Lab #4: Estimation of Time to Collision

In this experiment, you will use the Raspberry Pi Camera V2 on your TurtleBot3 to estimate the Time to Collision (TTC) with an obstacle placed in front of the robot.

To simplify the visual detection task, a paperboard with a simple pattern will be provided as an obstacle (Figure 1). The board must be mounted **vertically** so that it remains approximately parallel to the camera image plane during the experiment.

Figure 1: Raspberry Pi Camera V2

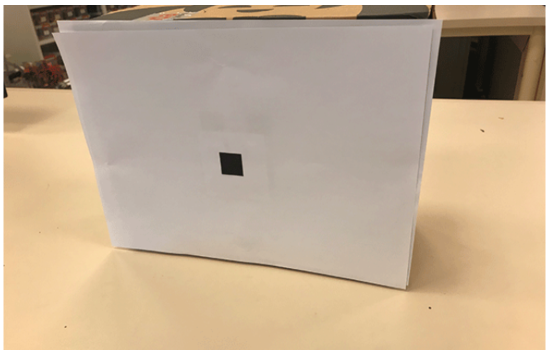
Figure 2: The paperboard representing the obstacle

By utilizing **/image_raw/compressed** ROS topic and MATLAB **Computer Vision Toolbox**, you will acquire and process the images. Particularly, you will acquire two sequential images in each cycle and extract the height of the obstacle (black rectangle) in **pixels**. Note that, the obstacle should be always parallel to the camera image plane. Make sure that the acquired images cover only the white background and the black rectangle for simplicity.

In order to measure the height of the black rectangle, extract corners in the obtained image by utilizing `detectHarrisFeatures` function, select the strongest 20 features and find maximum and minimum values. By using those values, you can plot a rectangle around the object and calculate the length of the right/left edge.

Once you find the height of the obstacle (ℓ) in sequential images, you can use the change in ℓ over time to estimate the TTC as follows:

$$\text{TTC} = \frac{\ell}{\dot{\ell}}. \tag{1}$$

In discrete form, the TTC can be approximated as

$$\text{TTC} = \frac{\ell_1}{\ell_1 - \ell_2}\,\Delta t = \frac{\ell_1\,\Delta t}{\ell_1 - \ell_2}. \tag{2}$$

where ℓ_1, and ℓ_2 are the height values in the sequential images and Δt is the time interval between the acquisition of these images. Note that if the robot is stationary (or almost stationary), we have $\ell_1 \approx \ell_2$ and the TTC tends to infinity. In that case, you must set TTC to a large number.

You are required to test the following cases:

- **Test 1 – Fixed Obstacle Moving Robot:** Place the obstacle at a certain distance and let the robot move towards the obstacle. By checking the time to collision, stop the robot after an appropriate time at a safe distance.
- **Test 2 – Fixed Obstacle Moving Robot Repeatedly:** Complete Test 1, move the robot back for a few seconds and let it go towards the obstacle again, repeatedly.
- **Test 3 – Fixed Robot Moving Obstacle:** Place the robot at a distance to the obstacle and keep it fixed. Then, move the obstacle towards the robot. By checking the time to collision, move the robot back when the obstacle exceeds the safe distance.
- **Test 4 – Moving Robot Moving Obstacle:** Place the obstacle at a distance and let the robot and the obstacle move towards each other. By checking the time to collision, move the robot back when the obstacle exceeds the safe distance.

Environment Set-up

Refer to the Lab #0 document to connect to the **TurtleBot3** and prepare the MATLAB environment in order to start the algorithm implementation.

Things To Do

- You are expected to implement the algorithm presented on pages 3-4.
- Before you start calculating TTC, make sure that you can take images, detect the obstacle by extracting the corner features and highlight the rectangle around it.
- **Record a video** demonstration for Test cases **1, 2, 3, and 4**, upload the videos to a cloud platform (e.g., Google Drive, YouTube), and include the shared links in your report.
- Submit your report **via SUCourse** until the report submission deadline.

Post-Lab Report Deadline: 10 December 2025, 23:55 via **SUCourse**

- Your report must include **introduction**, **procedure**, **results**, **conclusion**, **discussion** and **appendix**. Provide your **MATLAB** codes in the Appendix section appropriately.

Answer the following questions in the Discussion section of your Post-lab report:

1. Complete all the **Test** cases and comment on your results.
2. How did you determine the TTC threshold for stopping the robot?
3. In Test 1, how did changing the TurtleBot3's forward linear velocity affect the stopping distance and TTC curve?
4. How did you choose the sampling interval Δt?
5. What happens to your TTC calculation when the obstacle leaves the camera field of view (e.g., too close, too far, or moved sideways)?

Algorithm

You will write 3 MATLAB code blocks

- `main`
- `getImgFindCorners`
- `calculateTTCandMove`

getImgFindCorners.m

```
imgMsg = receive(imgSub, 5);
img = rosReadImage(imgMsg);
I = rgb2gray(img);
[rows, cols] = size(I);
Crop to Center (Region of Interest)
roiPercentage = 0.4;
r_min = floor(rows*(1-roiPerc)/2); r_max = floor(rows*(1+roiPerc)/2);
c_min = floor(cols*(1-roiPerc)/2); c_max = floor(cols*(1+roiPerc)/2);
I_roi = I(r_min:r_max, c_min:c_max);
corners = detectHarrisFeatures(I_roi);
corners = corners.selectStrongest(20);
corners = corners.Location;
Offset corners back to original image coordinates,
X = corners(:,1) + c_min; Y = corners(:,2) + r_min;
Now you have x and y coordinates of the detected corners (as vectors)
imshow(img); hold on;
Plot the rectangle in red;
Plot the right edge in green;
Calculate the length of the right edge;
```

calculateTTCandMove.m

```
if Robot is stationary then
    TTC = 100;
else
    Calculate TTC as in equation (2);
end
Move the robot according to Test cases;
Calculate the travelled distance of one wheel (you can use /odom topic);
Save the travelled distance to an array;
```

main.m

```
clear all; close all; clc;
Setup /cmd_vel publisher;
Setup /image_raw/compressed subscriber;
odomSub = ros2subscriber(node,"/odom","nav_msgs/Odometry", ...
"Reliability","besteffort","Durability","volatile","Depth",10);
deltaT = 0.1;
startMsg = receive(odomSub, 10);
startX = startMsg.pose.pose.position.x;
startY = startMsg.pose.pose.position.y;

while Target not locked do
    | getImgFindCorners;
    | pause(1/25);
end

Adjust motor speed parameters and start;

while Experiment Running do
    | start = tic;
    | getImgFindCorners;
    | Assign the calculated length to $L_1$;
    | pause(deltaT);
    | getImgFindCorners;
    | Assign the calculated length to $L_2$;
    | calculateTTCandMove;
    | Plot the travelled distance;
    | elapsed = toc(start);
    | pause((1/25) - elapsed);
end
Stop motors;
```