Integrating obstacle avoidance, global path planning, visual cue detection and landmark triangulation in a mobile robot

David Kortenkamp, Marcus Huber, Clare Bates Congdon, Scott Huffman,
Clint Bidlack, Charles Cohen, Frank Koss, Ulrich Raschke and Terry Weymouth

Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109
korten@engin.umich.edu

ABSTRACT

This paper describes the design and implementation of an integrated system for combining obstacle avoidance, path planning, landmark detection and position triangulation. Such an integrated system allows the robot to move from place to place in an environment, avoiding obstacles and planning its way out of traps, while maintaining its position and orientation using distinctive landmarks. The task the robot performs is to search a 22m X 22m arena for 10 distinctive objects, visiting each object in turn. This same task was recently performed by a dozen different robots at a competition in which the robot described in this paper finished first.

1 INTRODUCTION

Many mobile robots systems can plan paths and avoid obstacles. Many other systems use vision to detect landmarks and triangulate their position. Few mobile robot systems combine obstacle avoidance, path planning, landmark detection and triangulation within a single, working architecture that performs tasks in a dynamic world. Such an integrated system would move from place to place in the world, avoiding obstacles and planning its way out of traps, while maintaining its position and orientation using distinctive landmarks, whose positions were not prior knowledge. This paper describes the design of such a system and its implementation on an actual mobile robot.

1.1 Task description

The task our robot performs is to explore an up to 22m X 22m arena searching for 10 distinctive objects (which were also the landmarks) and then visit each of the objects. Visiting was defined as moving to within two robot diameters of the object. Our robot had 20 minutes to perform this task. This task was one stage of a three stage robot competition held at the National Conference for Artificial Intelligence in San Jose California in July 1992. A dozen robots participated in the competition, with the system described in this paper finishing first. For more details on the competition rules and scoring see [6].

The arena boundaries were defined by walls, and the arena floor was strewn with obstacles (cardboard boxes). Objects were ten foot tall, three-inch diameter poles. To each pole we attached an omni-directional, barcode tag, which allowed our computer vision system to distinguish one pole from another (see Section 2.3). The objects could be seen above the obstacles, while the clearance between obstacles was a minimum of 1.5m. See Figure 1 for a diagram of the arena.

1.2 Robot description

Our robot is a Cybermotion K2A named CARMEL (Computer Assisted Robotics for Maintenance, Emergency and Life Support). CARMEL has three synchronously driven and steered drive wheels. The body of CARMEL does not turn with the wheels. For sensing, CARMEL has a ring of 24 Polaroid sonar sensors and a single, black and white

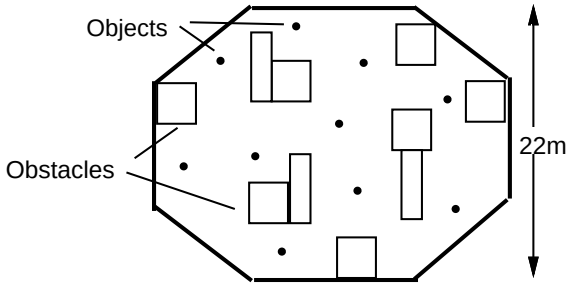

Figure 1: Diagram of the competition arena. Objects are drawn slightly larger than scale.

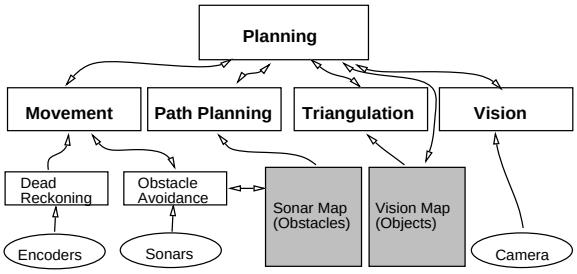

Figure 2: CARMEL's system architecture. Gray boxes are data structures, ovals are sensors, boxes are routines and arrows represent the flow of information.

CCD camera. The camera is mounted on a rotating table that allows it to rotate 360 degrees independently of robot motion. For computing, CARMEL has three processors: one controls the drive motors, one fires the sonar ring and the third, a 486-based PC clone, executes all of the high-level modules described in this paper. This latter computer also has a framegrabber to allow for image acquisition. All computation and power is contained entirely on-board.

1.3 Overview

CARMEL's software architecture is arranged hierarchically (see Figure 2). At the top-level is a supervising planner that decides what the robot should do next and then calls appropriate sub-modules. The next level are sub-modules for moving the robot, planning global paths, determining the robot's position in a world coordinate system, and finding objects. The MOVEMENT sub-module contains two sub-routines: one to compute the robot's current location using the wheel encoders and another to avoid obstacles using the sonar sensors. Two main data structures contain information about the world. One holds a certainty grid of obstacles constructed from the sonar sensors; this is used by the path planning and obstacle avoidance routines and is constructed and maintained by the obstacle avoidance routines. The second data structure contains the locations of objects as determined by the vision routines and is constructed and maintained by the supervising planner. These two sets of data, although represented in the same coordinate system, remain independent and are never fused. Robot motion (driving and steering) is controlled by the obstacle avoidance sub-routine. Camera motion (i.e., the rotating table) is controlled by the supervising planner.

The rest of this paper will briefly describe each sub-module and then detail how the supervising planner integrates all of them to perform the required task. Finally, the system is evaluated by analyzing its performance over many experimental runs in different environments, including the winning run during the actual competition.

Figure 3: Left figure is CARMEL and an environment. Right figure shows CARMEL's Histogram Grid of obstacle locations after moving from Start to O.

2 SUB-MODULES OF THE ARCHITECTURE

CARMEL's architecture is highly hierarchical. There are many sub-modules that the planner can call to accomplish various tasks. This section gives a brief description of each sub-module and the task it performs.

2.1 Obstacle avoidance

Obstacle avoidance on CARMEL is done solely with its sonar sensors and has two components: (a) a unique method for detecting and rejecting noise and crosstalk with ultrasonic sensors, called error eliminating rapid ultrasonic firing (EERUF) [4]; and (b) an obstacle avoidance method called the vector field histogram (VFH) [2, 3].

The innovative feature of EERUF is its ability to detect and reject ultrasonic noise caused by other mobile robots in the environment, or by crosstalk (a phenomenon where one sensor receives the echo from another), caused by CARMEL's own sensors. Since EERUF dramatically reduces the problem of crosstalk, CARMEL can fire its ultrasonic sensors at a rate of 160 ms per sensor, two to five times faster than in most other mobile robot applications. This faster firing rate is a crucial factor that allows CARMEL to react in time to unexpected obstacles, even when traveling at high speeds (up to 780 mm/sec).

To map the obstacles in the environment, CARMEL uses another innovative approach called a Vector Field Histogram (VFH). The VFH method uses a two-dimensional Cartesian grid, called the Histogram Grid (illustrated in Figure 3), to represent data from ultrasonic (or other) range sensors. Each cell in the Histogram Grid holds a certainty value that represents the confidence of the algorithm in the existence of an obstacle at that location. This representation was derived from the certainty grid concept that was originally developed by Moravec and Elfes in [8]. In the Histogram Grid, certainty values are incremented when the range reading from an ultrasonic sensor indicates the presence of an object at that cell. Based on data in the Histogram Grid, the VFH method creates an intermediate data representation called the Polar Histogram. The spatial representation in the Polar Histogram can be visualized

Figure 4: The Polar Histogram (left) has "mountains" in the direction of obstacles. CARMEL is steered toward a "valley" in the direction of the target. Right figure shows the same Polar Histogram in polar coordinates.

---

as a mountainous panorama around the robot, where the height and size of the peaks represent the proximity of obstacles, and the valleys represent possible travel directions (see Figure 4). The VFH algorithm steers the robot in the direction of one of the valleys, based on the direction of the target location.

2.2 Global path planning

In addition to VFH, CARMEL uses a global path planner that searches the Histogram Grid created during obstacle avoidance and creates a list of via points (intermediary goal points) that represents the shortest path to the goal. The algorithm was developed at the Oak Ridge National Laboratory and reimplemented on CARMEL [1]. The algorithm first expands each occupied cell of the Histogram Grid by half the robot diameter. Then a potential field proportional to the distance from the start to the goal is computed. A gradient descent is performed from the start position to the goal and via points are selected whenever the slope of the path changes. This algorithm is very quick, taking less than one second to run on a grid size of 256 X 256 cells (each cell represents 20 cm in the environment). The speed of the algorithm lets it run on-the-fly to extract CARMEL from potential trap situations.

2.3 Visual cue detection

Objects (10 ft. poles in our environment) were tagged with an omni-directional barcode to facilitate detection. The object tag design used for CARMEL consists of a black and white stripe pattern placed upon PVC tubing with a four inch diameter, allowing the tags to be slipped over the object poles. An example object tag is shown in Figure 5. The basic stripe pattern is six evenly spaced horizontal black bands of 50 mm width, with the top of the top band and the bottom of the bottom band spaced 1000 mm apart. The white gaps between the black bands correspond to the bit positions in a five-bit word. A white space between two bands corresponds to an "off" bit, while filling the space with black corresponds to an "on" bit. The five bits between the six bands can then represent 32 unique objects.

The actual algorithm for extracting objects from an image required no preprocessing of the image. The algorithm makes a single pass over the image, going down each column of the image looking for a white-to-black transition that would mark the start of a potential object. A finite state machine keeps track of the number and spacing of the bands. After finding enough bands to comprise a tag the algorithm stores the tag id and pixel length. Once a column is complete, the eligible objects are heuristically merged with objects found in previous columns. The

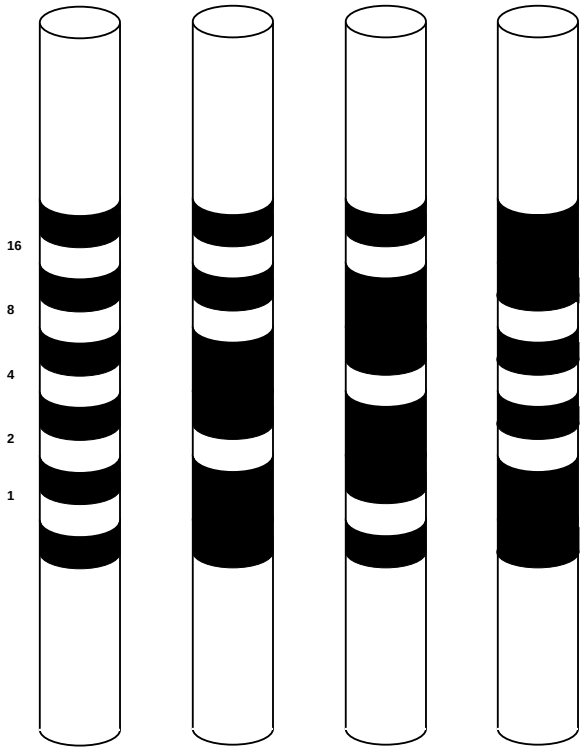

Figure 5: Example object tags showing the basic pattern and objects with bit pattern of 0, 5, 10 and 17.

distance between the top of the top band and the bottom of the bottom band, in terms of the number of pixels in the image, is then used to estimate the actual distance from the camera to the object. The algorithm has an effective range of about 12 meters. See [7] for a detailed description and analysis of the vision algorithm.

2.4 Landmark triangulation

The method used on CARMEL for triangulation is based on circle intersection (see [5] for an overview of three-object absolute-positioning algorithms). Given objects A, B, and C, a circle is formed with objects A, B, and the angle between them as viewed from the robot, as shown in Figure 6. Because this angle is a relative measure, the known x and y coordinates of the robot are not required. A second circle is formed similarly with objects B, C, and the angle between them. The two circles intersect at two points: object B's location and the robot's location. Therefore, the robot's location and orientation can be determined.

3 THE SUPERVISING PLANNING SYSTEM

The design of our planning system was motivated by several goals. First, and most importantly, we wanted to keep the planner very simple. The hierarchical design eliminated many problems such as contention for resources between sub-modules and lack of coordination between sub-modules. All scheduling was done through the top-level planner. Second, we wanted the overall system to be extremely robust. Because, the competition rules prohibited outside interference with the robots after they had started the task, the planner needed to handle even catastrophic errors during run-time. Third, we wanted to use existing research as sub-modules. For example, the obstacle avoidance routines we used had been developed over many years as stand-alone processes. We wanted to integrate them within the framework of the task without substantial modifications. Finally, we wanted to develop and test each sub-module

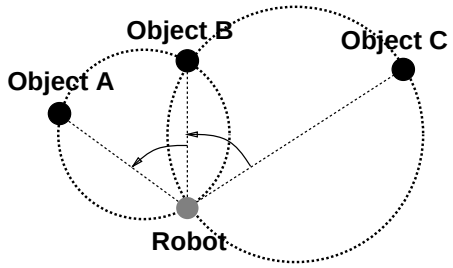

Figure 6: CARMEL's three object triangulation using the circles method.

independent of each other. This allowed several groups to work in parallel in developing the software system.

What emerged from these design goals was a strict hierarchy of modules with calls to each sub-module and information exchange between each sub-module being completely controlled by a supervising planner. No sub-module was allowed to run by itself and all sub-modules were guaranteed to return, even if they return an error condition.

3.1 Overview

The planner has three primary tasks: 1) It has to decide *what* to do (i.e., which sub-module to call); 2) It has to decide *when* to do it; and 3) It has to decide *where* to move the robot. The basic strategy for performing the task described in Section 1.1 is for the planner to 1) tell the MOVEMENT sub-module to move the robot to a particular location where objects might be seen, 2) tell the VISION sub-module to find all objects within a certain sweep area (e.g., the 180 degrees in front of the robot), 3) tell the MOVEMENT sub-module to approach the nearest object, 4) tell the VISION sub-module to verify the location of the object and finally, 5) tell the MOVEMENT sub-module to approach nearer to the object if the robot is still too far away from it. These steps are repeated until all of the objects have been visited. In some environments (such as the competition), steps (1) and (2) are repeated before beginning to visit objects. This allows many objects to be placed in the map before dead-reckoning errors accumulate.

At any time during this cycle, the planner can decide that CARMEL has moved far enough so that it's dead reckoning is probably in error. The planner then interrupts the cycle and directs the MOVEMENT sub-module to move the robot to a location from where it can see at least three objects, the minimum number needed to triangulate position and orientation. The planner points the camera at each of the three objects, and asks the VISION sub-module to determine new headings to the three objects. These headings are passed to the TRIANGULATION sub-module. That sub-module returns the new position and orientation of CARMEL, which the planner uses to update CARMEL's internal position and orientation.

3.2 Error recovery

There are two major causes of errors that the planner has to deal with: 1) Errors in movement, whereby the MOVEMENT sub-module cannot move the robot to the position requested by the planner; and 2) Errors in object location, whereby the VISION sub-module cannot verify that the robot has moved to an object. Recovery from these two types of errors is discussed in the following sub-sections.

3.2.1 Errors in movement

The MOVEMENT sub-module can detect two types of movement errors: trap situations, such as U-shaped obstacles and failures to attain the goal location. Traps are detected automatically by the obstacle avoidance algorithm. The planner's solution to this error is to call the global path planner to plot a set of via-points that will lead the robot out of the trap and to the goal location. The MOVEMENT sub-module is then called with this list of via-points instead of a single goal location. The MOVEMENT sub-module guides the robot through each via-point.

Failure to attain the goal location is not detected automatically by the obstacle avoidance algorithm. The algorithm was designed to repeatedly attempt to reach its goal location, running forever if necessary. Obviously, this was not desirable, since the goal location could be located inside of an obstacle or behind a long wall. To force the MOVEMENT sub-module to terminate, the planner estimates how long it should take the robot to reach the goal location. This time is passed to the MOVEMENT sub-module and the sub-module stops the robot and returns when this time limit is exceeded (or it has reached the goal location).

When the planner is informed of a failure to attain the goal location, it first determines if CARMEL is close enough to the goal location so that no further action is necessary. 'Close enough' varies depending on the situation. For example, when CARMEL is just trying to take an image a few meters away may be close enough, but when CARMEL is approaching within two robot diameters of an object, fractions of a meter are important. If the planner determines that CARMEL is not close enough, the planner asks the MOVEMENT sub-module to try again, maybe the obstacle was only temporary. After repeated failures, the planner chooses a new goal location that will also be suitable (such as the other side of the object) and starts the movement over.

3.2.2 Errors in object location

In the course of visiting an object, the planner will move CARMEL to a spot approximately three meters from the object. At this point, the planner will request another image to verify the location of the object. It is this new heading and distance that is used to make the final approach to within two robot diameters of the object; using this new, robot-relative information reduces the reliance on dead reckoning. However, there may be situations when the object cannot be seen from the verification location. In this case, the planner rotates the camera left and right in an ever-widening arc, searching for the object. If this fails, the planner assumes that CARMEL is too close to the object to detect it (the vision system has a minimum range of one meter) and so it backs CARMEL up. If this fails the object is assumed to be occluded and the planner chooses a new goal location on the opposite side of the object and starts the verification process over again.

4 RESULTS

An integrated system to perform a specific task has been described. In practice this system performed above expectations. The decision to dedicate specific sub-modules to different tasks paid off handsomely at execution time. Since sub-modules are encapsulated and guaranteed to return, they can have the entire resources of the computer at their disposal, without needing to sacrifice CPU time for the top-level planner. For the MOVEMENT sub-module, this results in movement that is fast and continuous, displaying none of the move-stop-move behavior of many other robots at the competition. For the VISION sub-module, this results in analysis of images that takes only a few seconds.

It also became apparent during the course of implementing the system, that the performance of each sub-module affected the design of the supervising planner. In particular, the VISION sub-module evolved over time and became much more accurate and could extract objects at much greater ranges. This reduced the need for dealing with uncertainty and error.

The system was tested over many runs in three different environments, with the largest being 22m X 22m. Error recovery routines for movement errors were tested by forcing CARMEL into traps. Error recovery routines for object location errors were tested by manually moving objects after their positions had been recorded by the planner. In practice, the error recovery routines for visiting objects allowed us to be less concerned with the precise position of the robot, thus we didn't need to perform landmark triangulation as often. Since landmark triangulation is time-consuming, requiring a movement and three images, the less often that it needs to be performed, the better.

In the actual competition, CARMEL found and visited all ten objects in just over nine minutes. This far outpaced the competition, none of whom could complete the task in the allotted 20 minutes. During the competition, CARMEL's maximum speed was set at 500 mm/sec, although it moved more slowly when approaching objects and when avoiding densely packed obstacles. The planning system's error-recovery mechanisms were barely used, only a few times did the camera need to pan in one direction or another to recover from vision and dead reckoning errors. In the actual competition we did not use the landmark triangulation system. We felt that CARMEL's dead-reckoning would be sufficient during the short amount of time that it took us to complete the task and we also had some last minute difficulties with the code. The landmark triangulation routine was tested extensively both in simulation and on actual runs of the robot prior to the competition.

4.1 Limitations

The most significant limitation of the system described in this paper is that it cannot handle parallel tasks; one sub-module must finish before anything else can be done. An obvious example of when parallelism would be desired is in positioning the camera so that it is pointing at the object while the robot is still moving towards the object. If we use a single processor computer, we cannot overcome this limitation without sacrificing our advantage in dedicating the processor to sub-modules performing tasks, which would result in slower execution of some sub-modules. This is a serious problem in the MOVEMENT sub-module, as the less often the robot can re-evaluate the obstacles in the world, the more slowly it has to move. If we use a multi-processor computer (or several connected computers) the planner must be much more sophisticated in scheduling sub-modules and dealing with contention for resources. Adding parallel task capability is an on-going research issue.

5 CONCLUSION

We have developed and implemented an integrated system of obstacle avoidance, global path planning, visual cue detection and landmark triangulation on an actual mobile robot. The robot's task was to explore a 22m X 22m arena, searching for and then visiting 10 objects. The system is rigidly hierarchical with a supervising planner controlling all processing. The system proved itself in a competition held in July 1992, finishing first out of a dozen competitors performing the same task.

6 ACKNOWLEDGMENTS

The authors wish to thank the other members of the CARMEL team, including Dr. Johann Borenstein, Doug Baker, Chris Conley, Roy Feague, LiQuang Feng, Rob Giles, Kevin Mangis, Alex Woolf, Annie Wu, and Cigdem Yasar. Support for the CARMEL team was provided by The University of Michigan College of Engineering, The University of Michigan Rackham School of Graduate Studies, the American Association for Artificial Intelligence, ABB Robotics Inc. and ABB Graco Robotics Inc. Co-authors Marcus Huber and Frank Koss are supported by DARPA grant no. DAAE-07-92-C-R012. Purchase of CARMEL and support for research using it is provided by Department of Energy grant no. DE-FG0286NE37969.

REFERENCES

[1] Claus S. Andersen, Claus B. Madsen, Jan J. Sorensen, Neils O. S. Kikeby, Judson P. Jones, and Henrik I. Christensen. Navigation using range images on a mobile robot. To appear in *Robotics and Autonomous Systems*, 1992.

[2] Johann Borenstein and Yoram Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal of Robotics and Automation*, 7(4), 1991.

[3] Johann Borenstein and Yoram Koren. The Vector Field Histogram for fast obstacle-avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3), 1991.

[4] Johann Borenstein and Yoram Koren. Noise rejection for ultrasonic sensors in mobile robot applications. In *The Proceedings of the IEEE Conference on Robotics and Automation*, 1992.

[5] Charles Cohen and Frank Koss. A comprehensive study of three-object triangulation. In *SPIE Mobile Robots VII*, 1992.

[6] Thomas Dean and R. Peter Bonasso. 1992 AAAI robot exhibition and competition. To appear in *AI Magazine*, Spring, 1993.

[7] Marcus J. Huber, Clint Bidlack, Kevin Mangis, David Kortenkamp, L. Douglas Baker, and Annie Wu. Computer vision for CARMEL. In *SPIE Mobile Robots VII*, 1992.

[8] Hans P. Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.