CS437 Midterm Report

Prithvi Balaji
University of Illinois at Urbana-Champaign
pbalaji3@illinois.edu

Joseph Stack
University of Illinois at Urbana-Champaign
jkstack2@illinois.edu

1 INTRODUCTION OF COMPETITION

We participated in a competition that aimed to advance indoor localization technology while simultaneously addressing the security and privacy concerns associated with hidden spying cameras. The event took place in the Siebel Atrium, designed to simulate an office space with standard furniture. All teams, including ours, began from the same point in Siebel 1214, and our mission was to locate a hidden wireless spying camera using wireless traffic data. On the first day, the camera continuously transmitted packets, while on the second day, it only sent data when motion was detected.

We were encouraged to make the most of the various sensing modalities available in the RPI SenseHAT to create practical solutions that could be applied in scenarios like hotel rooms or Airbnb accommodations. Our performance was evaluated based on the accuracy of our camera location estimates in meters, with awards given to the top teams in each category. Additionally, we had the opportunity to present our approaches and gain media coverage.

2 SYSTEM OVERVIEW

The IMU+RSSI merging method, Your walking estimate method, the RSSI processing scheme, and your final localization system that merges all of these modules, any pre-processing or postprocessing or calibration methods you have used. Please use figures to better show the building blocks or your processing flow)

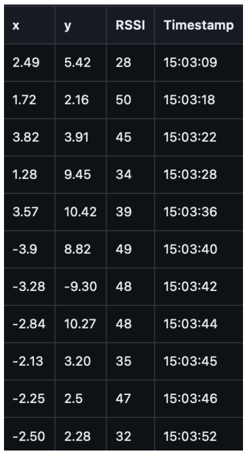
Day1: We decided to walk around the entire area (all the zones) first and then center in on the zone that we believed to have received the strongest RSSI values from. We utilized the senseHat on our Raspberry Pi to notify us of what range the RSSI signals of the transmitted packets were in. Our order went green, yellow, orange, and then red. This allowed us to get a much better prediction on where the spy camera was located. However, our original plan was to use step detection to predict the exact location of the camera because in the lab we were obtaining very accurate results of the number of steps we took to reach the camera (by exaggerating our z-axis movements). We were originally going to use all 3 trials to get accurate measurements on the predicted spy camera location. Unfortunately, we didn't know we were only allowed a single trial. As a result, we made some changes and relied heavily on IMU data, and double integrated our acceleration values in the x and y axis. We had a separate thread collecting our x and y acceleration values then the thread used to sniff packets so we could have improved position estimates. We then used these position values and retrieved the corresponding packet's RSSI value and plotted these points on a graph. Although our data was all over the place, we were able to pick up a couple strong RSSI value points that correspond to relatively accurate position (x,y) points, and were able to make an estimate on where we believed the spy camera to be located.

Day2: We had a similar walking strategy in Day 2 as in Day 1, where we walked around all the zones before centering in on where we believed the spy camera was located based on the RSSI values. We also used the coloring on our senseHat to indicate how strong the sniffed packets' RSSI values were (green, yellow, orange, red). However, we weren't able to get as many packets because we weren't in frame for as long. Unfortunately, we forgot to account for possible delay in the packets, which lasted for a few seconds. Therefore, when we passed by the frame of the spy camera, we didn't receive the packet for another 3 seconds, and because I was walking relatively quickly, this was a major source of error in predicting the location of the spy camera. We got a burst of packets about 3 seconds after we walked directly in front of the camera frame (close to camera), so if we had just walked a lot slower, we would have received a perfect estimate of the spy camera location.

3 DATA BENCHMARKING

Visualization of the data you collected during the two competition days over time and interpretation of the data

Day1:

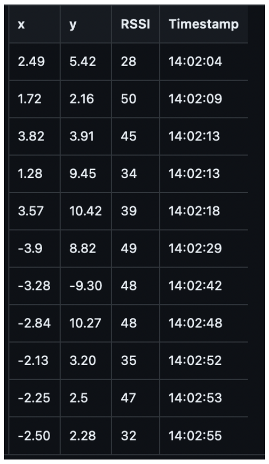


x	y	RSSI	Timestamp
2.49	5.42	28	15:03:09
1.72	2.16	50	15:03:18
3.82	3.91	45	15:03:22
1.28	9.45	34	15:03:28
3.57	10.42	39	15:03:36
-3.9	8.82	49	15:03:40
-3.28	-9.30	48	15:03:42
-2.84	10.27	48	15:03:44
-2.13	3.20	35	15:03:45
-2.25	2.5	47	15:03:46
-2.50	2.28	32	15:03:52

Interpretation:

Above we have our CSV file of our sniffed packets' RSSI values from the spy camera. From here we will have to process our data and plot the RSSI values with respect to it's x and y coordinates to further visualize our data and have a stronger prediction of where we think the spy camera might be located.By integrating the x and y acceleration values and using the packets with the strongest RSSI values (especially bursts of packets – low time deltas), we attempt to localize the exact location of the hidden RaspberryPi.

Day2:



x	y	RSSI	Timestamp
2.49	5.42	28	14:02:04
1.72	2.16	50	14:02:09
3.82	3.91	45	14:02:13
1.28	9.45	34	14:02:13
3.57	10.42	39	14:02:18
-3.9	8.82	49	14:02:29
-3.28	-9.30	48	14:02:42
-2.84	10.27	48	14:02:48
-2.13	3.20	35	14:02:52
-2.25	2.5	47	14:02:53
-2.50	2.28	32	14:02:55

Interpretation:

Above we have our CSV file of our sniffed packets' RSSI values from the motion detection spy camera, as well as our step detection data that we obtained that we decided to utilize as well in Day 2 in addition to IMU data. We then calculate the number of steps that we had taken to get a better understanding of how many meters we may have walked during the entire exploration of the area, which will assist us in obtaining the distance of where we walked to from the start of the exploration. Using this data and the integration of the x and y acceleration data, we attempt to predict the exact location of the hidden spy camera.

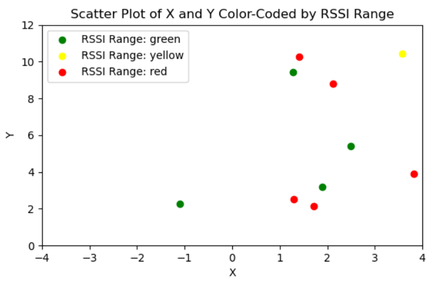
In both days, we received very few packets.

4 EVALUATION

Plots showing your processed data and estimated location for both days.

Day1:

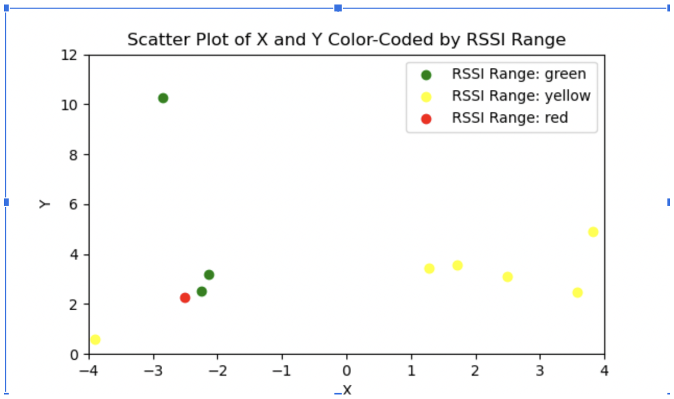
Estimated Location: **(1.62, 7.2)**

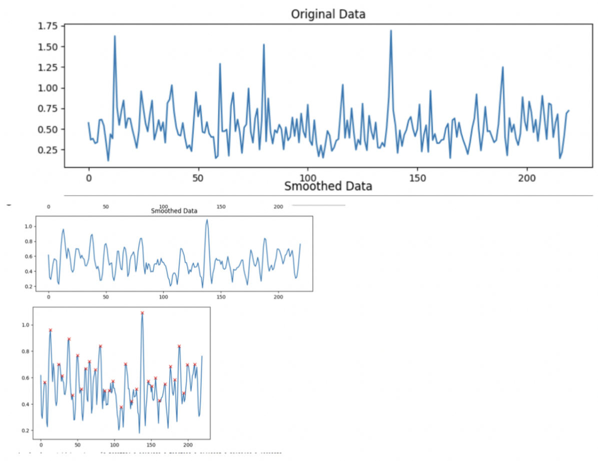


We observed that we were receiving packets with the strongest RSSI values in zone number 4; however, the harder part was closing in on the actual location in zone 4 of the spy camera. Using only IMU, which was didn't give us as accurate data as we desired (as you can see in our plots and data), we observed that the spy camera should be located at approximately the coordinates (1.62, 7.2) in zone 4.

Day2:

Estimated Location: **(-2.25,2.5)**





We observed that the area where we received the strongest RSSI values was where we predicted the spy camera to be located. Because using the IMU isn't always perfectly accurate, we had to use some guestimations on where the camera should be located. However, because of combination of the delay in the transmitted packets from the spy camera and our relatively fast walking speed, our predicted camera location was much south of where it actually was as we received the packets when we are already far past it. We also attached the images of our step detection graph, which we didn't use in Day 1, using the z-axis acceleration to determine whether or not a step was detected. As you can see in the step detection plots, they are a variety of peaks in the certain time limit, which signify when a step was taken .

In both days, we received very few packets.

5 REFLECTION FOR DAY 1

On the first day of our lab report project, we encountered some initial challenges. We found that the data we collected from various sources had differing sampling rates, resulting in data loss during the merging process. To address this, we explored the idea of utilizing two threads for merging in the preprocessing stage. We also tackled common filtering and faced some difficulties with lagging and buffering, which led to missed moments, particularly when moving slowly. Additionally, we realized that using VNC viewer introduced significant buffering. We questioned whether we were allowed to walk and communicate with our partner during the experiment, as it could impact data accuracy. Furthermore, we noticed that the IMU sensor in the Raspberry Pi exhibited some limitations. To address these issues, we began working on a simple code to combine Visual-Inertial Odometry (VIO) data from phones with Received Signal Strength Indicator (RSSI) data from the Raspberry Pi, along with IMU data from the phone. We contemplated using the human body as a shield to estimate direction. These were some of the high-level problems and questions that framed our discussions on the first day, such as mapping our movement, assessing RSSI values at various locations, making inferences about the camera, and distinguishing between the inferences made in different project stages.

6 REFLECTION FOR DAY 2

As we progressed into the second day of our project, we confronted several key challenges and implemented strategies to enhance our system's performance. Notably, we focused on improving the accuracy of our data. One approach involved mapping each received RSSI value to its closest reference, thereby increasing accuracy. We also introduced a process of averaging close data points together to minimize noise. We found that we could take a step approximately every 0.9 seconds, which contributed to smoother data acquisition. We diligently prepared the Raspberry Pi for the experiment, despite recognizing that IMUs, in general, tend to produce noisy data. Additionally, we explored the concept of "following the light" to better understand our surroundings. One significant advancement involved creating a threshold for RSSI values, allowing us to identify areas with strong signals. We saved the best RSSI values and noted the current direction we were facing, which became pivotal for our data analysis. Although these measures were not immediately helpful on the first day, they proved crucial for our work on the second day. Throughout our project, we were continuously addressing the high-level issues of mapping our trajectory, understanding the RSSI data at specific locations, making inferences about the camera, and discerning the differences between inference in various project phases.

7 DISCUSSION FOR IMPROVEMENTS

In order to only receive packets from only the motion detection camera, we will use the MAC addresses on the spy cameras and only investigate the RSSI signals that correspond with the motion detection cameras MAC address. By doing this, we are able to filter out transmissions from the other cameras, which will result in a much lower RSSI sampling rate. One way we could utilize the senseHat LED to guide the user to move around the space in order to maximize data collection is by having our LED light on our Raspberry Pi blink faster when we are facing the motion camera, and slower when we are facing in the opposite direction. For instance, when our back is turned to the camera it could block some of the radio frequency signals. Therefore, our RPI wouldn't be able to receive as many RSSI signals, or they will be weaker, and the LED blinking rate would be reduced, notifying us that we are facing in the opposite direction of the spy camera. To implement this in our Raspberry Pi, we decided to use different coloring on our SenseHat to depict how strong the RSSI signals are from the packets that we are receiving. We ordered the colors from red, orange, yellow, and green – green being the strongest and red being the weakest. Using the motion detection camera we constantly created motion in front of the lens, and had it transmit as many packets as possible and walked around with our raspberry pi with the SenseHat on in order to test how close we were to the spy camera.

8 POSTLAB3 EC DISCUSSION AND ALGORITHM

Our approach: We run through the program and walk around the room to simulate receiving RSSI values from different locations. Our RPI will receive rewards at points in the room (greater reward when the RSSI is stronger; smaller reward when RSSI is weaker) After the learning is finished → we use the sense hat to indicate the direction in where we should move to find the spy camera located in the room using the reward system (follow direction where there is greater reward)

The captured-packet-callback function is called whenever a packet is captured by the network interface. This function extracts MAC addresses, RSSI values, and accelerometer data (x and y) from the captured packet and sensor readings. The code initializes a packet sniffer (AsyncSniffer) to capture packets on the specified network interface (iface-n). The captured-packet-callback function is used as the callback to process each captured packet.

The code starts by initializing various parameters and variables, such as MAC addresses, network interface, the duration of the sniffing process, and Q-learning parameters like learning rate, discount factor, and exploration probability. This code comes from the PostLab3 prior before we are adding a reinforcement learning algorithm.

In our EC algorithm, we are having the state space is defined based on an 8x8 grid, where each cell represents a state which is the LED space. The action space consists of four possible actions: "moveup," "movedown," "moveleft," and "moveright." The Q-table is initialized as a dictionary with states as keys and sub-dictionaries as values, where each sub-dictionary maps actions to their associated Q-values. Initially, all Q-values are set to 0. The chooseaction function is used to select an action for the agent in a state based on an epsilon-greedy strategy. With probability explorationprob, a random action is chosen, and with probability 1 - explorationprob, the action with the highest Q-value in the current state is selected. Rewards are assigned based on the RSSI values. The code customizes reward values depending on the RSSI value for a specific MAC address. The higher the RSSI value, the higher the reward. After calculating the reward and selecting an action, the Q-value for the current state-action pair is updated using the Q-learning update rule. It incorporates the current Q-value, the learning rate, the reward, and the maximum Q-value of the next state.

Printing Results: The code prints the start time of the sniffing process. In summary, this code uses Q-learning to make decisions for an agent based on RSSI data and accelerometer readings. The agent learns to choose actions in different states by updating a Q-table, and it aims to maximize cumulative rewards by selecting actions that lead to higher RSSI values.

Reinforcement Learning Algorithm:

```
import numpy as np


learning_rate = 0.1
discount_factor = 0.9
exploration_prob = 0.2


num_states = 10
num_actions = 2


q_table = np.zeros((num_states, num_actions))

#example RSSI
rssi_values = [0.7, 0.6, 0.5, 0.4, 0.3]

packet_detected = [1, 0, 1, 1, 0]

def choose_action(state):
    if np.random.uniform(0, 1) < exploration_prob:

        return np.random.choice(num_actions)
    else:

        return np.argmax(q_table[state, :])


def q_learning():
    for episode in range(100):
        state = 0
        for t in range(len(rssi_values)):
            action = choose_action(state)
            next_state = state + 1
            reward = rssi_values[next_state]


            if packet_detected[next_state] == 1:
                reward += 10

            x = (1-learning_rate)
            y = q_table[state,action]
            xy = reward+discount_factor
            z = (xy * np.max(q_table[next_state, :]))
            z = learning_rate*z
            q_table[state, action] = x*y+z

            state = next_state


q_learning()


print("Learned Q-table:")
print(q_table)
```

9 WHAT WE LEARNED:

- Data collection from multiple sources, including RSSI data and IMU sensor data.
- Data integration by aligning and merging data based on timestamps.
- Data analysis, including the integration of acceleration and velocity data to estimate positions.
- Algorithm implementation, such as the Dead Reckoning (PDR) algorithm for position and orientation estimation.
- Calibration of sensors to reduce biases and errors in data.
- Experimental design for controlled data collection.
- Programming skills for creating data collection, analysis, and visualization scripts.
- Data visualization using plots and graphs.
- Problem-solving skills to address data accuracy issues and sensor biases.
- Effective communication of experimental findings.
- Enhancement of algorithms for better accuracy, including sensor fusion, data cleansing, and spatial interpolation.
- Critical thinking to propose strategies for locating hidden spy cameras based on sensor data and hardware limitations.
- Advanced Reinforcement Reward ML Algorithms

10 CONCLUSION:

In conclusion, Day 1 and Day 2 of our spy camera location tracking project brought about significant insights and challenges. On Day 1, our initial approach involved a systematic exploration of all the zones, utilizing RSSI values to gauge our proximity to the camera. By progressing through the colors on our SenseHAT, we narrowed down our search and, despite limited trials, leveraged IMU data for position estimates. Our strategy of double integrating acceleration data allowed us to estimate the camera's potential location with reasonable accuracy, even though the data displayed some variability. However, in Day 2, we encountered unforeseen delays in receiving packets, which introduced a source of error in our measurements. The lesson learned was the importance of accounting for potential packet delays, and had we walked more slowly during this phase, we might have obtained a more precise estimate of the spy camera's location. In both days, we adapted and improvised as we encountered challenges, underlining the iterative nature of experimental work and the need for meticulous planning in future endeavors.