. Ettinger [11]). However, while satisfactory computation times are often reported, the complexity of these techniques is not generally analyzed because it depends heavily on the particular situation. Grimson [15] did rigorously analyze the complexity of recognizing certain objects using geometric constraints. He showed that the search is expected to be exponential (in M', the visible portion of M), but he also described techniques that reduced the search cost an unspecified amount on actual problems. We have thus found it necessary to develop a new expression to describe the actual expected cost of matching.

For our matching cost estimate we assume a sequential process in which scene features are compared to each model feature in turn (for each model in turn). Table I-2 illustrates the process. For each object model, S scene features are compared to the first model feature. Some fraction β_1 of these comparisons will result in a match after unary constraints are applied. For each of the $\beta_1 S$ matches, $(S-1)$ scene features are compared to the second model feature, for a total of $\beta_1 S(S-1)$ comparisons. Of these potential matches, a fraction β_2 will satisfy the unary and binary constraints, resulting in $\beta_1 S \beta_2 (S-1)$ valid partial (two-feature) interpretations. This process continues until $(S-(M-1))$ scene features are compared to the last model feature. Based on the enormous reductions in the search space that have been reported in the literature, we estimate that $\beta_1 \approx \beta_2 \approx 10^{-2}$, and that for later steps $\beta_i \approx 1/S$. In other words, after the second step the number of valid interpretations will not increase—for each interpretation, only one scene feature will match the current model feature m_n and satisfy all of the n-ary constraints. The total number of comparisons (for each object model) is obtained by adding the number in each step in the table:

$$\textit{Comparisons per model} \approx S + 10^{-2} S(S-1) + 10^{-4} S(S-1)\left((S-2) + (S-3) + \ldots (S-(M-1))\right)$$

$$\approx S + 10^{-2} S^{2} + 10^{-4} S^{3} M.$$

In our case, the number of features is $S \approx \alpha P$ and we estimate $M \approx 20$ features per model, so the number of comparisons for a single object model is approximately $\alpha P + 10^{-2}(\alpha P)^{2} + 10^{-4}(\alpha P)^{3}$. Thus

Step	Model Feature	# Scene Features Remaining	# Comparisons	# Resulting Matches
1	m_1	S	S	$ß_1 S$
2	m_2	S - 1	$ß_1 S$ (S - 1)	$ß_1 ß_2 S$ (S - 1)
3	m_3	S - 2	$ß_1 ß_2 S$ (S - 1) (S - 2)	$ß_1 ß_2 S$ (S - 1)
4	m_4	S - 3	$ß_1 ß_2 S$ (S - 1) (S - 3)	$ß_1 ß_2 S$ (S - 1)
• • •				
M	m_M	S - (M-1)	$ß_1 ß_2 S$ (S - 1) (S - (M-1))	$ß_1 ß_2 S$ (S - 1)

Total (for $ß_1 = ß_2$): $\sim ß^2 S^3 M + ßS^2 + S$

Table I-2:

Number of comparisons in a sequential model matching process. The number of comparisons at each step is the product of the number of valid interpretations at the previous step and the number of remaining scene features. The number of successful matches is a fraction β of these comparisons for the first two steps; for later steps, only one scene feature is successfully matched for each previous interpretation.

---

the cost to apply all object models is

$$\textit{Cost of matching} \approx N\,d\,(\alpha P + 10^{-2}(\alpha P)^2 + 20\times 10^{-4}(\alpha P)^3) = 34P + 3.4\times 10^{-3} P^2 + 6.8\times 10^{-6} P^3 \quad (3)$$

where N is the number of objects (about 34 from Table I-1) and $d \approx 100$ is the cost of one feature comparison.

Secondary matching. As we mentioned above, after signs and signals are found the perception system must examine them in more detail to identify their contents. This involves computation similar to that described above, only for a more limited set of features (color vision only) and for only the pixels in the sign objects just found. P' for the signs found will be several orders of magnitude smaller than P for the entire scene in the expression above, so this recognition step does not add any significant cost to the overall process.

Searching for markings. The search for road markings was delayed until after road regions were found, but now the cost of finding markings must be included. In this case, unlike the case of signs and signals above, the number of pixels in the secondary search is comparable to the number in the entire scene. This difference is due to the fact that the markings are horizontal and can be greatly foreshortened at long ranges. The markings can therefore appear to be very small and require many small pixels to see (see Figure 2-2). The feature size in Table I-1 is 5cm, which requires 36 times as many pixels as are needed for the road (30cm feature); scenes from our simulated scenarios had about 3.5% of the image covered by road regions, on average. Thus we estimate that approximately the same number of pixels are used to search for markings. We assume that feature extraction would be simpler than before because the "objects" are all flat markings on the ground plane; thus

$$\textit{Cost of extracting marking features} \approx 10^3 P + 10^{-1} P^2 \tag{4}$$

in contrast to Equation 2. The cost of matching models is calculated using Equation 3, except that $N \approx 50$ from Table I-1:

$$\textit{Cost of matching markings} \approx 50\,P + 5\times10^{-3} P^2 + 10^{-5} P^3. \tag{5}$$

The total cost of perception is then the sum of the costs in Equations 2 through 5:

$$\begin{aligned}\textit{Total Cost} \approx\ & \textit{Cost of featureextraction} + \\ & \textit{Cost of matching} + \\ & \textit{Cost of extracting marking features} + \\ & \textit{Cost of matching markings} \\ \approx\ & 1.1\times10^4 P + 1.1\,P^2 + 1.7\times10^{-5} P^3.\end{aligned} \tag{6}$$

Pixel count. The above analysis indicates that in a scene with a fairly uniform distribution of features, the cost of perception is dependent on the number of pixels. The number of pixels is in turn dependent on the angle subtended by each pixel and the size of the field of view.

The angle subtended by a pixel depends on the size of the objects the system needs to see. Without any knowledge to constrain where objects might be, a naive perception system is forced to look for everything of potential interest in all parts of the scene. From Table I-1 we see that the smallest required feature size is 1.5cm, for finding small signs. We require that objects such as signs be recognizable even when turned away from the line of sign by $45°$; thus the signs require pixels $1.5\text{cm} \times \cos(45°) \approx 1.1\text{cm}$. This is for a "vertical" object; we must also consider roads, because they are horizontal and potentially foreshortened to a small size. While the 1.1cm feature covers 4.2×10^{-3} degrees at a range of 150m, a 30cm road feature covers only 1.5×10^{-3} degrees at 150m. Therefore, we assume that each pixel subtends 1.5×10^{-3} degrees.

We assume that the field of view extends $45°$ above and below the horizon, which allows the robot to see lines and other markings on the road within 2m of the robot, and overhead signs within several meters. The range in azimuth is the entire $360°$. Thus, if the perception system uses the same resolution over the entire scene, the number of pixels will be given by

$$P = \frac{\textit{Azimuth variation}}{\textit{pixel angle}} \times \frac{\textit{Elevation variation}}{\textit{pixel angle}} = \frac{360}{1.5\times10^{-3}} \times \frac{90}{1.5\times10^{-3}} \approx 1.4\times10^{10}.$$

The minimum size of pixels can be increased if the range to an object is less than the maximum of 150m. We assume a flat world, and therefore can guarantee a reduced range at some elevations because of the ground plane and an assumed ceiling on traffic objects. The ceiling is determined by signs, which can be up to 7m above the road. Figure I-2 shows how the range is limited by a floor and ceiling, and how the apparent size of an object increases when it is closer. For the lowest row of pixels in the sensor image, 45 degrees down in elevation, the pixel size corresponding to 30cm on the ground is $4°$. Pixels contintue to get smaller at longer ranges until at a range of 150m, 30cm on the road covers only 1.5×10^{-3} degrees. For signs, 1.1cm features vary in angle from 8.9×10^{-2} degrees to 4.2×10^{-3} degrees. Our perceptual routines take advantage of these relations to keep the resolution as low as possible. In order to keep our estimate of naive perception conservatively low, we will also apply a variable-resolution sensing technique to the pixel estimate above. We assume that sign features determine the resolution in most of the scene, except at long ranges and depressed elevations where road features appear smaller. When calculated this way, the scene requires about $P = 8.1{\times}10^7$ pixels.

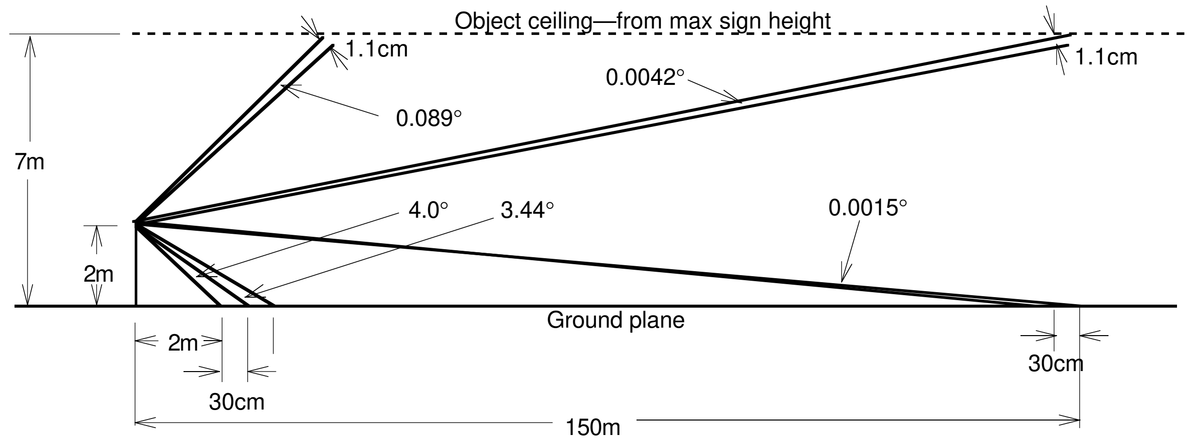


Figure I-2: An upper and lower limit on the location of objects limits the maximum range and increases the minimum pixel size needed. 1.05cm object is for a vertical sign; 30cm horizontal object is for a road.

It is possible to reduce the pixel count further by limiting the search for some objects to, say, the hemisphere in front of the robot. However, such gross constraints can only reduce the search cost by a small factor without using specific task knowledge.

Net cost. Using the reduced pixel count and Equation 6, we calculate the total cost to be

$Cost \approx 8.9{\times}10^{11} + 7.2{\times}10^{15} + 9.0{\times}10^{18} \approx 9.0{\times}10^{18}$.

II. The Cost of Perceptual Routines

Cost Equations

In this appendix we describe how we calculate the cost of each routine. In general we use the same assumptions about extracting features and matching models as we did for the naive model. The cost is again calculated using a polynomial in the number of pixels:

$$Cost = aP + b\alpha P^2 + Nc(\alpha P)^3,$$

where αP is the number of features in an image with P pixels, and N is the number of object models. For the naive model we used

$\alpha \approx 10^{-2}$,
$a \approx 10^4$,
$b \approx 10^2$,
$c \approx 0.2$
and
$N \approx 34$

for primary feature extraction (Equation 2) and model matching (Equation 3). For the routines we generally use these same values. When the routines are more limited, we modify the parameters. For example, routines that search only for features on the road use $a \approx 10^3$ and $b \approx 10$ to reflect the reduced complexity of processing two-dimensional features in a plane. These are the same values used for extracting road marking features in Equation 4. The equation for the naive model also had terms for secondary model matching and finding road markings; these costs are also included in routines where appropriate.

The number of pixels P in the above equation depends on the geometry of the area in the world scanned by the routine. Thus the cost of each routine varies with each call. For the routines that define the corridor—track-lane, find-intersection-path, etc.—we assume that the perception system can track the road or lane in the image. That is, from a start marker where the road is identified, the routine looks in the adjacent area for the continuation of the road, and then in the area next to that, etc. Such incremental procedures are frequently used in other machine vision tasks, such as finding roads in aerial photographs [25]. Under this assumption, the routine only searches an area slightly larger than the road or lane. Routines that search for objects such as signs or signals use a road or lane as a reference, and so can compute the entire region in the world in which to search.

Routines generally search regions that are over or next to a lane. The regions thus tend to resemble long, narrow "boxes" as in Figure II-1, with a width and length corresponding to a lane, and a height corresponding to the maximum height of the objects above the ground. The number of pixels required to cover the region depends on whether the object of interest has height (the "vertical" objects in Appendix I), or is entirely in the ground plane. In the former case, we count the number of steps necessary to sweep the entire far side of the box in one horizontal plane, as shown in Figure II-2. The number of vertical steps is approximated by dividing the region height by the object feature size. The total number of pixels is the product of horizontal and vertical steps.

For flat objects in the ground plane, the routines scan the the entire search area on the ground. The angle subtended by each pixel is determined by the size of foreshortened object features. For example, in tracking a lane we assume that the routine must detect both longitudinal lane lines and transverse stop and crosswalk lines. As Figure II-3 shows, either type of line may be foreshortened and limit the pixel angle, depending on the relative orientation of the road. The total number of

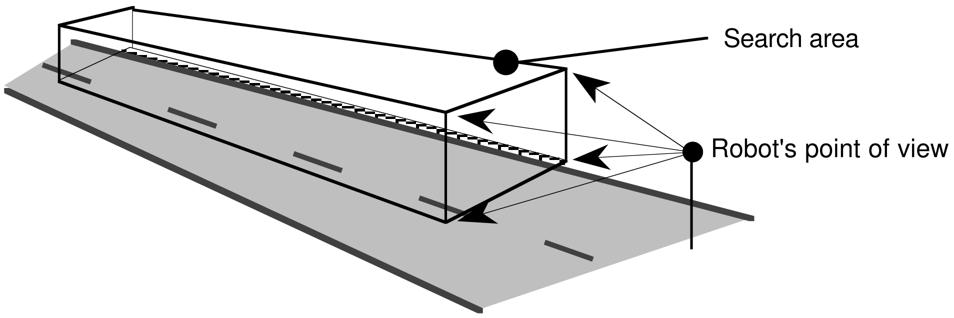


Figure II-1: When routines search within a fixed height above a lane, the resulting region is a long box.

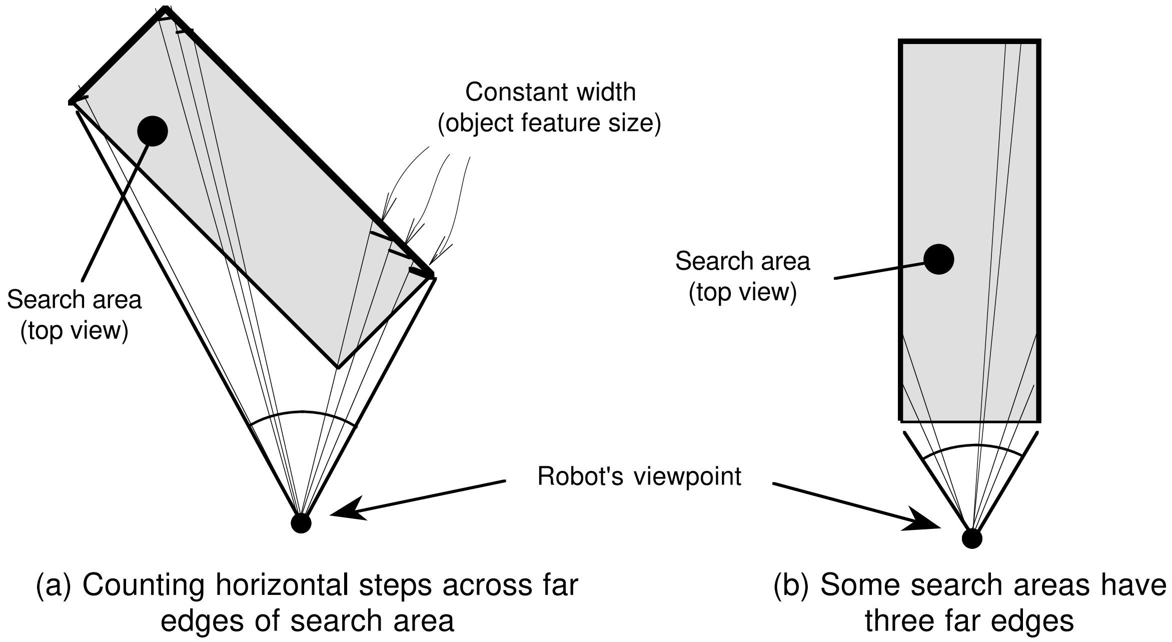


Figure II-2:

Counting horizontal steps across a search area. To see a given object feature, the routine must use small enough angular steps to see the feature even at the back of the area. For most areas (a), this involves stepping across the far two sides; for areas directly ahead of the robot (b), there are three far sides.

pixels that it takes to cover the search area is the product of the number of transverse steps and the number of longitudinal steps.

For our analysis of naive perception we used the smallest resolution size from Table I-1 to determine pixel angle. Since the routines all look for different objects, they use the size appropriate

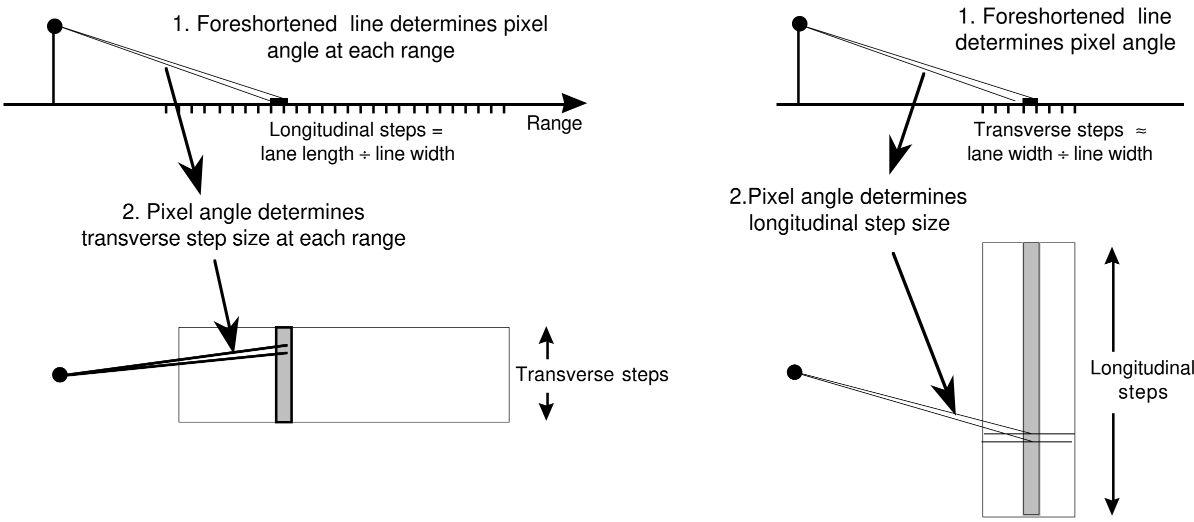


Figure II-3:

Counting the number of pixels in a search area in the ground plane. Lines in the road nearly perpendicular to the line of sight are foreshortened and determine how small the pixel angle is at each range. The total number of pixels is the product of the number of transverse and longitudinal angular steps.

---

for their object. Similarly, the number of model variations N depends on the particular object. The values of these parameters are taken from Table I-1. As with the naive case, we assume that signs and signals must be identified even if turned away by 45°, so the sizes given in Table I-1 are reduced by cos (45°) ≈ 0.7.

Individual Routine Costs

Each of the routines uses the same cost equation with the same parameter values as the naive case except where noted.

Find current lane. This routine finds and marks the lane directly in front of the robot. It is always the first routine that Ulysses uses. From the lane line information, this routine is able to tell if the robot is between two lanes, and what angular offset there is between the robot and the lane. The robot must scan an area about 4m wide by 3m deep in front of the robot with 5cm resolution. The routine counts pixels in the ground plane, then computes the cost using $a = 10^3$, $b = 10$, and $N = 6$. The number of pixels is constant for this routine at 7580, and the cost is always 1.33×10^7 operations.

Mark adjacent lane. This routine looks for a lane adjacent to a given marker. It also searches an area about 4m by 3m for lane lines. The cost parameters are the same as above: $a = 10^3$, $b = 10$, and $N = 6$. The routine cost is thus

$$\textit{Routine cost} = 10^3 P + 10^{-1} P^2 + 1.2 \times 10^{-6} P^3.$$

Track lane. This routine starts at a marker in a lane and traces the lane as far as possible. The search area is about 2m wider than the lane. The cost for track-lane is also computed using Equation .

Profile road. This routine scans transversely across the road at a marker to find all of the lanes in one direction. It reports the types of the lane lines and the relative position of the marked lane. The size of the scan area depends on the width of the road. As with the other lane-searching routines, this routine's cost is computed using Equation .

Find intersection roads. This routine finds all of the approach roads and lanes at a marked intersection. The assumed procedure is to first find intersecting *roads* using 30cm resolution, and then identify lanes at the intersection using 5cm resolution. The search area for the first phase is a region somewhat larger than the intersection; for the second it is the last 3m of each approach road. The parameters for the first search are the almost the same as for the lane searches above, except that $N = 1$. For the second phase the parameters are the same as for other lane searches ($a = 10^3$, $b = 10$, and $N = 6$). The cost is thus computed with

$$\begin{aligned} \textit{Routine cost} &= 10^3 P_1 + 10^{-1} P_1^{\,2} + 2{\times}10^{-7} P_1^{\,3} + \\ &\quad \textit{roads} \times [10^3 P_2 + 10^{-1} P_2^{\,2} + 1.2{\times}10^{-6} P_2^{\,3}] \end{aligned}$$

Find path in intersection. This routine is similar to finding intersection roads, except that it finds a departing lane in one particular direction and creates a path for the robot through the intersection. The cost is calculated using the same equation, . The number of pixels involved is smaller in the second phase of the routine because only one road is being analyzed.

Find next lane marking. This routine scans down a given lane looking for markings such as crosswalks, turn arrows, etc. The cost is just the same as tracking a lane, except that the scan area is limited to the confines of the lane, and the number of model variations $N = 44$. The cost is thus computed with

$$\textit{Routine cost} = 10^3 P + 10^{-1} P^2 + 8.8{\times}10^{-6} P^3.$$

Find next car in lane. This routine scans down a given lane looking for a car. The routine doesn't have to find the lane again, so does not have to search the ground plane for lane markings. As in the naive perception case, the routine is assumed to search with low resolution first, and then search within the object for details. The low resolution search covers a region 2.5m high with 10cm pixels and uses $N = 15$. At high resolution the routine scans the area of a car (assumed to be 2.5m by 7m) with 3cm pixels and uses $N = 1$. The total cost is thus

$$\begin{aligned} \textit{Routine cost} &= 10^3 P_1 + 10^{-1} P_1^{\,2} + 3{\times}10^{-6} P_1^{\,3} + \\ &\quad \textit{car?} \times [10^3 P_2 + 10^{-1} P_2^{\,2} + 2{\times}10^{-7} P_2^{\,3}] \end{aligned}$$

Find crossing cars. This routine scans a marked intersection to see if there are any cars on or approaching the robot's path. The search area is limited to the intersection, and does not include approach roads. The routine searches for cars as in find-next-car-in-lane, but does not bother with the higher resolution scan of the cars found.

Find next sign. This routine searches for signs in the area to the right of the road. Because the search area is referenced to the road and not a particular lane, the routine must track the road edge. Thus the cost includes the cost of finding ground plane features, the cost of finding sign objects at low resolution, and the cost of reading the signs (secondary matching). The road search portion uses the lane tracking parameters above, except that $N = 1$ and the feature size is 30cm. The low resolution sign search covers a region about 4m wide and about 7m high along the right edge of the road. The feature size is $1.5cm \times \cos(45) \approx 1.1cm$, and $N = 8$. Reading the signs at high resolution requires seeing object features as small as $0.7cm \times \cos(45) \approx 0.5cm$, and matching $N = 50$ object models. We assume that an average sign is 91.4cm (36") on a side. This higher resolution search must examine every candidate sign that the low resolution search finds. We assume that there is a potential sign every 5m along the road. In our scenarios actual signs are spaced no closer than 30m; however, cost is calculated as if the routine examines every potential sign. The total cost is computed with

$$\begin{aligned} \textit{Routine cost} = {} & 10^3 P_1 + 10^{-1} P_1^{\,2} + 2\times10^{-7} P_1^{\,3} + \\ & 10^3 P_2 + 10^{-1} P_2^{\,2} + 1.6\times10^{-6} P_2^{\,3} + \\ & \textit{candidate signs} \times [10^3 P_2 + 10^{-1} P_2^{\,2} + 10^{-5} P_2^{\,3}]. \end{aligned}$$

Find next overhead sign. This routine is similar to find-next-sign, except that the robot looks in the region above a given lane. Thus there is no road-finding cost. Overhead signs are assumed to be at least 2m off the ground, and no higher than 7m. In addition, we assume that the low resolution search finds potential signs every 20m instead of every 5m. We expect there to be fewer objects extending over a lane than there are erected beside the road. The equation for computing the routine cost is

$$\begin{aligned} \textit{Routine cost} = {} & 10^3 P_2 + 10^{-1} P_2^{\,2} + 1.6\times10^{-6} P_2^{\,3} + \\ & \textit{candidate signs} \times [10^3 P_2 + 10^{-1} P_2^{\,2} + 10^{-5} P_2^{\,3}]. \end{aligned}$$

Find back-facing signs. This routine looks for STOP and YIELD signs along the left side of the road, facing opposing traffic. The robot only looks about 25m up the road before giving up. The cost of the routine includes a road-edge tracking cost, as with find-next-sign, and a low resolution sign search cost. A higher resolution scan to read signs is not necessary since they are facing away from the robot. For the road, the feature size is again 30cm and $N = 1$. When searching for the signs, the robot only needs to look for two shapes—a triangle or an octagon—so $N = 2$. STOP and YIELD signs are at least 76cm $\times$ 76cm, so the feature size is $7.6cm \times \cos(45°) = 5.4cm$. The equation for computing the cost is thus

$$\begin{aligned} \textit{Routine cost} = {} & 10^3 P_1 + 10^{-1} P_1^{\,2} + 2\times10^{-7} P_1^{\,3} + \\ & 10^3 P_2 + 10^{-1} P_2^{\,2} + 4\times10^{-7} P_2^{\,3} \end{aligned}$$

Find signal. This routine scans a marked intersection to find signal heads facing the robot. The search region covers the whole area of the intersection because signals can be on the near or far side, and on the left or right. The routine first finds the intersection area using 30cm features and parameter values $a = 10^3$, $b = 10$, and $N = 1$. Next there is a low-resolution search for signal heads using $2cm \times \cos(45°) = 1.4cm$ features, and $N = 10$. Finally, the lenses found in the low-resolution search are examined at higher resolution to detect arrows. This process uses 0.7cm features and $N = 5$. We assume that there are three lenses on each signal head at the intersection, and each is examined to determine what symbols it displays. Each lens is assumed to be 30cm in diameter. The total cost of the routine is given by

$$\begin{aligned} \textit{Routine cost} = {} & 10^3 P_1 + 10^{-1} P_1^{\,2} + 2\times10^{-7} P_1^{\,3} + \\ & 10^3 P_2 + 10^{-1} P_2^{\,2} + 2\times10^{-6} P_2^{\,3} + \\ & \textit{lenses} \times [10^3 P_2 + 10^{-1} P_2^{\,2} + 10^{-6} P_2^{\,3}]. \end{aligned}$$

Marker distance. This routine calculates the (distance in the world, not an image) between two markers. It is used primarily to estimate the distance of another car to the intersection it is approaching. We assume that the perception system records world coordinates of each marker it creates, so this "routine" does not involve any perception at all. The cost is therefore zero.

III. Routine Costs in the Left Side Road Scenario

This appendix shows in more detail the cost of performing various perceptual routines in a specific situation. The situation we illustrate is a moment in the left side-road scenario, shown again in Figure III-1. This is a point in time between notes (4) and (5) in Figure 4-15 The use of routines here was already described by Figures 4-3 through 4-8 in Section 4; the planner uses (only) a model of the driving corridor to select perceptual targets. We repeat those figures here and list the estimated cost (in operations) of each perceptual action.

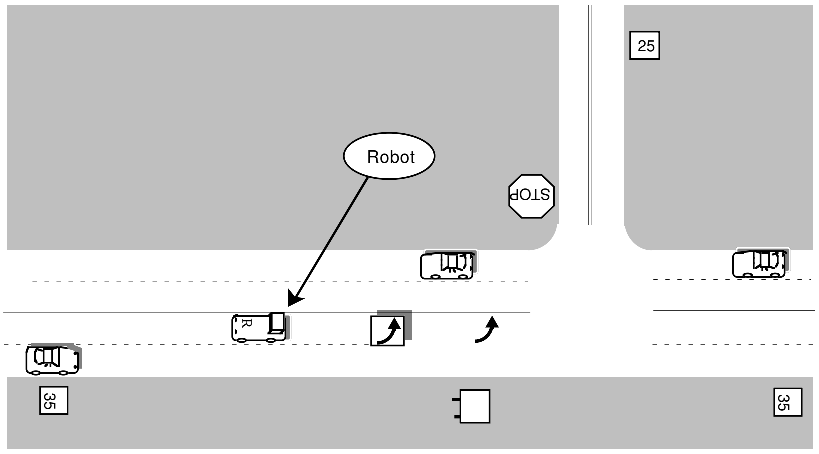
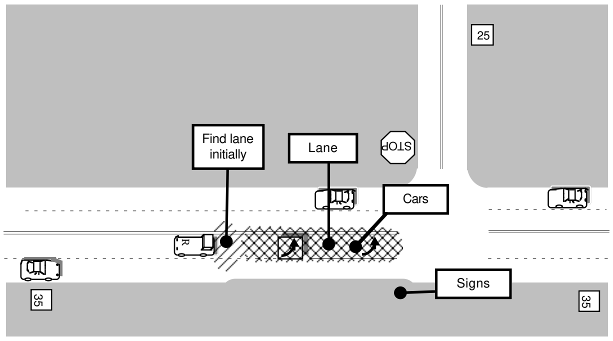


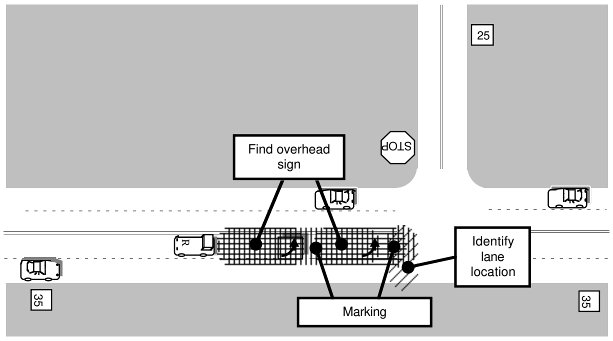
Figure III-1: The moment of the left side-road scenario analyzed in this appendix (not to scale).

Most actions correspond to single perceptual routine calls, except where noted. Multiple routine calls are made when the first scan terminates on an object, requiring another call to scan the rest of the corridor.

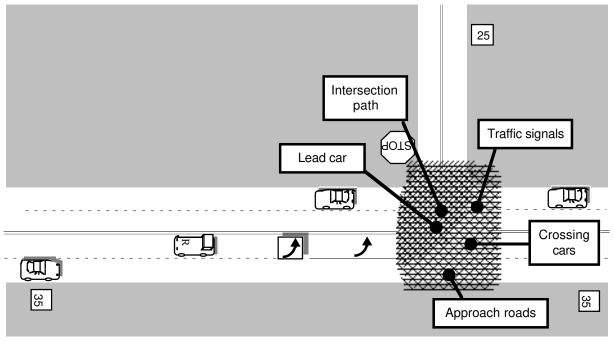
Several costly actions have be highlighted in bold face. These actions dominate the perceptual cost at this moment in the scenario. The actions all involve tracking lanes—i.e., looking for lanes bounded by narrow lines. The search for lanes is expensive in several cases because the lanes extend all of the way to the range limit of the sensors (150m), where very high resolution is needed to resolve the lines.



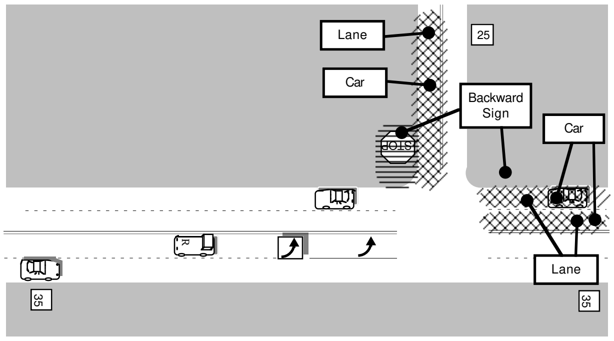
Find lane initially: 1.3×10^{7}
Track lane: 5.0×10^{12}
Look for car: 6.6×10^{7}
Look for signs (2 scans): 1.1×10^{13}



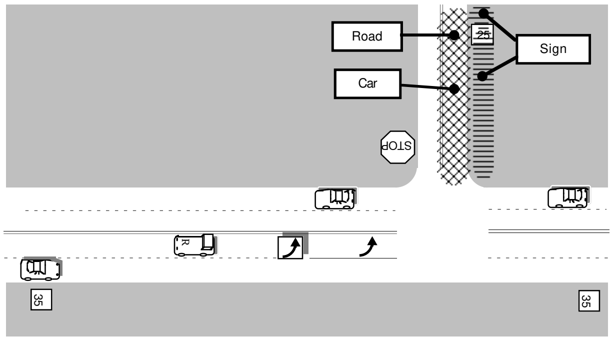
Find lane location: $5.0{\times}10^{12}$
Find overhead signs (2 scans): $1.1{\times}10^{12}$
Find markings (2 scans): $4.6{\times}10^{12}$



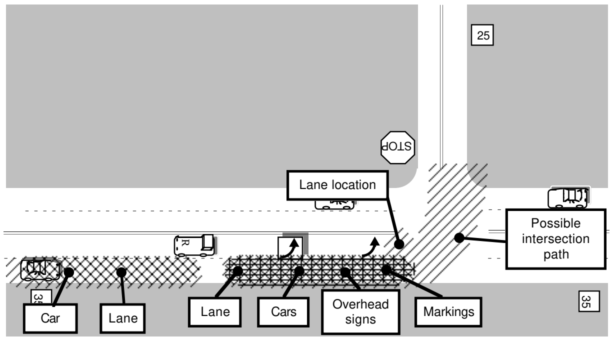
Find path in intersection: $2.2{\times}10^{8}$
Look for crossing cars: $1.9{\times}10^{8}$
Look for signals: $2.8{\times}10^{12}$
Find approach roads: $1.7{\times}10^{12}$
Look for lead car: $6.4{\times}10^{8}$



Find back-facing signs (2 approaches): $3.7{\times}10^{8}$
Track lanes upstream (3 lanes): $1.8{\times}10^{15}$
Look for approaching cars (in 3 lanes): $1.3{\times}10^{9}$



Track lane: $4.7{\times}10^{14}$
Look for cars: $6.6{\times}10^{8}$
Find speed limit signs (2 scans): $9.5{\times}10^{13}$



Find and track adjacent lane: $4.8{\times}10^{12}$
Look for cars ahead: $7.4{\times}10^{7}$
Find lane location at intersection: $5.0{\times}10^{12}$
Look for signs above lane: $1.9{\times}10^{12}$
Look for markings in lane: $1.0{\times}10^{13}$
Look for intersection path: $9.3{\times}10^{10}$
Track lane upstream: $2.0{\times}10^{15}$
Find car upstream: $6.6{\times}10^{8}$

References

[1] AASHTO.
A Policy on Geometric Design of Highways and Streets
American Association of State Highway and Transportation Officials, Washington, D.C., 1984.

[2] Agre, P. E. and D. Chapman.
Pengi: An Implementation of a Theory of Activity.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268-272. Morgan Kaufman Publishers, Los Altos, 1987.

[3] Akatsuka, H. and I. Shinichiro.
Road Signposts Recognition System.
In *Proceedings of the SAE International Congress*. SAE, Detroit, 1987.

[4] Aloimonos, J., I. Weiss and A. Bandyopadhyay.
Active Vision.
International Journal of Computer Vision 1(4):333 - 356, 1988.

[5] Ballard, D. H.
Animate Vision.
Artificial Intelligence 48:57-86, 1991.

[6] Besl, P. J. and R. C. Jain.
Three-dimensional Object Recognition.
ACM Computing Surveys 17(1), March, 1985.

[7] Binford, T. O.
Survey of Model-Based Image Analysis Systems.
International Journal of Robotics Research 1(1), 1982.

[8] Brown, C.
Gaze Controls with Interactions and Delays.
IEEE Transactions on Systems, Man, and Cybernetics 20(1):318-327, March/April, 1990.

[9] Burt, P., J. Bergen, R. Hingorani, S. Peleg, and P. Anandan.
Dynamic Analysis of Image Motion for Vehicle Guidance.
In O. Kaynak (editor), *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, pages 75-82. IEEE, August, 1990.

[10] Crisman, J. and C. Thorpe.
Color Vision for Road Following.
In *Proceedings of the SPIE Conference on Mobile Robots*. SPIE, 1988.

[11] Ettinger, G. J.
Large Hierarchical Object Recognition Using Libraries of Parametrized Sub-parts.
In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 32 - 41. IEEE, Ann Arbor, Michigan, June, 1988.

[12] FHWA.
Manual on Uniform Traffic Control Devices
Federal Highway Administration, U.S. Department of Transportation , Washington, D.C., 1978.

[13] Firby, R. J.
Task Directed Sensing.
In *Vol. 1198, Sensor Fusion II: Human and Machine Strategies*, pages 480-489. SPIE, 1989.

[14] Fujimori, T. and T. Kanade.
Knowledge-Based Interpretation of Outdoor Road Scenes.
In C. Thorpe and T. Kanade (editors), *CMU-RI-TR-88-4: 1987 Year End Report for Road Following at Carnegie Mellon*, pages 45-96. Carnegie Mellon University, 1988.

[15] Grimson, W.E.L.
The Combinatorics of Object Recognition in Cluttered Environments Using Constrained Search.
Artificial Intelligence 44:121-165, 1990.

[16] Griswold, N.C. and B. Bergenback.
Stop Sign Recognition for Autonomous Land Vehicles (ALVs) Using Morphological Filters and Log-Conformal Mapping.
1989.
Texas A & M University, College Station, Texas.

[17] Hanson, A. and E. Riseman.
Segmentation of Natural Scenes.
Computer Vision Systems.
In A. Hanson and E. Riseman,
Academic Press, New York, 1978, pages 129 - 163.

[18] Hebert, M. and T. Kanade.
3-D Vision for Outdoor Navigation by an Autonomous Vehicle.
In C. Thorpe and T. Kanade (editors), *CMU-RI-TR-88-4: 1987 Year End Report for Road Following at Carnegie Mellon*, pages 29-41. Carnegie Mellon University, 1988.

[19] Hutchinson, S. and A. Kak.
Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities.
IEEE Transactions on Robotics and Automation 5(6), 1989.

[20] Huttenlocher, D. and S. Ullman.
Recognizing Solid Objects by Alignment with an Image.
International Journal of Computer Vision 5(2):195-212, 1990.

[21] Ikeuchi, K. and M. Hebert.
Task Oriented Vision.
Proceedings of the DARPA 1990 Image Understanding Workshop :497 - 507, 1990.

[22] Kluge, K. and H. Kuga.
Car Recognition for the CMU Navlab.
In C. Thorpe and T. Kanade (editors), *CMU-RI-TR-88-4: 1987 Year End Report for Road Following at Carnegie Mellon*, pages 99-115. Carnegie Mellon University, 1988.

[23] Kluge, K. and C. Thorpe.
Explicit Models for Road Following.
In *Proceedings of the IEEE Conference on Robotics and Automation*. IEEE, 1989.

[24] Legislative Reference Bureau.
Pennsylvania Consolidated Statutes, Title 75: Vehicles (Vehicle Code)
Commonwealth of Pennsylvania, Harrisburg, PA, 1987.

[25] McKeown, D. M. Jr. and J. L. Denlinger.
Cooperative Methods For Road Tracking in Aerial Imagery.
In *Proceedings of the 1988 DARPA IUS Workshop*, pages 327-341. Morgan Kaufmann, 1988.

[26] McKnight, J. and B. Adams.
Driver Education and Task Analysis Volume I: Task Descriptions.
Final Report, Department of Transportation, National Highway Safety Bureau, Washington, D.C., November, 1970.

[27] Michon, J. A.
A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?
In L. Evans and R. Schwing (editors), *Human Behavior and Traffic Safety*. Plenum, 1985.

[28] Reece, D. A. and S. Shafer.
An Overview of the Pharos Traffic Simulator.
In J. A. Rothengatter and R. A. deBruin (editors), *Road User Behaviour: Theory and Practice*. Van Gorcum, Assen, 1988.

[29] Reece, D. and S. Shafer.
A Computational Model of Driving for Autonomous Vehicles.
Technical Report CMU-CS-91-122, Carnegie Mellon University, April, 1991.

[30] Solder, U. and V. Graefe.
Object Detection in Real Time.
In *Proceedings of the SPIE Symposium on Advances in Intelligent Systems*, pages 121-165. SPIE, November, 1990.

[31] Tsotsos, J. K.
A 'Complexity Level' Analysis of Immediate Vision.
International Journal of Computer Vision 1(4):303 - 320, 1987.

[32] Ullman, S.
Visual Routines.
Cognition 18:97-160, 1984.

Table of Contents

List of Figures

List of Tables