Software Robotics Spring Co-op 2025 Assignment
Visual Servoing

October 18, 2024

1 Introduction

The Nokia AIMS team is counting on your expertise to help us shine at an upcoming technology showcase, where our work will be presented to a large audience. Your applied knowledge in computer vision, robotics, and mathematics will be critical to our success.

The challenge involves demonstrating a drone in flight, performing a vertical lawn-mower-like pattern in front of a wall displaying an image of a tower (think of a square signal).

The solution to the problem statement outlined in this document will determine how well the drone flies, thereby contributing to how well we perform in front of hundreds of spectators at the technology showcase. Your ability to solve this challenge will directly impact the audience's experience, and the more points you score, the better our drone will navigate.

Help us make this showcase a resounding success!

2 Problem Description

The drone must navigate along a tower structure while maintaining a fixed distance and following a specified pattern. To achieve this, the drone relies solely on the visual input from its onboard camera. As a research engineer in the team, your task is to process the camera image data and implement visual servoing techniques to effectively control the drone's motion.

Your evaluation will be based on the accuracy and effectiveness of your implementation, which should enable the drone to follow the tower structure and land safely. However, there is an important constraint that you must keep in mind when developing your solution:

You do not have knowledge of the drone's state when designing the controller, in other words, the drone does not have knowledge of its own position, orientation, or velocity estimates.

Hence, you must use the camera feed and then calculate the necessary control inputs inspired from pixel coordinates.

3 System Description

3.1 State & Control Inputs

In this problem, we consider a robotic system characterized as a 4 degree-of-freedom (DOF) platform.

The states and control inputs of the system are defined in a three-dimensional space (see Figure 1). The body frame of the drone aligns with the world frame.

- The state is represented by, $\mathbf{x} = [x,\ y,\ z,\ \theta,\ \dot{x},\ \dot{y},\ \dot{z}]^\top \in \mathbb{R}^7$, where $x,\ y,\ z$ are the positions, θ is the roll angle, and $\dot{x},\ \dot{y},\ \dot{z}$ are the velocities, all expressed in the world frame.
- The control input is, $\mathbf{u} = [f,\ \omega,\ a_y]^\top \in \mathbb{R}^3$, where f is the thrust, ω is the roll-angle rate, and a_y is the forward acceleration.

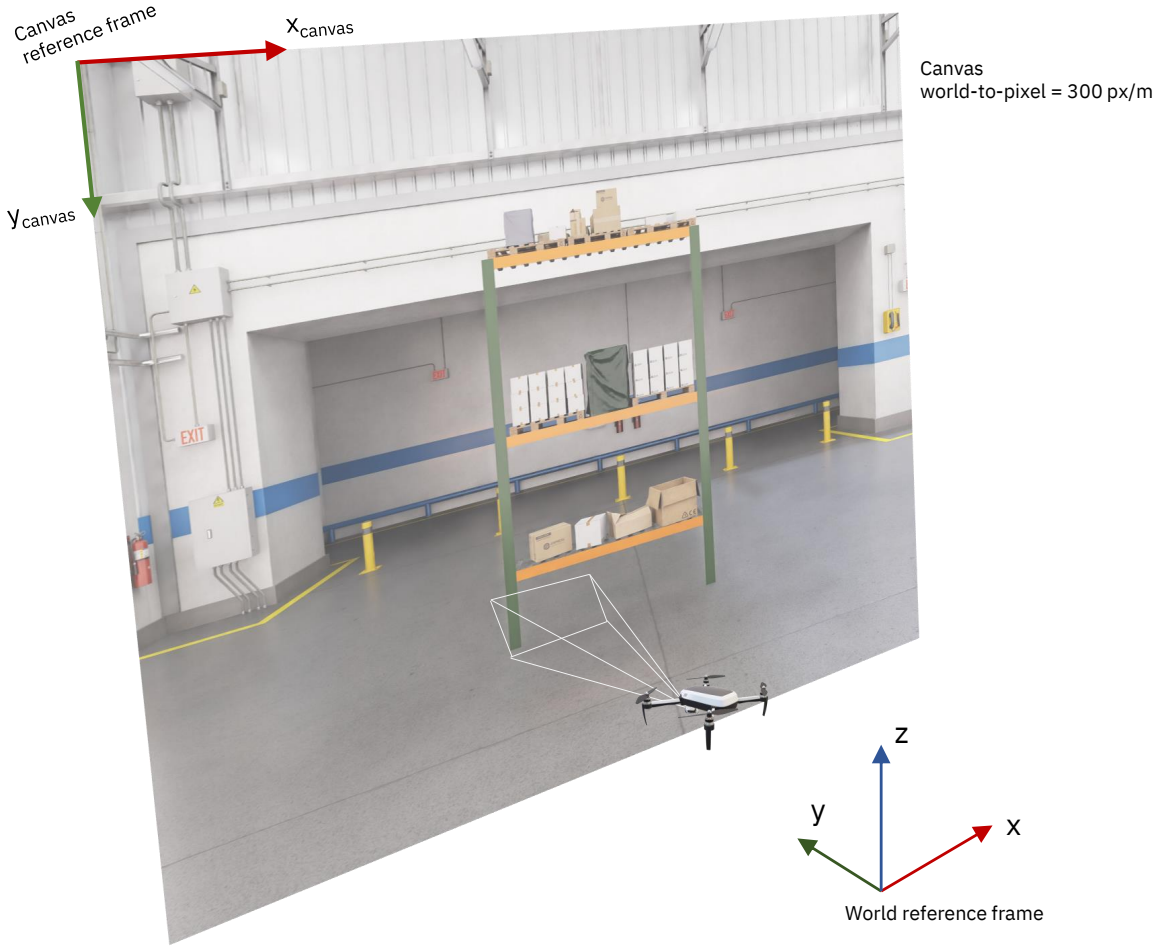


Figure 1: The world coordinate system and the canvas' coordinate system are shown. The drone can move freely in 3D position, but can only orient along its body y-axis, i.e., it can only roll.

3.2 Vehicle Dynamics

The equations of motion or the vehicle dynamics are described by the following equations:

$$\ddot{x} = f\ sin(\theta) - d\ \dot{x}$$
$$\ddot{y} = a_y - d\ \dot{y}$$
$$\ddot{z} = f\ cos(\theta) - g - d\ \dot{z}$$
$$\dot{\theta} = \omega$$

where,

- $g = 9.81\text{m/s}^2$, pointing in a direction opposite to the world frame's z-axis.
- $d = 4$ is the drag coefficient.

What you should do?

1. Implement the dynamics shown above in the function `vehicle_dynamics()`.
2. Implement the integrator of your choice in the function `step()`.

3.3 Camera View & Bounding Box

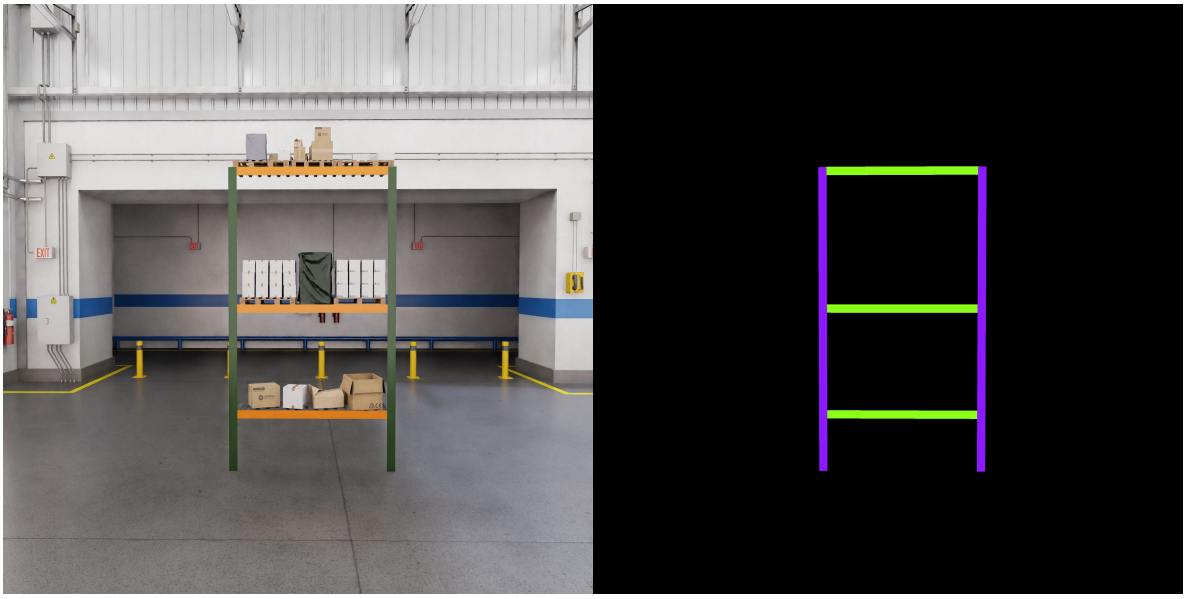
For the drone's environment, we have provided you with an image canvas and its corresponding masked image (see Fig. 2). The canvas is an RGB 3K image of the environment, with a warehouse background and a single tower with some boxes and pallets on the shelves. The mask represents the tower's vertical and horizontal pixels - the vertical and horizontal bars are differently colored. The image canvas is of shape 3072px × 3072px, and the drone will always look at a part of the image canvas at any point in time, providing it visual feedback.

What you should do?

1. Implement the function `get_camera_view()` to produce a 300px × 300px image based on the ground-truth state of the drone.
2. Implement the function `get_bounding_box()` to produce an oriented bounding box of each detected bar, $[p_x,\ p_y,\ w_{bb},\ h_{bb},\ \theta_{bb},\ \texttt{bar\_label}]$, where (p_x, p_y) is the bounding box origin in pixels, (w_{bb}, h_{bb}) is the bounding box width and height in pixels, θ_{bb} is bounding box orientation in image frame, and `bar_label` will be `vertical` or `horizontal`.

You can make the following assumptions,

- The canvas is at a distance of 2.75m from the world origin.
- The canvas pixel width is $\frac{1}{300}$m.
- You can assume that the camera's optical center and the drone's body center coincides, and the focal length is 3.2px.
- You can use any method you like to segment the bars and get the necessary bounding boxes.



(a) Canvas (b) Mask

Figure 2: A single tower in a warehouse environment comprises the complete canvas (left) and corresponding mask (right).

3.4 Visual Servoing

This is where we tie all the items together. The drone should fly a certain pattern as shown in Fig. 3. The expected maneuver for the drone is as follows:

- The drone should maintain a depth of 2m from the canvas.
- Drone first ascends, following the left vertical bar such that the vertical is 0.1m to the left of the drone.
- Once the drone reaches the top, about 0.1m above the horizontal bar, it starts flying right such that it maintains 0.1m distance above the horizontal bar.
- Finally, the drone descends following the right vertical bar such that the vertical is 0.1m to the right of the drone.

What you should do?

1. Implement the function `get_visual_servoing_inputs()` to calculate the control input. You cannot use any knowledge of the drone's state in this function.

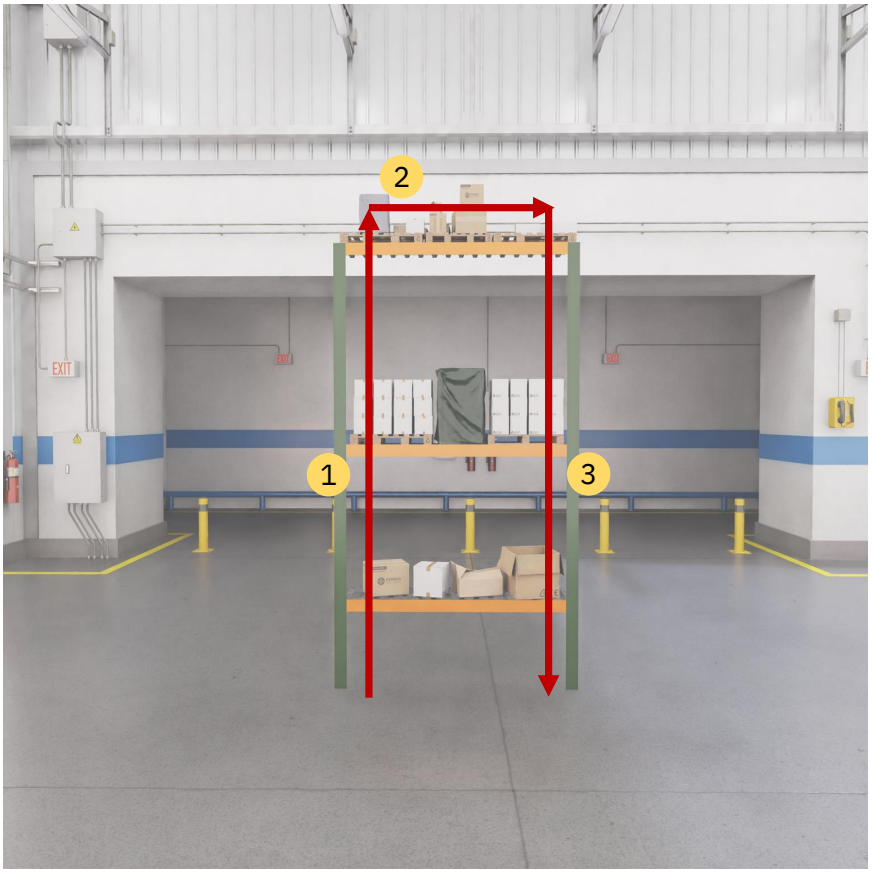


Figure 3: The expected path for the drone to follow.