AE 5098 - Directed Research

Estimation of Spaciotemporal Fields with Mobile Sensing Robot

Dr. Michael Demetriou and Dr. Nikolaos Gatsonis

**Project Overview**

This research project aimed to develop a platform for conducting real-world experiments with a simulated dynamic threat field based on solutions to a 2D advection-diffusion field. An array of projectors was set up to project a composite image of the threat field onto the laboratory testing grounds. This design allows for the creation and testing of various dynamic fields, where the colors of the projected image directly correlate to the gas concentration of the solution of the field. A mobile robot is then equipped with four camera systems that are capable of measuring and interpreting concentration readings from the color images taken which can then be used for targeted navigation or derivation of the original parameters of the threat field.

This document contains detailed information about the various subsystems mentioned in the introduction. A quick start guide is provided at the end of the document which gives instructions for starting up the system and running an experiment. All the code and other files for this project can be found on GitHub: https://github.com/WPI- AVMI/threat-field-simulator.

**Equipment/Login Information**

**Wi-Fi Network**

The NETGEAR R6080 Wi-Fi router in the lab was configured to provide a network for this research project. The dual-band router provides both 2.4G and 5G connectivity. The login information for the network is provided below.

Table 1: Wi-Fi Network Information

|  |  |
| --- | --- |
| **SSID** | **Password** |
| AVMI-LAB | letmeinplease |
| AVMI-LAB-5G | letmeinplease |

The network settings can be configured by opening the page at IP address 192.168.1.1.

Static IP addresses were assigned for the various devices on the network for convenience. A list of these IP address reservations is given below. These devices can be easily accessed from the rugged laptop via SSH.

Table 2: Static IP Address Assignments

|  |  |
| --- | --- |
| **Device Name** | **IP Address** |
| AVMI-LAB-RUGGED | 192.168.1.10 |
| AVMI-LAB-01 | 192.168.1.11 |
| AVMI-LAB-02 | 192.168.1.12 |
| AVMI-LAB-03 | 192.168.1.13 |
| AVMI-LAB-04 | 192.168.1.14 |
| AVMI-LAB-CAM-SYS | 192.168.1.15 |
| AVMI-TB4-01-RPI | 192.168.1.20 |
| AVMI-TB4-01-CREATE3 | 192.168.1.21 |

**Rugged Laptop**

The control computer for the experimental setup is a Dell Latitude 7330 Rugged Extreme Laptop. The laptop has been configured with a dual boot of Windows 11 and Ubuntu 22.04. The login information for this computer is the same for both operating systems.

Table 3: Rugged Laptop Login Information

|  |  |  |
| --- | --- | --- |
| **Device Name** | **Username** | **Password** |
| AVMI-LAB | AVMI-LAB-USER | letmein |

Everything for this research project was done within the Ubuntu environment. Ubuntu 22.04 is a Linux distribution which is compatible with ROS2. ROS is a middleware that is used for wireless communication with mobile robots over the network. ROS2 humble is the distribution used in this project.

**Raspberry PIs**

Table 4: Raspberry Pi Login Information

|  |  |  |
| --- | --- | --- |
| **Device Name** | **Username** | **Password** |
| avmi-lab-01 | pi | letmein |
| avmi-lab-02 | Pi | letmein |
| avmi-lab-03 | Pi | letmein |
| avmi-lab-04 | pi | letmein |
| avmi-cam-sys | avmi-cam-sys | letmein |

**Threat Field Image Generation**

The images representing the threat field are created in a Dell Latitude 7330 Rugged Extreme Laptop that serves as a base station for all the heavy computing needed for simulation and navigation. They are generated using the Threat Field Generator.py script; this script imports many helper functions from the file GCS Functions.py. It works by calculating a 2D solution to the advection-diffusion equations for a fluid field. The iterates over a 1920x1080 image and calculates the concentration of the field at any given point based on the initial parameters describing the position, strength, and movement of the source within the field. This concentration takes the form of a normalized value between 0 and 1 for every pixel with the image array. These values are then mapped onto RGB with the red part of the spectrum representing higher concentrations and violet purple representing lower concentrations, the values are then taken to produce a PNG image representing the computed field. This process is repeated every 5 seconds with a different time step to show the progression of the field over time.

After generating the visualization of the thread field, some additional processing is required to display the field on the projectors. The full image is divided into four quadrants and each component image is saved with a different filename and sent directly to the Raspberry Pis mounted to the projectors. The images for the top quadrants (projectors 1 and 2) are flipped 180 degrees because they will be projected from the opposite side of the workspace as the bottom quadrants (projectors 3 and 4).

**Projector System**

The projector system comprises four Optoma ZH450ST projectors which work are interconnected through a local network to display one large composite image onto the floor in the laboratory test area. The projectors are mounted onto large camera tripods arranged around the perimeter of the workspace facing downwards at a 90-degree angle. The protectors are numbered 1 through four and are placed in a counterclockwise order in the field. The Optoma ZH450ST projectors have a resolution of 1920x1080, a throw distance of 1.3' - 10.5', and an aspect ratio of 16:9.They are placed in 3 DOF mounts that can be easily adjusted to align the projectors properly to display an unwrapped image of the original threat field.

Each Projector is controlled by a Raspberry Pi 4 model B connected via HDMI. The Raspberry Pis are mounted along a leg of the tripods with 3d printed clamp mounts, which also hold the power supply and extension cords required to power the system. The CAD files for these mounts are included in the GitHub repository. The Raspberry Pis are intended for headless operation and are controlled remotely from a laptop that serves as the base station. They must be powered via the USB-C port.

The system works by generating a color image of the threat field on the base station laptop based on the needed initial parameter of the experiments, splitting the images generated into 4 separate files using a Python script that corresponds to the four quadrants covered by the projectors. Then using the SCP protocol, it sends the images directly to the Raspberry PI overwriting the previously displayed image. The Raspberry PIs each run a script that searches for the image in its file systems and displays it in Fullscreen on the projectors with the FEH image viewer application. The PIs will continually check the folder and refresh the display when a new image is uploaded to allow for dynamically changing fields.

**Camera System**

**Hardware:**

The Camera System is comprised of a Raspberry PI 4 mounted on top of the mobile robot connected to four Logitech C920x webcams. The system is powered by a portable power bank which powers the Raspberry PI via the USB-C port. The two portable power banks in the lab are interchangeable so one can charge while the other is in use.

The cameras are connected to the USB-A ports on the Raspberry PI and should be automatically recognized by the system. Afterward, the camera settings such as white balancing and autofocus are edited using the v4l2Linux program, which runs automatically on setup. The webcams are secured to the Turtlebot4 with 3D-printed mounts that attach to the screw holes on the robot chassis intended for the standoffs for the optional mounting plate. The Raspberry PI for the camera system is also mounted on this structure. The camera mounts were designed such that the four cameras are arranged in a radially symmetric pattern about the robot, each an equal distance from the center point directed at 15-degree angles and lowered to a height of 50mm from the top of the robot. The cameras are numbered 1 through 4 and correspond to the direction of the robot for navigation. To

check whether the webcams are properly connected to the raspberry pi, SSH into the pi and use the command ”v4l2-ctl –list-devices” to show the connected USB devices. The four webcams should be shown at the bottom of the list. Each device will be listed as ”UVC Camera” and should list a path of the form ”/dev/videoX.”

**Camera System ROS node:**

The Raspberry PI mounted on the camera system is running the ubuntu operating system with ROS or robot operating system for interconnection with the laptop base station as well as the mobile robot itself. This system has a ROS workspace in the ”ros2 ws” folder and runs a ROS node called “cam\_sys” which contains a ROS service called “get\_color\_vals” that when called accesses the cameras attached, and captures an image using the OpenCV library, The script then computes the average RGB values for a square of pixel values located at the center of the image. The RGB values are then converted to HSV, from which the hue value is extracted and normalized between 0 and 1. This is effectively an inverse of the encoding process used to generate the image of the threat field. The process is repeated for all four cameras, and the values are returned to the node that made the service call.

Starting the cam-sys node remotely from the rugged laptop can be accomplished by opening a new terminal and running the following commands:

$ sudo su

$ cd ros2 ws

$ source /opt/ros/humble/setup.bash

$ source install/setup.bash

$ ros2 run avmi lab cam sys cam sys

**Mobile Robot**

**TurtleBot 4:**

This project makes use of the Turtlebot4 mobile robot platform. The Turtlebot4 is a two-wheel, differential-drive robot. The robot has two onboard computers which are a Raspberry Pi 4 and a Create3. These devices must both be connected to the Wi-Fi network to operate the robot. The Turtlebot4 uses ROS2 Humble to communicate with the rugged laptop. The Turtlebot4 automatically launches the bring-up node automatically upon power up, so the user does not need to run any nodes on the robot manually. The robot communicates with remote devices over ROS topics to report information from sensors and to receive actuation commands. The Turtlebot4 may be teleoperated with the game controller joysticks as explained in the TurtleBot documentation.

**ROS Platform:**

The robot is controlled by running ROS nodes on the rugged laptop. The ROS workspace on the laptop is contained in the folder ”turtlebot4 ws”. This workspace currently contains two ROS packages. ”avmi lab interfaces” contains custom message and service definitions that are used within other packages in the workspace. The ”avmi lab threat field” package contains the nodes used for controlling the robot for various experiments. The simple demonstration of the simulator capabilities makes use of the ”threat field navigator” node which is contained within this package. ROS nodes can be run on the robot by opening a terminal and executing the command $ ros2 run package name¿ ¡node name¿. An example of this is shown in the quick start guide at the end of this document.

The threat field navigator node provides basic navigational abilities and lots of important underlying functionalities. This node sets up publishers and subscribers to interact with the robot. Specifically, the node subscribes to the ”/odom” topic to get information about the robot’s current state and publishes to the ”cmd vel” topic to send movement commands. Some basic movement functions have been written to command desired velocities and to facilitate more complex maneuvers like driving to a specified point in the workspace. This node also sets up a service call to the get color vals service provided by the camera system and provides a handler to process the camera data upon receipt. For any future work, it is recommended to start from a copy of this node to maintain these critical functions as described.

The goal of the threat field navigator node is to direct the robot to the maximum of the threat field. This is accomplished by using data from the camera system to essentially measure the local gradient at the robot’s current position in the field. From the four camera readings, the program selects the two highest readings and performs linear interpolation to compute a new heading direction in the direction that the value of the field is increasing. The robot then performs a point turn to orient itself in this direction and drives forward for a short distance. This process is repeated such that the robot will make a service call to the camera system to sense the field, perform some computation, then execute a movement to get closer to the goal. This functionality is shown in the simple demonstration videos in the GitHub repository.

There is significant room for improvement in the performance of the mobile robot. More sophisticated motion control techniques and navigation algorithms could be implemented to improve driving performance. Termination conditions could also be introduced to stop the robot after some goal is accomplished. Many variations could be made to this starter code to achieve the desired functionality needed for conducting various experiments with this real-world simulation platform

**Simulated Environment**

Lastly, there is the fully simulated environment component of the experiment. The objective of this full simulation was to test the viability of the underlying navigation and sensor reading components of the system.  Through this simulation we were able to verify that the navigation algorithm of the threat field is both accurate and reliable in any different positions and initial conditions, therefore most improvements that can be made would be in terms of the physical hardware of the experiment itself. The simulation runs in a ROS-based physics simulator called Gazebo where a virtual test environment was set up with a static projected field as the mesh for a ground plane, which would then be read by virtual cameras mounted upon a model of the TurtleBot 4s used in the experiment itself. With this configuration of the simulation, it is not only possible to test the fully virtual behavior of the system but also to visualize the response of the physical experiment due to the interconnection of the ROS nodes being identical for both the simulation and the actual robot. This portion of the project was fully my addition and was part of my responsibility during the duration of the fellowship.

**Running an Experiment**

This section provides a quick start guide to running an experiment with this simulation platform. Conducting an experiment can be done in three simple steps as described below.

1. **Start the Experiment:** Power on all four projectors and raspberry pis. From the rugged laptop, plug power bank into the robot mounted raspberry pie and open a terminal and change to the Scripts directory ($ cd Documents/Scripts). Then run the shell script to start up the display with the command $ ./init\_threat\_field.sh. If any errors occur where the images are not found, power cycling the raspberry pis should fix the issue. The experiment can be shut down by running the script $ stop\_trear\_field.sh. if more specific systems are needed run step 2,3, and 4.
2. **Start the projector system**: Power on all four projectors and raspberry pis. From the rugged laptop, open a terminal and change to the Scripts directory ($ cd Documents/Scripts). Then run the shell script to start up the display with the command $ ./init projectors.sh. If any errors occur where the images are not found, power cycling the raspberry pis should fix the issue. The projector system can be shut down by running the script $ stop projectors.sh.
3. **Start the camera system**: Connect a power bank to the raspberry pi for the camera system. Allow it to connect to the lab Wi-Fi network and open a terminal and change to the Scripts directory ($ cd Documents/Scripts). Then run the shell script to start up the display with the command $ ./init\_cam\_sys.sh.
4. **Start the Dynamic Threat field:** from the rugged laptop, open a terminal and change to the Scripts directory ($ cd Documents/Scripts). Then run the shell script to start up the display with the command $ ./ini\_dynamicfield.sh.
5. **Run a program on the robot:** Run a program on the robot. Place the Turtlebot4 on the floor of the workspace. Open a new terminal on the rugged laptop and run the ROS node for whichever program you would like to execute. For example, the simple demonstration program which navigates the robot to the maximum value of the threat field can be run with $ ros2 run avmi lab threat field threat field navigator. The robot program can be stopped by interrupting the terminal with ”ctrl” + ”C”.