

**Mechatronics Engineering Department
Al-Khwarizmi Engineering College
University of Baghdad
Iraq**



DESIGN AND CONTROL OF MOBILE ROBOT VIA VISION SYSTEM

**A project report submitted to Mechatronics Engineering Department in partial
fulfilment of the requirements for the award of the degree of Bachelor of
Science in Mechatronics Engineering**

IBRAHIM YASIR

AYMEN NAWAF

AYA KHDAIR

Dr. Malik M. Ali

Dr. Alaa A. Hasan

2018-2019

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations, which have been duly acknowledged.

Signature : _____

Name : _____

Date : _____

Signature : _____

Name : _____

Date : _____

Signature : _____

Name : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitle “**DESIGN AND CONTROL OF MOBILE ROBOT VIA VISION SYSTEM**” was prepared by **IBRAHIM YASIR, AYMEN NAWAF** and **AYA KHUDAIR** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Mechatronics Engineering at University of Baghdad.

Approved by,

Signature : _____

Supervisor : _____

Date : _____

Signature : _____

Supervisor : _____

Date : _____

Specially dedicated to
Our beloved families, friends, teachers
And to science...

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisors, **Dr. Malik M. Ali and Dr. Alaa A. Hasan** for their invaluable advice, guidance and their enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to our loving parents and friends who had helped and given us encouragement...

DESIGN AND CONTROL OF MOBILE ROBOT VIA VISION SYSTEM

ABSTRACT

The robotics industry is growing more and more each year, and they are becoming a part of our lives if we like it or not. All robots can become intelligent machines when equipped with a powerful controller and a few sensors. One of these sensors is the camera. Perhaps it is even the most important one.

This project is the design and control of a mobile robot via an efficient vision system; this includes guidance, navigation and control of the said robot. This robot can be used to access environments where it is difficult or dangerous for humans to enter for whatever reason.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF SYMBOLS / ABBREVIATIONS	ix

Chapter One	10
Introduction	10
1.1 Background	10
1.2 Aims and Objectives	11
1.3 Literature Review	11
Chapter Two	12
Methodology	12
2.1 The Hardware	12
2.1.1 The Vehicle's Body	12
2.1.2 The Actuators	12
2.1.3 The Cameras and Cameras Mount	13
2.1.4 Arduino and Ultrasonic Sensor	13
2.2 Principle of Operation	14
2.2.1 Stereo Vision	14
2.2.2 The Disparity	14
2.2.3 Depth Extraction	14
2.3 Implementation	15
2.3.1 Cameras Calibration	15

2.3.2 Image Processing	15
2.3.3 Method	16
2.3.4 Matlab App Designer	16
Chapter Three	18
Results and Discussions	18
3.1 Evaluation and Tests	18
3.1.1 Tests	18
3.1.2 Notes	19
3.2 Future Work	20
Chapter Four	21
Conclusion and Recommendations	21
4.1 Conclusion	21
4.2 Recommendations	21
References	22
APPENDICES	23

LIST OF SYMBOLS / ABBREVIATIONS

Mobile Robot via Vision System 1	MRVS-1
User Interface	UI
Focal length	f
Baseline	b

Chapter One

Introduction

1.1 Background

Object distance measurement against certain references, such as an observer, is of great necessity, especially in the fields of monitoring and security, industry, navigation, robotics, and even for many applications in smart vehicles. [4]

We can classify the distance measurement operation into two classes, Active Measurement and Passive Measurement.

The active measurement means that the measuring process is done by sending a signal to the object and receiving its reflection. This includes methods such as radar, infrared, radio signal, ultrasonic sound, laser, and so on [2].

While the passive measurement is done without sending any signal to the object. It is carried out by the use of computer vision that makes use of visual images of objects. This research attempts to calculate the distance of an object by using visual information gained from a pair of images taken by two cameras, which is known as stereo image. Using trigonometry, correlations between object positions from right and left tells the distance between the object and an observer this is the core concept of stereo vision and triangulation [1].

In our project -that we will call ‘Mobile Robot via Vision System 1’ and we will abbreviate it to MRVS-1- we will use both active and passive measurement types.

Active type in Ultrasonic Sensor to measure the distance from an obstacle. In addition, a passive type in the stereo cameras system.

1.2 Aims and Objectives

The goal of our project is to design and implement a stereo vision system with regular mono cameras and to use it with a mobile robot to guide and navigate it accordingly. The implementation of a vision system on a mobile robot enables the later to be controlled and navigated with ease by the user, more easily than without a vision system.

We aim to design everything ourselves to minimize the cost and to ensure we will get the product to be exactly how we want it.

1.3 Literature Review

We needed a method of measuring distance by a camera and the best approach we could find is using a stereo vision as suggested by (Kusworo Adi & Catur Edi Widodo, 2017)[1]. However, in the beginning we must know what a stereo vision is, how many methods can we use, and what is the best one.

Stereo vision is a method of using two cameras with the same performance fixed together with constant distance between them. This allows us to extract the depth between the cameras and the objects in front of them. (Hou A-Lin, Cui Xue, Geng Ying, Yuan Wen-Ju, Hou Jie, 2011)[4] This paper explains thoroughly the process of measurement and the equations used. We will discuss this also in the following chapter.

(Elena Bebeșelea-Sterp, Raluca Brad, Remus Brad, 2017)[3] Proposed a thoroughly comparison between stereo vision algorithms. However, we will compare only two methods in this literature review. The first method is the use of features extraction to determine the objective. The second method is choosing a certain shape, and then identify it via a certain software.

The problem with the first method as stated by the said paper is that the features extracted from the same image can be different each time a picture is captured.

The second method provides a better and a more concrete approach. No matter how many pictures captured, the object is always the same.

Chapter Two

Methodology

2.1 The Hardware

2.1.1 The Vehicle's Body

The vehicle's body was manually made of Alcopond of 5mm thickness. Part (a) is 200mm x 400mm. (Figure 2.1 [a]) Part (b) is 100mm x 200mm. (Figure 2.1 [b])

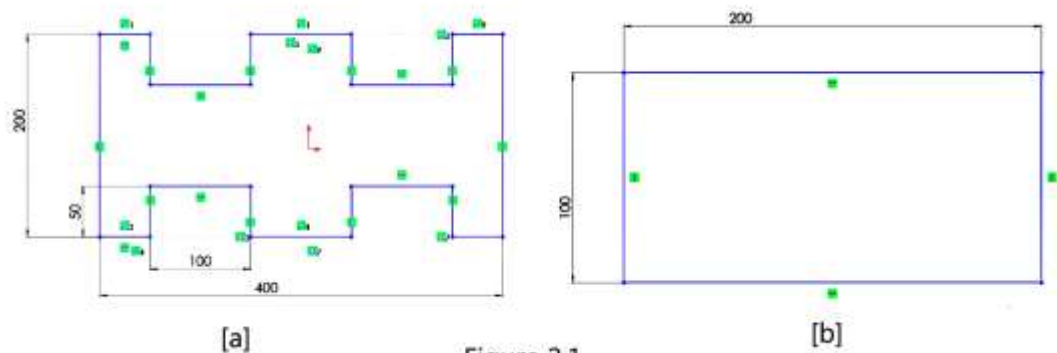
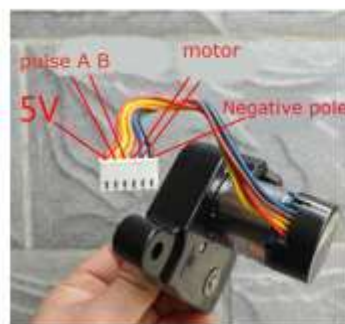


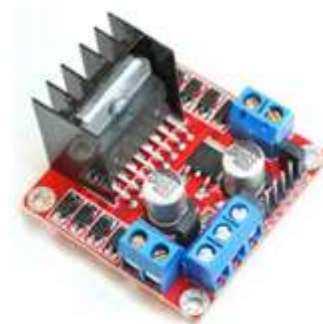
Figure 2.1

2.1.2 The Actuators

For the actuators, four [14390w DC] motors (Figure 2.2 [a]) were used for each tyre of the vehicle and were driven by two [L298N DC] drivers (Figure 2.2 [b])



[a]



[b]

Figure 2.2

2.1.3 The Cameras and Cameras Mount

Two Logitech C270 cameras were used in the project Figure 2.3 shows a list of their specifications.

- Product Specifications**
- Dimensions (L x W x H): 71 mm x 31 mm x 24 mm
 - Weight: 75 grams (including clip and cable)
- For HD 720p video calling (on Logitech Vid™ HD):
- 2.4 GHz Intel® Core™2 Duo
 - 2 GB RAM
 - 200 MB hard drive space
 - USB 2.0 port
 - 1 Mbps upload speed or higher
 - 1280 x 720 screen resolution



Figure 2.3

The cameras were mounted vertically on the vehicle by the 'Cameras Mount', which was designed by Solidworks and printed with a 3D printer. (Figure 2.4)

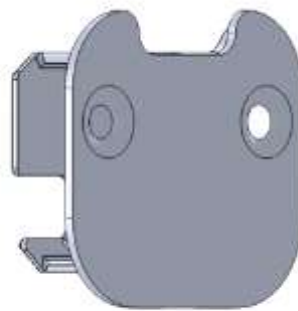


Figure 2.4

2.1.4 Arduino and Ultrasonic Sensor

One Arduino Uno was used as control unit for the actuators. (Figure 2.5 [a])

In addition, one HC-SR04 ultrasonic was used. (Figure 2.5 [b])



[a]



[b]

Figure 2.5

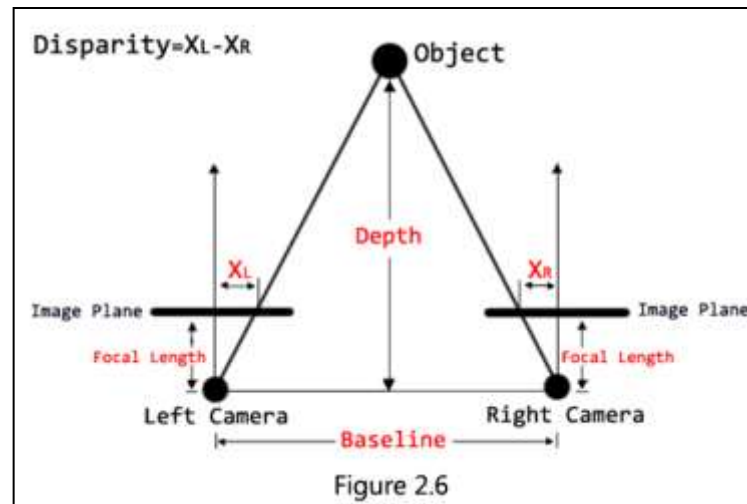
2.2 Principle of Operation

2.2.1 Stereo Vision

The term stereo vision indicates the use of stereo camera or two regular cameras. In our scope, we used two ordinary identical webcams (Logitech C270). This approach allows us to identify the depth as well as the shape and colour by means of a technique named The Disparity. In the following paragraph, we will discuss what disparity is.

2.2.2 The Disparity

This technique is exclusive to stereo vision because it relies on two images taken by identical cameras calibrated with fixed distance between their centres. Disparity is the difference of the x-axis position between the two images taken by the cameras; Figure (2.6) illustrates this concept well. More in appendix A.



2.2.3 Depth Extraction

Depth is extracted using equation (2.1) which relies on the disparity, focal length of the cameras and the baseline distance between the cameras centres. [1]

$$\text{Depth} = \frac{\text{Focal Length} * \text{Baseline}}{\text{Disparity}} \quad \text{Equation 2.1}$$

2.3 Implementation

2.3.1 Cameras Calibration

The two cameras were calibrated as a stereo camera with Matlab's Stereo Camera Calibration App. The baseline distance between the two centres was taken as 10cm. The calibration is done by taking 10 to 20 pictures[3][5] by each camera (while fixed in position and orientation) of the same scene. The scene must be a checkerboard with known dimensions. Matlab configures both cameras' parameters, which will be needed in the image processing later. Figure (2.7) shows the results of calibration session previously done for the project.

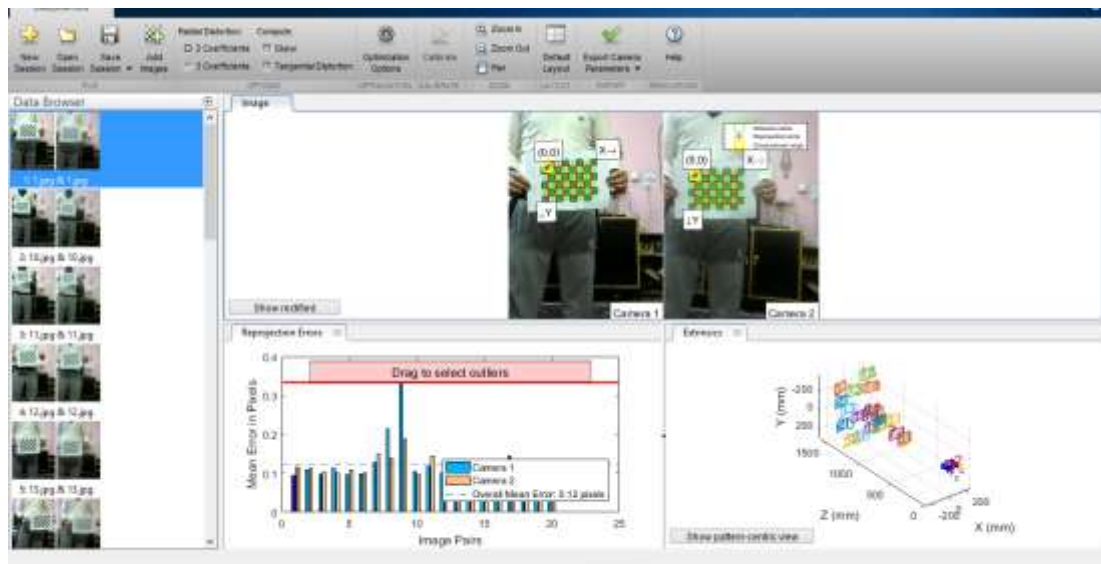


Figure 2.7

2.3.2 Image Processing

After calibrating the cameras, they will be used to identify a circle shape; which we will consider as our objective. Then the depth (or the distance) between the cameras and the objective will also be found.

Hough Transform is used to find the circles. Which is a feature extraction technique used in image analysis, computer vision, and digital image processing.[4]

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972.

2.3.3 Method

Before finding the circle and operation of adjusting must be performed on the images to ensure that they are in the same orientation. This operation is based on the camera parameters extracted by the calibration.

Matlab facilitates task of finding circles drastically so that one command is enough to detect circles with chosen radius in an image. This command is:

```
[centers, radii] = imfindcircles(image,[minimumRadius maximumRadius]);
```

Matlab also makes the task of finding the disparity easy with one command resulting a new image with pixel values proportional to it's disparity value. Figure (2.8)

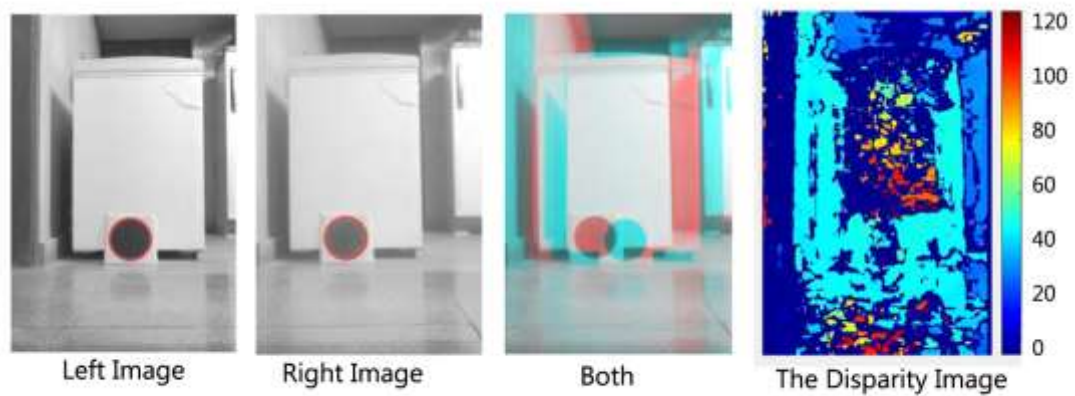


Figure 2.8

After we have the centres of the circles as (x, y) sets and the disparity image as a matrix, the real value of the centre is taken by averaging the two centres' (x, y) sets then extracting the value of the average (x, y) set pixel in the disparity image. This value is then placed in the denominator of equation (2.1) with the remaining values being known constants, the depth is extracted.

2.3.4 Matlab App Designer

This method is implemented and executed by Matlab through App Designer, which allows for a user interface (UI) to be designed to execute the codes easily.

Figure (2.10) shows the UI that was designed and used for the project. Table (2.1) demonstrates the function of each button.

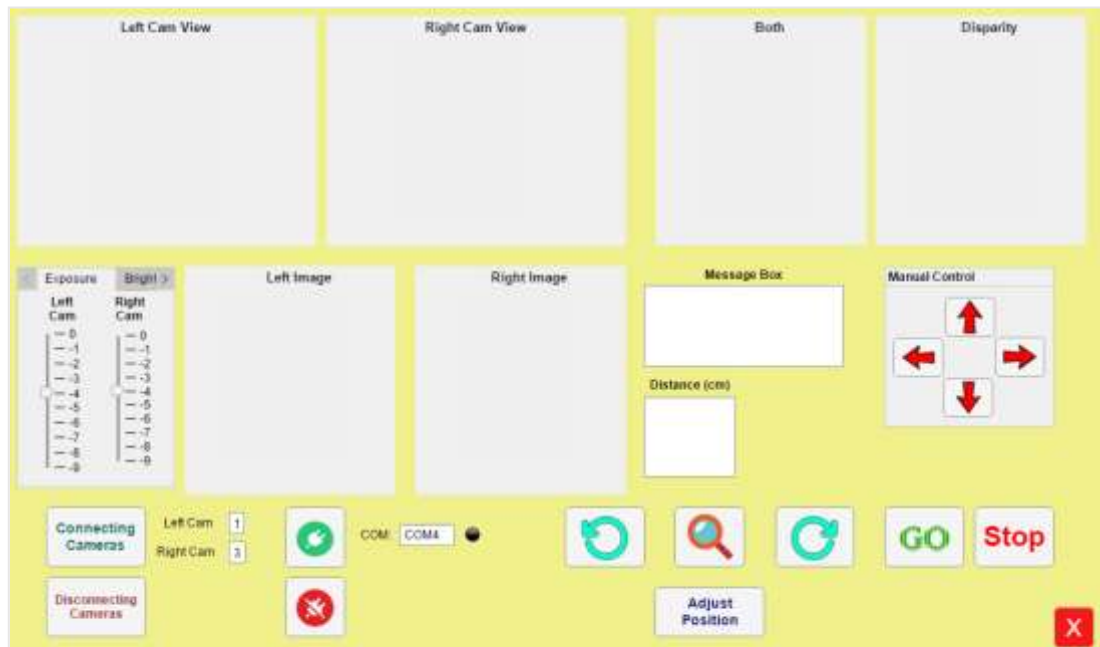


Figure 2.10










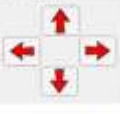





Button	Function	Button	Function
	Connects cameras to Matlab		After finding a circle, this button lets the robot go to it's location
	Disconnects cameras to Matlab		Stops the robot
	Connects Arduino to Matlab		Rotates the robot counter-clockwise by 45 degrees
	Disconnects Arduino from Matlab		Rotates the robot clockwise by 45 degrees
	Searches for circles and displaying the distance if it exists		Manual control of the robot with high resolution. Each arrow moves the robot in it's direction
	Adjusts the position of the robot so that the centre of the circle is in front of the robot in a straight line		Displays messages in case of an error
	Allows the user to adjust the exposure and brightness of both cameras seperately		Displays the distance of the objective if found
			Closes the application

Table 2.1

To have a better understanding of the operation of this algorithm, see (appendix B) that contains the whole code of the UI.

Chapter Three

Results and Discussions

3.1 Evaluation and Tests

The project (MRVS-1), as we like to call it, is considered as a success as it passed several tests conducted by our team.

3.1.1 Tests

The first test was putting the MRVS-1 in front of the target object in a straight line and see how it is going to respond. The MRVS-1 was able to measure the distance to a certain precision as shown by the Figure (3.1).

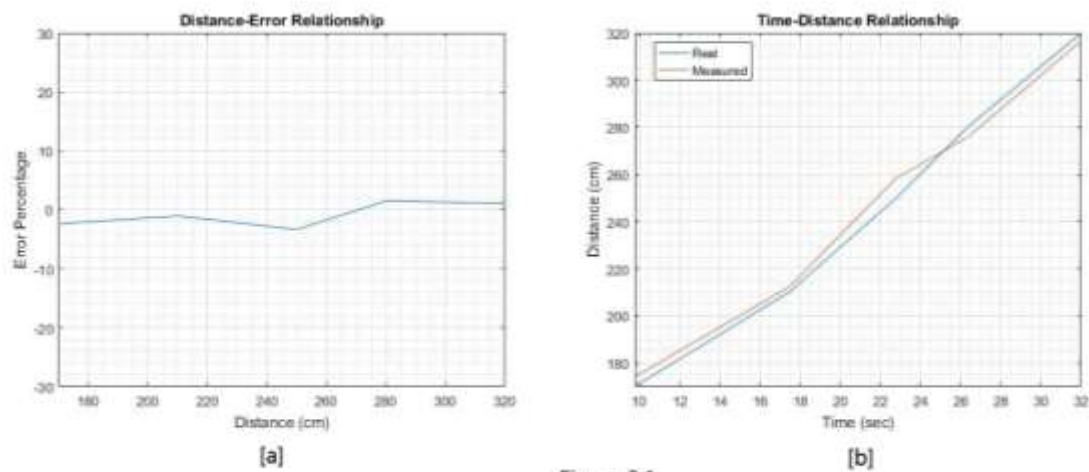


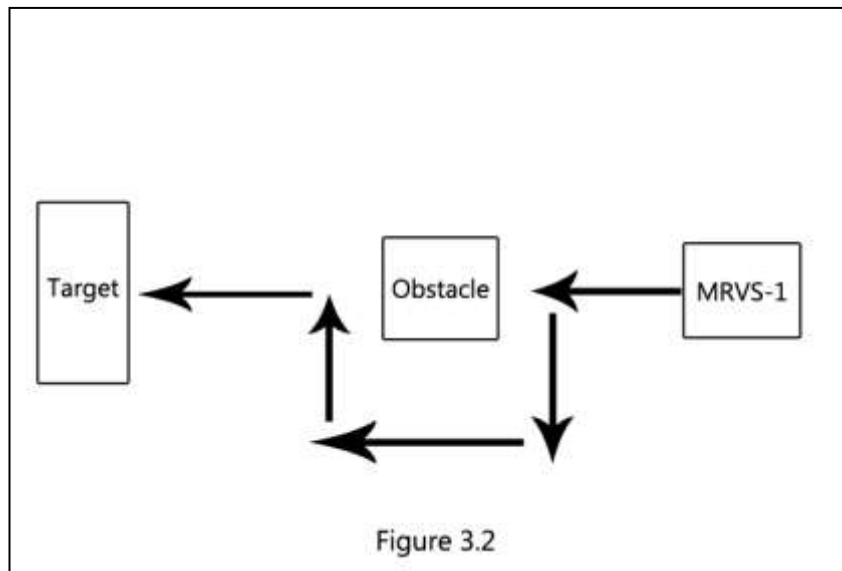
Figure 3.1

Figure (3.1 [a]) demonstrates the total error percentage with the distance, the maximum error occurs roughly at 250cm and is less than 3.5% which is considered fairly acceptable.

While Figure (3.1 [b]) demonstrates the real distance (red) compared to the measured distance (blue) corresponding to time.

The second test was similar to the first test but with an obstacle in between the MRVS-1 and the target. Conducted to see the robot's ability to avoid obstacles.

The MRVS-1 behaved in the way shown in Figure (3.2).



To see this thoroughly we filmed it and uploaded it to Youtube. You can check the video on <https://www.youtube.com/watch?v=QlvtstH1SLQ>

3.1.2 Notes

One would notice that the MRVS-1 is particularly slow and takes a lot of time to process an image. This is due to several factors including:

- The main controlling unit is the PC containing the UI, and our PC's processor is kind of out of date, hence, the delay.
- The secondary controlling unit is the Arduino Uno, known for its time problems.
- The cameras were common webcams and not specialized for the machine vision purpose. As this was one of the goals of our project, to create a cheap stereo camera.
- The Arduino and the cameras, connected via USB 2.0 cables, are not very fast, compared to the USB 3.0.

3.2 Future Work

Overtime, we grew fond of the idea of our project and we set our minds on some new techniques regarding our scope of work to develop MRVS-1. Some of these techniques are:

- a. Using Raspberry Pi instead of Arduino and Matlab. Raspberry Pi provides the specialized tools for image processing and wireless communication with the PC. This would increase the mobility and durability of MRVS-1.
- b. Using better cameras. One of the main inconveniences that we encountered was the resolution of the cameras, as they were not as good as we needed.
- c. Using machine-learning techniques to increase the intelligence of our target identification algorithm.

Perhaps one day, we will even develop a self-driving car as this is definitely on our goals-list.

Chapter Four

Conclusion and Recommendations

4.1 Conclusion

As stated before, the goal of the project was to design and control a mobile robot by a vision system. We chose to work with stereo vision and were successful with the implementation. We used Matlab App Designer for the User Interface (UI) and Arduino Uno with the Arduino library in Matlab for controlling the motors.

The tests we conducted on MRVS-1 (The name we call our project) proved that our project is a successful laboratorial device, which can be, further developed to work in real life applications.

4.2 Recommendations

Although our tests' results were acceptable, they could have been better, had we used different methods of control, for example, instead of using Matlab and Arduino we could have used Raspberry Pi. Matlab is reliable but on slow computers, it could be a burden to the overall integrity of the project.

Using cheap cameras may lead to certain complications, so; it is advised to use cameras with high resolution and efficiency.

Buying a vehicle model that is ready to be used can save a lot of time considering the time we spent on designing the vehicle that could be spend on implementing other components.

References

Journal Article

- [1] DISTANCE MEASUREMENT WITH A STEREO CAMERA. Kusworo Adi, Catur Edi Widodo, Diponegoro University, Semarang, Indonesia (2017)
- [2] Sukhla, N, A Review on Image Based Target Distance and Height Estimation Technique Using Laser Pointer And Single Video Camera For Robot Vision, International Journal of Engineering Research And Reviews, Vol. 3, Issue 1, (2015).
- [3] A Comparative Study of Stereovision Algorithms, Elena Bebeșelea-Sterp, Raluca Brad, Remus Brad, International Journal of Advanced Computer Science and Applications, (2017)

Book

- [4] Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. (2001)

Conference Proceedings

- [5] Hou, A. L., Cui, X., Geng, Y., Yuan, J. and Hou, J., (2011), Measurement of Safe Driving Distance based on Stereo Vision, Sixth International Conference on Image and Graphics (ICIG), Hefei, Anhui, China: p. 902-907.

Website

- [6] [https://www.mathworks.com/help/vision/referencelist.html?type=function&category=.](https://www.mathworks.com/help/vision/referencelist.html?type=function&category=)
- [7] https://en.wikipedia.org/wiki/Computer_stereo_vision

APPENDICES

APPENDIX A: Graphs

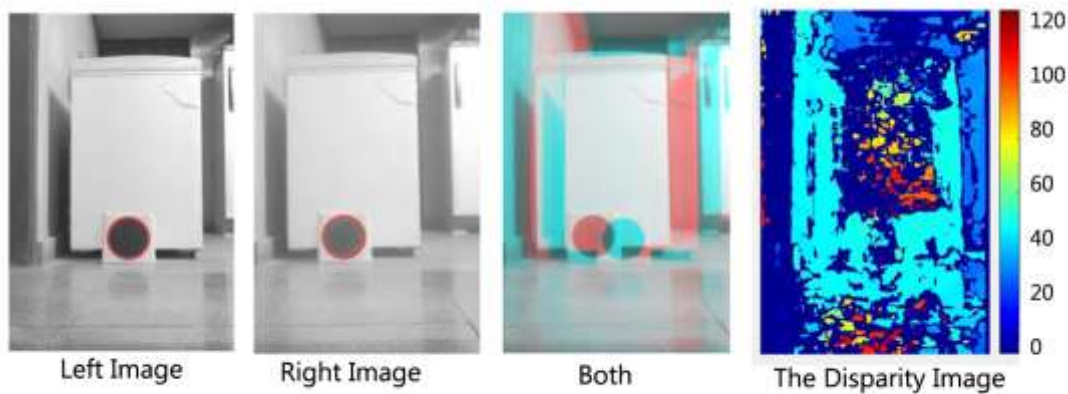
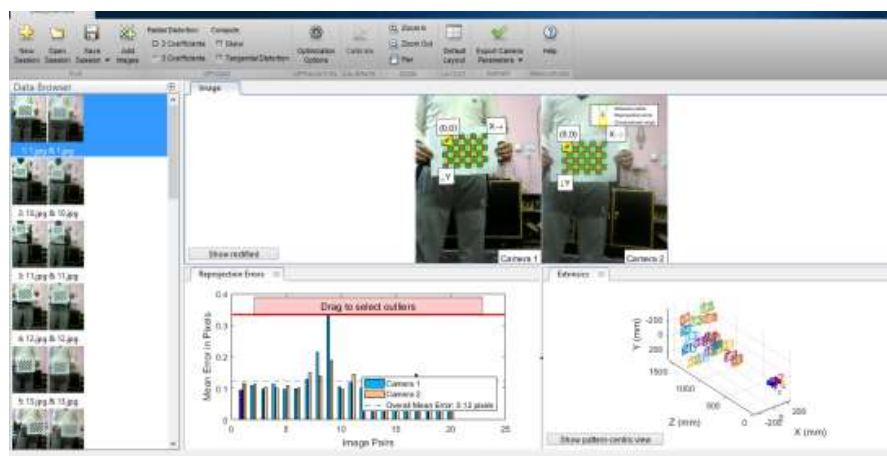
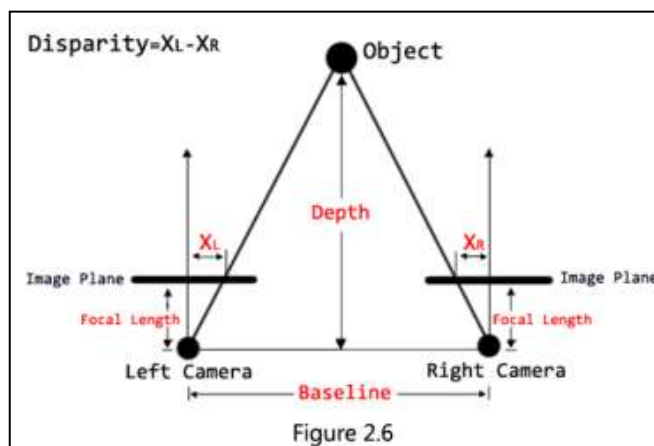


Figure 2.8

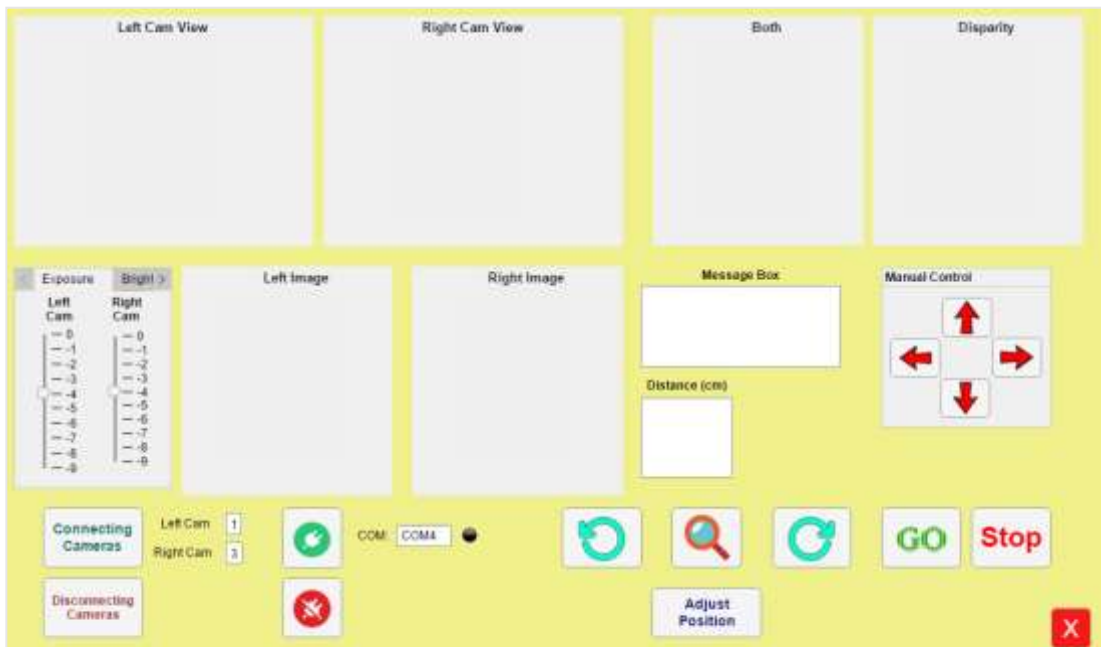


Figure 2.10

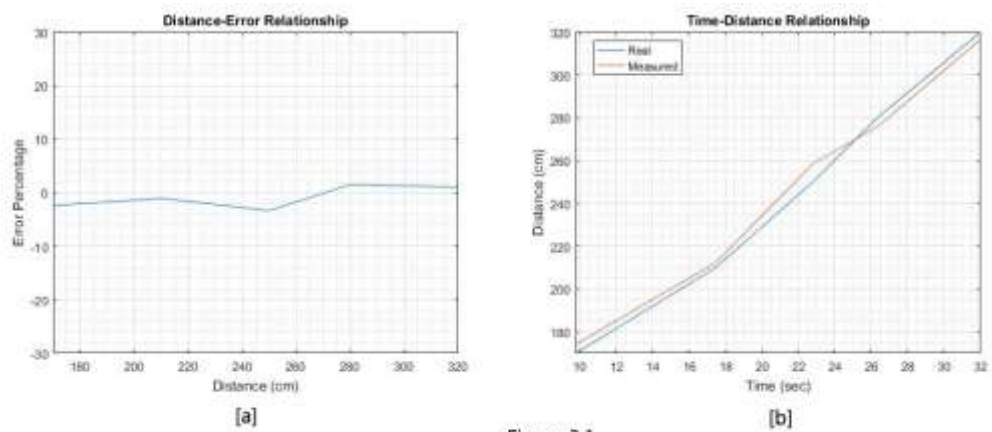


Figure 3.1

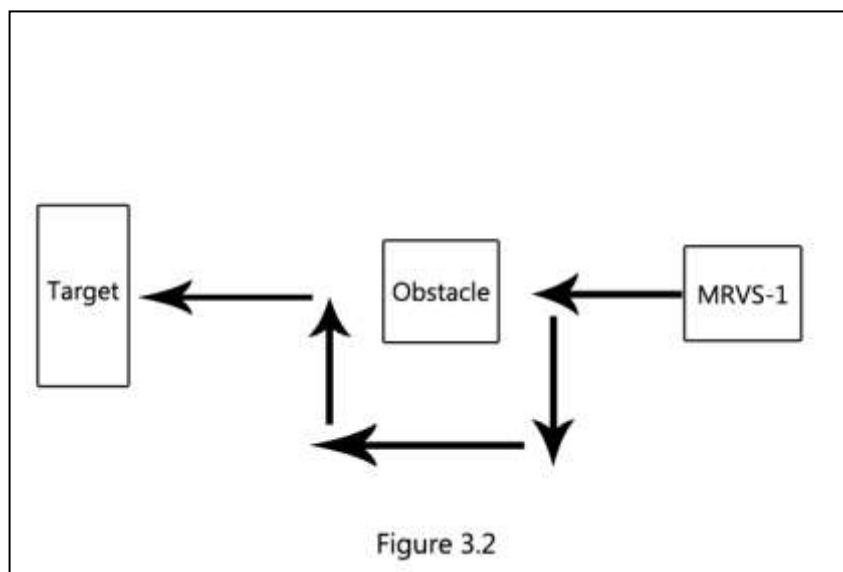


Figure 3.2

APPENDIX B: Computer Programme Listing

```
% Code that executes after component creation
function startupFcn(app)
    delete(instrfind({'Port'},{'COM4'}));
    global cam1;
    global cam2;
    global a;
    delete(cam1); delete(cam2);
    clear cam1; clear cam2; clear a;
    global blankPic;
    blankPic=imread('blank.jpg');
    imshow(blankPic, 'parent', app.UIAxes);
    imshow(blankPic, 'parent', app.UIAxes2);
    imshow(blankPic, 'parent', app.UIAxes3);
    imshow(blankPic, 'parent', app.UIAxes4);
    imshow(blankPic, 'parent', app.UIAxesLeft);
    imshow(blankPic, 'parent', app.UIAxesRight);
    app.connectLamp.Color='black';
    cd 'C:\Users\ibrahim\Desktop';
end

% Button pushed function: connectButton
function connectButtonPushed(app, event)
    %load('calibrationSession5.mat');
    global a; global sensor;
    global r1;global r2;global l1;global l2;
    global m1;global m2; global m3; global m4;
    global trigPin; global echoPin;
    comVariable= app.COMEditField.Value;
    a = arduino(comVariable, 'Uno','Libraries','JRodrigoTech/HCSR04');
    app.connectLamp.Color='green';
    % Right-hand side motors:
    r1='D2';
    r2='D4';
    % Left-hand side motors:
    l1='D7';
    l2='D8';
    % Analog signal for motors speed:
    m1='D10';
    m2='D3';
    m3='D6';
    m4='D9';
    % Ultrasonic code:
    trigPin = 'D11';    % Trigger
    echoPin = 'D12';    % Echo
    sensor = addon(a, 'JRodrigoTech/HCSR04', trigPin, echoPin);
    configurePin(a,r1, 'DigitalOutput');
    configurePin(a,r2, 'DigitalOutput');
    configurePin(a,l1, 'DigitalOutput');
    configurePin(a,l2, 'DigitalOutput');
    configurePin(a,m1, 'DigitalOutput');
```

```

        configurePin(a,m2, 'DigitalOutput');
        configurePin(a,m3, 'DigitalOutput');
        configurePin(a,m4, 'DigitalOutput');
        writePWMPWMVoltage(a,m1,(255/255)*5);
        writePWMPWMVoltage(a,m2,(245/255)*5);
        writePWMPWMVoltage(a,m3,(245/255)*5);
        writePWMPWMVoltage(a,m4,(255/255)*5);
    end

% Callback function
function DisconnectButtonPushed(app, event)
    app.connectLamp.Color='black';
    delete(instrfind({'Port'},{'COM4'}));
    clear
end

% Button pushed function: ConnectingCamerasButton
function ConnectingCamerasButtonPushed(app, event)
    global cam1; global cam2;
    L= app.LeftCamEditField.Value;
    R=app.RightCamEditField.Value;
    cam1=webcam(L);
    cam2=webcam(R);
    value1 = app.LeftCamSlider.Value; %setting exposure of cam1
    cam1.Exposure=value1;
    value2 = app.RightCamSlider.Value; %setting exposure of cam2
    cam2.Exposure=value2;
    frame1=snapshot(cam1);
    frame2=snapshot(cam2);
    imwrite(frame1, 'livefeedLeft.jpg');
    imwrite(frame2, 'livefeedRight.jpg');
    frame1=imread('livefeedLeft.jpg');
    frame2=imread('livefeedRight.jpg');
    frame1=imrotate(frame1,90);
    frame2=imrotate(frame2,90);
    im1=image(app.UIAxesLeft, zeros(size(frame1),'uint8'));
    axis(app.UIAxesLeft, 'image');
    im2=image(app.UIAxesRight, zeros(size(frame2),'uint8'));
    axis(app.UIAxesRight, 'image');
    preview(cam1,im1);
    preview(cam2,im2);
end
function DisconnectingCamerasButtonPushed(app, event)
    global cam1;
    global cam2;
    global blankPic;
    delete(cam1); delete(cam2);
    clear cam1; clear cam2;
    imshow(blankPic, 'parent', app.UIAxesLeft);
    imshow(blankPic, 'parent', app.UIAxesRight);
end

```

```

% Button pushed function: searchButton
function searchButtonPushed(app, event)
    global cam1; global cam2;
    % global a; global r1;global r2;global l1;global l2;
    global x;
    global distance;
    global blankPic;
    try
        img = snapshot(cam1);
        img2 = snapshot(cam2);
        imwrite(img, 'picLeft.jpg');
        imwrite(img2, 'picRight.jpg')
    catch
        app.MessageBoxTextArea.Value='Cameras are not connected,
connect to cameras Please.';
    end;
    J1=imread('picLeft.jpg');
    J2=imread('picRight.jpg');
    J1=imrotate(J1,90);
    J2=imrotate(J2,90);
    load('calibrationSession5.mat');
    [J1,J2] =
rectifyStereoImages(J1,J2,calibrationSession.CameraParameters);
    I1=rgb2gray(J1);
    I2=rgb2gray(J2);
    J1=imgaussfilt(I1,1.2);
    J2=imgaussfilt(I2,1.2);
    J1=255-J1;
    J2=255-J2;
    try
        %K1=255-I1;
        [Center1,Radius1]=imfindcircles(I1,[10
150], 'ObjectPolarity',...
        'dark', 'Sensitivity',0.85);

        Center1=Center1(1,:);
        Radius1=Radius1(1,:);
    catch
        answer='No Circle Found!';
        app.MessageBoxTextArea.Value=answer;
        app.DistancecmTextArea.Value=' ';
    end
    try
        [Center2,Radius2]=imfindcircles(I2,[10
150], 'ObjectPolarity',...
        'dark', 'Sensitivity',0.85);
        Center2=Center2(1,:);
        Radius2=Radius2(1,:);
    catch
        answer='No Circle Found!';
        app.MessageBoxTextArea.Value=answer;
        app.DistancecmTextArea.Value=' ';
    end

    if mean(Radius1) > 0 && mean(Radius2) > 0

```

```

disparityRange = [0 128]; % arbitrary number and divisible
by 8, max = 128
disparityMap = disparity(J1,J2,'BlockSize',
15,'DisparityRange',disparityRange);

%disparity image
imshow(disparityMap,disparityRange, 'parent', app.UIAxes4);
answer='Circle Found!';
app.MessageBoxTextArea.Value=answer;
try

    theCenter=(Center1+Center2)/2;
    x=floor(theCenter(1));
    y=floor(theCenter(2));
    f=846;
    b=10;
    distance=f*b/disparityMap(y,x);
    if distance < 40
        distance=f*b/disparityMap(y,x-10);
    end
    if distance < 40
        distance=f*b/disparityMap(y,x-20);
    end
    if distance < 40
        distance=f*b/disparityMap(y,x-30);
    end
    if distance < 40
        distance=f*b/disparityMap(y,x-40);
    end
    if distance < 40
        distance=f*b/disparityMap(y,x-50);
    end
    distanceRead=distance;
    distance=0.0016*distanceRead^2+0.87*distanceRead+3.4;
    %showing messages:
    distanceStr=num2str(distance);
    app.DistancecmTextArea.Value=distanceStr;

    if distance < 0
        app.DistancecmTextArea.Value=' ';
        app.MessageBoxTextArea.Value='Try again :(';
    end
catch
    answer='No Circle Found!';
    app.MessageBoxTextArea.Value=answer;
    app.DistancecmTextArea.Value=' ';
    imshow(blankPic, 'parent', app.UIAxes);
    imshow(blankPic, 'parent', app.UIAxes2);
    imshow(blankPic, 'parent', app.UIAxes3);
    imshow(blankPic, 'parent', app.UIAxes4);

end
else
    answer='No Circle Found!';

```

```

        app.MessageBoxTextArea.Value=answer;
        app.DistancecmTextArea.Value=' ';

        imshow(blankPic, 'parent', app.UIAxes);
        imshow(blankPic, 'parent', app.UIAxes2);
        imshow(blankPic, 'parent', app.UIAxes3);
        imshow(blankPic, 'parent', app.UIAxes4);
    end
    %showing proper images:
    %left
    imshow(I1, 'parent', app.UIAxes);
    %right
    imshow(I2, 'parent', app.UIAxes2);
    %both
    imshow(stereoAnaglyph(I1,I2), 'parent', app.UIAxes3)

    java.lang.Thread.sleep(100);
end

% Button pushed function: goButton
function goButtonPushed(app, event)
    global sensor;
    global sensorDistance;
    global distance;
    global cam1;
    global cam2;
    global blankPic;
    global oldDistance;
    oldDistance=distance;
    obstacleExist=0;
    obstacleDistance=10;

    velocity= 20.387 ; %cm/sec
    distanceReal=distance-60;
    if distance==0
        distanceReal=0;
    end

    time = abs(distanceReal/velocity); %time needed to get there in
seconds

    if distance==0
        app.MessageBoxTextArea.Value='Press the Search button
first.';
    end
    if floor(time)>=4 && floor(time)<=6
        n=2;
    elseif floor(time)>6 && floor(time)<=9
        n=4;
    elseif floor(time)>9 && floor(time)<=14
        n=6;
    elseif floor(time)>14
        n=8;
    else
        n=1;
    end
end

```

```

end
for i=1:n

    if time==0
        break;
    end

    straightAhead(app);
    t1=now;

    t=0;
    while t < time/n
        sensorDistance = readDistance(sensor);
        sensorDistance=sensorDistance*100;

        if sensorDistance <= obstacleDistance
            %avoid function:

            app.MessageBoxTextArea.Value='Obstacle is found!
Avoiding now.';

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            turnCounterClockwise(app);    %1) turning left
            java.lang.Thread.sleep(1000);

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            straightAhead(app);          %2) going
            java.lang.Thread.sleep(1500);

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            turnClockwise(app);          %3) turning right
            java.lang.Thread.sleep(1000);

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            straightAhead(app);          %4) going straight
            java.lang.Thread.sleep(3000);

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            turnClockwise(app);          %5) turning right
            java.lang.Thread.sleep(1100);

            stopBoy(app);                %stopping
            java.lang.Thread.sleep(100);

            straightAhead(app);          %6) going straight

```

```

        java.lang.Thread.sleep(1500);

        stopBoy(app);                                %stopping
        java.lang.Thread.sleep(100);

        turnCounterClockwise(app);                    %7) turning left
        java.lang.Thread.sleep(1150);

        stopBoy(app);                                %stopping
        java.lang.Thread.sleep(100);
        obstacleExist=1;
        break; %break while
    end %end if

    t2=now;
    t=t2-t1;
    t=t*60*60*24;

end %end while
if distance==0
    obstacleExist=2; %will need it after the for loop
    %                                %and has nothing to do with obstacles.
    break; %break for
end
app.MessageBoxTextArea.Value='Following circle.';
stopBoy(app);

if obstacleExist ==1
    app.MessageBoxTextArea.Value='Obstacle Avoided.';
    break; %break for
end
%here was the search code;
searchButtonPushed(app);

end      % end of for

if obstacleExist == 1
    distance=0;
    searchGo(app);

elseif obstacleExist==2
    app.MessageBoxTextArea.Value='Press the Search button
first.';
    app.DistancecmTextArea.Value=' ';

else

    answer='We are here!';
    app.MessageBoxTextArea.Value=answer;
    app.DistancecmTextArea.Value=' ';
    distance=0;

end

end
end

```

```

properties (Access = private)
end

methods (Access = private)

function straightAhead(app)
    global a;
    global r1; global r2; global l1; global l2;
    try
        writeDigitalPin(a,r1,1);
        writeDigitalPin(a,r2,0);
        writeDigitalPin(a,l1,1);
        writeDigitalPin(a,l2,0);
    catch
        app.MessageBoxTextArea.Value='Please Connect to Arduino';
    end
end

function turnClockwise(app)
    global a;
    global r1; global r2; global l1; global l2;
    try
        writeDigitalPin(a,r1,0);
        writeDigitalPin(a,r2,1);
        writeDigitalPin(a,l1,1);
        writeDigitalPin(a,l2,0);
    catch
        app.MessageBoxTextArea.Value='Please Connect to Arduino';
    end
end

function turnCounterClockwise(app)
    global a;
    global r1; global r2; global l1; global l2;
    try
        writeDigitalPin(a,r1,1);
        writeDigitalPin(a,r2,0);
        writeDigitalPin(a,l1,0);
        writeDigitalPin(a,l2,1);
    catch
        app.MessageBoxTextArea.Value='Please Connect to Arduino';
    end
end

function reverse(app)
    global a;
    global r1; global r2; global l1; global l2;
    try
        writeDigitalPin(a,r1,0);
        writeDigitalPin(a,r2,1);
        writeDigitalPin(a,l1,0);
        writeDigitalPin(a,l2,1);
    catch
        app.MessageBoxTextArea.Value='Please Connect to Arduino';
    end
end

```



```

function stopBoy(app)
    global a;
    global r1; global r2; global l1; global l2;
    try
        writeDigitalPin(a,r1,0);
        writeDigitalPin(a,r2,0);
        writeDigitalPin(a,l1,0);
        writeDigitalPin(a,l2,0);
    catch
        app.MessageBoxTextArea.Value='Please Connect to Arduino';
    end
end
function searchGo(app)
    global distance;
    global oldDistance;

    java.lang.Thread.sleep(1000)
    searchButtonPushed(app);
    java.lang.Thread.sleep(1500);

    AdjustPositionButtonPushed(app);
    java.lang.Thread.sleep(1000);

    searchButtonPushed(app);
    java.lang.Thread.sleep(1500);
    if distance > oldDistance
        java.lang.Thread.sleep(1000)
        searchButtonPushed(app);
        java.lang.Thread.sleep(1500);

        AdjustPositionButtonPushed(app);
        java.lang.Thread.sleep(1000);

        searchButtonPushed(app);
        java.lang.Thread.sleep(1500);

        if distance > oldDistance
            app.MessageBoxTextArea.Value='There is an error, please
search again';
        else
            goButtonPushed(app);
            java.lang.Thread.sleep(1000);
            searchButtonPushed(app);
            java.lang.Thread.sleep(1500);
            goButtonPushed(app);

        end
    else
        goButtonPushed(app);
        java.lang.Thread.sleep(1000);
        searchButtonPushed(app);
        java.lang.Thread.sleep(1500);
        goButtonPushed(app);
    end
end
end
end

```

تصميم ومراقبة الروبوت المحمول عبر نظام رؤية

الخلاصة

صناعة الروبوتات هي من الصناعات النامية التي تنمو كل عام وتكاد ان تصبح جزءاً لا يتجزأ من حياتنا. جميع الروبوتات تصبح الات ذكية عندما يتم تزويدها بمعالج قوي ومجموعة حساسات ربما يكون من أهمها هي الكامرة. هذا المشروع هو عبارة عن تصميم وسيطرة على روبوت متنقل عن طريق نظام رؤية فعال وهذا يتضمن توجيه والسيطرة على ملاحه هذا الروبوت. ويمكن استخدامه في دخول بيئات حيث يكون دخول الانسان صعب او خطر لأي سبب من الأسباب.