ME 425 LAB 4 Report
Computer Vision On Turtlebots

Arda Gencer
34124
Instructor: Mustafa Ünel

Fall 2025-2026
Due Date: 10/12/2025

1 Introduction

In the lab #5 of ME425, a more complex vision algorithm was implenented to be able to detect Aruco markers. This algorithm was then used to implement a proportional motion controller that makes the robot follow the Aruco marker until it reaches a desired distance to it. Several MATLAB CV toolbox functions such as extrinsics were used alongside the camera parameters to be able to determine the extrinsic parameters of the camera and get the necessary control inputs. More details are given in the procedure section.

2 Procedure

The tracking algorithm was developed in 3 steps:

1. First, ArucoDetect.m was developed. In this script, the robot is completely stationary and the camera constantly searches for any Aruco markers by using the readAruco-Marker function. It also sends the pictures it takes to the MATLAB for visualization. From the readArucoMarkers function, an ids vector is received and this is checked to see if it is empty or not. If it is empty, this means that no markers were detected. If it isn't empty, it means that an Aruco marker was detected. If a marker is detected, its borders are indicated with green lines on the image that the robot sends to the MATLAB.

2. For the second step, a SearchAndStop.m script was developed. This scriptp uses the marker detection logic from the previous script but also incorporates motion to it. If the robot doesn't see any markers, it turns around itself with a constant angular velocity. If any marker is detected at any time, the robot stops rotating. When the marker disappears from the robot's vision, it starts spinning with the same angular velocity again.

3. Finally, the Aruco marker tracker was developed in VisualServoing.m by expanding upon the previous developed scripts. In this script, first, the required parameters such as the location of the markers in the 3D world, the camera parameters, the desired stopping distance and the control gains are defined. In the main loop, if the marker isn't detected, the robbot spins around itself just like the algorithm in step 2. However, if a marker is found, the chasing algorithm and the proportional controller starts. Here, the extrinsics function was used with the corner coordinates in the image pixel frame and also their coordinates along with the camera parameters that were provided. From this function, the extrinsic parameters were extreacted (namely the rotation matrix and transformation vector). Then, the contents of the transformation matrix were used as control inputs to the motion controller. The controller was developed as follows:

Forward distance and lateral deviation of the marker:

$$x_R = t_z$$

$$y_R = -t_x$$

Errors:

$$e_\theta = \text{atan2}(y_R, x_R)$$

$$e_d = x_R - d_{\text{desired}}$$

The inputs to the robot:

$$v = k_{lin}\ e_d$$

$$\omega = k_{ang}\ e_\theta$$

During the process the errors were also collected and saved to different vectors. After the code is ran, a seperate script was used to plot these data to see how the errors change in each iteration. A video demonstration of this task can also be found in the results section.

3 Results

The link to the video demonstration of Task 3:
https://drive.google.com/file/d/12Ny95SaOb7IvO2PwbE7Zw920EG5IQ230/view?usp=sharing

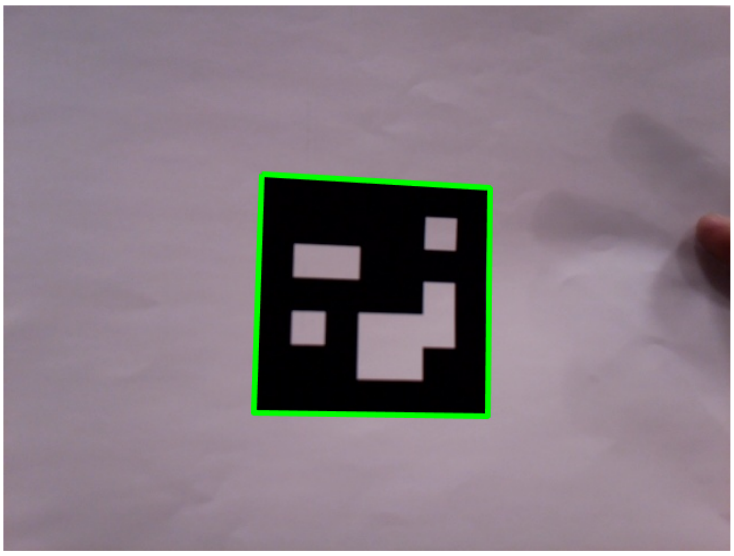

Figure 1: The image result from Task 1

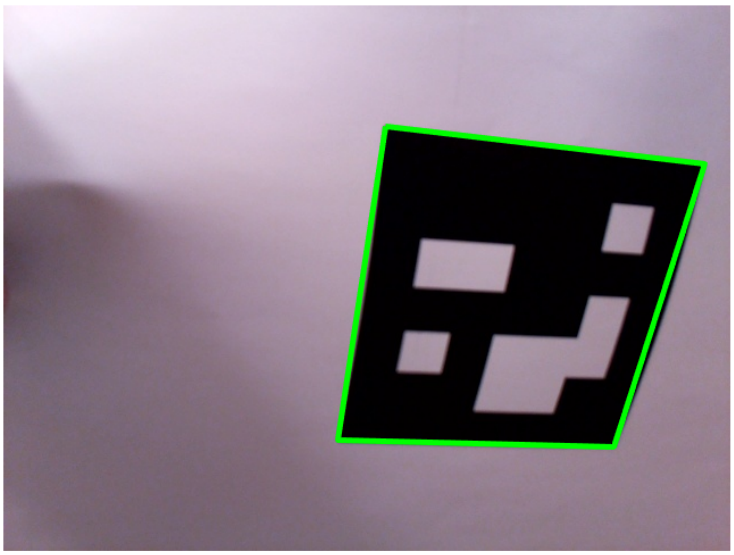

Figure 2: The image result from Task 2

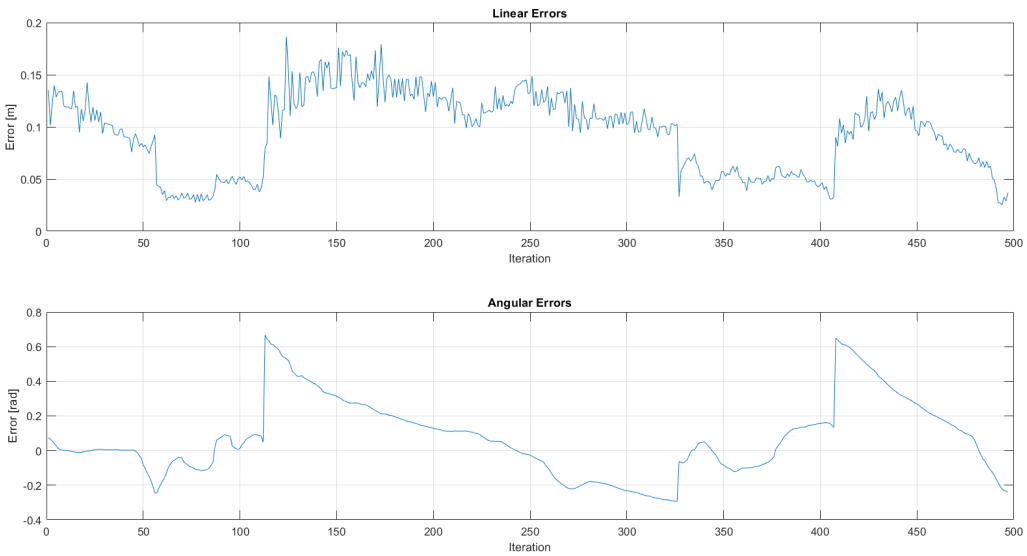

Figure 3: The linear and angular error data from Task 3

4 Conclusion

In this lab, a continuation to computer vision applications using Turtlebbot 3 and MATLAB was made. An example to how the vision data can be used in other applications such

as motion control was also developed.

5 Discussion

1. Explain the significance of the Camera Intrinsic Parameters in distance estimation. What happens if these parameters are incorrect?

   For the equation given in the lab document that represents the relation between the world coordinates of a point and the image coordinates to give out correct results, these parameters must be correct. If not, a wrong extrinsic parameter matrix will be received and the estimated distances will also be incorrect.

2. How did the estimated distance vary as you moved the marker physically closer or further? Was the measurement noisy?

   In the lab, only the error data was saved. However, looking at the linear error data, it can be seen that there are many spikes and quite a bit of noise. This is most likely caused by the sudden movement of the Aruco marker. As the robot suddenly sees the marker much closer/ farther away than where it was previously.

3. Describe the robot's behavior in Task 3 when the marker was moved laterally quickly. Did it overshoot or lose track?

   Due to the lag between the Tutlebot and the MATLAB, when the marker is moved too much in too little time laterally, the robot may lose track or not be able to adapt in enough time to the marker's new position. But, if the marker is moved gradually; usually, no issue is encountered.

4. How did lighting conditions affect the ArUco detection? Did you experience any false negatives or detection jitter?

   The lighting conditions that were present in the lab usually didn't affect the detection in any negative way. However, iff the lighting conditions are way too dim (or bright), this may affect the camera's ability to be able to detect the markers.

6 Appendix (All MATLAB Codes That Were Used)

Listing 1: ArucoDetect.m. Detects any Aruco markers and visualizes it on a image plot.

```
%% Environment Setup
clc; clear; close all;
clear node; % clear previous node handle if it exists

setenv('ROS_DOMAIN_ID','37'); % Set to your robots ROS_DOMAIN_ID (check
    the robot"s ID card)
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
node = ros2node('ArucoDetect');
%% Subscribers And Publishers
imgSub = ros2subscriber(node,"/image_raw/compressed","sensor_msgs/
    CompressedImage", ...
"Reliability","besteffort","Durability","volatile","Depth",10);
%% Visualization
% SETUP (Before Loop)
figure("Name","ArucoDetect");
hIm = imshow(zeros(480, 640, "uint8")); hold on;
hPlot2D = plot(0, 0, "g-", "LineWidth", 3);
receive(imgSub, 10);

%Update The Visualizatiion
while true
    %% Read Image
    imgMsg = receive(imgSub, 10);
    I = rosReadImage(imgMsg);

    %% Detect Markers
    [ids, locs] = readArucoMarker(I, "DICT_5X5_250");
    set(hIm, "CData", I);
    %% Update The Plot
    if ~isempty(ids)

        corners = locs(:,:,1);
        boxX = [corners(:,1); corners(1,1)];
        boxY = [corners(:,2); corners(1,2)];
        set(hPlot2D, "XData", boxX, "YData", boxY, "Visible", "on");
    else
        set(hPlot2D, "Visible", "off");
    end

end
```

Listing 2: SearchAndStop.m. Spins the robot until it detects any Aruco markerrs. The robot stops if any markers are detected.

```
%% Environment Setup
clc; clear; close all;
clear node; % clear previous node handle if it exists

setenv('ROS_DOMAIN_ID','37'); % Set to your robots ROS_DOMAIN_ID (check
    the robot"s ID card)
```

```
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
node = ros2node('ArucoRotate');
%% Subscribers And Publishers
imgSub = ros2subscriber(node,"/image_raw/compressed","sensor_msgs/
    CompressedImage", ...
"Reliability","besteffort","Durability","volatile","Depth",10);

velPub = ros2publisher(node,"/cmd_vel","geometry_msgs/Twist", ...
"Reliability","reliable","Durability","volatile","Depth",10);
velMsg = ros2message(velPub);

%% Visualization SETUP (Before Loop)
hIm = imshow(zeros(480, 640, "uint8")); hold on;
hPlot2D = plot(0, 0, "g-", "LineWidth", 3);
receive(imgSub, 10);
%% Main Loop That Will Run Indefinetely
while true
     %% Read Image
    imgMsg = receive(imgSub, 10);
    I = rosReadImage(imgMsg);
    %% Detect Markers
    [ids, locs] = readArucoMarker(I, "DICT_5X5_250");
    %If marker found
    set(hIm, "CData", I);
    if ~isempty(ids)
        velMsg.angular.z = 0;
        send(velPub, velMsg);
        corners = locs(:,:,1);
        boxX = [corners(:,1); corners(1,1)];
        boxY = [corners(:,2); corners(1,2)];
        set(hPlot2D, "XData", boxX, "YData", boxY, "Visible", "on");
    %If Marker Not Found
    else
        set(hPlot2D, "Visible", "off");
        velMsg.angular.z = 0.25;
        send(velPub, velMsg);
    end
end
```

Listing 3: VisualServoing.m. Spins the robot if no marker is found. Chases the marker iff the robot detects one.

```
%% Environment Setup
clear node; % clear previous node handle if it exists

setenv('ROS_DOMAIN_ID','36'); % Set to your robots ROS_DOMAIN_ID (check
    the robot"s ID card)
setenv('ROS_LOCALHOST_ONLY', '0'); % 0 implies multi-host communication
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp'); % Middleware

% Create a unique node name for this MATLAB session
```

```
node = ros2node('ArucoTrack');
%% Subscribers And Publishers
imgSub = ros2subscriber(node,"/image_raw/compressed","sensor_msgs/
    CompressedImage", ...
"Reliability","besteffort","Durability","volatile","Depth",10);

velPub = ros2publisher(node,"/cmd_vel","geometry_msgs/Twist", ...
"Reliability","reliable","Durability","volatile","Depth",10);
velMsg = ros2message(velPub);

%% Defining Constant Parameters and Specifications
markerSize = 0.1;

%The Corners Positions In The World Frame
worldPoints = [-markerSize/2 -markerSize/2 0;
    markerSize/2 -markerSize/2 0;
    markerSize markerSize 0;
    -markerSize/2 markerSize/2 0];

%Desired Stopping Distance
d_desired = 0.2;

%% The Control Parameters
k_lin = 1.5;
k_ang = 0.75;

%% Visualization SETUP (Before Loop)
hIm = imshow(zeros(480, 640, "uint8")); hold on;
hPlot2D = plot(0, 0, "g-", "LineWidth", 3);
receive(imgSub, 20);

%% The Error Data Vectors Initialization
linearErrors = [];
angularErrors = [];

%% The Program Loop
while true
     %% Read Image
    imgMsg = receive(imgSub, 20);
    I = rosReadImage(imgMsg);
    set(hIm, "CData", I);

    % Detect ArUco markers
    [ids, locs] = readArucoMarker(I, 'DICT_5X5_250');
    %locs = transpose(locs);
    % If No Marker Detected
    if isempty(ids)
        % STATE: SEARCHING - rotate in place
        velMsg.linear.x = 0;
        velMsg.angular.z = 0.3; % constant rotation speed
        send(velPub, velMsg);
        set(hPlot2D, "Visible", "off"); %Update Plot
    % If Marker Found
    else
```

```
        % STATE: TRACKING - control to chase marker
        %% Get the Rotation Matrix and Translation Vector
        [rotMatrix, tVector] = extrinsics(locs, worldPoints(:,1:2),
            cameraParams);
        x_R = tVector(3);
        y_R = -tVector(1);

        %% Compute The Errors
        e_theta = atan2(y_R,x_R);
        e_d = x_R - d_desired;
        angularErrors = [angularErrors e_theta];
        linearErrors = [linearErrors e_d];

        %% Compute The Control Inputs
        v = k_lin * e_d;
        w = k_ang * e_theta;

        %Anti Reverse Logic
        if e_d < 0.05
            v = 0;
        end
        if e_theta < 0.05
            w = 0;
        end
        %% Assign the Velocities
        velMsg.linear.x = v;
        velMsg.angular.z = w;
        send(velPub, velMsg);

        %% Plotting The Picture
        corners = locs(:,:,1);
        boxX = [corners(:,1); corners(1,1)];
        boxY = [corners(:,2); corners(1,2)];
        set(hPlot2D, "XData", boxX, "YData", boxY, "Visible", "on");
    end
end
```

Listing 4: Plots the error data that is received from the VisualServoing.m script.

```
figure;
subplot(2,1,1);
plot(linearErrors);
grid on;
title("Linear Errors");
xlabel("Iteration");
ylabel("Error [m]");


subplot(2,1,2);
plot(angularErrors);
grid on;
title("Angular Errors");
xlabel("Iteration");
ylabel("Error [rad]");
```