Mailbot:

An Autonomous Mail Collecting Robot

Frank Bond
EEL 5666
IMDL
Spring 1997

Table of Contents

Abstract

The focus of this class is to design an autonomous agent while being introduced to the ideas of machine intelligence. The primary goal is to learn that intelligence comes from interaction with the agent's surroundings. To learn how to create a rudimentary intelligent machine and have it interact with its environment while performing some useful function is the goal and the following paper will provide an account of the experience.

The function of the mailbot is to autonomously retrieve outgoing mail from an office setting. It is assumed that receptacles for the mail will be placed outside of each office along a path to be traversed by the mailbot. As the mailbot passes under these receptacles, a hook will grab the front edge of the receptacle and empty the contents into a payload area on the mailbot. The mailbot will continue emptying these receptacles until it reaches the end of its path in the mail room. Here the mail will be emptied and the mailbot will be recharged.

To allow the mailbot to perform its task, line following sensors, object avoidance sensor, bump sensors and a payload capacity sensor are incorporated into the mailbot's design. These sensors will allow the mailbot to execute its behaviors: path following, collision avoidance, bump relief, and payload dumping.

Along the way, this paper will discuss short comings and exceeded performance as compared to the original design.

Executive Summary

The goal of this project was to design an autonomous agent that would perform a certain task. The task selected was that of a mail retrieval system designed for an office environment. The mailbot was intended to navigate a course around the interior of an office building and retrieve outgoing mail from receptacles placed outside of the offices.

To do this, the mailbot performs certain behaviors. The line following behavior allows the mailbot to follow a path around the office building. This path leads the mailbot under mail receptacles that are to be emptied by the mailbot. The mailbot will also perform collision avoidance. Infrared light sensors mounted on the front of the mailbot allow the robot to sense an obstacle in its path. If detected, the mailbot avoids the impending collision.

In the case that an object gets through the IR array, the mailbot is equipped with a bumper that will detect when contact has been made. The mailbot will then retreat on a semi-circular path and choose a new direction so as to avoid contact with the obstacle again.

The mailbot also has a dumping behavior that is enacted when the payload reaches capacity. A sensor under the cargo basket detects when the payload has reached this threshold limit and the dumping mode is started.

Introduction

Machine intelligence means many things to many people, from the purely software side to the purely hardware side and everywhere in between. The focus of machine intelligence for this class is the use of sensors to intelligently interact with the environment to perform the desired task.

The mailbot is an attempt to create an autonomous agent that uses sensors to interact with its environment and perform some task. The task that was chosen is that of a mail retrieval agent. The mailbot will navigate around a fixed path in an office building while retrieving mail from receptacles placed outside of the offices.

In the following pages, the components of the mailbot's existence will be discussed. First the system as a whole will be inspected. Then the components of the mailbot will be detailed, including the mobile platform itself, the actuation of the behaviors, a description of the sensors, and a summary of the behaviors.

The paper will end with a description of the final demonstration and where the mailbot was good and bad, or designed well and poorly, depending on the point of view.

Integrated System

System Components

The final form of the mailbot, as of this report, consists of a wooden, two section chassis on which the controller, sensors, motors and wheels are located on the primary section. The purpose of the primary section is to “drive” the mailbot to its destination.

The locomotion is achieved by two motor and wheel combinations on each side of the primary section. Two infrared (IR) emitters and detectors are placed in a cross-eyed fashion, as shown in Figure 1, to act as collision avoidance sensors. These sensors detect an impending collision by bouncing IR light off of an object in the path of the mailbot. Once an impending collision is detected, the mailbot takes action to avoid the object.

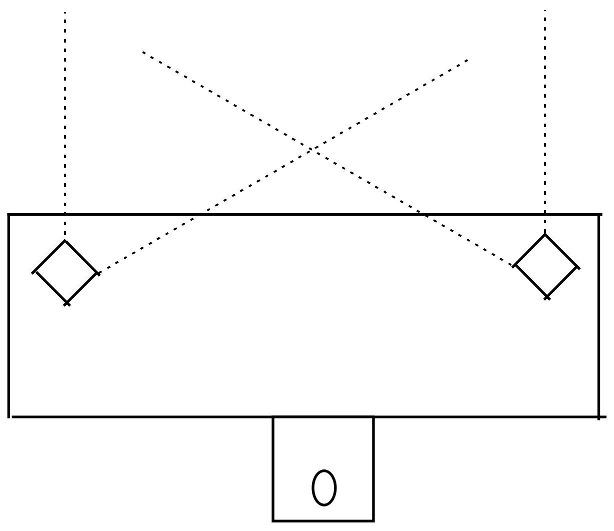

Figure 1

The mailbot is also equipped with three cadmium sulfide (CdS) cells to enable the line following behavior. The CdS cells are aligned as shown in Figure 2. Aligned in this fashion, the CdS sensors can determine relative variances in the amount of reflected light. The path to be followed will reflect less light than the surface on which the path lies, so the mailbot will be able to correct its course to follow the pre-determined path.

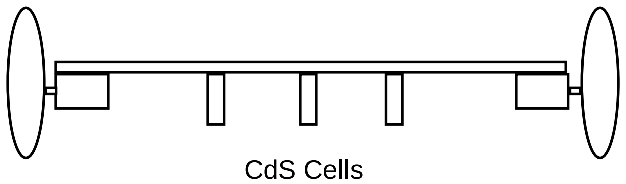
CdS Cells

Figure 2

The bump sensor is enables by three push button micro-switches connected by a light weight metal rod that acts as the bumper. When contact is made with the bumper, the switch is closed and the mailbot retreats in a manner to avoid contact with the object again.

The controller is the Motorola MC68HC11 EVBU. The EVBU is located on the drive section of the chassis. All of the sensor components are connected through the controller to the behavior actuators. The controller polls the sensors and determines the desired state of the mailbot depending on the sensor values. Based on the code running at the time, the controller places different priority weights on the program modules that actuate the behaviors. The controller runs a shell that time slices program modules so that virtual multi-tasking is achieved.

The code that ran on the mailbot during the final demonstration is located in Appendix A.

The secondary (payload) section of the chassis is a basket where the collected mail is stored. This basket rests on a spring supported scale with a push button micro-switch located directly under the line of action of the scale as shown in Figure 3. When the payload reaches a certain weight threshold the switch is contacted and the mailbot actively seeks the dumpsite so it can unload its cargo.

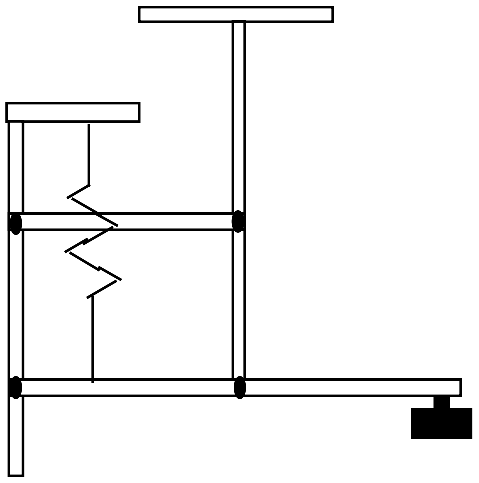

Figure 3

The two chassis sections are joined by nut and bolt. The wood platforms are separated by two flat metal washers to lower the friction between the two sections. The sections are joined in this fashion in order to decrease the turning radius.

System Operation

The mailbot was designed to begin in line following mode. Theoretically the mailbot would follow its path under outgoing mail receptacles. These receptacles would be designed in such a way that as the mailbot passed under, a hook on the mailbot would engage the spring supported receptacle and as the mailbot continued to pass, the receptacle would pivot about its hinge and dump the contents into the payload area of the mailbot. Figure 4 diagramming this process is shown below. The line following algorithm was not operational at the time of the demonstration so this behavior was deleted.

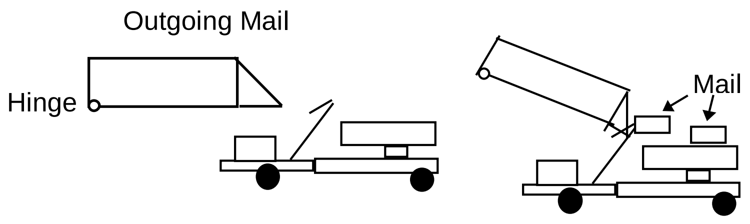

Figure 4

The line following behavior is the default behavior. In the case that an object is detected in the path or a bumper collision occurs, the mailbot changes priority from line following to collision avoidance or bump relief, depending on the which occurs. Once the object in the path is avoided, the mailbot returns to the path and the line following becomes the priority behavior again.

The mailbot continues along its path until the path terminates at the dumpsite. At this point, the mailbot will have passed by all of the receptacles and retrieved all of the mail or the payload sensor will have been tripped and a number of receptacles have been neglected. Once the payload reaches capacity, the mailbot continues along the path, with its retrieval hook lowered, until it reaches the dumpsite. At the dumpsite the payload area can be emptied by a human, another robot, or the mailbot can be modified to mechanically dump the mail autonomously. If the mailbot reached the dumpsite because the payload was at capacity then it will return to the path, form the beginning, and collect the rest of the mail. The idea here is that little enough time will have elapsed that the receptacles that have been emptied will not have been refilled. The mailbot will be able to get to those receptacles previously neglected.

Source Code

The code is based upon a modular format. The different behaviors of the mailbot are each done by a different module and the modules are integrated by an arbitration module. A priority module sets the priority of the behaviors depending on sensor values. For example, the default priority is line following so its priority is set to one and the others (avoid, bump, dump) are sequentially numbered. When the collision avoidance system notices an object in the path, the avoid module needs priority so the priority module sets the avoid priority to one and numbers the rest accordingly.

The arbitration module determines which behavior to realize based upon the current values of the behavior priorities. Each behavior has its own desired wheel value but the arbitrator decides which values the motors will use.

The modular format is desired because of ease of integration of new behaviors and ease of debugging. Once a module such as collision avoidance works, it only has to be included in the main module and integrated into the priority module and arbitration module.

The main module is the module that runs by default. Its sole duty in the case of the mailbot is to start the processes (sensor sweep, line following, etc.). The IC compiler loaded on to the EVBU time slices the modules so that virtual multi-tasking is achieved. All of the processes appear to run simultaneously, so the robot appears to be doing many things at once.

Mobile Platform

The Drive Section

This section of the mobile platform contains most of the hardware essential to the operation of the mailbot. The dimensions of this section are 8” X 11.5” X 4.5” (LXWXH). The drive section contains the wheels and motors on either side, three CdS cells attached under the platform, two IR emitter/detector pairs on either side, the EVBU, and the batteries. A diagram of this section is shown as Figure 5 below.

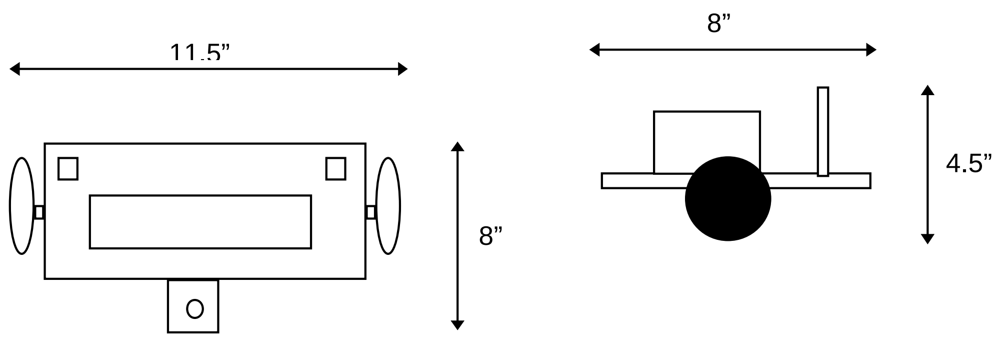

Figure 5

The Payload Section

This section is where the basket for cargo storage is attached. This section also has the payload capacity sensor. The dimensions for this section are 10.5” X 10.5” X 10”. A pair of low friction wheels are attached to the rear of this section. Figure 6, shown below, diagrams the payload section.

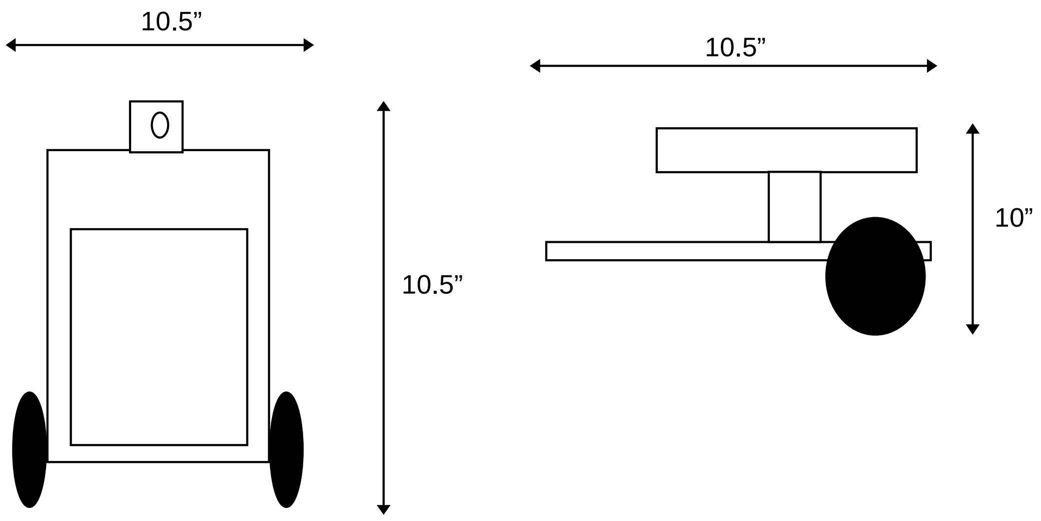

Figure 6

The two sections are joined in truck-trailer fashion, shown in Figure 7, by a nut and bolt. The two wooden surfaces are separated by two flat, metal washers to lower the amount of friction between the sections.

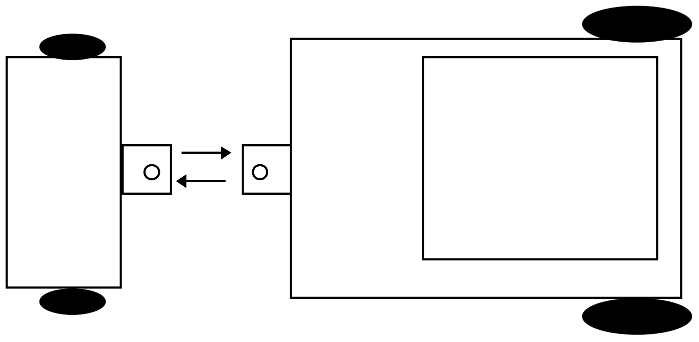

Figure 7

The two section format has advantages and disadvantages over the traditional single piece agent.   The two-piece design allows for a smaller turning radius. This is important because path following in an office building situation will necessitate turning in a relatively small area.

The modular design also has the advantage of interchangeability. A different type of section (for example, a mail delivery cargo area with a mechanical delivery arm) can be attached to the drive area. With a simple download to the EVBU, the function of the mailbot can be changed dramatically.

A disadvantage is the lack of control of the payload area during back-up. For example, when an object is in the path of the mailbot the mailbot has to back-up and correct its path. Depending on the orientation of the payload area in relation to the drive area, the back-up process may cause the mailbot to “jack-knife” as shown in Figure 8.

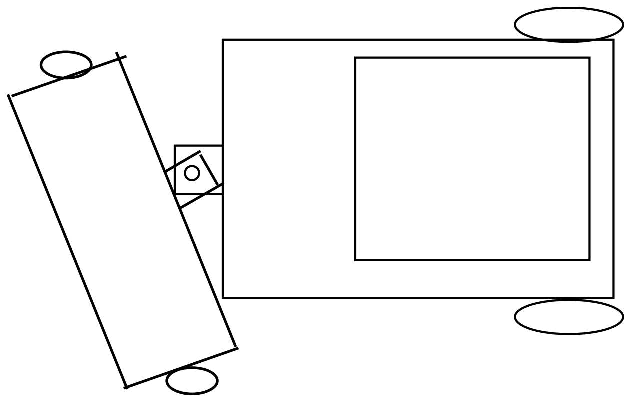

Figure 8

The payload section of the mailbot was fitted with wheels that have a low coefficient of friction on the surface in contact with the floor. The drive wheels have enough power to make the rear wheels slide in the case of a jack-knife.

Actuation

The main type of actuation on this robot is locomotion based. The motors are servos with a 180° range of motion that have been “hacked” to allow for a full circular range. The feedback is removed and the motors are controlled by sending a voltage relative to the desired speed of the wheel.

The line following sensors cause the motors to change speed based on the amount of reflected light detected. If the center CdS sees less light than either of the other two then the mailbot is on course. If the right CdS sees much less than the center CdS then the mailbot corrects its course by increasing the output of the left motor and decreasing the output of the right motor, thereby turning the mailbot to the right.

The collision avoidance sensors detect an object in the path of the mailbot and cause a path change by changing the speed or direction of the wheels. If an object is detected on the left side of the path, the mailbot will back-up in a left semi-circle until the object is no longer in the line of sight. The mailbot will then straighten its path and seek the line to follow. Figure 9 diagrams this action.

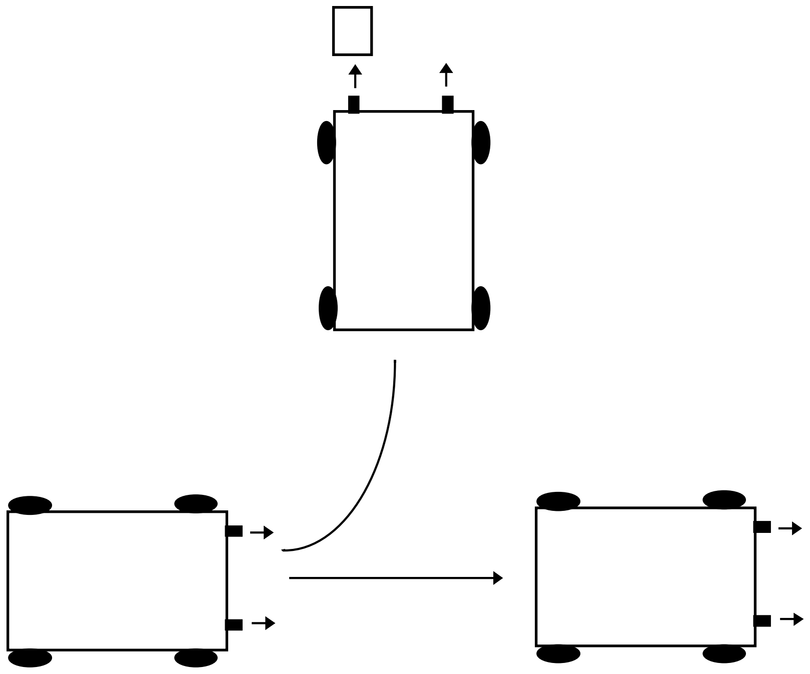
Figure 9

The bump sensors are similar to the collision avoidance sensors in that when a collision occurs, the wheel motors cause the mailbot to back-up in a right semi-circle. The right wheel is given a negative value and the left wheel is stopped, so the effect is that the robot backs up to the right. The mailbot then goes forward and seeks the line to follow.

The payload capacity sensor causes the mailbot to go into a dumping mode. For the demonstration, the mailbot was programmed to simply drive in a circle to show that the dumping state had been achieved. For an unknown reason the mailbot simply caused the motors to stop when the dump mode was achieved. All of the variable values necessary to put the mailbot in dump mode were in the correct state, but the wheel values were set to zero instead of the values that were programmed for that state.

After the mailbot has avoided the obstacles in its path it must return to the path. The mailbot passively seeks its path because of the layout of the office building. The mailbot is designed to travel on the right side of the hallways, as close to the wall as possible. Once the mailbot has ventured out into the middle of the hall to avoid obstacles, it returns to its path by virtue of the default motor values in the line following state. The right wheel turns more slowly than the left. This causes a tendency to the right, where the path is located. The CdS cells allow the mailbot to constantly check its alignment on the path and correct if necessary.

Sensors

Collision Avoidance

This sensor is comprised of an IR light emitting diode (LED) that is modulated at 40 KHz and a 40 KHz Sharp IR detector. The Sharp detector was "hacked" to make it an analog detector. Now the detector gives a reading based on the amount of IR light instead of whether it is present or not.

The LED, which is turned on and off at a frequency of 40 KHz, emits IR light in front of the mailbot. If an object is in the path of the mailbot the IR light will reflect back toward the mailbot. The Sharp detectors constantly read the value of the local IR light. The closer an object is to the sensor, the more light it will reflect. When the detected IR level reaches a certain threshold, a collision is impending and evasive action is taken.

Figure 1 shows the orientation of the collision avoidance sensors. They are placed in a cross-eyed fashion to detect objects that would have been missed if they were oriented parallel to the course of the robot. The mailbot is wide enough that two "eyes" pointing straight ahead aren't capable of seeing an object passing directly between the two sensors. The sensors were crossed to provide an overlap area in the center of the mailbot's vision field.

A deficiency of the mailbot is that it cannot see to its immediate left or right. The orientation of the IR sensors allow for better center coverage, but if an object is encountered in the path to the left of the left eye or to the right of the right eye, it will not be detected. The mailbot demonstrated a hard time getting away from walls that were just less than parallel to its path but were outside of the field of vision.

Bump Detectors

The bump sensor is a bar that stretches along the front edge of the mailbot. It is connected to three push button micro-switches that connect two pins when the button is pressed. If an object hits the bumper, a contact is made and the mailbot senses the object and changes its course.

The three switches are connected in parallel to a pull up resistor attached to the analog input pin as shown in Figure 10 below. This way, if any switch is closed, the pin voltage level drops to zero volts and the bump is sensed. The metal bumper bar is connected to the switches by a small drop of Krazy Glue. Care must be taken to apply only a small drop or the glue may prevent the movement of the button and the switch becomes useless.

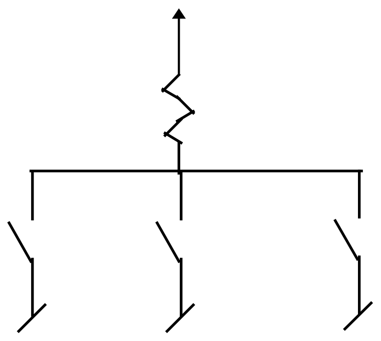

Figure 10

Line Following Sensors

Three Cadmium Sulfide (CdS) cells are attached to the bottom of the drive section. These cells are analog devices that sense ambient light. These cells are used to read in the amount of light reflected from the floor. The idea is that the path to be followed is of such a color that less light is reflected than the surrounding area. The mailbot follows this path of diminished reflected light.

The cells are inserted into the a pen cap that has been modified to provide a hole on either side, in essence becoming a small hollow tube. This restricts the area that the CdS cells see and allows for a more focused vision field. The three cells are each connected to an analog input pin and are read constantly so that the line following module can continually correct its path.

On the day of the demonstration, the line following module was not working so it was deleted from the program code. Future work includes getting this module to function correctly.

Payload Capacity Sensor

The mailbot needed a sensor to detect when the payload area was full so it could dump the contents and not have letters fall out due to overflow. The payload capacity sensor is a spring supported kitchen scale with a push button micro-switch placed directly under the line of action of the scale as shown in Figure 3. The switch is connected to ground potential and an input pin through a pull-up resistor to five volts. When contact is made, the pin voltage drops to zero volts and the mailbot changes to dump mode.

The sensor uses a switch because the question is “Has capacity been reached?” and not “What is the present capacity?” The mailbot does not care what the analog value of the payload weight is, it only cares if the capacity threshold has been met.

Behaviors

Line Following

The mailbot was intended to follow a path of black tape set on a lightly colored floor. The CdS cells detect the reflected light and depending on their relative levels, the line following module sets the motor values. For example, if the right CdS sees less much less reflected light than the center CdS, then the black tape is under the right CdS. The motor values are changed such that the right wheel turns slower than the left wheel until the black tape is under the center CdS again. At this point the motor values are set back to their default values.

While this behavior seems simple, this module was not functioning correctly at the time of the demonstration so it was deleted from the program code. Future work includes getting this module to work properly and integrating it into the mailbot’s behavior structure.

Collision Avoidance

This behavior is based on the idea of getting out of the way of a collision so that no damage is caused. If the collision avoidance sensors notice something in the path of the mailbot, this behavior changes the wheel values so that the path is altered and the object is avoided. Figure 9 diagrams the collision avoidance behavior.

This behavior works well for objects in the line of vision. The problem discovered at the final demonstration was that a wall oriented in a less than parallel path immediately outside of the field of vision caused the mailbot to skim the wall. This can be alleviated by placing a side looking IR emitter/detector pair on either side of the drive section. This creates a larger field of vision and would allow for avoidance of obstacles out of the original field of vision but in the path of travel.

Bump Relief

This behavior uses the bump sensor to indicate when a collision occurs. Upon collision, the mailbot backs up in a right semi-circle. The mailbot will pass the object on the left and will return to line following behavior. This behavior calls for the mailbot to pass on the left because in the environment for which the mailbot is designed, a wall will be on the right and a relatively large area (the remainder of the hallway) will be on the left.

The only problem encountered with this behavior was designer error. When attaching the metal bumper rod to the switches, a little too much Krazy Glue was used and one of the switches became inactive. The bumper still operated for about 90% of the width of the bumper so the behavior was proven effective.

Payload Dumping

Once the payload capacity sensor notices that capacity has been reached, this behavior takes effect. The mailbot continues on its path, but it no longer picks up outgoing mail from the receptacles because the hook is in a position that won’t engage the receptacles. This can be accomplished by attaching the hook in such a way that as the basket lowers, due to the weight of the cargo, the hook that is directly attached to the basket also lowers. At capacity weight, which is related by displacement to basket height, the hook is at an elevation that cannot engage the receptacles.

Another approach is to connect the hook to a servo. When the payload capacity has been reached, the servo is moved so that the hook is lowered. Upon entry into non-dumping mode the servo is moved to its original, hook extended position.

The path terminates at the mail room. The mailbot will stop in a designated place and will be unloaded by either a human or another robot. The basket can also be modified to be servo controlled so that once the dumpsite is reached, the servo is enabled and the basket (now hinged on one side) is raised and the cargo is dumped into an outgoing mail bin.

Conclusion

About 80 - 85% of the designed specifications were met at the final demonstration. The line following was the only behavior that truly did not work, along with a slight bug in the module to demonstrate dumping mode. All of the sensors performed well. The IR detectors could have been oriented to cope better with the problem encountered with the wall. Also the basket was a little wobbly but the payload capacity sensor worked well.

The area that most pleased me was the coding. Although simple by coding standards, the modular format worked well. I am pleased that the priority setting module and the arbitrator module worked so well. I was skeptical about using that model, opting originally for the long, all inclusive main module, but the modular format proved very valuable for debugging purposes.

The areas with which I was most displeased were my time management and asking for help. Instead of working continuously at a constant rate, I worked very hard in spurts and very little the rest of the time. While I accomplished many of my design goals, I'm sure I could have completed nearly all if not all I would have managed my time better.

I also tried to figure too much out on my own. When I had trouble I spent too much time trying to correct the problem. After failing to solve the problem in a reasonable amount of time, I should have asked someone with experience instead of wasting more time on the problem. I was arrogant to think that I was the only one who would understand my specific problem.

If I had to begin this project again, I would meet with the professor or a teaching assistant and create a timeline based on my design goals. The timeline would serve as progress checkpoints to help with my time management.

The future work on this project includes getting the line following behavior to work. I also plan to add some IR detectors to alleviate the wall skimming problem. Possible future work includes adding a servo to the retrieval hook so it can be raised and lowered based on the status of the payload capacity sensor.

_References

1994 6.270 LEGO Robot Design Competition Course Notes. 6.270 Organizers. 1994.

Appendix

```
/* This file is a program for my final demo. */

/* Global Sensors*/
int left_eye, right_eye, thresh=100;
int bump_tmp, bump; /* bump sensor */
int pay_cap_tmp, pay_cap;


void sensor_module() /* Read Sensors*/
{
      left_eye = analog(1);
      right_eye = analog(2);
      bump_tmp = analog(6);
      if(bump_tmp < 128){
         bump=1;
      }
      else {bump=0;}
      pay_cap_tmp=analog(7);
      if(pay_cap_tmp<128){
         pay_cap=1;
      }
      else{pay_cap=0;}
      defer();
}

/* Globals */
float left_wheel, right_wheel;
float avoid_left, avoid_right;

void avoid() /*Go but avoid collision */
{
    /* if there is nothing in the path */
    avoid_left=80.0;
    avoid_right=80.0;

    if(left_eye>thresh || right_eye>thresh){
    /* if there is something in the path */

      /* if it is on the left side, backup right */
      if(left_eye >= (right_eye+5)){
         avoid_left = -55.0;
         avoid_right = 0.0;
      }
      /* if it is on the right side, backup left */
      else if(left_eye < right_eye){
         avoid_left = 0.0;
         avoid_right = -55.0;
      }

      /* if it is the center, go straight back */
      else{
         avoid_left = -55.0;
         avoid_right = -55.0;
      }
    }


}

/* bump relief wheel values */
float bump_left, bump_right;

/* collision relief module */
void bump_relief(){

     if(bump){
        bump_left = -80.0;
        bump_right = 0.0;
        sleep(2.0);                    // note: don't use sleep, use wait.
     }
     defer();

}

float dump_left, dump_right;

void dump_load(){
   if(pay_cap){
      dump_left=60.0;
      dump_right=30.0;
   }
   defer();
}

/* priority globals */
int avoid_pri, bump_pri, dump_pri;
```

```
/* set the priority for the arbitration module */
void set_priority(){

     if(bump){
        bump_pri = 1;
        avoid_pri = 2;
        dump_pri=3;
     }
     else if(pay_cap){
        dump_pri=1;
        avoid_pri=2;
        bump_pri=3;
     }
     else{
        avoid_pri = 1;
        bump_pri = 2;
        dump_pri=3;
     }

}

void arbitrate() /* behavior arbitrator */
{

     if(avoid_pri==1){
       left_wheel = avoid_left;
       right_wheel = avoid_right;
     }
     else if(bump_pri==1){
        left_wheel = bump_left;
        right_wheel = bump_right;
     }
     else if(dump_pri==1){
        left_wheel=dump_left;
        right_wheel=dump_right;
     }
     else{
        left_wheel=10.0;
        right_wheel=10.0;
     }
     motor(0,left_wheel);
     motor(1,right_wheel);

}
```

```
void main()
{
  poke(0x7000,0xff);
  while(1){
  /*  start_process(get_thresh()); */
   start_process(sensor_module());
   start_process(avoid());
   start_process(bump_relief());
   start_process(set_priority());
   start_process(arbitrate());
  }
}
```

Mini-Abstract

This robot is an autonomous mail collecting agent that performs the following behaviors: line following, collision avoidance, bump relief, and payload capacity determination.