DOCUMENTATION FOR ARK PERCEPTION TEAM TASK

TASK-3 Localization in known environment

I.INTRODUCTION

When a UAV (Unmanned Aerial Vehicle) needs to navigate from one place to another, it relies on two sources of information: the environment it needs to navigate and its initial position relative to the environment. In real-life situations, satellite imagery is used to visualize the environment, and Global Navigation Satellite Systems (GNSS), such as GPS, have been used for UAV localization. However, GNSS systems have been prone to inaccuracies caused by the use of radio signals to determine the distance between the UAV and satellites, as well as numerical integration approximations. This has led to the need for an alternative localization method. Localization refers to the process of estimating the position of an object relative to a given reference based on sensor inputs. In the case of drone localization, visual localization involves using camera images to estimate the drone's position in a given environment. The objective is to develop a strategy to localize the drone with respect to the maze using the provided files: player.py, utils.py, and MapGeneration.py.

II. PROBLEM STATEMENT

The problem is to localize the position of a UAV (represented by the Player object) within a maze using the maze image, local snapshots from the drone's camera, and control actions. The Player object provides methods to interact with the environment, including retrieving the maze map, capturing a snapshot of the surroundings, and controlling the drone's movement. The task is to complete the **strategy()** function in player.py to localize the drone's position. The function should ensure that the program is appreciably certain about the drone's position by minimizing the error. At the end of the execution of the **strategy()** function, the program should print the estimated position of the drone.

III. INITIAL ATTEMPTS

The initial approach was to use template matching with the getMap() and getSnapShot() images and hence find the location of drone. But sometimes there were more than one region in the entire Map which matched with the template. It this case it was hard to conclude where the drone was present. Thus, I used the motion of drone in horizontal and vertical directions to finalize its actual position

IV. FINAL APPROACH

In the final approach I first used template matching for finding the points where the drone might be present (the regions which matched the template). Then for each such region, I moved the drone up, down, right and left and compared the snapshot images with the corresponding images in the surrounding of those points. If everything matches then that is the actual position, if not then move to the next point.

The entire explanation of the player.py file is as follows:

1. The code starts by importing the necessary libraries: Player and WINDOW_WIDTH from a module named "utils", cv2 (OpenCV), and numpy.

2. An instance of the Player class is created and assigned to the variable player. This player object represents the player in the maze and has methods and attributes related to its movement and the maze.

3. The strategy() function is defined. This function is responsible for localizing the position of the player with respect to the maze. Inside the strategy() function, the getMap() method of the player object is called to get the current maze map. The getSnapShot() method is also called to capture a snapshot of the current position of the player.

4. A template is created using the captured snapshot. The size of the template is extracted using the shape attribute.

5. The matchTemplate() function from OpenCV is used to perform template matching between the map and the template. The result of the matching is stored in the res variable. A threshold value of 0.8 is set to determine the similarity between the template and the map. The where() function from numpy is used to find the locations where the similarity is above the threshold.

6. A loop is used to iterate over each possible location (pt) obtained from the template matching. Within the loop, four checks are performed to validate if the current location is the correct position of the player:

    a) The UAV (Unmanned Aerial Vehicle) represented by the player object is moved horizontally to the right, and a new snapshot is captured. A sub-image of the map is extracted based on the current location, and the difference between the new snapshot and the extracted sub-image is calculated using subtract() from OpenCV. If the number of non-zero pixels in the difference is greater than 0, it means the snapshot on moving right is not

equal to the corresponding section of the map, indicating that the current location is incorrect. The original position is recovered by moving the UAV horizontally in the opposite direction.

b) Similar to the previous step, the UAV is moved horizontally to the left, and the same check is performed.
c) The UAV is moved vertically up, and the same check is performed.
d) The UAV is moved vertically down, and the same check is performed. If all four checks pass, the current location (pt) is considered as the correct position of the player.

The vertical and horizontal distances of the centre of the current location from the top-left corner of the map are printed.

7. The __main__ block initializes the maze map and calls the strategy() function.

V. RESULTS AND OBSERVATION

Upon executing the code, the strategy() function attempts to localize the drone within the maze using visual localization techniques. It performs template matching, horizontal and vertical movement, and snapshot comparisons to validate potential positions. If successful, it prints the estimated position of the drone.

Observations:

- The localization strategy relies on accurate template matching and snapshot comparisons to identify the correct position.
- The accuracy of the estimated position depends on the quality of the snapshots and the matching threshold used.
- The program might encounter challenges if the environment contains similar patterns or if there are multiple sections in the maze that match the template.

VI. FUTURE WORK

Here are some potential improvements and future enhancements for the UAV localization using visual localization:

- Optimize the template matching algorithm to improve the accuracy and efficiency of position estimation.
- Implement image processing techniques, such as feature extraction and matching, to enhance the robustness of the localization algorithm.

VII. CONCLUSION

The localization of UAVs using visual localization techniques is a challenging problem. In this project, a strategy was implemented to estimate the position of a drone within a maze using visual inputs and control actions. The strategy involved template matching, snapshot comparisons, and movement control to validate potential positions and determine the drone's location. The success of the localization depends on the accuracy of template matching and the correctness of the comparisons. Further improvements and enhancements can be explored to enhance the accuracy and robustness of the localization algorithm.