ECE/COE 1896

Senior Design

Swarm Robotics Final Report

Prepared By: Nick Davis
Joe Lynch
Austin Whited

Table of Contents

Table of Figures

1. Executive Summary

In this project, we implemented a swarm intelligence that will detect a fire and swarm it. Large geographic areas are notoriously hard to patrol and maintain. As a result of this, forest fires often spread without any kind of warning until it has caused irreparable damage. Our system will be able to patrol a large area for any stimulus that implies there is a fire. If any of our patrol robots see the stimulus, it will communicate the location with the other patrol robots and they will all swarm to that location. In this way we will demonstrate a swarm intelligence that will be able to find and react to a stimulus.

Swarm intelligence in a robotic context is defined as a collective behavior of decentralized, self-organizing systems. The key points in this definition are that the system needs a collective behavior and needs to be decentralized. The system must have a global behavior once a certain objective is complete, and a central controller can't make decisions for each individual node of the system.

We will have multiple robotic platforms equipped with sensors that will patrol our search area in order to find the stimulus. In our testing scenario, our robotic platforms will be searching for a spot of light from a flashlight that will represent our fire. We believe this is a good testing scenario because the diffused light beam and the fact that it is strongest at its center, closely resembles a fire. A fire's light is also most intense at its center, as is the temperature. Searching for this stimulus is each node's local behavior. There is no controller that tells each node where to search, or what to do in certain cases. Each node has complete freedom in their local behavior.

Once the stimulus is found, our robot platforms will all travel to the location. This is the collective behavior of the system. The platforms will switch between their local and global behavior without a central controller telling them to do so. They will complete their swarm behavior with the other platforms, then return to their local behavior when the task is complete.

We considered several different swarming search algorithms, but settled on the particle swarm (PSO) algorithm, as this algorithm is designed to continue to search the area even if the stimulus is found. This was the best fit for our forest fire scenario because there might be several hot spots, not just the main stimulus, so we wanted our platforms to continue to search.

Overall, we were able to implement the algorithm and demonstrate its ability to find the stimulus. There were some issues with losing the location data of some of the robots, the speed at which the secondary robots swarm after the stimulus is found, and collision avoidance due to the location data.

2. Problem Definition

In simple terms, the problem our project addressed was how to optimize searching in a 2 dimensional space. We operated under the assumption that multiple eyes are better than one when it comes to searching a space, so we applied this mantra to our design. Having multiple robotic platforms searching for the stimulus allows for a quicker response to the stimulus on average than if only one robotic platform was searching.

We had to implement a swarming search algorithm to solve this problem using robot platforms. In order to be considered swarm intelligence, each platform had to make its own decision while searching. We couldn’t use a central controller to tell each robot what to do in response to the stimulus. Each robot had to search the area, collect sensor data, send it to the central hub, and continue its search pattern. The central hub had to provide the location data for all the robot platforms, collect the sensor data and keep track of the global max. It had to send the location of the global max to all the platforms so they could adjust their search patterns in response.

3. Background

In 2016 there were over 67,000 fires across the United States, burning over 5.5 million acres and costing $1.975 billion (source). There is currently no good solution to reduce these numbers. The forest service is only able to react after much damage has already been done. We want to introduce a system that will search over very large geographic areas in search of a potential fire, swarm the fire in a containment behavior, and alert the authorities that a fire has broken out. This way, the swarm will provide some fire suppression until firefighters arrive and can fully contain it.

We believe swarm intelligent robots is the best way to accomplish this, as many algorithms have their basis in a swarm search pattern. We find that modifying a bee swarm algorithm or a particle swarm will be most effective as these algorithms are already suited for collaborative searching (*New Progresses in Swarm Intelligence Based Computation- Duan and Luo).* We will modify an existing algorithm to suit our needs, and so it may not follow exactly the natural model it was based off of. We will also have the advantage of modern robotic communication to make data transfer quicker than the natural model.

The particle swarm algorithm is based off of social foraging behavior of certain animals like flocking birds and schooling fish. The goal is to locate the maximum value of a certain parameter. Initially each particle (node) has its local max and a vector. After each cycle, the local maximum is updated, along with a global maximum that is known across all the nodes. Over time, the nodes will cluster around a local maximum, or several local maximum if they satisfy the objective. In this algorithm, we like the idea of having a local maximum and a global maximum. We believe this is the best way to compare the data received by each node and make decisions accordingly.

4. System Requirements

The following section details the requirements for a system that will monitor a geographic area and swarm to a fire. The robots will then alert an operator once it is verified to be a fire. This system will be used by any group (forest rangers, national park rangers, etc.) who have an interest in autonomously patrolling a large geographic area for fires before they get too powerful. The search function will require autonomous behavior, that is, without human input. The system must also alert the human monitor of the system if a fire is found, so the human may dispatch necessary resources.

4.1 The robots shall have autonomous behavior

Each unit will be autonomous and not require direction from the operator. Their behaviors are designed to be performed without human input, because that is the most effective way for the robots to survey a large geographic area.

4.1.1 Local Behavior

The local behavior of each unit is to randomly search the area for the fire. The robots will keep a record of the local maximum and global maximum of the temperature/light levels of the environment to determine when to converge. The robots will use a hybrid bee/particle swarm algorithm (PSO) to determine their local behavior. Each unit will perform their local autonomous behavior to converge on the stimulus, while also searching around for a higher value than the current global max. The theory of the local behavior is that when replicated multiple times in the same search domain, multiple robots will produce an emergent, Global Behavior.

4.1.2 Global Behavior

The global behavior of the swarm is that all robots eventually converge on the location of the fire. The robots will use a hybrid bee/particle swarm optimization algorithm. Each unit will actively search out a stimulus while tending toward the one with the current global max. When iterated over multiple times by individual robots they will produce a faster and more robust search than a single robot or non-swarming multiple robot implementation.

4.2 The robots shall have the ability to communicate

To make the swarm algorithms function, the robots must be able to communicate with each other. They must be able to communicate their current best sensor value, and their location to the rest of the bots in the swarm. They will also need to be given location data which will be supplied by the computer vision system on the central hub. This implementation will ensure that each robot is placed on the same X-Y coordinate system, and that they may locate each other using a common set of coordinates.

The units will communicate to each other and the central hub over WiFi. We used a raspbery pi single-board computer, running Ubuntu linux 16.04 lts, and ROS, the Robot Operating System. The linux operating system was chosen because of its robustness and control over the running services and system components. It made it easy to run only the programs we needed to run for the robots to function. Among those programs were the wifi driver, wpa_supplicant, and network interface provided by the Ubuntu. the linux kernel has drivers which allow us to utilize the wifi hardware on the pi. We used ROS to implement the application and transport layers of our system. Once all the robots can communicate over the wifi network, they subscribe to a ROS master and several ROS topics. A ROS topic provides a common medium for the various subsystems and robots to share messages with one another.

Each unit will have the ability to communicate their position and their sensor data. The particle swarm algorithm dictates that the units keep a record of the historical local and global maximums. The local maximum is kept in each individual robot while the global maximum covers all robots and will be shared to the other units over the network.

4.3 Each unit shall collect data with sensors

Each unit will have a temperature or a light sensor to collect data since heat and light are the 2 main outputs of a fire. For simplicity sake of this demonstration only 1 sensor will be included.

This decision has been made based on the testing of several different heat and light sources that light will be much safer to test with, and provides a nice, tightly bounded spot for a stimulus. If this were to be the real system, it would be helpful to have both types of sensors to distinguish between where there is just smoke (high temperature low light) and an actual fire (high temperature and light intensity). We will determine a realistic threshold of some source of light, then use that threshold to detect when there is a fire in our demonstration.

4.4 The system shall track each unit's position

The system will have some method of providing a relative or absolute position for each robot. This will simulate the gps location services that would be used on a full scale, deployed system. Each unit will use that information to traverse the search area.

We went with a computer vision implementation of this. Where a camera is mounted above the search area of the robots and computer vision software is run on the central hub computer to detect the robots. The software will describe the location of each robot and send it to the robots via several ROS topics.

4.5 The system shall have a central hub that displays information to the user

The central hub will receive sensor and location data from each of the robots. The central hub will combine this data and put it on a screen in a way that is readable to the user. The central hub will not use the data to command other units to one position.

We found that outputting data to the shell of each robot provided us with enough information for debugging, the values read by each robot will be displayed on the shell, which will be accessible by using tmux to attach to the running session. These shell sessions will be created by the launchfile.

5. Design Constraints

5.1 Robots shall operate within the range of location detection by the kinect and the wifi router.

Since the robots will communicate through 802.11n wifi, we will lay out our robot area within the range of the router. The camera has a maximum area of 8.16m x 5.29m, and the wifi-router, ideally, has a range of a circle with a radius of 32m.

5.2 The central hub shall have sufficient compute power to communicate with each robot and run the computer vision software

With in order to have sufficient infrastructure, we will need to select a central hub with enough processing power to communicate location data to and from the robots, as well as run the kinect tracking software.

Testing has shown that a laptop computer running a dual-core x86 architecture seems to work fine for this task.

5.3 The system shall be secure enough to deter basic attacks.

No matter the platform we use, changing passwords and using encrypted communication protocols will help deter cyber attacks. The wifi network and tcp/ip stack we are using is ubiquitous in modern device communication. While the tools are standard, they must be configured such that anybody can't spy on the network using default passwords or packet sniffing techniques.

We chose to implement custom passwords and several layers of encryption. WPA2 was used to secure the connection between router and devices, and 256-bit RSA keys were used to authenticate devices over ssh in lieu of passwords.

5.4 The sensors used shall adequately be able to detect a stimulus.

The sensors that we choose to detect light and/or heat, must have a wide enough input range and sensitivity to be able to show light differences with different levels and configurations of ambient light.

5.5 Each node shall be identical in design

This constraint helps define swarm operation. Each node should be identical in software and hardware, with the only differences being hardware identification stickers, and Physical and Logical network addresses (MAC & IP addresses).

6. Technical Standards

6.1 Standard Terminology for Driverless Automatic Guided Industrial Vehicles - Active Standard ASTM F3200

Using proper terminology is important to ensure quick and easy communication of ideas between professionals. Writing all of our design documents using the standard terminology is important to ensure the design is well received by our peers and to avoid any misunderstandings in intent or design. This standard provides the terminology and definitions necessary to have clear and concise communication between professionals.

6.2 IEEE 1609.0-2013 IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture

This standard is designed to meet the needs of mobile elements primarily in the transportation sector. Our application will not be in the transportation sector, but it is still relevant as we aim to make autonomous ground based vehicles that communicate wirelessly with each other. The standard enables the design of low latency and low overhead systems, which is important to our application.

6.3 IEEE 802.11ba - Battery Life Improvement

Because they are mobile systems, battery life will be an important factor for our robot platforms. At the same time, sharing data between the nodes is vital for swarm intelligence. To balance the battery life of our platforms and ensure timely communication between the nodes, we will need a low battery and low latency solution. This standard discusses how to achieve low latency and low battery consumption at the same time.

7. Evaluation of Design Concepts

Our system consists of two main hardware subsystems: the central hub and the robots.

The central hub consists of a laptop running linux connected to an Xbox 360 Kinect. The purpose of the central hub is to detect and calculate the location of each robot and act as a central server that facilitates the exchange of data between all of the robots.

Each robot is made up of a predesigned robot chassis from adafruit that consists of the robot body, two motors and wheels. A raspberry pi was used as the “brain” of each robot with custom designed electronics interfacing between the pi and the robot. The custom designed electronics includes a motor driver, RGB light sensor, 8xAA battery pack to power the motors, 5V UBEC, and a USB portable battery pack to power the pi.

The RGB light sensor was chosen after trying out both a temperature and light sensor. The temperature sensor was tested by using a hair dryer held roughly 3 feet above the sensor. The temperature sensor quickly registered the increase in temperature but upon turning off the hair dryer the temperature sensor took 10+ seconds for the temperature reading to return to room temperature. This was too long to provide accurate data for the robots so the RGB light sensor was used in its place. The R component of the RGB light reading was chosen since we are mimicking a fire using a flashlight and fire is mostly red.

The original robot design did not call for the rechargeable USB portable battery pack as the 8xAA battery pack was intended to power both the raspberry pi and the motors. However the batteries drained quicker than expected so we switched to a separate rechargeable battery pack to power the raspberry pi that would allow us to spend less on AA batteries.

The software for both the robots and the central hub was written in python and run on the Robot Operating System (ROS). The purpose of ROS is to simplify communication between the robots and the central hub. Each robot was connected to the same WIFI router to enable all ROS communication.

The modularity of the system created by separating each main software component into separate nodes is a key design feature that greatly simplifies future changes to the system. If we were to change the algorithm all that needs to be done is to create one new file that is run in place of the current PSO Algorithm node. As long as this new algorithm node still outputs the same x and y velocity vector the rest of the system would operate the same way as with the original PSO Algorithm node. If we were implementing the full fire detection and response system drones would be used instead of robots. In this case all the algorithm and data collections nodes would be unaffected since we would only need to change the Robot node and Robot Controller node to interface with propellers instead of interfacing with 2 wheel motors.

8. Team

8.1 Nick Davis - COE

I will be taking charge of researching and implementing the swarm algorithms. This includes the algorithms for the robot movement when randomly searching the area as well as the higher level movements when converging around the location of the fire.

I will also handle all reading of the local sensor data on each robot as well as calculating the global maximum sensor data.

8.2 Joe Lynch - COE

As a computer engineering student who is also pursuing a mechanical engineering minor, my interest and skills lie in the area of robotics. I will be taking charge of the robot construction, as well as the lower level robot motion and perception.

The lower level robot motion includes creating an easy to use interface and set of simple commands for the higher level algorithms to use. These simple commands (x + y velocity vectors) will then be translated into commands to be sent to each motor. The lower level motion is also responsible for obstacle avoidance and preventing the robots from crashing into each other and remaining in the search grid.

Robot perception includes each robot knowing where they are in space and in relation to every other robot. This was done with the use of the Xbox 360 Kinect and computer vision software to track each robot and calculate related distances.

8.3 Austin Whited - COE

I will be taking charge of the networking communication aspects of this project as well as the robot PCB.

The network communication protocols will define how the data flows between the parts of the system. I'll be responsible for realizing the arrows on the flow diagram in Figure 1 with a wifi network.

The robot PCB which I will design will connect directly to the GPIO pins of a standard raspberry pi board and provide the motor controllers, ADC, and sensor inputs to the pi.

9. Final Prototype

9.1 System Overview

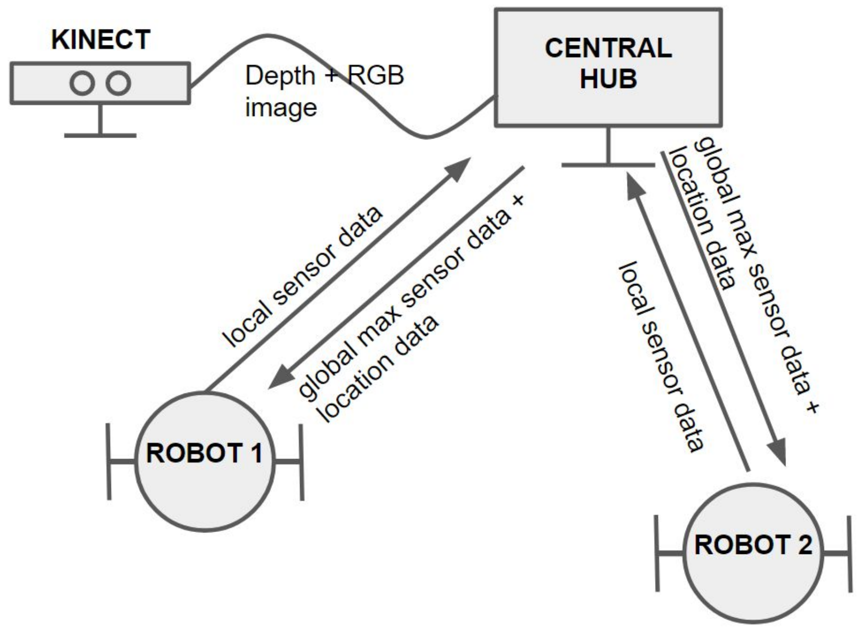

Figure 1: Design 1 Overview

Figure 1 shows the all the components in the system and what data is transferred between each component. The central hub receives the depth and RGB images from the Kinect camera which are used to calculate the location data of the robots. This location data is both sent to the robots and displayed on the central hub's screen.

Each robot collects local sensor data about its environment. This sensor data is sent to the central hub. The central hub also displays the sensor data on its monitor. The central hub calculates the global maximum of the sensor data and sends that value to each robot.

All data communication will be done over wifi. This requires each robot and the central hub to be connected to the same wireless network. The Robot Operating System (ROS) will also be used to facilitate the communication between devices.

9.2 Central Hub Hardware

The central hub contains no custom hardware. However it does consist of a laptop running the newest version of Ubuntu (16.04.3) and an Xbox 360 Kinect camera. The Kinect is connected to the laptop via USB.

The Xbox 360 Kinect will be placed directly above the search area to simulate a gps satellite that provides location data to each robot. In the real-world application each robot would contain a military grade gps to provide accurate location data. However that would not be practical to use in such a small-scale demonstration.

9.3 Central Hub Software

9.3.1 Software Overview

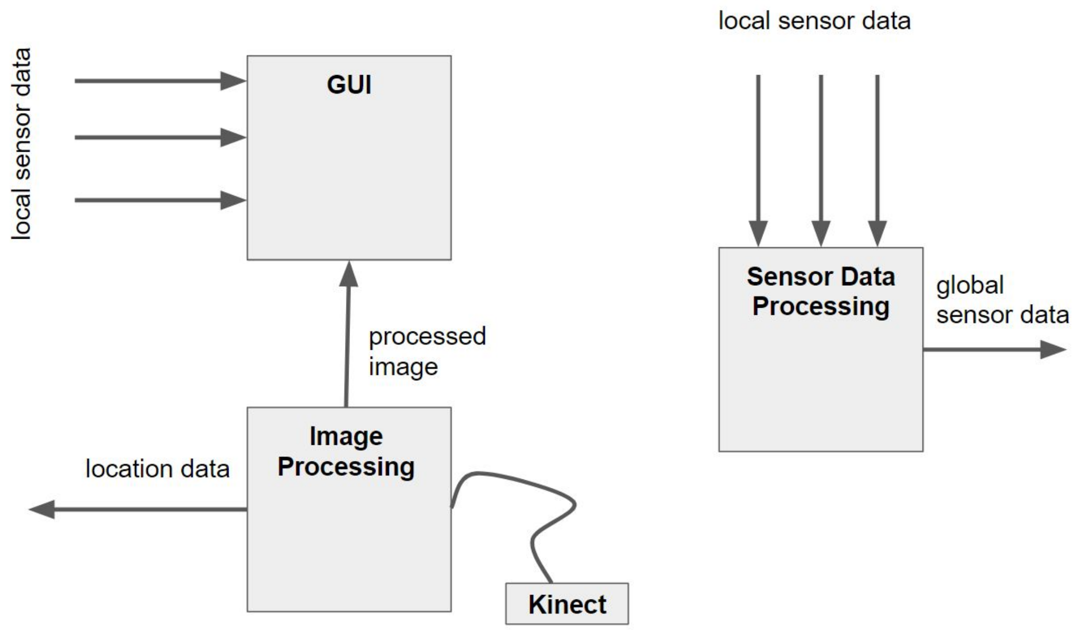

Figure 2: Central Hub Software Overview

Figure 2 outlines the custom ROS nodes that will be written for the central hub. Each box represents a node and the arrowed lines represent the data each node is publishing and subscribing to. The curved lines represent data that comes via a wired connection to an external sensor. In this case the only wired connection is the Kinect camera.

9.3.2 Sensor Data Processing

Topics Publishing To:

Global Sensor Data - global maximum value of the sensor data as well as the coordinates of the robot when it observed that sensor value

Topics Subscribing To:

Local Sensor Data - robot ID, sensor data and the coordinates of the robot when it observed that sensor data

Functionality / Algorithms:

This node will read all of the local sensor data and compare it to the current global maximum. If the current local sensor data value is greater than the global maximum then the global maximum data will be updated and that new data will be published on the Global Sensor Data topic.

9.3.3 Image Processing

Topics Publishing To:

Location Data - list of each robot ID along with the x and y coordinates and angle of each robot

Processed Image -RGB image taken by the Kinect with circles drawn on to show the location of each robot along with arrows showing the direction each robot is facing.

Topics Subscribing To:

RGB Kinect image - color image taken by the kinect

Depth Kinect image - point depth map taken by the kinect

Functionality / Algorithms:

The goal of this node is to calculate the location of each robot by identifying the uniquely colored isosceles triangle on each robot. OpenCV functions are used for the image processing. There are several steps to calculate this information:

1. Triangle shape detection

   The RGB image is converted to grayscale, blurred using a gaussian blur technique and a threshold is applied to create a binary mask of any colored shapes. The OpenCV function cv2.findContours is applied to the binary mask. Any contours without 3 edges are filtered out. This leaves a list of triangles and their pixel coordinates. These triangles are the pixel coordinates of each robot.

2. Color detection

   Several rgb pixel values from the center of each triangle are sampled and averaged together. These average pixel values are compared to the list of robot ID's and known triangle colors in order to determine which robot is represented by which detected triangle.

3. Coordinate Calculations

   The pixel coordinates of each robot must be converted into actual distance measurements in order for the robots to have accurate location data. The pixel height and width along with the Kinect's field of vision are known constants. The distance at all 4 corners of the image are calculated from the depth image. Multiple pixel distance measurements are averaged to reduce noise. Trigonometry is used to calculate an actual distance measurement per pixel. This calculation is used to convert the pixel coordinates into distance coordinates.

9.4 Robot Platform Hardware

9.4.1 Robot Platform

Figure 3: Robot Chassis from Adafruit

We will use the predesigned robot chassis kits shown in Figure 3 to allow our sensor nodes to move around and demonstrate the swarm algorithms. The benefit of using a predesigned chassis is to allow us more time to focus on the implementation of the actual algorithms.

9.4.2 Robot Electronics Overview

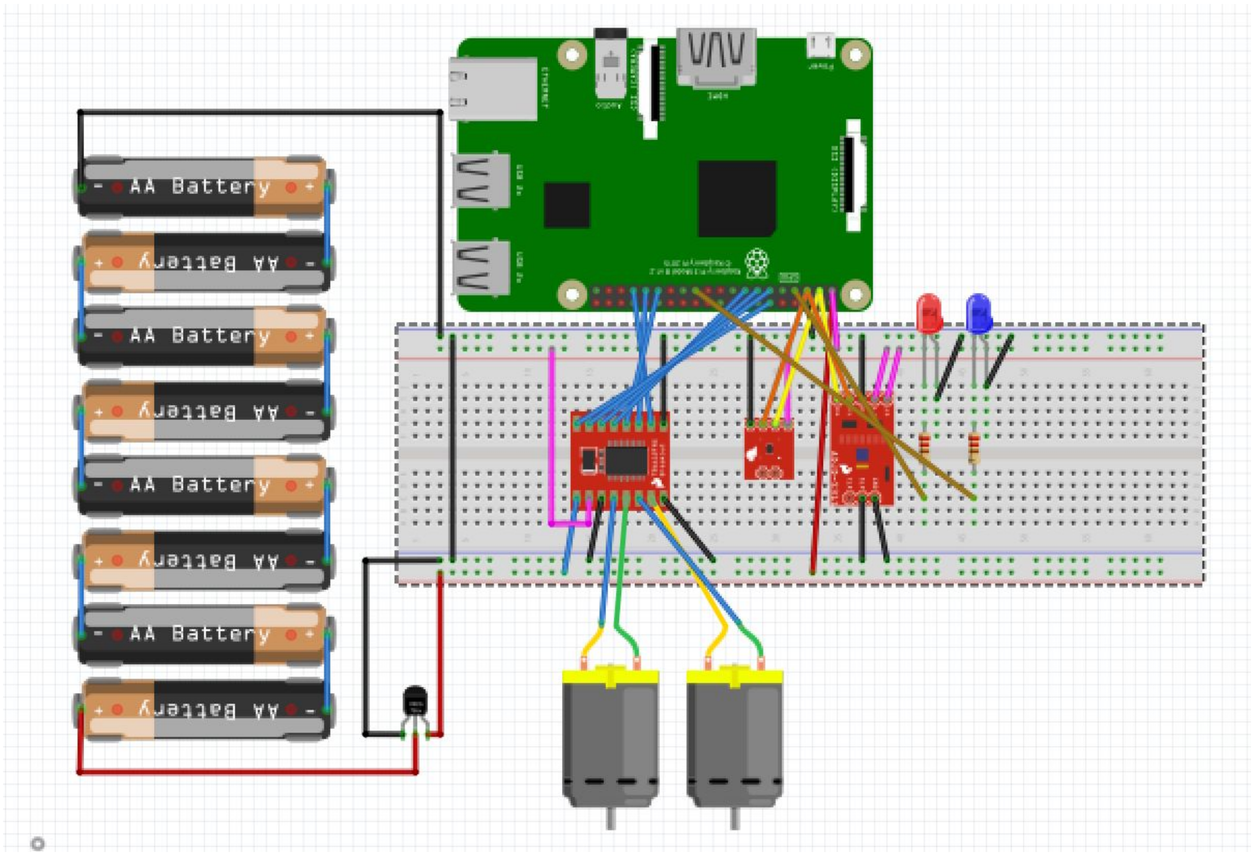

Figure 4: Robot Electronics Schematic

Figure 4 shows an overview of each electrical component in the robot and how they are connected. Our breadboard prototype is shown. The same above components were wired and soldered onto a protoboard that was plugged directly into the Raspberry Pi. A custom PCB was also created however it was not actually used on the robots due to manufacturing issues.

9.4.3 Environmental Sensing

An RGB light sensor was used on each robot to detect a “fire” since the 2 main sensory outputs of fire are light and heat. The R component of the detected robot was used to measure the strength of the fire since forest fires are mostly red and orange. In order to simplify the algorithm and the calculation of the global and local best data points, only the light sensor was used and not both a light sensor and temperature sensor.

A standard flashlight was used as the simulated fire and was detected by the RGB sensor.

9.4.4 Robot “Brain”

A raspberry pi will be used as the main “brain” of the robot. The benefits of using a raspberry pi are to be able to run linux so that we can use the Robot Operating System (ROS) in order to facilitate communication between each robot. The 40 exposed GPIO pins also provide a simple way of controlling our motors and sensors.

9.4.5 Motors + Motor Control

The robot chassis kit contains two DC motors. A single dual motor driver is included in the design to provide control of the motors. The motor driver connects to the Raspberry Pi’s pwm pins to allow for operating each motor at various speeds.

9.4.6 Battery Power

Each robot contains an 8 AA battery pack that powers the motors. A UBEC is connected between the battery pack and the robot to step the battery pack voltage down to a consistent 5 Volts. A portable USB battery pack was also added to the robot in order to power the Raspberry pi.

9.5 Robot Platform Software

9.5.1 Robot Platform Software Overview

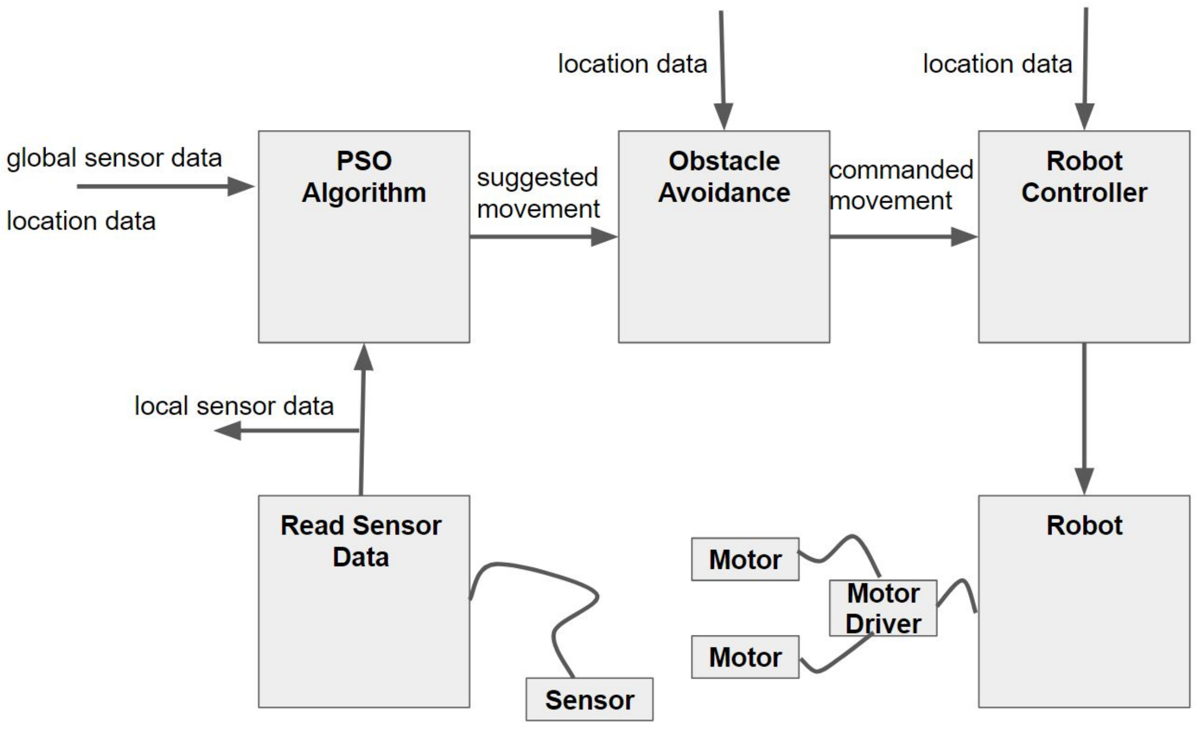

Figure 5: Robot Platform Software Overview

Figure 5 outlines the custom ROS nodes that will be written for the central hub. Each box represents a node and the arrowed lines represent the data each node is publishing and subscribing to. The curved lines represent data that comes via a wired connection to an external sensor. For the robot these external sensors include the light/temperature sensor, the motor driver, and 2 DC motors.

9.5.2 Read Sensor Data

Topics Publishing To:

Local Max Sensor Data - robot ID, local maximum sensor value and the coordinates of the robot when it observed that sensor data values

Local Sensor Data - robot ID, local real time sensor values and the coordinates of the robot when it observed that sensor data values

Topics Subscribing To:

None

Functionality / Algorithms:

The Read Sensor Data node will periodically read in new temperature/light data via the Raspberry Pi I2C Bus. This node will keep track of its own local sensor data maximum. On each successive read of sensor data the node will compare the newest data to the local maximum. If the maximum value changes, this new data is then published on the maximum local data topic.

Every read of sensor data is published on the Local Sensor Data topic regardless of whether it is a new local maximum.

9.5.3 PSO Algorithm

Topics Publishing To:

Suggested Movement - an x and y velocity vector that the robot should implement if it is able

Topics Subscribing To:

Local Max Sensor Data - robot ID, sensor data and the coordinates of the robot when it observed that sensor data

Global Sensor Data - robot ID, sensor data and the coordinates of the robot when it observed that sensor data

Location Data - robot ID, x and y coordinates, robot angle

Functionality / Algorithms:

This node controls the high level behavior of each robot using the PSO algorithm [1]. This algorithm controls the x and y velocities of the robot when searching for the fire in a pseudo random function that also takes into account the coordinates of the local and global historical best sensor data. That function is:

Vx = Vx + 2*rand()*(localBestX-currentX) + 2*rand()*(globalBestX-currentX)

Vy = Vy + 2*rand()*(localBestY-currentY) + 2*rand()*(globalBestY-currentY)

Where rand() is a random number generator that outputs a number between 0 and 1. The purpose of the random number generator is to introduce some randomness that increases the likelihood of the entire search area is covered. Without it, it is likely that the swarm would settle on local minima instead of finding the global largest fire.

Testing of the algorithm will need to be done in order to determine when to switch between the local and global behavior. The global behavior differs between the minimum project standard and the goal project result. For the minimum project standard all robots will stop once one robot found the fire. Testing will need to be done to determine the threshold of the sensor data that constitutes finding the “fire” and switching from the local PSO search behavior to the global behavior of stopping the robots. For the goal project result the local behavior will continue until all the robots have converged on the fire location. All of the robots will then stop to signal that the fire has been found and contained.

We will also determine if any changes of the PSO search algorithm are needed to optimize it for our system through testing. A possible variation may be to add the sensor data values into the PSO velocity equations instead of just the location of the sensor values. The purpose of adding in the sensor data values is to add a larger emphasis to the global best position when there is a larger difference between the local and global values.

The new commanded x and y velocities are then published on the Suggested Movement topic.

9.5.4 Obstacle Avoidance

Topics Publishing To:

Commanded Movement - an x and y velocity vector that the robot must implement

Topics Subscribing To:

Suggested Movement - an x and y velocity vector that the robot should implement if it is able

Location Data - robot ID, x and y coordinates, robot angle

Functionality / Algorithms:

This node ensures that robots will not crash into each other and will not exit the search area. The suggested velocity and current location of the robot is used to calculate the next position of the robot. If this next position is outside the search area then the suggested velocities will be changed to keep the robot in bounds. If this next position is too close to a current position of another robot then these velocities will also be changed to avoid a collision.

This node will do everything it can to implement the suggested velocities but priority will be given to keeping the robots operating in a safe manner.

9.5.5 Robot Controller

Topics Publishing To:

PWM Wheel Commands - a PWM command (0-100%) for each wheel on the robot

Topics Subscribing To:

Commanded Movement - an x and y velocity vector that the robot must implement

Location Data - robot ID, x and y coordinates, robot angle

Functionality / Algorithms:

A reference angle vector will be calculated based on the commanded x and y velocities. This reference angle will be along with the actual angle of the robot will be passed into a PID loop to determine how fast each wheel should be spinning relative to each other. Spinning the right wheel faster will turn the robot to the left while spinning the left wheel faster will make it turn right.

The PWM commands for both wheels will be published to the PWM Wheel Commands topic.

9.5.6 Robot

Topics Publishing To:

None

Topics Subscribing To:

PWM Wheel Commands - a PWM command (0-100%) for each wheel on the robot

Functionality / Algorithms:

This node controls the physical motors on the robot. This node reads in the PWM commands and sends them to the motors. Based on the commanded PWM value, the mode of each motor is also set. The possible modes of each motor is clockwise, counter-clockwise, brake and coast.

10. Testing, Data Analysis and Results

To test our system, we made several diagnostic tests and a few global functioning tests

Diagnostic tests
Motor Driver test - In the motor driver test, we verify the output of the motor drivers using a program which sends 50% wheel speed to the motor. We then used an oscilloscope to verify the pwm signal had a 50% duty cycle.

Computer Vision test - To test the computer vision, we manually measured a few locations of the robots and the size of the search area, then compared them with the measurements made by the computer vision system. Values given by the computer vision were within 5cm of our measured values.

Sensor test - for our sensor test, we check to see if the light sensor is working on each robot. to do this, we run a program which will output the value of the sensor to the terminal, a group member shines the light on the light sensor in question. The sensor passes the test if the lux value increases sharply when the light is shone upon it.

Network connectivity test - the network connectivity test will first send an ICMP ping request to all robots, when they all respond, they have passed the first part of the test. The second part involves each node writing data to a particular ROS topic, if all the messages were received on the ROS master, then we have full network connectivity.

Integration tests
Collision Avoidance test - Robots execute random movements. If they do not hit each other after 5 mins. Collision avoidance is working properly. This will test the computer vision, motors, and ros communication between robots and the central hub. We found that our system will pass the test if ambient lighting conditions are not too bright or too dark.

Boundary test - Another test for computer vision, motors, and ros communication is to see if the robots stay within the predefined area after trying to send them out. This test will be considered to pass if all robots stay within the boundary for 5 minutes.

Algorithm test - The full algorithm was run several times with different sized areas of light and different locations of the light source. The time for the robots to swarm to the light was tracked and the results are listed below in table 1. Each run corresponds to a video of a test trial [2]. Table 2 details what test was run in each video test trial.

Video	Time swarming before light introduced	Time for 1st robot to find the light after light is introduced	Time for all to swarm to the light after it was found by 1 robot
SwarmDemo1	16.71	1.5	70
SwarmDemo2	25.82	1.09	2.12
SwarmDemo3	0	5.87	18.18
SwarmDemo4	0	16.65	10.86
SwarmDemo5	N/A	N/A	15.12

Table 1: Algorithm Stats (All times in seconds)

Video Title	Notes
SwarmDemo1	This was an early test before the algorithm had been refined.
SwarmDemo2	This was with only 2 robots and they had somewhat already swarmed together based on the room ambient light maximum before the light was introduced.
SwarmDemo3	An ideal test of 3 robots.
SwarmDemo4	An ideal test of 3 robots.
SwarmDemo5	Demoing all 3 robots swarming to the room maximum.

Table 2: Video Notes

11. Project Timeline

1st Graded Checkoff / Demonstration

Austin:

- Get raspberry pi's installed with Linux + ROS
- Get raspberry pi's connected to wifi network
- Communicate to a robot via ssh & tmux from the central hub

Joe:

- Move robots in a straight line
- Kinect setup on central hub
- Simple image processing proof of concept – color blob detection to identify a robot

Nick:

- Collect sensor data from environment
- Send sensor data to central hub – for demonstration purposes
- Setup a test flashlight/lamp to mimic light from a fire
- Setup a hairdryer or other heat source to mimic heat from a fire

Overall:

- Show sensor data changing as the robot moves in a straight line towards the stimulus
- Decide which stimulus to use and the corresponding sensor
- One full robot setup on a breadboard with sensor(s) and motor drivers

2nd Graded Checkoff / Demonstration

Austin:

- Full PCB Layout
- More automation of wireless network
  - hostfiles
  - public key authentication
  - move network manager from GUI to terminal based, using config files.
- Basic ROS communication between devices

Joe:

- Full robot location
  - Eliminate noise from computer vision to prevent finding false triangles
  - Detect color of triangles to distinguish between robots
  - Translate pixel location to actual distance location
- Move robots based on simple simulated command types from algorithm node
  - These simple commands will just be velocity and NOT turning the robot to face a certain angle
- All robots assembled with breadboards

Nick:

- Decide on temperature vs light sensor and light/heat source
- Simple Central Hub to send sensor data via ROS
- Simple fire detection - stop robot when the stimulus (light/temperature) is over a certain threshold

3rd Graded Checkoff / Demonstration

Austin

- GUI for central hub
    - for each robot
        - lux value
        - location
- Launch files for individual robots
    - remap (names) topics for individual robots.
- Create scripts for batch configuration
    - environment variables

Nick

- random search pattern for each individual robot.
- Have all robots stop when the stimulus is found.

Joe

- Turn robots to specific angle
    - The goal of this is to be able to receive commands from the algorithm node and move around the search grid
- Keep robots in the search area + in view of the kinect camera
    - Override the commands from the algorithm node if the robot is about to leave the search area

12. Conclusions and Future Work

Ultimately we accomplished the goal we set out to accomplish which was to demonstrate an improvement over a basic random search and demonstrate basic swarm intelligence. The robots are able to exchange data between each other and alert others when they've discovered the stimulus (in this case the light from the flashlight). Once a single robot has discovered the light it has been shown that all other robots can converge on the location of that single robot.

One current issue with the system are the fact that the computer vision system is dependent on the ambient light levels of the room. A potential fix to this is converting the RGB image to an HSV image. The HSV image is less dependant on the ambient light so it would be potentially easier to provide accurate thresholding when isolating the triangles on top of the robots. Another issue is that the algorithm does not take into account the actual data reading at the maximum location so if the actual light source is not found quickly the robots tend to swarm to the location of the maximum ambient light level of the room. The cardboard plates placed on top of the robots sometimes interfered with the operation of the light sensors by covering them up. A 3d printed bracket could have been made to hold the cardboard plate in place and ensure the light sensors could accurately sample the light.

However, there were some goals we had for the project that we did not get a chance to tackle due to the large overhead/setup that this project required. Some of these things included building small robots that could carry out commanded movements as well as reading in sensor data from a light sensor. Our project was not building a robot but we did need to build robots in order to demonstrate that the robots could swarm to the location and that our swarm algorithm was functional.

Some of those extra items we wanted to implement that would be implemented in a second semester of senior design are an improved initial random search algorithm, implementations of different swarm algorithms and tailor the algorithm more towards fire detection and suppression.

For the initial random search location the algorithm could be changed to give little weight to the location of the global max if the data value of the global max is below the threshold. The robots could also take into account the distance away from other robots to ensure the robots are spread out as much as possible around the search area.

We would also like to implement different swarm algorithms in order to compare and contrast the response time and accuracy of the robots swarming to the “fire”. A completely random search would be implemented to determine a baseline response time for all robots to find the “fire”. The bee algorithm could also be implemented to compare to the PSO swarm algorithm. In the bee algorithm, once the stimulus is discovered all robots will immediately take their shortest path to the stimulus location. This will likely improve the response time but will likely make the accuracy lower if there is multiple fire locations.

In order to improve the response to a fire the robots should evenly spread out in a circle surrounding the fire to speed up putting it out and to contain the fire. Currently the robots do not take into account the location of other robots when swarming to the fire.

13. References

- [1] *Particle Swarm Optimization* by James Kennedy and Russell Eberhart
- [2] Swarm Demo videos included in the project submission folder.