Perception-Driven Robotic Manipulation and Motion Planning for Packaging of Dynamic Compressible Objects

by

Rohith Venkataramanan

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Masters of Science

in

Robotics Engineering

May 2023

Approved by:

Dr. Siavash Farzan, Advisor

Dr. Jing Xiao, Thesis Committee Member

Dr. Berk Calli, Thesis Committee Member

Abstract

This thesis presents the development of an innovative robotic system for the efficient and accurate manipulation of envelope stacks moving on a conveyor belt in the mailing and packaging industries. The research aims to address the challenges in envelope stacking and packing by leveraging vision-based techniques for detecting stacks of envelopes and estimating conveyor velocity, as well as trajectory planning and control for robotic manipulation. The proposed solution integrates a custom-designed pneumatic gripper, a camera-based perception system, motion planning techniques, and velocity servoing to precisely pick and place multiple stacks of envelopes from a conveyor belt into a cardboard box. The camera system is designed to detect the envelope stacks and estimate the conveyor velocity by tracking envelope positions over time. The robotic manipulator employs velocity servoing to accurately probe the envelopes and slice into the stacks using the elastic and flexible properties of envelopes.

A series of experiments and performance metrics are used to evaluate the system's effectiveness in terms of grasping success rate, localization accuracy, conveyor velocity estimation error, stack detection performance, cycle times, and box-filling accuracy. Our proposed system demonstrates high success rates in grasping, localization accuracy, and envelope stack packing efficiency while maintaining the integrity of the envelope stacks. The research findings and proposed solution hold the potential to significantly improve automation processes in the mailing and packaging industries, paving the way for further advancements in robotic manipulation, motion planning, and perception-guided control.

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Dr. Siavash Farzan, for his unwavering support and patience throughout the course of my research. His expertise, guidance, and willingness to help me overcome challenges, especially during periods of research failures, have been instrumental in the completion of this thesis. I want to thank Dr. Jing Xiao and my advisor for giving me the opportunity to work on such an interesting project. I would like to express my sincere gratitude to my committee members, Dr. Jing Xiao and Dr. Berk Calli, for providing valuable insight and advice that helped shape my thesis. Their profound knowledge and constructive feedback have been invaluable in refining my research and ensuring its quality.

I am also extremely grateful to my family for their love, encouragement, and faith in my abilities. Their constant support has provided me with the strength to persevere through difficult moments and celebrate the successes along the way. I would like to extend my heartfelt thanks to my close friends, who have stood by me and offered both moral support and valuable advice during this journey. Their camaraderie and shared experiences have made this process a memorable and enjoyable one. I highly appreciate Zhaoyuan Ma for his valuable input and ideas that helped shape my research project.

I would like to thank everyone who has contributed to my academic and personal growth during my time in the program. The knowledge, experiences, and relationships I have gained from my course of study and research will remain with me for years to come.

Contents

1 Introduction **1**

2 Related Work **6**

2.1 Motion Planning and Control 6

2.2 Manipulation of Deformable Objects 7

2.3 Pick-and-Place Optimization 9

2.4 Gripper Design 9

2.5 Perception for Object Handling 11

3 Envelope Gripper Design **14**

3.1 Mechanical Design Considerations 14

3.2 Gripper Design and Fabrication 19

3.3 Gripper Integration and Evaluation 22

4 Envelope Stack Detection and Localization **25**

4.1 Notch Detection in Simulation 25

4.2 Notch Detection in Hardware 31

4.3 Notch Position and Stack Dimensions Identification 40

5 Motion Planning and Servoing **44**

6 Experimentation and Results **60**

6.1 Experiment Setup 60

6.2 Perception Experiments 62

6.2.1 Notch Detection Under Different Lighting Conditions 62

6.2.2 Notch Velocity Tracking Performance 64

6.3 Pick-and-place Experiments 65

6.3.1 Pick and Place Analysis 65

6.3.2 Grasp Failure Modes 67

6.3.3 Safe Retract Analysis 70

7 Conclusion and Future Work **72**

List of Figures

1.1 Notches that stick out of envelopes to denote the end of a stack . . . 3

1.2 The Festo HGPL Parallel gripper [8](left) and the Zimmer Pneumatic GHK6000 series gripper [13](right) both of which are double-acting pneumatic grippers and can reach stroke lengths up to 200mm. 4

3.1 Free body diagram of envelope stack when grasped by the gripper jaws [37] 18

3.2 CAD model of the envelope gripper in closed (left) and open (right) configurations 20

3.3 Mechanical assembly of the envelope gripper 21

3.4 Schematic layout of electrical and pneumatic gripper connections . . . 23

3.5 Hardware integrated gripper with electrical and pneumatic connections 24

3.6 Linear motion test: Robot repeating movement from the pose in (a) to pose in (b) and vice versa; Circular oscillatory motion test: Robot repeating movement from the pose in (c) to pose in (d) and vice versa 24

4.1 Simulation of a virtual RGB-D camera mounted from the ceiling which captures images of envelope stacks 26

4.2 CAD model of an envelope stack used in the Gazebo simulation . . . 26

4.3 Sides of notches forming trapezoidal shapes from the RGB camera's perspective 27
4.4 Actual notch positions highlighted in red - leftmost edge for notches to the left and rightmost edge for notches to the right of the image center 29
4.5 RGB Region of Interest (left) and color thresholding mask (right) . . 30
4.6 Edge detection using Canny Edge Detection 30
4.7 Line segment detection and filtering on the basis of slope (highlighted in red) 31
4.8 Bounding boxes (green) identified using the detected notches 31
4.9 Depth images acquired in simulation (left) and hardware (right) . . . 33
4.10 Envelope segmenting using depth image filtering 35
4.11 (a) Isolated envelopes with notches; (b) Result of applying filters to the isolated stacks 38
4.12 Mask created from color thresholding 39
4.13 Plots of detected (a) pairs of notch points; (b) true notch points in mask 39
4.14 True notch locations highlighted as yellow circles 40

5.1 High-level overview of motion planning and servoing processes for envelope packaging from a conveyor belt 45
5.2 Cardboard box inclined at an angle using a simple mechanical structure 46
5.3 A diagram representing parameters used for calculating placing poses 55
5.4 Step-by-step depiction of the dynamic envelope stack grasping process 57
5.5 Step-by-step depiction of the envelope pick and place 58
5.6 Notch envelope in contact with the jaw sliding outside the stack due to friction 58

5.7 Axes for sliding motion from the envelope in contact while retracting 58
5.8 Velocity servoing commands executed to avoid adhesive effect of envelope in contact . . . 59

6.1 Hardware setup of the robotic system for experimental analysis . . . 61
6.2 MoveIt collision added structured environment . . . 61
6.3 Top view of envelope holding mechanism . . . 62
6.4 Measuring the localization error by commanding the robot exactly to the notch position . . . 63
6.5 Notch detection in the studied light Illuminance conditions from 418 Lux to 42 Lux. . . . 64
6.6 Plot of velocity estimation error under different conveyor speeds . . . 65
6.7 Picking failure caused by a combination of the other failures or inadequate pressure . . . 68
6.8 Notch probing overshoot caused by accuracy errors in perception pipeline . . . 68
6.9 Failure to slice into the stack safely . . . 69
6.10 Failure to slide the notch into the stack due to inadequate force applied during sliding motion . . . 70

List of Tables

6.1 Notch Detection Error for Different Light Intensities 64

6.2 Performance of the Robotic System for Different Conveyor Velocities 67

6.3 X and Z axis speeds for executing motion to overcome the friction between the jaw and envelope in contact; corresponding displacement if arm retract was successful 70

Chapter 1

Introduction

The packaging of compressible items on conveyor belts is a frequent and difficult operation in numerous sectors including e-commerce, manufacturing, and logistics. To optimize supply chain processes and cut costs, it's imperative to be able to package such items precisely and effectively. The time-consuming, labor-intensive, and error-prone nature of conventional manual packing processes involving human workers, however, can result in inefficiencies and errors. Robotic systems that automate the packaging of compressible objects have been developed in order to get around these difficulties.

Robotic systems are designed to work in real-time and with high accuracy, making them suitable for handling large volumes of objects quickly and efficiently. However, designing an effective and efficient robotic packaging system that can handle compressible objects presents several unique challenges. By proposing a robust and efficient robotic system that solves these challenges, this thesis aims to contribute significantly to the handling of delicate items, increase efficiency, and reduce the reliance on manual labor in sorting facilities. The successful implementation of this robotic system can inspire solutions for other industries requiring precision handling,

such as food processing, pharmaceuticals, and electronics manufacturing, where the handling of delicate or sensitive materials is crucial. This thesis presents the development and implementation of a robotic manipulator system designed to automate the picking and placing stacks of envelopes moving on a conveyor belt.

Compressible objects, such as envelope stacks, have a variable and dynamic shape, making them difficult to handle and package using conventional grippers. For example, when handling stacks of envelopes, they tend to bend and deform, making it challenging to pick and place them accurately. Furthermore, the robotic system must be equipped to handle different dimensions and variations of envelopes that would be encountered in the production line. Kicki et al. [23] proposed a method for perceiving and localizing elongated compressible objects, enabling successful grasping and exploratory movements with a human-like gripper, by incorporating prior shape knowledge from visual system computations and estimating object elasticity through raw force signals from the gripper.

The proposed 6 degrees of freedom robot is equipped with a custom-designed pneumatic gripper, enabling it to securely hold and release long stacks of envelopes without causing damage or disturbance. A conveyor belt running at constant velocity is loaded with a long line of envelopes that will be produced by machines. Envelope-producing machines have the ability to count the envelopes as they are manufactured, and can make the final envelopes stick out after each stack of a specified number of envelopes is produced. A camera system is mounted above the conveyor belt to detect "notches," which are the final envelopes in each stack that protrude slightly relative to all the other envelopes as shown in Figure 1.1. These notches are useful for workers in industries to determine the number of stacks and their positions for packaging. By tracking the notch positions over time, the camera system also calculates the conveyor velocity, which is useful for precise and

synchronized operation between the robot and the conveyor.

Figure 1.1: Notches that stick out of envelopes to denote the end of a stack

A major factor in finalizing a gripper for packaging stacks of envelopes is it must have the necessary stroke length (nearly 0.2m) and normal forces required to pick and place at least 200 envelopes during each cycle. On average, a stack of 200 envelopes will be 15-16 cm long. While there are some grippers available in the market that satisfy these requirements, most of them are either too bulky or slow-acting. There is the additional challenge of isolating envelope stacks from the major stack using the notches, which will be discussed in Chapter 5. A fixed gripper jaw makes it much easier to probe these notches, slice into the envelopes and grasp one stack at a time with a moving jaw on the other end. Moreover, after careful consideration, pneumatic actuation was chosen for the gripper mechanisms due to reasons discussed in Chapter 3.

Off-the-shelf pneumatic grippers were initially considered with stroke length, payload, and operation mode as the main factors. There are industry-standard

parallel two-jaw grippers as shown in Figure 1.2 which have enough stroke length and payload to be suitable for this application. It would be much more challenging to isolate and grasp a moving envelope stack if both the gripper jaws were closing. The lack of single-acting pneumatic grippers in the market with high stroke lengths and low payloads necessitates the design of a custom gripper.

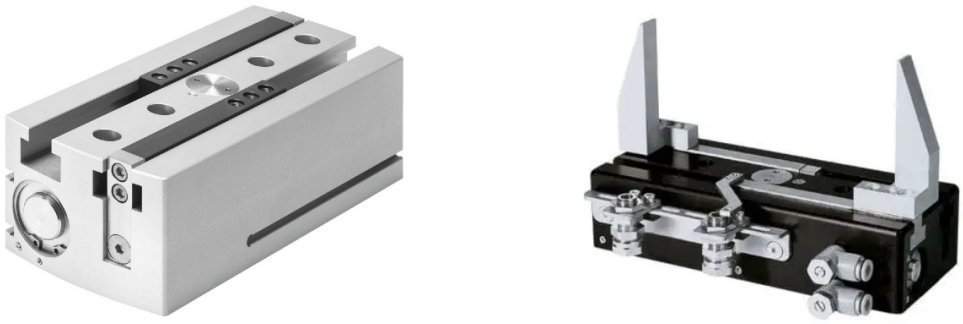

Figure 1.2: The Festo HGPL Parallel gripper [8](left) and the Zimmer Pneumatic GHK6000 series gripper [13](right) both of which are double-acting pneumatic grippers and can reach stroke lengths up to 200mm.

In developing a comprehensive robotic system for this application, special attention must be given to several key components. Firstly, the gripper design must be tailored to cater to the specific requirements of handling envelopes with precision and care. Secondly, an accurate perception pipeline is crucial for the effective detection and tracking of notches, even in the presence of sensor noise and varying lighting conditions. This ensures a high degree of reliability and consistency in the system's performance. Lastly, the motion planning pipeline plays a significant role in ensuring that the robotic system can efficiently and safely pick and place the envelopes from a conveyor into cardboard boxes of different dimensions.

This thesis has been organized into several chapters, each addressing a crucial aspect of developing an efficient and reliable robotic system for picking and placing stacks of envelopes:

- Chapter 2 delves into the review of literature work in the main areas relevant to the application of packaging envelopes.
- Chapter 3 focuses on the design considerations and fabrication methods for developing a novel gripper and how it is integrated into the robotic system.
- The next chapter discusses the development of a robust vision pipeline for notch detection and envelope stack dimension identification.
- Chapter 5 presents a motion planning and servoing-based approach to securely grasp, pick up, and place moving envelopes into a corrugated cardboard box.
- In chapter 6, the experimentation for analyzing the system's reliability and performance and the corresponding results are presented.
- The final chapter provides an overview of the methodology, evaluation, and results of the previous sections and suggests future work to further improve the robotic system.

Chapter 2

Related Work

This chapter explores the foundational research and advancements in the crucial areas in the development of our robotic system for picking and placing stacks of envelopes on a conveyor belt. In the field of robotics, several approaches have been proposed to achieve successful grasping and manipulation of objects, taking into account various factors such as the robot's capabilities, the object's shape, and the environment. In this chapter, we discuss some recent research works in this field, dividing them into subsections based on the approach used.

2.1 Motion Planning and Control

Motion planning and control are crucial components of object manipulation in robotics. In this subsection, we present some recent works that focus on motion planning and control of grasping.

Fontanals et al. [11] proposed a method for obtaining an end effector's goal pose by considering the robotic arm's capabilities and a successful grasp configuration using independent contact regions. Vahrenkamp et al. [46] presented a planning strategy using rapidly exploring random trees (RRTs) to build a tree of reachable

and collision-free configurations. Haustein et al. [16] proposed a method that plans fingertip contacts and arm motion simultaneously by exploiting the hierarchical structure of the Hierarchical Fingertip Space grasp planner. Their method uses RRTs to establish a basis for the grasp search space, considering both the local geometry of the object surface and fingertip geometry.

Reaching control is an essential aspect of object manipulation that involves the robot's ability to move toward a desired object. Gaskett et al. [12] implemented a control system that combined closed and open loops to enhance the robot's ability to move toward a desired object by correcting positioning errors after open-loop reaching. Jamone et al. [19, 20] proposed a strategy that combines exploration and exploitation to enable the robot to autonomously learn to reach objects through three-dimensional space. Jørgensen et al. [21] present a robot system for performing pick and place operations with deformable objects that uses a structured light scanner to capture a point cloud of the object to be grasped, and the determined action is executed by the robot to solve the task. To achieve smoother movement without any abrupt changes, [36] proposes an automated trajectory planner for the near-optimal trajectory of 6-DOF robotic manipulators for the given Cartesian description of end-effector at pick and place destinations under kinematic constraints acceleration bounded S-curve trajectory. [31] modifies this approach as a jerk-based bounded S-curve trajectory.

2.2 Manipulation of Deformable Objects

Handling deformable objects is a challenging problem in robotics, and several recent works have focused on this area.

Bodenhagen et al. developed an adaptable robotic system to handle silicon el-

ements as a test-friendly replacement for meat [5]. Misimi et. al [35] proposed the GRIBBOT, a 3D vision-guided robotic system for front half chicken harvesting that uses a compliant multifunctional gripper tool with a beak and a supporting plate to grasp and hold the fillet. Balaguer et. al [2] addressed the issue of handling deformable objects using cooperative manipulators, specifically for towel folding tasks, using a momentum fold and a learning algorithm combining imitation and reinforcement learning. Jørgensen et. al [4] discussed optimization schemes for solving robotic optimization problems related to grasping and manipulating deformable objects, using a dynamic simulation framework to model the performance of the solutions. Wu et al. [47] proposed a model-free visual reinforcement learning approach to tackle the problem of manipulating deformable objects, using iterative pick-place action space and learning only the placing policy conditioned on random pick points.

The paper [41] introduces a suite of simulated benchmarks for manipulating deformable objects in 1D, 2D, and 3D structures. It proposes embedding goal-conditioning into Transporter Networks, which rearrange deep features to infer displacements that represent pick and place actions. The authors demonstrate that goal-conditioned Transporter Networks enable agents to manipulate deformable structures into flexibly specified configurations without test-time visual anchors for target locations. The paper also extends prior results using Transporter Networks for manipulating deformable objects by testing on tasks with 2D and 3D deformable.

These recent works demonstrate the significant progress made in the planning and control of object manipulation in robotics, highlighting the importance of developing effective strategies for grasping, reaching, and handling deformable objects.

2.3 Pick-and-Place Optimization

Pick and place cycles in an industrial setup is a non-value-added activity that cannot be eliminated, but it can be framed in a way that has a minimal lead time and gives maximum production for economic reasons. To implement the trajectory in real time, the method must have a low computational complexity and must possess a smooth jerk profile to reduce the stress induced in the actuators and increase the tracking accuracy [28,34]. The movement of an object manipulated by an industrial robot, especially in pick-and-place applications, involves both optimizations of the movement of the object manipulated with respect to the end-effector and the optimization of the positioning of the robot base relative to the application defined by the two positions [43].

Working in an optimization framework involves minimizing or maximizing a set of objective functions. [38] provides more details on the most commonly considered optimization criteria used in path planning literature such as minimum time trajectory planning, minimum actuation effort, or minimum jerk trajectory planning. Often, multi-criteria optimization involving two or more objective functions is considered to attain better results. The challenges of path accuracy, the importance of minimum-time trajectory, and the performance of motion controllers faced in an industrial setting are discussed in [24].

2.4 Gripper Design

Successful grasping is critical for robotic systems to efficiently manipulate objects in industrial applications, and this requires a reliable and robust gripper design. The design characteristics of a gripper determine its ability to carry out its intended functions. Thus, robustness, durability, and adaptability to varying conditions should

be the focus of the design. The design should also consider the sensitivity of the gripping surface, as well as the sensor integration. Passive gripping mechanisms allow for lifting objects without controlling the robot's hand.

H. Itoh et al. [18] introduces an electroadhesive paper gripper with a pad consisting of interdigital electrodes that adhere to a sheet of paper when a high voltage is applied. Although this method works, it is limited as it requires high voltage and can pick up only one paper at a time. A. Saboukhi et al. [40] proposes a 2-jaw gripper with parallel jaws and FSR sensors that are calibrated to detect object presence, prevent damage, and evaluate object weight through the coefficient of friction. The gripper is optimized to maximize the weight it can hold and can be used in palletizing with a 3-DOF robotic arm. However, the sensor requires calibration before each pickup.

A. Hassan et al. [15] presents a single DOF robot gripper with four fingers that move simultaneously using a four-bar and slider-crank mechanism. The design is optimized for grasping objects of various shapes and is intended for educational and research purposes, which can be considered a limitation of our application. M. Guo et al. [14] uses data-driven optimization of gripper jaw surfaces to improve grasp robustness. The design process involves creating and evaluating 37 variations of jaw surfaces using physical grasp experiments and hill-climbing in parameter space. The optimized design is outperforming the factory-provided gripper tips and gecko-inspired surfaces. This approach shows promise for improved grasp robustness and performance in industrial settings.

In [10], a novel soft robotic gripper design with variable stiffness is introduced. The design employs an adaptive optimization method for stiffness adjustment to provide effective object grasping. The gripper is validated with objects of different sizes and weights, and it is suggested to use pneumatically powered systems

for safety. Hua, H et al. [17] presents a pneumatic underactuated robotic gripper (PURG) with a feedforward grasping force control method based on learned kinematics. The PURG's control system includes an air compressor, servo valve, pressure regulator, buffer gas tank, and microcontroller. The approach is validated through actuating force control and grasping experiments.

In [33], a novel dual-stage shape memory alloy (SMA) actuated gripper is proposed. The design improves grasp performance through primary and secondary actuation. The proposed numerical analysis method provides approximate solutions validated through test experimentation. Results show that the design can handle objects of varying sizes, weights, and shapes, with a maximum grasping force of 2 N.

2.5 Perception for Object Handling

Over the last few decades, the development of robotic tracking and grasping of moving objects has attracted significant attention due to its potential applications in several areas such as manufacturing, assembly, and material handling. This literature survey aims to provide an in-depth analysis of ten research papers that have contributed to this field, highlighting their methodologies, techniques, and findings.

P.K. Allen et al. in [1] developed a system that tackled three distinct issues in robotic hand-eye coordination for grasping moving objects, namely fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and interception and grasping. The system was able to operate at approximately human arm movement rates, and experimental results demonstrated successful tracking, stable grasping, and picking up of a moving model

train.

K. Kondak et al.'s [27] introduced a non-linear time-optimal controller for manipulators that considered the position, velocity, and acceleration parameters of the moving object determined by a sensor system. The goal was to achieve the grasping conditions in the shortest possible time while adhering to the constraints of the manipulator's joints.

Li Ge and Zhao Jie, in the paper [30], implemented parallel algorithms to improve the computational efficiency of the stereo visual servoing system. These parallel algorithms divided the computation tasks into smaller, independent subtasks that could be executed concurrently by multiple processors or cores. By distributing the workload, the system was able to process the visual data more quickly and effectively. The use of parallel algorithms in stereo vision processing allowed for faster extraction of 3D information about the object's position, orientation, and motion, resulting in more accurate tracking and grasping.

In [3], a new position-based visual servoing approach for the dynamic manipulation of moving objects was presented. They employed a global-local vision tracking method using two cameras and a dynamic extended Kalman filter (EKF) to estimate the kinetic parameters. A biomimetic motion planning algorithm for the manipulator was also developed, along with a two-stage task planning process consisting of tracking and grasping stages.

Ming Lei and B.K. Ghosh's [29] discussed position-based and image-based tracking schemes for a robot manipulator. The authors demonstrated that both schemes were equivalent when the motion of the object was accurately estimated. They also showed that the image-based controller could be improved by incorporating the image of the gripper, which helped cope with system calibration errors.

In [42], a position-based visual tracking system using a Kinect camera for a

7 DOF PowerCube manipulator from Amtec Robotic was presented. The authors employed a Kalman filter to predict the target position and velocity, and the stability analysis was derived from real-time experiments on both static and moving targets.

Dellen et al.'s [9] introduced a framework for joint segmentation and tracking in-depth videos of object surfaces. The method used the 3D colored point cloud obtained from a Kinect camera to segment the scene into surface patches, which were then used to partition the depth image of the subsequent frame consistently with the precedent frame. The algorithm was tested for scenes showing human and robot manipulations of objects, proving the effectiveness of their proposed method.

Overall, these research papers have made significant contributions to the development of robotic tracking and grasping of moving objects. By using different approaches, including visual servoing, motion planning, and controller design, these studies have led to the development of advanced and efficient systems that have potential applications in various areas.

Chapter 3

Envelope Gripper Design

Developing an efficient, accurate, and safe gripper is crucial for successfully picking and placing envelope stacks from a conveyor belt and packaging them into a cardboard box. This chapter details the design and fabrication process of our custom-designed pneumatic gripper, highlighting the key considerations and innovations that distinguish it from existing solutions and make it suitable for the system requirements.

3.1 Mechanical Design Considerations

When it comes to picking compressible and fragile moving objects, having a fixed jaw and a single moving jaw in the gripper offers certain advantages over a design with two moving jaws. A key benefit is an enhanced ability to place stacks of envelopes precisely at the edges of the cardboard box. With a fixed jaw, the gripper can maintain a stable reference point, enabling more accurate and controlled placement of the stacks especially when positioning them in tight or confined spaces near the box edges. In contrast, two moving jaws may introduce additional complexity in controlling and synchronizing the movement of both jaws, potentially leading to

reduced precision in stack placement. A fixed jaw design can simplify the gripper's mechanical structure, reducing the number of actuators, components, and potential failure points.

Multiple actuation methods were considered for the gripping mechanism. Pneumatic actuation offers several benefits over alternative actuation methods, such as electric, hydraulic, and mechanical systems, making it a more suitable choice for our gripper design. Here are some of the key advantages of pneumatic actuation:

- **Fast Response Time:** Pneumatic actuators are known for their rapid response times, as compressed air can be quickly channeled to the actuator. This allows the gripper to quickly open and close, enabling efficient operation and minimizing the time spent handling each envelope stack. Electric actuators, in contrast, can be limited by the speed of the motor, while hydraulic actuators may experience delays due to the movement of fluid within the system.

- **Safe Handling:** The inherent compliance of pneumatic systems makes them well-suited for delicate operations, such as handling envelopes. The compressed air within the system provides a degree of cushioning, allowing the gripper to apply force gently and avoid damaging the envelopes during the grasping process. Electric actuators can be more difficult to control in terms of applied force, while hydraulic actuators, due to their reliance on incompressible fluids, typically offer less compliance.

- **Lightweight and Compact:** Pneumatic actuators are generally lighter and more compact than their electric and hydraulic counterparts, making them an ideal choice for integration with robotic manipulators. A lightweight gripper minimizes the impact on the manipulator's payload capacity and ensures that the overall system remains agile and responsive. Electric actuators often re-

quire bulky motors, while hydraulic systems can be cumbersome due to the presence of fluid reservoirs and hoses.

- **Scalability:** Pneumatic systems can be easily scaled to accommodate various envelope sizes and stack heights, as the actuation force can be adjusted by simply modifying the air pressure supplied to the gripper. This flexibility is advantageous for envelope-handling applications, where the gripper must adapt to a range of stack configurations. Electric and hydraulic actuators can be more challenging to scale, often requiring the replacement of components or the use of additional gear ratios.

- **Reliability and Low Maintenance:** Pneumatic actuators are known for their high reliability and low maintenance requirements, as they have fewer moving parts than electric and hydraulic systems. The absence of motors, gears, and fluid components reduces the likelihood of mechanical failure, wear, and leakage, resulting in lower maintenance costs and less downtime for the gripper.

- **Safety:** Pneumatic systems are generally considered to be safer than electric and hydraulic actuators, as they do not pose risks associated with electrical hazards or high-pressure fluids. In the event of a system failure, the compressed air can be safely vented, minimizing the potential for damage or injury.

Amongst linear pneumatic actuators, there exist traditional and rodless pneumatic cylinder options in the market. Rodless pneumatic cylinders are preferred for the design of the gripper over traditional pneumatic cylinders due to several advantages they offer, particularly in terms of compactness and adaptability. The absence of a protruding piston rod in rodless cylinders allows for a more streamlined design, making them more compact and better suited for confined spaces around conveyor

belts and packing stations. Additionally, rodless cylinders can provide a longer stroke length compared to rod-style cylinders of the same overall length, enabling the gripper to handle a broader range of envelope sizes and stack heights without the need for additional modifications. Their direct connection between the piston and the external carriage minimizes side loading and reduces the risk of bending or misalignment, ensuring accurate positioning and smooth operation while handling envelopes. Furthermore, rodless pneumatic cylinders offer a more simple design while making the gripper more compact and lightweight.

To ensure that the gripper is capable of lifting reasonable payloads of at least 200 envelopes, it is necessary to conduct a force analysis and determine the normal forces required to create frictional forces that can safely handle the payload. This analysis is crucial in selecting an appropriate pneumatic cylinder to achieve the desired force requirements. To aid in this analysis, a free-body diagram of the envelope stack when grasped by the gripper jaws is shown below in Figure 3.1.

The normal forces F_{y1} and F_{y2} applied by the fingers are considered to be equal since only the moving jaw applies force on the envelopes (object) and the fixed jaw. As a result, the frictional forces F_{z1} and F_{z2} applied by the jaws are equal, assuming both jaws have the same coefficient of friction (μ). Considering each envelope has a maximum mass (m) of 10g, the net mass of a stack of envelopes (M) is 2kg. Hence the gravitational force experienced by the stack under gravitational acceleration (g) is given by

$$F_{po} = M * g = 2kg * 9.81m/s^2 = 19.62N$$

If the gripper jaws have reasonable friction for holding the envelopes, we can assume a friction coefficient (μ) of 0.6. The minimum normal force (F_n) required to

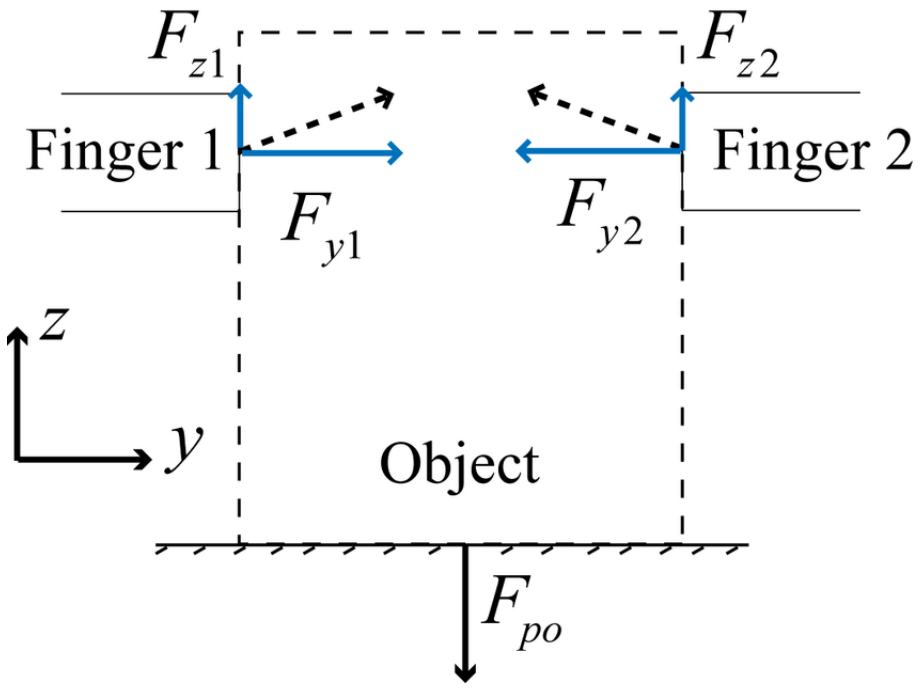

Figure 3.1: Free body diagram of envelope stack when grasped by the gripper jaws [37]

prevent slipping is given by

$$F_n = F_{po}/\mu = 19.62N/0.6 = 32.7N$$

We can add a safety factor (S) to account for uncertainties, inaccuracies, and variations that may arise during the operation. Since the application involves simple pick and place operations, we can consider $S = 2$. Then the required gripping force F_{grip} is given by

$$F_{grip} = F_n * S = 32.7N * 2 = 65.4N$$

To evaluate the pneumatic actuator's force, we consider an operating pressure (P) and minimum piston diameter (D). At an operating pressure of 3.5 bar (350000

Pa), the actuator force (F_{act}) is given by

$$F_{act} = P * \pi * (D/2)^2 = (274889.357 * D^2)N$$

Equating $F_{act} = F_{grip}$, we get the minimum piston diameter $D = 0.0154m$. Hence a rodless pneumatic cylinder with a bore diameter of 16mm was chosen as a suitable actuator for the gripper.

For the seamless electronic operation of the pneumatic cylinder, it is essential to incorporate a solenoid valve that facilitates reliable opening and closing of the jaws. In selecting the most compatible solenoid valve, critical factors such as operating pressure, flow rate, and port size compatible with the pneumatic cylinder were evaluated. Given that the application employs a double-acting cylinder, a 4-way (5-port) 2-position single solenoid spring return valve powered by a 24V supply emerged as the optimal choice for gripper operation.

3.2 Gripper Design and Fabrication

Beyond the design considerations outlined in the previous section, the gripper must also exhibit a compact form factor to accommodate placement within narrow boxes, and feature slender jaws capable of delicately slicing into envelope stacks without causing damage. Taking these constraints into account, a Computer-Aided Design (CAD) model of the gripper was developed, as illustrated in Figure 3.2. The model was designed to be streamlined for minimal overall weight while maintaining a careful balance of form and function.

The gripper jaws, side brackets, and top flange (colored in orange in Figure 3.2) were designed to be 3D printed. The external frame is an assembly of 20mm x 20mm Aluminum extrusions to maintain a rigid structure to support the pneumatic cylin-

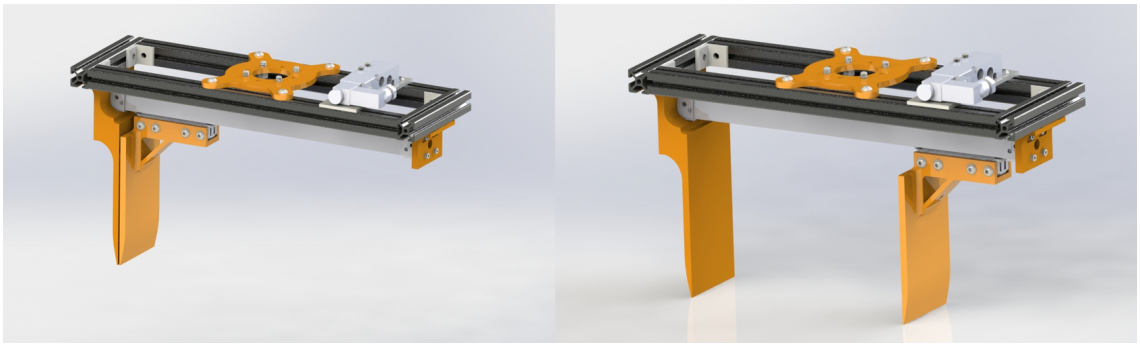

Figure 3.2: CAD model of the envelope gripper in closed (left) and open (right) configurations

der and jaws which are connected using the side brackets. Moreover, the solenoid valve is mounted on top of the extrusions with a 3D-printed plate. On completion of the CAD model design, the parts were fabricated and assembled (excluding the solenoid valve) as shown in Figure 3.3.

The jaws, flange, and brackets were printed in Polycarbonate (using an Ultimaker 3 Extended printer) with a layer height of 0.15mm and infill of 90% to maximize tensile strength and impact resistance. Other common Fused Deposition Modeling (FDM) materials such as PLA, ABS, and PETG were compared with Polycarbonate for printing. PLA, while easy to print and environmentally friendly, lacks the strength, rigidity, and temperature resistance of polycarbonate, making it less suitable for applications requiring high durability and performance. ABS, another popular 3D printing material, offers better impact resistance and temperature stability compared to PLA, but it still falls short of polycarbonate's strength and rigidity. PETG, known for its balance of strength, ease of printing, and chemical resistance, still does not compete with the mechanical properties of polycarbonate, especially when it comes to impact resistance and dimensional stability. By choosing polycarbonate for the 3D printing of gripper components, the resulting structures can be expected to outperform those created with alternative materials in demand-

ing pick-and-place operations. To improve the coefficient of friction between the jaws and the envelopes, 2mm thick Silicone rubber sheets were stuck to the jaw surfaces.

Figure 3.3: Mechanical assembly of the envelope gripper

A notable challenge arises when employing the pneumatic cylinder for operating the gripper, which stems from its rapid actuation capabilities. The high-speed nature of the cylinder's operation presents potential risks to the envelopes during the picking and placing processes. The swift movements exerted by the gripper could not only result in damage to the envelopes but also to the motors of the robot due to the sudden jerks and disturbances while executing trajectories.

Mechanical flow control valves offer a viable solution to the challenge of rapid opening and closing of the pneumatic cylinder during operation. These valves regulate the flow of compressed air entering and exiting the cylinder, allowing for precise control over the actuation speed of the gripper. By adjusting the flow rate, the gripper's opening and closing speed can be fine-tuned, ensuring a smoother and more controlled movement. Incorporating mechanical flow control valves in the pneumatic system enables a gradual acceleration and deceleration of the jaw's motion.

This mitigates the risk of damage to the envelopes during the picking and placing processes by preventing abrupt or forceful contact. By integrating mechanical flow control valves into the gripper's pneumatic system, the delicate balance between speed and precision can be effectively maintained.

3.3 Gripper Integration and Evaluation

The integration of the gripper with the robot necessitates electrical and software interfacing with the solenoid valve responsible for controlling the moving jaw. The robot has a tool I/O connection near the flange, allowing for seamless connectivity between the solenoid and the robot via an industrial-grade cable. This connection enables control of the solenoid through the robot's teach pendant. For the pneumatic connections, 6mm ether-based polyurethane tubes are used.

The solenoid valve interfaces with a pressure regulator, which supplies the input pressure, and two outlet ports that connect to flow control valves on either side of the pneumatic cylinder. The gripper's opening and closing actions are contingent upon the digital input signals received by the solenoid valve and the corresponding pneumatic cylinder connections. Figure 3.4 provides a schematic diagram illustrating the electrical and pneumatic connections of the solenoid valve and the robot. Upon completing the mechanical assembly of the gripper to the robot and establishing the necessary electrical and pneumatic connections, as illustrated in Figure 3.5, the gripper was successfully integrated into the robotic system. This enabled the gripper to be operated through the robot's teach pendant, demonstrating the compatibility of the custom-designed gripper within the larger robotic system.

To thoroughly evaluate the gripping stability of the custom-designed gripper, a series of tests were conducted using a stack of 200 envelopes. The primary objective

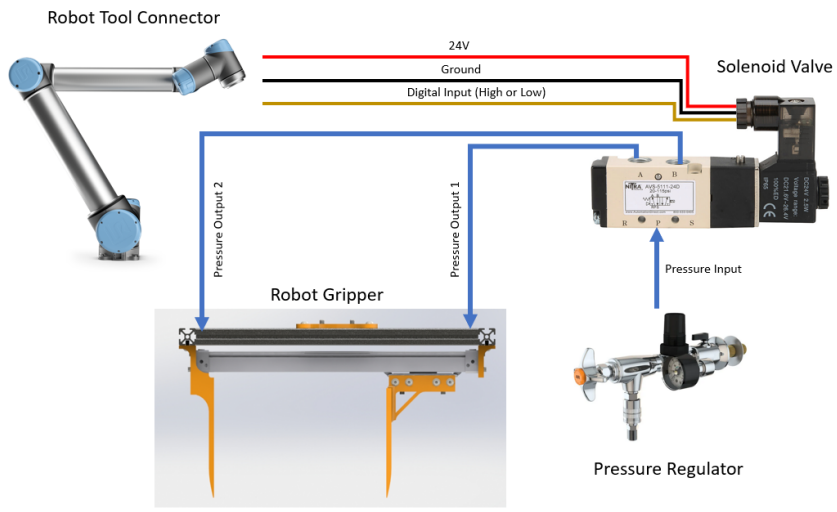

Figure 3.4: Schematic layout of electrical and pneumatic gripper connections

of these tests was to determine the gripper's ability to securely grasp and maintain its hold on the envelopes during various motion profiles executed by the robot. During these tests, the gripper jaws maintained contact with the envelopes in a square surface area of 7cm x 7cm.

Initially, the robot was tasked with grasping the stack of envelopes and executing a series of linear motions repeatedly 20 times. Throughout this test, the gripper's performance was closely monitored for any signs of instability, such as envelope slippage or unintended disturbances to the grasped stack. The linear motion test was considered successful when the robot was able to perform the motion at full speed without causing any disruptions to the gripped envelopes.

Following the successful completion of the linear motion tests, the robot was programmed to perform an oscillatory circular motion 20 times, emulating the behavior of a pendulum. This test aimed to assess the gripper's stability under dynamic conditions, which often involve higher forces and accelerations compared to linear movements. The pendulum-like motion subjected the gripped envelope stack to a range of accelerations, challenging the gripper's ability to maintain a secure grasp.

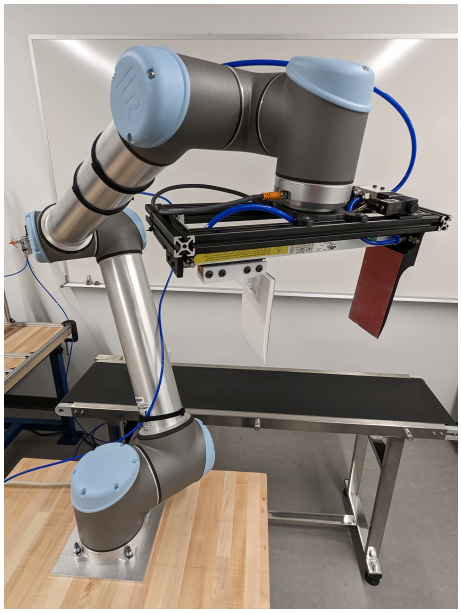

Figure 3.5: Hardware integrated gripper with electrical and pneumatic connections

Throughout both the linear and oscillatory circular motion tests as shown in Figure 3.6, the gripper was able to securely hold the envelope stack and maintain its integrity.

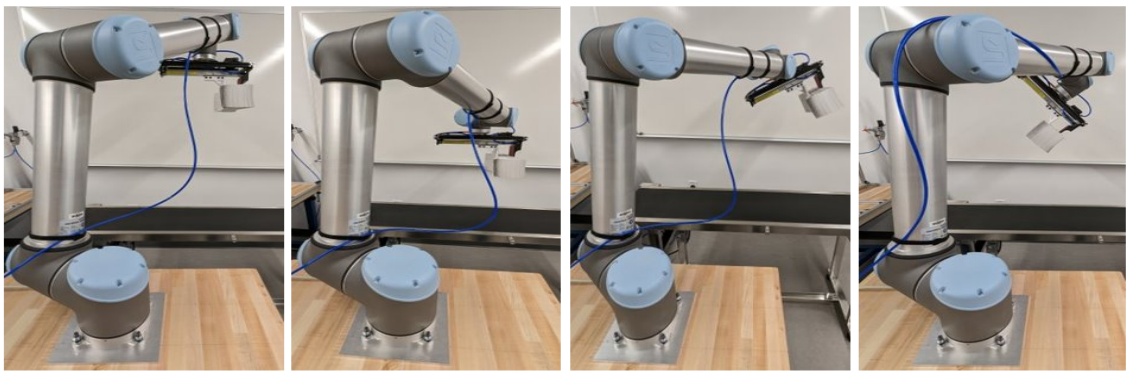

Figure 3.6: Linear motion test: Robot repeating movement from the pose in (a) to pose in (b) and vice versa; Circular oscillatory motion test: Robot repeating movement from the pose in (c) to pose in (d) and vice versa

Chapter 4

Envelope Stack Detection and Localization

This chapter presents the methodology and techniques employed for detecting envelope stacks' notches, dimensions, and tracking their movement on the conveyor belt. The key components of this process include image acquisition, pre-processing, notch detection, envelope stack segmentation, dimension identification, and tracking.

4.1 Notch Detection in Simulation

Simulation environments provide an effective platform for testing and refining perception algorithms while mitigating the risks and costs associated with hardware experimentation. A simulation of the system was set up in the Gazebo simulator [26] for prototyping and testing notch detection algorithms. An RGB-D camera is mounted above static envelopes placed on a table as shown in Figure 4.1. The virtual camera is simulated using the Realsense plugin with the same intrinsic, field of view, resolution, etc., as the hardware camera. The envelope stacks are approxi-

mately modeled as cuboids with notch structures as shown in Figure 4.2. The virtual camera acquires both RGB and Depth images from the environment and publishes them in separate streams.

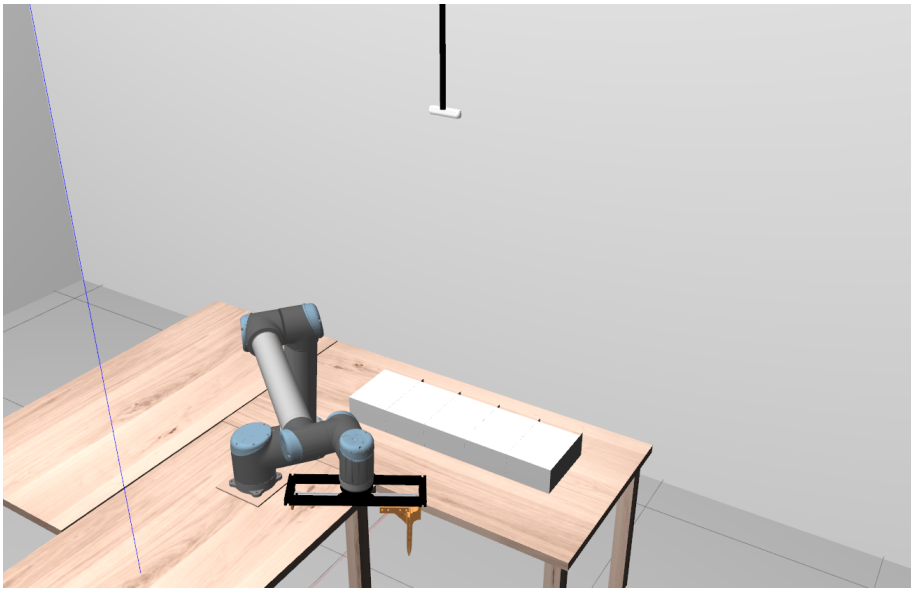

Figure 4.1: Simulation of a virtual RGB-D camera mounted from the ceiling which captures images of envelope stacks

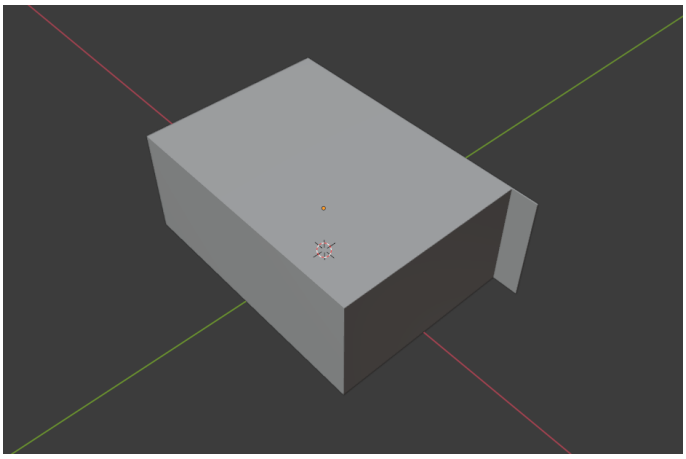

Figure 4.2: CAD model of an envelope stack used in the Gazebo simulation

The ceiling-mounted camera provides a top-view perspective of the envelope stacks, which plays a crucial role in detecting the notches that mark the end of each stack. However, this perspective introduces a unique challenge in accurately determining the position of the notches. Depending on their location relative to the center of the image, the true position of the notches varies. Moreover, even though the notches are vertically aligned and straight, their radial distance from the camera causes the sides of the notches to be visible, resulting in a trapezoidal shape in the captured RGB image rather than a thin rectangle as shown in Figure 4.3. This effect is most pronounced for notches located further from the center, while those closer to the center appear as thin rectangles from the camera's perspective.

Figure 4.3: Sides of notches forming trapezoidal shapes from the RGB camera's perspective

To address this challenge, perspective transformations can be a possible solution. Perspective transformations when applied to the captured images, can effectively compensate for the distortion and enable more accurate notch detection. Xin

Li et al. [32] proposed a new method for rectifying image deviation in circular instruments using perspective transformation on images preprocessed by the Canny operator that detects equipment areas and computes regional parameters. While perspective transformations can give good results, they rely on accurate camera parameters, such as intrinsic and extrinsic matrices to correct for the distortions in the image. Any error in these parameters can lead to inaccuracies in the transformed image, which may affect the notch detection performance. Moreover, perspective transformations might not perform well under varying lighting conditions or when the envelope surface has significant reflections or irregularities. A much simpler and more accurate solution can be considered for this problem instead.

The perspective of the notches that are away from the center of the RGB camera is always such that the actual notch position is at the radially most distant edge of the trapezium. This can be seen in Figure 4.4 where the actual position of the notches (highlighted in red) to the left of the image center is at the leftmost edge of the trapezium formed from the camera's perspective and the rightmost edge for notches to the right of the image center. This logic can be used to determine the true position of the notches once they are segmented.

The notch detection in the simulation was carried out in 4 main steps:

- **Color Thresholding:** Color thresholding is a widely used image segmentation technique that separates regions of an image based on specific color ranges. In this simulation setup, the envelopes are spawned as white blocks for simplicity. Before performing color thresholding, a region of interest (ROI) was defined to get only the envelopes and the supporting surface. This ROI is then converted from RGB to HSV color space. To segment out the white envelopes, a mask was created by setting all pixels in the HSV range of (0,0,0) to (0,0,255) to white as shown in Figure 4.5.

Figure 4.4: Actual notch positions highlighted in red - leftmost edge for notches to the left and rightmost edge for notches to the right of the image center

- **Edge Detection:** Edge detection is a crucial step for identifying the boundaries of the notches accurately. This process allows the system to recognize the trapezoidal or rectangular shapes corresponding to the notches, and subsequently track their positions. Once a mask of envelopes generated using color thresholding was isolated from the RGB image, Canny Edge Detection [6] was used to identify the edges as shown in Figure 4.6.

- **Line Segment Detection:** After getting the edges in the mask, line segment detection can be employed to locate the straight edges of the notches. The probabilistic Hough Transform [25] was employed to get the line segments from the edge mask. The straight line segments were then filtered on the basis of slope - only retaining lines with $x_2 - x_1 \approx 0$ where the endpoints are given by (x_1, y_1), (x_2, y_2) pixel coordinates. Here x coordinates are column values and

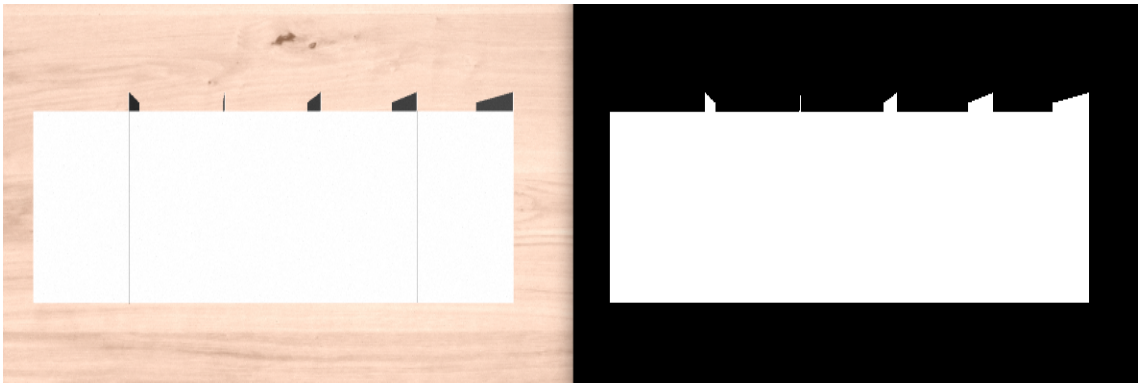

Figure 4.5: RGB Region of Interest (left) and color thresholding mask (right)

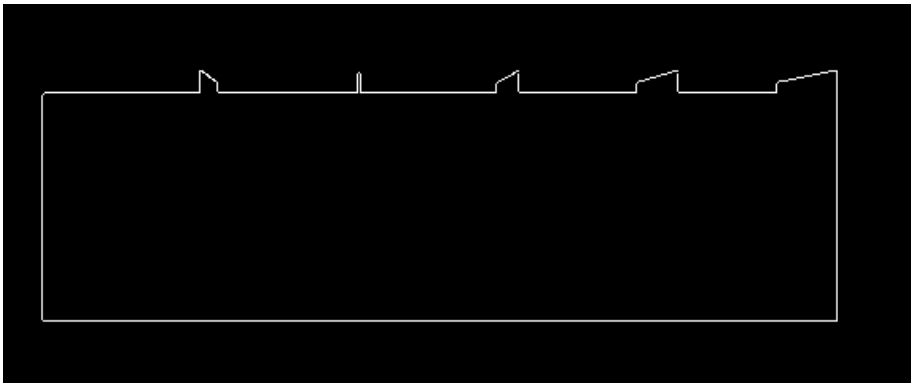

Figure 4.6: Edge detection using Canny Edge Detection

y coordinates are row values. Figure 4.7 depicts these filtered lines highlighted in red.

- **Notch and Stack Detection:** These filtered line segments are iteratively traversed through and grouped into "notch pairs" based on a fixed maximum distance (50 pixels in this case). This maximum distance can be determined by having notches at the extreme edges of the ROI and getting the greater pixel distance between the parallel edges of the trapeziums formed. Two line segments are notch pairs if they form the parallel sides of the trapezium or rectangle shape of the notches. These notch pairs are iterated through, and based on their x values the true notch line segments are identified. Based

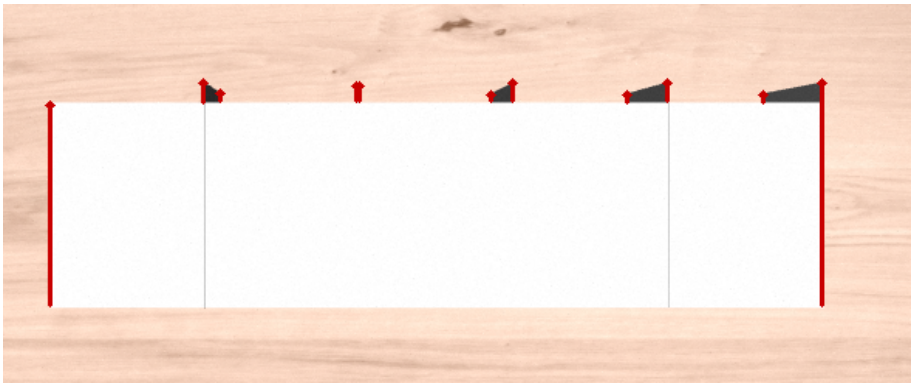

Figure 4.7: Line segment detection and filtering on the basis of slope (highlighted in red)

on these true notch positions, bounding boxes are created for each envelope stack as shown in Figure 4.8. These bounding boxes are used by the motion planning pipeline to keep track of the envelope stack lengths while packaging.

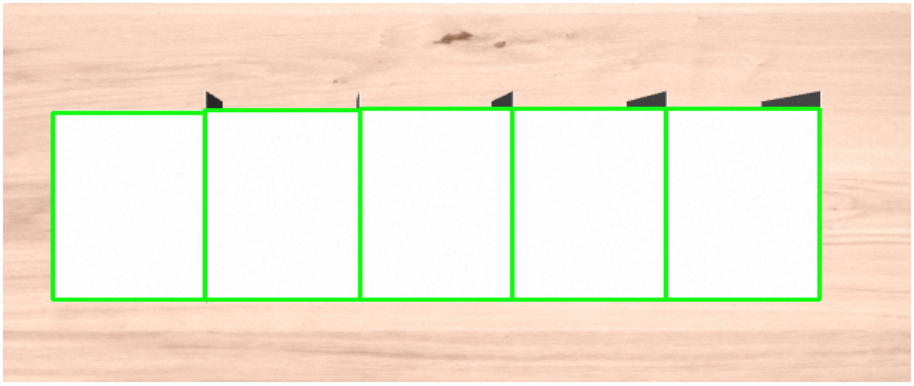

Figure 4.8: Bounding boxes (green) identified using the detected notches

4.2 Notch Detection in Hardware

Hardware perception plays a critical role in the successful detection and localization of notches for accurate slicing, grasping, picking, and placing of envelope stacks by the robot. Transferring perception methodologies from simulation to hardware

presents several challenges that need to be addressed to ensure the successful implementation of a robotic system. These challenges arise due to the differences between simulated and real-world environments, as well as the limitations of hardware sensors. Some of the key challenges include:

- **Sensor Noise:** Real-world sensors, such as RGB-D cameras, are subject to noise, which can significantly affect the accuracy and reliability of the perception algorithms. Sensor noise can be caused by various factors, including temperature fluctuations, electronic interference, and manufacturing imperfections. In contrast, simulated environments typically produce clean and noise-free data, which may not adequately represent the challenges faced in real-world scenarios.

- **Limited Accuracy at Greater Distances:** In many cases, depth sensors struggle to accurately capture point clouds of objects located more than 1 meter away. This limitation can make it challenging to detect notches in envelope stacks when the robotic system is operating at a distance from the conveyor belt. While simulation environments may not fully account for these distance-related inaccuracies, they must be considered when transferring perception methodologies to hardware systems.

- **Real-World Lighting Conditions:** Simulation environments often assume ideal and uniform lighting conditions, which rarely exist in real-world settings. In practice, lighting conditions can vary significantly due to shadows, reflections, and ambient light changes. These variations can cause problems in the detection and segmentation of objects, particularly when relying on RGB images. To address this challenge, it is crucial to develop perception algorithms that are robust to varying lighting conditions and can adapt to the dynamic

nature of real-world environments.

A primary factor for utilizing RGB and depth images separately, rather than relying on point clouds for notch detection and localization stems from the inherent inaccuracy of hardware sensors. Notches, being quite thin, pose a challenge for RGB-D cameras to reliably detect and identify when positioned at a height of 1 meter or more from the envelopes. This limitation arises due to the diminished resolution and increased noise in the depth data as the distance between the camera and the notches increases. Consequently, the point cloud representation may not accurately capture the finer details of the notches, leading to less reliable detection.

One potential solution to this issue involves mounting the camera directly onto the robot, which would allow for closer proximity to the notches during detection. However, this approach introduces a new set of challenges, as it could impede the robot's ability to place envelopes in narrow spaces. The increased risk of collisions due to the additional hardware on the robot could compromise the efficiency and safety of the overall system. Figure 4.9 shows a comparison between the depth images acquired from the simulation and hardware cameras.

Figure 4.9: Depth images acquired in simulation (left) and hardware (right)

Aligning depth images with color images in an RGB-D camera is necessary for ensuring that the data from both sources are accurately combined for further processing. This process is called "registration". Registration between color and depth

images is done by obtaining extrinsic parameters, depth image rectification, projecting depth data into the color image, and interpolating missing data. The relative transformation (rotation and translation) between the depth and color cameras represents the extrinsic parameters that are offered by most sensor manufacturers. Using the intrinsic and extrinsic parameters, the depth image is rectified to match the color image's coordinate system. For each pixel in the rectified depth image, its corresponding 3D point is projected into the color image's coordinate system using the extrinsic and intrinsic parameters of both cameras. During the projection of depth data into the color image's coordinate system, some pixels may not have corresponding depth values due to occlusions or differences in the camera's fields of view. Interpolation techniques (e.g., nearest-neighbor or bilinear interpolation) are used to fill in the missing depth values. This step results in a new depth image that is aligned with the color image.

Depending on the environment and system setup, the ROI of the aligned color and depth images must be set to include all the regions where the envelopes can be viewed and tracked. The notch detection in hardware was carried out in 5 main steps:

1. **Envelope Segmentation:** Since the envelope stacks form a well-defined surface on top, we can approximate the overall height of the collective stacks using the minimum depth from the depth images. The ROI is first defined to only include the regions where envelopes can be located. Using the depth images, the envelope stacks were segmented by using minimum depth filtering to create a binary mask with only the corresponding pixels. A bounding box of the mask is then processed to get contours using Suzuki's Contour tracing algorithm [44] and drawn in the aligned color image ROI as shown in Figure 4.10.

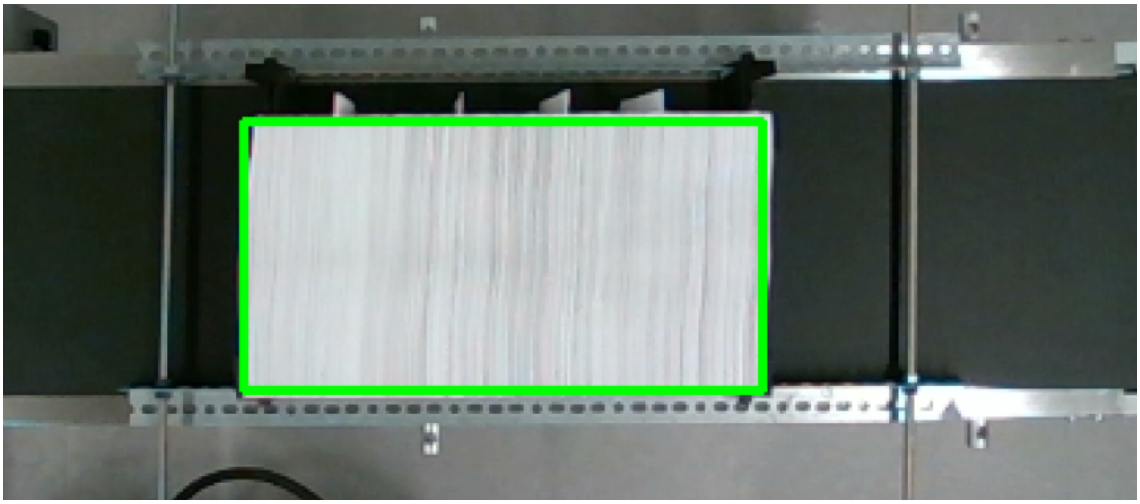

Figure 4.10: Envelope segmenting using depth image filtering

2. **Envelope and Notch Isolation:** Using the corners of the bounding box acquired in the previous step, the envelopes and the notches are isolated using image slicing of the original color image. Assuming the notches are always at a fixed length, an offset to extend the bounding box region from the previous step by a few pixels on top is found by trial and error. Similarly, the bounding box region is slightly clipped from the sides to remove any inaccuracies caused by the envelope-holding mechanisms. This isolation of envelopes and notches is done to remove the unnecessary background which can be susceptible to more noise from other objects such as glare from highly reflective surfaces (seen in Figure 4.10). The resulting isolated envelopes with notches are shown in Figure 4.11(a).

3. **Image Filtering:** Filtering RGB images in hardware is an important preprocessing step to improve the accuracy of the notch segmentation process. Hardware-based filtering can reduce noise, enhance color contrast, and improve the overall quality of the RGB images captured by the camera. The isolated envelopes with notches are preprocessed using these 3 filters in the

order of appearance:

(a) **Gaussian Blur Filter:** The Gaussian blur filter was employed to smoothen the ROI further by convolving the image with a Gaussian function, which reduces high-frequency noise and smoothens the edges. To apply the Gaussian blur filter, a kernel is generated using the Gaussian function with a specified σ value, which is the standard deviation that controls the amount of blurring applied. This kernel is then convolved with the input image, effectively averaging the pixel intensities in the neighborhood defined by the kernel. The Gaussian function can be expressed as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where x and y are the pixel coordinates.

(b) **Bilateral Filter:** The bilateral filter [45] was employed to reduce noise in the RGB images while preserving edges, prior to color thresholding. The bilateral filter is a non-linear filter that combines domain filtering and range filtering, thus offering the advantage of both smoothing the image and preserving edges. The bilateral filter is defined as:

$$BF(I)(x) = \frac{1}{W_p} \sum_{y \in S} G_s(||x - y||) G_r(||I(x) - I(y)||) I(y)$$

where x is the target pixel, y is a neighboring pixel in the spatial neighborhood S, G_s is a Gaussian function for spatial distances, G_r is a Gaussian function for intensity differences, $I(x)$ represents the intensity value of the input image at pixel location x, $I(y)$ represents the intensity value of the input image at a neighboring pixel location y, and W_p is a normalization

factor to ensure that the weights sum up to 1:

$$W_p = \sum_{y \in S} G_s(||x - y||) G_r(||I(x) - I(y)||)$$

The spatial Gaussian function G_s measures the distance between pixels and is defined as:

$$G_s(||x - y||) = e^{-\frac{||x-y||^2}{2\sigma_s^2}}$$

where σ_s is the standard deviation for the spatial domain.

The range Gaussian function G_r measures the difference in intensities and is defined as:

$$G_r(||I(x) - I(y)||) = e^{-\frac{||I(x)-I(y)||^2}{2\sigma_r^2}}$$

where σ_r is the standard deviation for the intensity domain. The bilateral filter calculates a weighted average of neighboring pixel intensities, with weights depending on both the spatial distance and the intensity difference. This ensures that only similar pixels in close proximity contribute to the output, which results in effective noise reduction while preserving edges.

(c) **Blur Filter:** This filter blurs the image using the normalized box filter. The normalized box filter, also known as the mean filter or average filter, works by averaging the color intensity values of neighboring pixels within a specified kernel size. This process effectively reduces high-frequency noise while preserving the overall structure of the image. This filter

smooths the image using the kernel:

$$K = \frac{1}{\text{kernel width} \times \text{kernel height}} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

The resulting image from applying these 3 filters is shown in Figure 4.11 (b).

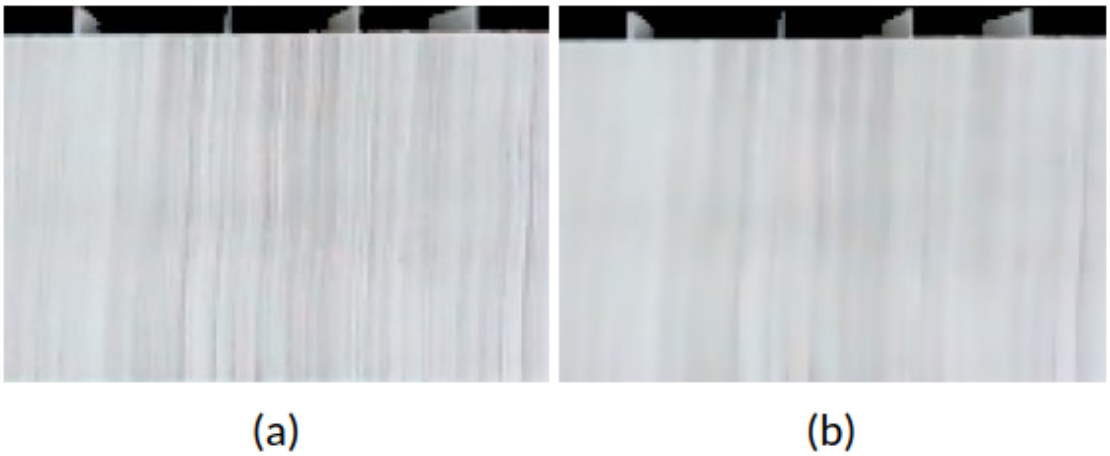

Figure 4.11: (a) Isolated envelopes with notches; (b) Result of applying filters to the isolated stacks

4. **Color Thresholding:** Similar to the color thresholding step in simulation perception, a binary mask of the envelopes and notches is created as shown in Figure 4.12.

5. **Notch Detection:** In the binary mask generated from the previous step, the notches can be easily identified by analyzing a row of pixels at a small pixel distance from the top. This row of pixels is iteratively checked for changes in color. Each time there is a change in color, it denotes the start or end of a

Figure 4.12: Mask created from color thresholding

notch as shown in Figure 4.13(a). Using a threshold distance between notches similar to the final step in simulation notch identification, these points are grouped together as notch points. If these pairs of notch points are to the left of the color image center, the true notch location is the leftmost pixel of the pairs, and vice versa for notches to the right of the center. The true notch points have been plotted in Figure 4.13(b) and highlighted as yellow circles on the actual color image as shown in Figure 4.14.

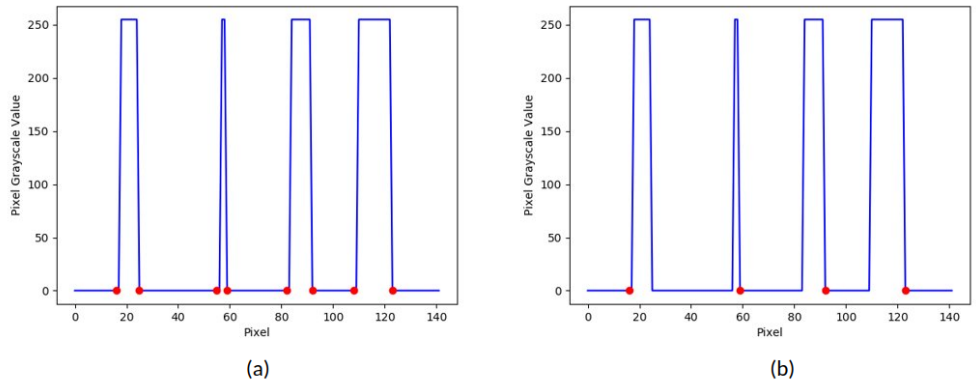

Figure 4.13: Plots of detected (a) pairs of notch points; (b) true notch points in mask

Figure 4.14: True notch locations highlighted as yellow circles

4.3 Notch Position and Stack Dimensions Identification

After successfully retrieving the notch positions in the color image space as shown in the previous sections, the real-world position of the notches with respect to the camera must be determined. This can be achieved using the intrinsics of the calibrated camera and corresponding pixel depths as follows:

1. **Depth Retrieval:** The depth value at the corresponding pixel coordinates of notches in the RGB image must be retrieved from the aligned depth image. The problem with this approach is that the depth image in the hardware is not precise enough to get the notch depths. However, since all envelopes are of the same height, the depth of the centroid of the bounding box (in Figure 4.10) was used for the same purpose.

2. **Conversion to Normalized Image Coordinates:** The pixel coordinates (u, v) and the depth value (z) are converted to normalized image coordinates

(x', y') using the camera's intrinsic parameters as:

$$x' = (u - cx)/fx$$
$$y' = (v - cy)/fy$$

where (cx, cy) are the principal points (image center coordinates), and (fx, fy) are the focal lengths in the x and y directions, respectively.

3. **Real Position from Camera:** The real-world coordinates (X, Y, Z) are computed using the depth value (z) and the normalized image coordinates (x', y'):

$$X = x' * z$$
$$Y = y' * z$$
$$Z = z$$

The resulting (X, Y, Z) coordinates represent the 3D position of the notches in the real world with respect to the camera's color sensor frame corresponding to the original pixel coordinates (u, v) in the color image.

The positions of the notches must be transformed into the world frame (which will be used as the reference plane for motion planning) from the RGB camera's frame using transformations. If W represents the world coordinate frame and C represents the color camera's frame, then the transformation from the world to the color camera is given by ${}^{W}T_{C}$ which is a 4×4 homogeneous transformation matrix known from the environment setup. Let P_C be the 4×1 position of the notch with respect to the camera frame with the first 3 elements being the X, Y, and Z values obtained above and 1 being the last value. Then the position of the notch with

respect to the world frame P_W is given by:

$$P_W = {}^{W}T_C * P_C$$

Similarly, the envelope stack dimensions are retrieved using the notch coordinates and the bounding box parameters(shown in Figure 4.10) as follows:

- The length of the stacks is calculated by getting the real-world position of the leftmost (or rightmost) corners of the bounding box with respect to the camera and finding the difference between the coordinates in the corresponding color camera frame's axis.

- The width (thickness) of the first stack is calculated by getting the real-world position of the top left corner of the bounding box and the first notch with respect to the camera and finding the difference between the coordinates in the corresponding direction. Similarly, for the subsequent envelope stacks, the distance between the corresponding real-world coordinates of the notches gives the stack thickness. This dimension is particularly important for the robotic system to keep track of the stack thicknesses while packaging into cardboard boxes.

- The height of the envelopes can be found using the depth of the center of the bounding box and the depth of a pixel on the supporting surface. The supporting surface pixel's row coordinate can be the same as that of the bounding box center, and its column coordinate is taken a few pixels to the left of the leftmost edge of the bounding box in the color image. The difference between these depths multiplied by the depth scale of the camera gives the envelopes' height.

- For the simulation, the envelopes' height is calculated the same way as the hardware, with the only difference being the notch depths were accurate in the virtual camera's depth images and can be directly used. The length and breadth of each stack can be computed similarly by using the real-world coordinates of the bounding box corners of each stack (shown in Figure 4.8).

This position for each notch along with the stack dimensions is streamed to the motion planning pipeline which uses the position, dimensions, and time information to accurately pick and place the envelope stacks. More information about this will be discussed in the next chapter.

This thesis chapter has presented a comprehensive methodology for envelope stack detection and localization. The techniques for notch detection and envelope stacks' dimension calculation in both simulation and hardware environments were discussed. Notch detection in simulation can be more applicable to the robotic system if the real sensor data is more precise and less noisy. The methodologies used for hardware perception are designed to be robust to noisy and inaccurate sensor data.

Chapter 5

Motion Planning and Servoing

In this chapter, we delve into the intricacies of motion planning and servoing for the purpose of envelope stack manipulation. A key aspect of any robotic manipulation task is the ability to plan and execute smooth, collision-free trajectories while maintaining a stable grasp and control with the end-effector. The proposed approach combines motion planning techniques with robust servoing strategies, ensuring precise and efficient pick and place of envelope stacks. We begin by providing an overview of the motion planning algorithms that were considered, followed by a discussion on the implementation of servoing methods for the project. Special emphasis is placed on addressing the unique challenges posed by the specific manipulator, gripper, and task requirements.

Figure 5.1 provides a high-level overview of the envelope packaging process once the perception pipeline is ready to stream notch positions and envelope stack dimensions. For dynamic envelope grasping from a conveyor belt, the robot must first plan a path toward the envelopes to get ready to isolate the stack and proceed to pick it. Once the robot executes path planning to get close to the notch, it switches to a velocity controller in order to probe the notch and slice the envelopes. To achieve

high accuracy and smooth isolation of the envelopes using the slender fixed jaw of the gripper, cartesian servoing methods are used to perform the required actions. After slicing through the envelopes, the robot must also slide the notch into the envelope stack to prevent damage to the notch envelopes while placing the stack into the box. For obtaining a stable grasp of the envelopes, the gripper needs 1-2 seconds to apply full pressure onto the stacks. Hence the robot has to keep matching the conveyor velocity until it has a stable grasp and retracts.

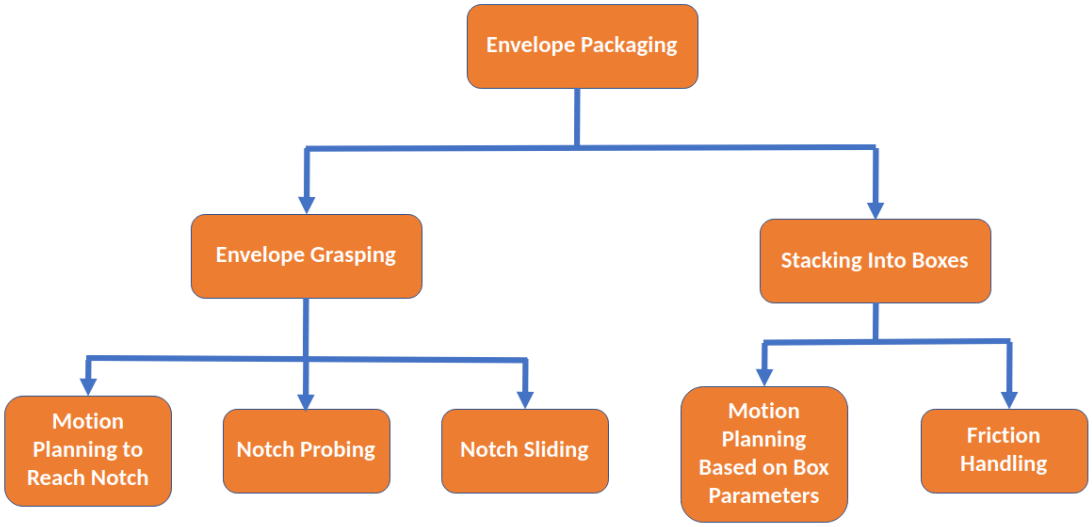

Figure 5.1: High-level overview of motion planning and servoing processes for envelope packaging from a conveyor belt

The next step in envelope packaging is stacking the grasped envelopes into cardboard boxes. The robot has to generate trajectories to reach the box location and match the box's position and orientation while placing envelopes inside. Once the envelope stack is placed inside and the gripper opens, there are two challenges that arise:

- **Envelope toppling:** Envelopes in a stack tend to expand when compressed and released, especially when placed in an upright cardboard box. While it can be solved by a placeholder that can be used to temporarily hold together

the stack upright, an easier solution is to mount the cardboard box at an orientation as shown in Figure 5.2. This makes the motion planning problem for stacking slightly more complicated but is a much simpler way of preventing envelopes from falling over.

Figure 5.2: Cardboard box inclined at an angle using a simple mechanical structure

- **Envelope adhesion from friction:** The gripper jaws have a good coefficient of friction which is important for achieving a good grasp. However, this leads to a disadvantage after placing the envelopes into the box when the robot starts retracting to pick the next stack. The envelope in contact with the jaw tends to stick with it and move up alongside the gripper when the robot retracts. In order to overcome this challenge, the robot can be pre-programmed to execute some motions that enable it to slowly retract while overcoming the frictional force. These motions will be explored later in this chapter.

At the start of each pick and place cycle, the motion planning pipeline tracks the conveyor velocity by calculating the change in the position of notches with time. If the envelopes start moving at a non-zero speed, the robot needs a reliable planner to generate optimal collision-free trajectories quickly and reliably. The motion planners

must be fast and accurate in order to reach the notch on time and slice into the stack. Multiple sampling-based algorithms like RRT, RRT-Connect, RRT*, and PRM, as well as trajectory optimization algorithms like CHOMP [39] and STOMP [22], were considered for this task. Sampling-based planners tend to take excessive time to plan paths, especially in collision-rich environments, and can be suboptimal. Both CHOMP and STOMP are optimization-based planners that try to find a collision-free path with a minimum cost. However, they can get stuck in local minima and fail to find a globally optimal solution, especially in highly constrained environments. Both planners use iterative optimization techniques, and the maximum number of iterations or the convergence tolerance can influence their success in finding a valid path. If the allowed iterations are insufficient, or the convergence tolerance is too strict, the planner might not find a valid path within the given constraints. A planner that can generate deterministic and repeatable trajectories, which can ensure consistent and predictable robot behavior would be more applicable to such structured industrial environments.

Linear and circular trajectories in the Cartesian space that are precise and repeatable will be useful for rapid initial approach to the notches and pick-and-place tasks. Such linear trajectories can be produced using linear interpolation between the start (P_{start}) and end (P_{end}) positions, and quaternion slerp between the start (q_1) and end (q_2) rotations. Linear interpolation is computed as follows:

$$P(t) = P_{start} + (P_{end} - P_{start}) * t/T$$

where $P(t)$ is the linear interpolated position at time t, T is the time to complete the motion, and $0 <= t <= T$.

Quaternion slerp is used to smoothly interpolate between two quaternions, which

represent the starting and ending orientations in 3D space. Given two quaternions, q_1 and q_2, representing the starting and ending orientations, quaternion slerp (spherical linear interpolation) can be computed as follows:

1. Calculate the dot product of the two quaternions:

$$\cos(\theta) = dot(q_1, q_2)$$

2. If $\cos(\theta)$ is negative, reverse one of the quaternions to ensure the shortest path is taken:

$$q_2 = -q_2$$

$$\cos(\theta) = -\cos(\theta)$$

3. Calculate the interpolation parameter, t, which represents the fraction of the total distance between the two quaternions required to interpolate ($0 <= t <= 1$). A lower t results in more waypoints in the trajectory. Since the motion is following a circular trajectory, the value of t will be adjusted based on the current position along the circular path.

4. Compute the slerp:

$$\sin(\theta) = \text{sqrt}(1 - \cos^2(\theta))$$

$$\theta = \text{atan2}(\sin(\theta), \cos(\theta))$$

$$\text{slerp} = (\sin((1-t) * \theta) / \sin\theta) * q_1 + sin(t * \theta) / \sin(\theta) * q_2$$

A circular trajectory is a path that follows the arc of a circle between two points in the Cartesian space. To generate a circular trajectory, we need the start point P_{start}, end point P_{end}, center point of the circle P_{center}, and time to complete the

motion (T). It is computed as follows:

1. Compute the radius (r) and the angle (θ) of the circular trajectory:

$$r = ||P_{start} - P_{center}||$$

$$\theta = acos(dot((P_{start} - P_{center}), (P_{end} - P_{center}))/(r^2))$$

2. To interpolate the positions along the circular trajectory, the angular velocity (ω) is calculated and at each time step (t), the position is updated:

$$\omega = \theta/T$$

3. For each time step t ($0 <= t <= T$), the planner computes the current angle (θ_t):

$$\theta_t = \omega * t$$

Finally, the position at time t ($P(t)$) is calculated using the following equation:

$$P(t) = P_{center} + r * R_z(\theta_t) * (P_{start} - P_{center})$$

where $R_z(\theta_t)$ is a rotation matrix about the z-axis by angle θ_t.

The quaternion slerp for circular trajectories is calculated the same way as mentioned in linear trajectories. The main drawback of using this method of waypoint generation is the lack of collision avoidance capabilities and the ability to handle more dynamic scenarios. However, in a structured workcell environment with known collisions, this should work without any issues.

For generating the joint trajectories in this application, the KDL inverse kine-

matics solver in the MoveIt! [7] motion planning framework is used to determine joint positions for the interpolated points. The KDL solver is based on the Newton-Raphson [48] method and uses the Jacobian pseudoinverse to iteratively update joint angles. The Pilz industrial planner is used to minimize the total execution time while adhering to the robot's kinematic constraints, such as joint limits, maximum velocities, and accelerations. Trajectory execution on the robot is implemented using MoveIt, which has postprocessing collision-checking algorithms that abort the execution when any self-collisions or collisions with the environment and the robot are found. Hence the motion planning and execution framework is "collision-aware", but cannot plan for collision avoidance. This makes the system safe from any major collisions in a structured workcell.

In the pick and place cycle, after the robot executes motion plans to reach the back of the moving notch, it has to probe it until the notch deforms by a fixed length and isolates the stack for slicing in with its static jaw. For such accurate and seamless task requirements, Cartesian velocity servoing is effective. Cartesian velocity servoing allows the robotic system to follow a specified velocity profile in the Cartesian space, resulting in precise control over the end-effector's position and orientation. It also allows for smooth motion of the end-effector, which is essential for delicate operations like probing notches and slicing envelopes. Smooth motion reduces the risk of damaging the envelopes or causing the stack to shift during the operation.

Cartesian velocity control can be achieved with the following steps:

1. Get the desired velocity in the robot's base frame. If the velocity is in any other frame X, use Adjoint transformations to transform the velocity to base frame:

$$V_{base} = \text{Adjoint}(T_{base\_X}) * V_X$$

where V_{base} and V_X are 6D twist vectors (3D linear velocity and 3D angular velocity) in the robot's base and X frames, respectively, T_{base_X} is the transformation matrix representing the frame X's pose in the base frame, and `Adjoint()` represents the adjoint transformation.

2. Compute the Jacobian matrix, which relates the joint velocities ($\dot{q}$) to the Cartesian velocities (V_{base}) of the end-effector:

$$V_{base} = J(q) * \dot{q}$$

where $J(q)$ is the Jacobian matrix as a function of the current joint configuration q.

3. Calculate the pseudo-inverse of the Jacobian matrix, J^+:

$$J^+ = (J^T * J + \lambda^2 * I)^{(-1)} * J^T$$

where J^T is the transpose of the Jacobian, λ is a damping factor to improve the numerical stability of the inversion (and mitigate potential singularities), and I is the identity matrix.

4. Compute the joint velocities required to achieve the desired Cartesian velocities:

$$\dot{q} = J^+ * V_{base}$$

The `moveit_servo` ROS package was used to execute precise and smooth Cartesian velocity commands. Furthermore, this package can also integrate with MoveIt's collision avoidance capabilities to ensure that the robot's trajectory is collision-free.

For performing pick and place, it is assumed that the packaging box's location,

angle of inclination, and dimensions are already known. The first envelope stack's dimensions are streamed from the perception pipeline. Using the motion planning and servoing methods described above, the following steps describe the pick and place cycle for picking each stack from the conveyor and placing it into a box:

1. Set controller to position control and plan and execute a linear trajectory to move to a predetermined pose that does not occlude the envelopes from the camera

2. Initialize the filled length inside the box to 0

3. While the box is not completely filled, keep tracking for notches to start moving and the first stack's dimensions

4. If notch movement is detected, track the conveyor velocity. If the stack is thin enough to be placed in the box, open the gripper and start the pick-and-place cycle.

5. Set a desired duration (t_d) for reaching the notch position. This means desired system time to reach behind notch is:

$$t_1 = \text{current time} + t_d$$

6. Move in a linear path to a hovering pose directly above the position the notch will reach after t_d seconds.

7. Plan a linear path downward to reach behind the notch when it comes close to the fixed jaw. Let t_{down} be the time it will take to execute this linear trajectory. Wait for $(t_1 - \text{current time} - t_{\text{down}})$ seconds and then execute the trajectory to directly reach behind the notch. Switch to the velocity controller.

8. Now for probing the notch, change the controller to the velocity controller. To get a sense of the direction of servoing for the next steps, let us define a coordinate frame A with an x-axis (A_x) along the direction the robot would slice through the envelopes if they were static, y-axis (A_y) along the direction of conveyor velocity and z-axis (A_z) to pointing vertically downward towards the floor.

9. Set a deformation (d_{def}) of the notch in the direction of conveyor velocity to be achieved by pushing the notch using the fixed jaw. Start servoing the robot in the direction of conveyor velocity (at a higher speed) until it bends the notch by moving ahead of its actual position by d_{def}. This step is called probing since the fixed jaw is used to touch, bend and separate the first stack from the other stacks.

10. After probing, define a certain amount of time (t_{slice}) to slice into the notch. To slice into the stack, the robot has to maintain its velocity component along A_y at the same velocity as the conveyor while traveling about A_x by d_{x_1}, which half of the envelopes' length subtracted by the notch length, and going down along A_z by a few millimeters (d_{z_1}). The A_x and A_z velocity can be calculated as d_{x_1}/t_{slice} and d_{z_1}/t_{slice} respectively.

11. Define an amount of time ($t_{\text{move down}}$) for moving down into the stack to get a good grasp. Move along the A_z axis by a distance d_{z_2} while maintaining conveyor velocity along A_y and 0 velocity along the A_x axis.

12. Now the notch must be slid inside into the envelope stack. Otherwise, the notch envelope might get damaged while being placed into the box. This can be done by defining a $t_{\text{notch sliding}}$ during which the robot must go at a velocity slightly higher than the conveyor speed along the A_y axis and travel the notch

length l_{notch} along the A_x axis. Hence velocity along A_x can be calculated as $l_{\text{notch}}/t_{\text{notch sliding}}$.

13. Close the gripper while matching the speed of the conveyor for about 2 seconds to get a good grasp. After that, change the controller to position control. Retract the gripper.

14. Move towards the cardboard box with the goal pose as the pre-place pose which matches the box's angle of inclination.

15. Move down into the box according to the filled length and place the envelopes.

16. Open the gripper and retract it to move outside the box

17. Move back to the starting pose of the cycle.

Algorithm 1 represents a brief pseudocode that captures the pick-and-place logic of the motion planning pipeline.

For generating waypoints to place the envelope stack into the box after grasping and picking it, a stacking algorithm was created. This algorithm uses information like the box dimensions, angle of inclination, and filled length. If the box has the parameters shown in Figure 5.3 and is located at (cb_x, cb_y, cb_z) with respect to the world frame, any coordinate inside the box can be calculated as:

$$x_{world} = cb_x + (L_1/2 - a) * \cos(|\theta|) + (L_3 - b) * \sin(|\theta|)$$

$$y_{world} = cb_y$$

$$z_{world} = cb_z + (L_3 - b) * \cos(|\theta|) - (L_1/2 - a) * \sin(\theta)$$

The stacking algorithm uses these equations to calculate end-effector poses for sequencing waypoints during envelope placing. For example, when the robot picks

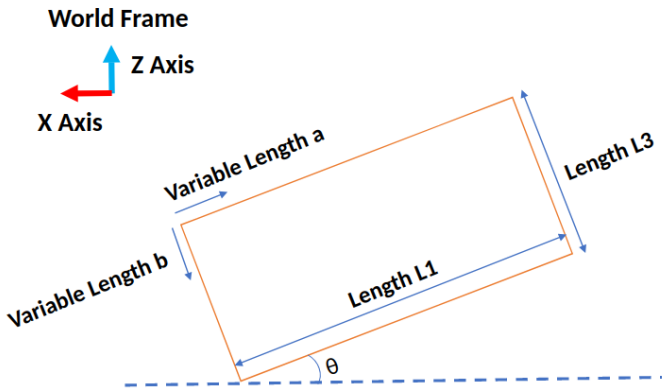

Figure 5.3: A diagram representing parameters used for calculating placing poses

up its second stack, its placing position would be at a=filled length of the stack, and b=height of envelopes.

In the process of placing envelope stacks into the box, a friction problem arises that affects the efficient execution of the robotic task. When the robot opens the gripper and attempts to retract it after placing the envelope stack, the last envelope in contact with the gripper jaw tends to stick to it due to the friction force between the surfaces as shown in Figure 5.6. This unintended adhesion of the envelope to the gripper not only impacts the precision of the robotic system but also disrupts the overall process. To mitigate this issue, various strategies can be considered, such as modifying the gripper material or surface texture, implementing a controlled release strategy, employing methods for envelope separation, or exploring alternative gripper designs.

This friction problem can be solved by slowing down the retracting motion and following a zig-zag pattern using Cartesian velocity servoing. By executing a constant slow retracting velocity along the Z-axis and a constant speed with oscillating directions along X-axis as shown in Figure 5.7, the envelope's adhesive effect due to friction was reduced considerably. This motion can be visualized in Figure 5.8.

Algorithm 1 Pick and place algorithm for moving envelopes

Set controller to position control
Move to a fixed pose (P_{start}), that doesn't occlude the envelopes from the sensor
Filled length inside box = 0
while not completely filled **do**
    Keep tracking for notch movement & first stack length
    if stack thin enough to be placed in the box & movement detected **then**
        Open gripper
        Move to an estimated pose notch will reach in the future (using conveyor velocity)
        Compute, wait and execute a downward motion to reach the back of the notch
        Convert to velocity controller
        Cartesian servoing to probe the notch
        Cartesian servoing to Slice the notch and move down into the stack
        Cartesian servoing to slide in the notch
        Close gripper, grasp & retract vertically upwards
        Move to a pre-place pose calculated using box position, angle of inclination & dimensions
        Placing pose inside the box using filled length, angle of inclination
        Open gripper & retract
        Add stack length to filled length
        Move to P_{start}
    end if
end while

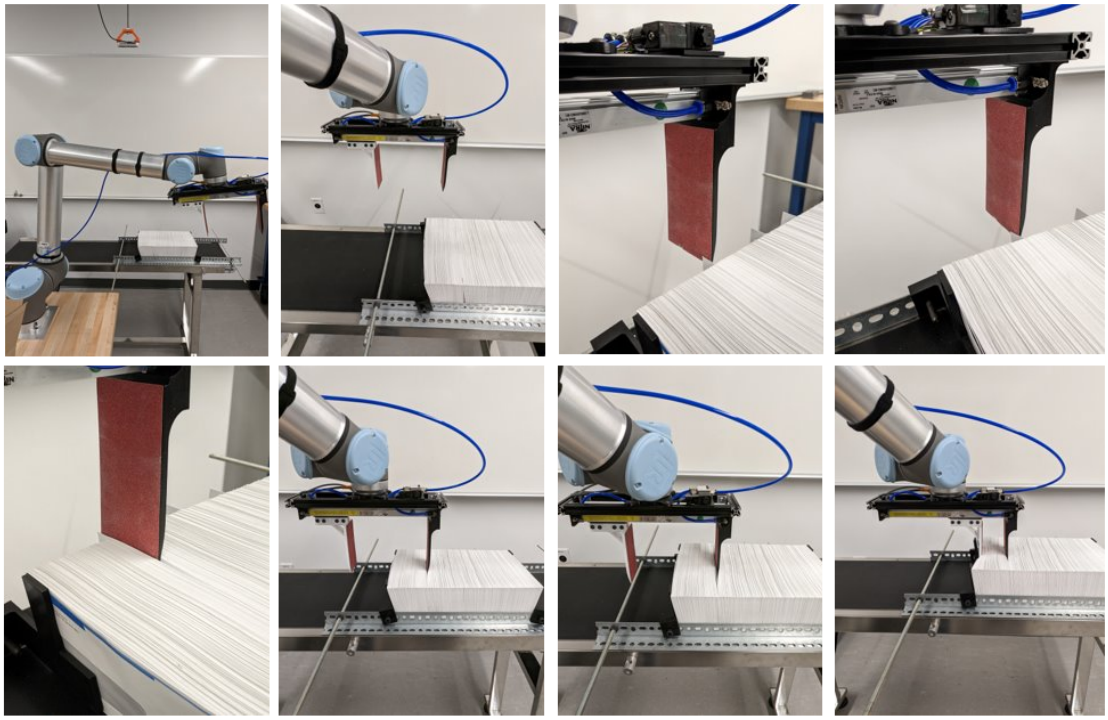

Figure 5.4: Step-by-step depiction of the dynamic envelope stack grasping process

Figure 5.5: Step-by-step depiction of the envelope pick and place

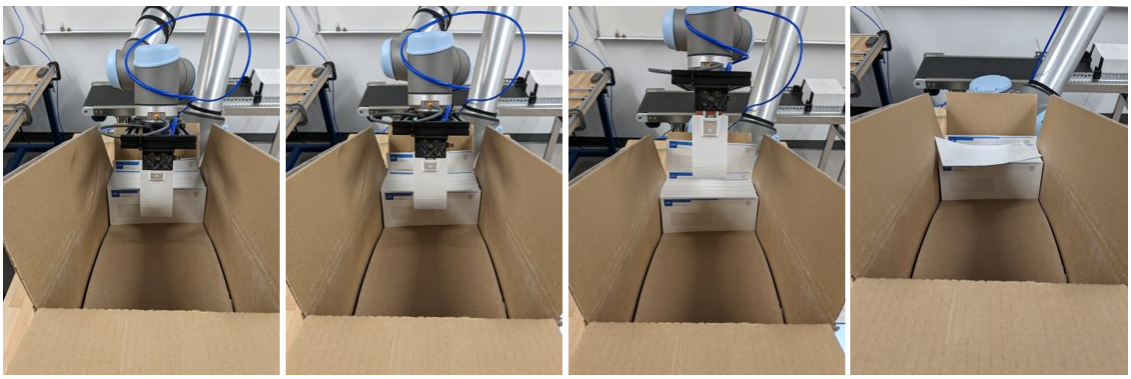

Figure 5.6: Notch envelope in contact with the jaw sliding outside the stack due to friction

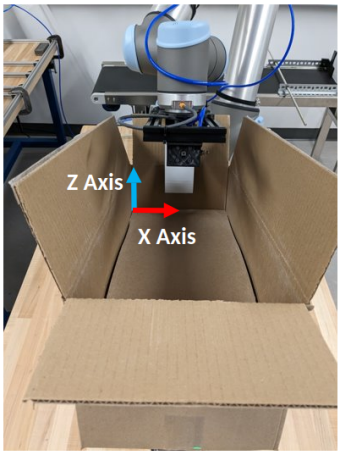

Figure 5.7: Axes for sliding motion from the envelope in contact while retracting

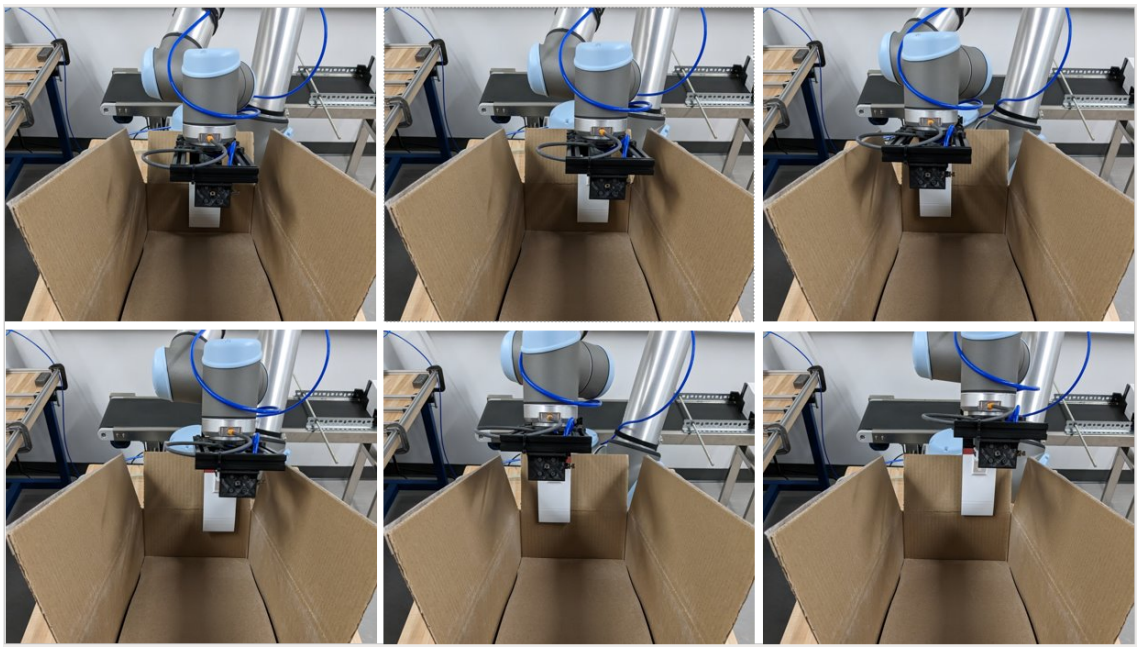

Figure 5.8: Velocity servoing commands executed to avoid adhesive effect of envelope in contact

Chapter 6

Experimentation and Results

6.1 Experiment Setup

This section details the setup of the robotics system, envelope holding mechanism, and packaging box configuration for all the experiments conducted to evaluate the system's performance. The UR10 robot with the integrated gripper is mounted onto a table. A cardboard box for stacking is mounted on the same table behind the robot. The cardboard box is of dimensions (0.568m, 0.254m, and 0.35m) denoting its length, breadth, and height respectively. It is inclined at an angle of 0.3785 radians to prevent envelope stacks from falling over after being placed. The conveyor belt with envelope stacks is located in front of the robot close enough for the robot to plan and execute its motions with ease as shown in Figure 6.1. The Realsense D455 camera was used for perception and was mounted directly above the conveyor belt.

The environment collisions are modeled in MoveIt! using primitive shapes and meshes for trajectory postprocessing and collision checking. The collisions are shown in green while the camera is shown in red in Figure 6.2. The robot is completely connected and controlled using the Robot Operating System (ROS) framework.

Figure 6.1: Hardware setup of the robotic system for experimental analysis

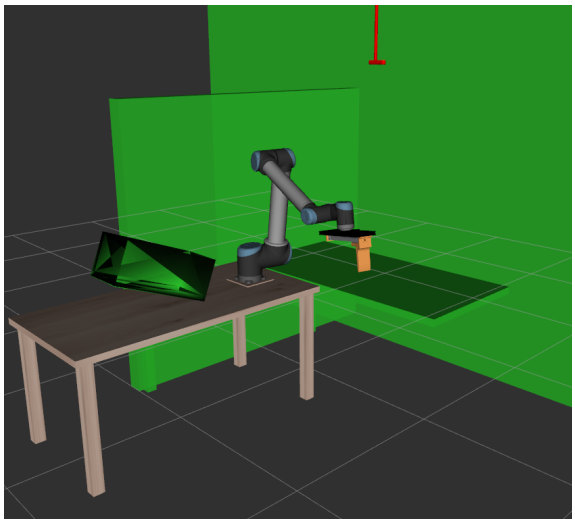

Figure 6.2: MoveIt collision added structured environment

Moreover, the custom gripper's CAD model was used to create a URDF (Unified Robotics Description Format) file which was later used to create a MoveIt! configuration package for this robotic system with a gripper. Moreover, a simple gripper interface was programmed to control the gripper directly from ROS.

The envelopes are held together upright by a simple mechanical structure using 3-D printed brackets and a pair of slotted angle rails as shown in Figure 6.3. The brackets were printed using PLA since they are only used to hold the envelopes in

space and didn't have many mechanical requirements.

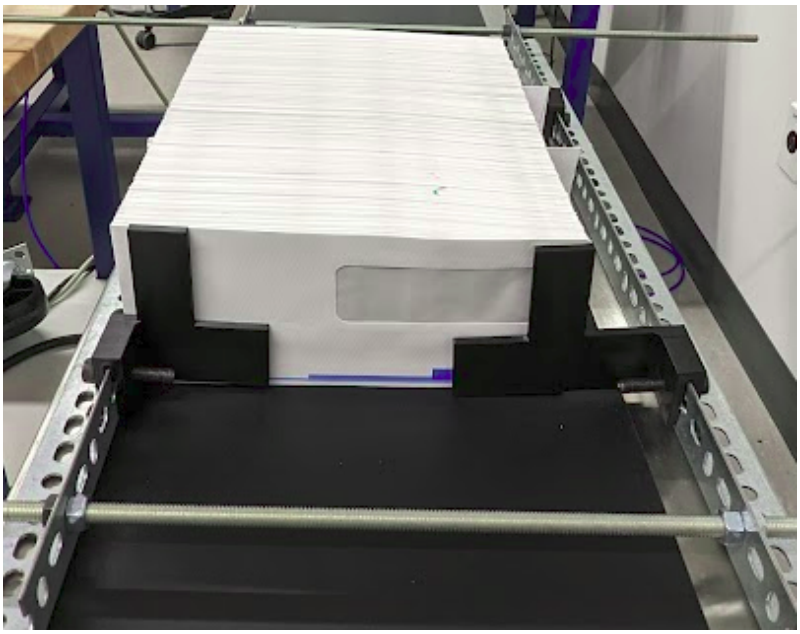

Figure 6.3: Top view of envelope holding mechanism

6.2 Perception Experiments

The perception pipeline's performance is extremely critical for the high-precision motions the robot needs to execute. Two experiments were carried out to (1) test the performance of notch detection in different lighting conditions, (2) analyze the accuracy of conveyor velocity estimation.

6.2.1 Notch Detection Under Different Lighting Conditions

To see how different lighting conditions affect the perception performance, 4 arbitrary positions within the region of interest of the camera were chosen for localizing a notch. For different lighting conditions, the notch is detected and the robot is commanded to move exactly to the position of the notch as shown in Figure 6.4.

The parallel difference between the gripper position and the notches is recorded under different lighting conditions as shown in Table 6.1. A positive error means the gripper reached behind the notch; a negative error means it reached in front of the notch instead.

Figure 6.4: Measuring the localization error by commanding the robot exactly to the notch position

For each of these positions, it can be observed that there is not a significant change in error due to lighting conditions changing. This means the perception subsystem is robust to changes in lighting, even though sensor noise can lead to some errors especially when mounted at more than 1m height from the conveyor. The notch detected for a position with the above-studied light illuminance conditions is shown in Figure 6.5. While these errors are in the acceptable range for reliable grasping, the camera could be further calibrated to give better accuracy.

Table 6.1: Notch Detection Error for Different Light Intensities

Light Illuminance (Lux)	Notch Detection Error (mm)			
	Position 1	Position 2	Position 3	Position 4
418	0	3	0	-4
374	0	2	5	-2
275	0	3	5	-2
164	0	2	0	1
101	0	2	6	-2
42	-1	5	2	1

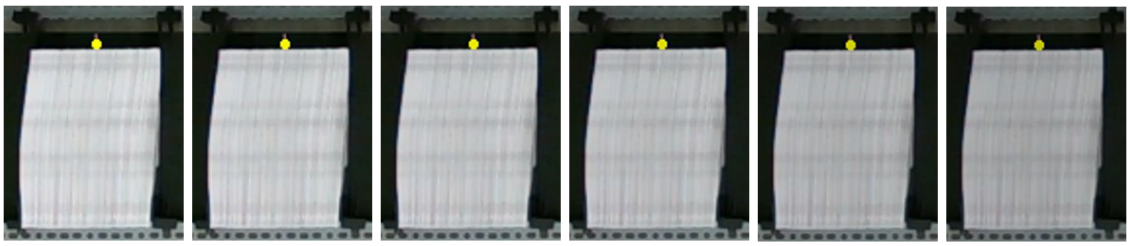

Figure 6.5: Notch detection in the studied light Illuminance conditions from 418 Lux to 42 Lux.

6.2.2 Notch Velocity Tracking Performance

To test the notch velocity tracking accuracy of the perception system, the conveyor belt was run at different speeds with the envelope stacks on top. The rate of change of notch positions between consecutive frames was used to compute the speed of the conveyor in the perception pipeline and the error between the actual and estimated speeds are plotted in Figure 6.6.

The velocity estimation of the conveyor needs to be very accurate for the slicing and notch sliding operations performed by the robot. From repeated experiments and testing, it was concluded that this method of velocity estimation was not reliable enough to be used by the motion planning pipeline because of sensor noises in depth images. Hence, for all pick and place testing and experiments in this thesis, the conveyor speeds were measured manually and fed as input to the motion planning pipeline.

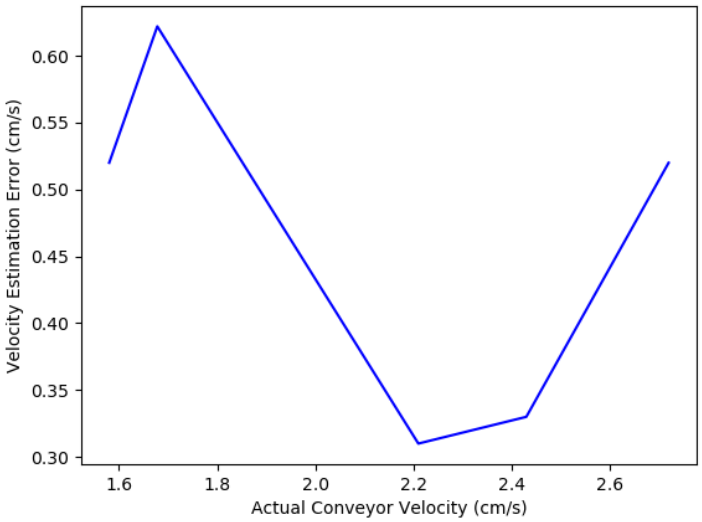

Figure 6.6: Plot of velocity estimation error under different conveyor speeds

6.3 Pick-and-place Experiments

In this section, two experiments were carried out to analyze the pick-and-place process under different conditions. The first experiment involves a comparison of grasp success rate, picking and placing times, cycle times, and box-filling accuracy for different velocities of the conveyor belt. The 2nd experiment involves testing different end-effector velocities while performing the zig-zag motions for placing the envelopes without the notch envelope sticking out of the stack. It aims to minimize the distance the notch envelope sticks out of the stack after placing it and retracting the arm.

6.3.1 Pick and Place Analysis

Analyzing the pick and place cycles for different conveyor velocities is important to ensure this system is robust enough to handle more dynamic manufacturing lines. For different conveyor velocities, we try to get the system to fill the complete

cardboard box which is capable of holding 1000 envelopes at a time. Hence, we run the pick and place cycle 5 times for each conveyor velocity where 200 envelopes are packaged each time. Moreover, the box-filling process for each conveyor velocity scenario is analyzed using these 5 evaluation metrics:

- Grasp Success Rate: The percentage of times when the gripper has a stable grasp of envelopes after probing, slicing into the moving stack, picking, and placing it. A grasp is considered a success if the robot is able to get the stack inside the box without causing any damage or disturbances to the envelopes or the box.

- Average Pick Time: The average of the times it takes to reach the moving envelopes, probe the notches, slide the notches inside, close the gripper, and retract during each cycle for a given conveyor velocity.

- Average Place Time: The average of the times it takes for the robot to reach the cardboard box, reach inside the place location, release the stack, execute friction-related behaviors, retract, and move to its start pose during each cycle for a given velocity of conveyor.

- Average Cycle Time: The average of the times it takes to execute a complete pick and place cycle for a given conveyor velocity.

- Average Box Filling Error: The average of the errors measured between desired placing pose inside the box for each envelope stack during each pick and place cycle, for a given conveyor velocity.

The box-filling error is updated after each pick and place cycle by measuring gaps between the previously and currently placed envelope stacks.

Table 6.2: Performance of the Robotic System for Different Conveyor Velocities

Conveyor Velocity (cm/s)	Grasp Success Rate (%)	Average Pick Time (s)	Average Place Time (s)	Average Cycle Time (s)	Average Box Filling Error (cm)
1.4	100	17.83	26.86	44.69	2.6
1.7	80	17.87	26.88	44.75	2.2
1.9	80	17.9	26.73	44.63	1.1
2.2	100	17.98	27.01	44.99	2.5
2.4	80	17.84	27.3	45.14	3.6

Each time the robot is unable to grasp/successfully place a stack during these cycles, its grasp success rate decreases by 20% and the cycle is executed again. The failure modes of grasping are discussed in the next subsection.

A noteworthy feature of this robotic system is its repeatability in pick and place times. The pick, place, and cycle execution times are almost equal regardless of the conveyor speed. This is particularly useful when the system is deployed into industrial settings where accurate modeling of cycle times is necessary for simulating large-scale packaging of envelopes. At the current rate, the robot can fill the entire cardboard box in 3.75-4 minutes. This can be further improved by increasing the maximum velocity and acceleration factors for trajectory generation and execution.

6.3.2 Grasp Failure Modes

Grasp failure during the pick and place analysis can happen while performing any of the following operations in decreasing order of failure criticality:

- **Stack Pickup:** In this failure case, the robot fails to pick up the grasped envelope stack successfully as shown in Figure 6.7. This can be caused by relative slipping between envelopes, unforeseen collisions with envelopes due

to improper slicing into envelope stack, or inadequate input pressure.

Figure 6.7: Picking failure caused by a combination of the other failures or inadequate pressure

- **Notch Probing:** This is the robot's failure to probe the notch correctly and is caused by notch detection accuracy issues in the perception pipeline as shown in Figure 6.8. Failure to probe the notch accurately can lead to the packaging of inaccurate lengths of envelope stacks, or damage to envelopes due to incorrect slicing location.

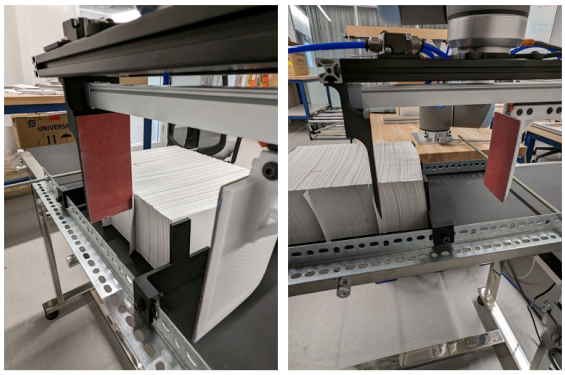

Figure 6.8: Notch probing overshoot caused by accuracy errors in perception pipeline

- **Stack Slicing:** After probing the notch, the robot has to slice into the isolated stack while matching the conveyor velocity along the respective component of end-effector velocity. This can sometimes cause damage to envelopes if the stack is not isolated enough before slicing as shown in Figure 6.9.

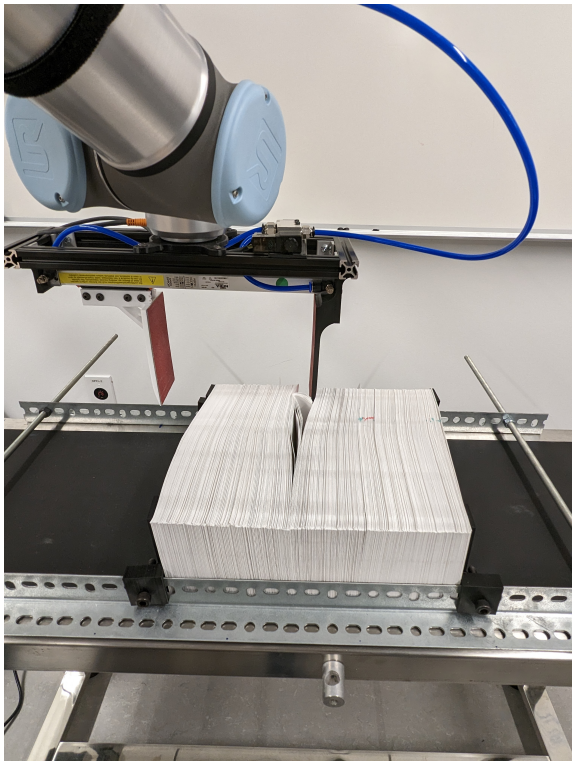

Figure 6.9: Failure to slice into the stack safely

- **Notch Sliding:** This is characterized by the robot's failure to slide the notch envelope into the stack (Figure 6.10). Failure to slide in the notch can happen due to insufficient forces applied to it while executing sliding behaviors. If the robot does not slide the notch into the stack, the sticking-out envelope can be damaged while packed into cardboard boxes.

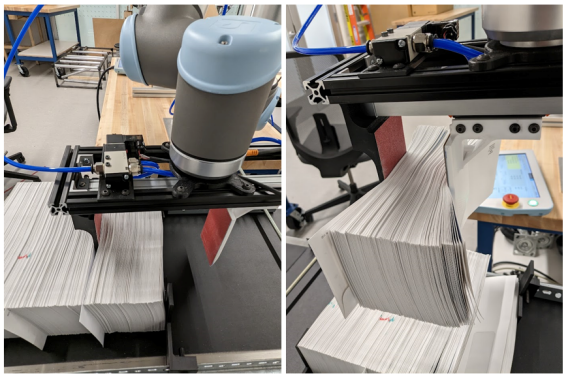

Figure 6.10: Failure to slide the notch into the stack due to inadequate force applied during sliding motion

6.3.3 Safe Retract Analysis

The next experiment aims to study the effect of the speed of servoing while performing zig-zag motions to solve the friction problem of envelopes sticking to the gripper jaws while retracting after placing. For this analysis, the X and Z axis speeds are varied (refer to Figure 5.6 for axis directions), and the subsequent displacement of the last envelope after the arm successfully retracts from the box. The trial is considered a failure if the envelope in contact gets completely isolated from the stack/falls onto the top of the stack, and the corresponding displacement is marked as -.

Table 6.3: X and Z axis speeds for executing motion to overcome the friction between the jaw and envelope in contact; corresponding displacement if arm retract was successful

Speed along X axis (m/s)	0.03	0.05	0.1	0.2	0.03	0.05	0.05	0.1
Speed along Z axis (m/s)	0.01	0.01	0.01	0.01	0.005	0.015	0.02	0.02
Displacement (cm)	0.5	0.3	2	-	0.45	5	1.7	4

It can be noted from this experiment that the slower the velocities and smoother the profile, the robot is better at overcoming the friction between its jaws and

retracting to get ready for the next pick-and-place cycle. Out of these velocities, an X-velocity of 0.05 m/s and Z-velocity of 0.01 m/s give the best results and are best suited out of the other trials to solve the friction problem after placing the envelopes into the box.

Chapter 7

Conclusion and Future Work

In this thesis, a novel robotic system was proposed that can reliably pick and place moving envelope stacks. For cutting into the stacks and grasping reliably, a custom pneumatic gripper capable of manipulating stacks of 200 envelopes was designed and integrated with the robot. The perception pipeline involved an RGB-D camera mounted above the conveyor belt to track the notches and localize them with respect to the robot's frame. Two approaches to detecting the notches, one in simulation and the other in hardware were proposed, and the challenges in detecting them were discussed. The motion planning pipeline uses the initial position of the notches to extrapolate it into the future using already-known conveyor velocity. The different strategies for motion planning and servoing for each step of the pick and place cycle were proposed. The problem of the notch envelope sticking with the jaw and protruding out of the stack while retracting the gripper was introduced and solved using a smooth servoing strategy. The perception and motion planning pipelines were finally evaluated through a series of experiments to understand how to better optimize the system.

Several avenues of improvement and exploration can be considered to enhance

the performance and applicability of the developed robotic system in the future. Firstly, incorporating position control in the gripper to enable partial opening while retracting could provide a more effective and safer solution for handling edge cases. Secondly, efforts can be made to improve the perception pipeline's accuracy of notch tracking, allowing the robotic system to better identify and manipulate the envelope stacks. The perception pipeline can be further improved by enabling it to detect envelopes and notches with a mix of colors, color gradients, etc.

The addition of an in-hand camera for tracking dynamic objects with varying velocities could significantly expand the range of applications for the robotic system. Furthermore, implementing dynamic collision avoidance techniques would enable safe human-robot collaboration, broadening the scope of potential use cases and industries.

Closed-loop probing and slicing of notches using appropriate sensors could improve the overall reliability and precision of the system. The integration of grasp quality checks using sensing technology would allow for real-time evaluation of the effectiveness of each grasp, potentially leading to better performance. Finally, the development and application of grasp correction methods could further ensure a higher success rate in handling and placing envelope stacks. Pursuing these future research directions would contribute to the ongoing optimization and refinement of the robotic system and its capabilities.

Bibliography

[1] Allen, P., Timcenko, A., Yoshimi, B., and Michelman, P. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation* (07 1994).

[2] Balaguer, B., and Carpin, S. Combining imitation and reinforcement learning to fold deformable planar objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 1405–1412.

[3] Bing, W., and Xiang, L. A simulation research on 3d visual servoing robot tracking and grasping a moving object. pp. 362 – 367.

[4] Bo Jørgensen, T., Debrabant, K., and Krüger, N. Robust optimization of robotic pick and place operations for deformable objects through simulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 3863–3870.

[5] Bodenhagen, L., Fugl, A. R., Jordt, A., Willatzen, M., Andersen, K. A., Olsen, M. M., Koch, R., Petersen, H. G., and Kruger, N. An adaptable robot vision system performing manipulation actions with flexible objects. *IEEE Transactions on Automation Science and Engineering 11*, 3 (2014), 749 – 765.

[6] Canny, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 6 (1986), 679–698.

[7] Chitta, S., Sucan, I., and Cousins, S. Moveit![ros topics]. *IEEE Robotics & Automation Magazine 19*, 1 (2012), 18–19.

[8] Corp., F. Festo parallel gripper hgpl, 2023.

[9] Dellen, B., Husain, F., and Torras, C. Joint segmentation and tracking of object surfaces in depth movies along human/robot manipulations. *VISAPP 2013 - Proceedings of the International Conference on Computer Vision Theory and Applications 1* (01 2013), 244–251.

[10] Dinakaran, V., Balasubramaniyan, M., Le, Q., Jawad Alrubaie, A., Al-khaykan, A., Muthusamy, S., Panchal, H., Jaber, M., Dixit, A.,

and Prakash, C. A novel multi objective constraints based industrial gripper design with optimized stiffness for object grasping. *Robotics and Autonomous Systems 160* (10 2022), 104303.

[11] fontanals, J., Dang-Vu, B.-A., Porges, O., Rosell, J., and Roa, M. A. Integrated grasp and motion planning using independent contact regions. vol. 2015, pp. 887–893.

[12] Gaskett, C., and Cheng, G. Online learning of a motor map for humanoid robot reaching.

[13] Group, Z. Zimmer pneumatic gripper ghk6000 series, 2023.

[14] Guo, M., Gealy, D., Liang, J., Mahler, J., Goncalves, A., McKinley, S., Ojea, J., and Goldberg, K. Design of parallel-jaw gripper tip surfaces for robust grasping. pp. 2831–2838.

[15] Hassan, A., and AboMoharam, M. Design of a single dof gripper based on four-bar and slider-crank mechanism for educational purposes. vol. 21.

[16] Haustein, J., Hang, K., and Kragic, D. Integrating motion and hierarchical fingertip grasp planning. pp. 3439–3446.

[17] Hua, H., Song, J., Zhao, J., and Liao, Z. Sensor-less grasping force control of a pneumatic underactuated robotic gripper. *Journal of Mechanisms and Robotics 16* (03 2023).

[18] Itoh, H., Okamoto, T., Fukumoto, H., and Wakuya, H. An electroadhesive paper gripper with application to a document-sorting robot. *IEEE Access 10* (2022), 113598–113609.

[19] Jamone, L., Natale, L., Metta, G., Nori, F., and Sandini, G. Autonomous online learning of reaching behavior in a humanoid robot. *International Journal of Humanoid Robotics 9* (07 2012), 1250017.

[20] Jamone, L., Natale, L., Sandini, G., and Takanishi, A. Interactive online learning of the kinematic workspace of a humanoid robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), pp. 2606–2612.

[21] Jørgensen, T. B., Jensen, S. H. N., Aanæs, H., Hansen, N. W., and Krüger, N. An adaptive robotic system for doing pick and place operations with deformable objects. *Journal of Intelligent & Robotic Systems 94* (2019), 81–100.

[22] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation* (2011), IEEE, pp. 4569–4574.

[23] Kicki, P., Bednarek, M., and Walas, K. Robotic manipulation of elongated and elastic objects. In *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)* (2019), pp. 23–27.

[24] Kim, J., and Croft, E. A. Trajectory planning for robots : the challenges of industrial considerations.

[25] Kiryati, N., Eldar, Y., and Bruckstein, A. M. A probabilistic hough transform. *Pattern recognition 24*, 4 (1991), 303–316.

[26] Koenig, N., and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* (2004), vol. 3, pp. 2149–2154 vol.3.

[27] Kondak, K., Binner, S., Hommel, G., and Neumann, M. Time optimal manipulator control for sensor guided grasping of moving objects. vol. 4, pp. 1912 – 1917 vol.4.

[28] Kyriakopoulos, K., and Saridis, G. Minimum jerk path generation. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation* (1988), pp. 364–369 vol.1.

[29] Lei, M., and Ghosh, B. Visually guided robotic tracking and grasping of a moving object. pp. 1604 – 1609 vol.2.

[30] Li, G., and Jie, Z. A real-time stereo visual servoing for moving object grasping based parallel algorithms. pp. 2886 – 2891.

[31] Li, H., Gong, Z., Lin, W., and Lippa, T. Motion profile planning for reduced jerk and vibration residuals.

[32] Li, X., Li, S., Bai, W., Cui, X., Yang, G., Zhou, H., and Zhang, C. Method for rectifying image deviation based on perspective transformation. In *IOP Conference Series: Materials Science and Engineering* (2017), vol. 231, IOP Publishing, p. 012029.

[33] Li, X., Liu, S., Tong, L., and Gao, R. A novel dual-stage shape memory alloy actuated gripper. *Industrial Robot: the international journal of robotics research and application 50* (12 2022).

[34] Macfarlane, S., and Croft, E. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation 19*, 1 (2003), 42–52.

[35] Misimi, E., Øye, E. R., Eilertsen, A., Mathiassen, J. R., Åsebø, O. B., Gjerstad, T., Buljo, J., and Øystein Skotheim. Gribbot – robotic 3d vision-guided harvesting of chicken fillets. *Computers and Electronics in Agriculture 121* (2016), 84–100.

[36] Perumaal, S., and N, J. Automated trajectory planner of industrial robot for pick-and-place task. *International Journal of Advanced Robotic Systems 10* (02 2013), 1–17.

[37] Rabenorosoa, K., Clevy, C., Chen, Q., and Lutz, P. Study of forces during microassembly tasks using two-sensing-fingers grippers. *IEEE/ASME Transactions on Mechatronics 17*, 5 (2012), 811–821.

[38] Ratiu, M., and Prichici, M. Industrial robot trajectory optimization- a review. *MATEC Web of Conferences 126* (01 2017), 02005.

[39] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation* (2009), IEEE, pp. 489–494.

[40] Saboukhi, A., Rahimi Gorji, M., Amirpour, E., Savabi, M., Fesharakifard, R., Ghafarirad, H., and Rezaei, S. Design and experimental analysis of a force sensitive gripper for safe robot applications.

[41] Seita, D., Florence, P., Tompson, J., Coumans, E., Sindhwani, V., Goldberg, K., and Zeng, A. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 4568–4575.

[42] Siradjuddin, I., Behera, L., Mcginnity, T., and Coleman, S. A position based visual tracking system for a 7 dof robot manipulator using a kinect camera. pp. 1–7.

[43] Spensieri, D., Carlson, J., Bohlin, R., Kressin, J., and Shi, J. Optimal robot placement for tasks execution. *Procedia CIRP 44* (12 2016), 395–400.

[44] Suzuki, S., et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing 30*, 1 (1985), 32–46.

[45] Tomasi, C., and Manduchi, R. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)* (1998), IEEE, pp. 839–846.

[46] Vahrenkamp, N., Asfour, T., and Dillmann, R. Simultaneous grasp and motion planning: Humanoid robot armar-iii. *IEEE Robotics Automation Magazine - IEEE ROBOT AUTOMAT 19* (06 2012), 43–57.

[47] Wu, Y., Yan, W., Kurutach, T., Pinto, L., and Abbeel, P. Learning to manipulate deformable objects without demonstrations.

[48] Ypma, T. J. Historical development of the newton–raphson method. *SIAM review 37*, 4 (1995), 531–551.