ME 425 Post Lab Report 5

Demir Demirel, Bati Berk Ozata

***Abstract*— This report include implementation of Two Wheeled Robot with Time To Collision implementation made with Lego Mindstorms™ EV3 and discussion on the test results gathered with MATLAB.**

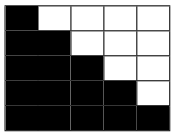
I. INTRODUCTION

In this lab we were required to construct a two wheeled robot and implement time-to-collision(TTC) algorithm to enable the robot stop before the collision to the object that we can determine with height of black square on the white background. The TTC is depending on the height of the black square on the obstacle we use and the amount of change of in size. This algorithm also can be used in autonomous cars to protect passengers to get damage. For that reason, camera used instead of sensors. Camera is enable us to implement computer vision algorithms based on Harris Corner detection. When the object is getting closer the robot should stop at a certain distance. Rest of the report we will be explaining the computer vision algorithm and the programming of the robot and how did the change in speed, distance and Δt affected our desired stop position.

II. PROCEDURE

In this lab, we have experienced implementation of differential-drive robot model and computer vision algorithms. To achieve this, we used MATLAB environement to give necessary commands to our mobile robot that we constructed from LEGO pieces. We managed, forward movement and stop action with the commands from the data gathered with camera. During the motion of the robot, we collected the rotation and camera data to plot to see our results.

First thing to do in the problem wants us to understand the image data. The camera data is gathered with the resolution of '320x240'. About the detection of corners; "large change in intensity in at least 2 directions" define as a corner. So when we shift the window in any direction, we detect corners. But the corner should not be as sharp as we expect, there might be some edges which could represented as the matrix given below. In this matrix computer could detect



many corners when we look with the 2x2 window matrix. After detecting of corners with *detectHarrisFeatures*, select the 20 strongest feature and determine the x,y coordinates. Than we apply the formulas for the time to collision calculation. Formulas are given as follows:

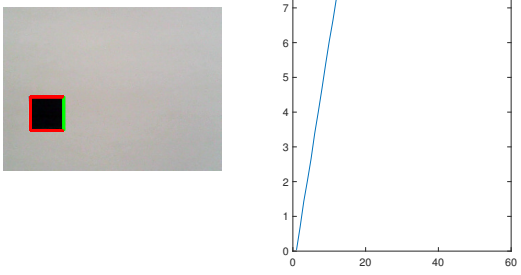
$$TTC = \frac{L_1}{\frac{L_1 - L_2}{\Delta t}} = \frac{L_1 \Delta t}{L_1 - L_2} \tag{1}$$

where L_1 and L_2 are the heights in sequential images.
After applying this formula in MATLAB we are assigned to do some objectives. One of them is named as 'Fixed Obstacle Moving Robot' which is move the robot towards the obstacle and stop at a safe distance. We get some results to compare with trials of different scenarios.

III. RESULT

In this lab we observed the distance of the two wheeled robots wheels with MATLAB plots. In the following three figures Fig.1, Fig.2 , Fig.3 ., TTC is limited as 8 and the motor speeds vary between 20,50,100. While the speed is increasing, between 20 and 50 the maximum distance is around 9 but the response time gets smaller. When we

Fig. 1. TTC limited by 8 and V =20



compare 50 and 100 the response time is stays same but the distance covered in that time increase dramatically. While 50 has distance 9, 100 has distance 23. There is also another comparison material in the figures; the image when the robot stops. First two seems like the length of the edges are equal. But when we look at the third figure, length of the edges becomes larger. So we could conclude that the longer starting distances has more error on the length of the square.

In the following two figures Fig.4 , Fig.5 the velocities are stays same but the TTC limit changes. When we compare the travelled distance, TTC limitted in 1 travelled around 4 but the TTC limit 5 travelled 9.

Fig. 2. TTC limited by 8 and V =50

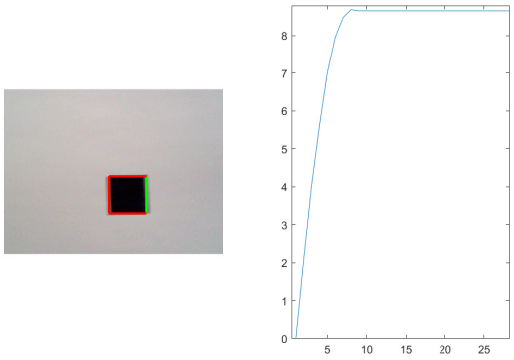


Fig. 3. TTC limited by 8 and V =100

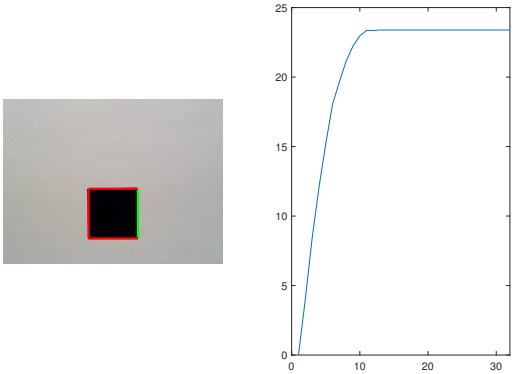


Fig. 4. TTC limited by 1 and V =30

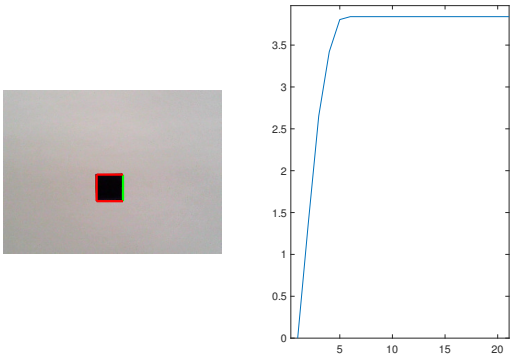
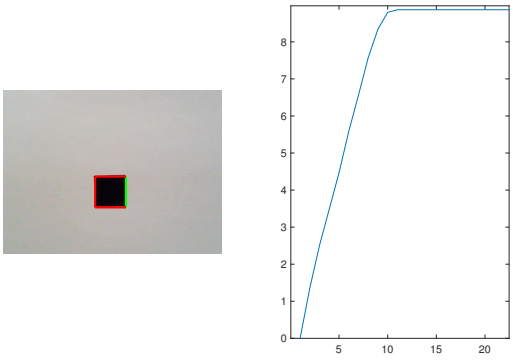


Fig. 5. TTC limited by 5 and V =30



A. CONCLUSION

The two wheeled differential drive robot experimental platform LEGO MINDSTORMS EV3 and Camera system is described in this lab report. The data gathered from actuators and camera are reviewed in terms of graphs that represent travelled distance of the robot before stop.

The theoretical expectations was not satisfied well. There are environmental constraints which prevent the accurate motion like friction. Beside environmental constraints, missing of a control algorithm also affect the real motion. Also measurements of the wheel base and wheel diameter has a small effect on the estimation observed as a result of the experiment.

IV. INDIVIDUALS

A. Bati Berk Ozata

This week we used the proximity sensor to calculate the size of a square printed on a paper that we use as an obstacle. We calculate the TTC which stands for time to collision. The robot calculates the time to collision using the size and the rate of size change comparing two sequential snapshots from the camera. Then according to TTC we coded the robot to move and when it gets close it stops in order not to collide.

B. Demir Demirel

In this lab we were tried to plot the travelled distance of the robot with the data that we gather from encoders of the motors. From the motor encoders we can only obtain the rotation data. This rotation data helps us to find out distance that traveled of each tire. This distance is obtain from equations of (2) and (3).

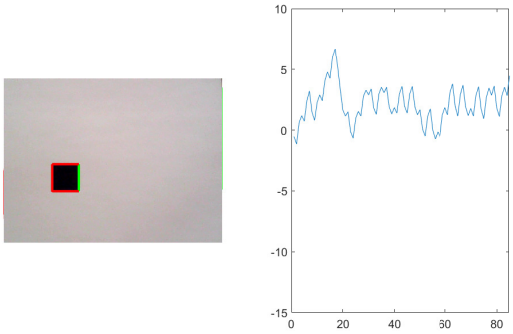
$$D_R = 2\pi R \frac{\Delta E_r}{360} \tag{2}$$

$$D_L = 2\pi R \frac{\Delta E_l}{360} \tag{3}$$

But the only one of the D_L or D_R is enough for this assignment since the robot is only capable of move towards. After this step the time to collision should be calculated with the equation (1). And this formulation gives us the values that are very high when there is no motion in the camera since L_1 == L_2 and $\frac{L_1}{L_1 - L_2}$ is corresponds to infinity. In such a case the value for time to collision should fixed to 0 to represent the robot is stationary. And our robot should move if the TTC value is between 0 and the TTC_limit that we decide.
About the first case the distances travelled stop at a certain points in each cases we can determine the stop point with changing in TTC limits. When the limit increased, the distance gets higher.
About the second case the robot should move between two

points. In other words between two limits. In the following figure, we apply 'Fixed Obstacle Moving Robot Repeatedly' method. In the graph we can see that the robot come and go 3-4 again and again. But there is some unexpected movements in that graph between time interval 50-60. The reason for that could be noise of the data or the slippage in the robot. The data could be noisy since the robot has some slippage. It makes the square located in many different locations. And makes some noises because of it.

Fig. 6. Task 2



About the time to stop we first need to discuss the reliability of the data. Normally time to collision is calculated with one simple formula without camera. TTC calculated with the data gathered from the IMU or accelerometer or encoders of the motors with the range finder sensors like sonar, LIDAR, etc. The formula could verbalized as; the distance between obstacle divided by difference in the velocity of the object and the robot.

$$TTC = \frac{L}{\Delta V} \tag{4}$$

In our case we have an image data which could leads some errors according to some vision based problem(Selecting strong correlated ones may be wrong corners etc.). For example we expect L1-L2 will be 0 but the real data is not that accurate.

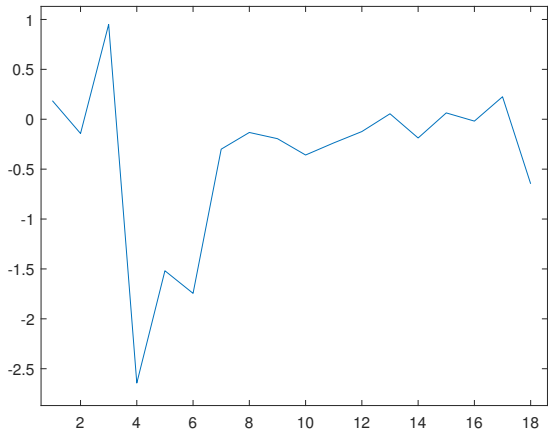
If we assume the data is clear enough to analyze, we need to stop at certain point with some formulation as follows:

$$TTC = \frac{v\Delta t}{v} \qquad v\Delta t = TTCv \tag{5}$$

If we assign 10 cm for the distance between obstacle and the velocity as 10cm/s, time to collision would be 1 second. When we think our high school knowledge, the TTC would be correct. So time to stop should determined with the formula above with given a certain distance between object and robot.

When we change the speed of the robot, lets say the speed is 2v. When we apply the formulation in equation 5, we get time to collision as TTC/2. So we can conclude that when we increase the speed the TTC should be decrease.

Fig. 7. L1-L2 for V=50, TTC=8



The another thing is the Δt. Δt determined by design parameters. If we want small difference between L_1 and L_2, we need to choose smaller Δt. If we want bigger difference between L_1 and L_2, we need to choose larger Δt.

The other possible problem is losing the tracked object. We try this objective with the object and some other objects in the image. For example we have a square as normal but in addition to the square, we have some other corners so that the algorithm detect another corners. And it makes confusion in the L values.

To sum up this lab shows us the vision is important sensor and used as instead of 2 sensor. With the exercises that we did also we have chance to see effects of speed and the sampling time to make correct estimations on TTC.

APPENDIX

main.m

```
clear all; close all; clc;

mylego=legoev3;
leftmotor  = motor(mylego,'D');
rightmotor = motor(mylego,'C');
start(leftmotor);
start(rightmotor);
resetRotation(rightmotor);
resetRotation(leftmotor);
ΔT=0.01;
FileName = 'case1_speed30_TTC7';
R = 2;

DEleft= [];
TTC=[];
Data.elapsed=[];Data.I={};
Data.X={}; Data.Y={};
Data.x={};Data.y={};
Data.L1=[];Data.L2=[];

cam = webcam(1);
cam.Resolution = '320x240';
while ¬readButton(mylego, 'down')
% Your code will be here.
getImgFindCorners;
pause(1/25);
end

while ¬readButton(mylego, 'up')
% Your code will be here.
start=tic;
subplot(1,2,1);
getImgFindCorners;
L1 =L;

pause(ΔT);
getImgFindCorners;
L2=L;


calculateTTCandMove;
subplot(1,2,2);
plot(DEleft);
title(['TTC: ' num2str(TTC) ]);


elapsed=toc(start);
pause((1/25)-elapsed);



Data.elapsed(end+1)= elapsed;
Data.X{end+1}  = X;   Data.x{end+1} = [x1 x2];
Data.Y{end+1}  = Y;   Data.y{end+1} = [y1 y2];
Data.L1(end+1) = L1;  Data.L2(end+1) = L2;
Data.DEleft=DEleft;   Data.TTC=TTC;
save(FileName,'Data');


end

stop(rightmotor);
stop(leftmotor);
```

calculateTTCandMove.m

```
if abs(L1-L2)< 0.1 %Robot is stationary then
    TTC(end+1) = 0;
```

```
else
    TTC(end+1) = L1*ΔT/(L2-L1);
end

if TTC(end) < 7 && TTC(end) > 0
       leftmotor.Speed  = -30; %stop(leftmotor);
       rightmotor.Speed = -30; %stop(rightmotor);
       pause(0.5);
else
       leftmotor.Speed  = 30;
       rightmotor.Speed = 30;
       pause(0.1);
%        start(leftmotor);
%        start(rightmotor);
 end

%Calculate the travelled distance of one motor;
DEleft(end+1)  = 2*pi*R*double(readRotation(leftmotor))/360;
```

getImgFindCorners.m

```
img = snapshot(cam);
I = rgb2gray(img);
corners = detectHarrisFeatures(I);
corners = corners.selectStrongest(20);
corners = corners.Location;
X = corners(:,1); Y = corners(:,2);
imshow(img); hold on;
x1=min(X);
x2=max(X);
y1=min(Y);
y2=max(Y);
plot([x1 x1],[y1,y2],'r-','linewidth',2);
plot([x2 x2],[y1,y2],'g-','linewidth',2);
plot([x1 x2],[y1,y1],'r-','linewidth',2);
plot([x1 x2],[y2,y2],'r-','linewidth',2);

L=y2-y1;
```