Lab 3 Report - Georeferencing Using UAV Payload Data

Team E: Aditya Jain, Apurv Mishra, Praharsha Abbireddy

1 Introduction

This lab focuses on building a system for georeferencing of targets on the ground using a drone with an on-board camera. The first part is about designing a trajectory for efficient scanning of the search area and collecting image data. The second part builds the computer vision module for locating targets of interest in the captured data and referencing them with respect to an inertial reference system using drone's pose information. The trajectory design for data collection is first tested in a ROS simulation, followed by real world experiments on a Parrot AR.Drone 2.0 and Vicon motion capture system.

2 Data Collection

For capturing data in the given area, a lawn mover trajectory has been designed with an overlap for redundancy and comprehensive coverage of the circular targets. The area covered by the field of view of the camera is calculated using the given angle of 64° and image aspect ratio of 16:9.

Starting from the bottom-left corner of the search area (viewing as a 2D coordinate system), the coordinates (x,y) are calculated by iterating the values of the length and width of the computed FoV, for every position of the drone. Figure 1 shows plot of the resulting trajectory, for visualization. In the figure, the red dotted line and the orange line represent the FoV of the camera and area to be covered by the drone, respectively.

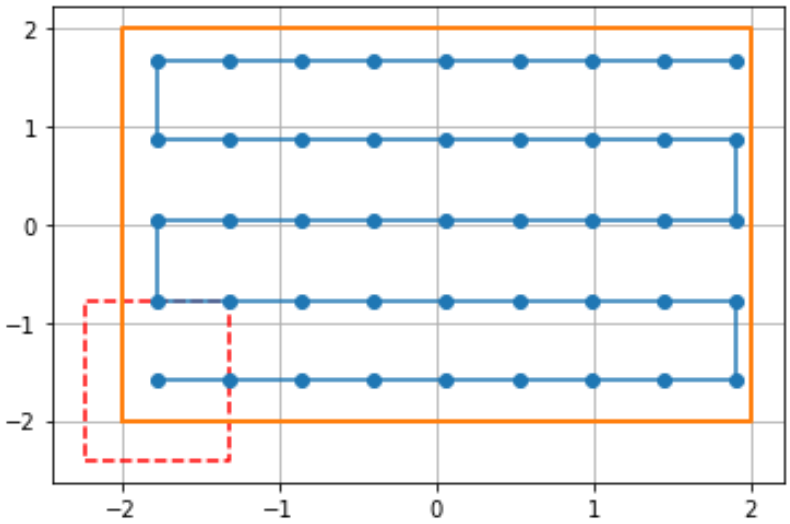


Figure 1: Plotted Trajectory

To measure the estimated flight time, the simulation was run for an overlap of 50% and height of 1.5 m. For the mentioned conditions, the drone flew for approximately 11 minutes to reach the end point.

3 Georeferencing of Targets

3.1 Image and Pose Data from Bag file

During the real world experiments for data collection, the data published on */ardrone/bottom/image raw* and */vicon/ARDroneCarre/ARDroneCarre* topics is recorded. The bag file is then used to fetch image and pose data for post-processing.

The bag file recorded 2,250 image and 23,336 pose messages. The bag file is then played in a time-synchronized fashion to save the images and corresponding pose (position and orientation) of the drone at the time instant of image capture.

3.2 Circle Detection using Hough Transform

To detect circles in the images, we used the Hough Circle Transform from OpenCV library. After a few iterative runs, the following parameters of cv2.HoughCircles() were found to be optimal for us: *minDist*=100, *param1*=50, *param2*=30, *minRadius*=25 and *maxRadius*=40.

For each image in the bag file, the green channel is extracted and a median blur filter is applied. This minimizes the detection of false positives. This processed image is then passed to the cv2.HoughCircles() function, which returns the circle centres and their corresponding radius. A total of 1780 out of 2250 images had circles detected. Figure 2 shows a few samples for the detection.

Figure 2: Examples of Circle Detection

Figure 3 shows the x and y positions of the drone where circles were detected in the images.

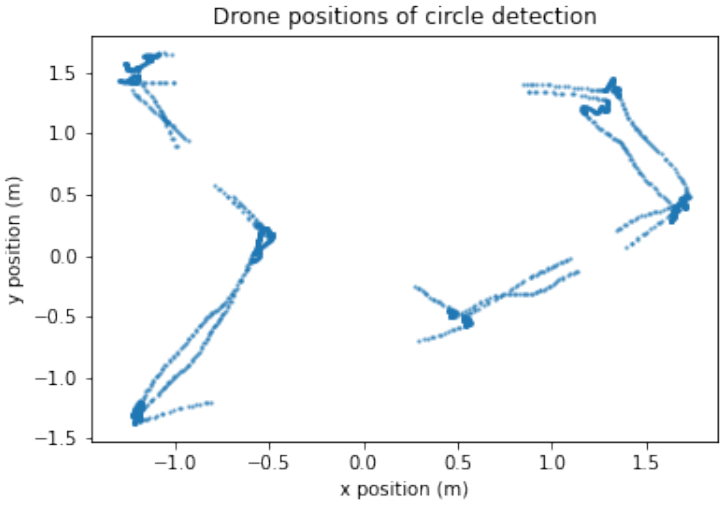


Figure 3: Drone locations where the circles were detected in images

3.3 Georeferencing Algorithm

After the circles were detected in the images, the pixel locations of the circle centres had to be transformed to Vicon inertial reference frame using drone's synchronized pose information. Below is the sequential procedure:

1. **Transformation from world to camera frame**:
   Vehicle's origin in camera frame, $vorigin_{camera} = vorigin_{body} * T_{CB}$, where
   $vorigin_{body} = [0, 0, 0, 1]^T$, $vorigin_{camera} = [x_c, y_c, z_c, 1]^T$ and $T_{CB} = \begin{bmatrix} 0.0 & -1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$

2. **Normalized Camera Coordinates**: The normalized camera coordinates of the vehicle's origin, $vorigin_{normal} = [x_c/z_c, y_c/z_c, 1]^T$

3. **Adding Distortion**: Radial and tangential lens distortion is added to $vorigin_{normal}$ to get $vorigin_{dist} = [x_d, y_d, 1]^T$, using distortion coefficients d = [0.159121, 0.549986, 0.003377, 0.001831, 0.000000]

4. **Pixel Coordinates Transform**: Finally, the vehicle's origin in image frame,
   $vorigin_{image} = K * vorigin_{dist}$, where camera matrix $K = \begin{bmatrix} 674.84 & 0.00 & 298.61 \\ 0.00 & 674.01 & 189.32 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$ and
   $vorigin_{image} = [x_o, y_o, 1]^T$

5. **Distance between circle centre and vehicle origin**: Now, the euclidean pixel distance $dist_{pixel}$ is calculated between the vehicle origin (x_o, y_o) and circle centre (x_c, y_c). Only circles that are less than 125 pixels from vehicle origin are considered for further processing because lens distortion increases as we move away from camera centre.

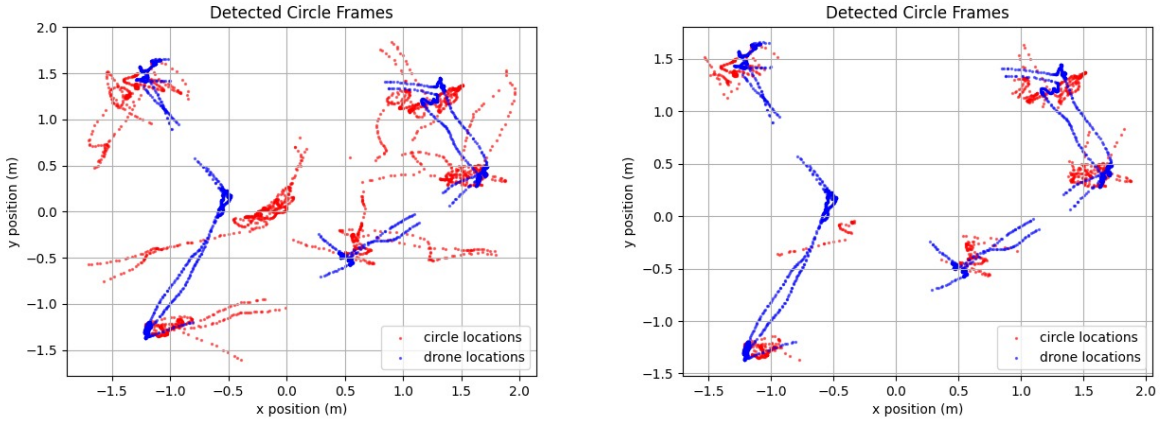
6. **Circle Location in Inertial Frame**: First, the euclidean pixel distance $dist_{pixel}$ is converted to metres, taking $pose_z$ is the height of the drone in metres for the captured image.
   $dist_m = dist_{pixel} * (2 * \tan(64) * pose_z)/\sqrt{640^2 + 360^2}$

   Then, the angle between the positive x-axis and the line joining vehicle origin and circle centre is defined, $angle_{circle} = arctan(y_c - y_o/x_c - x_o)$.

   Finally, the circle centre in inertial frame is calculated:
   $x_{c\_inertial} = pose_x + dist_m * cos(angle_{circle} + pose_{yaw})$
   $y_{c\_inertial} = pose_y + dist_m * sin(angle_{circle} + pose_{yaw})$,
   where $pose_x$, $pose_y$ and $pose_{yaw}$ are the vehicle's x position, y position and yaw respectively for the particular image capture.

3.4 Results

The two plots in figure 4 shows the overlay of drone locations where the image capture had circle detection and the corresponding circle centres transformed in the inertial reference frame. Figure 4a contains all the 1780 image frames where a circle was detected and figure 4b shows the post-filtering process (section 3.3, step 5), wherein we only consider circles whose centres are less than 125 pixels away from the vehicle origin in image frame.

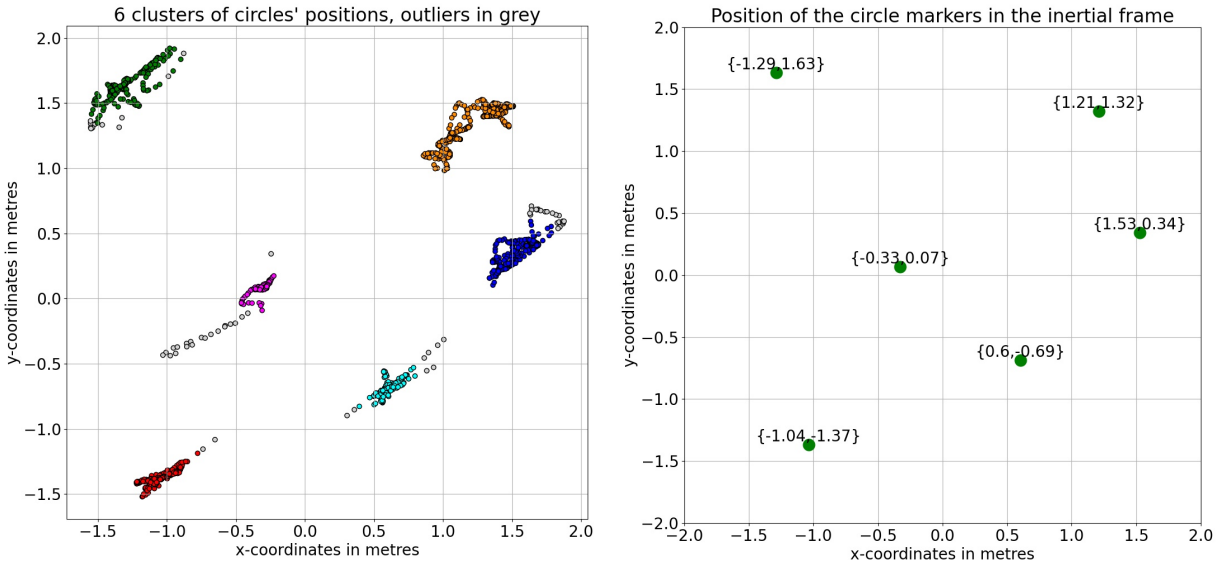


(a) All the 1780 images containing circles (b) Circle centres post-filtering

Figure 4: Overlay of drone locations and circle centres

The (red) filtered circle centres in figure 4b are then clustered using density-based clustering DBSCAN. Each cluster should have alteast 25 points and the results are depicted in figure 5a. After rejecting the outliers, the target locations are calculated by finding the centroid of every cluster, as shown in figure 5b. The final locations (in meters) are:

Target #1: (-1.29, 1.63, 0.00) m, **Target #2:** (-1.04, -1.37, 0.00) m,
Target #3: (-0.33, 0.07, 0.00) m, **Target #4:** (0.60, -0.69, 0.00) m,
Target #5: (1.21, 1.32, 0.00) m, **Target #6:** (1.53, 0.34, 0.00) m



(a) Clustering using DBSCAN (b) Final target locations

Figure 5: Clustering and final target locations