INVESTIGATING PRE-TOUCH SENSING TO PREDICT GRIP SUCCESS IN COMPLIANT GRIPPERS USING MACHINE LEARNING TECHNIQUES

BY

BENJAMIN THOMAS WALT

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Assistant Professor Girish Krishnan

Abstract

This work explores the application of pre-touch sensing to a compliant gripper in order to navigate the last few centimeters while grasping fruit in an occluded, cluttered environment. Machine learning was used in conjunction with pre-touch sensors to provide qualitative feedback about the success of the gripper in picking the target fruit prior to contact. Three compliant grippers were each designed to pick a specific fruit (miracle berries, cherry tomatoes and small figs) without damaging them. These grippers were designed to be mounted on the hybrid soft-rigid arm of a mobile field robot. An IR reflectance, time of flight and color sensor were used as pre-touch sensors and arranged on the gripper in various combinations to explore the contribution of each sensor. The gripper-sensor system was trained by positioning it relative to a dummy fruit using a 6 DOF arm and gripping the target. Using the training data, five machine learning methods were explored: nearest neighbor, decision trees, support vector machines, multi-layer perceptrons and a naïve Bayes classifier. The various sensor configuration-machine learning combinations were tested and evaluated based on their ability to predict grip success. Additional training was conducted to demonstrate the ability to differentiate fruit from foreign matter (e.g. leaves) that are in the gripper opening. Time of flight sensors using nearest neighbor and support vector machines along with the set of all three sensors using support vector machines and multi-layer perceptrons showed the highest prediction precision ($\approx 90\%$) with the color sensor playing a key role in detecting foreign objects. The machine learning methods were similar in their ability to predict grip success with nearest neighbor showing the best overall results, while sensor 'richness' play an important role in differentiating the sensors with the three sensor combination showing the best results.

This work is dedicated to my parents who have supported me every step of the way.

Acknowledgments

I would like to acknowledge everyone who has helped me complete this thesis. Firstly, I wish to express my sincere appreciation to my advisor, Professor Girish Krishnan, who has provided me with countless hours of advice and support as he has helped guide my development as a researcher. He has demonstrated his immense patience as he allowed me to freely explore a variety of topics and find an idea that I could explore fully. I am also indebted to all of my teachers who have put so much time and energy into me. They have instilled in me a curiosity and desire to learn and without them, it would not be possible for me to be where I am today. I would also like to thank the National Science Foundation (NSF), the United States Department of Agriculture (USDA) and the Department of Industrial and Enterprise Systems Engineering who have generously funded my research. Additionally, I would like to express my thanks to my lab mates and classmates who have helped me learn and grow in countless ways. In particular, I must thank my lab mate Naveen, who has been a mentor in the day-to-day workings of research. Finally, I would like thank my parents, who have been a constant source of support and encouragement. Returning to school to pursue this degree, would not have been possible without their help and advice. I am, as always, immensely indebted to them.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

The interest in autonomous robots that are capable of working independently, and safely in complex environments has led to an increase in the need for incorporating more and more perception into them [9–11]. One notable area of interest is perceptive grippers. Considerable work has been done with various tactile sensors to add a variety of abilities to grippers such as the detection of surface contours [12, 13], determining grip success [14], classification of objects [15, 16], force control [17, 18], and haptic feedback to the operator [19, 20]. Cameras also play a large role in robot perception. They aid in activities such as navigation and mapping [21], obstacle avoidance [22], and object identification and tracking [23]. Cameras are typically mounted remotely from where any object manipulation is taking place, either on the robot body or fixed in its environment. Between the up close tactile sensor and the distant visual sensor is pre-touch sensing [6]. Pre-touch sensing aims to bridge the gap between cameras and tactile sensors and can therefore provide details that a distant camera can not and without touching the target as a tactile sensor does (Figure 1.1). Obtaining detailed information about a target prior to gripping is very important when there is a risk of disturbing or damaging the target (e.g. fruit on the plant) if gripping is done incorrectly. Pre-touch sensing fills this role and makes use of a variety of kinds of sensors such as optical (e.g. photo-diodes [5] and time of flight [24] sensors), sonic [6], capacitive [7], and electrical [25] and magnetic field [26] sensors - all of which work at a distance. Visual servoing can also be a form pre-touch sensing when cameras are mounted on the end effector and work up-close to the target [27, 28]. Pre-touch sensing can determine if the target is present [5], help perform position correction [29], and provide information about object orientation [30]. Little has

been done to extend pre-touch sensing to compliant grippers as the cited methods implement large sensors or embed them directly into the fingers of the rigid gripper. The large sensors are heavy and will deform soft material, thus altering its performance and attaching hard sensors to soft material can be challenging and introduces rigidity into the soft material thus reducing its effectiveness.

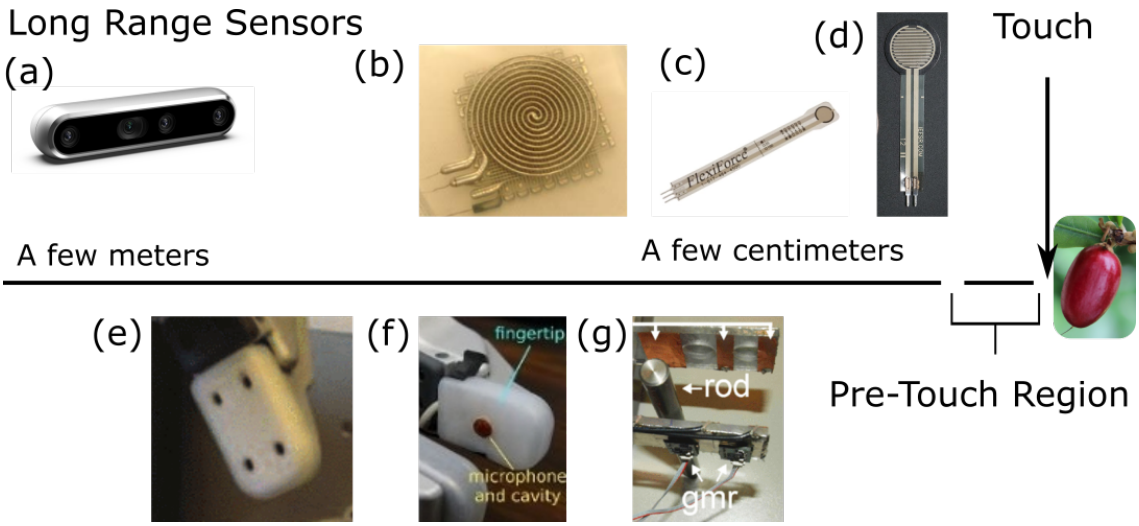


Figure 1.1: Examples of sensors with various ranges: Long Range (a) RealSense Camera [1] ;Touch (b) EGaIn tactile sensor [2] (c) FlexiForce flex sensor [3] (d) Resistive tactile sensor [4] ;Pre-touch (e) Optical sensors [5] (f) Sonic sensors [6] (g) Capacitive sensors [7]

There has been a flowering of gripper designs in recent years which seems largely driven by the development of the field of soft robotics. Traditional rigid grippers made with metal or hard plastic and driven by electric motors, hydraulic or pneumatic actuators is a well developed field with a huge variety of designs and capabilities. The development of soft robotics has introduced concepts like vacuum jamming grippers [31], whole arm grippers [32,33], fluidic elastomer actuators (FEAs) [34], magic ball grippers [35] and Fin Ray grippers [8,36]. These grippers have been designed to be mounted on a standard rigid manipulator (though in the case of the whole arm manipulator, the distinction between manipulator and gripper is not obvious or even possible). There are some examples of grippers being mounted on soft arms, such as one mounted on the end of a growing

vine robot [37] and one for use underwater [38], but the field is small due the extreme restrictions that soft arms place on the gripper design and both examples represent rather unique conditions.

This work seeks to expand the above areas by exploring a gripper system designed to work on a hybrid soft-rigid arm that will operate in an occluded, cluttered environment using perception to aid in successfully picking fruit. The hybrid soft-rigid arm is a manipulator that combines rigid links with an extendable soft arm. It is mounted on a mobile robotic platform and intended to pick fruit in a field environment. This robot presents several challenges related to the gripper design, such as:

- The arm has a very limited payload
- Wires, cables and tubes routed along the soft arm must not interfere with its operation
- The gripper should be able to pick the targeted fruit without damaging it or the environment

The end use also presents a major challenge related to gripping in the occluded, cluttered environment of a plant: ensuring that you have reached the target. Environmental interference such as wind and the plant contacting the manipulator can move the target fruit and change its visibility. Long range cameras can aid in locating fruit on plants by are not sufficient to complete the gripping process. Due to their distance from the target fruit, it is easy for their view of the target to become obscured either by the foliage or the manipulator. Sensors located on the gripper itself are necessary to navigate the last few centimeters to the target and provide qualitative feedback about the quality of the grip. Tactile sensors can aid in this situation, but as they require contact, there is a risk of damaging both the fruit and plants. Pre-touch sensing can fill this gap and safely provide the required feedback. To achieve the goal of the soft gripper system described above,

this work discusses the design of a soft gripper, the implementation of pre-touch sensors and the use of machine learning classifiers to predict the success or failure of a grip prior to touching the fruit. The contributions of this work are:

1. A set of three task specific grippers designed to work in a resource constrained set up

2. A simple, modular, soft actuator design capable of supporting many types of grippers for different tasks

3. A gripper system with pre-touch perception capable of predicting the success of a grip prior to gripping

4. A gripper system with pre-touch perception capable of classifying the object to be gripped

This work is divided into three chapters. Chapter 2 explains the gripper design. It describes the mobile berry picking robot for which the gripper system is designed and the resultant design requirements. For each gripper, details of the the design features and how they meet the needs of the robot are provided as well as an explanation of the actuator. Unused designs are discussed and analyzed. Chapter 3 describes the methods used to create, train and test the gripper system. It explains the hardware used the collected data, the sensors used in the gripper system, and the machine learning methods that were used. It also establishes a testing protocol for how data was collected and evaluated as well as how the machine learning methods were explored. Chapter 4 reviews the results of the experiment conducted. The experiment was divided into three phases: (1) an initial examination of all the sensor configurations with all of the machine learning methods, (2) a deeper examination of the best performing configurations and methods of Phase 1, and (3) Using the configurations and methods of Phase 1 and 2 to classify unwanted objects such a leaves.

2 GRIPPER DESIGN

Figure 2.1: SoftAgbot - A mobile berry picking system that uses a hybrid soft-rigid arm to safely reach inside a plant and pick fruit

This chapter covers the design of the gripper portion of the gripper system. Section 2.1 describes SoftAgbot, the hybrid soft-rigid robot for which the gripper is designed. This section explains how the hybrid robot works and the equipment available on the mobile platform and describes the design requirements for the gripper itself. Section 2.2 uses those design requirements and targeted fruit to detail the designs for three grippers for different fruit: miracle berries, cherry tomatoes and small figs. The final section, 2.3,

Figure 2.2: (a)Gripping an artificial tomato in an occluded environment using a VaLeNS arm with extended soft manipulator (b-c) A BR2 manipulator showing the range of motion achievable.

looks at several alternative designs that were developed and tested during the iterative design process.

2.1 Gripper End Use Case and Design Requirements

2.1.1 SoftAgbot

The gripper was designed to be the end effector on a mobile robotic platform intended to pick a variety of fruits and berries. The robot, known as SoftAgbot was built to make use of a Variable Length Nested Soft arm or VaLeNS arm. The VaLeNS arm is a hybrid soft-rigid manipulator which combines a BR^2 soft manipulator with a traditional rigid arm to take advantage of the properties of both traditional rigid robotics and soft robotics. Because of its soft arm, the SoftAgbot is capable of picking fruit on both the exterior of a plant and inside the leaf canopy without damaging the plant itself. It thus needs the ability to work in occluded, cluttered environments. A complete description of the

SoftAgbot can be seen in [39].

BR^2 Soft Arm

A BR^2 soft arm is made of three fiber reinforced elastomeric enclosures or FREEs [40]. These FREEs are pneumatically actuated and based on the way they are constructed, the application of air pressure causes them to deform. For a bending FREE, as pressure increases, the tubular FREE bends in the plane creating an arc or circle. For a rotating FREE, as pressure increases, the FREE rotates about its long axis. A BR^2 soft arm is constructed of a bending (B) FREE and two rotating (R) FREE - one clockwise and one counterclockwise. This configuration gives the soft arm a wide range of motion with great dexterity. The design, testing and analysis of the BR^2 soft arm can be found in [41] and [42].

VaLeNS Arm

The VaLeNS arm is a three link rigid manipulator where the final link is a hollow tube. Inside this hollow tube sits a BR^2 soft arm. The BR^2 soft arm is mounted on a lead screw so it can be extruded from or retracted into the hollow tube. When fully retracted, the VaLeNS arm acts just like a traditional rigid arm and is thus strong and capable of precise, task space control. When the soft arm is extruded, it able to take advantage of the capabilities of the soft arm - compliance and a large number of degrees of freedom. As much of the soft arm can be extruded as needed to complete the task at hand. The VaLeNS arm is described in [43] and its task space control is detailed in [39]

Mobile Platform

The VaLeNS arm is mounted on a mobile platform. On board the platform are all of the support equipment needed to operate the VaLeNS arm including an air compressor and pressure regulators. It is powered by LiPo batteries and a controlled by a myRIO

controller (National Instruments). An additional Raspberry PI 3 is used to control the soft arm. Details of the mobile platform can be found in [39].

2.1.2 Design Requirements

Based on the end use case of the the gripper, there are several design requirements that needed to be met:

- Gripper is compliant - A compliant gripper allows the manipulator to grip delicate fruit while minimizing the risk of damage. It also supports the over all compliant design of the manipulator intended to work safely around plants.
- Gripper is light weight - The soft arm has a very limited payload (approximately 50g), thus the gripper must be light in order to still be able to grip a larger fruit [43].
- Gripper has a single mode of actuation - Weight factors limit the number and kind of actuators that can be used. A pneumatic mode of actuation was selected due to its simplicity and the readily available source of pressurized air for the SoftAgbot.
- Gripper is capable of picking fruit - It is natural that the gripper must be designed to apply the required forces and torques need to detach the targeted fruit.

Instead of attempting to design a single gripper that could work in all cases, a task based design approach was used where a gripper was designed based only on the needs to accomplish a specific task. To this end, three fruits were selected as target fruit for the robot and grippers were designed to pick them. The fruit were the miracle berry, cherry tomato and a small fig (Figure 2.3). Each one provides interesting challenges that must be addressed.

(a) Miracle Berry

(b) Cherry Tomato

(c) Small Fig

Figure 2.3: Fruit used as design targets for gripper design

2.2 Designs

2.2.1 Miracle Berry

The gripper designed to pick the miracle berry (Figure 2.3a) can be seen in Figure 2.6a. Its design was inspired by the Festo's FinGripper [8, 44] design (Figure 2.4b) and has three fingers which have been 3D printed from thermoplastic polyurethane (TPU) filament (SainSmart Flxible TPU). As the miracle berry is a firm fruit, gripping it directly with the fingers is not problematic as it would be for a soft fruit like a cherry tomato. The miracle berry also sits close to the branch, so it is difficult to get behind it or enclose it. As such, this gripper was designed to directly grip the fruit along its major axis. The fingers, designed to make use of the Fin Ray® effect [36], conform to the shape of the fruit allowing for a firm grip spread out over a large surface area. This prevents damage to the surface of the fruit. The firm grip then allows the manipulator to apply a moment to the berry and separate it from the plant. The size of the fingers and the thickness of the structure were developed using an iterative design process. The actuator is described in detail in Section 2.2.4.

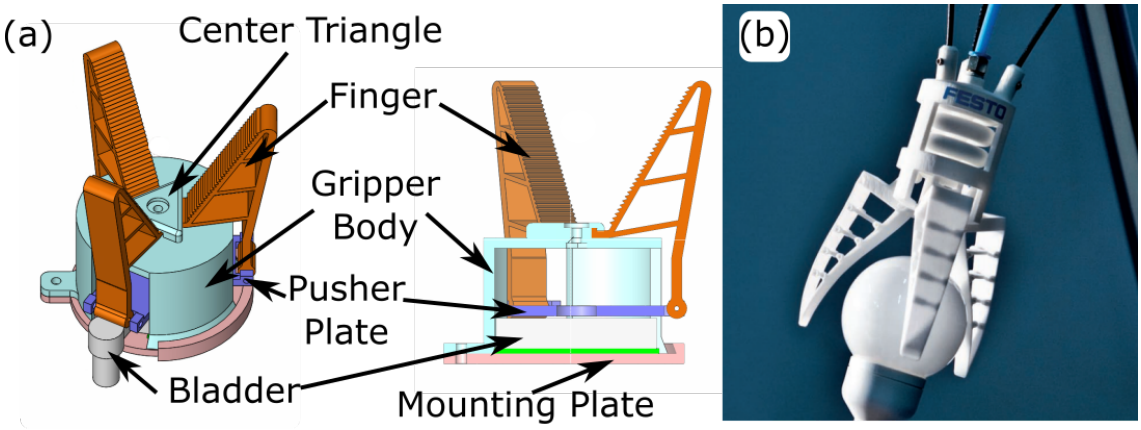


Figure 2.4: (a) Parts of the gripper (b) A Festo FinGripper that was used as the basis of the gripper design [8]

2.2.2 Cherry Tomato

The cherry tomato gripper (Figure 2.6b) was built from the starting point of the miracle berry gripper and was 3D printed from the same TPU. As the fruit is larger and softer, it required some changes to the approach and design. To prevent damage to the fruit, the design was modified to enclose the fruit rather than directly grip it. This is possible because the fruit sits on a short stem away from the branch of the plant. Tabs were added the end of the gripper to create a space within the fingers when it is closed and the fingers were spread wider to increase the volume of the enclosure. The tabs were also designed to sit behind the fruit and allow the robot to apply a moment to separate the fruit from plant. After being picked the fruit stays trapped with in the finger enclosure until it can be safely deposited in a collection bin. Due to the size of the fruit and its picking method, the fingers were made more robust. The same Fin Ray® effect design was employed to allow the fingers to conform to the fruit if needed (e.g. a large fruit is gripped), though the tabs should prevent an over exertion of force on the fruit. The size of the fingers, length of the tabs and the thickness of the structure were developed using an iterative design process. The actuator is described in detail in Section 2.2.4.

2.2.3 Small Fig

The fig gripper (Figure 2.6c) incorporates some of the same ideas as the miracle berry and cherry tomato gripper. The fig is a small, very soft fruit, that sits on a long stem. Because it is so soft, once again the approach to enclose the fruit is employed, but a much smaller enclosure was created. When closed, the tabs sit behind the fig and the fruit can be pulled from the plant without damaging it as the force is only applied to the back of the fruit. The fruit then stays enclosed in the gripper until it is deposited in a storage bin. The gripper is still made of TPU as the compliance is needed to create the closing motion. The size of the fingers, length of the tabs and the thickness of the structure were developed using an iterative design process. The actuator is described in detail in Section 2.2.4.

2.2.4 Actuator

The actuator design is common to all three gripper designs with minor modifications to parts to facilitate the different gripper designs. A labeled diagram of the gripper and parts can be seen in Figure 2.4a. The grippers are pneumatically actuated via an approximately 34.5kPa (5psi) air signal provided by an air pressure regulator. The low pressure air is used to inflate a small, round, disc-shaped air bladder made from silicone (Dragon Skin^{TM} 30, Smooth-On). This bladder acts on a pusher plate connected to the free end of each of the gripper fingers while the other end of the fingers are fixed in place by a center triangle piece which clamps them in place (Figure 2.5a,b). When the bladder is inflated, it raises the pusher plate which in turn raises the free end of the fingers. Since the other end is fixed the fingers move inward, thus closing the gripper (Figure 2.5c). When pressure is released, the compliant fingers return to their original shape and the gripper opens. Minor modifications include a larger pusher plate and center triangle for the cherry tomato gripper to accommodate its larger opening and thicker finger structure. The body of the actuator is designed to facilitate the modular nature of the gripper by

making it interchangeable. A raised lip on the outer edge of the body interlocks with a catch on the mounting plate. It is then held in place with a locking bolt (Figure 2.5d,e).

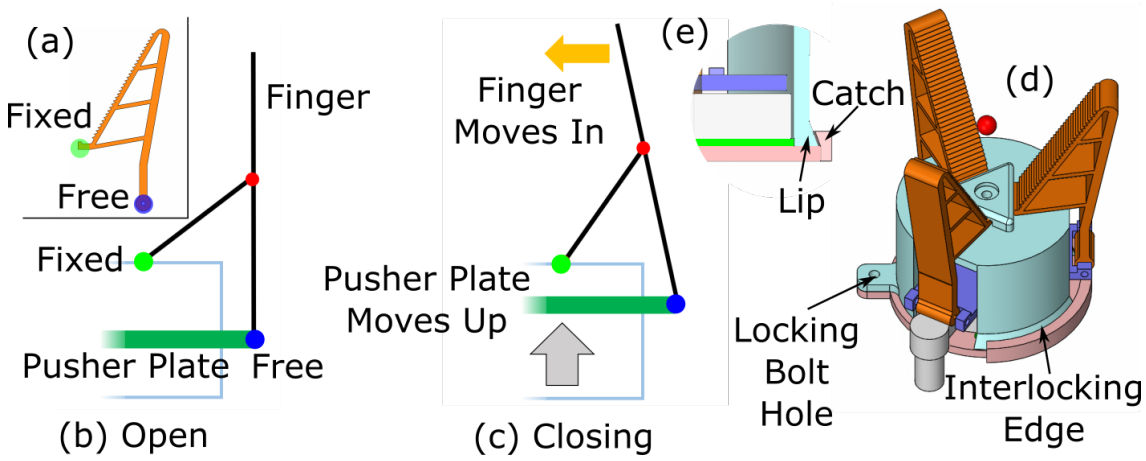
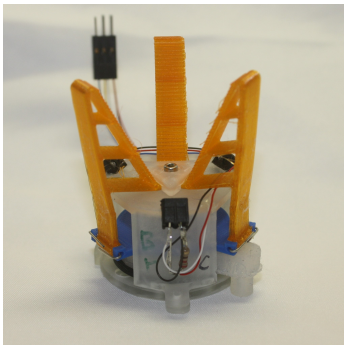


Figure 2.5: Details of the actuator (a) Image of finger showing free and fixed ends (b) open gripper (c) closing gripper - the expanding bladder pushes the pusher plate up which forces the fingers inward (d) the gripper attached to the mounting plate (e) cut away of interlocking lip and catch which locks the gripper to the mount plate

2.3 Other Designs Explored

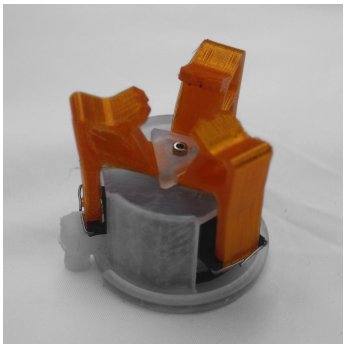
In addition to the final designs, many intermediate studies were completed to find the correct size and shape of the parts and the best mode of actuation. Also many other types of grippers were explored. Figure 2.6d and 2.6e were created to understand the motion of the fingers and the Fin Ray effect. Similar designs were made to explore the length of the fingers and thickness of the material required. These early designs were actuated by a cable which was another method of actuation that was tested. Cable actuation was decided against due to the potential for interference with the soft arm shape and position. Different materials for the gripper were tested such as Figure 2.6f, where the gripper fingers are printed from a flexible resin made by FormLabs (FLFLGR02). This material was ultimately to heavy and not durable enough to work. Figure 2.6g shows an early actuator design. In this design, the whole actuator body moves as the doughnut shaped

bladder expands. The design was abandoned because it was fragile, difficult to assemble and impossible to repair as it was glued together. It also required more than 137.9kPa (20psi) to actuate which often ruptured the bladder. Other than the finger based designs, two additional concepts were explored. Figure 2.6h, shows a vacuum driven magic ball gripper which is a miniaturization of the grippers developed by Li et al [35] and Figure 2.6i is an inflation based gripper inspired by Wang et al [45], which traps the fruit in a cup shaped enclosure and inflates bladders at the open end to trap the fruit inside. Neither design proved workable due to their complex design and their bulk, which made them ill-suited for cluttered environments.



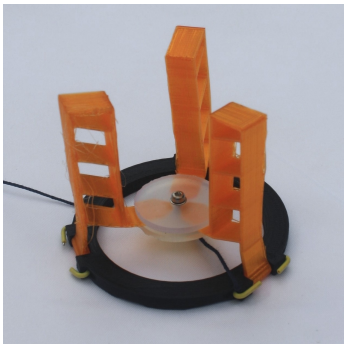
(a) Miracle berry gripper

(b) Cherry tomato gripper



(c) Fig gripper

(d) Early prototype

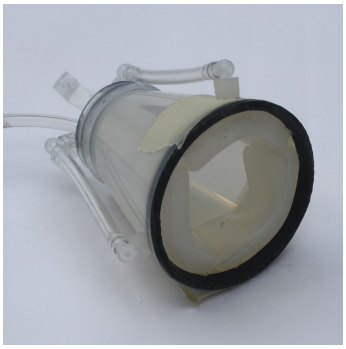


(e) Another early prototype

(f) Resin printed fingers

(g) Early actuator design

(h) Magic ball gripper



(i) Inflatable gripper

Figure 2.6: Evolution and variation of the gripper design

3 PREDICTION OF GRIP SUCCESS

This chapter explains how grip success prediction was achieved. Section 3.1, covers all the hardware and software used to train the system to detect grip success and verify the results of that training. Section 3.2, covers the methods used for training and testing, including the machine learning methods and the test protocol used.

3.1 Materials

This section explains all of the hardware and software used to train the gripper system to predict grip success. An overview of the hardware used can be seen in Figure 3.1. The major components include a laptop, a robotic arm, a variety of sensors (IR reflectance, time of flight and color), two microcontrollers (Arduino Mega and Arbotix-M), a user input subsystem and a pneumatic subsystem. Further details of each component and how they were integrated together follows.

3.1.1 Robotic Arm

In order to collect training data, it was necessary to precisely position a piece of fruit in and around the gripper system. This was a accomplished using a modified WidowX Robot Arm Mark II (Trossen Robotics). The modified arm consists of the lower three joints of the WidowX arm with a custom configuration of three additional joints made from servomotors (DS3128MG, DS Servo) which act as a spherical wrist and give it six degrees of freedom. Additional modification allowed for the mounting of the gripper as the end effector. This arm can be seen in Figure 3.2. The arm is controlled via the Robot

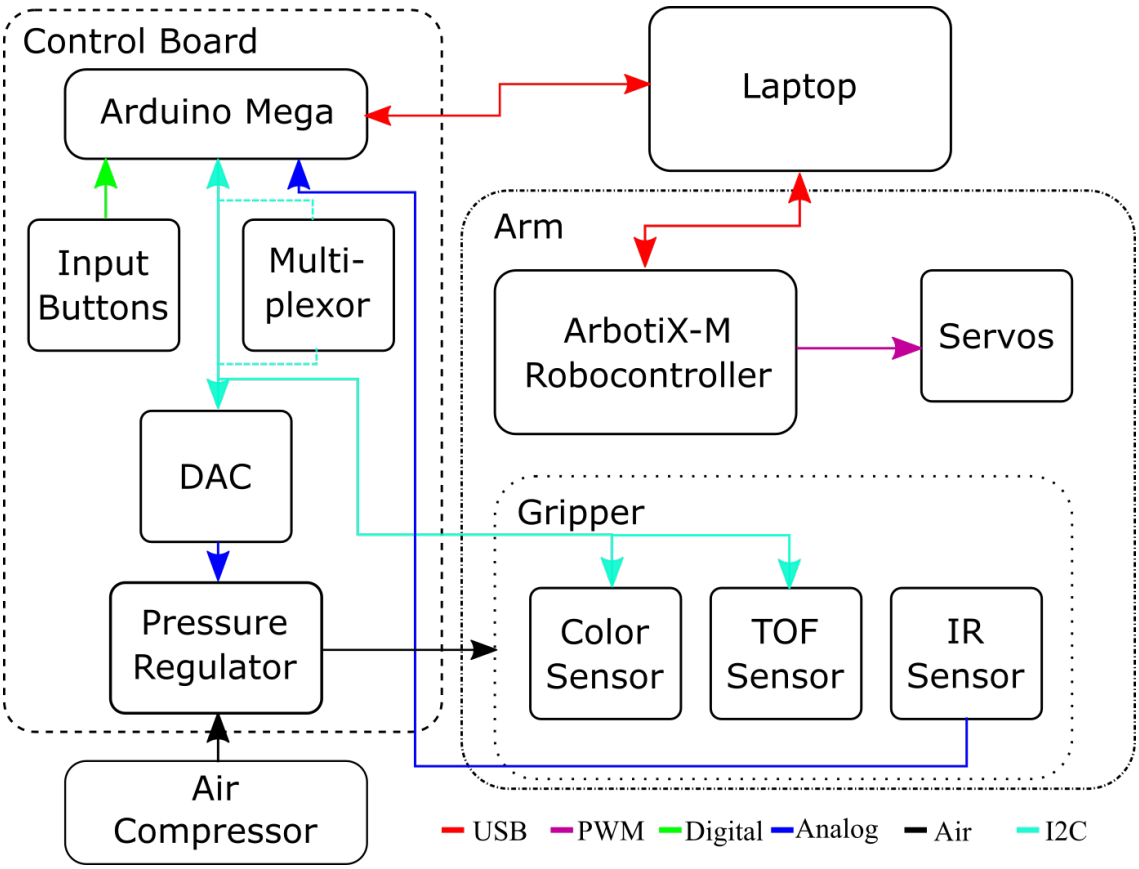


Figure 3.1: System Block Diagram - All components and subsystems with their mode of communication. Note the multiplexor is only used for select combinations of sensors

Operating System (ROS) on an Arbotix-M control board (Vanadium Labs) which is an, Arduino based microcontroller designed to control multi-joint robots. Custom control scripts were created to perform task space control of the end effector position and to make simple point to point linear moves.

3.1.2 Sensors

Three sensors types (IR reflectance, time of flight and color) were used as input to the prediction methods. The IR reflectance and Time of Flight (TOF) sensors both give data related to the position of the fruit relative to the gripper, while the color sensor helps separate between fruit an other objects such as leaves and branches. It is worth noting that a gripper mounted camera could also be useful to achieve the desired goal. It was

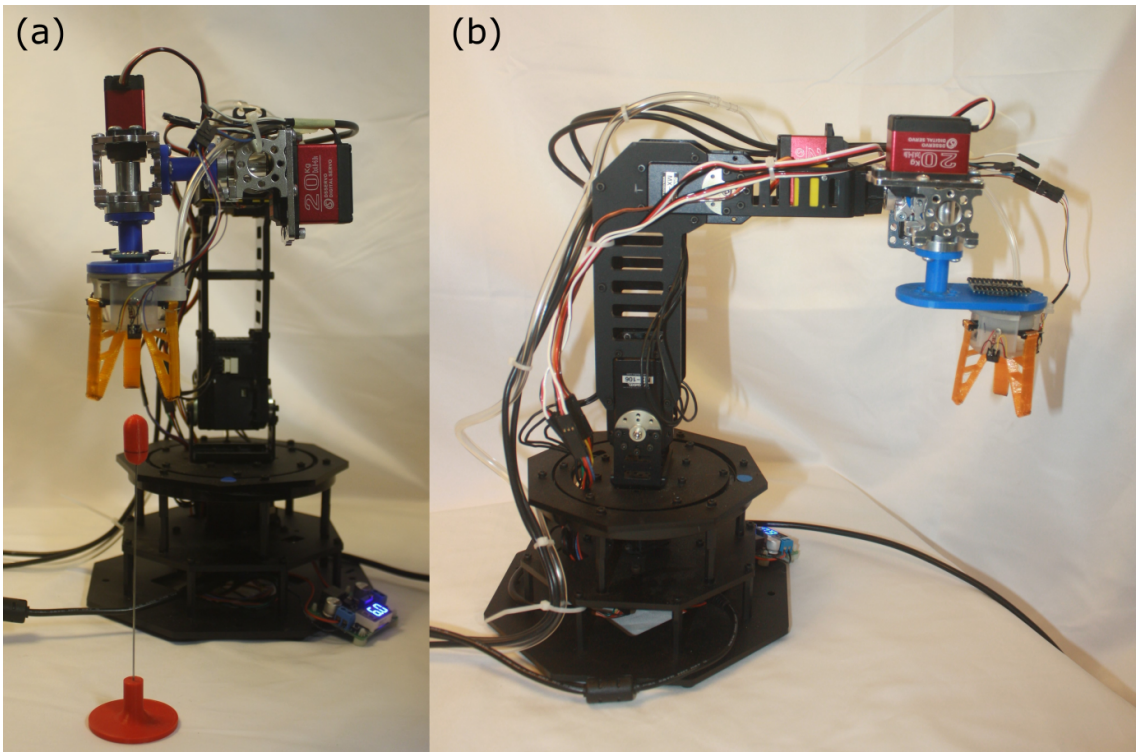
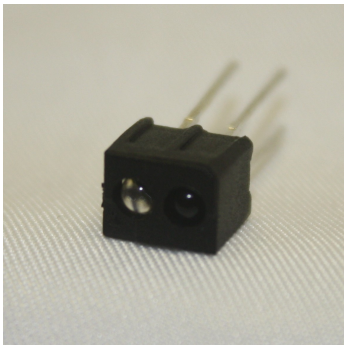


Figure 3.2: A modified WidowX arm used to position the gripper and collect training data (a) Front view (b) Side view

decided not to explore cameras in order to keep costs low and maintain the simplicity of the approach. While cameras have become relatively inexpensive, they require much more computational power than the simple sensors presented here which would add to the cost of the solution. Additionally, they require much more complex approaches to the development of the machine learning method and data collection and labeling.

IR Reflectance Sensor

The IR reflectance sensor (Everlite ITR-20001/T) consists of an GaAlAs LED emitter mounted side by side with a phototransistor detector. The LED emits at 940nm (the IR range) and the phototransistor is paired to be sensitive to this wavelength [46]. Depending on the surface and its properties, light emitted by the diode is reflected off the surface and back into the phototransistor where it is detected as a voltage proportional to the amount of light reflected. This voltage is read by the ADC of the Arduino Mega microcontroller.

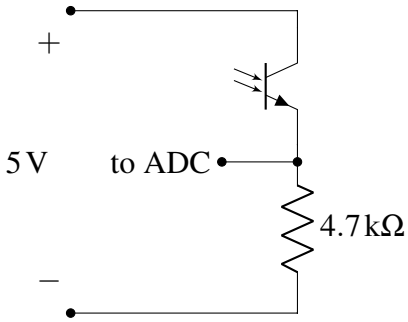


(a) IR Sensor

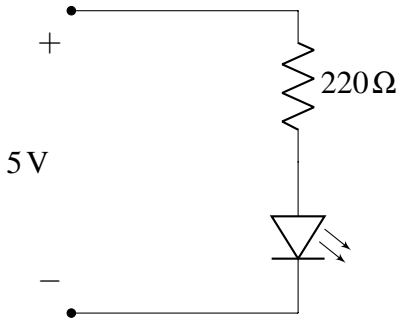
(b) TOF Sensor

(c) Color Sensor

Figure 3.3: Sensors used in the gripper system to detect features and predict grip success



(a) Phototransistor circuit



(b) LED circuit

Figure 3.4: Sensor circuits for the IR reflectance sensor

In the case of the gripper system, the amount of reflectance depends on how close the fruit is located to a sensor. The LED and phototransistor are powered by 5VDC as seen in Figure 3.4. This sensor requires two wires for power and one for each of the sensor outputs.

Time of Flight Distance Sensor

The Time of Flight (TOF) distance sensor (Figure 3.3b) (VL6180X Breakout, Adafruit) uses an IR laser to accurately measure the distance to objects. Unlike the reflectance sensor, it does not depend on how reflective the surface of the object is, but instead measures the time taken for the emitted light to bounce back. This allows it to very accurately determine the distance to an object between 0 and 100 mm away. It uses I^2C to communicate the distance to the Arduino Mega microcontroller [47]. In the case of the gripper

system, the distance depends on where the fruit is located in the gripper. The sensor was powered by 5VDC and requires two more wires for the I^2C communication for a total of four wires. Additional sensors share the I^2C bus, so more wires would not be needed.

Color Sensor

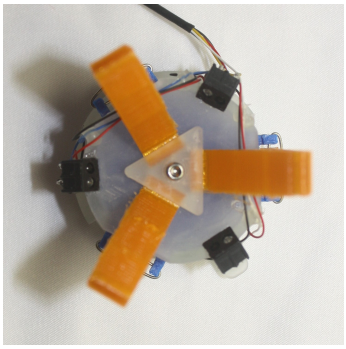
The color sensor (Figure 3.3c) (RGB Color Sensor with IR filter and White LED - TCS34725, Adafruit) is made of four kinds of filtered photodiodes: red-filtered, greed-filtered, blue-filtered and unfiltered (clear). ADCs convert the output of these photodiodes to digital signals that are transmitted via I^2C to the Arduino Mega microcontroller. It is IR filtered to better represent the human visual spectrum range [48]. A 4150 K LED is used to illuminate the sample area with a bright, neutral light. The sensor was powered by 5VDC and requires two more wires for the I^2C communication for a total of four wires.

Sensor Arrangements

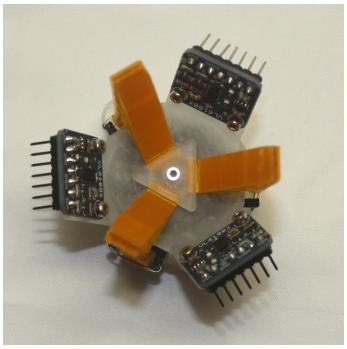
To fully explore the three types of sensors selected, they were combined and arrange in different ways:

- Three IR sensors
- Three TOF sensors
- Two IR sensors and a color sensor
- Two TOF sensors and a color sensor
- One each: IR, TOF and a color sensor

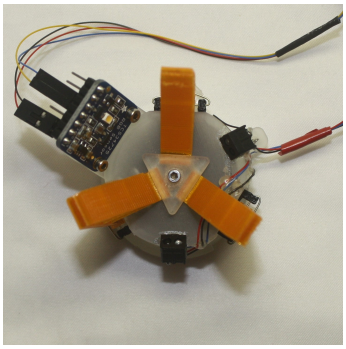
The sensors are all arranged 120° apart - opposite the gripper fingers as seen in Figure 3.5. They are aimed at the center of the fingers - a point approximately 20mm from the top of the actuator body as seen in Figure 3.5f.



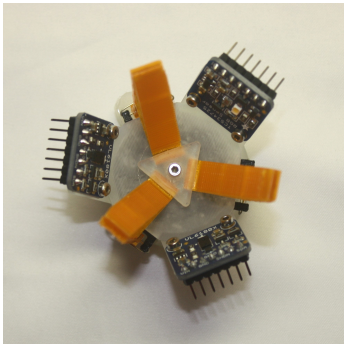
(a) IR Sensor



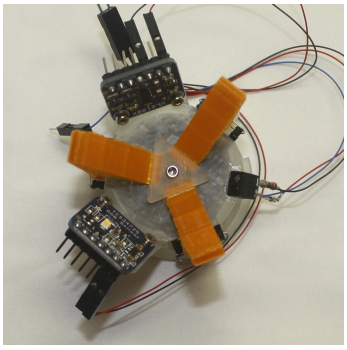
(b) TOF Sensor



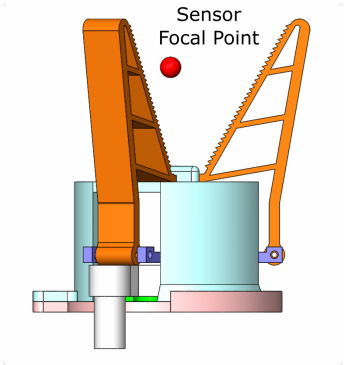
(c) IR-Color Sensor



(d) TOF-Color Sensor



(e) IR-TOF-Color Sensor



(f) Focal point of the sensors

Figure 3.5: (a-e) Sensors arrangements (f) Sensor focal point indicated by red dot

3.1.3 Additional Subsystems

Pneumatic Control Subsystem

To open and close the gripper, a 34.5kPa (5psi) air signal was required to be turned on (close) and off (open). This was done via an air pressure regulator (QB1XANKKZP50PSG, Proportion Air). This regulator was controlled with a voltage signal proportional to the desired output in pounds per square inch (psi) - i.e. a 0VDC signal gives 0psi (0kPa) and 5VDC signal gives 50psi (344.7kPa), thus a 0.5VDC signal gives the desired 5psi (34.5kPa) closing pressure. To create this signal, a DAC (SparkFun I2C DAC Break-out - MCP4725) was used and controlled via I^2C from the Arduino Mega. The supply of high pressure air (approximately 137.9kPa (20psi)) was supplied from a portable air compressor (Porter Cable, Model PCFP02003).

User Input Subsystem

To facilitate easy user input, an array of four normally open tactile buttons (E-Switch TL1100F160Q, Digi-Key) were connected to digital inputs of the Arduino Mega with button de-bounce being done in software. Pull down resistors (4.7 kΩ) were added to prevent false inputs. These buttons were used to collect data about the evaluation of gripper success. In this work, only two buttons were used, but additional buttons were installed for future development.

Laptop and ROS

A laptop (Dell Latitude E6540, Intel Core i5-4300MCPU@2.60GHz, 8GB RAM) running Ubuntu 16.04LTS with ROS Kinetic was used for control and data collection. Two custom ROS nodes were created: an arm control node and a training node, both written in Python 2.7. The arm control node contained the code needed to control the arm servos, the analytic inverse kinematics equations and a simple linear point-to-point path planner.

It interfaced with the Arbotix-M microcontroller via USB. The training node contained all the code needed to execute the test protocol described in Section 3.2.1, including a user interface, gripper operation, receiving input from the user input buttons and data collection. It communicated via USB with the Arduino Mega.

3.1.4 System Integration

The arm, gripper, sensors, and additional subsystems were all integrated together as seen in the system block diagram (Figure 3.1). User control was perform via the laptop and ROS which brought of the systems together to achieve the tasks needed to collect training data. One additional note is that due to I^2C address collision between the TOF sensors and color sensor, anytime more than one of these was used, an I^2C multiplexor (TCA9548A I2C Multiplexer, Adafruit) was used to control all the devices.

3.2 Methods

This section explains the methodology used to collect data, train and evaluate the effectiveness of the training. It includes the test protocol used and an explanation of each of the machine learning methods and how they were evaluated.

3.2.1 Test Protocol

To gather data, a consistent test protocol was developed. A dummy target based on the size, shape and color of a miracle berry (Figure 2.3a) was 3D printed from PLA (Figure 3.6b). This dummy target was mounted on a flexible metal shaft to simulate the flexibility of the branch or stem to which the fruit is attached. The target was mounted securely to the lab table to maintain a fixed origin during data collection. The gripper was mounted to the modified WidowX arm which was used to position the gripper around the dummy

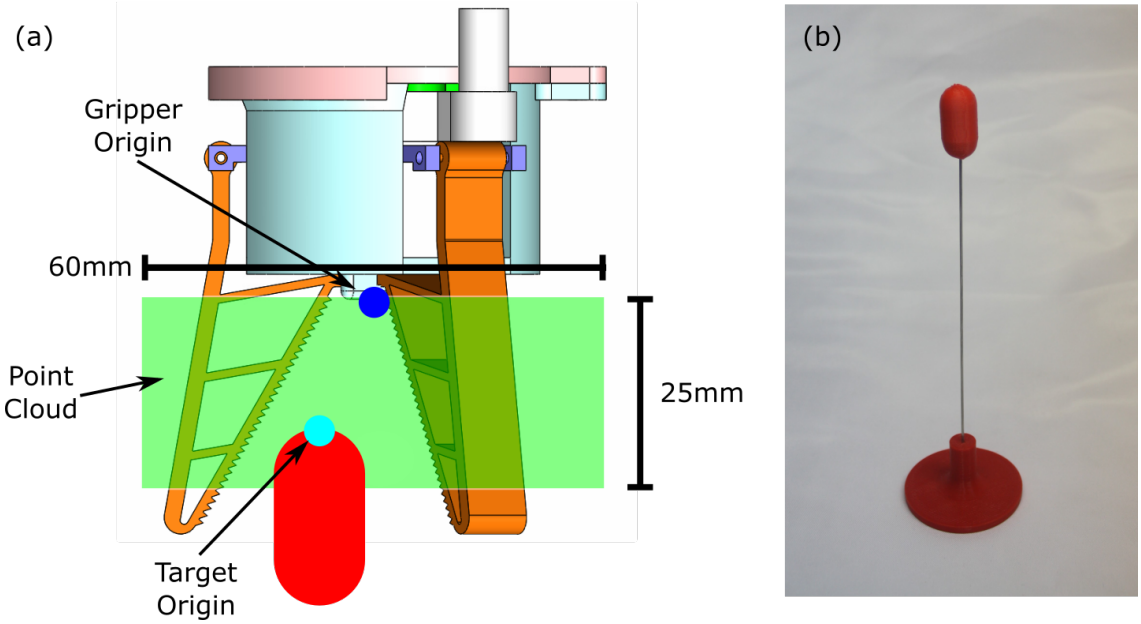


Figure 3.6: (a) The arrangement of the master point cloud from which training test points were selected (b) the dummy target based on a miracle berry and mounted on a flexible shaft

target. The final link was controlled so as to always be parallel to the work surface and the x-axis of the work space. This maintained the orientation of the gripper so that the major axis of the target was always aligned with the center line of the gripper, as seen in Figure 3.7a. Using this equipment, the target was positioned within or around the gripper while gathering data. To fully explore this space, a point cloud of 73,034 points was created as a grid of points spaced 1mm apart in the x, y and z directions in a 25mm cylinder with radius 30mm as seen in Figure 3.6a. These points represent the position of the top center of the dummy target relative to the bottom center of the gripper opening. From this master set of points, smaller sets were randomly selected. Instead of collecting data on all 500 (or 1000) points at once, data collection was conducted on smaller sets of 100 points until the total of 500 (or 1000) points were tested in order for the operator to maintain focus during data collection. The process was repeated until all of the desired points were collected and the data was combined together into the desired training data set. The full

details of the data collection protocol can be seen Algorithm 1.

Algorithm 1: Data collection Protocol

Result: A collection of 100 training elements

Create a training point set consisting of 100 points randomly selected from the master point cloud;

for *each point in the training set* **do**

    Move the gripper to a point in the training set;

    Collect sensor data;

    Close the gripper;

    Evaluate in accordance with Section 3.2.1: Evaluating the grip and record the status of the grip;

end

Save the collected data;

Evaluating the grip

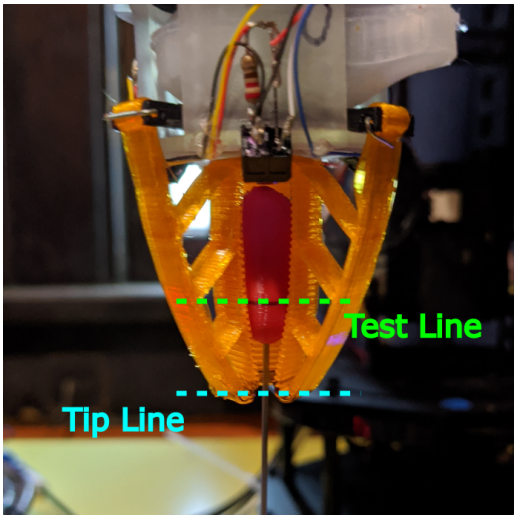
One important step in this process is the evaluation of the grip. Because training is being completed on a dummy target, it is not possible to pick the fruit in the same way that one would on a real plant. It is also not reasonable at this point in the development of the gripper system to attempt to train the system on real fruit as it would require access to thousands of live fruit and add layers of complexity that would make proper evaluation of the prediction process difficult. The goal at this stage is to evaluate the effectiveness of the sensors and machine learning techniques at predicting the success of a picking action, before implementing the results in a fully automated robotic picker system. To that end, a reasonable assumption about what makes a grip successful was made. Based on observation of the gripper in use, the position of the fruit within the closed gripper is good predictor of how successful the action will be. A good, secure grip will likely result in a successful picking, while a grip that misses the fruit or has little contact between fruit

and fingers will likely fail. It was further observed that the fruit's position in the closed gripper, can be predicted by its position relative to the gripper before closing. Thus, if the relative position of the fruit to the open gripper can be measured, it will be possible to use that data to predict the success or failure of the picking action. Since the final picking was not done, it was only necessary to match the collected sensor data with a consistently applied label based on the quality of the grip (and hence the likelihood of success). To aide in consistently evaluating the quality of a grip, a set of guidelines was created.

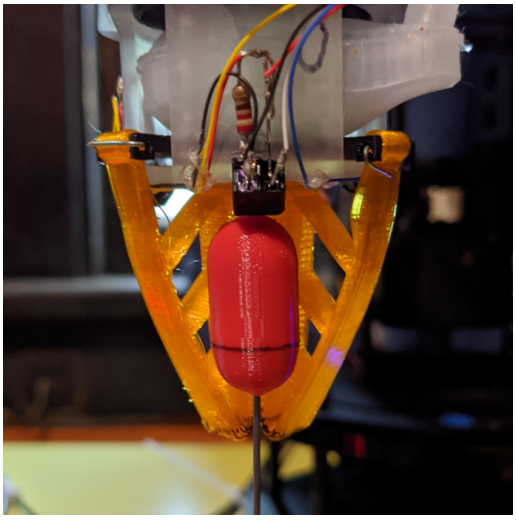
- The fruit should have contact with the face of all three fingers
- The fruit should be deep enough in the gripper so that the tip line is below the test line on the target (Figure 3.7a)

While this does not perfectly represent success and failure, it does represent a consistent method of evaluation that directly relates to the fruit's position in the gripper. If a machine learning method can successfully predict the outcome of this experiment, it can predict fruit position in the gripper and is a strong indicator that it can predict success or failure of picking.

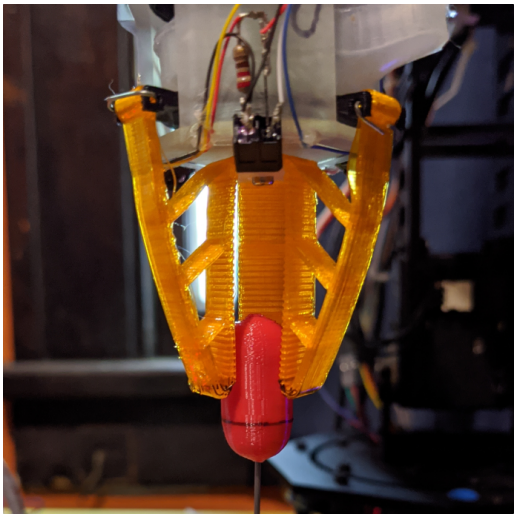
Figure 3.7 gives examples of how the grip was evaluated. In Figure 3.7a, the target is deep in the gripper with a lot of contact with the fingers. It is thus likely to succeed and labeled 'good'. Figure 3.7b shows the gripper completely missing the target which is obviously going to fail and is labeled 'bad'. In Figure 3.7c, the target is only barely in the gripper. Though it has a lot of contact with the fingers, it is unlikely to succeed. The cut-off for this situation was determined by the test line on the target. If the finger tips were above the line, it was labeled 'bad' and below the line it was labeled 'good'. Figure 3.7d shows a poorly positioned grip. The target is 'deep', but the contact with the fingers is poor and likely to be lost when any force is applied. This grip is likely to fail and is labeled 'bad'.



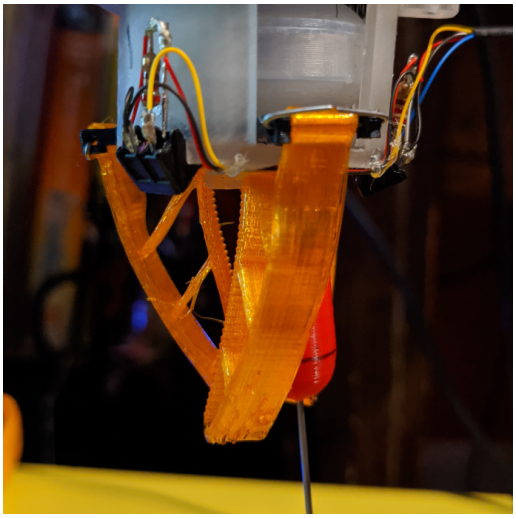
(a) A well positioned grip



(b) A complete miss of the target



(c) A poorly positioned grip that grips the fruit too highly



(d) A poorly positioned grip where the fruit is caught between only two fingers

Figure 3.7: Examples of various grip conditions used in the evaluation of the grip

Collecting Leaf Data

Leaf data was collected using an artificial leaf placed in the open gripper. It was manually manipulated to simulate a large number of ways that a leaf could become lodged in the gripper and fool the system into predicting a berry was present and well position when it is in fact another object. This testing was also not meant to be exhaustive of all foreign objects that might enter the gripper, but to show that the system can be trained to differentiate between kinds of objects to aid in the prediction of grip success.

Three Phases of Data Collection

Data collection and testing were performed in three phases:

- Phase 1: 700 elements were collected using the test protocol. The 700 elements were then divided into three sets. The *training* set consisted of 500 elements. This was used to create the classifier for each method. The *testing* set consisted of 100 elements. This was used in conjunction with the classifier to set the various hyperparameters and options for each method. The last 100 elements form the the *final testing set*. This set was used only to make a final evaluation of each method and played no role in tuning hyperparameters or setting options. Phase 1 testing was conducted for all five sensor combinations with all five machine learning methods and a limited set of hyperparameters.

- Phase 2: 1200 elements were collected using the test protocol (the original 700 plus 500 more). This resulted in a training set of 1000 elements, the same test set of 100 elements and the same final test set of 100 elements. Phase 2 was only conducted on the best performing methods and sensor combinations of Phase 1 and included a deeper exploration of these methods and sensor arrangements.

- Phase 3: For this phase, data was collected using an artificial leaf. Three hundred elements were collected of which 200 were used for the training set, and 50 each

were used for the testing set and final test set. Phase 3 was an extension of Phase 2 and used the results of both to guide the sensors and methods selected. These elements were combined with the elements of Phase 2 for training and testing.

3.2.2 Nearest Neighbor

Overview

Nearest Neighbor or k^{th} Nearest Neighbor (KNN) is form of supervised learning. Training data, consisting of states, is collected along with a label associated with each set of states. These states are treated as points in $\mathbb{R}^n$ (where n is the number of sensor states - three or five in this work) and an appropriate distance function, such as the Euclidean norm, is selected. When a new observation, y is created, the label for it is predicted by searching through the training data and finding the label of the state vector that has the smallest distance between itself and y. This is the "nearest neighbor" to y and its label is predicted for y. To account for outliers in the data, KNN finds not just the one nearest neighbor, but the k nearest neighbors, where k is a positive integer greater than 1. The results then count as a vote for a particular label and the label with the most votes is applied to y. This method has the advantage of being very easy to implement. Training data is collected and labeled, but no further processing of the data is required. The algorithm is very straightforward and there is only one hyperparameter, k, to tune. The major disadvantage of this method is the large computation cost at the time of prediction. A given observation must be compared with every point in the training data set, thus the larger the training data set, the more computationally intense the method is [49–51].

Setting the k value

To implement KNN, it is necessary to set the single hyperparameter k. Data was collected as described in the testing protocol (Section 3.2.1). The training data set was then used

to predict the outcome of the testing data set with Scikit-Learn's *KNeighborsClassifier* module [52]. This was repeated for k values from 1 to 25. An example of the results can be seen in Figure 4.1a. Based on the results, an appropriate value of k was selected to maximize the precision of the prediction. For each sensor combination, the best value of k was selected and used to predict the outcome of the final testing data set. These results were used to create confusion matrices (For example, Figure 4.2a) which were used to evaluate which method and sensor configuration provided the best results.

3.2.3 Decision Trees

Overview

Decision trees are a form of non-parametric, supervised learning. Training data, consisting of states, is collected along with a label associated with each set of states. These states and labels are then used to create a decision tree. A decision tree is a tree data structure which contains decision nodes and prediction leaves. To create the tree, the training data is split according to some feature of the states. This feature split then forms a decision node. The process of finding features by which to split the training data continues until the data can no longer be split (all labels are the same) or the max depth is reached. At this point a leaf is created which contains a label. This is repeated until all branches in the tree end with a leaf. The challenge of this method is how to select the feature and value for each decision node to efficiently and accurately split the data. This is done by an impurity measure which measures how well a decision splits the data. A pure split is one in which all the child nodes formed are leaf nodes i.e. they share the same label. Two common ways to measure impurity are via entropy and Gini index. To make a prediction, an observation y is tested at each node. It then follows whatever branch the feature comparison dictates until it reaches a leaf where a label is applied. The advantages of a decision tree are that it is relatively easy to train and once trained, it does not require

much computational power to use. It is also a white box method in that you can observe exactly what is happening in the tree. Trees are easily visualized as binary trees. The disadvantages are that some data does not work well with the method which can result in very deep, complex trees [50, 51, 53].

Setting the parameters and options for Decision Trees

Two parameters were explored for decision trees: maximum depth of the tree and the impurity measure. Data was collected as described in the testing protocol (Section 3.2.1). The training data set was used to create the decision tree and predict the outcome of the testing data set with Scikit-Learn's *tree* module [52]. This was repeated for maximum tree depths ranging from 1 to 25 for both the entropy and Gini index impurity measures. An example of the results can be seen in Figure 4.3a. Based on the results, an appropriate max depth and impurity measure were selected to maximize the precision of the prediction. For each sensor combination, the best parameters were selected and used to predict the outcome of the final testing data set. These results were used to create confusion matrices (For example, Figure 4.4a) which were used to evaluate which method and sensor configuration provided the best results.

3.2.4 Support Vector Machines

A support vector machine is a kind of linear classifier. It operates on the idea of splitting the space the data resides in into two regions with a hyperplane. For separable data, this plane is picked to maximize the geometric margin which is the minimum Euclidean distance between the hyperplane and the training data points. The selection of this hyperplane is an optimization problem. Specifically, it is a quadratic programming problem, which means there are many available solution methods - including techniques specifically designed for this type of problem. For non-separable data the problem is more complex. No hyperplane can be selected which can maximize the geometric margin for

all data points. Instead, the goal is to minimize the distance between the hyperplane and the points which violate it. This is done by relaxing the constraints of the optimization problem. For every point that violates the constraints, a slack variable is created which represents the distance by which the point violates the constraint. This then becomes an optimization problem with two competing desires: maximize the geometric margin and minimize the total slack [50, 51, 54].

Setting the parameters and options for Support Vector Machines

For SVMs the kind of kernel function was explored. The kernel function controls how the data can be split up e.g. a 'linear' kernel can only divide with lines and planes whereas a polynomial kernel can include curves. Four kinds of kernels were tested: Radial Basis Function ('RBF'), polynomial ('poly'), 'linear' and 'sigmoid'. Data was collected as described in the testing protocol (Section 3.2.1). The training data set was used to create the classifier and predict the outcome of the testing data set with Scikit-Learn's *SVC* module [52]. This was repeated for all four kernel functions. An example of the results can be seen in Table 4.9. Based on the results, an appropriate kernel function was selected to maximize the precision of the prediction. For each sensor combination, the best parameters were selected and used to predict the outcome of the final testing data set. These results were used to create confusion matrices (For example, Figure 4.6a) which were used to evaluate which method and sensor configuration provided the best results.

3.2.5 Multi-layer Perceptron

A multi-layer perceptron (MLP) is a type of artificial neural network. At its most basic form, it consists of a single perceptron which is designed to behave like a neuron in a brain. Mathematically, the perceptron creates a hyperplane and is only capable of creating a linear fit. It takes the form:

$$y = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

where y represents the predicted label, x is the feature vector, w is a vector of weights associated with each feature and w_0 is an offset or bias. To train a perceptron, training data is used to adjust the weights w and bias w_0. This is done by applying the perceptron to an element of training data. The error between the predicted label and the correct label is then used to update the weights and bias. If the prediction is good, there will be little to no correction, while if it is poor, there will be larger corrections. This is repeated for all the training data and if there is enough data, the weights and bias will converge and on values that work well as a predictor.

The multi-layer perceptron builds off of this idea and stacks perceptions together in order to fit to non-linear problems. It consists of an input layer, an output layer and a number of hidden layers in between. The size and number of these hidden layers is a hyperparameter that must be tuned by the user. In order to fit non-linear data, an activation function, such as 'relu', is used to make the output of the hidden layers non-linear. Without this, it would just be a linear function acting on another linear function which will yield a linear output. The hidden layers also make the weight correction more complex, which is solved through the backpropagation algorithm. Once a prediction is made, the error is then propagated back through the network to update all of the weights of all of the layers. Because it is no longer linear, a solver, such as stochastic gradient descent, is needed to optimize the updated weights [50, 51, 55].

Setting the parameters and options for the Multi-layer Perceptron

For MLPs the primary hyperparameter explored in Phase 1 was the size and number of the hidden layers. Setting the number and size of the hidden layers can be challenging. Typically there is little need for more than one hidden layer and the layer should not be bigger than the input layer (three or five in this case) or smaller than the output layer (two for Phase 1 and 2 and three for Phase 3) [56]. It was decided to explore two hidden layers and vary the size between the input and output sizes. Data was collected as described

in the testing protocol (Section 3.2.1). The training data set was used to create the classifier and predict the outcome of the testing data set with Scikit-Learn's *MLPClassifier* module [52]. This was repeated for all the variations of the hidden layers. An example of the results can be seen in Table 4.3. Based on the results, an appropriate hidden layer combination was selected to maximize the precision of the prediction. For each sensor combination, the best parameters were selected and used to predict the outcome of the final testing data set. These results were used to create confusion matrices (For example, Figure 4.5a) which were used to evaluate which method and sensor configuration provided the best results.

For Phase 2, the activation function and solver were also explored. The activation functions used were: 'identity', 'logistic', 'tanh' and 'relu'. The solvers used were: limited-memory Broyden–Fletcher–Goldfarb–Shanno ('lbfgs'), stochastic gradient descent ('sgd') and 'adam'. Using the larger training data set and the results of Phase 1, the results for the different combinations of these two parameters were compared.

3.2.6 Naïve Bayes

Naïve Bayes works from Bayes' Rule:

$$\Pr(a \mid b) = \frac{\Pr(a)\Pr(b \mid a)}{\Pr(b)}$$

But instead wants to find $\Pr(y \mid x_1, x_2, x_3, \ldots, x_n)$, where y is label and $x_1, \ldots, x_n$ are features of a state. This gives the equation:

$$\Pr(y \mid x_1, \ldots, x_n) = \frac{\Pr(y)\Pr(x_1, \ldots, x_n \mid y)}{\Pr(x_1, \ldots, x_n)} \tag{3.1}$$

The naïve part of Naïve Bayes is the assumption that all of pairs of features are con-

ditionally independent. Because of this assumption, Eqn 3.1 can be simplified to:

$$\Pr(y \mid x_1, \ldots, x_n) = \frac{\Pr(y) \prod_{i=1}^{n} \Pr(x_i \mid y)}{\Pr(x_1, \ldots, x_n)} \tag{3.2}$$

This can be further simplified to:

$$\Pr(y \mid x_1, \ldots, x_n) \propto \Pr(y) \prod_{i=1}^{n} \Pr(x_i \mid y) \tag{3.3}$$

as the denominator is a constant. Using Eqn 3.3, it is only necessary to find the value of y i.e. the label that maximizes the results. The value of $\Pr(y)$ comes from the frequency of the label in the training data and $\Pr(x_i \mid y)$ can be estimated using Maximum A Posteriori (MAP) estimation [50, 57, 58].

Naïve Bayes works only off the distribution of the training data, so there are no hyperparameters or options to select. For this work, a Gaussian distribution was assumed. Because there was no tuning, only the final testing data was used to create confusion matrices (For example, Figure 4.7a) which were used to evaluate which method and sensor configuration provided the best results. This was done with Scikit-Learn's *GaussianNB* module [52].

4 RESULTS

This chapter reviews the results of the experiment created in Chapter 3. It begins with a brief explanation of how the results were interpreted. It then covers the results by dividing them into three phases:

1. Phase 1: An initial evaluation of the all the methods and sensor arrangements.
    - k Nearest Neighbor - Section 4.2
    - Decision Tree - Section 4.3
    - Multi-layer Perceptron - Section 4.4
    - Support Vector Machine - Section 4.5
    - Naïve Bayes - Section 4.6
2. Phase 2: A deeper look at the best methods found in the first phase - Section 4.8
3. Phase 3: An exploration of the ability to identify foreign objects (leaves) that are in the gripper - Section 4.9

as described in Section 3.2.1. It concludes with a final summary of all the results in Section 4.10

4.1 Interpretation of Results

The results were evaluated based on the following criteria:

- A high precision for prediction of a good grip is desirable. This is given the highest priority.

- Prediction of a bad grip is also important, so a small reduction in good grip prediction precision is acceptable if it results in a reasonable gain in bad grip prediction precision.

Good grip precision is prioritized to enhance the efficiency of the gripper. As the gripper moves through its workspace, it will spend most of its time in places where it is a bad grip (i.e there is no berry). In this space, the prediction system is not very useful as the gripper position relative to the target will give better information. The prediction system plays a bigger role when the gripper has arrived at the target as it is much more likely to be a good grip. At this point, if it incorrectly identifies a bad grip as good, the robot will expend time and energy gripping, plucking and depositing nothing - so it is desirable to minimize this error as much as possible. If however, it mislabels a good grip as bad, the robot will re-position itself and make another prediction - a much less time and energy consuming action.

4.2 kth Nearest Neighbor

kth Nearest Neighbor (KNN) requires only the selection of a single hyperparameter, k. This was done as described in Section 3.2.2 and results can be seen in Figure 4.1. Some of the graphs show clear trends such as for the TOF sensor (Figure 4.1b) or the TOF-Color sensor (Figure 4.1d), while other such as the graph for the IR sensor (Figure 4.1a) are much noisier and thus more challenging to interpret. While interpreting, the goal was to maximize both the good and bad grip prediction accuracy and if these were in conflict with each other, to emphasize the good grip prediction as described in Section 4.1.

- For the IR sensor (Figure 4.1a), there are many large spikes that are difficult to interpret, but there seems to be an overall trend of good grip prediction decreasing until around $k = 10$ and then leveling off and the bad grip prediction increasing until

$k = 10$ and then leveling off. A value of $k = 6$ was selected to try to take advantage of both of these trends.

- For the TOF sensor (Figure 4.1b), the trends are much more consistent and thus a value of $k = 8$ was selected.

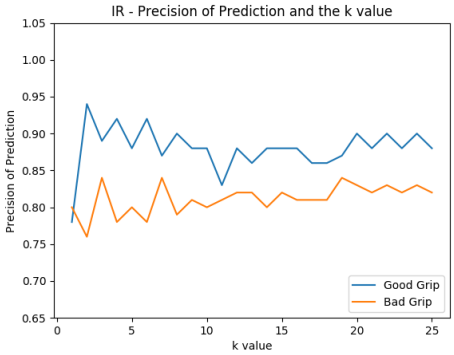
- For the IR-Color sensor arrangement (Figure 4.1c), there is a lot of noise, but the overall trend is clear with both values rising as k increases. A value of $k = 15$ was selected to best take advantage of this trend.

- For the TOF-Color sensor arrangement (Figure 4.1d), once again the values are very consistent. A value of $k = 8$ was selected.

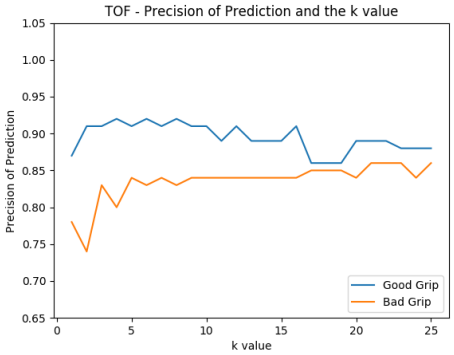
- For the IR-TOF-Color sensor arrangement (Figure 4.1e), there is once again a lot of noise, but the overall trend is clear with the good grip prediction value remaining relatively constant and the bad grip prediction precision rising slightly as k increases. A value of $k = 11$ was selected to best take advantage of this trend.

4.2.1 k^{th} Nearest Neighbor - Confusion Matrices

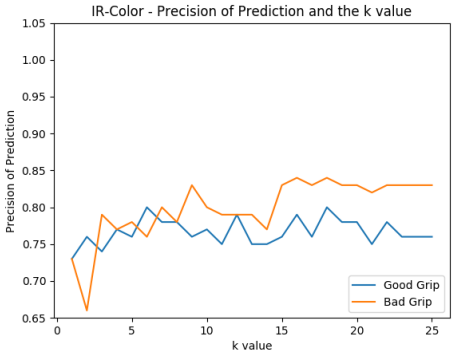
To complete the analysis of the KNN method, a confusion matrix was created for each sensor combination using the final test data set with the best performing hyperparameters as seen in Table 4.1. The confusion matrices can be seen in Figure 4.2. The best performing sensor set was the three TOF sensors (Figure 4.2b) with the combination of all three sensors (Figure 4.2e) also performing well. The other sensor combinations performed okay, but not exceptionally well. Overall KNN performed well, having the best average performance of all five methods.



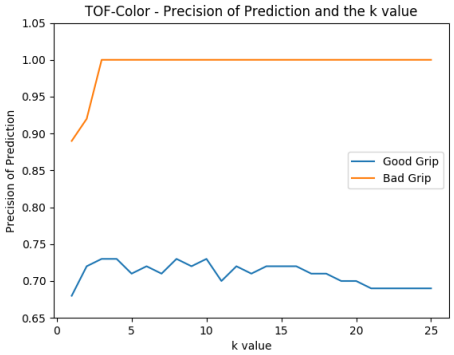
(a) IR KNN



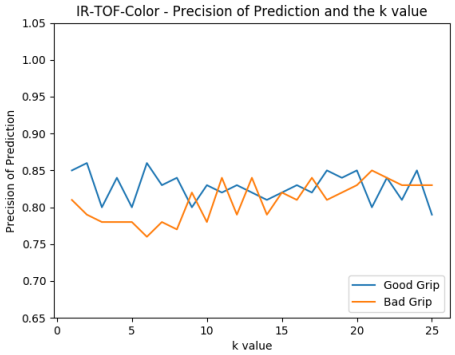
(b) TOF KNN



(c) IR-Color KNN

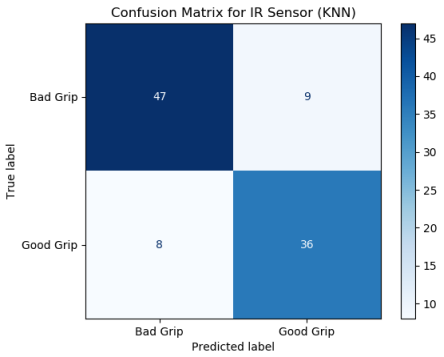


(d) TOF-Color KNN

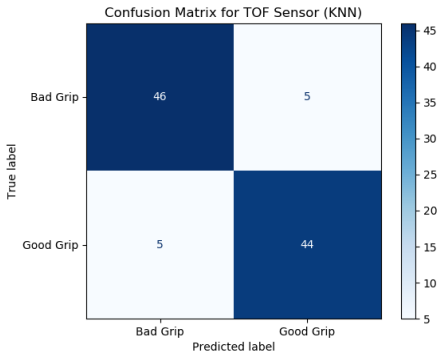


(e) IR-TOF-Color KNN

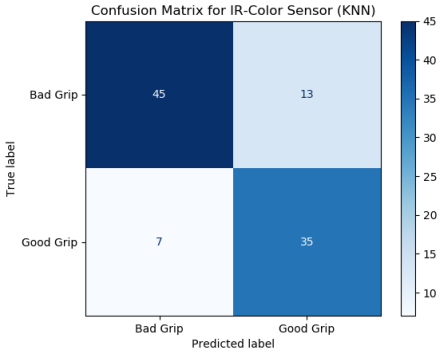
Figure 4.1: Phase 1: Selecting the k value for kth Nearest Neighbor



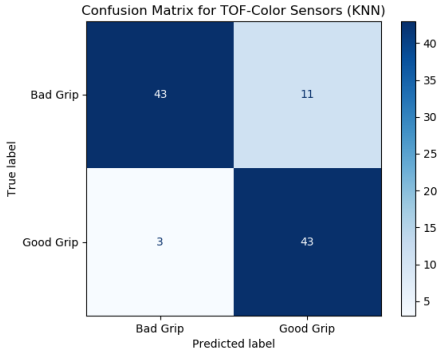
(a) IR KNN Confusion Matrix



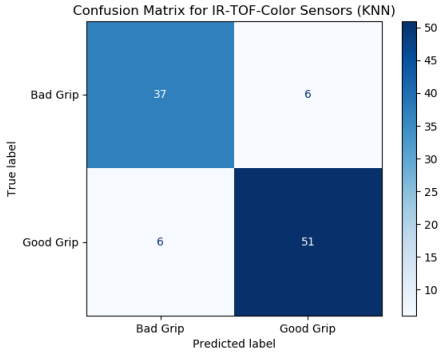
(b) TOF KNN Confusion Matrix



(c) IR-Color KNN Confusion Matrix



(d) TOF-Color KNN Confusion Matrix



(e) IR-TOF-Color KNN Confusion Matrix

Figure 4.2: Phase 1: KNN Confusion Matrices

Table 4.1: Phase 1: k value for the different sensor types

Sensor Type	k-value
IR	6
TOF	8
IR + Color	15
TOF + Color	8
IR + TOF + Color	11

4.3 Decision Trees

Two parameters were explored; the max depth pf the tree and the impurity measure.

- For the IR sensor (Figure 4.3a), there is a considerable amount of noise, but the overall trends are fairly clear. The entropy impurity measure shows slightly better results for bad grip prediction with deeper trees and for good grip with shallower trees. A depth of eight was selected to best take advantage of this trend.

- For the TOF sensor (Figure 4.3b), the trends are almost indistinguishable. A depth of eight was selected using the entropy method as it has shown better results in the rest of this work, though the performance in this sensor arrangement is indistinguishable.

- For the IR-Color sensor arrangement (Figure 4.3c), there is a lot of noise, but the trend for good grip prediction is clear and consistent between the two impurity measures with a shallower tree providing the best results. While the bad grip prediction methods diverge for deeper trees, they are very similar for shallower trees. A depth of four using the entropy method was selected.

- For the TOF-Color sensor arrangement (Figure 4.3d), the values are extremely noisy - especially for bad grip prediction with a deeper tree. A shallower tree depth of six was selected to avoid the noise along with the entropy method due to it slightly better performance in this region.

- For the IR-TOF-Color sensor arrangement (Figure 4.3e), there is once again a lot of noise especially for shallower trees. The good grip prediction for both methods is fairly consistent for deeper trees and entropy shows better performance for bad grip prediction. A depth of 10 was selected using the entropy method to take advantage of this trend.

4.3.1 Decision Trees - Confusion Matrices

To complete the analysis of the Decision Tree method, a confusion matrix was created for each sensor combination using the final test data set with the best performing hyperparameters and options as seen in Table 4.2. The confusion matrices can be seen in Figure 4.4. The best performing sensor set was the combination of all three sensors (Figure 4.4e). The other sensor combinations performed okay, with the exception of the TOF-Color combination (Figure 4.4d) which performed rather poorly.

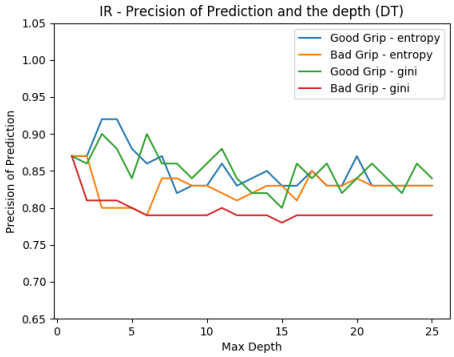
Table 4.2: Phase 1: Values for Decision Trees

Sensor Type	Depth	Impurity Measure
IR	8	Entropy
TOF	8	Entropy
IR + Color	4	Entropy
TOF + Color	6	Entropy
IR + TOF + Color	10	Entropy

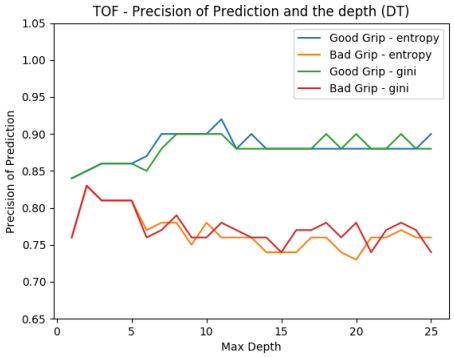
4.4 Multi-Layer Perceptron

Note: The tuple (3,2) represents the structure of the hidden layers - one with three nodes and a second with two nodes. (3,) would represent a single hidden layer with three nodes.

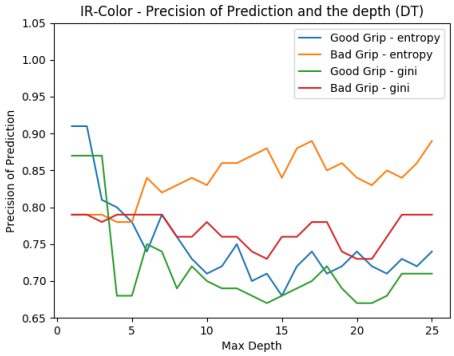
- For the IR sensor (Table 4.3), there is overall a strong ability to predict good grips. While the (3,3) arrangement showed the highest good grip precision. (3,) was selected due to the better balance between good and bad grip precision.



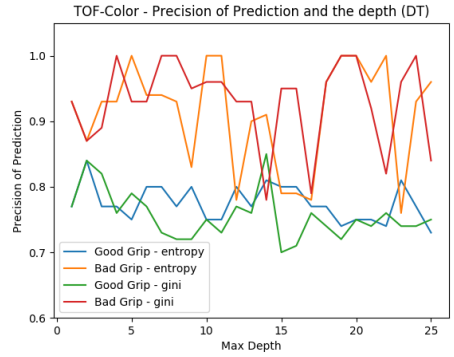
(a) IR DT



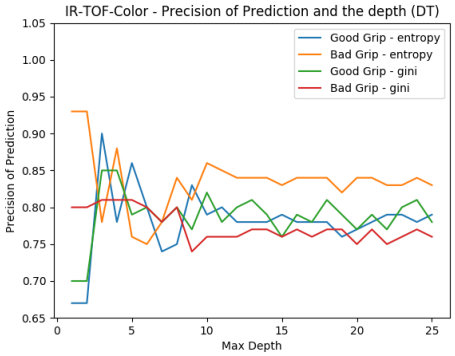
(b) TOF DT



(c) IR-Color DT

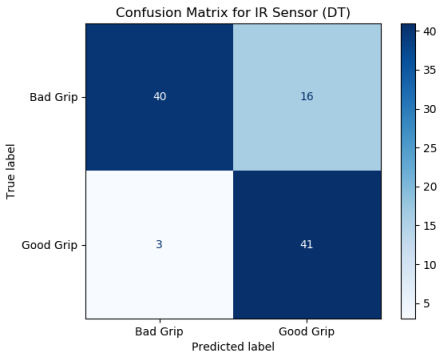


(d) TOF-Color DT

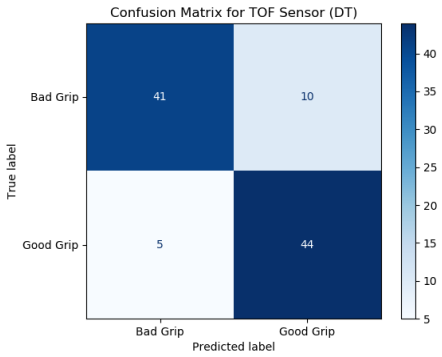


(e) IR-TOF-Color DT

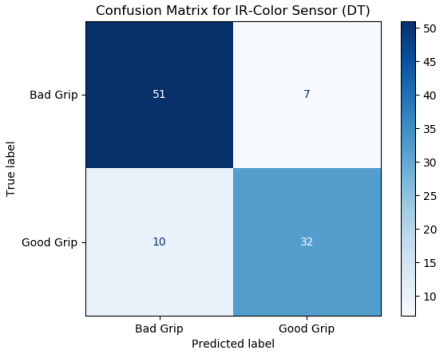
Figure 4.3: Phase 1: Selecting the depth value and split criterion for decision trees



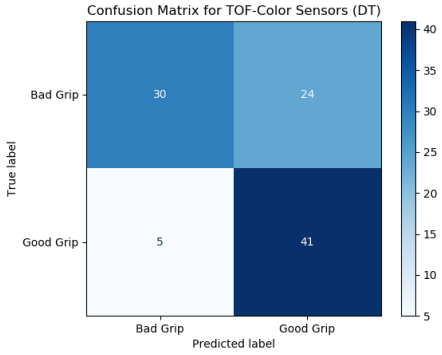
(a) IR DT Confusion Matrix



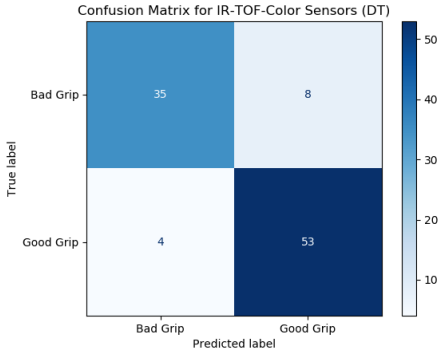
(b) TOF DT Confusion Matrix



(c) IR-Color DT Confusion Matrix



(d) TOF-Color DT Confusion Matrix



(e) IR-TOF-Color DT Confusion Matrix

Figure 4.4: Phase 1: Decision Trees Confusion Matrices

- For the TOF sensor (Table 4.4), most of the arrangements showed about the same results with the exception small first and second hidden layers. Several values showed identical results and as such from these (3,) was selected for the final test.

- For the IR-Color sensor arrangement (Table 4.5), with the exception of a few outliers, the results were fairly consistent across most of the arrangements. (4,2) was selected based on it slightly better performance.

- For the TOF-Color sensor arrangement (Table 4.6), there was considerable variation with in the results depending the arrangement. Two arrangements, (5,3) and (4,3) showed markedly better results and as they were identical, (5,3) was selected for the final test.

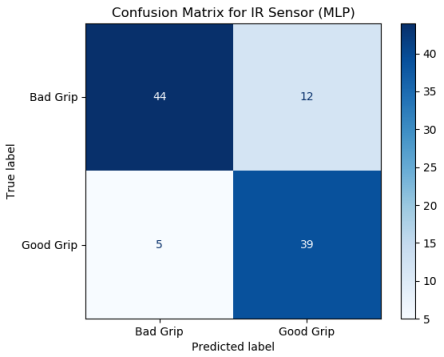
- For the IR-TOF-Color sensor arrangement (Table 4.7), the was not a large variation within the results. The (5,2) arrangement was used due to its slightly better performance.

4.4.1 Multi-Layer Perceptron - Confusion Matrices

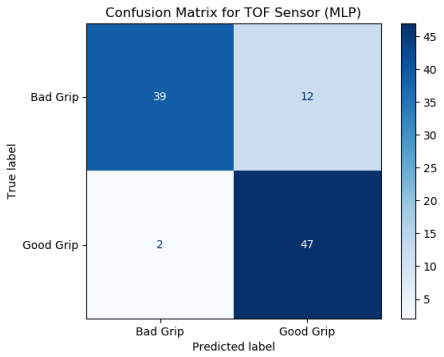
To complete the analysis of the Multi-Layer Perceptron method, a confusion matrix was created for each sensor combination using the final test data set with the best performing hyperparameters as seen in Table 4.8. The confusion matrices can be seen in Figure 4.5. The best performing sensor set was the combination of all three sensors (Figure 4.5e). The other sensor combinations performed poorly - especially the TOF-Color sensor set up which identified many bad grips as a good grip.

4.5 Support Vector Machines

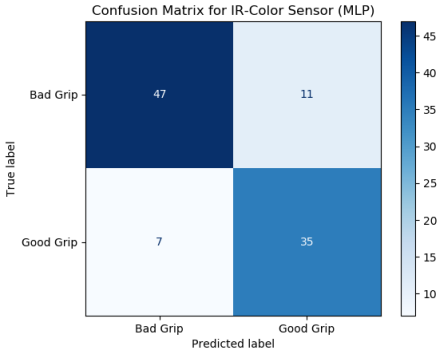
The Support Vector Machine (SVM) method has been tested in accordance with Section 3.2.4. The results have been summarized in table form with the columns representing



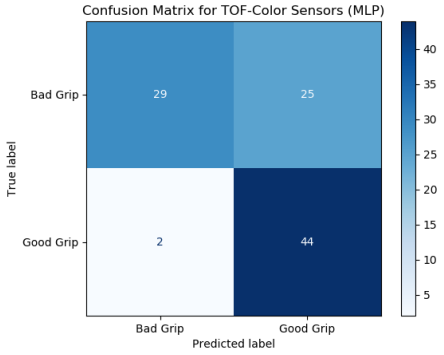
(a) IR MLP Confusion Matrix



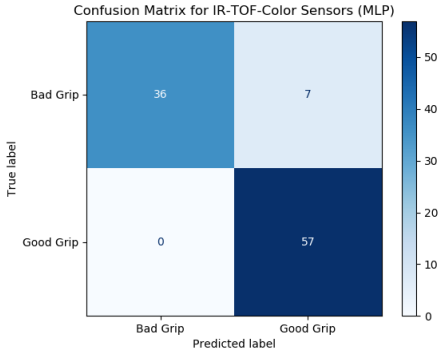
(b) TOF MLP Confusion Matrix



(c) IR-Color MLP Confusion Matrix



(d) TOF-Color MLP Confusion Matrix



(e) IR-TOF-Color MLP Confusion Matrix

Figure 4.5: Multi-layer Perceptron Confusion Matrices

Table 4.3: Phase 1: Selecting hidden layer size for MLP for IR

First Layer		**3**	**2**	**3**	**3**	**2**
Second Layer		**X**	**X**	**3**	**2**	**2**
True Label	**Predicted Label**					
Good	**Good**	40	38	33	39	39
	Bad	8	10	15	9	9
Bad	**Good**	4	4	1	4	4
	Bad	48	48	51	48	48
Good Precision		0.91	0.9	0.97	0.91	0.91
Bad Precision		0.86	0.83	0.77	0.84	0.84

Table 4.4: Phase 1: Selecting hidden layer size for MLP for TOF

First Layer		**3**	**2**	**3**	**3**	**2**
Second Layer		**X**	**X**	**3**	**2**	**2**
True Label	**Predicted Label**					
Good	**Good**	50	50	51	50	50
	Bad	7	7	6	7	7
Bad	**Good**	8	9	9	8	8
	Bad	35	34	34	35	35
Good Precision		0.86	0.85	0.85	0.86	0.86
Bad Precision		0.83	0.83	0.85	0.83	0.83

the four different kernels tested: Radial Basis Function ('RBF'), polynomial ('poly'), 'linear' and 'sigmoid'. The rows contain the same data as a confusion matrix, just in a more compact form. The results were analyzed as follows:

- For the IR sensor (Table 4.9), a polynomial kernel was selected primarily due to having the minimum number of mislabeled tests results.
- For the TOF sensor (Table 4.10), a polynomial kernel was selected primarily due to having the minimum number of mislabeled tests results and also the high 'Good Precision' results.
- For the IR-Color sensor arrangement (Table 4.11), a 'RBF' kernel was selected primarily due to having the minimum number of mislabeled tests results and also

Table 4.5: Phase 1: Selecting hidden layer size for MLP for IR-Color

First Layer		**5**	**4**	**3**	**2**	**5**	**5**	**5**
Second Layer		**X**	**X**	**X**	**X**	**5**	**4**	**3**
True Label	**Predicted Label**							
Good	**Good**	45	44	46	45	46	47	46
	Bad	7	8	6	7	6	5	6
Bad	**Good**	14	12	16	17	16	19	20
	Bad	34	36	32	31	32	29	28
Good Precision		0.76	0.79	0.74	0.73	0.74	0.71	0.7
Bad Precision		0.83	0.82	0.84	0.82	0.84	0.85	0.82
First Layer		**5**	**4**	**4**	**4**	**3**	**3**	**2**
Second Layer		**2**	**4**	**3**	**2**	**3**	**2**	**2**
True Label	**Predicted Label**							
Good	**Good**	46	45	45	45	45	52	46
	Bad	6	7	7	7	7	0	6
Bad	**Good**	17	13	13	12	16	48	15
	Bad	31	35	35	36	32	0	33
Good Precision		0.73	0.78	0.78	0.79	0.74	0.52	0.75
Bad Precision		0.84	0.83	0.83	0.84	0.82	NA	0.85

the high ‘Good Precision’ results.

- For the TOF-Color sensor arrangement (Table 4.12), a ‘linear’ kernel was selected primarily due to having the minimum number of mislabeled tests results and also the high ‘Good Precision’ results.

- For the IR-TOF-Color sensor arrangement (Table 4.13), a ‘linear’ kernel was selected primarily due to having the minimum number of mislabeled tests results and also the high ‘Good Precision’ results.

4.5.1 Support Vector Machines - Confusion Matrices

To complete the analysis of the Support Vector Machine method, a confusion matrix was created for each sensor combination using the final test data set with the best performing hyperparameters as seen in Table 4.14. The confusion matrices can be seen in Figure 4.6.

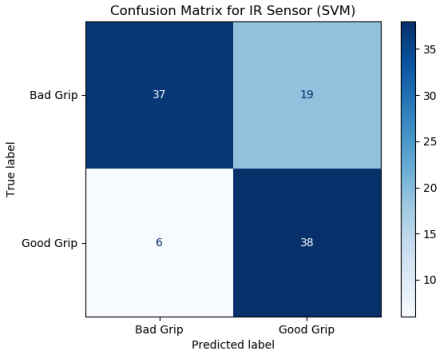
Table 4.6: Phase 1: Selecting hidden layer size for TOF-Color

First Layer		**5**	**4**	**3**	**2**	**5**	**5**	**5**
Second Layer		**X**	**X**	**X**	**X**	**5**	**4**	**3**
True Label	**Predicted Label**							
Good	**Good**	57	57	52	57	57	55	49
	Bad	0	0	5	0	0	2	8
Bad	**Good**	23	17	11	14	18	14	5
	Bad	20	26	32	29	25	29	38
Good Precision		0.71	0.77	0.83	0.8	0.76	0.8	0.91
Bad Precision		1.0	1.0	0.86	1.0	1.0	0.94	0.83
First Layer		**5**	**4**	**4**	**4**	**3**	**3**	**2**
Second Layer		**2**	**4**	**3**	**2**	**3**	**2**	**2**
True Label	**Predicted Label**							
Good	**Good**	57	56	49	57	57	57	50
	Bad	0	1	8	0	0	0	7
Bad	**Good**	18	14	5	16	14	19	8
	Bad	25	29	38	27	29	24	35
Good Precision		0.76	0.8	0.91	0.78	0.8	0.75	0.86
Bad Precision		1.0	0.97	0.83	1.0	1.0	1.0	0.83

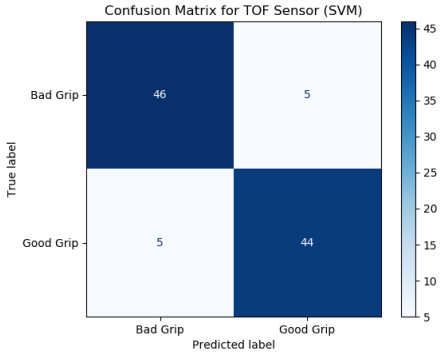
The best performing sensor sets were the set of three TOF sensors (Figure 4.6b) and the combination of all three sensors (Figure 4.6e). The other sensor combinations did not perform well.

4.6 Naïve Bayes

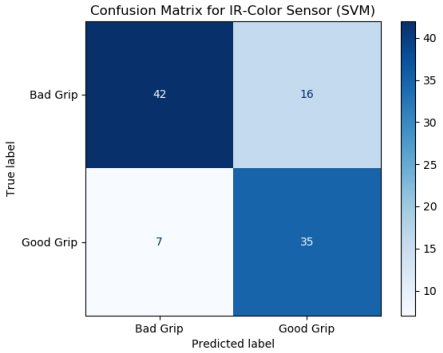
For the Naïve Bayes method, there were no parameters to adjust. Testing was performed in accordance with Section 3.2.6. To complete the analysis, a confusion matrix was created for each sensor combination using the final test data set. The confusion matrices can be seen in Figure 4.7. The best performing sensor set was the combination of all three sensors (Figure 4.7e). The other sensor combinations performed very well and overall Naïve Bayes was second only to KNN in average performance.



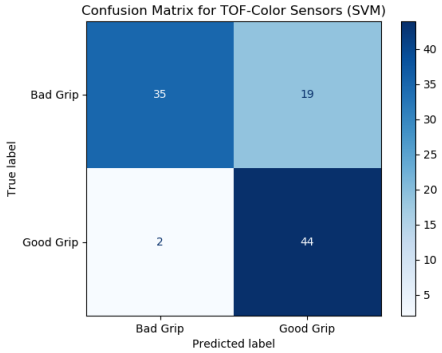
(a) IR SVM Confusion Matrix



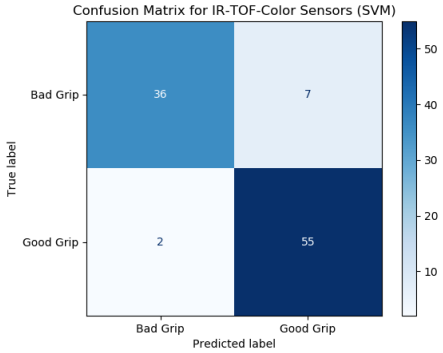
(b) TOF SVM Confusion Matrix



(c) IR-Color SVM Confusion Matrix

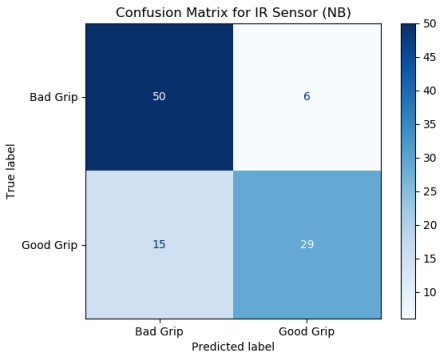


(d) TOF-Color SVM Confusion Matrix

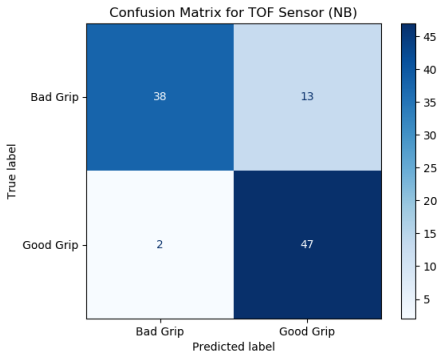


(e) IR-TOF-Color SVM Confusion Matrix

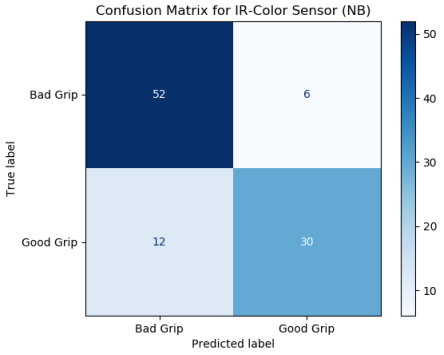
Figure 4.6: Phase 1: Support Vector Machines Confusion Matrices



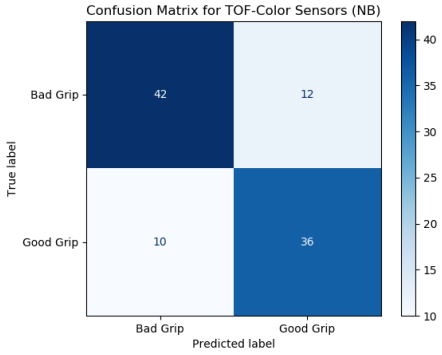
(a) IR NB Confusion Matrix



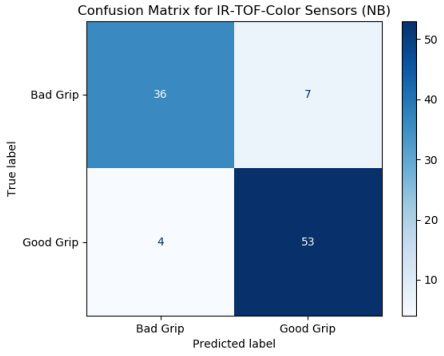
(b) TOF NB Confusion Matrix



(c) IR-Color NB Confusion Matrix



(d) TOF-Color NB Confusion Matrix



(e) IR-TOF-Color NB Confusion Matrix

Figure 4.7: Phase 1: Naïve Bayes Confusion Matrices

Table 4.7: Phase 1: Selecting hidden layer size for MLP for IR-TOF-Color

First Layer		**5**	**4**	**3**	**2**	**5**	**5**	**5**
Second Layer		**X**	**X**	**X**	**X**	**5**	**4**	**3**
True Label	**Predicted Label**							
Good	**Good**	46	43	45	43	46	45	43
	Bad	4	7	5	7	4	5	7
Bad	**Good**	11	9	10	10	12	11	10
	Bad	39	41	40	40	38	39	40
Good Precision		0.81	0.83	0.82	0.81	0.79	0.8	0.81
Bad Precision		0.91	0.85	0.89	0.85	0.9	0.89	0.85
First Layer		**5**	**4**	**4**	**4**	**3**	**3**	**2**
Second Layer		**2**	**4**	**3**	**2**	**3**	**2**	**2**
True Label	**Predicted Label**							
Good	**Good**	45	44	45	46	45	42	46
	Bad	5	6	5	4	5	8	4
Bad	**Good**	9	11	12	11	9	11	11
	Bad	41	39	38	39	41	39	39
Good Precision		0.83	0.8	0.79	0.81	0.83	0.79	0.81
Bad Precision		0.89	0.87	0.88	0.91	0.89	0.83	0.91

4.7 Summary of Phase 1 Testing

A summary of initial testing of of all five sensor configurations with all five classification methods can be seen in Table 4.15. This table consolidates all of the confusion matrix data presented in the individual sections. Based on a review using the criteria of Section 4.1, several sensor combinations and methods stood out.

- Overall, the best performing method was KNN with Naïve Bayes a close second.

Table 4.8: Phase 1: Values for MLP

Sensor Type	First Layer	Second Layer
IR	3	None
TOF	3	None
IR + Color	4	2
TOF + Color	5	3
IR + TOF + Color	5	2

Table 4.9: Phase 1: Selecting a kernel for SVM for IR

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	33	44	34	34
	Bad	15	4	14	14
Bad	**Good**	3	7	3	10
	Bad	49	45	49	42
Good Precision		0.92	0.86	0.92	0.77
Bad Precision		0.77	0.92	0.78	0.75

Table 4.10: Phase 1: Selecting a kernel for SVM for TOF

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	51	47	50	51
	Bad	6	10	7	6
Bad	**Good**	7	3	8	8
	Bad	36	40	35	35
Good Precision		0.88	0.94	0.86	0.86
Bad Precision		0.86	0.8	0.83	0.85

- The best performing sensor combination was IR-TOF-Color, which had strong, consistent results across all the methods.
- The three TOF sensors showed generally good results and showed the overall best results when paired with either the KNN or SVM methods.
- The IR-TOF-Color sensor combination showed its best results when paired with either the SVM or MLP methods.

The most surprising result of Phase 1, was the poor performance of the TOF-Color sensor combination. Given the strength of the TOF by itself and the mixed performance of the IR sensor, it seemed natural that the TOF-Color combination would perform strongly, but instead it was the combination of all three sensors that was the most consistent good performer. A possible explanation of this is that the three sensor combination gave a

Table 4.11: Phase 1: Selecting a kernel for SVM for IR-Color

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	44	46	45	35
	Bad	8	6	7	17
Bad	**Good**	13	20	17	26
	Bad	35	28	31	22
Good Precision		0.77	0.7	0.73	0.57
Bad Precision		0.77	0.7	0.73	0.57

Table 4.12: Phase 1: Selecting a kernel for SVM for TOF-Color

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	57	57	51	39
	Bad	0	0	6	18
Bad	**Good**	24	24	7	38
	Bad	19	19	36	5
Good Precision		0.7	0.7	0.88	0.51
Bad Precision		1.0	1.0	0.86	0.22

richer signal than the other combinations. The behavior of the IR reflectance sensor is related to distance, but not quite the same. It has a wider 'beam' and thus responds to a lot movements that the narrow 'beamed' TOF sensor would not detect. Thus combining the three gives three distinct features and helps create a rich, identifiable sensor reading. That still leaves the question as to why the TOF sensor did so well given its lack of richness. One idea is that the precision of the signal made up for that, but only with the three sensors combined. The removal of one sensor undermined this and led to the poor showing for the TOF-Color sensor combination.

In addition to the work presented here, KNN was tested on the SoftAgbot as discussed in [39]. In that work, the system was tested using real cherry tomatoes attached to an artificial plant. These tests showed the initial promise of the idea and helped identify its limitations. The system successfully identified when a fruit was in the gripper and

Table 4.13: Phase 1: Selecting a kernel for SVM for IR-TOF-Color

Kernel		**RBF**	**Poly**	**Linear**	**Sigmoid**
True Label	**Predicted Label**				
Good	**Good**	42	49	44	40
	Bad	8	1	6	10
Bad	**Good**	10	15	10	14
	Bad	40	35	40	36
Good Precision		0.81	0.77	0.81	0.74
Bad Precision		0.83	0.97	0.87	0.78

Table 4.14: Phase 1: Values for SVM

Sensor Type	Kernel
IR	Polynomial
TOF	Polynomial
IR + Color	RBF
TOF + Color	Linear
IR + TOF + Color	Linear

those fruit were successfully picked. However with human-in-the-loop control, it was impossible to independently judge the success of the system. It was also noted that leaves and stems presented obstacles that created false positives and thus additional testing was necessary.

4.8 Phase 2: Further Exploration of the Best Performing Sensors and Methods

Based on the results of Phase 1, the best performers were selected for further exploration. As seen in Table 4.15, the best sensor-method combinations were: TOF with KNN, TOF with SVM, IR-TOF-Color with SVM and IR-TOF-Color with MLP. For both of the sensor combinations, an additional 500 training elements was collected

Table 4.15: Phase 1: Summary of the confusion matrices

		KNN					Decision Trees				
True Label	**Predicted Label**	**IR**	**TOF**	**IR+ Color**	**TOF+ Color**	**IR+ TOF+ Color**	**IR**	**TOF**	**IR+ Color**	**TOF+ Color**	**IR+ TOF+ Color**
Good	**Label**	36	44	35	43	51	41	44	32	41	53
	Bad	8	5	7	3	6	16	10	10	5	4
Bad	**Good**	9	5	13	11	6	3	5	7	24	8
	Bad	47	46	45	43	37	40	41	51	30	35
Good Precision		0.8	0.9	0.73	0.8	0.89	0.72	0.81	0.82	0.63	0.87
Bad Precision		0.85	0.9	0.87	0.93	0.86	0.93	0.89	0.84	0.86	0.9
		Support Vector Machines					**Multi-Layer Perceptron**				
Good	**Good**	38	44	35	44	55	39	47	35	44	57
	Bad	6	5	7	2	2	5	2	7	2	0
Bad	**Good**	19	5	16	19	7	12	12	11	25	7
	Bad	37	46	42	35	36	44	39	47	29	36
Good Precision		0.67	0.9	0.69	0.7	0.89	0.76	0.8	0.76	0.64	0.89
Bad Precision		0.86	0.9	0.86	0.95	0.95	0.9	0.95	0.87	0.94	1.0
		Naïve Bayes									
Good	**Good**	29	47	30	36	53					
	Bad	15	2	12	10	4					
Bad	**Good**	6	13	6	12	7					
	Bad	50	38	52	42	36					
Good Precision		0.83	0.78	0.83	0.75	0.88					
Bad Precision		0.77	0.95	0.81	0.81	0.9					

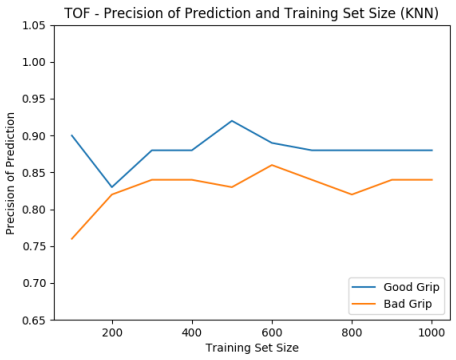
4.8.1 Further Exploration of the Time of Flight Sensor

k^{th} Nearest Neighbor

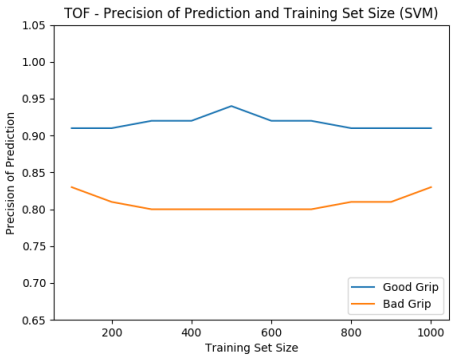
For the TOF sensor combined with KNN, the effect of training data size was tested. Training test sets from 100 to 1000 elements in 100 element increments were evaluated using the test data set. This was done with the k value determined in Phase 1 as seen in Table 4.1. Figure 4.8a showed that beyond 3-400 elements, the size of the training data played little role in the precision of the results. The resulting confusion matrix (Figure 4.9a) showed results consistent with the earlier tests (Figure 4.2b).

Support Vector Machine

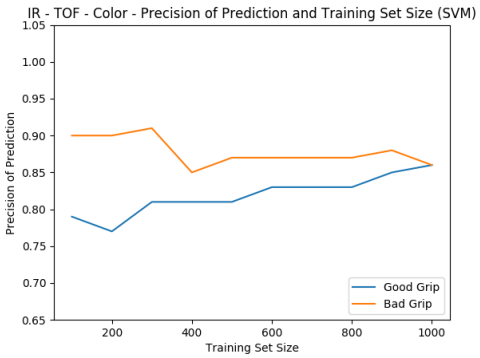
For the TOF sensor combined with SVM, the effect of training data size was tested. Training test sets from 100 to 1000 elements in 100 element increments were evaluated



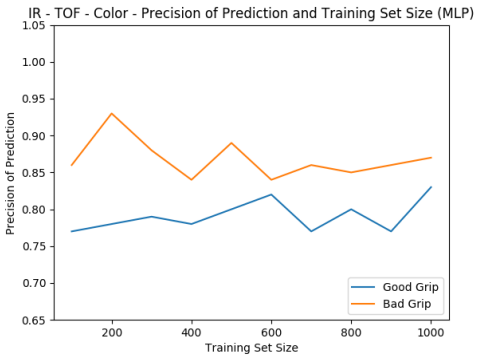
(a) TOF KNN Training set size



(b) TOF SVM Training set size



(c) IR-TOF-Color SVM Training set size



(d) IR-TOF-Color MLP Training set size

Figure 4.8: Phase 2: Effect of training set size on precision

using the test data set. This was done with the parameters determined in Phase 1 as seen in Table 4.14. Figure 4.8b showed that the size of the training data played little role in the precision of the results. The resulting confusion matrix (Figure 4.9b) showed results consistent with the earlier tests (Figure 4.6b).

4.8.2 Further Exploration of the IR-Time of Flight-Color Sensor

Support Vector Machine

For the IR-TOF-Color sensor arrangement combined with SVM, the effect of training data size was tested. Training test sets from 100 to 1000 elements in 100 element increments were evaluated using the test data set. This was done with the parameters determined in Phase 1 as seen in Table 4.14. Figure 4.8c showed that the size of the training data played little role in the bad grip prediction precision, but an increasing trend for the good grip prediction precision. It is possible that further training data could lead to a further increase in the results. The resulting confusion matrix (Figure 4.9c) showed results consistent with the earlier tests (Figure 4.6e).

Multi-layer Perceptron

For the IR-TOF-Color sensor arrangement combined with MLP, the effect of training data size was tested. Training test sets from 100 to 1000 elements in 100 element increments were evaluated using the test data set. This was done with the parameters determined in Phase 1 as seen in Table 4.8. Figure 4.8d showed that the size of the training data played little role in the precision of the results. In addition to training set size, the activation functions and solvers were tested. Table 4.16 shows how the combination of different activation functions and solvers affected precision. In each entry in the chart, the upper number is the ‘good precision’ and the lower number is the ‘bad precision’. There were several strong performing combinations with ’lbfgs’-’identity’, ’sgd’-’identity’, ’sgd’-’logistic’,

and 'adam'-'relu' all showing the best results. The resulting confusion matrix (Figure 4.9d) was created with 'sgd'-'identity', but all the combinations showed identical results. It shows strong results consistent with the earlier tests (Figure 4.5e).

Table 4.16: Phase 2: Selecting an activation function and solver for MLP with IR-TOF-Color sensors - Note: top entry is good precision and bottom entry is bad precision

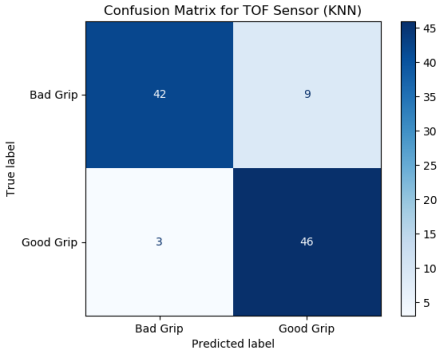
		Activation Function			
		identity	**logistic**	**tanh**	**relu**
Solver	**lbfgs**	0.86	0.72	0.78	0.84
		0.84	0.82	0.93	0.91
	sgd	0.86	0.86	0.82	0.83
		0.84	0.84	0.89	0.87
	adam	0.84	0.81	0.8	0.86
		0.84	0.91	0.87	0.84

4.8.3 Summary of Phase 2 Results

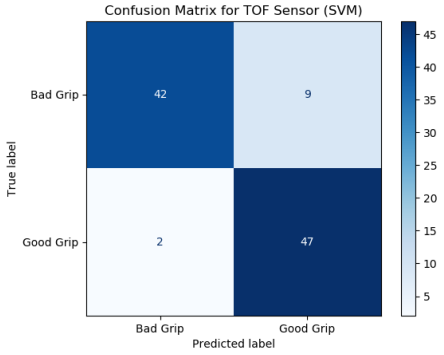
Table 4.17: Phase 2: Summary of the confusion matrices

		KNN	**SVM**		**MLP**
True Label	**Predicted Label**	**TOF**	**TOF**	**IR+ TOF+ Color**	**IR+ TOF+ Color**
Good	**Good**	46	47	53	54
	Bad	3	2	4	3
Bad	**Good**	9	9	7	6
	Bad	42	42	36	37
Good Precision		0.84	0.84	0.88	0.9
Bad Precision		0.93	0.95	0.9	0.93

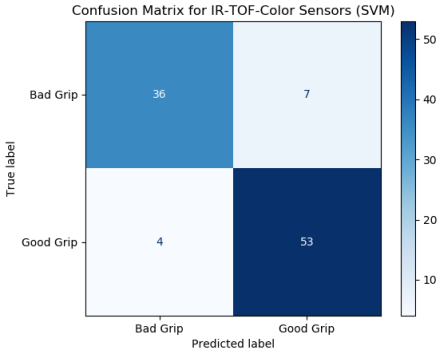
The results of Phase 2 can be seen in Table 4.17. As expected, all the methods and sensor configurations did well and performed consistently with the previous results as seen in Table 4.15. The IR-TOF-Color sensor combination performed slightly better than the TOF flight sensor by itself. It was surprising that the addition of training data did not have an significant effect on the precision - even to the extent that 2-300 training elements may be sufficient to achieve the best available results. This suggest a few possibilities,



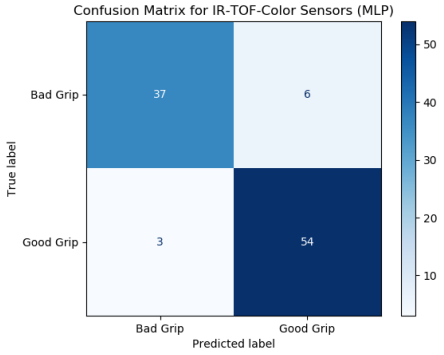
(a) TOF KNN Phase 2 Confusion Matrix



(b) TOF SVM Phase 2 Confusion Matrix



(c) IR-TOF-Color SVM Phase 2 Confusion Matrix



(d) IR-TOF-Color MLP Phase 2 Confusion Matrix

Figure 4.9: Phase 2 Confusion Matrices

such as the improperly classified test elements were just particularly stubborn to properly classify - perhaps they were marginal cases or even mislabeled, though the later seems unlikely for such a large fraction of the data. It is more likely that additional data was simply not richer in information. It was just a repetition of existing data and thus was not able to help resolve these outliers properly. Given a real situation where the fruit would have more variety in size, shape, color and gripping angle, a larger data set may be key, but with the simplified nature of this experiment, it did not play a big role.

4.9 Phase 3: Addition of Leaves

Using the lessons learned from Phase 1 and Phase 2, one more exploration was conducted by adding an additional category to classify - a leaf. The best methods of Phase 1 were used with the additional training data of Phase 2, but because the addition of a third label significantly changes the problem, the analysis of Phase 1 was re-performed to find the best hyperparameters and options.

KNN with Time of Flight Sensors

With the new leaf data, the k value was found as described in Section 3.2.2 and results can be seen in Figure 4.10. Based on these results, a value of $k = 6$ was selected. Using this value, it was then tested with the final test data set. The results can be seen in Figure 4.11a. The method did a good job of correctly labeling a leaf, but the additional class had a negative effect on the ability to identify the quality of the grip.

Support Vector Machine with Time of Flight Sensors

With the new leaf data, a new kernel function was selected as described in Section 3.2.4. The results can be seen in Table 4.18. The 'RBF' kernel provided the best results. Using this, the method was tested with the final test data set. The results can be seen in Figure

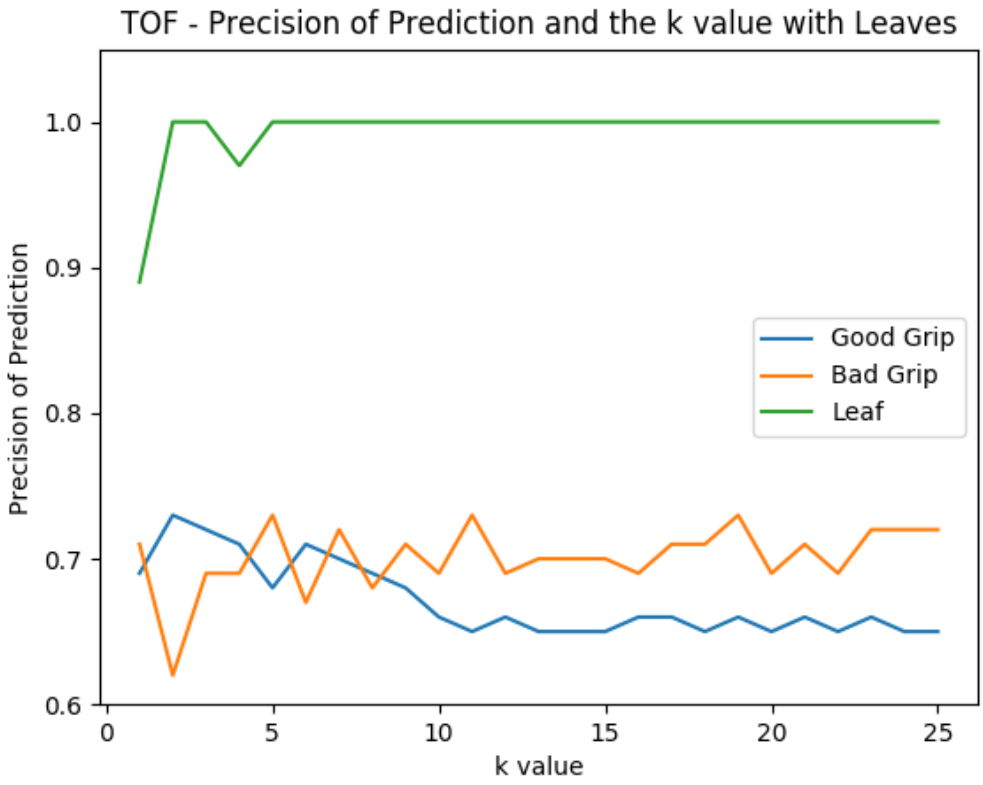
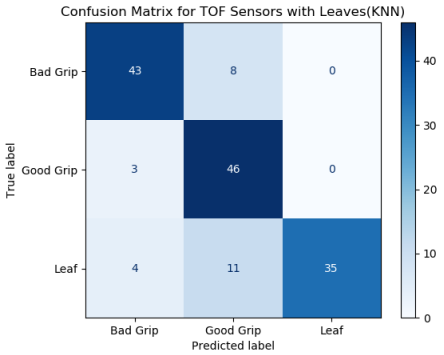


Figure 4.10: Phase 3: Selecting the k value for TOF with the addition leaves

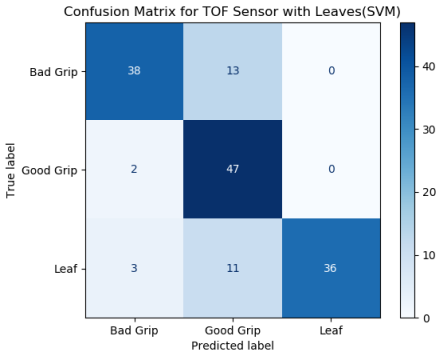
4.11b. The method did a good job of correctly labeling a leaf, but the additional class had a negative effect on the ability to identify the quality of the grip.

Support Vector Machine with IR-Time of Flight-Color Sensor Combination

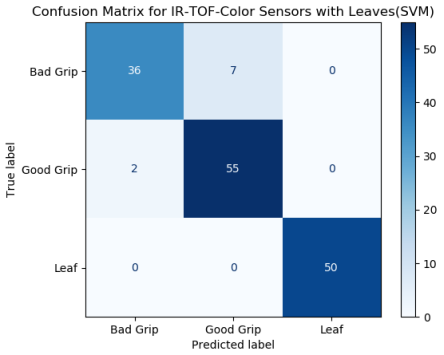
With the new leaf data, a new kernel function was selected as described in Section 3.2.4. The results can be seen in Table 4.19. The 'linear' kernel provided the best results. Using this, the method was tested with the final test data set. The results can be seen in Figure 4.11c. The method did a good job of correctly labeling a leaf, while still maintaining a high precision at predicting a good or bad grip.



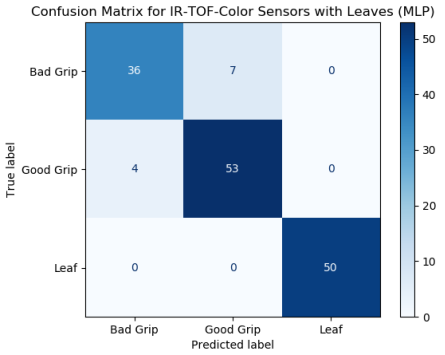
(a) TOF KNN with leaves Confusion Matrix



(b) TOF SVM with leaves Confusion Matrix



(c) IR-TOF-Color SVM with leaves Confusion Matrix



(d) IR-TOF-Color MLP with leaves Confusion Matrix

Figure 4.11: Phase 3 Confusion Matrices - include the addition of leaves as a classification

Multi-layer Perceptron with IR-Time of Flight-Color Sensor Combination

With the new leaf data, the number of hidden layers along with an activation function and solver were selected as described in Section 3.2.5. The results can be seen in Table 4.20 and 4.21. A hidden layer of size (3,3) was selected along with the 'identity' activation function and the 'lbfgs' solver. Using this, the method was tested with the final test data set. The results can be seen in Figure 4.11d. The method did a good job of correctly labeling a leaf, while still maintaining a high precision at predicting a good or bad grip.

Table 4.18: Phase 3: Selecting a kernel function for SVM with TOF sensors and the addition of leaves

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	52	53	52	43
	Bad	5	4	5	4
	Leaf	0	0	0	10
Bad	**Good**	8	8	8	8
	Bad	35	35	35	34
	Leaf	0	0	0	1
Leaf	**Good**	14	30	39	30
	Bad	4	10	11	14
	Leaf	32	10	0	6
Good Precision		0.7	0.58	0.53	0.53
Bad Precision		0.8	0.71	0.69	0.65
Leaf Precision		1.0	1.0	NA	0.35

4.9.1 Summary of Phase 3 Results

With leaves added to the system, all four methods/combinations showed a strong ability to identify leaves, but without the color sensor, this came at the expense of correctly identifying the other classes. With the addition of the color sensor, leaves were always correctly identified and other classes were not mistaken for leaves. This is not a surprising result as the color sensor is a much richer source of information and the distinction between the leaf and other objects is much greater. With only the distance information provided by the TOF sensor, the difference is present, but much less distinct. A summary of the results of Phase 3 can be seen in Table 4.22

4.10 Final Results Summary

The results of this experiment are best seen in Table 4.15, which summarizes the results of Phase 1, the general testing, Table 4.17, which summarizes the results of Phase 2, the deeper exploration, and Table 4.22, which summarizes Phase 3, the addition of leaves as a

Table 4.19: Phase 3: Selecting a kernel function for SVM with IR-TOF-Color sensors and the addition of leaves

Kernel		RBF	Poly	Linear	Sigmoid
True Label	**Predicted Label**				
Good	**Good**	44	49	44	44
	Bad	6	1	6	6
	Leaf	0	0	0	0
Bad	**Good**	10	23	10	12
	Bad	40	27	40	38
	Leaf	0	0	0	0
Leaf	**Good**	2	14	0	1
	Bad	0	0	0	5
	Leaf	48	36	50	44
Good Precision		0.79	0.57	0.81	0.77
Bad Precision		0.87	0.96	0.87	0.78
Leaf Precision		1.0	1.0	1.0	1.0

class. In Phase 1, KNN performed well on average for all sensors and the IR-TOF-Color combination performed well for all the machine learning methods. Most tests performed well and there was only a small difference between them and the top performers. Given the variation seen in some of the tests, it is possible they could perform better under slightly different circumstances. The top performers TOF with KNN and SVM and IR-TOF-Color combination with SVM and MLP continued to perform well in Phase 2. The addition of more training did not show much of an effect on the precision nor did the testing of other parameters in the case of MLP. This suggest that given the test protocol and the hardware, around 90% good prediction precision is as good as can be achieved. To improve the results, a different approach needs to be taken. With the addition of the leaf class in Phase 3, the sensors and methods behaved as expected. The TOF performed quite poorly and while having a high leaf prediction precision, it came at the expense of prediction of the grip. The IR-TOF-Color combination performed quite well with the addition of the leaf data and shows promise as a way to identify foreign material in the gripper.

Table 4.20: Phase 3: Results of varying the number and size of hidden layers for MLP with IR-TOF-Color sensors and the addition of leaves

First Layer		**5**	**4**	**3**	**5**	**5**	**5**	**4**	**4**	**3**
Second Layer		**X**	**X**	**X**	**5**	**4**	**3**	**4**	**3**	**3**
True Label	**Predicted Label**									
Good	**Good**	44	41	41	42	42	44	42	41	39
	Bad	6	9	9	8	8	6	8	9	11
	Leaf	0	0	0	0	0	0	0	0	0
Bad	**Good**	10	11	8	11	10	9	10	10	7
	Bad	40	39	42	39	40	41	40	40	43
	Leaf	0	0	0	0	0	0	0	0	0
Leaf	**Good**	0	0	0	0	0	0	0	0	0
	Bad	0	0	0	0	0	0	0	0	0
	Leaf	50	50	50	50	50	50	50	50	50
Good Precision		0.81	0.79	0.84	0.79	0.81	0.83	0.81	0.8	0.85
Bad Precision		0.87	0.81	0.82	0.83	0.83	0.87	0.83	0.82	0.8
Leaf Precision		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 4.21: Phase 3: Selecting an activation function and solver for MLP with IR-TOF-Color sensors with and the addition of leaves - Note: top entry is good precision, middle is entry is bad precision and bottom entry is leaf precision

		Activation Function			
		identity	**logistic**	**tanh**	**relu**
Solver	**lbfgs**	0.86	0.78	0.81	0.83
		0.84	0.83	0.87	0.87
		1.0	1.0	1.0	1.0
	sgd	0.81	0.81	0.82	0.85
		0.85	0.87	0.82	0.8
		1.0	1.0	1.0	1.0
	adam	0.81	0.78	0.79	0.81
		0.83	0.88	0.90	0.87
		1.0	1.0	1.0	1.0

Table 4.22: Phase 3: Summary of the results of the addition of the 'Leaf' class

		KNN	SVM		MLP
True Label	**Predicted Label**	**TOF**	**TOF**	**IR+ TOF+ Color**	**IR+ TOF+ Color**
Good	**Good**	46	47	55	53
	Bad	3	2	2	4
	Leaf	0	0	0	0
Bad	**Good**	8	13	7	7
	Bad	43	38	36	36
	Leaf	0	0	0	0
Leaf	**Good**	11	11	0	0
	Bad	4	3	0	0
	Leaf	35	36	50	50
Good Precision		0.71	0.66	0.89	0.88
Bad Precision		0.86	0.88	0.95	0.90
Leaf Precision		1.0	1.0	1.0	1.0

5 CONCLUSIONS AND FUTURE WORK

The results of Chapter 4 and particularly those of Phase 2 (Table 4.17) and Phase 3 (Table 4.22), show the promise of the idea of predicting grip success without contact. The TOF sensor configuration with the KNN and SVM classifiers and the IR-TOF-Color sensor configuration with the SVM and MLP classifiers showed the ability to predict grip success and classify leaves with great precision. With the given approach, however, the precision seems to top out at around 90%, which while good, may not be good enough for a robot that needs to pick 1000s of fruit a day. The failure rate would significantly impact the harvest rate and potential damage the improperly gripped fruit. It could also leave a significant number of fruit unpicked. Therefore work needs to continue to improve the precision of the results. Some possible areas of improvement and future work:

- Sensor aim and position - varying the aim of the sensors could provide a richer feature set with more spatial information.
- A more informative label system - The current system only classifies the grip into good and bad, but one that separates between a total miss, a marginal grip and a good grip could allow the robot to make better decisions about what to do.
- Sequential classification - performing classification sequentially i.e. first deciding if the grip will miss or not and then decide if is good or bad could result in more separable data fewer false positives.
- Compare to a camera based system - A camera mounted on the gripper is a logical alternative to the sensors used in this work and is worth exploring.

- Reduce the bulk of the TOF and color sensors - The actual sensors are quite small, but the breakout boards used for the prototype are bulky and could interfere with gripper operations. Rearranging the sensors and their support circuits could drastically reduce their foot print.

- Field test the sensors - This work was done indoors, but an actual sensor would need to function in the heat, moisture and dirt of a field environment.

In addition to improving the precision of the results, future work will include implementing the gripper system into the control loop of an automated fruit picker. This will allow training of the system on real fruit in a real occluded environment. A real environment will also provide a more accurate representation of foreign objects than just a single leaf. It will also allow classification of success or failure to be based on the real results of the picking action. Once it is working on a real system, work will be done to not only help the robot identify when it is ready to pick a fruit, but to adjust the gripper position to improve the picking success rate.

References

[1] "Intel RealSense Depth Camera D455," 2020. [Online]. Available: https://www.intelrealsense.com/depth-camera-d455/

[2] Y.-L. Park, B.-r. Chen, and R. J. Wood, "Soft artificial skin with multi-modal sensing capability using embedded liquid conductors," in *SENSORS, 2011 IEEE*, 2013.

[3] "4 Inch FlexiForce 0-1 lbs. Resistive Force Sensor," 2020. [Online]. Available: https://www.trossenrobotics.com/flexiforce-1lb-resistive-force-sensor-4inch.aspx

[4] "Round Force-Sensitive Resistor (FSR) - Interlink 402," 2020. [Online]. Available: https://www.adafruit.com/product/166

[5] D. Guo, P. Lancaster, L. T. Jiang, F. Sun, and J. R. Smith, "Transmissive optical pretouch sensing for robotic grasping," in *IEEE International Conference on Intelligent Robots and Systems*, 2015.

[6] L. T. Jiang and J. R. Smith, "Seashell effect pretouch sensing for robotic grasping," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012.

[7] T. Schlegl, M. Neumayer, S. Muhlbacher-Karrer, and H. Zangl, "A pretouch sensing system for a robot grasper using magnetic and capacitive sensors," *IEEE Transactions on Instrumentation and Measurement*, 2013.

[8] Festo, "BionicTripod with FinGripper," 2009. [Online]. Available: https://www.festo.com/rep/en_corp/assets/pdf/Tripod_en.pdf

[9] H. Wang, M. Totaro, and L. Beccai, "Toward Perceptive Soft Robots: Progress and Challenges," *Advanced Science*, vol. 5, no. 9, p. 1800541, 9 2018. [Online]. Available: http://doi.wiley.com/10.1002/advs.201800541

[10] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Sensors and systems for fruit detection and localization: A review," *Computers and Electronics in Agriculture*, vol. 116, pp. 8–19, 8 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169915001581

[11] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, "Robotic tactile perception of object properties: A review," 2017.

[12] H. Zhao, K. O'Brien, S. Li, and R. F. Shepherd, "Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides," *Science*

Robotics, vol. 1, no. 1, p. eaai7529, 2016. [Online]. Available: http://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aai7529

[13] N. Kuppuswamy, A. Castro, C. Phillips-Grafflin, A. Alspach, and R. Tedrake, "Fast model-based contact patch and pose estimation for highly deformable dense-geometry tactile sensors," *IEEE Robotics and Automation Letters*, 2020.

[14] J. Zimmer, T. Hellebrekers, T. Asfour, C. Majidi, and O. Kroemer, "Predicting Grasp Success with a Soft Sensing Skin and Shape-Memory Actuated Gripper," in *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 7120–7127.

[15] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Haptic identification of objects using a modular soft robotic gripper," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9 2015, pp. 1698–1705. [Online]. Available: http://ieeexplore.ieee.org/document/7353596/

[16] L. Chin, J. Lipton, M. C. Yuen, R. Kramer-Bottiglio, and D. Rus, "Automated recycling separation enabled by soft robotic material classification," in *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, 2019.

[17] R. Kumar, U. Mehta, and P. Chand, "A Low Cost Linear Force Feedback Control System for a Two-fingered Parallel Configuration Gripper," in *Procedia Computer Science*, 2017.

[18] J. Becedas, I. Payo, and V. Feliu, "Two-flexible-fingers gripper force feedback control system for its application as end effector on a 6-DOF manipulator," *IEEE Transactions on Robotics*, 2011.

[19] M. R. Motamedi, J. B. Chossat, J. P. Roberge, and V. Duchaine, "Haptic feedback for improved robotic arm control during simple grasp, slippage, and contact detection tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2016.

[20] J. M. Walker, N. Zemiti, P. Poignet, and A. M. Okamura, "Holdable Haptic Device for 4-DOF Motion Guidance," in *2019 IEEE World Haptics Conference, WHC 2019*, 2019.

[21] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," in *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications, IS 2018 - Proceedings*, 2018.

[22] D. Ball, B. Upcroft, G. Wyeth, P. Corke, A. English, P. Ross, T. Patten, R. Fitch, S. Sukkarieh, and A. Bate, "Vision-based Obstacle Detection and Navigation for an Agricultural Robot," *Journal of Field Robotics*, 2016.

[23] B. Arad, P. Kurtser, E. Barnea, B. Harel, Y. Edan, and O. Ben-Shahar, "Controlled lighting and illumination-independent target detection for real-time cost-efficient applications. The case study of sweet pepper robotic harvesting," *Sensors (Switzerland)*, 2019.

[24] B. Yang, P. Lancaster, and J. R. Smith, "Pre-touch sensing for sequential manipulation," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[25] J. R. Smith, E. Garcia, R. Wistort, and G. Krishnamoorthy, "Electric field imaging pretouch for robotic graspers," in *IEEE International Conference on Intelligent Robots and Systems*, 2007.

[26] T. Schlegl, S. Muhlbacher-Karrer, M. Neumayer, and H. Zangl, "A GMR based magnetic pretouch sensing system for a robot grasper," in *2012 IEEE I2MTC - International Instrumentation and Measurement Technology Conference, Proceedings*, 2012.

[27] T. Dewi, P. Risma, Y. Oktarina, and S. Muslimin, "Visual Servoing Design and Control for Agriculture Robot; A Review," in *Proceedings of 2018 International Conference on Electrical Engineering and Computer Science, ICECOS 2018*, 2019.

[28] L. Wang, L. Zhang, Y. Duan, and T. Zhang, "Fruit localization for strawberry harvesting robot based on visual servoing," *Nongye Gongcheng Xuebao/Transactions of the Chinese Society of Agricultural Engineering*, 2015.

[29] Y. Xiong, P. J. From, and V. Isler, "Design and Evaluation of a Novel Cable-Driven Gripper with Perception Capabilities for Strawberry Picking Robots," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018.

[30] P. Lancaster, B. Yang, and J. R. Smith, "Improved object pose estimation via deep pre-touch sensing," in *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[31] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, "Universal robotic gripper based on the jamming of granular material," *Proceedings of the National Academy of Sciences of the United States of America*, 2010.

[32] S. Neppalli, B. Jones, W. McMahan, V. Chitrakaran, I. Walker, M. Pritts, M. Csencsits, C. Rahn, and M. Grissom, "OctArm-A soft robotic manipulator," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, p. 2569.

[33] N. K. Uppalapati and G. Krishnan, "Towards Pneumatic Spiral Grippers: Modeling and Design Considerations," *Soft Robotics*, p. soro.2017.0144, 7 2018.

[34] J. Shintake, V. Cacucciolo, D. Floreano, and H. Shea, "Soft Robotic Grippers," *Advanced Materials*, vol. 30, no. 29, p. 1707035, 7 2018. [Online]. Available: http://doi.wiley.com/10.1002/adma.201707035

[35] S. Li, J. J. Stampfli, H. J. Xu, E. Malkin, E. V. Diaz, D. Rus, and R. J. Wood, "A vacuum-driven origami 'magic-ball' soft gripper," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019.

[36] W. Crooks, G. Vukasin, M. O'Sullivan, W. Messner, and C. Rogers, "Fin Ray® Effect Inspired Soft Robotic Gripper: From the RoboSoft Grand Challenge toward Optimization," *Frontiers in Robotics and AI*, vol. 3, 11 2016.

[37] S.-G. Jeong, M. M. Coad, L. H. Blumenschein, M. Luo, U. Mehmood, J. H. Kim, A. M. Okamura, and J.-H. Ryu, "A Tip Mount for Carrying Payloads using Soft Growing Robots," *arXiv Computer Science*, 2019.

[38] Z. Gong, B. Chen, J. Liu, X. Fang, Z. Liu, T. Wang, and L. Wen, "An opposite-bending-and-extension soft robotic manipulator for delicate grasping in shallow water," *Frontiers Robotics AI*, 2019.

[39] N. K. Uppalapati, B. Walt, A. Havens, A. Mahdian, G. Chowdhary, and G. Krishnan, "A Berry Picking Robot With A Hybrid Soft-Rigid Arm: Design and Task Space Control," in *Robotics: Science and Systems Foundation*, 2020.

[40] J. Bishop-Moser, G. Krishnan, C. Kim, and S. Kota, "Design of soft robotic actuators using fluid-filled fiber-reinforced elastomeric enclosures in parallel combinations," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 4264–4269.

[41] N. K. Uppalapati and G. Krishnan, "Design of Soft Continuum Manipulators Using Parallel Asymmetric Combination of Fiber Reinforced Elastomers," 8 2018.

[42] N. K. Uppalapati, G. Singh, and G. Krishnan, "Parameter estimation and modeling of a pneumatic continuum manipulator with asymmetric building blocks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 4 2018, pp. 528–533.

[43] N. K. Uppalapati and G. Krishnan, "VaLeNS: Design of a Novel Variable Length Nested Soft Arm," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1135–1142, 4 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8961979/

[44] Festo, "Getting to Grips with the Adaptive Gripper DHDG," 2011. [Online]. Available: https://www.festo.com/net/sv_se/SupportPortal/Details/210356/PressArticle.aspx

[45] G. Wang, Y. Yu, and Q. Feng, "Design of End-effector for Tomato Robotic Harvesting," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 190–193, 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896316315981

[46] EVERLIGHT, "Opto Interrupter ITR20001/T," 2016. [Online]. Available: https://cdn-shop.adafruit.com/product-files/3930/ITR20001-T.pdf

[47] STMicroelectronics, "Proximity and ambient light sensing (ALS) module VL6180X," 2014. [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/037/608/original/VL6180X_datasheet.pdf

[48] Texas Advanced Optoelectronic Solutions, "TCS3472 COLOR LIGHT-TO-DIGITAL CONVERTER with IR FILTER," 2012. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf

[49] "Nearest Neighbors," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html

[50] E. Alpaydın, *Introduction to Machine Learning*, 2nd ed. Cambridge, Massachusetts: The MIT Press, 2010.

[51] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. Cambridge, Massachusetts: The MIT Press, 2018.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and È. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[53] "Decision Trees," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html

[54] "Support Vector Machines," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[55] "Neural network models (supervised)," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[56] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition*, 2nd ed. Heaton Research, Inc., 2008.

[57] "Naive Bayes," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html

[58] H. Zhang, "The Optimality of Naive Bayes," in *Proc. FLAIRS*, 2004.