UNIVERSITY OF MORATUWA

Faculty of Engineering

Registered Module No: EN3992

INDUSTRIAL TRAINING REPORT

LE Robotics (Pvt.) Ltd.

From: 04/ 01 / 2022 To: 20 / 06 / 2022

Date of Submission:
20 / 06 / 2022

Thalagala B.P.

180631J

Department of: Electronic and Telecommunication Engineering

Preface

This report was composed as partial fulfillment of the requirements of Module EN3992 - Industrial Training, in the curriculum of B.Sc. in Engineering (Electronic and Telecommunication) at the University of Moratuwa, Sri Lanka. The experience and knowledge that I gained during the six months of my industrial training were used and were the inspiration to create this report.

Acknowledgment

I would like to gratefully acknowledge all of the people who helped me to make this six months of special industrial training period a massive success, starting from the point of applying to a company to the point I left the training organization at the completion of six months.

First of all, I would like to express my heartfelt gratitude to Professor Kapila Jayasinghe who was our supervisor for us throughout the six months of the internship period. The advice he provided us with regard to professional engineering practices and ethics was invaluable. In addition to that the directions he provided us to gain the required technical skills required for the allocated project were priceless. I believe the mindset that he built in us, towards working under minimum supervision, will be a massive support for us to thrive in the fast-moving industry.

Next, I would like to express my gratitude toward Miss. Laknie Jayasinghe who was the Engineer in charge of us in our training period. The support she provided us to improve our soft skills as well as technical skills as a professional engineers, is highly appreciated. The support she provided when composing the technical documentation of my allocated task by pointing out the mistakes and the areas of improvement was invaluable and must be mentioned at this point. Moreover, her support in validating the project deliveries at the end of the training period is highly appreciated.

Last but not least, I would like to express my heartiest gratitude to Mr Janka Kulathunga, the manager of the Lanka Electronics research and development section and Mr Aloka Perera, an electronic engineer in LE Robotics (Pvt.) Ltd for being available for us whenever we needed their support.

Contents

PDF is clickable

List of Abbreviations

CAN Controller Area Network

CPLD Complex Programmable Logic Device

CV Computer Vision

FOV Field of View

FPGA Field Programmable Gate Arrays

GUI Graphical User Interface

IDE Integrated Development Environment

IPC Inter-process communication

LAN Local Area Network

ML Machine Learning

OSS Open-Source Software

PC Personal Computer

PCB Printed Circuits Boards

R&D Research and Development

ROI Region of Interest

RPi Raspberry Pi

SCARA Selective Compliance Articulated Robot Arm

SDLC Software Development Life Cycle

SE Software Engineering

SFTP SSH(Secure shell) File Transfer Protocol

SIFT Scale Invariant Feature Transform

SVM Support Vector Machine

UART Universal Asynchronous Receiver/Transmitter

List of Figures

List of Tables

Chapter 1

Description of the organization and business, its past, present and future

For my special industrial training, I got the opportunity to work as an engineering trainee at LE Robotics (Pvt.) Ltd. located at 100/4, Divulapitiya Road, Minuwangoda, Sri Lanka.It is a local Research and Development (R&D) facility where they work on industrial articulated robot arms and related technologies. This chapter provides an extensive description of the organization and business as well as information about its past, present and future.

1.1 Organization and Business

1.1.1 Introduction

LE robotics (Pvt.) Ltd. is a local R&D facility that has been established by Prof. J.A.K.S. Jayasinghe who is a senior professor in the Department of Electronic and Telecommunication Engineering at the University of Moratuwa in Sri Lanka.

LE robotics (Pvt.) Ltd. is the first in the Sri Lankan market to offer fully customisable robotics solutions 'Made in Sri Lanka' for various automation needs for an affordable price with expertise based in Sri Lanka. In the year 2005, the company designed and manufactured the first custom robotics solution in their affiliated company, Lanka Electronics (Pvt.) Ltd. Since then, they have been developing various robotics solutions and related technologies in the facility.

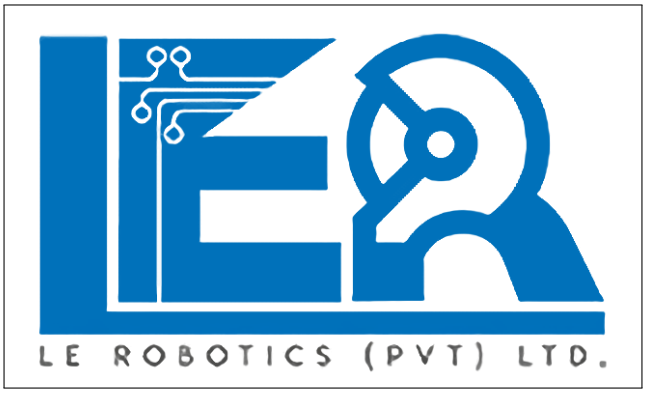


Figure 1.1: Logo of the LE Robotics (Pvt.) Ltd.

1.1.2 Services and Products

As mentioned previously LE Robotics Pvt. Ltd. provides services and tailor-made products for various automation needs of its customers. When it comes to the services they provide, the following key services can be highlighted.

1. Process Analysis to identify the automation needs and challenges of organizations
2. Product customization to provide the best tailor-made solution for requirements
3. Providing local expertise with ease of access
4. Providing lifetime support for the products

In addition to the services they provide, the following products are manufactured at LE Robotics Pvt. Ltd. R&D of the related technologies of those products is a key activity among the day-to-day activities in the facility.

1. **6 DOF Robots** - Robots with six degrees of freedom (Figure 1.2 depicts such a model)
    - Robotics solutions with the capability to mimic human arm operations
    - Offers the flexibility to provide you with fully customized robotic movements to suit your requirement
    - Around 1 m reach and 0.1 mm placement precision

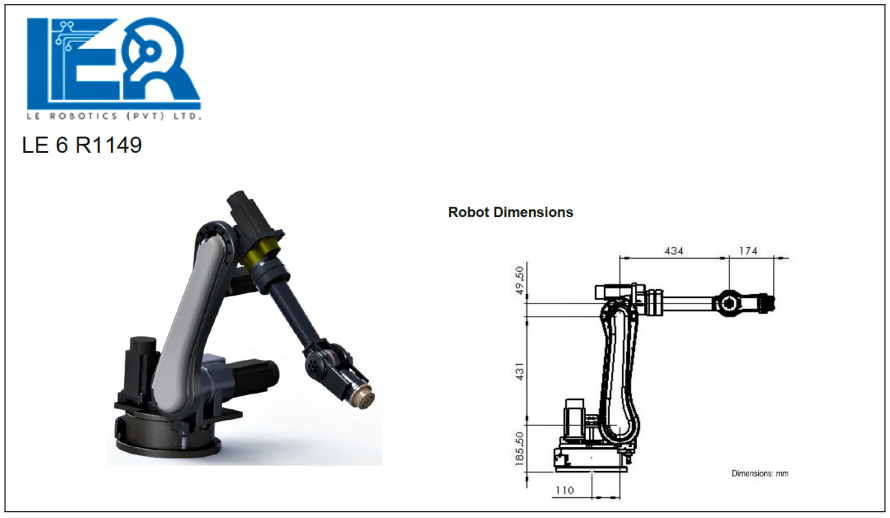


Figure 1.2: Model *LE 6 R1149* 6-DOF articulated robot arm[2]

2. **4 DOF Robots** - Robots with four degrees of freedom (Figure 1.3 depicts such a model)
    - Robotics solutions with superior high-speed performance, high rigidity and high accuracy
    - Array of options with a compact design
    - Around 500 mm reach and 0.1 mm placement precision

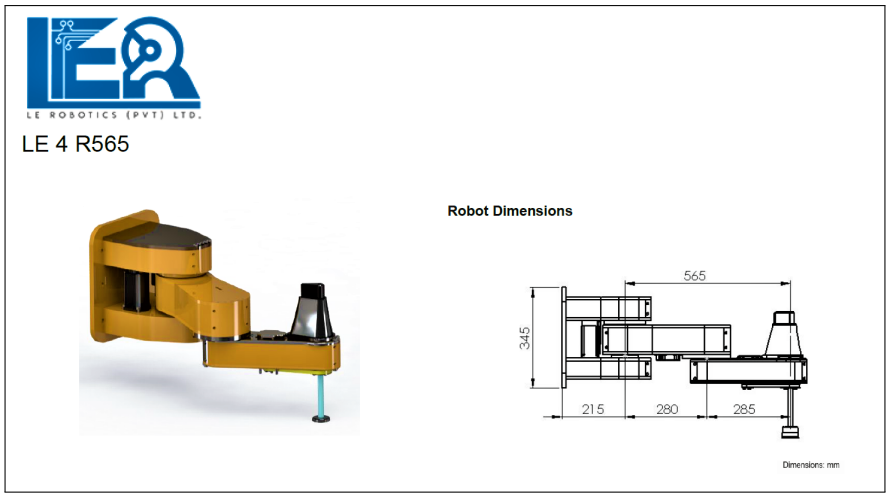


Figure 1.3: Model *LE 4 R565* 4-DOF SCARA robot arm[3]

3. **Custom Made Robots** - Robots designed according to the customer's needs
    - Reach and Payload specifications are possible to be customized as per your process requirements
    - Robotics Solutions to carry out simple pick and place operations

Chapter 2

Description of familiarization work carried out

Although the official internship period commenced on Thursday 4^{th} of January in 2022, we had had our first meeting with the supervisor about a month prior to that. In that meeting, our supervisor introduced the available projects that we would be working on in our training period. He offered us two separate paths we can select to work during our internship period as described in Table 2.1. Depending on our interests, we chose 'Plan A' to continue with.

TABLE 2.1
PLANS WE WERE OFFERED PRIOR TO BEGINNING OUR INDUSTRIAL TRAINING

Plan A	**Plan B**
Motion planning, Path planning (welding, drilling)	Advanced servo drivers - Controller Area Network (CAN) (Ethernet to CAN bus interface + Drivers)
CV, Perception (picking objects from a moving conveyor)	Advanced servo drive - Ethercat (Ethercat servo drive Printed Circuits Boards (PCB) + Firmware)
Multiprocessor robot controllers (modify the present design using Field Programmable Gate Arrays (FPGA)/Complex Programmable Logic Device (CPLD) pulse generator such that precision welding and real-time picking is possible)	Industrial Personal Computer (PC) based robot controllers (Developed Inter-process communication (IPC) software incorporating CAN/Ethercat servo drivers)

Chapter 3

Exposure to systems (HSE, Financial, Administration, Logistics, etc.)

3.1 Health, Safety and environment (HSE) Department

Health, safety and environment (HSE) refers to a branch, or department, within a company that is responsible for the observance and protection of occupational health and safety rules and regulations along with environmental protection.

3.2 Financial Department

Chapter 4

Project Work

By title, the project that I was assigned, was ***Machine vision based Real-time Motion Planning for an Industrial Articulated Robot Arm***. In simple words, it was a project related to an automatic pick and place machine which can be used to pick objects placed on a conveyor belt and pass them to the next stage of processing.

My contribution to that project was to develop the following three aspects of the system.

I. Development of an object detection framework

II. Development of an application for camera calibration (referred as *Camera Calibrator GUI* in this document)

III. Development of an application to train an object classification model (referred as *Classifier Trainer GUI* in this document)

Subsequent sections explain the mentioned sub-projects that were undertaken by me during the industrial training period. Implementation details are not exposed in this report as it does not comply with professional engineering ethics.

4.1 Development of an Object Detection Framework

The first project, that I was assigned as a trainee electronic engineer was related to the CV field. In this project, an Object Detection Framework was developed to be deployed in an Automatic Pick and Place Machine. In a summary, the framework is capable of identifying Region of Interest (ROI), detecting and classifying objects, and determining the location and orientation of objects with respect to a real-world coordinate system for grasping (picking). This section explains everything related to the development of that framework.

4.1.1 Problem Definition

Most of the robotic arms used in industrial environments operate in a pre-programmed cycle. When comes to the way a human does the same task is much different as the path planning for picking an object may change from cycle to cycle because of the perception obtained through human vision. CV is the technology and methods incorporated to mimic the human vision in order to gain insights into the operating environment of the robotics system.

When it comes to real-time object detection using CV, there is an inevitable trade-off between the accuracy and the speed of the operation. This depends entirely on the used CV algorithms and the computational power of the available hardware. If the robotic system/ arm in interest is not controlled through a dedicated industrial PC with adequate computational resources, the amount of resources that can be allocated to the CV unit becomes limited. This will eventually result in great delays (which is not desirable when it comes to real-time operations) to produce the required outputs by processing the acquired images through the associated camera.

Due to the limitations in the resources, the initial problem of real-time object detection on a moving conveyor was subdivided into three problems.

- **Case 1:** Detection of objects placed in a ***static environment*** (no conveyor belt is associated). Picking and placing is carried out by the robot in the usual way.
- **Case 2:** Detection of objects placed on a ***non-continuous conveyor belt***. The conveyor belt can be paused when objects are detected and picking and placing is carried out thereafter.
- **Case 3:** Detection of objects placed on a ***continuously moving conveyor belt***. Picking and placing happen while the conveyor is moving.

4.1.2 Solution

As the initial stage of the project, case 1 mentioned in Section 4.1.1 was developed. It consisted of developing a CV subsystem for detecting ***stationary objects*** placed in a ***static environment***. In this case, no conveyor belt is used and objects are placed in the Field of View (FOV) of the camera.

Figure 4.1 shows the interconnection between inputs and outputs of the developed system. Once the video stream from a camera is fed into the system, it is capable of,

1. Determining the grasping location of an object in a real-world coordinate system. The centroid of the 2D view of the object was considered as the grasping location.

2. Determining the gripper's angle. Orientation of object with respect to the positive x direction of the same real-world coordinate system was used for this.

3. Determining the class/ type of the detected objects using a Scale Invariant Feature Transform (SIFT) and Support Vector Machine (SVM) based classification algorithm.

and providing them to the main controller of the pick and place machine as requested by it.

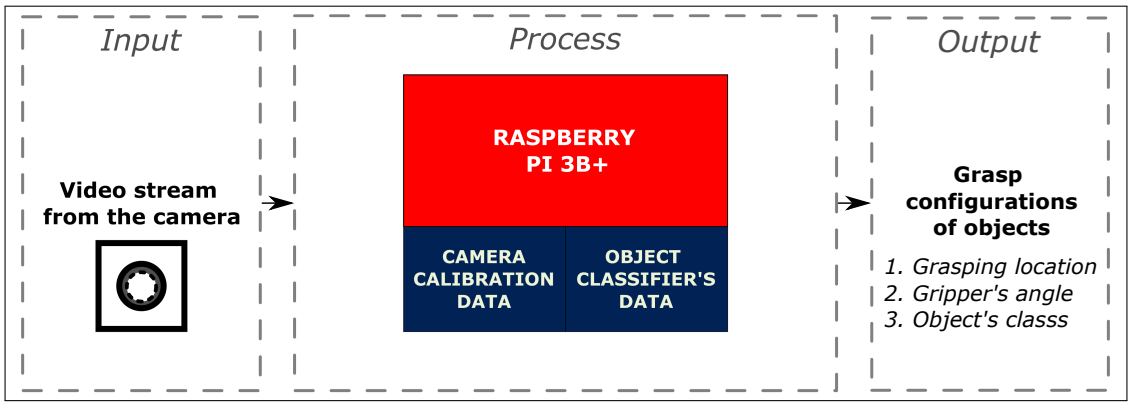


Figure 4.1: Interconnection between inputs and outputs of the CV subsystem

Figure 4.2 depicts the interconnections between the existing robot controller and the newly developed computer vision subsystem. Outputs shown in Figure 4.1 are provided to the robot controller through a serial communication link built using an Universal Asynchronous Receiver/Transmitter (UART) circuitry. For transferring data to the Raspberry Pi from the working PC, SSH(Secure shell) File Transfer Protocol (SFTP) over Wi-Fi technology was used.

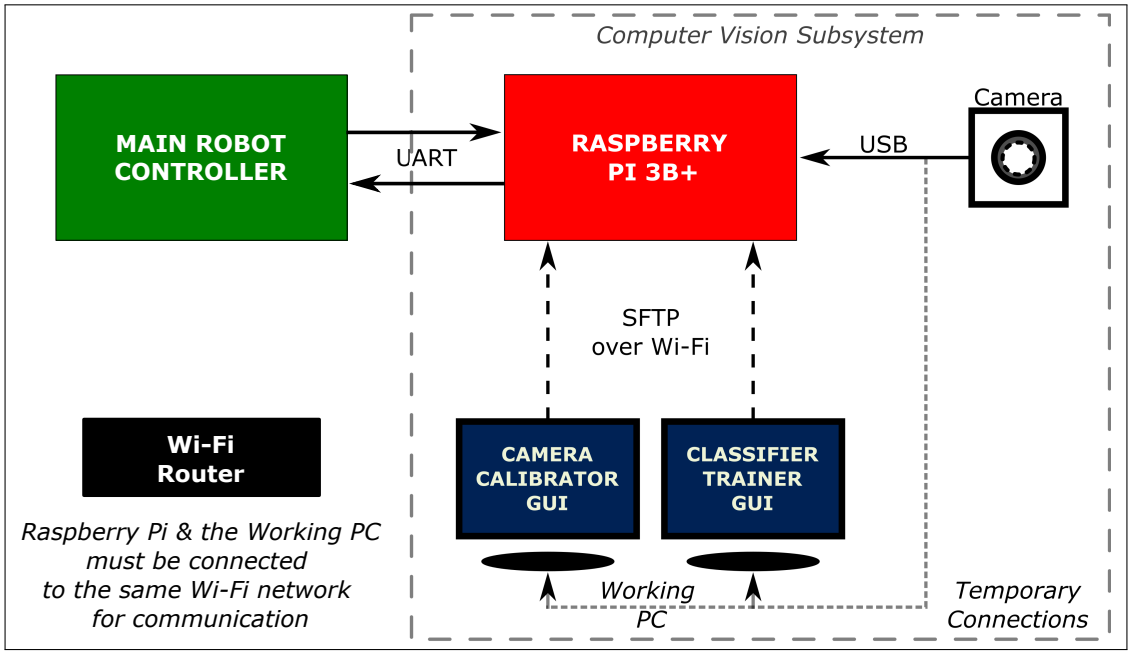


Figure 4.2: Interconnections between the existing robot controller and the CV subsystem

For the implementation of the prototype, third-party tools given below are used.

1. **Raspberry Pi (RPi) 3 B+ Single Board Computer (SBC)**
   Product brief: Online available at https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf.

2. **Logitech C310 HD Webcam, 720p Video**
   Technical specifications: Online available at https://support.logi.com/hc/en-us/articles/360023464573-Logitech-HD-Webcam-C310-Technical-Specifications.

3. **OpenCV open source computer vision library stable release 4.4.0**[6]
   Stable and development releases: Online available at https://github.com/opencv/opencv; A pre-compiled version of this library, which is optimized for CV and Deep Learning on Raspberry Pi was used.

4. **Gordon's Arduino wiring-like WiringPi Library for the Raspberry Pi**[8]
   Unofficial Mirror for WiringPi bindings: Online available at https://github.com/WiringPi/WiringPi.

During the operation of the pick and place robot, the developed object detection algorithm runs only inside the RPi. However, for the operation of that algorithm, it requires the data shown in Table 4.1. A Windows GUI was developed to generate those data and the 'Description' column of Table 4.1 provides only a simple overview of that data . Subsequent sections explain the functionality and the working principles of the mentioned GUI, in detail.

TABLE 4.1
DATA FILES REQUIRED TO RUN THE OBJECT DETECTION FRAMEWORK

File Name	Description
calibration_data.xml	This file includes camera calibration data. That is, intrinsic and extrinsic parameters of the associated camera.
objects_data.xml	This file includes object classification model's data. That is, the data required for the object classification (class names, trained support vector machines and etc.).

4.2 Development of an Application for Camera Calibration

The second project, that I was assigned as a trainee electronic engineer was related to the Software Engineering (SE) field. In this project, a Windows GUI was developed to calibrate a given camera. In a summary, the application is capable of generating the required data, to remove the distortions of captured images and to transform 2D image points back to a given 3D real-world coordinate system with an accuracy of $\pm 0.5\ mm$.

4.2.1 Problem Definition

The end goal of the CV subsystem mentioned in the Section 4.1.2 is to provide the location and orientation of the identified objects, with respect to a known real-world coordinate system. However, when an image is captured, the details about the depth of the object relative to the camera are lost due to the real world to image plane transformation (the forward projection [7]) happens inside the camera. This operation is non-invertible.

There are several methods used in CV literature to back-project image plane coordinates to the real-world coordinate system. In this project, a method which uses a single calibrated camera and analytic geometry was implemented. It was observed that the back-projected coordinates are accurate at least to $\pm 0.5\ mm$. As mentioned for this back-projection process, a *calibrated camera* is required and the rest is engineering mathematics. Therefore, the actual problem to be solved was to calibrate the camera associated with the pick and place machine.

4.2.2 Solution

Camera Calibration refers to the extraction of various parameters related to the camera that will be used to capture the video stream/ images. These parameters are required when the back-projection is done. For the camera calibration, the method that uses an asymmetric chess board was used[4]. Because that method was really simple and could be implemented easily using the resources that were available in the facility.

The important input data needed for calibration of the camera is a set of 3D real world points and the corresponding 2D coordinates of these points in the image. 3D points are called object points and 2D image points are called image points.

I. **2D Image Points:** These points can be easily found from the images. These image points are the locations where two black squares touch each other in chess board. Functions in OpenCV is used to extract those points[4].

II. **Object Points:** For simplicity, the chess board was kept stationary at XY plane, (so Z=0 always) and camera was moved accordingly. This consideration helps us to find only X,Y values. Now for X,Y values, we can simply pass the points as (0,0,0), (1,0,0), (2,0,0), ... which denotes the location of points. In this case, the results we get will be in the scale of size of chess board square[4]. But if we know the square size, (for our implementation square size of the chess board was 30 mm), we can pass the values as (0,0,0), (30,0,0), (60,0,0).

The process of camera calibration is given below. Once the camera calibration is done, we will be able to extract the intrinsic and extrinsic parameters of a given camera. Those parameters

can then be used for back-projection which was my problem at hand.

1. Keep the chess board stationary in the XY plane of the real world coordinate system as shown in the Figure 4.3 and do not let it move due to any reason.

Figure 4.3: Real world coordinate system and the position of the camera with respect to it.

2. Capture about 10-15 images (higher the better) as shown in the Figure 4.4 using the camera that will be used for object detection, in several distances and orientations.

   The actual number of required images is around 6. However, in some scenarios OpenCV is unable to detect the chess board pattern properly in every image. Two of such reasons are given below.

   - **Only a part of the chessboard is visible in the captured image**: For the program to work, whole chessboard must be visible in the captured image.
   - **No proper lighting conditions**: if the environment that the chessboard is placed, does not have proper lighting conditions application will not work as expected. Therefore make sure that the chessboard pattern receives enough light.

   Hence, it's better to capture images more than the required amount so that we can filter out damaged/ useless images later.

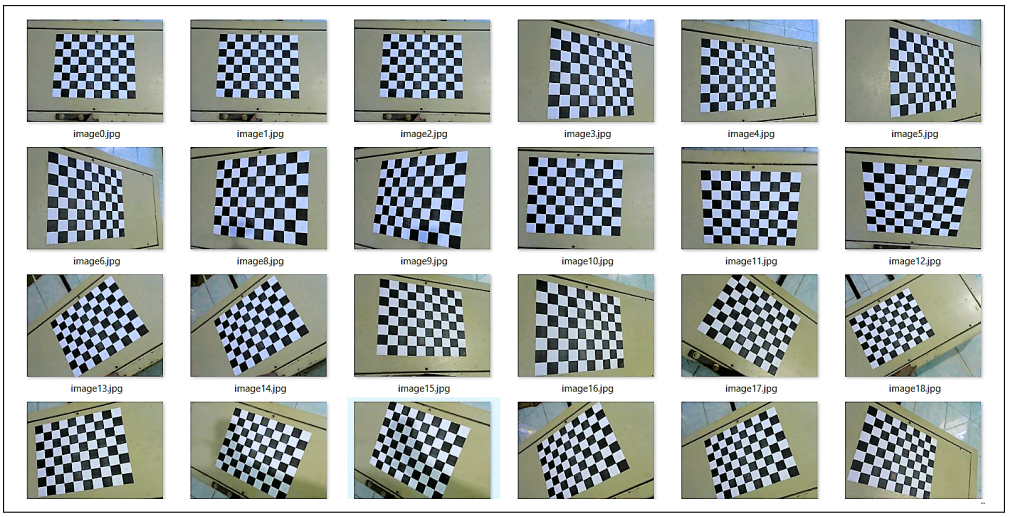
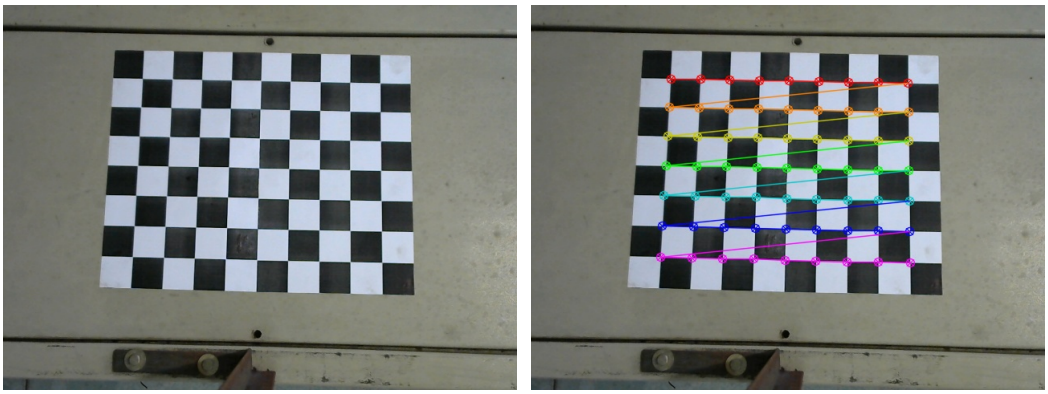


Figure 4.4: Raw images captured by placing the camera in various distance and orientations

3. Capture one additional special image, placing the camera in the place that it will be mounted when the actual object detection algorithm runs. Extrinsic parameters[7] corresponding to this image will be directly used in our Back-projection algorithm.

   Extrinsic parameters contains a translation vector and a rotation matrix. These two together brings the calibration pattern from the object coordinate space (in which object points are specified) to the camera coordinate space. In more technical terms, the rotation matrix and translation vector performs a change of basis from object coordinate space to camera coordinate space. Due to its duality, it is equivalent to the position of the calibration pattern with respect to the camera coordinate space [5].



(a) Raw image (b) Processed image to identify the pattern

Figure 4.5: Captured special image to extract the extrinsic parameters that are useful for back-projection

4. Use all of the captured images to calibrate the camera. The necessary documentation of the used functions are explained in [5]. After carrying out those functions you will be

able to extract the extrinsic and intrinsic parameters related to the special image and the camera respectively.

Extracted intrinsic parameters which are specific to the used camera, do not depend on the image. Among those parameters, *Camera Matrix* is what has the greatest important. *Radial/ Tangential Distortions* have a minimal importance and therefore can be neglected. Extracted extrinsic parameters of the image shown in Figure 4.5 are specific for that image. If in any case, orientation or the place of the camera is altered, these must be recalculated in order to be used in the back-projection algorithm.

As mentioned previously ultimate goal of this sub project was to develop a GUI application for camera calibration. The process mentioned above was therefore implemented in C# programming language using '*Visual Studio 2019*' software. For the implementation, the *Emgu CV* library which is a cross platform *.Net* wrapper to the OpenCV image processing library was also used. It allows OpenCV functions to be called from .NET compatible languages like C# and therefore make the software development process efficient. The developed GUI is shown in the Figure 4.6.

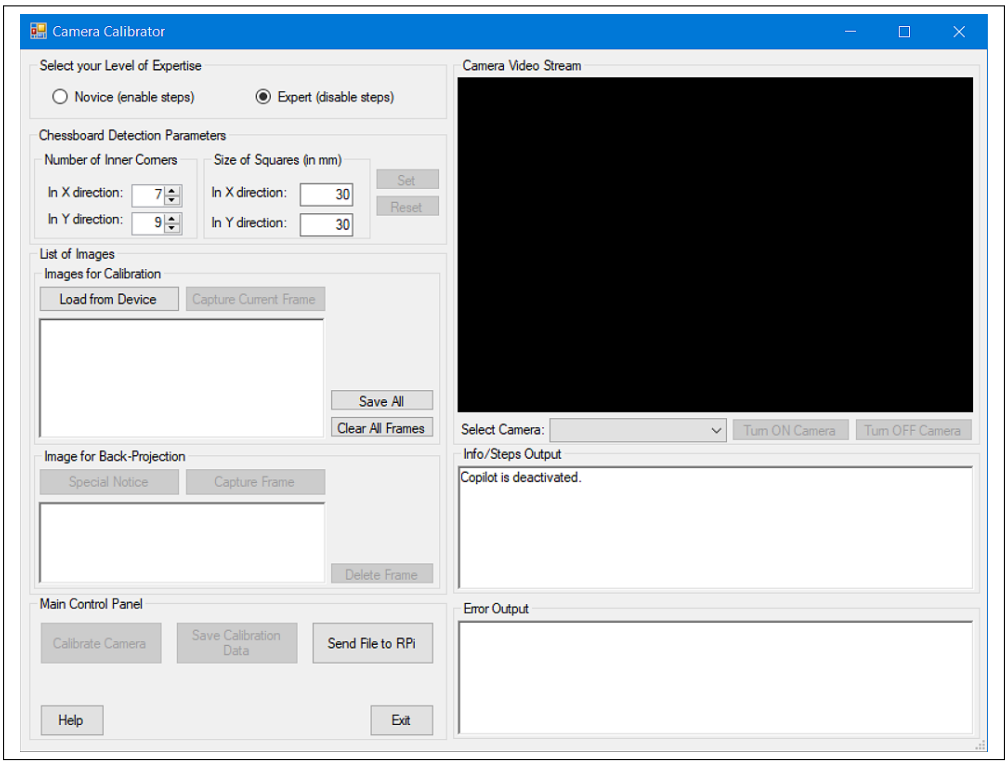


Figure 4.6: Camera calibrator GUI for Windows

Once the camera calibration is done, the software can also be used to send the file that includes the necessary data to the RPi. For that the RPi and the working PC where the software is run, must be connected to the same Local Area Network (LAN) over WiFi.

4.3 Development of an Application to Train an Object Classification Model

The third project, that I was assigned as a trainee electronic engineer was also related to the SE field. In this project, a Windows GUI was developed to train a Machine Learning (ML) model which can be used classify set of pre-defined objects. In a summary, the application is capable of generating the required data, to classify identified objects by the object detection framework explained in the Section 4.1. Figure 4.7 shows how the object detection framework classifies objects with the help of object classification model.

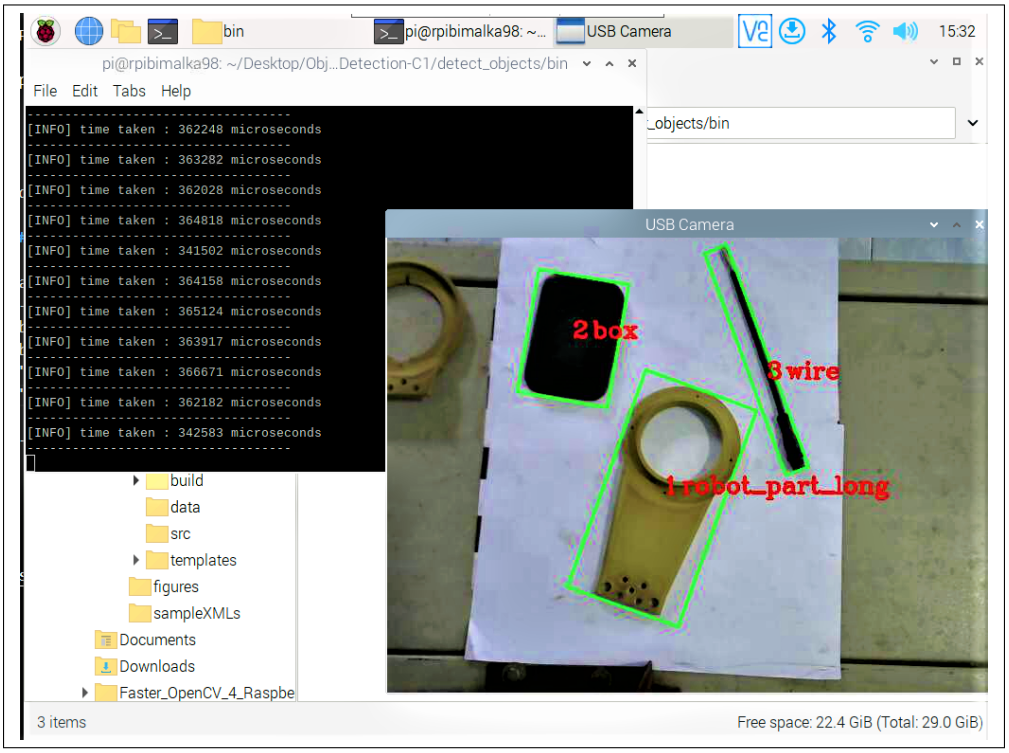


Figure 4.7: Output of the object detection framework with object classification capability

4.3.1 Problem Definition

One of the key aspects of an object detection framework is to classify the detected objects. Because, in order to effectively make use of the information of the detected objects, it is necessary to know the type/class of them. In CV literature there are various methods to determine the class of a detected object. Accuracy as well as the speed of the predictions was a major concern when selecting a method, as the end product was supposed to use in an industrial environment. By considering all of such aspects it was decided to use an ML model which is based on SIFT features and SVMs.

4.3.2 Solution

Prior to deploy any ML tool for predictions in any system, they must be trained for the target task by feeding it the necessary data. In literature this task is known as *training an ML model*. Similarly, in our case the object classification model must be trained prior to couple it

with the object detection framework explained in the Section 4.1. This training phase of the classification model is carried out using a Windows GUI application developed using Emgu CV (a cross platform .Net wrapper to the OpenCV image processing library) package[1]. Figure 4.8 depicts the layout of the designed Windows GUI.

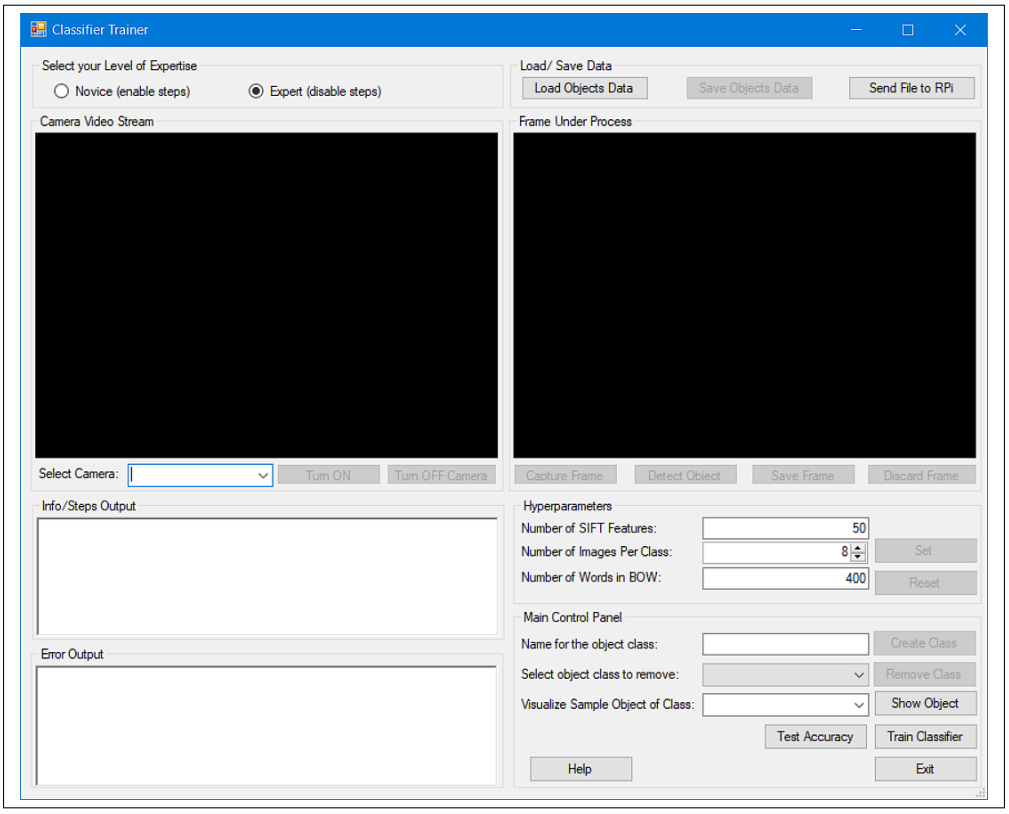


Figure 4.8: Classifier trainer GUI for Windows

Once the training is done, the software can also be used to test the accuracy of the prediction model to fine tune its performance. Ultimately, the file that includes the necessary data can also be sent to the RPi using the same software. For that the RPi and the working PC where the software is run, must be connected to the same LAN over WiFi.

When the model is trained on a set of object classes, we can keep using the model for classification as long as we do not introduce new objects classes to our object detection framework.In addition to that, SVMs were trained using the one-vs-all approach. Therefore, each object class has its own SVM which decides whether an object belongs to that class or not (a binary classification problem). Therefore, the calculated histogram of features, of a given ROI (an object) is passed through all of SVMs when deciding the class of an object. This improves the accuracy of predictions but sacrificing the time. Therefore, time it takes to run a single object detection routine will depend on two factors.

1. Number of non overlapping objects placed in the FOV of the camera: *Higher the number of objects, longer the time object detection takes.*

2. Number of classes of objects: *Lesser the number of classes shorter the time object detection takes.*

Chapter 5

Hands-on experiences

The entire six months period of my industrial training was full of hands-on experiences. It was an ideal combination of learning, unlearning and relearning. LE robotics Pvt. Ltd. had no experts in the CV field during the time I was working there. Therefore, I had to learn most of the things related to my assigned projects by actually doing them. This made an ideal opportunity for me to learn various technologies really fast and with minimum supervision.

5.1 Resources for Self-Learning

As I made a lot of design decisions on my own while doing the project, I had to extensively refer to various knowledge resources throughout my training period. This taught me the right skill set to identify reliable information sources among a lot of garbage on the internet. Following is a list of information sources that I used during my training.

- Google Search
- Stack Overflow
- OpenCV Documentations
- EmguCV Documentations

5.1.1 Google Search

Website: https://www.google.com/
Without any doubt, the starting point of exploring any information on internet was the *Google Search*. It is the search engine provided by Google. The power of modern google search is pretty amazing and it can quickly adapt to your search patterns. This lets it bring the most relevant information to the user. Therefore, using the correct terms during a search matters and the quality of retrieved information extensively depends on that.

5.1.2 Stack Overflow

Website: https://stackoverflow.com/
Stack Overflow is a question and answer website for professional and enthusiast programmers. It features questions and answers on a wide range of topics in computer programming[11]. If

you are not doing something really new that no one has ever done, there's a high probability of finding an answer to any question which you may come across. Stack Overflow is that rich of various solutions to fit exactly to your problem at hand.

Additionally, the number of *Scores* an answer has obtained for a particular question is an ideal measure to estimate the the reliability of the answer. Moreover, answers can be easily linked in your documentations and source codes using their associated *Share* links. Figure 5.1 depicts an answer to a question asked on the Stack Overflow platform. The actual answer is available at https://stackoverflow.com/a/10230489/15939357.

Figure 5.1: A sample answer on Stack Overflow and its associated shareable link

5.1.3 OpenCV Documentations

Website: https://docs.opencv.org/4.x/
OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms. It also includes high quality documentation as well, for each module of the library. These documentation contains theoretical concepts and a vast amount of example code snippets. Such code snippets can be used to understand the way of using a particular function in a given application.

5.1.4 EmguCV Documentations

Website: https://www.emgu.com/wiki/index.php/Documentation
EmguCV is a cross platform .Net wrapper for the OpenCV image-processing library. It allows OpenCV functions to be called from .NET compatible programming languages like C#. The wrapper can run on Windows, Android, iOS, Mac OS and Linux. EmguCV also includes a rich documentation which explains usage of various functions and their interfaces. This documentation comes handy when it comes to Windows GUI development in C# using Visual Studio software.

5.2 Usage of Open Source Software

5.2.1 Introduction

Open-Source Software (OSS) is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose[10]. The purpose may be either commercial or private use. Most of the time such OSS are developed in a collaborative manner through online platforms such as *GitHub (https://github.com/)* and general public has the access to examine the source codes. Additionally, any potential developer can contribute to such open source technologies and number of contributors has no limit.

5.2.2 Advantages

During our internship, we were encouraged to use open source technologies as much as possible inside our project implementations. Because it has a lot of advantages when it comes to R&D field. Few of them are listed below.

- Most of the open source software are free of charge and there is no need of purchasing a license to use the source codes.
- Since OSS are developed in collaborative manner through publicly available online platforms, software are highly optimized due to the diverse perspectives of their developers.
- There are well established community driven online platforms to sort out different problems that may encounter during project development. Most of the time answers to such problems are readily available.
- Due to the indefinite number of contributors, there is a rich set of documentations related to the software. This makes the production really efficient as there is no need of blindly experimenting with the software.

The EmguCV docs that were introduced in the Section 5.1.4 contain much description about functions in its library. However, EmguCV docs does not contain code examples. This was a challenge that I faced during the development of the software which were explained in the Sections 4.2 and 4.3. This is where the power of open source technologies can be used. EmguCV's source codes are hosted and developed at https://github.com/emgucv/emgucv. Therefore, they can be accessed by anyone without any charge and can be used to understand the usage of available functions in various applications. Situation is the same for OpenCV and its source codes are hosted and developed at https://github.com/opencv/opencv.

In addition to that, each such repository contains a section named as *Issues* with various modifications requests and questions raised by the community. This issues section can be a life saver when working with latest versions of the software, as most of the problems that may encounter when using the latest versions may not be answered anywhere else on the internet. Figure 5.2 depicts the issues section of the OpenCV project repository.

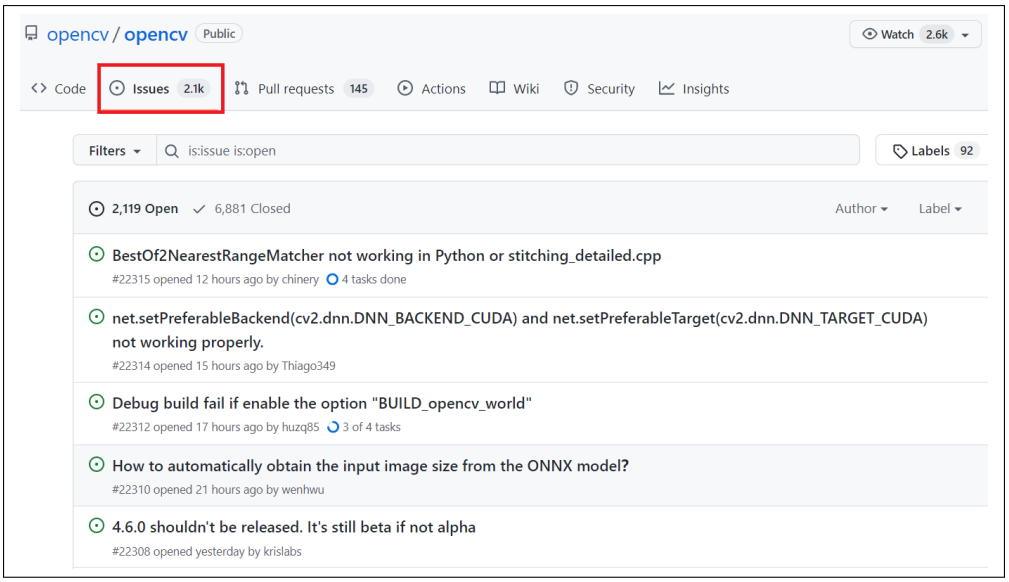


Figure 5.2: Issues section of the OpenCV project repository

5.3 Usage of Modern Tools

As described in the Section 4, throughout my internship period I had to work with technologies related to CV and SE. Therefore, I gained a lot experience of using various tools related to Software Development Life Cycle (SDLC). The tools I used can be classified into two categories depending on the area I used them. That classification is shown in the Table 5.1.

TABLE 5.1
MODERN TOOLS USED DURING THE DEVELOPMENT OF THE SOFTWARE

Windows GUI Development	**Object Detection Framework Development**
Visual Studio 2019 Git	Visual Studio Code Git CMake

5.3.1 Visual Studio 2019

Visual Studio 2019 is simply an Integrated Development Environment (IDE) owns and developed by Microsoft. It provides necessary facilities for computer programmers to develop software. It consists of a source code editor, build automation tools, debugger and many more features that an IDE should contain [9].

Visual Studio uses various Microsoft software development platforms to develop software. In my project I used what is known as *Windows Forms* platform. It is a free and open-source GUI class library included as a part of Microsoft .NET Framework, providing a platform to write client applications for desktop, laptop, and tablet PCs. An application developed using the Windows Forms platform is called as a *Windows Forms Application*. These applications are event-driven applications which means they spends most of their time simply waiting for

the user to do something, such as fill in a text box or click a button[12]. The code for the applications developed by me were written in C# language.

5.3.2 Visual Studio Code

5.3.3 Git

5.3.4 CMake

Chapter 6

Soft Skills Development

Chapter 7

SWOT Analysis of the organization and self

Chapter 8

Conclusion: Own perspective of areas to be improved (of the whole training process including self)

Appendices

Appendix A

Guidelines

References

[1] Emgu CV: a cross platform .Net wrapper to the OpenCV. https://www.emgu.com/wiki/index.php/Version_History#Emgu.CV-4.5.1.

[2] LE Robotics - Articulated Robots. http://www.lerobotics.lk/articulated-robots.php. publisher: LE Robotics Pvt. Ltd.

[3] LE Robotics - SCARA Robots. http://www.lerobotics.lk/scara-robots.php. publisher: LE Robotics Pvt. Ltd.

[4] OpenCV: Camera Calibration. https://docs.opencv.org/4.4.0/dc/dbb/tutorial_py_calibration.html.

[5] OpenCV: Camera Calibration and 3D Reconstruction. https://docs.opencv.org/4.4.0/d9/d0c/group__calib3d.html.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[8] G Henderson. Wiring Pi: GPIO Interface library for the Raspberry Pi. http://wiringpi.com/.

[9] Wikipedia contributors. Microsoft visual studio — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&oldid=1098834529, 2022. [Online; accessed 28-July-2022].

[10] Wikipedia contributors. Open-source software — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Open-source_software&oldid=1096922686, 2022. [Online; accessed 28-July-2022].

[11] Wikipedia contributors. Stack overflow — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Stack_Overflow&oldid=1100396097, 2022. [Online; accessed 27-July-2022].

[12] Wikipedia contributors. Windows forms — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Windows_Forms&oldid=1094827942, 2022. [Online; accessed 28-July-2022].