Shared Autonomous System for Robot-Assisted Sewing

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Alexandra Emrick

Sophia Gudenrath

Dominic Cupo

Elijah Eldredge

Mary Hatfalvi

Wyatt Henke

Advised By:

Professor Craig Putnam

Professor Jie Fu

Professor Jane Li

Abstract

The athletic industries in the United States produce apparel and equipment that contribute to the $60.4 billion sports market in North America [1]. However, the processes to produce these goods are inefficient, with over 1% of products defective. They also put factory workers in potentially uncomfortable environments. A shared autonomous robot-assisted sewing system is proposed to mitigate these issues. The system incorporates a computer vision subsystem to identify the sewing path and location of fabric, motion planning algorithms to optimize sewing trajectories, and intuitive input devices that allow a user to teleoperate the robot while sewing. This system has the potential to increase sewing efficiency and decrease the risk posed to workers.

Acknowledgements

We would like to acknowledge Jeff Lapalme and New Balance for sponsoring this project and bringing it to WPI. We would like to thank our advisors and WPI graduate students Akshay Kumar and Lening Li for their assistance on this project.

Contents

Abstract **i**

Acknowledgements **ii**

List of Figures **viii**

List of Tables **xi**

1 Introduction **1**

1.1 Project Statement 1

1.2 Customer Value Proposition 2

2 Background **4**

2.1 Manufacturing with Sewing 4

2.1.1 Existing Robotic Sewing Tasks 5

2.2 Computer Vision 6

2.2.1 Camera Parameters 6

2.2.2 Object Identification: Classic Computer Vision Techniques 9

2.3 Motion Planning 13

2.3.1 Trajectory Planning 14

2.3.2 Problems with High-Dimensionality and Potential Solutions 15

2.3.3 Existing Motion Planning Algorithms 16
2.3.4 Visualization Platforms 25
2.4 Joint Actuation and Control 26
2.4.1 Forward Kinematics and DH Parameters 27
2.4.2 Inverse Kinematics 29
2.5 Shared Autonomy 30
2.5.1 Input Devices 31

3 Concept of Operations **33**
3.1 Stakeholders 33
3.2 Use Cases 34
3.3 User Stories 37
3.4 Requirements 38
3.4.1 Functional Requirements 38
3.4.2 Nonfunctional Requirements 39

4 Technical Documentation **40**
4.1 Workspace Layout 40
4.2 The Robot 51
4.2.1 End of Arm Tooling 54
4.3 System Overview 56
4.3.1 Computer Requirements 56
4.3.2 ROS Architecture 57
4.3.3 Program Flow 59

4.4 User Interface 61

4.4.1 UI Design 62

4.4.2 Input Devices 67

4.5 Computer Vision 80

4.5.1 Camera Setup 80

4.5.2 CV Software and Node Architecture 82

4.5.3 Tasks 83

4.5.3.1 User Views 83

4.5.3.2 Calculate Sewing Machine and Robot Offsets 83

4.5.3.3 Vamp Detection 88

4.6 Motion Planning 92

4.6.1 Modeling the Workspace 92

4.6.2 Transformations 94

4.6.3 Simulating Motion: Moveit 95

4.6.4 Simulation Motion: Klamp't 101

4.6.5 IK Solving 105

4.6.6 Motion Planner Analysis 106

4.6.7 EEF Position Determination 107

5 Evaluation 115

5.1 Social Implications 115

5.2 Project Execution Evaluation 117

5.2.1 Execution Summary 117

5.2.2 Timeline Adjustments 118

5.2.3 Risk Management 119

5.2.4 Budget and Expenditure Justification 120

5.3 Product Evaluation 120

5.3.1 Hardware Requirements 120

5.3.2 Motion Planning Performance 122

5.3.3 Fabric Manipulation Performance 123

5.3.4 Overall System Performance 124

6 Recommendations for Future Work 125

6.1 Computer Vision 125

6.2 Control Interface 126

6.3 Motion Planning 127

6.4 Overall System Improvements 128

Appendices 131

A Robot Specification Sheet 131

B Robotiq 2-Finger Adaptive Gripper Specifications Sheet 132

C Trade Study 132

C.1 Camera Selection 132

C.2 Gripper Selection 134

D UI Feature Lists: Buttons 136

E UI Features Lists: Messages **137**

F UI Features Lists: Camera Views **137**

G UI Feature Lists: Input Device Interface **138**

H Pros and Cons of Potential Input Devices **138**

I Test Plans **140**

I.1 Offset Calculations 140

I.2 Inverse Kinematic Solving 142

I.3 Landmark Identification 143

I.4 Robot Controller 144

I.5 Gripper and Gripper Controller 145

I.6 Camera Nodes 145

I.7 User Interface 146

I.8 Input Devices 146

I.9 Integration 152

J Construction **153**

K Link Masses and Inertial Matrices **154**

L Motion Planning Analysis Criteria **155**

M Timeline **157**

N Budget and Expenditure Justification 164

O Motion Planner Evaluation Results 166

P Gap Analysis 169

Q Authorship Table 172

List of Figures

1 Athletic Footwear: Global Market Share 3
2 Pinhole Camera Representation 7
3 Examples of ArUco Tags 10
4 Canny Algorithm Steps Visualized 11
5 Example of a Singularity where Two Joints are Aligned 14
6 Exploration using RRT 19
7 Exploration of RRT (left) and RRT* (right) 21
8 Hinge Loss (blue) vs Zero-One Loss Functions (green) 24
9 Two Robot Joints and DH Parameters 28
10 Robot workspace with objects labeled: Right View 40
11 Robot workspace with objects labeled: Left View 41
12 Robot with Sewing Machine Simulated in RViZ 42
13 Vamp (bottom) and Toe Piece (top) 43
14 Communication between all Components 44
15 Warning and Safety Zones of OMRON Safety Sensor 45

16 Safety Mat (Black); turns off Robot's Servos when Stepped on 46
17 Durkopp Adler 868 Sewing Machine 47
18 Camera Mounted on (Right) Wrist 48
19 Camera Mounted on the Sewing Machine 49
20 Lighting mounted to sewing machine 50
21 Camera Mounted on the Tripod 50
22 Yaskawa Motoman SDA10F 51
23 SDA10F Model with Joint Names 52
24 FS100 Controller (left) and Pendant (right) 53
25 85mm Robotiq 1-Finger Adaptive Robot Gripper 55
26 ROS Node Diagram 58
27 Flowchart of System Progress 59
28 Robot Home Position 60
29 Robot Start Position 61
30 Initial UI Design in C++ with QT 63
31 Final UI using RQT and C++ 64
32 Final UI in Python Tkinter, Tripod Camera View 65
33 Wii Remote 70
34 Motion Plus Attachment for Wii Remote 71
35 Geomagic Touch Haptic Device in Home Position 72
36 Sewing Pattern specified using the Haptic Device 73
37 Wii Remote Input Device UI 75
38 Haptic Device UI 76

39 B Button on Wii Remote 77
40 Haptic device tool, buttons, and rotating axis 78
41 Example path of an input device 79
42 View and Select Plot 79
43 Camera calibration checkerboard 81
44 Camera calibration identifying image points 82
45 ArUco Tags on Sewing Machine 84
46 Wrist Cameras Mounted on the Robot's Grippers 85
47 Wrist Gripper Mount 86
48 Wrist Gripper Mount featuring adjustable pitch angle 87
49 OpenCV Key Points 89
50 OpenCV Key Point Analysis 90
51 Key Point Analysis with Vamp in Grippers 92
52 Sewing Machine Model from Kinect 94
53 Transformation Tree 95
54 Moveit Side Panel in demo.launch File 97
55 Klamp't Simulation World 103
56 Placement of Motion Capture Markers on Hands 108
57 Motion Capture Workspace with Origin Shown 109
58 Scatter Plots of X,Y positions of the Left and Right Hands . . 111
59 Vamp held in Robot Grippers 112
60 Vamp held incorrectly in robot's grippers 123
61 Curved and Straight Paths Traced During Testing 149

62 Exploded view of wrist camera mount in Inventor 153
63 Assembled view of wrist camera mount in Inventor 154

List of Tables

1 Stakeholders and Relevant Information 34
2 Use Case 1 35
3 Use Case 2 36
4 Use Case 3 37
5 Features of User Interface 67
6 Camera Decision Matrix 133
7 Gripper Decision Matrix 135
8 Desired Buttons on UI 136
9 Desired Messages on UI 137
10 Pros and Cons of Input Devices 140
11 Consistency of Offset Calculations 141
12 Accuracy of Offset Calculations 141
13 Accuracy of TRAC_IK Algorithm 142
14 Precision of Landmark Identification (in pixels) 143
15 Precision of Landmark Identification (in m) 143
16 Accuracy of Landmark Identification 144
17 Precision of Wii Remote Data when Zeta = 0.360094977 . . . 147
18 Precision of Wii Remote Data when Zeta = 0.3771788462 . . . 147

19 Precision of Haptic Device: Position (mm) 148
20 Precision of Haptic Device: Orientation (rad) 148
21 Accuracy of Haptic Device Positions along a Curved Path . . 150
22 Accuracy of Haptic Device Positions along a Straight Path . . 151
23 Accuracy of Wii Remote Positions along a Curved Path 151
24 Accuracy of Wii Remote Positions along a Straight Path . . . 152
25 Link Masses and Inertial Matrices 155
26 Budget and Expenditure Justification 166
27 RRT Evaluation Results 167
28 RRT* Evaluation Results 168
29 PRM Evaluation Results 168
30 LBKPIECE Evaluation Results 169
31 Gap Analysis 172
32 Authorship 173

1 Introduction

Manufacturing is a complicated process with many steps. The process must be fast, affordable, and safe while producing a high-quality product and minimizing waste. Human workers are an important part of the manufacturing process because they can operate machinery or perform tasks that are too complicated or expensive to be easily automated. However, this can put humans in unsafe working conditions or leave them doing repetitive, dull assembly tasks. Since the instantiation of the Occupational Health and Safety Act (OSHA) in 1970, robots in factories have helped decrease occupational hazards [2]. Incorporating robots into manufacturing processes can help people avoid harmful circumstances, such as loud noises or proximity to dangerous machinery. The opportunity for people and robots to collaborate on manufacturing tasks can improve the quality of a manufacturing line, and in the lives of employees.

1.1 Project Statement

Despite the advantages of factory automation, there are some tasks that are more difficult for robots to accomplish than humans. For example, only 8% of jobs can be mostly automated (more than 90% of tasks assigned to the job) [3]. Fine manipulation tasks are often too difficult to accomplish automatically. They require precise motion, dexterous end effectors, and fast planning. However, it is possible to combine the dexterous skills of a human and the automation benefits of a robot by implementing a shared autonomous process [4]. Shared autonomous processes allow robots and humans to work synchronously to accomplish tasks in a safe and efficient way and can also help train new workers by eliminating tremors while sewing and reducing errors during teleoperation. Due to the required precision, uncomfortable work environments, and time it takes to train a new seamster, shoe sewing is a specific fine manipulation task that could benefit from shared autonomy. Automating parts of the shoe sewing manufacturing process allows the employees to work in a safer environment while still performing intricate sewing tasks and lets the robot handle the more dangerous, monotonous tasks [5].

1.2 Customer Value Proposition

New Balance is invested in this project, hoping to improve their shoe sewing manufacturing process through automation. The athletic shoe market has changed drastically over the past decade as shoe companies diversify [6]. As of 2015, New Balance held 4.4% of the global athletic footwear market, while Nike held 22.9% and adidas held 9.7%, as shown in Figure 1 [7].

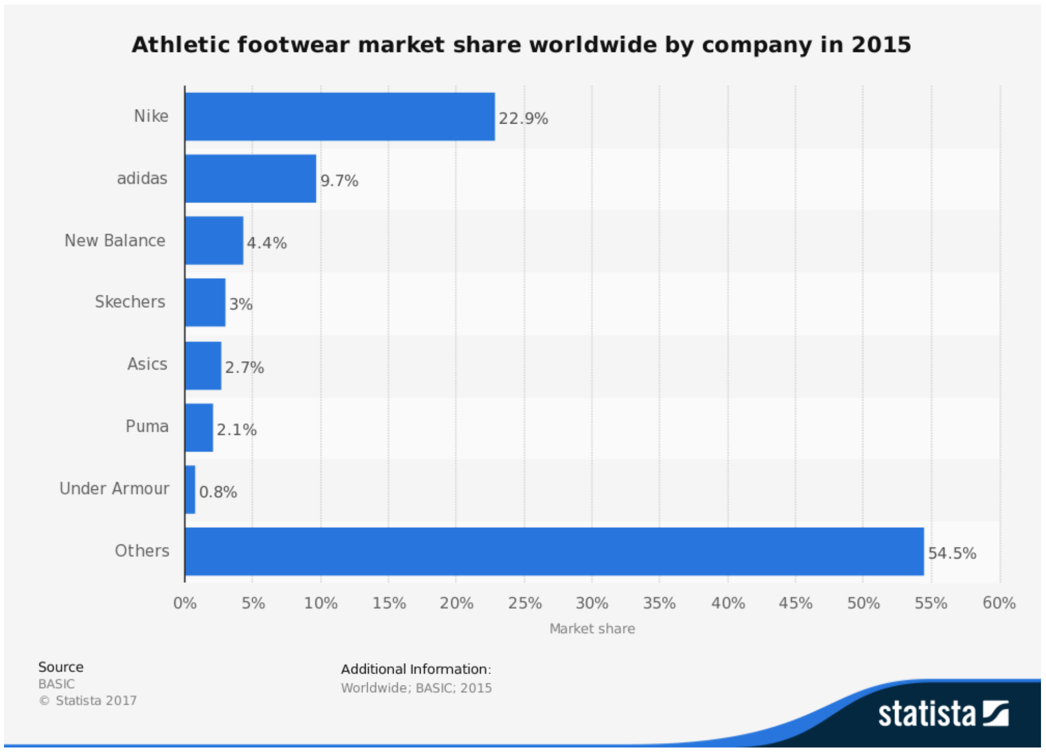


Figure 1: Athletic Footwear: Global Market Share

Implementing a shared autonomous system in the New Balance factory could help ensure the company gains a greater market share. In fact, global productivity is expected to grow 2.8% over the next 50 years due to automation [3]. Businesses such as New Balance have the opportunity to become more competitive in the market due to labor cost reductions, increased throughput, higher quality products, and decreased downtime [8].

Because of the benefits of automation, the goal of New Balance is to incorporate shared autonomous robotic systems into the shoe sewing manufacturing process to increase throughput, decrease sewing errors, move workers to safer environments, and train new workers faster. Typically, it takes one month for a new worker to learn how to properly sew the shoes, and even longer to do it quickly enough to meet their quota. Part of the difficulty lies in the strict, 1 millimeter tolerance for the sewing at New Balance, and manipulating the soft fabric can be challenging. The repeatability of an automated sewing system can help New Balance decrease production errors and save money. Integrating computer vision systems and advanced motion planning algorithms with a robot capable of very precise movements can help New Balance achieve its automation goals.

2 Background

2.1 Manufacturing with Sewing

In the United States, there is a decrease in manufacturing jobs and an increase in factory automation [9]. Automation has increased factory productivity, meaning companies must automate their processes to maintain a competitive advantage [10]. However, sewing is one of the industries that has been slow to adapt to the new market. The sewing industry hasn't changed a significant amount for about the last century. Today, it is still extremely reliant on human labor, even with the recent rise of manufacturing automation [11]. One reason that it is difficult and uncommon to automate sewing is the challenges that come with manipulating soft materials like fabric [12].

Textiles are a nonrigid material, which can buckle, slip, and rip as well as become misaligned if stacked during the sewing process [13]. Despite these limitations, there are successful implementations of automated sewing, which involve typically involve computer vision systems [11].

2.1.1 Existing Robotic Sewing Tasks

Dexterous robots with many degrees of freedom (DOF) have been successfully implemented in several experimental sewing operations. One of these operations used a robot, gripper, and photocells to sew a pucker-free seam with a constant width [14]. This system incorporated operation and control of the sewing machine while avoiding obstacles, singularities, and joint torque limits. Developed in 1990, this operation dissected the robot's tasks by finding and prioritizing sub-objectives within the main task and controlling the position and force of the joints throughout each task.

Another robotic sewing system involved two robots handling the fabric similar to how a human operator would [15]. The sewing pattern is composed of a series of points with the common coordinate system's origin where the needle and fabric meet. This system allows the different arms to operate asymmetrically, yet in cooperation to create an optimal product. The robots in this system rely on tactile feedback to ensure they are applying the correct force to press the layers of fabric together before they begin sewing. The system is operated by a user who selects two or three points that form the pattern the robot sews along. Although these robotic sewing systems were successful, they have not been widely implemented, and the opportunity for advanced, vision-based sewing automation is increasing.

2.2 Computer Vision

Computer vision is the ability to view and interpret an environment using cameras and machine learning algorithms [16]. It can be used to determine the position and orientation of objects in a workspace. Important considerations for a computer vision system are the camera mounting positions, number of cameras, and lighting, as well as the camera resolution and frame rate. Camera extrinsics, intrinsics, and classic computer vision techniques will be discussed in this section.

2.2.1 Camera Parameters

There are two main groups of parameters when analyzing a camera: intrinsics and extrinsics. Camera intrinsics are the internal characteristics of the camera, while extrinsics are parameters external to the camera, such as where they are mounted [17]. The camera intrinsics matrix, K, transforms the 3D camera coordinates into 2D image coordinates. When considering a traditional pinhole camera, the matrix contains five parameters shown in

Equation 1 that define the camera's geometry.

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Two parameters, f_x and f_y, describe the focal length, the distance from the pinhole to the image plane. They are the x and y components of the distance f, shown in Figure 2.

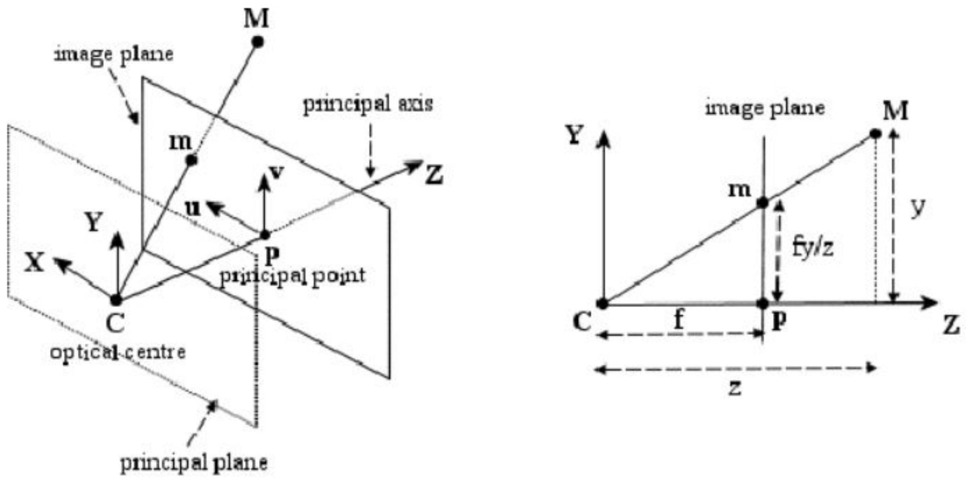


Figure 2: Pinhole Camera Representation

Ideally, f_x and f_y are the same but inconsistencies in camera sensors, distortion, and errors in calibration cause discrepancies. The parameters, x_0 and y_0, describe the principle point. The principal point offset is the location of the principal point, p in Figure 2, with respect to the optical centre, C in Figure 2. The fifth parameter of the K matrix is skew, s. Skew is the offset generated by factors of the lens and distance from object, caused mainly by manufacturing error [18]. Calibration routines incorporate these parameters in order to provide a clear and accurate image [17].

In addition to the internal properties of the camera, it is important to understand how the camera fits into the workspace [19]. Camera extrinsics integrate the camera's center and heading coordinates. This is how the frame of the camera is related to the frame of the world. Considering the camera frame, C, and the world frame, W, Equation 2 shows the perspective projection equation.

$$p = \frac{1}{z} M P \tag{2}$$

where P denotes the vector of homogeneous coordinates in the world frame, W.

As shown by these equations, there are 6 extrinsic parameters: three angles defining the rotation matrix R, and three coordinates of the translation vector t.

The matrix M in Equation 2 can be rewritten as a function of the intrinsic and extrinsic parameters shown in Equation 3.

$$M = \begin{bmatrix} \alpha r_1^T - \alpha \cot(\theta r_3^T) + u_0 r_3^T & \alpha t_x - \alpha \cot(\theta t_y) + u_0 t_z \\ \frac{\beta}{\sin\theta} r_2^T + v_0 r_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ r_3^T & t_z \end{bmatrix} \tag{3}$$

The camera calibration determines the extrinsic and intrinsic parameters of the transformation between an object in the real world and an object seen by the camera [20].

2.2.2 Object Identification: Classic Computer Vision Techniques

Classic computer vision techniques, as opposed to deep learning methods, are discussed in this chapter in terms of how they can detect and track objects in the workspace.

Object Detection

Object detection has three main parts: segmentation, shape identification, and object recognition [21]. In segmentation, the important features of the object are identified and groups of pixels, or segments, are formed based on similar features. Next, the shape identification routine takes the groups of pixels and attempts to determine their shapes using attributes such as shading, texture, and motion. Finally, the shape of the object is compared with a database of model shapes, and the object recognition routine outputs the object from the database that most closely matches the object in the image. After identifying the object, its location and orientation in the camera's coordinate frame must be determined. Programs using packages such as OpenCV can use camera images to detect edges, classify objects, and output locations and orientations [22].

Edge detection is an important aspect of shape identification. It is mainly used to decrease the amount of data in the image to make it easier to process

[23]. Differentiating colors can also help in object detection. High contrast of colors in a scene can make the edge detection, and thus shape identification, easier and more accurate [24]. Edge detection can be done using just cameras and computer vision algorithms, but some applications implement fiducial markers designed to track objects, such as ArUco Tags shown in Figure 3 [25]. These tags are low-cost, square black-and-white patterns that can be identified using OpenCV and Robot Operating System (ROS) tools. Some disadvantages of using markers like ArUco Tags are that they require a very high resolution camera and they must be attached to the objects, whereas edge detection algorithms do not need any extra hardware and can be more adaptable [26].

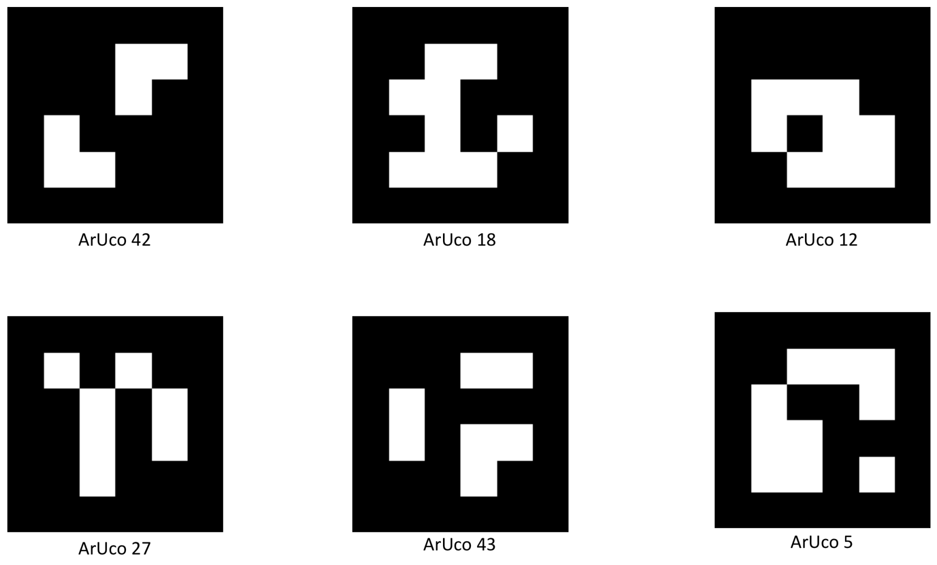


Figure 3: Examples of ArUco Tags

There are a variety of edge detection algorithms, and they are broken up into two different types: gradient-based or Laplacian-based. Gradient-based methods, such as Sobel, Prewitt, and Robert edge detection techniques, detect edges by finding the minimum and maximum values in the first derivative. These methods are implemented by using the magnitude of the gradient of the image [23]. Laplacian based methods, on the other hand, look for when the second derivative of an image is equal to zero. Where the second derivative equals zero, there is an edge in the image. Once the edges are located, all data from the image except the edges are eliminated [23]. One example of a Laplacian edge detection method is the Canny edge detection. The images in Figure 4 show the steps of edge detection using the Canny algorithm, with the raw image on the left and the detected edges on the right.

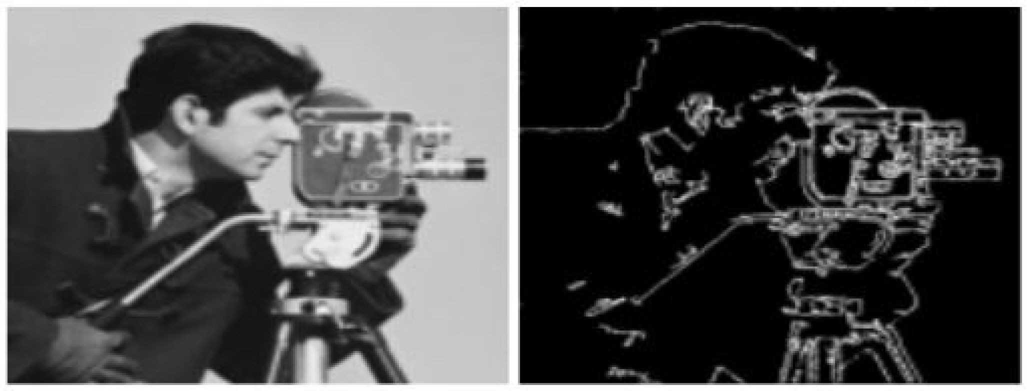


Figure 4: Canny Algorithm Steps Visualized

The Canny algorithm is one of the most commonly used edge detection methods [23]. It works by first reducing noise using a Gaussian filter [23]. Then, it finds the image gradient and removes all unwanted pixels [27]. This process is known as suppression. Hysteresis thresholding is then used to convert an image into a binary mask. This labels pixels in the background zero and all other pixels one [28]. The binary mask helps distinguish objects in the image from the background, which is especially helpful for object detection and edge detection in the foreground of images.

2.3 Motion Planning

Using computer vision systems can help track objects in the workspace. Whether a robotic system has a human operator or not, controlling the motion of a multi-DOF robot can be challenging. Motion planning is the task of creating a path between start and goal poses while avoiding collisions [29]. The start and goal poses must exist in the robot's workspace [30]. The workspace is determined using the range of each joint and is important because it defines the state space of the end effectors and joint locations, or all of the possible robot configurations [31]. Obstacles in the workspace and potential singularities of the robot must be noted. A boundary singularity is where the robot is commanded to leave the workspace whereas an internal singularity occurs when two or more of the robot's axes are aligned, resulting in a loss of a degree of freedom [31]. Figure 5 depicts an example of an internal singularity.

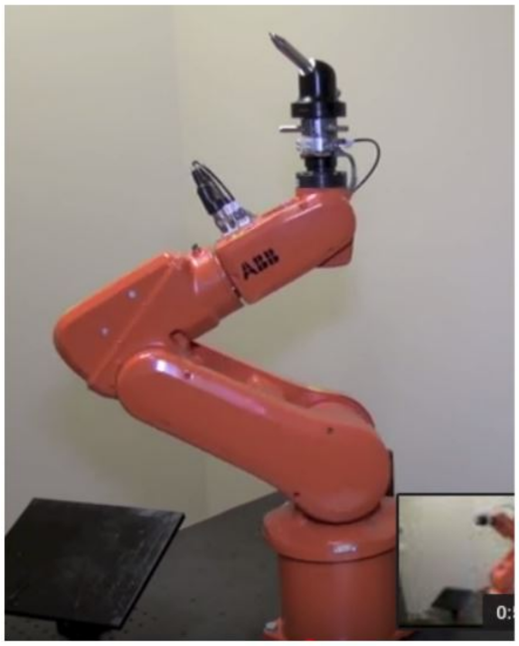


Figure 5: Example of a Singularity where Two Joints are Aligned

2.3.1 Trajectory Planning

Trajectory planning is generating motion control inputs to execute a planned trajectory to a goal pose. Trajectories include timing constraints such as velocities or accelerations, whereas a path is simply the series of points in the workspace [32]. Some requirements of trajectory planning are: minimizing computational demand, creating continuous functions for joint positions and velocities, and avoiding collisions. Optimizing trajectory plans creates smoother and more efficient paths to reach an end goal faster. Trajectory optimization algorithms can shorten already generated trajectories or initialize a trajectory with collisions and optimize it along the way. Therefore, numerical optimization methods and collision checking methods are necessary for trajectory optimization algorithms [33].

2.3.2 Problems with High-Dimensionality and Potential Solutions

Implementing motion planning on robotic systems of high-dimensionality can be time and computationally intensive [34]. In addition, redundant manipulators have an infinite number of inverse kinematic solutions, so without imposing some constraints, it can be impossible to reach a solution [35]. Several solutions have been proposed to compensate for these problems caused by high-dimensionality. One solution specifically for 7-DOF robots is to reduce the redundancy in planning by treating the robot as if it had six degrees of freedom instead of seven. If the proposed path contains inefficiencies or potential singularities, the seventh degree of freedom can be used to move the robot arm into a more ideal position than originally planned [36].

Another potential solution only works for robots whose wrist joints are independent of the other arm joints. If the robot has an independent wrist, the 7-DOF planning problem can be separated into two lower dimensional planning problems: one 4-DOF problem for the arm joints that determines end effector position and one 3-DOF problem for the wrist joints that determines end effector orientation [37]. As opposed to changing the planning problem, other solutions use motion primitives to construct paths. Instead of replanning paths for each iteration of the problem, small segments of frequent paths, called motion primitives, are saved and connected to make new paths

[38]. Motion primitives could also be combined with other potential solutions for high-dimensionality issues to further simplify planning problems.

2.3.3 Existing Motion Planning Algorithms

This section describes the two categories of motion planning algorithms, sampling- and optimization-based, as well as some common existing motion planning algorithms applicable to multi-DOF robot arms.

Sampling-Based Methods

In a sampling-based planning method, the obstruction space, defined as all the points in the workspace that would create a collision with the robot, is not explicitly defined. Instead, the workspace is probed using a sampling scheme and the obstruction space is constructed [39]. The advantage of a sampling-based method is that because the obstruction space is not defined, the algorithm is independent of a particular geometric model and can adapt to different workspaces. Many different sampling-based methods exist and will be explored in this section.

Optimization-Based Methods

The goal of an optimization-based method is to define the motion planning problem as an optimization problem, which finds a minimum-cost path using constraint functions [40]. Constraint functions describe criteria such as the minimum distance the robot must be from obstacles and maximum accelerations of the joints [41]. Optimization planning aims to combine the smoothness of the trajectory function with potential collisions of the obstacle function [42]. These two elements work together to determine the speed of the motion and path of the motion, respectively. Optimization-based methods tend to produce higher quality trajectories than sampling-based methods because they include a secondary step to improve plans [43].

PRM

Another method for motion planning is the Probabilistic Road Map, or PRM. It computes collision-free paths and works particularly well when planning for robots with many degrees of freedom. It starts with a learning phase where the probabilistic roadmap is constructed and stored as a graph. The nodes of the graph are configurations and the edges are paths. The paths are computed using a simple and fast local planner that is ideally deterministic. Next begins the query phase, where the paths that connect start and goal configurations are identified. One experiment with PRM cites the time of the learning phase as “a few dozen seconds” and the time of the query phase as “a fraction of a second” [44].

RRT

The RRT algorithm, or Rapidly-exploring Random Tree, is an example of a sampling-based method. It uses a randomized data structure that is designed for a broad class of path planning problems. The algorithm has been applied to holonomic, nonholonomic, and kinodynamic problems with up to 12-DOF robots. A system is nonholonomic, as opposed to holonomic, when the system has constraints on the velocity that are not derivable from the position constraints [45]. Kinodynamic problems provide constraints on positions, velocities, and accelerations of all joints [46]. The following is an overview of the RRT planning algorithm.

Given a space X, there is an initial state x_{init}, a goal state x_{goal}, and an obstacle region x_{obs}. RRT creates edges that are paths in x_{free}, where X - $x_{obs} = x_{free}$. Below are the steps of the RRT algorithm.

1. Initialize first state, x_{init}
2. Select a random state from X, x_{rand}
3. Find the closest vertex, x_{near}, to the random state in terms of ρ, a predefined distance metric in X
4. Select and input u that minimizes the distance from x_{near} to x_{rand}, ensuring that it avoids collisions
5. Create a vector from x towards x_{goal} of magnitude rho
    - The vector ends at x_{new}, which is added as a vertex to the tree
6. Add an edge from x_{near} to x_{new}
7. Repeat until x_{goal} is found

There are many advantages to using the RRT search algorithm. First, the expansion using RRT is heavily biased toward unexplored areas of the state space. This increases the chance of finding the goal state sooner. In addition, the distribution of vertices approaches the sampling distribution, producing consistent behavior. The search is also probabilistically complete under general conditions and the graph remains connected with minimal edges. It is simple to implement, leading to high performance, and can create entire paths without human interaction [47].

Figure 6 is a visual representation of the exploration performed by RRT. The red dot is the starting point, the blue dot is the goal point, the black boxes are obstacles, and the red line is the path from the start to goal found [48]. All the green lines (referred to as the tree) represent edges that the algorithm has explored, but are not on the path to the goal point.

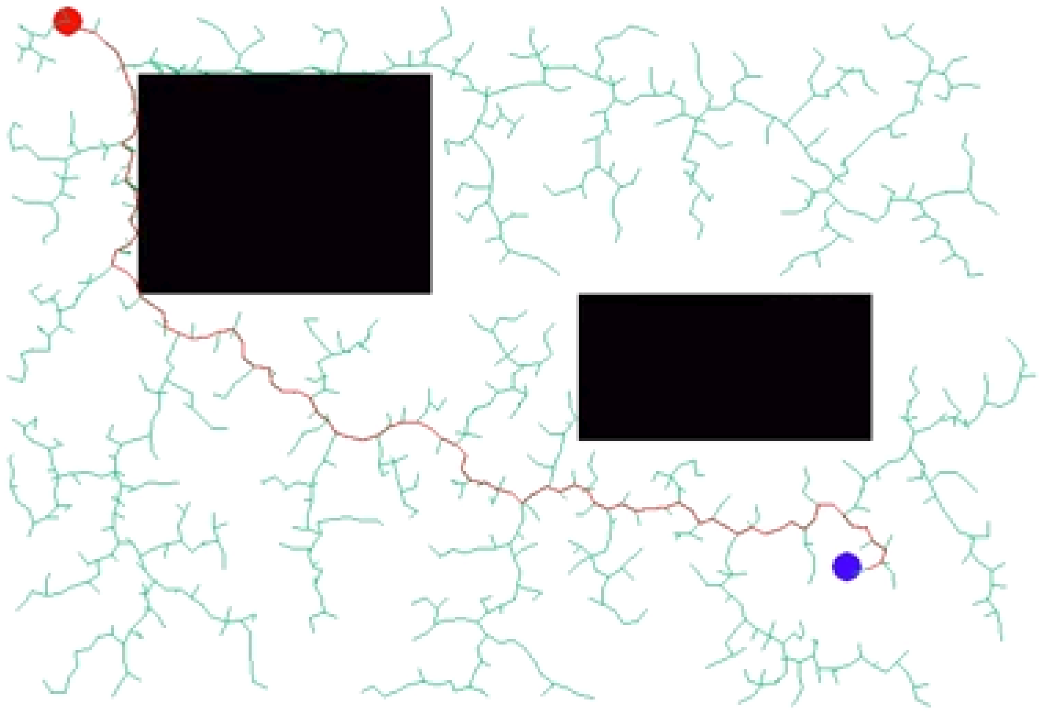


Figure 6: Exploration using RRT

RRT*

The RRT* algorithm is a variation of RRT that is more efficient because the tree is more organized. It is an efficient incremental sampling-based algorithm with provable optimality properties, as opposed to RRT whose quality is unknown but is almost surely non-optimal. The complexity of RRT* is asymptotically within a constant factor required of RRT because it is a modified Rapidly-exploring Random Graph (RRG) as opposed to a tree. RRT* uses a Probabilistic RoadMap (PRM), which is probabilistically complete and constructs a graph of feasible paths offline. RRT* combines the benefits of RRT and RRG. Like RRT, RRT* is a relatively easy extension to motion planning problems with differential constraints and can cope with modeling errors. Like RRG, RRT* is asymptotically optimal and computationally efficient [49]. Figure 7 is an image comparing the paths explored by both RRT and RRT* [50].

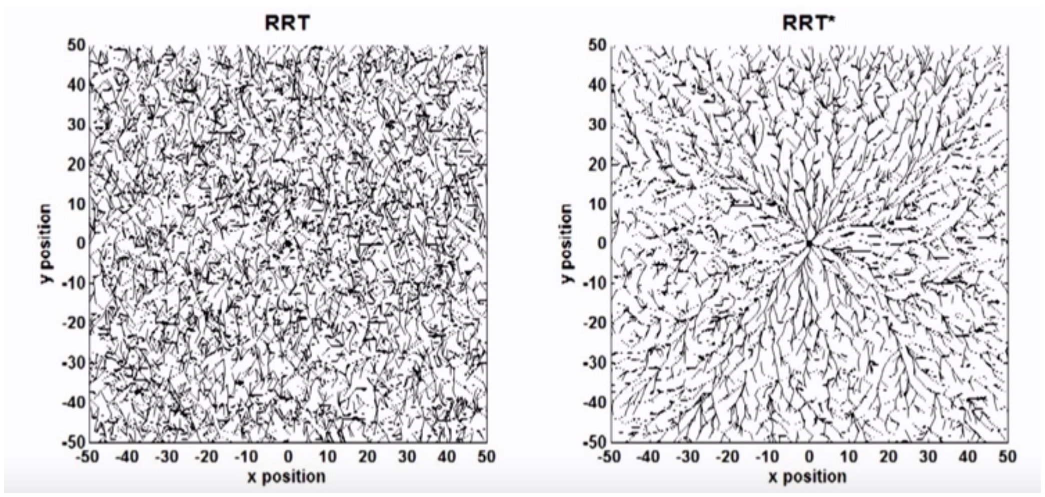


Figure 7: Exploration of RRT (left) and RRT* (right)

ARA*

Anytime Repairing A*, or ARA* is a variation of the A* search that has provable bounds on suboptimality. A* search is a best-first search algorithm that computes path costs, f(n), as the sum of the cost from the start node to the current node, g(n), and the estimated cost of the cheapest path from the current node to the goal node, h(n) [21]. The A* cost function is shown in Equation 4.

$$f(n) = g(n) + h(n) \tag{4}$$

Provided that the estimated costs are relatively accurate, A* search is complete and optimal. Anytime algorithms are useful when time is limited because they find an initial solution very fast and then improve the solution until there is no time left. Sub-optimality bounds are used to set limits on the quality of a plan and decide if the existing plan is good enough or if it is necessary to continue improving the plan.

The naive approach to ARA* is to run a series of A* searches with decreasingly inflated heuristics. However, it is computationally inefficient because each iteration repeats what was done in the previous iterations. A non-naive ARA* approach eliminates these inefficiencies by reusing previous search efforts. One experiment demonstrated that finding a path on a simulated robot arm with six degrees of freedom was six times faster using ARA* than A* [51]. There are several examples of ARA* being used for motion planning for pick and place robotic manipulation tasks [52] [53] [36] [38].

Other Relevant Algorithms

CHOMP

The CHOMP (Covariant Hamiltonian Optimization for Motion Planning) algorithm works by formulating the goal and performing a numerical path optimization [33]. CHOMP uses a covariant gradient descent to produce locally optimal trajectories. The algorithm starts with a naive guess of the trajectory, and optimizes the trajectory as the robot moves. A time-independent cost function is used to determine how optimal a possible trajectory is [33]. The cost function has an obstacle component, f_{obs}, which determines how close to an obstacle the trajectory would place the robot, and a prior component, f_{prior}, which determines the accelerations and smoothness of the trajectory [54]. The CHOMP algorithm selects the trajectory with the lowest cost using the cost function.

STOMP

The STOMP (Stochastic Trajectory Optimization for Motion Planning) algorithm is mainly used for kinematic motion planning [33]. This method uses a gradient-free stochastic system for optimization of the path. Since no gradient information is required, STOMP can be used more generally than CHOMP and can avoid the problem of only finding a local maximum. To begin, a vector, is generated. The space around this initial trajectory is explored and the trajectory is iteratively optimized using a cost function that accounts for obstacles, joint constraints, energy requirements, and smoothness [43].

TrajOpt

The Trajectory Optimization (TrajOpt) algorithm produces a locally optimal trajectory that is free of collisions [55]. TrajOpt uses a sequential, convex optimization method that penalizes collisions [33]. The hinge loss function, Equation 5, is used for increasing penalty coefficients as needed [56].

$$l(y) = max(0, 1 - t * y) \tag{5}$$

where t is the intended output (+1 or -1) and y is the classifier score.

Penalty coefficients are the weights applied to each attribute of the cost function. The hinge loss function is used as opposed to other functions, like the zero-one loss function, because it is more versatile and provides a more realistic cost calculation. Figure 8 shows a graph of the hinge loss and zero-one loss functions.

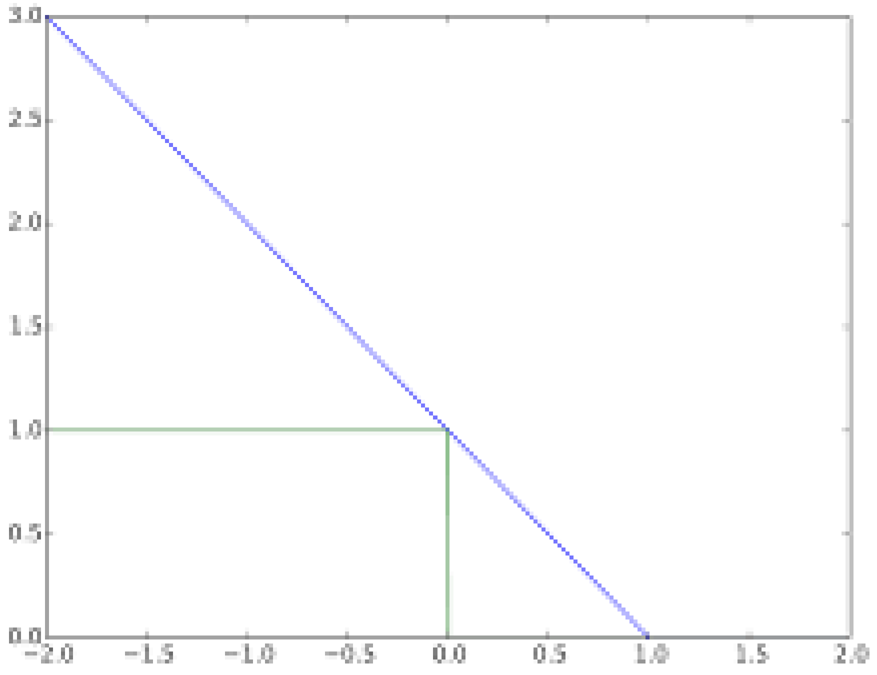


Figure 8: Hinge Loss (blue) vs Zero-One Loss Functions (green)

TrajOpt has been proven faster than alternative trajectory optimization algorithms, such as CHOMP, and is able to solve more problems with higher quality paths than competing solutions. The main differences between CHOMP and TrajOpt are the collision detection and numerical optimization methods [33]. While CHOMP uses Euclidean distance transforms for collision checking, TrajOpt uses convex-convex collision checking, which detects two shapes colliding and calculates the minimum translation required to clear the obstacle. The Euclidean distance transforms determine the amount that each point of the robot needs to move so the robot is no longer in a collision. This method does not provide an overall estimate of the shape of the obstacle whereas the convex-convex collision detection method does. When using Euclidean distance transforms, the points of the robot can potentially move to conflicting locations when avoiding obstacles [55]. While the Euclidean distance method is independent of changes in the trajectory, the convex-convex collision detection method uses a more specific definition of the overall workspace.

2.3.4 Visualization Platforms

There are a variety of visualization platforms designed for robotic use that are particularly helpful in testing and evaluating different motion planning methods. The first, RVIZ, is a 3D visualizer made specifically for the Robot Operating System (ROS) framework. It is a powerful tool that has been used in many robotic projects and supports different robot file types [57]. Gazebo is another simulation tool and can include physics engines and the simulation of sensor data. ROS packages are also available to integrate Gazebo and ROS [58]. Another useful simulation tool is OpenRAVE, a simulation tool with a ROS plugin and 3D physics to provide quality simulation and planning for humanoid robots. OpenRAVE makes it easy to tune planning parameters and port code from simulation to a real robot [59]. A newer robot modeling and simulating tool is Klamp't (Kris' Locomotion and Manipulation Planning Toolbox) that was developed particularly for testing and analyzing algorithms for robot manipulation and locomotion [60]. A visualization platform is important because it is unsafe and much slower to test every iteration on the physical robot. Different visualization platforms will be explored in this project.

2.4 Joint Actuation and Control

After trajectories have been planned for the robot's arms, the joints must be quickly and accurately actuated. There are two main categories of controlling a robot's joints: point to point control and continuous path control. In point to point control, the controller is provided start and stop points only. Within point to point control there are three main methods. In one joint at a time control, each joint is actuated individually to achieve a goal pose. In slew motion, all joints are actuated at the same time, all with the same default speed. In joint interpolation, all the joints are actuated at the same time but the speed of each joint is proportional to the distance to travel. That is, the joint with the longest displacement starts at the default speed and every other joint moves slower according to their displacements. This produces a smooth motion that avoids singularities, making joint interpolation the most popular point to point control method [61].

Continuous path control defines points along the path instead of just start and stop points like in point to point control. To find the path, low order interpolating polynomials with imposed velocities are used [32]. Equations 6 and 7 describe the start and end of the robot arm's trajectory [62].

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \tag{6}$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{7}$$

Continuous path control is more difficult to implement and is more computationally expensive, but it provides a more precise and efficient trajectory because more than just two points are used to define it. These control methods provide different ways for a robot to execute a planned trajectory.

2.4.1 Forward Kinematics and DH Parameters

Forward kinematics involves mapping coordinates from the robot's joint space to Cartesian space [31]. Forward kinematics uses transformation matrices to determine translational and rotational transformations. To select the reference frames of the robot's joints, DH parameters can be used to represent the axis between joint pairs [63]. There are four different DH parameters: a, α, d, and θ, shown in Figure 9, where O_0 and O_1 represent the origins of two of a robot's joints.

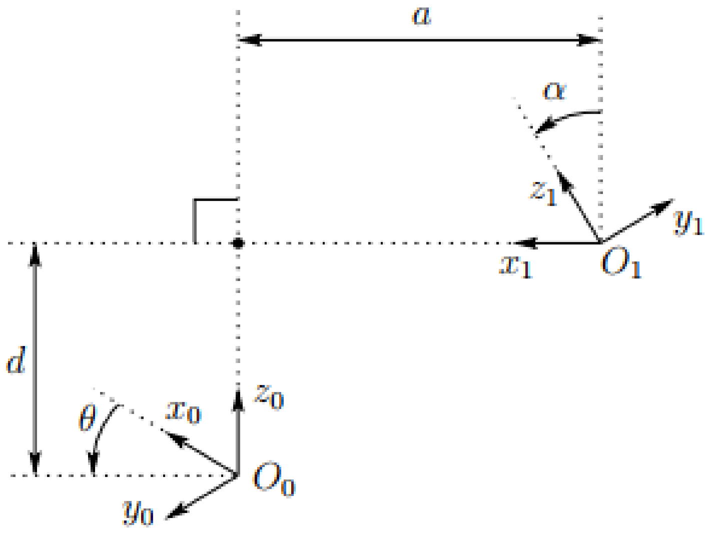


Figure 9: Two Robot Joints and DH Parameters

Between two joints, a is the length of the common normal, α is the angle about the common normal, d is the offset along the z_0 axis to the common normal, and θ is the angle about the z_0 axis [64]. From these parameters, a transformation matrix is constructed shown in Equation 8.

$$T = \begin{bmatrix} cos(\theta) & -sin(\theta)cos(\alpha) & sin(\theta)cos(\alpha) & rcos(\theta) \\ sin(\theta) & cos(\theta)cos(\alpha) & -cos(\theta)sin(\alpha) & rsin(\theta) \\ 0 & sin(\alpha) & cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \qquad (8)$$

where R is the rotation matrix and T is the translation matrix. Using DH parameters is a convenient way to define a coordinate frame for controlling a robot's joints.

2.4.2 Inverse Kinematics

Inverse kinematics (IK) determine the set of joint configurations that correspond to the desired position of an end effector of a robot [65]. These calculations are used in robotics, as well as computer animation, ergonomics, gaming, and others. Inverse kinematics can be applied to robots specifically as a "method for computing the posture via estimating each individual degree of freedom in order to satisfy a given task that meets user constraints" [65]. IK calculations can also be applied to a robot to compute the area and shape of the reachable workspace of a robot, plus its end effector [66].

Inverse Kinematics solvers, or IK solvers, are algorithms used for finding solutions to inverse kinematics problems with added constraints [67]. These constraints can include joint limits, joint singularities, or instantaneous joint velocities. While IK problems can be solved without the use of IK solvers, it is often not practical because inverse kinematics problems can have infinite solutions [67].

There are several different inverse kinematics solvers. One of these methods is called FABRIK, or Forward And Backward Reaching Inverse Kinematics. This method is a heuristic iterative method, which means it finds each joint position by locating a point on a line, makes computation costs, and then produces visually realistic poses for the robot [65]. FABRIK is efficient for both simple and complex position calculation problems, and produces good results for all problems quickly. Other IK solvers include CCD, Jacobian IK solvers, target triangulation, TRAC IK, FAST IK, and KDL IK [65]. Many of these solvers have built-in ROS packages to make IK calculations in a robotic system easy to implement.

2.5 Shared Autonomy

Inverse kinematics calculations are necessary so the robot controller can receive a desired joint configuration to execute. Precise execution of joint configurations is necessary to complete the sewing task. Previous examples of robotic sewing tasks were fully autonomous, requiring no user input during the sewing task. However, robotic systems with shared autonomy incorporate a human operator. Shared autonomy describes a control task that is accomplished using both a robot and a human operator. Overall, the two main goals of shared autonomy are to predict the user's goal and provide assistance to accomplish that goal [68]. One example is where the robot is controlled by the human operator using teleoperation, directly mapping the human operator's inputs to the robot's operation [69]. Shared autonomy can alleviate some of the challenges of teleoperation, such as filtering noise from input devices and utilizing all of the degrees of freedom of the robot that the human cannot directly control. In one manufacturing application, a human operator controlled multiple robots remotely, checking the robots' activity using a simulated display [70]. In another application, an assistive shared autonomous robot augmented the motor abilities of patients with motor impairment to perform tasks that need fine motor control for the human [71].

Other forms of shared autonomy do not involve teleoperation, but require a human operator for other information [72]. In one application, a tablet is used as an intuitive user interface and abstracts the robot's motion so that the human only needs to input a desired task [73]. When a robot can correctly execute the procedure to complete that task independently, it can diminish the effects of time delays and simplify the control interface for the human operator [74]. Regardless of the level of teleoperation involved, shared autonomy utilizes the complementary advantages of humans and robots.

2.5.1 Input Devices

When implementing a shared autonomous system, it is important to consider which robot control device the user will find most intuitive. Because there are a wide variety of input devices available for robotic operation, it is important to determine the criteria the device must meet. These criteria also depend on the specific use case and can include cost, intuitivity, or accuracy. Simple devices such as buttons, switches, keyboards and mice are easy to use and implement but do not provide advanced functionality necessary to directly control a robot. Several companies design industrial robotic controllers that have increased functionality but require a steep learning curve to operate [75]. Other robotic systems utilize common handheld controllers used in video games that can track movements. These devices can be very intuitive to use, but typically have low accuracy [76]. A variety of input devices were researched and evaluated in this project.

3 Concept of Operations

Many different stakeholders are involved in this project of sewing using with a shared autonomous robotic system. This chapter defines all the stakeholders as well as use cases and user stories that apply to them. This chapter also outlines the function and nonfunctional requirements that the project needs to fulfill.

3.1 Stakeholders

These are the relevant stakeholders and information regarding them as it pertains to this project. Table 1 shows the various stakeholders and their roles in this project.

Title	**Description**	**Role**	**Representation**
Students	Developers	Directly involved	Self
Advisors	Advise/grade	Directly involved	Self
New Balance Innovation Team	Sponsor	Funding/Project Specifications	Self
WPI	Sponsor	Graduation Requirements	Registrar personnel
RBE/CS Departments	Sponsor	Funding/MQP Requirements	Advisors

Table 1 continued from previous page

New Balance Seamsters	Use robot	Operate robot	Self
Future Students	Continue work on project	Continue work on project	Not represented

Table 1: Stakeholders and Relevant Information

3.2 Use Cases

The following use cases, described in Tables 2, 3, and 4, are used to define the problem as it pertains to the relevant stakeholders.

Name	Autonomous Mode Operation
Actor (s)	New Balance Seamster
Initial Conditions	Vamp is assembled Toe guard is cut
Flow Of Events	1. Seamster places vamp and toe guard in the workspace 2. Seamster powers on robot, UI, and sewing machine 3. Seamster selects ‘auto’ on the screen 4. Robot picks up the vamp and toe guard 5. Robot aligns the pieces 6. Computer vision system determines points along sewing pattern

Table 2 continued from previous page

	7. Motion planning subroutine determines end effector positions based on sewing pattern points 8. Motion planner determines joint space configurations based on end effector positions and IK calculations 9. Robot actuates the planned trajectory
Exit Conditions	Vamp and toe guard have been sewn together Robot releases assembled pieces

Table 2: Use Case 1

Name	Assist Mode with Input Device as Sewing Pattern
Actor	New Balance Seamster
Initial Conditions	Vamp is assembled Toe guard is cut
Flow Of Events	1. Seamster places vamp and toe guard in the workspace 2. Seamster powers on robot, UI, and sewing machine 3. Seamster selects 'assist' on screen; opens up input device screen 4. Seamster imitates sewing pattern using the input device 5. Seamster selects desired sewing trajectory on the input device screen 6. Robot picks up and positions the vamp and toe guard

Table 3 continued from previous page

	7. Motion planning subroutine determines end effector positions based on sewing pattern points 8. Motion planning subroutine determines end effector positions based on sewing pattern points 9. Computer vision system provides visualization of workspace to the user 10. Robot actuates the planned trajectory
Exit Conditions	Vamp and toe guard have been sewn together Robot releases assembled pieces

Table 3: Use Case 2

Name	Assist Mode with input device as end effector positions
Actor	New Balance Seamster
Initial Conditions	Vamp is assembled Toe guard is cut
Flow Of Events	1. Seamster places vamp and toe guard in the workspace 2. Seamster powers on robot, UI, and sewing machine 3. Seamster selects assist on the screen - opens up an input device screen 4. Seamster imitates sewing using input device 5. Seamster selects desired end effector positions on input device screen

Table 4 continued from previous page

	6. Robot picks up and positions the vamp and toe guard 7. Motion planner determines joint space configurations based on end effector positions and IK calculations 8. Computer vision subsystem provides visualization of workspace to user 9. Robot actuates the planned trajectory
Exit Conditions	Vamp and toe guard have been sewn together Robot releases assembled pieces

Table 4: Use Case 3

3.3 User Stories

The following user stories are specific to each stakeholder and demonstrate general criteria for the system based on the needs of the stakeholders.

As a New Balance factory employee, I want a robot that is safe to work near so I know I won't get hurt.
As a New Balance factory employee, I do not want to work on the factory floor because it is cramped, loud, and uncomfortable.
As a New Balance factory employee, I want very simple user interface so I can learn how to interact with the robot quickly.
As a New Balance factory employee, I don't want the robot to do everything

because I don't want to lose my job.

As a New Balance factory employee, I want the tolerance of the fabric input to be not too tight because I don't want to have to focus too much on how I hand the fabric to the robot.

As a New Balance expert employee, I want to show the robot how to sew so that it can teach other employees.

As a New Balance expert employee, I want to easily and effectively teach the robot so that I can show it how to accurately sew a shoe.

As a New Balance expert employee, I want to easily be able to correct a small segment of the trajectory so I can improve the robot's product.

As a New Balance Innovation Team Member, I want a robotic system to decrease the number of sewing defects to reduce waste and costs.

As a New Balance Innovation Team Member, I want to increase the number of shoes assembled to increase our sales.

As a New Balance Customer, I want a robotic system to be implemented to decrease the cost of the product for me.

3.4 Requirements

3.4.1 Functional Requirements

The functional requirements of the system describe what it should do and the outcomes of the product. Below is a list of the functional requirements of the system.

- Use computer vision to identify the location of the vamp
- Sew the toe guard onto the vamp
- Plan trajectories in joint space
- Have a user-friendly control interface
- Safely interact with factory workers
- Use computer vision
- Robot must avoid collisions
- Trajectories are optimal
- Implement an intuitive teleoperation method
- Eliminate tremors and collisions from human input
- Manipulate the soft material appropriately
- Be adaptable for future applications (different materials, sizes, etc.)
- Fit well in a factory setting

3.4.2 Nonfunctional Requirements

The nonfunctional requirements are criteria of how the system operates and performs. Most contain numeric specifications, shown below.

- Sew a pattern within 1mm
- Do not come within 3 inches of an obstacle
- Complete the sewing task within 60 seconds
- Sew less than 1% defects

4 Technical Documentation

4.1 Workspace Layout

The workspace is the area around the robot, including the objects that the robot can potentially interact with. Figures 10 and 11 shows the workspace in the lab and Figure 12 shows the workspace in simulation.

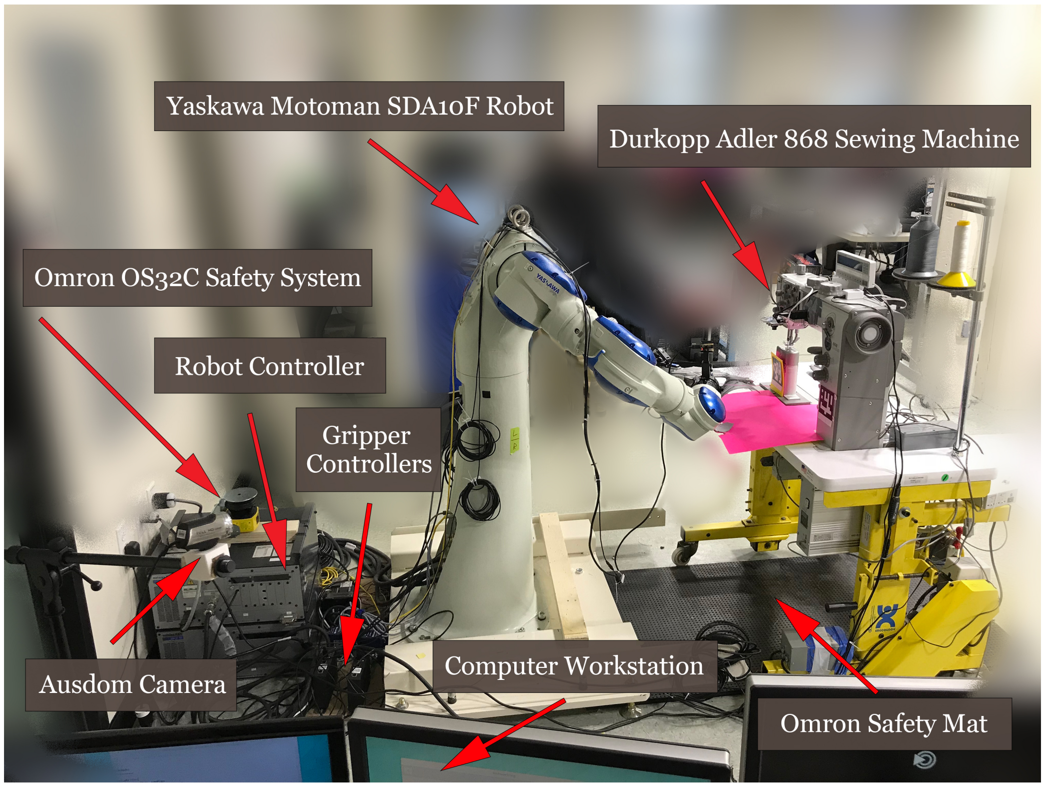

Yaskawa Motoman SDA10F Robot
Durkopp Adler 868 Sewing Machine
Omron OS32C Safety System
Robot Controller
Gripper Controllers
Ausdom Camera
Computer Workstation
Omron Safety Mat

Figure 10: Robot workspace with objects labeled: Right View

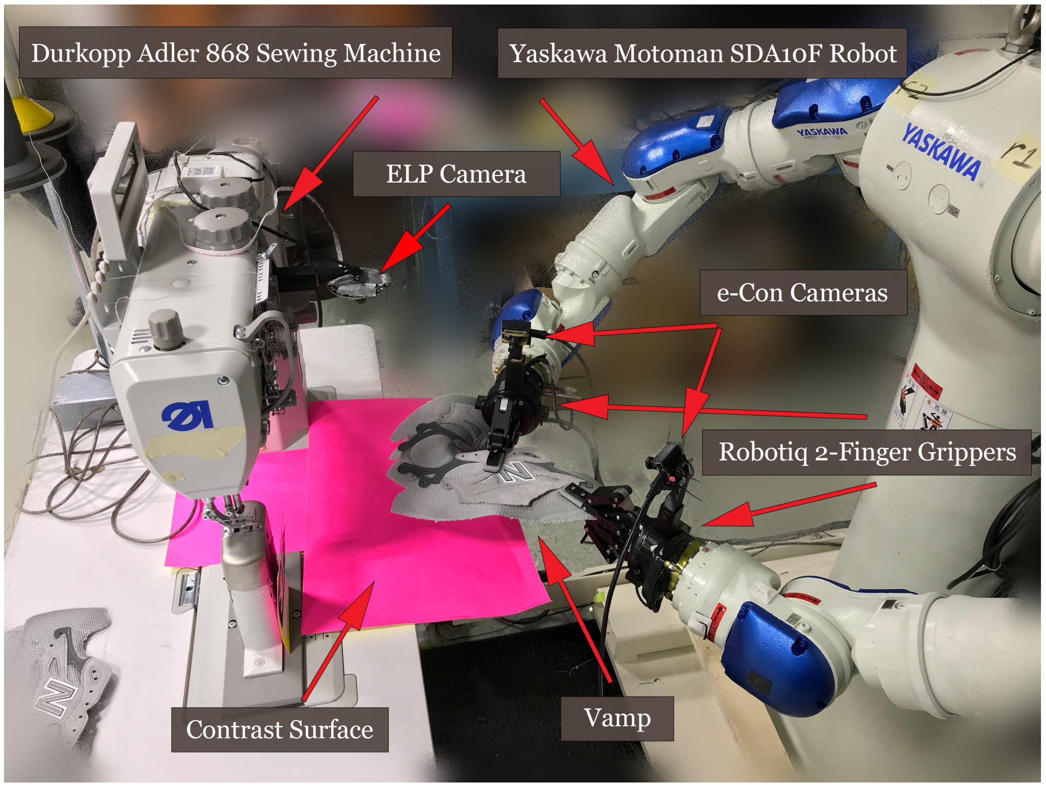

Durkopp Adler 868 Sewing Machine
Yaskawa Motoman SDA10F Robot
ELP Camera
e-Con Cameras
Robotiq 2-Finger Grippers
Contrast Surface
Vamp

Figure 11: Robot workspace with objects labeled: Left View

Note that the robot controller and safety system are not represented in simulation since they cannot be reached by the robot and therefore are not relevant to motion planning problems. The construction of the simulation is discussed in Section 4.6.1.

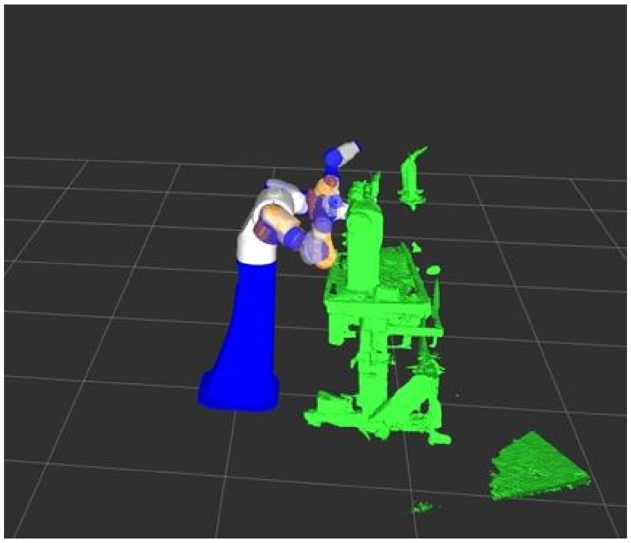


Figure 12: Robot with Sewing Machine Simulated in RViZ

The fabric pieces the robot manipulates are the vamp and the toe piece, shown in Figure 13. The vamp forms into the top of the shoe, and the toe piece covers the seam between the vamp and the sole.

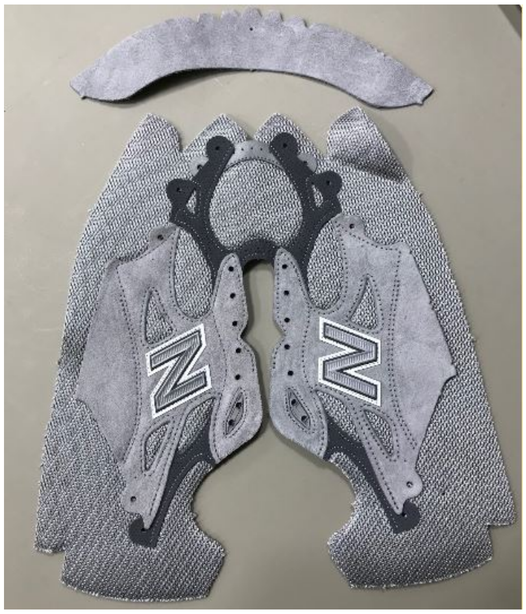


Figure 13: Vamp (bottom) and Toe Piece (top)

Described in Section 4.2, the robot, a Yaskawa Motoman SDA10F, has its own controller and communicates with it over Ethernet. Each of the grippers communicate with their own Robotiq K-Model controllers over Ethernet as well. The gripper controllers can communicate with the computer using USB or Ethernet, and USB was selected for this project. Figure 14 shows the communication between all the components in the workspace.

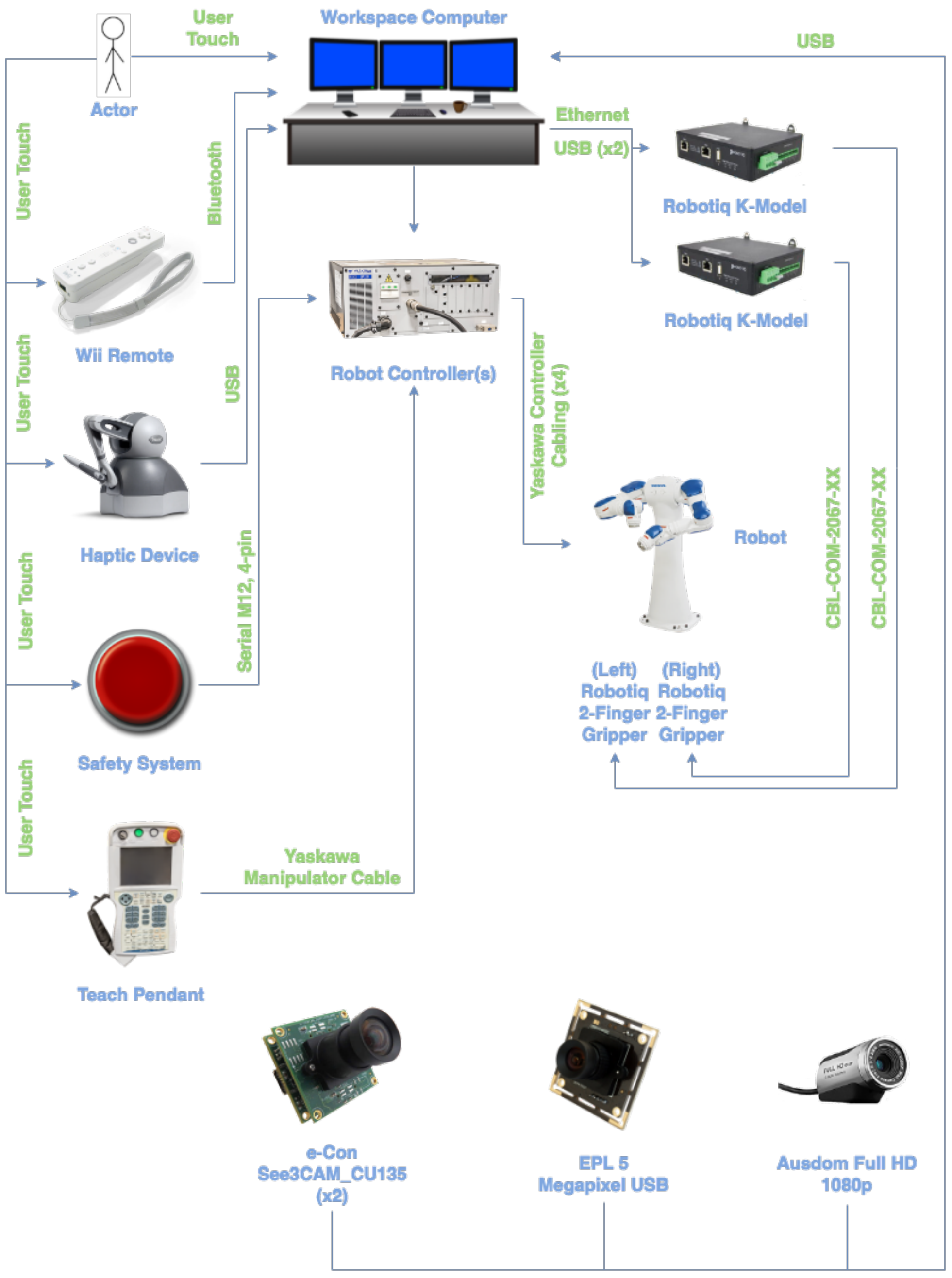


Figure 14: Communication between all Components

A safety system was installed to protect human workers who enter the robot's workspace. The OMRON OS32C safety sensor was used to detect when humans get close to the workspace (Warning Zones) and then shut the robot's servos off when humans are within reach of the robot (Safety Zone). Figure 15 shows a simulation of the boundaries of the zones.

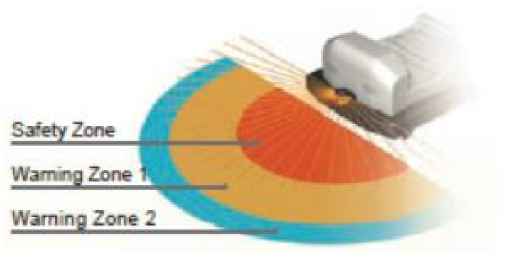


Figure 15: Warning and Safety Zones of OMRON Safety Sensor

The safety system also includes a mat shown in Figure 16 that is located in front of and beneath the sewing machine. The sewing machine is in the laser's blind spot, so this ensures no human workers are standing where the safety system cannot detect.

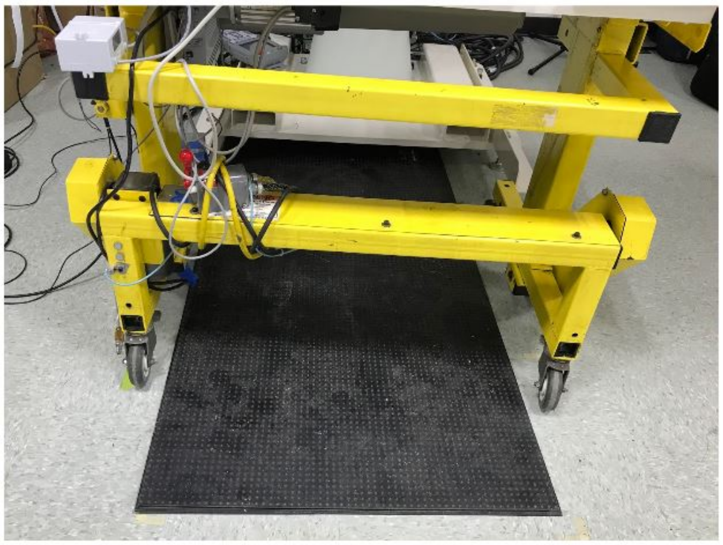


Figure 16: Safety Mat (Black); turns off Robot's Servos when Stepped on

There are visual and auditory warnings that display the state of the safety system. The safety warning zones are shown above in Figure 15. The light rod on the top of the robot shown in Figure 11 has three LEDs: one red, one yellow, and one green. When Warning Zone 1 is entered, the yellow light flashes and there is a loud, pulsing beep. When the Safety Zone is entered, the red light flashes and the robot is halted. When the user leaves the workspace, the red and yellow lights both illuminate, indicating that the system needs to be reset using the reset button at the workstation. Once the safety system is reset and the zones are clear, the green light indicates that the workstation is clear of human workers. The system also utilizes three emergency-stop buttons: one next to the computer, one on the robot controller, and one on the teach pendant. Each button allows the user to halt the robot.

The sewing machine used is a Durkopp Adler 868 sewing machine shown in Figure 17.

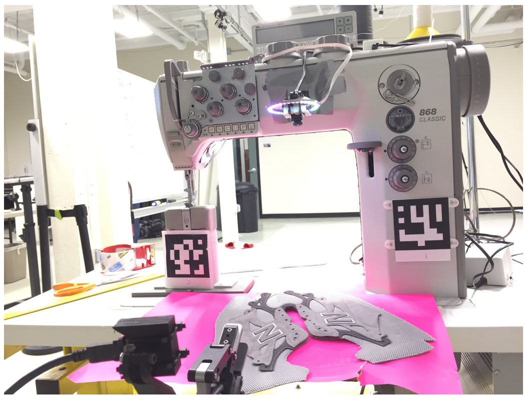


Figure 17: Durkopp Adler 868 Sewing Machine

This system is a twin needle post bed machine with a built in sewing motor, which is capable of variable-rate repetitions via a continuous foot pedal. This means the sewing machine can be operated over a range of speeds. The sewing machine is located 71 cm in front of the robot. It is on an adjustable table, allowing the sewing machine height to be configured. For this project, the sewing table was set to a height of 82 cm.

Also included in the workspace are the cameras that the computer vision subsystem uses to identify landmarks on the vamp and that the operators use to monitor sewing. Two of the four cameras are e-Con See3CAM_CU135 USB webcams, one is an ELP 5 Megapixel USB, and the fourth is a Ausdom Full HD 1080P. The two e-Con See3CAM_CU135 cameras are mounted on the robot's wrists using custom brackets, shown in Figure 18.

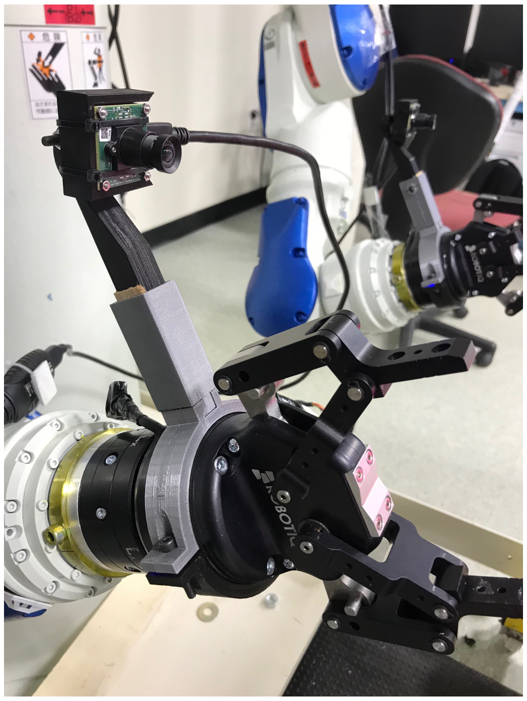


Figure 18: Camera Mounted on (Right) Wrist

The ELP camera is mounted on the front of the sewing machine facing down, as shown in Figure 19

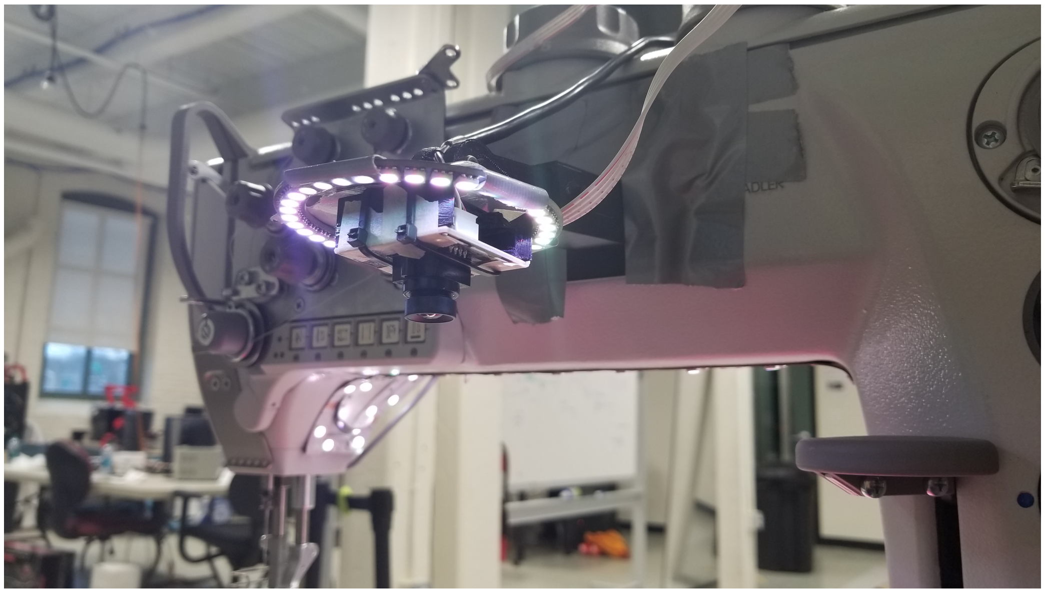


Figure 19: Camera Mounted on the Sewing Machine

White LEDs were mounted around the sewing machine camera and on the sewing machine to maximize light in the workspace and keep the footprint small. Figure 20 shows the lighting mounted on the sewing machine.

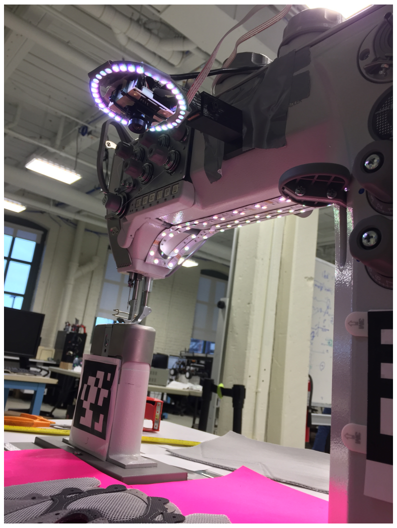


Figure 20: Lighting mounted to sewing machine

The EPL 5 Megapixel USB camera is mounted on a tripod and is depicted in Figure 21.

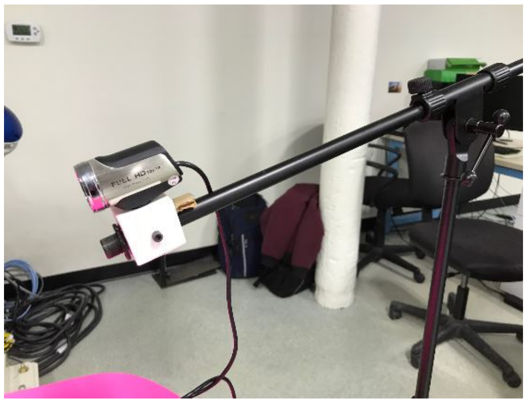


Figure 21: Camera Mounted on the Tripod

The cameras are used by the operator to monitor the robot's sewing and by the computer vision subsystem to generate sewing waypoints. The details of each workspace component and operation are discussed in-depth in this chapter.

4.2 The Robot

The robot used in this project is a Yaskawa Motoman SDA10F, shown in Figure 22.

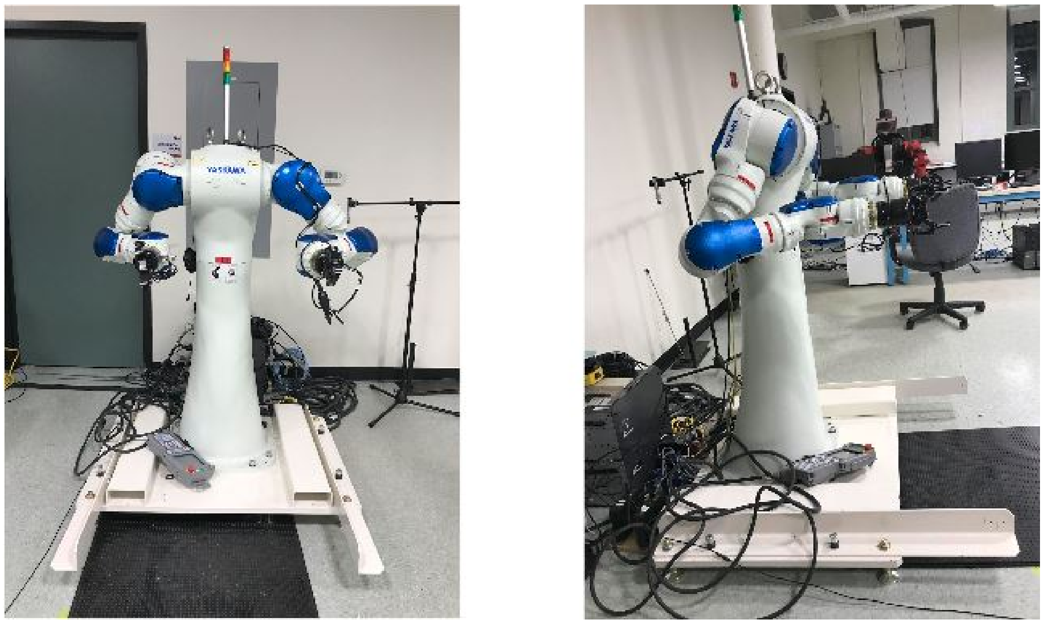


Figure 22: Yaskawa Motoman SDA10F

This robot is a 15 degree of freedom, dual arm robot with a maximum payload of 10kg per arm. Each arm has seven rotational joints, and there is an additional joint at the base to rotate the torso. Each arm has a maximum horizontal reach of 720mm and maximum vertical reach of 1,440mm. The joints in the arms have a rotation between ±110 to ±180 (see Appendix A).

The joint angles for each motor are known through absolute encoders placed at each joint servo. The repeatability of motors is ±0.1mm, which is ideal precise sewing. A model of the robot with the joint names is shown below in Figure 23.

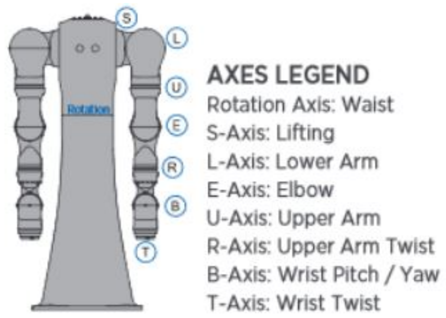


Figure 23: SDA10F Model with Joint Names

The SDA10F comes with the option of three controllers: the DX200, FS100, or MLX200. The controller for this particular robot is the FS100 controller. Two controllers are required for control of the SDA10F because it has 15-DOF. The FS100 controller is ideal for handling small payloads. The compact nature and high communication speeds and of the controller make it suitable for this application. This controller also comes with a handheld pendant for manipulating the robot and creating programs, pictured below in Figure 24.

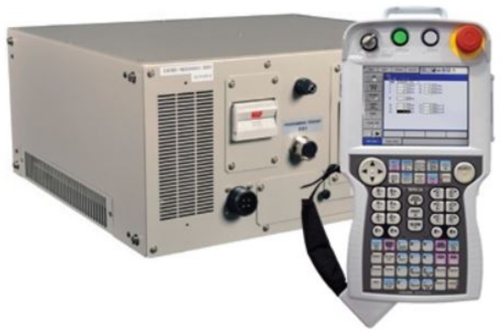


Figure 24: FS100 Controller (left) and Pendant (right)

The pendant gives the option to switch between teach, play, or remote mode. Teach mode allows the operator to use the pendant to create and save robot programs. Play mode executes a selected program on the robot. Remote mode allows the robot to be controlled by a system other than the pendant. The pendant has three other buttons along the top: start, hold, and emergency stop. Start is used to run a program with the robot in play mode, hold is used to pause a program while it is running, and the emergency stop is pressed when the robot needs to come to a complete halt immediately.

Inside the controller are general settings for the robot. These can be accessed and adjusted through the pendant. Some of these settings include the maximum joint speeds for each joint during the different modes of operation. The robot controller also determines if it should perform a safety stop depending on the signals from each safety sensor attached.

The controller also contains multiple digital input and output connections. External elements, such as end of arm tooling, safety systems, and sensors, can be connected, controlled, and read using the robot's controller and pendant. The controller also has 10Base-T/100Base-TX Ethernet for connecting to the robot over a network or directly with a computer. The laser safety sensor and safety mat are connected to the robot controller via the safety input connections.

This robot is suited to the sewing task because of the ±0.1mm repeatability and the high dexterity of the robot. The dexterity comes from the fifteen controllable degrees of freedom. The robot has two separate arms, mounted so they mirror each other, resembling the structure and position of human arms. This will make it easier for the robot to replicate human like motion.

4.2.1 End of Arm Tooling

The end of arm tooling chosen for this application is the Robotiq 2-Finger Adaptive Robot Gripper with a maximum opening of 85mm. The gripper is able to close with the fingers coming to a point, the Encompassing Grip, or with the fingers parallel to each other, the Parallel Grip (Figure 25). The gripping type is important because it affects how the fabric is held. The parallel grip was chosen for this project because it increases contact area with the fabric. This will help manipulate the soft fabric in a predictable manner.

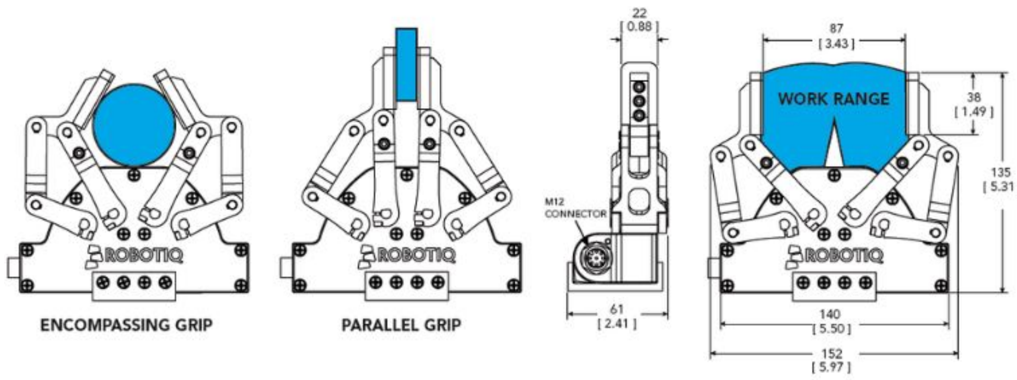


Figure 25: 85mm Robotiq 1-Finger Adaptive Robot Gripper

The grippers are able to exert a force from 20N to 235N. The full list of specifications of the 85mm Robotiq 2-Finger Adaptive Robot Gripper can be found in Appendix B. End effectors to grip the shoe fabric needed to be chosen. The factors used to evaluate the grippers were degrees of freedom, grip strength, compatibility with other hardware and software, cost, manipulability, reliability, and contact area size. The decision making matrix comparing different grippers is shown in Appendix C.

Both arms of the robot were fitted with identical grippers. Between the right arm wrist and the right gripper, a Robotiq Force-Torque sensor was installed. Although this was not used during this project, it remains on the robot for the possibility of future research. Each gripper has a separate controller, which communicates with the computer via USB. The computer can tell each gripper when to open and close, how much to open or close, how quickly to open or close, and the force to use. The position can also be controlled by sending a value from 0-255 to the gripper controller. This gives the gripper a resolution of 0.332mm. Each of the grippers as well as the robot controller have their own ROS nodes, described in detail in Section 4.3.

4.3 System Overview

This section gives a general description of the system as a whole.

4.3.1 Computer Requirements

Computer

The sewing task had to be done in sixty seconds for this project iteration. This is reasonable to achieve and quick enough to be useful in a production setting. This speed requirement dictated that the software had to quickly process images, generate waypoints, and execute the planned trajectories. To help accomplish this goal, efficient hardware was chosen. A desktop computer workstation was repurposed to serve as the processing center and upgraded with a Solid State Drive, more RAM, and more USB 3.0 ports for the cameras. The workstation computer came equipped with a dedicated graphics card and a quad core i7 Intel processor. A second computer workstation was also prepared and later repurposed for Windows applications.

Computer Operating System

The operating system on the workstation was Ubuntu Linux Mate 14.04. It was chosen because it is relatively light on system resources when idle, while still maintaining the usability of a normal Ubuntu install. The Ubuntu version, 14.04, was needed as the Robotiq package needed for the grippers only supports up to ROS Indigo, which only runs on Ubuntu 14.04.

Security

The system was designed to be secure by minimizing attack vectors. To do this, the communication with the robot was limited to one computer, as opposed to connecting the robot to the Local Area Network (LAN). Only a single computer was used to run all of the ROS nodes, so only one computer needed to be secured.

4.3.2 ROS Architecture

In order to work simultaneously on the different subsystems, one or more ROS nodes were used for each subsystem that communicate over ROS topics. In addition, using multiple nodes allows for easy addition of features, as each feature is simply a node that subscribes and publishes to ROS topics. Figure 26 below is a diagram that summarizes the different nodes.

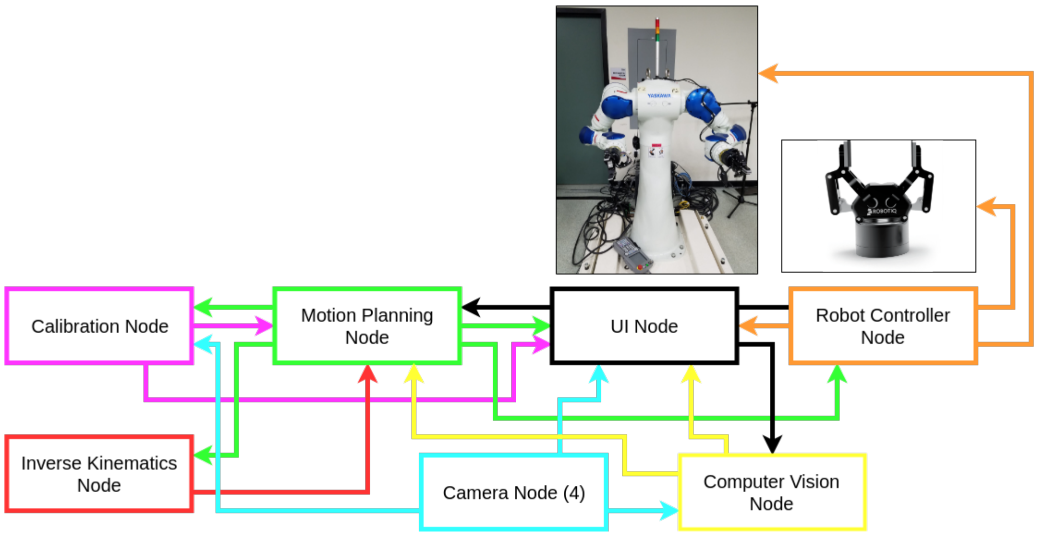


Figure 26: ROS Node Diagram

In summary, there are ten different nodes. Four nodes are camera nodes, one for each camera, and they provide the same functionality. There is a calibration node that handles the calibration during startup and publishes the distance and orientation offsets of the sewing machine. Another node is the computer vision node, which takes the view from the sewing machine camera, determines the position of the vamp, and publishes the positions of designated waypoints. The motion planning node subscribes to the waypoints on the vamp published by the computer vision node and determines the trajectory for the robot to follow. The trajectory is then published to a topic to which the inverse kinematics node subscribes. The inverse kinematics node calculates the robot's joint angles in order to execute the calculated trajectory. The joint angles are published, and the robot controller node subscribes to them. The robot controller node then publishes the trajectories in order to command the robot to move. The robot controller node also communicates with the gripper controllers, and instructs them to open and close the grippers based on commands from the user interface node. The user interface node is the node that begins the sewing process and integrates the motion planning, computer vision, robot controller, and input device nodes. A benefit of independent nodes for each feature is that they can be launched and used independently. This is useful for debugging and testing as well as reducing complexity for the addition of future tasks.

4.3.3 Program Flow

Figure 27 below is a flowchart depicting how a user uses the system.

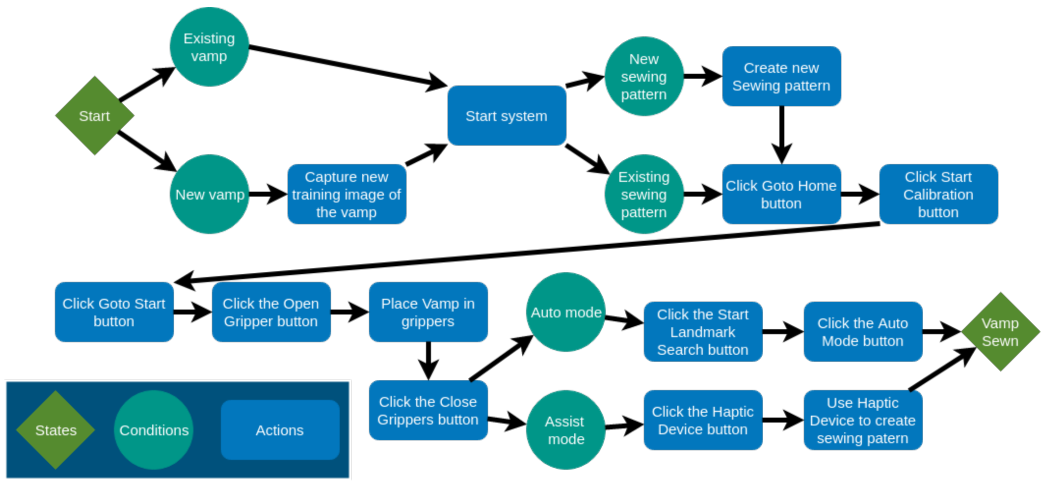

Start
Existing vamp
New vamp
Capture new training image of the vamp
Start system
New sewing pattern
Create new Sewing pattern
Existing sewing pattern
Click Goto Home button
Click Start Calibration button
Click Goto Start button
Click the Open Gripper button
Place Vamp in grippers
Click the Close Grippers button
Auto mode
Click the Start Landmark Search button
Click the Auto Mode button
Assist mode
Click the Haptic Device button
Use Haptic Device to create sewing patern
Vamp Sewn
States
Conditions
Actions

Figure 27: Flowchart of System Progress

The system has two modes: auto mode and assist mode. In auto mode, the robot sews autonomously with no user input, while in assist mode, the user teleoperates the robot using an input device. To start the process, two predefined poses were saved shown in Figure 28.

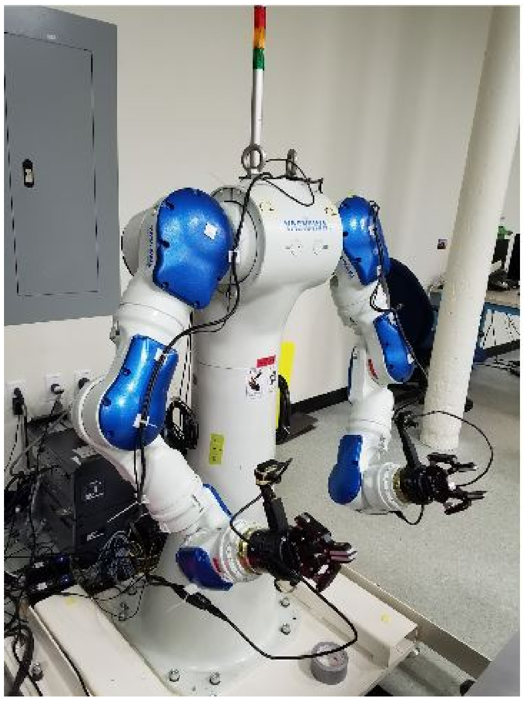


Figure 28: Robot Home Position

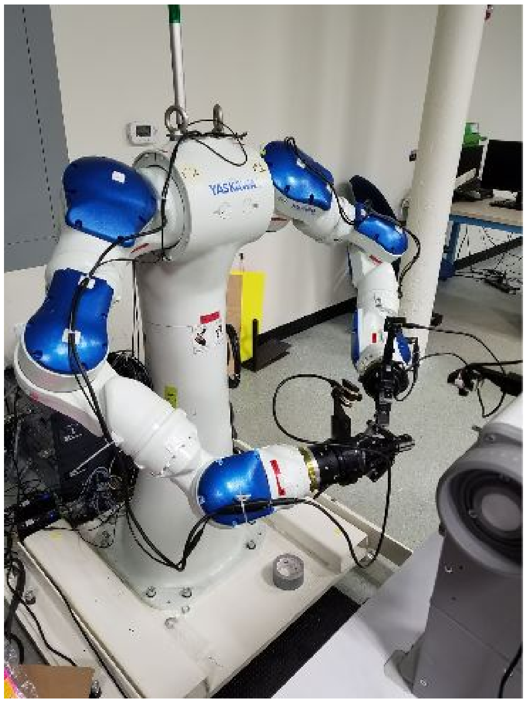


Figure 29: Robot Start Position

4.4 User Interface

The user interface (UI) allows the user to control and understand the sewing process. The UI must be intuitive and communicate with other subsystems. The intended users are seamsters or factory workers, who are not expected to have knowledge in computer programming. These users need an understandable UI that they can easily use with minimal training. The full lists and explanations of desired UI features are available in Appendices D, E, F, and G. Along with the UI, an input device is needed to enable the user to accurately control the robot's end effector position while in assist mode.

The input device needs to be intuitive to use and record accurate data. This section describes the user interface design and functionality as well as the input devices researched.

4.4.1 UI Design

In the process of creating the UI, several different programming languages and software packages were researched such as QT, RQT, Matlab, and Python. Since RQT is the framework most frequently used for developing user interfaces with ROS, it was used with C++ to design the first version of the UI. However, this version only worked as a standalone plugin and could not be integrated with the rest of the system. Therefore, the UI was redesigned using a different programming language and UI development library. One possibility was Matlab and its ROS library, but this solution was not implemented because Matlab is both computationally and financially expensive. Python was chosen instead because the team members were more familiar with it, and it can utilize the Tkinter software package for simple UI designs. This Python UI became the final design.

RQT/QT C++ Layout Design

The software development platform QT is commonly used for designing user interfaces and can be used in programs in different languages. RQT is an QT based GUI design platform for integration into ROS systems. The first UI implementation for this project was programmed in C++ using the QT framework. It was easier to program and test the core functionality of the UI in QT before integrating it into RQT to add ROS functionality. Figure 30 below shows a the first UI in C++ using QT.

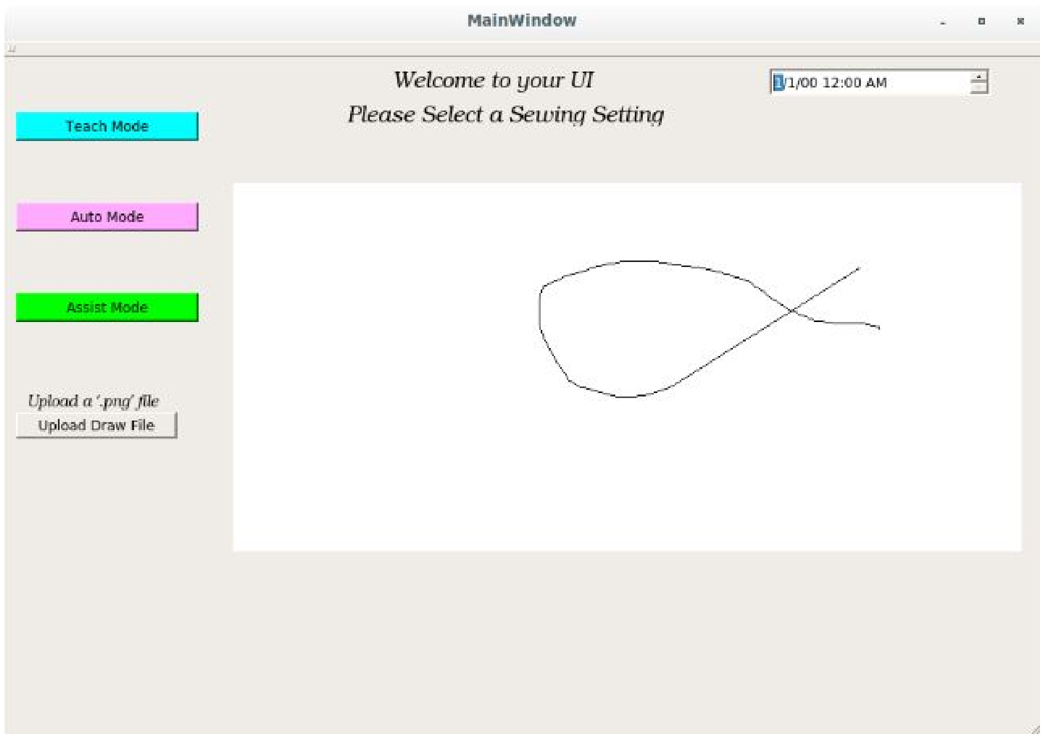


Figure 30: Initial UI Design in C++ with QT

The three colored buttons represent the sewing modes of the system, although only the assist mode and auto mode were implemented. One initial idea for the UI was to allow the user to upload a desired pattern from a file and convert the image into a desired sewing pattern. This was implemented in the first UI. However, it was unnecessary for users to upload an image since they could use the input devices to generate trajectories so this feature was removed from future versions of the UI. The final UI using RQT, pictured in Figure 31, was able to be opened and users were able to click buttons. This allowed the developer to test the camera views separately before integrating them into the ROS system.

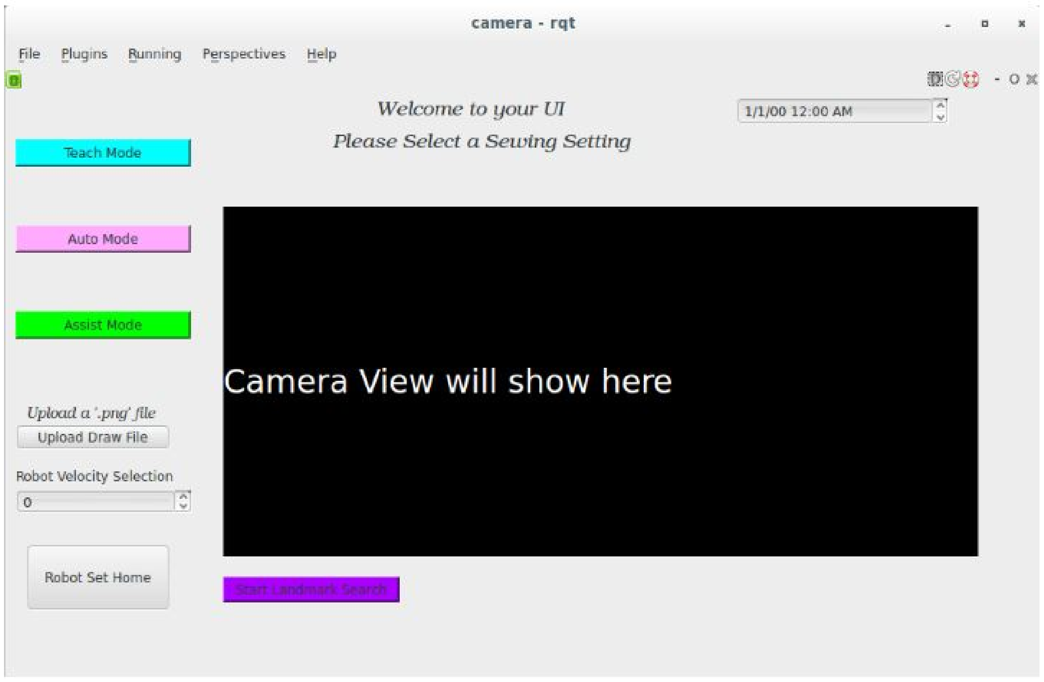


Figure 31: Final UI using RQT and C++

In the end, this UI worked as a standalone ROS plugin but not work as an executable subsystem. Limited experience in QT presented a steep learning curve to overcome before anything could be integrated with the rest of the system. Another problem was that the RQT tutorials and example UI were broken during the UI development of this project. These reasons are why another programming language was used to develop the final UI.

Python and the Tkinter library were chosen since ROS commands are easy to incorporate in Python and the members of the team have knowledge of the language. Further, it has no additional cost, unlike Matlab.

Python Tkinter UI Layout Design

After the programming language and available libraries changed, the layout design of the UI changed as well. Figure 32 shows the UI designed in Python.

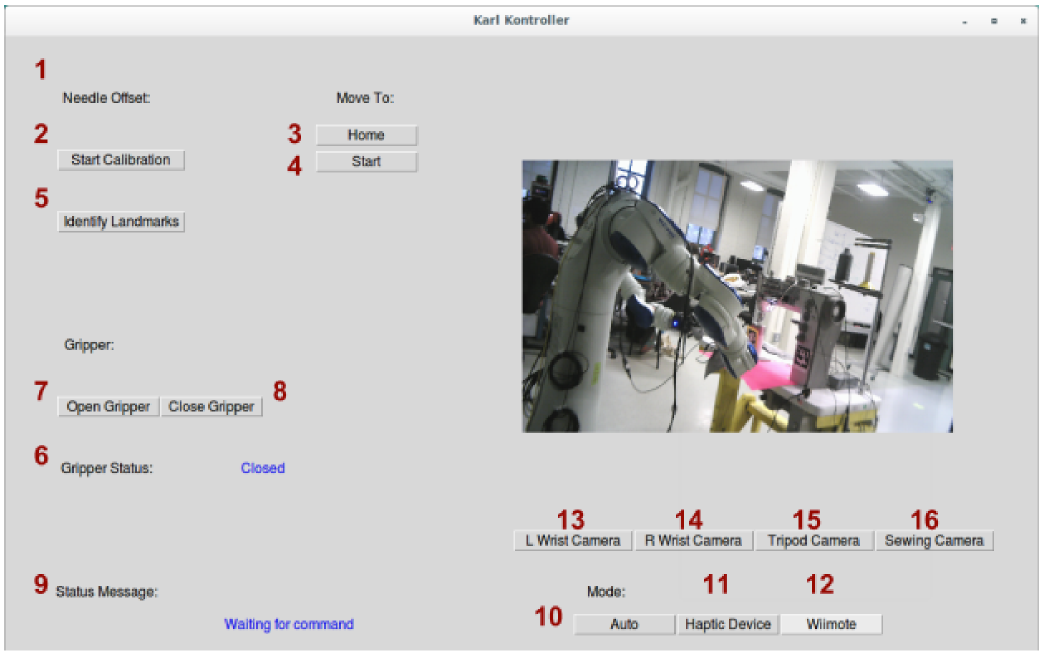


Figure 32: Final UI in Python Tkinter, Tripod Camera View

Table 5 describes the messages and buttons that correspond to the labels in Figure 32.

1: Needle Offset	Displays current offset between robot and sewing machine
2: Start Calibration	Boolean ROS message published, Calibration node performs offset calibration and publishes current offset
3: Home Button	Boolean ROS message published, Robot controller moves robot to home pose shown in Figure 28
4: Start Button	Boolean ROS message published, Robot controller moves robot to start pose shown in Figure 29
5: Identify Landmarks Button	Boolean ROS message published, CV subsystem begins routine to find the x, y locations of landmarks on vamp
6: Gripper Status	Displays current gripper position (Opened or Closed)
7: Open Gripper Button	Boolean ROS message published, right and left gripper controllers command gripper positions of 255
8: Close Gripper Button	Boolean ROS message published, right and left gripper controllers command gripper positions of 0
9: Status	Displays messages of current task being executed
10: Auto Button	Boolean ROS message published, motion planning subsystem plans path and robot controller actuates the path
11: Haptic Device Button	Haptic Device UI window is opened (Figure 38); user uses haptic device to record sewing pattern and publish a pose array of the pattern

Table 5 continued from previous page

12: Wii Remote Button	Input Device UI window is opened (Figure 37); user uses input device to record end effector positions along the sewing trajectory and publish a pose array of the trajectory
13: L Wrist Camera Button	Changes the camera view displayed to the view from the left wrist camera
14: R Wrist Camera Button	Changes the camera view displayed to the view from the right wrist camera
15: Tripod Camera Button	Changes the camera view displayed to the tripod camera
16: Sewing Camera Button	Changes the camera view displayed to the camera mounted on the sewing machine

Table 5: Features of User Interface

4.4.2 Input Devices

An input device is needed to enable the user to accurately control the robot's end effector position while in assist mode. The input device needs to be intuitive to use and record accurate and precise data. Three input devices were chosen and two were implemented. The input device investigated but not implemented was a Virtual Reality (VR) glove. The two implemented input devices were the Geomagic Touch Haptic Device and the Wii Remote. The input devices have separate user interfaces than the main UI. This section describes the input device interfaces, how to use the devices, and which of the devices is most effective.

Requirements

The data from the input device is used to determine a desired end effector position and orientation. Four input devices were compared on accuracy, cost, ease of use, and other parameters. Appendix H lists the pros and cons for each device researched. The evaluated devices are virtual reality gloves, haptic devices, fiducial markers or tags, and handheld controllers. After some research, the virtual reality glove chosen was the CaptoGlove. This glove is much cheaper than other VR gloves and has sensors accurate enough for this application. The CaptoGlove was also readily available in the lab, making this a convenient solution. The haptic device researched was the Geomagic Touch because it was already available in the lab. ArUco tags were chosen as the fiducial markers to analyze because they are cheap and accessible. The handheld controller chosen was the Wii Remote because of its popularity as a position tracking device. After this initial analysis, only the VR glove, Wii Remote, and haptic device were researched further. Fiducial markers were not a satisfactory input device because they take too long to set up and would not fit in a factory setting well. This section discusses the implementation of the remaining input devices.

Virtual Reality - CaptoGlove

Virtual Reality (VR) is a relatively new technology that allows a user to experience an environment that does not exist in physical form. Virtual reality gloves give the operator control in a way that simulates reality. For this sewing task, VR gloves give the user an intuitive way of controlling the robot with minimal training. The seamster can put on the gloves and sew any pattern. The position of the gloves could be recorded while the seamster is using one sewing machine, translated into robot end effector positions, and then mimicked by the robot on a separate sewing machine.

The CaptoGlove was chosen since it was readily available in the lab, had a lower cost than other VR gloves, and had sensors accurate enough for the sewing task. CaptoGlove provides SDKs in C++ and C, which implement features such as reading data from the sensors and managing the Bluetooth connection for the CaptoGlove. The plan was to use the CaptoGlove and the C++ SDK to record data from the sensors and use it to calculate the position and orientation of the user's hand. The biggest problem with the CaptoGlove was that it is an underdeveloped, new technology. It was tedious to connect to the CaptoGlove and the software examples did not work.

The libraries were only developed to run in Microsoft Visual Studio 2017 on Windows 10, and not on Linux with ROS. A ROS package would need to be developed to use the CaptoGlove as an input device for this project. This would be complex and time-consuming. Communication protocols like SSPP were researched to see if it was possible to use the CaptoGlove software on a Windows 10 machine and communicate the data to the Linux machine. This was done successfully for the haptic device, which is explained in Section 4.4.2. However, the CaptoGlove SDK was difficult to use and did not provide any filtering, resulting in noisy data. Therefore, this device was not implemented in this project. If a VR glove is desired in the future, a well developed ROS package and SDKs designed to work in Linux environments are suggested.

Handheld Controller - Wii Remote

The Wii Remote, shown in Figure 33, is a handheld controller used with the Wii gaming console.

Figure 33: Wii Remote

It uses three accelerometers for motion capturing and an IR camera for 2D position tracking. The Wii Remote has a Bluetooth transmitter, which can connect to any device with a Bluetooth receiver. The Motion Plus attachment, pictured in Figure 34, has a dual and single axis gyroscope used for 3D motion.

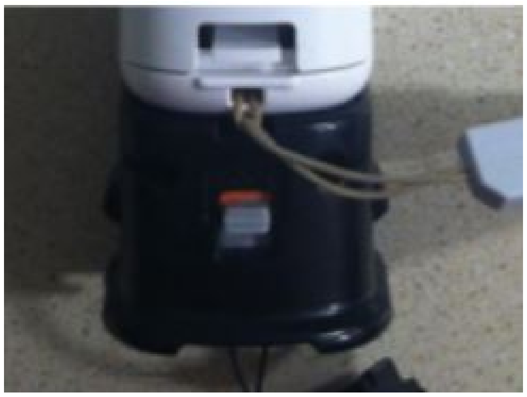


Figure 34: Motion Plus Attachment for Wii Remote

This attachment was used to gather motion more accurately than the Wii Remote by itself. This technology is well developed and has open source, easy-to-use libraries and ROS packages. The Wii Remote ROS package used is called "wiimote." It implements functions that can communicate with the Wii Remote through Bluetooth on the computer. Using this package, gyroscope and accelerometer readings were used to calculate orientations and positions.

The Wii Remote was the least accurate input device researched. According to a research paper from the University of Missouri, the Wii Remote's accuracy was within 1.275 degrees for its pitch measurement and 0.701 degrees for its roll measurement [77]. To increase the accuracy of the Wii Remote, the imu_filter_madgwick package was used to filter and fuse the gyroscope and accelerometer data. This greatly improved the accuracy of the orientation calculated using the Wii Remote. The testing methodology and the drifts measured are shown in Appendix I. The gyroscope and accelerometer only output angular velocity and acceleration, so to calculate position, the velocity and acceleration were integrated. This produced inaccurate positions and large drifts, as shown in Appendix I. In addition to the low accuracy, the Wii Remote is less intuitive because using the Wii Remote does not replicate the sewing task.

Haptic Device - Geomagic Touch

Haptic devices are used for tasks such as simulating touch interactions between robots and humans in a real and remote environment. This is a good input device because it is used frequently in research and industrial fields. Figure 35 shows the Geomagic Touch, the haptic device used in this project.

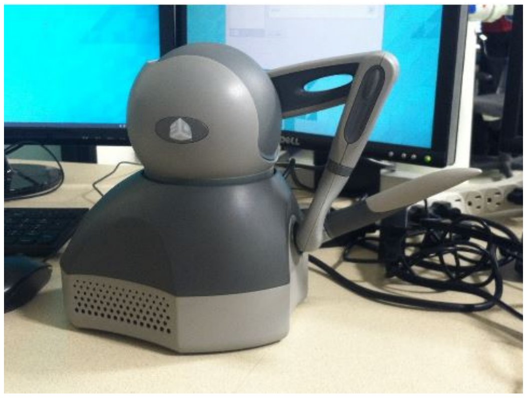


Figure 35: Geomagic Touch Haptic Device in Home Position

The haptic device allows a user to capture a series of poses that represent a series of waypoints along the desired sewing pattern. The user can use the tool point on the haptic device to specify a pattern that the robot should sew, seen in Figure 36.

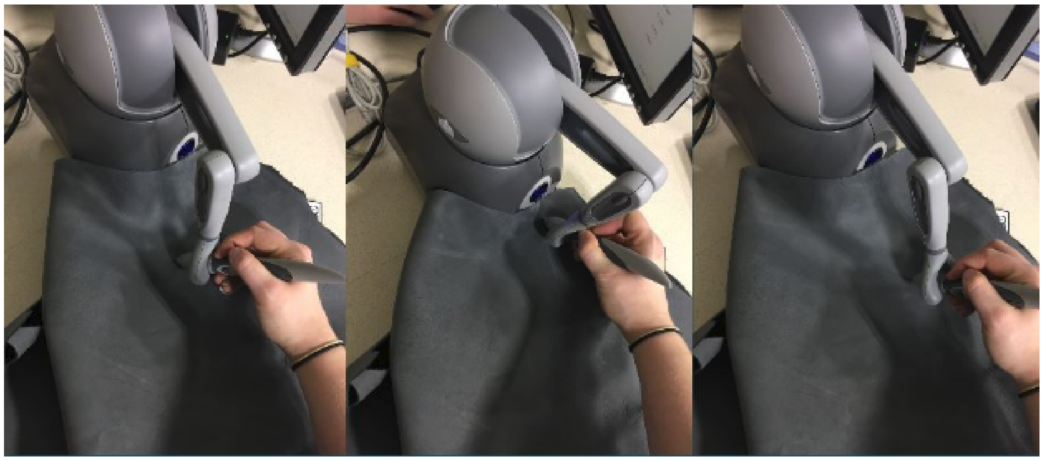
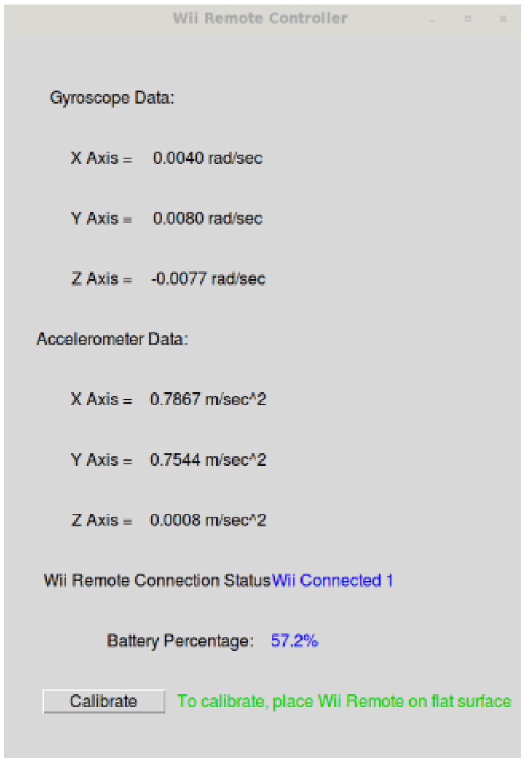


Figure 36: Sewing Pattern specified using the Haptic Device

The positions recorded using the haptic device were very precise and accurate (see Appendix I), unlike those calculated using the Wii Remote. Accuracy is important so that the robot's sewing does not deviate more than 1mm from the pattern, per the sewing precision requirement. Researching the haptic device was not a top priority initially. Additionally, while researching the haptic device, there was no ROS package available. To use the haptic device, it would have to be connected to a Windows machine and communicate to a custom ROS package using SSPP protocol. This was too challenging for the time allotted, but more resources for the Geomagic Touch device were available when the CaptoGlove was abandoned. At this time, the ros_geomagic ROS package had been developed by Adnan Munawar and was used for recording data from the device. The Ubuntu drivers provided by the OpenHaptic forums were used for connection and device calibration. Using the ROS package, the UI and Geomagic Touch nodes could communicate. The Geomagic Touch node collected pose messages from the haptic device and published them, which the UI then displayed. Then, the UI node published them so that the motion planning node could use them as waypoints along a desired sewing pattern.

Input Device UI

For the input device interface, the user can see the filtered data from the input device updated every 0.01 seconds. Figure 37 shows a picture of the input device interface when the Wii Remote is connected, and Figure 38 shows the input device interface with the Geomagic Touch haptic device is connected.


Wii Remote Controller

Gyroscope Data:

X Axis = 0.0040 rad/sec

Y Axis = 0.0080 rad/sec

Z Axis = -0.0077 rad/sec

Accelerometer Data:

X Axis = 0.7867 m/sec^2

Y Axis = 0.7544 m/sec^2

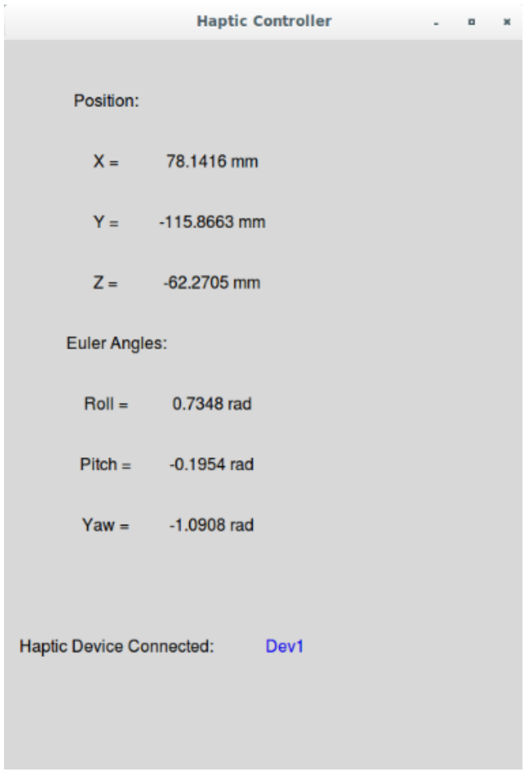
Z Axis = 0.0008 m/sec^2

Wii Remote Connection Status Wii Connected 1

Battery Percentage: 57.2%

Calibrate To calibrate, place Wii Remote on flat surface

Figure 37: Wii Remote Input Device UI



Position:

X = 78.1416 mm

Y = -115.8663 mm

Z = -62.2705 mm

Euler Angles:

Roll = 0.7348 rad

Pitch = -0.1954 rad

Yaw = -1.0908 rad

Haptic Device Connected: Dev1

Figure 38: Haptic Device UI

The calibration button is not present in Figure 38 since calibration is done in the Geomagic Touch diagnostic application outside of the ROS system. Buttons on the input devices are used to start and stop the recording. When using the Wii Remote, the starting point, (0, 0, 0) in space, is set by pressing the B button shown in Figure 39.

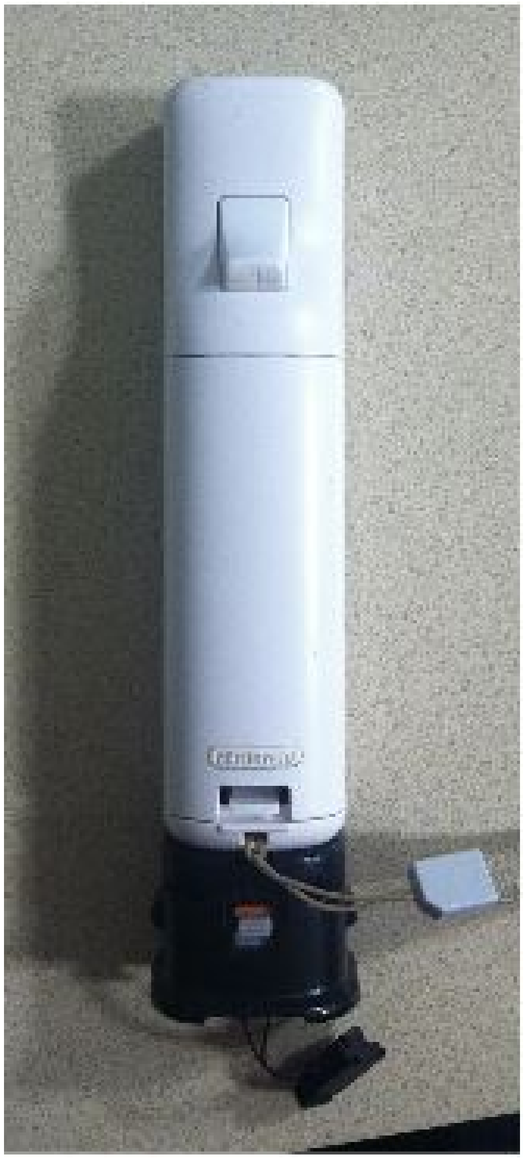


Figure 39: B Button on Wii Remote

Then, the B button must be held throughout recording. When the B button is released, the recording stops and a CSV of the raw IMU data from the remote is saved. That data is then used to calculate positions in mm.

To use the haptic device, the grey button on the tool is pressed to start recording and the white button is pressed to stop recording. The tool and buttons are shown in Figure 40. The position of the tool tip and the orientation of the tool's rotating axis are saved to a CSV file.

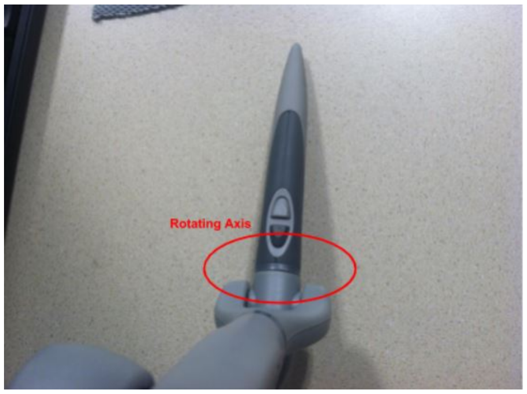


Figure 40: Haptic device tool, buttons, and rotating axis

The positions recorded by an input device are graphed on an x, y plot for the user to see what has been recorded. The plot of an example path recorded by an input device is pictured below in Figure 41.

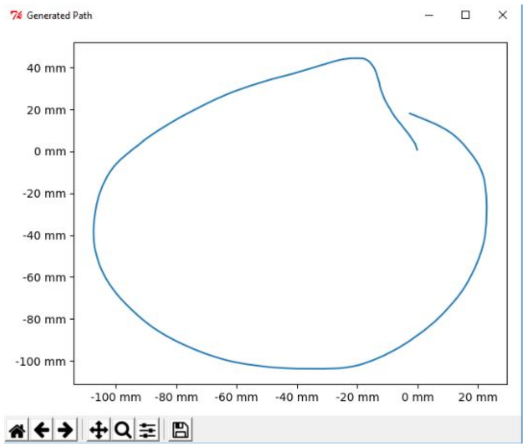
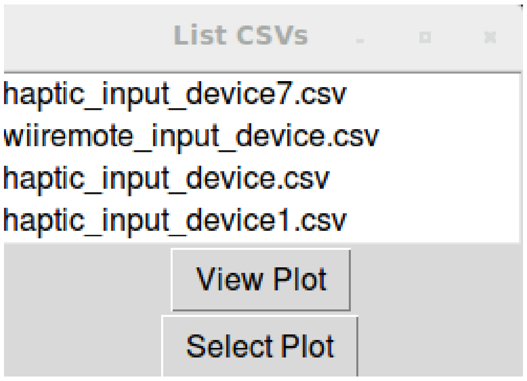


Figure 41: Example path of an input device

The paths can be seen by selecting a dataset from the list pictured in Figure 42 and selecting the 'View Plot' button. If the user approves of the path recorded by the input device, they click 'Select Plot', which publishes the positions in a pose array.



haptic_input_device7.csv
wiiremote_input_device.csv
haptic_input_device.csv
haptic_input_device1.csv

Figure 42: View and Select Plot

If the user does not like the path, they can either choose from the list of recorded datasets, or record a new path.

Input Device Recommendation

Out of the three input devices that were researched, the haptic device was the best. The VR glove was underdeveloped and the Wii Remote was not as precise or accurate. In conclusion, the haptic device is recommended as an input device because of its high accuracy. A similar haptic device is used at the factory for other industrial processes, so users would be familiar with how it works, minimizing the learning curve for this input device.

4.5 Computer Vision

Computer vision (CV) was implemented to aid the robot's sewing and locate the vamp in the workspace. Cameras were used to identify sewing patterns, as well as provide views of the workspace to an operator. This section explains what tasks needed to be solved with computer vision and how they were accomplished.

4.5.1 Camera Setup

Good quality, high resolution images were required for image processing and recognition. Different cameras were evaluated on resolution, frame rate, cost, size, interface ability, and codec usage as shown in Appendix C. Four cameras were used to provide multiple views of the workspace. One camera was placed on each wrist of the robot to provide a close-up view while sewing. Another camera was placed on the sewing machine in order to identify the sewing pattern before sewing began. The last camera was placed on a tripod so the user can put it where ever they want. A calibration of the cameras is performed once to correct the natural distorsions in the camera lens. The calibration works by taking images of a checkerboard pattern of alternating black and white squares, shown in Figure 43.

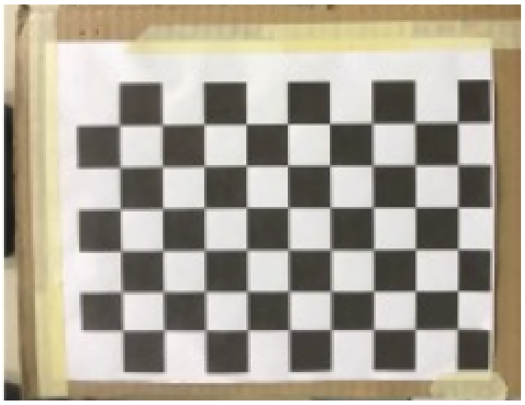


Figure 43: Camera calibration checkerboard

A preset number of images is captured while the user moves the checkerboard around the camera's view to cover all of the sensor. Then, the corners of each square are identified. The calibration program receives the dimensions of the squares and looks for differences in the size of the squares at different points shown in Figure 44.

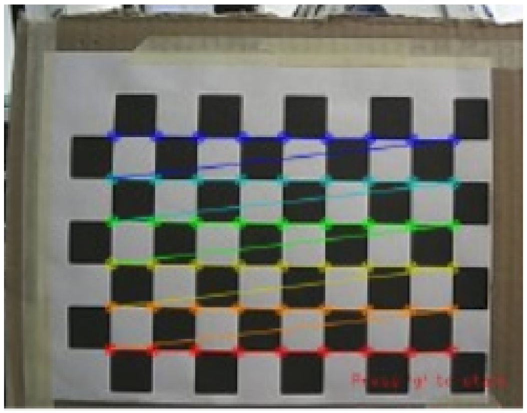


Figure 44: Camera calibration identifying image points

4.5.2 CV Software and Node Architecture

The software package OpenCV was used because it is efficient enough to run most calculations in real time and provides functions that can complete all of the desired tasks. It is written natively in C++, and has hooks for Python as well. Test nodes were written in both C++ and Python to compare the ease of use and effectiveness of each language. The nodes written in C++ struggled to compile, but even after the compilation issues were fixed, the C++ node produced numerous SEGFAULTS due to incompatibility with OpenCV, ROS, and necessary libraries. The Python node, however, did not produce errors when executed. Therefore, Python was chosen as the language to write the computer vision nodes.

Each camera stream is processed by a dedicated ROS node. OpenCV's VideoCapture object is used to open the live video stream in these nodes. Finally, the node publishes the video as a ROS Image message to be accessed by other nodes.

4.5.3 Tasks

4.5.3.1 User Views

The first task for the cameras was to provide the operator with views of the work being done because it is unsafe for people to be in the robot's workspace. However, the operator needs to make sure the robot is completing its task correctly. To accomplish this, an additional camera was mounted on a movable tripod to provide auxiliary views of anything the user needs. All the camera views are streamed from their respective camera node to the UI, where the operator can toggle between the different views.

4.5.3.2 Calculate Sewing Machine and Robot Offsets

In order to avoid collisions and sew accurately, computer vision was needed to calculate the distance and orientation offsets between the robot and the sewing machine. This is necessary for when the sewing machine and robot are reconfigured in the factory. The offsets from the robot to the sewing machine needed to be calculated quickly and with high repeatability. To do this, ArUco tags were placed on the sewing machine facing the robot as shown in Figure 45.

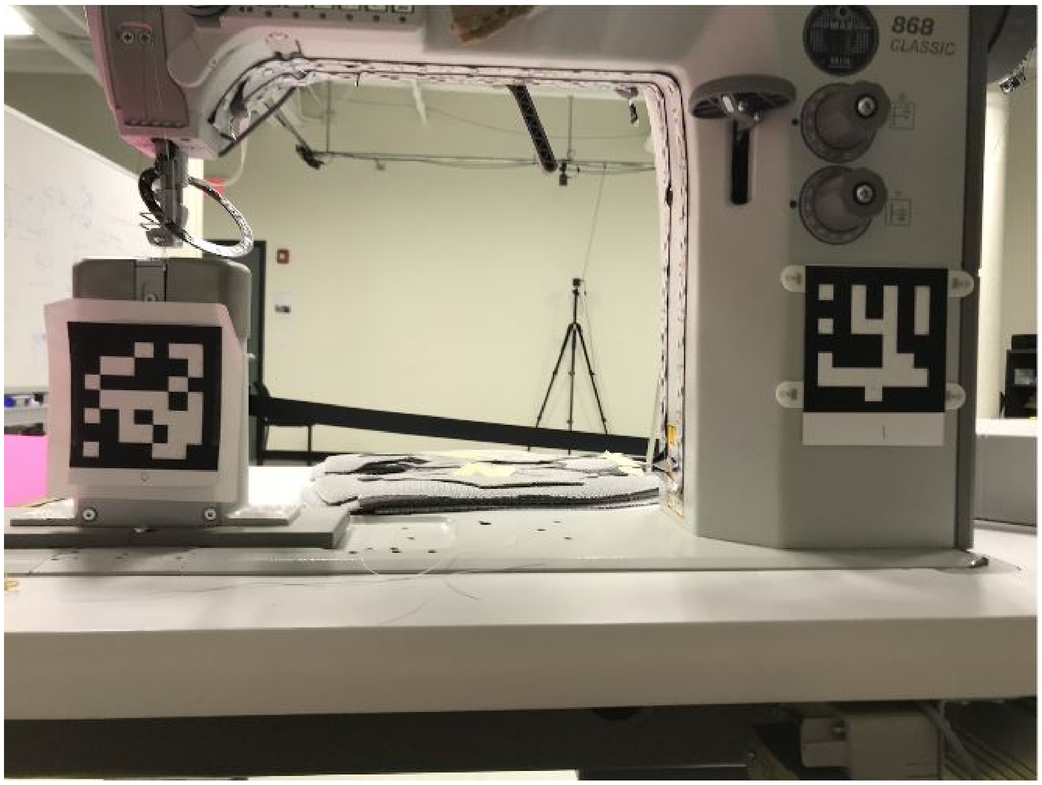


Figure 45: ArUco Tags on Sewing Machine

A camera was placed on each of the robot's wrists so that two ArUco tags can be used at once to provide a more accurate calculation versus a single tag. While multiple ArUco tags can be used with one camera, the addition of a second camera reduces the calculation error by providing another pose to compare against.

Each camera was mounted with a 3D printed clamp. The clamps were designed in Autodesk Inventor to fit the circumference of the gripper. They were 3D printed and mounted to the wrists as shown in Figure 46.

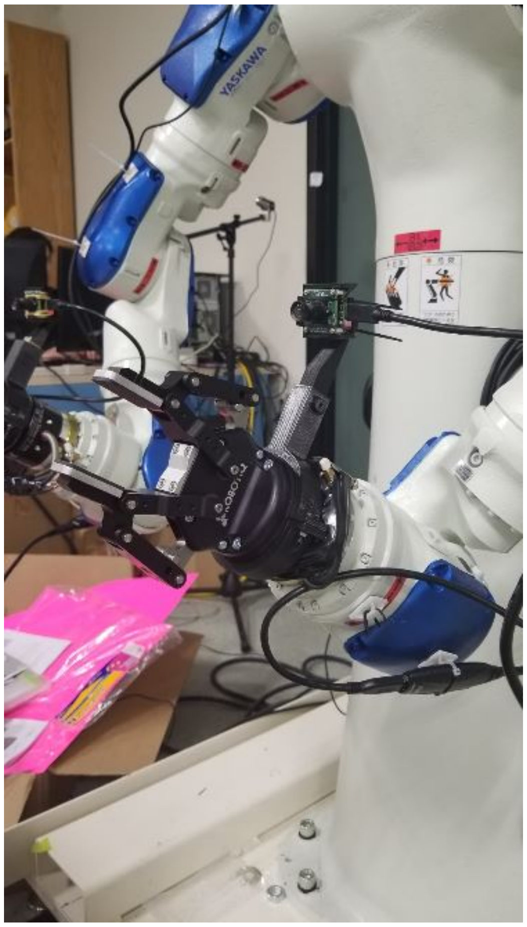


Figure 46: Wrist Cameras Mounted on the Robot's Grippers

The clamps were attached using two bolts on either side of the mount. Electrical tape was wrapped around the gripper at the location of the mount to provide more friction between the plastic of the mount and the metal of the gripper so the clamp does not slip while the robot moves. The mount, shown in Figure 47, also has an adjustable height and pitch angle (Figure 48) to quickly adjust the camera to provide the best view. The assembly and construction instructions are available in Appendix J.

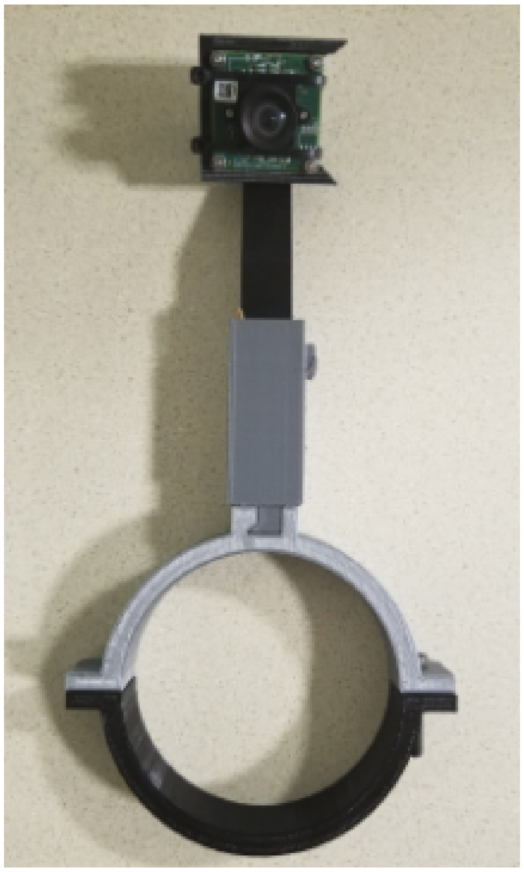


Figure 47: Wrist Gripper Mount

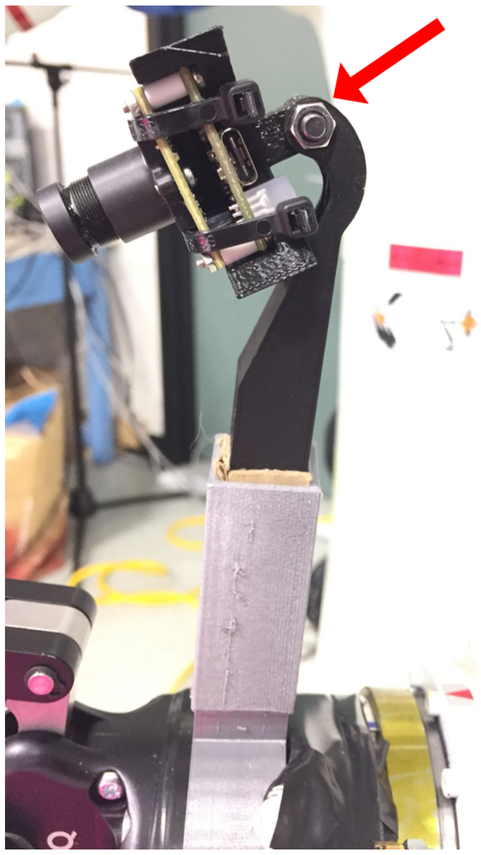


Figure 48: Wrist Gripper Mount featuring adjustable pitch angle

A calibration node was written to perform the sewing machine offset calculations. This node sits idle until a start message is published on the "startCalibration" topic. It then accesses the images from the two wrist cameras before performing the ArUco tag pose calculations. The overall pose is then published to the '/base_to_needle_offset' topic for the motion planning node to use.

For the offsets to be calculated, the robot arms need to be in the home position shown in Figure 28. This position is constant for every offset calculation, and known to the camera calibration node beforehand. Once the arms are in the correct pose, a start command is sent to the camera calibration node. The camera calibration node then retrieves the most recent images published by the LeftWrist and RightWrist camera nodes. Each image is then independently searched for valid ArUco tags using OpenCV's 'detectMarkers' function. This function returns a list of the corners in the markers measured in XY pixel coordinates. The location is then passed to another function, 'estimatePoseSingleMarkers', which also receives the camera's distorsions and the size of the ArUco tag, in millimeters. The pose of the ArUco tag with respect to the camera lens is returned. Because the location of the cameras with respect to the robot's frame is known, the marker's pose with respect to the robot frame can be calculated.

At this point, the node has the poses of each ArUco tag with respect to the robot frame. As the tags are static on the sewing machine, the offsets between each tag's pose and the sewing machine frame is known. Therefore, each tag can return its calculated sewing machine to the robot frame pose. These poses are then averaged together to provide a more accurate final pose.

4.5.3.3 Vamp Detection

To plan a successful path, the CV system needs to identify the location of the vamp and a series of sewing waypoints. To do this, a training image of the vamp is captured and processed manually, using GIMP to remove the background. This is done before the system is active, and has to be repeated if the shoe is changed. That image is then fed into the Image Processing node, which uses the OpenCV Oriented FAST and Rotated BRIEF (ORB) algorithm to calculate descriptors for key points in the training image. Those key points are decided by OpenCV, which looks for distinctive areas in the image. Figure 49 shows what the key points look like (key points are colored green).

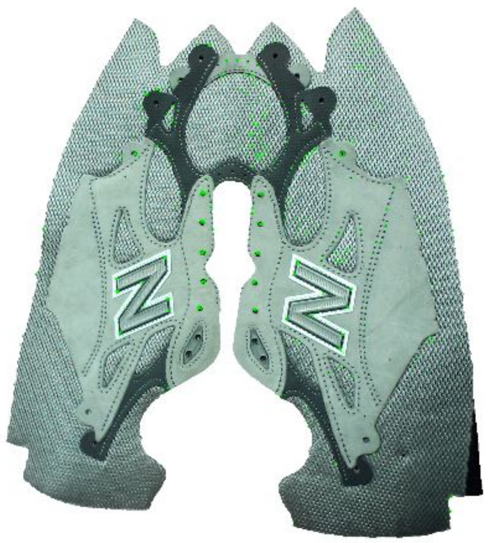


Figure 49: OpenCV Key Points

The training image processing is done when the node is launched. In the main loop of the program, ORB is used again on the camera feed to calculate descriptors for key points in that image. Descriptors are any identifying characteristics of the points, including shape, color, and orientation. During each iteration of the program, both sets of descriptors are compared to try to find any matches between the training image and the camera stream. To compare the descriptors, the OpenCV Brute Force Matcher (BFMatcher) is used to find the closest matching descriptor as the training image (Figure 50).

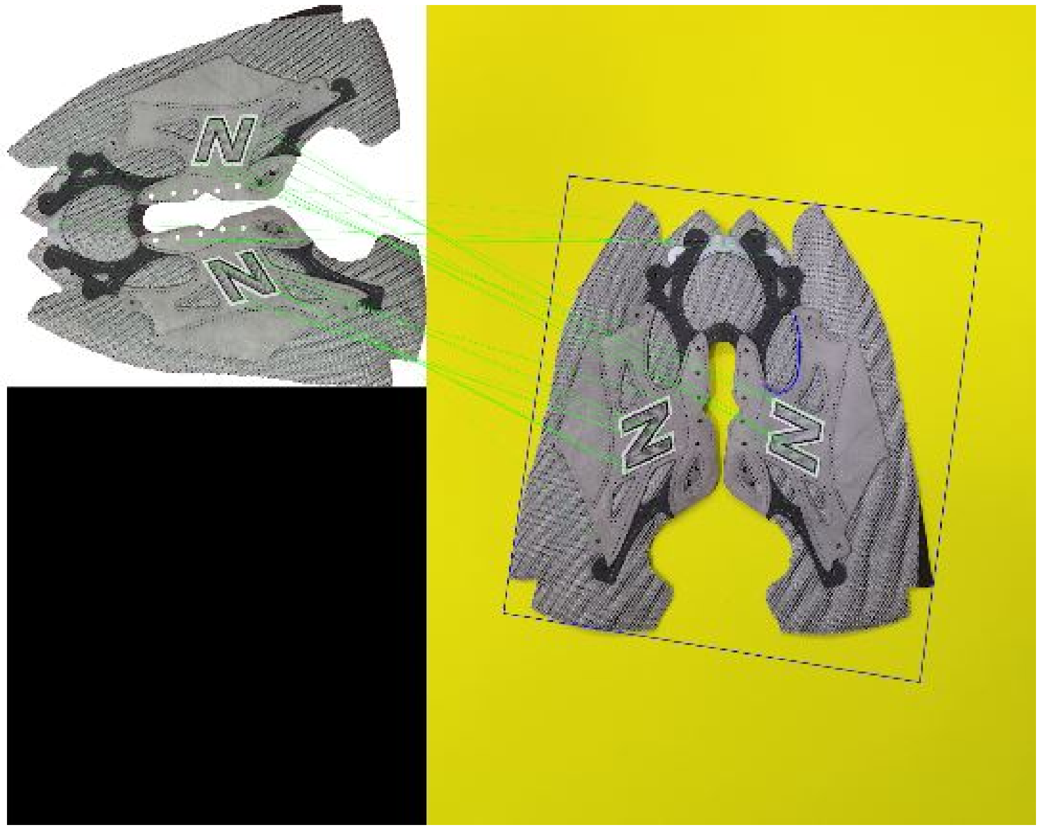


Figure 50: OpenCV Key Point Analysis

These matched key points are used in the OpenCV function findHomography to find the perspective transformation matrix between the two images. This transformation matrix is multiplied by a matrix of the known points from the training image to get the new positions of the desired points in the camera image. This path is then translated to the live image and a sewing pattern is determined (blue line in Figure 50).

Later in the project, a new step was added in the computer vision subsystem to identify the locations of the sewing waypoints again, once the vamp is in the robot's grippers. Previously, it was assumed that the user placed the vamp in exactly the same place in the robot's grippers every time. This was an unrealistic expectation, so the computer vision subsystem was used to find the relative distance between the grippers and the desired sewing pattern. This landmark identification step worked the same as the first vamp detection step but with different descriptors. More descriptors were added around the N on the vamp because the grippers limited the area on the fabric available for detection.

In addition to changing the descriptors used in the landmark identification step, some autocorrecting functionality was implemented to improve the accuracy of the landmark identification step. This was done by performing a sign analysis on the resulting transformation matrix that describes the transformation of the points from the training image to the live camera image. When the landmarks were correctly identified, certain numbers had a certain sign, while when the landmarks were incorrectly identified, those numbers had the opposite sign. This took some trial and error to figure out which numbers should have which signs, but, once implemented, increased the reliability of the CV subsystem.

Figure 51 shows the vamp detection step with the vamp in the robot's grippers.

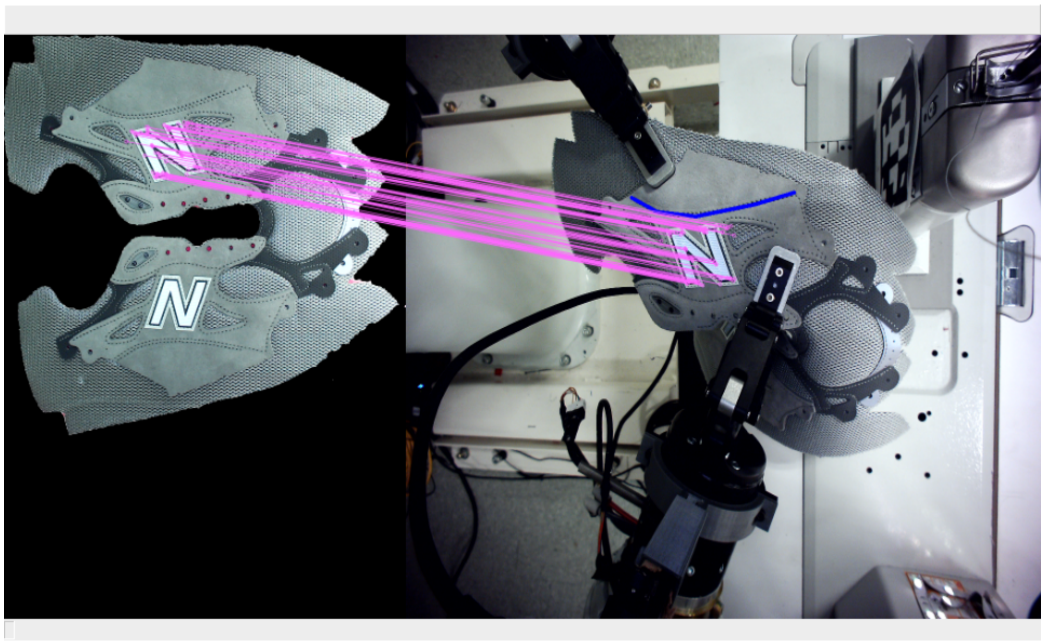


Figure 51: Key Point Analysis with Vamp in Grippers

4.6 Motion Planning

The motion planning subsystem needs to comprehend the output from the computer vision and user interface subsystems and create collision free joint trajectories. This chapter details how these tasks were accomplished.

4.6.1 Modeling the Workspace

The first step of the motion planning problem was to create a realistic simulation environment to test and evaluate different planning methods. Simulations save time because it's much faster and safer to run a simulation than a physical test. The simulation must include the robot, its grippers, the Force-Torque sensor, and the sewing machine. The model of the robot was obtained from the Motoman ROS package as a Unified Robot Description Format (URDF) file. This URDF was manually edited to include the grippers. The Robotiq gripper URDFs were obtained from the Robotiq ROS package. The base link of the gripper was attached to the last link of the robot arm. On the right arm, where the Force-Torque sensor was installed, the base of the sensor was added to the last link of the robot arm. Then, the gripper was added as a child link of the sensor. A XACRO file was used to simultaneously reference the robot, gripper, and sensor models.

To obtain a model of the sewing machine, a Kinect for Xbox One was used with the Kinect Fusion Explorer for Windows software. This software is used to create a solid object model (.stl file) from a 3D point cloud. To construct the model, one team member held the Kinect such that the entire sewing machine was in frame while another team member monitored the Kinect's output to ensure that the model was being properly constructed. The first team member moved with the Kinect around the entirety of the sewing machine to create a 3D scan. The sewing machine model produced is shown in Figure 52.

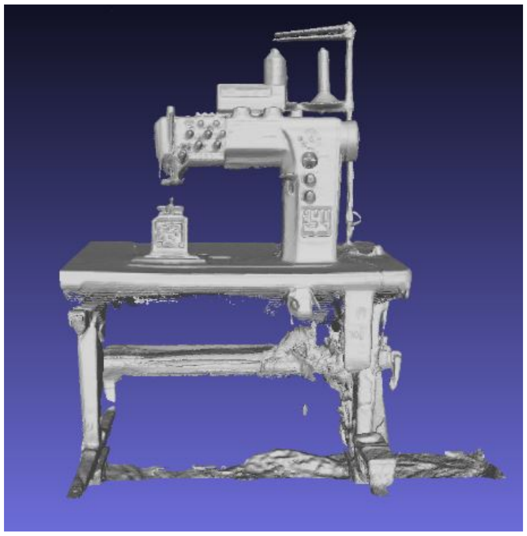


Figure 52: Sewing Machine Model from Kinect

4.6.2 Transformations

After modeling all the robot components, the transformation matrices between coordinate frames of the workspace need to be defined. Each link of the robot as well as the robot base and end effectors have independent coordinate frames. In addition, there is a coordinate frame at each camera and a coordinate frame where the sewing needle meets the fabric. The tf2 ROS package was used to publish the transformation matrix between coordinate frames as messages of type TransformStamped. Figure 53 shows the transformations and connections between the different coordinate frames in the workspace.

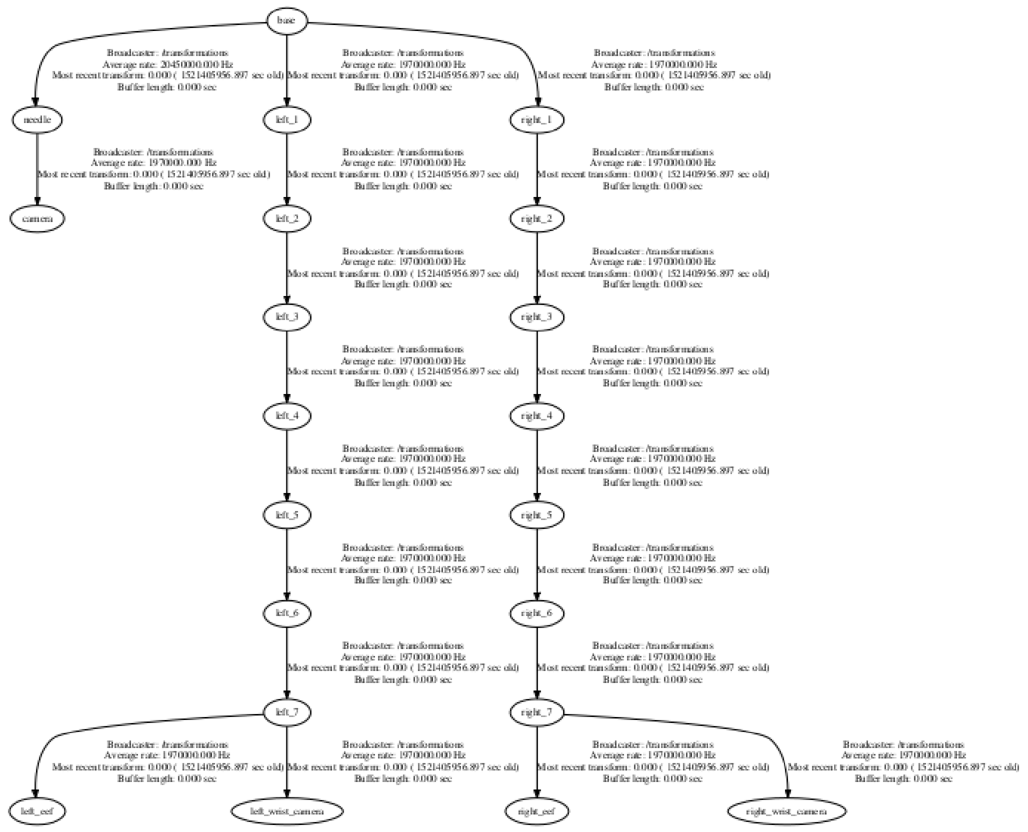
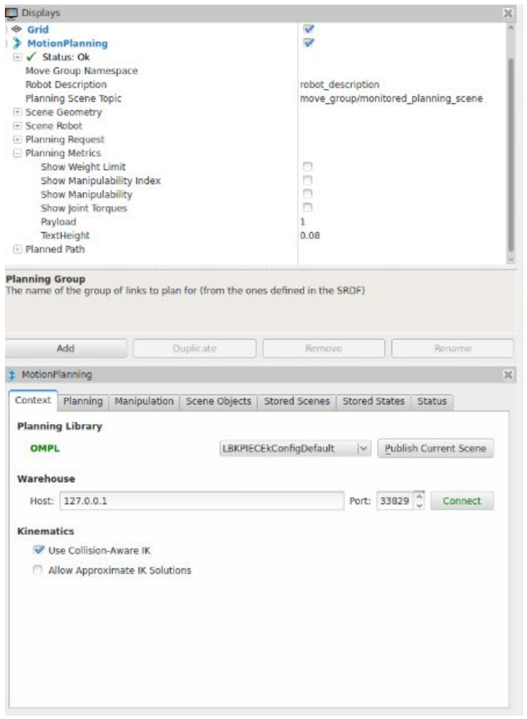

base
Broadcaster: /transformations
Average rate: 20450000.000 Hz
Most recent transform: 0.000 (1521405956.897 sec old)
Buffer length: 0.000 sec
Broadcaster: /transformations
Average rate: 19700000.000 Hz
Most recent transform: 0.000 (1521405956.897 sec old)
Buffer length: 0.000 sec
needle
left_1
right_1
camera
left_2
right_2
left_3
right_3
left_4
right_4
left_5
right_5
left_6
right_6
left_7
right_7
left_eef
left_wrist_camera
right_eef
right_wrist_camera

Figure 53: Transformation Tree

4.6.3 Simulating Motion: Moveit

To simulate the robot and gripper movement, the Moveit ROS package was used. Moveit is an open-source software package designed for testing and implementing different motion planning and manipulation solutions. Moveit has been used on over 65 robots, and the setup tutorial guide for the Yaskawa robot includes how to set up the robot in Moveit. This was the main reason Moveit was chosen as the initial motion planning software. It combines the collision avoidance capabilities of the Flexible Collision Library (FCL) with the variety of motion planners available in the Open Motion Planning Library (OMPL). Both of these work as standalone ROS packages, but Moveit integrates them along with simple testing and motion planning tools.

Moveit comes with a setup assistant that allows the user to define planning groups, end effectors, and the self-collision matrix of the robot. A planning group is a robotic chain that the user needs to define for inverse kinematic calculations. After this setup has been completed, Moveit generates several files, one of which is called ‘demo.launch’, which launches the robot model in RVIZ, a 3D visualization tool for ROS. This launch file also launches the Moveit side panel shown in Figure 54, which allows the user to select the desired motion planner as well as plan, execute, and visualize paths between start and goal joint configurations.



Displays

Grid	✓
MotionPlanning	✓
Status: Ok	
Move Group Namespace	
Robot Description	robot_description
Planning Scene Topic	move_group/monitored_planning_scene
Scene Geometry	
Scene Robot	
Planning Request	
Planning Metrics	
Show Weight Limit	☐
Show Manipulability Index	☐
Show Manipulability	☐
Show Joint Torques	☐
Payload	1
TextHeight	0.08
Planned Path	

Planning Group
The name of the group of links to plan for (from the ones defined in the SRDF)

Planning Library
OMPL — LBKPIECEkConfigDefault

Warehouse
Host: 127.0.0.1 Port: 33829

Kinematics
[✓] Use Collision-Aware IK
[ ] Allow Approximate IK Solutions

Figure 54: Moveit Side Panel in demo.launch File

Moveit also comes with a Move Group Interface, which is the user interface for the MoveGroup class, implemented in both C++ and Python. The MoveGroup class allows users to set joint and pose goals, create motion plans, add obstacles to the environment, and more. The C++ MoveGroup implementation was used for this project because it has more features and functions than the Python implementation. To add the sewing machine scan from Figure 52 to the simulation, the .stl was loaded as a collision object using the MoveGroup interface. The position of the sewing machine was set by measuring the distance from the robot to the sewing machine in real life and replicating the distance in the simulation. Making the sewing machine a collision object as opposed to a rigid object ensured that the robot and the sewing machine would not collide.

The next feature implemented using the MoveGroup interface was planning the path from the robot's home position to the start position. The start and home positions were found by manually jogging the robot into the positions using the teach pendant. Then, the joint locations were recorded by echoing the joint_states topic where all the robot's joint values are constantly updated. Then, the joint values were copied into the example C++ MoveGroup interface file as start and goal positions in joint space. Using the specified motion planner, MoveGroup creates a path between the start and goal positions. This path is published to the ROS topic optimized_trajectory with the message type trajectory_msgs/JointTrajectory. The robot controller node subscribes to this topic and reads the path message. The robot controller node sends the message to the robot controller to execute the preplanned path.

After the MoveGroup interface was successfully used to plan between joint space goals, plans to Cartesian pose goals were attempted. The Cartesian pose has four components: x-position, y-position, z-position, and orientation as a quaternion. A Pose object in the geometry_msgs ROS package was constructed from these four components. The Pose for this test of planning with Cartesian poses is the desired end effector location for the robot's 'start' position. The MoveGroup command 'setPoseTarget' was used to set this Pose as the target for the specified planning group. Each of the arms is a separate planning group, so for this initial testing of the Cartesian pose goals, the planning group was the seven joints of the right arm. Next, the 'plan' function of the MoveGroup class was used to create a path for the given planning group to reach the set Pose using the specified motion planner. However, the planner could not find any valid paths. According to the Moveit Wiki, this is for one of the following reasons: pose is unreachable by the robot, not enough time or planning iterations were given to the planner, the allowable angle error was too small, or the defined end effector of the robot was not actually the end effector. Each of these potential failure methods were evaluated.

The pose is known to be reachable because the robot was manually jogged into this joint position in simulation with no collisions or singularities. Also, planning to the pose was possible when done in joint space instead of Cartesian space, eliminating this as the problem. Several other goal poses were also tested and the planner behaved the same.

To eliminate planning time as the limitation, the planning time was increased from the default of 5.0 seconds to 60.0 seconds, and a plan was still not found. The time was not increased further because a requirement of the system is that the sewing task is completed in 60 seconds. The number of planning iterations were also increased from 1 to 100, and no plan was found still. The angle tolerance was increased from the default of 1.0x10-5 radians to 0.1 radians. Even with this very large tolerance, which would not be feasible for this project, a plan was not found.

The final potential error mentioned by the Moveit Wiki was a problem with defining the end effectors of the robot. The end effectors and their parents links were defined using the Moveit Setup Assistant. The Moveit Wiki included a tutorial on how to use the setup assistant. This tutorial was followed closely to ensure that the end effector was defined properly. The MoveGroup interface was also implemented on PR2, the robot used in the Moveit tutorials, to ensure that there was not an issue with the installation of Moveit. The interface was able to find a plan using the PR2 robot. Based on this and the fact that none of the other potential fixes impacted the plan produced, we determined that the hyper-redundancy of the 7-DOF Yaskawa robot was too complex for the Inverse Kinematic (IK) solver built into Moveit.

Moveit implements the Kinematic and Dynamics Library (KDL) for IK solving. Moveit is also able to implement the trac_ik ROS package for IK solving, which differs from KDL in a variety of ways. First, track_ik accepts a maximum planning time as a parameter and restarts the solving process if it fails, providing there is enough time left. KDL does not restart planning if it fails, unless told to do so by receiving a number of planning iterations as an optional parameter. KDL uses a joint-limited pseudoinverse Jacobian implementation, which researchers at TRACLabs found to produce errors frequently due to the joint limitations of robotic arms [78]. Trac_ik however, runs two different IK methods simultaneously. This first is similar to the KDL solver but mitigates local minima. The second is a Sequential Quadratic

Programming IK formula using quasi-Newton methods. The combination of these methods results in better performance for non-smooth search spaces and hyper-redundant robotic arms.

After failing to find planning solutions using KDL, we implemented the TRAC_IK Moveit plugin. Unfortunately, this plugin did not produce a plan either. In addition, the joint vectors generated by path planning in joint space do not have associated velocities. This means the velocities of the joints would be chosen by the controller, so the speed of the sewing machine would need to adapt to match the velocity of the robots' joints. The only way to change the speed of the sewing machine is to manually actuate the foot pedal beneath the sewing machine. The robot cannot reach the pedal and it would be unsafe for a human to operate the pedal, so controlling the robot's joint velocities is critical in a successful sewing task. We attempted to mitigate these problems by trying a different 3D visualization and motion planning software, Klamp't.

4.6.4 Simulation Motion: Klamp't

Klamp't is a 3D modeling and simulation software package developed for robot manipulation and locomotion. Its fast trajectory optimization, real-time motion planning routines and IK constraint solving promise a robust and versatile tool for robotic manipulation tasks.

To load a file in Klamp't, it must be a URDF file, not a XACRO file like in Moveit. A Python script provided by ROS was used to convert the

XACRO file to a URDF file. Klamp't comes with many different example programs to test and analyze robot models and robot motions. First, the Klamp't tool called 'RobotPose' was used to open the robot URDF to ensure that the robot appears as expected and the joints actuate as expected and respect their joint limits. All of the arm joints actuated properly, as well as the torso, but the Robotiq grippers referenced in the URDF did not appear in the Klamp't visualization because the grippers and the robot files are in different ROS packages, and Klamp't cannot load from multiple packages at once. This problem was ignored for the time being because there were other tasks of higher priority.

To put the sewing machine model into the Klamp't simulation, the .stl file was converted to an .obj file. The .obj file was loaded as a rigid object in the XML file that also loaded the robot model and a floor plane as shown in Figure 55.

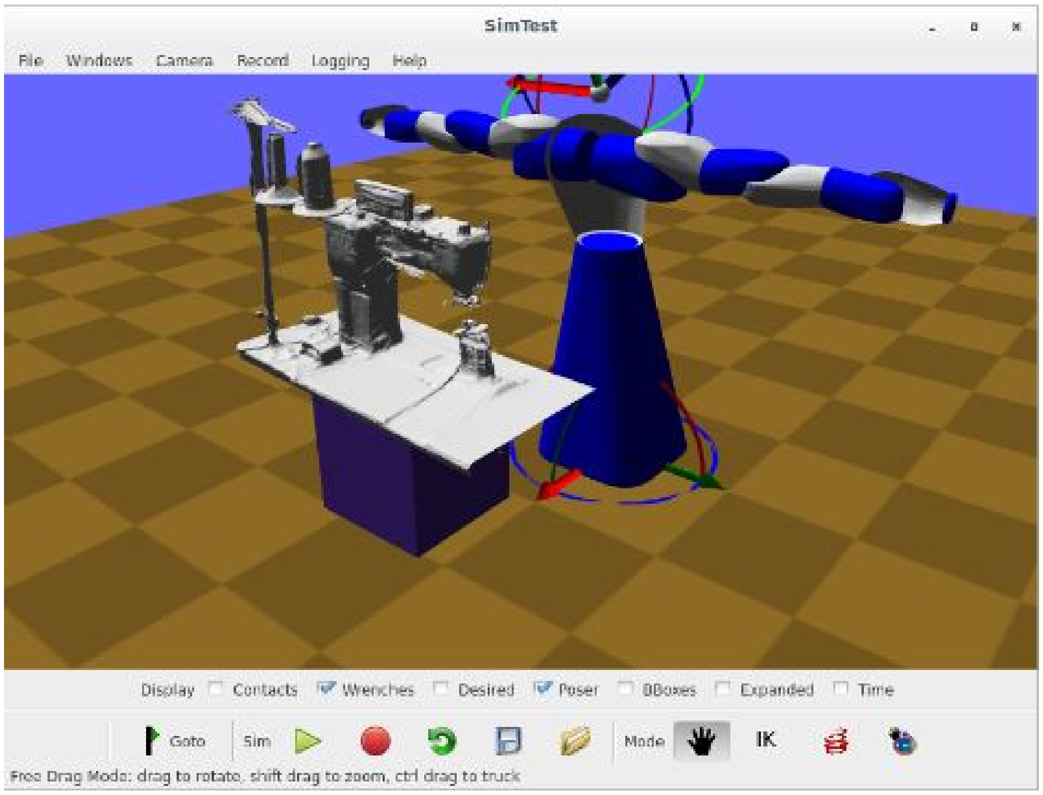

SimTest
File Windows Camera Record Logging Help
Display Contacts Wrenches Desired Poser BBoxes Expanded Time
Goto Sim Mode IK
Free Drag Mode: drag to rotate, shift drag to zoom, ctrl drag to truck

Figure 55: Klamp't Simulation World

A translation and rotation were applied to the sewing machine so it matched the location in the real world. The translation was found by measuring the distance from the robot to the sewing machine in the real workspace and replicating it in simulation. The appropriate rotation, however, was found by trial and error until the sewing machine table was parallel to the floor.

After the workspace was properly modeled in Klamp't, the masses and inertial matrices of each link were added to the URDF. These values were not included in the URDF because RVIZ loads them from a .yaml configuration file instead of the robot file itself. The inertial matrices were found by performing a geometric analysis of each link individually in Meshlab. The masses were found by performing a uniform mass distribution analysis of the robot in Solidworks. The calculated masses and inertial matrices are shown in Appendix K. After these parameters were found, the RobotPose Klamp't tool was used to create a test path. The purpose of this test path was to use it to calibrate the simulated motors. Klamp't comes with a MotorCalibrate tool that compares a robot's planned trajectory and executed trajectory and sets the dry friction, viscous friction, and PID values for each motor. Calibrating the motors ensures that they act as similarly to the real motors as possible.

Creating the test path for the motor calibration involves moving each joint throughout its full range of motion. The path was constructed by manually jogging the simulated robot's joints to their lower limits and saving the robot's pose. Then, the robot's joints were manually jogged to their upper limits and the robot's pose was saved. The path optimizer built into Klamp't was used to create a trajectory between these two poses. This optimizer generates and optimizes a trajectory between the first and second poses to minimize execution time. This outputs a .xml file with a series of joint angles and associated velocities.

After the test path was constructed, the robot and the test path were loaded into the SimTest Klamp't tool. SimTest takes the test path and outputs the executed trajectory necessary for the motor calibration. However, when the robot and test path were loaded into SimTest, the executed trajectory path was not produced.

For assistance with this error, the team emailed Dr. Kris Hauser and Hayden Bader, two developers of the Klamp't software. Unfortunately, they did not know why this error was occurring and suggested that the motor calibration parameters be hand tuned as opposed to using the built-in tool. Hand tuning these values is infeasible because there are 75 values total and there is no way to determine if the calibration is done properly. Therefore, there was no way to create a simulation environment that is accurate to the real world. For this reason, Klamp't was abandoned as a simulation and modeling tool. The team moved forward without a simulation tool temporarily and focused on IK solving and path optimization techniques.

4.6.5 IK Solving

After not being able to successfully solve the Inverse Kinematics using Moveit or Klamp't, the ROS package trac_ik was used to solve the inverse kinematics. An example from the trac_ik repository was modified to work as an IK node in the motion planning subsystem of this project. This node subscribes to the topic where the Pose Arrays of goal end effector poses are published. The IK node uses the TRAC_IK algorithm described in Section 4.6.3 to solve for the desired joint angles, and publishes the joint angles as a JointTrajectory. As described in Appendix I, the average deviation for each angle was 3.4698e-05 radians, or 0.001988 degrees, which is a negligible distance. This confirms that the TRAC_IK algorithm is very accurate. Only six joints were used to calculate the IK to simplify the planning problem, since the seventh degree of freedom is redundant and there are often many solutions for a 7-DOF IK problem. This seventh degree of freedom is still available to the IK solver in case the robot needs to avoid a collision or singularity. After the inverse kinematics calculations are complete, the desired joint angles are passed to the motion planner node to create paths between them.

4.6.6 Motion Planner Analysis

The motion planning ROS package used in this project is the Open Motion Planning Package (OMPL). This package can be used on its own or within Moveit or Klamp't. The package contains over 30 planning algorithms for various applications. They come in two categories: geometric planners and control-based planners. Because of the many degrees of freedom in the robot, geometric planners were the focus of the analysis for this project. The desired planner solves motion planning problems quickly and is able to find a solution using six or seven degrees of freedom. LBKPIECE is the default planner in OMPL because it has been shown to work in a wide variety of problems. For problems where a projection of state space is not available, the default planner is RRT.

The motion planner analysis of this project explored several different geometric planners to determine the optimal ones for this application. They were evaluated criteria including planning time, repeatability, and distance to obstacles. The full list of criteria is available in Appendix L. The results of this analysis showed that the default LBKPIECE algorithm performed the best of the planners tested. This is the algorithm used to plan the sewing motions.

4.6.7 EEF Position Determination

The motion planning subsystem needs to plan a path based on the locations of the waypoints defined by either the computer vision subsystem or an input device. If using the input devices, the UI node publishes the recorded path from the user as a PoseArray to the desired_trajectory topic. The motion planning node subscribes to this topic and optimizes the path created by the user. This is done by passing the poses to the motion planner algorithm to optimize the path between all the poses. These paths are constructed into a JointTrajectory message and published to the topic the robot controller subscribes to.

The motion planning subsystem must determine where the desired end effector positions are based on the given sewing pattern. Finding the desired end effector positions was performed in two ways. In the first method, the Vicon motion capture system was used to determine the natural placement of hands while sewing. Markers were placed on the hands as shown in Figure 56.

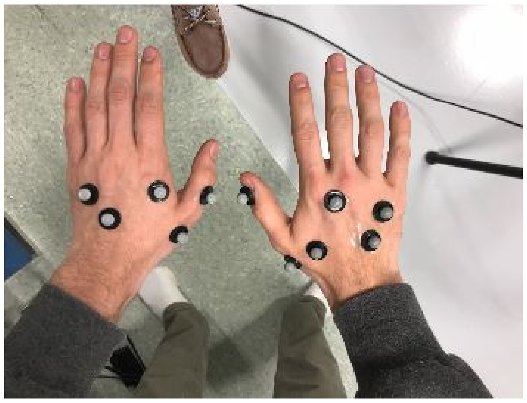


Figure 56: Placement of Motion Capture Markers on Hands

Two of the markers were placed along the thumb, and the rest were placed on the top part of the hand. Other markers were placed on the side of the hand because the others were placed in the same plane, making it difficult to determine the orientation of the hand. One extra marker was placed on the right hand so the markers were not identical and the motion capture system could distinguish between the hands. The origin of the workspace was defined as a point directly above the sewing needle, so the data could be easily translated into a reference frame that was already defined. The origin, defined by the five red LEDs of the wand, and the motion capture workspace are shown in Figure 57 below.

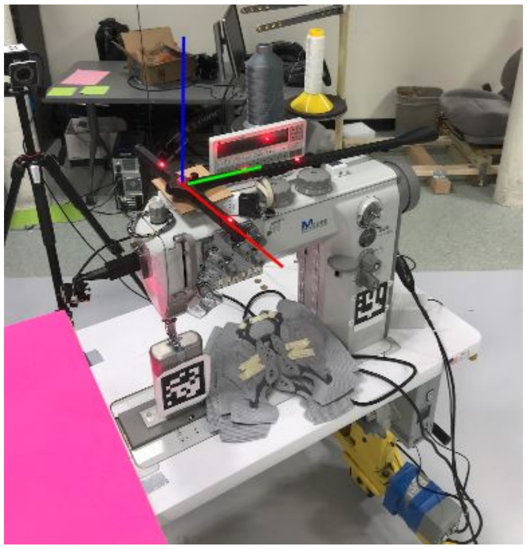


Figure 57: Motion Capture Workspace with Origin Shown

The motion study was conducted as follows. After the cameras, workspace, and origin were calibrated, the markers on the subject's right hand were defined as one object and the markers on the left hand were defined as another object. Three different subjects were used in this study and all of their hands were defined as different objects because their hands were different sizes and the markers were not placed in identical locations. After the subject's hands were being tracked by the cameras, a recording of the positions was started and the subject sewed a path back and forth four times. Every time the subject reached an end point, they notified the recorder at the computer who saved the hand positions and started a new data collection before the subject sewed back the other direction. This was done to save time so the subject did not have to reset their hands each time they began to sew. The subjects tried to sew one pass in about three seconds so the data could be compared between subjects.

All of the subjects sewed one straight path and one curved path. The curved path, however, produced sporadic data because the hands often got blocked by the sewing machine and the camera lost view of one or both hands. Therefore, the data processed was only for the straight paths. The distances of the hands from the origin in the x, y, and z directions, as well as the roll, pitch, and yaw about the origin, were recorded for each sewing pass for both hands. This data is recorded every 0.001 seconds and output to a CSV file.

The data was filtered to eliminate outliers. However, data points that appear to be outliers cannot simply be eliminated. This is because, occasionally, the motion capture system confuses which hand is which. This is because even though the marker placement is different on each hand, sometimes a marker on the side of the hand can get blocked by the sewing machine. To ensure that useful data was not deleted, scatter plots were made of the right and left hands. A combined scatter plot of the final data for both left and right hands is shown in Figure 58.

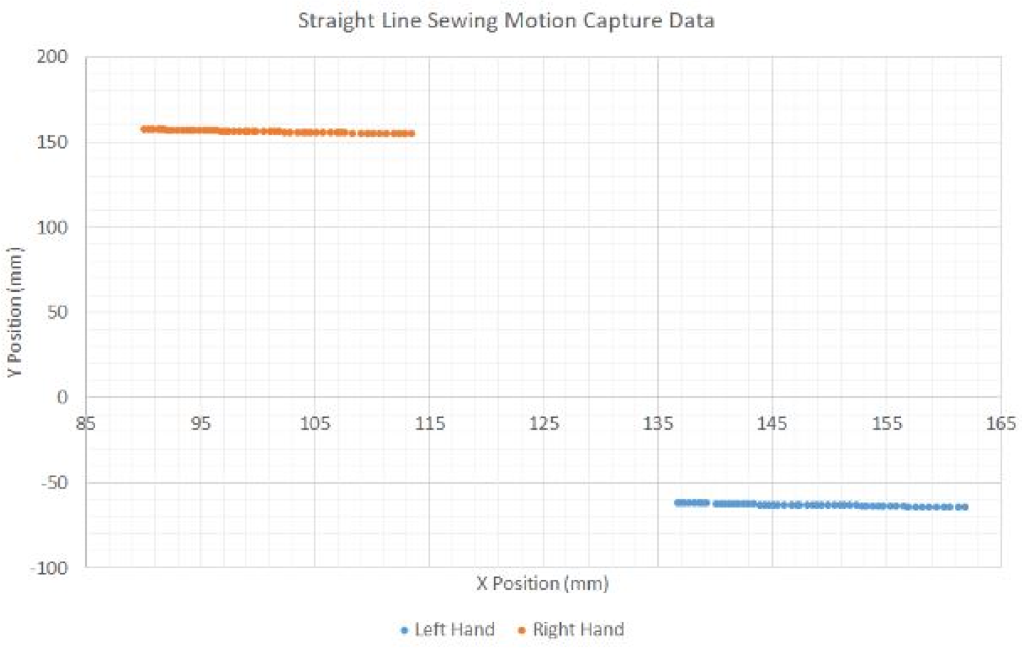


Figure 58: Scatter Plots of X,Y positions of the Left and Right Hands

If it appeared that both datasets had a misplaced set of points, the misplaced data was switched to match the appropriate hand. Additional filtering of outliers was completed after this to eliminate spurious points that can happen from changes in the motion capture workspace, a sudden change in relative position of the markers, or other unpredictable circumstances.

To further eliminate erroneous data, all of the data for each time step was averaged. The standard deviation was calculated and all data points that were greater than three times the standard deviation plus the average or less than the average minus three times the standard deviation were removed. The averages were updated after removing these points. The final averages per time step were kept as the final data set to replicate on the robot.

This data needed to be correlated with the waypoint positions calculated by the computer vision system. To do this, the vamp was placed in the robot's grippers similarly to how the human subject's hand held it during the sewing motion capture study, as shown in Figure 59. Note, all sewing testing was performed on the left half of the vamp.

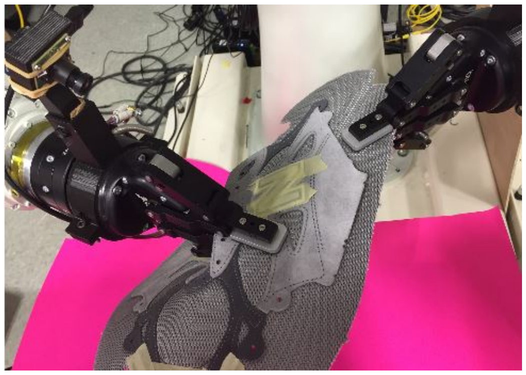


Figure 59: Vamp held in Robot Grippers

A video of the vamp in this position was taken over the pink background. The CV landmark identification software identified the x, y locations of the points on the vamp with respect to the camera coordinate frame that form the sewing pattern as shown in Figure 50.

To know where to start sewing, the waypoints were transformed into the sewing needle's coordinate frame. The first waypoint should be placed at the origin of the sewing needle's frame, where the sewing needle meets fabric. This means the starting joint configuration and the goal end effector position were both known. The goal end effector position was passed to the IK solver and the motion planner created a path between the start and goal poses. This path was used to move the vamp from under the camera to under the sewing needle.

The rest of the motion capture data was stored in the motion planning node in order to correlate the desired end effector positions with the CV waypoints throughout the sewing process. To do this, a C++ object containing the CV waypoint X position, CV waypoint Y position, and corresponding end effector pose was constructed by reading the data from text files. The data from each time step was used to create a new object. These objects were added to a vector and sorted by CV waypoint X position. A function was written to find the corresponding end effector pose given a CV waypoint X position. The Y positions are currently ignored because only straight line paths were recorded and there was very little movement in the Y direction. If there is no available data for the desired CV waypoint X position, the end effector poses corresponding to the two closest positions are used to interpolate a new predicted pose.

The second method of determining the end effector positions without using motion capture data was implemented because only one sewing pattern can be sewn using the motion capture method. Also, the motion capture method assumes the human worker places the vamp in the robot's grippers exactly the same every time. This is unrealistic, so it was necessary to develop a more dynamic way to determine the end effector positions. The first step in the second method is to determine the location of the grippers on the vamp. This was done by using the computer vision system to publish the coordinates of the waypoints along the sewing pattern in the camera's reference frame while the vamp is in the grippers. These coordinates were transformed into the robot's base frame and converted from number of pixels to meters. Next, the distance between the two grippers and the distance between the grippers and the sewing pattern were determined. This step is important because these distances must be kept constant throughout the entire sewing process, or the fabric will bunch or warp.

After determining the relative positions of the grippers and sewing pattern, the first point of the sewing pattern is placed under the sewing needle. This is performed by determining the distance between the sewing needle and the sewing pattern. The distance is found by transforming the origin of the sewing needle coordinate frame into the robot base frame. Because the sewing pattern waypoints are already transformed into the robot base frame, the position of the first sewing pattern waypoint and the position of the sewing needle can simply be subtracted. This distance is applied as a translation to the right and left end effectors.

Once the first sewing point is under the sewing needle, a trajectory to sew the rest of the path must be created. To do this, a cubic spline is fit to the sewing waypoints. A cubic spline was chosen because it fits the majority of sewing patterns well. Then, 1000 points along the spline are chosen to be intermediate sewing points. A large number of points were chosen because if the path is curved, too few points would not create a smooth sewing pattern. The distance between the sewing point currently under the needle and the next sewing point is determined and applied as a translation to the end effectors. This is repeated until all the sewing points have been reached.

5 Evaluation

It is important to reflect on the success of the end result of the project, including how the team functioned and planned tasks. This chapter evaluates how the product performed as well as how the team executed the project over the course of the year.

5.1 Social Implications

Regardless of the technical successes of this product, it is important to evaluate how the project could impact New Balance factory workers. If the system achieves the goal of being able to sew autonomously, the project has the possibility of displacing a small amount of workers. Since the system is made for a specific shoe, size, width, and color, there are few opportunities to permanently decrease factory staff. As the system expands to sew different shoes, the potential to eliminate jobs increases. The shared autonomy feature, and external factors could make immediately downsizing an irresponsible option for New Balance.

The use of shared autonomy in the system reduces the need to decrease the number of employees, and instead changes their job to one that is less dangerous and dirty. Sewing with a shared autonomous system decreases the risk of injuries for seamsters. By educating seamsters on how to sew with the robotic system and improve the sewing trajectories, the quality of the shoes produced would increase, while seamsters would not lose their jobs or personal value at the company. It is possible that the autonomous capabilities of the system could discourage or demotivate workers currently employed in the factory. Seeing complex tasks, like sewing paths in 3D space, become automated could decrease the workers' drive to perform their tasks and discourage potential applicants from joining New Balance.

To mitigate some of these risks, training programs could be offered to New Balance employees at risk of being replaced by a robot. The programs could train seamsters to perform other valuable tasks, such as robotic maintenance. This way, as more robots are added to factories, the workers would have more job opportunities with robotic upkeep and repairs. Jobs would also be created in designing and programming the robot systems, although these jobs would be in research and development instead of the factory environment.

The shoe seamsters could also be reassigned to a different department, such as sewing other textiles in a production line that has yet to be automated. While this solution is not permanent, it takes advantage of the seamsters' abilities while managers decide whether they want to repurpose the seamsters' existing skills, develop new skills, or downsize factory staff. While reducing the number of workers in the factory might seem like the natural reaction to an influx in robotic equipment, it is not the only solution or the ideal solution.

5.2 Project Execution Evaluation

It is important to evaluate how the project was executed. This section analyzes the project's overall execution, timing, and planning from the team's perspective.

5.2.1 Execution Summary

The overall execution of this project was much slower than expected. Several factors contributed to this, including slow delivery times and difficulty learning some topics in this project. The first step was to set up the robot workspace. It is important that the setup be done properly for future researchers as well as to keep workers in the lab safe. Setting up the workspace correctly will allow future researchers to spend more time improving the system. The setup included getting the robot and its controller delivered, purchasing and installing leveling feet to mount the robot on, and installing a 220V three phase power panel to power the sewing machine. A safety system also had to be selected, installed, and calibrated. The robot controller proved especially difficult to set up, due to slow support from Yaskawa and a confusing installation procedure for the new firmware. The robot controller needed firmware updates in order to be able to receive ROS messages. These updates were difficult to obtain and time consuming to install. After a slow delivery of the robot, the firmware updates delayed progress on the rest of the project significantly. In addition to the robot's delayed arrival, other setbacks such as receiving the wrong cables for the cameras occurred. In total, the time to set up everything in the workspace took close to three months as opposed to the two weeks originally anticipated.

The number of topics that were unfamiliar to the team prior to the project kickoff also impacted the slow execution of the project. Learning the theory as well as practical implementations of computer vision, motion planning, and user interface design greatly reduced the time spent on implementation, testing, and troubleshooting.

5.2.2 Timeline Adjustments

While the robot was being delivered and the robot controller was being configured, all motion planning work had to be done exclusively in simulation. This required scanning the robot's environment to represent all the possible collision objects in the simulation and combining the robot and gripper description files into one. This also took more time than expected due to the complex shape of the sewing machine and limitations of using an Xbox Kinect as a 3D scanner.

Other equipment delays and lack of knowledge blocked the team from making further progress. Lack of QT knowledge slowed development of the UI, and the lack of an SDK for the CaptoGlove slowed research on input devices for the project. The cameras were delivered two months late, which slowed the development of the computer vision system significantly. This was considered when ordering additional cameras, which were purchased from a website with a more reputable shipping and delivery operation. A full outline of the time line of this project is shown in Appendix M.

5.2.3 Risk Management

While timeline setbacks were plentiful, especially in the first half of the project, the team continued making progress. To counteract delays, the team developed different aspects of the project simultaneously. This allowed the team to make improvements while other parts of the project were stalled.

The biggest difficulty was the team's lack of knowledge at the beginning of the project. The team members all had different backgrounds, which is why the team was divided into three subteams: motion planning, computer vision, and user interface. The team spent a significant amount of time doing background research, which ultimately slowed down the development progress. Because of the team's lack of experience, lots of different methods were tested, such as the different user interfaces and methods for determining end effector positions. This trial and error slowed the team's overall progress but helped us gain new knowledge.

Another issue the team needed to overcome was communication. With the large size of the team and the disconnect between the sub teams, communication was especially strained. Conflicting schedules frequently disrupted the team's workflow and ability to meet. To combat this, the team met for short durations twice each week. This allowed the team to share updates on progress made, blockers that prevented progress from being made, and what each team member planned to work on next. Later in the project, longer meetings were added with at least one member of each subteam to work on integrating all the components.

5.2.4 Budget and Expenditure Justification

All of the items purchased for this project, excluding the robot, its controller, and the sewing machine, are shown in Appendix N. Although New Balance funded all of these items, minimizing cost was important so that this system could be a financially viable solution to implement in factories. Future research teams will not have to purchase a lot of equipment because it is already purchased and installed. The travel budget was set at $5,000, but did not limit the team due to the minimal travel required for the project.

5.3 Product Evaluation

In addition to the project's execution, it is important to evaluate the final product. There are several parameters that can be used to evaluate the performance of the project's end result. The success of the product can be evaluated using the requirements described in Section 3.

5.3.1 Hardware Requirements

Several of the requirements from Sections 3 and 4 describe features of the hardware of the system such as the cameras, workstation computer, and input devices. The e-Con See3CAM_CU135 USB cameras chosen satisfied the requirements discussed in Appendix C, and were able to complete the computer vision and user interface tasks. The workstation computer also satisfied its requirements, with the additions of a second network card and internal USB hub. The workstation launched all of the ROS nodes simultaneously, updated the camera views in the user interface with a frame rate of up to 30 FPS, and supported all four cameras at once.

The Wii Remote and Geomagic Touch haptic device satisfied the input device requirements to varying success. The fiducial markers, such as ArUco tags, were not researched because they were not conducive to a factory setting and required too many separate pieces that had to be reassembled and calibrated frequently. The CaptoGlove was intuitive for the user, but could not interface with the rest of the system and was therefore not a feasible option. The Wii Remote was easy for the user to understand but not intuitive while operating, as it does not closely resemble the action of sewing and has low accuracy. Appendix I shows the accuracy and precision of the positions calculated using the Wii Remote.

The Geomagic Touch haptic device was extremely accurate. Using the haptic device requires the user to specify where the sewing pattern should be instead of replicating the hand motions during sewing. This process is intuitive for the user because the pattern they specify on the fabric is exactly what the robot sews. Appendix I demonstrates the accuracy of the Geomagic Touch device.

Overall, the haptic device was the best input device based on its accuracy.

Another benefit is that factory workers already have experience using haptic devices for other industrial purposes, so they could easily learn to control the robot with it as well. The main limitation while using the Geomagic Touch device was that it could not be used quickly with all nodes running. If moved too quickly, the force sensor on the haptic device reaches its upper limit and crashes the node.

5.3.2 Motion Planning Performance

Several geometric planners available in OMPL were evaluated to determine which planner would fit this application the best. The results of this evaluation are shown in Appendix O (attached as a separate document). Based on this analysis, the LBKPIECE algorithm was chosen. This algorithm provided satisfactory trajectories for straight and curved lines; the motion planner was able to optimize trajectories and avoid collisions and singularities. However, the robot was not able to sew within the 1mm tolerance. The robot sewed between 0.5mm and 3.5mm away from the desired pattern at different points during the sewing process. Tests were inconsistent and rarely produced similar results. This is attributed to how the desired end effector positions were selected. The sewing pattern is discretized into 10 small lines, which does not output a smooth path. The path would be smoother if the number of lines the pattern is discretized into was increased, but this would dramatically increase planning time. The accuracy of the inverse kinematics (using the trac_ik package) and the repeatability of the robot's motors did not contribute to the sewing inaccuracies. The robot's motors have a repeatability of 0.1mm and the TRAC_IK algorithm was proven to be accurate enough for this application (see Appendix I).

5.3.3 Fabric Manipulation Performance

Another factor that impacted the low sewing accuracy was the difficulty in holding the fabric. The distance between the grippers must be maintained throughout the sewing task, or the fabric will lose its tension and sag between the grippers. A person has to place the vamp into the robot's grippers, and the robot's grippers do not move positions on the vamp during the sewing task. This means the person is required to place the vamp into the robot's grippers with enough tension that the fabric does not flop during sewing as shown in Figure 60.

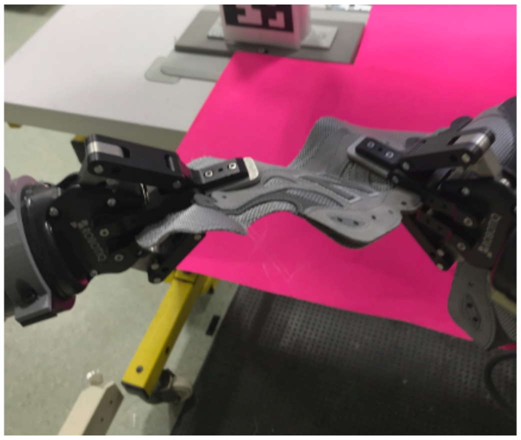


Figure 60: Vamp held incorrectly in robot's grippers

Despite the low accuracy of the sewing produced, the task was completed in the allotted 60 seconds, and the system showed great potential for increased efficiency and accuracy, discussed further in Section 6.

5.3.4 Overall System Performance

While this iteration of the project was not able to sew in 3D, many other requirements were satisfied and significant progress was made on the computer vision, motion planning, and user interface subsystems. First, the workspace setup was completed, which included enabling the robot controller to accept ROS messages, installing cameras and routing their cables, and mounting and calibrating a safety system. This safety system met the requirement of protecting human workers and also took up a small footprint, so it could be easily integrated into a factory setting. The computer vision system was also used to automatically calibrate the distance and orientation offsets between the robot and the sewing machine. This increased the mobility of the system and made it more applicable to a factory setting. Since the robot sews similarly to a human, no custom brackets or extra steps in the process were added; the robot can be seamlessly integrated into the factory. This also means the system can be easily adapted to sew different sizes or styles of shoes.

6 Recommendations for Future Work

This project is designed to be completed over multiple years. This chapter describes the next steps in the project for the computer vision, control interface, and motion planning subsystems, as well as improvements for the system as a whole.

6.1 Computer Vision

One of the main weaknesses of the computer vision subsystem is that it only performs well in a consistent and ideal environment. The ArUco tags are especially susceptible to environmental factors, such as lighting. The lighting must be very bright and constant, and the cameras must be mounted at the correct angle to maintain consistency. The system would improve if the computer vision subsystem could deal with inconsistencies such as shadows, slight changes in the angle of the cameras, or different colored backgrounds. A potential solution to these problems would be to apply brightness and contrast matching before the images are published.

It is also important to adjust and correct trajectories during the sewing process. Currently, all the trajectories are preplanned, but recalculating the trajectory while sewing would result in a more accurately sewn shoe. The cameras could be used for real-time feedback to adjust and correct sewing trajectories. There is already a camera on each wrist (two cameras total) pointed at the vamp, making this improvement solely software-based.

A future goal of this project is to sew in 3D. Some changes will need to be made to the computer vision system, such as redefining descriptors, so the landmark detection will function once the extra dimension is added. This could also require the addition of a wide FOV camera or a stereo camera, so that the entire shoe can be seen in one frame.

6.2 Control Interface

Most of the future work to be done on the control interface focuses on input devices. One improvement would be to implement real-time replication of the input device's trajectory executed by the robot. This would be more similar to true teleoperation, instead of recording the input device's movements and then mimicking them later with the robot. This would be helpful for teaching novice users how to sew better, as well as save the users a lot of time because they would not have to go through the process of recording, viewing, and choosing the best path. In addition, more input devices could be researched such as virtual reality gloves, other than the CaptoGlove, or other types of handheld controllers. This would give the users more options and provide them with more intuitive teleoperation of the robot. After the input device research is completed, future teams should test input devices in the factory with different seamsters to see which ones they prefer.

Other control interface improvements revolve around the user interface. Some visual enhancements like interesting colors, the ability to alter the text size, and other small tweaks could make the UI more appealing. One potential change would be hiding certain buttons while the robot is executing a task. The future UI should also be able to plot positions recorded from the haptic device in 3D instead of 2D. Plotting the positions recorded from the input devices in real time instead of recording the positions, saving them to a file, and then plotting them later, would also help provide feedback to the user. Displaying projected versus desired sewing patterns on the UI would also be useful to the user to see what waypoints the CV subsystem identified and what the robot plans on sewing. In order to evaluate the usability of the UI and identify future modifications, it is important to collect feedback from factory workers.

A handy feature that we think should be implemented is giving the user the ability to specify the sewing pattern. Currently, the landmark identification is performed by the computer vision system on the backend, meaning the user does not have the ability to specify a sewing pattern without an input device. This means only the programmers have the ability to change which sewing pattern is executed by the robot. This would be a disadvantage if the robot needs to change tasks.

6.3 Motion Planning

The most significant change to the motion planning subsystem will be implementing sewing in 3D. Currently, the system assumes a constant z-position for the robot's end effectors, but this would need to dynamically change to sew 3D shoes instead of flat, 2D fabric.

Another improvement to the motion planning subsystem would be to implement a true learning-by-demonstration approach to correlate the CV waypoints with the end effector positions. This subsystem needs to expand to include curved patterns, 3D patterns, and patterns, which require the robot to reposition its hands on the fabric while sewing. Learning by demonstration could help teach the robot to sew more, different patterns as well as make its motions more natural, resulting in fewer sewing errors.

6.4 Overall System Improvements

There are a lot of small improvements that could be made to the overall system. Currently, the entire sewing process takes approximately 45 seconds. Writing code more efficiently could decrease the sewing time, as well as computational resources. Another small improvement would be disabling all buttons on the user interface during autonomous mode so the user does not interfere with the predefined trajectory execution.

A significant flaw within the current system is the grippers' inability to grasp fabric from a flat surface. Currently, a person must pick up the vamp and hand it to the robot's grippers. This system is not ideal because it would be unsafe in a factory environment where the robot is moving at high speeds and cannot be easily disabled. A system could be devised where the vamp is picked from a non-flat surface, such as an elevated platform. Another alternative solution would be developing custom grippers for the robot that allow fabric grasping from flat surfaces. Implementing these improvements would make the system safer to use and increase efficiency in factories.

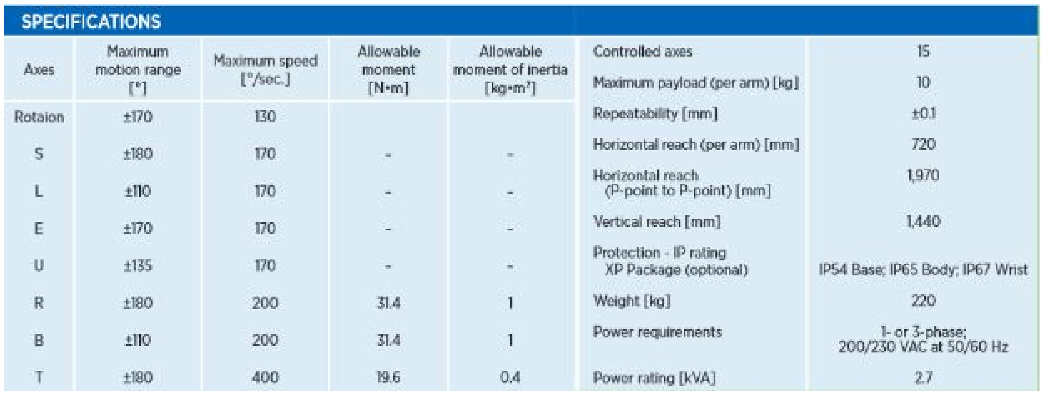
While this system was developed for use in a factory, no testing with factory workers was done. The final product of the project could be used in a factory setting, so the system needs to fit into New Balance's existing workflow. In the future, feedback from factory workers would help cater the product more towards its intended users. Feedback from future users would also illuminate problems with the practical usability of the system. Overall, there are many improvements could be made to the system, but the framework and equipment exist in order to make future improvements simpler. Appendix P describes in more detail the shortcomings of the current implementation of this system.

Glossary

CV	Computer Vision. 80
DOF	Degrees Of Freedom. 5
FCL	Flexible Collision Library, ROS package for collision detection and avoidance. 96
FOV	Field Of View. 126
FPS	Frames Per Second. 121
OMPL	Open Motion Planning Library, ROS package providing motion planning algorithms. 96
ORB	Orientated FAST and Rotated BRIEF algorithm. 89
ROS	Robot Operating System. 10
SDK	Software Development Kit. 69
SSPP	Simple Structure Passing Protocol, a lightweight, cross-platform messaging protocol for structured data. 69
UI	User Interface. 61
VR	Virtual Reality. 67

Appendices

A Robot Specification Sheet

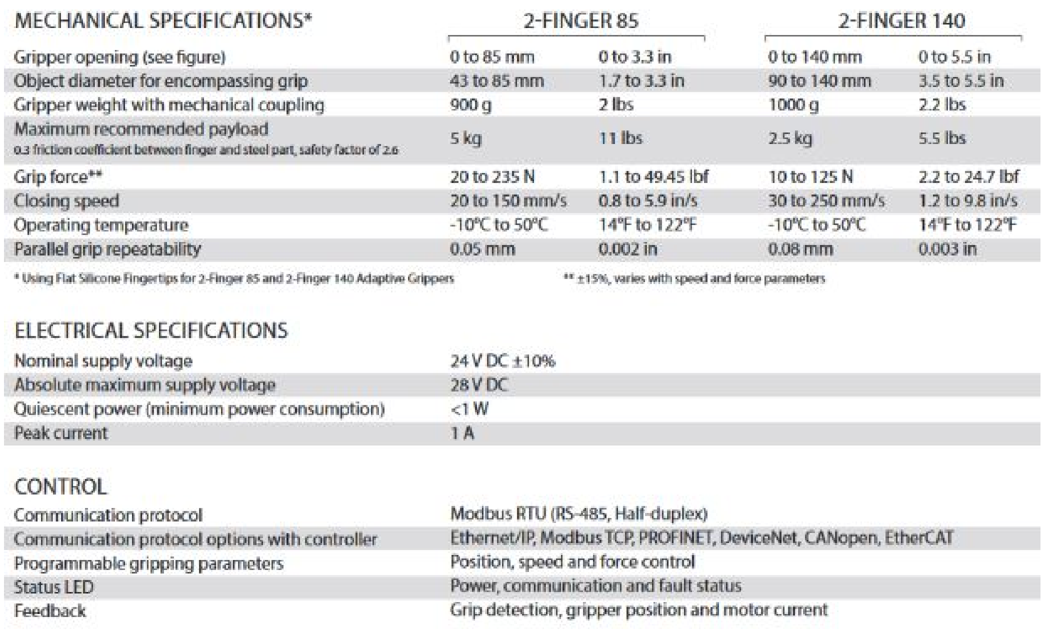


SPECIFICATIONS

Axes	Maximum motion range [°]	Maximum speed [°/sec.]	Allowable moment [N•m]	Allowable moment of inertia [kg•m²]
Rotaion	±170	130		
S	±180	170	-	-
L	±110	170	-	-
E	±170	170	-	-
U	±135	170	-	-
R	±180	200	31.4	1
B	±110	200	31.4	1
T	±180	400	19.6	0.4

Controlled axes	15
Maximum payload (per arm) [kg]	10
Repeatability [mm]	±0.1
Horizontal reach (per arm) [mm]	720
Horizontal reach (P-point to P-point) [mm]	1,970
Vertical reach [mm]	1,440
Protection - IP rating XP Package (optional)	IP54 Base; IP65 Body; IP67 Wrist
Weight [kg]	220
Power requirements	1- or 3-phase; 200/230 VAC at 50/60 Hz
Power rating [kVA]	2.7

B Robotiq 2-Finger Adaptive Gripper Specifications Sheet



MECHANICAL SPECIFICATIONS*	2-FINGER 85		2-FINGER 140	
Gripper opening (see figure)	0 to 85 mm	0 to 3.3 in	0 to 140 mm	0 to 5.5 in
Object diameter for encompassing grip	43 to 85 mm	1.7 to 3.3 in	90 to 140 mm	3.5 to 5.5 in
Gripper weight with mechanical coupling	900 g	2 lbs	1000 g	2.2 lbs
Maximum recommended payload 0.3 friction coefficient between finger and steel part, safety factor of 2.6	5 kg	11 lbs	2.5 kg	5.5 lbs
Grip force**	20 to 235 N	1.1 to 49.45 lbf	10 to 125 N	2.2 to 24.7 lbf
Closing speed	20 to 150 mm/s	0.8 to 5.9 in/s	30 to 250 mm/s	1.2 to 9.8 in/s
Operating temperature	-10°C to 50°C	14°F to 122°F	-10°C to 50°C	14°F to 122°F
Parallel grip repeatability	0.05 mm	0.002 in	0.08 mm	0.003 in

* Using Flat Silicone Fingertips for 2-Finger 85 and 2-Finger 140 Adaptive Grippers

** ±15%, varies with speed and force parameters

ELECTRICAL SPECIFICATIONS	
Nominal supply voltage	24 V DC ±10%
Absolute maximum supply voltage	28 V DC
Quiescent power (minimum power consumption)	<1 W
Peak current	1 A

CONTROL	
Communication protocol	Modbus RTU (RS-485, Half-duplex)
Communication protocol options with controller	Ethernet/IP, Modbus TCP, PROFINET, DeviceNet, CANopen, EtherCAT
Programmable gripping parameters	Position, speed and force control
Status LED	Power, communication and fault status
Feedback	Grip detection, gripper position and motor current

C Trade Study

This trade study describes how the cameras and grippers were chosen.

C.1 Camera Selection

Different cameras were chosen for different purposes. There are three main tasks that the CV subsystem needed to complete: generate a series of waypoints along the sewing pattern, calculate the distance between the robot and the sewing machine, and provide the user high definition views of the workspace at a frame rate of at least 24 FPS. Table 6 below shows the Decision Making Matrix constructed to help select the cameras to mount on the robot's wrists. The weights of the categories signify the importance of the category, with higher weights being more important. Each camera is given a score in each category out of ten, which is then multiplied with the category's weight. The weighted scores for each category are totaled for each camera, providing each camera's overall score.

	Weight	See3CAM_CU135	See3CAM_CU51	See3CAM_CU40
Resolution	0.3	10	6	4
Frame rate	0.3	9	9	10
Cost	0.2	9	10	10
Size	0.1	10	10	10
Interface	0.05	10	10	10
Codec Usage	0.05	10	10	10
SCORE		9.5	7.5	7.2

Table 6: Camera Decision Matrix

The cameras for the offset calculations needed to be high resolution, because ArUco tags were used. While these calculations only use one frame, the cameras are also used to provide the user close-up views of the sewing, so a high frame rate is needed. The camera chosen to meet these requirements was the eCon CU135 camera. This camera is also 5MP, has a field of view of 60°, and a USB3.0 connection. This camera has a fixed focus that is changed by rotating the lens assembly on the camera module. Using a fixed focus camera avoids the issue of the camera refocusing during a task. USB3.0 was chosen over USB2.0 as it allows the camera to provide a 4K video stream at a stable 30FPS with the camera module's built-in MPEG compression format.

In order to view the entire vamp in one camera frame, the camera used for landmark detection needed to have a wide FOV. It also needed to have a high resolution for accurate landmark detection. This computation is done on a single still image, so the camera does not need to have a high frame rate. The ELP 5MP camera with a 120° lens and USB2.0 connection was chosen and mounted on the sewing machine. One more camera was needed to provide another viewing angle to the operator. A Microsoft 1080P webcam with autofocus and USB2.0 connection was chosen because it was cheap, available, and easy to mount on the tripod.

C.2 Gripper Selection

Table 7 below shows the Decision Making Matrix used to decide which grippers to use as the robot's end of arm tooling. The weights of the categories signify the importance of the category, with higher weights being more important. Each gripper is given a score in each category out of ten, which is then multiplied with the category's weight. The weighted scores for each category are totaled for each gripper, providing each gripper's overall score.

Parameter	Description	Weight	Robotiq 3 Finger	Robotiq 2 Finger	Custom Gripper
DOF	How many DOF does gripper have?	0.1	7	6	6
Grip Strength	How strong is grip on target object?	0.1	5	8	3
Compatibility	Is it compatible with current hardware/software?	0.3	8	8	5
Cost	How expensive is it?	0.2	1	6	8
Manipulability	How many ways can it grip an object?	0.1	7	7	7
Reliability	Will gripper work for extended periods of time?	0.1	9	9	5
Contact Area Size	What is the area of the gripper that contacts the object?	0.1	3	7	3
Totals		1	5.7	7.3	5.5

Table 7: Gripper Decision Matrix

The Robotiq 2-Finger grippers were chosen because they have a higher grip strength and contact area than the 3-Finger grippers, and were more cost effective than the custom-made grippers.

D UI Feature Lists: Buttons

Label	Function
Start camera needle offset calibration	Calibrate distance between robot and sewing machine
Acknowledge calibration complete	User acknowledges calibration is complete
Go to Home	Sends robot to home position
Go to Start	Sends robot to the start sewing position
Find landmarks	Use CV to locate sewing waypoints
Close Gripper	Closes grippers on the robot
Open Gripper	Opens grippers on the robot
Auto mode	Sends command to sew CV waypoints
Assist mode	Opens window for input devices to be used
Confirm sewing waypoints	Sends command to robot that it found the waypoints to sew and can start to sew

Table 8: Desired Buttons on UI

E UI Features Lists: Messages

Label	Function
Offset Calibration in Progress	Tells the user needle-camera offset calibration is in progress; Wait to continue until this is complete
Sewing Waypoints Identified	Tells the user the sewing waypoints on the fabric have been identified and it is ready to sew when the ”Acknowledge Offset Calibration Complete” button has been selected
Done	Lets the user know that the sewing task is complete

Table 9: Desired Messages on UI

F UI Features Lists: Camera Views

- Live camera feed
- Live edge detection video
- Display landmarks on live camera feed
- Display sewing pattern waypoints on camera feed
- Display desired sewing path on camera feed
- Display projected sewing path on camera feed

G UI Feature Lists: Input Device Interface

- Display raw data
- Display the input device name that is connected
- Display output of input device position recording
- Storing input device position trajectory
- Button for calibration
- Button for recording the data
- Button for sending the desired trajectory to the robot for execution

H Pros and Cons of Potential Input Devices

Device	Pros	Cons
VR Glove (CaptoGlove)	• IMU - calculate position AND orientation • Comfortable for user • Low cost (CaptoGlove) • Intuitive to use • Low latency	• SDK recently developed - potential for bugs • IMU could be noisy • Low sampling frequency • Need to calculate position and orientation

Table 10 continued from previous page

Haptic device (Geomagic Touch)	• High precision • Can track EEF position • Well-developed SDK • Available to use in lab	• Difficult to operate two hands simultaneously • Less intuitive • Take up more space
Fiducial marker (ArUco tag)	• Can track anything • Low cost	• Difficult to track fingers • Some noise • Difficult to adapt to factory (need to recalibrate) • Takes long time to set up

Table 10 continued from previous page

Wii Remote	• Low cost • IMU - calculate position AND orientation • Easy to set up • ROS package exists • Used to track position	• Not intuitive • Built-in IMU is noisy • IMU has low accuracy and precision • Only sensor is IMU

Table 10: Pros and Cons of Input Devices

I Test Plans

This section describes how each subsystem of the project was tested and any data produced from the testing.

I.1 Offset Calculations

The purpose of the offset calculations is to measure the distance between the robot and the sewing machine. To test it, the true distances between the robot base frame and the sewing machine frame in the x, y, and z directions were measured in the workspace with a tape measure. These measurements were recorded and then compared to the offsets calculated using the ArUco tags. Table 11 below shows the offset calculations performed five times. This is shown to determine if the offset calculations are consistent. Because the standard deviations of each set of calculations is small (less than 5), the offset calculations are consistent between executions.

	Run 1	Run 2	Run 3	Run 4	Run 5	Std Dev
X (m)	0.6253645	0.6247383	0.6253444	0.6253444	0.6257054	0.0003492
Y (m)	0.0148355	0.0149498	0.0150294	0.0150294	0.0150895	0.000098
Z (m)	0.0391812	0.0394361	0.0393592	0.0393592	0.0397369	0.0002030

Table 11: Consistency of Offset Calculations

Table 12 below shows the measured distances and the output of the offset calculations. The Offset Calculation Average column averages the calculations of the five executions shown in the table above. The final column shows the difference between the measured distances and the average of the calculated distances. The difference for the distances of each axis is below one centimeter, so the offset calculations were determined to be accurate.

	Measured Distance	Offset Calculation Average	Difference
X (m)	0.65	0.02	-0.3
Y (m)	0.6252994483	0.01498675434	0.03941457778
Z (m)	0.02470055166	0.005013245656	-0.3394145778

Table 12: Accuracy of Offset Calculations

I.2 Inverse Kinematic Solving

The TRAC-IK algorithm was used to solve the inverse kinematics of the robot. To test the accuracy of the IK solving, the forward kinematics were solved using a set of desired joint angles. Then, the desired end effector position from the forward kinematics was given as an input to the TRAC-IK solver. The output from the TRAC-IK solver should closely match the original desired joint angles. The original desired joint angles, output from the TRAC-IK solver, and differences between the two values are shown below. This test was performed on only one arm because the arms are symmetrical and should produce the same IK solutions.

Joint	Desired Joint Angle (rad)	TRAC-IK Output (rad)	Difference (rad)
1	0	1.39621e-06	-1.39621e-06
2	1	0.999888	1.12e-04
3	0	-5.81496e-07	5.81496e-07
4	1.04	1.0399	1.0e-04
5	-0.62	-0.620006	6.0e-06
6	-0.76	-0.759991	-9.0e-06
		Average	3.4698e-05

Table 13: Accuracy of TRAC_IK Algorithm

As shown in Table 13 above, the difference between the desired joint angles and the TRAC-IK output was very small, meaning the accuracy of the TRAC-IK output is high.

I.3 Landmark Identification

The goal of the landmark identification step in the computer vision subsystem is to locate certain points in space. To verify that the locations identified were correct, the landmark identification routine was performed five times and the locations of the landmarks were recorded. Then, these locations were converted from number of pixels to mm. Then, the locations of the landmarks were measured in the physical space. Tables 14 and 15 below show the locations of the first landmark as identified by the computer vision subsystem in both pixels and meters. These calculations were computed five times, and the average standard deviations are shown as well. Note, only the x and y locations are recorded because this project only sewed in 2D.

	Run 1	Run 2	Run 3	Run 4	Run 5	Average	Std Dev
X	580.3879	583.2025	577.3796	577.9849	575.81	578.953	2.89
Y	155.1264	163.1964	173.3475	215.6062	235.8049	188.6163	35.1997

Table 14: Precision of Landmark Identification (in pixels)

	Run 1	Run 2	Run 3	Run 4	Run 5	Average	Std Dev
X	0.153560	0.154305	0.152765	0.152925	0.152349	0.153181	0.000764
Y	0.041043	0.04317	0.045864	0.057045	0.06239	0.049904	0.009313

Table 15: Precision of Landmark Identification (in m)

As demonstrated by the low standard deviation, the landmark identification routine is consistent between computations. Table 16 below shows the locations of the landmarks found by approximating the distance of the landmark from the image origin, using the size of the image (1280x1024 pixels). The difference between the measured and identified landmark locations is also shown.

	Identified	Measured Location	Measured - Identified
X (m)	0.1531813203	0.153749375	0.0005680547102
Y (m)	0.049904736	0.05566833333	0.005763597327

Table 16: Accuracy of Landmark Identification

Because the distance between the measured and identified landmark locations is small, the locations identified for the first landmark are accurate. The accuracy of the y coordinate is less accurate, but still accurate enough for this application.

I.4 Robot Controller

To test that the robot controller is properly connected to the workstation computer, the following topics were echoed: /sda10f/sda10f_r1_controller/joint_states (left arm joint angles) /sda10f/sda10f_r2_controller/joint_states (right arm joint angles)

These two topics are where the robot's joint states are published. If no data is being published, the robot controller is not connected properly.

I.5 Gripper and Gripper Controller

To test that the grippers and the gripper controllers were connected correctly, the grippers were actuated using the Robotiq gripper control examples. Both grippers were opened, closed, and opened half way to ensure they responded to commands as expected. Once we verified that the example Robotiq program controlled the grippers correctly, the gripper control functionality was added to the user interface.

Even though the vamp fabric is very thin at 3.35mm, the grippers are able to securely hold the fabric when completely closed, in a parallel position. The fabric could not be removed until the grippers were opened. This affirmed that the grippers had enough force to grip the fabric while sewing.

I.6 Camera Nodes

The functionality of the camera nodes was tested to determine if the images from the camera were being published properly. The camera nodes publish the camera images to the following topic: /send_feeds

This topic was echoed to determine if the feeds were being published. After the publication was verified, the user interface was used to see if the camera images being published had the proper focus and contrast. As described in Section 4.4, there are four buttons on the user interface that, when clicked, display the view from the requested camera. Each button was clicked to view each camera view and the focus and bandwidth were manually adjusted as necessary.

I.7 User Interface

The purpose of the user interface is provide a simple way to control and understand the sewing process. The buttons on the UI are used to signal other processes to start or stop. To test the functionality of the buttons before the rest of the system was done, each button was clicked and the topics to which the buttons publish were echoed.

I.8 Input Devices

Before integrating the input devices into the main UI, the accuracy and precision of the poses produced using the haptic device and Wii Remote. The Wii Remote has a gyroscope and an accelerometer, so the position was calculated by integrating the values from the sensors, and the orientation was calculated by fusing the values from the sensors. The value zeta is the gyroscope drift gain used in calculating the orientation of the Wii Remote. Zeta had to be tuned so the drift was as close to zero as possible. Tables 17 and 18 show the maximum, average and standard deviation of the drift measured by the Wii Remote with two different values of zeta. These were measured by taking 6000 recordings over 60 seconds with the Wii Remote on a flat surface.

	Average Drift (rad/sec)	Max Drift (rad/sec)	Std Deviation (rad/sec)
X	-0.100534214	-0.071807796	0.011311008
Y	0.070685799	0.109823688	0.014569631
Z	-0.058525334	-0.022175937	0.013323598

Table 17: Precision of Wii Remote Data when Zeta = 0.360094977

	Average Drift (rad/sec)	Max Drift (rad/sec)	Std Deviation (rad/sec)
X	0.018459795	0.255551274	0.059429037
Y	0.122382797	0.259775262	0.070016938
Z	-0.151184914	-0.23256821	0.052722812

Table 18: Precision of Wii Remote Data when Zeta = 0.3771788462

As shown above, both the average and the maximum drift was lower for all three axes when Zeta = 0.360094977. The drift produced with this zeta value was low enough for this application. The standard deviation was also very low, indicating that the orientation calculations are consistent.

The same test was performed using the haptic device. The position of the Geomagic Touch haptic device tool tip and the orientation of the rotation axis were recorded for 60 seconds while the tip was stationary. The average drift of the position in each axis is shown below. Table 19 shows the data from this test.

	Average Drift	Standard Deviation
X	0	0
Y	-1.16699E-05	1.01894E-20
Z	3.22266E-06	5.0947E-21

Table 19: Precision of Haptic Device: Position (mm)

As shown above, the average drift and the standard deviation were very small, indicating very precise and consistent measurements. Similar results were seen for the orientations recorded by the haptic device as well, shown in Table 20.

	Average	Standard Deviation
X	0.002443841	0.000455467
Y	-0.021791834	0.001681716
Z	-0.094716878	0.007997061
W	-0.022084072	0.00222304

Table 20: Precision of Haptic Device: Orientation (rad)

Due to the low average drift and low standard deviation, the orientation measurements were also proven to be precise and consistent.

The previous input device measurements only demonstrated precision, as the input device was held still. The accuracy of the haptic device was determined by tracing a sewing pattern with the haptic device tool. The curved and straight paths used in this testing are shown below in Figure 61.

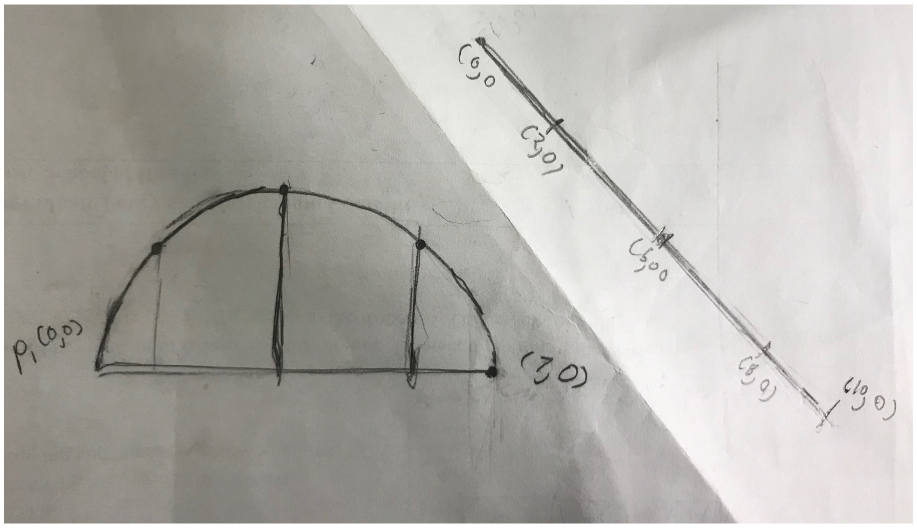


Figure 61: Curved and Straight Paths Traced During Testing

A high accuracy would mean the positions recorded by haptic device match the x,y points along the paths. Tables 22 and 21 below show the expected and measured positions of five sample points from a straight path and a curved path, as well as the difference between the expected and measured positions along each axis.

(mm)	Point 1	Point 2	Point 3	Point 4	Point 5
Expected					
X	0	11	32	55	70
Y	0	21	33	24	0
Z	0	0	0	0	0
Measured					

Table 21 continued from previous page

X	-0.000793457	14.04677582	35.72566223	56.53177643	64.48575106
Y	0	23.09552574	31.45517635	23.80514336	6.117397308
Z	-0.001449585	-0.774894714	-0.955627441	-0.958740234	-0.87693787
Difference					
X	0.000793457	3.04677582	3.72566223	1.53177643	5.51424894
Y	0	2.09552574	1.54482365	0.19485664	6.117397308
Z	0.001449585	0.774894714	0.955627441	0.958740234	0.87693787

Table 21: Accuracy of Haptic Device Positions along a Curved Path

(mm)	Point 1	Point 2	Point 3	Point 4	Point 5
Expected					
X	0	20	50	80	90
Y	0	0	0	0	0
Z	0	0	0	0	0
Measured					
X	0.069301605	16.77160835	45.70610142	71.60544968	88.64770126
Y	0.037990570	0.968406677	1.022716522	2.219299316	1.856254578
Z	0.002616882	-0.045356751	0.052444458	-0.026756287	1.098648071
Difference					
X	0.069301605	3.22839165	4.29389858	8.39455032	1.35229874
Y	0.037990570	0.968406677	1.022716522	2.219299316	1.856254578

Table 22 continued from previous page

Z	0.002616882	0.045356751	0.052444458	0.026756287	1.098648071

Table 22: Accuracy of Haptic Device Positions along a Straight Path

As shown above, the differences between the expected and measured positions were very low, so the accuracy of the haptic device is very high. A similar testing process was performed to determine the accuracy of the Wii Remote. Below in Tables 23 and 24 are the recorded and measured positions.

(mm)	Point 1	Point 2	Point 3	Point 4	Point 5
Expected					
X	0	11	32	55	70
Y	0	21	33	24	0
Z	0	0	0	0	0
Measured					
X	-0.00456677	-406.419484	-2510.70599	-8383.18530	-181853.518
Y	-0.00254000	-135.320333	-872.803610	-3506.33695	-117120.295
Z	0.002770094	339.3697409	2334.804566	7737.231903	203298.9882
Difference					
X	0.004566773	395.4194843	2478.705999	8328.1853	181783.5185
Y	0.002540000	114.3203335	839.8036101	3482.336951	117120.2953
Z	0.002770094	339.3697409	2334.804566	7737.231903	203298.9882

Table 23: Accuracy of Wii Remote Positions along a Curved Path

(mm)	Point 1	Point 2	Point 3	Point 4	Point 5
Expected					
X	0	20	50	80	90
Y	0	0	0	0	0
Z	0	0	0	0	0
Measured					
X	0.014423513	1189.192756	6254.908336	21230.34552	62417.38022
Y	-0.00254707	-15.5346527	-71.5417416	-129.251083	-1555.94361
Z	-0.00676015	-131.600906	-480.112922	-1316.88057	-3328.30970
Difference					
X	0.014423513	1169.192756	6204.908336	21150.34552	62327.38022
Y	0.002547079	15.53465272	71.54174166	129.2510835	1555.943619
Z	0.006760154	131.6009067	480.1129222	1316.880574	3328.309709

Table 24: Accuracy of Wii Remote Positions along a Straight Path

The differences between the expected and measured orientations and positions show that the Wii Remote has much lower accuracy than the haptic device. This is why the haptic device is the recommended input device.

I.9 Integration

After each component was tested individually, they needed to be integrated together. First, a launch file was written that launched all of the

ROS nodes at once. With all the nodes running, the functionality of all components was tested again. Functions occurred significantly slower with all nodes running at once, but they still worked.

J Construction

There is only one component of the project that needs to be constructed: the wrist camera mounts discussed in Section 4.5. The mount was designed in Autodesk Inventor and consists of five pieces in total. Figure 62 shows an exploded view of the parts in Inventor.

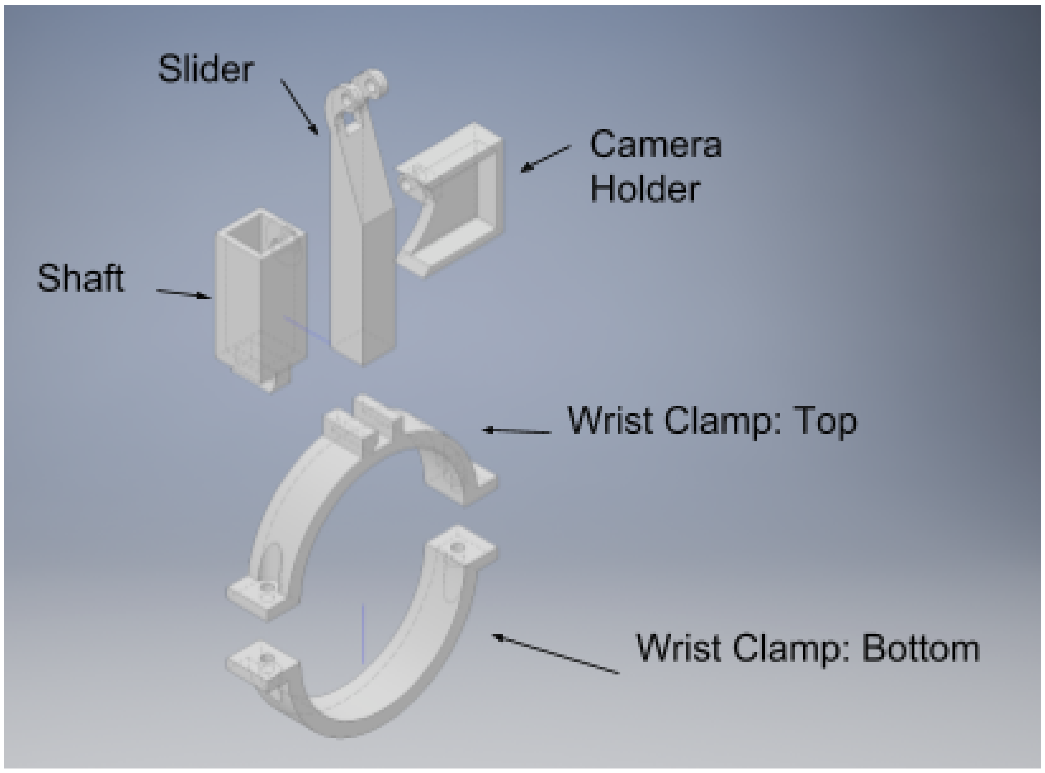


Figure 62: Exploded view of wrist camera mount in Inventor

The two wrist clamp pieces fit together around the robot's wrist and are attached using two 2cm 3M bolts and lock nuts. The camera slides into the camera holder and is secured with zip ties. A 3cm 3M bolt and lock nut are used to secure the camera holder onto the slider. Then, the slider fits into the shaft and is secured with a 1cm 2.5M set screw. The shaft slides into the groove on the top wrist clamp. Figure 63 shows an assembly of the pieces in Inventor.

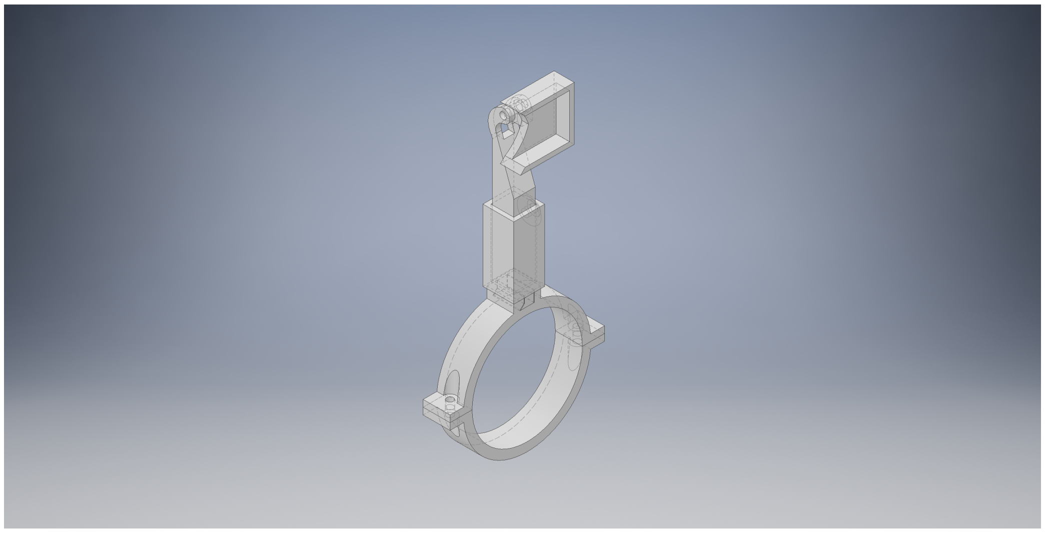


Figure 63: Assembled view of wrist camera mount in Inventor

K Link Masses and Inertial Matrices

Link	Mass(kg)	Inertial Matrix
1_s	6	xx=.000054, xy=0, xz=0, yy=.000054, yz=-.000006, zz=.000022
2_l	6	xx=.000022, xy=0, xz=0, yy=.000009, yz=.000006, zz=.000022

Table 25 continued from previous page

3_e	6	xx=.000013, xy=0, xz=0, yy=.000013, yz=.000001, zz=.000005
4_u	6	xx=.000015, xy=0, xz=0, yy=.000004, yz=-.000003, zz=.000015
5_r	6	xx=.000007, xy=0, xz=0, yy=.000007, yz=-.000001, zz=.000002
6_b	3	xx=.000005, xy=0, xz=0, yy=.000002, yz=.000001, zz=.000005
7_t	3	xx=.001, xy=0, xz=0, yy=.001, yz=0, zz=.001

Table 25: Link Masses and Inertial Matrices

L Motion Planning Analysis Criteria

- Planning time = Time it takes from when the Motion Planner receives desired EEF positions to when the robot starts moving
  - Less than 45 seconds
- Execution time = Time it takes from when the robot starts moving to when it stops moving and presents UI with "Done" message
  - Less than 15 seconds
- Obstacle Distance = Since the only obstacle is the sewing machine, this is the distance between the point on the robot closest to the sewing machine and the sewing machine
  - More than 3cm
- EEF Travel distance = distance traveled by the end effectors from the time the fabric is grasped to the time the fabric is dropped

  - EEF R: Less than 50cm
  - EEF L: Less than 50cm
- Joint Travel = total radians that each joint rotates
  - R1: Less than pi/2
  - R2: Less than pi
  - R3: Less than pi
  - R4: Less than pi
  - R5: Less than pi/8
  - R6: Less than pi/8
  - R7: Less than pi/8
  - L1: Less than pi/2
  - L2: Less than pi
  - L3: Less than pi
  - L4: Less than pi
  - L5: Less than pi/8
  - L6: Less than pi/8
  - L7: Less than pi/8
- Repeatability = consistency between planner outcomes in the exact same scenario
  - Measure the criteria mentioned above for several planner outcomes. Each planning criteria from all planning outcomes must be within 1 standard deviation of the mean. A small standard

deviation is also desirable for all criteria, although it is impossible to state what it should be.

- Ability to perform on a variety of problems = consistent performance in different scenarios (sewing different patterns with different sewing machine offsets)
  - Measure all the criteria mentioned above for each sewing scenario. Each planning criteria from all sewing scenarios must be within 1 standard deviation of the mean. A small standard deviation is also desirable for all criteria, although it is impossible to state what it should be.

M Timeline

Research

The first step of the project was to research the relevant topics including motion planning, computer vision, and ROS.

- Subtasks: background research, background chapter writing, gap analysis
- Duration: 4 months (through November 2017)
- Start Time: August 2017

Requirements Analysis

This step included developing the concepts of operations as described in

Section 3.

- Subtasks: stakeholder identification, use case and user story development, creating lists of requirements
- Duration: 1 month (through October 2017)
- Start Time: October 2017

Setting up the Workspace

Many different components needed to be installed to properly setup the workspace. This was an ongoing process throughout the first half of the project.

- Subtasks:
  - Workstation computer setup (obtain desktop, research requirements, install appropriate OS and other libraries)
    - Duration: 1 week (through Mid-September 2017)
    - Start Time: Mid-September 2017
  - Installing robot (delivered from New Balance, moved to CIBR lab, leveling feet installed)
    - Duration: 2 weeks (through Mid-September 2017)
    - Start Time: September 2017
  - Assembling sewing machine (delivered from New Balance, assembled in CIBR lab)
    - Duration: 2 weeks (through September 2017)

    * Start Time: Mid-September 2017

  - Power installation for sewing machine
    * Duration: 1 month (through mid-October 2017)
    * Start Time: Mid-September 2017
  - Enabling the robot controller to receive ROS messages (update firmware, change language from Japanese to English)
    * Duration: 2 months (through Mid-December 2017)
    * Start Time: Mid-October 2017
  - Installing Robotiq Grippers (receive custom mounts, connect controllers to computer, install mounts and grippers on end effectors)
    * Duration: 2 weeks (through October 2017)
    * Start Time: Mid-October 2017
  - Installing safety system (order sensor, calibrate sensor for workspace, install sensor and LED warning lights)
    * Duration: 1 month (through November 2017)
    * Start Time: November 2017

User Interface Development: QT and C++

This phase is the initial user interface development described in Section 4.4.1

- Subtasks: display status messages from robot controller, allow user to select a desired sewing pattern, display camera view
- Dependencies: workstation computer setup complete
- Steps:

  - Design: October 2017 - Mid-October 2017
  - Build: Mid-October 2017 through December 2017
  - Test: Mid-October 2017 through December 2017
  - Document: December 2017 through February 2018
  - Review: December 2017

User Interface Development: Tkinter and Python

This phase is the final user interface development described in Section 4.4.1

- Subtasks: display status messages from robot controller; allow the user to command the robot to: go to home position, go to start position, open grippers, close grippers; display all four camera views
- Dependencies: camera setup complete, workstation computer setup complete, robot controller enabled to receive ROS messages, home and start positions defined
- Steps:
  - Design: January 2018
  - Build: January 2018 through February 2018
  - Test: February 2018 through March 2018
  - Document: March 2018 through April 2018
  - Review: March 2018 through April 2018

Input Device Research and Implementation

This phase incorporated researching and attempting to implement multiple input devices for operating the robot in 'assist' mode.

- Subtasks: obtain devices, communicate with the devices using ROS messages, calculate and/or publish the poses recorded by the device
- Dependencies: workstation computer setup complete
- Steps:
  - Design: Mid-October 2017 through November 2017
  - Build: Mid-November 2017 through February 2018
  - Test: February 2018 through March 2018
  - Document: March 2018 through April 2018
  - Review: March 2018 through April 2018

Computer Vision Camera Setup

This phase involved choosing, ordering, and mounting the cameras for the computer vision subsystem.

- Subtasks: Select cameras, order cameras, design mounts for the cameras, mount the cameras

Computer Vision Subsystem Development

After all the cameras were installed, the computer vision subsystem development began.

- Subtasks: identify the position and orientation of the vamp in space, find the position and orientation offsets from the robot to the sewing machine, identify the position and orientation of the sewing pattern in the robot's grippers
- Dependencies: workstation computer setup, camera setup

- Steps:
    - Design: September 2017 through January 2018
    - Build: November 2017 through March 2018
    - Test: December 2017 through March 2018
    - Document: February 2017 through March 2018
    - Review: March 2018 through April 2018

Simulating the Workspace

In order to test the motion of the robot, the workspace needed to be modeled in simulation. Two simulation tools were tested, as described in Sections 4.6.3 and 4.6.4.

- Subtasks: load robot, grippers, Force-Torque sensor, and sewing machine in simulation environment; define collision space
- Dependencies: workstation computer setup
- Steps:
    - Design: September 2017 through October 2017
    - Build: October 2017 through January 2018
    - Test: October 2017 through January 2018
    - Document: January 2018 through Mid-March 2018
    - Review: February 2018 through March 2018

Motion Planning Subsystem Development

After the robot's motion was simulated, the development of the motion planning subsystem began.

- Subtasks: create collision-free trajectories, perform inverse kinematics calculations, define transformation matrices between all coordinate systems of the workspace
- Dependencies: workstation computer setup, workspace setup
- Steps:
  - Design: October 2017 through Mid-December 2017
  - Build: November 2017 through February 2018
  - Test: December 2017 through Mid-March 2018
  - Document: February 2018 through Mid-March 2018
  - Review: March 2018 through Mid-April 2018

End Effector Pose Determination

Once a motion planner was selected and the inverse kinematics of the robot could be solved, the desired end effector poses needed to be determined. These poses depend on the desired sewing pattern as well as how the robot grasped the fabric.

- Dependencies: workstation computer setup, workspace setup, computer vision system setup
- Steps:
  - Design: December 2017 through Mid-January 2018
  - Build: February 2018 through April 2018
  - Test: Mid-March 2018 through April 2018
  - Document: March 2018 through April 2018

– Review: Mid-March 2018 through April 2018

N Budget and Expenditure Justification

Item and Link	Vendor	Price	Justification
See3CAM_CU135	E-Con	3 x $109	High FOV, Adjustable Frame Rate and Resolution
5 MP HD USB Camera Module	EPL	$45.90	Wide FOV, high resolution
HDCQ12 12ft 2.1mm x 5.5mm	Hanvex	4 x $7.99	Power cable extensions to power the barrel lights on the sewing machine
12V DC Output Lithium Ion Battery	Talentcell	2 x $22.99	High capacity power sources for LEDs mounted on sewing machine
AC to DC 5V 2A Regulated Power Supply	Ziumier	1 x $10.59	AC to DC adapters to charge batteries and temporarily power LEDs
SuperSpeed USB 3.0, Active Extension Cable	Cable Matters	4 x $34.99	USB 3.0 cables, capable of handling the bandwidth of 4K cameras at 10 meters

Table 26 continued from previous page

2.1mm x 5.5mm to 3.5 x 1.35mm DC 12V Adapter	DEYF	2 x \$7.46	Extensions to reach lighting on the sewing machine
SATA Power Extension Cable	CRJ Elect	\$9.49	Mount close to the cameras and sewing workspace
Cable tie mounts 25x25(mm)	Monoprice	\$6.38	Ensure cables do not get caught in motion of robot
85mm adaptive 2-Finger gripper and controllers	Robotiq	2 x \$4,800	Grip the fabric securely, Easy installation onto robot
MS7701B Tripod Boom	OnStage	\$24.95	Mount camera in a configurable location
Swivel Smartphone Holder	Acc. Basics	\$12.95	Securely mount camera on tripod
Leveling Feet	McMaster	4 x \$6.99	Make sure robot is level, strong enough to support robot (220kg)
16.4 feet of LED cabling	Flykul	\$16.99	Illuminate workspace to improve computer vision
2x RGB 80mm Multi-color COB	Yinatech	\$21.99	Illuminate workspace to improve computer vision

Table 26 continued from previous page

850 Evo, 250GB SSD	Samsung	$91.98	Improve speed and efficiency of computer
PCI-E to USB 3.0 4 Port	Mailiya	$50.99	Support all USB cameras at once with one machine
OS32C safety sensor	Omron	$141.39	Ensure a safe work environment for human workers
Wii Remote	Nintendo	$48.00	Input device for teleoperation of the robot
Motion Plus Attachment	DTOL	$13.33	Add extra sensor for calculating positions
Bluetooth Receiver	Insignia	$18.69	Communicate Wii Remote with workstation computer
TOTAL		$10,701.40	

Table 26: Budget and Expenditure Justification

O Motion Planner Evaluation Results

The following appendix contains the results of the motion planner evaluation. Planners evaluated are RRT, RRT*, PRM, and LBKPIECE. Planning and execution times are measured in seconds. Obstacle distances and EEF Travel distances are measured in millimeters, and joint travel is measured in radians. A run is failed if the value is outside one sigma.

Run#	1	2	3	4	5	Avg	SD	Avg+SD	Avg-SD
Planning Time	5.14	5.02	5.17	5.29	5.16	5.156	0.0960	5.252072	5.05992
Execution Time	11.92	12.98	12.09	11.76	11.84	12.118	0.4971	12.6151	11.6208
Obstacle Distance	1.9	2.2	2	2.9	2.8	2.36	0.4615	2.8215	1.8984
EEF Travel Distance	80.9	82.8	78.2	78.9	83.1	80.78	2.2174	82.9974	78.562
Joint Travel	0.21	0.36	0.17	0.23	0.41	0.276	0.1033	0.37934	0.17265
Conclusion	Pass	Fail	Pass	Fail	Pass				

Table 27: RRT Evaluation Results

Run#	1	2	3	4	5	Avg	SD	Avg+SD	Avg-SD
Planning Time	5.26	5.18	5.39	5.17	5.14	5.228	0.1008	5.3288	5.1271
Execution Time	11.16	11.52	11.97	11.04	11.84	11.506	0.4074	11.91	11.0985
Obstacle Distance	3.2	4.8	5.1	3.8	4.2	4.22	0.7628	4.9828	3.45711
EEF Travel Distance	81.1	87	79.2	82.3	83.1	82.54	2.8936	85.433	79.6463

Table 28 continued from previous page

Joint Travel	0.12	0.34	0.21	0.27	0.32	0.252	0.0892	0.3412	0.16272
Conclusion	Pass	Pass	Fail	Pass	Pass				

Table 28: RRT* Evaluation Results

Run#	1	2	3	4	5	Avg	SD	Avg+SD	Avg-SD
Planning Time	5.01	5.24	5.17	5.35	5.02	5.158	0.1454	5.30349	5.01250
Execution Time	11.09	11.26	11.06	11.05	11.31	11.154	0.1217	11.2757	11.0322
Obstacle Distance	1.2	1.9	1.6	2.1	1.8	1.72	0.3420	2.0620	1.37794
EEF Travel Distance	86	81.2	81.7	82.2	84.9	83.2	2.1201	85.320	81.0798
Joint Travel	0.33	0.23	0.21	0.28	0.31	0.272	0.0511	0.32318	0.22081
Conclusion	Fail	Pass	Pass	Fail	Pass				

Table 29: PRM Evaluation Results

Run#	1	2	3	4	5	Avg	SD	Avg+SD	Avg-SD

Table 30 continued from previous page

Planning Time	5.01	5.15	5.11	5.04	5.14	5.09	0.0620	5.15204	5.02795
Execution Time	10.95	11.08	11.11	10.98	11.05	11.034	0.0673	11.1013	10.9666
Obstacle Distance	3.5	4.1	3.7	3.8	3.9	3.8	0.2236	4.02360	3.57639
EEF Travel Distance	79.8	81.2	83.3	80.1	83.1	81.5	1.6385	83.1385	79.8614
Joint Travel	0.12	0.29	0.31	0.19	0.31	0.244	0.0853	0.32932	0.15867
Conclusion	Fail	Pass	Pass	Pass	Pass				

Table 30: LBKPIECE Evaluation Results

P Gap Analysis

The gap analysis investigates the current state of the technology used in this project and compares it to the features desired for this project. For gaps in the system, we will analyze the desired feature and compare that to the current solution, or lack thereof. Desired features come from Sections 3 and 6. The state of the art technology defined in this case is the best option available, regardless of the price or difficulty of implementation. The implemented feature is the version of the desired feature that is currently used in the project, if there is a solution. The risks column identifies and discusses potential drawbacks with the implementation of the best technology available.

Desired Feature	Current State of the Art	Implemented Feature	Risks of State of the Art
Sewing lines and curves in 2D	Automatic 2D sewing with planar machine	Calculate many waypoints on path for small motions	Expensive
Intuitive UI	UI with usability and product features	UI with few buttons and features	Takes time to develop
3D sewing	None exists	Not implemented	Low ability to execute paths
Fabric gripping	Human-like finger grippers to pick up fabric from flat surface	2-Finger Robotiq grippers	Custom gripper might not be as reliable or durable
Learning by demonstration	Use motion capture system to record seamster sewing; robot uses machine learning to sew from seamster data	Seamster can specify path with Geomagic Touch haptic device	Robot would sew identically to human - no machine advantage

Table 31 continued from previous page

Interface with sewing machine	Use signal processing to control sewing machine speed through ROS	Not implemented	Not thoroughly tested
Robust CV system	3D cameras with high frame rates and resolution; environmental factors (lighting, etc.) do not affect system	CV system with four cameras and max bandwidth	Expensive - \$250+
Robot replicates input device path in realtime	Robot executes path at same time as user defines path with input device	When user clicks button on UI, robot sews path specified with haptic device after optimization	System cannot plan paths instantaneously
Robot sews pattern from input device	None exists	Robot sews path from Geomagic Touch device	Not thoroughly tested
Robot sews pattern using CV	None exists	Robot sews predefined path using CV	Not thoroughly tested

Table 31 continued from previous page

Quickly plan optimal path with speed control	Group joints into groups of 6-DOF or less and use geometric planner	LBKPiece algorithm implemented; no joint speed control	Not able to plan trajectories for many DOF robots
Plot 3D sewing patterns	Augmented reality	Plot 2D sewing paths with a canvas in UI	Expensive: \$7,000 - \$14,000 to develop AR app using markers

Table 31: Gap Analysis

Q Authorship Table

Section	Primary Author(s)
Abstract	Alex Emrick
Project Statement	Sophia Gudenrath
Customer Value Proposition	Alex Emrick
Manufacturing with Sewing	Sophia Gudenrath
Computer Vision	Alex Emrick
Motion Planning	Alex Emrick and Sophia Gudenrath
Joint Actuation and Control	Sophia Gudenrath
Shared Autonomy	Alex Emrick

Table 32 continued from previous page

Stakeholders	Alex Emrick
Use Cases	Alex Emrick
Requirements	Alex Emrick
Workspace Layout	Elijah Eldredge
The Robot	Sophia Gudenrath
System Overview	Wyatt Henke
User Interface	Mary Hatfalvi
Computer Vision	Dominic Cupo
Motion Planning	Alex Emrick
Product Evaluation	Alex Emrick and Sophia Gudenrath
Project Execution Evaluation	Sophia Gudenrath and Alex Emrick
Recommendations: Control Interface	Mary Hatfalvi and Alex Emrick
Recommendations: Computer Vision	Dominic Cupo and Alex Emrick
Recommendations: Motion Planning	Sophia Gudenrath and Alex Emrick
Overall System Improvements	Alex Emrick
Appendices	Alex Emrick and Sophia Gudenrath

Table 32: Authorship

References

[1] D. Heitner, "North american sports market at 75.7 billion by 2020, led by media rights," Oct 10, 2016. [Online]. Available: https://www.forbes.com/sites/darrenheitner/2016/

10/10/north-american-sports-market-to-reach-75-7-billion-by-2020/
#6fe22d38217b

[2] M. Groover, "automation - advantages and disadvantages of automation," June 27, 2017. [Online]. Available: https://www.britannica.com/technology/automation/Advantages-and-disadvantages-of-automation

[3] F. Berruti, G. Nixon, G. Taglioni, and R. Whiteman, "Intelligent process automation: The engine at the core of the next-generation operating model." [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/

[4] S. Hayati and S. T. Venkataraman, "Design and implementation of a robot control system with traded and shared control capability," 1989, p. 1315 vol.3. [Online]. Available: http://ieeexplore.ieee.org/document/100161

[5] A. Mody and D. Wheeler, *Automation and world competition*, 1st ed. Basingstoke u.a: Macmillan, 1990.

[6] W. Duggan, "How the athletic footwear market has changed in recent years," -01-31 2017. [Online]. Available: https://www.benzinga.com/news/earnings/17/01/8972585/how-the-athletic-footwear-market-has-changed-in-recent-years

[7] BASIC, "Global athletic footwear market share (nike, adidas, etc.)," 2017. [Online]. Available: https://www.statista.com.ezproxy.wpi.edu/statistics/246501/athletic-apparel-companies-ranked-by-global-market-share-in-footwear-sales/

[8] J. Rock, "How robots will reshape the u.s. economy," Mar 21, 2016. [Online]. Available: http://social.techcrunch.com/2016/03/21/how-robots-will-reshape-the-us-economy/

[9] D. M. West, "What happens if robots take the jobs? the impact of emerging technologies on employment and public policy," October 2015. [Online]. Available: https://www.brookings.edu/wp-content/uploads/2016/06/robotwork.pdf

[10] D. Rotman, “How technology is destroying jobs,” June 12, 2013. [Online]. Available: https://www.technologyreview.com/s/515926/how-technology-is-destroying-jobs/

[11] A. Peters, “This t-shirt sewing robot could radically shift the apparel industry,” -08-25 2017. [Online]. Available: https://www.fastcompany.com/40454692/this-t-shirt-sewing-robot-could-radically-shift-the-apparel-industry

[12] T. Gottschalk and G. Seliger, “Automated sewing of textiles with different contours,” *CIRP Annals*, vol. 45, no. 1, pp. 23–26, July 2, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850607630091

[13] K. P. Reddy, “The rise of robotic automation in the sewing industry,” p. 26, May 1, 2016. [Online]. Available: https://search.proquest.com/docview/1796771370

[14] D. Gershon, “Parallel process decomposition of a dynamic manipulation task: robotic sewing,” *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 357–367, 1990. [Online]. Available: http://ieeexplore.ieee.org/document/56654

[15] M. Kudoa, Y. Nasua, K. Mitobe, and B. Borovac, “Multi-arm robot control system for manipulation of flexible materials in sewing operation,” July 12, 1999. [Online]. Available: http://ac.els-cdn.com/S0957415899000471/1-s2.0-S0957415899000471-main.pdf?_tid=1061c73e-9c85-11e7-a73c-00000aacb362&acdnat=1505748278_571cd0d124a18a134f1ba3d574d22cf2

[16] R. Szeliski, *Computer vision.* London [u.a.]: Springer, 2010.

[17] K. Simek, “Dissecting the camera matrix, part 3: The intrinsic matrix,” August 13, 2013. [Online]. Available: https://ksimek.github.io/2013/08/13/intrinsic/

[18] R. Hartley and A. Zisserman, “Multiple view geometry in computer vision,” *Robotica*, vol. 23, no. 2, p. 271, 2005. [Online]. Available: https://www.cambridge.org/core/article/multiple-view-geometry

[19] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, 2nd ed. Pearson Education M.U.A, 2012. [Online]. Available: http://lib.myilibrary.com?ID=645682

[20] G. Medioni, *Emerging topics in computer vision*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 2005.

[21] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson Education, Jul 4, 2016. [Online]. Available: http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9781292153971&uid=none

[22] P. G. Ranky, "Advanced machine vision systems and application examples," *Sensor Review*, vol. 23, no. 3, pp. 242–245, Sep 1, 2003. [Online]. Available: http://www.emeraldinsight.com/doi/abs/10.1108/02602280310481869

[23] S. Kaur and I. Singh, "Comparison between edge detection techniques," *International Journal of Computer Applications*, vol. 145, no. 15, pp. 15–18, Jul 15, 2016. [Online]. Available: http://au4sb9ax7m.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info%3Asid%2Fsummon.serialssolutions.com&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.atitle=Comparison+between+Edge+Detection+Techniques&rft.jtitle=International+Journal+of+Computer+Applications&rft.au=Kaur%2C+Satbir&rft.au=Singh%2C+Ishpreet&rft.date=2016-07-15&rft.issn=0975-8887&rft.eissn=0975-8887&rft.volume=145&rft.issue=15&rft.spage=15&rft.epage=18&rft_id=info:doi/10.5120%2Fijca2016910867&rft.externalDBID=n%2Fa&rft.externalDocID=10_5120_ijca2016910867&paramdict=en-US

[24] R. Chatterjee, "Advanced color machine vision and applications," Mar 26, 2014. [Online]. Available: https://www.visiononline.org/userAssets/aiaUploads/file/CVP_Advance-Color-Machine-Vision-and-Applications_Romik-Chatterjee.pdf

[25] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, *Automatic generation and detection of highly reliable fiducial markers under occlusion*, 2014, vol. 47.

[26] "Aruco: a minimal library for augmented reality applications based on opencv." [Online]. Available: https://www.uco.es/investiga/grupos/ava/node/26

[27] "Opencv: Canny edge detection," Dec 18. 2015. [Online]. Available: https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

[28] M. A. Najjar, M. Ghantous, and M. Bayoumi, *Video Surveillance for Sensor Platforms : Algorithms and Architectures*, 1st ed. New York, NY: Springer, 2014, vol. 114. [Online]. Available: http://lib.myilibrary.com?ID=547507

[29] W. Burgard, C. Stachniss, M. Bennewitza, and K. Arras, "Introduction to mobile robotics: Robot motion planning," July 2011. [Online]. Available: http://ais.informatik.uni-freiburg.de/teaching/ss11/robotics/slides/18-robot-motion-planning.pdf

[30] K. C. Gupta, "On the nature of robot workspace," *The International Journal of Robotics Research*, vol. 5, no. 2, pp. 112–121, Jun 1986. [Online]. Available: http://journals.sagepub.com/doi/full/10.1177/027836498600500212

[31] M. R. de Gier, "Control of a robotics arm: Application to on-surface 3d-printing," April 6, 2015. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:a674a3fa-2534-44c4-b251-1e49a5194079/datastream/OBJ

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics.* London: Springer London, 2009.

[33] P. Abbeel, J. Ho, A. Lee, I. Awwal, H. Bradlow, and J. Schulman, "Finding locally optimal, collision-free trajectories with sequential convex optimization," Tech. Rep., 2012. [Online]. Available: http://joschu.net/docs/trajopt-paper.pdf

[34] J. Fan and R. Li, "Statistical challenges with high dimensionality: Feature selection in knowledge discovery," Feb 7, 2006. [Online]. Available: http://arxiv.org/abs/math/0602133

[35] A. S. Deo and I. D. Walker, "Minimum effort inverse kinematics for redundant manipulators," *IEEE Transactions on Robotics and*

Automation, vol. 13, no. 5, pp. 767–775, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/631238

[36] K. Gochev, V. Narayanan, B. Cohen, A. Safonova, and M. Likhachev, "Motion planning for robotic manipulators with independent wrist joints," May 2014, pp. 461–468.

[37] M. F. Ghajari and R. V. Mayorga, "Specialized prm trajectory planning for hyper-redundant robot manipulators," 2017. [Online]. Available: http://www.wseas.org/multimedia/journals/systems/2017/a605902-779.pdf

[38] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," 2010, pp. 2902–2908. [Online]. Available: http://ieeexplore.ieee.org/document/5509685

[39] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 2006. [Online]. Available: http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf

[40] P. Zips, M. Böck, and A. Kugi, "Fast optimization based motion planning and path-tracking control for car parking," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 86–91, January 1, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016316408

[41] R. Geraerts and M. H. Overmars, "Clearance based path optimization for motion planning," vol. 3. IEEE, 2004, p. 2392 Vol.3. [Online]. Available: http://ieeexplore.ieee.org/document/1307418

[42] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. Bagnell, and S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning¡br¿," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, Sept 13, 2013. [Online]. Available: http://journals.sagepub.com/doi/abs/10.1177/0278364913488805

[43] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," May 2011, pp. 4569–4574.

[44] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. [Online]. Available: http://ieeexplore.ieee.org/document/508439

[45] A. M. Bloch, *Nonholonomic Mechanics and Control*, 2nd ed. New York, NY: Springer New York, 2015, vol. 24.

[46] A. Masoud, "Kinodynamic motion planning," *IEEE Robotics and Automation Magazine*, vol. 17, no. 1, pp. 85–99, 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5430384

[47] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," May 3, 2010. [Online]. Available: http://arxiv.org/abs/1005.0416

[48] ——, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun 2011. [Online]. Available: http://journals.sagepub.com/doi/full/10.1177/0278364911406761

[49] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," vol. 2, 2000, p. 1001 vol.2.

[50] S. Jain and A. Chhabra, "Tracking path control of robotic manipulators using radial basis function neural network," *i-Manager's Journal on Instrumentation and Control Engineering*, vol. 2, no. 3, p. 11, May 1, 2014. [Online]. Available: https://search.proquest.com/docview/1693778622

[51] M. Likhachev, G. J. Gordon, and Thrun, "Ara*: formal analysis." [Online]. Available: http://repository.cmu.edu/compsci/2174

[52] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects." IEEE, 2013, pp. 816–823. [Online]. Available: http://ieeexplore.ieee.org/document/6696445

[53] S. Klanke, D. Lebedev, R. Haschke, J. Steil, and H. Ritter, "Dynamic path planning for a 7-dof robot arm." IEEE, 2006, pp. 3879–3884. [Online]. Available: http://ieeexplore.ieee.org/document/4059012

[54] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," May 2009, pp. 489–494.

[55] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, Aug 2014. [Online]. Available: http://journals.sagepub.com/doi/full/10.1177/0278364914528132

[56] C. Gentile and M. Warmuth, "Linear hinge loss and average margin." [Online]. Available: http://papers.nips.cc/paper/1610-linear-hinge-loss-and-average-margin.pdf

[57] D. Coleman, "rviz - ros wiki," June 15, 2016. [Online]. Available: http://wiki.ros.org/rviz

[58] "Gazebo," 2014. [Online]. Available: http://gazebosim.org/

[59] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," July 2008. [Online]. Available: https://www.ri.cmu.edu/pub_files/pub4/diankov_rosen_2008_2/diankov_rosen_2008_2.pdf

[60] K. Hauser, "Klamp't manual v0.7," March 30, 2017. [Online]. Available: http://motion.pratt.duke.edu/klampt/KlamptManualv0.7.pdf

[61] C. Zhou, "Isye 4256 supplemental material," Fall, 1999. [Online]. Available: http://www2.isye.gatech.edu/~czhou/MotionTypes.pdf

[62] "What's the difference between rrt and rrt* and which one should we use." [Online]. Available: https://www.youtube.com/watch?v=JeEk_CWcRFI

[63] S. V. Shah, S. K. Saha, and J. K. Dutt, "Denavit-hartenberg parameterization of euler angles," *Journal of Computational and Nonlinear Dynamics*, vol. 7, no. 2, p. 21006, 2012.

[64] S. A. Nugroho, A. S. Prihatmanto, and A. S. Rohman, "Design and implementation of kinematics model and trajectory planning for nao humanoid robot in a tic-tac-toe board game," vol. 4. IEEE, 2014, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7111783

[65] A. Aristidou and J. Lasenby, "Fabrik: A fast, iterative solver for the inverse kinematics problem," *Graphical Models*, vol. 73, no. 5, pp. 243–260, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1524070311000178

[66] J. Zhao and N. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 4, pp. 313–336, Oct 1, 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=195827

[67] J. Karpinska, K. Tchon, and M. Janiak, "Approximation of jacobian inverse kinematics algorithms: Differential geometric vs. variational approach," *Journal of Intelligent and Robotic Systems*, vol. 68, no. 3-4, pp. 211–224, Dec 1, 2012. [Online]. Available: https://search.proquest.com/docview/1113738401

[68] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," Mar 26, 2015. [Online]. Available: http://arxiv.org/abs/1503.07619

[69] R. Gordon, "Teleoperating robots with virtual reality," October 11, 2017. [Online]. Available: http://news.mit.edu/2017/mit-csail-new-system-teleoperating-robots-virtual-reality-1009

[70] K. Kinugawa and H. Noborio, "A shared autonomy of multiple mobile robots in teleoperation." IEEE, 2001, pp. 319–325. [Online]. Available: http://ieeexplore.ieee.org/document/981922

[71] S. Jain, A. Farshchiansadegh, A. Broad, F. Abdollahi, F. Mussa-Ivaldi, and B. Argall, "Assistive robotic manipulation through shared autonomy and a body-machine interface," vol. 2015. IEEE, 2015, pp. 526–531. [Online]. Available: http://ieeexplore.ieee.org/document/7281253

[72] W. S. Kim, B. Hannaford, and A. K. Fejczy, "Force-reflection and shared compliant control in operating telemanipulators with time delay," *IEEE*

Transactions on Robotics and Automation, vol. 8, no. 2, pp. 176–185, 1992. [Online]. Available: http://ieeexplore.ieee.org/document/134272

[73] P. Birkenkampf, D. Leidner, and C. Borst, "A knowledge-driven shared autonomy human-robot interface for tablet computers." IEEE, 2014, pp. 152–159. [Online]. Available: http://ieeexplore.ieee.org/document/7041352

[74] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, vol. 35, no. 4, pp. 241–254, Nov 2013. [Online]. Available: https://search.proquest.com/docview/1437177065

[75] "Robot control devices." [Online]. Available: http://www.trossenrobotics.com/c/robot-control-devices.aspx

[76] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," Apr 20, 2017. [Online]. Available: http://arxiv.org/abs/1704.06053

[77] I. Muttschall, "Wii remote accuracy," Tech. Rep., 12/17/ 2009. [Online]. Available: http://vigir.ee.missouri.edu/~gdesouza/ece4220/Projects/F2009/Isaac%20Muttschall/Embedded%20Final%20Report.pdf

[78] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics." IEEE, 2015, pp. 928–935. [Online]. Available: http://ieeexplore.ieee.org/document/7363472