Introduction to Intelligent Robotics
Dr. Dean Hougen
Spring 2004
Project 1
February 16, 2003

Alpha 4.0

Team 5
Patrick Bradley ( )
Mayank Murarka (000257487)
Tandy Jones (4419073191)

1. Introduction

1.1. Project Objectives

Design, build, program, and demonstrate an autonomous robot that carries out the following simple task: To move around a course of colored "tiles." Pieces of black electrical tape will be attached to the floor to form one-foot squares. These squares will be placed no more than six feet apart from one another and no closer than one foot apart from one another on an orthogonal grid. Within each square will be a piece of colored paper indicating the action the robot should take in that square. The colored paper will be 2 inches by 2 inches and located in the center of the square. There will be five colors indicating five different possible actions:

1. Green - Forward. The robot should not turn (turn 0 degrees) then move forward to the next square.
2. Pink - Reverse. The robot should turn around (turn 180 degrees) then move forward to the next square.
3. Blue - Right. The robot should turn to its right (turn clockwise 90 degrees) then move forward to the next square.
4. Orange - Left. The robot should turn to its left (turn counterclockwise 90 degrees) then move forward to the next square.
5. Yellow - Stop. The robot should stop moving and give an audio indication of having successfully completed its task.

2. Robot Design and Hardware

The design of the robot contains the CMUCam camera, camera switch relay, optical encoders, infrared (IR) sensors, wheels, motors, gearing, and enough legos to keep the robot from bending under its own weight or inhibiting its functionality. We made the robot as small as possible so that the turning would be easy to control, and to make the construction simpler.

2.1. Design Process

The first prototype that was designed was a three wheeled robot. Two large front wheels drove the robot with a caster wheel for the back. While testing this design, it was found that changing the direction depending on the alignment of the caster wheel would be unacceptable when making lots of turns.

The next design used a fixed third wheel that would allow the robot to move straight without any trouble, but this design had problems of its own. Turning the robot became a problem because the friction of the third wheel was so great that the turns could not be made smoothly.

The next design was a tread design to keep the robot moving in as straight a line as possible giving better flexibility for turning. The zero radius turning feature of the tracked wheels showed a very good incentive to be used here as that would ease the difficulty of aligning the robot after each turn. The camera was mounted near the turning point of the

robot but then the shading the robot inflicted on the camera was too great to get accurate readings.

So, finally the camera was moved out in front. And the distance between both the treads was also decreased to getter better stability and compactness for the robot which was then named "alpha 4.0" Fig 2.1. Also, for better tracking of the movement of the treads encoders were used. A gear train was used to increase the gear ratio between the tracks and the encoder to get better accuracy.

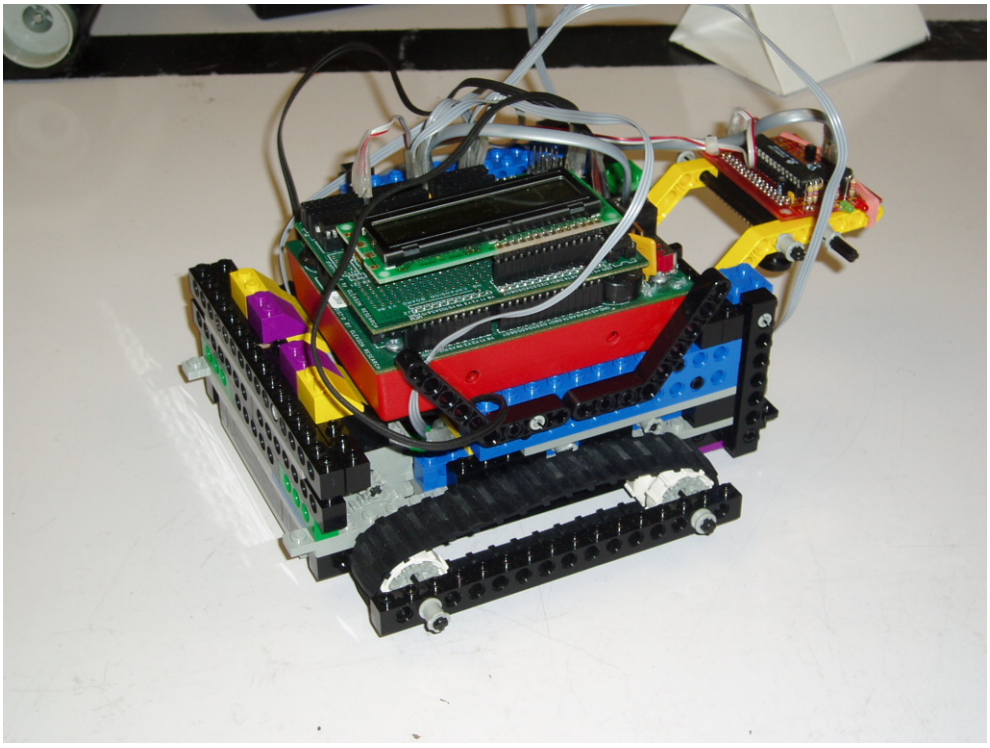

Fig 2.1 Alpha 4.0 - Final design

2.2. Alpha 4.0

2.2.1. Drive Train

The tracks had two tread wheels each separated by 11 holes in a long lego brick. Bricks were placed on both sides of the treads to keep the wheels stable. The robot had a motor for each tread connected to one tread wheel, and the other tread wheel had a series of gears on the other end connected to the encoders. The motors each had 8 tooth gears meshed with 24 tooth gears directly connected to the back tread wheels of the robot. This gave a reduction of 3 times between the motor and the tread. The encoders were attached on the front wheel with a compound gear train giving a gear ratio of 10:1 as shown in Fig 2.2. This gave reasonable precision when making turns. Every axel had at least two lego bricks to keep them from bending under pressure.

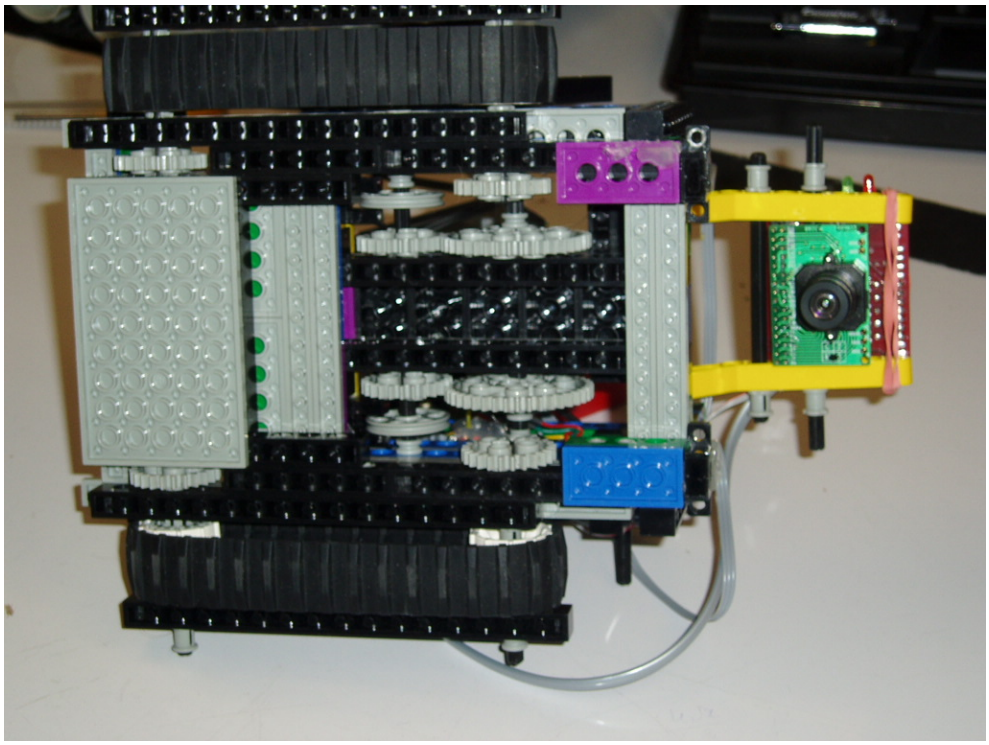

Fig 2.2 Bottom view of the robot with gear train

2.2.2. Sensors

The IR sensors and encoders were hot glued to the legos to keep them from moving. The camera was held in place by legos and a single rubber band. The encoders were placed just where they lined up with the lego encoder wheels. The IR sensors were glued to the very front right and front left of the robot after the design was finalized. The IR sensors were plugged into the digital ports of the handy board which gave us a reading of 0 when the sensors sensed black and 1 when they sensed white.

The camera was attached to the front of the robot using yellow L shaped legos and black axels to hold it in place. This placed the camera approximately 1 to 2 inches in front of the robot pointing straight down and giving it a view as free of shadows as we thought possible. The camera switch relay was attached along the left side of the robot to keep it out of the way as much as possible.

The only additional concern that was dealt with was moving the handy board above the drive train so that the center of gravity is as close to the axis of rotation and keeping it from sliding off in any direction.

2.2.3. Possible Improvements

The only major problem encountered with the robot was turning and moving certain distances. The gear ratio we used for the encoders worked well, but we could have increased it and thus the precision with which the encoders ran. This would allow more accuracy when moving.

3. Software Organization

3.1. Forward Motion

One of the foremost concerns for us was to see that the robot goes in a straight line. This was needed because it seemed that one of the motors had some friction in it and was slower than the other. For this, the voltage pulse to the left and the right motors were so adjusted that when both of them are actuated they move in a straight line. There were three forward commands which differed in speed. One was "goFast" which was used for traversing the area between the black squares which could have been anywhere from 1 feet to 6 feet. Next was "forward" to get the camera on the colored 2-inch square once the blacktape was detected. The other was "goSlow" method was used to move inside the black square after the detection of the centered colored square.

3.2. Robot Alignment

For ensuring that the robot traveled in a straight line optical encoders were used in the "forward" method. This checked for the number of ticks (cuts in the optical encoder beam) made by the left and the right motor separately. If any one of the motors recorded higher number of ticks than the other then that motor was slowed down so that both the motors lined up. This same technique was employed in the turns too. This idea was implemented being inspired by the code presented by last year's teams. [1] [2].

If the robot did not go straight or after some turns was not in the correct path an alignment check by infrared sensors was done whenever the black tape was detected. Here, of the left IR sensor detected the black tape before the right one the left motor would be moved in the reverse direction keeping the same direction for the right one till both of them aligned. Similar thing was done if the right IR sensor registered before the left one.

3.3. Turning

To get the exact 90^0 turns the code was adjusted based on brute force method. The number of ticks required to get the right, left and turnaround turns were observed and the average reading of both the encoders was set to this. It can be seen that the number of ticks for left turn are not same as the number of ticks for right turn which are not half of that for around turn as would be logical. The encoders were also used to detect any slip and rectify that as described in the earlier section.

3.4. Camera Initialization

The camera was initialized in the RGB mode instead of the YUV mode. The CMU manual came in handy for this purpose [3] to tweak the camera initialization code for YUV mode in Dr. Miller's library file [4]. It was seen that the performance of the robot in color detection was greatly improved by this. In the YUV mode the values registered for each color overlapped with some of the other colors. For example, pink was very close to orange and sometimes even green would be detected as yellow. So, switching to the RGB mode gave non-intersecting values and increased the performance of color tracking.

3.5. Color Detection

The CMU JavaCamGUI software was used for observing the RGB values given by the CMU camera for different colored papers. The deviations for these values depending on the lighting conditions were also given consideration and the maximum and minimum values of RGB for each color was observed. This was put in the "trackRaw" method for detecting various colors and decision was taken to either turn the robot left or right or around. Separate functions were written for tracking blue and orange also as the "track_blue" and "track_orange" operated for YUV modes.

4. Results of Demonstration

The robot was able to make nearly 90 degree turns affectively. It was not able to make a 180 degree turn, but would have been able to had more time been given to testing and debugging. It would drift only by a few inches during a six foot straight away, and would line itself up with the black tape when it approached the tape at an angle. The camera was able to recognize all the colors affectively.

The course that was set up for the demonstration contained 13 tiles, repeating 3 of them twice, making the course 16 tiles long. The course was successfully completed by alpha 4.0 needing adjustment only 4 or 5 times so it would line up with the center of a following tile.

References

[1] http://www.cs.ou.edu/~hougen/classes/Spring-2003/Robotics/materials/project1/team05/project1_team5.ic
[2] http://www.cs.ou.edu/~hougen/classes/Spring-2003/Robotics/materials/project1/team03/behavior1.ic
[3] http://www-2.cs.cmu.edu/~cmucam/Downloads/CMUcamManual.pdf
[4] ..\Interactive C 4\Handyboard\cmucamlib.ic

Team Organization Evaluation and Plans

The project was originally divided into three tasks, construction of the robot, getting the robot to sense and getting the robot to move appropriately. Each group member was assigned one task and expected to complete the task on their own.

Following the construction of the first robot, changes were immediately needed and it was clear that changing the construction of the robot would be an ongoing process. Everyone from the group therefore helped build the robot at one point or another, altering its design with each problem faced.

The person assigned to sensing spent a great deal of time getting the camera to sense colors correctly. However, the team member assigned to movement also helped with the testing. Once the correct values for the camera to read were found, the final testing began.

The majority of the programming for the robot was done during testing, finding which implementation would work the best took several tries and several different ideas.

Evaluation

The division of labor did not work well for this project because each section blended into the others. A better organization would be to have the group decide on a design, and then have more than one person work on each part of the project.

The robot was kept at a lab that was usually occupied by someone so that group members could work on it when available. This worked out well so that we did not need to coordinate free time very much.

The design of the robot took so much time that we had little time to work on movement. To remedy this, the team should start work on the robot design earlier so we could have more time for programming and testing. Also, having a better design at the beginning would enable less rebuilding which took a considerable amount of time.

Alpha 4.0 Support Software

```
#use "cmucamlib.ic"

/*****************************************************************************
Intro to intellegent robotics
Dr. Hougen
Spring 2004
Project 1
Team 5
*****************************************************************************/


int TICKS_PER_LEFT = 96;
int TICKS_PER_RIGHT = 101;
int TICKS_PER_TURNAROUND=185;
float TIME_TO_SLEEP = 1.5;
int TICKS_TO_CAMERA = 65;
int TICKS_TO_COLOR = 85;
int R_MOTOR = 3;
int L_MOTOR = 0;
int R_SENSOR = 14;
int L_SENSOR = 15;

//goFast() moves the robot at a reasonably fast pace
void goFast()
{
    motor(L_MOTOR,-50);
    motor(R_MOTOR,-48);
}

//goSlow() moves the robot half the speed of goFast()
void goSlow()
{
    motor(L_MOTOR,-25);
    motor(R_MOTOR,-24);
}

//forward(int x) moves the robot x encoder ticks
void forward(int x)
{
    reset_encoder(0);
    reset_encoder(1);

    motor(L_MOTOR,-50);
    motor(R_MOTOR,-48);


    while(1)
      {

        if(averageEncoders() >= x)
          {
            stop();
            reset_encoder(0);
            reset_encoder(1);
            return;
        }
      }
    return;
}//end foward(int x)
```

```
//findBlack() runs while the robot is heading toward black tape
//  it will align the robot with the black tape and then return control
void findBlack()
{
    while(1)
      {

        if ((!digital(R_SENSOR)) && (digital(L_SENSOR)))
          { // when the right IR sensor is on the blacktape and the left sensor is not then,
            // the right motor is moved backward and left in forward mode to get the robot aligned
            //to the blacktape
            while(!((!digital(L_SENSOR))&&(!digital(R_SENSOR))))
              {
                motor(R_MOTOR,30);
                motor(L_MOTOR,-35);
            }
            stop();
            return;
        }
        if ((!digital(L_SENSOR)) && (digital(R_SENSOR)))
          { //this is the opposite of the above
            while(!((!digital(L_SENSOR))&&(!digital(R_SENSOR))))
              {
                motor(L_MOTOR,30);
                motor(R_MOTOR,-35);
            }
            stop();
            return;
        }
        if ((digital(L_SENSOR) && digital(R_SENSOR))) // both sensors see white
          {

        }
       if (!((digital(R_SENSOR)) || (digital(L_SENSOR)))) // both sensors see black
          {
            printf("Black found\n");
            return;
        }
    }

}

//backward() moves the robot backwards
void backward()
{
    motor(L_MOTOR,25);
    motor(R_MOTOR,24);

    return;
}

//turn off motors
void stop()
{
    off(L_MOTOR);
    off(R_MOTOR);
    return;
}

//diffEncoders() take the difference of the encoders
//  we had our right encoder as 0 and left as 1
int diffEncoders()
```

```
{
    return (int)(read_encoder(0)-read_encoder(1));
}

//averageEncoders() adds the encoder values and divides by 2
int averageEncoders()
{
    return (int)((read_encoder(0)+read_encoder(1))/2);
}

//left() turns the robot left 90 degrees
void left()
{
    reset_encoder(0);
    reset_encoder(1);
    motor(L_MOTOR,50);
    motor(R_MOTOR,-50);
    while(1)
      {
        if (averageEncoders() >= TICKS_PER_LEFT)
          {
            stop();
            return;
        }
        if (diffEncoders() > 0) //right has traveled farther
          {
            motor(L_MOTOR,40);
            motor(R_MOTOR,-50);

        }
        else if (diffEncoders() < 0) //left has traveled farther
            {
              motor(L_MOTOR,50);
              motor(R_MOTOR,-40);
          }
          else  //the encoders are the same
            motor(L_MOTOR,50);
            motor(R_MOTOR,-50);
    }
}//end left

//right() turns the robot right 90 degrees
void right()
{
    reset_encoder(0);
    reset_encoder(1);
    motor(L_MOTOR,-50);
    motor(R_MOTOR,50);
    while(1)
      {

        if (averageEncoders() >= TICKS_PER_RIGHT)
          {
            stop();
            return;
        }
        if (diffEncoders() > 0) //right has traveled farther
          {
            motor(L_MOTOR,-50);
            motor(R_MOTOR,40);

        }
        else if (diffEncoders() < 0) //left has traveled farther
```

```
            {
              motor(L_MOTOR,-40);
              motor(R_MOTOR,50);
          }
          else
            motor(L_MOTOR,-50);
            motor(R_MOTOR,50);
    }
}//end right()

//turnaround turns the robot right 180 degrees
void turnaround()
{
    reset_encoder(0);
    reset_encoder(1);
    motor(L_MOTOR,-50);
    motor(R_MOTOR,50);
    while(1)
      {

        if (averageEncoders() >= TICKS_PER_TURNAROUND)
          {
            stop();
            return;
        }
        if (diffEncoders() > 0) //right has traveled farther
          {
            motor(L_MOTOR,-50);
            motor(R_MOTOR,40);

        }
        else if (diffEncoders() < 0) //left has traveled farther
            {
              motor(L_MOTOR,-40);
              motor(R_MOTOR,50);
          }
          else
            motor(L_MOTOR,-50);
        motor(R_MOTOR,50);

    }
}//end turnaround()

//all the track_color() methods return the confidence
//  that the camera sees that color

int track_pink()
{
    int crmin;
    if (trackRaw(100,180,12,23,12,20) > 0) {
        return track_confidence;
    } else return 0;
} // end track_pink() //

int track_green()
{
    int crmin;
    if (trackRaw(95,120,135,170,15,25) > 0) {
        return track_confidence;
    } else return 0;
} // end track_green() //

int track_yellow()
```

```
{
    int crmin;
    if (trackRaw(140,180,140,180,13,19) > 0) {
        return track_confidence;
    } else return 0;
} // end track_yellow() //

int track_orange2()
{
    int crmin;
    if (trackRaw(125,185,45,65,13,19) > 0) {
        return track_confidence;
    } else return 0;
} // end track_yellow() //

int track_blue2()
{
    int crmin;
    if (trackRaw(30,65,125,185,75,130) > 0) {
        return track_confidence;
    } else return 0;
} // end track_yellow() //


//clamp_camera_rgb() came from clamp_camera_yuv only changed the numbers
//  to set the camera in RGB mode
int clamp_camera_rgb()
{
    int i;
    printf("Point at white  and press start\n");
    while (!start_button());
    send_R_command_require_reply("CR 18 44\r",9);
    /*    for (i= 15; i > 0; i--) {
        printf("Setting white   balance %d..\n", i);
        sleep(1.0);
    }
*/  //we commented this out because the white balance was
    //  taking too long in testing.  It worked fine without it.
    send_R_command_require_reply("CR 18 40\r",9);
    printf("Done\n");
} // clamp_camera_RGB() //


void main()
{
    int i;
    init_camera(); // initialize the camera
    clamp_camera_rgb(); //clamp camera white balance (RGB mode)

    while (!start_button()) //starts moving after you press start
      {}


    enable_encoder(0);//enable encoders
    enable_encoder(1);
    sleep(1.0);

    //move until you find black tape
    goFast();
    findBlack();

    while (1)
      {
```

```
        //move the camera to the center of the square
        forward(TICKS_TO_CAMERA);
        sleep(1.5);
        //look for a color
        if (track_orange2() > 0) //if orange is found
          {
            printf("orange found\n");
            forward(TICKS_TO_COLOR); //move the turning point on top of the color
            left();
            goSlow(); //start moving
            findBlack();//look for the black tape of the square you are leaving
            sleep(1.0);
        }
        if (track_blue2() > 0)// if blue is found
          {
            printf("blue found\n");
            forward(TICKS_TO_COLOR); //move the turning point on top of the color
            right();
            goSlow();  //start moving
            findBlack(); //look for the black tape of the square you are leaving
            sleep(1.0);
        }
        if (track_green() > 0) //if green is found
          {
            printf("green found\n");
            sleep(1.0);
        }
       if (track_pink() > 0) //if pink is found
          {
            printf("pink found\n");
            forward(TICKS_TO_COLOR); //move the turning point on top of the color
            turnaround();
            goSlow(); //start moving
            findBlack(); //look for the black tape of the square you are leaving
            sleep(1.0);
        }

        if (track_yellow() > 0) //if yellow is found
          {
            printf("yellow found\n");
            forward(TICKS_TO_CAMERA); //move the turning point on top of the color
            for (i=0;i<20;i++) //beeping course completed
              {
                beep();
            }
            stop();
            return;
        }
        goFast(); // go to next square
        findBlack(); // look for the tape of the next square
    }
    return;
}
```