

# A Case Study for Behavioral Design

Emre Uğur

November 18, 2003

## Abstract

The aim of this work the design and implementation of a set of behaviors for an autonomous, mobile robot to accomplish a given task. The task is to navigate towards a target while avoiding from obstacles. To simulate the world, a simple 2-D simulator is employed. Behaviors are designed in an object-oriented approach. Potential fields methodology is proved to work well in unstructured environments and reactive paradigm is successfully applied to solve the problem in the unknown world.

## 1 Introduction

The objective of this study is the design and implementation of a set of behaviors for an autonomous, mobile robot to accomplish a given task. The task is first finding its target and home colors using the camera modality, moving to target while avoiding from obstacles around and returning back to its home as fast as possible. Furthermore, the robot has no a priori information about the unstructured world it is operating, and reactive paradigm should be applied while designing robot actions. Player/Stage software is employed as the simulation environment which provides a world with walls, obstacles and robot. A variety of sensors are simulated in a non-realistic way, where a camera, a ring of sonar sensors, a ring of infrared-sensors and four bumpers are used in our case.

In the following section, the simulation environment is described throughly. Design of behaviors are made in Section 3 and test results are given in the preceding section. At last section, conclusion of the

study is presented.

## 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-robot and robot-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

Environment is composed of a circular robot, a rectangular wall, obstacles with different size and shapes, and two colored objects which serve as food and home (Fig 1). 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. Since robot should view its targets anytime (food or home), obstacles are set to be invisible not to avoid the visibility of the targets. Robot has a circular shape with a diameter of 2.

---

<sup>1</sup>[www.sourceforge.playerstage/](http://www.sourceforge.playerstage/)

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

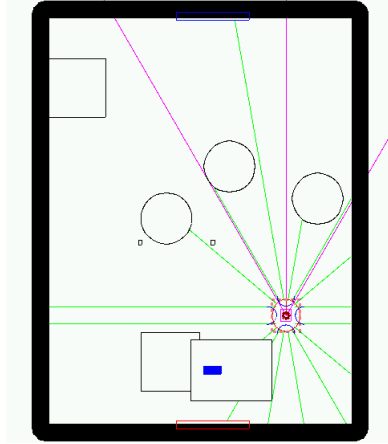


Figure 1: A sample environment snapshot with obstacles, food, home, and robot components

Figure 1 shows the environment with its components. Now let's describe each component,

### 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 (jumping above) 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.

### Sensors

Sensors are the indirect way of perceiving environment. There are 4 different sensors located on robot

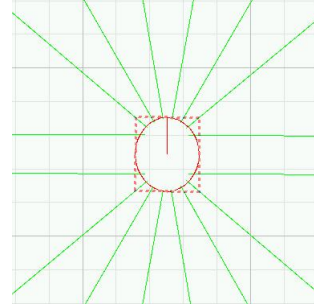


Figure 2: 16 sonar sensors with a range of 25 (robot diameter is 2).

body, which are 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 and infrared sensors, specular reflexivity and absorption problems do not occur in the simulation environment. Although sonar sensors are not used for the given task, I will explain each of the sensors one by one.

**Sonar Sensors:** 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 [4]. 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.

**Infrared Sensors (IR):** Other member of proximity sensor class is infrared sensors, which emit near-infrared energy and measure whether any significant amount of light is returned. Therefore, this sensor should give a binary signal, object presence or absence in the range of field. However, in simulator, the sensor is designed to measure the intensity of re-

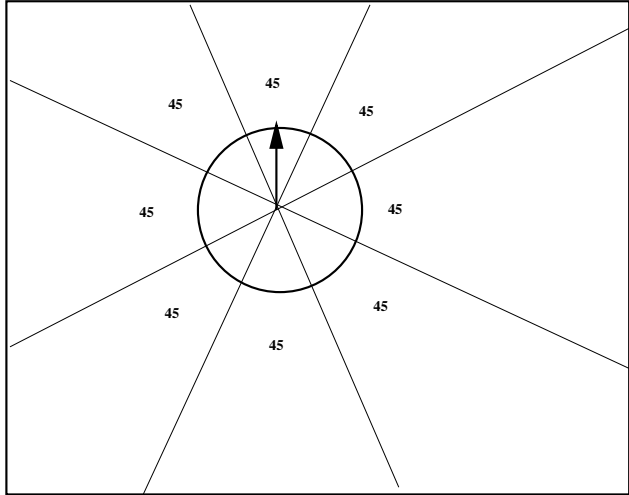


Figure 3: 8 infrared sensors with a range of 5 (robot diameter is 2).

flected light with perfect accuracy, thus it is a trivial task to extract the distance of the nearest object in its field of view. 8 infrared sensors which are homogeneously distributed over the robot body, which cover 360 degrees (Fig 3). Each infrared sensor has 45 degrees of field of view and approximately a range of 5. When compared with sonar sensor, infrared could be viewed as short-range sensor, and sonar as long-range. Thus no collision occurs on the view fields among sensors.

**Bumpers:** In order to understand if robot collides with any other object, bump sensors are employed widely. A two layer ring is usually used in bump sensors, when any collision occurs, they touch each other and an electrical signal is generated. In real life, the difficulty with bump sensors is the adjustment of sensitivity [1]. 4 semi-circular bumpers are located around robot bulk, which are activated when the robot came too close to any object. Since program stalls in any collision case, bumpers are activated just before collision. The 4 semi-circular bumpers are illustrated in Figure 4.

**Camera:** Computer vision is one of the most difficult fields of robotic perception. There are differ-

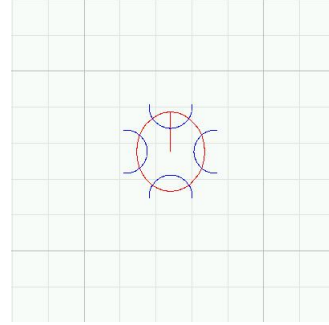


Figure 4: 4 semi-circular bumpers are activated when an object is too close.

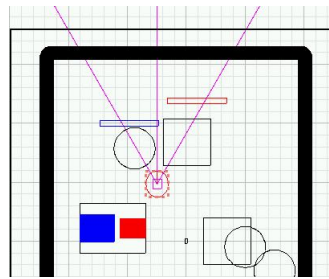


Figure 5: The field of view is illustrated in figure. The obstacles have no effect on vision.

ent tools to be employed like cameras, thermal sensors, X-rays; various problems to be solved, like image segmentation, range from stereo or shading etc.; and a number of state-of-art methods for each problem. While acquiring the whole picture of the world in the field of the view, the processing of this data and extraction of relevant information becomes very complex and slow. However, simulator provides a pan-tilt-zoom camera and a blob finder proxy which directly extracts the number and color of the blobs as well as the area of the blob surfaces. Segmentation for this case is a trivial task, since there are only a small number of different colors in the environment, and they are easily separable. Figure 5 shows the camera sensor and visualization of the output of the blob finder proxy.

**Infrared vs. Sonar:** Both sensors are proxim-

ity sensors. In reality they have different characteristics and they are appropriate in different environments for different tasks. Because of the uncertainties of real world data, and the intrinsic limitations and noisy measurements of the sensors, qualitatively different sensors are coupled for better perception of the world [3]. However, sensor fusion has no usage in our case since both proximity sensors have same functionalities with perfect accuracy. Furthermore, infrared sensors have a wide range of field when compared with sonar sensors which emit only sound rays. Therefore, I selected only sonar sensor for all proximity measurements (bumpers are used for very close obstacle detection).

### 3 Design the Behavioral System

Careful design of behaviors has crucial importance because overall performance of the system depends on the structure of it. Designing behaviors has two important deficits [1], it is more an art than science, and handling emergent behaviors is very difficult. Emergence means **more than sum of the parts**, where in robotics it means the unexpected results of the combination of primitive or complex behaviors.

There are a number of different methods for specifying and designing robotic behaviors. Mainly three of them are differentiated in [2]:

- Ethologically guided : Animal behaviors guide design steps.
- Situated activity-based : Some situations are defined and for each situation, an action is assigned to the robot.
- Experimentally driven : First a minimal behavior is constructed, it is tested and according to the evaluations, new modules or behaviors are added.

In this study, modular behaviors are tried to be developed as much as possible. Employing schema-based approach, each behavior has a perceptual-schema, a motor schema and may contain other behaviors. Although schemas will not be defined and

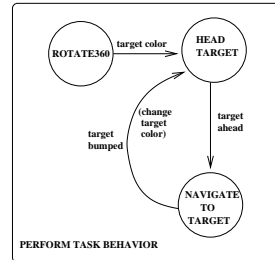


Figure 6: FSA of PERFORM TASK BEHAVIOR.

described explicitly, by employing Finite State Automata (FSA) [1], series of behaviors, releasers of each behavior, hierarch among different behaviors and inhibitors will be explicitly shown.

*Abstract Behavior* is the behavior which compose of many different behaviors. In our case, the top level abstract behavior compose of two behavior which controls different actuators. *PERFORM TASK BEHAVIOR* controls the position of robot, the speed and direction of the robot to accomplish the global task. As can be seen from Figure 6, this abstract behavior is composed of 3 sub-behaviors, some of which are again abstract. Robot first rotates around 360 degree to differentiate the colors of food and home. After food color is set as target color, robot continue rotating to come up with an orientation which heads to the target color. In *NAVIGATE* behavior, robot tries to approach the target color while avoiding from obstacles. When a bump occurs between target (currently food) and robot, target color is set to be home color and 6 behavior again employed to head to home and 8 behavior is activated to return back to its home. The modular usage of 6 and 8 behaviors should be noted (when these are thought as schemas, the color information become argument to them). Figure 7 shows how 7 behavior controls the actuator of camera sensors.

#### 3.1 Camera Trace Target Behavior

Active sensing [1] is employed in this study to follow the target direction. The target location, either food or home targets are unknown. The only available in-

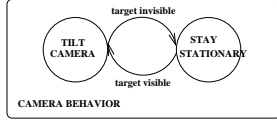


Figure 7: FSA of CAMERA BEHAVIOR.

formation is the direction of target (target color) for the robot. Robot extracts the color information by employing its ptz camera, thus the location of the target is always perceivable since other objects in the environment are defined to be invisible. While approaching to the target, robot may face with different obstacles and should change its path to avoid from them. While the robot modifying its path, since camera only heads towards the robot’s front direction, target may be disappear from the camera view. Thus, I used active sensing capability of the pant-tilt-zoom camera, to monitor the trace of the target in all conditions. In order to establish the requirement, in each step, the center of the target color blob is extracted from camera sensor. According to the distance between the center of camera and center of the blob, camera is tilted. If blob is at the left side of the camera screen, camera is tilted to left and if blob is at right, camera is tilted to right. For the sake of simplicity, it is sufficient to rotate camera a constant degree per second. However, simulator constrains the maximum allowable tilt angle to  $pm$  100 degrees, robot may miss its link to target, target may disappear from the screen. In such situations, last rotation angle of the robot just before missing the target is stored and used as the target direction. The direction information is used when computing the attraction vector of the target in Section 3.2. Figure 7 shows the FSA for this behavior.

### 3.2 Navigate Behavior

Potential fields method is employed to control robot movements. Figure 8 shows the two constituents of the NAVIGATE behavior. The output vectors of *AVOID OBSTACLE* and *GOTO TARGET* are summed to specify the direction of movement. How-

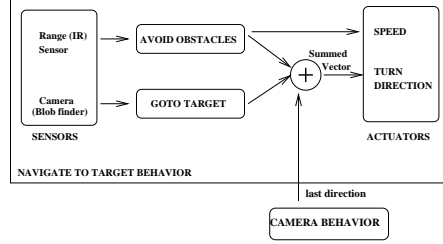


Figure 8: NAVIGATE BEHAVIOR.

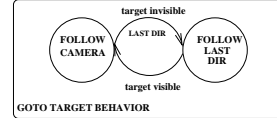


Figure 9: FSA of GOTO TARGET BEHAVIOR.

ever, the desired speed of the robot does not set by directly magnitude of the vector. Instead velocity is determined by the density of obstacles ahead the robot. When there are no obstacles around, robot moves with its maximum speed, and when an obstacle is very close to the robot, it moves slowly to enable robot make desired rotations. Last direction information comes from CAMERA behavior, and is used when target direction cannot be extracted from blob finder proxy.

#### Goto Target Behavior

The direction of the target, either food or home, will be determined by camera direction whenever the target can be seen by camera (Figure 9). Color is used as the affordance [1] measure for the attraction since it is directly perceivable and represents a potential for robot movements. The size of the blob represents closeness to the target. Since field of the view of the camera is limited, target may be invisible to the camera (and robot) for some situation (ie. when it is behind the robot). For such situations, as it is stated in Camera Trace Target Behavior (Section 3.1), last rotation angle (direction) is stored, and an attraction vector is composed using the opposite of the stored

direction. Thus, in all situations, an attraction field vector can be computed. Constant magnitude is employed for the attraction field vector. It is not a true field vector since when robot miss the track of the target, an attraction vector is computed dynamically and approximately, assuming target locates in some direction. There may be an alternative method to tackle with losing target track problem. When target is in the field of view, a vector to the target location could be computed relative to robot. This vector, coupled with odometer functionalities, then can be used to approximately determine target direction. This method is not followed from two reasons. First in real robots when errors are accumulated for a long among of time, the estimation of the location of the target may be misleading. The other reason is the complexity of the calculation of orientation changes in robot.

### Rotate behavior

Robot has no prior knowledge about the environment, as well as the food and home locations. Colors of the food and home is the only cue for the robot to accomplish its task. Robot only knows that it has a starting point which is closer to its home. Since home and target are in the same size and in different colors, rotation of the robot should enable it to decide the colors of walls. Rotation behavior is composed of two sequential behaviors. In the first behavior, robot makes a 360 degree rotation. In each step of rotation, colors and areas of the corresponding color blobs are extracted. For each color, the maximum area is computed by comparing the sequential measures. At the end of the 360 degree turn, the color with a greater surface is referred as home color and other color is assigned for food. Odometer is used in the inhibition of this first sub-behavior and in the release of second one. After the 360 degree turn, robot keep on rotation in order to head to the target. When the camera sense the target color, robot stops turning.

### Avoid Obstacle Behavior

Quadratic drop-off is used for the potential field of the obstacle. The vector is determined with the us-

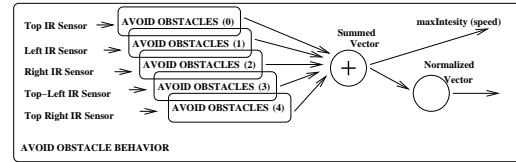


Figure 10: OBSTACLE AVOIDANCE BEHAVIOR.

age of proximity sensors (IR sensors) as perception. However, since more than one sensor are employed, the tangential fields should be combined for the extraction of total tangential field. There are 2 different approaches to combine multiple sensor case ([1]). In the more common approach, a behavior is instantiated multiple times, for each sensor. Other approach is to combine sensor data in one behavior. First one is desirable, since the function is independent and will not introduce side effects to the main program. If first approach is used, addition of more sensors would be very simple, not changing structure of the program or other modules(object oriented approach). However, if avoid obstacle behavior is instantiated for each sensor, the summation of the vector in main program will have undesirable results for big obstacles since whole obstacle avoidance behavior will dominate general behavior. Therefore, there should be a hierarchical summation procedure. First all behavior vectors summed, a direction is found. Since the magnitude is the main reason for an unwanted domination, magnitude is directly assigned to be the maximum from different sensor instantiations. Figure 10 illustrates this behavior. Especially for this project in the highly unstructured environment, a crucial part of the design of this behavior is the selection of tangential vector direction with respect to robot. The density of the obstacles is computed using infrared sensors and robot selects the direction of less dense area.

Because of the difficulty, obstacle avoidance behavior is designed as experimentally driven [2]. In experiments, when obstacle is very close, tangential vector field may result in the collision of the robot to obstacles. Therefore, when robot is too close to the obstacle (ie. infrared intensity value is very high) re-

pulsive field vector is employed to make sharp turns and avoid from collision. Since the response lasts as long as stimulus exists and response magnitude is directly proportional to the stimulus, this behavior can be classified as a *reflex*. Additionally, because the movement direction of robot depends on the orientation of the response, it can also be located in *taxes* type behaviors.

## 4 Test Runs and Results

Following the design steps in [1], I test each behavior independently for certain contexts. Figure 11 shows a difficult test world and hand-drawn trace of the robot in this environment. <sup>3</sup> As it is shown tangential vector field and storage of last target direction before missing it, are successfully applied to local minima conditions.

## 5 Conclusion

A set of sequential and cooperative behaviors are designed and implemented in this project to tackle with the unknown environment. The navigation to a target employing affordance measure is applied while avoiding from obstacles using taxes and reflexes. Potential fields methodology is proved to work well in unstructured worlds. Tangential vector field selection for obstacle avoidance and direction storage for target position when target view not available are successfully applied to enable robot escape from local minima.

## References

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

---

<sup>3</sup>Full movie can be obtained from [www.ceng.metu.edu.tr/e116526/test.mpg](http://www.ceng.metu.edu.tr/e116526/test.mpg).

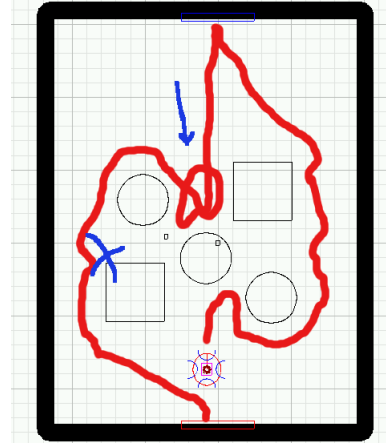


Figure 11: Robot trace in an unstructured environment, robot is able to tackle with local minima. The oscillation on the top (circular movement) is due to the erroneous direction storage of target. Robot collides with the left rectangular obstacle, where a sign is drawn.

- [3] Elfes, A. Using occupancy grids for mobile robot perception and navigation. Computer , Volume: 22 Issue: 6 , June 1989.
- [4] Elfes, A. Sonar-based real-world mapping and navigation, Robotics and Automation, IEEE Journal of [legacy, pre - 1988] , Volume: 3 Issue: 3 , Jun 1987.