Computer Vision With Scribbler Robots

Jillian Pyle And Clark Rinker
The Evergreen State College

June 3, 2010

1 Introduction

For the last five months we have been working on our project involving the scribbler robots. The goal was to program the scribblers to find each other and interact in a semi inelegant way. One reason behind picking this project was to learn about an aspect of computer science that we would not touch on in our studies in Student Originated Software. The project started with research in the fields of robotic behavior, including swarm robotics and basic interactions with each other. Then we moved onto computer vision, a topic that would take up most of our time. Lastly was mapping and navigation in a hope for the robots to inelegantly search for each other rather than blindly wandering around until they by random chance found each other. Swarm Robotics is a robotics paradigm where robots react to each other in order to accomplish a goal. Intelligence emerges from the swarm rather than from the individual robots. Using the Scribblers for swarm robotics however produces a few issues. Firstly, with only two robots it would be difficult to produce said swarm intelligence. While the scribbler robots are able to detect their environment, being equipped with a variety of sensors their ability to interact with their environment is somewhat limited, really only being able to drive around and beep. In particular, using the scribblers the project addressed two points of Artificial Intelligence:

1) Using the attached camera, is it possible for the Scribblers to "see" each other? 2) Using the radio and various sensors could the robots find each other within a room?

Scribbler Specification The Scribbler robots are battery operated cars created by Prallax Inc. The Scribblers come equipped with the following sensors: o Two binary IR sensors designed to follow a black line drawn on the ground o Two front mounted binary IR sensors for collision detection o A stall sensor that can detect if the robot is moving o Three front mounted continuous light detection sensors o A two-frequency tone generator o No odometer o Scribbler can hold a pen in center of robot (Thus "The Scribbler")

In addition the Institute For Personal Robots In Education (IPRE) that created the Myro interface adds the following functionality with their Fluke, an add on circuit board: o The capability to program the scribbler with the IPRE firmware wirelessly using a Bluetooth Radio. o Extra LEDs (front and back) for communication o Three infrared obstacle detectors o A battery voltage sensor o A 220x220 pixel color camera

Myro Specification The Myro interface is a framework for programming robots equipped with Flukes written in Python and developed by the IPRE. Myro communicates via Bluetooth with the Fluke using the Fluke Microcontroller's byte-code and is able to interface with the Scribbler to make use of its drive control and sensors. Functions include: o Duel and independent control of the drive motors o Readings from the various collision and light sensors o Producing pitches from the two tone speaker (including an interface for producing a string of notes from their associated letter pitches) o Color and Black and white photos.

A robot is initialized by first pairing the robot with the computer's bluetooth controller. The serial port associated with the particular piconet (see below for brief bluetooth specification,) is then sent to the myro's init function. The Python interpreter is then able to run the commands in the interpreter shell or take prepared Python scripts to execute a series of commands. Because the Myro framework uses global variables to store data about the scribbler multiple robots cannot be controlled within the same Python interpreter process. Thus controlling multiple Robots from the same computer becomes a inter-process communication problem, something briefly touched on by the scope of this project.

Bluetooth Specification In order to give some scope of the scalability of this project's work with the Scribbler robots a brief overview is given for the Bluetooth Radio Protocol. Bluetooth makes use of a portion of the Industrial,

Scientific, and Medical (ISM) 2.4GHz radio band and can transmit across 79 radio frequencies from 2402-2480 MHz. A master bluetooth controller can control up to seven slave devices in what's called a piconet and can use its 79 channels to transmit at speeds up to 1 Mbit/s. Thus this project could be scaled to seven scribblers pair to one computer running seven instances of Python running the Myro interface. Furthermore, multiple piconets can exist in the same area (thus why your phone doesn't receive data from someone else's cell phone) in time-sliced mode called a Scatter net. With the relatively low bandwidth needs of the Scribbler byte-code multiple piconets of robots could be run in the same area.

2 Related Work

Computer Vision Computer Vision encompasses the use of computer imaging to infer data including object recognition, event recognition, and modeling more. This project focuses on object recognition, using the Scribbler's camera to first search for specific colors and then identifying key points and attempting to infer if an image is a match. Because images are made up of an array of pixels some simpler image analysis could look for particular colors or combinations of colors to test if an image matches the desired search criteria. In the insistence of the Scribbler there happens to be a large green circuit board sticking out of the top of the robot, and thus an algorithm could be written that test if there were green pixels in the image. This however produces a few problems. 1) While most things aren't bright green (thus the idea behind Green Screens) different light levels will produce different pixel values and thus writing such an algorithm would be heavily based on empirical data gathered about the environment the robot is run in. Our environment, The Evergreen State College, also happens to be in a forest. 2) Sending a 220x220 color picture across Bluetooth and then looping through all of the pixels is time consuming. 3) And probably most importantly any image not taken of the front of the Scribbler wouldn't have any green in it.

Therefore simply looking for green is not general enough for the project's application.

Rather than looking for colors an algorithm that attempted to do object mapping could be used (looking for the shape of the circuit board, wheels, etc) Simple object recognition for exact matches of photos would also be easy, just match up point by point on the dot. However, looking for exact images

does not allow for partial occlusion or even the smallest amount of rotation and scaling, making it nearly useless in a practical environment. Rotation, scaling and occlusion can be accounted for by using Geometric Hashing and Linear Translation.

Geometric Hashing And SIFT The expensive problem of comparing two images can be somewhat mitigated by making use of Geometric Hashing. Rather then attempt to do object mapping by checking for exact images the image can be split into a grid of buckets. Key points in the picture are then hashed into these buckets on their x an y coordinates within the picture. Instead of storing and comparing entire pictures storage and comparisons are only made using full buckets, resulting in much quicker object recognition. Geometric hashing also allows for some degree of scaling and rotation, as objects that are closer or further from the image source within a threshold are likely to hash to the same bucket. One way these key points can be selected is using the SIFT algorithm that was published by David Lowe. The SIFT algorithm first identifies key points in an image and turns them into feature vectors. These vectors can then be stored and used in the linear transformation by using the vectors and transforming the other points in the image to match this vector. Once the points are transformed they can then be then run through the geometric hashing function in order to determine if the object is in the image.

Navigation Within the scope of the project navigation encompasses both obstacle navigation, the Scribbler being able to avoid objects and continue towards its destination and mapping, the Scribbler being able to plot a path between it's start and end location. Collision avoidance is handled using the Scibbler's onboard IR sensors. A call one of Myro's IR sensor functions returns an integer value specifying a distance, with 0 signifying no reflection detected (no object detected) and 6400 being a collision. For instance, a Scribbler that wanted to avoid running into anything could turn left every time it got too close to an object:

```
while(true): #Run Until the Robot is switched off
while(getObstacle("middle",0)<4000):#Read from the middle sensor
forward(5,2)#go forward for two seconds
turnLeft(.5,2)#Turn left for two seconds.
```

Mapping of an area can be done by creating a Roadmap. A Roadmap a graph whose vertices represent key points in an area and whose edges represent a distance and direction to the next vertex. In the case of the Scribbler, a graph containing a roadmap could be traversed to determine the path between two points in an area. This poses an interesting challenge for the Scribbler in a few areas: 1)The Scribbler has no compass, and therefore would have difficulty discovering what direction it was heading in. 2)The Scribbler cannot measure its speed; it can only measure how much power is going to the motors. Furthermore speed is affected by the battery life of Scribbler. The lack of these features on the Scribbler means that these features would need to be improvised from the Scribbler's IO devices, either through the use of the sensors or by marking up the area beforehand and relaying on image recognition. This project proposes solutions based on using the various sensors on the Scribbler.

3 Implementation

Object Recognition Testing for key points rather than colors allows for use of the grayscale mode of the camera and tests a much smaller subset of pixels, removing the two bottlenecks from the color recognition implementation. Eight images in total were used as the base case for the object recognition, taken at 45 degree angles of the Scribbler. Key points were then chosen using an interface mocked up that allowed the user to click on sections of the image and return pixels at those coordinate. The program was used to collect 15 key points, hashed into a 10x10 array representing the Geometric Hash buckets. Choosing the points allows for ease of testing: the same image can be fed to the algorithm to test for a match. In the case of this hashing function, collisions are good. The key points from a test image are hashed along the same function, and a collision results in a potential match. A collision results in a "vote" in favor of a match, with enough votes pointing towards a positive image match. Use of geometric hashing results in the case of this implementation 15 comparisons per base image; very inexpensive compared to comparing bitmaps. Images being tested in real-time however cannot make use of the interface user picking key points. Test images therefore have their key points chosen by SIFT and are then hashed against the base images.

Using Geometric Hashing With a working geometric hashing function we can use our stock images to have one scribbler attempt to identify the other. Because the camera can only take relatively low resolution photos (220 X 220 pixels) there will be a certain range that one scribbler will be able to "see" the other scribbler in. Through testing we can derive a distance that we can reasonably assume the geometric hashing function will be able to identify a scribbler. With this distance D we can define an area that, if both scribblers are inside, we will get a successful match. Our search pattern for this area is defined as follows:

1) Robots decide which one will search by generating a random number. (highest number searches) 2) The robot doing the searching will execute the following code:

```
found=false
timeout=false;#Robot will make one full rotation before timing out.
while(!timeout):
photo=takePhoto()
found=checkPhoto(photo)
  if(found==true):
  return true
else:
rotate()#scribbler will rotate half of a photo frame
checkTimeout()
return found
```

The scribbler will thus take a photo, check to see if it matches, and then if no match is found rotate half the width of its photo frame. If the scribbler makes one full rotation without finding the other robot it will time out and return a false, and go to another point in the room to check for other robot again.

Color Recognition The Myro interface provides its own set of functions for image processing that we used to attempt color recognition as an alternative to geometric hashing. Due to the fact that the room we were running the scribblers in was mostly the same color blue as the body of the robots, we decided to look for the green in the fluke board. This turned out to be much easier to understand, program, and implement then the geometric hashing. The algorithm that we use searches through all the pixels searching for a

specified range of color. It breaks the image into nine sections, and searches through those sections for specific colors. The reason behind the nine sections is if the color is identified in say the top right hand corner, we want the robot to turn a little to the right and then move forward. If the color was found in the middle, we want the robot to go strait. The search will go from the center out to the sides with each section being 74x74 pixels. If none of the specified color in the image is found, the scribbler will do a default search pattern (move forward until it gets close to an object, then randomly turn right or left), periodically taking pictures to search for the color again.

While the color recognition method had some success in identifying the target Scribbler it runs into a few pitfalls that make it a suboptimal solution to the image recognition problem: 1) The transmission and analysis of the color image is expensive. A 220x220 bitmap works out to be 57,600 pixels or 1.3MB in 24bit color. This translates to a bottleneck at the Bluetooth radio, which rarely seems to operate at its advertised 1Mbit/s. 2) For the algorithm to identify the target Scribbler it must be taken head on for enough of the circuit board to be in view. Two Scribblers already driving towards each other doesn't provide a very interesting computer vision problem. 3) The image capturing functions requires the robot to not accept commons while it sends the photo back to the computer. This slow reaction time translated into the robot running into things before it was able to make a decision based on the image test. 4) While the computer center mostly blue tones, any introduction of Evergreen State College memorabilia into the experiment produced false positives.

Implementing Navigation Navigation can be implemented using a Roadmap whose vertices represent circles whose radius is the distance that the Geometric hashing algorithm can identify a target Scribbler and whose edges represent a distance an direction to adjacent vertices. Two Scribblers looking to find each other in a room would begin by reporting their starting vertices and complete a graph traversal between their starting points. In order to prevent the Scribblers from missing each other it would be necessary for both to use the same traversal. Each Scribbler will then use local navigation to travel between vertices. Once the Scribblers arrive at the same vertex they can then perform a search using the Geometric Hashing Algorithm. In order to traverse two vertices the Scribblers would need to know their direction and their speed. However as the robots come equipped neither with a

odometer nor a compass these values need to be improvised using some of the other sensors on the Scribbler.

Speed on the Scribbler is controlled by sending a value from 1-5 to the movement functions which translate to an amount of power generated by the drive motors. However actual speed of the robot depends on the amount of battery life and the surface the robot is traversing. Using the voltage test function and timing the robots it would be possible to generate some average speeds for different battery charge levels. However these would be inaccurate if the robot was moved to a new terrain or after enough hair from the carpet gets wrapped around the drivetrain of the robot, making it pretty unreliable data.

Speed could be instead estimated by calculating a change in magnitude of the IR sensors on the front of the Scribbler. Assuming there are no moving objects in the area or readings that produce large jumps in magnitude are thrown out it would be possible to produce an average speed over a given interval. However the Scribbler would be unable to estimate its speed if crossing an expansive (in robot terms) area as the IR sensor would not return a value.

Using the Bluetooth radio's signal strength reading it is possible to produce a distance from the computer to the Scribbler Robot. The Bluetooth protocol allows a device to be paired with masters on different piconets using time slicing. By pairing a Scribbler with two computers there is enough information to produce a speed and a direction: Using the two signal strengths as radiuses two circles can be drawn whose intersects represent the two possible locations for the Scribbler. By placing the two computers against the edge of the test area one point can be discarded as it will always appear outside of the boundary. By Measuring the distance between the two computers and using the two signal strengths we can draw a triangle and calculate the angles between the three vertices. Two signal strength measurements over a time interval allows the calculation of a speed and a direction

While triangulation would actually yield enough information to show the locations of the Scribblers for the purpose of road mapping it will produce a speed and a direction for the robot to reach its next vertex. Within the scope of the project triangulation using the Bluetooth controller somewhat violates its intent as the goal was to simulate swarm robotics when computing their decisions on the computer and the distance between the two computers uses data that the Scribbler can't actually measure.

4 Results

Our work produced successful implementations of navigation of local obstacles, Color Recognition, and the geometric hashing function using the key point selection interface. Our color recognition implementation produced successful matches of the target Scribbler (as well as a few other green colored items) within a limited range taking pictures head on. Our geometric hashing function could compare two images without implementing linear transformation to compensate with large rotations and occlusion. We were unable to interface our code with the SIFT algorithm and thus key points for the geometric hashing function had to be selected manually. In addition we were unable to implement a Roadmap navigation algorithm.

5 Conclusions

In conclusion, we were not able to complete all aspects of our project that we had hoped due to time constraints and difficulties that we had not expected to encounter. If we could start the project over from the beginning, one hope would be to spend more time on the robot behavior rather than the computer vision part. Were we to continue with our project, the behavior aspect will be our next goal. We were able to learn much about the challenges and puzzles that are involved with robotics. Doing something that was mostly unrelated to our class subjects allowed us to explore different areas of computer science and give us a better understand of what is out there. We were able to learn the importance of having a clear and laid out plan when implementing a project and the team work and communication that is required when working with others.

6 References

Citations
1)Institute For Personal Robots In Education. "Myro Hardware." IPRE. March 6th 2009. Web. http://wiki.roboteducation.org/Myro_Hardware Accessed June 1st, 2010.

2)Institute For Personal Robots In Education. "Myro Reference Manual." IPRE. March 30th 2010. Web. http://wiki.roboteducation.org/Myro_Reference_Manual A

3)Wikipedia Users. "Bluetooth." Mediawiki Foundation. June 2nd,
2010.http://en.wikipedia.org/wiki/Bluetooth Accessed June 2nd 2010.

Bibliography

Wolfson J. Haim, Rigoutsos Isidore. Geometric Hashing: An Overview IEEE Computati

Lowe G. David. Object Recognition from Local Scale-Invariant Features.
Proc. Of the Internationa Conference on Computer Vision, Corfu
(Sept. 1999)