CMU Sidewalk Navigation System:

A Blackboard-Based Outdoor Navigation System

Using Sensor Fusion with Colored-Range Images

Y. Goto, K. Matsuzaki
I. Kweon, T. Obatake
Robotics Institute, Carnegie-Mellon University
Pittsburg, PA. 15213

Abstract

We describe the CMU Sidewalk Navigation System, which can drive a vehicle in the outdoor environment of the CMU campus. The system includes all modules necessary for outdoor navigation -- modules for route planning, local path planning, vehicle driving, perception, and map data. The perception module uses sensor fusion with color and rage data to analyze complex outdoor scenes accurately and efficiently.

1. Introduction

The goal of the CMU SCVision group is to create an autonomous mobile robot system capable of operating in outdoor environments.[1] The complexity of the environment requires the system to have a powerful perception ability, capable of analyzing natural objects, and a planning ability which can work in non-uniform conditions. Because this navigation system will be very large, we need mechanisms to combine programs into whole systems and mechanisms for parallelism in computation.

We already have several systems towards the goal: a road following system with color classification [5], road network navigation with a simple map [1], scene analysis with a laser range sensor [2], and the blackboard [4].

The CMU Sidewalk Navigation System is a milestone system toward our goal. In this system, we focus on two points. The first is to build a whole system based on a good system architecture so that the system is both complete (containing every necessary module) and efficient. We achieve that goal by adopting a blackboard-based architecture. The second point is to create perception modules with sensor fusion that work well in our outdoor environment.

The test site for the CMU Sidewalk Navigation System is the CMU campus, containing a network of sidewalks and intersections, along with grass, slopes, and stairs. The system can drive the vehicle through these objects to get to its destination.

[1]This research was sponsored by the Defense Advanced Research Projects Agency, DOD, through ARPA Order No. 5351, and monitored by the U. S. Army Engineer Topographic Laboratories under contract DACA 76-85-C-0003. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

2. System Architecture for the Outdoor Navigation System

2.1. Hardware Configuration

The hardware for the CMU sidewalk navigation system consists of three SUN-3 workstations, the vehicle, the color TV camera, and the laser range sensor. The workstations are linked together with Ethernet, and the workstations and the vehicle are linked with radio communication. Figure 1 shows the vehicle called *Terregator*.

Figure 1: Terregator

2.2. System Architecture

2.2.1. Stages of Navigation

In order to create a reasonable system architecture, we have to start by analyzing outdoor navigation.

If the navigation system uses only one uniform navigation mode, the system architecture issue is not essential. But, in general, outdoor navigation includes several navigation modes. The example shown in Figure 2 illustrates this situation. The vehicle running from the starting point to the destination has to *follow the road*, *turn at the intersection*, *climbing the slope* and *cross the terrain*. *Turning at the intersection* needs a more complex method to drive the vehicle than *following the road*. Perception for *crossing the terrain* is different from perception for *turning at the intersection*. In *following the road* we can use assumption that the ground is flat, which makes perception easier. But *climbing the slope* does not satisfy this assumption. This is one reason why the

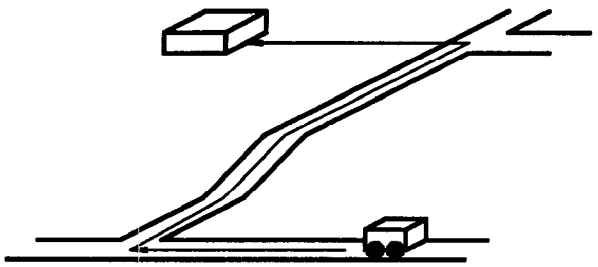


Figure 2: Outdoor Navigation

outdoor navigation system needs good system architecture.

We decompose navigation into two processing stages. The first stage is the *route planning stage*, and the second stage is *local navigation*. In the route planning stage the system selects the best navigation route, from several possible routes to get to the destination from the starting point. The system divides the whole route into a sequence of *route segments*. In each route segment, objects on which the vehicle can run are constant. The navigation system can drive the vehicle using a single uniform driving mode, for example, *following the road* or *turning at the intersection*, and a single perception mode. In this stage, using the map data is essential.

Local navigation is navigation within one route segment. In the local navigation stage, the navigation mode is constant and the main task is to drive the vehicle along the route segment. Local navigation uses perception to find a safe passage for the vehicle, and to determine the actual vehicle driving path.

In contrast, our earlier and simpler navigation system did not have the route planning capability and has only one navigation mode for local navigation.

The system architecture of the CMU sidewalk navigation system is indicated in Figure 3. We decomposed the whole system into several modules. The modules indicated with blocks are separate processes, running independently, and communicating with each other through the BLACKBOARD. In selecting this decomposition of the whole system into these modules, we followed the principle of *information hiding*. The CAPTAIN module and the MAP NAVIGATOR module are responsible for the route planning, and they do not know the result of perception or how to drive the physical vehicle. The PILOT module, the PERCEPTION module and the HELM module are responsible for the local navigation, and they do not know the destination, the whole route, or the sequence of route segments. What they know is limited to only one route segment at one time. We will explain the system architecture in detail in the following sections.

2.2.2. The Blackboard-Based Architecture

Our BLACKBOARD provides modules with communication and synchronization facilities [4]. Using a blackboard-based architecture brings two main advantages to building our navigation system.

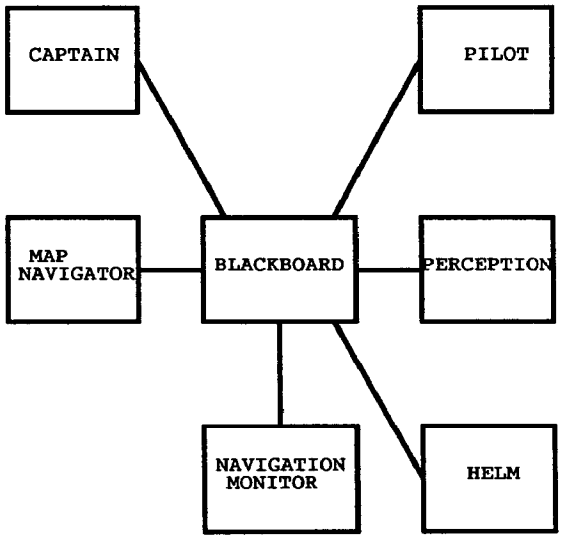


Figure 3: System Architecture

The first advantage is parallelism in execution. We decompose the whole system into several parallel modules. Because the most time consuming operation is perception, it is an independent process, the PERCEPTION module, running on its own machine, and not disturbing other modules. Because the HELM module which drives the physical vehicle needs real-time response, it is another separate process. Communication and synchronization of all modules are handled by mechanisms of the BLACKBOARD.

The second advantage is that using a blackboard makes it easier to combine several programs into a whole system. Our BLACKBOARD provides a good mechanism to connect modules, and limits the interactions among modules. For instance, each module can work in its most natural and convenient coordinate frame, with the BLACKBOARD converting among reference frames. We use the principle of *information hiding* so that the interfaces between modules are small. This keeps communication costs low and allows good modularity. The details of the BLACKBOARD are explained in following sections.

2.3. Module Structure

In this subsection we explain each module.

2.3.1. The CAPTAIN Module and the Mission

At the upper level of the system is the *CAPTAIN* module that receives instructions from the controlling person and oversees the mission. The mission consists of a number of steps, and the CAPTAIN sequences through the steps. For each step, there is a *destination* that tells where to go and one or more *constraints* that tell how to go. For example, "go to intersection D" gives a destination and "keep right" gives a constraint. Each mission step also has a *trigger* condition and an action which will be executed if the trigger condition is satisfied. Triggers can be used to move on to the next mission step when one step is completed.

The CAPTAIN sends the destination and the constraints of each mission step to the MAP NAVIGATOR one step at a time, and gets the result of mission step, *success* or *fail*, from the MAP NAVIGATOR.

2.3.2. The MAP NAVIGATOR and Route Planning

The MAP NAVIGATOR does the route planning based on a destination and a constraint sent from the CAPTAIN, gives the PILOT directions for driving along the route, and reports the result of the mission to the CAPTAIN.

The MAP NAVIGATOR contains two main parts, the ROUTE SELECTOR, and the ROUTE SEGMENT DESCRIBER (see Figure 4). The ROUTE SELECTOR creates the route plan, and decomposes it to a sequence of the route segments so that each route segment has only one *navigation mode*. The current system has several navigation modes: *follow-road*, *turn-at-intersection*, *go-through-intersection*, and *go-through-slope*. Our future system will have another navigation mode, *cross-country*, in order to navigate on open terrain.

The ROUTE SEGMENT DESCRIBER generates the description of the route segment. The purpose of *route segment description* is to provide the PILOT with the information necessary for navigation within the route segment. It includes path objects (e.g., pieces of road, intersections), navigation modes, the conditions to exit from the route segment, the constraints to drive the vehicle, and object descriptions. *Path objects* are the objects on which the vehicle should run. *Object descriptions* describe the location and the shape of the objects (such as landmarks) which the PERCEPTION module can see while running on the route segment. This description is created by copying a part of the Map data, and is used as a prediction for the PERCEPTION module. One important point is that only the MAP NAVIGATOR maintains the Map data.

The route segment description is sent to the BLACKBOARD and forwarded to the PILOT. When the PILOT finishes the route segment, it reports the result. If the result is *success*, the ROUTE SEGMENT DESCRIBER sends next route segment description.

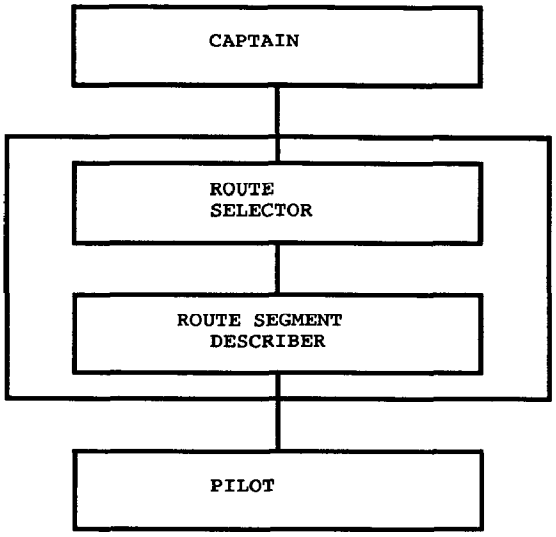


Figure 4: The MAP NAVIGATOR Module

2.3.3. The PILOT and Local Path Planning

The PILOT, the PERCEPTION and the HELM work together for local navigation. The PILOT operates continuously to conduct the navigation within the route segment. The PILOT contains several sub-modules that form a sequence as shown in Figure 5, to process each area to be traversed.

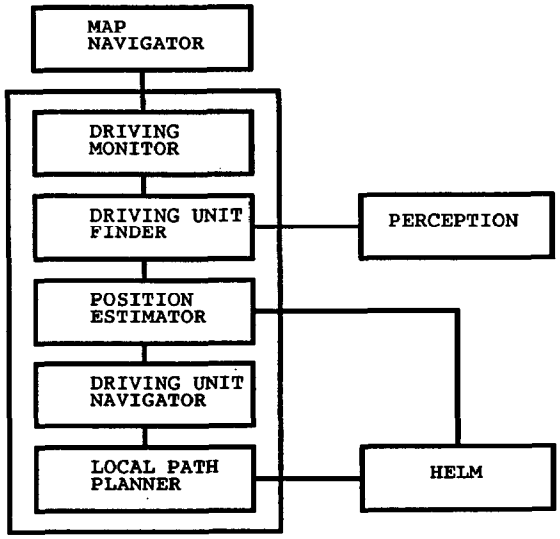


Figure 5: The PILOT module

The DRIVING MONITOR, the top level of the PILOT, receives route segment descriptions whenever a newly created route segment appears in the BLACKBOARD. The DRIVING MONITOR breaks the route segment into pieces called *driving units*, so that the PERCEPTION can detect one driving unit separately (see Figure 6). The DRIVING MONITOR builds a *driving unit description* for each driving unit, describing the location of the driving unit and the objects which PERCEPTION should see. Following the principle of information hiding, the driving unit description tells where and what objects the PERCEPTION should see, but it does not tell how to detect them.

The DRIVING UNIT FINDER works as an interface to the PERCEPTION, getting the newly created driving unit description from the DRIVING MONITOR and sending it to the PERCEPTION through the BLACKBOARD. If the result of perception is written in the driving unit description in the BLACKBOARD, the DRIVING UNIT FINDER retrieves it into the PILOT.

The POSITION ESTIMATOR determines the vehicle position using position estimations generated both by perception and by dead reckoning. When PERCEPTION sees objects which can be landmarks, PERCEPTION can estimate the vehicle position using predicted object locations and shapes. For example, stairs and intersections are good landmarks. Sidewalks are not sufficient as a landmark, because PERCEPTION looking at only a sidewalk cannot tell the location along a sidewalk. It can tell only the distance from the edge of a sidewalk. Therefore, the position estimation by the PERCEPTION is sometimes complete and sometimes is not complete. The position estimation by dead reckoning is given by the HELM. The POSITION ESTIMATOR combines both of them and determines the vehicle position.

The BLACKBOARD stores two kinds of vehicle positions. One is *perceived vehicle position*, and the other one is *moving vehicle position*. Because the perceived vehicle position is estimated by only the POSITION ESTIMATOR once per driving unit, it is discrete. On the other hand, the moving vehicle position is estimated by not only the POSITION ESTIMATOR but also the HELM using dead reckoning. Because the HELM updates the moving vehicle position frequently, it tells the vehicle current position. Because both positions are stored in the BLACKBOARD, all modules can use them. Currently they are used by the PERCEPTION and the LOCAL PATH PLANNER.

The DRIVING UNIT NAVIGATOR plots *local path constraints* in the driving unit using the result of perception and the driving constraints given by the MAP NAVIGATOR.

The LOCAL PATH PLANNER gets the local path constraints and creates the *local path plan* from the vehicle's current position, through all intervening driving units, reaching to the far edge of the newly scanned driving unit. Because the vehicle is not in the newly scanned driving unit, the LOCAL PATH PLANNER keeps the old local path constraints to calculate local path plan, and discards them if they are not necessary. The algorithm for local path planning is based on Lozano-Perez's method [3]. This method generates a sequence of line segments, which the LOCAL PATH PLANNER converts to a smooth path. Therefore, the vehicle turns along a curved line. The generated local path plan is passed to the HELM through the BLACKBOARD.

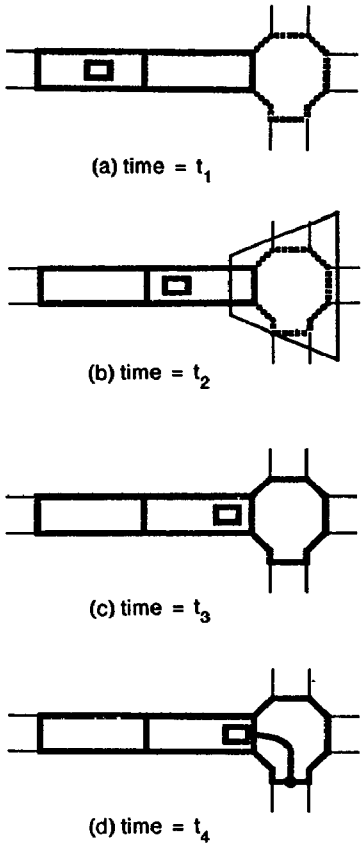


Figure 6: Process Sequence in the PILOT

Figure 6 illustrates the process sequence in the PILOT. The black painted box is the vehicle. The bold line indicates the driving unit detected by the PERCEPTION already. The dotted line shows the driving unit newly created by the DRIVING MONITOR. At time t_1 the DRIVING MONITOR creates new driving unit description and sends it to the DRIVING UNIT FINDER. The DRIVING UNIT FINDER sends it to the PERCEPTION. When the vehicle reachs the best place to detect the required driving unit, the PERCEPTION inputs image data with the sensors (time t_2). The thin line shows the sensor view frame. At time t_3, the PERCEPTION finishes processing and reports the object shapes and the vehicle position. The DRIVING MONITOR starts creating next driving unit description. And at time t_4 the LOCAL PATH PLANNER generates new local path plan and passes to the HELM.

2.3.4. The HELM and Driving the Vehicle

Whenever a new local path plan appears in the BLACKBOARD, the HELM discards the old path plan and picks up new one. The HELM converts the local path plan, which tells only trajectory, into vehicle driving commands, and feeds them to the vehicle. In addition to driving the vehicle, the HELM is responsible for maintaining the *vehicle moving position* stored in the BLACKBOARD. Because the task of the HELM needs quick response to control the vehicle, it is implemented as an independent process.

2.3.5. The PERCEPTION Module

PERCEPTION picks up a driving unit description from the BLACKBOARD when a new one appears. PERCEPTION has two tasks: detecting navigable passages, and, if possible, estimating vehicle position. The details of PERCEPTION are explained in the next section.

2.3.6. The BLACKBOARD

The BLACKBOARD provides the modules with facitilies for communication and data management. Our BLACKBOARD looks like a traditional blackboard, with several additional properties that make it useful for navigation:

1. **parallel asynchronous execution of modules.** This property makes it possible to execute all modules in parallel.

2. **transparent networking between processors.** This property makes it easier to build interfaces between modules.

3. **no pre-compilation of data retrieval specification.** This property makes it easier to pick up desired data from the BLACKBOARD.

4. **geometric reasoning.** Coordinate transformation and geometric calculations are done by the BLACKBOARD. Data retrieval from the BLACKBOARD with geometry is used in several places.

2.3.7. The MAP

The MAP is the main data base in our navigation system. It consists of two parts, GEOGRAPHICAL MAP and OBJECT DATA BASE.

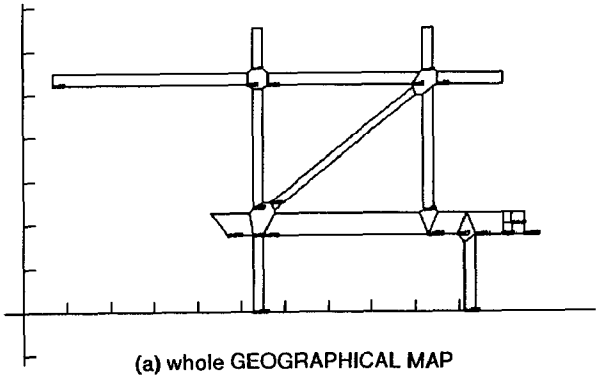
The GEOGRAPHICAL MAP is similar to usual maps which we use in daily life, and tells object locations with their outlines. The MAP NAVIGATOR uses the GEOGRAPHICAL MAP to pick up objects belonging to the route segment. Figure 7 shows current GEOGRAPHICAL MAP.

The OBJECT DATA BASE stores the object descriptions.

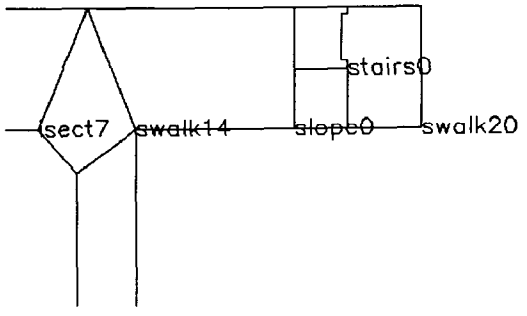
1. **Perception Feature:** Both of the object's three dimensional shape and its color are stored to produce object description for the PERCEPTION. Three dimensional shape is expressed with surfaces.

2. **Role in Navigation:** Some objects can be landmarks, and other objects can be paths. Therefore, the OBJECT DATA BASE indicates roles of objects: *landmark*, *path*, *obstacle* and *no meaning*. Also, navigation costs on objects are stored. These data are useful when the MAP NAVIGATOR performs route planning.

The MAP is stored in the BLACKBOARD and accessed only by the MAP NAVIGATOR. Currently we assume that the MAP has all necessary information and it is correct and complete. But our future work will include "map revising", starting with an incomplete map, and updating it during navigation.



(a) whole GEOGRAPHICAL MAP



(b) stairs and slope

Figure 7: The GEOGRAPHICAL MAP

2.3.8. The NAVIGATION MONITOR

The NAVIGATION MONITOR is a graphics system which displays the current navigation situation. It displays the route segment, the driving unit, the sensor view frame, the vehicle position, and the local path plan. Because the NAVIGATION MONITOR is implemented as an independent process, it does not disturb other modules. Whenever the data to be displayed appear on the BLACKBOARD, the NAVIGATION MONITOR retrieves and displays them.

3. Perception Using Colored-Range Image

3.1. Requirements and Approach

The basic problem for the perception is caused by the complexity of outdoor scenes. Some objects have very complicated shapes and colors from which it is difficult to extract surfaces or edges. This requires powerful sensors and algorithms for object detection. Also, the processing time of the perception is critical, because it eventually constrains vehicle speed. Even if we have a very powerful perception program which can detect complicated objects, it will be computationally expensive. Therefore, having a *single* powerful object detection program is not an adequate solution.

To overcome these difficulties, we take a sensor fusion approach. There are several types of sensor fusion methods [4]. This PERCEPTION module uses two types of sensor fusion.

The first type is low level sensor fusion, doing segmentation using color and range data simultaneously. We call the image which has color information in addition to range values a *colored-range* image. Using this method, we can segment objects with uniform color but varying surface orientation, as well as objects with smooth surfaces and varying colors. This method is used to analyze complicated scenes.

The second type of fusion is a higher level sensor fusion or sensor selection. The PERCEPTION module has both a *colored-range* segmentation program, mentioned above, and a color segmentation program, and uses these programs selectively. The former program can extract segments in complicated scenes, while the later program is adequate for simple flat scenes and uses much less processing time. This type of sensor fusion achieves both powerful perceptual ability and fast processing.

3.2. PERCEPTION Module Architecture for Sensor Fusion

The PERCEPTION receives perceptual requests from the PILOT, and analyzes sensor data to compute its response. The main effort to design the PERCEPTION module is how to combine several types of sensors and sensor data processing modules into one system and make them work efficiently. We designed a hierarchical structure and a monitor module which manages all parts of the hierarchy.

Figure 8 illustrates the structure of the PERCEPTION module. This is composed of the PATCH MAKER, the OBJECT FINDER, the POSITION CALIBRATOR, and the PERCEPTION MONITOR. Sensor data go through in the order of the PATCH MAKER, the OBJECT FINDER, and the POSITION CALIBRATOR.

The data interface between each module is designed to be independent of the algorithms used by each module. This allows each layer in the hierarchy to have several modules based on different algorithms. In the current system, the PATCH MAKER includes two types of segmentation modules, and one object finding module can work on the results of both segmentation modules because of the common data interface. The PERCEPTION MONITOR is a key for this hierarchical processing. We describe it in detail in the next section. Other modules are explained in following paragraphs.

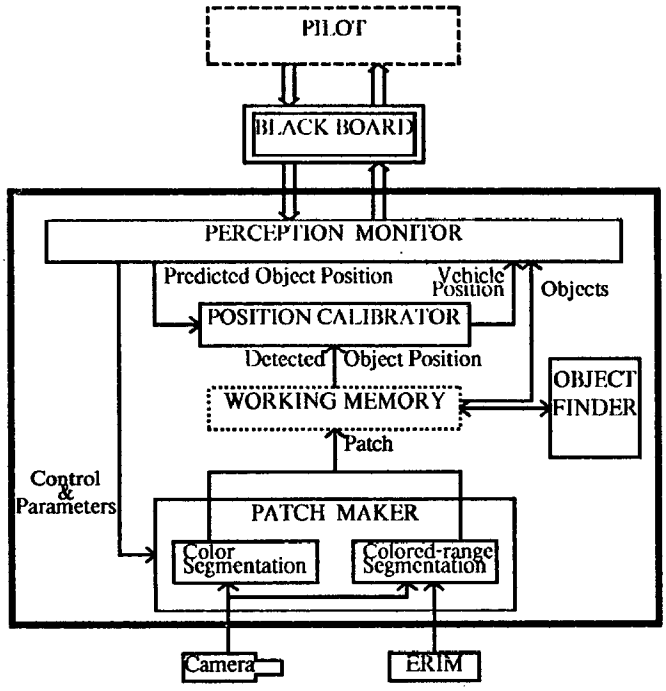


Figure 8: Structure of PERCEPTION Module

3.2.1. The PATCH MAKER

As a segmentation module (PATCH MAKER), this system has a color segmentation module, a range segmentation module, and a *colored-range* segmentation module. The color segmentation, the range image segmentation, and *colored-range* image segmentation are described in Section 3.4.

The data structure which holds patch data is common to both segmentation modules. These data include *color type, surface type and normal, polygons for boundary shape*, and *relation to neighbor segments.*

3.2.2. The OBJECT FINDER

The OBJECT FINDER identifies each segment as a part of a predicted object. This algorithm is described in rule-base style, with two kinds of rules. One kind identifies detected segments as parts of predicted objects, and the other type of rule finds the actual correspondence between perceived and predicted polygons. In other words, the first set of rules deals with symbolic matching, while the second knows about detailed geometry.

The OBJECT FINDER uses a WORKING MEMORY. The PATCH MAKER assigns the Patch data into the WORKING MEMORY. Also, the PERCEPTION MONITOR assigns predicted object shape and feature data into the WORKING MEMORY. These data include color, surface type, and shape. The OBJECT FINDER uses WORKING MEMORY data to match predicted with detected data.

3.2.3. The POSITION CALIBRATOR

The predicted objects are described in the current coordinate system, but the vehicle coordinate system is used to describe the detected objects. The POSITION CALIBRATOR then computes the vehicle position in the current coordinate system, applying the transformation matrix between two coordinate systems. The problem for this computation is that the predicted object shape and the detected object shape are not same because of imperfections in the MAP and the perception. Therefore, the POSITION CALIBRATOR has to find the most appropriate matching for these two shapes.

To get the best matching point, the POSITION CALIBRATOR calculates the distance between the predicted vertices and the detected vertices of object polygons, and finds the position which minimizes the distance. Sometimes, a scene is composed of only parallel lines (e.g., *sidewalk*) or a point (e.g., *tree*), which are insufficient to decide a matching point. In this case, the POSITION CALIBRATOR derives a line equation on which the vehicle is located instead of a point for vehicle position.

3.3. The PERCEPTION MONITOR

The PERCEPTION MONITOR has two major roles: communication with other modules (the PILOT) and control of internal submodules. As mentioned before, a design principle of this system is to provide a common structure for different sensors and algorithms. This tends to make the module interface rather high level. For example, an image input position is usually decided by an external module using sensor parameters. However, if there are several types of sensors with different view angles, the common interface for those modules will be ***where the PERCEPTION should see*** instead of ***where the PERCEPTION should look from.*** This means the PERCEPTION module itself has to decide where the best position is from which to see the requested place. The communication with other modules means doing such kinds of interpretation between the high level module interface commands and actual commands to internal submodules.

Control flow of the perception process is rather simple. It progresses in order of segmentation, object finding, and position calibration. The PERCEPTION MONITOR activates the PATCH MAKER, the OBJECT FINDER, and the POSITION CALIBRATOR in this sequence. The functions for the interpretation of the high level commands from the other planning module (the PILOT) are described in following paragraphs.

3.3.1. Selection of Sensor and Segmentation Modules

The PILOT requests what objects to see, but does not say which sensor should be used. The PERCEPTION MONITOR decides which sensor and segmentation module is the best for the requested objects. The current system has two sensors and segmentation modules. If all requested objects are *sidewalks* or *intersections* on a flat plane, the PERCEPTION MONITOR selects the color segmentation module as a PATCH MAKER. If three-dimensional objects such as stairs, and slopes are included in the requested objects, the PERCEPTION MONITOR selects the *colored-range* segmentation module.

3.3.2. Decision of View Frame and Resolution

The PILOT requests to see objects as quickly as possible within a tolerable error allowance, but does not specify the resolution or view frame of PERCEPTION. The view frame of PERCEPTION depends on a tolerable error allowance for the PERCEPTION. If the error allowance is small, a meaningful view frame will be limited to near the vehicle, even if the view angle of the sensor covers a greater area. Also, the view frame is a function of PERCEPTION resolution. If the resolution is very fine, the view frame can be wider, keeping the same error allowance.

An interesting point is that the processing time of the PERCEPTION is a function of the resolution. Therefore there are two typical strategies for equal error allowance. One is a seeing a closer area with rough resolution in a short time. The other is a seeing a farther area with fine resolution in a longer time. The PERCEPTION MONITOR decides the optimal view frame and the resolution which allows the vehicle to run at maximum speed.

Figure 9 shows the relation between the velocity (V) and the distance the vehicle can run in one cycle (perception distance, D) for several error allowances. We got this result from two relations: the relation of resolution and the maximum perception distance for equal error allowance, and the relation of the resolution and the processing time (T). Velocity is defined as $V = D/T$.

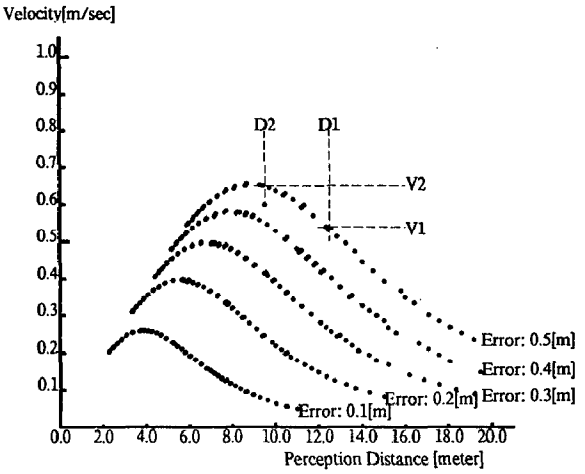


Figure 9: Distance and Velocity by Error Allowance

It is interesting to note that Figure 9 indicates that the maximum speed point of our system is at a rather short distance and low resolution. When the vehicle runs at a speed of V_1, the best perception distance and resolution is at D_1. This is because the PERCEPTION can see the widest area at that point. This increases the chance of finding landmarks or obstacles. Then, if the vehicle trys to run faster (at speed of V_2), the perception distance becomes shorter (D_2). This shows that as speed increases, the vehicle becomes short-sighted.

The PERCEPTION MONITOR has a data table in which are stored the values of optimal resolution and perception distance for the error allowance. Using this table, the PERCEPTION MONITOR provides the optimal resolution and perception distance.

3.3.3. Decision of Image Input Position

The PILOT indicates which region the PERCEPTION should see, but does not indicate when the PERCEPTION should see it. This is because the view frame of the PERCEPTION depends on the sensor used, and the PILOT does not know which sensor will be used. Furthermore, when the PERCEPTION uses a pan and tilt mechanism, only the PERCEPTION can decide image input timing and position.

The position decision algorithm has two steps. First, this module simulates the view frame and the vehicle's future path which is posted in the BLACKBOARD by the LOCAL PATH PLANNER. When the simulated view frame covers the region which the PILOT has requested the PERCEPTION to see, this vehicle position is defined as the image input position. Second, this module monitors current vehicle position by watching the moving vehicle position on the BLACKBOARD. And, when the moving vehicle position reaches the image input position, this module controls sensors to take an image.

3.3.4. Creation of Segmentation Parameters

To have good segmentation results, we need not only appropriate segmentation parameters, but also good algorithms. It is very difficult to know appropriate parameters unless PERCEPTION has scene knowledge. In this system, the PILOT predicts objects for the PERCEPTION, but the description is general and does not indicate segmentation parameters for the objects.

THE PERCEPTION MONITOR creates appropriate parameters, and the segmentation modules use them. Currently, the PERCEPTION MONITOR creates color type and minimum area of segments as segmentation parameters. For example, if the predicted objects are only a *sidewalk* and *grass*, the PERCEPTION MONITOR decides that types of color are *GRAY* and *GREEN*, and reports them to the segmentation module (the PATCH MAKER). Another example is the minimum area for segments. If there are no small predicted objects, the PERCEPTION MONITOR sets a rather large value for minimum segment area. This eliminates the noisy segments, and makes a simple segment list which is easy to analyze in the later object finding phase.

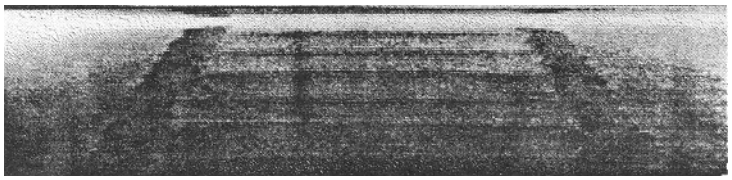
3.4. Colored-Range Image Analysis

It is very difficult to recognize complex objects in outdoor scenes using only one kind of sensor, but several different sensors can provide a lot of clues about the environment. For example, use of both range data and color images provides a very powerful vision system for outdoor scene analysis, because range data provide information about the geometry of a scene, and color images provide an important physical property of objects. In order to use these different sensor data, we must integrate them using sensor fusion techniques. The registration between range data and color images can be a first step to sensor fusion. In the following sections, we describe the registration algorithm for color and range image, the segmentation procedure for range data, the color segmentation algorithm, and how to use a *colored-range* image to recognize stairs and slopes.

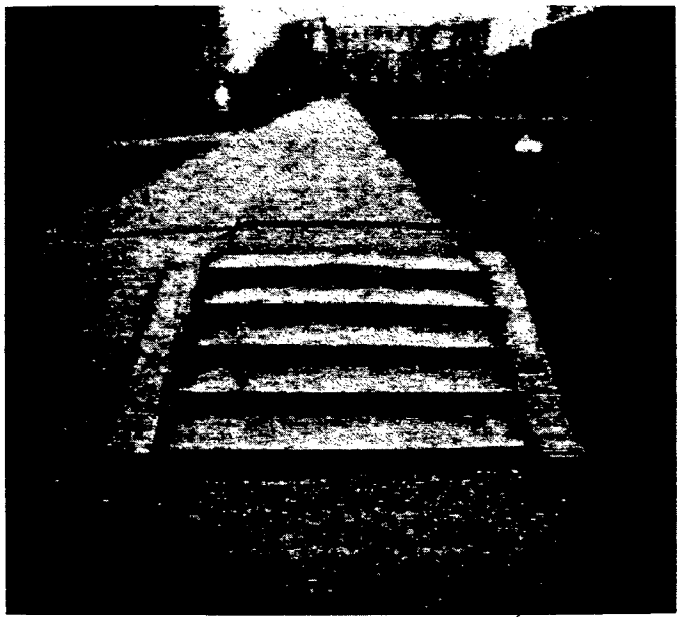
3.4.1. Registration

We must know the relative positions and orientations of the color camera with respect to the range scanner to register range data into camera-centered coordinates. These can be computed using a calibration procedure. The calibration step consists of initial value estimation and optimum value finding procedures. In the initial value estimation step, the measured angles are used to simplify the problem, so the position of the camera relative to the range scanner and its focal length are the only unknown

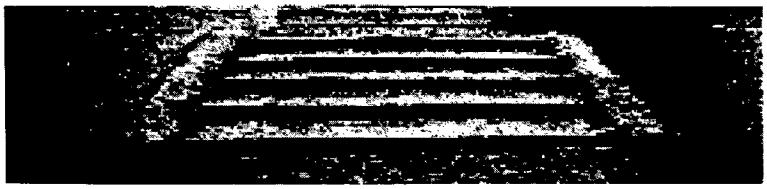
parameters. The unknown parameters are computed by solving a least-squares criterion. Once the initial values of the camera parameters are computed, we can obtain more accurate parameters using a modified Newton-Gauss method. In the experiment, the wide-angle lens is used to capture color images, which prevents us from using a linear perspective transformation for projection. Use of a third-order polynomial for projection provides good registration. The projection model is developed through a conventional camera calibration procedure. Since the camera/scanner transformation and the perspective transformation are computed, the range data can be registered with the color image. This is done by applying the transformation to each point of the range image, so the corresponding pixel point in the color image can be computed. The color intensity for the computed pixels in color image then are assigned to the range image. The registration procedure is illustrated in Figure 10.



(a) Range image



(b) Original color image



(c) *Colored-range* image

Figure 10: Colored-range Image Registration

3.4.2. Range Image Segmentation

The range segmentation module generates surface segments using three basic attributes at each pixel: jump edges, surface normal, and surface curvature. Each segmented region has a surface normal vector, a surface curvature, 3-D edges, and a label indicating smooth region or rough region. The detailed algorithm for range data analysis can be found in Hebert's recent work [2].

3.4.3. Color Segmentation

The color segmentation module uses Wallace's color classification algorithm [5] which uses a standard quadratic discrimination function for multivariate normal distributions of mean vectors and covariance matrices for the Red, Green, and Blue components of an image. This module cannot detect 3-D position data for segments. However, when we assume all objects are on a flat ground plane, this module can calculate 3-D values for segments using the relation between the perspective image and the scene [1].

3.4.4. Overlap Segmentation

The registration process creates an image in which each pixel has a color value and a range value. The segmentation for the *colored-range* image is done as illustrated in Figure 11. First, the color label and the surface label are obtained by color segmentation and surface segmentation. Then, the color-surface label for each pixel is obtained by an AND operation of the color label and the surface label. Overlap segmentation is done using this label value.

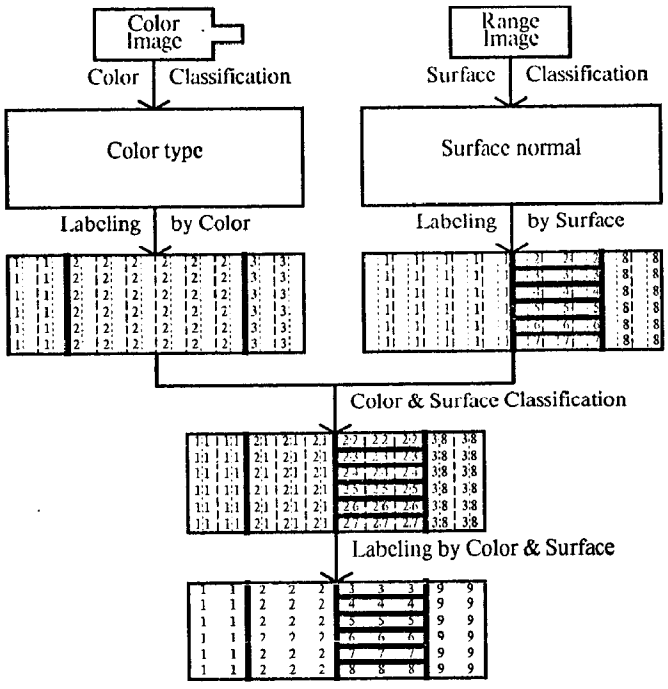
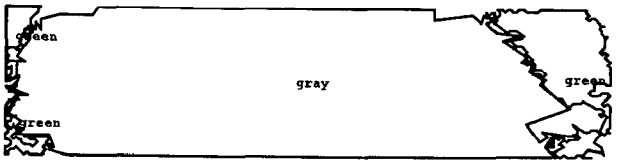


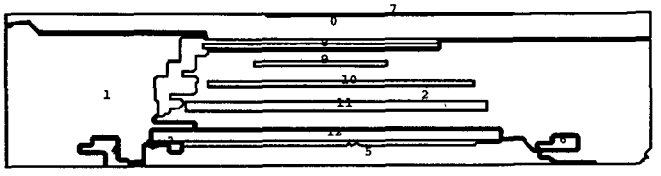
Figure 11: Overlap Segmentation by Color and Range Data

3.4.5. Result of a Real Scene Analysis

One good example to show the effectiveness of the *colored-range* segmentation module is a slope and stairs scene of a campus sidewalk. The slope and the stairs are made of concrete and have the same gray color. The slope and road-side grass are on almost the same plane. Therefore, segmentation using only color can not separate the slope and the stairs, and segmentation using only range can not separate the slope and the road-side grass. Overlap segmentation using *colored-range* image can extract the slope which is only navigable region in this scene. Figure 12 shows the results of color segmentation, range segmentation, and colored-range segmentation drawn with polygons.

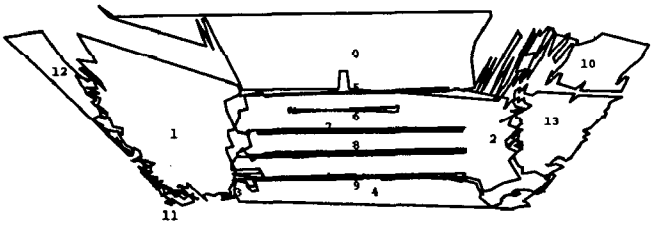


(a) Color segmentation



(b) Range segmentation

(c) *Colored-range* segmentation



(d) *Colored-range* segmentation projected on x-y plane

Figure 12: Colored-Range Segmentation

4. Conclusion

We have developed the Sidewalk Navigation System which can drive the vehicle on the test site, the CMU campus. Because our test site involves sidewalks and intersections, slopes, stairs, and grass, the navigation system should have ability to select the best navigation mode depending on the situation. Our architecture works well in this environment, using both route planning and local path planning, selecting vehicle driving mode, and selecting sensors. The scene which includes the stairs, the slope, and the grass is hard to detect with only a TV camera or only a range sensor. But our perception module can analyze even this scene using sensor fusion with color and range data. And sensor selection dependent object prediction saves computation time. Our current system demonstrates the framework of an outdoor robot navigation system.

Our future work will include pipeline processing in the PILOT module, expanding the PERCEPTION module to detect other objects, and map revising. And we will add a capability for *cross-country navigation*.

Acknowledgements

The CMU Sidewalk Navigation System is created as a part of the Strategic Computer Vision Project at CMU. This system includes several programs produced by project members. The BLACKBOARD was produced by Steven Shafer, Anthony Stentz, and Charles Thorpe [4]. The color classification program was based on the code made by Richard Wallace [5]. The range image segmentation program was made by Martial Hebert [2]. The system architecture design is based on the proposal by Steven Shafer, Anthony Stentz and Charles Thorpe [4].

We would like to thank Regis Hoffman for creating the LOCAL PATH PLANNER, Paul Allen for writing the NAVIGATION MONITOR, and Ralph Hyre for useful blackboard facilities. Thanks to Mike Blackwell, Kevin Dowling, and James Moody for hardware, system interface, and video support. The civil engineers Red Whittaker, Chuck Whittaker, Francois Bitz, Steve Berman, and Kai Lee developed and maintain the Terregator vehicle.

References

[1] Wallace, R., Matsuzaki, K., Goto, Y., Crisman, J., Webb, J., Kanade, T.
Progress in Robot Road-Following.
In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pages 1615-1621. April, 1986.

[2] Hebert, M.
Outdoor Scene Analysis Using Range Data.
In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pages 1426-1432. April, 1986.

[3] Lozano-Perez.
An Algorith for Planning Collision Free Paths Among Polyhedral Obstacles.
In *Communications of the ACM*. October, 1979.

[4] Stentz, A., Shafer, S., Thorpe, C.
An Architecture for Sensor Fusion in an Autonomous Land Vehicle.
In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pages 2002-2011. April, 1986.

[5] Wallace, R.
Robot Road Following by Adaptive Color Classification and Shape Tracking.
In *Forthcoming in Proceedings 1986 AAAI*. 1986.