Design and Construction of Robotic Palletizer

Guided by a machine vision system

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

Author: Akram GHEDIRI

Supervisor: Prof. Abdelbaki DJOUAMBI

June 2017
Dedication

First of all, all Praise be to Allah Almighty for his blessing, for the courage and the patience he gave to me to accomplish this humbled work.

To my Mom & Dad
Thank you for a lifetime of unfailing love, caring and sacrifices. I am very proud of you. I would not be the man I am today without Allah and your guidance. I cannot repay you for all that you have done for me, I love you.

To my beloved brothers: Djihad & Ayoub
To my best buddies in the world: Chouaib, Tareq & Mouaad
To my dearest friends: Amar, Baamour, Sami, Walid, Imed & Naoui

I owe you my deepest gratitude, for being there for me when I needed you, you made me a better person, a better brother and better friend. Thank you for being a part of my life, I love you.
Acknowledgements

I owe my deepest thanks to my supervisor prof. Abdelbaki DJOUAMBI for his leadership and guidance, he provided me with a deep understanding for the various aspects of the subject.

I also thank my institution and my faculty members without whom this project would have been a distant reality.
Abstract

The aim of this work, is to design and construct a robotic palletizer, guided by a machine vision system, which makes the robot capable of recognizing parts based on color and shape, and sorting them through pick-and-place application. This project combines the knowledge of mechanics, image processing, electronics and programming. The project is mainly oriented towards educational purposes, which is totally open source to help both students and amateurs to understand the basic concepts of robotics.

**Keywords:** Robotics, Palletizer, Pick-and-Place, Machine Vision, Shape Recognition, Color Recognition
MOLAKHES
Résumé

Le but de ce travail est de concevoir et de construire un palettiseur robotique, guidé par un système de vision qui permet au robot de reconnaître les pièces en fonction de la couleur et de la forme, et de les trier par une application pick-and-place. Ce projet combine la connaissance de la mécanique, du traitement d’image, de l’électronique et de la programmation. Le projet est principalement orienté vers des fins éducatives, qui est totalement open source pour aider les étudiants et les amateurs à comprendre les concepts de base de la robotique.

Mots clés: Robotique, palettiseur, Pick-and-Place, Vision machine, Reconnaissance de la forme, Reconnaissance des couleurs
## Contents

Dedication ii  
Acknowledgements iii  
Abstract iv  
MOLAKHES v  
Résumé vi  
List of Tables ix  
List of Figures x  

1 INTRODUCTION 1  
1.1 Historical Background .............................................. 2  
1.2 Review of Robotics .................................................. 3  
  1.2.1 Robot manipulator .................................................. 3  
  1.2.2 Positioning and orientation ........................................ 5  
  1.2.3 Kinematics ............................................................ 5  
  1.2.4 Dynamics .............................................................. 6  
  1.2.5 Path and trajectory ................................................ 6  
  1.2.6 Motion types ......................................................... 6  
  1.2.7 Robot Programming Languages ................................... 7  
  1.2.8 Robotics and Machine Vision ..................................... 7  
1.3 Objective ............................................................... 8  
1.4 Overview of Thesis ................................................... 8  

2 MACHINE VISION 9  
2.1 Introduction ............................................................ 9  
2.2 Machine Vision System Components .................................. 10  
2.3 Image Acquisition .................................................... 11  
2.4 Digital Image Processing ............................................. 11  
  2.4.1 Image Preprocessing ................................................ 12  
  2.4.2 Image Segmentation ............................................... 14  
  2.4.3 Feature Extraction ................................................ 18  
  2.4.4 Object Classification ............................................. 20  
2.5 Summary ................................................................... 23  

3 MECHANICAL MODELING AND ANALYSIS 24  
3.1 Introduction ............................................................. 24  
3.2 Kinematic Analysis .................................................... 25
4 SOFTWARE AND HARDWARE IMPLEMENTATION

4.1 Introduction ................................................. 35
4.2 Robot Components ......................................... 36
   4.2.1 Main Body ........................................... 36
   4.2.2 Controller ............................................ 36
## List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Median Filter Example</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Extracted Measurements</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Corners Coordinates</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Shape Factors</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>Shape Classification Results</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>DH parameters</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Rotation Range Constraints</td>
<td>32</td>
</tr>
<tr>
<td>3.3</td>
<td>Links Lengths and Offsets</td>
<td>32</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Different Classes of Manipulators ............................................. 3
2.1 Machine Vision System Components .......................................... 10
2.2 Simulation Image ...................................................................... 11
2.3 Saturation Adjustment Results .................................................. 12
2.4 Noise Filtering Results .............................................................. 13
2.5 Edge Enhancement Results ........................................................ 14
2.6 Color Filtering Results .............................................................. 15
2.7 RGB to Grayscale Conversion Results ........................................ 16
2.8 Histograms of Converted Grayscale Images ............................... 17
2.9 Binary Thresholding Results ...................................................... 17
2.10 Cropped Images Of The Obtained Objects .................................. 18
2.11 Corners Detection Results ......................................................... 19
2.12 Square Part ............................................................................ 21
2.13 Equilateral Triangle Part .......................................................... 21
2.14 Final Result ............................................................................ 22
3.1 Definition of standard DH link parameters ................................ 25
3.2 Mechanical Structure ................................................................. 26
3.3 Kinematic Configuration of Palletizer Robot ............................... 27
3.4 Work Envelope ......................................................................... 33
Chapter 1

INTRODUCTION

In the last twenty years, robot technology has developed very quickly which is a comprehensive technology integrating mechanical technology, automatic control technology, computer technology, artificial intelligence, microelectronics, optical, communication technology, sensor technology, bionics science, etc. It represents the frontier of high-tech development. It is a popular direction of the current scientific and technological research. Palletizing technology is a new technology in the field of logistics automation technology.

Palletizing means the adoption of the idea of integration unit to stack the materials according to certain models to realize logistic activities such as storage, handing, loading, unloading and transportation of the materials units. In recent years, with the expansion of production scale and improvement in the level of mechanical automation, stacking technology, especially palletizing robot technology, is more and more extensively applied and is leading the trend due to its advantages in the mechanical structure, scope of application, the device footprint, flexibility, cost and the maintenance. So far, palletizing robot is widely used in the pharmaceutical industry, petrochemical industry, food, household appliances as well as agriculture and many other areas. In this chapter, we will talk about the different industrial robots classes, as well as what we need to build a palletizing robot, with a machine vision system, that guides the palletizer to perform part sorting, based on shape and color.
1.1 Historical Background

George Charles Devol, often called the father of robotics, invented the first industrial robot, the Unimate, in 1954. A few years later, Devol and entrepreneur Joseph F. Engelberger were discussing interested at a part and their company, Unimation, was born.

The first prototype, Unimate, was produced in 1961 and installed in GM’s factory for die casting handling and spot welding. It cost $65,000 to produce yet was sold for $18,000. After that, GM installed 66 more Unimates and Ford became interested as well. The industrial robot future was certain to be bright with all of the automotive interest and investment.

Modern industrial robot arms continued to evolve in the 1960’s and 70’s around the globe. The competition from companies around the world continued to produce a high demand for industrial robots. This spurred further research and technical development and items such as the development of the microprocessor helped to create cost-effective control systems that were still powerful.

In 1963, the six-jointed Rancho Arm was created to assist handicapped. This was followed by the tentacle arm, designed by Marvin Minsky in 1968. It was able to lift a person and had 12 joints.

The first successful story of a business developing a specific robot based on their needs was created in 1967. This company developed a robot to complete a spray painting application and eventually became ABB. This is only one example of when large companies began to develop their own industrial robots.

So, industrial robotic development continued to quickly evolve, and in 1969 the Standford Arm eventually led to commercial arm production. The Stanford Arm was one of the first electronically powered, computer-controlled arms. By 1974, it reached a level of sophistication where it could assemble a Model T water pump.

The Stanford Arm was followed by the Silver Arm in 1974. The Silver Arm was created by MIT’s David Silver to perform precise assembly using touch and pressure sensors and a microcomputer. During that same year, these arms led to Victor Scheinman, the inventor of the Stanford Arm, to form Vicarm, Inc. to manufacture industrial robotic arms. Scheinman was instrumental in the creation of the PUMA (programmable universal manipulator for assembly) for Unimation.

By the middle of the 1970s, industrial robots had boomed and were expected to grow at rates around 30%

In the 1980’s, automotive companies showered robotic companies with investments. The enthusiasm and funding were not always matched with understanding. General Motors Corporation spent more than $40 billion on new technology in the 1980’s, but a lack of understanding led to costly robot fiascos. In 1988, robots at the Hamtramck Michigan plant wreaked havoc - smashing windows and painting one another. Unfortunately, the premature introduction of robotics began to create financial instability.

It wasn’t until recently that the robotics industry has regained mid-1980 revenue levels. The American robotics market disappeared as Japanese and European bought up companies.

2010 brought a huge acceleration in demand due to the continued innovative development and improvement of industrial robots. By 2014, there was a 29% increase in robot sales across the globe.

It is an exciting time to be a part of the robotic world as our globe becomes more aware of the amazing benefits of industrial robots.
1.2 Review of Robotics

1.2.1 Robot manipulator

The earliest robots for industrial applications were developed in the 1960s. The technology has, from then, greatly progressed and can now be said to have reached mature levels. Through the automation of processes the technology is not only trying to replace human operators in the execution of operations but also in the intelligent processing of information. Automation is the synthesis of mechanical technology used in the industry and computer technology allowing information management. The industrial robot is, due to its programmability and flexibility, an essential component in the programmable automated systems growing larger within the industry.

By ISO 8373, an industrial robot is defined as a re-programmable, automatically controlled, multipurpose manipulator programmable in three or more axes. A robot manipulator is classified as a robot with a fixed base (compared to a mobile robot with a mobile base) and consists of a number of rigid bodies referred to as links. These links are connected to one and another by joints that are either prismatic or revolute. Prismatic joints creates a relative translational motion between two links where revolute joints creates a relative rotational motion. The displacement is referred to as joint offset and joint angle respectively.

Manipulators can be classified as Cartesian, polar, cylindrical, spherical, SCARA or jointed-arm depending on the type and sequence of the degrees of freedom (DOFs).

![Different Classes of Manipulators](image)

Figure 1.1: Different Classes of Manipulators

For a manipulator to perform an arbitrarily positioning and orientation of an object in a three-dimensional space six DOF are required, three to position the object and three to orient the object. The links in a manipulator can either form an open or closed kinematic chain. In an open kinematic chain there is only one sequence of links connecting the two end points compared to a closed kinematic chain where some sequence of links form a loop. In an open kinematic chain each joint provides the robot with one DOF.
The use of industrial robots in the manufacturing process industry can be divided into three major categories:

- material handling
- manipulation
- measurement

Material handling refers to all the moments in a manufacturing process where material and parts are moved from one place to another for storage, manufacturing, assembly and package. A robot is ideal for pick and place operations such as:

- palletizing
- part sorting
- loading and unloading
- packing...

In a similar fashion manipulation is referring to all the moments during the manufacturing process where parts and material are altered, added and finally turned into a finished product. Typical manipulation applications where the robot can perform such tasks are:

- painting and coating
- arc and spot welding
- gluing and sealing
- square, wiring and fastening...

Finally, within the process of manufacturing there are several moments where the parts and material are examined to ensure correct manufacturing and product quality. These kind of measurements can be performed by manipulators and some examples of applications are:

- object inspection
- operation supervision
- contour finding...
1.2.2 Positioning and orientation

To be able to manipulate and handle objects it is necessary to describe the position and orientation of both the object and the robots end-effector in space. Assuming there exists a universal coordinate system to which everything can be referred to, and an object is provided a coordinate system or a frame, the position and orientation of the object can now be described with respect to the universal coordinate system. It might be desirable to describe the position and orientation of the object with respect to some other reference system and since any other frame also is described with respect to the universal coordinate system it is possible to recalculate the position and orientation of the object with respect to the new reference frame. This is called transforming, mapping or changing the description of the object.

The position and orientation of a robot’s end-effector is usually referred to as the pose, and can either be described in Cartesian space or in joint space. In Cartesian space the pose of the end-effector is described by its position in the global x-, y, and z-coordinates as well as its rotation around the global x-, y- and z-axis. In joint space the pose of the end-effector is described by the angles on each of the joints in the robot. The angles of each joint in the robot also determine the posture of the robot.

1.2.3 Kinematics

For an industrial robot the relationship between the joints position and the end-effector pose is described through its kinematics. The kinematics of a robot refers to all the geometrical and time-based properties of a motion, such as position, velocity, acceleration and all higher order derivatives of position, while ignoring the forces causing it. The relationship between the joint motions and the motions of the end-effector is described through the Jacobian matrix and referred to as the robots differential kinematics. The kinematic relationships can be divided into two categories, the direct, or forward, kinematics and the inverse kinematic.

The direct kinematics describes the end-effectors pose as a function of the joint angles, the posture of the robot, and is also known as transforming from joint space description to Cartesian space description and is relatively straightforward. Describing the end-effectors tool frame relative the base frame is an example of a direct kinematic problem.

The inverse kinematic problem consist of computing the joint angles given the end-effectors pose. This is usually more complicated and might not provide a unique solution as a robot can reach the same end-effector pose through different joint angles and postures. Figure 2.3 shows one example of two different robot postures giving the end-effector the same position and orientation. The inverse kinematic problem might also not have a solution and the robots work envelope is defined by the existence or nonexistence of an inverse kinematic solution. If no solution exists the given end-point position lies outside of the robot’s work envelope.
1.2.4 Dynamics

A dynamic model of a robot describes the relationship between a motion, position, velocity and acceleration, and the forces required to cause the motion. Parameters such as mass of the robot, the inertia of different components, the geometry and gear characteristics (gear torque) as well as external forces such as gravitation takes into consideration and determines the dynamic performance of the robot [21]. The dynamic model is of great importance when designing a control system. In a similar way as the kinematics, there is a direct and inverse problem with the dynamic model. The inverse dynamic calculates the force and torques in each joint related to the position, velocity and acceleration while the direct dynamics calculate the motion with respect to internal or external forces and the torques acting on it [21]. In a direct model calculation the current joint position, velocity and torques along with active forces are entered in to the model to output the resulting joint acceleration.

1.2.5 Path and trajectory

The terms path and trajectory are commonly incorrectly used as synonyms. Since both are being used frequently throughout the thesis the difference between the terms are to be explained to avoid confusion. A path is a geometric description of motion and only denotes points in space, Cartesian or joint, that the end-effector has to move through when executing the assigned task. In each point the pose of the end-effector and the posture of the robot is defined. A trajectory is the path motion and defines the velocity and acceleration of each joint in each point of the path. The path can be seen as the road the car has to follow and the trajectory defines how the car is traveling along that road, where it slows down or speeds up.

1.2.6 Motion types

Both path and trajectory are crucial when programming robots. In principle the description and constraints of a path along with the dynamic constraints can be seen as the inputs to the trajectory planning algorithm, while the outputs are a time sequence for the end-effectors position, velocity and acceleration. The function of a trajectory planning algorithm is to create a smooth trajectory defining the motion of the robot without violation of any constraints. In an operators point of view, a trajectory is created by defining one or more motions after each other. The motions can be of different type but the starting point of one motion is always the end point of the previous motion. Depending on the desired application of a robot its trajectory is programmed differently. Some general motion types are:

- Point-to-point (PTP)
- Linear motion (LIN)
- Circular motion (CIRC)
- Spline motions (SPLINE)
1.2.7 Robot Programming Languages

Almost every robot manufacturer has developed their own proprietary robot programming language, which has been one of the problems in industrial robotics. ABB has its RAPID programming language. Kuka has KRL (Kuka Robot Language). Comau uses PDL2, Yaskawa uses INFORM and Kawasaki uses AS. Then, Fanuc robots use Karel, Stäubli robots use VAL3 and Universal Robots use URScript.

In recent years, programming options like ROS Industrial have started to provide more standardized options for programmers.

1.2.8 Robotics and Machine Vision

A machine vision system is employed in a robot for recognizing the objects. It is commonly used to perform the inspection functions in which the industrial robots are not involved. It is usually mounted in a high speed production line for accepting or rejecting the work parts. The rejected work parts will be removed by other mechanical apparatuses that are in contact with the machine vision system.

A machine vision system can be incorporated with an industrial robot for performing the following three important tasks such as:

a) Inspection

The industrial robots are only used to support the machine vision system when it performs the inspection tasks. During this process, it checks for accurate surface finish, exact dimension, errors in labeling, presence of holes in the work parts, and other factors. The machine vision system carries out this inspection processes automatically with less time and errors. In addition, the human workers can also perform these operations manually, but there is a high possibility of error occurrence and increased operation time.

b) Identification

In this process, the machine vision system performs recognizing and categorizing of work parts instead of inspecting it. It also helps in determining the work part’s position and orientation. Some of the operations accomplished by a machine vision system in the identification process are work part palletizing and depalletizing, object sorting, and gripping the parts oriented from a conveyor. A robot is used in these tasks to take successive action and decision.

c) Visual Servoing and Navigation

In this application, a machine vision system controls the actions of a robot according to the visual input. For example: In robot visual servoing process, a machine vision system directs the path of robot’s end effector to a work part in the work cell. Some applications of this category consist of positioning of work parts, seam tracking, bin picking, and retrieving and re-orienting the work parts that are moving along a conveyor. With the help of visual data, the navigational control can be used in collision protection and automatic path planning of a robot.
1.3 Objective

In every project, there must be a reason why it was conducted. Objective defined how successful the project has been, it gives the benefits to organize the efforts towards accomplishing the desired goal. Thus, The objective of this thesis project focusses on how to build a palletizer robot which is capable of sorting parts through the guidance of a machine vision system. The project will be oriented for educational purposes to help both students and amateurs to understand basic robotics.

This project scopes are:

1. To design a machine vision system for color and shape recognition
2. To analyze the mechanical structure of the robot
3. To build the control circuit and program the robot

1.4 Overview of Thesis

Each chapter in this thesis is initiated with a brief introduction of the content, and ended with a short summary concluding the main contributions. Following this introduction to the subject and description of robotics in general, objectives and scopes, is Chapter 2, giving the main steps and algorithms to build an object classification computer software based on shape and color recognition. Chapter 3 describes the mechanical structure of the palletizer as well as kinematic and velocity analysis, and Chapter 4 is about the implementation of software and hardware to build a real world palletizer, ending the thesis with a conclusion dedicated the discussion of encountered difficulties in this project as well as proposals for future work.
Chapter 2

MACHINE VISION

2.1 Introduction

Traditionally, machine vision refers to the use of both computer vision and image processing in an industrial, practical application or process. Based on the image analysis done by the vision system, it is necessary to execute certain function or outcome. Based also on the findings, the system can be used to trigger a variety of set "actions".

Vision systems are widely used in industry, mainly for inspection and quality control processes. Their use has been increased in applications related to improving the safety of workers in the industrial environment and for robot guidance. Robots need machine vision to move around the working space avoiding obstacles, to work collaboratively with humans, to identify and locate the working parts, to improve their positioning accuracy, etc. Depending on the objective, the vision system can be scene-related or object-related. In scene-related tasks the camera is usually mounted on a mobile robot and applied for mapping, localization and obstacle detection. In object-related tasks, the camera is usually attached to the end-effector of the robot manipulator or above a conveyor belt, so that new images can be acquired by changing the point of view of the camera.

In this chapter, we will illustrate the different steps and algorithms, to build an object classification computer software based on shape and color recognition, and extracting some proprieties such as location and rotation of the parts. The software will be used to guide a palletizing robot to execute pick-and-place application.
2.2 Machine Vision System Components

The main goal of a machine vision system, is to automate complex or mundane visual inspection tasks and precisely guide handling equipment during product assembly. Applications include Positioning, Identification, Verification, Measurement, and Flaw Detection.

A machine vision system typically consists of digital cameras and back-end image processing hardware and software. The camera at the front end captures images from the environment or from a focused object and then sends them to the processing system. Depending on the design or need of the MVS, the captured images are either stored or processed accordingly. The vision system is composed of two main processing components, executed one after the other:

1. Image Acquisition
2. Digital Image Processing

![Figure 2.1: Machine Vision System Components](image)

A common output from automatic inspection systems is pass/fail decisions. These decisions may in turn trigger mechanisms that reject failed items or sound an alarm. Other common outputs include object position and orientation information from robot guidance systems. Additionally, output types include numerical measurement data, data read from codes and characters, displays of the process or results, stored images, alarms from automated space monitoring machine vision systems, and process control signals.
2.3 Image Acquisition

The first stage of any vision system is the image acquisition stage. Image acquisition can be defined as the action of retrieving an image from capturing sources usually, a hardware-based source. So, it can be passed through any processes needed to occur afterward. The first step in a machine vision sequence is always performing image acquisition in image processing because without an image no processing is possible. The image that is acquired is completely unprocessed. Therefore, the application of some image processing techniques is necessary in order to create an understandable image for the machine.

For that reason, USB Web camera is used as capturing device which outputs an 24-bits RGB image with a size of 640 by 480 pixels, and a resolution of 96 DPI.

2.4 Digital Image Processing

Digital image processing is always an interesting field as it gives improved pictorial information for human interpretation and processing of image data for storage, transmission, and representation for machine perception. Image Processing is a technique ( . . . ) to enhance raw images received from various capturing devices such as cameras and sensors, and/or extracting features and measurements for object detection and classification applications. The field of image processing significantly improved in recent times and extended to various fields of science and technology. The principle advantage of Digital Image Processing methods is its versatility, repeatability and the preservation of original data precision. The various Image Processing stages are:

1. Image preprocessing
2. Image segmentation
3. Feature extraction
4. Object classification

We used AForge.NET framework for building the vision software. AForge.NET is an open source C# framework designed for developers and researchers in the fields of computer vision, image processing, artificial intelligence, neural networks, genetic algorithms, fuzzy logic, machine learning, robotics, etc. For the sake of demonstration, we used a simulation image to perform processing on it.

Figure 2.2: Simulation Image
2.4.1 Image Preprocessing

In image preprocessing, image data captured by cameras or sensors can be accompanied by errors related to various reasons. These errors such as brightness level, contrast, and different noises, are corrected using appropriate mathematical models. Image enhancement is the modification of image by changing its pixels values to improve its visual impact, or to convert the image to a form which is better suited for human or machine interpretation. The used techniques in this section are:

a) Saturation adjustment
b) Noise reduction
c) Edge enhancement

a) Saturation Adjustment

Color saturation refers to the intensity of color in an image. The term hue refers to the color of the image itself while, saturation describes the intensity (purity) of that hue. When color is fully saturated, it is considered in purest version. Primary colors red, green and blue are considered purest version color as they are fully saturated. As the saturation increases, colors appear to be more pure or vivid and vice-versa.

So, in order to increase color classification efficiency, we increase the color’s purity by adjusting color saturation, this could be done with Aforge.NET predefined class “SaturationCorrection”.

Figure 2.3 shows result of saturation adjustment on the simulation image with +60% gain.
b) Noise Reduction

Images taken with both digital cameras and conventional film cameras will pick up noise from a variety of sources. Further use of these images will often require that the noise be (partially) removed, noise reduction is a process used to remove various types of noises from the images, for this job we need to pick a noise filter. There are several filters to choose from, however, due to available noise filters in Aforge.NET, we are limited with 3 filters which are convolution, mean, and median. Considering testing results obtained after applying previous filters on a test image, the median filter does "better" job of reducing noise and preserving useful details in the image.

Median filter consist of replacing each pixel of the original source image with the median of N\times N neighboring pixel values including the concerned pixel. The median is calculated by first sorting all the neighborhood pixel values into numerical order and then replacing the pixel being considered with the middle pixel value.

Example: supposing we have the following 8-bits gray-scale image that we want to apply filtering on pixel P(3, 3) with 3x3 median.

<table>
<thead>
<tr>
<th>Raw Image</th>
<th>Filtering Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 125 126 130 140</td>
<td>123 125 126 130 140</td>
</tr>
<tr>
<td>122 124 126 127 135</td>
<td>122 124 126 127 135</td>
</tr>
<tr>
<td>118 120 150 125 134</td>
<td>118 120 124 125 134</td>
</tr>
<tr>
<td>119 115 119 123 133</td>
<td>119 115 119 123 133</td>
</tr>
<tr>
<td>111 116 110 120 130</td>
<td>111 116 110 120 130</td>
</tr>
</tbody>
</table>

Neighbourhood values: 115, 119, 120, 123, 124, 125, 126, 127, 150

median value: 124

Using predefined Aforge.NET class “Median”, we could apply median filter on 24-bits RGB images, as figure 2.4 shows results of noise filtering on the image obtained after saturation adjustment with 3x3 neighboring pixels.

Figure 2.4: Noise Filtring Results
c) Edge Enhancement

Edge enhancement is an image processing filter that enhances the edge contrast of an image or video in an attempt to improve its acutance (apparent sharpness).

The filter works by identifying sharp edge boundaries in the image, such as the edge between a subject and a background of a contrasting color, and increasing the image contrast in the area immediately around the edge. This has the effect of creating subtle bright and dark highlights on either side of any edges in the image, called overshoot and undershoot, leading the edge to look more defined when viewed from a typical viewing distance.

The reason behind the need to enhance edges, is that applying a noise filter is basically removing high frequencies from image, since noise generally is a high frequency signal, however, eliminating high frequencies also leads to losing some of useful informations such as edges, which makes the image somehow looks blurry, and that can effect processing results. There is several algorithms to enhance edges, however there is only one edge sharping class available on Aforge.NET which is “GaussianSharpen” class.

Figure 2.5 shows the impact of edge enhancement on the obtained noise filtered image from previous phase.

![Edge Enhancement Results](image)

Figure 2.5: Edge Enhancement Results

2.4.2 Image Segmentation

Image segmentation is the process that subdivides an image into its constituent regions (or their contours) corresponding to objects. We usually try to segment regions by identifying common properties that connected pixels share. Or, similarly, we identify contours by identifying differences between regions (edges). The level to which the subdivision is carried out depends on the problem being solved, so the segmentation should stop when the objects of interest in an application have been isolated from the rest of the image.

The reason why segmentation is essential, is that we need to isolate every potential part individually for features extraction stage. For achieving the goal, there are two techniques that we need to apply:

a) Color filtering

b) Binary thresholding
Note: Aforge.NET framework considers by default that regions are 8-connectivity pixels.

a) Color Filtering

As a part of object classification in this application requires color recognition, its necessary to decompose the image to its constituent colors, by subdividing the image to regions that share the same, or similar colors within a range. Due to simplicity reasons, we chose 3 colors to be recognized, which are the basic colors Red, Green and Blue. The idea here is to eliminate all regions that do not match with the colors of interest, and keeping the ones that match, by applying color filtering.

The concept of a color filter, is keeping pixel value (color) if it belongs to a specific color range, otherwise, putting pixel value to zero (black). The reason why we chose to perform color filtering with a color range, not a single value such as mean color, is to prevent problems caused by illumination conditions.

If \( g(x, y) \) is the output of color filtering of \( f(x, y) \), then

\[
g(x, y) = \begin{cases} 
    f(x, y) & \text{if } f(x, y) \in \text{Range}_{\text{color}} \\
    0 & \text{otherwise}
\end{cases} \tag{2.1}
\]

However, some background pixels can have values that are included in the colors ranges, and be considered as regions of interest. The solution here is limiting the regions size, so every region has a length and width lower than certain limit (in our case we took the limit as 64x64 pixels) will not be considered as a region. By using Aforge.NET classes ”YCbCrFiltering” and ”Blob”, we can perform color filtering with region size limitation, and outputting 24-bits RGB images as the figure 2.6 shows.

![Color Filtering Results](image)

(a) Preprocessing Resulted Image
(b) Red Regions
(c) Green Regions
(d) Blue Regions

Figure 2.6: Color Filtering Results
b) Binary Thresholding

The simplest property that pixels in a region can share is intensity. So, a natural way to segment such regions is through thresholding, the separation of light and dark regions. Thresholding creates binary images from Grayscale ones by turning all pixels below some threshold to zero (black) and all pixels about that threshold to one (white). If \( g(x, y) \) is a thresholded version of \( f(x, y) \) at some global threshold \( T \), then

\[
g(x, y) = \begin{cases} 
1 & \text{if} \ f(x, y) \geq T \\
0 & \text{otherwise}
\end{cases}
\]  

(2.2)

The output images obtained from color filtering are 24-bits RGB, so we need to convert them to Grayscale images. This could be done with "Grayscale" class, as the figure 2.7 shows.

![Figure 2.7: RGB to Grayscale Conversion Results](image)
Now, after obtaining Grayscale images, we need to set threshold value. A common method for determining threshold value is Histogram analyzing. Figure 2.8 shows Histograms of the obtained Grayscale images.

![Histograms of Converted Grayscale Images](image)

By observing figure 2.8, we can clearly see a gab between the background (black) and the potential parts (shades of gray). The threshold could be a value within the interval $[1, 70]$, however, due to illumination factor, this is not always the case. After testing different values of threshold with different illuminations conditions, value $T = 10$ seems enough for our application. Binary thresholding can be applied with Aforge.NET class "Threshold", as figure 2.9 shows the obtained results.

![Binary Thresholding Results](image)
2.4.3 Feature Extraction

The features extraction stage is very important. This technique extracts high-level features needed in order to perform classification of objects. Features are those items which uniquely describe an object, such as size, location, corners etc. Segmentation techniques are used to isolate the desired object from the scene so that measurements can be made on it subsequently. Quantitative measurements of object features allow classification and description of the image.

It is essential to focus on the feature extraction phase as it has an observable impact on the efficiency of the recognition system. There are four main features that we need to extract for the application, which are:

a) Area
b) Perimeter
c) Center of Gravity (Location)
d) Corners Coordinates

a) Area

By region area we mean, the count of pixels that constitute the region. We can extract region area by calling the "Area()" function contained in Aforge.NET class "Blob".

b) Perimeter

Region perimeter refer to, the count of pixels that surrounds the region (region contour pixels). We can extract region perimeter by calling the "Perimeter()" function contained in Aforge.NET class "Blob".

c) Center of Gravity

Region center of gravity is, coordinates of the pixel that is situated in the center of gravity of the region, which is calculated as mean value of X and Y coordinates of each pixel contained in the region. We can extract center of gravity by calling the "CenterOfGravity()" function contained in Aforge.NET class "Blob".

The extracted measurements so far, are listed in table 2.2.

![Figure 2.10: Cropped Images Of The Obtained Objects](image)

(a) Object (1)  (b) Object (2)  (c) Object (3)
Table 2.2: Extracted Measurements

<table>
<thead>
<tr>
<th>Object (i)</th>
<th>Area</th>
<th>Perimeter</th>
<th>Center of Gravity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object (1)</td>
<td>21408</td>
<td>540</td>
<td>(303, 144)</td>
</tr>
<tr>
<td>Object (2)</td>
<td>11612</td>
<td>493</td>
<td>(454, 305)</td>
</tr>
<tr>
<td>Object (3)</td>
<td>20012</td>
<td>564</td>
<td>(178, 333)</td>
</tr>
</tbody>
</table>

d) Corners Coordinates

A corner can be defined as the intersection of two edges. Corners can be extracted by using two classes, which are "SusanCornersDetector" and "MoravecCornersDetector". However, Susan Corners Detector seems "better" of detecting corners according to testing results for this application.

The obtained results are shown in figure 2.11, along with corners coordinates in table 2.3.

![Figure 2.11: Corners Detection Results](image)

Table 2.3: Corners Coordinates

<table>
<thead>
<tr>
<th>Object (i)</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object (1)</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Object (2)</td>
<td>(498, 228)</td>
<td>(366, 310)</td>
<td>(503, 383)</td>
<td>none</td>
</tr>
<tr>
<td>Object (3)</td>
<td>(119, 409)</td>
<td>(254, 396)</td>
<td>(240, 260)</td>
<td>(101, 275)</td>
</tr>
</tbody>
</table>
2.4.4 Object Classification

As the name suggests, the object classification aims to classify full objects, based on some extracted features and labeling the defined objects. The goal of the application, is the classification of objects based on shape and color, however, for simplicity reasons, we chose to perform classification on three shapes, which are the basic shapes squares, circles and equilateral triangles. The used method for shape classification is based on calculating "Compactness measure" or "shape factor".

Shape factors are calculated from measured dimensions, such as diameter, area, perimeter, centroid, etc, which allow us to define a shape based on its geometric properties. However, in this application, we will use two features to define the shape, which are area and perimeter properties that we extracted from the previous features extraction stage. The shape factor in this case can be defined as following:

\[ C = \frac{1}{4\pi} \left( \frac{(Perimeter)^2}{Area} \right) \]  

(2.3)

The criteria of shape classification is the following:

\[ |C_{Object} - C_{Shape}| \leq error, \text{ then } Object = Shape \]

Note: After multiple tests to define the error value, error = 0.2 seems enough for the classification.

Now, by calculating and comparing shape factors of the objects, to results shown in table 2.4, the table 2.5 shows shape classification results

<table>
<thead>
<tr>
<th>Object (i)</th>
<th>Object (1)</th>
<th>Object (2)</th>
<th>Object (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>[ C = \frac{1}{4\pi} \frac{(4a)^2}{a^2} = \frac{4}{\pi} \approx 1.27 ]</td>
<td>[ C = \frac{1}{4\pi} \frac{(2\pi R)^2}{\pi R^2} = 1 ]</td>
<td>[ C = \frac{1}{4\pi} \frac{(3a)^2}{\sqrt{3}} = \frac{3\sqrt{3}}{\pi} \approx 1.65 ]</td>
</tr>
<tr>
<td>Class</td>
<td>Circle</td>
<td>Eq Triangle</td>
<td>Square</td>
</tr>
</tbody>
</table>

Due to color filtering phase in the segmentation stage, we already classified regions to three color classes: Red, Green and Blue, so by combining color filtering results with shape classification, we get an object classifier based on shape and color.
So far, we classified objects by shape and color, and each part location (center of gravity) has been extracted. The next thing to do, is defining the parts rotation angles, that depends on the corners coordinates and the shape itself. We can understand the procedure of calculating the angles with some examples:

**Example 1:** In this example, we will determine rotation angle of square shaped part.

![Figure 2.12: Square Part](image)

**Step One:** sorting corners by descending Y order, so we should get this order: P4,P3,P1,P2

**Step Two:** calculating the angle between the first two corners after sorting, which are in this case P4\((x_4, y_4)\) and P3\((x_3, y_3)\)

\[
\tan(\alpha) = \frac{y_4 - y_3}{x_4 - x_3} \quad (2.4)
\]

\[
\alpha = \arctan\left(\frac{y_4 - y_3}{x_4 - x_3}\right) \quad (2.5)
\]

\[
\alpha = \arctan\left(\frac{275 - 260}{101 - 240}\right) \approx -6^\circ \quad (2.6)
\]

**Example 2:** In this example, we will determine rotation angle of equilateral triangle shaped part.

![Figure 2.13: Equilateral Triangle Part](image)
**Step One:** Sorting corners by descending Y order, results the following order : P3,P2,P1

**Step Two:** calculating the angle between the first two corners after sorting, which are in this case P3($x_3$, $y_3$) and P2($x_2$, $y_2$)

\[
\tan(\alpha) = \frac{y_3 - y_2}{x_3 - x_2} \quad (2.7)
\]

\[
\alpha = \arctan\left(\frac{y_3 - y_2}{x_3 - x_2}\right) \quad (2.8)
\]

\[
\alpha = \arctan\left(\frac{383 - 310}{503 - 366}\right) \approx 28^\circ \quad (2.9)
\]

Finally, after labeling the parts and defining locations and rotation angles, we get the final result shown in figure 2.14.

![Figure 2.14: Final Result](image)

In order to make locations of parts readable to the robot, we need to convert centers of gravity from pixels to millimeters. The conversion formula is expressed by

\[
\text{output}[\text{mm}] = 25.4 \frac{\text{input}[\text{pixel}]}{\text{resolution}[\text{dpi}]} S \quad (2.10)
\]

Where $S = 1.18$ is the scale factor related to the capturing distance.
2.5 Summary

This chapter has described a number of different steps and algorithms from the image acquisition to part classification. First thing we started with is capturing the input image, followed by image preprocessing phase where the image is enhanced in order to apply segmentation and extracting some measurements, to be used later in classification phase. The result is a machine vision recognition system for guiding a robotic palletizer to perform pick-and-place sorting application.
Chapter 3

MECHANICAL MODELING AND ANALYSIS

3.1 Introduction

Nowadays, industrialized countries with their high labor costs have to rely on production automation to keep their competitive advantage. One of the most flexible and powerful automation technology available today is industrial robotics. Equipped with the right tool, standardized industrial robots can perform numerous production tasks. Since acquisition and programming of an industrial robot are very expensive, the feasibility of using robots in production facilities depends on the efficiency with which the robot can perform its task: the more production steps a robot can performing a given time interval, the higher the production rates, as a consequence the faster the robot can compensate for its initial acquisition and programming costs, and the higher is the competitive advantage it provides to the company.

One of the main tasks performed in factories is palletizing, which refers to the operation of loading or placing an object such as cartons on a pallet or a similar device in a defined pattern. However, this task can be time consuming and expensive, and it can also put unusual stress on workers.

Many factories and plants today have automated their application with a palletizing robot solution of some kind. Robotic palletizing technology increases productivity and profitability while allowing for more flexibility to run products for longer periods of time. The robot is chosen based on requirements such as the speed, product weight, and size of the packages. Various end-of-arm-tooling styles allow flexibility of different types of robot palletization. Bag grippers encompass an item and support it on the bottom, while suction and magnetic grippers typically handle more ridged items and grip them from the top.

The robotic kinematics is essential for describing an end-effector’s position, orientation as well as motion of all the joints, while dynamics modeling is crucial for analyzing and synthesizing the dynamic behavior of robot. In this chapter, we will illustrate kinematic and dynamic analysis, of 4DOF parallelogram hybrid mechanism palletizing robot.
### 3.2 Kinematic Analysis

The robotic kinematics studies the motion of a robot mechanism regardless of forces and torque that cause it. It allows to compute the position and orientation of robot manipulator’s end-effector relative to the base of the manipulator as a function of the joint variables. Robotic kinematics is fundamental for designing and controlling a robot system. In order to deal with the complex geometry of a robot manipulator, the properly chosen coordinate frames are fixed to various parts of the mechanism and then we can formulate the relationships between these frames. The manipulator kinematics mainly studies how the locations of these frames change as the robot joints move.

Kinematics focuses on position, velocity, acceleration, and an accurate kinematics model must be established in order to investigate the motion of a robot manipulator. Denavit–Hartenberg (DH) notations are widely used to describe the kinematic model of a robot. The DH formation for describing serial-link robot mechanism geometry has been established as a principal method for a roboticist. Standard DH notations are used to create a kinematics model of robot.

In DH convention notation system, each link can be represented by two parameters, namely the link length $a_i$ and the link twist angle $\alpha_i$. The link twist angle $\alpha_i$ indicates the axis twist angle of two adjacent joints $i$ and $i-1$. Joints are also described by two parameters, namely the link offset $d_i$, which indicates the distance from a link to next link along the axis of joint $i$, and the joint revolute angle $\theta_i$, which is the rotation of one link with respect to the next about the joint axis [4]. Usually, three of these four parameters are fixed while one variable is called joint variable. For a revolute joint, the joint variable is parameter $\theta_i$, while for a prismatic joint, it will be $d_i$. The DH convention is illustrated in figure 3.1.

![Figure 3.1: Definition of standard DH link parameters](image-url)
The four parameters of each link can be specified using DH notation method. With these parameters, link homogeneous transform matrix which transforms link coordinate frame $i - 1$ to frame $i$ is given by

$$T_{i-1}^{i} = Rot(z, \theta_{i}) Trans(z, d_{i}) Trans(x, a_{i}) Rot(x, \alpha_{i}) \quad (3.1)$$

$$T_{i-1}^{i} = \begin{bmatrix} c_{i} & -s_{i} & 0 & 0 \\ s_{i} & c_{i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_{i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & a_{i} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \lambda_{i} & -u_{i} & 0 \\ 0 & u_{i} & \lambda_{i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

$$T_{i-1}^{i} = \begin{bmatrix} c_{i} & -s_{i} & \lambda_{i} & s_{i}u_{i} & a_{i}c_{i} \\ s_{i} & c_{i}\lambda_{i} & -c_{i}u_{i} & a_{i}s_{i} \\ 0 & u_{i} & \lambda_{i} & d_{i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

where $c_{i} = \cos(\theta_{i})$, $s_{i} = \sin(\theta_{i})$, $\lambda_{i} = \cos(\alpha_{i})$, $u_{i} = \sin(\alpha_{i})$

### 3.2.1 Mechanical Structure

The concerned robotic palletizer is a 4DOF hybrid manipulator arm, with parallelogram mechanism. Its size not only affects the work efficiency and kinematic accuracy of the robot, but also determines the size of the robot’s workspace. The significant propriety about this structure, is that the end-effector situation is always conserved (parallel to the ground).

![Mechanical Structure](image)

**Figure 3.2: Mechanical Structure**

**Note:** The mechanical structure is a modified version of Ufactory palletizing robot structure, which is an open source robot modeled after the ABB industrial Pallet-Pack robot.
3.2.2 Forward Kinematics

Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters. Transformation matrices relate two frames and can be used to transform positions and orientations from one frame to the other. In robotics, a transformation matrix can be used to calculate the position and orientation of a manipulator’s wrist or end effector given the values of each of the manipulator’s joints. These specific transformation matrices are referred to as the manipulator’s forward kinematics. The transformation from one joint to the next can be computed using the DH parameters.

The first thing to begin with, is defining DH parameters, depending on the kinematic configuration shown in figure 3.4.

![Kinematic Configuration of Palletizer Robot](image)

Table 3.1: DH parameters

<table>
<thead>
<tr>
<th>i</th>
<th>$\theta_i$</th>
<th>$d_i$</th>
<th>$a_i$</th>
<th>$\alpha_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>$d_1$</td>
<td>$l_1$</td>
<td>$90^\circ$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>0</td>
<td>$l_2$</td>
<td>$0^\circ$</td>
</tr>
<tr>
<td>3</td>
<td>$\theta_3$</td>
<td>0</td>
<td>$l_3$</td>
<td>$0^\circ$</td>
</tr>
</tbody>
</table>

Where $\theta_i$ is the the $i$th joint angle and $\alpha_i$ is the $i$th twist angle between axes $i$ and $i + 1$. Moreover $a_i$ is the length of link $i$ and $d_i$ is the offset distance at joint $i$. 

27
\[ T_1^0 = \begin{bmatrix} c_1 & 0 & s_1 & l_1 c_1 \\ s_1 & 0 & -c_1 & l_1 s_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (3.4)

\[ T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (3.5)

\[ T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & l_3 c_3 \\ s_3 & c_3 & 0 & l_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (3.6)

For brevity, let \( c_{ij} = \cos(\theta_i + \theta_j) \), and \( s_{ij} = \sin(\theta_i + \theta_j) \)

\[ T_3^0 = T_1^0 T_2^1 T_3^2 = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 & l_1 c_1 + l_2 c_1 c_2 + l_3 c_1 c_{23} \\ s_1 c_{23} & -s_1 s_{23} & -c_1 & l_1 s_1 + l_2 s_1 c_2 + l_3 s_1 c_{23} \\ c_{23} & 0 & 0 & d_1 + l_2 s_2 + l_3 s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (3.7)

The homogeneous transformation matrix that transform the base frame \( R_0 \), to the end-effector frame \( R_e \), can be expressed by

\[ T_0^e = \begin{bmatrix} c_{14} & -c_{14} & 0 & l_1 c_1 + l_4 c_1 + l_2 c_1 c_2 + l_3 c_1 c_{23} \\ s_{14} & s_{14} & 0 & l_1 s_1 + l_4 s_1 + l_2 s_1 c_2 + l_3 s_1 c_{23} \\ 0 & 0 & 1 & d_1 - d_4 + l_2 s_2 + l_3 s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (3.8)

Note that the kinematic-chain keeps the orientation of \( Z_e \) normal to the planar formed by \( X_0 \) and \( Y_0 \), so the orientation of the last frame, respective to the O-XYZ only changes by rotation angle about \( Z_0 \) axis with \( \theta_1 \), and the last joint angle \( \theta_4 \).

From the obtained homogeneous transformation matrix (3.8), we can extract the end-effector position \((x_e, y_e, z_e)\), as well as the the yaw angle \( \varphi \)

\[
\begin{pmatrix} x_e \\ y_e \\ z_e \\ \varphi \end{pmatrix} = \begin{pmatrix} c_1 (l_1 + l_4 + l_2 c_2 + l_3 c_{23}) \\ s_1 (l_1 + l_4 + l_2 c_2 + l_3 c_{23}) \\ d_1 - d_4 + l_2 s_2 + l_3 s_{23} \\ \theta_1 + \theta_4 \end{pmatrix}
\] (3.9)
### 3.2.3 Inverse Kinematics

In robotics, inverse kinematics makes use of the kinematics equations to determine the joint parameters that provide a desired position of the robot’s end-effectors. An analytic solution to an inverse kinematics problem is a closed-form expression that takes the end-effector pose as input and gives joint positions as output.

Analytical inverse kinematics solvers can be significantly faster than numerical solvers and provide more than one solution for a given end-effector pose. Based on figure 3.3 and equation (3.9) we have

\[
\tan(\theta_1) = \frac{y_e}{x_e} \iff \theta_1 = \arctan\left(\frac{y_e}{x_e}\right) \quad (3.10)
\]

Let \( r = \sqrt{a^2 + b^2} \), where \( a = x_e - l_1 - l_4 \), and \( b = z_e - d_1 + d_4 \). By using cosine rule:

\[
\cos(\theta_2 - \gamma) = \frac{a^2 + b^2 + l_2^2 - l_3^2}{2l_2\sqrt{a^2 + b^2}} \quad (3.11)
\]

\[
\theta_2 - \gamma = \arccos\left(\frac{a^2 + b^2 + l_2^2 - l_3^2}{2l_2\sqrt{a^2 + b^2}}\right) \quad (3.12)
\]

\[
\theta_2 = \arccos\left(\frac{a^2 + b^2 + l_2^2 - l_3^2}{2l_2\sqrt{a^2 + b^2}}\right) + \gamma \quad (3.13)
\]

\[
\theta_2 = \arccos\left(\frac{a^2 + b^2 + l_2^2 - l_3^2}{2l_2\sqrt{a^2 + b^2}}\right) + \arctan\left(\frac{b}{a}\right) \quad (3.14)
\]

Same thing can be applied, to calculate \( \theta_3 \)

\[
\cos(\alpha) = \frac{l_2^2 + l_3^2 - a^2 - b^2}{2l_2l_3} \quad (3.15)
\]

\[
\alpha = \arccos\left(\frac{l_2^2 + l_3^2 - a^2 - b^2}{2l_2l_3}\right) \quad (3.16)
\]

\[
\theta_3 - \alpha = -180 \iff \theta_3 = \arccos\left(\frac{l_2^2 + l_3^2 - a^2 - b^2}{2l_2l_3}\right) - 180 \quad (3.17)
\]

Finally, \( \theta_4 \) can be calculated by

\[
\theta_4 = \varphi - \theta_1 \quad (3.18)
\]

\[
\begin{pmatrix}
\theta_1 \\
\theta_2 \\
\theta_3 \\
\theta_4
\end{pmatrix} =
\begin{pmatrix}
\arctan\left(\frac{y_e}{x_e}\right) \\
\arctan\left(\frac{b}{a}\right) + \arccos\left(\frac{a^2 + b^2 + l_2^2 - l_3^2}{2l_2\sqrt{a^2 + b^2}}\right) \\
\arccos\left(\frac{l_2^2 + l_3^2 - a^2 - b^2}{2l_2l_3}\right) - 180 \\
\varphi - \theta_1
\end{pmatrix} \quad (3.19)
\]

There is multiple configurations (solutions), that leads to reach the desired pose, however, due to the mechanical constrains, there is one possible configuration. Rotation range constraints will be discussed later on work envelope section.
### 3.2.4 Velocity Kinematics

We have so far established the mathematical models for the forward kinematics and-inverse kinematics of the palletizing manipulator. These models describe the relationships between the static configurations of a mechanism and its end-effector. The focus here is on the models associated with the velocities and accelerations of articulated mechanisms and the Jacobian matrix which is central to these models.

The Jacobian generalizes the gradient of a scalar-valued function of multiple variables, which itself generalizes the derivative of a scalar-valued function of a single variable. In other words, the Jacobian for a scalar-valued multivariate function is the gradient and that of a scalar-valued function of single variable is simply its derivative.

Assuming that the end-effector coordinates and orientation \((x_e, y_e, z_e, \varphi)\), are multivariable functions, which the variables are the joints angles \(\theta_i\). These functions can be expressed by

\[
\begin{pmatrix}
    x_e \\
y_e \\
z_e \\
\varphi
\end{pmatrix}
= 
\begin{pmatrix}
f_1(\theta_1, \theta_2, \theta_3, \theta_4) \\
f_2(\theta_1, \theta_2, \theta_3, \theta_4) \\
f_3(\theta_1, \theta_2, \theta_3, \theta_4) \\
f_4(\theta_1, \theta_2, \theta_3, \theta_4)
\end{pmatrix}
\tag{3.20}
\]

The Jacobin matrix in this case, can be expressed by

\[
J(\theta) = \frac{\partial f}{\partial \theta} = 
\begin{bmatrix}
\frac{\partial f_1}{\partial \theta_1} & \cdots & \frac{\partial f_1}{\partial \theta_4} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_4}{\partial \theta_1} & \cdots & \frac{\partial f_4}{\partial \theta_4}
\end{bmatrix}
\tag{3.21}
\]

\[
J = 
\begin{bmatrix}
-s_1(l_1 + l_4 + l_2c_2 + l_3c_23) & -c_1(l_2s_2 + l_3s_23) & -c_1l_3s_23 & 0 \\
c_1(l_1 + l_4 + l_2c_2 + l_3c_23) & -s_1(l_2s_2 + l_3s_23) & -s_1l_3s_23 & 0 \\
0 & l_2c_2 + l_3c_23 & l_3c_23 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}
\tag{3.22}
\]

In general, the Jacobian allows us to relate corresponding small displacements in different spaces. If we divide both sides of the relationship by small time interval (i.e. differentiate with respect to time), we obtain a relationship between the velocities of the mechanism in joint and Cartesian space.

\[
\dot{f} = J(\theta)\dot{\theta}
\tag{3.23}
\]

\[
\dot{f} = J\dot{\theta}
\tag{3.24}
\]

\[
\begin{pmatrix}
    \dot{x}_e \\
    \dot{y}_e \\
    \dot{z}_e \\
    \dot{\varphi}
\end{pmatrix}
= 
J
\begin{pmatrix}
    \dot{\theta_1} \\
    \dot{\theta_2} \\
    \dot{\theta_3} \\
    \dot{\theta_4}
\end{pmatrix}
\tag{3.25}
\]
For brevity, let $v = (\dot{x}_e \ y_e \ z_e \ \dot{\phi})^T$ be the velocity vector, and $\omega = (\dot{\theta}_1 \ \dot{\theta}_2 \ \dot{\theta}_3 \ \dot{\theta}_4)^T$ be the angular velocity vector.

$$v = J\omega \quad (3.26)$$

The relationship that relate accelerations and joints angles, can be obtained by the derivation of both sides of the equation (3.26), with respect to time.

$$a = \dot{v} = \dot{J}\omega + J\dot{\omega} \quad (3.27)$$

where

$$J = \begin{bmatrix}
-e_1(l_1 + l_4 + l_2c_2 + l_3c_3)\dot{\theta}_1 + s_1(l_2s_2 + l_3s_3)\dot{\theta}_3 - s_1l_3s_3\dot{\theta}_1 - c_1l_3s_3\dot{\theta}_2 & 0 \\
-s_1(l_2s_2 + l_3s_3)\dot{\theta}_2 + s_1l_3s_3\dot{\theta}_1 & c_1(l_2c_2 + l_3c_3)\dot{\theta}_3 - c_1l_3c_3\dot{\theta}_1 - c_1l_3c_3\dot{\theta}_2 & 0 \\
c_1(l_2s_2 + l_3s_3)\dot{\theta}_3 - c_1l_3s_3\dot{\theta}_1 & c_1(l_2c_2 + l_3c_3)\dot{\theta}_3 - c_1l_3c_3\dot{\theta}_1 - s_1l_3s_3\dot{\theta}_2 & 0 \\
0 & -(l_2s_2 + l_3s_3)\dot{\theta}_2 - l_3s_3\dot{\theta}_3 & 0 \\
0 & -l_3s_3\dot{\theta}_2 - l_3s_3\dot{\theta}_3 & 0 \\
0 & 0 & 0
\end{bmatrix} \quad (3.28)$$

The inverse velocity problem seeks the joint rates that provide a specified end-effector twist. This is solved by inverting the Jacobian matrix. It can happen that the robot is in a configuration where the Jacobian does not have an inverse. Both inverse velocity and acceleration can be obtained by

$$\omega = J^{-1}v \quad (3.29)$$

$$\dot{\omega} = J^{-1}(a - \dot{J}\omega) \quad (3.30)$$
3.2.5 Work envelope

the working envelope refers to the working volume which can be reached by some point at the end of the Robot arm, this point is usually the center of the end-effector mounting plate. There are often areas within the working envelope which cannot be reached by the end of the Robot arm. Such areas are termed dead zones.

The workspace of this kind of mechanism is a bit different from normal serial vertical joint mechanism because of a closed-loop. For tradition serial robot, only rotation range of waist joint, shoulder joint and elbow joint are taken into consideration. But the closed loop introduces an additional constraint, which is the rotation range of $l_5$. Here the rotation range constraints, are listed in the table below.

Table 3.2: Rotation Range Constraints

<table>
<thead>
<tr>
<th>Joint</th>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waist</td>
<td>$\theta_1$</td>
<td>$-90^\circ$ $+90^\circ$</td>
</tr>
<tr>
<td>Shoulder</td>
<td>$\theta_2$</td>
<td>$+10^\circ$ $+130^\circ$</td>
</tr>
<tr>
<td>Elbow</td>
<td>$\theta_3$</td>
<td>$-20^\circ$ $-145^\circ$</td>
</tr>
<tr>
<td>Wrist</td>
<td>$\theta_4$</td>
<td>$-90^\circ$ $+90^\circ$</td>
</tr>
<tr>
<td>$l_5$</td>
<td>$\theta_2 + \beta$</td>
<td>$+90^\circ$ $+180^\circ$</td>
</tr>
</tbody>
</table>

Now that the forward kinematics is solved, the workspace can be got by these steps:

a): $\theta_1$ changes from its minimum value to maximum.

b): for each $\theta_1$ there exits a set of $\theta_2$ changing from its minimum value to maximum.

c): analogically for those $\theta_2$ there exits a set of $\theta_3$ changing from its minimum value to maximum. But if $\theta_2 + \beta$ reaches its maximum, $l_3$ must maintain its orientation for forming a parallelogram. So if $\theta_2 + \theta_3 + \pi \geq max(\theta_2 + \beta)$, then $\theta_3 = max(\theta_2 + \beta) - \theta_2 - \pi$. And if $\theta_2 + \theta_3 + \pi \leq min(\theta_2 + \beta)$, then $\theta_3 = min(\theta_2 + \beta) - \theta_2 - \pi$.

d): substitute all sets of $(\theta_1, \theta_2, \theta_3, \theta_4)$, and the value of $(l_1, l_2, l_3, l_4, d_1, d_4)$ into equation (3.9).

Table 3.3: Links Lengths and Offsets

<table>
<thead>
<tr>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
<th>$l_4$</th>
<th>$d_1$</th>
<th>$d_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>21mm</td>
<td>148mm</td>
<td>160mm</td>
<td>35mm</td>
<td>143mm</td>
<td>53mm</td>
</tr>
</tbody>
</table>
Figure 3.4: Work Envelope

(a) XOZ Plane

(b) XOY Plane
3.3 Summary

This chapter has described the kinematic model and analysis of a hybrid mechanism 4DOF palletizing robot. We started with a brief description of the mechanical structure, followed by the forward and the inverse kinematics analysis along with velocity kinematics model, and we ended up with work envelope analysis and simulation. The obtained results in this chapter are used to build a real world palletizer robot, which is the topic of the next chapter.
Chapter 4

SOFTWARE AND HARDWARE IMPLEMENTATION

4.1 Introduction

The robots play important roles in our lives and are able to perform the tasks which cannot be done by humans in terms of speed, accuracy and difficulty. Robots can be employed to imitate human behaviors and then apply these behaviors to the skills that leads the robot to achieve a certain task. They do not get tired or face the commands emotionally, and since they are designed by humans. They can be programmed and expected to obey and perform some specific tasks. In some cases the use of a robotic hand becomes remarkable. Robotic is applied in different forms and fields to simulate human behavior and motions.

With the evolution of electronics and open source plug and play boards such as arduino and raspberry, along with the development of software solutions, both professionals and amateurs are capable of creating very interesting projects for instance, home automation, robotics, gaming, etc. These projects are not only easy to build, but also very cheap due to the technology of CNC machines and 3D printing.

Our goal here is to build a palletizer robotic arm which is capable of recognizing object with the guidance of a machine vision system, and sorting the parts through pick-and-place application. The robot will be an open source project both software and hardware to help students and amateurs to understand some basic robotics and build their own robot and adding custom features. thus, in this chapter the implementation and combination of both software and hardware to build a robotic system which can be used in real-world industrial application and not only for educational purposes.
4.2 Robot Components

Robots consist of a variety of high tech components which are designed for accuracy, speed and convenience of operation. Some are for general purpose usage, while some are custom made to handle specific parts. The main components are

1. Main body (Mechanical Structure)
2. Controller
3. Drive
4. End-effector
5. Costume Features

4.2.1 Main Body

An important component of a robot is the main body which holds the actuators and manipulators that create the activity for each axis of movement. The manipulator carries the end effectors which grip the objects. Mechanisms that provide response regarding the location are included for identification and rectification of any difference between the chosen position in accordance with the command and the existent position. The intelligence of the robot is in the control element, which directs the manipulator along the ideal route. A power supply is essential to activate the actuators. The considered mechanical body is a hybrid parallelogram mechanism which is discussed in details in chapter 3.

4.2.2 Controller

Every robot is connected to a computer controller. The controller is the "brain" of the industrial robotic arm and allows the parts of the robot to operate together. It works as a computer and allows the robot to also be connected to other systems. The robotic arm controller runs a set of instructions written in code called a program. The program is inputted with a teach pendant. Many of today’s industrial robot arms use an interface that resembles or is built on the Windows operating system. In the future, controllers with artificial intelligence (AI) could allow robots to think on their own, even program themselves. This could make robots more self-reliant and independent. There is two parts that consist the controller

1. Hardware
2. Software