1. **What is the sampling time of your system(hint: how fast can you get sensor data)? Why does sampling time matter, even with a subscriber based controller?**
   LIDAR:
   scan rate = 300±10 rpm scan rate ≅ 5 Hz **sampling time = 200 ms**
   Camera:
   Max frame rate = 90 fms **sampling time = 11.11 ms**
   However, the queue size (set to 10 in create_subscription) affects how fresh the processed data is, which can introduce latency in real-time applications. Even with a subscriber-based controller, sampling time affects system responsiveness. If the sampling rate is too low, the robot may react too slowly to changes in object position, causing lag or oscillations in control. If it's too high, noise and rapid fluctuations may lead to instability.

2. **What variant of PID control did you use? Why?**
   We used P-control only. This is because P-control adjusts the robot's speed proportional to the error (how far the object is from the desired distance). Since object tracking doesn't require precise position control, a simple proportional response is sufficient

3. **If you use a proportional controller (P) why does the robot achieve its steady state (distance from the object) when disturbances like friction from the floor, wheel shaft, or even Sean's finger on the tire are present?**
   A proportional (P) controller achieves a steady-state distance from the object despite disturbances like friction or interference because it continuously adjusts the robot's speed (or motor power) in proportion to the error between the desired distance and the actual distance. The P controller doesn't prevent the disturbances from affecting the robot's motion. Instead, it reacts to the effect of the disturbances on the robot's distance from the target. It continuously adjusts the motor output to minimize the error, eventually reaching a steady state (though possibly with an offset). This continuous adjustment is why it works even with unpredictable disturbances.

4. **What does it mean for a system to be unstable? What behaviors will your robot display if it uses an unstable controller?**
   A system is considered unstable when small changes in its input or disturbances cause large, uncontrollable changes in its output, leading to unpredictable or diverging behavior over time. In an unstable system, the error between the desired and actual output may continue to grow, and the system fails to reach or maintain a steady state. If the robot uses an unstable controller, then the robot will display behaviors like oscillations, divergence and erratic movements.

5. **Describe your algorithm to determine where the object is relative to the robot. Specifically, how do you use the camera and LIDAR data to produce a desired velocity vector? Include mathematical expressions used and supportive figures where appropriate.**

The algorithm uses vision-based object detection with LIDAR range measurements to compute a velocity command that drives the robot toward the object while centering it in the camera's view.

Algorithm w/ equations:

Object Detection with the Camera

- **Color Selection:** The user clicks on the video image to select a target HSV color. A small neighborhood around the selected pixel is blurred and its median HSV value is computed. This HSV value is then used to set a color range (with margins) for detecting the object.
- **Object Coordinates (in Pixels):** In each incoming camera frame, the node converts the BGR image into the HSV color space and applies a mask based on the set color range. Contours are used and the largest contour is assumed to be the object. The code computes the bounding rectangle and its center:

$$(x\_center, y\_center) = (x + w/2, y + h/2)$$

where x, y, w, h are the coordinates and dimensions of the bounding box.

Converting Image Coordinates to an Angular Position

- **Calculating Angular Position:** the horizontal pixel coordinate is converted to an angular measurement. If the center of the image is at image_center = image_width/2 and the horizontal FOV is FOV_x, then the angular offset in radians is computed as:

$$\theta = (x\_center - image\_center) * FOV\_x / image\_width$$

where x_center is the the center of the object that is being detected

Extracting Range Information with LIDAR

- **Mapping Angle to LIDAR Index:** the LIDAR index corresponding to the object's computed angle θ (relative to the LIDAR frame) is calculated using the following equation:

$$index = (-\theta - \theta\_min) / \Delta\theta$$

where θ_min is the minimum scan angle and Δθ is the angular resolution of the LIDAR.

- **Calculating the Average Range (distance):** A small window around the computed index is sampled (from index-2 to index+2). After filtering out any invalid (NaN) values, the average range d is computed:

$$d = (1/N) * \Sigma(i \in window)\ r\_i$$

where r_i are the range values in the valid window and N is the number of valid readings.

Computing a Desired Velocity Vector

- **Angular (Rotational) Control:** The robot rotates to center the object in the image. The error in pixel coordinates is given by:

  $$e_{angular} = image_{center} - x_{center}$$

  A proportional control law determines the angular velocity command:

  $$\omega = K_{p,angular} * e_{angular}$$

  where $K_{p,angular}$ is a gain.
- **Linear (Forward/Backward) Control:** To approach the object, a similar proportional control is used based on the distance error:

  $$e_{linear} = d - d_{desired}$$

  The forward (linear) velocity is then:

  $$v = K_{p,linear} * e_{linear}$$

  with $d_{desired}$ being the desired stopping distance and $K_{p,linear}$ the gain.
- **Combined Velocity Command:** The computed linear and angular velocities are packaged into a velocity command vector (ROS Twist message) that is sent to the robot's motor controllers.