

# Koolio: An Autonomous Refrigerator Robot

Sara Keen  
maxpower@ufl.edu

Wenxing Ye, Lavi Zamstein,  
Blake Sutton, Gene Shokes,  
Gorang Gandhi

Eric M. Schwartz  
ems@mil.ufl.edu

Antonio Arroyo  
arroyo@mil.ufl.edu

Machine Intelligence Laboratory (MIL)  
MAEB 325, Building 720  
University of Florida  
Gainesville, FL 32611-6300  
00-352-392-2541

## ABSTRACT

In this paper, we describe the electrical and mechanical makeup of an autonomous delivery robot. We will discuss what processors were used and the tasks each one performs, the sensors that were utilized to evaluate the robots environment, and the steps the robot takes to complete the desired actions.

## Keywords

I2C, PID control, reinforcement learning, Q-learning.

## 1. INTRODUCTION

Koolio is a combination of R2D2 and a vending machine. His home is the Machine Intelligence Laboratory at the University of Florida. Faculty and students working in the Mechanical Engineering and Aerospace Building B (MAEB) can simply access the Internet to request that Koolio deliver a snack or a cold drink to their office. Koolio then leaves his docking station and proceeds down the hallway, using cameras to find the correct office number, avoiding any obstacles in his path. After a successful delivery he returns to the docking station to await the next order and the process begins anew.

Koolio has two modes of operation: one in which every decision he makes is based purely on the current sensor data, and one in which every decision is based on previous information he has collected about his environment. These two modes, and the electrical and mechanical systems that facilitate their operation will be discussed in detail in this paper.

## 2. INTEGRATED SYSTEM

The majority of the data analysis and decision-making take place in a single board computer that serves as the heart of the integrated system. Two smaller processors, an Atmega128 and an Atmega8, are used to monitor sensors and perform motor control. Figure 1 shows the hierarchy of these microprocessors. Although it may have been possible to control Koolio using only the single board computer, it was more practical to use multiple processors each dedicated to a small number of tasks. This ensured speedy sensor readings and decision-making, and reduced the risk of Koolio losing all power at once.

## 2.1 Single Board Computer

With similar capabilities to a low-end desktop computer, but a vastly reduced size, the NOVA-7896 CPU board exerts high-level control of decision-making and behavior. The current installed operating system is Linux Ubuntu, selected for its relatively low overhead and ease of use. With a Pentium III processor, USB support, and VGA monitor support, this embedded board is ideal for the robot's more complicated tasks, including the vision processing, wireless order taking, and machine learning. The computer runs Ubuntu 5.10 and is powered by 12V and 5V power supplies. Communication with the Atmega128 is achieved via a serial connection that is set for a Baud rate of 9600. The LCD screen that serves as Koolio's face is connected to the VGA output of the board. A USB hub is plugged into one of the two USB ports, which

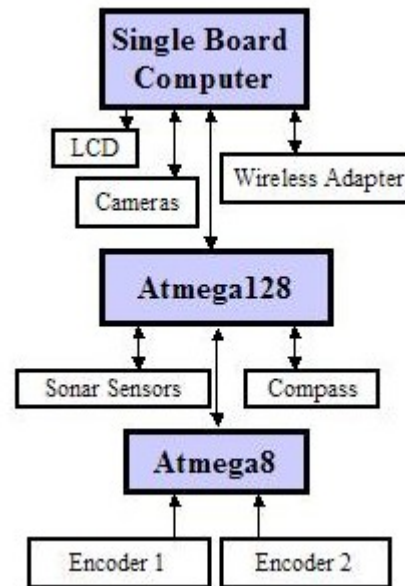


Figure 1. Block diagram of integrated system

allows both cameras and the wireless adapter to all connect to the board. All programming for this computer was written in C. When the main program is executed Koolio begins to communicate with the Atmega128, monitor the cameras and can receive wireless orders. During normal operation, the main program is constantly comparing sonar readings to previously determined thresholds and

performing image processing to determine if Koolio has reached the correct office when making a delivery. Based on the sensor readings the single board computer calculates the appropriate motor control commands and sends these to the Atmega128. Thus, the main computer performs obstacle avoidance. While this program is being executed the LCD is updated accordingly with pictures of various facial expressions that depict what Koolio is 'thinking.'

## 2.2 Mavric-IIB

The Atmega128 resides in a Mavric-IIB development board. The Mavric board is an appropriate microcontroller for any robotic project because it is equipped with numerous I/O ports, analog inputs, an onboard RS232 chip, and uses a 16MHz oscillator. The primary responsibilities of the Atmega128 are to continuously read incoming sensor data from the sonar sensors, digital compass and Atmega8, and send commands to the motor drivers. The sonar rangefinders, digital compass and motor drivers communicate with the Atmega128 via an I2C bus. The I2C interface requires only one power, ground, clock and serial data line per device. Every I2C address can be programmed with a unique 8-bit address, making it possible to have multiple devices on one bus. To access a device the processor places the device address on the data line, and only the correct address will obey the following commands.

The Atmega128 records the incoming data and transfers it serially to the embedded board, which interprets the data and determines Koolio's next action. This decision is sent to the Atmega128 through the serial connection, and the corresponding commands are sent to the motor drivers. The Mavric board sends data in 10-byte packets that contain the current sensor readings, as seen in Figure 3a.

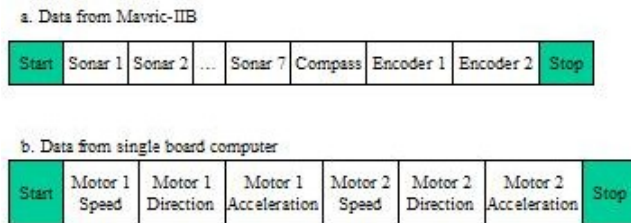


Figure 2. Serial data packets

After each packet is received the main computer decides what commands to send each motor and transmits this data serially to the Atmega128. The 6-byte packet is shown in Figure 3b.

Although the main computer controls steering, the Atmega128 is programmed to automatically stop Koolio if the sonar sensor readings are too far below the threshold values. This program acts as a backup obstacle avoidance routine and prevents Koolio from being damaged due to erroneous calculations.

## 2.3 Python Board

A Python development board contains the Atmega8 controller that continuously reads both encoders. The Atmega8 uses an 8MHz oscillator and has 3 I/O ports available on the Python Board. This board was selected for its small size and ease of use. The 8MHz clock is sufficient for the speed at which the encoders are read as long as no other computing is taking place.

An interrupt service routine that is executed every 80 us reads the encoder inputs and keeps track of each encoder count. A separate

interrupt service routine that is entered approximately every 8ms sends the Atmega128 the encoder count total since the previous interrupt. The Atmega8 outputs this data onto an 8-bit I/O port then sets a separate I/O line low to initiate an external interrupt in the Atmega128. The I/O line that is lowered depends on which encoder data is currently on the 8-bit port, and the Atmega128 catalogs the data according to which external interrupt occurred. This ensures that the Atmega128 always knows the speed of either motor.

## 3. MOBILE PLATFORM

Koolio stands at nearly 5 feet tall, weighs approximately 60lbs and has a base diameter of 20in. His height is necessary in order for the cameras on either side of his 'face,' to read room numbers. To minimize the swaying of the cameras while driving, Koolio's center of gravity is as low as possible. The motors, batteries, docking circuitry and motor drivers reside in his circular base. These items and the frame make up most of the robot's weight. The base is shielded by an aluminum skirt, which serves to hide and protect electronics as well as muffle the noise of the motors.

The refrigerator is mounted atop the base. The single board computer and the Atmega128 are stored in a large plastic box on the back of the refrigerator. These items were placed higher on the platform to easier access and to minimize the length of wires running from the higher sensors.

Two aluminum pipes support the LCD, which is mounted on a welded frame. The wireless adapter is mounted behind the LCD, at the highest point possible on the robot.



Figure 3. Front view of Koolio

## 4. ACTUATION

Koolio navigates using two 12 V motors mounted on the base of the platform. The motors share a single axle, which serves to keep them from moving under the force of the robot's weight. Because Koolio has a circular base, this allows him to rotate in place about his origin.

## 4.1 Electrical Considerations

In order to prevent the processors from being effected by current spikes, a separate battery powers the refrigerator and motors. It is impossible to prevent the rapid change in current because the motors may need change direction or speed quickly if an obstacle is spotted nearby. Although these two batteries share a common ground, the two power supplies are optically isolated from each other, which prevents damage to the circuitry [1].

## 4.2 Motor Control

The two 12 V motors are not matched; when the same signals are sent to both they turn at different speeds. Using PID control for both motors compensates for this. Koolio uses one optical encoder per motor to form to closed-loop feedback systems as shown in Figure 2. All motor control calculations take place within the Atmega128.

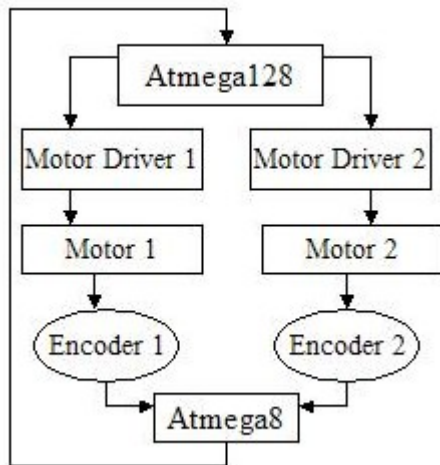


Figure 4. Block diagram of motor control system

### 4.2.1 Optical Encoders

Two optical encoders are used to determine the exact speed at which both motors are turning. The US Digital S1-50 encoders were each connected to one of the gears in the gear train of each motor. The encoders use two-channel quadrature output and have fifty counts per revolution. The motors are geared down such that this equates to 416 counts per wheel revolution, leaving almost no room for error. By monitoring the two channels, it is possible to know which channel is leading, and thus, the direction of the wheels rotation. The outputs channels use TTL levels and are input directly into an I/O port on the Atmega8. The encoder inputs are read every 80us and compared to the previous inputs to find the direction of rotation, if any. The total number of encoder counts may be positive or negative, depending on the direction of the wheel rotation. The Atmega8 always keeps a running sum of the total number of encoder counts, and transmits this total to the Atmega128 every 8ms. After transmitting the total, the sum is set to zero and a new sum is found for the next 8ms period. This allows the average speeds of both motors to be compared every period. All motor control calculations take place within the Atmega128.

### 4.2.2 PID Motor Control

PID stands for Proportional Integral Derivative. PID control is a method of smoothly adjusting motors to operate at a certain speed, or a set point. This set point is determined by averaging the current speed of both motors. When the encoders measure that both motors are operating at the same speed Koolio is traveling in a straight line. If in the process of straightening his path causes Koolio to then travel straight, but in the wrong direction, his sensors will alert him to change his course and he will correct his path until it is correct. Koolio uses the velocity PID algorithm, meaning that the control loop calculates only the offset to be added to the current motor commands. Thus, if there is no difference between the current speed and desired speed of either motor, the offset is calculated to be zero, and the motors do not alter their speeds [2]. The motor control program used in Koolio will calculate the offset for each motor with a separate PID loop and adjust the speed commands being sent to the motor drivers by adding these offsets. The offset is comprised of three terms and can be represented by the following equation:

$$\text{Offset} = K_p * (\text{P-term}) + K_i * (\text{I-term}) + K_d * (\text{D-term})$$

The coefficients  $K_p$ ,  $K_i$  and  $K_d$  are decimals between zero and one that are adjusted to alter the effects of each term on the output.

To calculate the proportional term, the Atmega128 finds the error or the difference between the set point and the actual motor speed. This term can be positive or negative depending on the speed of the motor as compared to the set point. The proportional term increases overshoot and decreases the rise time and steady state error of the response [3]. The settling time is not significantly effected by the proportional term. The coefficient  $K_p$  can be adjusted to increase or decrease the affect of the term. It was found that the right motor worked best with a coefficient of .55 and the left with a coefficient of .9. This is because the left motor is considerably weaker.

The integral term is used to minimize the past error that has accumulated over time. It is calculated by adding up the error for certain period of time and multiplying this sum by a coefficient that averages, or integrates, this error over time. The set point is then subtracted from this average to calculate the difference. This term is then multiplied by the integral coefficient  $K_i$  to adjust the terms effect on the output. The integral term decreases the rise time of the response and increases percent overshoot and settling time. It is very effective in eliminating the oscillations that cause steady state error. Both the right and left motor worked best with the integral coefficient set to .1.

The derivative term is found by subtracting the current error from the previous error. This term will also be positive or negative as needed. The response to a change in a system can be limited by this term, as which prevents large errors from occurring in the future. The derivative term decreases the overshoot and settling time of the response. Testing proved that the left motor performed best with a derivative coefficient of .25, and the right motor performed best with a coefficient of .45.

By experimenting with different combinations of coefficients it was possible to tune both motors precisely enough for Koolio to maneuver through small doorways and crowded rooms.

## 5. SENSORS

Koolio is equipped with numerous sensors, including seven sonar rangars, a digital compass, and two cameras. These sensors allow

Koolio to always have a clear idea of his surroundings and the walls on either side of him.

## 5.1 Sonar Rangers

Seven Devantech SRF08 Ultrasonic Rangers are mounted in various location on the platform. These sensors have a range of six meters and take one reading every second. These sensors were selected because they are I2C compliant and do not use I/O lines. It is not necessary to read the sonar sensors continuously, meaning all devices can share one bus. One sensor in the back prevents Koolio from backing into walls while maneuvering out of offices and leaving the docking station. Four sensors mounted near the base of the refrigerator monitor the 180° section in front of the robot. By reading these sensors it is possible to detect any obstacles that Koolio could possibly drive into. An additional sonar sensor is mounted below either camera facing outward. These sensors make it possible to know the exact distance of the image being captured with the cameras. Furthermore, they alert Koolio if there is a wall on either side of him, making it possible to determine when he is next to a door or in a hallway.

## 5.2 Digital Compass

A Devantech CMPS03 Compass allows Koolio to know what direction he is facing. The compass uses a Philips KMZ51 magnetic field sensor to compare the alignment of the robot with the Earth's magnetic field. The compass is also assigned a unique address and connected to the I2C bus. The compass outputs 1 byte of data to represent the direction the robot is facing. This results in 3-4 degree accuracy of readings. Koolio compensates for the error in directional reading by attempting to stay in the center of any hallway he travels through. If he believes he is traveling in the correct direction but is moving further from one wall he can realign himself and despite compass error.

The compass is mounted atop the refrigerator to minimize noise from the motors. Furthermore, the refrigerator is a fairly level and stable surface, ensuring the most accurate compass readings possible.

## 5.3 Bump Sensors

During the testing stages of the artificial intelligence programming the robot may ignore other sensor readings and run into a wall or other obstacle. For this reason, it is necessary to have sensors to detect when anything comes into contact with the robot's circular base. A bump ring composed of several metallic whiskers will be used to detect contact at any point. When Koolio collides with an object a whisker will be compressed, completing a circuit and alerting Koolio that a collision has occurred. The circumference of the base will be divided into six sections, each with one whisker. This will allow the robot to have a fairly accurate sense of the point of contact.

## 5.4 Cameras

Two Creative Video Blaster II Webcams are used to identify room numbers as Koolio navigates hallways. The main computer receives sequences of images from either camera. A sample image can be seen in Figure 5.



**Figure 5. Sample image from one of Koolio's cameras**

The program in the main computer will locate the room number plate in the image and separate it into several blocks. Standard digit templates stored in memory are used to match the segmented blocks to determine the most likely room number. Then the detected room number is compared with the desired room number to see if there is a match.

### 5.4.1 Image Processing

The image-processing program was developed in Matlab to simplify debugging. The final program is written in C, as are all programs running on the main computer. Difficulties encountered when analyzing an image include blurriness from the resolution and focus settings on the camera and from poor lighting, differences in the size and angle of plates, and disturbances from other objects, such as name plates and door frames. The program can lessen the effects of these problems by searching for plates that are the correct distance from a doorway and match the width-to-height ratio. By focusing on the plate and wall, as well as the plate and the room numbers, it is much easier to locate the plate in an image.

#### 5.4.1.1 Room Number Recognition Algorithm

The recognition algorithm makes the following assumptions based on real-world knowledge: the width-to-height ratio is the same for both plates and digits, the color of the number plate does not change, the camera is not angled too much, and the number plate is not located on the border of the image because of the position of the camera.

It is first necessary to define the color of the number plate in a way that the computer can understand. A covariance matrix  $c$  is introduced to represent the core of the color. The value of  $c$  will be the result from analyzing the marked image shown in Figure 6.



**Figure 6. Image marked with white to represent training data for covariance matrix  $c$ .**

All the RGB values of pixels in the white areas are our training data. With supervised training, we can get a rather reliable covariance matrix  $c$  using follow expression:

$$c_i = \begin{bmatrix} r_i - \text{mean}(r) \\ g_i - \text{mean}(g) \\ b_i - \text{mean}(b) \end{bmatrix} [r_i - \text{mean}(r) \quad g_i - \text{mean}(g) \quad b_i - \text{mean}(b)], \text{ where}$$

$$\text{mean}(x) = \frac{1}{N} \sum_{i=1}^N x_i$$

Finally, we have:

$$c = \left[ \frac{1}{N} \sum_{i=1}^N c_i \right]^{-1}$$

The next step is to cut the edge off of the image to eliminate the edge effect caused by the camera.

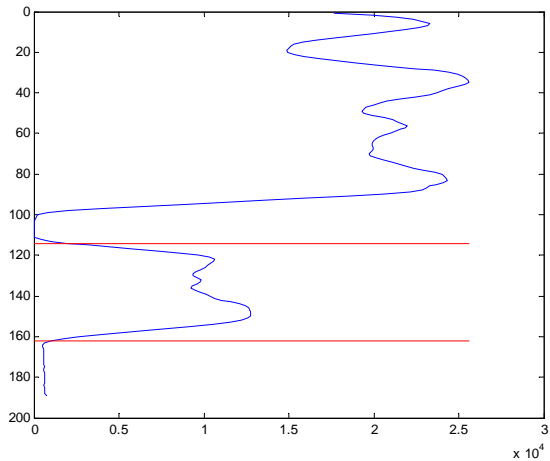
Then, a threshold  $t_d$  is set. All pixels with

$$d = [r \quad g \quad b] c \begin{bmatrix} r \\ g \\ b \end{bmatrix} > t_d$$

are set to be part of the background.

With this threshold, it is possible to introduce some noise control. We can calculate all the connected areas and throw away those with fewer pixels than the set amount.

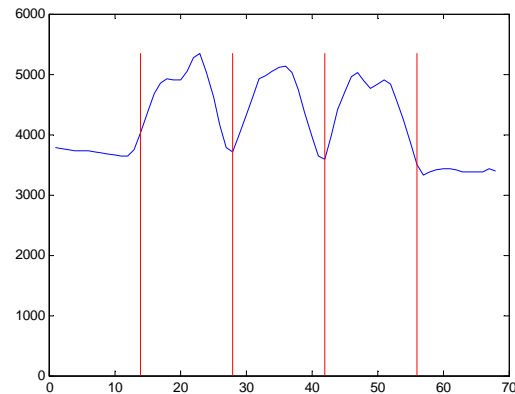
Next the image is converted to grayscale. The sum of the grayscale values of pixels is calculated for every row. Figure 7 shows a plot of the values.



**Figure 7. Plot of sums of grayscale pixel values in every row**

The number plate corresponds to the lower section of peaks. By locating the peak, the vertical location of the number plate is found. By summing and plotting the grayscale values of the pixels in each column, we can find the horizontal location of the number plate as well. The location of the plate is then fixed.

Similarly, by summing the pixel values for every column the digit segmentation can be determined. An example of this is shown in Figure 8.



**Figure 8. Plot of sums of grayscale pixel values in every row**

It is obvious that every peak in the plot represents a digit. Based on the assumption that each digit has a fixed height-to-width ratio it is possible to determine exact digit areas. By using a simple template matching method on these areas, it is possible to decipher the room number.

## 6. BEHAVIORS

As mentioned before, Koolio has two modes of operation. In the first mode, or 'sensing mode,' the robot remembers nothing from the past, despite following the same path on every delivery. Every decision is made as if for the first time, based purely on what data is collected about the environment. In the second mode, Koolio uses artificial intelligence to make choices. When Koolio is operating in the 'intelligent' mode his decisions will be based on the outcome of decisions that were made in the past.

## 6.1 Operation is Sensing Mode

A typical delivery in this mode of operation would take place as follows. The robot begins parked on the docking station, recharging both batteries and listening for an order to arrive by using the wireless adapter. One of the professors in an office down the hall is thirsty, and logs onto Koolio's wireless server. There the professor can peruse the selection of refreshments available and enter their room number. After the order is submitted, Koolio receives the order and immediately leaves the docking station. He is able to navigate out of the lab by using the sonar sensors to avoid obstacles and detect walls on either side. He is programmed to turn right as soon as there are no walls in the way and proceed straight until the sonar sensors alert him that he is in a doorway. At this point the robot is entering the hallway. Koolio will always turn right, and begin proceeding down the center of the hallway, examining room number on both sides. The sonar sensors will alert Koolio if any obstacle is blocking his path, and he will attempt to maneuver around it. If he cannot pass the obstacle, he has reached the end of the hallway and will then continue in the same manner in the opposite direction until the correct room is found. Koolio is able to travel in a straight line down the center of the hallway using the digital compass. After passing an object he can realign by matching his direction with the correct angle. When the cameras detect the correct room number, Koolio stops moving forward and turns to face the doorway. He proceeds into the office, using the sonar sensors to avoid walls. Then he waits for the client to press a button to signal they have taken their order from the refrigerator. Koolio then makes a 180° turn and reenters the hallway. He will return to the Machine Intelligence Laboratory in the same way he found the office, by always turning the same direction and searching for a matching room number. When he enters the lab, he will follow a similar procedure to find the docking station, using the sonar sensors and compass to know where he is in the room. Then he drives onto the docking station to recharge and await the next order.

## 6.2 Reinforcement Learning

Reinforcement learning is a method of learning by means of rewards and punishments. The learning agent will seek choices that result in high rewards and avoid actions that result in low rewards or punishments (negative rewards). In this way, an agent will learn to follow the decision path that results in the best possible reward [4]. Q-Learning was the reinforcement learning method chosen for use in Koolio. This method uses learned action value functions to approximate the optimal action value function independent of the current policy [4]. Q-learning was chosen for ease of use and availability of reference materials.

## 6.3 The Learning Process

The entire learning process, from development to actual execution using a robot, can be divided into three steps.

### 6.3.1 States and Actions

For a mobile robot, the states are simply the set of all sensor input. Because most of the sensors used on Koolio have analog outputs, there is the potential for a very large number of possible states. Therefore, states must be defined using ranges of sensor values. Table 1 shows some examples of states and the sensor inputs defining them.

State	7. Sensors
Wall on Left	Left sonar < Threshold
Bump on Right	Right bump sensor
Middle of Hallway	All sonars > Threshold
Sign sighted on left	Left sonar < Threshold, Left camera registers sign shape

Table 1. State Examples

### 6.3.2 Simulation

Reinforcement learning in episodic tasks requires a very large number of repetitions to learn. Along with this, many repetitions on the platform can result in wear of parts, and the testing area of a hallway may not always be available since it is used on a daily basis. Because of these factors, the initial parts of learning must be done in simulation. Fortunately many simulations can be run in the time of a single real robot episode. By developing an optimal policy in the simulated environment, much of the time of real robot learning can be done before involving the actual robot [4].

### 6.3.3 Real Robot Learning

Once the simulation has reached an optimal policy, it can be brought to the robot to continue the learning. Because a good deal of learning has already taken place, this phase of the learning process is much faster than if the learning was done solely in the real environment. Despite the shortcuts of using a simulator for the initial learning process, this phase of real robot learning is still the most time-intensive, as episode runs of the robot can take several minutes instead of the accelerated time used in simulation. Because the policy is already refined, however, only a relatively smaller number of episodes are required for reaching a new real environment optimal policy.

## 8. FUTURE WORK

When set to the reinforcement learning mode, Koolio can be transferred into another environment with similar makeup and learn fairly quickly how to operate optimally in the new environment. For instance, if Koolio was moved into another hallway with the same physical characteristics (such as wall color and room number signs), it could learn to find a room in much less time than it took to initially learn how to navigate a hallway.

The next mechanical development for Koolio will be a robotic arm that will remove the correct item from the refrigerator when making deliveries. The arm could also be used for other tasks such as opening doors or pressing elevator buttons. With the addition of an arm would come new sensors, to detect pressure or the color of the item it picks up, as well as new behaviors. Other plans include installing a voice recognition chip in the robot, and teaching him several words. When taught a basic vocabulary, it will be possible to place orders and give behavior and steering commands verbally.

## 9. ACKNOWLEDGMENTS

Our thanks to Brian Pietrodangelo and Kevin Phillipson for their work on the original Koolio design and the picture used in Figure 3, and the University of Florida Machine Intelligence Laboratory (MIL).

## 10. REFERENCES

- [1] Flynn, A. M., Jones, J. L., and Seiger, B. A.. *Mobile Robots: Inspiration to Implementation..* A. K. Peters, Natick, MA, 1999, 193-263.
- [2] Tham, M. *Discretised PID Controllers.*  
<http://lorien.ncl.ac.uk/ming/digicont/digimath/dpid1.htm>.1998.
- [3] Carnegie Mellon University. *The Characteristics of P, I and D Controllers.*  
<http://www.engin.umich.edu/group/ctm/PID/PID.html>. 1997.
- [4] Zamstein, L. *Koolio: Path Planning Using Reinforcement Learning on a Real Robot Platform.* University of Florida, 2006.