MicroMouse: An Autonomous Maze-Solving Robot

Mack Ward[1], Scott Will[2], Kyle Thompson[2], Dr. Jennifer Zirnheld[2]

1. Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260
2. Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260

University at Buffalo
The State University of New York

CURCA
CENTER FOR UNDERGRADUATE RESEARCH AND CREATIVE ACTIVITIES

Background

- MicroMouse is an event in which teams of participants construct robots that autonomously solve mazes in as little time as possible.
- In an attempt to reduce the amount of time that our robot spends exploring and solving the maze, we investigated the usefulness of a variety of novel technologies, including computer vision for wall detection.
- In addition to designing, constructing, and testing the physical robot, we also built a MicroMouse simulator to assist with algorithm development.

Computer Vision Overview

- The regularity of MicroMouse competition mazes, composed of equally-sized unit squares, uniform-height walls, and so on, makes it possible to visually extract wall information.
- The wall positions can be most easily determined from an orthographic, top-down view; thus, the goal of our algorithm is to transform a series of panoramic images, taken from the robot's starting position, into a single top-down image of the maze.
- First, we stitch together pairs of images using a script derived from a publicly available GitHub project called *stitcher*.
- Then, we use a variety of image processing techniques to determine the correct homography for each stitched image and perform the perspective transformation.
- Lastly, we stitch together the transformed images and use a combination of filtering and thresholding to identify walls in the maze.

Computer Vision Implementation

1. Obtain a "skeletonized" binary representation of the image by thresholding color values to obtain only red regions, and then reducing the region to single-pixel width by iteratively performing morphological dilation and eroding operations (Figure 2).
2. Find all lines in the skeletonized image using the Hough line transform– these lines usually lie along, or near, maze walls.
3. Since skeletonized walls are not perfectly straight, the Hough transform will find multiple lines with similar slope and intercept for each wall. Divide the lines into groups according to slope, and further subdivide by intercept (Figure 3).
4. Obtain a single line for each wall by removing outliers in every cluster and then averaging slope and intercept (Figure 4).
5. Compute all points in the image where the averaged Hough lines intersect.
6. Treating the intersection points as polygon vertices, find all closed polygons in the image.
7. Identify the polygon enclosing the smallest area, since this is likely to be the perspective view of a square maze cell (Figure 5).
8. Compute the homographic transformation between the stitched image and a top-down view of the same scene by using the corners of the aforementioned polygon and the corners of a square centered in a 2500 x 2500 pixel image as key points. The large size of the target image is chosen so that the source image, after being transformed by the homography (which stretches some regions) does not experience boundary clipping.
9. Crop the transformed image by finding the smallest rectangular region that bounds the image information (most of the pixels of this image will be zero, and the transformed, stitched image will occupy a small quadrangle near the center) (Figure 6).

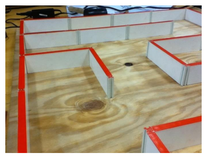


Figure 1
A sample Input image

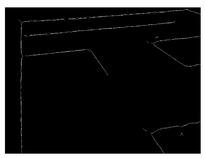


Figure 2
Skeletonized image

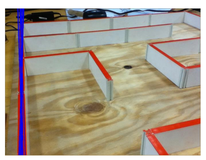


Figure 3
A group of Hough Lines

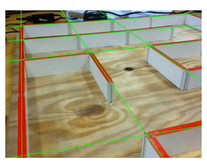


Figure 4
All calculated Hough lines

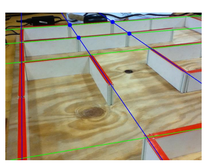


Figure 5
The smallest square in the image

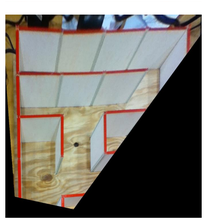


Figure 6
Resultant image after performing the perspective transformation

Robot Design and Construction

- The printed circuit board is used as the chassis to conserve weight and make the robot smaller (Figure 9).
- Infrared emitters and receivers are used to sense the walls around the robot. They can sense changes very quickly which allow the robot to move at faster speeds without crashing.
- The Teensy 3.1 microcontroller is overclocked to 96MHz. This allows for quick processing so the robot can detect walls, make corrections, and decide where to go next in less than 1ms.
- The geared four wheel drive allows the robot to move without any parts dragging on the ground.
- 3D printed motor mounts and wheel hubs allow for low-cost and quick prototyping (Figure 11).
- DC motors with encoders were chosen for their small size and fast speeds. The motors encoders enable accurate dead-reckoning within the maze.
- Surface-mount electronic packages were used to achieve small size and light weight (Figure 10).
- Six sensors are utilized to sense left and right, diagonally and forward. The sensors are quickly pulsed on then off to allow a higher brightness than would normally be permissible. All sensor readings are taken within 480µs.

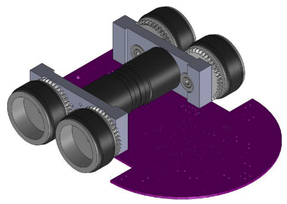


Figure 9
3D Model of mechanical construction

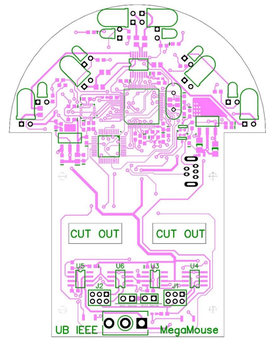
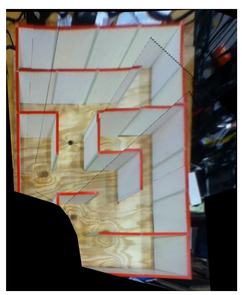


Figure 10
Printed Circuit Board (PCB) design

Figure 11
Completed Robot

- The four wheels are free to spin via ball bearings. They are turned by a pinion gear attached to the motor with a 40:9 gear ratio.

Computer Vision Results



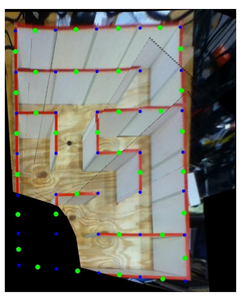


Figure 7
The final, stitched, perspective-warped image. This image is useful because the location of the wall pixels are representative of the actual wall locations, and thus we can simply inspect pixels to determine the existence of walls.

Figure 8
The walls of the MicroMouse maze, as detected by out algorithm. The blue dots represent the calculated wall vertices, and the green dots represent detected walls.

Simulation	Brute Force	FloodFill	Our Method
1	714	206	114
2	750	236	128
3	858	236	144
4	700	199	82
5	630	190	74
6	666	191	96
7	744	546	90
8	721	273	65
9	755	191	37
10	944	220	160

Table 1
Total discrete steps required to completely solve a maze using a brute force method, FloodFill (a popular algorithm in MicroMouse competitions), and our method.

Note that these results assume the complete functionality of our algorithm on a full size maze, which we haven't yet had the opportunity to confirm.

Simulator for Algorithm Development

- As an aid to MicroMouse algorithm and control code development, we built a MicroMouse simulator application entirely from scratch.
- The simulator interfaces with both continuous and discrete algorithms; that is, it provides two different interfaces, each at a different levels of abstraction.
- The continuous algorithms rely solely on sensor readings to control the robot (Figure 12).
- Furthermore, the only mechanism available to continuous algorithms for controlling the robot is the ability to set the motor speeds; this helps to ensure that the algorithms are written as realistically as possible.
- Discrete algorithms have the ability to move around the maze in discrete jumps: forward one square, turning ninety degrees, etc.
- Discrete algorithms also have the ability to speed up, slow down, and pause the simulator; this allows algorithms to be tested much more quickly than in real life
- Each type of algorithm as the ability to display the walls of the maze as perceived by the robot; this allows the programmers to understand exactly what the robot is thinking, and makes for easier debugging (Figure 13).
- All code is open source, and written to be as accessible as possible so as to encourage use by other teams.

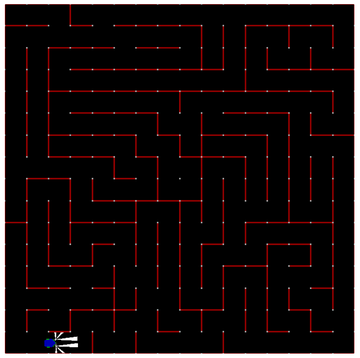


Figure 12
The simulator running in continuous mode, showing the range of each of its sensors.

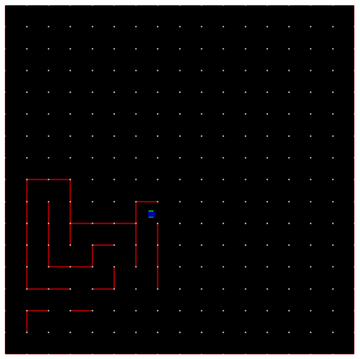


Figure 13
The simulator running in discrete mode, showing the walls as perceived by the robot.

Future Work

- First, we would like to test our computer vision algorithm on larger mazes; doing so would allow us to determine whether or not our approach is robust enough for competition.
- Next, we would like to continue our work on the MicroMouse simulator. We'd like to develop more features, such as displaying the shortest possible path, to make it easier for developers to determine the efficacy of their algorithms.
- We would also like to continue experimenting with various closed-loop feedback systems, especially those that include advanced feedback mechanisms such as the use of a gyroscope and magnetic motor encoders.

Acknowledgements

We would first like to thank Dr. Jennifer Zirnheld for her guidance and unwavering support our three-year-long mission to build a winning MicroMouse robot. Without her, this project wouldn't have come to fruition, nor been nearly as successful as it has. We would also like to thank the Center for Undergraduate Research and Creative Activities (CURCA) for supporting the project financially. The funding we received allowed us to purchase the materials needed to develop and test the methodologies described by this poster. Lastly, we would like to thank all of our fellow IEEE members who contributed to the project. We couldn't have done it without them.

References

[1] Wan, Soon. "Micromouse Competition Rules." N.p., 2013. Web. 21 Mar. 2015.

[2] N. Chen, "A vision-guided autonomous vehicle: An alternative micromouse competition," IEEE Transactions On Education, vol. 40, no. 4, pp. 253–258, 1997.

[3] Hongshe Dang; Jinguo Song; Qin Guo, "An Efficient Algorithm for Robot Maze-Solving," *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2010 2nd International Conference on* , vol.2, no., pp.79,82, 26-28 Aug. 2010