

# Design of a Control Architecture for Indoor Mobile Robot Navigation

Emre Uğur

January 28, 2004

## Abstract

The aim of this project is the study of hybrid method in a sample navigation task. The task of the robot is to navigate in an indoor world and reach to a target point. The map of the world is known a priori, however there exists randomly distributed small obstacles. Robot plans, decomposes and executes its actions, following an optimal route, while avoiding from obstacles in a reflexive manner. A simple 2-D simulator is used in experiments.

## 1 Introduction

Robotics research has a wide application area from industrial manipulators to space exploration missions. Each application field requires certain and special properties for robot morphology and control. For example, in a factory application, a robot arm should function as accurate as possible to accomplish its task. In order to be precise, robot should know or perceive all details of the environment, and should make complex calculations for the task. For an exploration task, robot should be able to move on an unknown and possibly unstructured terrain, and should tackle with unexpected situations with fast response rates.

Functioning in different environments requires different control strategies, that can be divided into three main groups [1]. First and the oldest approach, *Hierarchical paradigm* stems its roots from traditional artificial intelligence methods, where in each step, robot makes plans in a top-down manner using symbolic languages. Second paradigm, *Reactive paradigm* investigates the cognitive and biological sciences in order to design animal-like behaviors for the robots. *Hybrid deliberative/reactive*

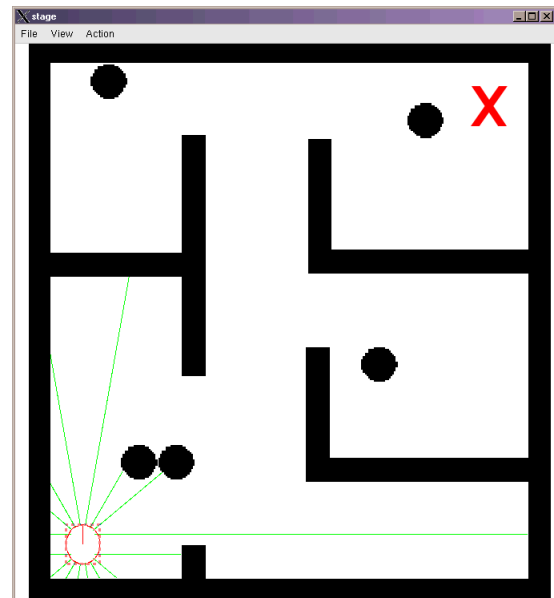


Figure 1: A sample environment snapshot with obstacles, walls and the robot.

*paradigm* tries to couple the crucial properties of the first two approaches in order to create intelligent robots which can exhibit fast responses when needed.

Selection of the robotic paradigm which is employed in this project, requires careful analysis of the problem. The task of the robot is to navigate in an indoor environment and reach to a target point. The indoor world is composed of corridor and doors that are connected to each other and the map is available to the robot. A number of small ob-

stacles are placed at different positions and their locations are unknown to the robot. This world can be viewed as an office with accurately known general structure, plus unknown and possibly dynamic objects inside, like furnitures or employees around. Robot is equipped with a perfect odometer, that all times it knows its exact position and orientation. Additionally, as a proximity sensor, a ring of sonar sensors locate on the robot body. Therefore, the objective of the robot is to navigate to a goal position whose coordinates are given, from an initial point. Figure 1 shows a sample world, with robot, walls, obstacles and goal location.

As demonstrated in Figure 1, instead of going ahead in the direction of the goal, robot should plan an optimal path through the corridors in order not to be trapped in a room. Therefore, robot should deliberately plan its action sequence, and in each step it should control its own state and performance. However, only planning the route is not sufficient since there are obstacles around which do not appear in the map. Thus, reflexive actions should be inserted into robot control to avoid from obstacles that appear on the path of the robot. Hybrid deliberative/reactive paradigm is applied to couple route planning and obstacle avoidance.

In Section 2, the simulation environment, robot sensing modalities and actuators will be discussed. The details of control architecture of the robot will be given in Section 3. A number of experiments are conducted for a number of situations, and results of these tests are given in Section 4. At the last section, future work and conclusions of the study is given.

## 2 Simulation Environment

As mentioned in Section 1, player/stage<sup>1</sup> software is used as the simulation environment. There exists a variety of computer programs that serve different functionalities for different applications. Robust rigid-body dynamics, collision detection and realistic response are some crucial functionalities that a simulation environment should provide for all robotic applications. Various computer programs<sup>2</sup> serve as physics engines to enable robots deal with real-world conditions. In this study, robot to robot

and robot to environment interactions do not have crucial importance, thus a simple and fast 2 dimensional simulator is sufficient. Additionally, player/stage environment provides a variety of sensors, which will be described later in this section.

### Environment

An indoor office environment is simulated for the robot to accomplish the given navigation task. The world is composed of a circular robot, corridors, rooms and obstacles with different size and shapes. The world map which is composed of a set of walls is known a priori by the robot, however the location and size of the obstacles are not given to the robot. Robot has a circular shape with a diameter of 2. The simulation does not provide any realistic interaction interface, so whenever the robot collides with any other object in the world, it freezes on that intersection position. Figure 1 shows the environment with its components.

### Locomotion

A positional proxy enables user to control robot movements. There is no real actuation mechanism in the robot like wheels or legs, but there are different ways to control robot position. I selected function which sets desired speed and turning rate where desired speed is directly assigned as the speed of the robot which results in unrealistic movements. For example if robot speed is increased a very high value, at that instant robot changes its position, discarding the objects in its path and locating itself according to that speed. Thus, I control the velocity of the robot by inserting a velocity control module which increments the speed of the robot to a certain amount. On the other hand, turning rate is more realistic, the robot cannot turn more than a certain amount in one instant and turning is performed gradually by the simulator.

A truth proxy which functions as a perfect odometer provides the position and orientation information of the robot in any instant. It works perfect such that it does not suffer from the most important problem of odometer measurements, accumulation of error, since there is no error in the device.

<sup>1</sup>www.sourceforge.playerstage/

<sup>2</sup>ie. Vortex, ODE

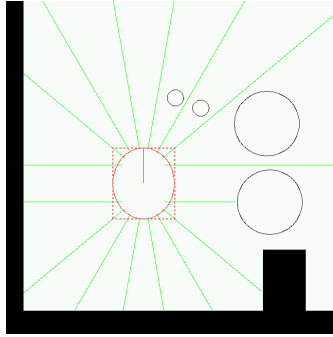


Figure 2: 16 sonar sensors with a range of 25 (robot diameter is 2). Since calculation of range using sonar sensors relies on ray intersection, the obstacles in the figure are not detected.

### Sensors

**Sonar Sensors:** Sensors are the indirect way of perceiving environment. There is only one type of range sensor located on robot body, which is simulated to measure accurately the characteristics of the world, without any noise or error. This approach is very dangerous from the robotic point of view, since real sensors always make erroneous and noisy measurements. Additionally, common problems of the sonar sensor, specular reflexivity and absorption problems do not occur in the simulation environment.

Sonar locates in the class of proximity sensors, where a sound wave is emitted and an external object reflects this wave to its sender [1]. The range of the object with respect to robot is computed using the time of flight of the wave. Sound sensors are commonly used in robotics since they are cheap and they provide range information directly. In player/stage, sonar sensors are simulated as linear rays emitted from robot body to different directions while in reality they are employed to view a certain degree of field, an area. There are 16 sound sensors located on the robot body, which is illustrated in Figure 2. Sonar sensors have a range of 25.

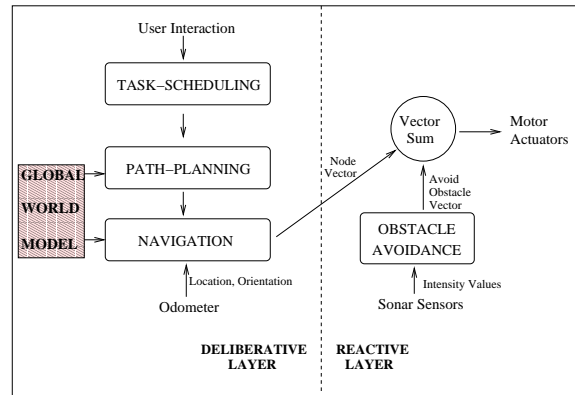


Figure 3: Layout of the Hybrid deliberative/reactive control architecture

## 3 Robot Control

In Section 1, different robotic paradigms and their application areas are discussed. Furthermore, the appropriate method for the navigation task in a known and structured environment with unknown small obstacles is derived to be the hybrid deliberative/reactive method. In 1990's, this approach was first applied to robotics in an effort to put back the cognitive problem solving ability, and planning (deliberation) of the agent to robot control without disputing the success of reactive behavioral control. In [1], it is proposed that hybrids are the best general purpose architectures and solutions to a variety of robotic problems. While asynchronous processing techniques allow reactive and deliberative processing performed simultaneously, good software modularity allows subsystems to be mixed effectively. There exists a planning modality on top of control architecture, which includes all deliberation and global world modeling. According to the directives of planning module, employing the current world state, using sensors, a set of behaviors execute until the plan is completed.

Objectives are set and methods are selected however finely grained decisions are not made, and that phase is handled by reactive behaviors. In the navigation task, the route is computed approximately and outputs to motor actuators are given by low level reactive behaviors.

Three loosely divided categories are listed as architectural styles of hybrids. *Managerial styles* divides the

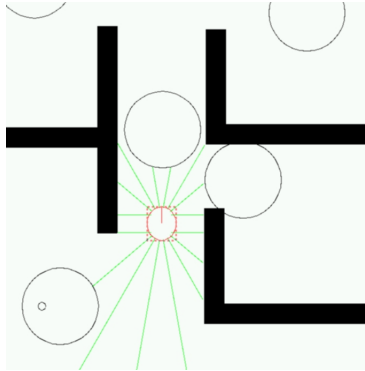
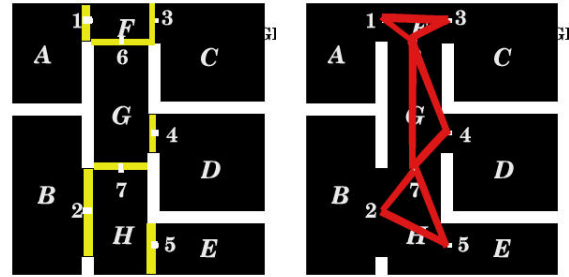


Figure 4: It is assumed that a situation in the figure cannot occur since obstacles are small.

behaviors according to the scope of control. *State hierarchies* use knowledge of robot state to distinguish between reactive and deliberative activities. Both of them are evolved from reactive paradigm. In this study, a *model-oriented style* is employed which is characterized by behaviors that have access to portions of the world model. Figure 3 shows the layout of the control architecture which resembles *Task Control Architecture*, employed by NASA especially in service robotics. **Task scheduling** is the upper-most module, where robot interacts with user, prioritize tasks, and optimize schedule. In the context of this study, since there is only one navigation task, this module is responsible from extracting desired goal position from the user. **Path planner** plans the route that robot should follow, using the global world model which is known a priori. The details of the world model and optimal path planning will be given later in this section. **Navigator** is responsible from the processing the state information. Navigator should extract the direction of next subgoal relative to the robot. At last, in the reactive layer, **Obstacle avoidance** behavior executes when there is a close obstacle ahead. It is employed for fast (reflexive) response to unexpected obstacles.

### Path Planning

Metric path planning procedure is applied in order extract the optimal path to the desired goal. The path planning algorithm decomposes the path into subgoals, in our case



(a) Convex polygons

(b) Graph nodes and edges

Figure 5: Meadow map and graph creation for the world in Figure 1. White areas are walls, yellow lines in (a) represents the boundaries of convex polygons and red lines in (b) demonstrates the graph edges.

into way points which are characterized by their  $(x, y)$  coordinates. The very first step is to partition the world into a structure amenable for planning. *Meadow Map* approach is employed to create a representation of the world and enable the optimal path extraction algorithm work since map is highly accurate and known in advance. Obstacles are unknowns of the maps however, they are small and they are assumed not to stuck the way. (Figure 4). Using meadow maps, free spaces are transforms into convex polygons where each polygon represents a safe region for a robot to traverse without colliding with any wall. Figure 5(a) shows manually created convex polygons, and safe region that robot can traverse through each polygon. Therefore, the problem is reduced to pick the best series of polygon to transit through. The next step of the algorithm is to specify candidate subgoal points according to the polygons. In figure 5(b), middle point of each border line segment becomes a node, and by connecting the edges between these nodes, an undirected graph is obtained. Thus, the navigation problem on continuous world is reduced to *graph search problem*. Finding optimal path is straightforward after the undirected graph is created. Closest nodes to the initial and destination coordinates of the robot is found. Then A\* search algorithm is applied to find the optimal path from initial node to destination node.

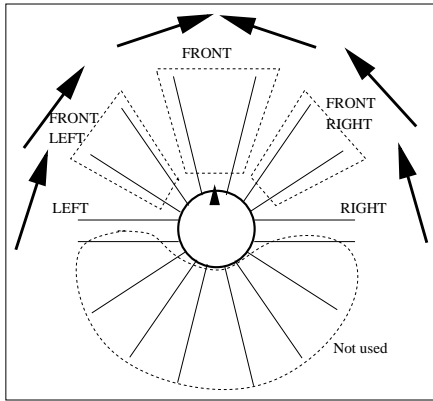


Figure 6: The usage of sonar sensor ring in selection of orientation of the response vector. The response rate of avoid obstacle behavior is independent of the direction of sensor group, rather it is discretely determined by the sensor which perceives maximum stimuli intensity.

### Navigation

Path-planner extracts the optimal path and a sequence of subgoal (node) coordinates. In each time step, navigator computed the direction of the subgoal, either graph node or final destination. Additionally, it controls whether subgoal point is inside the circular robot body region. In order to find the direction of the subgoal, and monitor the state of the robot, odometer data is employed. As it is mentioned in Section 2, player/stage simulator provides a truth proxy which gives precise location and orientation of the robot in any instant. If the subgoal is reached, navigator gives a signal to path planner and deserves for a new subgoal. If robot is inside the desired goal coordinates, a STOP flag is set to be true and speed of the robot is made zero. The output of the navigation module is a vector with constant magnitude.

### Obstacle Avoidance

Since there is no need for any cognitive computation in order to avoid from obstacles, *reflexive behavioral* [1] approach is followed. Response to the obstacle lasts only as long as sonar sensors give the stimuli of the obstacle, and the response is proportional to the intensity of the stimu-

lus. Thus, simple *reflexes* method is employed. The sonar sensors give the stimulus when the range of any object is smaller than a pre-defined threshold. *Discrete encoding* [2] scheme is used to specify the action, a response vector is selected from a vector set, which consists of a number of different directions and magnitudes.

Figure 6 demonstrates the employed sonar sensors. There are mainly 5 groups of sensors, LEFT, FRONT-LEFT, FRONT, FRONT-RIGHT, RIGHT. For any instant, the sensor group which perceives the greatest intensity will be active in a winner-take-all manner. The orientation of the winning group is selected as the orientation of the avoid obstacle response vector. For FRONT group, there are two different orientations in the figure, orientation is determined by the intensity of two sensor readings which locate in front of the robot. However, the magnitude of the response vector is independent of the winning sensor group. Response rate is directly set by the intensity of stimulus in 3 discrete steps.

The output vectors of navigation and obstacle avoidance modules are summed to obtain the desired orientation and speed. If there is no obstacle ahead, pure navigation will be on-line. Range and orientation of the closest frontal obstacle determines the effect of obstacle avoidance behavior to navigator. Figure 7 demonstrates a possible scenario where obstacle avoidance and navigation is coupled by summing up their vector outputs.

## 4 Experiments

Deliberative and reflexive behaviors which are described in previous section are tested independently before combined. Figure 8 shows a difficult test world, which is full of obstacles. As it is illustrated, robot can successfully deal with obstacles, while following its route.

## 5 Discussion

Hybrid deliberative/reactive paradigm is applied to the navigation task of the robot, given a map of the environment. Mainly two modules, path-planner and navigator, are employed in deliberative layer of the control architecture. Path-planner used meadow maps to generate an undirected graph of the environment, applied A\* algo-

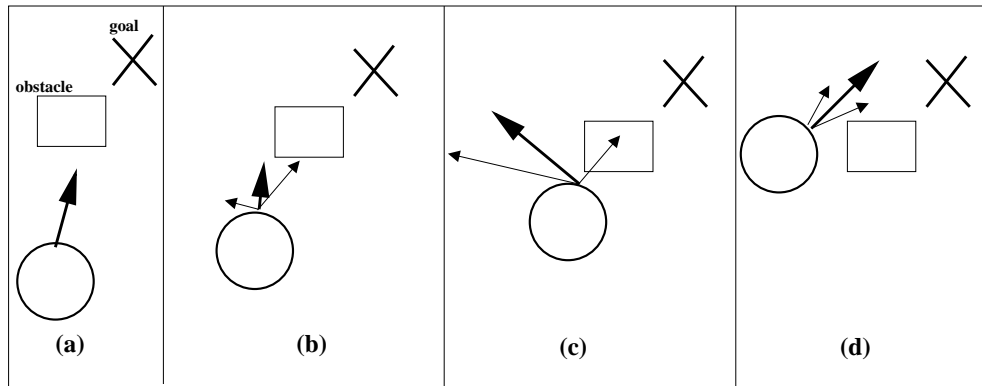


Figure 7: An imaginary scenario. In first step, since obstacle is very far away, only navigator vector defines the desired turn rate of the robot. In (b), obstacle starts to effect the result vector, but since stimulus is very low, the effect of the obstacle is not much. In (c), robot becomes close to the obstacle, thus reflexive obstacle avoidance determines the resultant turn rate. In (d), because a different sensor group perceives the obstacle, an avoidance vector with different orientation is summed up with the navigator vector.

rithm to extract the optimal route efficiently, and decompose path into subgoals. Navigator used subgoals in order to find an output vector to define desired speed and turn rate. As a module in reactive layer, obstacle avoidance behavior group its proximity sensors to discretely encode stimulus into *avoidance vectors*. The resultant turn rate is determined by summing up avoidance and navigator vectors.

As a future work, path relaxation method will be used in order to create the graph of nodes. Additionally, a more reliable proximity sensor type, infrared sensors, will be employed. The modular structure of the control architecture will enable the modification of obstacle avoidance implementation without changing any interface in other modules.

## References

- [1] *An Introduction to AI Robotics*, Robin Murphy, MIT Press, 2000.
- [2] *Behavior-Based Robotics*, Ronald C. Arkin, MIT Press, 1999.

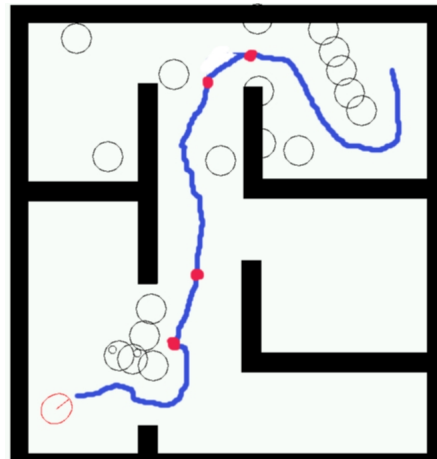


Figure 8: Trace of the robot's route on the way to the target. Red blobs demonstrates nodes of the graph where A\* algorithm is applied.