University of California San Diego

CSE276A: Introduction to Robotics

HW2: Robot control with Vision

October 31, 2023

Author	Student ID
Mazeyu Ji	A59023027
Zejia Wu	A59026752

Link to the Video:

https://drive.google.com/file/d/1cmmG9-dQCbBEPLOafFAWd1xzDOPxRHC5/view?usp=sharing

1 Diagram of Closed-loop Control

Figure 1 presents the diagram of our closed-loop control system, which is composed of the following parts:

- **Reference Pose**: The target position and orientation, which is the input.
- **Error Calculation**: Module to compute the difference between the target and estimated positions and orientations.
- **PID Controller**: The proportional term (P) generates control commands based on pose error. The integral term (I) is introduced to eliminate steady-state errors, with bounded magnitude to suppress drift. The differential term (D) has a minor control weight, which is used to suppress noise. The primary control relies on the proportional part.
- **Car**: The car moves according to the commands from the PID controller.
- **Environment**: The surroundings where the car operates, which might introduce noise.
- **Camera**: The camera detects landmarks in the environment and updates the pose of the robot.
- **Noise**: Random or unpredictable disturbances introduced by the environment.

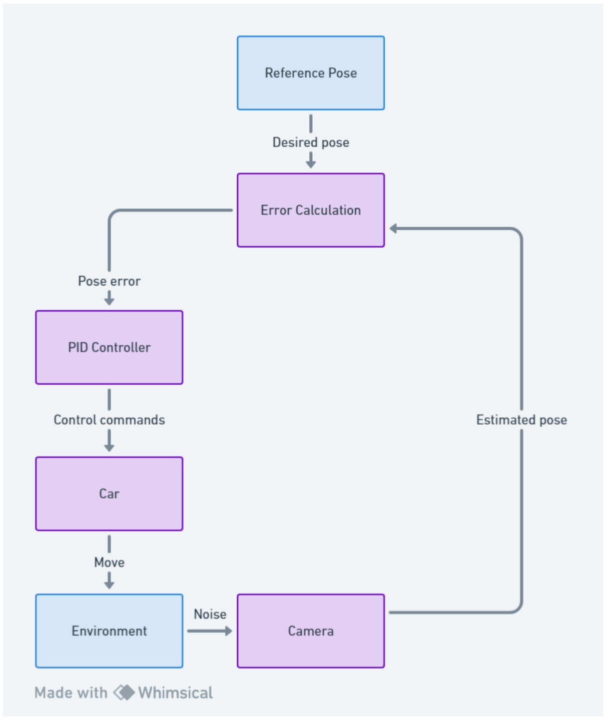

Figure 1: Diagram of the control system.

To summarize, our system calculates the error between the camera-estimated pose and the target pose, utilizes the PID controller to generate control commands to adjust the pose, achieving closed-loop control.

2 Description of Landmarks

We utilized AprilTags as landmarks. Despite having a wide-angle camera, we discovered that the detection of landmarks is not stable when at excessive distances or at skewed angles. Hence, we ensured that at every point along the trajectory, the robot could locate the target landmark with a rotation less than 180 degrees, and the distance to the landmark allowed for robust detection, with minimal number of landmarks. Specifically, we positioned the AprilTags in the world coordinate system at $(2.4, 0.7, -\pi)$, $(1, 2.4, -\frac{\pi}{2})$, and $(0, 2, 0)$, with the unit length being 0.5 meters. The relationship between the landmarks and the target trajectory is shown in the Figure 2.

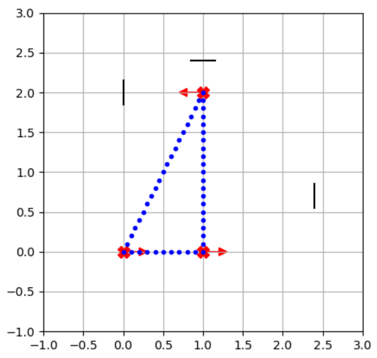

Figure 2: The positional relationship of landmarks, where the blue dots represent the ideal trajectory of the robot, the red dots and arrows represent the waypoints, and short lines represent the position of landmarks.

3 Estimating the Pose

Camera-Landmark pose estimation. Suppose that we have the coordinates of the four corners of the AprilTag in the image (p_1, p_2, p_3, p_4). Given the actual size of the landmark (width w and height h) and the intrinsic camera matrix K, we can estimate the pose of the landmark relative to the camera, $T_{\text{camera-tag}}$, using the **Perspective-n-Point**(PnP) method.

$$T_{\text{camera-tag}} = \text{PnP}(p_1, p_2, p_3, p_4, w, h, K). \tag{1}$$

Camera-Robot pose transformation. The fixed pose of the camera relative to the robot, $T_{\text{robot-camera}}$, is known. Then We can compute the pose of the landmark relative to the robot:

$$T_{\text{robot-tag}} = T_{\text{robot-camera}} \times T_{\text{camera-tag}}. \tag{2}$$

Robot-World pose estimation. Given the pose of the landmark in the world frame, $T_{\text{world-tag}}$, we can compute the pose of the robot in the world frame:

$$T_{\text{world-robot}} = T_{\text{world-tag}} \times T^{-1}_{\text{robot-tag}}. \tag{3}$$

Pose smoothing. The pose estimations can be noisy and exhibit sudden jumps. Therefore it is essential to smoothen these values for control purposes. One effective method is to use a moving average filter. By averaging pose estimates over a certain time window or a certain number of samples, abrupt changes can be mitigated. This results in a more stable and smooth estimate which is less prone to causing erratic behaviors in the differential controller. Mathematically, if we have n recent pose estimates $p_1, p_2, \dots, p_n$, the smoothened pose p_{smooth} can be calculated as:

$$p_{\text{smooth}} = \frac{1}{n}\sum_{i=1}^{n} p_i. \tag{4}$$

4 When The Robot Cannot See the Landmarks

During the robot's movement, situations may arise where the camera fails to detect any landmarks. For such scenarios, we employ the following methods:

- **Open-loop control when landmarks are absent.** The robot continues to move based on the previously calibrated relationship between input parameters and wheel speed. If open-loop control determines that it has reached the target point but still has not located any landmarks, it will then rotate in place to search.
- **Error correction upon landmark detection.** Once a landmark is detected again, the robot utilizes it to correct the cumulative error accrued during the open-loop phase.

This strategy ensures that the robot maintains movement when landmarks are temporarily out of sight and corrects its position when landmarks become visible again.

5 Analysis of the Trajectory

The 2D trajectory of the robot. Our program estimates and records the two-dimensional coordinates of the robot in real-time during operation, with the motion trajectory shown in Figure 3. Despite the presence of estimation errors, we can observe that the output trajectory essentially aligns with the actual motion trajectory of the robot in the video. As the robot approaches the target waypoints, the trajectory points become dense due to the small errors at this stage, which result in lower speeds output by the controller. In the process of returning to the starting point, the robot exhibits characteristics of moving along the x-axis first and then along the y-axis. This is because we did not

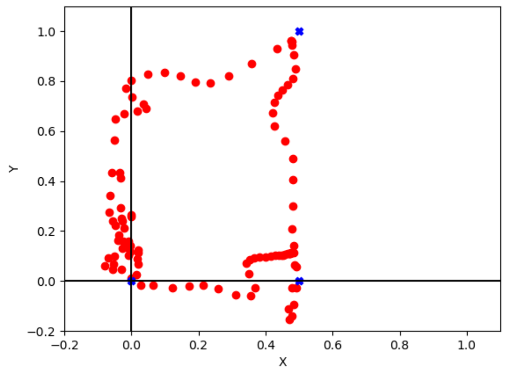

Figure 3: The 2D trajectory of the robot.

control the speeds in the x and y directions separately, and the orientation of the robot at that time was along the x-axis, given that forward and backward movements are often faster than sliding.

The total moving distance. Based on the trajectory presented above, we can calculate the total moving distance of the robot as 9.88 unit length (4.94 meters), while the length of the ideal trajectory is 5.22 unit length (2.61 meters). This discrepancy arises because the robot patrols the vicinity when approaching target waypoints. Furthermore, while the ideal trajectory back to the starting point is a straight line, the robot requires constant movement to adjust its pose. Therefore, the robot did not travel along the ideal trajectory perfectly.

Smoothness analysis. We calculated the speed of the robot between each two points on the trajectory. Throughout the trajectory, the average speed of the robot was 0.24 m/s, with a standard deviation of 0.31 m/s. The speed varied significantly with adjustments in control signals, hence the standard deviation is on the same order of magnitude as the mean. We also calculated the curvature of the trajectory corresponding to each trajectory point, with an average value of 8.65. Apart from rotation, the curvature of the trajectory is relatively smooth.

The location error on each waypoint:

Waypoint	x (m)	y (m)	θ (rad)
$(1, 0, 0)$	$+0.019$	-0.039	-0.002
$(1, 2, \pi)$	$+0.032$	-0.014	-0.012
$(0, 0, 0)$	-0.029	$+0.046$	-0.042

.

6 Overall Performance

In summary, our robot successfully navigated all target waypoints with small pose errors. Compared to the results of open-loop control, the introduction of visual closed-loop control significantly reduced errors, allowing the robot to adjust its pose at any time. However, there were instances when the robot failed to detect landmarks, indicating a need to enhance detection robustness. Furthermore, refining the control algorithm could make the travel more stable and smooth, reducing oscillation near target points.

7 Customized Landmark Detection

We also experimented with detecting our custom-designed landmarks. For the sake of robust detection, we had to use both color and shape information concurrently in the process of identifying landmarks to distinguish them from the environment. Thus, we set up a marker composed of a red square with a blue circle inside it. We first determined the color ranges to be identified and detected contours in the image within these ranges. Then, we applied the Ramer–Douglas–Peucker algorithm to fit the contour points into polygons. This detection was implemented on the robot, as shown in Figure 4. However, although the detection can be successful, it proved to be highly non-robust, with detection becoming challenging at greater distances. Therefore, we were unable to utilize this method for the robot localization.

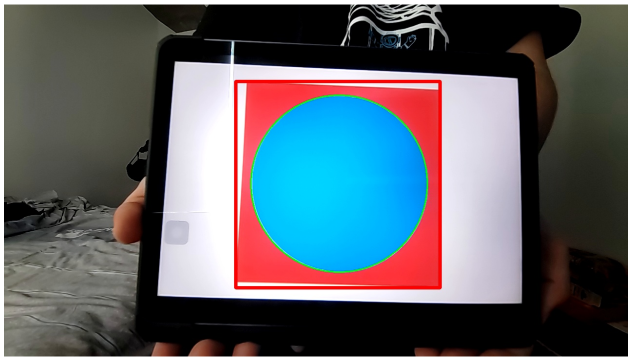

Figure 4: Detection of the customized landmarks.