Intelligent Robotic Recycling of Flat Panel Displays

by

Adam Garry Sanderson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2019

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

As the population and prosperity continue to rise the demand for high-tech products is rapidly increasing. Displays are a huge part of this market with millions being created each year. We have a finite amount of resources on this planet and it is imperative that we properly dispose of our devices.

In this thesis an intelligent robotic workcell for dismantling and recycling of flat panel displays is proposed and tested. This system utilizes industrial robots and open source tools to process a flat panel display (FPD) with the goal of removal of cold compact fluorescent tubes (CFLs). These are removed so the FPD can be shredded and properly recycled without the mercury present in the CFLs contaminating the materials or harming workers.

This system utilizes many new and innovative techniques including deep learning algorithms for object detection and image segmentation. These deep learning techniques specifically Faster R-CNN for object detection and DeepLab for image segmentation have been shown to be extremely robust and capable in this application.

The system was tested both with a real world system and with simulated geometry with a live vision feed. During simulation, the system was capable of processing flat panel FPDs in between 110 and 230 seconds per unit. It has been calculated using the simulation results, that this system can be profitable and has an approximate payback period of between 0.19 and 4.87 years depending on the material being fed into the system. The large range is due to the difference in value between monitor and TV style FPDs. The monitors have far less value than the TVs and are far more difficult to process.

Acknowledgements

First, I would like to thank everyone who has supported me on the research and academic side. Thank you to my supervisors Dr. Hamid Karbasi and Dr. Amir Khajepour for putting their faith in a me and supporting me both in my studies and career. Both providing great advice during my two years working on this project.

My coworkers at the waste electrical and electronic equipment (WEEE) lab Cristian Pop, Dr. Alireza Sharifi, and Dr. Ali Tehrani. Who have always been around to bounce ideas off of and talk about projects and work. Especially Cristian Pop who has been a great confidant and friend in my time at the lab. I would also like to thank Conestoga College for supporting the lab, my research and myself.

My friends who have been a great distraction in the times that I needed to get away and enjoy myself. Specifically, Gabriel DeSousa, Brandon Cole, and Conrad Chow who are (mostly) good influences.

I would like to thank my family, my parents Dean and Susan, who without their support none of this would be possible. They have supported me in all aspects from housing and advice to proof reading my thesis (thanks for that Mom). I would also like to thank my brother Aaron, not only because he said I had to but because he deserves it.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

In recent decades, consumer electronics have become an integral part of daily life, revolutionizing the way we communicate, retrieve information, and entertain ourselves. With all of these devices being created it is important to manage how they are disposed. Waste management and resource recovery for electrical and electronic equipment such as computing,displaying,s and mobile telecommunications devices includes waste stream sorting, chemical separation and treatment, decontamination, and waste logistics. Another key segment includes material recovery: metal recovery (e.g. gold, silver, and platinum) and plastic recycling contribute to both economic and environmental sustainability.

Waste Electrical and Electronic Equipment (WEEE) is the fastest growing sector of solid waste, with 40 to 50 million tons generated globally each year. Only 15-20% of WEEE is recycled; the rest ends up in landfills, riverbanks and deserts, or is exported to Third World countries where it is incinerated to liberate precious metals. In the U.S. alone, 2.2 million tons of electronic waste (e-waste) is disposed annually (Environmental Protection Agency). Ontario, Canada's largest e-waste producing province, recycled 71,018 tons in 2014 (more than 50% of total e-waste recycled in Canada) according to the Ontario Electronic Stewardship and Electronics Product Recycling Association [1, 2, 3, 4].

Automation and specialization among small and medium size enterprise (SME) e-waste recyclers is very limited, resulting in lost opportunities to improve productivity and business sustainability. For SMEs, proper e-waste recycling can be challenging as hazardous and toxic substances must be extracted and disposed of safely before any valuable materials can be recovered. The recycling process is time-consuming and labor-intensive, especially when dealing with high volumes of mixed materials.

E-waste is far too valuable of a commodity and should be treated more responsibly. For example, a ton of circuit boards is estimated to contain 40-800 times the gold of a ton of gold ore and 30-40 times more copper than copper ore [1]. It is also less energy intensive to recycle than it is to mine

which further increases the potential profit from recycling. Recycling can bring billions of dollars' worth of materials back into the market that would otherwise be buried in landfills and polluting the environment.

Recyclers process electronic waste (i.e. e-waste) in order to recover precious metals and valuable plastics that can be reused to manufacture new products. To remain sustainable within the competitive recycling marketplace, many recyclers must constantly improve their existing recycling processes and look for new processes in order to expand into new high-demand areas. Each process for recycling e-waste must be safe, fast, accurate, agile, and cost-effective. Robotic automation can return its initial investment in a short period of time by increasing the productivity and reducing labor costs. This project is aimed to improve the capability of Canadian recyclers to process Flat Panel Displays (FPDs). This, in turn, will result in an increase in revenue generation, and assist with developing long-term sustainability.

Flat panel televisions are a continually growing market with hundreds of millions of sales worldwide each year. In today's ever-evolving technology market, the life of our products is getting shorter which means that the rate at which these products are being recycled or thrown away is increasing. This creates a great opportunity for many recyclers but also a large problem. Figure 1 shows the worldwide TV market by technology. There will be a great economic and environmental advantage if innovative and effective recycling solutions become available to recyclers [5]. However, all the current processes recycling solutions require manual intervention to remove high value or dangerous components which makes recycling less attractive as a healthy business practice.

The purpose of this project is to solve these problems via the creation of a fully automated flat panel display recycling cell utilizing robotics. In order to accomplish this and remain economical different tooling, vision, and artificial intelligence (AI) techniques will need to be utilized.

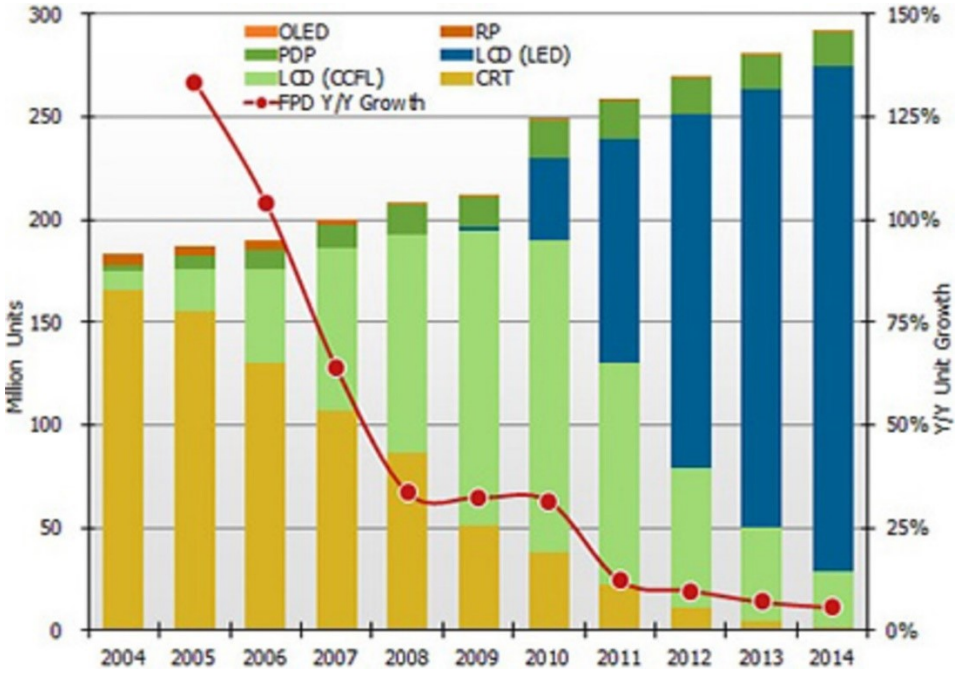


Figure 1 - A Breakdown of TV Units by Technology [5]

1.2 Challenges

The challenge within recycling (or de-manufacturing) is the dynamic range of products that need to be recycled. For example, one manufacturer of FPD might have a dozen unique products. In this project FPDs will refer to monitors and televisions 16" and larger, that do not use cathode ray tube or rear projection technologies and are not an "all-in-one" computer system or laptop computer. FPDs can use many different types of technologies; we are most concerned with FPDs using cold compact florescent tubes (CFLs). However, it is rarely apparent if the FPD contains CFLs before they are opened. A recycler needs to be able to process not only the dozen units from one manufacturer, but ideally all the manufacturers within the specific marketplace. In the case of FPDs, a recycler could be looking at hundreds of varieties of products. As such, each recycling process must be defined precisely to handle the complex array of parameters in order to make a system effective.

The variety of monitors is not the only challenge with the application. Nearly all the FPDs that come into a recycling facility are in various states of disrepair. This damage not only causes difficulty with fixturing and processing but also with part identification. Modifications to the FPDs are also not

limited to damage as some units can have seals and markings removed or even be painted. Figure 2 shows examples of damage and modifications done to FPDs.

Figure 2 - FPD States from Recycling Samples

1.3 Literature Review

Automation in e-waste processing is a very under explored area of research. There are not many academic papers covering this application most of the work done in this field has been by equipment manufacturers. Equipment manufactures do not tend to release papers outlining their process so marketing materials must be used to determine what they systems do and what type of sensors are being utilized. This section is broken down into three areas; current recycling practices for FPDs, FPD recycling systems that are available on the market, vision systems for recycling, and previous work done at Conestoga College to create a system to recycle FPDs.

1.3.1 Current Recycling Practice

One of the most prevalent ways to process used electronics in the e-waste recycling industry is to use large shredders to break the electronics into small pieces. These pieces are then sorted into

material types including circuit boards, plastics, ferrous metals, and non-ferrous metals. This practice is not possible with FPDs as, until the recent popularity of LED TVs, the most common way to backlight a TV was via the use of CFLs. These tubes contain mercury vapor which is used to create ultraviolet light which is then converted into visible light via a phosphor powder on the inside of the tubes. When an FPD containing CFLs is put into a shredder the mercury content contaminates the material. The material then either needs to be decontaminated or landfilled.

The current accepted practice of FPD recycling in industry is either a complete manual dismantling process or manual dismantling to remove CFLs followed by shredding of the other components. In order to process an FPD a worker requires a station and is typically given a pneumatic screw driver and dismantles the FPD one screw at a time working their way through all of the layers of the FPD.

This process is both time consuming and potentially dangerous for a laborer. There are significant ergonomic issues with lifting large FPDs onto a work table and the operator leaning over to remove each screw. There is also a risk of the laborer being exposed to toxins from the FPDs such as lead solder, biological contaminants, and mercury vapor from CFL tubes. The amount released per bulb is relatively small in typical conditions it takes several days for the mercury to vaporize [6]. However, repeated exposure can cause adverse health effects and requires recyclers to monitor the health of their laborers.

Manual dismantling is also an inherently slow process. As the FPD screws could have variable drive types and sizes the operator may require multiple bit changes. Every FPD that comes in will also be slightly different increasing the learning curve for laborers. Manual dismantling for a large FPD can take upwards of 11 minutes if the laborer is just removing the CFLs and upwards of 20 for full disassembly.

1.3.2 FPD Recycling Systems

There are some FPD recycling systems and plants that exist currently but many of their practices are far too large and expensive for medium to small scale recycling companies. Many of these systems are also semi-automated systems in which employees are required to do some of the

dismantling themselves. As discussed previously, this exposes the workers to health and safety hazards which should be avoided if possible.

An example of a large scale FPD recycling system is the FPD processing system developed by BluBox [7]. This company does not use a robotic dismantling approach but rather shreds the FPD into pieces and decontaminates the pieces. This approach also includes density, eddy current, magnetic, and color sorters which create several output streams. These streams include circuit boards, aluminum, ferrous materials, screen components, and plastics. By using this approach, the system can process up to 1000kg of displays per hour. The major downside to this approach is the cost of the system and the floorspace required to run it. The system is over two stories high and costs several million dollars to purchase. Only the largest e-waste recyclers can justify a system of this size.

There are also companies that have developed smaller semi-automated solutions to this problem. One of these companies is Erdwich. Erdwich created a semi-automated FPD dismantling system that removed the screen to the FPD then requires an operator to remove the CFLs [8]. This system utilizes a six-axis industrial robot and a custom fixture and loading conveyor. To detect where to cut the system utilizes a camera with a laser projector. The laser projector projects two lines in a cross, which is used to help find the edge of the screen. The camera is then used to determine the location of this edge. This is done in two opposite corners of the screen to find the locations to cut. The weakness of this approach is with damaged TVs where the bezel has been removed. Though it does increase processing speed but? the system does not remove the operator from the process. The operator will still be exposed to the mercury bulbs. The system also only works for TV style FPDs.

Erdwich also makes a system for processing monitors. This system is not robotic it is simply a fixture with circular saws that cuts off the top and bottom edges of the monitor. This is essentially a table saw with a guide on the side that spaces the blade a specific distance away from the edge. This is a static distance and is does not change based on the monitor. There is no vision used in this system.

There is also a large scale FPD processing site in UK called Veolia [9]. This site is a massive construction that utilizes a number of large 6 axis robots but it is still a semi-automated process. To remove the bezel a large 6 axis robot moves an FPD into a custom jig which uses 4 wedges to go under the lip of the bezel and break it. Once the bezel is removed workers then snap off any pieces of the bezel that remain and remove the screens by hand. Once that is done the FPD then is sent to a robot

which uses a laser line scanner across the middle of the screen to find the location of the bulbs. This location is used to put glue on the bulbs to attach it to the backing of the screen. Once this is done a worker than removes the bulbs manually.

1.3.3 Vision Systems for E-waste Applications

The purpose of this section is to review vision system applications in e-waste. This is one of the most significant areas of research in e-waste recycling. These systems tend to be used for material identification for bulk sorting and are relevant to this project as one of the key problems to be solved is the identification of parts and component types of the FPD being processed.

The most common form of machine vision in e-waste applications is color sorters. These devices use either color line scan cameras or color sensors to detect the color of components. This is done as circuit boards tend to be green. These systems then use air-jets or other actuators to remove these items from the material stream. There are issues with these systems, for instance if there are green plastics or other items in the material stream then they are commonly misidentified as circuit boards. They often also remove green capacitors. There has been work in utilizing NIR for color sorters to sort metals and plastics. [10]

There has been some work in academia for detecting circuit boards and electronics for recycling. These systems mainly focus on materials that have already been dismantled or shredded. An example of this is work that has been done in the WEEE lab at Conestoga College [11]. I was involved in this during my time in the lab. This system utilizes an object detection network, specifically Faster R-CNN to detect and classify shredded electronics on a conveyor belt and utilizes a delta robot to remove contaminants from a material stream. The system requires the components to be liberated and does not attempt to find the boundary of the object just a bounding box. Similar imaging processing techniques were utilized in this project.

Other systems focus on detection of components on a circuit board [12] or detecting the value of removed circuit boards [13]. There is very little work done on applying deep learning algorithms and especially image segmentation networks in the recycling industry. This is a field in which these new deep learning algorithms could be incredibly valuable due to the variety of inputs material states and types. All these vision systems use color cameras to detect and classify the components.

1.3.4 Previous Work

This project is the second phase of ongoing research to create a robotic flat panel display dismantling system. The previous project focused on a creating a semi-automated approach that would allow the operator quickly and easily access to the FPDs to remove circuit boards and CFLs. This work was done under the supervision of Dr. Hamid Karbasi in the WEEE lab, Conestoga College. The students from Conestoga College including the author did mechanical and electrical design and programming.

To accomplish this task the system uses a larger circular saw and performs three cuts at specific locations which avoid CFLs. These locations are below and above the bezel on the top and bottom respectively, and near the edge of the FPD on the short side. All cuts are performed from one end of the FPD to the other and are done at almost full depth. Figure 5 shows this system performing a cut.

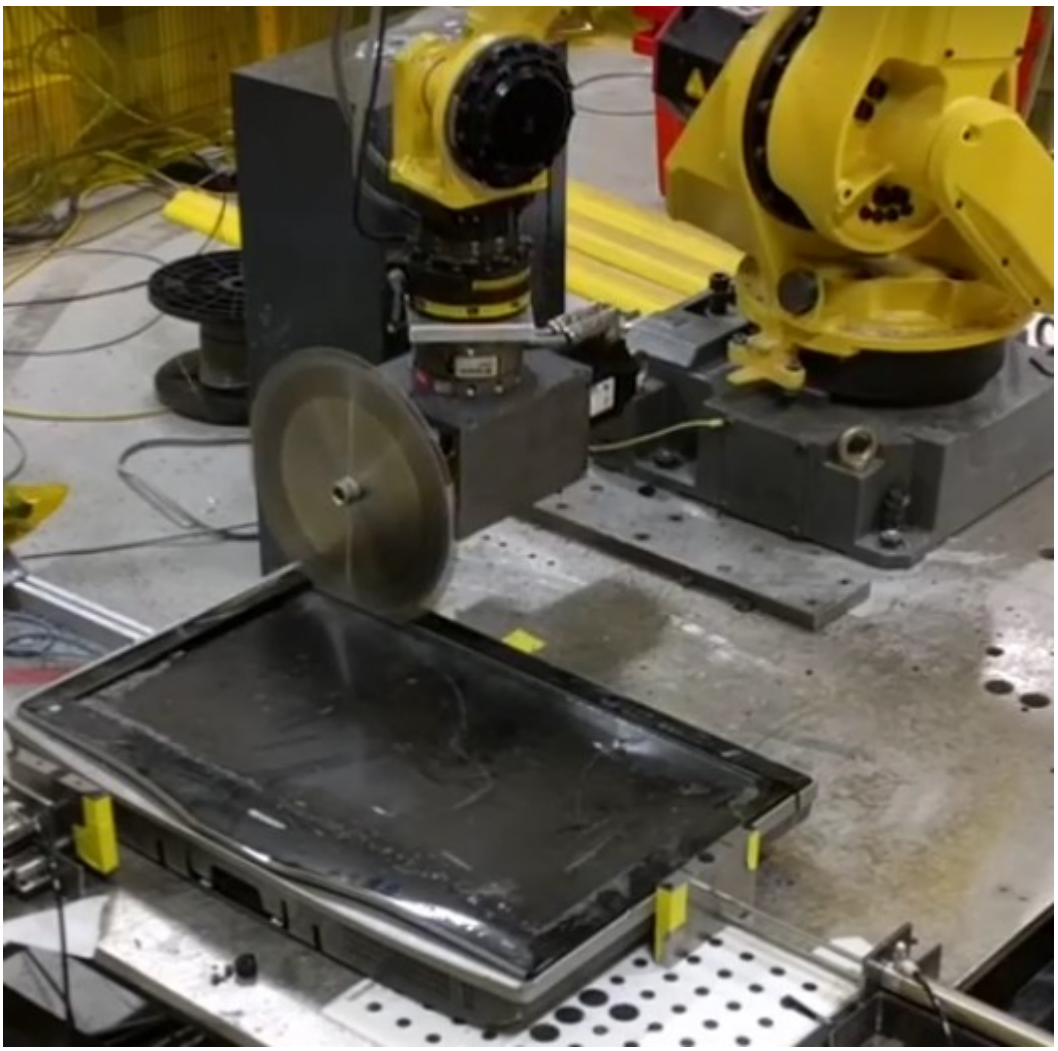


Figure 3 - Previous FPD Recycling Project

The cut locations are found using a computer-based vision specifically VisionPro [14] which is developed by Cognex. There are two cameras in the system both with a resolution of one megapixel mounted above and to the side of the fixture. The overhead camera is used to find the corners of the screen of the FPD which is used to determine the cut locations. This is done by converting the image into grayscale and performing line detection from the center of the FPD. These lines are then intersected to find the corners of the screen. The side camera is used to find the height of the FPD. Corner values are then modified using the height. This is done as the detection is transformed to real work co-ordinates

using a calibration grid on the surface of the fixture. Since the FPD screens are above the surface of the FPD perspective will make them look larger. The height is also used to find an approach value which allows the robot to move rapidly to the surface of the FPD reducing cycle time.

In order to control the Robot Fanuc's PC developer's toolkit (PCDK) was used. PCDK allows for access to the Fanuc robot controller over Ethernet. It can read and write system variables, monitor alarms, and start programs [15]. The PCDK and VisionPro were integrated into a single Visual Basic (VB) script which runs the VisionPro code to get the location of the corners of the screen, these values are then sent to the robot and used to modify a standard processing code. The code performs the steps shown in Algorithm 1. This system can process an FPD in approximately 2 minutes.

Despite these successes there were considerable issues that need to be addressed in this solution. One of the key issues was the vision system. The line detection process worked approximately 70% of the time and would take between 3 and 10 seconds to find the boundaries of the FPD's screen. The points provided were also inaccurate which is caused by the line detection and intercept finding process. If the found line angles are off by a couple of degrees, this can lead to several centimeters of error. Some mechanical issues were also present in this design, the noise levels from the saw were significant and significant sparks are generated during the cutting process.

The project completed in December of 2016.

Algorithm 1 - Previous System Cutting Steps

Turn on saw
Clamp FPD
Approach top left corner
Plunge cutter
Cut from top left to top right
Pull out of cut
Approach top left corner
Plunge cutter
Cut from top left to top right
Pull out of cut
Approach top left corner
Plunge cutter
Cut from top left to top right
Pull out of cut
Unclamp FPD

1.4 Typical FPD Design

Prior to the onset of the FPDs that utilized LEDs as their light source, all FPDs utilized cold compact fluorescent tubes as the source of the FPDs backlight. These CFL backlights are the reason why the FPDs need to be dismantled. The mercury content inside the CFLs contaminate the material if a shredder is used. As only some of these contain CFLs it is important to be able to detect which units contain them, but this is not always possible. FPDs and most screens tend to have some similarities. They are rectangular with a screen with standard aspect ratios of either 4 by 3, 16 by 9, and recently 21 by 9.

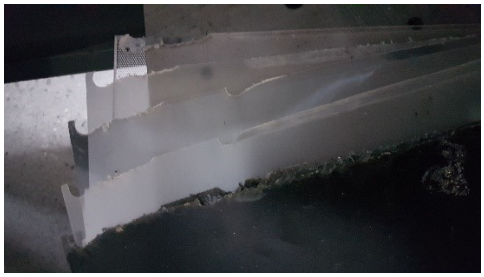
FPDs typically fall into two different design type groups, which we will define as Monitors and TVs. These two groups have many different FPD designs within them but have some consistent design features and paradigms. These features and paradigms act as the basis of processes in the proposed FPD dismantling system. Processing parameters also heavily rely on the construction of the unit as monitors and TVs require different materials to be cut. This section will cover the typical construction of FPDs and how this can help drive the dismantling process.

1.4.1 Monitor

The “monitor” construction style is typically a smaller size FPD. These FPDs tend to be less than 35” in size and thinner than TVs. These FPDs are made up of a screen assembly, which is encased by a plastic shell. This shell is mainly in place to hold buttons and speakers. It is usually made of ABS. The inner screen assembly is very tightly packed containing a stainless steel or aluminum outer shell, which holds the assembly together. Inside this shell the monitor contains an acrylic panel, several filters, the LCD panel and either CFL or LED lighting. The lighting is placed on the top and bottom of the screen surrounded by a reflective covering, which channels the light into the side of the acrylic panel. This panel is used to refract the light from the sources to the back of the filters and LCD panel toward the user. This technique is the reason why many older monitors had were significantly brighter towards the top and bottom of the screen.

Figure 3 shows a breakdown of a monitor.

(A)



(B)

(C)

(D)

Figure 4 - Breakdown of Monitor Style FPD. (A) Bezel removed showing screen package. (B) Filters and Acrylic (C) Screen package with Acrylic removed (D) Back of monitor with screen assembly removed

1.4.2 TV

The “TV” construction style is typically a larger thicker style of FPD containing a large amount of free space and many more light sources than a monitor. A TV is typically upwards of 35” in size with a large stainless steel or aluminum backbone/core. This backbone is typically a formed piece in which every other piece of the system connects. Starting from the screen of the system, back the construction typically goes as follows. A bezel connects to the backbone piece and is used to hold the filters and LCD panel. This panel is free floating and behind it is an air gap until the lighting assembly. The lighting assembly is an array of either LEDs or CFLs which provide the backlight for the TV. This lighting assembly also connects to the backbone piece of the TV. The next piece is the backbone which separates the screen from the circuit boards and control side of the system. The circuit boards are then bolted to the backbone and a plastic covering is the final component. Speakers can be on either the bottom or side of the screen in front of the backbone piece or installed behind it with the circuit boards.

(A)

(B)

(C)

Figure 5 - Breakdown of TV Style FPD. (A) Front with screen and bezel removed. (B) Back with plastic removed. (C) Filter and screen stack

1.5 Thesis Objective

In this project, the focus is on creating new techniques to find and remove key components to disassemble the FPD in the safest, fastest, and most cost-effective method possible while using minimally destructive techniques. To achieve this goal, an automated robotic system is developed that can load, process, and unload an FPD.

The objective of this project is to cut through low value plastics and metals and carefully remove high value or hazardous circuit boards and backlight mercury tubes layer by layer. In order to achieve this goal, different sensor, tool and fixture types and ideas are examined. The sensor types included robot mounted cameras and 3D sensors including laser displacement, time of flight, and structured light sensors. Tooling studies are focused on destructive tools such as saws, waterjet cutting, and mills, and nondestructive tools such as parallel, angular, fin, sand (coffee bean) grippers as well as suction cups.

These new techniques will not be useful unless it is known when and how to use them. To compensate for the variability in the input stream, this project requires significant investment in the creation of an intelligent control system. This system will use machine learning to interpret the data available and determine whether previous actions are successful. As the value of each FPD is small, the system needs to be protected against any collisions or anything that could cause damage to expensive tooling and actuators.

1.6 Contributions of other team members

This project was done with the help of team members at the waste electrical and electronic equipment (WEEE) lab at Conestoga College. Several? undergraduate students were vital in completing the work discussed. These students are Ivan Terziev and Nathan Nequest. Both of these students assisted with mechanical design.

Ivan assisted with designing a gantry version of the system tested, the fixture, and revising the end of arm tooling after testing. He was also responsible for gathering the images used to create the neural network. He also did the preliminary labelling of the dataset. The dataset labels were later refined and updated.

Nathan Nequest was responsible for assisting with preliminary end of arm tooling and fixture designs. Ivan and I later updated these models.

1.7 Thesis Organization

This thesis consists of six chapters including the introduction. The details of the proposed system are described in Chapter 2. This covers the basics of the control architecture as well as the mechanical, pneumatic, and electrical design. It also covers the safety considerations for the cell. The process used to dismantle the FPDs is discussed in Chapter 3. The vision and considered depth sensors are covered in Chapter 4. The vision algorithms specifically deep learning and 3D sensors are the core of this system. Chapter 5 covers the results of testing, how errors found can be addressed. Finally, conclusions and areas for future work are given in Chapter 6.

Chapter 2
Proposed System

The goal of this system is to create a flexible workcell to assist recycling FPDs. This provides a unique challenge due to the myriad of FPD models available on the market. Another issue is that as these models are in a recycling stream, they are likely either damaged or end of life. These difficulties make processing the FPDs via recipes or by model next to impossible. Due to this, the proposed system is vision driven and utilizes an articulated robot and custom fixture to create and determine paths on the fly. The main processing is done on a PC, which utilizes ROS and ROS-Industrial packages to control the system. For tooling a mill and two different grippers were chosen.

2.1 System Overview

The following Figure shows an overview of the components of the system. Everything in this system is controlled by the main control system/ubuntu pc. The other components are all external actuators and effectors.

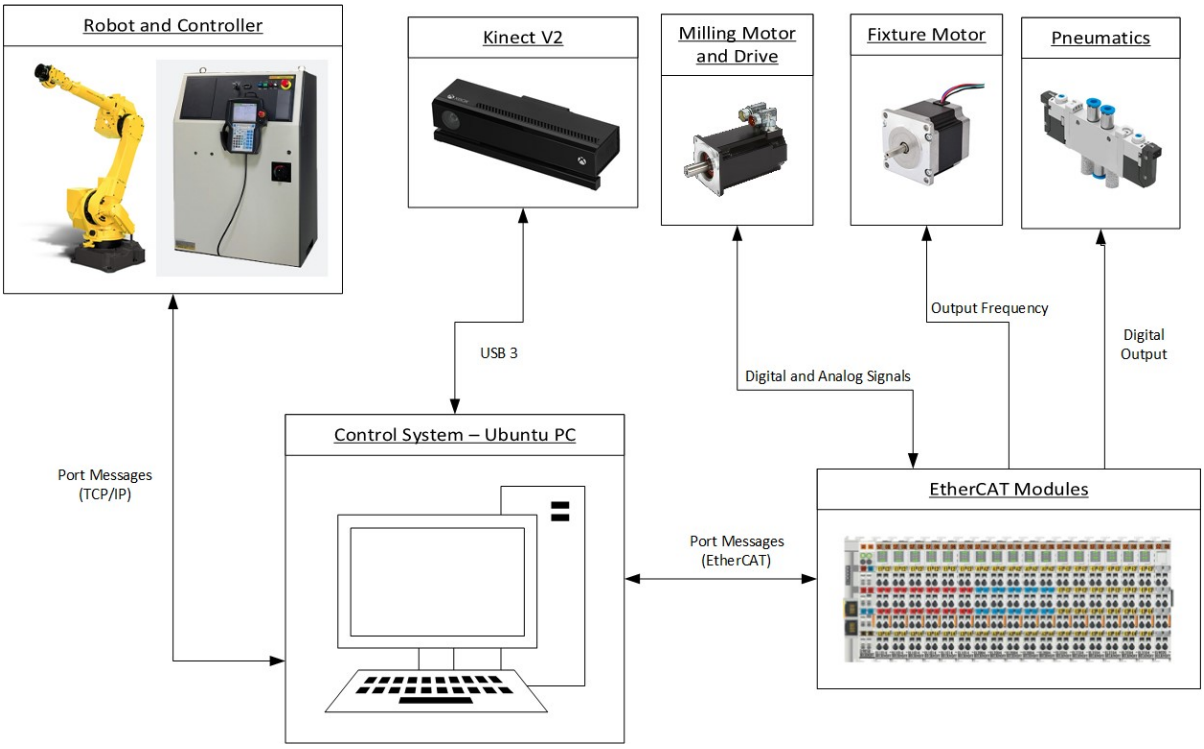


Figure 6 - System Component Overview

2.2 Control System

The main control center for this project is a Linux PC running ROS. The Linux PC runs Ubuntu 16.04 and has the specifications seen in table 1. This system monitors and sends commands to external devices via port messages, EtherCAT, and USB 3.1. This Linux PC does all of the heavy computation in the system. It is responsible for collision free path planning, image processing, process monitoring, some motion control, providing a human machine interface (HMI), and running neural network models.

CPU	Intel i7-4790K
RAM	16GB DDR3-1866
GPU	Nvidia GTX 1080Ti
Storage	250GB SSD
Accessories	4 Port PCIE Gigabit Ethernet Network Adapter

Table 1 - Control system computer specifications

When controlling external axis, a motion planning framework known as MoveIt [16] is utilized. This framework has many add ins, trajectory filters, planning libraries, and collision checkers. This planning framework also keeps track of joint states and action interfaces to allow for a single interface that can control all axis of a system allowing for co-ordinate motion of movement groups. This library also integrates with a transformation library which keeps track of all the transformations between co-ordinate frames. This allows goal poses to be given in any frame in the system. This system is further discussed in section 3.2.

This system also controls all external IOs via an open source EtherCAT library known as Simple Open EtherCAT Master (SOEM) [17]. This library allows for a non-real-time control of EtherCAT IOs and is a class B EtherCAT master as it is currently unable to sync distributed clocks. The library is used with Beckhoff EtherCAT terminals to interface with a servo drive to control the milling tool, control a stepper motor, and pneumatic valves.

This system is also set up to run deep learning and machine vision algorithms. Utilizing the GPU to accelerate these workloads. It is also set up as the system development PC. When running the full control system 16GB of RAM is not required but this is necessary when training a neural network. An SSD is used as the storage drive to speed up system loading and increase responsiveness.

2.3 Robot

The key component in this system is a 6-axis articulated robot, which the rest of the system is built around. This robot is a Fanuc M710iC/50 with an R30iA controller. The robot is shown in Figure 7 and specifications are found in appendix A. The payload of this robot is approximately 50kg. The

robot has a circular workspace of approximately 2 meters. This robot system was previously used for deburring applications and is equipped with a wrist mounted force sensor. The force sensor was not utilized but could potentially be hooked up to EtherCAT inputs and outputs (IOs) to provide force monitoring.

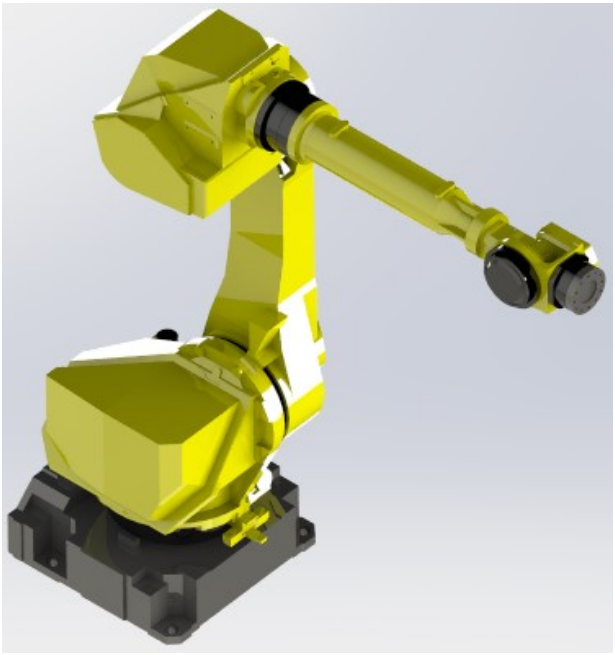


Figure 7 - Fanuc M710iC/50 Render

ROS Industrial provides a ROS driver as well as KAREL and Teach Pendant code providing communication between a Linux PC running ROS and the robot controller. This driver utilizes socket messaging to communicate with ROS and requires the socket messaging and KAREL options for the Fanuc controller. The role of the Teach Pendant code in this driver is to provide motion control and follow buffered points. This is the front end of the driver that controls motion but Teach Pendant code is rather limited and slow. KAREL on the other hand is a compiled low level coding language for Fanuc robot controllers. The KAREL code has two major roles. The first of which, is to receive trajectory messages from ROS and convert them into a form that the robot controller can use and the second is to provide feedback to the ROS driver via port messages. This feedback is either the current position of the robot, a request for more trajectory points, a signal that the motion is done, or a signal that the motion request has started. The loop rate of this communication is 42 Hz on an R30iA controller and

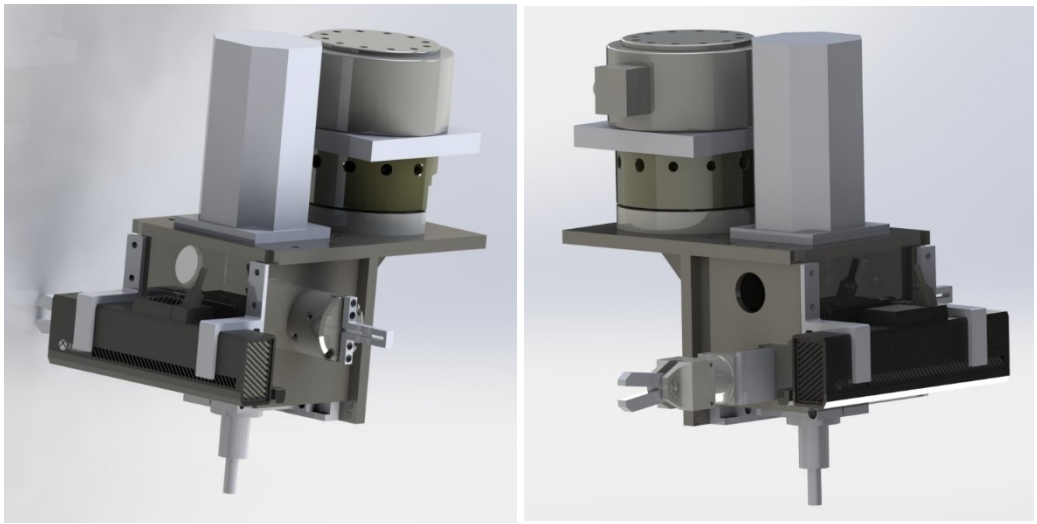
83 to 125 Hz on an R30iB controller. The ROS driver provides an interface to send and receive these messages and transfers the data to ROS topics that can be utilized by the rest of the system.

2.4 Mechanical

The mechanical systems for this project mainly focus on creating specialized and flexible tools for the FPD dismantling. The mechanical components consist of an end of arm tooling (EOAT) consisting of three separate tools and a camera, a custom fixture with a variable fixture height, and a loading system.

2.4.1 EOAT

The end of arm tooling (EOAT) is the main effector of the entire system. It is responsible for all value-added work in the system and contains cameras, destructive tooling and handling tooling. Each of these tools are on a different side of the EOAT so they can be utilized without the other tools creating interference. Renders of the EOAT are shown in Figure 8.



(A) (B)

Figure 8 - Rendering of EOAT where (A) is from the right side showing the CFL gripper and (B) is from the left side showing the CFL gripper

2.4.1.1 Destructive Tooling

The destructive tooling chosen for this operation is a mill. The milling tool uses a FINLEY 50-13-BDS spindle, which includes a collet for tool mounting. This spindle is designed for CNC milling and routing machines and contains bearings that are able to support several kilo Newtons of force. This spindle is attached via a coupler to a Kollmorgan S6SM57S-3000 servo motor. This servo motor has a maximum speed of 3000 RPM, a standstill torque of 4.6Nm and a total power of 0.95 kilowatts or approximately 1.3 hp. This is a reasonable sized motor when compared to many milling systems which tend to have motors rated at 1 to 2 hp. The collet supports tool shank sizes of up to ½" which is the size of mill being used in this operation. This size was chosen to increase rigidity and reduce the likelihood of the tool breaking during operation.

The mill chosen was a 4 flute ½" roughing and center cutting high speed steel (HSS) end mill with a titanium aluminum nitride (TiAlN) coating. A roughing end mill was chosen after performing testing both a roughing and finishing cutter. It was found that the roughing end mill had a larger maximum feed rate than a finishing end mill and provided a better-quality cut. If a traditional mill was used to cut FPDs carbide would be a good cutter choice. However, since this is a robot mounted milling cutter. This is because carbide is a brittle material and cannot handle shock loading. A robot is far less rigid than a traditional mill and would likely cause the tool to break during cutting. The coating was chosen to increase tool longevity and reduce friction on the cutting surface helping the tool cut through stainless steel and reducing the likelihood of plastics adhering to and clogging cutting surfaces.

2.4.1.2 Handling

For handling, two separate grippers were chosen. The first was a parallel gripper which is utilized mainly for gripping and breaking CFL tubes and the second was an angular gripper which performs heavy duty gripping tasks removing components that may still be connected to the FPD and require high forces to remove. Both of these grippers were pneumatic and were controlled with air at 90 PSI.

The CFL gripper is an SMC MHS2-40D. This gripper has two fingers and an effective stroke of 8mm. This gripper has a bore diameter of 40mm and an effective gripping force of 123 N at 0.5MPa. This force is high enough to shatter the CFLs so to grip them fingers were 3D printed with a small

opening in the middle. This opening can be seen in Figure 8 (A). The opening was then filled with a flexible foam. This foam provides support to the CFLs and increases friction while greatly reducing the potential for them to break.

The angular gripper was a VESSEL GT-NS20. This is a modified air nipper, which is not designed for gripping but actually for cutting steel and crimping connector. With a side cutter insert, this model is capable of cutting 2mm of steel and 7mm of hard plastic. The side cutter insert was replaced with a custom tool for gripping. The system provides the same amount of force but uses it with a series of small spikes which can rip and tear apart plastics and metals. Figure 8 (B) shows this gripper setup.

2.4.1.3 Vision

The vision system mounted on the EOAT is a Microsoft Kinect V2 sensor. This sensor contains an IR camera with a resolution of 512 by 424 pixels, a color camera with a resolution of 1920 by 1080, and a frame rate of 30 fps. The Kinect V2 sensor also contains active IR generators that illuminate the camera's field of view. This is an RGB-D sensor meaning that it outputs not only a color image but also outputs a depth map. During runtime these images were resized via bilinear interpolation and rectified to a consistent frame.

The Kinect V2 was affixed to the front of the tooling via two mounting clamps and a tripod mount, which connects through a Lexan cover. These three mounting points provide stability to the camera. In its standard mounting configuration, the tripod mount is used to stop lateral motion and two clamps on the sides prevent the sensors from tilting. The placement of the camera was chosen to protect it from debris from the milling operation while reducing tool movement. The cameras field of view is such that it does not see the milling tool or any of the EOAT. As there is no direct line of sight from the end mill to the sensor likelihood of chips from the cutting operation striking the sensor is greatly reduced.

2.4.2 Fixture

As shown in Figure 9, the fixture is a custom designed assembly made to fixture variable sized FPDs. The fixture includes a variable height back plate, two hard stops, and two pneumatic cylinders. The cylinders are TA-MX0-1.50x52-HC-BSPT=.25-CS-MPR-SST. These cylinders have a 1.5 inch

bore with a total stroke of 52 inches. When run with an operating pressure of 90 psi, the cylinders provide an approximate clamping force of 153 lbf per cylinder. Plates were affixed to these cylinders to provide more even pressure on the FPD reducing the likelihood of the plastic being deformed or breaking due to the clamping pressure.

The purpose of the variable height plate is to bring the edge of the FPD to the same height as the hard stops. By bringing the FPD to the same height as the hard stops, the mill can be fully engaged. This system is controlled via a ball screw mechanism with a diameter of 15mm and a lead of 10mm. The motor connected to this system is a Beckhoff AS1060 that has a holding torque of 5Nm. The drive torque required for a given load is can be calculated by rearranging Equation 1. When rearranging this equation and using a value of 10mm for the lead (P), 0.85 for efficiency (η), and 5Nm for the drive torque (T) the calculated maximum axial force (F) is 2670 N or approximately 600lbf. This holding force is well above the maximum axial load that the milling operation will impart. This ball screw is supported via a lateral member. The structure is made of all either 80 by 80 mm and? 40 by 80 mm aluminum extrusion. The profiles more than support the applied load.

$$\boldsymbol{T = \frac{F * P}{2 * \pi * \eta}} \tag{1}$$

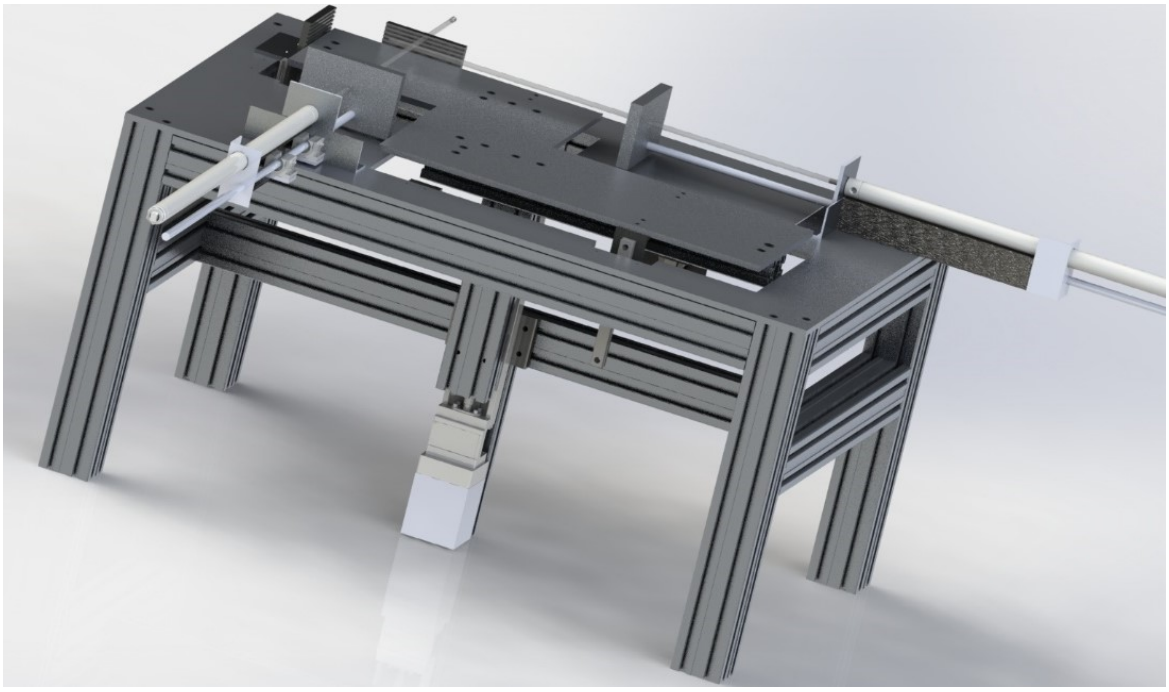


Figure 9 - Fixture Rendering

The shape of the variable height backing plate was designed to allow the monitors center of mass to be supported while still having room available to allow the clamping cylinders to clamp the FPDs to the hard stops. The plate supported with four linear guides and three pieces of aluminum extrusion. The linear guides reduce lateral motion and rotation while the aluminum extrusion supports increase the rigidity of the aluminum plate to reduce flexing.

2.4.3 Loading System

The loading system drawings can be found in appendix B. This system is a conveyor that is controlled by a variable frequency drive (VFD). This VFD is configure to use an analog and digital input to control the conveyor speed and turn the system on and off. The conveyor has a maximum speed of 3m/s and is has an allowable maximum payload of 10lbs per ft^2. The conveyor runs off 600VAC 3-phase and has a length of 11 feet, has a width of 3 feet with a bed height of 36 inches. Figure 10 shows a picture of this conveyor. The loading system has been specified but the states to implement automated loading have not yet been implemented.

Figure 10 - Conveyor for Automated Loading

2.5 Pneumatics

Pneumatics were utilized for fixture and gripping tasks. The only pneumatics in the system are the heavy gripper, CFL gripper, and fixture cylinders. Pneumatics was chosen for these applications to reduce tool size and increase gripping force. Pneumatic actuators tend to be smaller, lighter, and more powerful than comparable electric actuators. Appendix C shows the pneumatic circuit in this system.

2.6 Electrical

The electrical diagram can be found in appendix D. This shows how each of the components are interconnected. Some improvements could potential be made in this system. Firstly, the stepper motor is designed to run on 50VDC not 24VDC. A separate power supply should be used to get the maximum torque out of this stepper motor. Secondly, a transformer is being used to step down 480VAC to 208VAC to provide input power to the 24VDC power supply. This transformer could be removed if a power supply with a 480VAC input supply was used instead.

2.7 Safety

In the creation of this system safety was a key priority. The system is capable of producing a large amount of noise, upwards of 90dB, and the robot is a large potential hazard. To attempt to mitigate

any issues the robot was put in a work cell. This work cell fully encloses the robot with walls made of a panel of steel below a panel of Lexan. There was also a roof added with sprinklers passing though for fire suppression. The cell is approximately 5.5 meters in length and width and 2.7 meters high.

There is a single entrance which has a safety interlock. This safety interlock contains two keys which have to be installed for the door to be fully closed. The robot cannot run in automatic mode if the door is open. The interlock also includes two estop buttons, one on the inside of the cell and one on the outside. The safety circuit is set up so that the milling tool cannot be run unless the robot is operational.

Chapter 3
System Architecture

3.1 Computational Graph

The overall control system chosen for this project was ROS [18]. This was because of the flexibility offered and due to the large amount of open source code that could be quickly added to this project to speed up development time. ROS is not a control system but an architecture to help with distributed and inter-process control and communication. The basis of ROS is a large computational graph.

This graph defines how information flows and updates each state in either a synchronous or an asynchronous manner. The basic flow of the computation graph is shown in Figure 11. This graph outlines the interactions between different nodes in this system during runtime. These nodes are what control the system and allow it to operate. A full computation graph showing all topics and connections is shown in appendix E.

The nodes used to control the Fanuc M710iC/50 are defined in the FanucM710iC50 block/namespace. This block contains several key components. The first of which is joint state feedback. This provides consistent feedback to MoveIt which is the main system for motion control, path planning, and collision checking. It also provides a topic which connects to the robot via port messaging and provides an interface for MoveIt to send movement requests. These requests are done in what is called an action. An action has a request, feedback and results. In this case the request is sent to the robot node which forwards the points to the robot, feedback is sent back via joint positions and error amounts, and the result is either success or failure.

Soem Master is a tool that provides EtherCAT communication to external devices. This communication is non-real-time and runs at approximately 100hz. Though it has been seen to go between 101 and 99hz during typical operation. Soem Master connects to several external nodes which provide an easy interface to control the EtherCAT IOs and provide some level of abstraction to allow the stepper motor to be controlled via MoveIt and to allow the EtherCAT IOS to be set individually

instead of having to write an entire output block to each module just to control a single actuator. This works well enough for unstructured movement but if a lot of accuracy is required this state should be run by a dual kernel system where one kernel is real-time or by a full real-time system. This can be accomplished by running Xenomai or doing a PREMPT-RT patch respectively.

Another key area of this computational graph is the RobotStatePublisher node. This is a key part of the system which defines transformations between co-ordinate frames in the system at any given time. This is used to track where every joint in the system is and track the transformations from one co-ordinate frame to another. This is one of the key features of this system that makes ROS so appealing. This is used to keep track of where pictures where when they were taken, to transform points from one frame to another, and much more.

The driver for the Microsoft Kinect V2 used for the vision system is also on this graph it exists within the eoat_kinect2 block/namespace. This driver is responsible for transforming depth and color images into point clouds to add collision geometry, communicating with the Kinect V2 sensor, and publishing the transformations between the Kinect's camera frames. It also publishes color and depth images in 3 resolutions SD, QHD, and HD. These resolutions correspond to a resolution of 512 by 420, 960 by 540, 1920 by 1080. This system also rectifies the images and aside from SD, which rectified the color image to the depth image, the depth images are rectified to the color images. To decrease computation time and maintain reasonable accuracy the proposed system uses mostly QHD resolution images. This topic is also connected to filtering and image gathering nodes. This is not shown in the computation graph as these nodes connect to the Kinect driver only when they wish to gather an image. This is done to help reduce communication overhead in the system as when a node connects to the Kinect driver it continuously streams images to that node. This could lead to gigabytes of information being transferred per second for no real reason.

Another key node is the plan move path server node. This node is the main control center for all movement in the system and is a custom layer of abstraction for MoveIt. It is the single control point for motion in this system and is used to simplify movement requests. The specifics of this node are covered in the coming section MoveIt! - Motion Planning and Collision Avoidance.

The final point in the computation graph of interest is the SmachServer. This server is what is explained in process flow and is the main point of control for the whole system. This node contains a

state machine which runs the full process of the system from start to finish. It runs when it is called externally and ends when the user publishes a value of True to the "/finishing" topic. It does not end immediately but waits until the last FPD that has been loaded into the system has been removed.

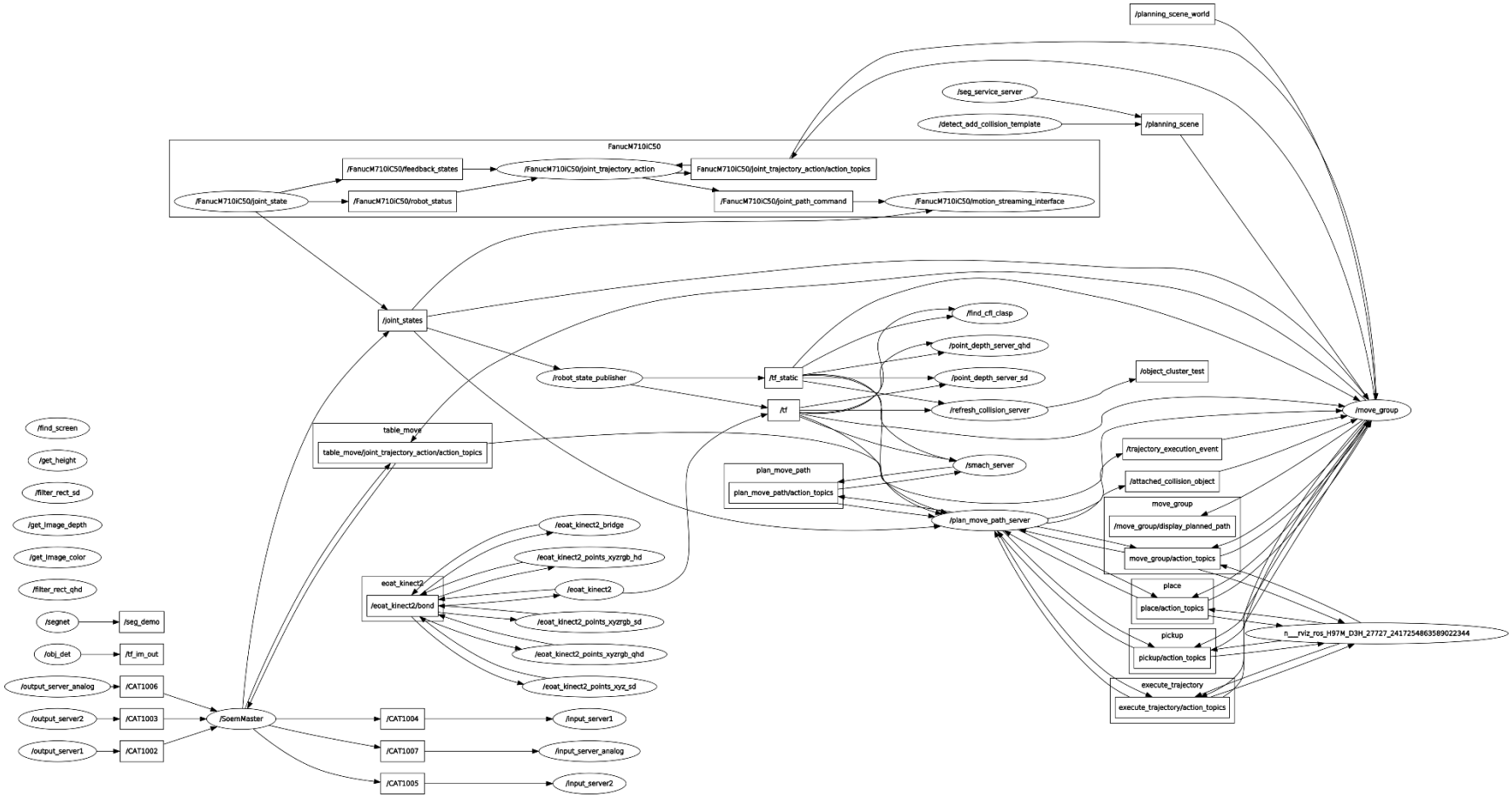


Figure 11 - System Computation Graph

3.2 MoveIt! - Motion Planning and Collision Avoidance

The core of this control system is MoveIt which is an open source and ROS integrated package of software that was initially designed for mobile manipulation tasks. Though this application is not mobile this tool set is immediately applicable to the application. There is also a consortium which focuses on extending this software and ROS as a whole into the industrial robotics and manufacturing space called ROS-Industrial. They develop drivers for most industrial robots from many manufacturers including Motoman, Fanuc, ABB, and Universal Robotics. This section will focus on the covering the motion groups which are implemented in this project, what collision geometry was implemented, and motion planners that are implemented in MoveIt.

MoveIt itself is more of a switchboard allowing for quick and easy usage of planning libraries, trajectory filters, collision checking, and motion groups. Motion groups are either a single or a collection of joints that MoveIt can control. When MoveIt controls these groups, it will check the geometry in the system to ensure that motion does not cause collision. MoveIt will also keep track of the potentially changing collision geometry and can replan paths if the geometry changes to ensure the path is still collision free. MoveIt is mainly designed with free motion planning in mind. Free motion planning in this context is planning from the stating pose to the goal pose but allowing the motion planner to decide the path. The motion planner can have many parameters including collision geometry, waypoints, orientation constraints and position constraints. MoveIt also monitors joint state topics to see the position, velocity, and acceleration of all motion groups in the system. Joint state topics are ROS topics which a specific joint's or robot's driver publishes the current state of its joints periodically.

3.2.1 Motion Groups

The general motion groups for this system are shown in Figure 12. The specific components are explained in the previous chapter under the mechanical system. These motion groups are each defined as kinematic chains starting from the base link to each end effectors axis. Aside from the fixture axis each of the motion groups consist of joints 1 to 6 of the Fanuc M710iC/50 with a different static transformation from the robot tool frame to the end effectors tool frame. The motion groups are defined as follows: manipulator, vision, cfl_gripper, heavy_gripper, and fixture_axis. These groups are set to links which are defined through transformations found in the tree diagram shown in Figure 13.

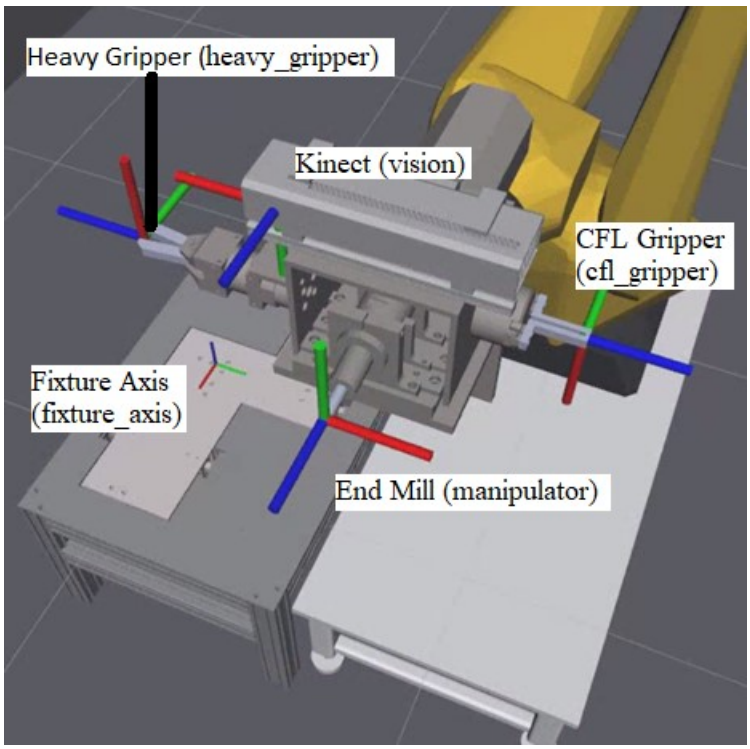
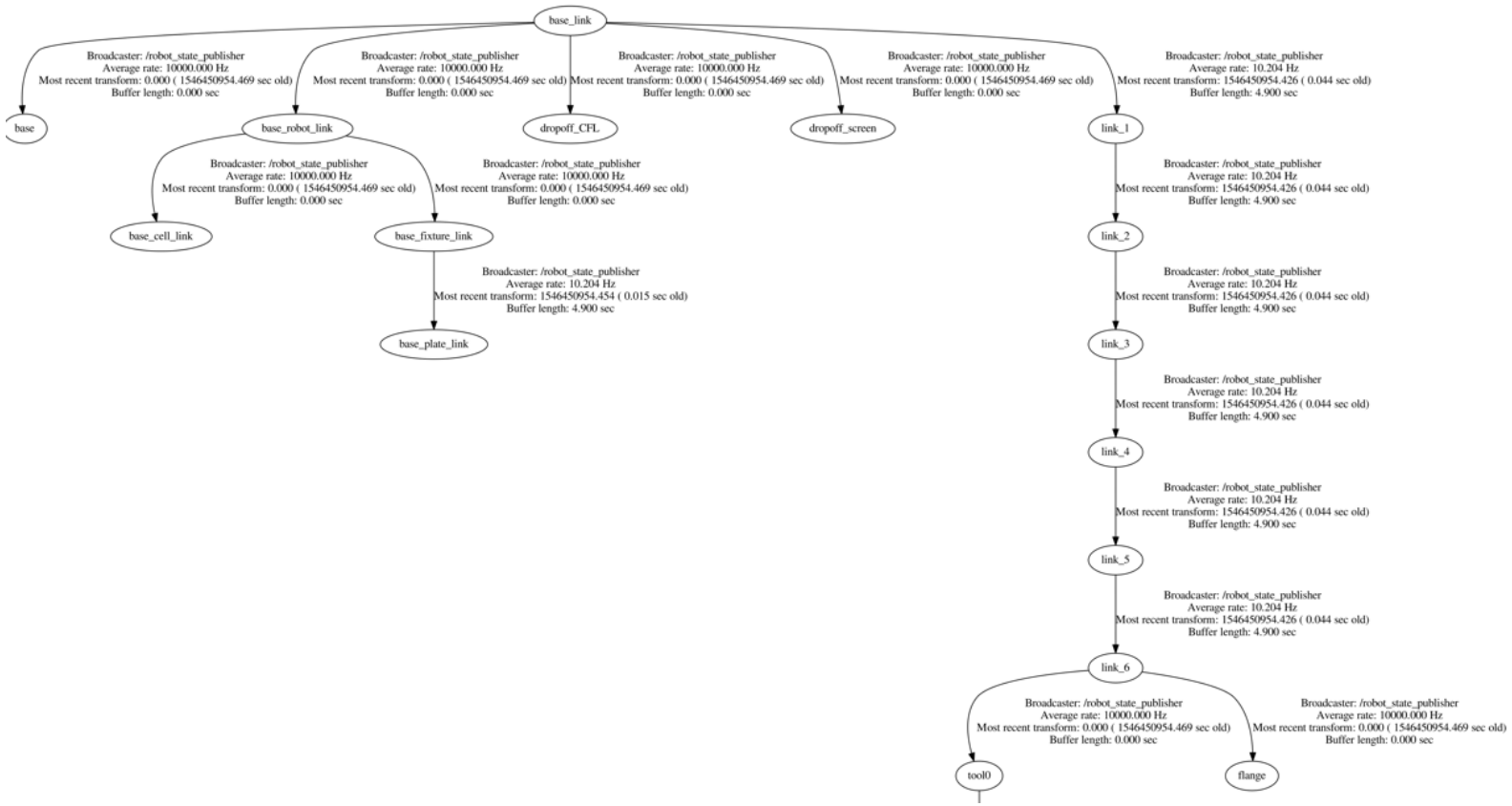


Figure 12 - Motion Group Links where X, Y, and Z axis are red, green, and blue respectively


base_link
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1546450954.469 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 1546450954.426 (0.044 sec old)
Buffer length: 4.900 sec
base
base_robot_link
dropoff_CFL
dropoff_screen
link_1
base_cell_link
base_fixture_link
link_2
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 1546450954.454 (0.015 sec old)
Buffer length: 4.900 sec
base_plate_link
link_3
link_4
link_5
link_6
tool0
flange

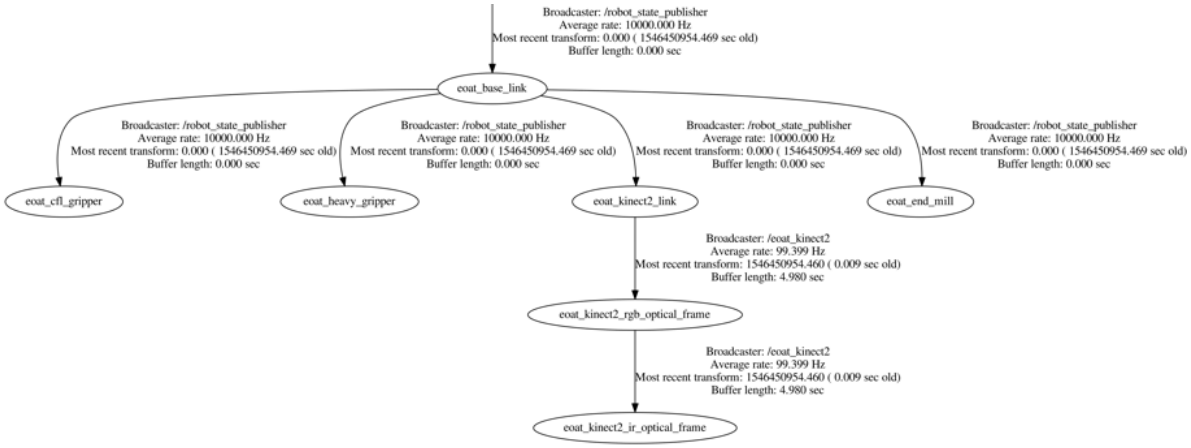


Figure 13 - Coordinate Tree Diagram

3.2.1.1 Fixture Group and EtherCAT Driver

The fixture group is slightly different than the others in this system. It is the only motion group that is not controlled by the Fanuc driver and the only linear axis in the system. This axis is controlled via a custom driver made for ROS utilizing simple open EtherCAT master (SOEM). This driver is designed to work in pulse mode and has several parameters. This driver directly interfaces with ROS joint state topics and contains its own action server to send motion commands.

The specific driver is designed to interface with a Beckhoff EL7041-1000 stepper motor terminal. This terminal is capable of controlling a 50V stepper motor with a maximum current draw of 5 A. The terminal has an interface of an incremental encoder and two limit switches. It has a two-phase output and is capable of running a bipolar stepper motor with 64-fold micro-stepping with a max frequency of 8000 steps per second.

The only control that this system has for speed is a single signed 16-bit integer which is percentage of max speed or V_{con}. In order to provide control for this system the driver must be able to convert between stepper/encoder counts and real-world units, keep track of counter over and underflow, and monitor position and velocity. To keep track of overflow and underflow value a signed long integer was used to hold onto an offset count. This is added to the current stepper count to find position. The equations to convert between V_{con} and velocity as well as find the position are shown in Equations (2.1) and (2.2) where there are several constants including the steps per rotation, reduction, full speed, and max step speed. The steps per rotation is a number which contains the step angle and micro stepping amount. This parameter is used to convert the steps being done in the system to radians on the output shaft it is equal to 360 degrees divided by the step angle multiplied by the number of micro steps per step. Reduction is used to convert between the output value of the motor to the actual output of the actuator. This allows the stepper driver to directly control a ball screw and output its velocity and position and/or to compensate the output to a gearbox on the output shaft. The full speed is the maximum speed of the stepper driver in steps per second and the max step speed is a constant equal to 32768 or the maximum value V_{con} can be set to.

$$Postion\_Actual = \frac{counter\_value + counter\_offset}{steps\_per\_rot * reduction} \tag{2.1}$$

$$V_{con} = \frac{ReqVel * steps\_per\_rot * reduction * full\_speed}{max\_step\_speed} \tag{2.2}$$

This driver is controlled by being sent a standard follow “joint trajectory” request from MoveIt. This request contains a list of position, velocity, and acceleration waypoints that the joint must go through. The driver loops at a rate of 100Hz and run through the control loop defined in Algorithm 2. This control loop works well but is not run in real-time and does not compensate for the acceleration that should have been done between the steps of the system. This means that the driver is always running a little bit behind what is requested by MoveIt. This is acceptable for single axis control but should not be used for coordinated motion. The system also has a setup to compensate for overshoot by jogging back at a set safe speed. This allows the system to be accurate but can lead to slow response times. The driver also has a tolerance parameter which defines how close we need to be to the next waypoint in the request to transition. The driver tries to remain accurate within two steps of the final requested position. During waypoint transition the driver sets the current velocity to the new waypoints starting velocity. This is done to try to remove any errors associated with the velocity update procedure. This control procedure does not operate as a PID control loop but with rudimentary velocity control.

Algorithm 2 - EtherCAT stepper motor driver control loop

```
If there is an active goal
    If current position is within tolerance of next waypoint position
        If the next waypoint is the last waypoint
            Set velocity to zero and return success
        Else
            Increase waypoint number
            Set current velocity to waypoint velocity
    Else
        Increase current velocity by waypoint acceleration times period (100hz)
        Convert current velocity to V_con using (2.2) and set
Else
    Set velocity to zero

If overflow or underflow
    Offset counter_offset by max counter value
Get Postion_Actual via (2.1) and publish joint state
```

The homing routine is a simple procedure. The motor is moved at approximately 5% of maximum speed in a single direction and goes until an input is triggered. In the current system, this input generates by an inductive sensor. Once this is done motion is stopped and the counter and counter offset variables are reset to zero.

3.2.2 Collision Detection

Collision detection in this system is mainly done on a self-collision or mesh to mesh basis. The collision library used is Flexible Collision Library (FCL) [19]. This library is used by selected motion planner and MoveIt to determine collisions and to create cost maps for trajectory optimization libraries. FCL is capable of the following tasks; collision detection, continuous collision detection, self-collision detection, penetration estimation, distance computation, and broad-phase collision. FCL is not often used directly but is utilized by MoveIt and the selected collision planning library.

In this system the collision geometry defined can be seen in the Figure 14. This collision geometry includes a translucent cell which is used to represent the guarding around the robot. This system almost completely relies on FCL checking the movement in the system to ensure that there are no collisions. During operation there are added collision boxes generated for the clamping cylinders that are used on the system. These are used to ensure that there are no collisions with the hard stop during cutting. There is however one key area of collision checking that cannot be done during runtime and that collision with the FPD that is loaded into the cell. If the FPD was added as collision geometry the cutting processes would fault every time it was run due to collisions being detected between the milling tool and the FPD. This would also lead to errors with gripping processes. To attempt avoiding this all free motion being done near the FPD is planned to a safe point far above the FPD and linear motion is then used. This should avoid the robot path planning the EOAT into the FPD during free movement.

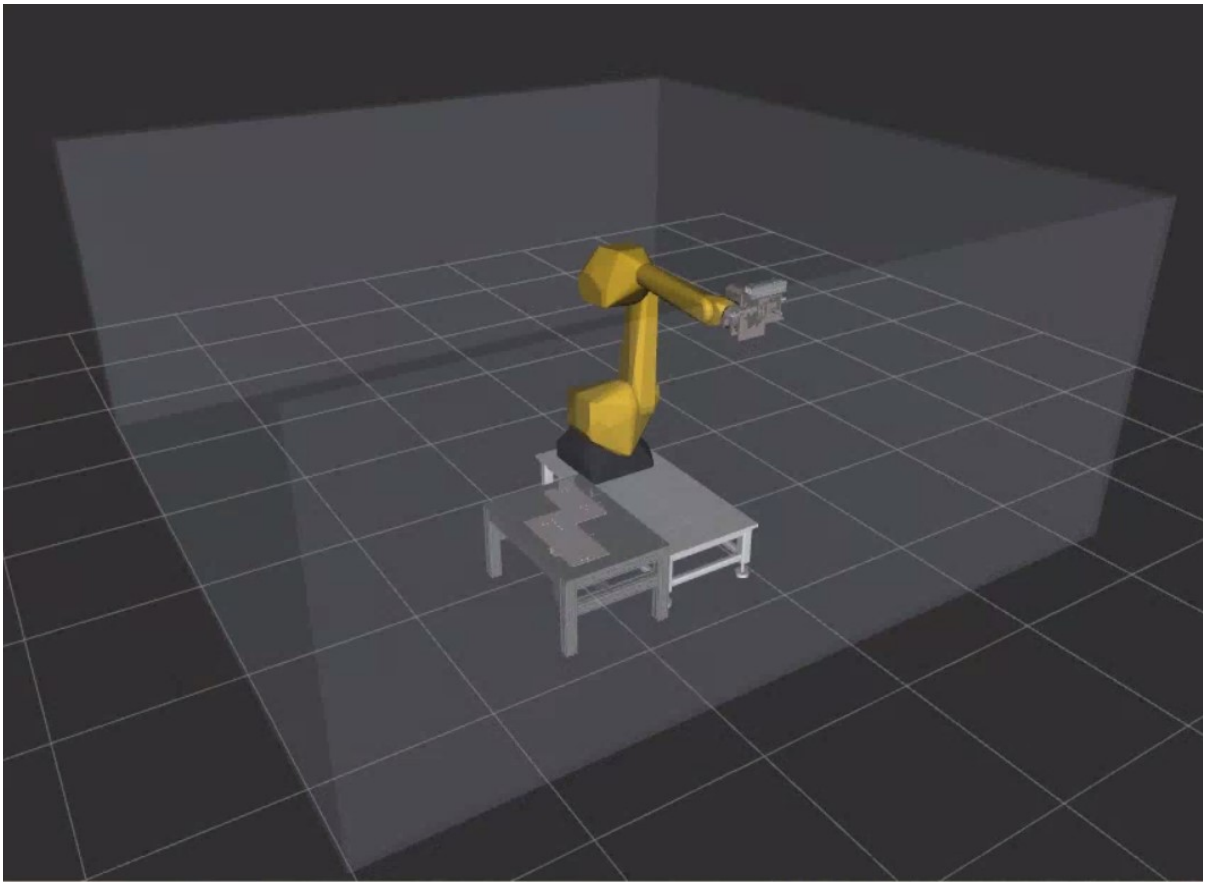


Figure 14 - Collision Geometry in System

3.2.3 Motion Driver – Plan Move Path Server

The plan move path server is a layer of abstraction that was made to operate between MoveIt's planning API and the state machine that controls the system. This layer serves several distinct tasks. It allows for simple generation of complex linear paths, enables speed of linear paths, unifies path planning to a single action, enables monitoring of conditions to stop motion, and ensures that all motion planners are properly initialized prior to use.

This action has an interface outlined in table 2,where the Request is what is sent to the server, Response is what the server responds with, and Feedback is a message that the server sends during runtime. The Request includes an array of poses. These poses include a 3D point and quaternion that are defined in the base frame. There are also parameters for the movement including arrays for speed in meters per second,

step size in meters, and joint poses, an integer which indicates the group to move, a float which scales the joint velocities for free motion, and Booleans that say whether to do a full plan and where the plan is in joint state. By changing these parameters, there are four different ways to interact with the server. There are two parameters which always need to be set. These parameters are group and joint state, which tell the server which motion group to use and whether we are setting the goal pose as a 3D pose or joint position. The input parameters are shown in Table 2.

The first way to run motion in this server is by setting the Boolean for joint state to true and defining a joint pose and joint move speed. This is used to do free motion planning of a group to a specific pose. Free motion is allowing the motion planning library to define a collision free path from the starting position to the goal pose. The joint move speed has an expected value of between 0 and 1. This parameter is used to scale the planned paths velocity so that slower of faster free motion can be done. This is done depending on the situation. As if the area is unknown, it may be best to do a slower motion to allow the path to be re-planned to avoid moving collision geometry. This is not commonly used in this system. Joint move speed is commonly set to a value of one. An example use of this application is adjusting the fixture axis or moving the robot to a joint position.

The second way to run motion is by including a single pose in poses, setting joint state to false, and a value for joint move speed. This tells the system to perform free motion planning to the goal pose while scaling the velocity by the amount in joint move speed. This is commonly done for tasks that don't need a specific path but just to get the motion group to a position. An example of where this would be useful is moving the vision group to a static point to find the FPD.

The third way this can be used is by including n poses, n-1 speeds and step sizes, a group, the joint move speed, and setting both joint state and full plan to false. In this situation free motion is done to the first pose just like in the previous way, then linear motion is performed to move to all subsequent poses. In this linear motion the speed and step size for the linear path planning are defined in the speed and step size arrays. The step size is required as ROS plans all points a joints positions and does the inverse kinematics on their end. Due to this, the step size is required to find the number of points to put between the start and goal pose to create the linear path. The full plan Boolean says whether or not to plan all of the linear paths at the same time. When this is set to false each linear path is performed one at a time alternating between moving and planning each path. This tends to be done for longer paths where it would take too long to plan

all at once or when the driver it is sending the data to has issues with large paths. This is the case with the Fanuc driver where it often returns an error when full plan is used.

The final situation is the same as before but with full planning set to true. This behaves the same as previously except the linear motion is planned and executed as one large path containing multiple linear moves. It is of note that this and the previous two ways of interacting with the server check the current position of the move group and does not perform free motion if it is in the desired position. Requests like this are commonly sent to the plan move path server when complex linear motion must be accomplished. Whenever linear motion is done in this system either this or the previous method have been used.

Request	geometry_msgs/Pose[] poses float32[] speed_m_s float32[] step_size_m float32 joint_move_speed int32 group bool full_plan bool joint_state float64[] joint_pose
Response	bool success string end_message int32 failed_path int32 error_code
Feedback	int32 step string step_message

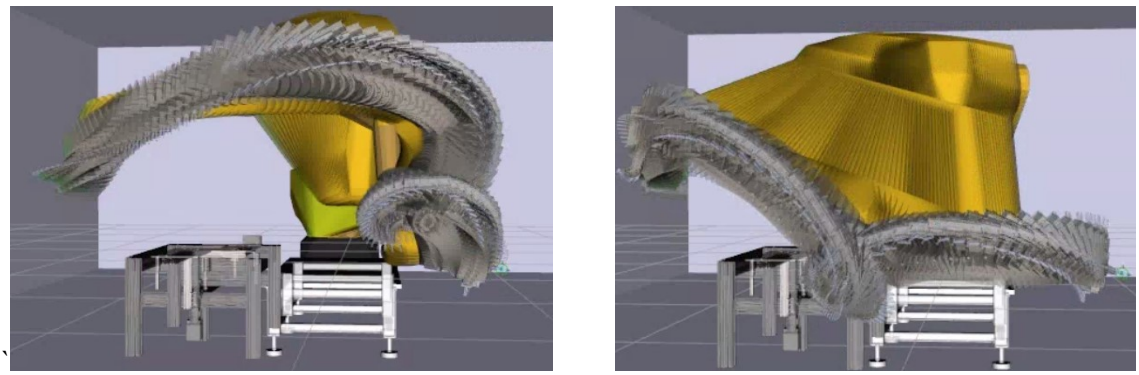
Table 2 - Plan move path server action specification

3.2.4 Available Motion Planners

There are several open source motion planners available to MoveIt. Some of planners include many well know algorithms while others are only a single algorithm. In the course of this project, three different motion planners were tested. They were Open Motion Planning Library (OMPL) [20], Covariant

Hamiltonian Optimization for Motion Planning (CHOMP) [21], and Stochastic Trajectory Optimization for Motion Planning (STOMP) [22].

OMPL is a toolkit with a large number of planning algorithms it contains a large variety of algorithms including single-query, multi-query, and optimizing planners. The two planning algorithms tested where RRT-Connect [23] and KPIECE [24]. KPIECE is a tree-based planner that works by adding motions to an expanding tree. It is not searching for an optimal but instead trying to minimize computation time and find a path to the goal. It works well and is very robust and capable of finding paths that many other planners would fail. KPIECE and OMPL are the default planners for MoveIt and were used for the majority of the project until it was found that KPIECE would commonly lead to unpredictable and far less than optimal paths. When utilizing this algorithm, a path would sometimes take over 5 seconds to plan and it would often lead to the robot going far out of its way. KPIECE would occasionally make the path over 10 times longer than the shortest path possible and having large arching joint movements. When this was experienced, it was necessary to try a new method. This is when RRT-Connect was considered. RRT-Connect tree-based planner that builds two rapidly-exploring random trees (RRT) at the start and goal pose of the path. These trees expand and converge towards each other via greedy heuristics. This method is designed for use in relatively sparse environments. Experimentally, this was found to solve all of the problems noticed by KPIECE. The greedy heuristic appeared to create well behaved direct paths to the goal, there were no longer any large swings in the arm during planning, and the computation time was reduced to under 1 second for path generation in most situations; often under 0.2 seconds. Figure 15 shows an example of the paths generated by KPIECE and RRT-Connect.



(A) (B)

Figure 15 - Paths Generated by (A) KPIECE and (B) RRT-Connect

STOMP and CHOMP are both optimization-based algorithms designed to find not just a path but an optimal path. STOMP was built to supersede CHOMP by removing issues it had with local minima. When tested both of these methods performed well but they were very costly requiring more time than even KPIECE. STOMP and CHOMP appear to have these issues due to the optimization approach requiring the inverse kinematics of the robot and collision to be calculated a very large number of times when compared to other methods. This could be sped up by replacing the standard inverse kinematics library used in MoveIt, but it was not possible to implement this method with the configuration of this system. Due to this, both STOMP and CHOMP were abandoned.

The final choice for the planning library was RRT-Connect due to it having by far the fastest computation time and predictable well-behaved paths.

3.3 Process Flow

The overall system is controlled by a finite state machine [25]. The state machine that interfaces with several drivers and services to control the robot and external devices and contains several states and sub states. The main system is broken into six main states. These starts are Initialize System, Recover, Process Monitor, Process TV, Load System, and Unload. All states and state machines in this process can be stopped early or preempted. This was included to allow an operator to stop the system immediately if necessary but still allow for recovery afterwards. The transitions are governed by the outcomes of other states. Initialize System then transitions into either Recover when preempted, Load System upon success, and exits the state machine with a failure result upon failure. Load System transitions into Process Monitor

when a monitor is detected during loading, Process TV when a TV is detected during loading, and Recover upon failure or preemption. The type of FPD is detected via an object detection neural network in the Load System state machine. The Process TV and Process Monitor states have the same transitions the system transitions to Recover upon failure or preemption, and Unload upon success. Recover transitions to an output of success when done, and output of preempt when preempted, and to Load System upon success. This state machine is shown in Figure 16. The system also includes demo and auto modes. In demo mode "wait for yellow push button states" are added which pause program execution until an input is pressed. These states allow an operator to test sections of the system and check the results prior to continuing. The state machine was developed with a ROS library called SMACH [26]. This library includes tools that assist with creation and introspection of task level state machines.

Nearly all states have typical return values of "Error" or "failure", and "Done" or "success". An "Error" or "failure" value is returned if there is a collision during motion planning or movement, an error in code, or vision algorithm cannot be completed in the state. "Done" or "success" is typically returned when the state is successful. Certain states also have a "preempt" return value which is typically returned when preemption is requested in the state machine. This is normally used to initiate immediate shutdown. Not state machines can return a "preempt" value, typically state machines that do active work or interface with IOs will have preemption enabled. Many other states have different options available which are used to branch the code depending on typically vision results. The branching outcomes will be discussed on a per state basis but "Error", "Done" and "preempt" returns will not be noted. State machine diagrams have been included in which these outcomes can be found.

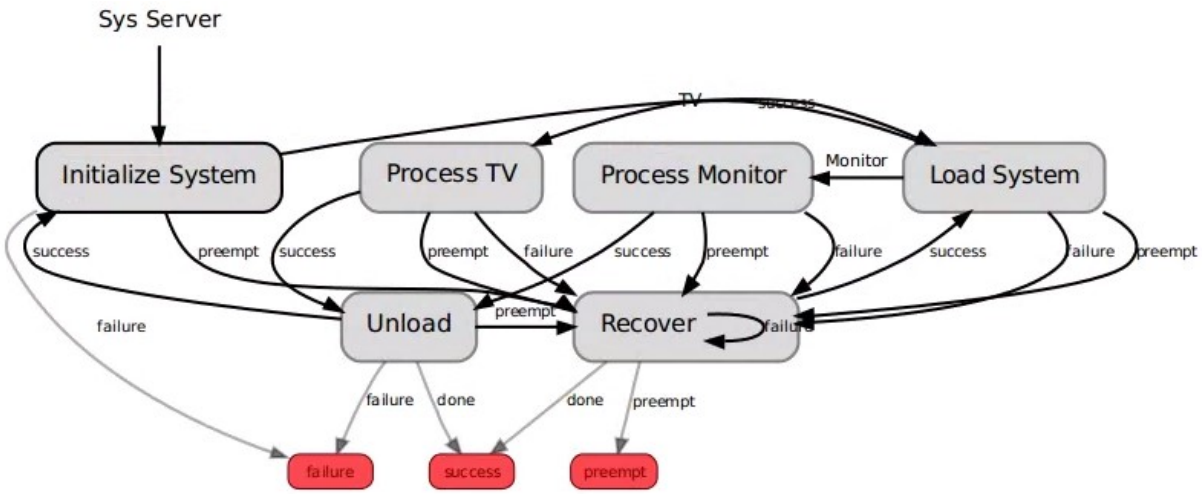


Figure 16 - Full System State Machine Diagram – High Level View

3.3.1 Input Output States

To keep control consistent and segregate operations that create motion in the state machine each IO operation is put into individual states. This helps ensure that the designer is aware of what the states of the IO in the system should be at any given time.

In this discussion we will cover the most common IO states. Some specific examples of this are states that interface with end effectors such as the grippers and mill.

3.3.1.1 Start and Stop Mill

This IO state is used to stop and start the mill. The start mill operation not only starts the mill but also enables the servo drive. When the mill starts it runs at a speed of around 1000 RPM clockwise. The electrical connections have been setup to allow for variable speed control, but this has not currently utilized. This however may not be useful as there is no way to change the robot speed without canceling and replanning motion. This would cause the system to stop and may increase the processing speed significantly. Due to this, a constant speed milling speed is used.

3.3.1.2 Vacuum On and Off

This state is responsible for sending a signal to a vacuum system to remove chips and debris from the cutting surface. The mechanical for this has currently been designed but the mechanical and a vacuum system is not in place to interface with this system. For now, this state is just a placeholder.

3.3.1.3 Open and Close Heavy and CFL grippers

These states are responsible for opening and closing the pneumatic grippers in the system. The solenoids for these grippers are single action so only a single IO needs to be switched to interact with these grippers.

3.3.1.4 Reset Robot

The Reset Robot state simply turns on an EtherCAT output, which is connected to the robot, waits for a second, then turns it off and waits for another second. This output is connected to an input on the robot which is configured to reset it on a falling edge signal. The first wait is done to give the robot time to register the input signal while the second wait is done so that the robot has time to reset prior to the state machine transitioning back to Check Robot again. If the system can set this output with no errors, then it returns a value of Done. If an error is encountered, it returns Error.

3.3.1.5 Wait for Yellow Push Button

This state is added in when the system is in demo mode. Once this is turned off all Wait for Yellow Push Button states are removed from the system. The purpose of this state is to block the system and wait for the input from a yellow push button on the cell. This is useful when testing and demoing the system to stop motion at key points to inspect and potentially fix any issues with the system during runtime.

3.3.1.6 Clamp Screen

This state just swaps two EtherCAT outputs. The solenoids used to run the clamping cylinders are individually actuated with one solenoid to move it to each state. To ensure that the solenoids on the pneumatic valves don't oppose the retracting outputs are turned off. There is then a wait command for half a second, extend one cylinder, wait again, then extend the other. The cylinders are turned on one at a time to try to avoid the chance of the FPDs getting stuck at an odd angle. If one is closed first, then the hard stops and cylinder will line up the FPD. The other cylinder will then push the FPD, so it slides along these

surfaces to the other fixture hard stop. When doing the clamping the cylinder on the short side of the fixture is closed first.

3.3.1.7 Unclamp Screen

This state is the opposite of the Clamp Screen state. In this state first, the extending outputs are turned off, there is then a 1 second pause before the retracting outputs are turned on. Unlike the Clamp Screen state, these are not actuated separately.

3.3.2 Common States

Common states are states that are used in more than one sub-state machine. These states often behave the same where they are used but have different parameters that are set to change the output. However, it is not always the case. These common states are typically due to needing the same action to take place at different points in the system.

3.3.2.1 Move to start

Move to start is a free motion command that tells the Robot to move the camera frame above the fixture, so it will be able to run vision commands on the loaded FPD. This moves to a static position of (0.9, -0.76, 1.05) meters in the base frame.

When conveyor loading is fully integrated this state will be duplicated with a different static position which will jog the robot above the conveyor to detect the presence of an FPD. If there is and FPD present, it will be loaded. If there is not, the conveyor will be incremented, and the states will repeat.

3.3.2.2 Home Table

This state sends a home command to the EtherCAT stepper motor driver that is connected to the ball screw driving the fixture back plate. This home command runs the ball screw at a constant speed backwards until an inductive sensor detects the plate. When this happens, the motion is stopped, and the joint position is updated to zero. This ensures that the output height is being reported and the actual system height is consistent.

3.3.2.3 Find Screen Seg

This state has two input parameters, which are pixel offset and the number of times to attempt to find the screen bounds. The state has an output of 4 points, which are ordered based on where they are located in the input image. The points are sorted into the following order top left, bottom left, bottom right and top right relative to the image taken from the EOAT. It is ordered in this way to ensure that future states can assume the order of the points without having to spend extra time post-processing and ordering these points. The offset parameter is used to increase or decrease the size of the detected bezel to help with point transformation. As the bezel of an FPD often has a small height gap between the bezel and screen the offset is often used to ensure that a proper consistent height is given when the points are transformed.

The steps of this algorithm are outlined in Algorithm 3. In this process the segmented image is found by running an image segmentation neural network. This algorithm tends to work well with TVs but in not as successful on monitors.

Algorithm 3 - Find screen boundary via image segmentation

Repeat until number of tries are complete or loop broken

Get Depth and Segmented Images

Binarize Segmented Image so screen class is 255 and all other classes are 0

Detect contours and use contour approximation on the detected contours

Prune contours so that they have 4 sides after approximation, internal angles are around 90 degrees, are convex, and have an area greater than 20000 pixels squared

If no contours are left continue loop

Keep the detected contour with the largest area as the detected bezel and sort contour points

Offset contour points towards the center of the contour in both X and Y by offset amount parameter. For instance, the top left point will be offset by positive offset in X and Y. While the bottom right will be offset by negative offset in X and Y. If the offset parameter can be negative.

Use depth image and contour points to find 3D points relative to base frame

Break loop and return success

Return failure

3.3.2.4 Find Screen Depth

Find Screen Depth is very similar to Find Screen Seg. The post processing is very similar but the information to find the bezel comes from a different source. The image is not segmented but rather the detected depth image is used to find the bezel. This state has three input parameters, which are pixel offset, a bounding box which is always only a part of the screen, and the number of times to attempt to find the screen bounds. The state has an output of 4 points, which are ordered based on where they are located in the input image. The processes for this state are found in Algorithm 4.

This process tends to work well with Monitors but not with TVs. The reason this state works is due to some peculiarities in the depth sensor used. As described in chapter 4, the Kinect V2 depth sensor will not see the filters on the screen. The depth found in the screen region is instead the depth of the back of the screen where the CFLs are located and not the screen surface. This creates a definite step in depth, which can be used to find the boundary between the bezel and the screen.

Algorithm 4 - Find screen boundary via depth thresholding

Repeat until number of tries are complete or loop broken
Get Depth Image
Use region parameter and find the minimum and maximum depth values inside this region
Binarize the image using the minimum and maximum values found. Set to 255 if the depth is greater than the minimum minus 15 and less than the maximum plus 15. Set to zero otherwise.
Detect contours and use contour approximation on the detected contours
Prune contours so that they have 4 sides after approximation, internal angles are around 90 degrees, are convex, and have an area greater than 20000 pixels squared
If no contours are left continue loop
Keep the detected contour with the largest area as the detected bezel and sort contour points
Offset contour points towards the center of the contour in both X and Y by offset amount parameter. For instance, the top left point will be offset by positive offset in X and Y. While the bottom right will be offset by negative offset in X and Y. If the offset parameter can be negative.
Use depth image and contour points to find 3D points relative to base frame
Break loop and return success
Return failure

3.3.2.5 Check Robot

The Check Robot state is responsible for checking that the system is operational and if there are any errors present transitioning to states that attempt to remedy this. The state checks whether EtherCAT inputs are broadcasting their state, the robot is not broadcasting its joint states, or the robot is estopped. If the robot or the EtherCAT inputs are not broadcasting their states it returns a value of System Error. If the EtherCAT systems are working and the robot estop input indicates that the robot is estopped, it returns a value of Estop. If the state is preempted it returns a preempted result. If no errors are found the system returns robot operational.

3.3.2.6 Move to CFL Vision Position

Similar to Move to Start process, this state is used to jog the robot into a position where the FPD can be seen. This vision position is used in TV dismantling where CFL detection is required. The vision

position is calculated based off of the found screen bounds in the previous state. The X and Y positions are the center of the found screen from the previous state machine and the Z position is the screen size (length of opposite corners) plus 0.1 meters. This measurement was found experimentally and allows both the depth and color images to see the full screen.

3.3.2.7 Twist

The twist state is commonly renamed to Twist, and Twist Remove. This state rotates the CFL gripper tool by 5 degrees in either direct and moves it in positive and negative Y by 1 centimeter. This is done to break the end of a CFL tube to aid in its removal.

3.3.2.8 Move Safe

Move safe is a basic state that just moves the EOAT up in the Z-axis by 0.4 meters via linear motion. This state is commonly used to clear specific areas to start free motion, or for recovery where we are unsure of the current state of the workspace and want to jog the EOAT up away from the fixture to be safe. This state has a maximum jog height of 1.2 meters to avoid returning an error if the safe position is in collision with the roof of the cell.

3.3.2.9 Remove Collision

This state removes any extra collision added by the other states in the system. This is done mainly to remove the collision objects added in by the Add Pusher Collision sub state in the Loading state machine. This is done to remove extra collision geometry when the clamping cylinders have been retracted.

3.3.3 Initialize System

The initialize system state is used to ensure the system is in a functional state prior to starting, reset all the outputs to the desired initial states, and home the fixture axis. The sub states in this state machine are check robot, initialize, reset robot, and home table. Figure 17 shows the process flow for the initialization state machine.

3.3.3.1 Check Robot

The basics for this state are covered in common states. When this system returns Robot operational it transitions into Initialize. If the system returns System Error, then it transitions back into check robot.

This blocks the system as if the robot or the EtherCAT is not connected the state machine does not have a way of fixing it. Since they cannot be fixed in the system and the system cannot be run properly without them blocking is the only option. If these are functioning and it detects that the robot is estopped the system transitions into Reset Robot. If it is preempted it preempts the main state machine.

3.3.3.2 Reset Robot

The basics for this state are covered in common states. This state attempts to reset the robot and when that attempt is complete it transitions back to Check Robot. If there is an error in the attempt, it returns error on the full state machine.

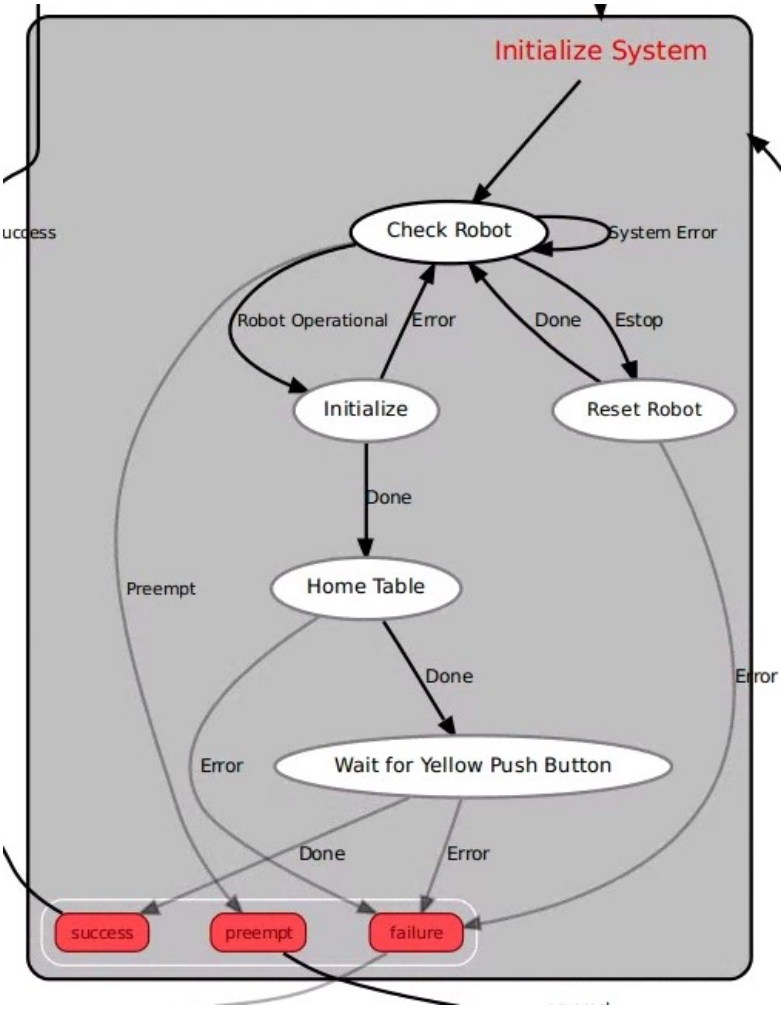


Figure 17 - Initialize System State Machine Diagram

3.3.3.3 Initialize

The Initialize state is a simple IO reset or initialization state. It is responsible for unclamping the fixture cylinders, ensure the mill is not on, and priming the camera. The first two actions are self-explanatory and are just simple calls to EtherCAT IO to change their state. The other action is a necessary hack to ensure the system functions properly. The Kinect V2 on startup will occasionally send blank depth data for the first several requested images. Priming the camera is requesting 10 images in quick succession to ensure that these blank depth images are not transferred to later states in the system where they could

cause the vision tasks to fail. Once this is complete the stepper motor is then homed, and the task is complete.

3.3.4 Loading

The loading state is responsible for adjusting the fixture and detecting the type of FPD in the system. It is an essential first step which each other state relies upon. It is also responsible for updating collision geometry, so the system can function properly. Figure 18 shows the process flow for this state.

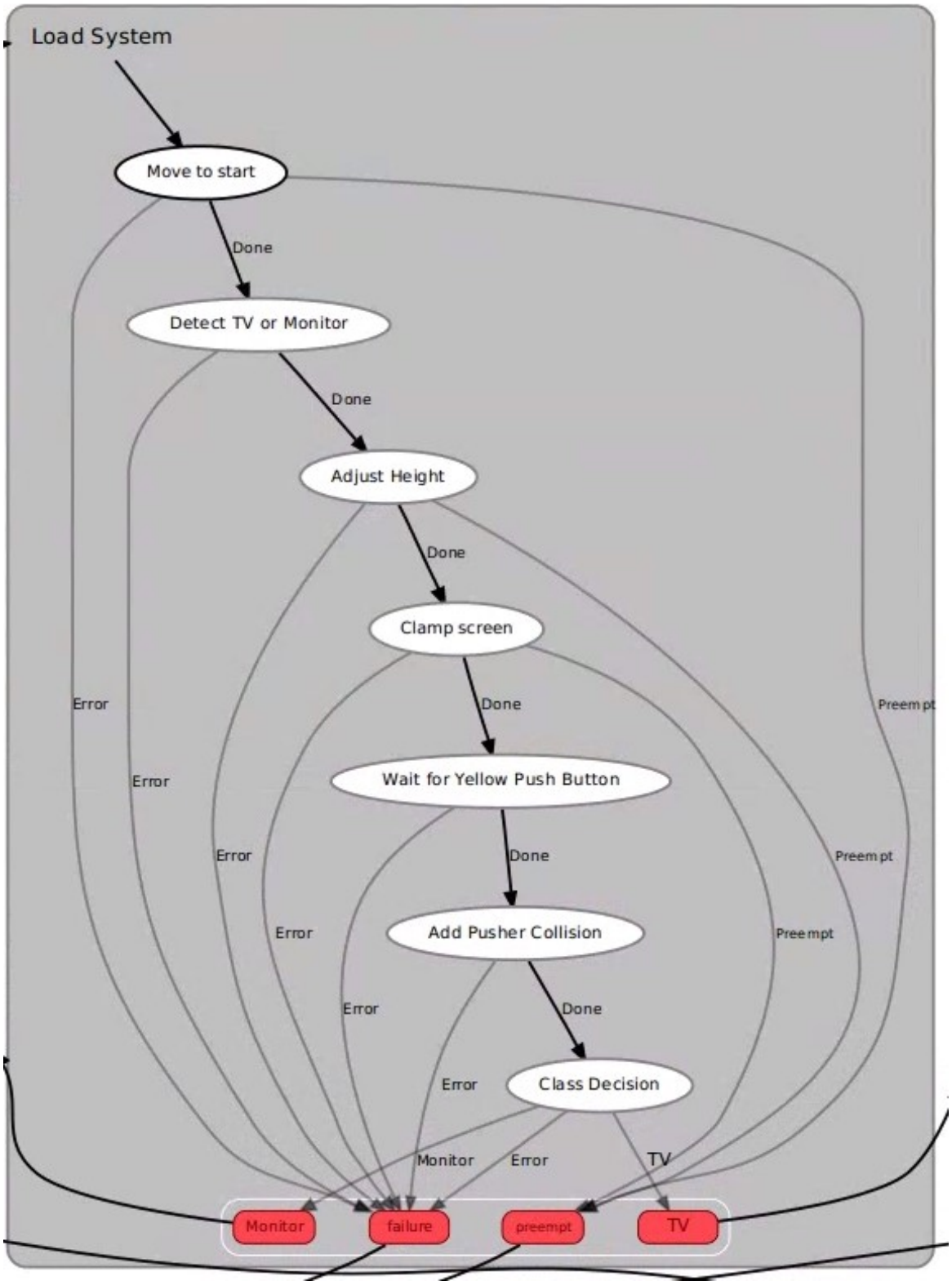

Load System
Move to start
Done
Detect TV or Monitor
Done
Adjust Height
Done
Clamp screen
Done
Wait for Yellow Push Button
Done
Add Pusher Collision
Done
Class Decision
Monitor
Error
TV
Error
Preempt
Monitor
failure
preempt
TV

Figure 18 - Load System State Machine Diagram

3.3.4.1 Detect TV or Monitor

This state makes a call to a service, which runs the regional convolutional neural network to be described in Chapter 4. This network is run on a color image from the robot mounted Kinect V2 sensor and detects the location and class of the FPD loaded in the fixture or on the conveyor belt. The bounding box co-ordinates of this FPD and its class are loaded and exported to the state machine for use in later states. If this state is successful it transitions into adjust height. If not, it returns an error and so does the loading state machine.

3.3.4.2 Adjust Height

This state is responsible for adjusting the base fixture plate so that the surface of the FPD is at the same height at the hard stops. This will increase the depth at which the tooling can reach without reducing the clamping force from the clamping cylinders.

The height to adjust the system is found by looking at a small region in the depth image from the Kinect V2 mounted on the robot's end effector. The location of the smallest depth location is transformed into a point in base frame. This point will be the highest location in this region of the image as the depth is relative to the camera location which is looking down at the fixture. The location of this will typically be on the bezel of the FPD.

By comparing the height given and the current height of the fixture plate, the thickness of the FPD can be calculated. This is then used to determine the amount to adjust the axis. The axis is then sent a movement command.

If this operation succeeds, then we transition to clamp screen. If it fails or is preempted the full state machine returns a failure or preemption result respectively.

3.3.4.3 Add Pusher Collision

The pushers are the plates used by the clamping cylinders to press against the hard stops. Prior to this state there is no collision geometry for these plates or the cylinders in which they are attached. The goal of this state is to add collision geometry for the flexible pushers. This collision geometry ensures that during

cutting and gripping operations the robot does not jog into these pushers, potentially causing serious damage to the EOAT.

This could potentially be done by tracking their position via linear encoders and adding the geometry based on the encoder position. This would be very easy to do if the pushers were controlled via motors with feedback but since they are pneumatic and have a very long stroke (52") this would be an expensive endeavor. Due to the difficulty of tracking the pneumatics via physical feedback, vision was chosen as the solution to find the ends.

The vision solution has been implemented via two different methods. The method starts by using vision to find the locations of the pusher plates in an image. The center of the pushers in the image are then correlated with a rectified depth image and transformed into a 3D point. These 3D points are used to insert a rectangular box which is the same height and width as the pusher plates that extends past the plates to cover the pneumatic cylinders.

The two vision methods being used to find the pushers are template matching and image segmentation. Template matching uses scaled templates and attempts to find regions that are similar to the images found in Figure 19. This method is sped up by binarizing the images prior to template matching turning yellow regions white and others black. The template matching is run at multiples scales from 0.8 to 1.2 with steps of 0.1. The other vision method used is image segmentation. In this method the deep learning techniques to be described in Chapter 4 are used to create a segmented image. The image is binarized, so the fixture components class pixels are white, and the other classes are black. The two largest regions are chosen as the fixture components. The method that is currently used is template matching.

(A) (B)

Figure 19 - Fixture Plate (Pusher) Templates (A) Left Plate (B) Right Plate

3.3.4.4 Class Decision

This state uses the detection information from the Detect TV or Monitor state and is essentially a switchboard for the state machine. If a Monitor was detected in the previous state, it returns Monitor and if a TV was detected it returns TV. This causes the full loading state machine to return either Monitor or TV ensuring that the proper process is used for dismantling.

3.3.5 Process TV

The Process TV state machine contains two states the first is TV screen removal. This state is responsible for removing the screen of a TV. This then leads into CFL removal. This state can be run either once or twice depending on system configuration. The remove CFL state also has two different versions one which utilizes line detection and an adaptive threshold function to find the CFLs and another which utilizes image segmentation and line detection. Figure 20 shows a mid-level representation of tis state machine.

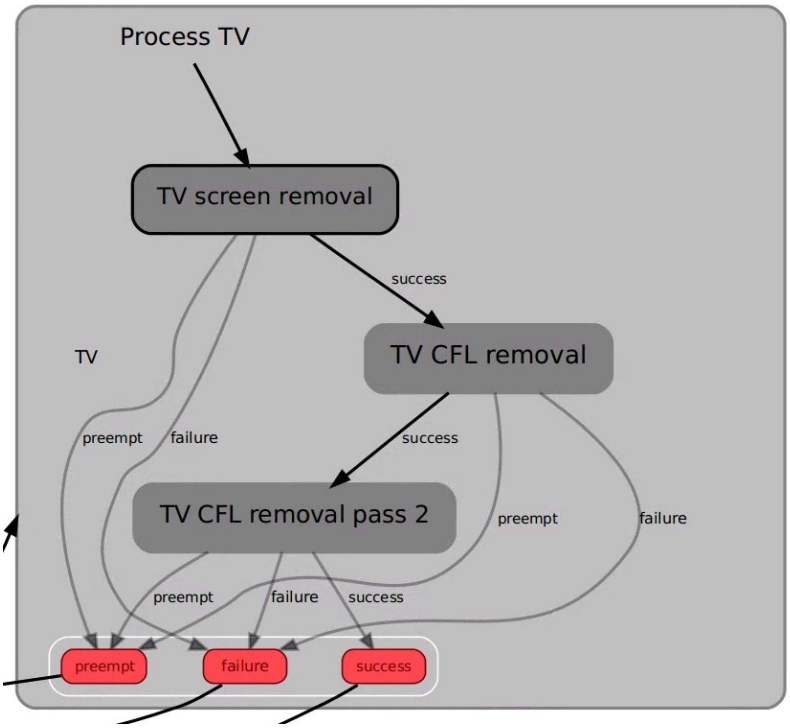
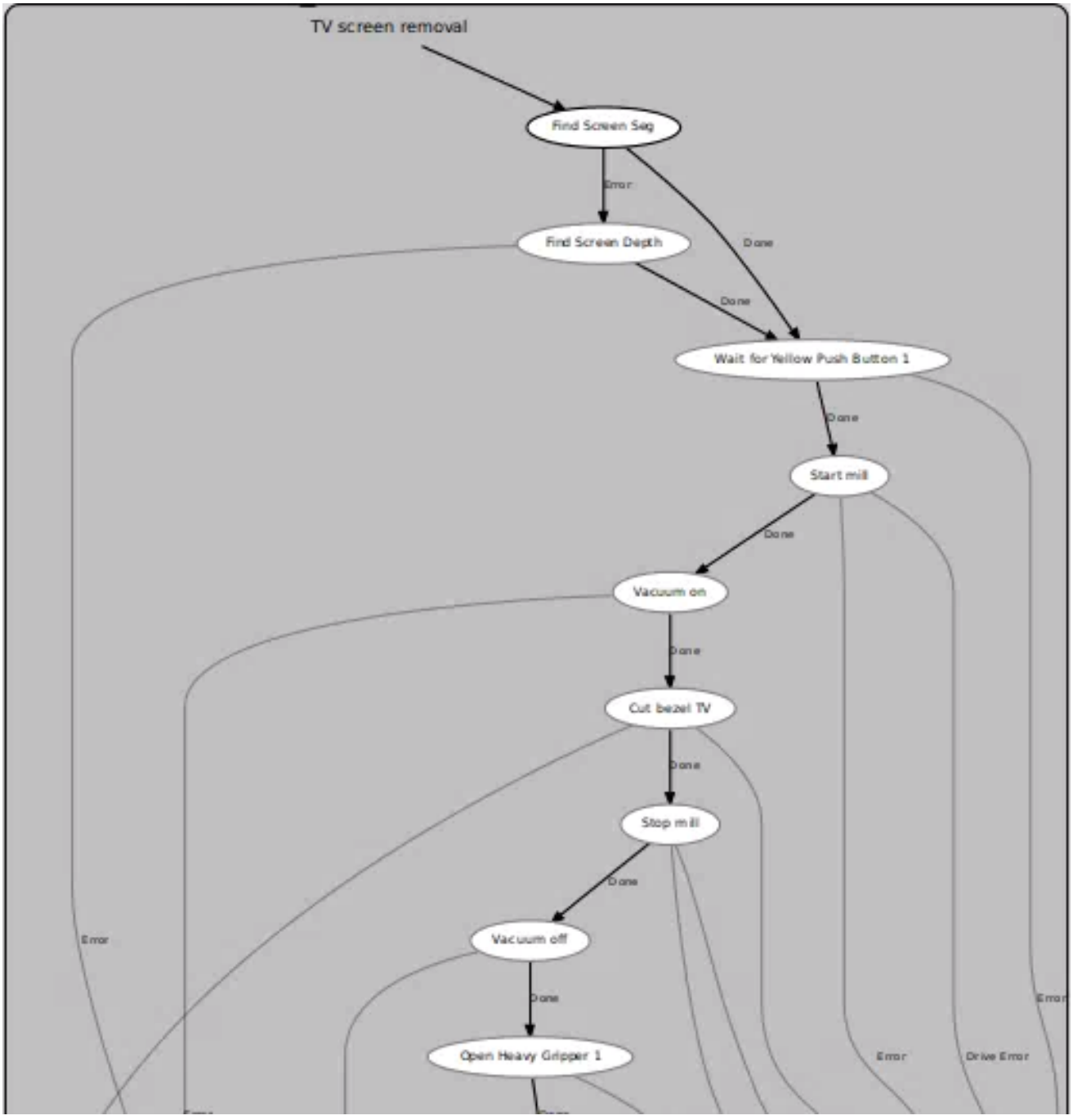


Figure 20 - Process TV State Machine Diagram - Mid Level View

3.3.5.1 TV Screen Removal

The TV screen removal state machine is one of the largest and longest state machines in the system but that is mainly due to the number of IO states. These IO states are described in Section 5.1. TV screen removal is a crucial first step in FPD processing. This operation must be done to reveal the CFLs for CFL removal. If any states return error or preempt values this main state also returns preempt or error. The transitions are shown in Figure 21.


TV screen removal
Find Screen Seg
Error
Find Screen Depth
Done
Done
Wait for Yellow Push Button 1
Done
Start mill
Done
Vacuum on
Done
Cut bezel TV
Done
Stop mill
Done
Vacuum off
Done
Open Heavy Gripper 1
Error
Error
Drive Error
Error

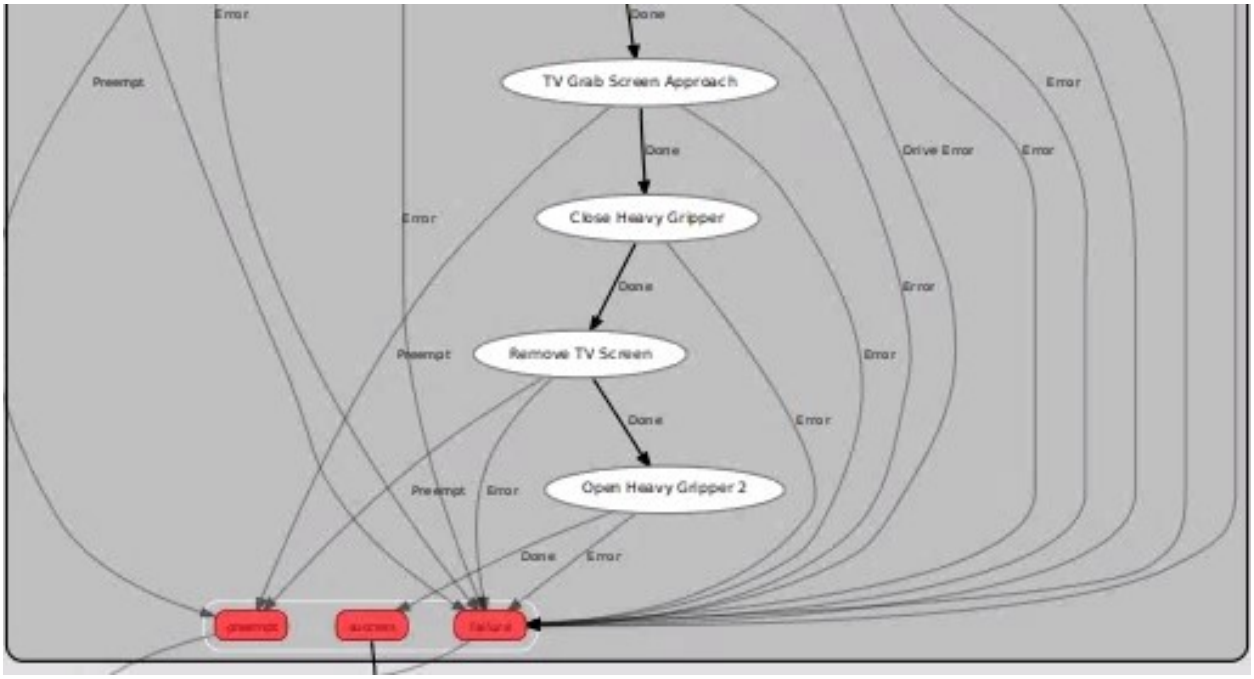


Figure 21 - TV screen removal State Machine Diagram

3.3.5.1.1 Find Screen Seg

This state is described in common states. In this state the input parameters of inner offset, and number of retries are set to five and two respectively. The reason for this offset is that the state that will use this information will be cutting the screen not the bezel. The main concern with this cut is whether the mill will hit the stainless steel or aluminum backing. The Kinect V2 can see through the screen and detect the depth of the backing. The points are offset to be inside the bezel-screen boundary. This is done as to ensure that the found depth is that of the screen and not the bezel. If this state is successful within the allotted two attempts, the found contour is transferred to a variable in the state machine. The state machine then transitions to Start Mill or Wait for Yellow Push Button. This then transitions into Cut Bezel TV. Upon failure the state machine transitions to Find Screen Depth to attempt finding the screen with a different algorithm.

3.3.5.1.2 Find Screen Depth

This state is described in common states. In this state the input parameters of inner offset, pixel window, and number of retries are set to five, a rectangle with bounds of 700 to 750 in X and 330 to 350 in Y, and two respectively. Aside from this, it does the same as Find Screen Seg. Upon failure, this state and TV Screen Removal return an Error value. If this state is successful within the allotted two attempts, the found contour is transferred to a variable in the state machine. The state machine then transitions to Start Mill or Wait for Yellow Push Button. This then transitions into Cut Bezel TV.

3.3.5.1.3 Cut Bezel TV

This state takes place after the mill has been started and the vacuum has been turned on. This state moves the end mill along a specific cutting path. The cutting path is described in Algorithm 5 and the points are taken from a previous find screen state. This state machine treats each of these movements as a different planning task. As such, the state machine may partially run the cutting path prior to discovering a point is in collision. This may reduce machine efficiency but should aid in manual dismantling of the FPD if the process fails. If this task succeeds, it saves the second last position in movement and exports it into the state machine for use as the pick position. The state machine will then transition through Stop Mill and Stop Vacuum to TV Grab Screen Approach. If there is an error or preemption the state machine returns error or preemption as well.

The path done in this ensures that all except for the final cutting path moves towards the hard stops on the table. This is done to attempt to reduce vibrations and to ensure that the FPD doesn't move due to the clamping cylinders being overpowered. The final cutting path creates a space for the heavy gripper to move between the layers of the screen and filters to remove them.

Algorithm 5 - TV screen cutting path

move to top left
plunge cutter
cut to bottom left
cut to bottom right
move to safe
move to top left
plunge cutter
cut to bottom top right
move to safe
move to midpoint of bottom and top right
plunge cutter
cut toward center and down slightly to make room for gripper
move to safe

3.3.5.1.4 TV Grab Screen Approach

This path approaches the saved position from the previous state angling the gripper by 30 degrees from the horizontal and approximately half a meter above and 10 cm towards the outside of the monitor. The gripper than moves down and towards the opening cut in the screen. The gripper is then moved so that it can grab the layers of the screen. If it is successful in this motion and not preempted the state machine transitions through closing the heavy gripper and to Remove TV Screen. If this state is preempted or has an error the state machine returns these results respectively.

3.3.5.1.5 Remove TV Screen

This is the final state in screen removal. This state does a linear motion towards the center of the screen when the screen is being gripped by the heavy gripper. This motion rips the screen from the TV. The heavy gripper then moves up to a safe position and does a free motion to the screen drop off position. If this is successful, the gripper is then opened and the screen is deposited in the screen drop off location. The TV screen removal state machine is then complete and returns a successful result. If this state is preempted or has an error the state machine returns these results respectively.

3.3.5.2 TV CFL Removal – Image Segmentation

This processing technique relies upon image segmentation and line detection to find CFLs. The strength of this technique is that if the neural network is properly trained the technique is very quick and accurate. The basic process being done is the detection and removal of CFLs. The CFLs can either be free floating due to damage to the FPD or be connected to one or either side of the FPD. The breakdown of the TV style FPD can be found in Chapter 1. If a CFL is connected it needs to be broken on that side so it can be removed. If no CFLs are detected the system moves on to the next FPD.

The main downside is that the reliance on image segmentation. If the network does not give proper results the system will not function. This can be avoided by having a large dataset of segmented images with CFLs but labelling CFLs is a very labor-intensive task. They are small and require fine image annotation. As this technique uses a segmented image other information can be utilized such as where the screen backing is and if there is any part of the screen left over from the previous operation. The line detection technique is also able to detect small CFL segments which if they are not close to the edge of the screen the system can pick them up without breaking the edge of the bulb. The image segmentation network, if properly trained, will also be far more robust than other potential detection methods, potentially being able to find the bulbs even with debris from the screen cutting operation in frame. This process is also run two times in case any bulbs were dropped or broken during removal. The transitions in this state are shown in Figure 22.

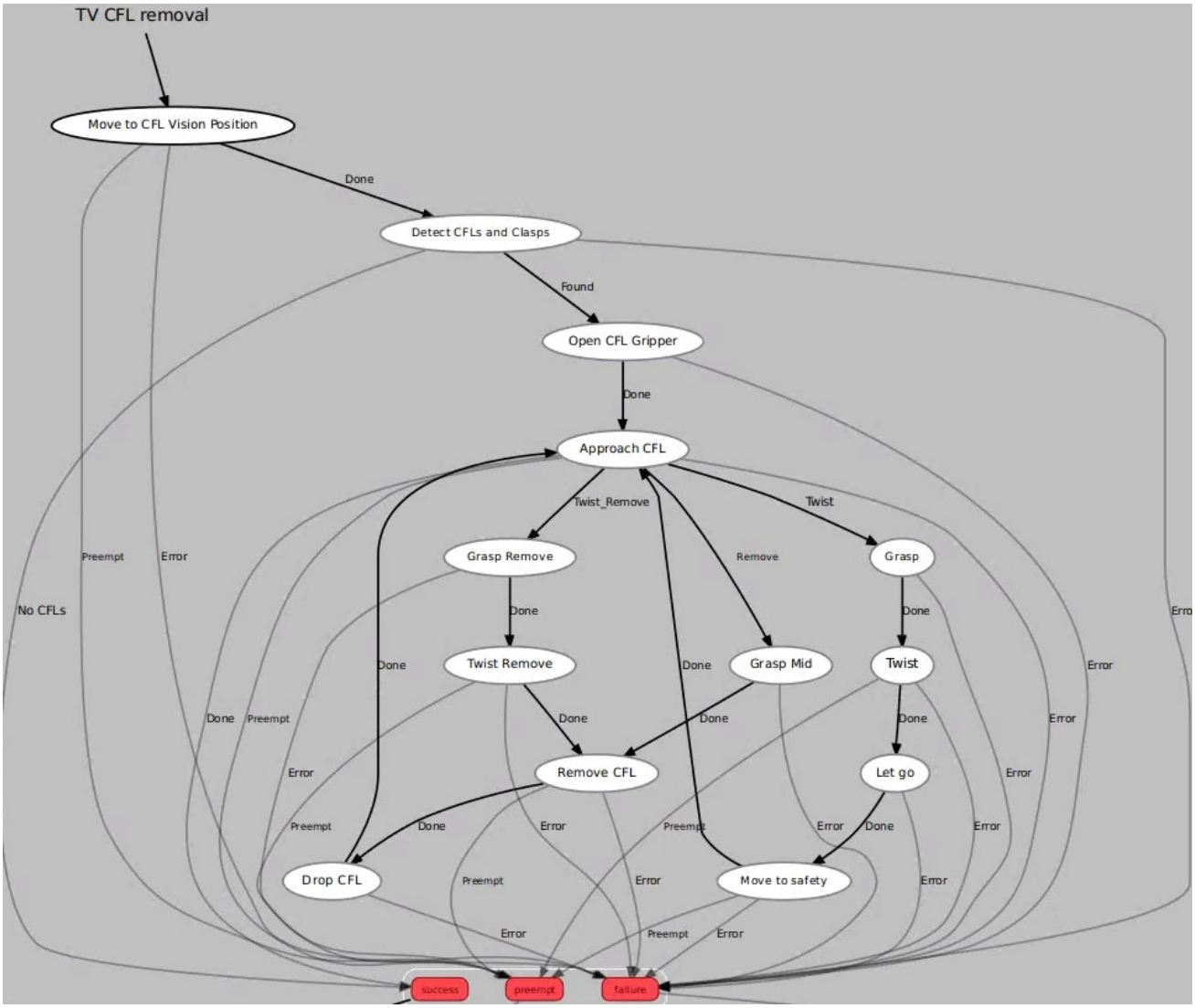


Figure 22 -TV CFL removal – Image Segmentation State Machine Diagram

3.3.5.2.1 Detect CFLs and Clasps

The name of this task is not fully descriptive. Though it is stated as Detect CFLs and Clasps, only CFLs are found. The reason for this name is to keep it constant with the name of a similar task in adaptive thresholding. This makes it easier swap these tasks at runtime.

This task utilizes a depth image and segmented image to find CFLs. The basic process is show in Algorithm 6. The system requires two input parameters the first being the number of attempts before returning a failure, and the second being the contour used to remove the screen. If any of the steps in the system fail, it will try again until it is successful. The screen contour is post process transformed detections as there cannot be any CFLs outside of the screen region. The output of this state is an array of CFL detections, and a connection array. The connection array says whether the CFL is close to the edge of the screen and will have to be broken on that end prior to being removed.

Algorithm 6 shows the basic process but several steps required further explanation. Step three uses probabilistic line transformation. This algorithm has five parameters including distance resolution, angular resolution, threshold, minimum line length, and maximum line gap. These five parameters have a value of 1 pixel, 0.5 degrees, 300, 50 pixels, and 20 pixels, respectively. All distance parameters are given in pixels. These parameters were tuned via experimentation.

Step four is filters the detected line segments by looping through them and comparing the bounds of the detected line segments. This filter process works by looping through the detected line segments and comparing them to a list of unique line segments. The list of unique lines is initialized with the first line segment in the list of detected line segments. This comparison checks for three things. If the unique line is expanded does the other line fall inside a minimum fitting rectangle made by that line, if the other line is expanded does the same happen, and are the corners of the line segment within 20 pixels of each other in X and Y. When the lines are expanded, they are expanded away from their midpoint in X by 20 pixels and in Y by 4. Figure 23 shows this expansion and the bounding rectangles and examples of lines which would be removed. If any of these statements are true, the longest line segment is kept. If none of them are true, then it is compared to the next unique line. If the list of unique lines is exhausted, then the line is added to the list of unique lines. The unique lines are then the final filtered output.

Step seven creates an array which predicts whether the CFLs need to be broken prior to removal. It does this by comparing the points of the CFL line segments to the detected screen backing rectangle. The

screen backing rectangle is reduced by ten percent of its size along the longest size and if any of the points of the detected CFLs are outside of this rectangle a value of true is set to the connection array for that point of that CFL. This array will be used to optimize processing later on in the system. It is assumed that these areas are connected as the bulbs have connection points on either end and always run along the longest side of the TV so the segment is close to that end it is likely still connected and has yet to move.

Once this process is complete the CFL gripper is opened to make the system ready to start removing bulbs and then the system transitions to Approach CFL. If there is an error in this process the full state machine returns an error.

Algorithm 6 - Find CFL via image segmentation

Loop number of attempts

1) Get Segmented and Depth Images

2) Create Binarized CFL Image where CFL class is 255 and all other classes are 0

3) Find line segments using Progressive Probabilistic Hough Transform algorithm

4) Filter and combine line segments keeping only the longest lines and removing overlapping

5) Create Binarized Screen Backing Image where Screen Backing class is 255 and all other classes are 0

6) Find rectangular region of screen backing – Finds contour and uses contour approximation

7) Create connection array to tell the system whether the bulb's ends need to be broken using rectangle

8) Remove all CFLs detected outside of screen region from the list

9) Use Depth Image to transform points to 3D space

10) Remove all CFLs outside of region detected in screen removal state machine

11) Return success

Return failure

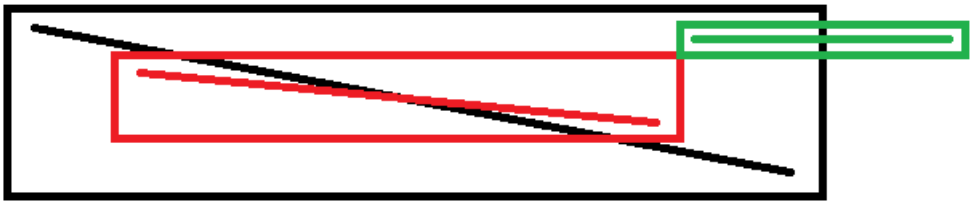


Figure 23 - Example of filtering technique where the boxes are the area being checked where the black line is from the unique line list, the red line would be filtered out, and green line would not

3.3.5.2.2 Approach CFL

This is the main control center of the CFL removal process. It is responsible for the first approach of the CFL gripper to the bulb and tells the system which process to take when removing it. The general approach path is free motion planning to a safe point approximately 0.5 meters above the pick point and then a linear move down to the pick point. This approach path is used for all CFL approaches.

This state starts with the first CFL in the list and looks at its connection array. If neither of the sides are connected the state approaches the midpoint of the CFL bulb using the standard approach path, increments the bulb counter, and returns a Remove result. After this result is returned the gripper is closed, the robot moves up in Z by half a meter before moving to the CFL drop off position and opening the CFL grippers. If the connection array only has one of the sides connected the state approaches that side of the CFL bulb using the standard approach path, increments the bulb counter, and returns a Remove_Twist result. After this result is returned the gripper is closed, the robot twists its end effector to break the bulb, moves up in Z by half a meter before moving to the CFL drop off position and opening the CFL grippers. If both the sides are connected the system checks whether the increment state machine variable is true. When the state machine is first run this is set to false. If it is set to false then increment is set to true, the robot approaches the left side and returns a Twist result. After this the system closes the CFL grippers, twists, opens the grippers and moves up in Z by 0.5 meters. If increment is set to true, increment is reset to false, approaches the right side of the CFL bulb using the standard approach path, increments the bulb counter, and returns a Remove_Twist result. After this the standard Remove_Twist actions are run. In all of these instances after the other tasks are run the state goes back to Approach CFL. If the bulb number is

greater than the number of detect bulbs, then a Done result is returned, and the TV CFL Removal state machine returns success. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.

3.3.5.3 TV CFL Removal – Adaptive Threshold

This processing technique relies upon adaptive thresholding and line detection. This technique relies on the bulb being darker than the CFL backing. This allows the adaptive thresholding function to determine which areas are CFLs and which are the backing. Unlike image segmentation, this system only detects full CFLs and not small sections. Once the CFLs are detected they are broken on both sides and removed.

The main downside to this approach is that adaptive thresholding relies on the CFLs being the darkest part of the monitor. Since during normal operation some of the screen material that was milled may have fallen into this area and the screens are a darker color than the CFLs this can cause false detections and cause the CFLs to bridge from one to another. Due to this line segments are not found but full lines. This makes this system incapable of finding broken small segments. It is also slower and less reliable than the image segmentation-based system. This system does detect small clasps that hold the CFLs unlike the segmentation-based system. These clasps are then removed via a milling operation. This is necessary as if the CFLs break due to being connected to these clasps they cannot be salvaged in a secondary processing step. Figure 24 shows the transitions in the state machine.

3.3.5.3.1 Detect CFLs and Clasps

This state does two different tasks the first is clasp detection and the second is CFL detection. Clasp detection uses template matching and a standard clasp shape. This process scales the input image between 0.7 and 1.1 times with a step of 0.1 when attempting to detect the template. This process only works on clasps with a specific shape and should be replaced with a deep learning process in the future.

To detect CFLs the algorithm in Algorithm 7 is used. The algorithm works by using adaptive thresholding to binarize an image. A quarter of each image is then removed. This is done to remove the change of extra components on the outside adding to lines. This is very helpful in reducing false positives especially on TVs with front facing speakers. Speakers will commonly have perforated areas so that sound

can propagate these can lead to lines being detected on angles and in regions that are not associated with CFLs.

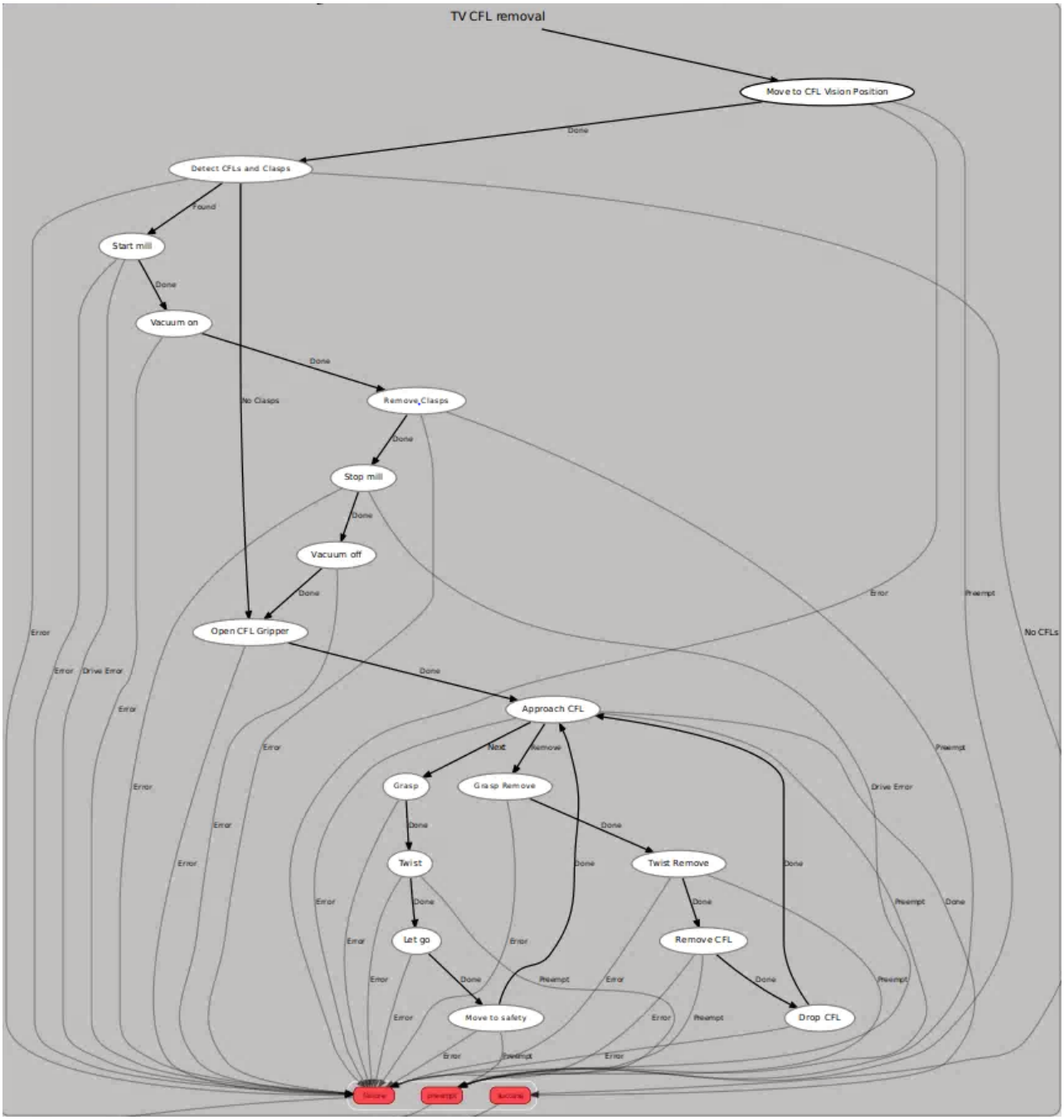


Figure 24 - TV CFL removal – Adaptive Threshold State Machine Diagram

After the image has been prepared lines are detected and filtered into bulb detections. The filtering process is done as follows. First all lines that do not have an angle within 3 degrees of the horizontal are pruned from the list. This is done as the larger TVs can only be mounted the long way in the fixture and as the CFLs are always in one direction they should be roughly horizontal. The CFLs are then grouped into bulbs. This is done by sorting through the filtered list of lines and finding groups of two lines that have the exact same angle and are between 15 and 5 pixels apart. This is the average distance that a bulb can be and less than the typical distance between bulbs in an image. Bulbs are then put into a list and compared to create a list of unique bulbs. The bulbs are compared by their midpoints and if a bulb in the list of unique bulbs midpoint is within 5 pixels of another bulb it is ignored. This list is populated on a first come first serve basis. This is then run a total of 5 times and with each unique list being added to a total detection list each cycle. When adding to the total detection list the same uniqueness check is performed. Once this is done the bulbs midpoints are found and transformed to a point and a pose is found by taking the current pose of the mill tool and rotating it by the angle of the bulb. The rotation ensures that the X axis of the pose is along the bulb and the Z axis is down. This is then the final output list of bulb detections.

When this state is successful if clasps have been detected it transitions into Start Mill, Vacuum on and then into Remove Clasps. If there were no clasps detected it transitions immediately into Approach CFL. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.

Algorithm 7 - Find CFL via adaptive thresholding

Loop number of attempts

1) Get Color and Depth Images
2) Convert color image to grayscale and use adaptive thresholding to binarize the image
3) Remove one quarter of image from either side along X axis
4) Find lines using Hough Transform algorithm
5) Filter detected lines into bulbs
6) Add bulbs to total bulbs list

No bulbs in list return failure

Transform total bulb list into list of joint poses where point of the pose is the center of the bulb and the orientation quaternion is the angle to pick and return success

3.3.5.3.2 Remove Clasps

Prior to this state the Mill and Vacuum on were run. The goal of this state is to move the mill approximately 0.3 meters above the surface of the clasp and plunge the cutter into the clasp. This state does linear moves from one clasp to another and removes each of them. Once this is complete it transitions to Stop Mill and Vacuum off before going to Approach CFL. If there are errors in the state or if any of the motions are preempted/aborted the state returns failure and preempt respectively.

3.3.5.3.3 Approach CFL

As this state relies on finding full lines and not segments there are no lengths for the bulbs defined when they are found. As such, it cannot be inferred whether the any side of the CFL tubes are connected. Therefore,, both sides are twisted to break them. Each bulb in this is defined as a single pose since there is no length defined so to find the ends of each bulb the screen size is used. The bulb positions are found by offsetting the given pose by its own X axis by half of the width of the screen minus two centimeters. The approach path is the same path utilized in the image segmentation version, the tool frame used is the CFL gripper. If increment is false, the bulb approaches the bulb pose with an offset in the X axis. It performs free motion to a safe approach position, which is 0.3 meters above the offset bulb pose in Z in the base frame before performing linear motion to the bulb. This state then returns “Next” once this is done the following states grasp the bulbs, run a twist motion, releases the bulb, and then returns to a safe position by running linear motion upwards by 0.5 meters. When this is done increment is set to true and we transition back to Approach CFL. If increment is true, the offset bulb pose is offset along the negative X axis by the same amount. This state then returns “Remove”. The next states then grasp the CFL, twist, perform a linear move up in the Z axis to remove the bulb, perform free motion to the CFL drop position, and finally drop the CFL. When the CFL is dropped, the bulb number is increased, and increment is set to false. The state machine then returns to Approach CFL. If the bulb number is larger than the number of bulbs detected the state returns “Done” and the full state machine returns success. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.

3.3.6 Process Monitor

The monitor process is very similar to the TV process. There are however, some key differences between the states. The cutting process for the monitor does not cut the screen but the bezel holding in the

screen. This is due to the way the monitor style is built. The CFLs are to the side of an acrylic panel that is behind the screen. The bulbs are below the bezel and the acrylic panel is held in by a small amount of stainless steel that is also below the bezel. To remove the acrylic panel and get to the CFLs the bezel must be removed. Since the acrylic panel must be removed, we also need to modify the screen removal states to handle the acrylic panel as well.

For screen removal two methods have been implemented the first uses the mill to remove the panel, screen and filter all at once. The other uses the heavy gripper and mills in a step to allow the gripper to grasp the acrylic panel, screen, and filters at the same time. Both of these removal techniques find the screen the same way. Due to the change being part way through the Monitor screen removal state that state has been split into three headings; Find and Cut Screen, Screen Removal – Heavy Gripper, and Screen Removal - End Mill. The full monitor state machine is shown in Figure 25.

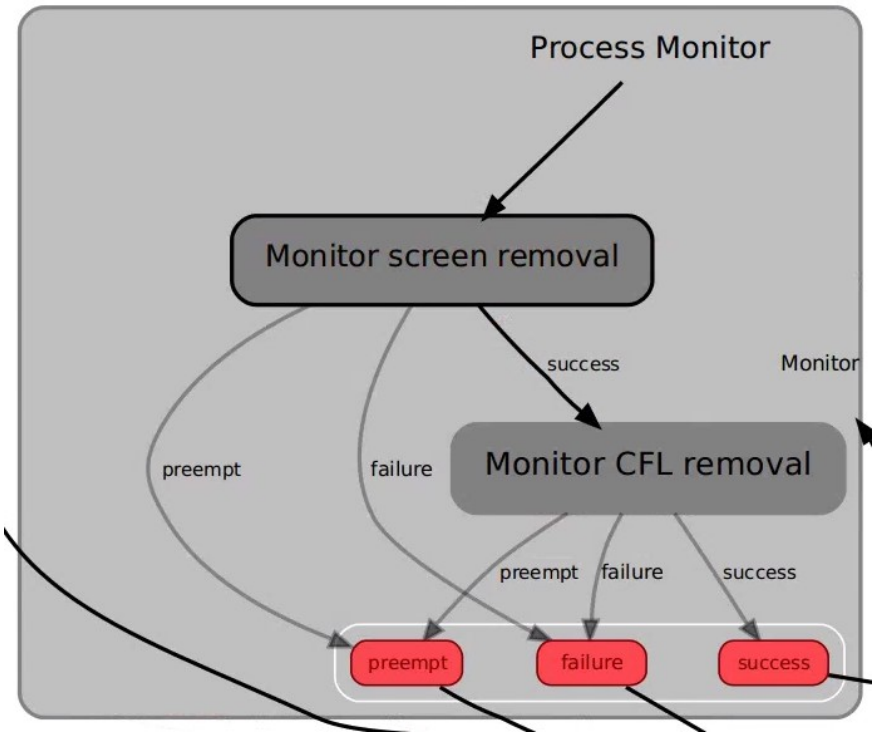


Figure 25 - Process Monitor State Machine Diagram - Mid Level View

3.3.6.1 Find and Cut Screen

The find screen state is the same as it is in the TV state machine except for some minor changes in the state parameters. The order has also been changed. This is done as Find Screen Depth tends to work better on Monitors than Find Screen Seg. This is likely due to the dataset in the image segmentation network containing more TVs than Monitors.

3.3.6.1.1 Find Screen Depth

This state is described in common states. In this state the input parameters of inner offset, pixel window, and number of retries are set to negative five, a rectangle with bounds of 700 to 750 in X and 330 to 350 in Y, and two, respectively. The negative offset is done to expand the found boundary to land on the bezel. This ensures that the cut depth is proper and does not go too far below the surface of the bezel. If this offset was not used the state could potentially cut too deep and cut into the bulb. This would spread the mercury content from the bulbs around the internals of the screen defeating the purpose of the system. If this state is successful within the allotted two attempts, the found contour is transferred to a variable in the state machine. The state machine then transitions to Start Mill or Wait for Yellow Push Button. This then transitions into Cut Bezel. Upon failure, the state machine transitions to Find Screen Depth to attempt to find the screen with a different algorithm.

3.3.6.1.2 Find Screen Seg

This state is described in common states. In this state the input parameters of inner offset, and number of retries are set to negative five and two respectively. Aside from this, it does the same as Find Screen Depth. Upon failure, this state and TV Screen Removal return an Error. If this state is successful within the allotted two attempts, the found contour is transferred to a variable in the state machine. The state machine then transitions to Start Mill or Wait for Yellow Push Button. This then transitions into Cut Bezel.

3.3.6.1.3 Cut Bezel

This state is responsible for the milling operations required for FPD monitor screen removal. Though it is called Cut Bezel more than that takes place. The beginning process is the same whether the heavy gripper or end mill is used to remove the screen. The first phase cuts the bezel with a cut depth of 1

centimeter below the bezel height. When using the mill to remove the screen the bezel all cuts are done towards the hard stops to increase stability. Once the bezel is cut the mill is then plunged in the center of the screen. If the heavy gripper is being used several other cuts are performed the purpose of these cuts is to provide clearance and a ledge to allow the heavy gripper to grasp the acrylic panel, screen, and filters. These cuts include a clearance cut in X this cut is approximately 5 centimeters in length. Once this cut is done the mill moves back to the position it was before making the clearance cut and rotates 30 degrees around the Y axis. The mill not moves 1 centimeter in negative X creating a small angled ledge. The mill then moved back 1 centimeter in X, un-rotates, and moves to safe. This process is shown in algorithm 8. If this is successful it returns success. If the heavy gripper is being used to remove the screen the state machine then transitions to stop mill, vacuum off, and if it is using the heavy gripper to remove the CFL the heavy gripper is opened, and approach remove screen is run. If the mill is being used the state machine still turns off the mill and vacuum but then transitions into Remove Screen. If there were any collisions, errors, or if the motion takes longer than anticipated the state returns an error result and so does full state machine. If the state is preempted, then the state machine returns a preempted result and so does the Process Monitor state machine.

Algorithm 8 - Monitor bezel cutting path

```
move to top left
plunge cutter
cut to top right
cut to bottom right
move to safe
move to top left
plunge cutter
cut to bottom left
cut to bottom right
move to safe
move to screen midpoint
plunge to create area to grab - stay in plunged area
if using heavy gripper
        cut towards x
        move back to previous position
        rotate cutter about Y axis
        cut in negative x
        move back to previous position
        rotate back about Y axis
        move up in Z to safe position
```

3.3.6.2 Screen Removal – Heavy Gripper

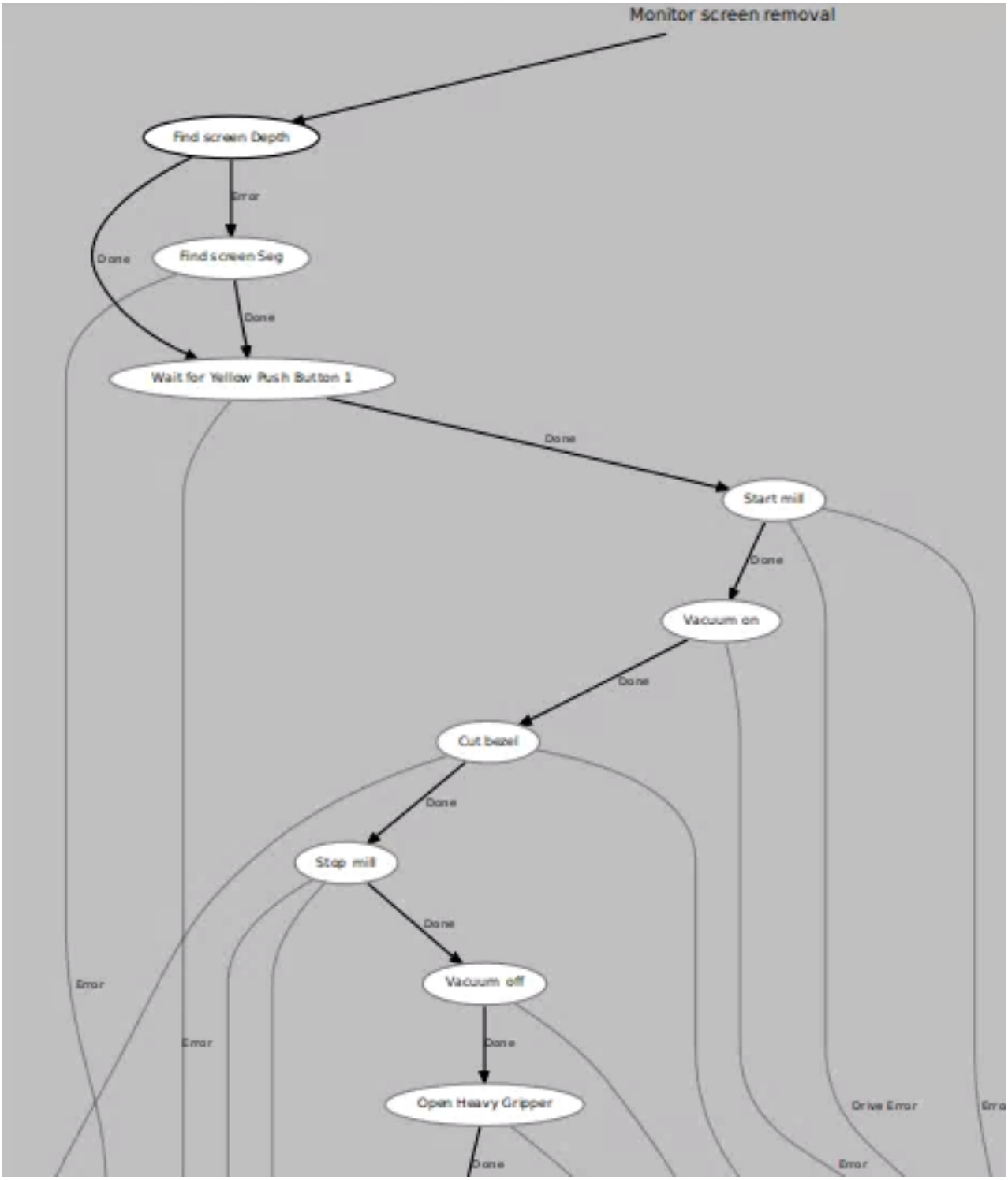
This state uses the heavy gripper and a ledge that was milled into the acrylic panel. The strength of this approach is that it will cause minimal damage to the end mill but due to the ledge being angled it is possible that the heavy gripper may slip out from the area. To try to mitigate this it is jammed into the ledge which leads to another issue. It could get stuck on the gripper. There is no way to ensure that this doesn't happen.

3.3.6.2.1 Approach Remove Screen

This state is essentially just an approach state. It starts by bringing the heavy gripper in line with the X axis and rotated 45 degrees along the Y axis and offset two centimeters in X and 30 centimeters in Y from the position in which the mill was plunged to create the clearance cut and ledge. While keeping this orientation the heavy gripper performs a linear move down to the screen height and then another linear move one centimeter in the x axis. This will bring the gripper around the ledge milled in the previous state. Once this is complete if there were no errors or collision in the movement this state returns a "Done" result and transitions into closing the heavy gripper and then into Remove Screen. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.

3.3.6.2.2 Remove Screen

When this state starts the screen will be in the grasp of the heavy gripper. The heavy gripper is then jogged in the X axis by two centimeters and in the Z axis by 40 centimeters. The X axis motion is towards the ledge and is added to reduce the chance of the gripper losing hold during screen removal. Once this is done the state moves to the screen drop off position via free motion. If this is completed successfully the state returns "Done" and transitions into Drop Screen which is just an Open Heavy Gripper state. This drops the screen and completes the Monitor screen remove state machine with a success result. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.


Monitor screen removal
Find screen Depth
Error
Done
Find screen Seg
Done
Wait for Yellow Push Button 1
Done
Start mill
Done
Vacuum on
Done
Cut bezel
Done
Stop mill
Done
Vacuum off
Done
Open Heavy Gripper
Done
Error
Error
Drive Error
Error
Error

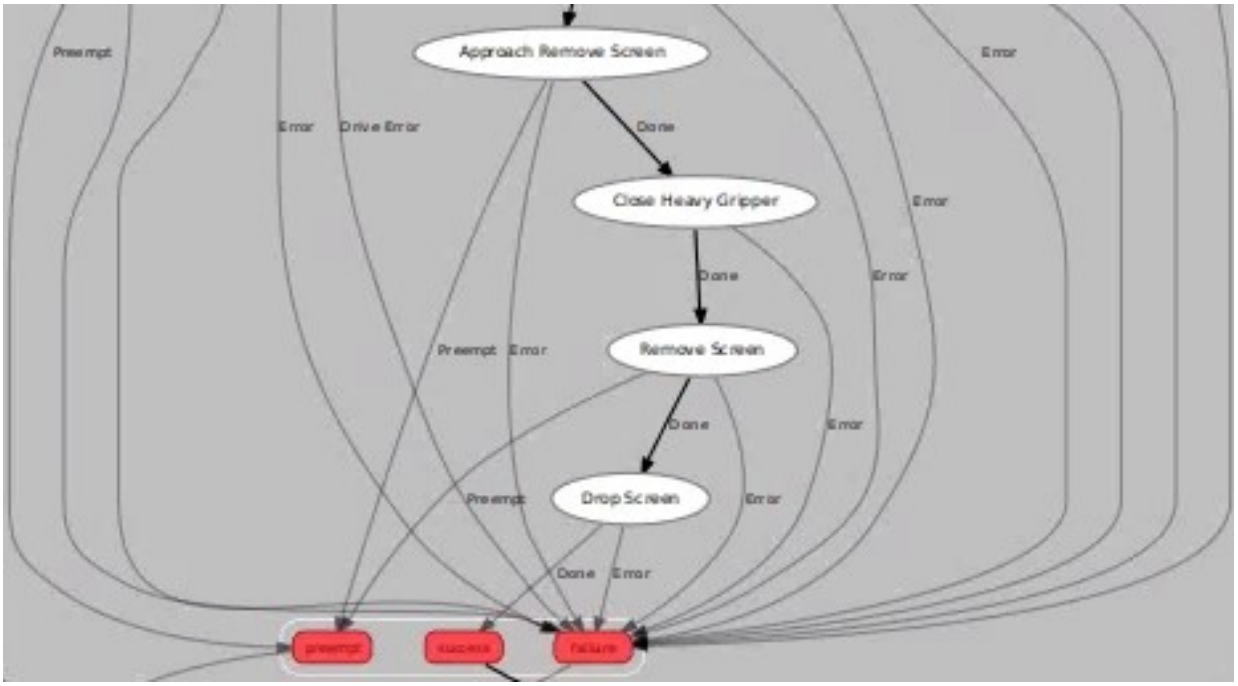


Figure 26 - Monitor screen removal – Heavy Gripper State Machine Diagram

3.3.6.3 Screen Removal – End Mill

The process to remove the screen with the end mill is done in the Remove Screen state. When this state starts the mill had just finished cutting a hole in the screen and is still inside it. The mill and vacuum have also been turned off. This stat starts by pushing down by 2 millimeters to flex the screen backing and ensure that the mill is below the acrylic, the mill then rotates by 20 degrees in about the X axis while moving the screen up in the Z axis by 2.5 centimeters. This should jam the mill into the edge of the acrylic and pull it out from the FPD. The mill then performs free motion to the screen drop off position. Once this is done the end mill is vertically oriented and the system shakes it off by moving quickly up and down in the Z axis by two centimeters two times. This state relies on the bezel being almost completely removed as the friction from the mill will not be enough to overcome the full bezel. The friction works well due to the roughing surfaces on the end mill. This method can potentially cause damage to or break the end mill, so it is not the preferred method of screen removal. Figure 27 shows the state machine transitions in this state.

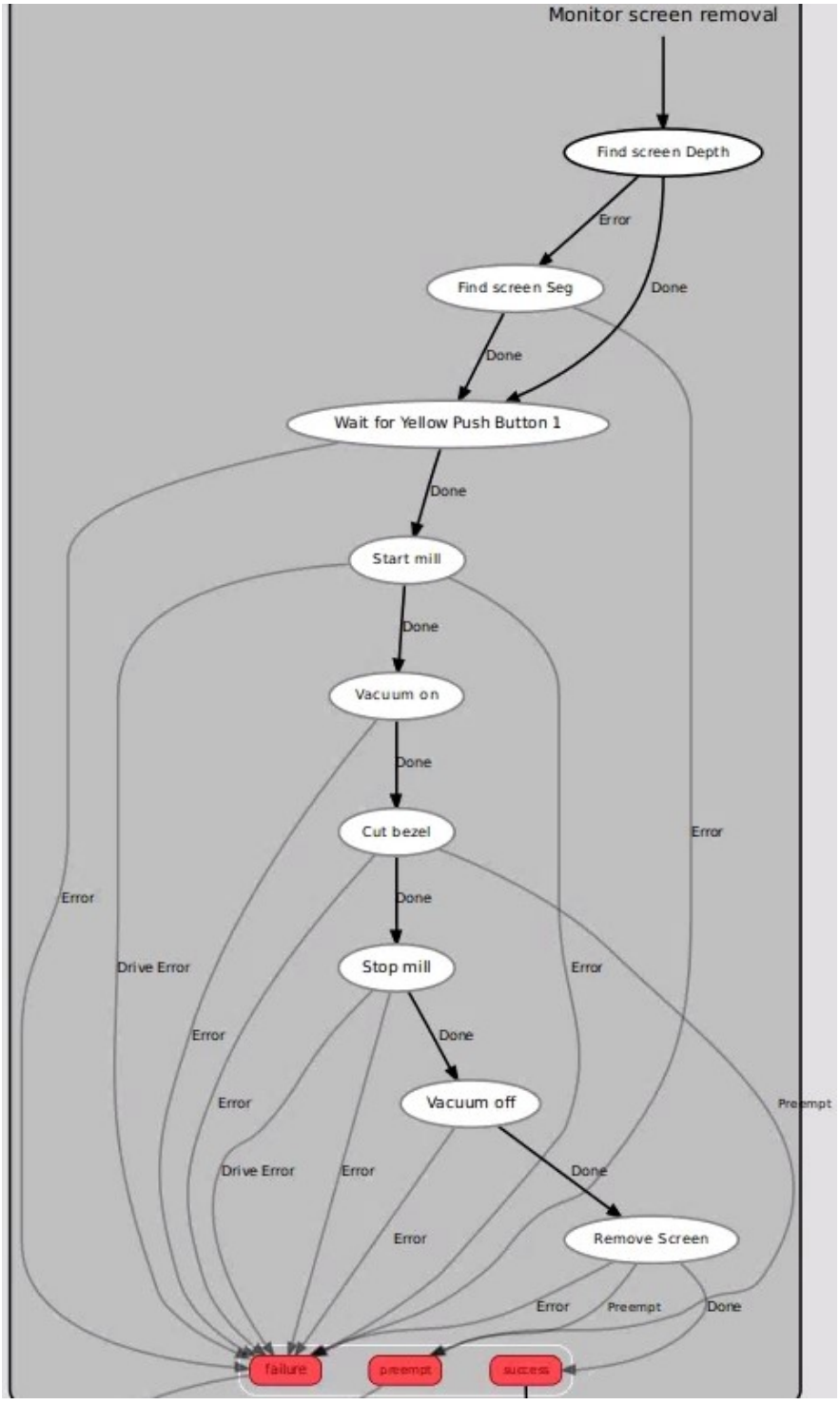


Figure 27 - Monitor screen removal - End Mill State Machine Diagram

3.3.6.4 Monitor CFL Removal

This state always attempts to remove the lighting on the top and bottom of the screen. As there are only two bulbs stored inside some reflective metal foil around the end of the acrylic panel, it is more efficient to remove the bulbs even if they are not CFLs then to check if they are CFLs fist. The CFL removal state has no variations and always runs the same tasks. The state diagram for this state machine is shown in Figure 28. In this system the any states ending with 1 or 2 are the same as each other but they are processing a different CFL. This is set up in CFL Decision and is controlled by a hyperparameter in the state initialization. Move to Start is used to put the robot EOAT into a good position to approach the CFLs.

3.3.6.4.1 CFL Decision

This state is responsible for setting up the information needed for the following states in the Monitor CFL Removal state machine. This system starts by taking the screen removal points used to mill out the bezel in the previous state and finds the midpoints of each of the edges of the bezel. Once the midpoints are found they are exported to the state machine and it then checks which is the longest edge. When the longest edge is found, that edge and the edge opposite of it are chosen as the CFL edges. In monitors the CFLs are always along the longest side. The state machine then exports the indexes at which the midpoints of these sides are stored. When this state is done it transitions to Move to Start 1 upon success and then to Approach CFL 1. If there is an error the state and Monitor CFL removal both return failure.

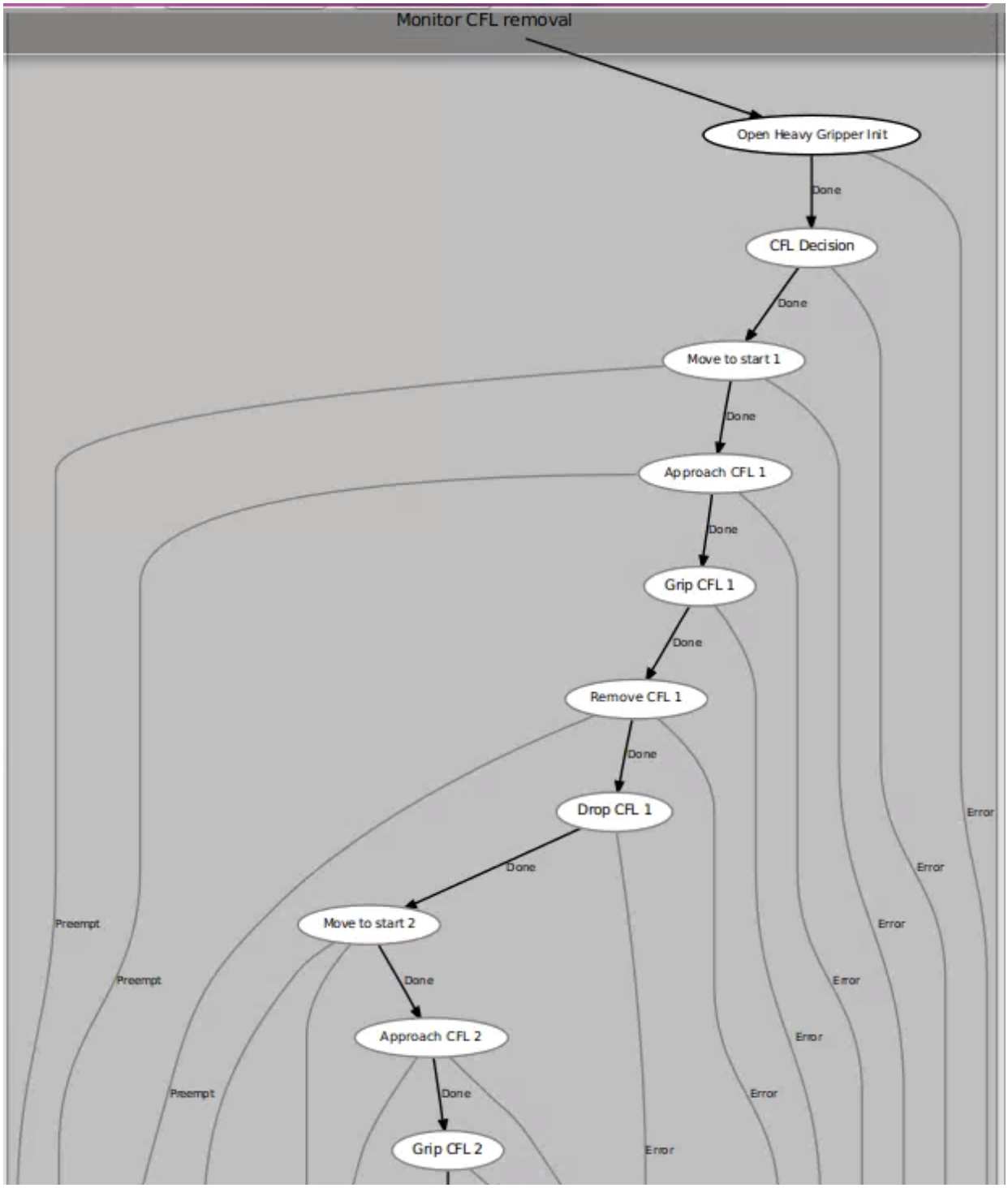
3.3.6.4.2 Approach CFL

The approach CFL state contains a parameter that tells it which index to use when selecting the CFL midpoint. This is the only difference between Approach CFL 1 and Approach CFL 2 and this is what causes them to grab different CFLs. Aside from this the state is a simple lookup. As the hard stops are in a constant spot and each of the corners and thus the sides are sorted, it is known which side of the monitor each index will be at. Since this is know we can use a static angle and offsets for each side. The basic movement is to rotate have the heavy gripper rotation be such that the X axis of the heavy gripper is aligned with the CFLs long side and rotated 35 degrees about that axis. This orientation is used and a standard approach path is then taken. This approach path starts by moving the heavy gripper with free motion to a point with an offset of 1cm toward the inside of the monitor and 0.4 meters up in Z from the CFL midpoint. The heavy gripper is then moved down to just 1cm above the bezel height, and then moved to the CFL

midpoint with linear motion. If this is done successfully the state machine either transitions to Close Heavy Gripper and Remove CFL 1 or 2 depending on which approach state is run. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.

3.3.6.4.3 Remove CFL

The Remove CFL state is just two linear motions followed by a free motion path. The two linear motions are one movement in the negative Z axis of the heavy gripper of 0.1 meters and relative move from that position up in the base Z axis of 0.4 meters. These motions are to rip the CFL out of the monitor and to move the EOAT into a safe position for motion planning respectively. The free motion is then a movement to the CFL drop off position. Once this is done the heavy gripper will be opened and the CFL removal will be complete. If this is Remove CFL 1 the state machine will then transition to Move to Start 2 and remove the next CFL and if this is Remove CFL 2 the full CFL removal will be complete and this state as well as the Monitor CFL Removal state machine will return success. If there are errors in the state or if any of the motions are preempted the state returns failure and preempt respectively.


Monitor CFL removal
Open Heavy Gripper Init
Done
CFL Decision
Done
Move to start 1
Done
Approach CFL 1
Done
Grip CFL 1
Done
Remove CFL 1
Done
Drop CFL 1
Done
Move to start 2
Done
Approach CFL 2
Done
Grip CFL 2
Preempt
Preempt
Preempt
Error
Error
Error
Error
Error
Error
Error

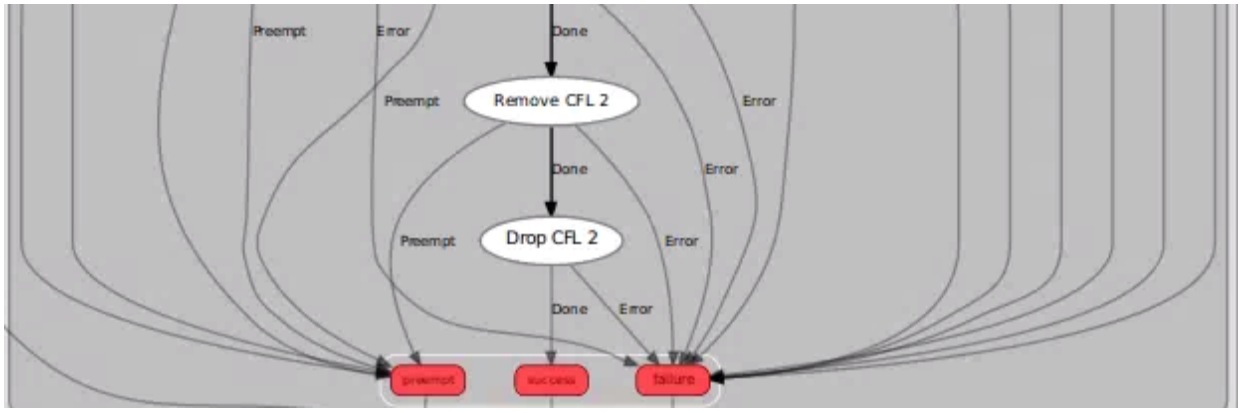


Figure 28 - Monitor CFL removal State Machine Diagram

3.3.7 Unloading

The Unloading state is responsible for removing an FPD when it has been fully processed. This state is very similar to the Recovery state but it assumes that everything is still functioning properly. It is of note that the Move to vision position state is the same state as Move to start. The name was changed so that it would make more sense in the context of this state machine but the state is functionally identical.

3.3.7.1 Unload

The unload state is the key and active state in this state machine. The first action this state does is jog the fixture axis to its maximum vertical height lifting it 1.2 meters up and just above all of the items on the table including the hard stops and cylinders. Once this is done the center point of the FPD is used to create a path to push the FPD off of the fixture. The center point is found using information from one of two sources. The first source is used only during the first unloading attempt and it is the found screen boundaries. The second source is used if unloading has failed previous and it is object detection boundaries from Detect TV or Monitor. Once this is done the location of the FPD found from Detect TV or Monitor is used. The center point is used to attempt to stop the FPD from rotating when being pushed off. From the center point only the X value is used.

The path the robot does to push off the FPD is as follows and is done with the heavy gripper tool frame. Free motion to a pose 0.3 meters above the surface of the fixture axis, with an X value equal to the X value of the midpoint and a Y value of -0.3 in the base coordinate frame the orientation is such that the

heavy gripper Z axis is facing the base frame's negative Z axis and the end mill Z axis is pointing in the base frame's negative Y axis. All of the linear motion is done in this orientation. The EOAT then moves to the same pose but with a Zvalue of one centimeter above the surface of the fixture axis via linear motion. The EOAT then does linear motion to a across the Y axis to a value of -1.5 meters.

This path should push the FPD off of the surface of the fixture to the side of the table.

3.3.7.2 Detect TV or Monitor

This state works the same as the Detect TV or Monitor state in loading but with an extra step. Instead of outputting the bounding box in image co-ordinates they are first transformed to global co-ordinates before being exported. This is done to allow the system to detect the FPD and export the bounding box values for use in the Unloading state if it needs to be rerun. This should enable the unload state to properly push off the FPD in the second try.

3.3.7.3 Check Success

This state exists for two reasons. Firstly, it checks the results of the Detect TV or Monitor state to see whether the FPD was properly unloaded. This allows the Detect TV or Monitor state to be a more general state that simply runs and exports the neural network results making it more useful and reusable for future work. Secondly, this state also looks at a parameter in the system to see if it is being requested to shutdown. If there are no detected FPDs from the previous Detect TV or Monitor state then the unloading is considered a success. If the state was successful it can take one of two paths. If the system is being requested to shutdown the system returns a "End" result and entire state machine finishes with a success result. This means that the system would need to be relaunched. This is the only exit point in this state machine aside from the same state in the Recover state machine. If the system was not requested to finish it will return "Done" transition to Home Table when this is done the Unloading state machine will finish with a "success" result and transition back to Initialize System. If an FPD was detected then a "failure" result is returned. This transitions the state machine back to unload and the attempts to remove the FPD with the location data from Detect TV or Monitor.

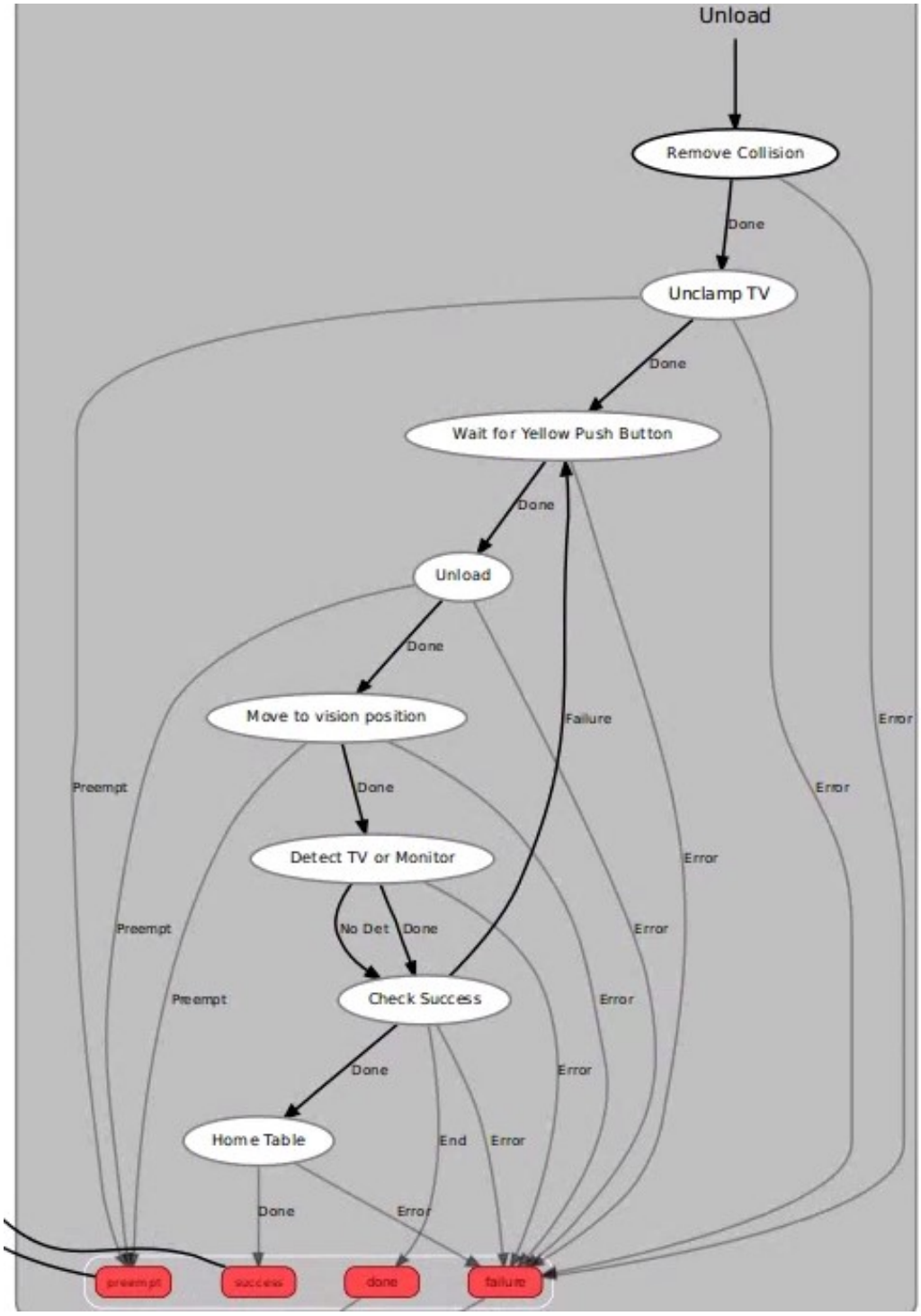


Figure 29 - Unload State Machine Diagram

3.3.8 Recovery

The Recovery state machine is essentially an Unload state machine with Check Robot, Reset Robot, and Move Safe states added to the beginning. The purpose of this state is to recover from robot errors or failures in the state machine. The extra states added on to the beginning of the state machine are responsible for this. The Check Robot and Reset Robot states check for any faults in the system and attempt to remedy them and the Move Safe command moves the EOAT away from the fixture to ensure that the rest of the operation can be completed successfully. In the current implementation it pushes the FPD off the same way unload does with no indication that the unloaded FPD was a failure to the operator. If a conveyor is added in to remove the FPDs this will need to be updated to include an indication of success or failure. This would likely be either a light or if automation is desired it can switch swap the conveyor path so the FPD is diverted to a secondary processing location. This is the reason why the Recovery state is not a small reset state machine that leads into Unload. It was left separate so these extra pieces can be added on a later date. This state machines state diagram is shown in Figure 30.

Most of the states used in this state machine are explained in the Unloading state machine or common states. It is of note that Move to Vision position is the same as Move to Start and the Detect TV or Monitor is not the same one used in Loading but is the Detect TV or Monitor state used in Unload.

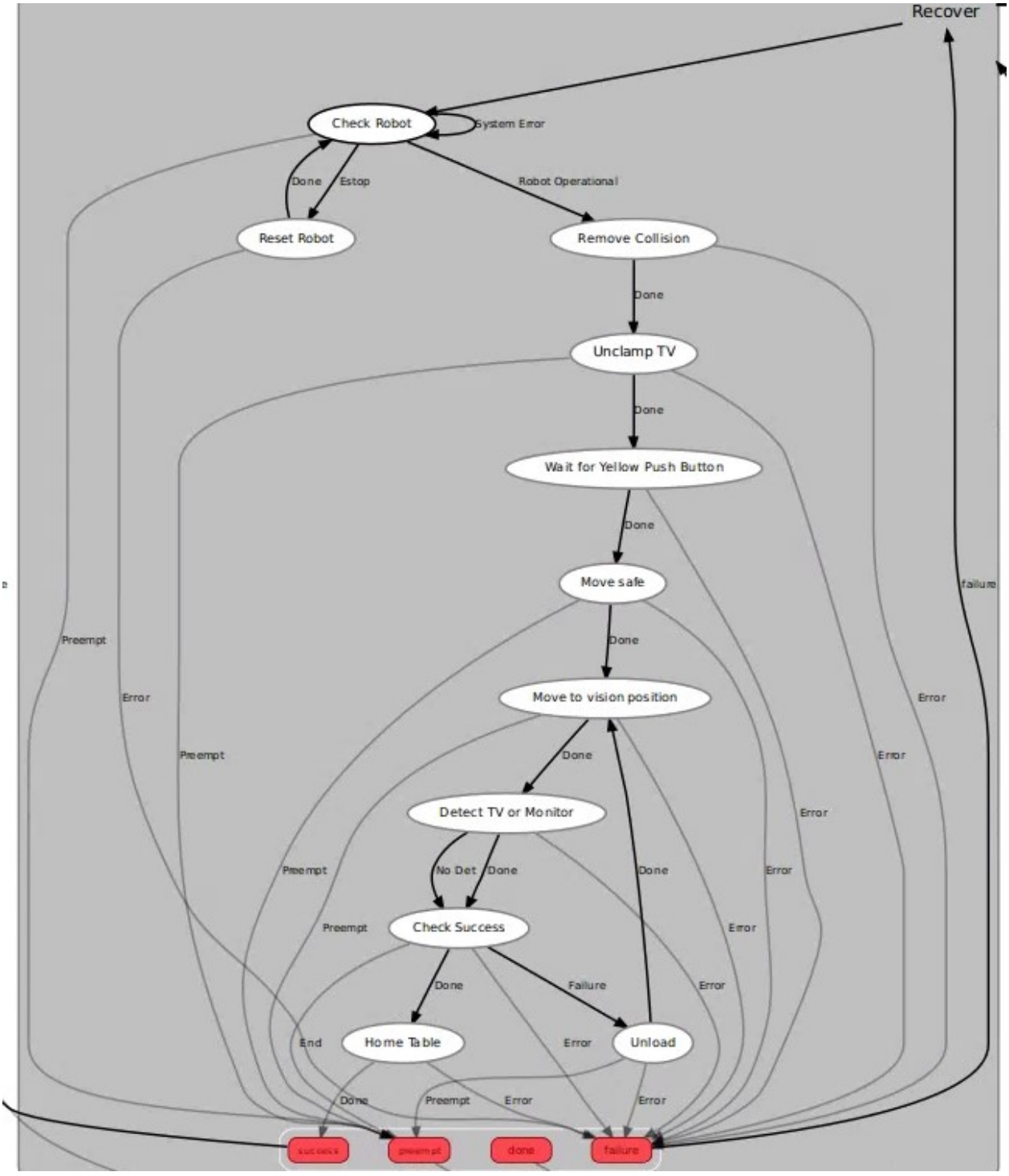


Figure 30 - Recovery State Machine Diagram

3.3.9 Semi-Implemented States

The states are areas where certain sub states and some of the background work has been implemented but the final state is not yet complete or tested. These states would be nice to have but could require significant changes to the system in order to implement.

3.3.9.1 Conveyor Loading

Most of the background of this state has been implemented. We are currently able to detect FPDs and could potentially control a conveyor by using a digital or analog output to move the system. The key thing that is missing is the driver for the conveyor and a system to take FPD detections from the conveyor and say how long and at what speed to move the conveyor to have it fall on to the fixture.

3.3.9.2 High Value Component Removal

This state may not be feasible with the current fixture. The big issue with doing high value component removal is that there is currently no system to flip the FPD to the back. All of the circuit boards and high value components are on the back of the FPD. It is possible that the FPD could be refed into the system upside down but the component removal would take far too long to be economical for the system.

The states that have been implemented that could help with this mainly involved deep learning. The system can currently determine if an FPD is upside down which could be used to go into a High Value Component Removal state machine. The image segmentation networks have also been trained to find circuit boards. This can be used to find and remove the circuit boards. The process may require other detection networks such as screw detection and detection of mounting components as the current system would not be able to cut through a steel wall mount which are sometimes left on recycled FPDs.

3.3.9.3 Completion Checks

Completion checks can be implemented using the image segmentation networks. As we are able to find screens we can detect whether removal has been successful. Completion checks have been somewhat implemented via the secondary CFL removal state that is run when using image segmentation for CFL removal in Process TV. There are also potential checks that could be done at drop off locations when components are being removed to see if anything was removed or if the pieces have been dropped. This would help with Monitor screen removal where there is a potential that the acrylic panel could get stuck.

Chapter 4
Vision

4.1 3D Sensors

One of the key focuses in this project was using the correct technology for detecting the FPD. Accurate depth detection is crucial when using a milling tool to process FPDs. When comparing vision sensors, the key areas of interest are accuracy, repeatability, cost, and usability.. Usability is tied to how easily and how efficiently a sensor can be used. Does the sensor driver have features or built-in functionality that makes it easier to use? Does a driver exist, or does it need to be implemented from scratch? Will the feature type fit the application or are there any specific features about the system that help with the applications? These questions define usability.

4.1.1 Stereo

Stereo vision uses two color cameras and compares the images to find a depth map. These cameras must go through calibration where a known pattern is used, and pictures are taken simultaneously by both cameras. OpenCV [27] is being utilized for most vision applications and includes calibration and reconstruction tools. The algorithm used for 3D reconstruction is Stereo Processing by Semi-Global Matching and Mutual Information [28]. This technique operates with black and white images and the implementation used by OpenCV does not have fine grained steps. To reduce potential computation time, this implementation finds a certain number of regions of depth. The key issue with this is the lack of fine-grained control. Unlike the other types of 3D sensors, this leads to a potential that the bezel height and screen height will be grouped into the same region and a large number of regions will be found in the background instead of in locations that are of interest.

In all stereo 3D these systems it is assumed that if the pixels are the same color or intensity and in similar regions they belong to a single object. This can cause serious issues when attempting to find the depth information of FPDs surfaces, as they are commonly black on black. This can lead to serious issues with accurate registration. The depth information also become less accurate as the distance increases. As the range of FPDs can potentially be up to 60 inches in size and we need sub centimeter accuracy, this system was quickly determined to be unusable. As such no extensive testing on this method was done.

Stereo vision is straight forward to implement using ROS and OpenCV with multiple sensor systems and calibration tools already available for the system [29]. If the bezel and screen were not so similar in color this would be a serious contender for 3D sensor of choice just due to this ease of implementation.

4.1.2 Structured Light

Structured light is a technique that uses pattern projection and a separate camera or often two cameras to detect the depth [30]. Common examples of sensors that utilize this technique are the Microsoft Kinect V1 and the Intel Realsense line of sensors. For testing in this application both a Kinect V1, and Intel Realsense Z300 and D435 sensors were utilized. Both sensor types use a similar projected pattern, but other systems utilize different patterns types. The pattern used for the Intel Realsense D345 is shown in Figure 31. Google has an open source toolkit to allow users to use different patters such as lines and checkerboard projections, but this would require purchasing a separate pattern projector and color camera [31]. This would likely cost more than the Microsoft or Intel options and have less accurate results.

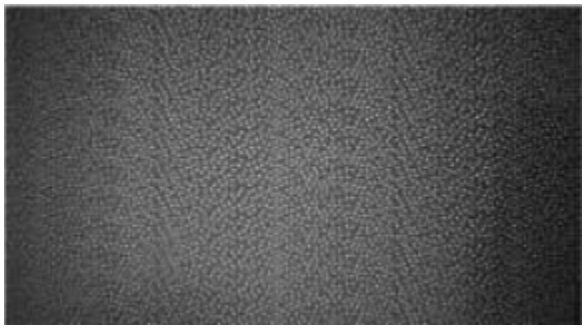


Figure 31 - Pattern Projected by Intel Realsense D435

The key difference between the Kinect V1 and the Intel Realsense line of sensors is the size and the sensor layout. Figure 32 shows the two sensors with labels on the cameras and projectors of each sensors. One of the key differences between the two sensor options is that the Realsense utilizes two IR cameras to detect the projected pattern. This leads to a much higher accuracy and higher resolution than the Kinect V1. Due to this increase in accuracy the Intel Realsense was tested and not the Kinect V1. There are some issues with this system as can be seen in Figure 33. The depth map is very noisy and has ripples propagating through it. This has been seen less with other models of the Intel Realsense such as the ZR300 but is still a concern. The ripples in the depth image could be mitigated by averaging progressive frames to

remove noise. This should be easy to do as the Intel Realsense cameras run at 60fps or more. The Intel Realsense D345 camera also has a high native resolution for a depth sensor of 1280 by 720 pixels. This is much higher than the Kinect V1s native depth resolution of 320 by 240 pixels. The Kinect V1 is no longer available but can be purchased for under $300 CAD while the Intel Realsense D345 camera costs around $250 CAD.

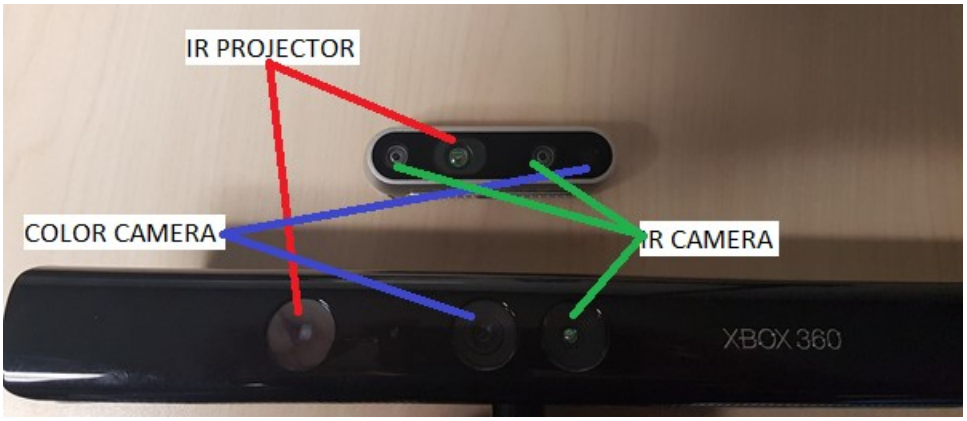
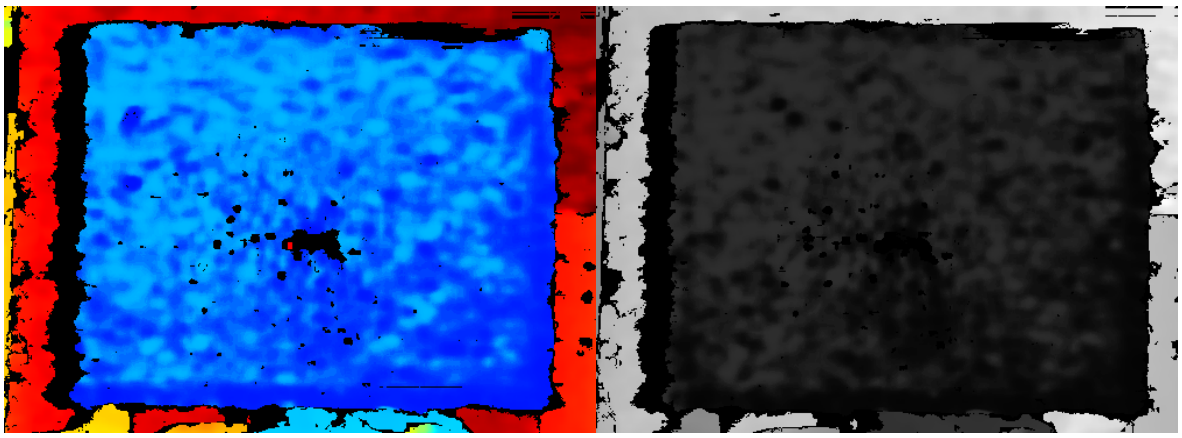


Figure 32 - Intel Realsense and Microsoft Kinect V1 Size Comparison and Sensor Indication

Another key issue is the reliance on a pattern being projected. Since the pattern is from a static projector and the camera has a fixed resolution the farther away the sensor is from the object it is trying to sense the less accurate the depth detection will be. Like with stereo vision this causes some serious concerns.

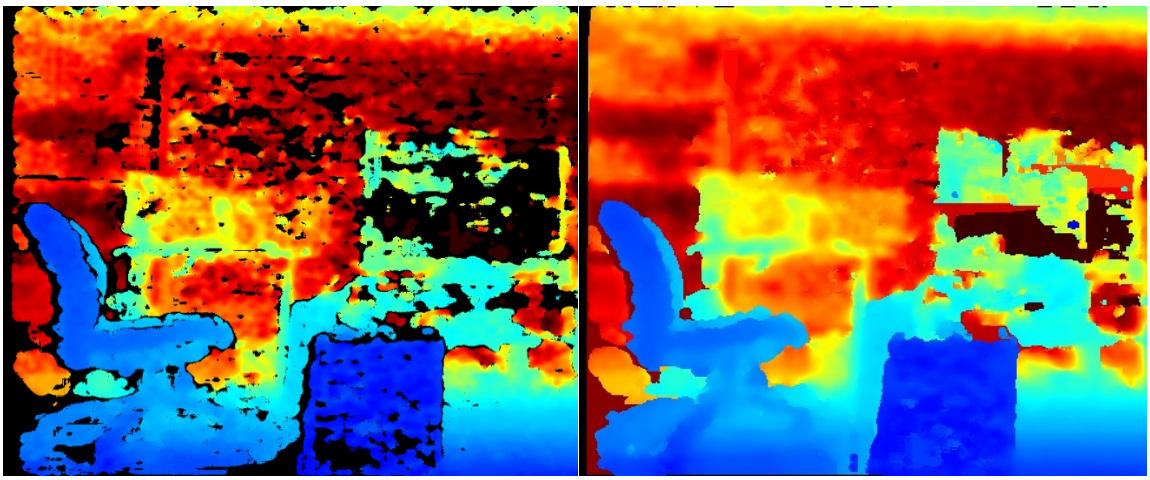


(A) (B)

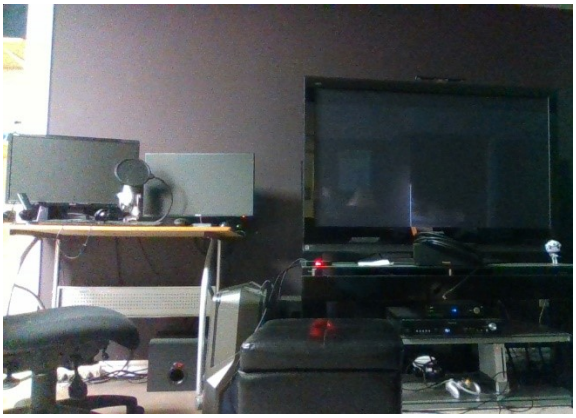
Figure 33 - Depth Image from Intel Realsense D345 with (A) Jet Coloring and (B) Mono

One of the strengths of this system is that drivers are readily available for both the Kinect V1 and Intel Realsense. The sensors are readily available and have been used extensively in systems made with ROS.

One of the key features that the Intel Realsense cameras has that makes it an appealing choice is that it has fine grained control of every part of the sensor including exposure, pattern projector intensity, white balance, saturation, gain, depth parameters, etc. This fine-grained control is not available on the Kinect V1 and many other sensor types. The Intel Realsense Application Program Interface (API) also has post processing algorithms available to help fill in gaps in the depth image, retain edges, and smooth images. Figure 34 shows the difference these filters can make. The filters also do not add a large amount of computation time with all of them reducing the frame rate from around 60 fps to 30fps which is still fast enough for use in Simultaneous Localization and Mapping (SLAM) and image capture applications. The results of the filtering are very promising, but this sensor works very poorly on reflective surfaces which can be seen by the reflection on the large TV on the left of the image. It does work well on matte materials as show by the chair.



(A) (B)



(C)

Figure 34 - Results of Intel Realsense Built-in Filters. (A) Unfiltered (B) Filtered (C) Color Image

4.1.3 Time of Flight

Time of Flight sensors use an invisible light source, commonly near infrared light emitting diodes (LED), and detect the amount of time it takes to return to the sensor. There are several common ways this is done such as RF-modulated light sources, range gated imagers, and direct time of flight [32]. The time of flight sensor tested in this system is a Kinect V2. This sensor uses a range gated imager. The way that this technique detects depth is by opening and closing the shutter at the same rate as pulses are being sent out. This method uses the speed of light to determine distance. Far away objects will appear darker than close objects as a portion of the light is being blocked by the shutter related to how far the pulse traveled.

The sensor then compares the amount of the light pulse that sent out and the amount that was received to determine the distance of the pixel. This causes some issues with reflective materials and materials that absorb near infrared light. If the exposure it to high bright sports will for in the IR image. Those bright sports will have a depth value of 0.

One of the strengths of this technique is that the Microsoft Kinect V2 and Time of Flight sensors is that they tend to have a very consistent error rate so is run in its typical range of operation. This leads to the sensor putting out less noisy and more consistent depth image.

There is also a strange feature of the depth gated time of flight implementation of the Kinect V2. This feature is that the depth camera is able to see through the screen of an FPD and detect the depth of the items behind it. Unfortunately, this is not accurate enough to see the CFLs behind the screen but it can be used to detect the boundary between the screen and the bezel of a monitor. Figure 35 shows a depth cloud output of an FPD. This shows the backing and bezel at very different heights.

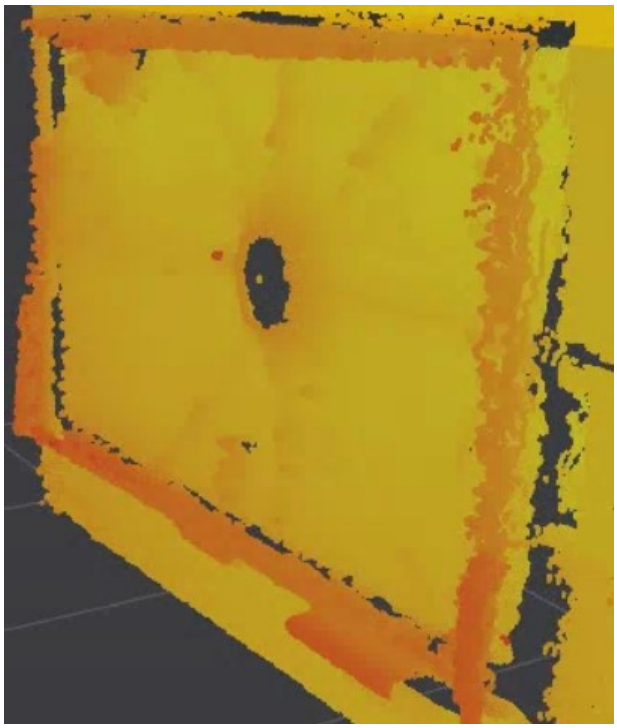


Figure 35 - Point Cloud from Kinect V2 of what? Front view of a FPD?

There are other time of flight depth sensors that are available aside from the Microsoft Kinect V2. Some of these are the Swiss Ranger, CanestaVision, and Basler ToF cameras. These systems work well but are not do not have as complete of a feature set as the Kinect V2. The Kinect V2 not only has a time of flight depth camera but also has a full HD color camera. The others only have depth sensors. There are also ROS drivers available that contain APIs for upsizing the depth image to the same resolution as the color images, rectifies the images, and contains calibration tools. This allows for quick setup and use of the Kinect V2. The Kinect V2 does however have a small native resolution of 512 by 424 pixels. This is why the Kinect V2 ROS driver has built in code to rectify and use bilinear interpolation to match the depth and color image scales.

4.1.4 Laser Line Scan

. During this project different laser line scanners were compared including ones from Riftek, and Gogater. These sensors all operate on a similar principle. There is a laser line projector on one side of the sensor which projects a line towards the subject and a camera at a known angle with a filter that only sees the color of the line projected. Since this distance and the angle of the sensor is known the height and location of an object can be inferred by the location of the line on the image [33]. These systems can either output a series of points or image information. General design of these scanners can be seen in Figure 36 which shows a potential laser line scanner; the Riftek RF625.

One of the upsides of this system is that the accuracy is very good at close and mid ranges with the Riftek RF625 having an error rate of plus or minus 0.2 percent of range. This means that at 2 meters away there is a potential error of plus or minus four millimeters.

There are some key issues with this sensor type. The first is that the either the sensor or the fixture needs to be moved across the laser line generator in order to scan the component. The potential laser line scanner for this project has points per profile of 1280. If a standard 1280 by 1024 image is required, since the scanner can take measurements at a rate of 500 Hz, it will take over 2 seconds to scan the system. This depth sensor also does not come packaged with a system to rectify it to a color image. As such, the usability of the system is greatly diminished as not only does it have the slowest framerate, but a lot of development time would need to be spent to create a driver that would turn the output of a laser line scanner into an image format that can be rectified to the a color camera.

Figure 36 - Riftek RF625

4.1.5 Comparison

The final chosen sensor was the Microsoft Kinect V2. This was chosen mainly due to the noticed ability for the sensor to see through certain FPDs. Though this does not work with all types of FPDs it was prevalent and useful enough to force the decision. In addition, the sensor tends to be accurate within a centimeter at any depth and is available for a reasonable price. Unfortunately, this sensor may need to be replaced as Microsoft stopped manufacturing the Kinect V2 in early 2017. Currently the only way to purchase these sensors are on the secondary or used market.

Table 4 shows a breakdown of the potential sensors and some of their key features. The accuracy of each sensor type was previously discussed in their sections

Type	Specific Sensor	Depth Resolution	Frame Rate (fps)	Other Sensors	Connection Type	Approximate Cost
Structured Light	Microsoft Kinect V1	320 by 240	30	Color Camera with resolution of 640 by 480	USB 2.0	250 USD Not currently manufactured
Structured Light	Intel Realsense D345	1280 by 720 pixels	90	Color Camera with resolution of 1920 by 1080	USB 3	179 USD
Time of Flight	Microsoft Kinect V2	512 by 424	30 fps	Color Camera with resolution of 1920 by 1080 and Stereo Microphones	USB 3.1	400 USD Not currently manufactured
Time of Flight	Basler ToF	640 by 480	14 fps	None	GigE	2340 USD
Laser Line Scan	Riftek RF625	1280 by X*	X/500 fps	None	Ethernet	9500 CAD

* X is the number of detections taken to build a depth image

Table 3 - 3D sensor comparison

4.2 Vision Algorithms

One of the key challenges in this system was how to process image data. As the system could be fed with monitors and TVs in various states it was important to ensure that the system was flexible enough to process each of them, and find key features and points in a 3D space.

In order to accomplish this task, many different types of features and classifiers were required. Different deep learning techniques (regional convolutional neural networks, and Image segmentation networks) were utilized in the final system as well as traditional statistical machine vision techniques.

4.2.1 Regional Convolutional Neural Networks (RCNN)

Regional convolutional neural networks are a specific class of neural network, which focus on object detection and classification. This is a rapidly evolving field, which has been utilized for everything from surveillance to self-driving vehicles.

The goal of utilizing regional convolutional neural networks in this application was to assist in fixture loading and unloading as well as determining the type of FPD in the fixture prior to processing. The network was broken into three classes. Those classes were monitors, TVs, and upside down. Upside down was included in the dataset to protect against operator error. If the system was loaded upside down, then the FPD cannot be properly loaded into the fixture or processed. In addition, this classification ensures that the upside down monitor can be unloaded promptly for increasing system efficiency. This classification of the monitor was especially important, as the processing steps for a Monitor and a TV are vastly different. This step also ensures that the system will not run if there is no FPD in place reducing the likelihood of crashes.

4.2.1.1 Faster Region-based Convolutional Neural Network (Faster R-CNN)

Faster R-CNN [34] was chosen for this network as it is a good balance between computation time and accuracy [35]. The general structure of Faster RCNN can be found in Figure 37. The network can be broken into several components; feature extractor (convolutional layers), region proposal network (RPN), and classifier. The classifier is comprising of a box predictor and box regressor. The box regressor is used the initial bounding box region and predict a more accurate bounding box. The feature extractor is typically a popular convolutional neural network architecture created for image

classification (ResNet [36], VGG16 [37], etc.). There are several reasons why using one of these convolutional neural networks is appealing. These networks have been tested extensively with image classification challenges, so it is relatively simple to determine the overall capacity and speed of an architecture relative to other classifiers. Many of the networks are also available in most deep learning APIs and have pre-trained weights from image classification challenges available which can help with initializing the network. Starting a network with weights from another trained network is called transfer learning and has been shown to increase training speed and final network accuracy [38].

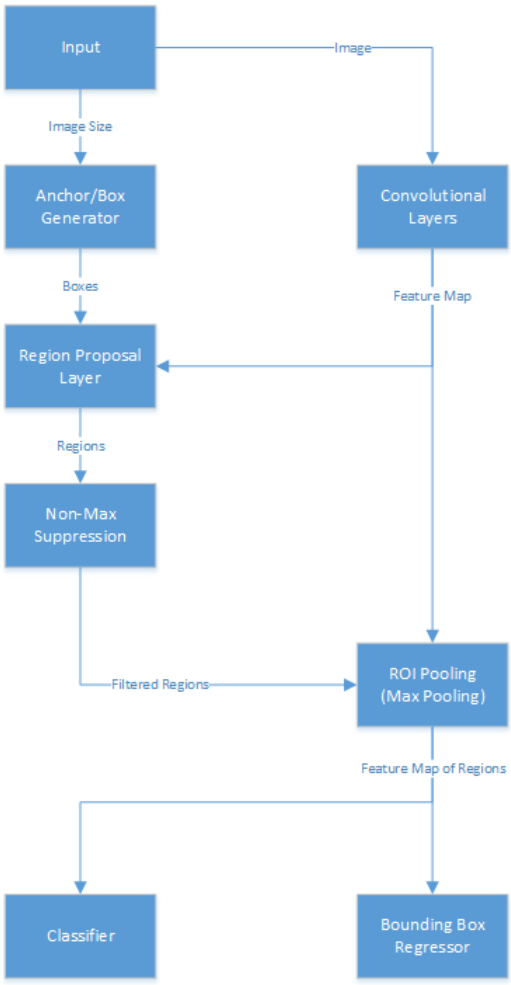


Figure 37 - Faster R-CNN Structure

The region proposal network is fed with a myriad of regions or proposals with varying scales, locations, and aspect ratios. These proposals are generated from a standard generator. For Faster R-CNN this is typically an anchor based sliding window generator, but this is not always the case. These windows are then fed into a network which further refines the bounding box and predicts whether this is an object of interest or not. This can be thought of as mid network filtering.

After these regions are proposed they are then resized via a max pooling layer to the size expected by box predictor. The box predictor includes a second bounding box regression layer and a final classification layer. These are both fully connected layers.

The bounding box regression is class dependent, so the bounding box regression layer has 4n outputs where n is the number of classes attempting to be predicted and the class predictor has n+1 outputs. This extra class is a background class used to remove any background classes that may have made it past the RPN. Finally, non-max suppression is used to remove any extraneous and same class overlapping predictions.

The losses for these networks are associated with the RPN and the box predictor. The losses for the RPN and box predictor are calculated using the same formulas and can both be broken into losses for the bounding box regressor and for the class predictor. Equation (3) shows a formula where p is the class prediction, t a scaled bounding box output and values with a * are target values. The bounding box regressor's loss L_{reg} uses robust smooth L1 loss defined in [39] and the classifier's loss L_{cls} uses cross entropy loss. λ is a hyper parameter which is normally set to a value of 3 and each N value is the number of losses and also the max value of i. It is of note that the bounding box regressor loss is set to zero when the class predictor does not predict the correct class or when it predicts background. The prediction is considered correct if the bounding box has an intersection area over union area of greater than 0.5 with a ground truth bounding box and the class is predicted correctly. This is done because the bounding box regressor is class dependent, so it should only be optimized for the correct class. Since there is no ground truth bounding box for background classes the loss cannot be created. For further insights into this network including calculation of t values see [34].

$$L(\{p_i, p_i^*\}, \{t_i, t_i^*\}) = \frac{1}{N_{cls}} \sum L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum p_i^* L_{reg}(t_i, t_i^*) \quad (3)$$

4.2.1.2 Dataset

Due to the limited number of FPDs available for gathering a dataset, the dataset was rather small containing only 118 images. The breakdown of the images per class is show in Table 4. Most of the images were gathered on site via a robot mounted camera in the work cell. The images were taken in this environment so that the lighting would be consistent with typical operating conditions. The images were saved as JPEG files to reduce size and have a total resolution of 960 by 540 pixels or approximately 0.5 megapixels.

In order to increase the number of images and thus the robustness of the network, each FPD was used for around three pictures; one in the condition it was received, one after being damaged, and one upside down. The damaged image included breaking parts of the FPD such as smashing or removing the screen and/or bezel. The FPDs were damaged to help the network become less sensitive to the condition of the FPD in its detection and classification.

Class	Number in Dataset	Percentage of Dataset
Monitor	60	44.12
TV	38	27.94
Upside-down	38	27.94

Table 4 - Object detection dataset breakdown

During training time, the dataset heavily augmented to make the system more robust and increase the total amount of different images the network may see. One of the key dataset augmentation techniques used in this network was image rotation. A total of three potential rotations were used each at 90-degree increments. As Faster-RCNN is not rotation invariant this leads to the number of images being used for training being increased by a multiple of four. The data was also augmented by mirroring the images in its X and Y axis creating multiplying the total images by four again. There was also random brightness and contrast scaling as well as bounding box jitter. This does not make the data incredibly different as it is still correlated in some way, but it does increase its robustness to lighting conditions. With all the augmentation in place the total amount of different images the network was trained on is 1888 excluding brightness and contrast augmentations.

Labeling was done via a custom labeling script, which uses OpenCV to load images and load or create xml files. The xml files contain the labels and bounding boxes for objects in the images. To speed up dataset creation, an automated labelling script was created. This script uses a neural network that was trained on a small dataset to label images.

This script can run one of two ways it either uses the class predictions from the network or all class predictions are overwritten with a single value. This is used when the images being labeled only have a single class in frame. The auto-labelled images are then validated and added to the original dataset. This modified dataset is then used for training. This process is then repeated until the network has achieved the needed accuracy.

4.2.1.3 Network Configuration and Hyper Parameters

The neural network used in this application is Faster-RCNN with ResNet 101 convolutional layers. ResNet 101 was chosen for the convolutional layers as it is a good balance between speed and accuracy. It is important that the network be both quick and accurate as no points are being detected to do value added work but the detection decides which process the FPD will go through. Since the work is not value added any reduction in the processing time will increase overall system efficiency. Faster RCNN with Resnet101 has a relatively high mean average precision (mAP) compared with other networks and a moderate GPU time depending on the network parameters [35].

The parameters for the region proposal network were setup specifically for this application. Due to the variety of sizes of the FPDs the anchor generator has a large variety of scales. The scales used are 0.25, 0.5, 1.0, 2.0, and 2.5 with aspect ratios of 0.5, 1.0, and 1.5. These scales and aspect ratios augment a base window size of 256 by 256 pixels. With an image size of 540 by 960 and a height and width stride of 16, this leads to a potential of 29700 regions for the anchor generator to check; though many of these regions go over the bounds of the image and are thus ignored.

4.2.1.4 Training

During training, the dataset was split into testing and validation sets. These set sizes were 94 images for training and 24 for evaluation or an approximately 80-20 split. This was done to attempt to avoid overfitting by utilizing early stoppage and to automatically test the networks accuracy prior to

deployment. Early stoppage is when the network training is stopped prematurely (before a preset number of training steps) to avoid overfitting. This is typically a manual process.

In Section 8.1.2,the final dataset augmentation was discussed. These were not the only attempted augmentation techniques. Random hue adjustments was attempted but caused a significant degradation in results lowering the overall validation accuracy from 95% mAP to a maximum of just above 60% mAP. This may be due to the some of the monitors and TVs having gray areas which when the hue is changed can drastically shift in color. This is would be why brightness and contrast changes do not have the same effect.

When training the networks various hyper parameters were adjusted. The hyper parameters include regularization weights, dropout keep percentage, learning rates and momentum optimizer decay values. In the final deployed network, the parameters used are shown in Table 5. These gave by far the best validation results.

When training this network, a gradient descent optimizer was used with a momentum value of 0.9. This optimizer keeps the previous update values and multiplies them by the momentum value before adding them to the current update values. Gradient decent optimizers are a typical error propagation optimizer used for training neural networks. This helps smooth the descent and has been shown to decrease training time.

Parameter	Value
L2 Normalization Weight	0.001
Dropout Keep Probability	70%
Max Detections	300
Max Detections per Class	200
Learning Rate	step 0 to 9999: 0.0003 step 10000 to 14999: 0.00003 step 15000 or more: 0.000003
Localization Loss (L_{reg})	2.0
Classification Loss (L_{cls})	1.0

Table 5 - Object detection training and model parameters

4.2.1.5 Results

The validation results are shown in Figure 38. The green line shows the validation precision of the graph which was implemented in the final system. During testing, this network has yet to make an incorrect classification but the FPDs in which it is being tested come from the same batch which was used for training. This could be leading to a misleadingly high accuracy rate.

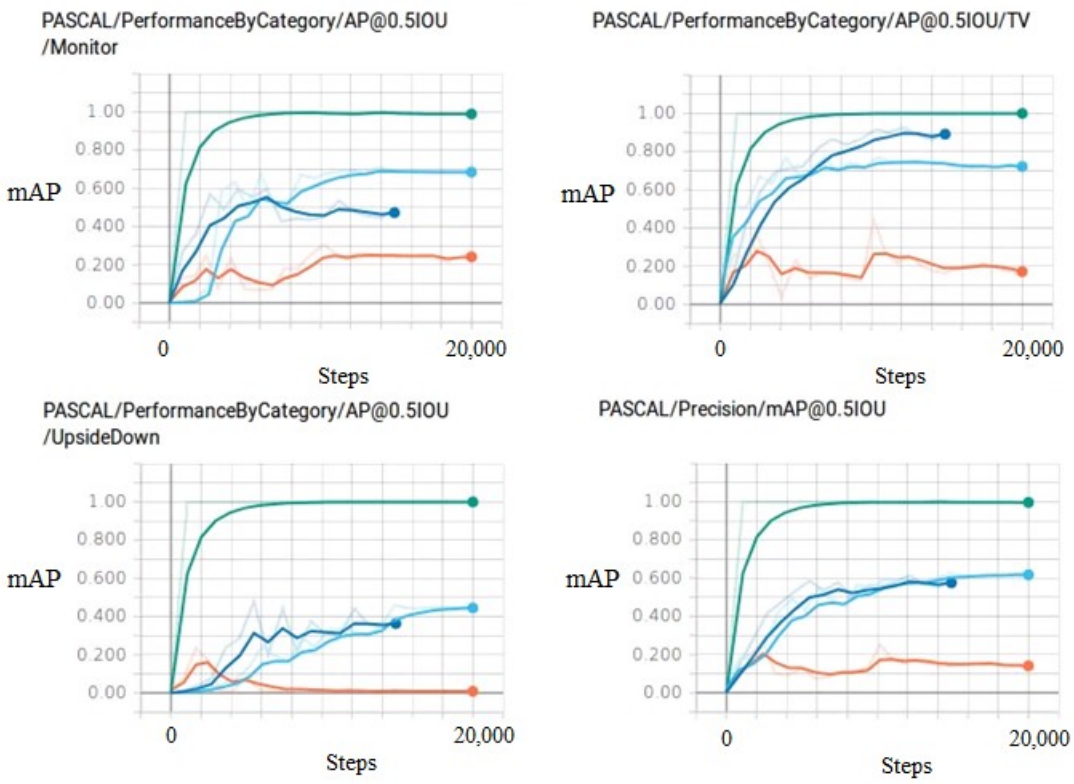


Figure 38 - Validation Results from FPD Object Detection Network

After the network is exported, the inference time is approximately 140ms when run on an Nvidia GTX 1080ti. Significant time was spent to decrease this inference time by using direct memory transfer to send the data to the TensorFlow. This reduced the inference time from 250ms to the final speed of 140ms.

4.2.2 Image Segmentation

Image segmentation has been a difficult challenge in computer vision for years. The goal of image segmentation is to take an unknown input image and correctly classify every pixel in the image into the correct class. This is one of the core technologies being utilized in research involving autonomous driving and has been utilized for detecting things like dogs, cats, pedestrians, bikes, road lanes and much more.

Much like other fields in deep learning this area has been evolving rapidly. In this application, two similar but different techniques have been examined to detect the components of FPDs. These techniques are SegNet [40] and DeepLab [41]. Both techniques were trained with the same datasets, but slightly different augmentation techniques were utilized.

4.2.2.1 Dataset

The dataset created to train this contains 138 labeled images. Each of these images has a corresponding label image where each pixel is assigned a value based on its class. The classes in the data set are background, fixture, plastic, screen, circuit board, CFL, metal, and screen back. Most of these classes are self-explanatory but fixture, and screen back are less obvious. The fixture class contains items in the dataset that are used to hold the screen in place during loading. This includes the fixture pusher plates and hard stops. The screen back class is the material behind the screen material and filters. It is typically a white film which is used to direct light from the light source (LED or CFL) through the screen. This class was included to be able to add error checking for screen removal and

Attempt to help with CFL detection. This will help with CFL detection as the CFLs and screen backing have the least amount of variance.

The dataset was created with multiple backgrounds including a concrete floor, an previous revision of the FPD fixture and the current fixture. This was done to try to make the network more reliable if changes to the fixture are required. The image resolution is 960 by 540 pixels or around 0.5 megapixels. When training both SegNet and DeepLab the dataset was split into training and validation sets with a split of 80-20 or 110 and 28 images, respectively.

In each of the networks examined, the same dataset augmentation techniques were utilized. During training the input images were scaled randomly between 0.8 and 1.0 times their original size. This scaling was done with a random floating-point value creating potentially hundreds of potential image sizes while keeping the same aspect ratio for the input image size of 960 by 540 multiplying the dataset by around 6 times. The images were also mirrored in X and Y. Mirroring was chosen over image rotation to keep the input aspect ratio consistent while still augmenting the dataset. This effectively multiplied the dataset by 4. The images also have contrast and saturation randomly modified. The

overall dataset augmentation multiplies the dataset by around 24 to a total size of 3,312 unique examples.

To create this dataset a custom labelling script was created. This script overlays a label and inputs image. There are three sliders, one changes the class being labeled, one changes the size of the brush that is doing the labelling, and the other changes the weight of the label and inputs image in the display or the opacity of the label. Figure 39 shows the labeling script developed. All of the images are saved in a portable network graphics (png) format. The png image format is used as it is a lossless and any compression in the label image will cause pixels to be miss labeled. In images this is not a concern but a change of a pixel value in the label image changes the class. This will lead to issues training the network.

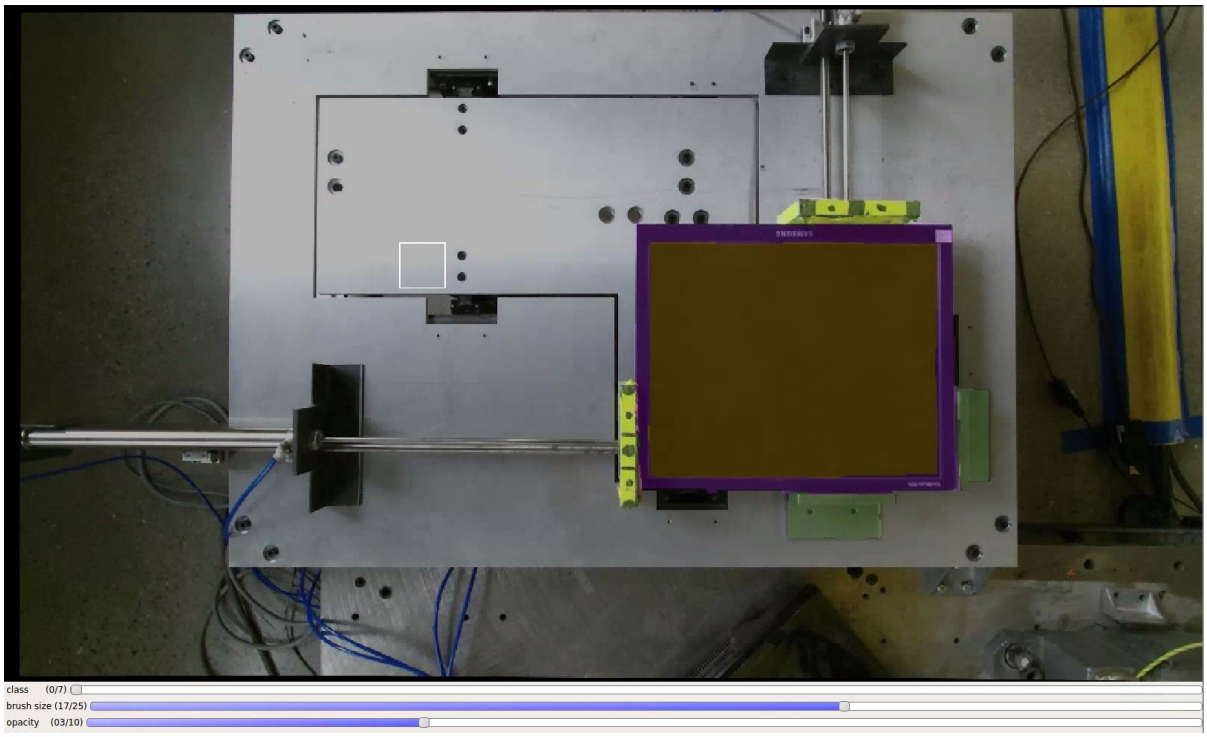


Figure 39 - Image Segmentation Labeling GUI

4.2.2.2 SegNet

SegNet is an encoder-decoder based image segmentation network. It is made up of essentially 4 different layer types. On the encoder side the network runs several convolutional layers before pooling

the network to a smaller size. The pooling layer that is typically used is max pooling. On the decoder side the small feature map is put through an upsampling layer followed by convolutional filters. The upsampling layer is either a layer which remembers the indices of the previously maxpooled values and puts the values back into those locations or can use other upscaling techniques such as bilinear interpolation. In the network run in this system, the up-sampling layer utilized was bilinear interpolation. The role of the upsampling layer is to bring the small feature map up to the full image size while still keeping the positive features of running max pooled convolutional layers such as translation invariance. The final feature map is then run through the final layer type, softmax, in order to get the final pixel wise output.

4.2.2.2.1 Training

The training for this network was rather simple. Dropout and other regularization techniques were not employed. The network was trained with a stair casing exponential decay with an initial learning rate of 0.01 and a decay factor of 0.98. The learning rate decays every 5 epochs. When training the network, a momentum optimizer was used with a value of 0.7. This value was chosen to increase training speed. When training this network, if the momentum value was above 0.7 the network had a high accuracy with regards to plastics but a poor accuracy for CFLs and circuit boards. For this reason, the momentum value was lowered.

4.2.2.2.2 Results

After training the network, it was tested using validation data at regular intervals. The graphs in Figure 40 show these testing results. The line in red is the final exported neural network. It achieved a mean intersection over union (mean IU) of 82% where the mean IU is the amount of true positives divided by the sum of the false positives, true positives, and false negatives. Due to the amount of background in the dataset, the mean IU is likely at this level due to the amount of false detections.

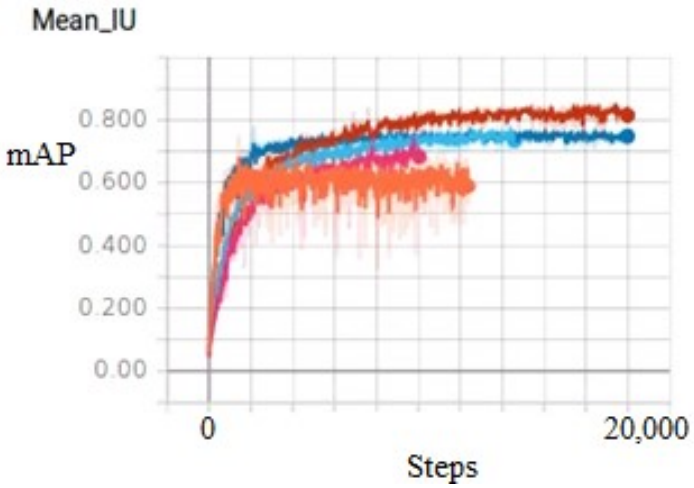


Figure 40 - SegNet Testing Results – Red is final network while others are previously trained networks

The mean IU reported in the validation set is far from what is seen during testing as show in Figure 41. The example Figure in this network does, however, show some of the strengths of SegNet. The network is able to detect fine features like the bezel with high accuracy. The boundary is almost the same as in the real image. It also shows the weakness SegNet, it can struggle with overall accuracy when compared to other techniques. It was very powerful for its day and still works well in certain use cases but does struggle with others.

The exported network has a typical run time of 180ms to create a fully segmented image when run on an Nvidia GTX 1080Ti.

Figure 41 - SegNet Segmentation Example

4.2.2.3 DeepLab

DeepLab is an image segmentation method that is not a true encoder-decoder structure. It uses atrous/dilated convolutions to find features of various scales and fully connected conditional random fields to recover object boundaries. The idea of this method is to use atrous convolutions and max pooling layers to create a feature map with translation and scale invariant features. This feature map is then run through a fully-connect conditional random field (CRF) and finally a softmax layer.

CRFs commonly used in image segmentation to refine image segmentations. This filter uses information from the input image such as pixel values, adjacent pixel values, and class confidence to refine the image segmentation.

Atrous convolutions work by inserting zeros in between the values of a normal convolutional filter. These gaps increase the field of view while keeping the computation speed consistent. Figure 42 shows an example of these atrous convolutions.

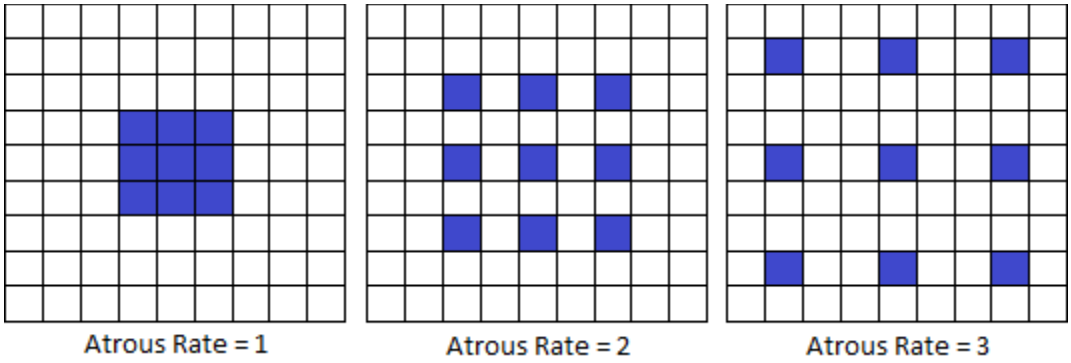


Figure 42 - Example of an Atrous or Dilated Convolutional Filter

In this specific implementation of DeepLab Xception-65 was used for the convolutional layers. Xception is a modified version of the traditional Inception network, which utilizes depth wise separable convolutions to decouple special and cross channel correlations in the data. By doing this the overall accuracy increases while only having a small impact on speed. Xception-65 has 65 convolutional layers. The atrous rates used in this network were 6, 12 and 18.

4.2.2.3.1 Training

When training this network some regularization techniques were employed. This network was run with dropout with a keep probability of 0.9 and an L2 regularization rate of 0.0001. After minor experimentation, this was found to help with generalization. The network was trained with a stair casing exponential decay with an initial learning rate of 0.003 and a decay factor of 0.99. The learning rate decays every 5 epochs. When training the network a momentum optimizer was used with a value of 0.8. This value was chosen after experimentation and for the same reason as 0.7 was chosen for SegNet.

4.2.2.3.2 Results

When training the network, it was tested using validation data at regular intervals. The graphs in Figure 43 show these testing results. The line in green is the final exported neural network. The achieved mean IU is 78%.

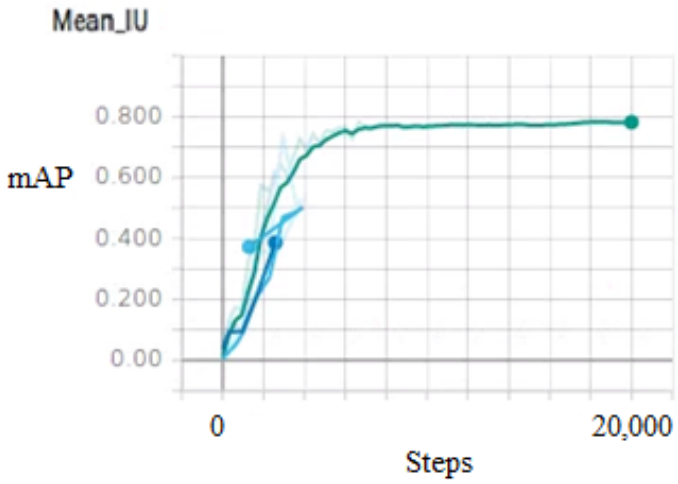


Figure 43 - Deep Lab Testing Results

Figure 44 shows an example detection from this neural network. One of the key issues in this detection is likely due to the CRF layer. This can be seen on the input image and segmented image. Areas where the pixel values are alike cause significant issues in the network. The bezel is not well defined in the output image, which is likely due to this issue. The output image compared to the results visible shows a smaller mean IU than the validation set. This is likely due to the high correlation between the input and validation dataset.

The exported network has a typical run time of 200ms to create a fully segmented image when run on an Nvidia GTX 1080Ti.

Figure 44 – DeepLab Image Segmentation Example

4.2.2.4 Comparison and Discussion

In this application, the clear winner is SegNet. Though there are some issues with false detections and classifications, the encoder-decoder structure has far less issues on boundaries than DeepLab's CRF. The SegNet architecture is also faster and more accurate with a mean IU of 82% and a speed of 180ms where as DeepLab has a mean IU of 78% and a speed of 200ms.

There is however one issue with using SegNet. The license for the code base from its creators does not permit commercial use. Since the provided code does not allow commercial use it is likely that the base algorithm is also non-permissive. Deeplab on the other hand was released with an Apache 2.0 license, which allows for commercial use. For this reason, when used in industry DeepLab may be the only choice for image segmentation.

To improve the overall accuracy the best option is to increase the size of the dataset. Most of the images were taken with a concrete background and the old fixture. This will cause the network to perform poorly on the new fixture when compared to the other background types. Another concern is that the amount of training images in the dataset is lower than many image segmentation challenges. For example, the KITTI semantic segmentation benchmark has 200 labelled images.

4.2.3 Traditional Machine Vision Techniques

In this system, traditional machine vision techniques supplemented image segmentation, object detection techniques were utilized to replace them in certain instances where a machine vision algorithm could out perform a deep learning-based alternative. These techniques were also utilized to find the value of points from an image in 3D space. These techniques, especially transforming points into a 3D space, are the backbone of this system.

This section will cover a brief overview of the techniques utilized and some of the tasks in which they were utilized. These tasks include filling/preprocessing rectified images, 3D transformations, and feature (edge, contour, shape, line) detection.

4.2.3.1 Image Preprocessing

Image rectification is an important practice when dealing with 3D sensors. Whether this is rectifying stereo cameras or a depth image into the same frame as color images. Unfortunately, there are times when this rectification process leaves significant gaps in the data due to large changes in height. There can also be significant gaps in generated depth images from stereo or time of flight depth sensors. This can be caused by the depth sensor having a high exposure or a section of the frame has a high reflectance. High reflectance causes significant gaps in data when using time of flight sensors.

When creating a point cloud these gaps in data can be ignored but in most of the processes in this system this is not the case as the depth image is used find image points in 3D. If there is a location or feature that is detected in the color image that needs to be transformed into a 3D position using the depth image and there is a gap in the data at that location the system will be unable to run. To solve this problem, several image inpainting techniques were examined such as fast marching method, Navier-Stokes, and a custom contour-based solution.

Navier-Stokes and fast marching method image inpainting are standard processes that deal with image inpainting by minimizing an equation taking into effect the boundary conditions around the pixels which must be infilled. Both of these techniques take into account the image gradient and assume that if the image is changing along a boundary it is likely to continue this change. These techniques are very reliable and lead to good results when the regions are small but when there is a large section that

is not in the image these techniques start to fail. Figure 45 (A) shows an example of this issue. Each of these techniques run at a speed of around 400ms on an image with a resolution of 960 by 540.

To solve this inpainting problem a mean infill solution was proposed. This method works by finding gaps and taking the filling them with the mean of all pixels that are not part of the gap in a minimum enclosing rectangle of the region. There are potential issues with this method. For instance if this method is filling an area that follows the gradient the method will fail. It is not recommended for use on color image infill but it does work well with rectified depth images. The processing time for this method is approximately 600ms.

Figure 45 shows a comparison of each of these techniques on the same image. This Figure shows that Navier-Stokes image infilling works better on this image type than the fast marching method. These methods also work well when attempting to recover regions like the bezel on the left side where some of the pixel information is available and there is not a large enough of a region for the gradient to change the value heavily. The mean infill works well in large regions such as the gap in the center of the screen.

Depending on available computation time, it may be possible to combine both solutions. Using mean infill only on regions whose circularity and area is high and using Navier-Stokes for the rest of the infill.

If using other sensor types aside from the Kinect V2 there are some good filters included in custom systems. A good example of this is the Intel Realsense which includes preprocessing filters for smoothing, hole filling, etc. These have already been optimized and are faster than the methods used. They are however contained inside the Intel Realsense drivers and are difficult to implement on other hardware.

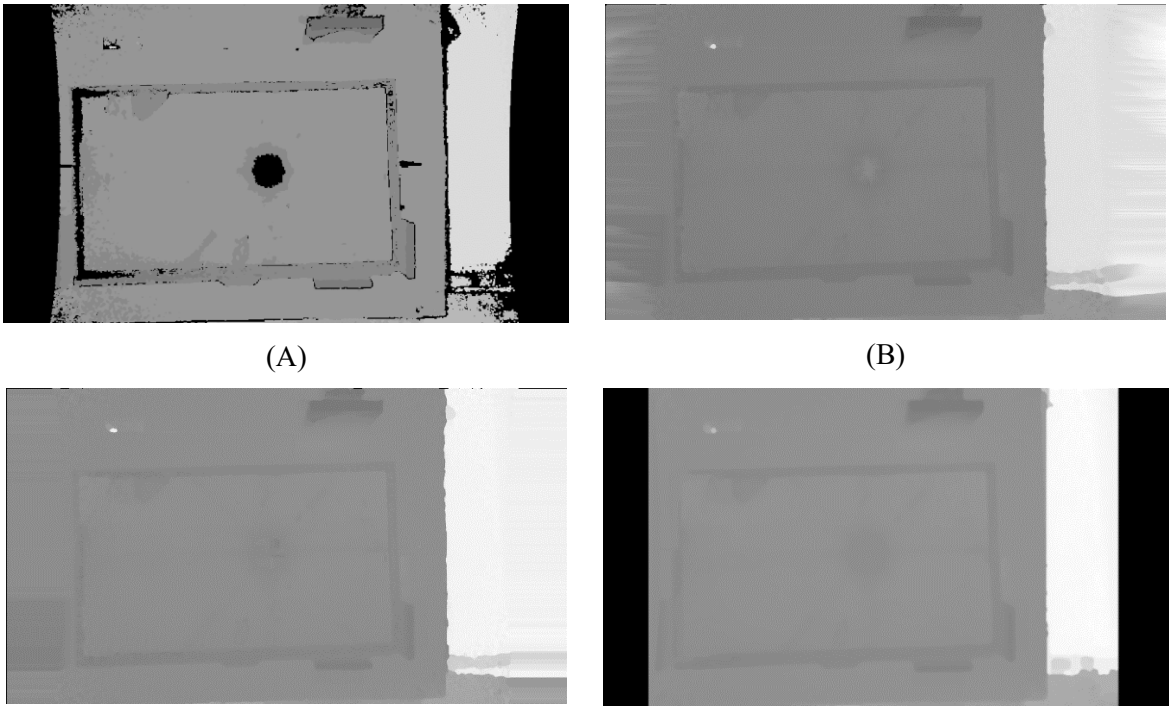


Figure 45 - Image Infill Techniques (A) Unfiltered Depth Image (B) Inpainting with fast marching method (C) Inpainting with Navier-Stokes method (D) Inpainting with mean infill

4.2.3.2 3D Vision

The potential 3D sensors used in this project all have drivers, which include depth and rectified images. One of the key problems that needs to be solved is how to transform detected points into a 3D location from the base frame, so they can be used to guide the robot. In order to solve this problem a service was made which takes a depth image and a series of points. These points are turned into a 3D location from the camera reference frame before being transformed into the base frame.

The set of equations (4) is to get the 3D value from the depth. Where X and Y are the pixel coordinates, c_x and c_y are the image center value in X and Y respectively, f_x and f_y are the focal distance in X and Y respectively, and $D_{y,x}$ is the depth at point (y,x) on the image. The focal and center values are camera intrinsics and are calculated via the camera's driver. This point is in the camera frame to

transform this into the base frame a transformation matrix is generated and the points are transformed. This is done via ROS's transformation (TF) library.

$$x_{3D} = (x - c_x) * D_{y,x}/f_x \tag{4}$$

$$y_{3D} = (y - c_y) * D_{y,x}/f_y$$

$$z_{3D} = D_{y,x}$$

$$P_{3D} = (x_{3D}, y_{3D}, z_{3D})$$

4.2.3.3 Feature Detection

Feature detection is used in conjunction with other techniques, mainly image segmentation, to find shapes and boundaries that are expected in an FPD. As discussed in Section 2.1, FPD designs tend to be relatively consistent and because of this shape and feature detection can be a useful tool to find regions of an FPD. For example, the screens of FPDs are rectangular so to find the boundaries of a screen can be accomplished by using a depth or segmented image and searching for rectangles.

In the previous phase of the project discussed in Section 2.3, feature detection was the only method used. Though it was somewhat successful feature detection with color images was not enough to be reliable. To combat this segmented and depth images were used for feature detection. The main techniques being utilized are contour detection and fitting these contours into shapes and line detection. This project uses the implementations done in OpenCV for detecting these features. The main techniques used are contour detection which uses the algorithm from [42], contour approximation which uses the algorithm from [43], and Hough line detection and Probabilistic Hough line detection [44]. These are used in algorithms discussed in Chapter 3.

Chapter 5
Results and Discussion

5.1 Testing

This project was tested extensively via simulation to ensure that movement and states were ready to be implemented in real world tests. When refer to simulation this means that the robot motion and collision checking was run in a simulated environment. All vision was run with the real robot mounted KinectV2 and actual FPDs in the fixture. This was done so that a large amount of testing could be accomplished with a limited number of FPDs as any real world tests would damage the samples to the point that they could not be used for more testing. This focus on simulation over real world tests lead to a robust process flow, and repeatable and reliable states. The mechanical side of the project lead to a large amount of unforeseen issues when these states were implemented.

There were also some unforeseen issues with the vision side of the system, but these are relatively minor when compared to the mechanical issues. During simulation live vision was used and it was possible to run the system from start to finish. Proving that developed vision and motion algorithms are valid but there are some key mechanical issues that must be addressed.

5.1.1 Simulation Results

During simulation the system was successful the majority of the time. Table 6 outlines the approximate success rate and the time to run each state. The run times assume that the system is not in demo mode and does not need to wait for push buttons and the recovery state and initialize state assume that the robot was not faulted and needs to be reset. The Unloading and Recovery states show the minimum possible runtime and assume that the FPD removal was successful on the first attempt. The run times are also all approximate and are based on experience with the simulations. They may not be indicative of final system performance.

The only state that cannot fail is Initialize. This is because this state just checks the robot and homes the fixture axis. There is nothing to go wrong. The other states have some potential issues during runtime. The next most successful states are Recovery and Unloading each with a 95% success rate. The five percent failure rate comes from using the Y value of the midpoint of the FPD to align the

monitor. If the FPD is too small, it is not possible to reach this value without the robot colliding with itself. The collision planner will not allow this to happen and the system will fault when trying to plan the motion and the state will then try to find the FPD again. This loop continues until the found midpoint Y value is reachable. This is uncommon but can happen.

Monitor and TV screen removal are a very similar operations with slightly different paths. The difference is processing time between the two operations is a result of the difference in cutting speed of each operation. TVs are far larger but can be cut at over twice the speed of a monitor as the only thing being cut is the filters while the Monitor requires the system to cut stainless steel. The path planning operations are almost always successful in these states. The lower rate of success comes from the vision process not getting correct values for the screen bezel boundary of the FPD. This can be improved by increasing the dataset for image segmentation.

Monitor CFL Removal is a simple pick and place operation. There is some complex free motion and there is the potential that the end mill can strike the fixture during this operation. Although this is uncommon, it can happen with smaller FPDs. The rate of success is also related to the Monitor Screen Removal operation as this is where the pick points originate. If the edge of the monitor is not found properly this state will not function properly.

TV's CFL Removal is a difficult task to qualify. If the measure of success was 100% removal of CFLs, this state would almost never succeed. The vision process finds the majority of the CFLs but very rarely manages to find all of them. Figure 62 shows the results of CFL detection. This state can also fail in the Twist state if the bulb breaks in multiple places rather than just on the edge of the CFL. This is the reason why TV CFL Removal is run twice. Typically, a TV will have 16 to 22 CFLs making the backlight.

When taking all of the times into account the processing time range for Monitors and TVs are 110 to 170 seconds and 160 to 230 seconds. These estimates were found from repeated simulation over a period of two weeks using 6 different FPDs. In the simulations were run approximately 50 times.

Process	Approximate Time	Approximate Success Rate
Initialize1	<5 seconds	100%
Loading	10 seconds	90%
Recovery	10+ seconds	95%
Unloading	15+ seconds	95%
Monitor – Screen Removal	60 to 120 seconds	80%
Monitor – CFL Removal	20 seconds	75%
TV – Screen Removal	50 to 90 seconds	80%
TV – CFL Removal (Image Segmentation)	5 seconds for vision and approximately 5 seconds per CFL	80%

Table 6 - Simulated process time and rate of success

5.1.2 Real-World Testing

5.1.2.1 Robot Control

The Fanuc robot driver released by ROS Industrial functions well but there are weaknesses to this system. The main weakness is that the robot may not precisely follow the planned robot path. This is due to the way that Fanuc's motion commands function and the way that ROS or more specifically MoveIt generates paths. When MoveIt creates a path, it assumes that there is fine grain control of each motor in the system. As such, MoveIt generates points which include not only joint positions but velocities and accelerations for each point in a path. Fanuc does not allow this level of control in their joint motions. Unlike ROS motion commands, there is only 1 speed value. When a Fanuc robot performs a joint move each joint starts and stops moving at the same time. This method is known as linear joint interpolation. Due to this, the performed and commanded speed will likely deviate from one another. Another potential issue is the implementation of the continuous motion type in Fanuc's robot controller. There are two ways to end a motion command on a Fanuc controller. The first is "FINE"

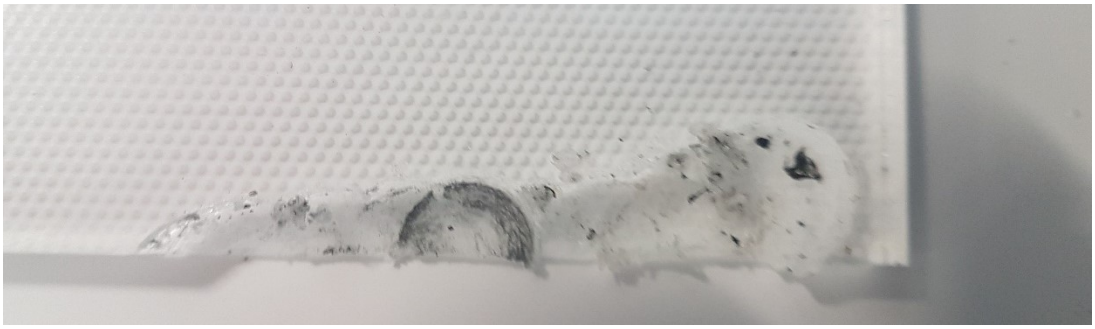
which performs a full stop and the second is “CNT” which blends motion together. This number defines how close the end effector gets to a point before decelerating. This is set to 50 on the system by default. If the points are relatively close together then this does not cause a noticeable decrease in accuracy. This was found during initial testing of this implementation.

Another potential issue with this system is the correlation between accuracy and speed. The system can only receive points at a loop rate of 42 Hz on an R30iA controller. Due to this, if a full speed and precise linear motion needs to be executed there is the potential that the system cannot keep up. If the end effector needs to move linearly at 2 meters per second and the system can only receive points at a rate of 42 Hz the closest these points can be together is 4.76 centimeters. If the system is not fed with points fast enough then the motion will stop while it waits for more points. This has not been an issue in this application.

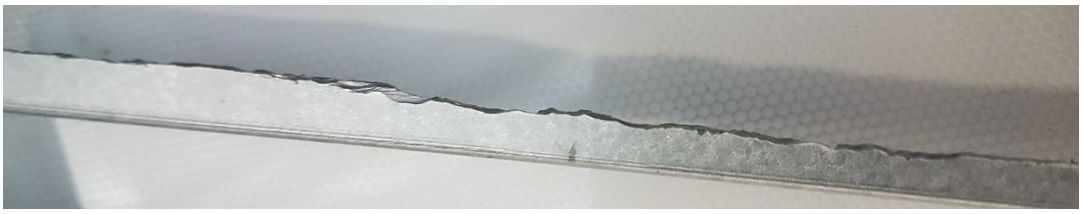
5.1.2.2 Cutting Tests

There were several key limitations with the mechanical side of this project that were discovered at the end when real world testing was implemented. These limitations are not unsolvable but cannot be fixed in the scope of this work. There were small mechanical limitations found in every part of the project including the tooling, fixture, and robot. However, the largest problems are related to the robot and tooling. The tooling problems can potentially be fixed and some work has gone into these designs but modifying or changing the robot is well beyond the scope of this project.

One key limitation in this project is significant tool chatter. Figure 46 shows the results of this chatter on two different FPDs. The chatter is mainly seen when processing monitors and seems to be caused by the mill attempting to cut the stainless-steel or aluminum bezel. The image shows significant deflection when attempting to perform cuts on a linear path into the bezel. The chatter is not present during plunging cuts but when a linear cut is done along a bezel the tool can move wildly.



(A)



(B)

(C)

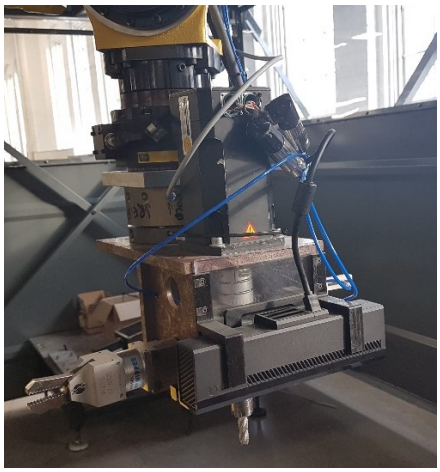
Figure 46 – Examples of chatter (A) milling tool deflection during cutting (B) roughness of cut on a bezel (C) cutter bounced between bezel edges.

Tool chatter was present during previous phases of the project but became more prevalent over time. This is likely due to a change in tooling design. The previous design (Figure 47 A) was less dexterous than the current but the distance to from the end of the milling cutter to the robot tool frame was far smaller reducing the moment on the robot substantially. The two designs are shown in Figure

47. Though the previous design reduced tool chatter, it was still present. This is likely due to the rigidity of the robot. The M710iC/50 is a powerful robot but even when just grabbing the milling tool by hand the tool can be moved several millimeters. This movement is not caused by flexibility? in the tooling but by the robot itself. The forces applied when doing this is substantially less than the maximum forces that may be seen during cutting.

The chatter is likely also exasperated by changes in speed caused by the speed conversion from ROS to Fanuc control. This leads to a less consistent cutting speed and potentially a large amount of acceleration during precise movements.

(A)



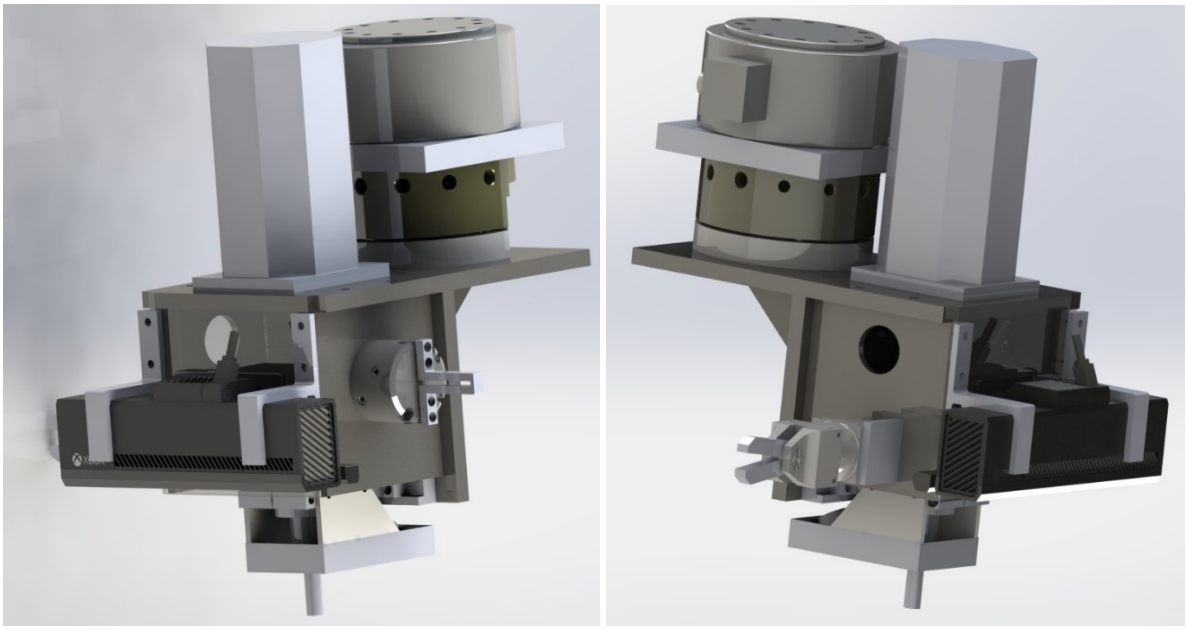
(B)

Figure 47 - EOAT Designs (A) Initial Tooling Design (B) Updated Design

Tool chatter is only present on the monitors. The TVs do not require the tooling to perform cuts on any metal components, only plastics, which reduces the load on the end mill. Unfortunately, this process does have its own problems. The main issue being the amount of debris generated by the TV cutting process and a specific filter. This can and will cause reduced accuracy with the vision used for CFL removal. Figure 48 shows the amount of chips put into a TV during cutting. This could be solved by adding more examples to neural network training, but it would also be a good idea to reduce the amount of chips as much as possible. A reference design has been made to do this but is currently untested. Figure 49 shows this reference design, which includes a shroud that has bristle brushes mounted to it. The bristles will trap the particles while allowing air from outside to get in. The bristles

deform against the surface of the FPD to allow for full use of the milling cutter. There is also an included hookup for a vacuum on the back behind the tool. The connection point is angled so the orifice is facing towards the end mill to help with chip removal.

Figure 48 - Chips and Debris from Screen Milling



(A) (B)

Figure 49 - Rendering of Vacuum Enabled EOAT where (A) is from the right side showing the CFL gripper and (B) is from the left side showing the heavy gripper

An additional issue with TVs is the inclusion of a small brittle filter which is used to make the light diffuse. This filter is typically the bottom most filter and provides some of the rigidity in the system. The problem with this filter is that it is brittle and tends to shatter instead of cut as seen in Figure 50. This filter can be removed by using the heavy gripper, but it will leave debris inside of the TV. This debris is large and will not be removed via the vacuum attachment to the tooling. This can be solved either by adding a separate removal process for these pieces of plastic or by blowing it out with pressured air. It is recommended to modify the tooling to add in an air jet output for application. The addition of an air jet will also help remove any debris missed by the vacuum.

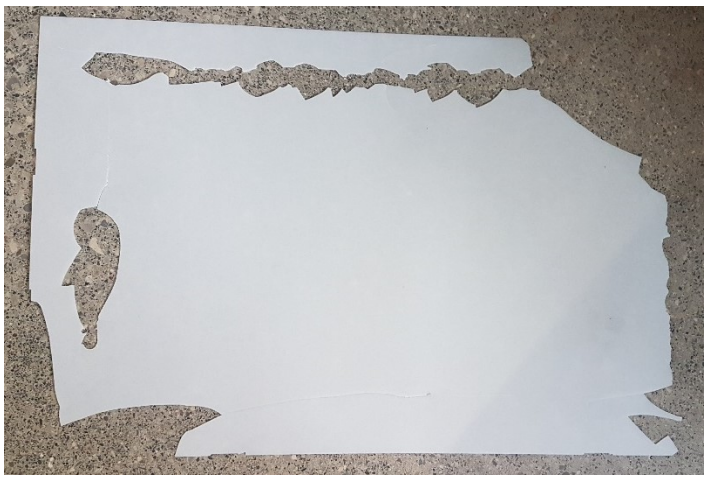


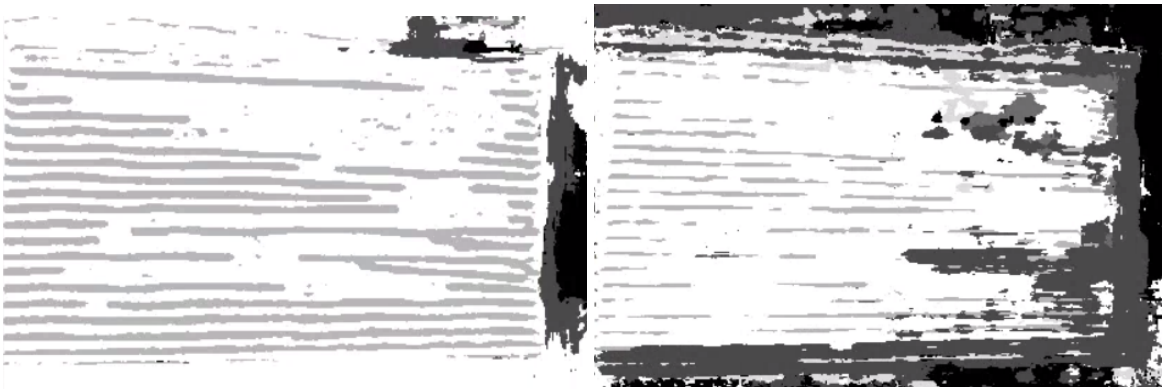
Figure 50 - Breaking of Plastic Diffuser after Screen Milling Operation

The fixture can be improved by modifying the fixture cylinders to increase rigidity when fully extended. The pushers can flex laterally by over a centimeter when fully extended. This can be done by replacing the guide rod with guide rails, which have more rigidity over large distances. This isn't a large concern as all cuts are performed towards the hard stops which reduces load on the pushers.

5.1.3 Vision

The vision system performs well but is sensitive to lighting conditions. During a demo the system was rendered unusable by an outside window and Lexan panel allowing in a large amount of light. This caused the system to fail almost outright. This can be addressed by blocking these unwanted light sources or by training the image segmentation network with varying lighting conditions.

There are significant improvements that can be made to the image segmentation network. The network does not work well for many FPD types especially monitors. This is likely due to the amount of training data used in the system. There were only 110 images used to train these neural networks from scratch. The background was also not the same as the current fixture as it had not arrived when the dataset was being made. This method shows considerable promise and the results can be improved substantially by increasing the dataset size. Figure 51 shows examples of CFL detection.



(A) (B)

(C)

Figure 51 - Example of Image Segmentation Networks and CFL Detection

(A) DeepLab (B) SegNet (C) Color Image

5.2 Return on Investment

Though there are some mechanical and vision limitations in this system the overall algorithms are very close to successful. With the processing times discussed previously and the cost of the each of the systems, it is possible to determine a payback period.

There was also development of a gantry based alternative system, which should be used in future instead of an articulated robot. This was not possible to implement in the project due to budget limitations. This system will cost less to make and be just as capable as the alternative. Figure 52 shows a rendering of the model. There are some strengths to the alternative system aside from just a reduced

cost. Since the gantry system is above the fixture there are four directions in which the system can be loaded and unloaded. The gantry system was specified to have a payload of 50kg which will have to be increase to remove mechanical limitations. Though the gantry systems payload will need to be increased, it shows the potential difference in cost when using an industrial robot and custom gantry system.

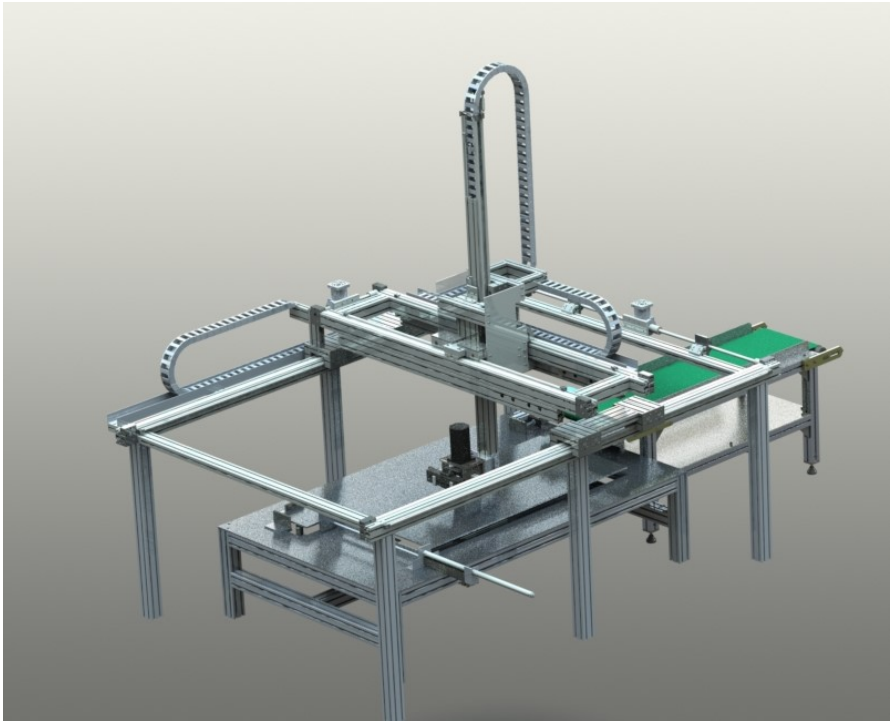


Figure 52 - Gantry System Reference Design

5.2.1 System Cost

Table 7 shows the approximate costs of large components in the system. The total cost of the system when using an articulated robot is $201,000. Where almost 70% of the cost is the robot. This cost can be reduced to $136,000, where approximately 55% of the cost comes from the gantry system.

Item	Approximate Cost
Gantry System	$75,000
Robotic System	$140,000
Fixture	$12,000
EOAT	$15,000
In Feed Conveyor	$10,000
Output Conveyor	$10,000
Guarding	$12,000
Electrical	$2,000

Table 7 - FPD recycling system component costs

5.2.2 Payback Period

The payback period for the proposed systems is based mainly off of the input feed to the system. There are some assumptions that need to be made prior to calculating the payback period of this system. Firstly, we assume that the system runs for a single shift or 7 hours a day. There is an assumed 20% downtime and only 241 work days a year. 241 work days is the number of work days in a year if weekends and 4 weeks of the year are removed. The system also requires a single operator which costs 80 thousand dollars a year and will use 6 thousand dollars of energy. We then assume that the average monitor weight is 5lbs and the average TV weight is 40lbs. We then need to calculate the amount of money that can be generated per pound. The recycled materials in an FPD can be sold for approximately 20 cents per pound for the materials inside and FPD and in Ontario if you are a level two recycler you can get $150 for collection and $600 for recycling of display devices [45]. This leads to a value of around 57.5 cents per pound. When using these assumptions there is a best-case revenue of $79,125.82 for monitors and $732,192 for TVs per year and a worst-case revenue of $27,904.94 for monitors and $ $489,264 for TVs per year. This clearly shows that you want to process TVs with this system due to the extra weight. The payback period for this system is between 0.19 and 4.87 years. This is a very

large gap but can be reduced by feeding more large TVs in the system and keeping the number of monitors to a minimum.

	Monitor		TV	
	Best Case	Worst Case	Best Case	Worst Case
Revenue	$ 79,125.82	$ 27,904.94	$ 732,192.00	$489,264.00
Payback years Robot	2.54	7.20	0.27	0.41
Payback years Gantry	1.72	4.87	0.19	0.28

Table 8 - Potential revenue and payback period of FPD recycling system

Chapter 6
Conclusion and Future Work

6.1 Conclusion

In this thesis, a system for automated dismantling of FPD was proposed and tested. This system was designed to use two optimized dismantling procedures based on the type of FPD loaded into the system. The processes have been tested and we have found that robot is unable to support the loads required to mill the FPDs. The payload of the system, 50kg, should be able to support it but the robot is not rigid enough to be able to stop the cutter from chattering. This chatter amplifies the load and causes the robot to fault.

The process to dismantle an FPD will vary due depending on type; monitor or TV. Each of these types are constructed differently and have a special process type. When cutting TVs, the system removes the screen to get to the CFL array below. These CFLs are then detected and removed. For monitors the system cuts away at metal holding an acrylic panel. This panel is removed to reveal the CFLs on the top and bottom of the system. If the loaded FPD is a monitor the system does not attempt to determine if the light source is LED or CFL. They are immediately removed to increase system efficiency.

For the system to work correctly it must be able to detect if an FPD is present in the fixture and what type of FPD is present. To accomplish this an implementation of Faster RCNN was trained and validated. This network uses ResNet 101 as a feature extractor. The network was found to have an accuracy in excess of 90 percent. This network detects FPD and puts them into three classes; TV, monitor, and upside-down. The first two classes are used to determine which dismantling procedure should be used while the final class was included to check if the fixture was improperly loaded. The upside-down class can also be used to allow the system to remove valuable components from the FPD as these are always located on the back of the FPD. This neural network was tested both with validation data and experimentally. It is possible that the accuracies reported are inflated as the system was trained and tested with FPDs taken from the same batch. Though no individual FPD was used for both training and testing the data may be correlated. Once the class is determined the specific dismantling process is done.

To find components both deep learning and statistical vision approaches were used. The processes were originally planned to use only statistical vision process but during testing these were found to be unreliable after implementation. To enable the system to find components within the FPD image segmentation networks were trained and implemented. These networks were used to segment the image via class. These segmented images can then be examined via statistical processes to determine the location of components. The networks tested were SegNet and DeepLab with an Xception-65 feature extractor. When running testing with the validation dataset both networks were found to be robust and to provide almost perfect segmentation with mean IOUs of 82 and 78 percent respectively. However, during testing the results were not as robust. The system had a hard time detecting screen boundaries on monitors but performed well on CFLs and circuit boards. This is fixable by increasing the size of the dataset and including more images with the new fixture as the background.

This system was tested in simulation and in a real-world environment. Due to the robot payload limitations mentioned previously a full test could not be completed on the real-world system. In order to properly test the vision and movement algorithms despite this the system was tested extensively in simulation where the robot motion was performed in a simulated environment. The vision was performed in a real-world environment with a live feed from the robot mounted Kinect V2 sensor. The system was then run and the FPD was swapped to another when materials were removed. Results of simulation were very promising with the system being capable of processing FPDs in between 110 and 230 seconds per unit.

The machine is currently ready to process FPDs. The software is nearly complete, and the vision is reliable, but the system cannot do the job required due to the robot and tooling. These should be relatively simple tasks to accomplish as the driver utilized allows for easy replacement of the robot and the tooling is relatively simple to design..

6.2 Future Work

To finalize this system, there are several key limitations which will need to be addressed. These limitations are easily addressable but stop the system from being a full solution. Many of these limitations and potential solutions to them have been mentioned in the body of the thesis. The future work is as follows in order of importance:

- Remove tooling chatter by replacing the robot system. The 50kg payload robot is not strong or rigid enough to handle milling loads and will require improvement.
- Increase fixture rigidity
- Improve neural network performance by increasing the dataset and including images under different lighting conditions.
- Include states to check for task success and recover from failures. In the current implementation, if any state fails it automatically unloads the conveyor and restarts with a new FPD. Each state needs to be recoverable to ensure that the system can complete the dismantling process. For example, if the system fails to remove the screen it should check and try again.
- Add states to allow for automated conveyor loading. All the components required to run this state are available, but the process needs to be implemented and tested.
- Reduce and/or collect cutting debris and tool chips to reduce image noise.
- Further testing and optimization of cutting paths and state machine.

Bibliography

1. K. Button, "20 Staggering E-Waste Facts - Earth911.com", Earth911.com, 2016. [Online]. Available: http://earth911.com/eco-tech/20-e-waste-facts/. [Accessed: 25- Nov- 2016].
2. Electronic Products Recycling Association. “Annual Report 2014.” Electronic Products Recycling Association (EPRA). Electronic Products Recycling Association, 2015. Web. 08 Oct. 2015.
3. Electronic Products Stewardship. “Design for Environment Report 2015 and Standard (ERS) 2015.” Electronic Products Stewardship Canada (EPSC). Electronic Products Stewardship Canada, 2015. Web. 08 Oct. 2015.
4. Ontario Electronics Stewardship. “2014 Annual Report.” Ontario Electronics Stewardship. Ontario Electronics Stewardship, 2014. Web. 08 Oct. 2015
5. H. Thompson, "Flat Panel TV sales growth expected to grow 4% in 2011", Digital Home, 2011. [Online]. Available: http://www.digitalhome.ca/2011/05/flat-panel-tv-sales-growth-expected-to-grow-4-in-2011/. [Accessed: 16- Nov- 2016].
6. Y. Li and L. Jin, "Environmental Release of Mercury from Broken Compact Fluorescent Lamps", Environmental Engineering Science, vol. 28, no. 10, pp. 687-691, 2011. Available: 10.1089/ees.2011.0027 [Accessed 11 November 2018].
7. "LAMP AND FLAT PANEL DISPLAY RECYCLING", Blubox.ch, 2019. [Online]. Available: https://www.blubox.ch/technologies/blubox/blubox-lamp-flat-panel-display-recycling. [Accessed: 05- Dec- 2018].
8. "LCD screen recycling", Erdwich.com, 2019. [Online]. Available: https://www.erdwich.com/en/application-areas/lcd-monitors/. [Accessed: 05- Dec- 2018].
9. "TVs ‘Moscow’ in the recycling bin", Veolia UK, 2018. [Online]. Available: https://www.veolia.co.uk/press-releases/tvs-moscow-recycling-bin. [Accessed: 13- Jan- 2019].
10. "Recycling technology : TOMRA", Tomra.com. [Online]. Available: https://www.tomra.com/en/sorting/recycling/recycling-technology. [Accessed: 11- Jan- 2019].
11. A. Sharifi, H. Karbasi, A. Sanderson, C. Wilson, “Robotic Sorting of Shredded E-waste: Utilizing Deep Learning”, Submitted to 20th International Conference on Artificial Intelligence (ICAI 2018), Las Vegas, NV, USA, Jul. 30 – Aug. 2, 2018

12. B. James, "Developing a vision system to value waste PCBs", Cambridge Consultants, 2018. [Online]. Available: https://www.cambridgeconsultants.com/insights/developing-vision-system-value-waste-pcbs. [Accessed: 15- Jan- 2019].

13. C. Pramerdorfer and M. Kampel: “A Dataset for Computer-Vision-Based PCB Analysis”, Machine Vision Applications, 2015

14. "VISIONPRO | Cognex", Cognex.com. [Online]. Available: https://www.cognex.com/products/machine-vision/vision-software/visionpro-software. [Accessed: 15- Dec- 2018].

15. "Industrial Robot Software | FANUC America", Fanucamerica.com. [Online]. Available: https://www.fanucamerica.com/products/robots/software. [Accessed: 15- Sep- 2016].

16. S. Chitta, I. Sucan and S. Cousins, "MoveIt! [ROS Topics]", IEEE Robotics & Automation Magazine, vol. 19, no. 1, pp. 18-19, 2012. Available: 10.1109/mra.2011.2181749.

17. "Open EtherCAT Society: Home of SOEM and SOES", Openethercatsociety.github.io. [Online]. Available: http://openethercatsociety.github.io/. [Accessed: 05- Dec- 2018].

18. "ROS.org | Core Components", Ros.org. [Online]. Available: http://www.ros.org/core-components/. [Accessed: 02- Nov- 2018].

19. J. Pan and D. Manocha. (2011). The Fast Collision Library (FCL) [Online]. Available:http://www.ros.org/wiki/fcl_ros

20. I. A. Sucan, M. Moll, and L. Kavraki. (2010).The Open Motion Planning Library (OMPL),[Online]. Available: http://ompl.kavrakilab.org

21. N. Ratliff, M. Zucker, J. A. D. Bagnell, andS. Srinivasa, “CHOMP: Gradient optimizationtechniques for efficient motion planning,” inProc. Int. Conf. Robotics and Automation, May2009, pp. 489–494.

22. M. Kalakrishnan, S. Chitta, E. Theodorou,P. Pastor, and S. Schaal, “STOMP: Stochastictrajectory optimization for motion planning,”in Proc. Int. Conf. Robotics and Automation,Shanghai, China, May 2011.

23. J. Kuffner and S.M. LaValle, “RRT-connect: An efficient approach to single-query path planning”, in Proc. 2000 IEEE Intl. Conf. on Robotics and Automation, pp. 995–1001, Apr. 2000. DOI: 10.1109/ROBOT.2000.844730

24. I.A. Şucan and L.E. Kavraki, “Kinodynamic motion planning by interior-exterior cell exploration”, in Workshop on the Algorithmic Foundations of Robotics, Dec. 2008

25. Perkowski Marek, Digital design automation: finite state machine design, dept. of electrical engineering, Portland State University.

26. "smach - ROS Wiki", *Wiki.ros.org*. [Online]. Available: http://wiki.ros.org/smach. [Accessed: 11- Nov- 2018].

27. "OpenCV library", Opencv.org. [Online]. Available: https://www.opencv.org/. [Accessed: 09- Dec- 2019].

28. H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 328-341, 2008. Available: 10.1109/tpami.2007.1166.

29. "stereo_image_proc - ROS Wiki", *Wiki.ros.org*. [Online]. Available: http://wiki.ros.org/stereo_image_proc. [Accessed: 19- Nov- 2018].

30. J. Pagès and J. Salvi, "Coded light projection techniques for 3D reconstruction", J3eA, vol. 4, p. 001, 2005. Available: 10.1051/bib-j3ea:2005801.

31. "structured-light", Code.google.com. [Online]. Available: https://code.google.com/archive/p/structured-light/. [Accessed: 22- Nov- 2018].

32. Y. He, B. Liang, Y. Zou, J. He and J. Yang, "Depth Errors Analysis and Correction for Time-of-Flight (ToF) Cameras", Sensors, vol. 17, no. 1, p. 92, 2017. Available: 10.3390/s17010092.

33. "Technology White Paper: Laser Line Scanning | Quality Digest", Qualitydigest.com, 2009. [Online]. Available: https://www.qualitydigest.com/inside/twitter-ed/technology-white-paper-laser-line-scanning.html. [Accessed: 17- Dec- 2018].

34. S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” CoRR, vol. abs/1506.01497, 2015

35. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” CoRR, vol. abs/1611.10012, 2016.

36. K.He,X.Zhang,S.Ren,andJ.Sun, “Deep residual learning for image recognition,” CoRR, vol. abs/1512.03385, 2015.

37. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2014.

38. [Ben-David and Schuller, 2003] Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In Proc COLT, pages 567-580.
39. R. B. Girshick, “Fast R-CNN,” CoRR, vol. abs/1504.08083, 2015.
40. V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481-2495, 2017. Available: 10.1109/tpami.2016.2644615.
41. L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834-848, 2018. Available: 10.1109/tpami.2017.2699184.
42. Satoshi Suzuki and others. “Topological structural analysis of digitized binary images by border following”. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.
43. Prasad, Dilip K.; Leung, Maylor K.H.; Quek, Chai; Cho, Siu-Yeung (2012). "A novel framework for making dominant point detection methods non-parametric". Image and Vision Computing. 30 (11): 843–859. doi:10.1016/j.imavis.2012.06.010
44. Jiri Matas, Charles Galambos, and Josef Kittler. “Robust detection of lines using the progressive probabilistic hough transform”. Computer Vision and Image Understanding, 78(1):119–137, 2000.
45. R. Kim, "Processor Incentive Program", Ontario Electronic Stewardship, 2014. [Online]. Available: http://ontarioelectronicstewardship.ca/service-providers/recycler-or-processor/processor-incentive-program/. [Accessed: 02- Jan- 2019].

Appendix A

Robot Data Sheet

M-710*i*C™ Series

FANUC Robotics

Basic Description

The M-710*i*C series is FANUC Robotics' latest-generation, six-axis, medium to medium-high payload, high-performance family of industrial robots. The M-710*i*C robots are designed for a variety of manufacturing and system processes. The M-710*i*C series provides one of the largest work envelopes in its class with one of the smallest footprints.

Engineered for extremely high speed, application flexibility and reliability, the M-710*i*C series delivers repeatable precision and unparalleled performance. With the FoundryPRO® option, the entire robot is IP67-protected for the M-710*i*C series, making these robots ideal for use in harsh environments.

M-710*i*C Series, the Solution for:

- Material handling
- Machine load/unload
- Parts transfer
- Assembly
- Waterjet cutting/routering
- Foundry applications
- Material removal
- Thermal spray
- Packing & palletizing
- Dispensing
- Arc welding

Benefits

- Multiple mounting solutions (see specifications for robot dependent configurations).

Note: FoundryPRO is a registered trademark of FANUC LTD.

- Slim arm and wrist assemblies minimize interference with system peripherals allowing operation in confined spaces.
- Generous supplementary J3 payload up to 15 kg.
- Pneumatic and electrical connections (8RI/8RO) available on front of J3 casting for easy and simple integration of end-of-arm tooling (EOAT).
- Multiple process attachment points allowing flexibility when integrating EOAT.
- Superior wrist capacity for handling heavy work pieces and fixtures.
- Best joint speeds in its class maximize cycle time and throughput.
- Wrist flange and base mounting are identical to the M-710*i*B series, allowing for seamless backward compatibility.
- Larger work envelope than the M-710*i*B series, making replacement easier.
- Designed to use minimal mechanical components to reduce down time, increase mean time between failure and minimize spare parts.
- Utilizes proven, reliable FANUC servo motors providing the greatest uptime and productivity.

Features

Mechanical:

- Given the ability to reach overhead and behind, the M-710*i*C series has one of the largest work envelopes in its class.
- 6 degrees of freedom.
- RV reducers for all axes eliminates belts, chains and pulleys and minimizes backlash.

- No motors at the wrist.
- Increased work envelope realized by the elimination of counterweight/balancer.
- 360° standard J1 rotation.

Software:

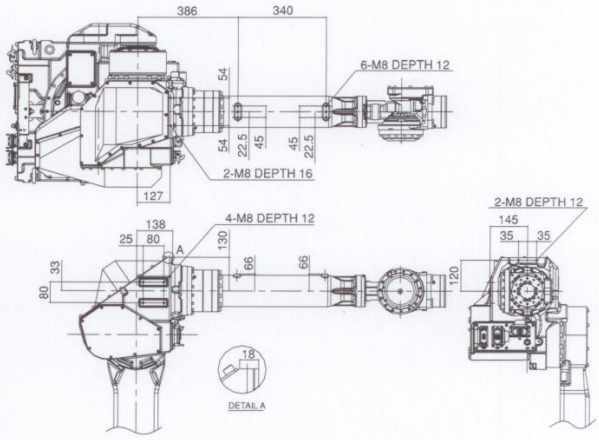
- Process specific software packages for various applications.
- Web-based software tools for remote connectivity, diagnostics and production monitoring.
- Built-in support for machine vision applications for error-proofing and robot guidance, without the need for a PC.
- *i*RVision™ (Integrated Robot Vision) system delivers high-performance 2-D and 3-D machine vision capabilities with FANUC reliability.

Options

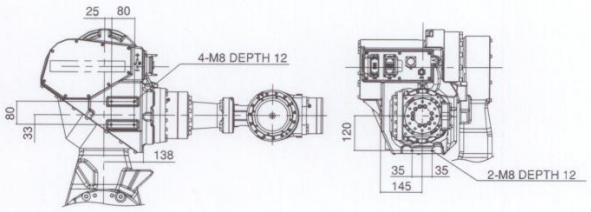
- Standard baseplate for quick robot installation.
- Auxiliary axes packages for integration of peripheral servo-controlled devices.
- Adjustable hard stops for J1, J2 and J3.
- Various robot connection cable lengths for flexible cabinet placement and optional track rated cables.
- Two-part epoxy paint available for harsh environments.
- Monochrome pendant available.
- FoundryPRO option available.
- ± 185 rotation for J1.

Auxiliary Equipment Mounting Provisions

M-710*i*C/50 & M-710*i*C/70

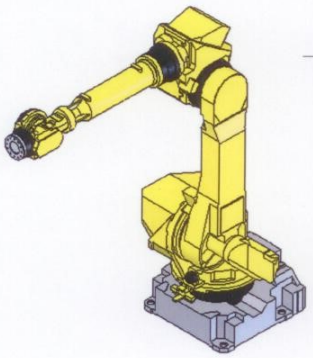


M-710*i*C/50S

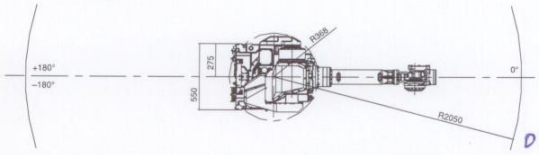


M-710*i*C/50 & M-710*i*C/70 Dimensions

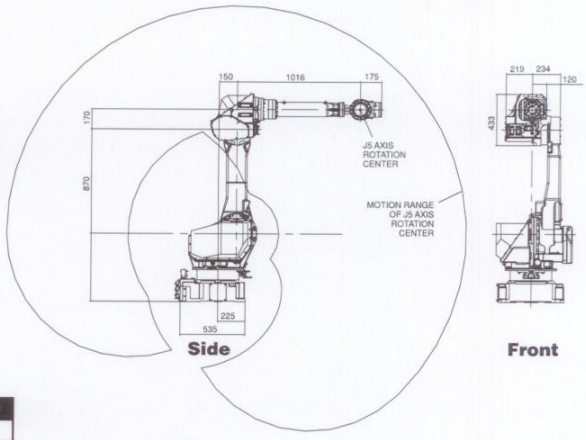
Isometric



Top



D 4100

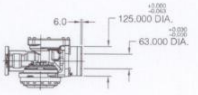


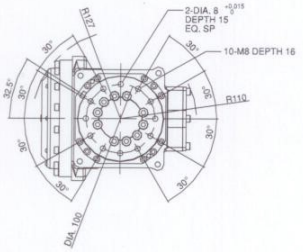
M-710*i*C/50 & M-710*i*C/70 Specifications

Items		M-710*i*C/50	M-710*i*C/70
Axes		6	6
Payload (kg)		50	70
Payload (kg) on J3 casting		15	15
Reach (mm)		2050	2050
Repeatability (mm)		±0.07	±0.07
Interference radius (mm)		368	368
Motion range (degrees)	J1	320/360/370	320/360/370
	J2	225	225
	J3	440	440
	J4	720	720
	J5	250	250
	J6	720	720
Motion speed (degrees/s)	J1	175	160
	J2	175	120
	J3	175	120
	J4	250	225
	J5	250	225
	J6	355	225
Wrist moment N·m (kgf·m)	J4	206 (21)	294 (30)
	J5	206 (21)	294 (30)
	J6	127 (13)	147 (15)
Wrist inertia (kg·m²)	J4	28	28
	J5	28	28
	J6	11	11
Mechanical brakes		All axes	
Mechanical weight (kg)(1)		560	560
Mounting method(2)		Floor, ceiling, angle and wall	
Installation environment:			
Ambient temperature °C		0 to 45	
Humidity		Normally: 75% or less Short term (within a month): 95% or less. No condensation	
Vibration m/s² (G)		4.9 m/s² or less (0.5G or less)	
IP Rating(s)		Body IP54 Std. (IP67 optional) Wrist and joint 3 arm IP67	

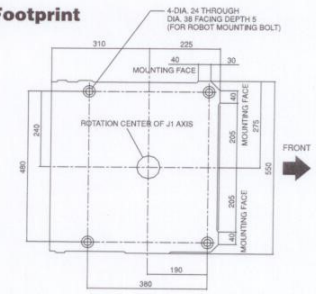
Notes:
(1) Without controller.
(2) J1 and J2 axis motion range will be limited for ceiling, angle and wall. See manual under installation condition.

Faceplate



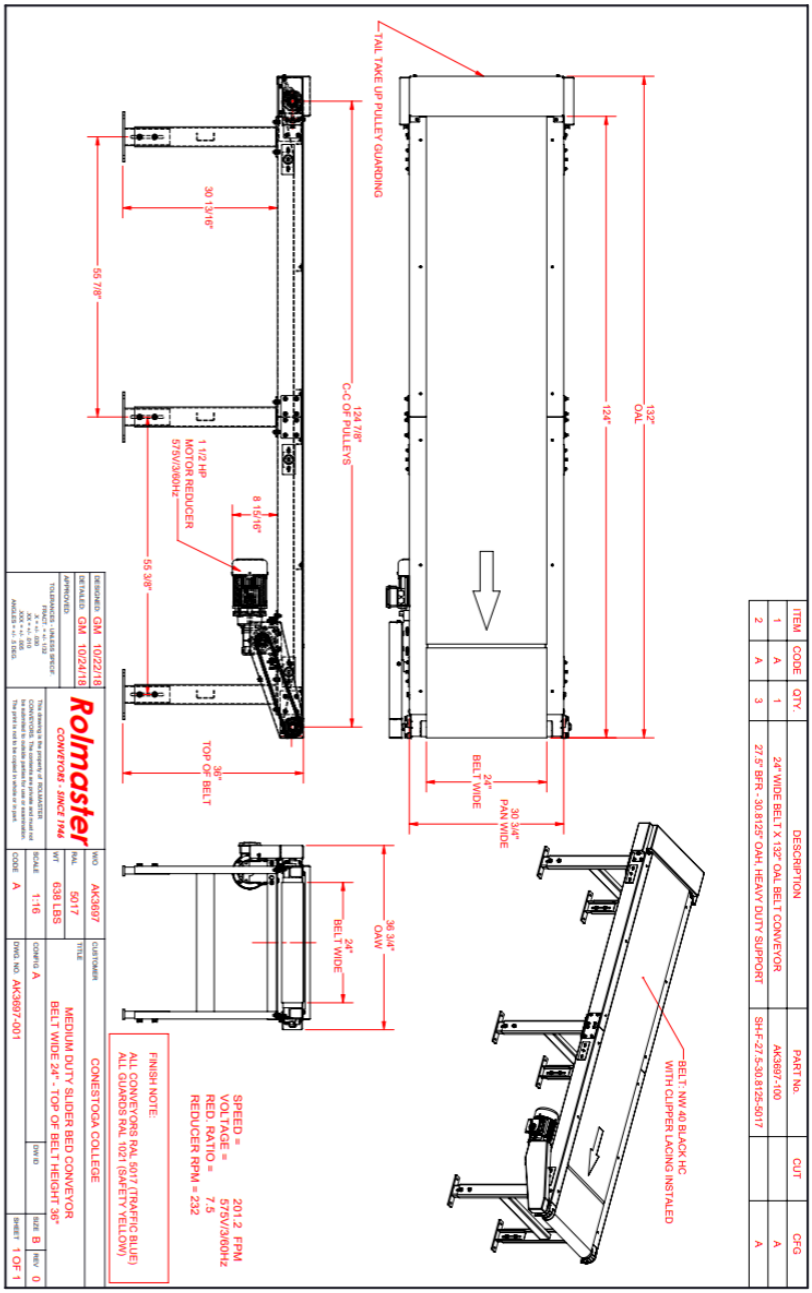


Footprint



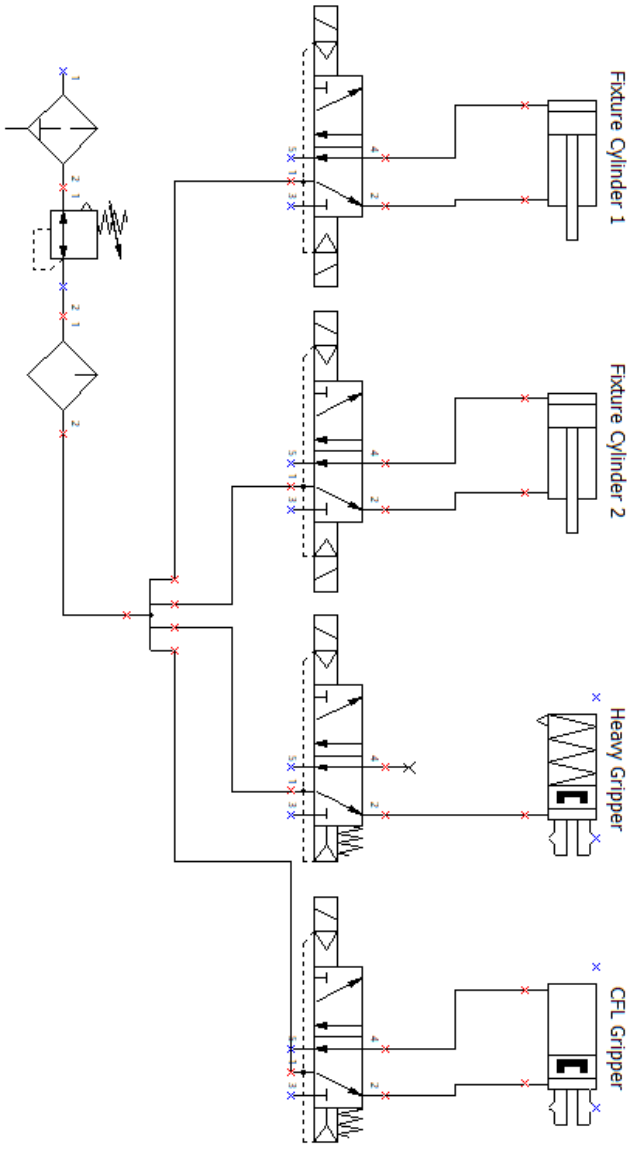
Appendix B

Conveyor Drawing

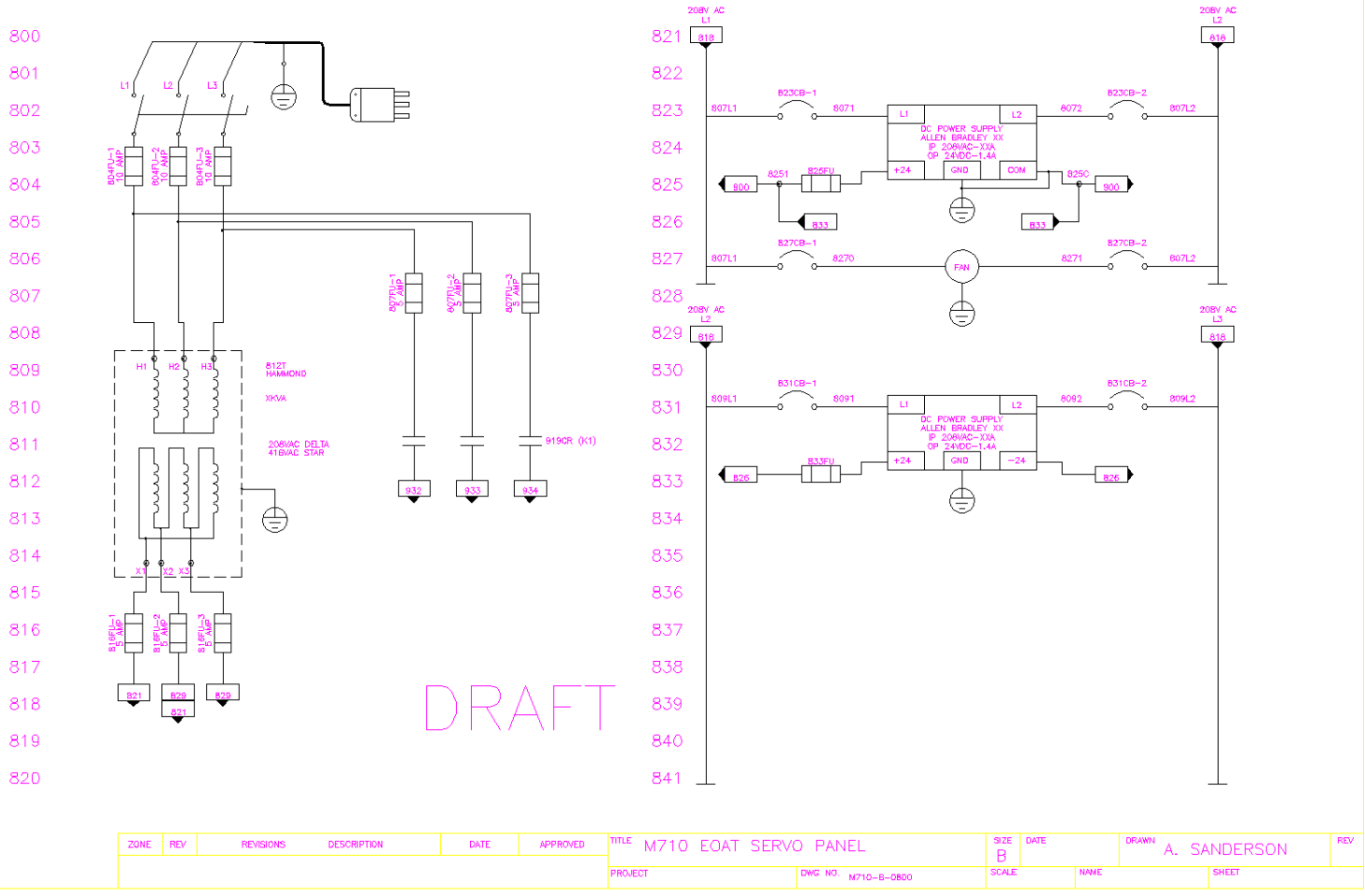


Appendix C

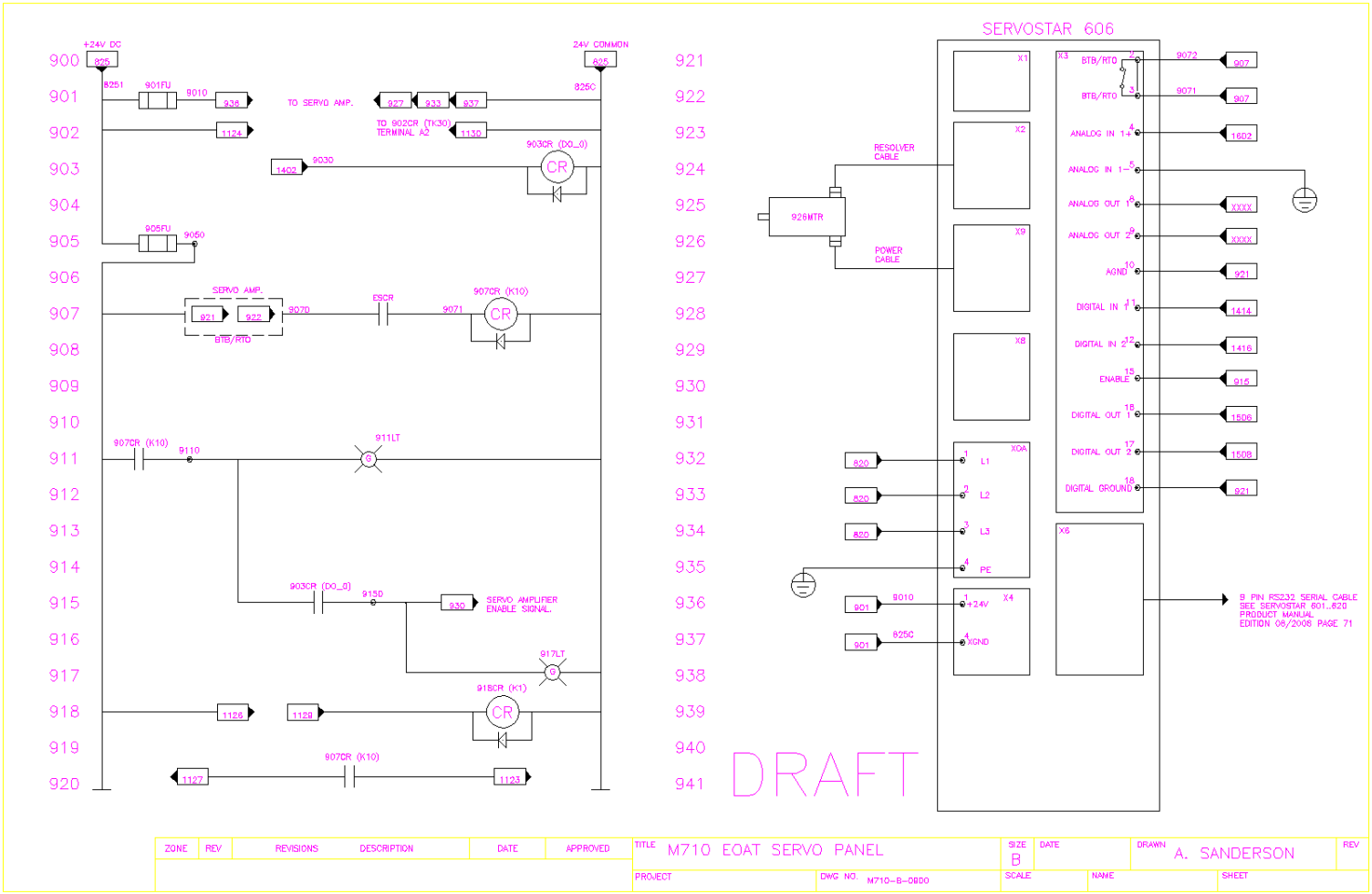
Pneumatic Schematic

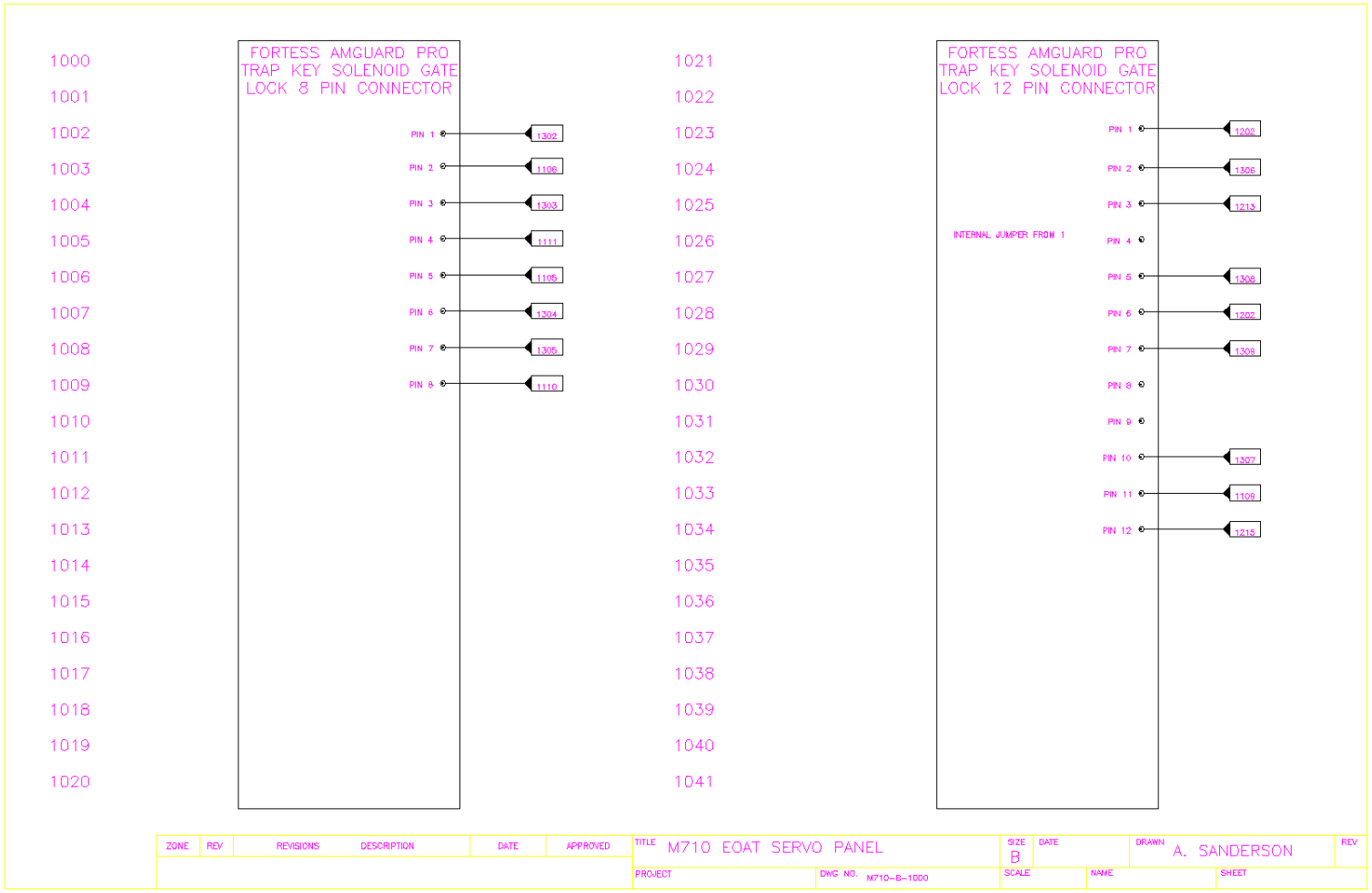


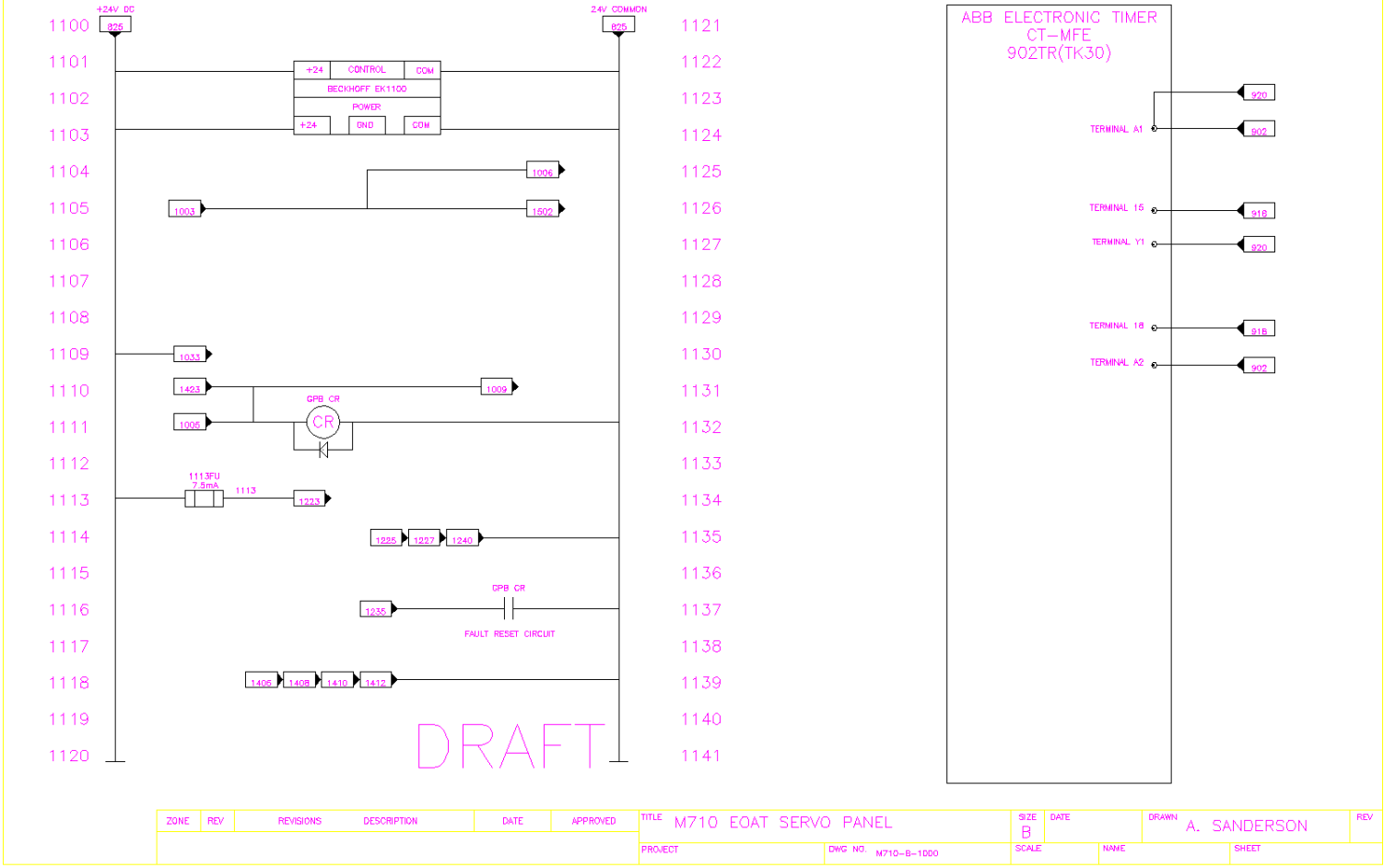
Appendix D
Electrical Diagram

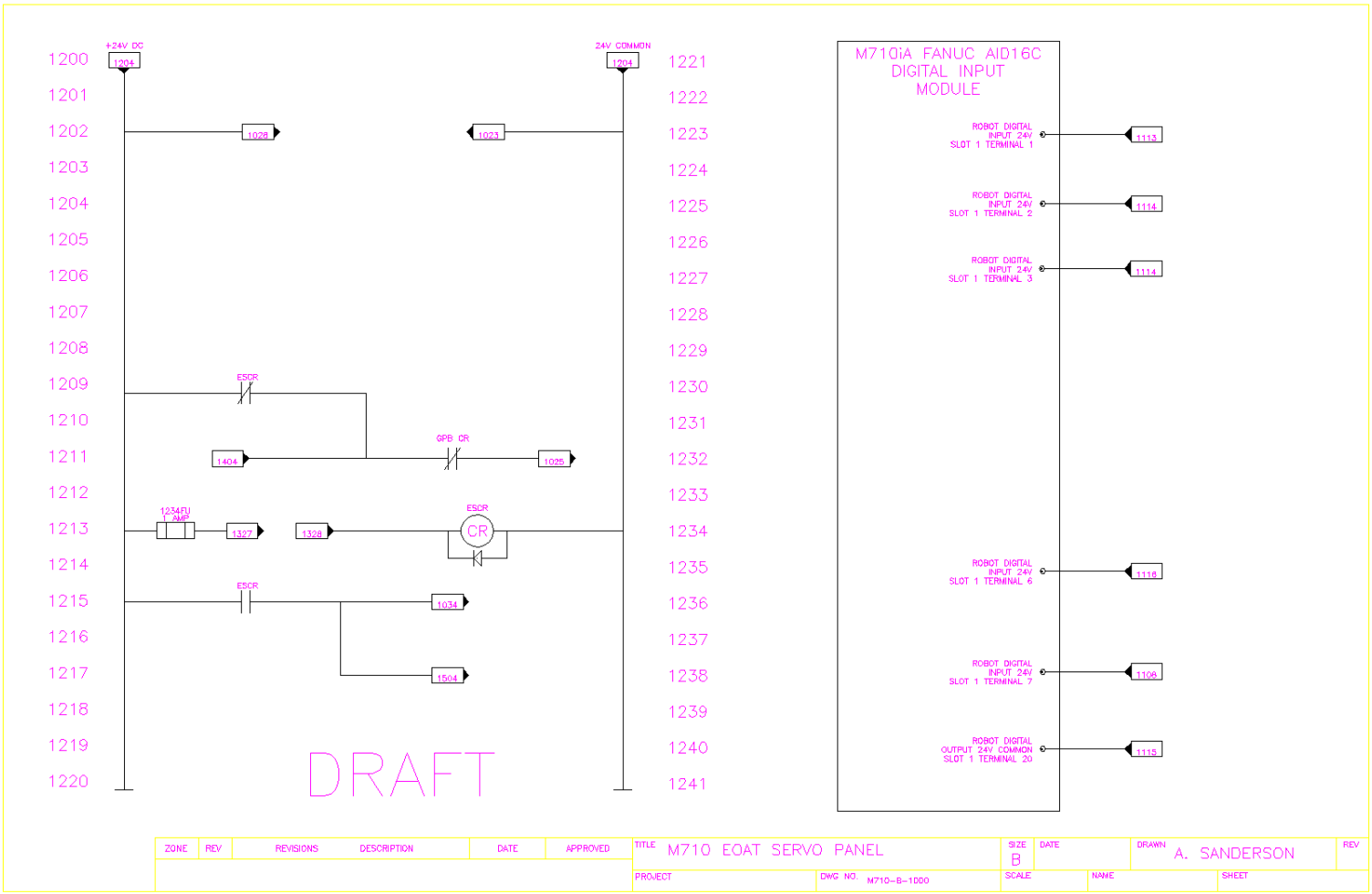


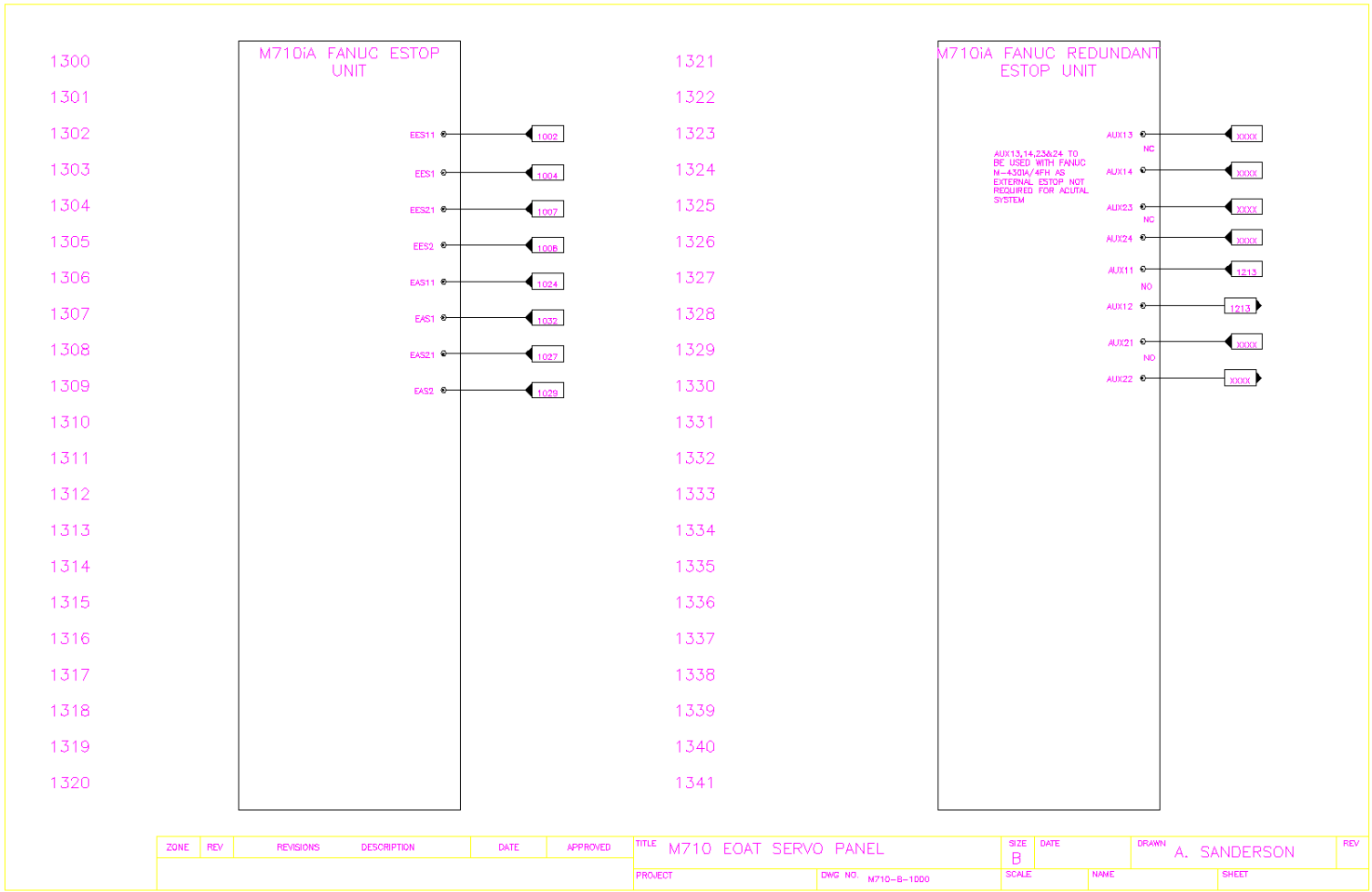
ZONE	REV	REVISIONS DESCRIPTION	DATE	APPROVED	TITLE M710 EOAT SERVO PANEL	SIZE B	DATE	DRAWN A. SANDERSON	REV
					PROJECT	DWG NO. M710-B-0800	SCALE	NAME	SHEET

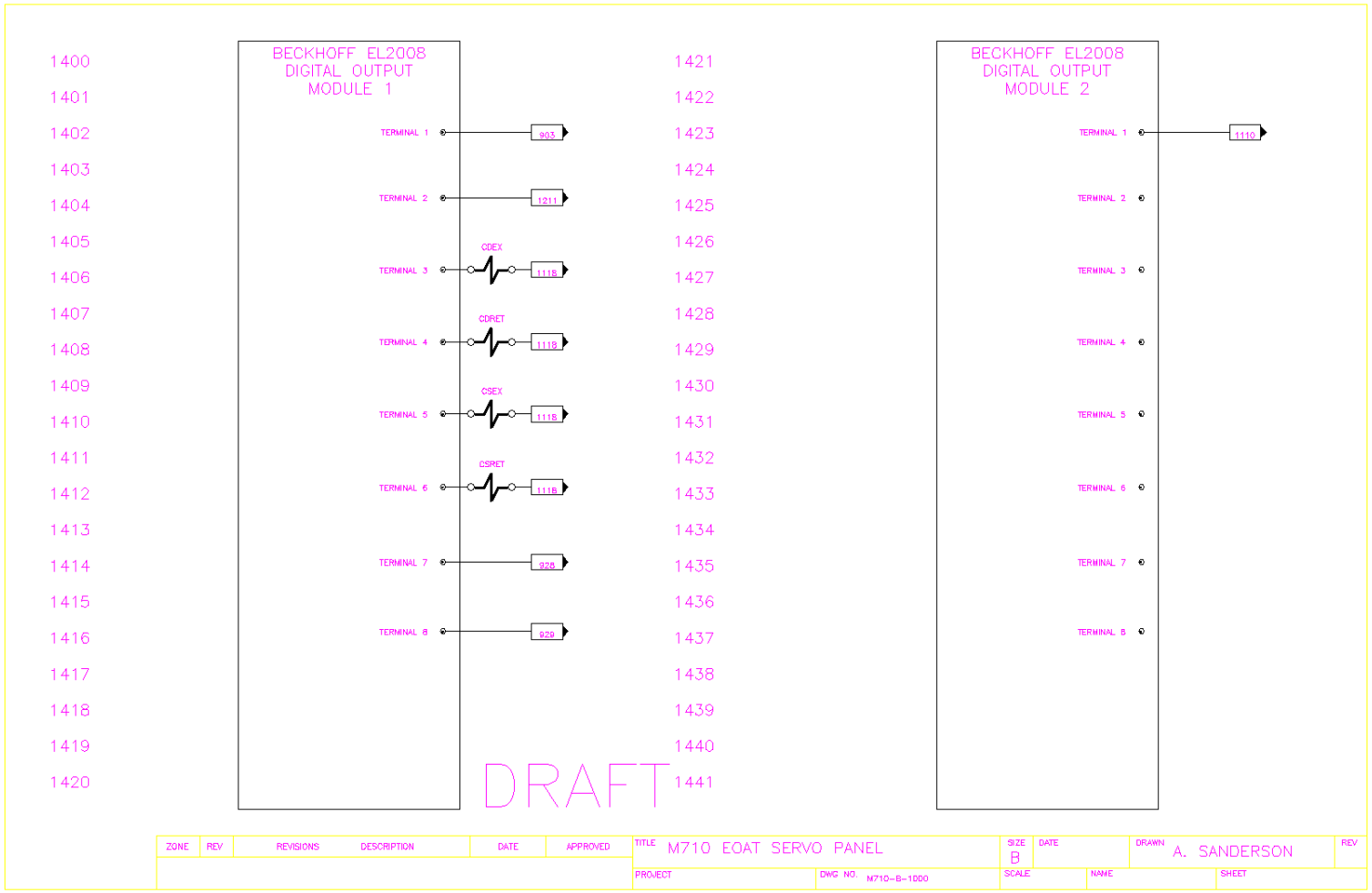

+24V DC
24V COMMON
TO SERVO AMP.
TO 902CR (TK30) TERMINAL A2
903CR (DO_0)
SERVO AMP.
BTB/RTD
907CR (K10)
ESCR
911LT
SERVO AMPLIFIER ENABLE SIGNAL
917LT
918CR (K1)
SERVOSTAR 606
RESOLVER CABLE
POWER CABLE
926MTR
BTB/RTO
ANALOG IN 1+
ANALOG IN 1-
ANALOG OUT 1
ANALOG OUT 2
AGND
DIGITAL IN 1
DIGITAL IN 2
ENABLE
DIGITAL OUT 1
DIGITAL OUT 2
DIGITAL GROUND
L1
L2
L3
PE
+24V
XGND
9 PIN RS232 SERIAL CABLE SEE SERVOSTAR 601..620 PRODUCT MANUAL EDITION 08/2008 PAGE 71
DRAFT
M710 EOAT SERVO PANEL
A. SANDERSON
M710-B-0000

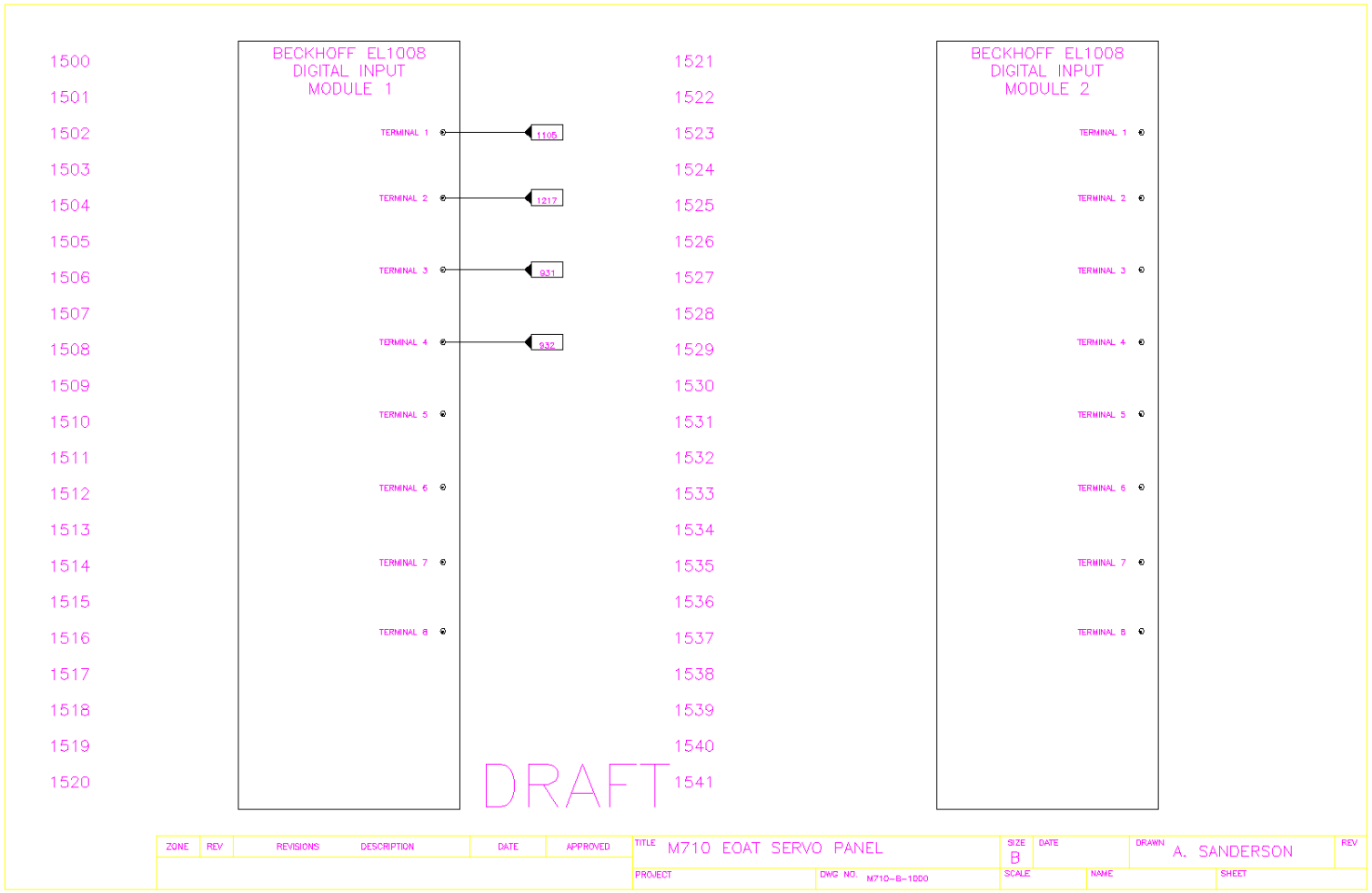

1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
FORTESS AMGUARD PRO TRAP KEY SOLENOID GATE LOCK 8 PIN CONNECTOR
PIN 1
PIN 2
PIN 3
PIN 4
PIN 5
PIN 6
PIN 7
PIN 8
1302
1106
1303
1111
1105
1304
1305
1110
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
FORTESS AMGUARD PRO TRAP KEY SOLENOID GATE LOCK 12 PIN CONNECTOR
INTERNAL JUMPER FROM 1
PIN 1
PIN 2
PIN 3
PIN 4
PIN 5
PIN 6
PIN 7
PIN 8
PIN 9
PIN 10
PIN 11
PIN 12
1202
1306
1213
1306
1202
1309
1307
1109
1215
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE
M710 EOAT SERVO PANEL
SIZE
B
DATE
DRAWN
A. SANDERSON
REV
PROJECT
DWG NO.
M710-B-1000
SCALE
NAME
SHEET

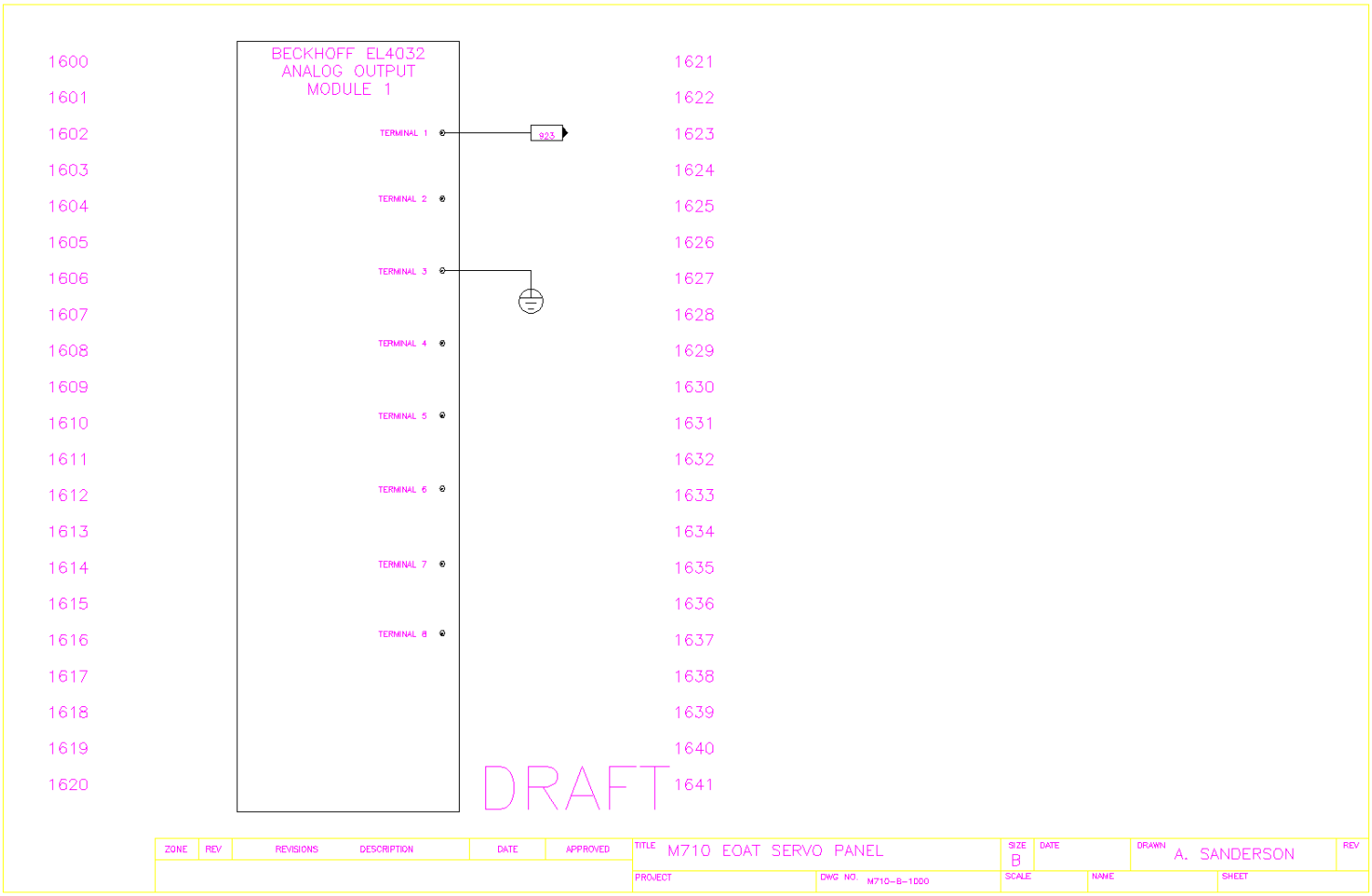

+24V DC
925
24V COMMON
925
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
+24
CONTROL
COM
BECKHOFF EK1100
POWER
+24
GND
COM
1006
1003
1502
1033
1423
1009
GPB CR
CR
1005
1113FU
7.5mA
1113
1223
1225
1227
1240
GPB CR
1235
FAULT RESET CIRCUIT
1406
1408
1410
1412
DRAFT
ABB ELECTRONIC TIMER
CT-MFE
902TR(TK30)
920
TERMINAL A1
902
TERMINAL 15
918
TERMINAL Y1
920
TERMINAL 18
918
TERMINAL A2
902
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE
M710 EOAT SERVO PANEL
SIZE
B
DATE
DRAWN
A. SANDERSON
REV
PROJECT
DWG NO.
M710-B-1000
SCALE
NAME
SHEET


+24V DC
1204
24V COMMON
1204
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1028
1023
ESCR
1404
GPB CR
1025
1234FU
1 AMP
1327
1328
ESCR
CR
ESCR
1034
1504
DRAFT
M710iA FANUC AID16C
DIGITAL INPUT
MODULE
ROBOT DIGITAL
INPUT 24V
SLOT 1 TERMINAL 1
1113
ROBOT DIGITAL
INPUT 24V
SLOT 1 TERMINAL 2
1114
ROBOT DIGITAL
INPUT 24V
SLOT 1 TERMINAL 3
1114
ROBOT DIGITAL
INPUT 24V
SLOT 1 TERMINAL 6
1116
ROBOT DIGITAL
INPUT 24V
SLOT 1 TERMINAL 7
1106
ROBOT DIGITAL
OUTPUT 24V COMMON
SLOT 1 TERMINAL 20
1115
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE
M710 EOAT SERVO PANEL
SIZE
B
DATE
DRAWN
A. SANDERSON
REV
PROJECT
DWG NO.
M710-B-1000
SCALE
NAME
SHEET

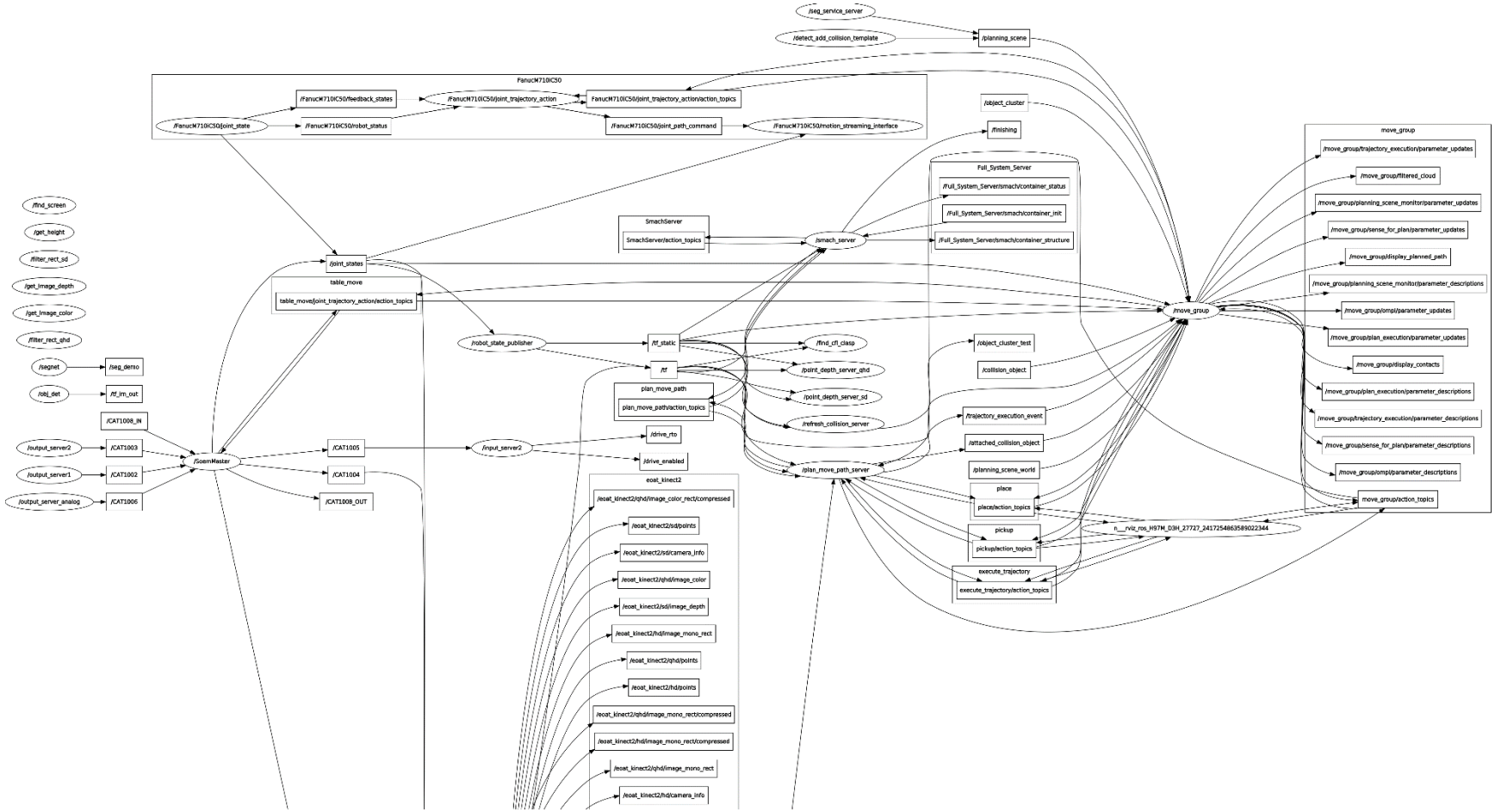

1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
M710iA FANUC ESTOP UNIT
EES11
1002
EES1
1004
EES21
1007
EES2
1006
EAS11
1024
EAS1
1032
EAS21
1027
EAS2
1029
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
M710iA FANUC REDUNDANT ESTOP UNIT
AUX13,14,23&24 TO BE USED WITH FANUC N-430IA/4FH AS EXTERNAL ESTOP NOT REQUIRED FOR ACUTAL SYSTEM
AUX13
NC
XXXX
AUX14
XXXX
AUX23
NC
XXXX
AUX24
XXXX
AUX11
NO
1213
AUX12
1213
AUX21
NO
XXXX
AUX22
XXXX
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE
M710 EOAT SERVO PANEL
SIZE
B
DATE
DRAWN
A. SANDERSON
REV
PROJECT
DWG NO.
M710-B-1000
SCALE
NAME
SHEET

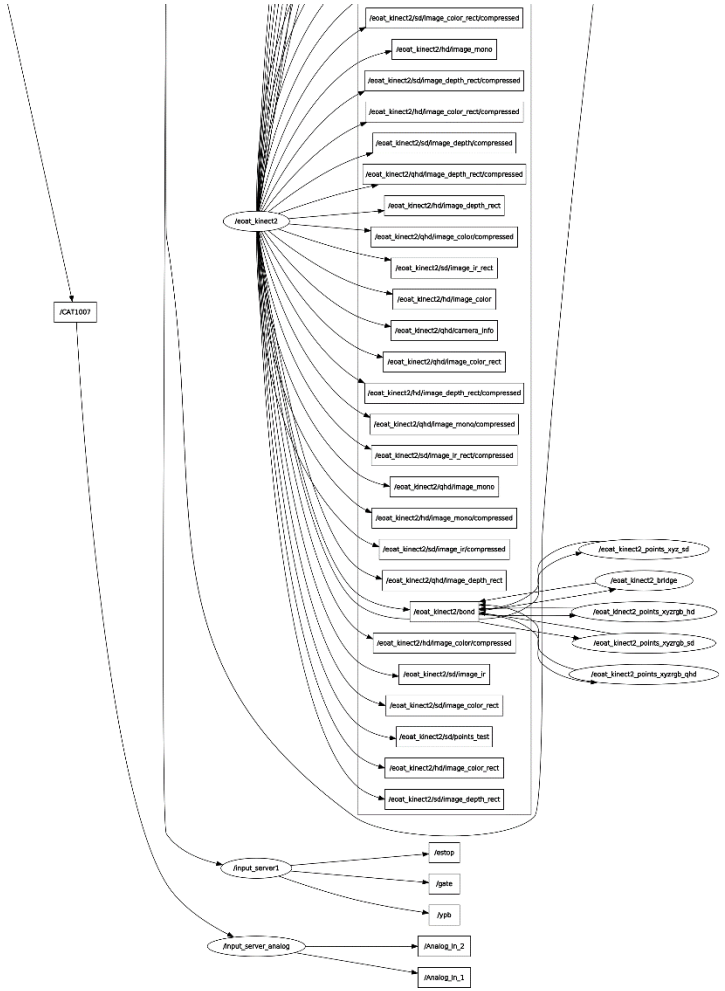

1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
BECKHOFF EL2008
DIGITAL OUTPUT
MODULE 1
TERMINAL 1
903
TERMINAL 2
1211
CDEX
TERMINAL 3
1118
CDRET
TERMINAL 4
1119
CSEX
TERMINAL 5
1115
CSRET
TERMINAL 6
1116
TERMINAL 7
928
TERMINAL 8
929
DRAFT
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
BECKHOFF EL2008
DIGITAL OUTPUT
MODULE 2
TERMINAL 1
1110
TERMINAL 2
TERMINAL 3
TERMINAL 4
TERMINAL 5
TERMINAL 6
TERMINAL 7
TERMINAL 8
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE
M710 EOAT SERVO PANEL
SIZE
B
DATE
DRAWN
A. SANDERSON
REV
PROJECT
DWG NO.
M710-B-1000
SCALE
NAME
SHEET


1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
BECKHOFF EL1008
DIGITAL INPUT
MODULE 1
TERMINAL 1
TERMINAL 2
TERMINAL 3
TERMINAL 4
TERMINAL 5
TERMINAL 6
TERMINAL 7
TERMINAL 8
1105
1217
931
932
DRAFT
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
BECKHOFF EL1008
DIGITAL INPUT
MODULE 2
TERMINAL 1
TERMINAL 2
TERMINAL 3
TERMINAL 4
TERMINAL 5
TERMINAL 6
TERMINAL 7
TERMINAL 8
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE M710 EOAT SERVO PANEL
SIZE B
DATE
DRAWN A. SANDERSON
REV
PROJECT
DWG NO. M710-B-1000
SCALE
NAME
SHEET


1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
BECKHOFF EL4032
ANALOG OUTPUT
MODULE 1
TERMINAL 1
TERMINAL 2
TERMINAL 3
TERMINAL 4
TERMINAL 5
TERMINAL 6
TERMINAL 7
TERMINAL 8
823
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
DRAFT
ZONE
REV
REVISIONS
DESCRIPTION
DATE
APPROVED
TITLE M710 EOAT SERVO PANEL
SIZE B
DATE
DRAWN A. SANDERSON
REV
PROJECT
DWG NO. M710-B-1000
SCALE
NAME
SHEET

Appendix E
Computational Graph


/seg_service_server
/detect_add_collision_template
/planning_scene
FanucM710iC50
/FanucM710iC50/feedback_states
/FanucM710iC50/joint_trajectory_action
FanucM710iC50/joint_trajectory_action/action_topics
/FanucM710iC50/joint_state
/FanucM710iC50/robot_status
/FanucM710iC50/joint_path_command
/FanucM710iC50/motion_streaming_interface
/object_cluster
/finishing
Full_System_Server
/Full_System_Server/smach/container_status
/Full_System_Server/smach/container_init
/Full_System_Server/smach/container_structure
move_group
/move_group/trajectory_execution/parameter_updates
/move_group/filtered_cloud
/move_group/planning_scene_monitor/parameter_updates
/move_group/sense_for_plan/parameter_updates
/move_group/display_planned_path
/move_group/planning_scene_monitor/parameter_descriptions
/move_group/ompl/parameter_updates
/move_group/plan_execution/parameter_updates
/move_group/display_contacts
/move_group/plan_execution/parameter_descriptions
/move_group/trajectory_execution/parameter_descriptions
/move_group/sense_for_plan/parameter_descriptions
/move_group/ompl/parameter_descriptions
move_group/action_topics
/find_screen
/get_height
/filter_rect_sd
/get_image_depth
/get_image_color
/filter_rect_qhd
/segnet
/seg_demo
/obj_det
/tf_im_out
/CAT1008_IN
/output_server2
/output_server1
/output_server_analog
/CAT1003
/CAT1002
/CAT1006
/SoemMaster
/CAT1005
/CAT1004
/CAT1008_OUT
/joint_states
table_move
table_move/joint_trajectory_action/action_topics
SmachServer
SmachServer/action_topics
/smach_server
/robot_state_publisher
/tf_static
/tf
/input_server2
/drive_rto
/drive_enabled
plan_move_path
plan_move_path/action_topics
/plan_move_path_server
eoat_kinect2
/eoat_kinect2/qhd/image_color_rect/compressed
/eoat_kinect2/sd/points
/eoat_kinect2/sd/camera_info
/eoat_kinect2/qhd/image_color
/eoat_kinect2/sd/image_depth
/eoat_kinect2/hd/image_mono_rect
/eoat_kinect2/qhd/points
/eoat_kinect2/hd/points
/eoat_kinect2/qhd/image_mono_rect/compressed
/eoat_kinect2/hd/image_mono_rect/compressed
/eoat_kinect2/qhd/image_mono_rect
/eoat_kinect2/hd/camera_info
/find_cfl_clasp
/point_depth_server_qhd
/point_depth_server_sd
/refresh_collision_server
/object_cluster_test
/collision_object
/trajectory_execution_event
/attached_collision_object
/planning_scene_world
place
place/action_topics
pickup
pickup/action_topics
execute_trajectory
execute_trajectory/action_topics
/move_group
n__rviz_ros_H97M_D3H_27727_241725486358902344


/eoat_kinect2/sd/image_color_rect/compressed
/eoat_kinect2/hd/image_mono
/eoat_kinect2/sd/image_depth_rect/compressed
/eoat_kinect2/hd/image_color_rect/compressed
/eoat_kinect2/sd/image_depth/compressed
/eoat_kinect2/qhd/image_depth_rect/compressed
/eoat_kinect2/hd/image_depth_rect
/eoat_kinect2
/eoat_kinect2/qhd/image_color/compressed
/eoat_kinect2/sd/image_ir_rect
/eoat_kinect2/hd/image_color
/eoat_kinect2/qhd/camera_info
/eoat_kinect2/qhd/image_color_rect
/eoat_kinect2/hd/image_depth_rect/compressed
/eoat_kinect2/qhd/image_mono/compressed
/eoat_kinect2/sd/image_ir_rect/compressed
/eoat_kinect2/qhd/image_mono
/eoat_kinect2/hd/image_mono/compressed
/eoat_kinect2/sd/image_ir/compressed
/eoat_kinect2/qhd/image_depth_rect
/eoat_kinect2_points_xyz_sd
/eoat_kinect2_bridge
/eoat_kinect2/bond
/eoat_kinect2_points_xyzrgb_hd
/eoat_kinect2_points_xyzrgb_sd
/eoat_kinect2_points_xyzrgb_qhd
/eoat_kinect2/hd/image_color/compressed
/eoat_kinect2/sd/image_ir
/eoat_kinect2/sd/image_color_rect
/eoat_kinect2/sd/points_test
/eoat_kinect2/hd/image_color_rect
/eoat_kinect2/sd/image_depth_rect
/CAT1007
/estop
/input_server1
/gate
/ypb
/input_server_analog
/Analog_In_2
/Analog_In_1