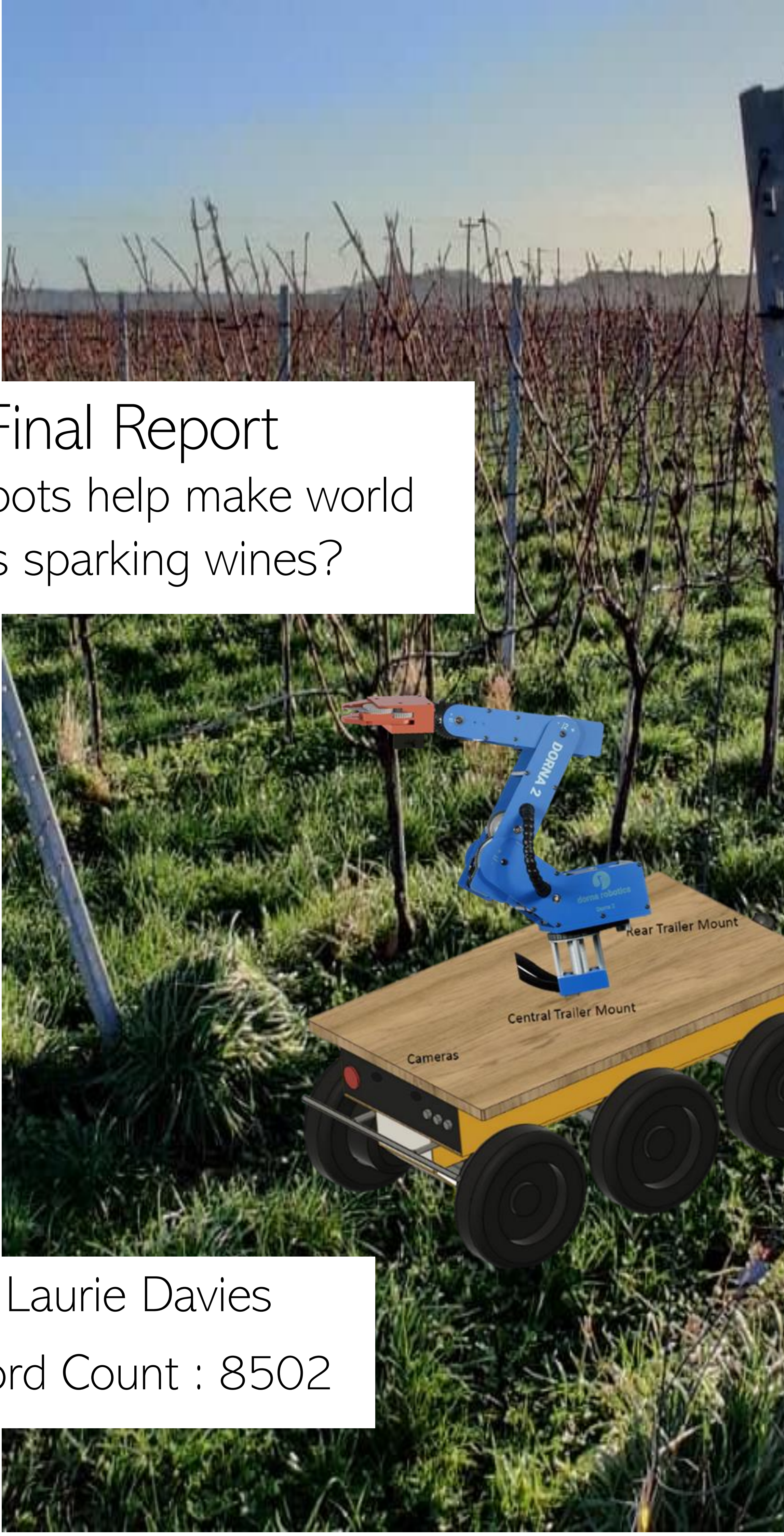


Final Report
Can Robots help make world class sparking wines?

Laurie Davies

Word Count : 8502

Summary

This report details the work undertaken over the last year to produce a prototype of a grape picking robot. Initially two mini-studies were undertaken to explore current solutions and the harvesting process. The research completed in the studies allowed for some feasible early designs and a specification to be made. The project was split into four sections: the rover, the vision system, the robotic arm and the end effector. It was decided that the rover and robotic arm should not be prototyped due to time constraints and that this semester should focus on the development of the vision system and the end effector.

The vision system aims to recognise features of grapes such as colour and shape. This was chosen over a machine learning approach as there was no data set which could be used at the time. A number of different filters and functions were tested individually to find the optimal way of recognising grapes. These included: a chroma-key filter, k-means clustering, contouring and a depth filter. In the final version a combination of these filters were used along with a percentage certainty calculation that was based around the shape of the identified bunch. The system can reliably identify multiple bunches of grapes and mitigates several problems initially outlined such as identifying similarly coloured leaves as grapes.

The end effector grips the stem and then cuts above it so that the bunch does not fall. Two pivot designs were initially prototyped to find the best method for doing this and were based around rotational and linear movement. A rotational movement design approach was chosen as it allowed for a broader design space and for higher force designs to be made. Several stages of prototyping were completed with the final design using an Arduino and two servo motors connected to moving arms which met static arms to complete the gripping and cutting processes. A gearing mechanism was included to increase the torque of the cutting blade so it could reliably cut through the stems. Other features such as knurled gripping surfaces were also added to improve its functionality.

The final prototypes prove that designing and building a complete grape cutting robot would be possible. Both systems can be improved by adding functionality such as including a second depth filter in the vision system and using more powerful motors on the end effector. Moving forward the project should aim to integrate the current systems by introducing a robotic arm. This should be done in time for harvest season in October so that more realistic testing can be done.

Table of Contents

Table of Figures

Introduction

Farmers across the world have struggled in the last few years to find enough workers to harvest their crops, partly due to the physical demands of the job and low wages. Prior to Brexit foreign nationals made up 98% of the fruit picking workforce, but when the UK left the European Union at the start of 2020 it became much harder for foreign nationals to obtain work visas so that they could work as fruit pickers here [1]. A number of schemes were introduced to try to solve this problem such as the Seasonal Workers' Visa Scheme which allowed farmers to hire workers (initially 10,000 nationally but increased to 30,000 after 1 year) from anywhere in the world [2]. This scheme was mildly successful but did not solve the problem completely as many Europeans who had previously worked in the UK could now get better paid or less labour-intensive jobs in their own country. This meant farmers had to look further afield and therefore had to spend more money on transporting their workforce from more distant countries. The demand for fruit-pickers has been further amplified by the COVID-19 pandemic over the last two years. International travel has been made almost impossible at times and this has further reduced the number of fruit pickers available to farm owners. Pick for Britain was created by the government to try and solve this problem and aimed to incentivise UK nationals to start working on farms. Due to many nationals being furloughed or made redundant there were initially a large number of applicants. However due to the long arduous hours and many of the new workers needing training and supervision [3] this scheme only lasted a year and was scrapped in April 2021 [4]. Even if it had been a success, it would have been only a short term solution as many nationals have now been taken off furlough and returned to their normal jobs.

In order to address the workforce problem several fruit picking robots have already been developed and deployed across the country. These range from simple aids for labourers to fully automated fruit pickers such as the Agrobot strawberry picker [5]. However, there is not currently a non-destructive robot being sold commercially that can be used on vineyards to pick grapes. This project will aim to discover if this is possible and how it could be achieved.

The project focused of on two of the sub-systems of a complete robot picker: the vision system and end effector. For prototyping an Intel RealSense d435i stereo camera was used in conjunction with Python to create a vision system to identify the grapes. The end effector was prototyped using an Arduino and a pair of Dynamixel motors along with a number of 3D printed parts.

Background

Over the first semester, two mini-studies were completed, a number of early prototypes were theorised, and an early specification was drawn up. Mini-study 1 was centred around a patent study into current grape pickers as at the time none were being sold commercially. Some other robotic fruit pickers were also researched to find parts or processes that could be used in the design of a grape picker. Mini-study 2 focused on finding out more about the current harvesting process. It also took a brief look at potential vision system solutions.

Grape Harvesting Process

Grapes are quite an unusual fruit to harvest as they are picked in bunches and from low height (around one metre). Other fruits are generally picked one by one from a tree or the ground. It is essential that the grapes are not damaged before they are pressed as the initial juice obtained makes the best wine and is therefore the most valuable. The harvesting process completed by humans is relatively simple with fruit pickers moving down the rows cutting bunches of grapes and dropping them into nearby buckets. These buckets (yellow) are collected once they are full, and the grapes are transferred into larger crates ready to be transported to the pressing site.

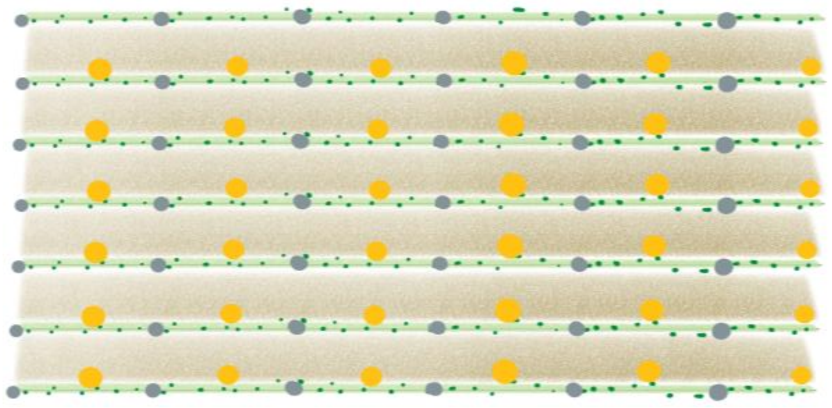


Figure 1 - Diagram of a Vineyard

There is around a two-week period each year in which the grapes are ripe and should be picked and approximately 10 workers (calculated as an average from the vineyards contacted) are needed per hectare. The two week harvesting period is usually in late September to early October (for white grapes). This means that the design of the robot would need to allow it to operate in several different conditions. The main problem is likely to be with the lighting conditions. The robot would be required to operate for long hours with changing light levels and varying shadows. Therefore, it is important that the vision system can adjust well to these changes.

While visiting the vineyard it became apparent that several other processes occur during the growing season which could also be automated. These ranged from watering and applying fertiliser to testing the ripeness of the grapes. There are already several robots that are capable of watering and fertilising crops, so it was decided to not include this functionality at this stage. Ripeness testing is integral to harvesting grapes at the optimal time, but the current tests require either human judgement or expensive machinery, so this functionality was also not explored.

Existing Solutions

Over the last 5 years there has been a large increase in the number of robots that have been designed to help farmers harvest their crops. Despite this there are currently no grape pickers being sold commercially. However, several patents have been filed for based around 3 distinct designs: rovers, drones and gantries. The picking methods were all relatively similar with most using 1 or 2 blades to cut the stems. The collection methods were also similar with all but one of the designs using a basket or a pincer to grip the stems.

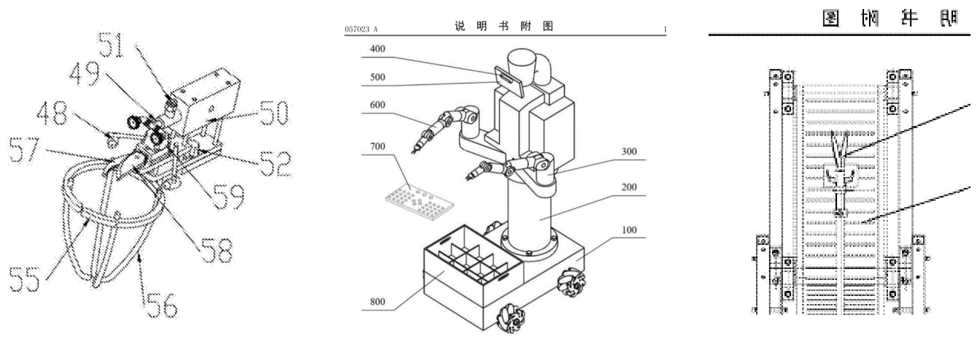


Figure 2 - Existing Gripping and Cutting Patents [6] [7] [8]

The vision systems of the designs were only shown on a few of the patents and all of them used a camera or cameras to achieve this.

Figure 3 - Cameras on Patents [6] [9]

A number of designs were chosen (each with differing features) and an MCDA was completed with a number of other engineering students to find the best design and any desirable features. The outcomes from this were that a rover-based robot would perform best as it would be less complex than the drones and also be able to move autonomously (unlike the gantries which would need to be moved). The robots with high dexterity and a large number of joints in their robotic arms also scored highly as they would allow for grapes to be picked from a large range of angles. Exploring existing patents helped to identify which features would be most useful but none were discounted before some initial designs were drawn up.

As previously mentioned, the vision system will be an integral part of the process. Therefore it was important to select appropriate hardware to complete this project. It was decided that an Intel RealSense d435i stereo camera would be used as it could produce both colour and depth images. While this camera is perfect for prototyping, a higher quality camera could be used in the final product to allow for more precise image analysis.

Project Aims and Initial Specification

The main aim of the project was to produce a proof of principle prototype. The project was split into 4 sub-systems. These were:

- Movement system – How the robot will move around the vineyard.
- Robotic Arm – How the robot will move the grapes from the vine to the storage containers.
- End Effector – How the robot will cut the stems of the bunches and hold them.
- Vision system – How the robot will detect a bunch of grapes.

At the end of semester, it was decided that the movement system would not be prototyped. Instead, the Navvy (a project run at the university by Ed Elias) would be used for some initial data and for prototyping if the project reached that stage. After re-assessing the required work at the start of semester 2 and given the time and financial constraints it was decided that the development of the robotic arm was not feasible, although some initial designs were drawn up.

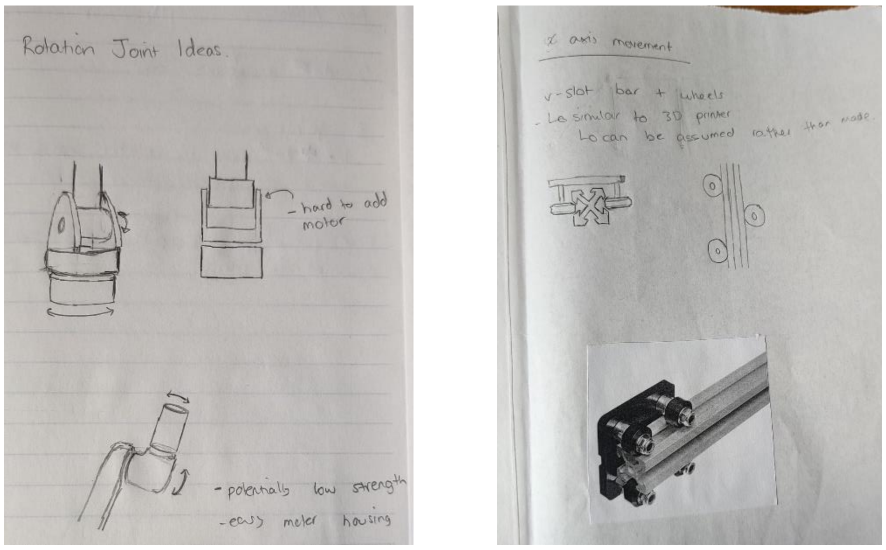


Figure 4 - Easy Arm Designs

There are many robotic arms on the market at present which could have been used if required and several that have been developed with farming in mind such as the DORNA [10].

Figure 5 - Dorna Robotic Arm [10]

This project therefore focused on the vision system and end effector. These were the most interesting and potentially innovative parts of the project as there are no current versions of each. Removing the design of the robotic arm meant that integration between the end effector and vision system would be difficult to achieve.

Project Plan Review

At the end of semester 1 a proposed timeline was drawn up which included the design of a robotic arm. At the start of semester 2 when it was decided that the robotic arm would not be included in the prototype a new timeline was made. This timeline also removed the integration phase as this would not be possible.

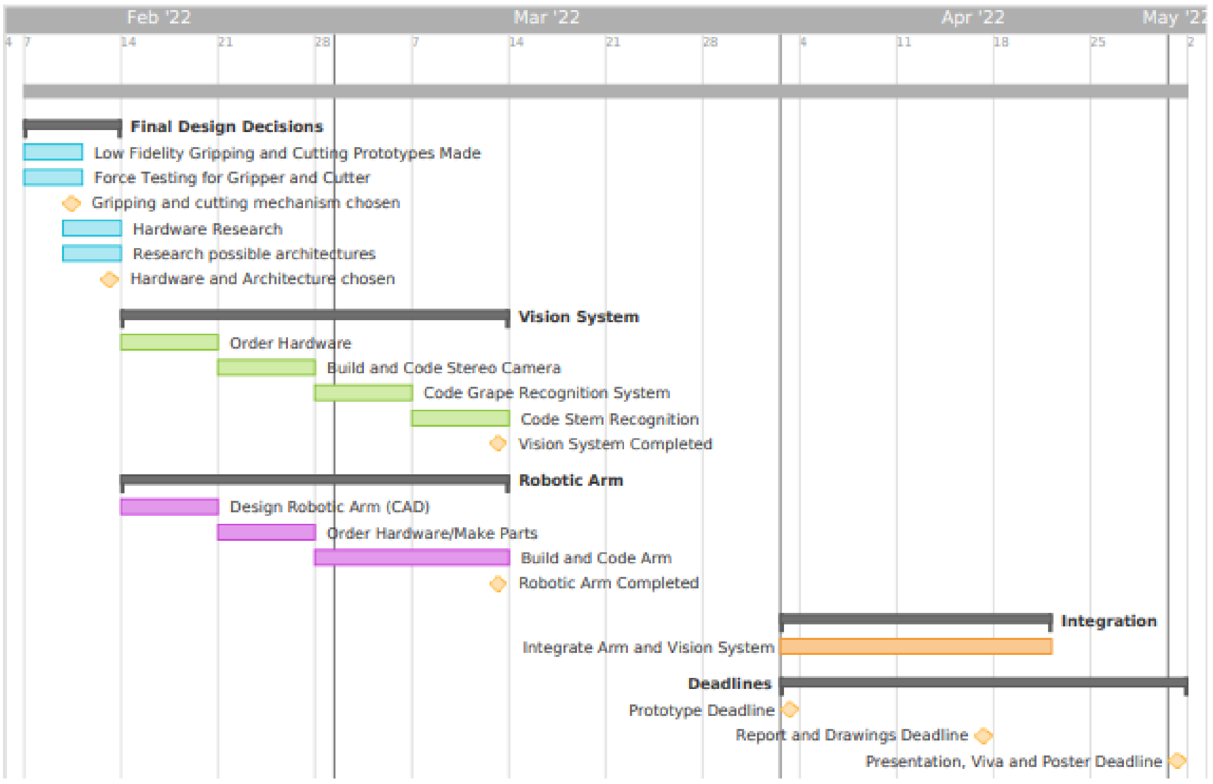


Figure 6 - Original Semester Plan

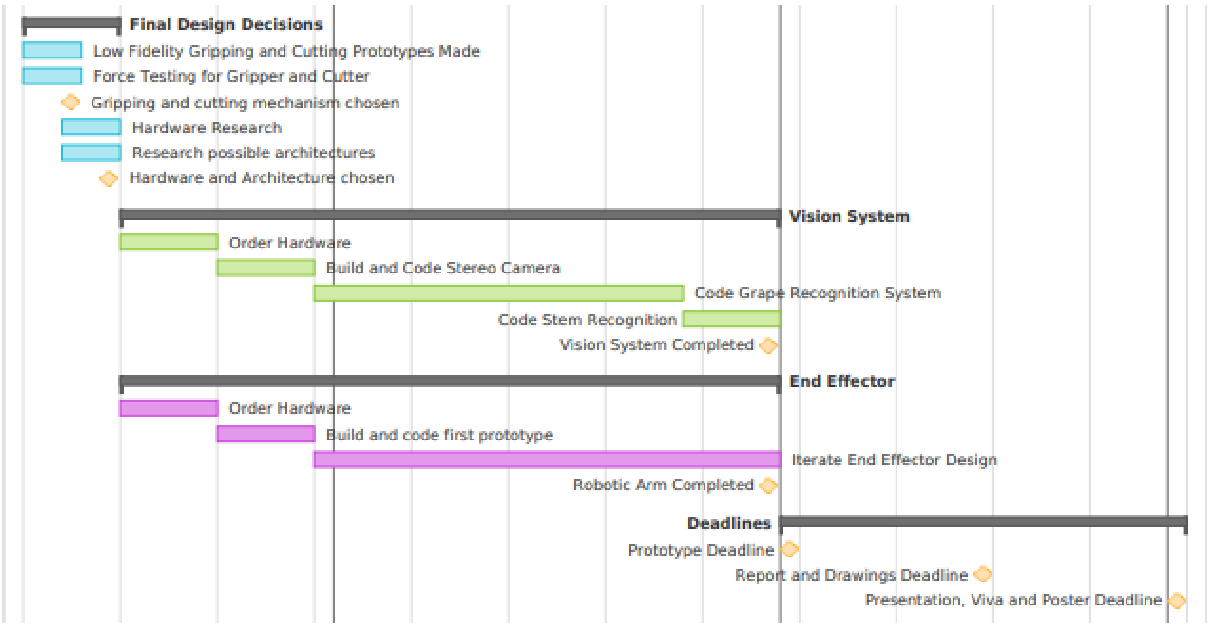


Figure 7 - Updated Semester Plan

Design Architecture

Vision System

As previously mentioned, an Intel RealSense d435i stereo camera would be used for the project and the code would be written in Python. There was already a library for the camera making prototyping easy. The code was run through a laptop. Other methods, such as using a Raspberry Pi were researched but it was difficult to know if the Pi would have enough processing power for the colour and depth image processing. As this was a proof of principle project it was decided that this functionality would not be needed at this stage.

End Effector

The end effector would need to grip the stem of the grapes and also cut above the gripper in order to remove the grapes from the vine. Two motors were used to do this, one for each process. Dynamixel motors [11] were chosen as they included a large number of features which would be useful during the project. For example, the motors could be run in position mode where they would turn to the input angle. The motors could also give feedback on the current usage and therefore the torque. This was useful for measuring both the gripping and cutting power of the arms and could provide some useful checks during the end effector's operation. The motors were controlled using an Arduino Nano and a shield which was designed for the motors. This made setup and operation very easy and allowed for more prototyping time.

System Operation

Although the entire system was not being designed it was important to have a clear idea of how each sub-system would interact with one another and how the robot would operate. The robot would use a rover (the NAVVY or similar) to traverse around the vineyard and once the vision system had detected a bunch of grapes the arm would move into position allowing the end effector to grip and then cut the grapes which could then be moved to a storage container.

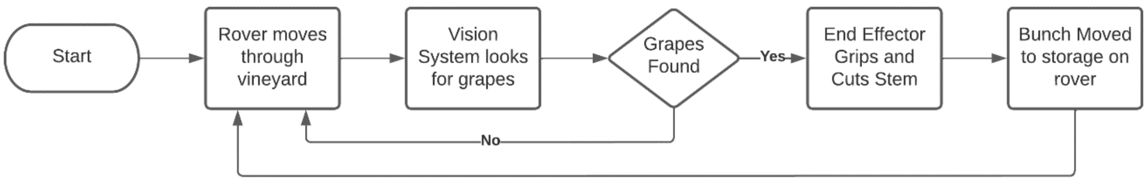
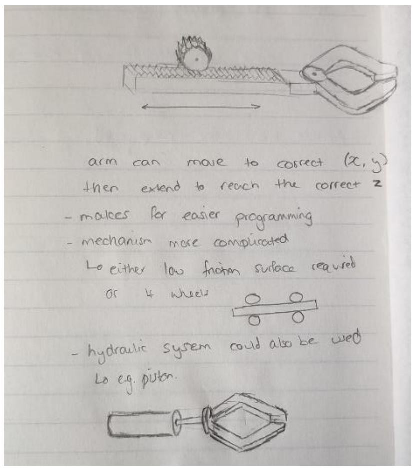

Start
Rover moves through vineyard
Vision System looks for grapes
Grapes Found
Yes
No
End Effector Grips and Cuts Stem
Bunch Moved to storage on rover

Figure 8 - Flow chart of system operation

The end effector was designed so it could feasibly be attached to any robotic arm but for the purposes of the project it was assumed it followed the movement pattern. The robot arm would move to a position close to the stem and then extend the end effector so the gripping and cutting mechanism were around the stem. This method was chosen to reduce the invasiveness of the arm into the vines to minimise the chance of damaging any grapes. Several designs for doing this were developed initially before the robotic arm was removed from the project scope.



arm can move to correct (x, y)
then extend to reach the correct z
- makes for easier programming
- mechanism more complicated
  ↳ either low friction surface required
    or 4 wheels
- hydraulic sysem could also be used
  ↳ e.g. piston.

Figure 9 - Initial Designs for End Effector Extension

Vision System

Approach

The requirements for the vision system were that it must be able to detect a bunch of grapes accurately and identify where the stem is so the end effector can move to the correct position. This was done by estimating an area at the top of the bunch in the centre where it was suspected the stem would be.

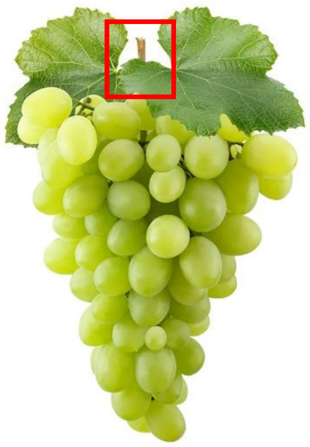


Figure 10 - Area where grape stem is assumed to be

Potential Problems

There are several features of vineyards that make detecting a single bunch of grapes difficult.

Firstly, vines often have large leaves that can be similar in colour and size to bunches of grapes. The leaves often do not completely face the camera and can be discounted that way, but this is not a certainty.

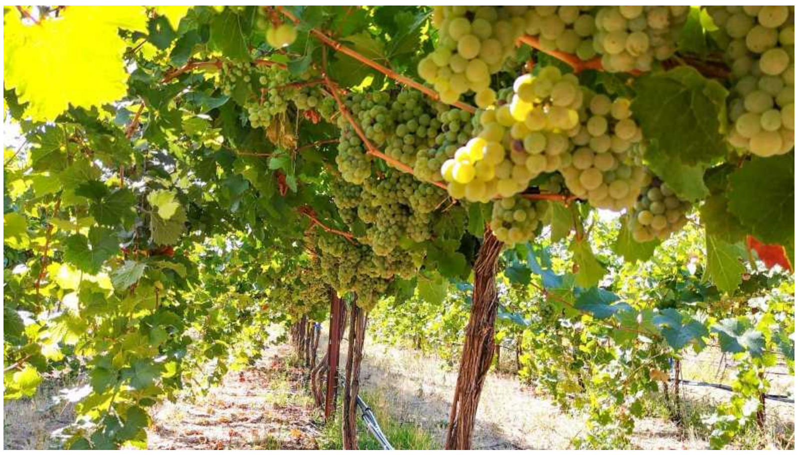


Figure 11 - Standard Vineyard with grape and leaves of similar colours

While picking the occasional leaf is not detrimental in terms of wine production (they would be removed with the stems at the press site), it is important that as little time as possible is wasted as the grapes must be cut with in a set timeframe.

Secondly, other bunches in the background can often be included in what the system thinks is one bunch of grapes.

Figure 12 shows what the system would ideally pick out as a bunch of grapes and figure 13 shows what could be picked out if some functionality is not added to mitigate it. This fault would have a knock-on effect for the system locating the stem of the bunch and would prevent it from being picked.

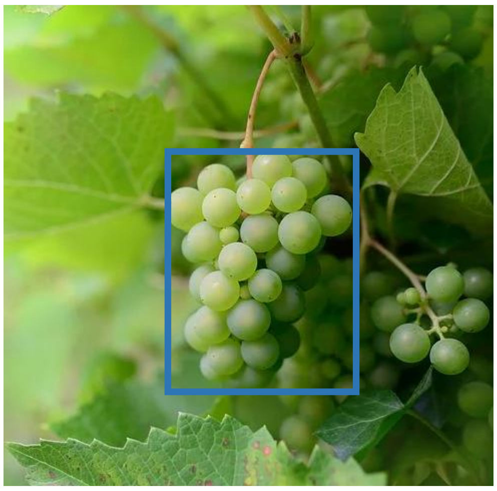


Figure 13 – Ideal Grape Recognition

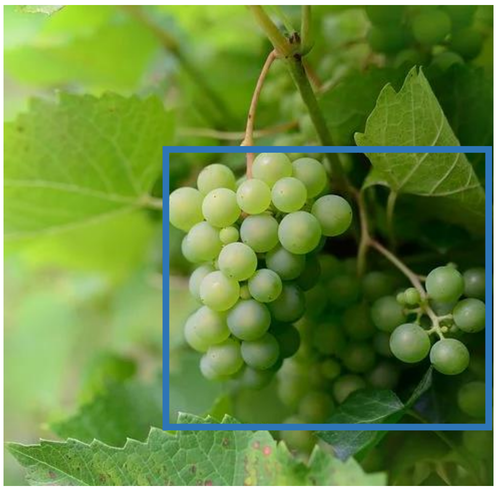


Figure 12 - Potential Problem with overlapping grapes

Finally, lighting and shadows can have a marked effect on the system being able to identify different features of the grapes. This could present as only part of the grape being recognised so that the stem is not located, or the bunch not being recognised at all.

Vision System Type

Initially two different approaches were considered to base the vision system around: machine learning and feature recognition.

Machine Learning

Machine learning aims to teach the system over time what a bunch of grapes looks like. As the system sees more and more grapes it becomes better at successfully recognising them. This method would be useful as initially obvious grape bunches could be used and as the system improves, harder images (that may include other grapes and shadows) could be used to further increase its accuracy. This method was not chosen however, as it requires a large dataset for the system to learn from. There were no datasets online that could be used other than google images which would be difficult to use as the lighting changed dramatically between many of the pictures. Similarly, a custom dataset could not be made as I started developing the system in January when the grapes had not had long enough to grow since harvest season. It was also thought that this method would require more processing power than simple feature recognition which is an important feature on a battery powered robot. [12]

Feature Recognition

Feature Recognition aims to identify grapes using classic characteristics. These normally revolve around shape and colour. This method was chosen as it could be developed in a small time period and could make good use of the stereo camera.

Testing Methods

Initial testing was done using 3 images of bunches of grapes.

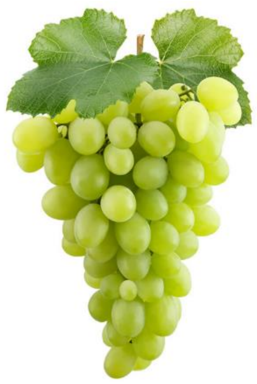


Figure 15 – Simple Bunch of Grapes

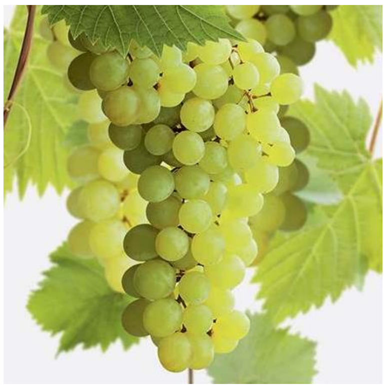


Figure 14 – More Complex Bunch of Grapes

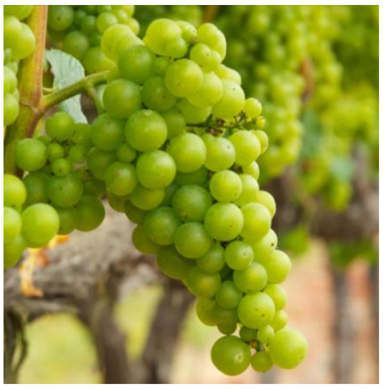


Figure 16 – Highly complex Bunch of Grapes

The first (figure 12) was a simple bunch on its own with a few leaves near the top on a white background. The second (figure 13) is also against a white background but has a few more leaves and grapes in the background. The last image (figure 14) is from a vineyard and has large stems, other grapes and grass in the background. These images were chosen as they provided increasing complexity for the system to be tested on.

Once initial testing of a few methods had been completed, the stereo camera was used with a simple mock gantry.

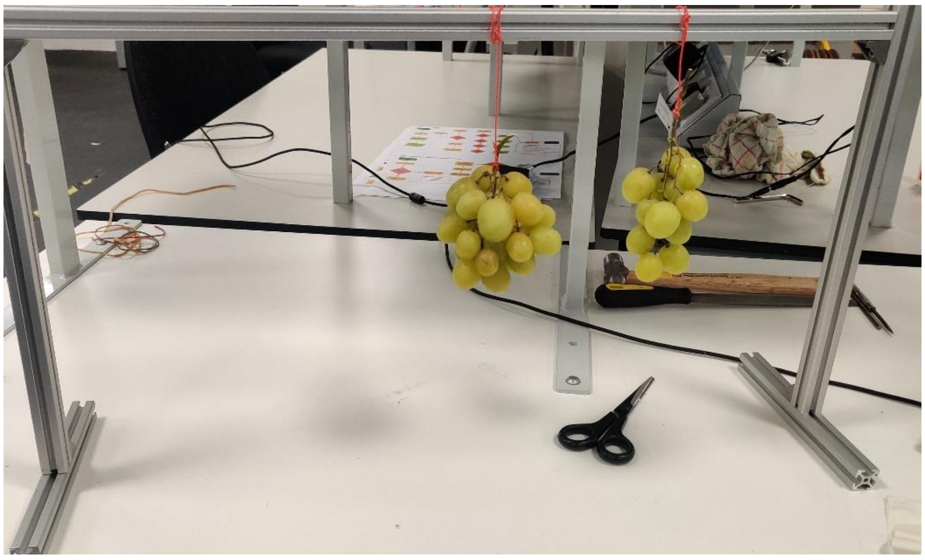


Figure 17 - Mock vineyard used for testing

This setup allowed for a depth filter to be added and also allowed for the system to run in real time on a live video. Supermarkets generally do not sell grapes in whole bunches so smaller bunches were used during testing.

Development

Chroma-Key

A chroma-key filter aims to filter out all colours that are not within a set range of RBG values. This is normally used when working with green screens meaning the background of the subject can be changed to a different image or video. For this project this is reversed in order to keep only the green pixels and replace the other colours with black. Initial results from this filter were good.

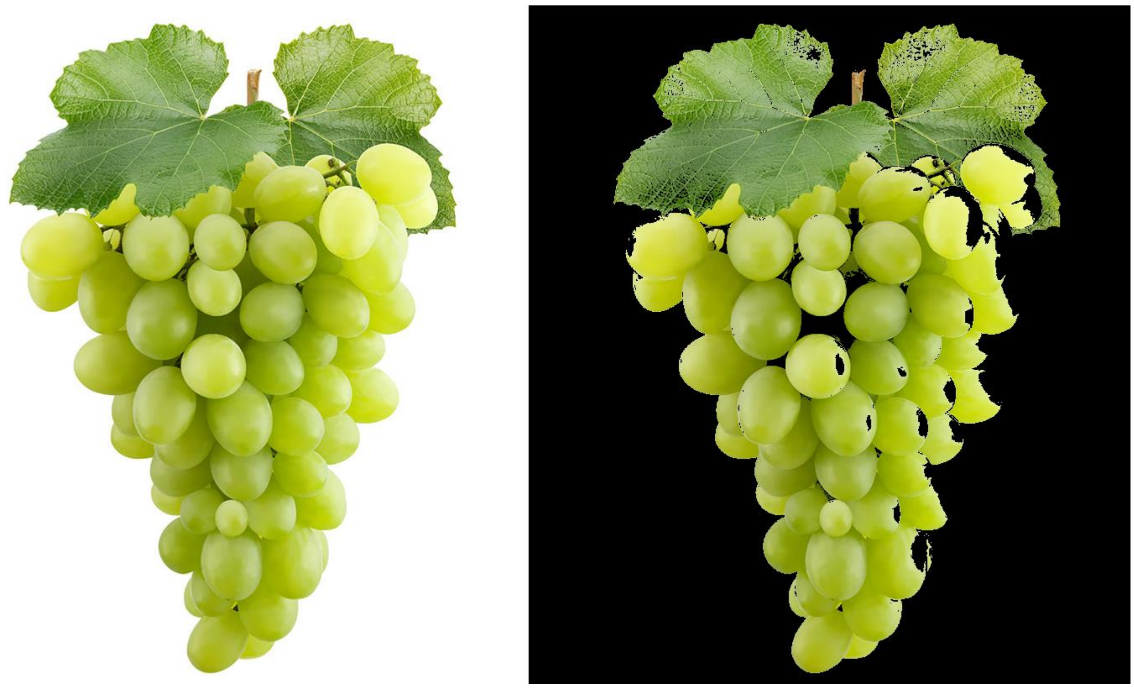


Figure 18 - Chroma-key filter applied to simple image

The images above clearly remove all the white background well. A few of the grapes' edges get cut off where they meet the white background but this is minimal and would not affect the rest of the system. The main problem is that the filter is not removing the leaves of the grapes. In this example it again is not a significant problem as the leaves are not covering many of the grapes. They may also affect the stem prediction area and make it slightly higher.

The other two images were used to test the functionality of this filter. The results were not as good.

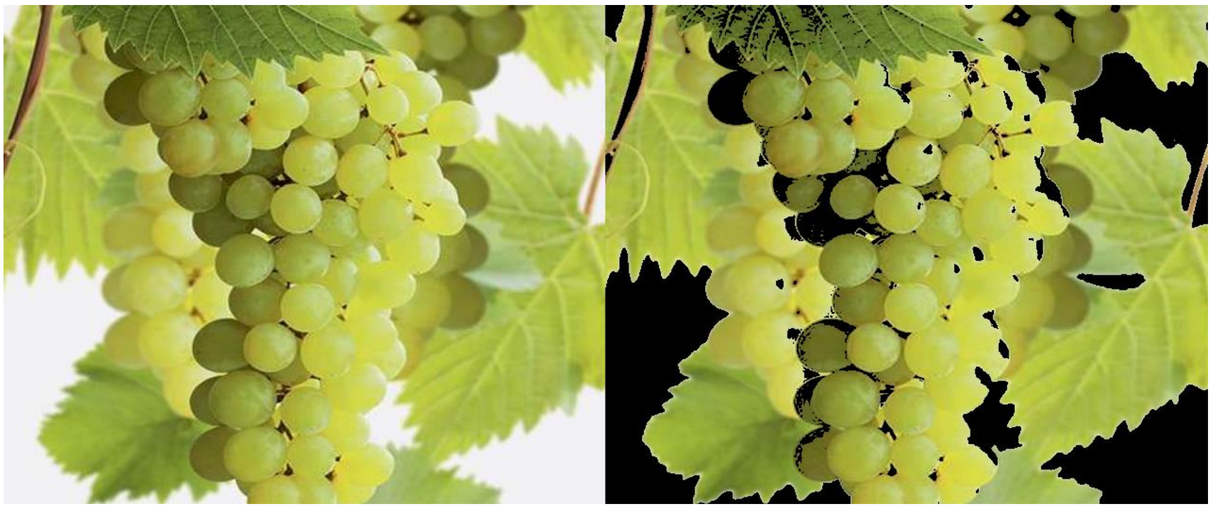


Figure 19 - Chroma-key filter applied to more complex image

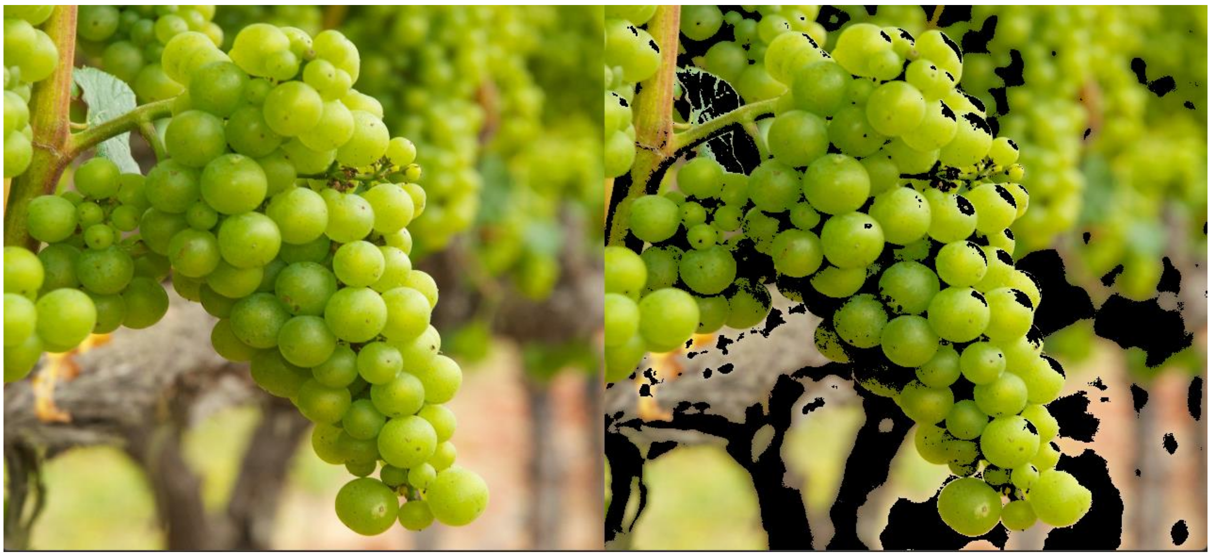


Figure 20 - Chroma-key filter applied to highly complex image

These figures show how lighting and shadows can severely affect the results of using a chroma-key filter. Figure 19 successfully removes the white background but keeps the leaves and grapes behind the main bunch. Figure 20 hardly removed any of the background or the branches or grass behind. The RGB values could be changed to make this better but possibly to the detriment of other images and lighting scenarios. This method showed some success but would not work by itself.

K-means Clustering

K-means clustering is a method of grouping data together based on shared or similar characteristics. It is normally used in machine learning to find similar patterns [13]. However, it can be used to separate images into a set number of clusters (k) based on their RGB values. Clusters containing unneeded parts of the image can then be removed or changed to black. This method also worked well on the simplest grape image and only required 2 clusters to achieve a good result. [14]

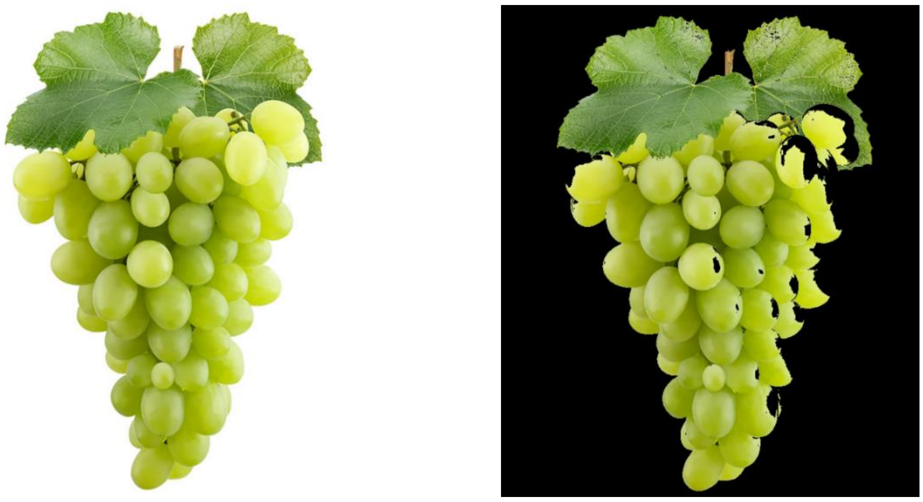


Figure 21 - K-means clustering filter applied to simple image (k=2)

Small parts of some of the grapes were missed but it is still very obviously a bunch of grapes. The leaves could still not be removed however despite increasing the number of clusters up to 5.

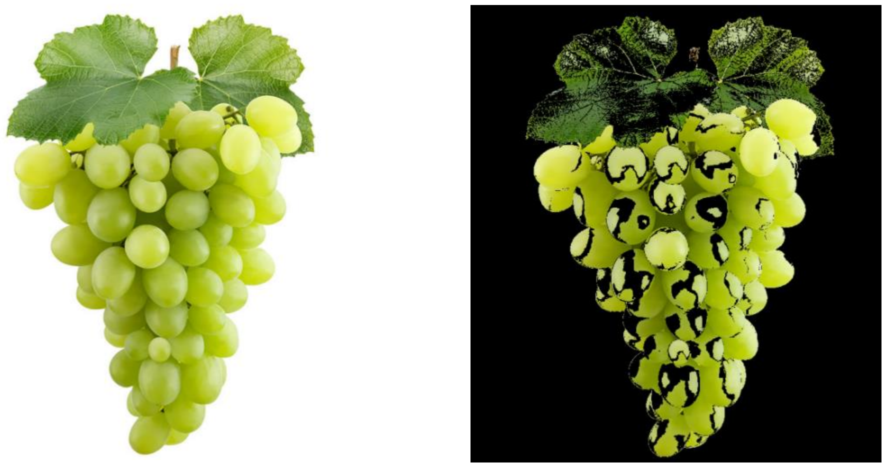


Figure 22 - K-means clustering filter applied to simple image (k=5)

When k-means clustering was applied to a more realistic picture of a bunch of grapes it cut out most of the irrelevant parts of the image but still allowed parts of the branches through.

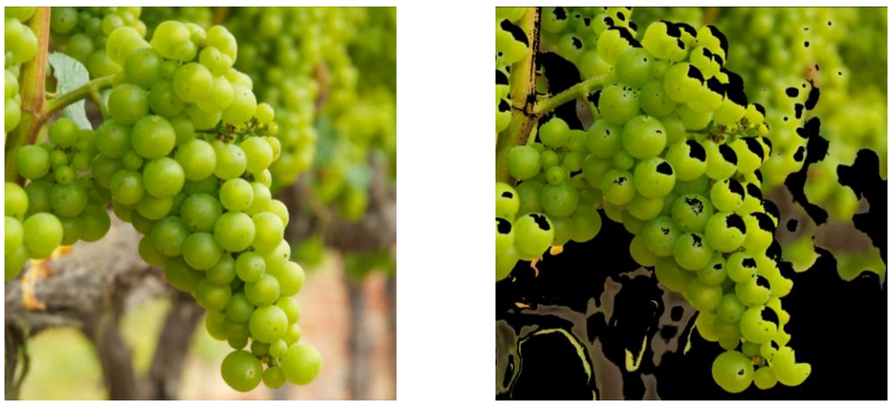


Figure 23 - K-means clustering filter applied to more complex image

K-means clustering generally seemed to produce better results across a range of images than the chroma-key filter. This normally required at least 5 clusters. Initially this method was going to be used in the project. However, the clusters that needed to be included to produce the best image for one picture often needed to be changed manually when using a different picture.

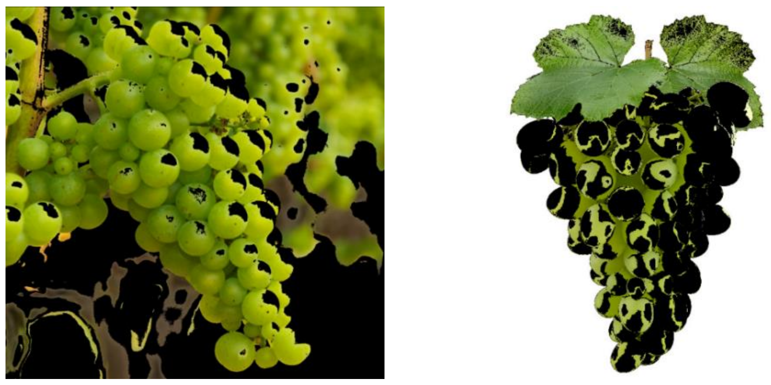


Figure 24 - Same clusters applied to two images

In order to select just the correct clusters, machine learning would need to be added which was not possible.

Contouring

Contouring aims to identify the outlines of objects in an image. This process is normally used to identify shapes based on the number of points identified along the contour curve [15]. However, in this case it can be used to look for edges around the grapes and potentially remove any leaves from around the bunch. The area of the contour can be found and used to remove any contours that are not big enough to be a bunch of grapes in pickable range. The number of points (corners) along the contours can also be found and used to remove any objects that have too many or few points. The standard number of points for the bunches that were being tested was around 10. A rectangle can easily be placed around the contour to identify an object. At the start of this project a smaller rectangle was also added at the top in the centre of the first rectangle which estimated the position of the stem. Ideally later on a more accurate process that actively looks for the stem would be used in preference to this method. This approach may be useful if a leaf is covering the stem.

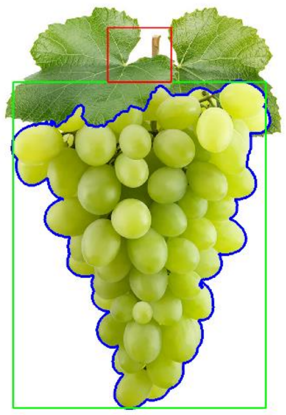


Figure 25 - Contouring applied to simple image

This method worked very well on the simpler grape images and managed to cut out the leaves which the previous methods had not been able to. The stem from the first image also fell inside the estimation box which would make for easy picking. On the final image the system still struggled to identify the grapes correctly but got a lot closer and was an improvement on the previous methods.

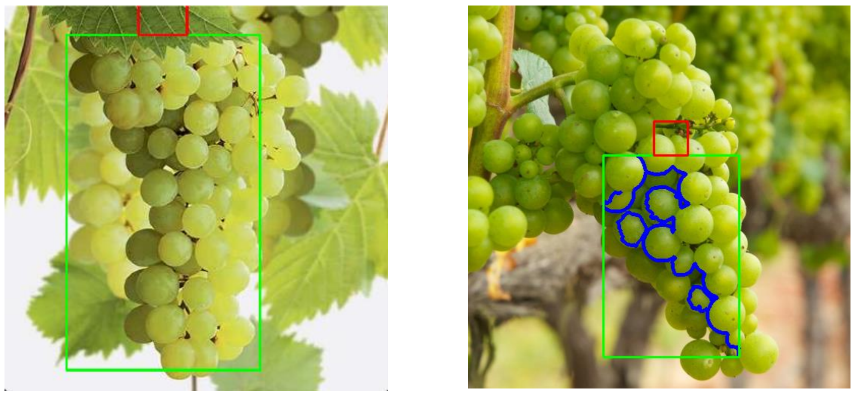


Figure 26 - Contouring applied to more complex images

Depth Filter

The next filter that was tested was a depth filter. This required using the Intel RealSense d435i stereo camera, so a mock vineyard was built to accommodate this (figure 15). Two different depth filters were initially developed and were tested to find the better option.

Background Removal

The first filter simply removed the pixels (turned them black) that were over a certain distance away from the camera.

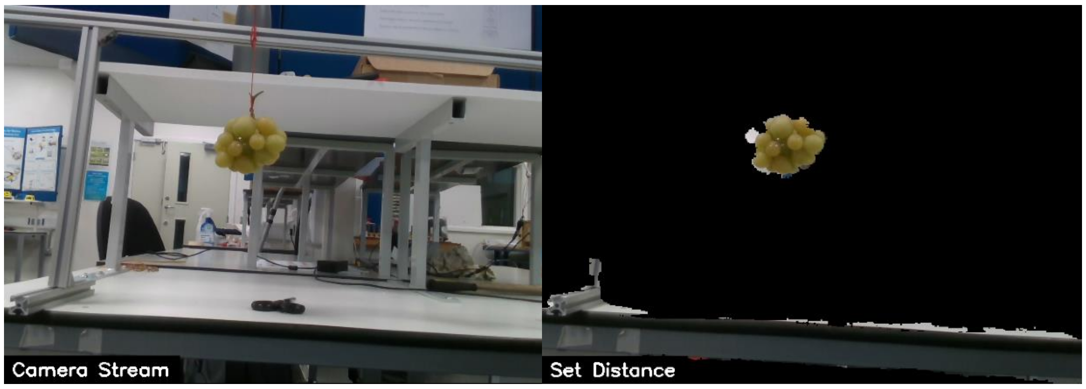


Figure 27 - Background removal at a set distance

This filter worked well and would mean that other objects in the background could be removed before any other filters or image processes were done. This reduces both the chance for error and also the processing power required.

Closest Object

The second implementation found the closest pixel on the screen and removed all the pixels that were not within a set range.

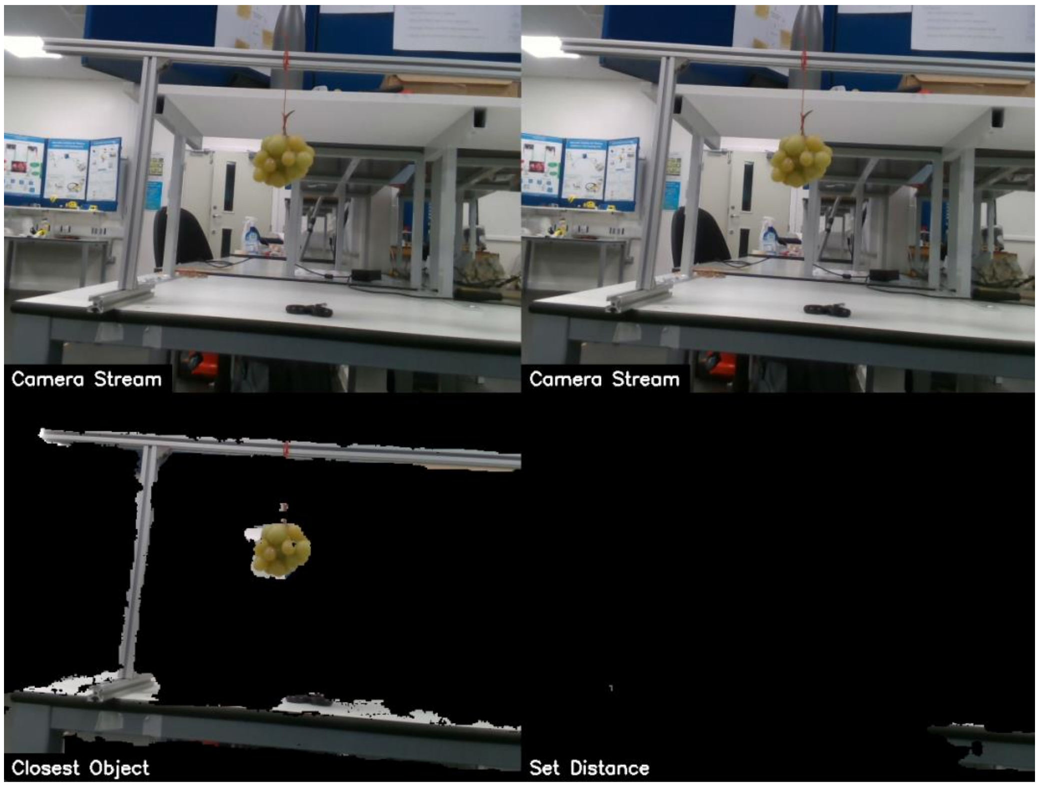


Figure 28 - Background removal at a set distance and based on closest object at large distance

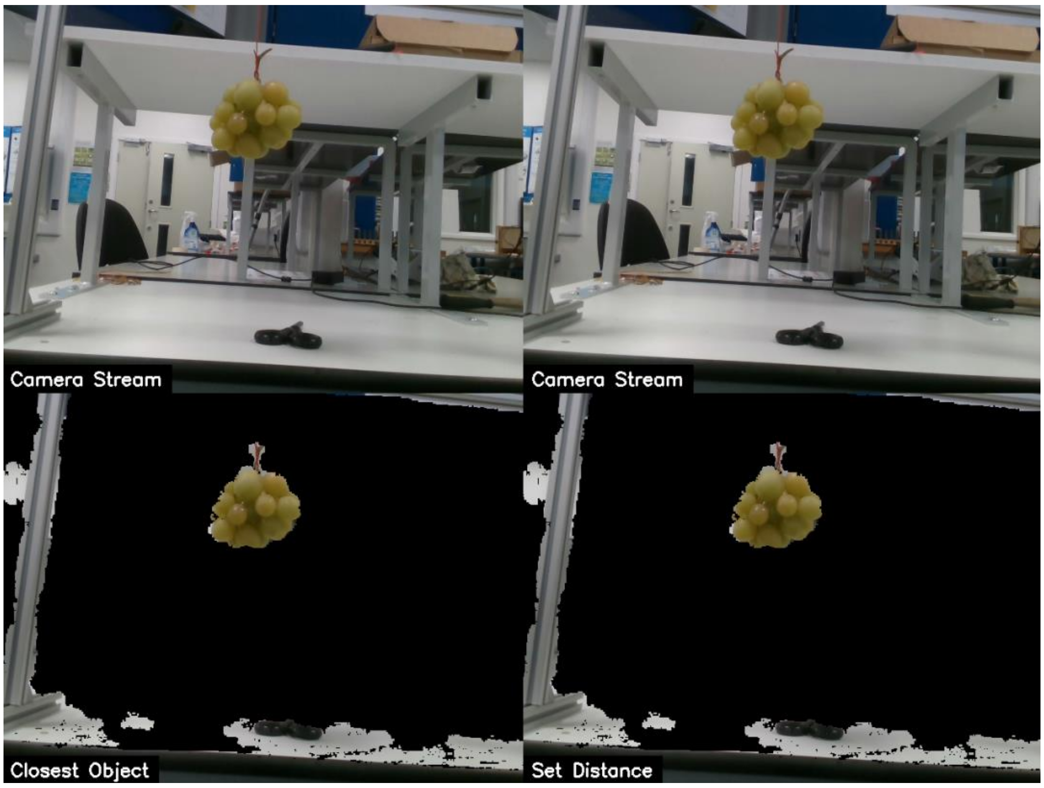


Figure 29 - Background removal at a set distance and based on closest object at small distance

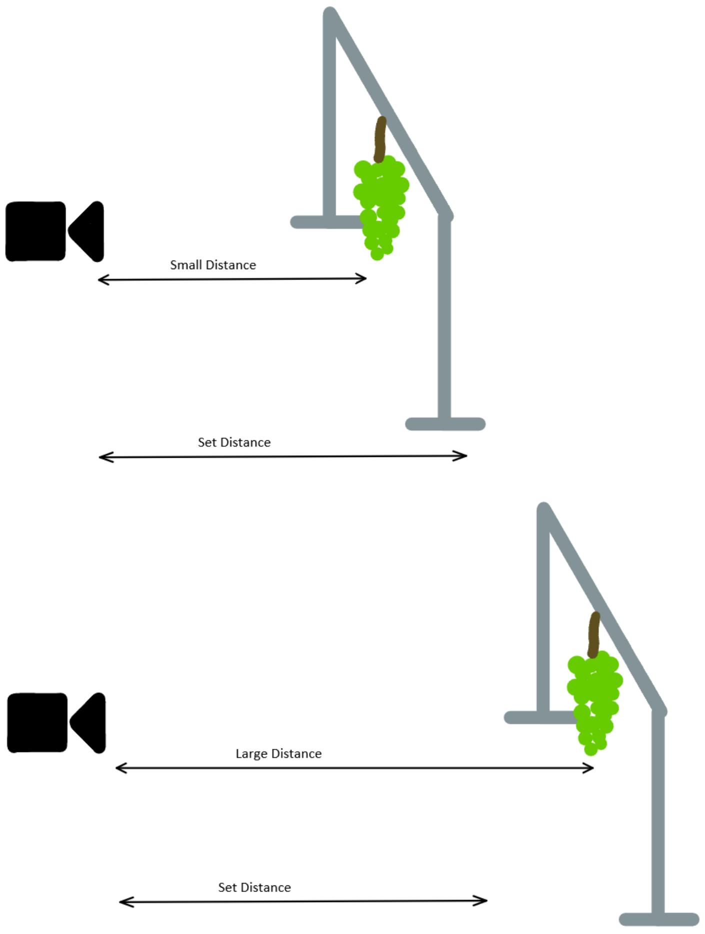


Figure 30 – Testing setup for background removal

While this implementation worked well on the mock vineyard it may not work as well on a real one for to a number of reasons. The biggest potential problem is that it assumes that a bunch of grapes is the closest object. If the closest object was a leaf or part of the trellis then the system would remove the grapes from the background and they would not be picked.

Although both these versions have their advantages, the background removal was selected as the most appropriate for the project.

HSV

While developing the contour system it was noted that converting the RGB image to a Hue, Saturation, Value (HSV) image greatly improved the results. A HSV model image is much closer to how humans perceive colour. The Value input is like a brightness value meaning that it can be changed to reflect the current light levels which is very useful for this project. Applying this model to the chroma-key filter greatly improved its effectiveness and meant it could be used as a reliable filter.

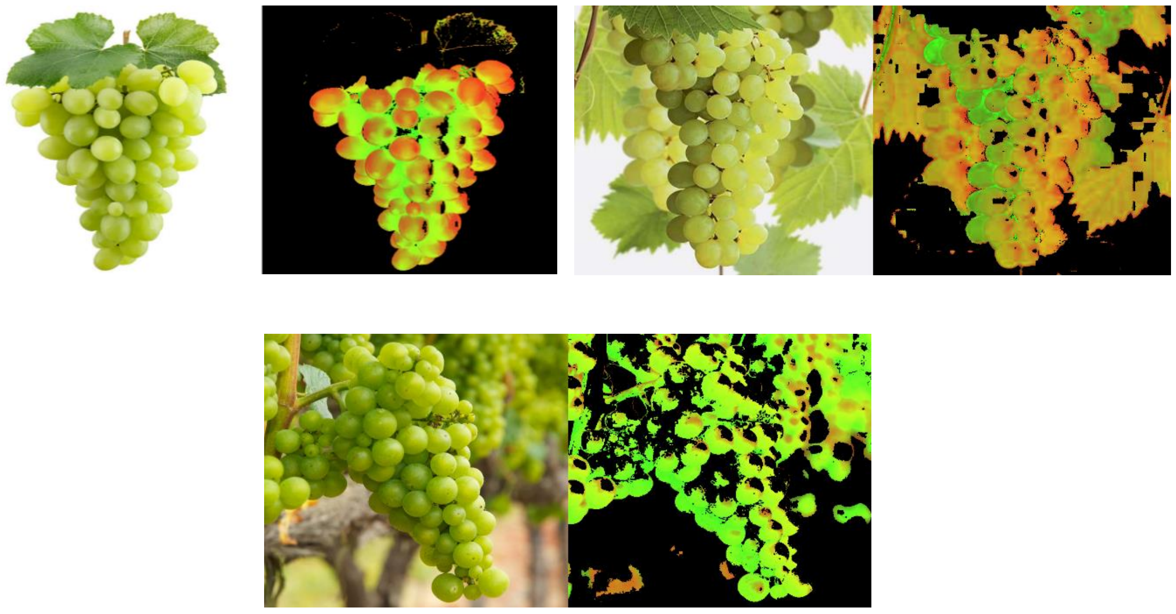


Figure 31 - Test images converted to Hue, Saturation, Value images

Final Prototype

The final prototype uses several of the previously mentioned filters and functions. K-means clustering was not used due to the requirements for machine learning. Initially a depth filter is applied to the image. The background removal approach is used, and the distance can be easily changed to accommodate the robot's environment and set up. The remaining parts of the image are then converted to HSV, and the chroma-key filter is applied. This removes all the remaining objects that are not grapes. Once just grapes are included in the image the contouring function is applied. This performs two functions. Firstly, it makes identifying an estimate for the stem easier and secondly, it can act as a final check to ensure that only grapes are included in the image.

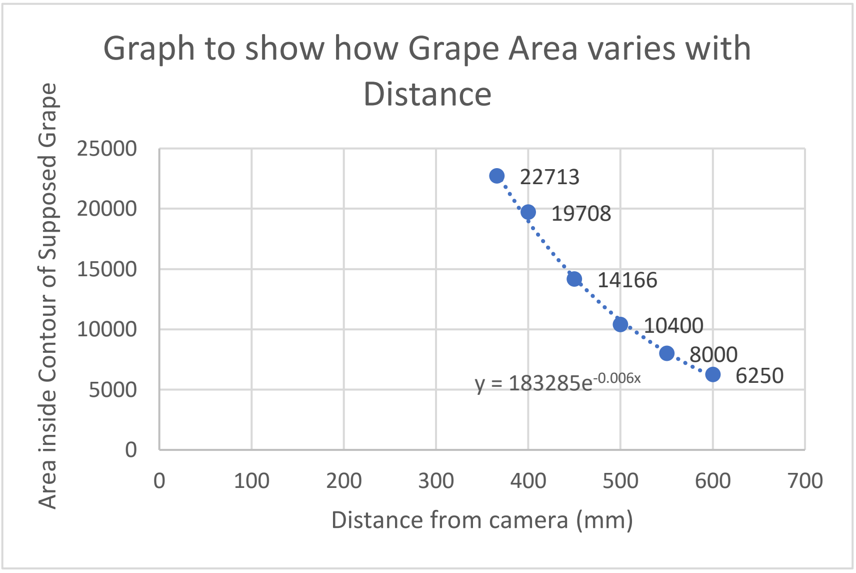
Percentage Certainty

Through testing it was found that occasionally if a leaf is a very similar colour to the grapes it can be allowed through the filters and therefore it was important to add further checks. A percentage certainty calculation was added to perform this task which checks several parameters of the supposed bunch of grapes. The parameters currently used are based on the area and shape of the image.

Area Certainty

The area inside the contour is used along with the distance from the camera to calculate an accurate area of the supposed bunch. Once this has been calculated it is compared to the area of an average bunch of grapes.

The average bunch's area at different distances was calculated by collecting data from a number of the test grapes being used. By testing how the area inside the contours changes with distance an equation could be formed that shows the relationship between them.



The equations below show how a final value for area certainty was calculated.

$$Supposed\ Grape\ Area = Area\ inside\ contor * distance\ from\ camera$$

$$Average\ Grape\ Area = 183285e^{-0.006*distance\ from\ camera}$$

$$Area\ Certainty = 1 - \left|1 - \frac{Average\ Grape\ Area}{Supposed\ Grape\ Area}\right|$$

Shape Certainty

This calculation aims to check that the supposed bunch is the correct shape. This was done by comparing the width and height of the bounding box added by the contour. An 'if' statement was used to check the values and the certainty value was set to 0.5 if any of the limits were exceeded. The code below shows how this was implemented.

```
if (width * 3) < height or height > (width*6):
    shape_certainty = 0.5
```

Percentage Certainty Calculation

The equation below shows how the certainty was calculated where the parameters have a value between 0 and 1.

$$Percentage\ Certainty = 100 * Area\ Certainty * Shape\ Certainty$$

The parameter values are calculated based on how close they are to a realistic bunch of grapes.

Final Outcomes and Testing

Figure 32 shows the different stages of the identification process.

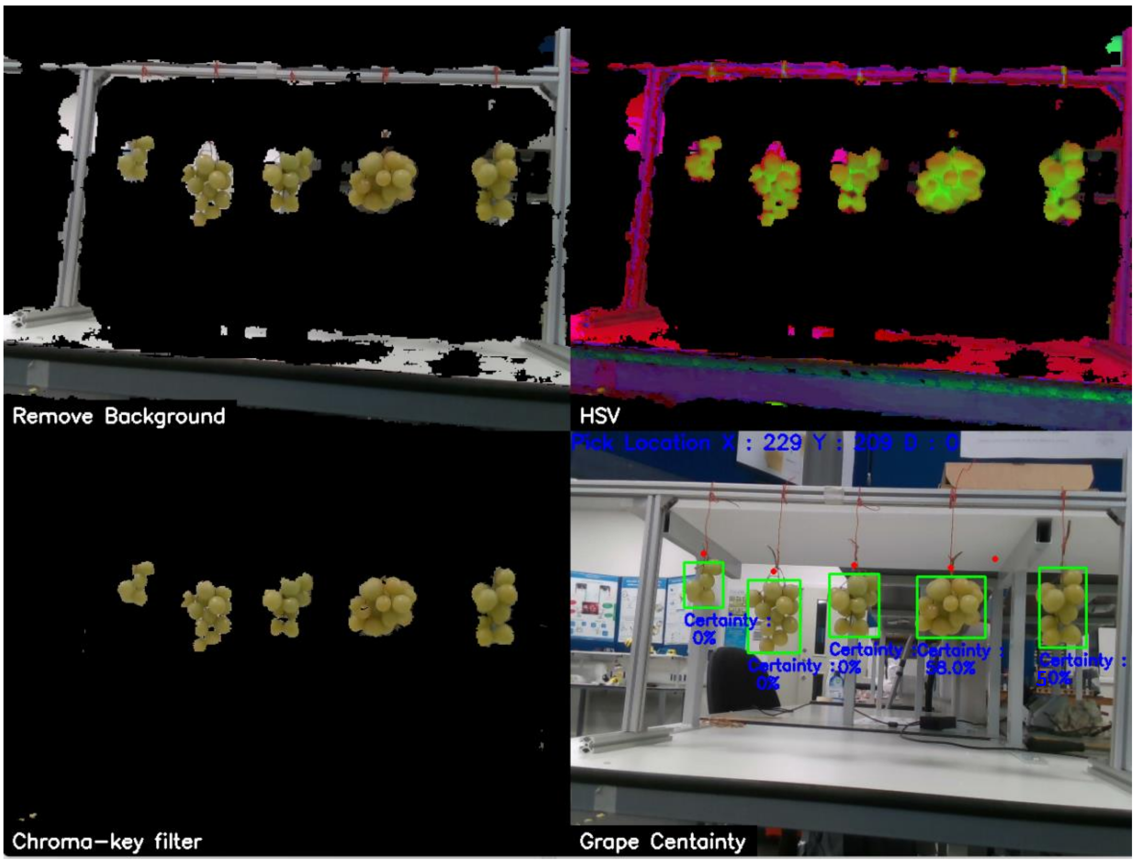


Figure 32 - Final Identification Process

The system can identify and calculate the percentage certainty for up to three bunches of grapes reliably. Once the number is increased beyond this the certainty calculation becomes less reliable. The system is currently running at 30 frames per second (fps) which may account for this and reducing it may rectify the problem. However, up to 5 bunches of grapes could still be recognised even if the percentage certainty is wrong. This number may be larger, but I was unable to fit more bunches of grapes on the mock vineyard I had built to test beyond 5. The system is able to filter out leaves very effectively and currently it has filtered out every leaf that has been tested (around 20).

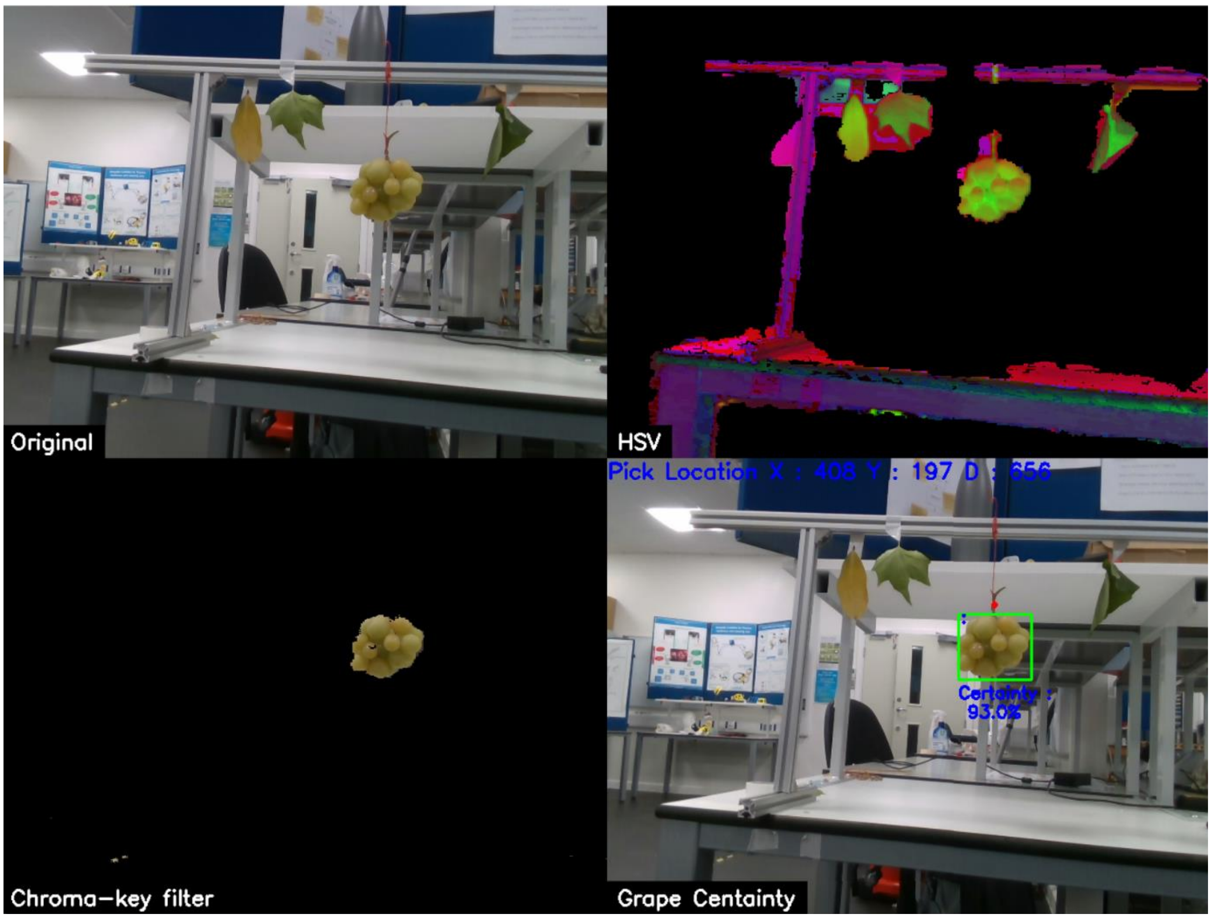


Figure 33 - Leaves being removed from image

Currently the stereo camera has been set up to read distances between 300 and 700mm very accurately. Outside of this range the readings become less accurate. If a reading is recorded outside of this range it is discounted and assumed to be an inaccuracy. This range can be increased but with a loss of accuracy.

Future Work

Improvements

Differentiating between overlapping grapes

One problem that was previously identified but was not rectified during the prototyping phase was when multiple bunches of grapes were behind each other. The system struggled to separate them as shown in figure 34.

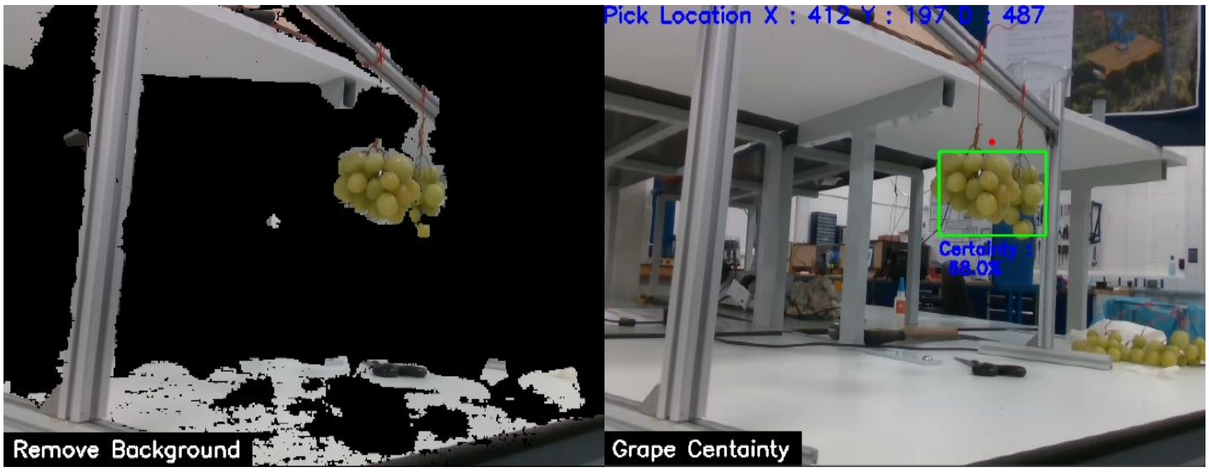


Figure 34 - Two bunches being identified as one

One way of fixing this would be to isolate the grape with the highest percentage certainty and then apply the closest object depth filter that was previously explained. This would filter out the grapes in the background leaving just the closest bunch. The figures below show how this could be achieved although it was not possible to implement this into the final code due to time constraints.

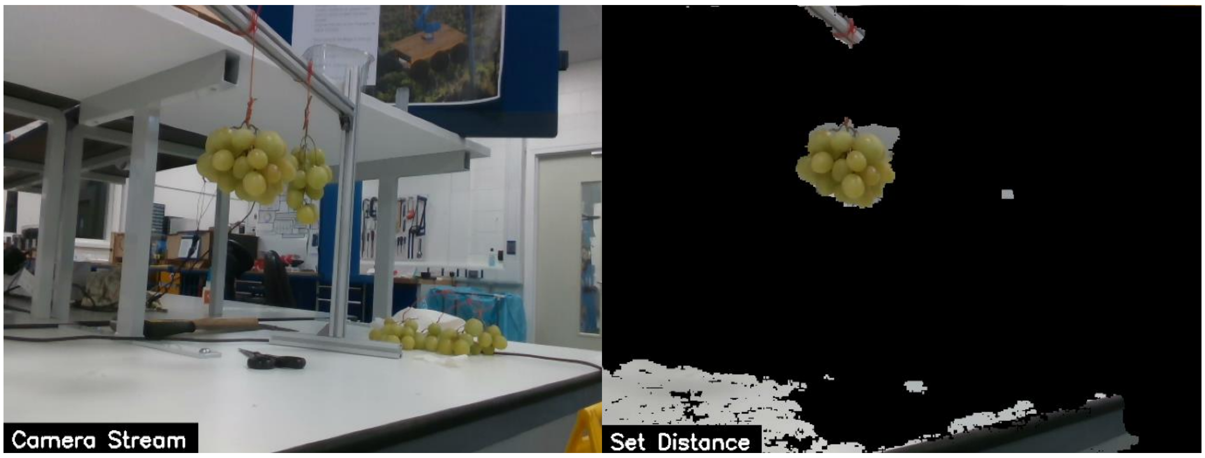


Figure 35 - Closest object background removal being used to isolate only one set of grapes

If the bunches of grapes are very close together then this method may not work but by fine tuning the allowed distance behind the closest pixel it would mitigate most circumstances.

Shape Recognition
Currently the system operates by checking the height and width of the bounding rectangle produced by the contours. However, in practice grape bunches are more triangular. The system could be improved to test against this better shape for example by comparing the top and bottom widths or comparing the angles of the sides to an ideal value.

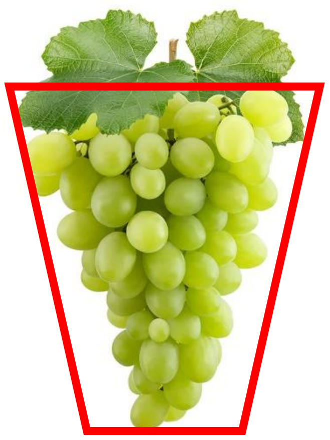


Figure 36 - Shape recognition that is closer to grapes shape

Stem Identification
The current system predicts the stem location based on where the centre of the bunch is. While grape stem location varies very little according to vineyard owners that were interviewed, there is still a chance that the stem could be missed. Adding a feature to recognise the stems of the grapes would be a useful feature. Unfortunately, this is quite difficult as the stems are normally green while they are on the vines. This reduces the chance that the chroma-key filter could be used for this purpose. By repeating some of the filters and functions on just the section of the image around the top of the bunch, the stem may be identifiable more easily. Another approach would be to try and discount areas where the stem could not be. The gripper could also be redesigned to have a large area where it could grip a stem to mitigate the need for a highly accurate set of coordinates for the stem.

Additional Features
As previously mentioned, leaves have been filtered out well in the limited testing performed so far. However, in different lighting conditions where the leaves are close in colour to grapes this may not be the case. One method of checking this again would be to look for individual grapes in a recognised bunch.

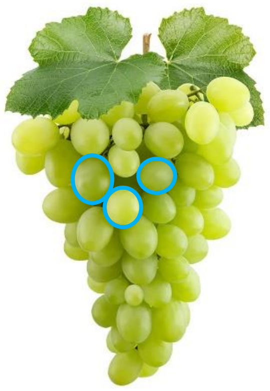


Figure 37 - Contouring being used to find individual grapes

This can be done by altering the method for finding contours. The system uses an external approximation and by using an internal one it could find individual grapes. By looking at the number of points on the contour the shape can be found (0 points for a circle) and therefore identify if it is a grape. Leaves will not have these features and therefore would have a negative score applied to their percentage certainty making them less likely to be picked.

End Effector

Approach

The end effector must be able to grip the stem of a bunch of grapes reliably as it will also be used to move the grapes from the vine to the storage solution on the rover. It must also be able to reliably cut through vines. This process will be done above the gripping method, so the grapes do not fall when cut. All prototyping was completed on a 3D printer as it is a fast and easily accessible prototyping method.

Testing

All tests were completed on supermarket grape stems. These differ from stems that are still on the vines. Most of them are smaller in diameter and are slightly harder due to the reduction in water content. The gripping mechanism was designed with the actual size diameters in mind. It was assumed that the increase in hardness would mitigate the reduction in diameter and therefore the stems from the supermarket grapes would still be realistic enough for testing.

Some very low fidelity tests were done to calculate an estimate for the force required to cut a stem. This was done by cutting a range of stems on a set of scales and recording the weight required to cut them.

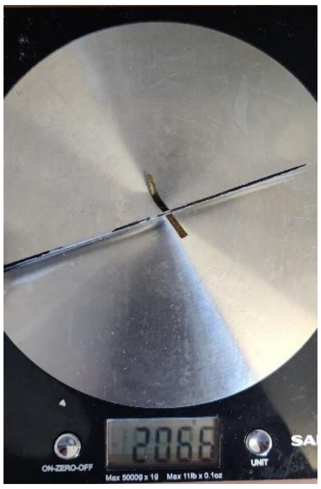


Figure 38 - Stem being cut on a set of scales

Most stems could be cut by applying around 2kg or 20N of force.

Potential Problems

The main potential problem is that the gripping and cutting mechanisms cannot provide enough force and either drop the grapes or cannot cut through the stems. Although this can be mitigated by using higher powered motors or a gear box it is important that the end effector keeps a small form factor so it does not damage any grapes or vines while operating in a small area.

A second potential problem is that the vines might not be long enough for the end effector to cut them. This would tend to occur if the gripping and cutting mechanisms are too far apart so it is important that they be kept as close together as possible.

Initial Design Review

At the end of the first semester several initial designs were drawn up. These took a range of forms and followed a ‘no idea is a bad idea’ approach which led to some quite quirky designs.

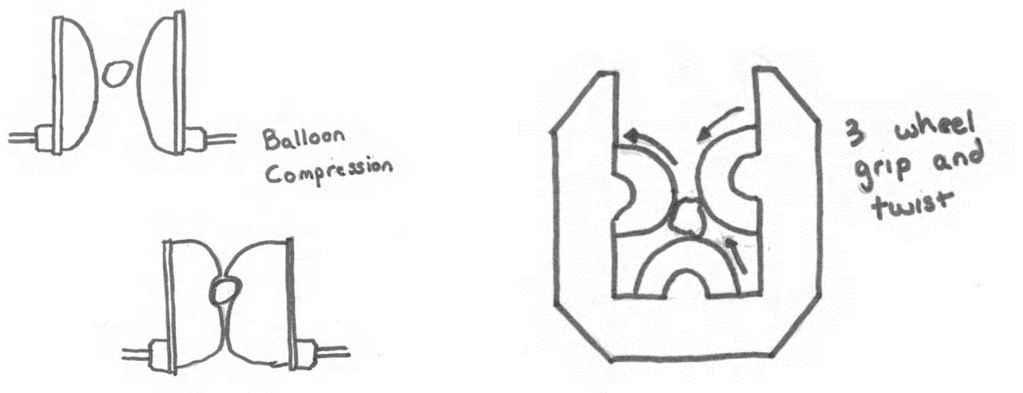


Figure 39 - 'Whacky' initial designs

However, after getting some feedback from peers and a vineyard owner it was decided that the most obvious designs would perform the best. These designs were all based around a pincer movement; one moving linearly and one rotating.

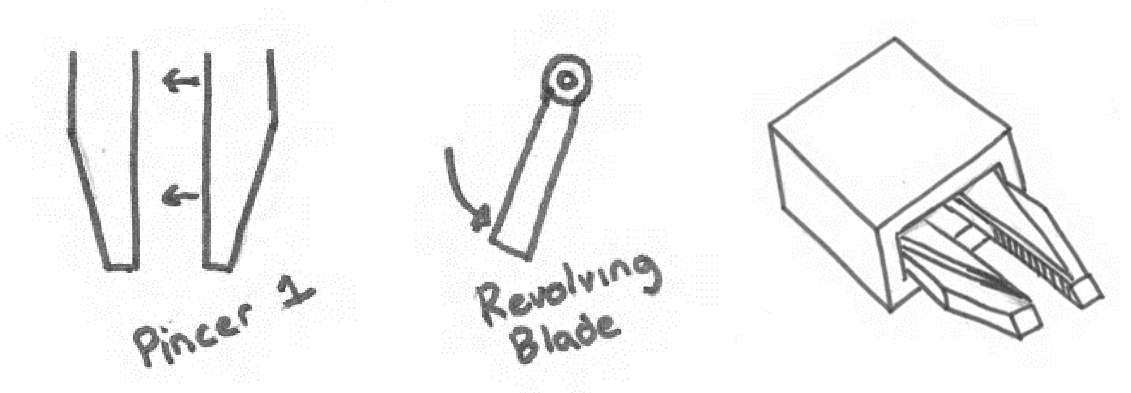


Figure 40 - Initial designs based around a pincer movement

The two mechanisms were initially designed separately and were later combined into one design. It was also decided that the gripping and cutting mechanisms should be designed using the same movement method as it would allow for faster prototyping and would allow for easy integration when the two mechanisms were combined.

Development

Initial Prototypes

Following on from the initial designs, two simple prototypes were made (V1.1 and V1.2). These used rubber bands to provide the force to close them.

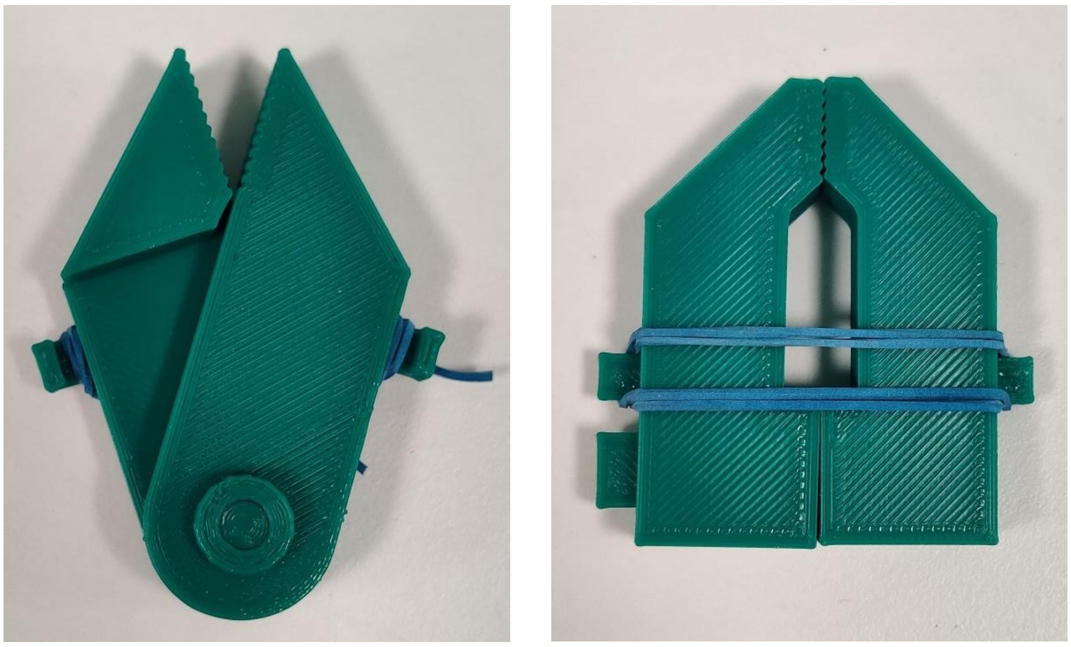


Figure 41 - First prototypes based on two types of pincers

At this stage it was important to also look at the hardware that could be used to perform the required task.

Rotational Movement

For rotational movement a servo motor would be used. Servo motors can supply a large amount of torque and therefore force to the stem. This can also be increased by adding a gear box. A large number of designs can also be created based around the rotational movement.

Linear Movement

For linear movement a linear solenoid would be used. These can also provide a large amount of force however this cannot be increased through mechanical design easily. It also limits the design space with only one design that would be expected to perform well.

Due to the reasons stated above a rotational design approach was taken forward. However, a linear mechanism was not discounted and could be returned to if the rotational method did not work.

Rotational Designs

Three rotational designs were initially made again using rubber bands to imitate the motors.

The first (V2.1, figure 42) was a redesign of the initial prototype with gripping surfaces now meeting parallel to each other. Each arm would move away from the other at the same speed.

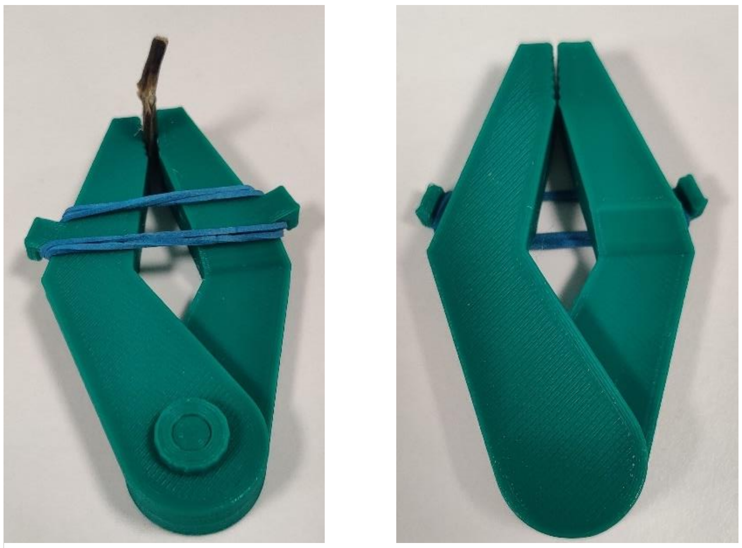


Figure 43 - Prototype V2.1

The second design (V2.2, figure 43) had just one offset arm that moved and met a static arm to grip against.

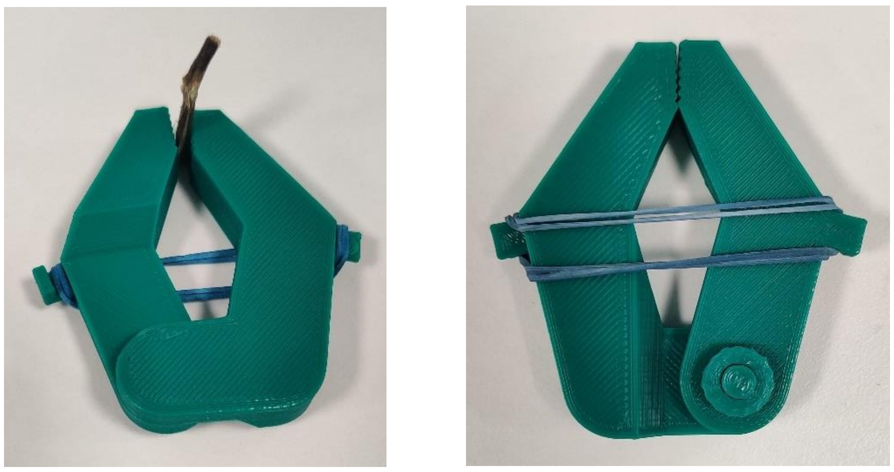


Figure 42 - Prototype V2.2

The third design used a gearing mechanism so that the arms mirrored each other’s movement. This design would allow for both arms to move while only using one motor.

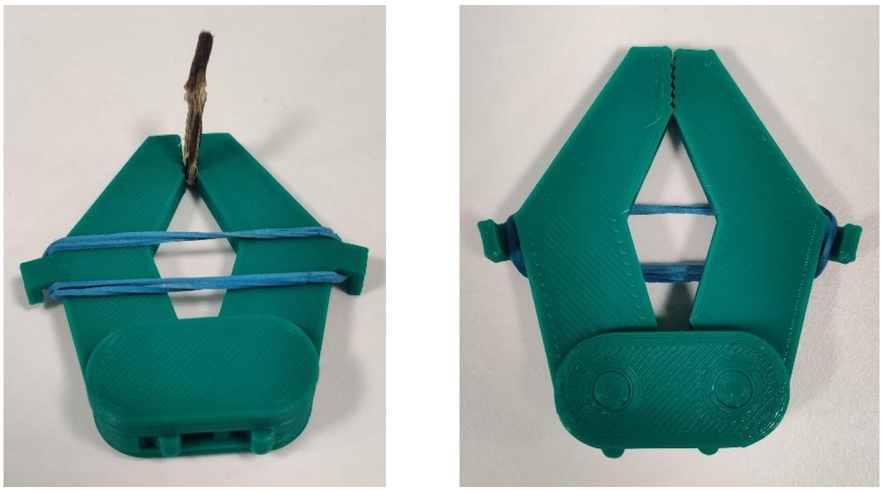


Figure 44 - Prototype V2.3

All of the designs worked well with a rubber band being used however it would be hard to move both arms in V2.1 using just one motor and therefore it was discounted. The other two designs were taken forward to be further prototyped.

Motorised Designs

At this stage it was important to start powering the designs properly. This would help to make important design decisions and allow for some early testing. Dynamixel XC330-M288-T servo motors were chosen to do this. These motors provided 0.93Nm of torque meaning when applied to the designs they can apply up to 20N of force which is enough to cut grape stems based on the early testing. They have a relatively small form factor meaning the designs can also be kept small. Dynamixel motors also come with several features that made prototyping with them very easy. Firstly, they can be controlled through an Arduino Nano using a shield. The shield allows the motors to be daisy chained together but still be controlled individually leading to a simple plug and play experience. They also only require 5V to be powered meaning additional circuitry was not needed.

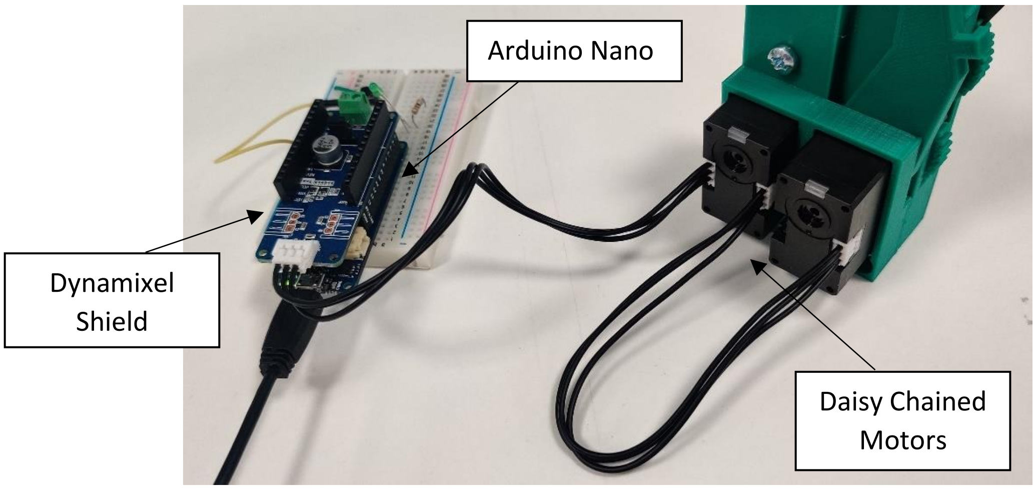


Figure 45 - Arduino, Shield and motor set up

Secondly, the motors have a large number of additional features such as angle position control and built in current (and therefore torque) feedback. Finally, they already have many examples built into the Arduino software and a good support community online. Arduinos are ideal for fast prototyping as they are cheap and easy to program. A simple button circuit (figure 44) was used to cycle through the different stages of the gripping and cutting process (figure 45).

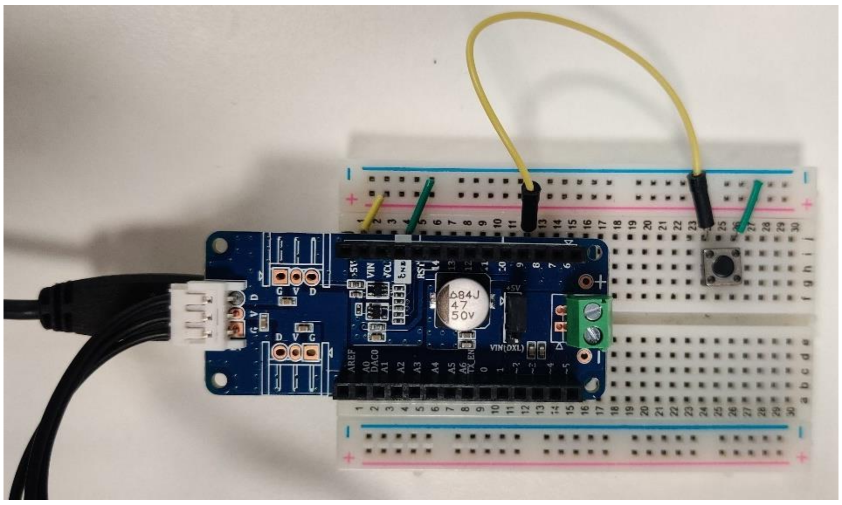


Figure 46 - Button Circuit

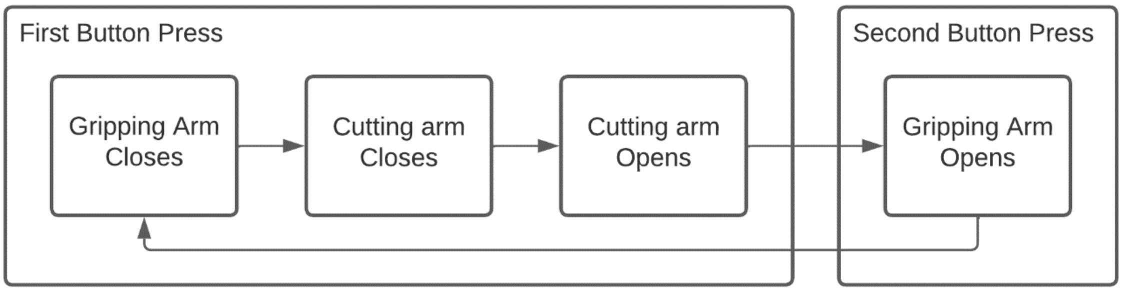


Figure 47 - Flowchart of end effector movements

The offset design (V3.1, figure 48) worked well with the motor and gripped the grapes securely. It was noted however that the size of the arms could be reduced dramatically. This would reduce the chance of them breaking and increase the force on the stem.

The geared design (V3.2, figure 49) also worked well when holding the grapes. However, it did require more fine tuning due to the gears allowing for a small amount of wiggle room. Specifically with smaller vines this led to the grip strength not being as robust as it should be. Although this could be rectified with a better gearing it was decided that this design was overly complicated so it was not taken forward.

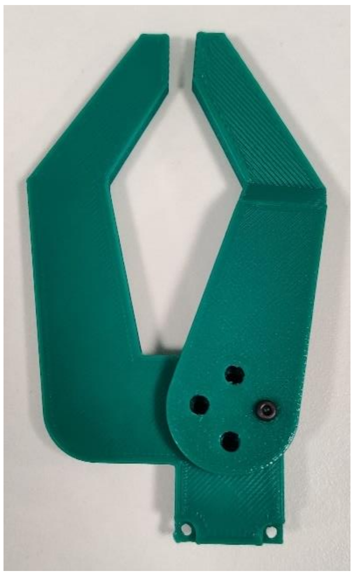


Figure 48 - Prototype V3.1

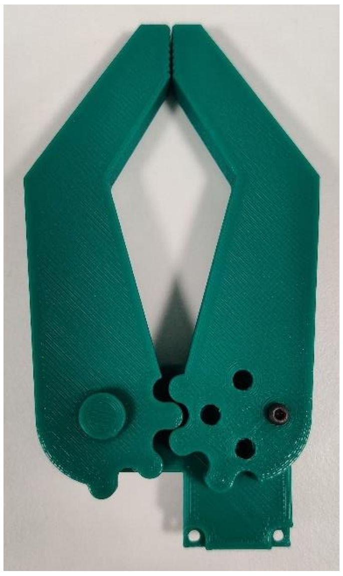


Figure 49 - Prototype V3.2

At this stage the offset design was also adapted to make a cutting version. This again performed well but struggled with some thicker vines. Again, the arm length was larger than it needed to be so it was decided that this would be reduced in the next prototype and that this may rectify the strength problem. A Stanley blade was used in the design as they are easy to acquire and can be fixed into the design easily. As stated in the initial specification, having blades that can be easily acquired and replaced is important so that custom parts do not need to be made and the user does not need specialist equipment or an engineer to have the blade replaced.

Combined Design 1

Having proved that the current gripper and cutter designs could work, they were combined into one prototype. This was very simple and placed the designs back-to-back. As previously mentioned, the arm sizes were reduced to increase their force.

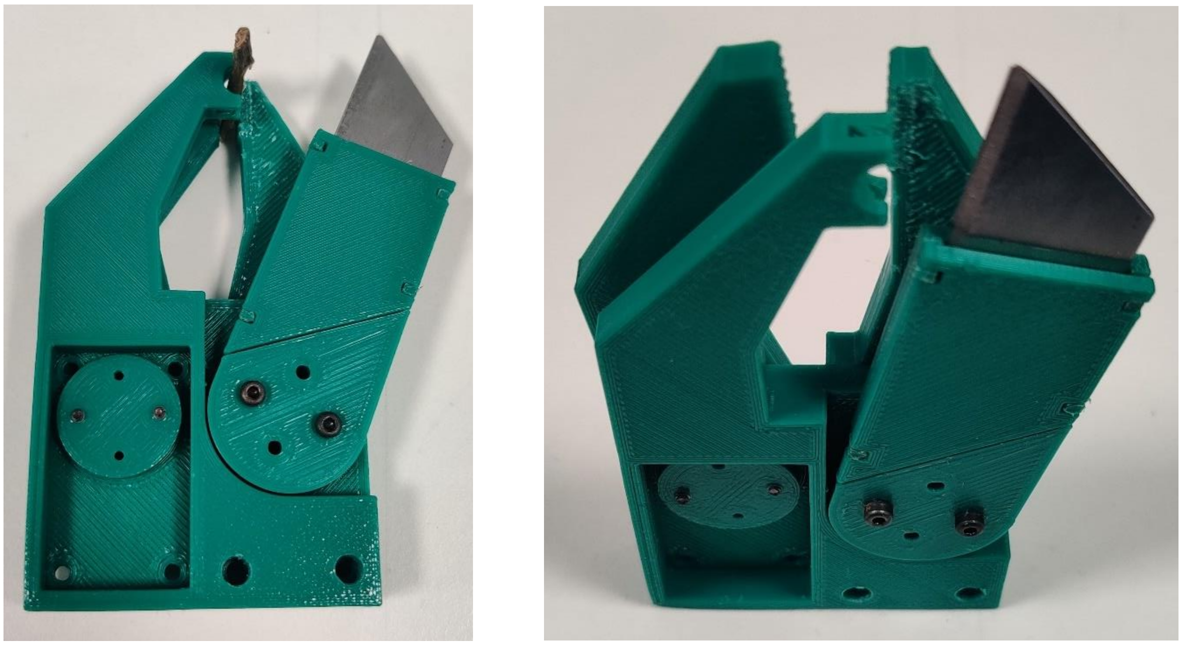


Figure 50 - Combined Design 1

This design had one major flaw. The arms moved in opposite directions to each other meaning that the static arms almost overlapped which made moving the vine into position very difficult. The cutting arm also still did not have quite enough power to successfully cut through thicker vines. Finally, when stress testing the gripping arm by shaking it with a bunch of grapes in it, the vine would occasionally slip a few millimetres through the arms. Although it was never dropped, this was not ideal and requires some modification if possible.

Combined Design 2

In order to solve the cutting problem and the arms being on the opposite sides to each other, a small gearing mechanism was added. This allowed the cutting arm to be moved to the same side of the gripping arm making it much easier for the vine to be moved into the optimal position.

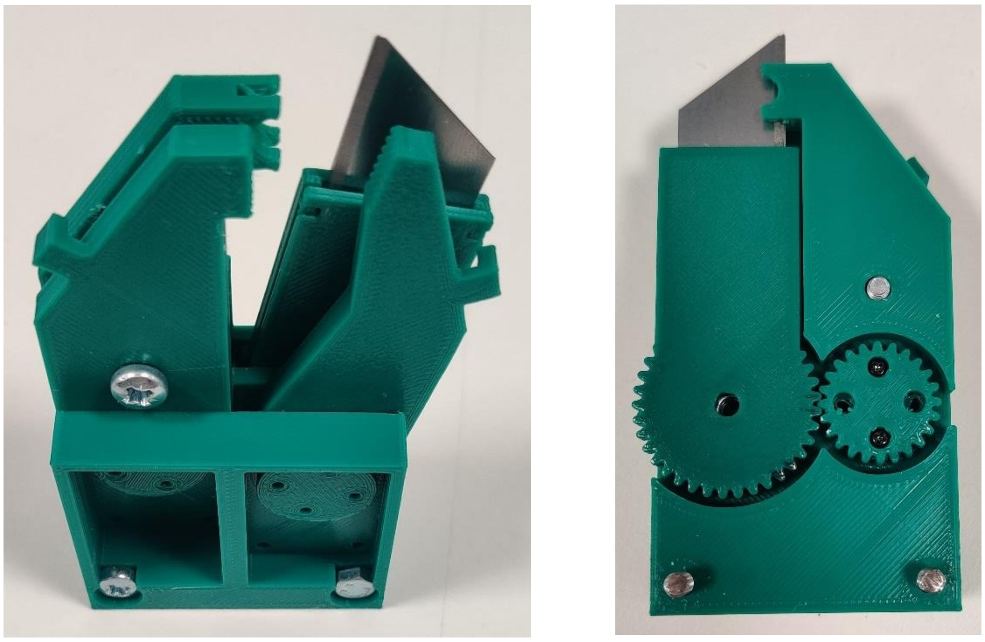


Figure 51 - Combined Design 2

A more knurled surface was added to the griping parts of the arms to reduce the likelihood of the stem slipping.

Small hooks for a rubber band to be stretched around were added to the design in a similar way to the initial low fidelity prototypes. The idea behind this was that the servo motors would not need to be powered while the gripper was closed. This would reduce the stress on the motors and reduce their power consumption.

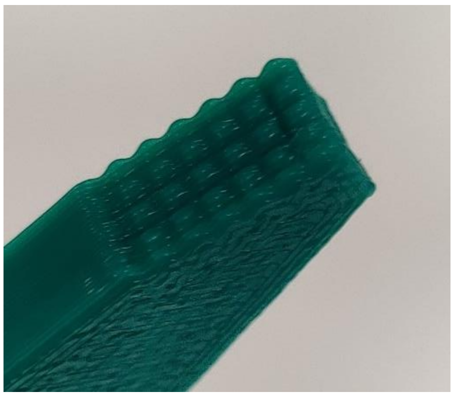


Figure 52 - Knurled surface

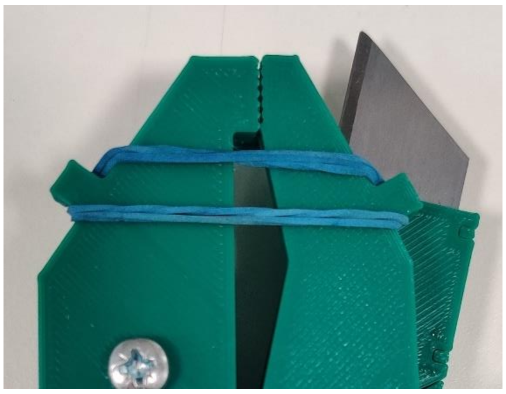


Figure 53 - Rubber band for unpowered closing

All the additions improved the design dramatically and allowed the grape stems to be cut fairly reliably. The cutter still struggled with larger stems but would reliably cut them on the second attempt. The knurling worked as intended and stopped the vines from slipping. The rubber band proved the principle however, it was hard to match the force applied to the motor. This meant that the rubber band either stopped the gripping arm from opening or did not apply enough force when it was closed.

Final Prototype

The final version aimed to increase the cutting force by adding further gearing. This increased the ratio to 3:1 and gave the design a much higher torque. The position of the hooks for the rubber band were moved slightly to try to improve their effectiveness.

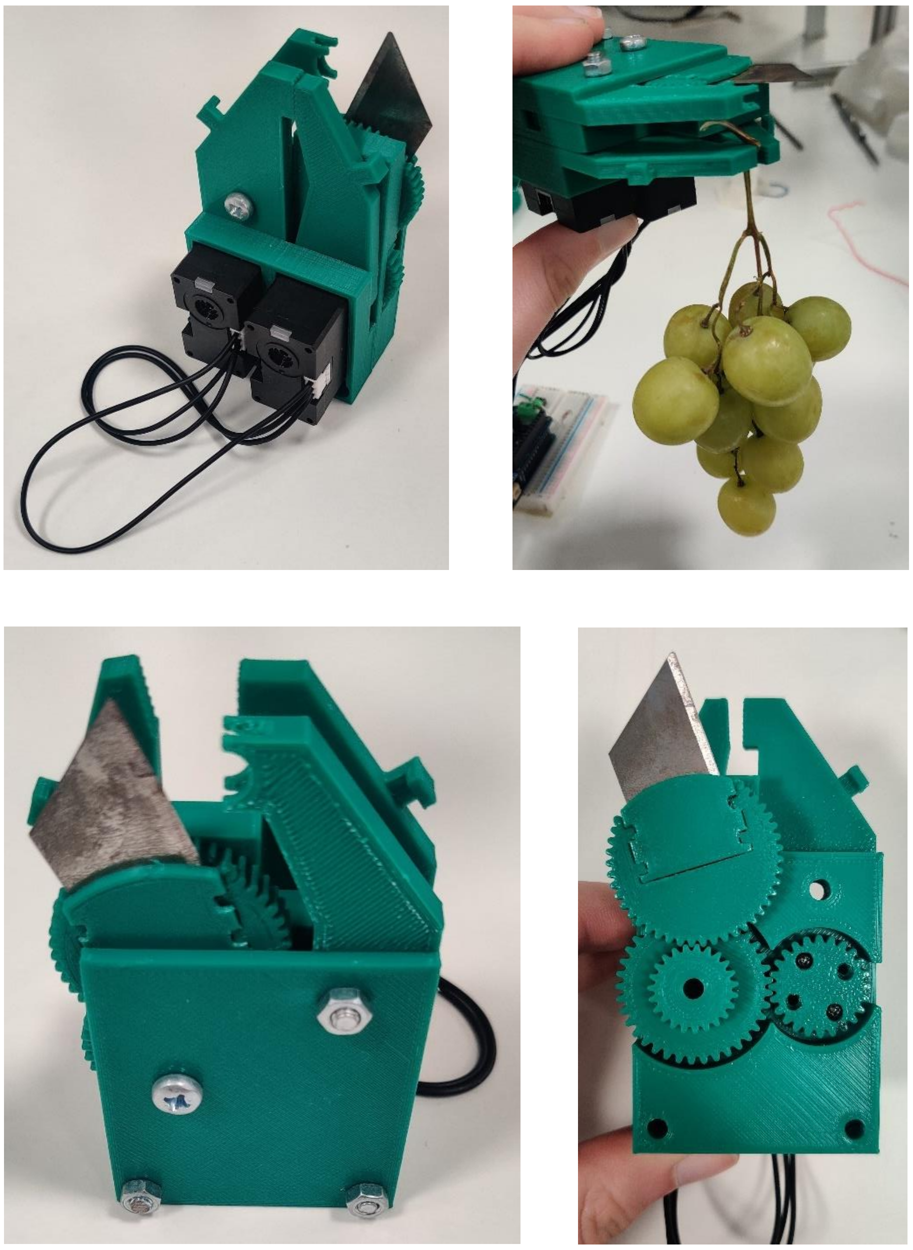


Figure 54 - Final Prototype (See Appendix B for exploded view)

With the increase in torque the design was able to grip and cut grapes very effectively. The unpowered when closed system still needs some refinement but this prototype showed that it could be done. Due to more gears being added, the Stanley blade had to be cut to fit into the design and the gripping and cutting arms had to be moved slightly further apart. This meant that a stem would need to be at least 15mm long for it to successfully be gripped and cut.

Future Work

Improvements

The gripping system worked well with the grape bunches which it was tested on. However, as previously mentioned, these are smaller than bunches found on an actual vineyard. Therefore further steps to improve the stability of the stem when it is being held should be investigated. Due to the quality of the 3D printer used the knurling was not highly accurate and could be improved by using a higher quality 3D printer or machined parts. Alternatively, a rubber end could be added to make the surface grip better. One easy way to prototype this would be to 3D print some rubber ends with knurling using Thermoplastic Polyurethane (TPU) which can be done on most modern printers.

In order to improve the unpowered when closed system further testing will need to be done in order to find the optimal way to match the motor's torque. This could be done initially by adding some adjustable hooks for the rubber band to go around. Alternatively using springs with a variety of spring constants could be used.

Although the cutting mechanism cut the all the stems that were tested, the stems in the vineyard will be larger. Therefore it is possible that this prototype will not reliably work when using full sized bunches of grapes. Using the current motors, it will be hard to increase the torque without adding a very high gear ratio which may cause other mechanical issues and increase wear. Using a higher torque motor would address this problem but would also have a larger form factor and require more power and additional electronics (assuming they require more than 5V). One alternative to this would be to use pneumatics which can provide large amounts of force in a small form factor. This would mean moving back towards using a linear movement system.

Another way to improve the cutting mechanism would be to look at other ways it could be done. Currently the blade is cutting against a hard surface however, lots of other cutting devices such as scissors and secateurs cut by using two blades moving against each other in a shearing movement. This method may improve the reliability of the cutting mechanism but it does detract from the idea of using easily replaceable parts.

In the next prototype the design should be altered so that the gripping and cutting mechanism are closer together to allow for shorter stems to be cut more easily. The design should also be altered so that the Stanley blade does not need to be shortened in order to fit into the design.

Additional Features

The Dynamixel motors used have several features which were useful for prototyping but could also be used to perform some useful checks. For example, when the system is trying to grip a stem the position feedback could be used to find if there is anything between the arms or not. This could also be used to check the cutting mechanism.

Moving Forward

Further Testing

The most obvious next step for the project is to integrate the current prototypes and create a working system. This would include the introduction of a robotic arm which could be custom designed or purchased. This should be done in time for harvest season in October so that some realistic tests can be completed. With further testing in mind, a more realistic mock vineyard should be constructed so that more complicated and difficult tests can be run. This would include different levels of depth for grapes to be hung from and the ability to hang more grapes up. Another useful addition would be to put the camera on a rail so it could mimic the movement of the rover. This would mean additional code could be added to stop movement when it finds a grape. It would also be useful to find an alternative to supermarket bought grapes for testing. Being able to test the system with full sized bunches is integral for both the vision system and the end effector. It would be useful to find a plant that grows all year round that has similar dimensions and properties to grape stems so more realistic testing can be done on the cutting mechanism. Given that harvest season is only about 2 weeks long, the system must be robust so that testing can be done efficiently without the need for trouble-shooting. The rover should not be integrated initially as this can be mimicked very easily and the time before harvest season would be better spent optimising the end effector, robotic arm and vision system.

Design Decisions

One of the first decisions to be made is how the cutting arm of the end effector will be powered. If pneumatics are going to be used for the cutting arm, it may be beneficial to power the entire system using pneumatics including the robotic arm. The main consideration when making this decision will be the power consumption of both systems. Running a compressor for the entire system will use more power than an electronic system. If this limits the picking time too much it will mean that the grapes cannot be harvested in time and will be lost.

The battery and charging method will also need to be considered. The rover will return to a charging station automatically when it gets low on power and ideally this would be done overnight when lighting conditions are worst. However, charging just once in 24 hours may not be possible. The battery size will be dictated by the power consumption but initially a battery like the one used in the NAVVY could be used as it will probably have the same or higher power consumption.

Research has shown that mounting the camera on the robotic arm can improve a vision system's accuracy greatly. Adding this functionality to the grape picker would help in several ways. Firstly, it allows the camera to be moved easily. This could allow the vision system to check low percentage certainty grapes from a different angle and confirm if they are grapes or not. It would also help with stem identification as the arm can move the camera much closer to the stem meaning the vision system can operate on a much higher resolution version of the image.

Conclusions

This project has been a useful proof of principle and the final prototypes show that with further development a fully working grape picker could be produced. I am confident that through researching and creating a large number of designs, some of the optimal solutions have been explored in detail and that if this project was taken forward the current prototypes would provide a useful base to work from.

The vision system has mitigated many of the potential problems that were outlined initially such as leaves being identified as grapes and background grapes interfering with the image. It can successfully identify five, and potentially more, bunches of grapes. Although the percentage certainty calculation function is currently limited to just 3 bunches this feature could be improved. In practice this might not be problematic because as the bunches are picked and removed from the image the remaining bunches will be able to have their certainty calculated. There are also several ways in which the vision system could be adapted such as improving the shape recognition and looking for individual grapes in a bunch which would reduce the chance of leaves or other objects being picked out as grapes.

The end effector can grip and cut a bunch of grapes reliably and has kept a small form factor which was initially specified as being important. The final prototype has a few small issues such as the need to cut the Stanley blade and the gripping and cutting arms being slightly too far apart. A relatively simple redesign might rectify these problems quite easily. Being able to test the end effector on stems still attached to the vine is the next important test that needs to be completed and will give a better guide to the function of this design. A number of possible changes to the system have been outlined to improve the force of the cutting mechanism such as higher-powered motors or two cutting blades. This issue has had to be regularly addressed several times throughout the project and is likely to be where the system may run into problems in the future.

Moving forward, the vision system and end effector would continue to be improved through testing and further prototypes but adding a robotic arm into the system would allow for the first integration between them. This should aim to be completed by harvest season so that more realistic tests can be completed. I believe that the designs and prototypes produced over the last semester give a firm base to work from and if taken forward could contribute to a high-quality grape picker.

References

[1] D. Pencheva, “Coronavirus: flying in fruit pickers from countries in lockdown is dangerous for everyone,” The Conversation, 21 April 2020. [Online]. Available: https://theconversation.com/coronavirus-flying-in-fruit-pickers-from-countries-in-lockdown-is-dangerous-for-everyone-136551. [Accessed 7 November 2021].

[2] L. Eccles and T. Calver, “Hunt for fruit pickers extends 6,000 miles as shortage bites,” The Times, 12 September 2021. [Online]. Available: https://www.thetimes.co.uk/article/hunt-for-fruit-pickers-extends-6-000-miles-as-shortage-bites-xtthzl3mp. [Accessed 08 November 2021].

[3] H. Al-Othman, “The Times,” The Times, 23 August 2020. [Online]. Available: https://www.thetimes.co.uk/article/britains-fruit-farmers-hanker-for-return-of-foreign-pickers-r0z8wj6qm. [Accessed 07 November 2021].

[4] H. Sandercock, “Pick for Britian scheme for UK-based farm workers scrapped,” The Grocer, 16 April 2021. [Online]. Available: https://www.thegrocer.co.uk/hiring-and-firing/pick-for-britain-scheme-for-uk-based-farm-workers-scrapped/655189.article. [Accessed 7 Novermber 2021].

[5] “Agrobot E-series,” Agrobot, [Online]. Available: https://www.agrobot.com/e-series. [Accessed 5 December 2021].

[6] W. YUN, W. ZIRUI, S. YUGENG and H. FAN, “Automatic pesticide spraying and picking device for single-trellis”. China Patent CN108605510A, 2 October 2018.

[7] Z. TAO, L. YONG, Z. YI, Y. JUBIAO, Z. ZHENGGUO and D. XIAOJUN, “Intelligent humanoid grape picking robot”. China Patent CN113057023A, 2 July 2021.

[8] X. Fuxiang, S. Jian, H. Hongpeng and H. Xinxin, “Grape picking device”. China Patent CN107371617A, 12 September 2017.

[9] C. JIAJIA, C. JIWEN, Y. HONGJUAN and C. QINGPENG, “Agricultural vineyard intelligent automatic picking and conveying device”. China Patent CN107079669A, 22 August 2017.

[10] “Dorna,” Dorna Robotics, [Online]. Available: https://dorna.ai/. [Accessed 28 January 2022].

[11] “Robotis e-Manual XC330-M288-T,” Robotis, [Online]. Available: https://emanual.robotis.com/docs/en/dxl/x/xc330-m288/. [Accessed 10 February 2022].

[12] “Advantages and Disadvantages of Machine Learning Language,” Data Flair, [Online]. Available: https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/. [Accessed 5 January 2022].

[13] D. M. J. Garbade, “Understanding K-means Clustering in Machine Learning,” 12 September 2018. [Online]. Available: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

[14] N. Dhanachandra, K. Manglem and Y. j. Chanu, “Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm,” in *Procedia Computer Science*, Bangalore, 2015.

[15] “Fundamentals of image contours,” Evergreen Technologies, 2 January 2020. [Online]. Available: https://evergreenllc2020.medium.com/fundamentals-of-image-contours-3598a9bcc595. [Accessed 25 February 2022].

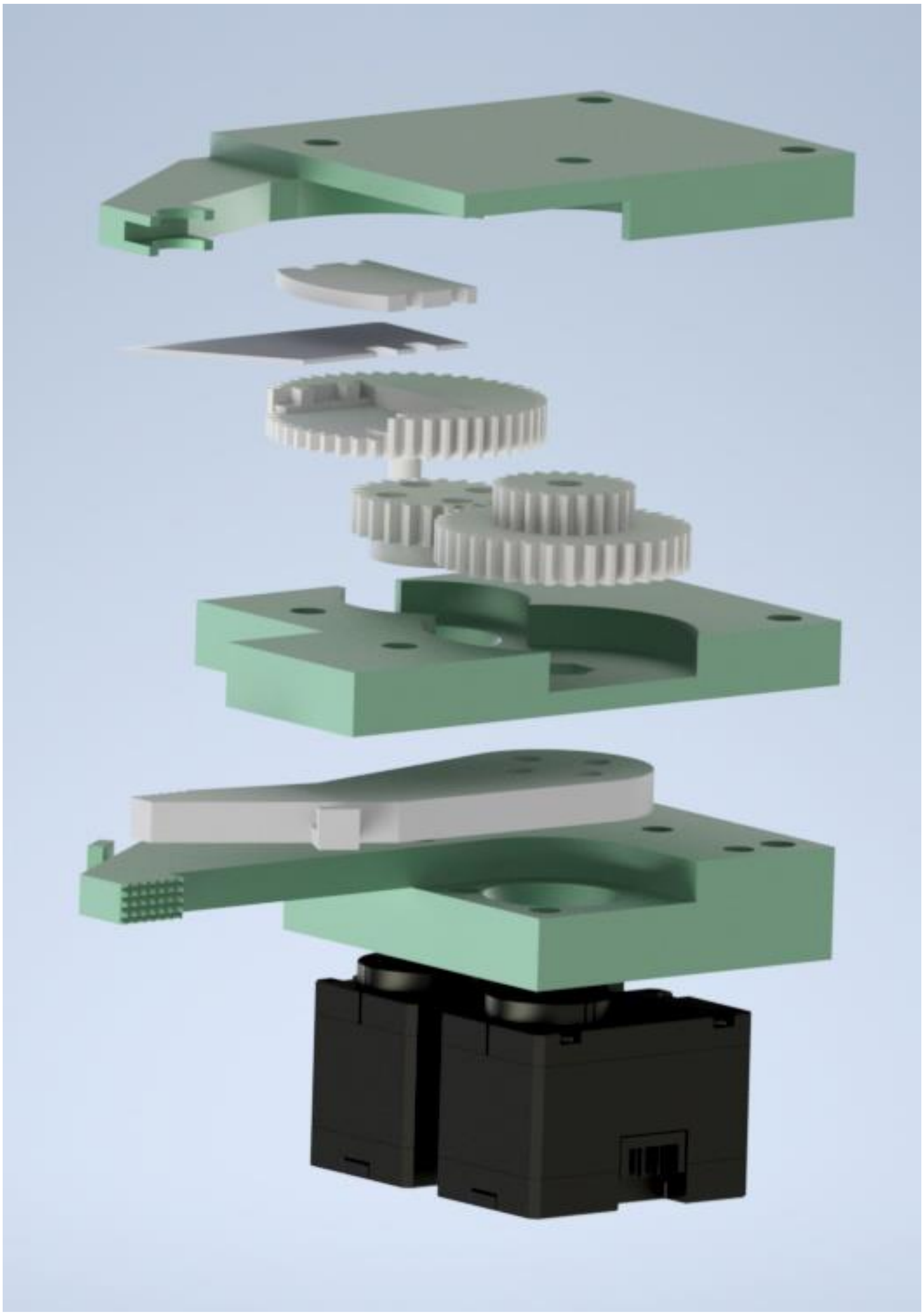
Appendix

A – Vision System Python Code

```python
import pyrealsense2 as rs
import numpy as np
import cv2
import math

# function to stop limit values
def clamp(input, min, max):
    if input < min:
        return min
    elif input > max:
        return max
    else :
        return input

#function used to lable final images
def label(text,image):
    text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.75, 2)
    cv2.rectangle(image, (0, 445), ((text_size[0])[0] + 20, 480), (0, 0,
0), -1)
    cv2.putText(image, text, (10, 470), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
(255, 255, 255), 2)


# Create a pipeline
pipeline = rs.pipeline()

# Create a config and configure the pipeline to stream
# different resolutions of color and depth streams
config = rs.config()

# Get device product line for setting a supporting resolution
pipeline_wrapper = rs.pipeline_wrapper(pipeline)
pipeline_profile = config.resolve(pipeline_wrapper)
device = pipeline_profile.get_device()
device_product_line = str(device.get_info(rs.camera_info.product_line))

found_rgb = False
for s in device.sensors:
    if s.get_info(rs.camera_info.name) == 'RGB Camera':
        found_rgb = True
        break
if not found_rgb:
    print("The demo requires Depth camera with Color sensor")
    exit(0)

config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 15)

if device_product_line == 'L500':
    config.enable_stream(rs.stream.color, 960, 540, rs.format.bgr8, 15)
else:
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 15)

# Start streaming
profile = pipeline.start(config)

# Getting the depth sensor's depth scale
```

```
depth_sensor = profile.get_device().first_depth_sensor()
depth_scale = depth_sensor.get_depth_scale()
#print("Depth Scale is: " , depth_scale)

# Create an align object
# rs.align allows us to perform alignment of depth frames to others frames
# The "align_to" is the stream type to which we plan to align depth frames.
align_to = rs.stream.color
align = rs.align(align_to)

#set HSV limits for the chroma key filter
lower_HSV = np.array([20,50,50])
upper_HSV = np.array([40,255,255])
#Set initial width-height coeficient
wh_coef = 1
# Streaming loop
try:
    while True:
        # Get frameset of color and depth
        frames = pipeline.wait_for_frames()
        # frames.get_depth_frame() is a 640x360 depth image

        # Align the depth frame to color frame
        aligned_frames = align.process(frames)

        # Get aligned frames
        aligned_depth_frame = aligned_frames.get_depth_frame() #
aligned_depth_frame is a 640x480 depth image
        color_frame = aligned_frames.get_color_frame()

        # Validate that both frames are valid
        if not aligned_depth_frame or not color_frame:
            continue

        depth_image = np.asanyarray(aligned_depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        # Find closest pixel and set clipping distance 200mm behind it
        additional_distance = 200
        closest_pixel = 600
        clipping_distance = (additional_distance + closest_pixel) /
depth_scale /1000

        # Remove background - Set pixels further than clipping_distance to
grey
        grey_color = 0
        depth_image_3d = np.dstack((depth_image,depth_image,depth_image))
#depth image is 1 channel, color is 3 channels
        bg_removed = np.where((depth_image_3d > clipping_distance) |
(depth_image_3d <= 0), grey_color, color_image)

        # Render images:
        #   depth align to color on left
        #   depth on right
        depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(depth_image,
alpha=0.03), cv2.COLORMAP_JET)
        images = np.hstack((bg_removed, depth_colormap))

        # Creates a chroma key mask using the predefined bgr limits and
sets
        # all other pixels to black
        HSV = cv2.cvtColor(bg_removed,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(HSV , lower_HSV, upper_HSV)
        masked_image = np.copy(bg_removed)
        masked_image[mask == 0] = [0, 0, 0]

        ## creates a contour around the image and then draws a rectangle
around the objects ##
        # convets image to grey
        imgray = cv2.cvtColor(masked_image, cv2.COLOR_BGR2GRAY)
        # adds a blur to the image
        imgblur = cv2.GaussianBlur(imgray, (5, 5), 1)
        # draws the contours around the object
        ret, thresh = cv2.threshold(imgblur,30, 255, 0)
        contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        percentages = []
        coords = []
        for cnt in contours:
            area = cv2.contourArea(cnt)
            peri = cv2.arcLength(cnt, True)
            approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)
            objCor = len(approx)
            x, y, w, h = cv2.boundingRect(approx) #calculate bounding
rectangle coordinates
            # adds a filter to remove any contours that have an area that's
too small or not enough points
            if area > 1200:
                if objCor > 7:
                    percentage  = 0 # set intial percentage value
                    #cv2.drawContours(color_image, cnt, -1, (255, 0, 0), 3)
                    # draws a rectangle around the contour based on the
coordinates found above
                    cv2.rectangle(color_image, (x, y), (x + (w), y + (h)),
(0, 255, 0), 2)

                    # finds the centre of the supposed grape and finds the
depth at that point
                    xcentre = clamp((int(x + (w / 2))),0,479) #clamp so
point is within image
                    ycentre = clamp((int(y + (h / 2))),0,639)
                    ystalk = y - 10 #assumed stalk coordinate
                    cv2.circle(color_image,(xcentre,
ystalk),2,(0,0,255),2,1) #add circle where stem should be cut
                    #calculate area coeficient
                    depth = depth_image[xcentre, ycentre]
                    exp_area = 183285 * math.exp(-0.006*depth)
                    area_coef = 1 - (abs(1-(area/exp_area)))
                    #calculate width/height coeficient
                    if (w * 3) < h or h > (w*6):
                        wh_coef = 0.5
                    else : wh_coef = 1
                    if depth != 0 and depth < 1000: #check if depth value
is realistic
                        percentage = clamp((round((100 * (area_coef) *
wh_coef), 0)),0,100) #calculate percentage and limit between 0 and 100
                        if (percentage is not None):
                            percentages.append(percentage)
                    if (xcentre is not None) and (ystalk is not None):
                        coords.append([xcentre,ystalk])
                    #add data to output image
                    cv2.putText(color_image,'Certainty : ',(x,y + h +
20),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,0,0),2)
                    cv2.putText(color_image, ' ' + str(percentage) + '%',
(x, y + h + 40),cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)
                    #cv2.putText(color_image,str(depth), (x, y),
cv2.FONT_HERSHEY_SIMPLEX,0.75, (255, 0, 0), 2)

        ## Display the final image ##
        # Change names of images
        Image1 = bg_removed
        Image2 = HSV
        Image3 = masked_image
        Image4 = color_image
        #Add title for each image (label function at start of code)
        label('Remove Background', Image1)
        label('HSV', Image2)
        label('Chroma-key filter', Image3)
        label('Grape Centainty', Image4)
        if coords:
            cv2.putText(color_image, 'Pick Location X : ' +
str((coords[0])[0]) + ' Y : ' + str((coords[0])[1]) + ' D : ' +
str(depth_image[(coords[0])[0],(coords[0])[1]]), (0,
20),cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 0, 0), 2)
        #Add all images to one screen
        img_concate_hori_1 = np.concatenate((Image1,Image2), axis=1)
        img_concate_hori_2 = np.concatenate((Image3, Image4), axis=1)
        img_concate_vert =
np.concatenate((img_concate_hori_1,img_concate_hori_2), axis=0)
        #Display the final images
        cv2.imshow('Grape Detection',img_concate_vert)
        key = cv2.waitKey(1)
        # Press esc or 'q' to close the image window
        if key & 0xFF == ord('q') or key == 27:
            cv2.destroyAllWindows()
            break
finally:
    pipeline.stop()
```

B – Final Design Exploded View



C – Arduino Code for End Effector

```
#include <DynamixelShield.h>

//set up motors

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560)

  #include <SoftwareSerial.h>

  SoftwareSerial soft_serial(7, 8); // DYNAMIXELShield UART RX/TX

  #define DEBUG_SERIAL soft_serial

#elif defined(ARDUINO_SAM_DUE) || defined(ARDUINO_SAM_ZERO)

  #define DEBUG_SERIAL SerialUSB

#else

  #define DEBUG_SERIAL Serial

#endif


const uint8_t DXL_CUTTER = 2;

const uint8_t DXL_GRIPPER = 1;

const float DXL_PROTOCOL_VERSION = 2.0;


//Set angles for arms to move to

int GripperClosed  = 0;

int GripperOpen = 30;

int CutterClosed = 0;

int CutterOpen = 40;

int BS;

int State = 1;


DynamixelShield dxl;


using namespace ControlTableItem;


void setup() {

  //set up button
  pinMode(8, INPUT_PULLUP);  // Pin 8 reads 1 if not connected to GND
  DEBUG_SERIAL.begin(115200);

  // Set Port baudrate to 57600bps. This has to match with DYNAMIXEL baudrate.
  dxl.begin(57600);
  // Set Port Protocol Version. This has to match with DYNAMIXEL protocol version.
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  // Get DYNAMIXEL information
  dxl.ping(DXL_CUTTER);
  dxl.ping(DXL_GRIPPER);
  // Turn off torque when configuring items in EEPROM area
  dxl.torqueOff(DXL_CUTTER);
  dxl.setOperatingMode(DXL_CUTTER, OP_POSITION);
  dxl.torqueOn(DXL_CUTTER);
  dxl.torqueOff(DXL_GRIPPER);
  dxl.setOperatingMode(DXL_GRIPPER, OP_POSITION);
  dxl.torqueOn(DXL_GRIPPER);

  //Angle Conversion
  GripperClosed = (GripperClosed + 80) * 11.377;
  GripperOpen = (GripperOpen + 80) * 11.377;
  CutterClosed = (CutterClosed) * 11.377;
  CutterOpen = (CutterOpen) * 11.377 * 3; //x3 for gearing
}
void loop() {
  BS = digitalRead(8); //read button pin
  if (BS == 0) { //wait for button press
    if (State ==0) { //State 0 - Grip stem, cut stem, keep gripper closed for movement
      dxl.writeControlTableItem(PROFILE_VELOCITY, DXL_GRIPPER, 75); //set movement speed for
gripping arm
      dxl.writeControlTableItem(PROFILE_VELOCITY, DXL_CUTTER, 500); //set movement speed for
cutting arm
```

```
      dxl.setGoalPosition(DXL_GRIPPER,GripperClosed); //Close gripping arm
      delay(2000);
      dxl.setGoalPosition(DXL_CUTTER, CutterClosed); //Close cutting arm
      delay(2000);
      dxl.writeControlTableItem(PROFILE_VELOCITY, DXL_CUTTER, 75); //set movment speed of
cutting arm
      dxl.setGoalPosition(DXL_CUTTER, CutterOpen); //Open cutting arm
      State = 1;
      delay(1000);
    }
  }
  BS = digitalRead(8); //read button pin
  if (BS == 0) { //wait for button press
    if (State == 1) { //State 1 - Release stem
      dxl.setGoalPosition(DXL_GRIPPER,GripperOpen); //open gripping arm
      State = 0;
      delay(1000);
    }
  }
}
```