White Maize Classification and Grading

Shane Erasmus
23549777

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Rensu P. Theart
Department of Electrical and Electronic Engineering

October 2023

Acknowledgements

I would first like to thank Johan Olivier, Arlo Steyn, and Werner at Timber King for their help in building and testing the system. I would also like to thank Wiana and Ben from SAGL for providing knowledge and insights about the agriculture industry and for supplying maize samples that were key to the development of the system.

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

23549777 Studentenommer / *Student number*	S Erasmus Handtekening / *Signature*
S Erasmus Voorletters en van / *Initials and surname*	27/10/2023 Datum / *Date*

Abstract

This report presents a innovative approach to maize sample grading. This is done by building a image capturing system coupled with image processing technologies and deep learning algorithms. The core portion of the report revolves around overcoming many of the limitations and difficulties in developing such a system and optimizing its performance. The developed system is finally tested on unseen batches of maize and compared with other technologies. The results gathered offer valuable insights and promise it's potential for developing further solutions that could, some day, be utilized in the industry.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

Ω	Ohms
A	Amperes
DIS	Discolored kernels
DoG	Difference in Gaussian blur
EUR	Euros
HD	Heat Damage
HI	Hyperspectral Imaging
ID	Insect Damage
LED	Light Emitting Diodes
MI	Multispectral Imaging
MSE	Mean Squared Error
nm	Nanometers
PCA	Principal Component Analysis
$PLS-DA$	Partial Least Square Discriminant Analysis
$ReLU$	Rectified Linear Unit
$RMSProp$	Root Mean Squared Propagation
SBC	Single-board Computer
V	Volts
W	Watts
WV	Wavebands

Acronyms and abbreviations

CNN	Convolutional Neural Network
$HDMI$	High-Definition Multimedia Interface
$NoIR$	No Infrared Filter
RSA	Republic of South Africa
$SAGL$	South African Grain Laboratory
USB	Universal Serial Bus
WM	White Maize
YM	Yellow Maize

Chapter 1

Introduction

1.1. Background

Maize is a staple food all over the world and accounts for a significant share of South Africa's agricultural exports. It is used to produce syrups, alcohols and breakfast cereals. It remains the most important feed for livestock in the country and for many of the people forms part of their staple diet. Attaining the highest quality in the industry is essential, as it ensures the most favourable market value for grain. Additionally knowing the quality of the maize is also beneficial for farmers, since this tells them how to properly store and preserve the batch.

The typical grading system to determine defective kernels is as follows. According to the South African department of agriculture [1], A 150g sample, or about 380 kernels is taken from the maize consignment and this sample is examined by hand by a qualified inspector. The inspector will sum the total number of defective kernels and then assign an overall grade to the batch. The traditional methods are slow and prone to inaccuracies. The project goal is thus to test alternative ways in which to grade defective maize kernels.

1.2. Problem Statement

This project aims to address the above problem by building camera system that captures images of both sides of the grain, and then uses a significant amount of data to train a convolutional neural network model which will be used to test on unseen kernels. The model results are then analyzed and compared with other alternative technologies.

1.3. Objectives

The end goal of the project will be that the system performs the following tasks:

1. Having a camera system that will be able to take a top and bottom image of a 150g sample with high quality.
2. Perform image processing to properly separate each of the kernels.
3. Use the labeled kernels to train a CNN model.
4. Pass the unlabeled kernels through the model to classify them correctly.
5. Optimize the system so that the process is relatively fast. So that multiple batches can be analyzed quickly.
6. Test and compare results with other methods of grading technologies.
7. Test model on unseen batches and evaluate results.

1.4. Project Scope

Due to limited time and resources, the project has some degree of constraints. These constraints are as follows:

- Maize kernels have many defects and unfortunately not all of the defects are accounted for. This project only examines a select few.
- Foreign material will not be detected and kernels won't be measured to the mm precision.
- A top and bottom side of the kernel will be examined. And thus not the entirety of the kernel, (the edges wont be taken into account)
- Chemical properties and interiors of kernel not be examined. Moisture will also not be accounted for.
- There are a limited supply of defective kernels,so the sample size is not as large as one would want to ideally be.
- We are limited to one camera which prevents the system from being fully automated
- The camera system will be designed bulky and not easily portable
- Other types of cameras such as NoIR, hyperspectral etc. will unfortunately not be tested.

1.5. Structure of Report

- Chapter 1 - Overview of the topic and problem at hand, as well as project objectives and an overview of the solution design.
- Chapter 2 - Providing background knowledge required in developing a solution. Explanation of other existing solutions to problem.
- Chapter 3 - How the system hardware is designed and why certain choices have been chosen above others in development.
- Chapter 4 - How the software operates and the specific details of each process that runs.
- Chapter 5 - Results in performance of algorithms, run time, training sets and testing batches.
- Chapter 6 - Conclusion and overview of the system.

Chapter 2

Literature review

Chapter 2 presents a extensive literature review, outlining the fundamental concepts of the project. It begins by giving an overview of how maize is normally classified in the industry, including the types of defects that will be dealt with. The chapter then dives into previous technologies and studies related to this classification problem. Finally some key software background concepts are briefly explained.

2.1. Grading of maize batches

According to the South African Department of Agriculture, white maize is categorized into 3 classes: WM1, WM2, WM3, with WM1 being the most pure and WM3 the least, [1]. To grade the maize, a percentage of deviations, discoloring, and foreign materials in a sample is assessed according to the RSA grading regulations. The following table is a snippet from the Department of Agriculture report which shows the percentages used to place the sample of maize into each of the above mentioned categories.

Table 2.1: "Regulations relating to the Grading, Packing and Marking of Maize intended for sale in the Republic of South Africa as published in the Government Gazette No. 32190, Regulation No. R. 473 of 8 May 2009." [1]

Deviation		**Maximum possible deviation (%)**		
		WM1	**WM2**	**WM3**
#1	Foreign matter	0.3	0.5	0.75
#2	Defective maize kernels, above and below the 6.35 mm round-hole sieve	7	13	30
#3	Other colour maize kernels	3	6	10
#4	Deviations referred to in items 1, 2, 3 collectively: Provided that the deviations are individually within the specified limits	8	16	30
#5	Pinked maize kernels	12	12	12

Although foreign materials and pink maize does affect the grade of the maize, for the scope of the project we will be dealing and examining point 2,3 and 4 only. The Table 2.2 illustrates the types of defective kernels one can expect to find during the grading process.

Table 2.2: Defective kernel types according to the SAGL grading Report [2]

Defective kernel types:
Heat Damaged
Frost / Water Damaged
Sprouted
Fusarium (Fungal infection)
Pest Damage
Immature
Soiled (smoke, fire, etc.)
Matter which can pass through 6.35mm sieve

Depending on seasons and harvests, it is quite difficult to obtain enough samples of every single defect type. This year in particular, water and frost damage, as well as soiled, sprouted and immature kernels were particularly hard to find. Thanks to the SAGL we managed to obtain enough samples of the following kernels and deviations which we will be used throughout the project.

Table 2.3: Available categories provided by SAGL

Available categories		**Approximate sample size (# kernels)**
(a)	White Maize	1566
(b)	Discolored White Maize	1481
(c)	Yellow Maize	1500
(d)	Insect Damaged White Maize	969
(e)	Heat Damaged White Maize	1157
(f)	Fusarium	786
(g)	Severe Fusarium	1257

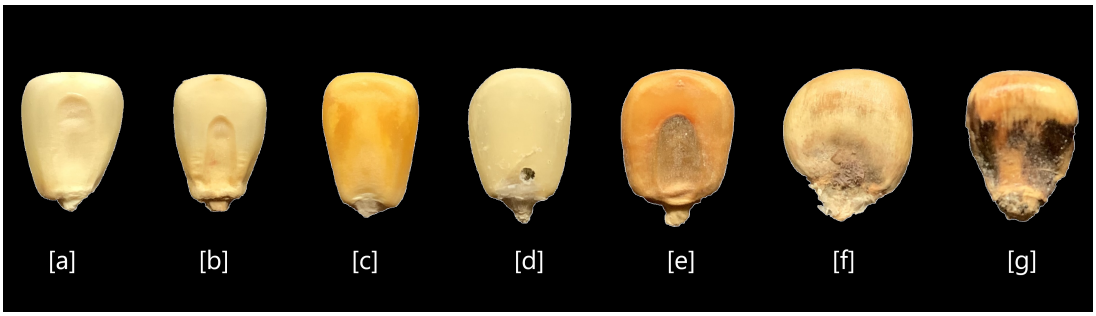


Figure 2.1: Visual representation of each category

2.2. Previous related studies

There have been a few approaches in solving this problem of classification, with the two main approaches being, using multi-spectral and hyper-spectral cameras. This allows one to detect all wavelengths of light in a specific range. These methods also involved using PCA coupled with PLS-DA to classify the kernels. The main difference between hyper-spectral and multi-spectral imaging comes down to wavelength resolution. Multi-spectral collects spectral data in a few specified bands, whereas hyper-spectral imaging collects a more detailed continuous spectrum using hundreds of narrow bands. The studies using these systems are documented in [3] and [4] respectively. These studies were conducted on behalf of the Department of Food Science at Stellenbosch University.

The main issue with these solutions is that the price of these cameras are quite high. The typical cost depends on the exact configuration setup, but typical solutions with added equipment are in the range of 45 000 EUR to 65 000 EUR which is quite expensive even by industry standards. Processing images across a wide spectrum is also computationally intensive. Additionally setting up the cameras and taking images is a lengthy process.

In the multi-spectral case the camera used 19 wavelengths ranging from 375 to 970 nm. In the hyper-spectral study, a comparison was made using the full spectrum model and three other images using the wavebands; 48, 21 and 13 respectively. The accuracies of the two studies for each of the wavebands are shown below in Table 2.4. As one can see, as the wavebands decrease, so does the relevant classification accuracy. These accuracies listed will be used as a benchmark to compare with the results obtained using our trained model. Following below is the Table 2.5 summarizing the benchmark accuracies that will be used to test our model against.

Table 2.4: Accuracies obtained in previous studies, where (HI - hyperspectural imaging), (MI multispectural imaging), (WV - wavebands),

Class	Accuracy (%)				
	HI (288 WV)	HI (48 WV)	HI (21 WV)	HI (13 WV)	MI
Sound white maize	**88.3**	**76.7**	**78.3**	**63.3**	-
Pinked white maize	**83.3**	**75.0**	**78.3**	**80.0**	**91.43**
Yellow maize	**75.0**	**60.0**	**56.7**	**48.3**	**100**
Defects (average)	**93.3**	**86.7**	**83.2**	**81.0**	**92.86**
- Screenings	93.3	78.3	78.3	73.3	100
- Fusarium	100	88.3	86.7	83.3	88.57
- Diplodia	90.0	90.0	71.7	78.3	82.86
- Heat	95.0	76.7	76.7	61.7	100
- Water	90.0	88.3	81.7	76.7	97.14
- Frost	96.7	90.0	86.7	85.0	-
- Pest	95.8	93.3	82.5	88.3	88.57
- Sprouted	86.7	90.0	86.7	93.3	-
- Immature	92.6	85.2	98.1	88.9	-
Foreign materials (average)	**100**	**96.66**	**95.66**	**94.32**	**100**
- Soy	100	100	100	100	100
- Sorghum	100	98.3	100	98.3	100
- Sunflower	100	100	100	100	100
- Wheat	100	85.0	83.3	83.3	100
- Plant	100	100	95.0	90.0	100

Available categories	**Benchmark Accuracy %**	
	Hyperspectral Camera	Multispectral Camera
Sound White Maize	88.3	-
Discolored White Maize	-	-
Yellow Maize	75.0	100
Insect Damaged	95.8	88.57
Heat Damaged	95.0	100
Fusarium	100	88.57

Table 2.5: Benchmark accuracy from previous studies

2.3. Background concepts

This section is aimed to introduce the reader to perhaps foreign concepts that will be used in the project. The three sections that will be discussed are image registration, image segmentation and convolution neural networks. All which play important roles in developing a solution to the problem.

2.3.1. Image registration

Image registration is a process which uses feature matching to align two images of the same object that are very similar, but not necessarily identical. The use of this technique is later shown in section 4 where it is used to align top and bottom images of maize kernels. While we will explore the main concepts of image registration, we will not dive deeply into the exact details of each step. The process is briefly explained in the following steps:

Feature Detection:

The first step involves feature detection of binary images, for which we use the SIFT (Scale-Invariant Feature Transform). Given a input image, a few copies of the original image is made, each with a certain amount of Gaussian blur and scaling. These images can be thought of as being stacked on top of each other, such as a ”pyramid”. At each level of the pyramid we calculate the difference in Gaussian blur (DoG). This is done by subtracting the top image from the bottom, or essentially a blurred version of the image from a slightly more blurred one. What this does is highlights certain key points in a image where spots changes significantly. Once the extreme key points have been found in both images they then need to go through a filtering process called key point localization. This involves removing elements like edge points which are sensitive to DoG and overly bright points.

Orientation Identification The next step involves determining the orientation of each of the key points. This is done looking at the magnitude of the gradients of the surrounding points. Each pixel's direction is calculated according to this region and is then placed in one of the bins of a orientation histogram. The histogram consists of 36 bins covering 360 degrees. When all the directions of the surrounding area is calculated then the highest bin will be considered as the final ”orientation” of the key point.

Key point Matching:

After finding key points, the next step is to find matches from those in the source and target image. A distance metric between each pair is calculated using Euclidean distance. The goal is to find the key point that most closely corresponds to its corresponding key-point in the other image. The matches found are then filtered, choosing only good matches (closest ones). Using these good matches and the orientations the source image is then finally warped to fit onto the target image. A more visual explanation can be found here: [5]

Figure 2.2: Example of two satellite images being aligned using image registration. Reproduced from [6].

2.3.2. Image segmentation

Watershedding

Watershedding segmentation is a technique that is used to separate objects in an image. Simple contour detection works well when trying to separate objects that are relatively well distanced from each other. However in this case, when taking pictures of hundreds of grain, there is bound to be kernels that are touching each other, sometimes even overlapping. The best way to properly separate and extract these types of images is by means of the watershedding algorithm.

The basic principle of watershedding is to take a gray scale image and then consider the image as topological surface. This means that high pixel values will be seen as ”high surface” regions and low values as ”low surface” regions. If one were to imagine ’flooding’ the image with water, where the water sprouts from each of the low lying points, then where the water meets from different sources, a ”dam wall” is built. At these points a contour line is then drawn to separate each of the objects. This can be best shown in a 1D topographic Figure 2.3

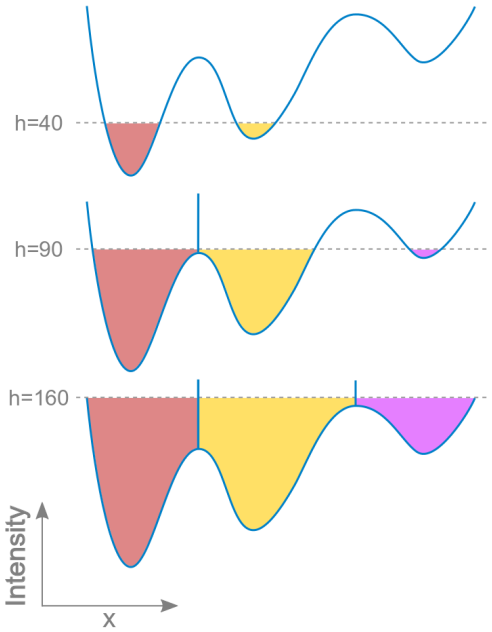


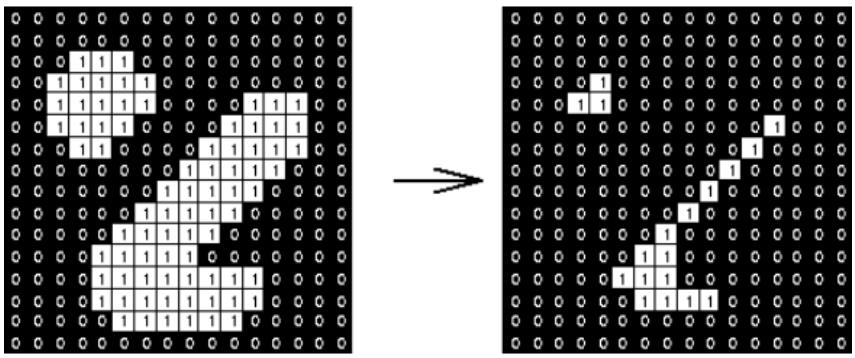
Figure 2.3: Watershedding in the 1D space. Image reporduced from [7].

Eroding

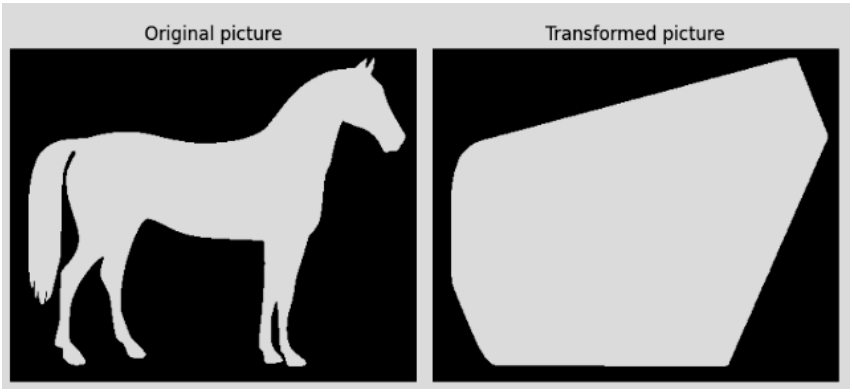
Erosion is used to strip away extrusions of a binary image. This aids in image segmentation. The way in which it works is by choosing a square kernel of size say PxP. This kernel is filled with 1's and is swept from the left to the right over the image for each row of pixels. For each pixel the kernel sweeps over, the program checks if the P adjacent pixels match with the kernel. If this is the case then the evaluated pixel is set to 1. Otherwise the pixel is set to 0. Changing the size of the value P will affect how much of the image is essentially striped away. One needs to be careful not to make the size of the kernel too large as then you lose too much information on your image. An example of how this is done is shown in Figure 2.4a.

Convex hull algorithm

After applying the watershedding algorithm, we are left with a mask image. Sometimes some of the kernel are cut through the middle by the algorithm and not properly segmented. To remedy this the convex hull algorithm is used to store the entirety of the shape of the kernel. The way in which this works is that the algorithm tries to create the smallest convex polygon that surrounds a white image. In the case of kernels this works pretty well since we assume that the shape will almost always be convex. A visual example is shown in Figure 2.4b



(a) Erosion using a 3x3 kernel. Image reproduced from [8]



(b) Convex hulling. Image reproduced from [9]

Figure 2.4: Visual examples of erosion and convex hulling on binary images

2.3.3. CNN

A CNN is a type of neural network that follows a Feed-Forward Neural Network Architecture which consists of multiple layers. We will not be going through all the layers in detail, however the basic concepts of each layer will be mentioned. The following is a rough overview of how the overall CNN architecture looks like:

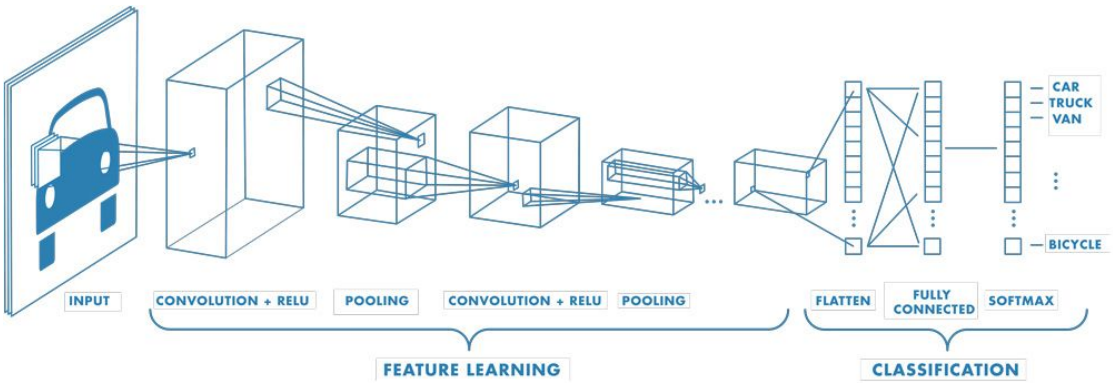


Figure 2.5: Typical CNN architecture. Reproduced from [10]

Convolutional Layer

Even though the network is referred to a convolutional neural network, it uses a cross-correlation operation for training.

$$G(x[n], h[n]) = \sum_{i=-\infty}^{\infty} x[i] \cdot h[i-n] \tag{2.1}$$

The cross-correlation equation calculates the similarity between two sequences. The use of cross-correlation comes to play when using filters. A filter is a small matrix of size PxP which "shifts" across the original image from left to right, top to bottom, forming a activation map. The more a patch of the original image "looks" like the filter, the higher the value in the activation map will be. The process allows for detection of meaningful information, which is vital for classifying images. Each filter has a specific set of weights that are updated during the training process. Figure 2.6 visually illustrates this convolution process.

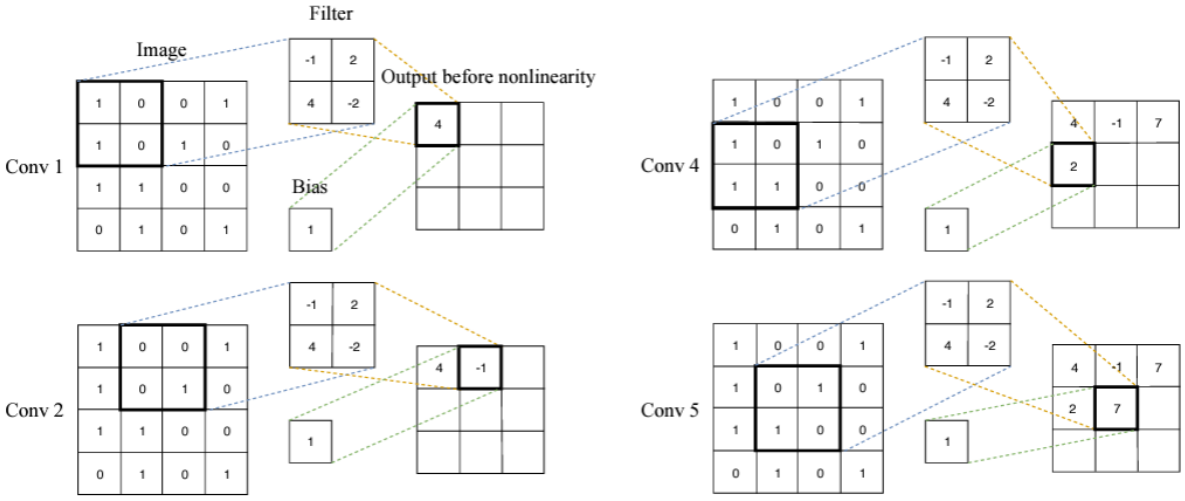


Figure 2.6: A simple visual example of convolution using a 2x2 filter. Image reproduced from [11]

Activation Layer

After the filtering step, the activation map is initially a linear layer. To add non linearity, which allows the network to learn complex patterns, a activation function is added. The typical activation function used is ReLU. Essentially ReLU sets all values that are less than 0 to 0, otherwise anything above remains unchanged. This activation function also helps address the vanishing gradient problem in training.

$$f(x) = max(0, x) \tag{2.2}$$

Max Pooling Layer

To minimize the spatial volume of the input image, a max pooling layer is used. Reducing the spatial size means that the images requires less weights and thus the eventual training is less computationally expensive. Max pooling is done by calculating the maximum value of each window frame and then storing this with a new output dimension.

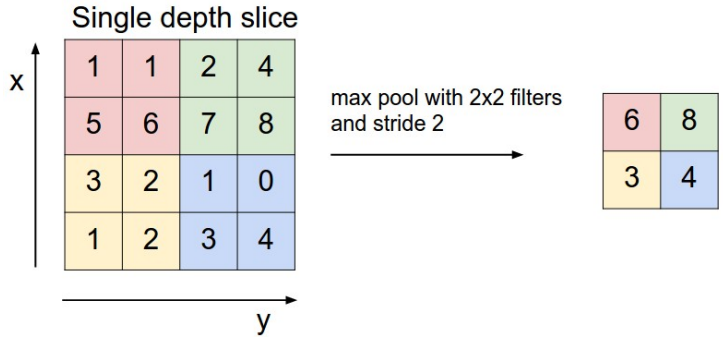


Figure 2.7: A simple Pooling example using a 2x2 filter and stride of 2. Image reproduced from [12].

Fully-Connected Layer

Finally, the output of the hidden layers is converted into a one-dimensional vector using flattening. In this layer, each neuron, with its now flattened weights and biases, is interconnected with each other. After simple multiplication is done with each of these neurons a softmax function is then used to finally classify these features into one of the available classes. The softmax function requires a 1-dimensional input and is thus the reason why the flattening layer is necessary. The classes assigned to the images are then compared to their actual class and the MSE or other loss metric is calculated. We won't go into too much detail but with the use of back-propagation, the system updates the weights of the neurons to decrease this overall loss.

Chapter 3

Hardware design

This chapter gives a overview of how the system works. It dives into the hardware components and the architectural design of the camera box system, as well as the configurations and explanations for various design choices made.

3.1. Components

3.1.1. Raspberry Pi 4 Model B

When a image is taken it will have to undergo intensive processing such as image registration, water shedding, etc. In Chapter 4 we go into more detail regarding this. It is also required that the processing should be fast enough for high quality images. A basic single-board computer, such as the Raspberry Pi Zero with its limited 512MB memory and 1GHz processor, would thus not be able to handle the tasks.

The Raspberry Pi Model 4 B is chosen as it is the newest in range. It offers a improved performance with a 1.5GHz processor and 2GB of memory. While some of additional features are essentially useless in this setup, such as the audio jack. It is still a viable choice due to its processing capabilities. It also has two micro HDMI ports, which allows us to visualize the camera feed on a monitor, which is very convenient. The USB ports allow us to make use of wireless mouse and keyboard so that one can control the device without having to touch it once the camera has been aligned. Finally the SBC also comes with SD Card storage which is useful when storing large amounts of images. More specs of Raspberry Pi's can be found at, Zero [13], Model 4 [14].

3.1.2. Raspberry Camera Module 3

The standard Pi module camera 3 has been chosen. The module 3 is 12MP, with the highest square resolution of 2592x2592. The camera also has auto focus which works well as the cameras position is consistently adjusted for proper alignment.

3.1.3. Camera holder

To keep the camera module and the SBC attached to each other and held in one place, a simple 3D printed attachment had been made. **********Figure **??** in the appendix shows how the attachment looks like.

3.2. Designs

3.2.1. Overhead tripod

The initial design was quite simplistic. Consisting of a open box to place the kernels in and a tripod holding the SBC, camera and light source. Although simple, there are a few fundamental flaws with this setup. Firstly one would like to take images in a consistent environment. In this case outside light will have a significant affect on the images taken. The other obvious flaw is that the images taken show only one side of the kernel. Essentially you are losing half of the information. This design was scrapped and an alternative solution had been developed.

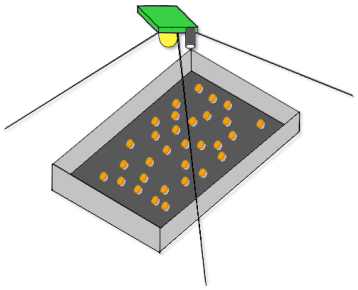


Figure 3.1: Initial tripod setup

3.2.2. Photo booth

The alternative design resembles a somewhat ‘photo booth‘ style box. This consists of a wooden box approximately a meter in height with a opening on the side in which a batch of maize is slid into. This creates a consistent environment for images to be taken. The process for capturing photos is as follows:

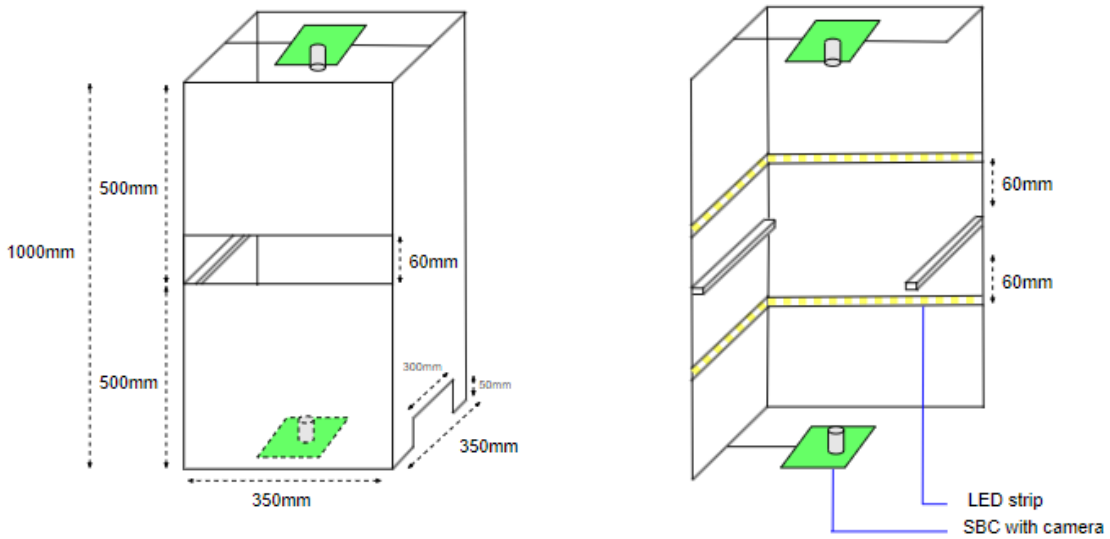
1. Activate LEDs and position the camera in place.
2. Throw 150g batch into a smaller container, make sure the kernels are well spaced.
3. Slide a black covering within the box.
4. Slide kernel container above the covering.
5. Take the top photo.
6. Slide the covering from out the bottom and place it on top of the kernel container.
7. Take the bottom image.

Ideally one would want to have two cameras, however due to the scope and budget we have only one camera that must be moved from the top to the bottom between each set of images taken.

3.3. Photo-booth design choices

3.3.1. Box

The specific dimensions of the box are show below in Figure 3.2. The inside view shows the built in internal slides which allow the container to be slid into. The green rectangles represent the SBC with the camera and the yellow line represent the LED strip.



(a) Outside view **(b)** Inside view

Figure 3.2: Photo-booth design

3.3.2. Kernel container

Size

The size of the platform had to be carefully considered, since we don't want the kernels to stack on top of each other, blocking the view. Instead we aimed to choose the size, so that all the particles can easily fit and lie flat on the surface. The platform's size also couldn't be unpractical large. Too much wasted space would mean that the camera would have to be placed higher, which would decrease the overall the image quality. The trick lies in minimising height while maximising space.

After weight tests were conducted, it was determined that a 150g sample contains approximately 375 kernels, with each maize kernel weighing approximately 0.4g. From measuring multiple kernels was also assumed with a high degree of confidence that the average surface area of a kernel is approximately 160 mm^2. The minimum total surface area needed to hold these kernels is then given by:

$$\text{Total Surface Area} = 375 \times 160\,\text{mm}^2 = 60,000\,\text{mm}^2$$

To create 50% more space between the kernels, we use a 30 cm × 30 cm container, it has a total surface area of $90,000\,\text{mm}^2$. Which is sufficient to ensure reasonably well spread of the kernels.

Glass

The kernel container is equipped with a 32cm x 32cm transparent glass sheet, which allows visibility from both sides. However, using glass does come with some challenges. The pictures taken through the glass are not as sharp as those taken directly. Refer to Chapter 5 for direct comparison. Another drawback, is that when kernels are placed on the glass repeatedly, after a while the glass gets dirty and requires to be cleaned. The glass also produces reflections when LEDs shine on the surface. To counter this the LEDs are placed at a specific height and location, as detailed in the next section. There are other alternatives like plexiglass, but the photo quality is not the same. Things like anti-glare glass does exist, which doesn't completely take away the glare but reduce it significantly. The price of such glass is quite expensive. Nevertheless, for the purposes of this project, the standard glass sheet performed well and proved satisfactory.

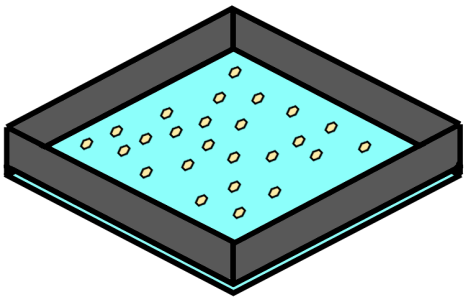


Figure 3.3: A 32cm x 32cm maize container with glass platform, which slides into larger box.

3.3.3. Camera height

According to the Raspberry Pi Camera Module 3 data sheet [15] the standard camera has a 75° diagonal range of view. The design is that of a square block and thus requires that our pictures be take in the square format. Consequently we are limited by the vertical view range which according to the data sheet [15] is 41°. Using simple trigonometry one can calculate the optimal height in which the camera should be placed. The final height equates to 430 mm.

$$\text{Camera Height (mm)} = 160\tan(69.5°) = 427.939139\,\text{mm} \approx 430\,\text{mm} \tag{3.1}$$

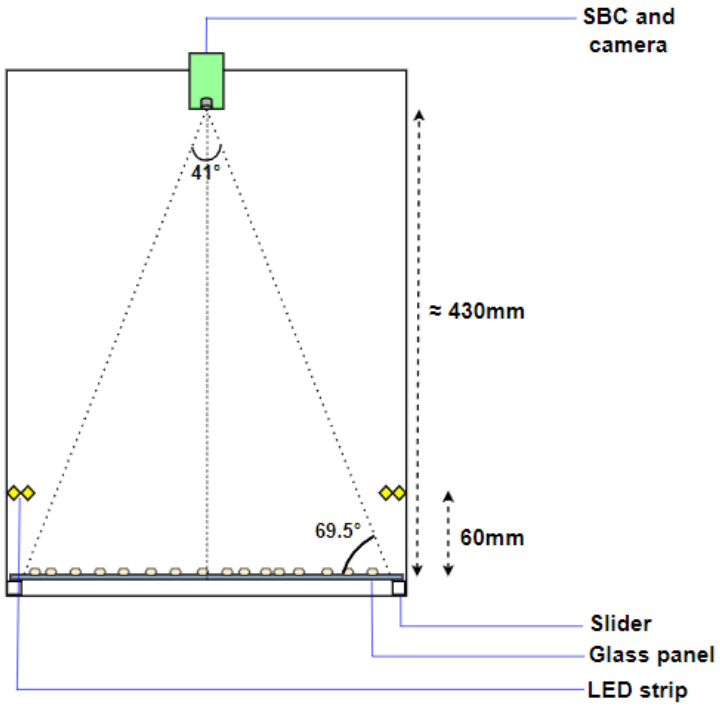


Figure 3.4

Figure 3.5: Side view of box,

3.3.4. Lights

To ensure that the kernels are all equally well lit, an LED strip has been placed all around the inside of the box. The LEDs are positioned 6cm in height from the surface of the glass on both sides. Placing the LEDs this way almost entirely removes any potential shadows. By placing the LEDs on the sides of the box and not instead at the top is to counter the effects of reflections of the glass. Essentially, the reflected area of the glass will only be confined to the edges and nowhere else.

To deal with the slight amounts of reflections on the edges we simply place a covering on the edges of the glass. Essentially 'cropping out' a bit our image container space. In this case isolation tape was used and seemed to work reasonably fine. There is however still room for improvement as shown in Chapter 5.

The LED strip is connected in series to 9V battery and has a power and voltage rating of 6W and 12V respectively. Furthermore, the Datasheet [16], provided no further information about the internal resistance. An assumption had thus been made that the LED had internal resistance of about 10Ω. After running multiple tests, the resistance providing optimal brightness was a 470Ω resistor.

For the bottom circuit, it is important to remember that the glass absorbs light. So in order to achieve the same lighting for the images taken through the glass, the LEDs had

needed to shine brighter. This required a lower resistor value to be used in the circuit. Based on the assumption that 3mm glass absorbs approximately 10% of light it was calculated that 420Ω resistor would be ideal for the bottom circuit. The following set of equations show the methodology used in getting this value. We assume brightness of the LED is directly proportionate to the current flowing through circuit.

$$\begin{aligned}
\text{Total Resistance in Top circuit} &: R_{\text{Top}} = 10\Omega + 470\Omega = 480\Omega \\
\text{Current for Top} &: I_{\text{Top}} = \frac{9V}{480\Omega} = 18.75 \text{ mA} \\
\text{Current for Bottom} &: I_{\text{Bottom}} = 1.1 \times I_{\text{Top}} = 20.625 \text{ mA} \\
\text{Total Resistance for Bottom} &: R_{\text{Bottom}} = \frac{9V}{20.625\text{mA}} = 436.36\Omega \\
\text{Bottom Resistor Value} &: R_{\text{value}} = R_{\text{Bottom}} - 10\Omega = 426.36\Omega
\end{aligned}$$

3.3.5. Sliding sheets

To create a solid background for the kernels a black cardboard sheet is slid into the top and bottom of the box. Cardboard was used as it is rigid enough to stay firm in one place without having creases, and thin enough to slide in and out underneath the container without moving the kernels. The following diagram shows how the sheets are slid out from the bottom and on top of the container after each photo is taken.

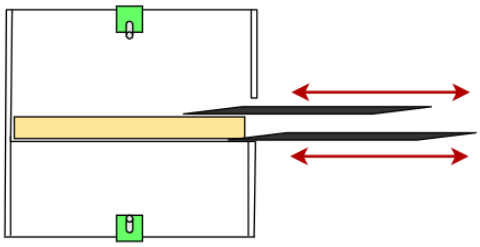


Figure 3.6: Side view, showing the sliding of the black sheets. Note: Diagram is not to scale.

Images of the entire setup can be found in the Appendix **??** **. The next chapter will focus on the image processing software, detailing how system processes the top and bottom images once they have been captured by the hardware system above.

Chapter 4

Software and System Design

This chapter separates the software design into three sections, namely: Processing, Training and Evaluation. Once images have been captured, they are initially subjected to the processing stage. Thereafter, the images are used either for training or evaluation.

4.1. Processing

Arguably the most complex and fundamentally important section of the project. This stage is further divided into 5 parts. The Figure 4.1 below highlights each stage and the order in which they happen.

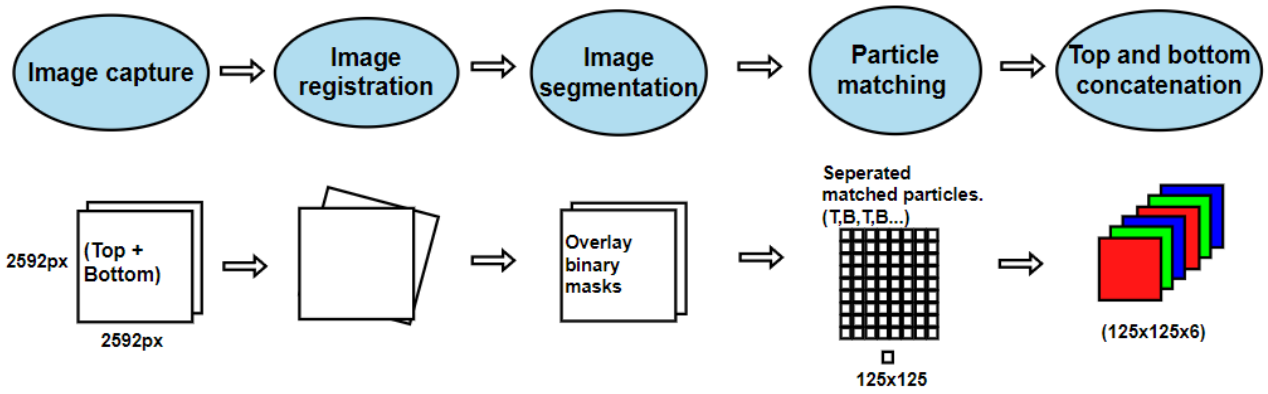


Figure 4.1: Processing flow diagram.

4.1.1. Image capture

A Python script, with the PiCamera2 library is used to capture the images. In the script the auto focus is enabled and the image resolution is set to the maximum square format of 2952x2952 pixels. A preview display is shown to the connecting monitor. This allows the user to fine tune and adjust the camera's positioning as needed.

4.1.2. Image registration

The goal after taking the top and bottom images is to have them perfectly aligned so that kernels can be identified. Being limited to one camera, it forces one to have to constantly

move it between the top and bottom sections of the box. This movement causes inaccurate placement which induces some skewness between the two images. To combat this, image registration is implemented in software. A python script using the OpenCV library is used to implement this functionality. The Figure **** **??** in appendix shows the before and after effects this has on our images.

4.1.3. Image segmentation

Once the images are correctly aligned the next step is to extract the kernels out of the image and save them as there own PNG files. We will dive deep into each step process of this image segmentation stage. For most of these steps, the Java plugin of ImageJ Fiji is used.

Color thresholding (b)

The first step in the image segmentation process is to distinguish all objects with the background. A black cardboard background works well here as the particles stand out. To choose the values which capture the particles and not the background, extensive 'eyeballing' tests were done to see if the range covers all the particles entirely. The lightest particle are the WM and the darkest are Fusarium. ImageJ Fiji uses HSB color space and the final values were chosen as: H = 0-255; S = 0-255; B = 35-255. Where, the Hue value represents all the colors on the color wheel. The Saturation value represents all the types of shades of gray. The Brightness values represents all colors that are above 14% brightness on the standard scale. (0 black and 100% white). When this thresholding takes place a red mask is drawn over the pixels that fall into this range. This red mask is shown in Figure 4.2b

Binary mask and eroding (c)

The next step involves converting the image into a binary format. This is done by pixelating the background as 0 and the selected range as 1. Since many of the kernels are packed closely to each other, slight erosion is applied to create space. Some of the information of the particle is temporary lost by doing this, but this step significantly aids the water shedding process.

Water-shedding algorithm (d)

A initial water-shedding algorithm runs and draws contour lines between each of the kernels. The fundamentals of this algorithm is explained in Chapter 2. Although this algorithm correctly separates most of the kernels, some are still not correctly separated. In some cases kernels are cut through the middle by the algorithm, refer to Figure 4.2d. To remedy this, the Convex Hull algorithm is introduced as a subsequent step.

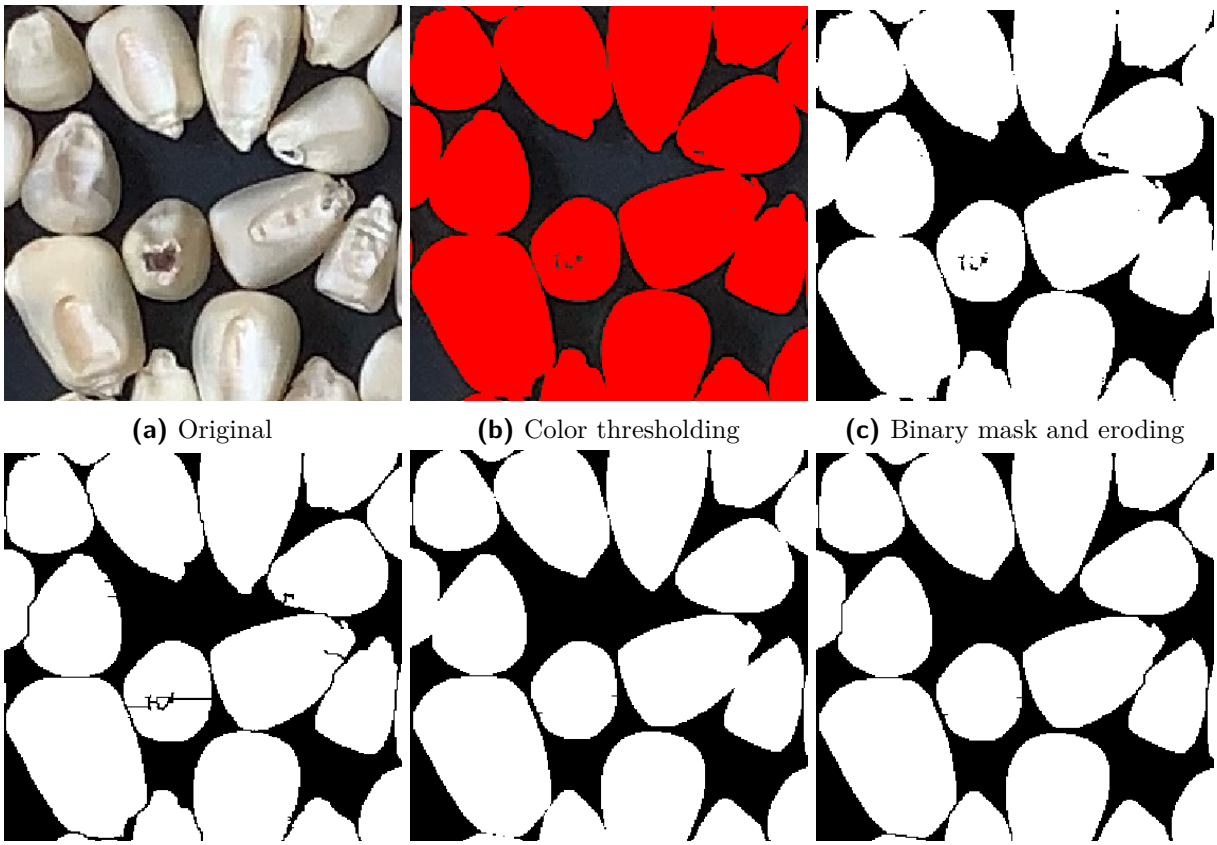
Convex hull algorithm (e)

This algorithm essentially aids in the separation by filling in holes in the kernels that were incorrectly made by the water-shedding algorithm. What one has after this is particles that are mostly correctly selected and also not as tightly overlapping. This is because there is still many initial contour lines separating them.

Final water shedding, contour detection (f)

Now that the particles are in a cleaner format, one can run the water-shedding algorithm once more. We are then left with a final mask which is then overlaid onto the original image. Using OpenCV's contour detection functionality we can get contour coordinates which are used to extract the particles. However before saving the images, they each go through the particle matching process which is explained in next step.

An example of how the image segmentation process works on a small subset of kernels. Take note of the circular kernel in the middle, which is initially incorrectly separated but then rectified by the script.



(a) Original **(b)** Color thresholding **(c)** Binary mask and eroding

(d) Water shedding **(e)** Convex hull **(f)** Final water shedding step

Figure 4.2: Image segmentation steps

4.1.4. Particle matching

Implementing these image segmentation steps doesn't always yield perfectly results. Sometimes small particles, glares and incorrect splitting happens. While missing a few kernels isn't the end of the world, what we don't want is to feed the model with incorrect data. Thus two checks have been added to decrease the chance of this happening.

After running some manual tests, is has been determined that the smallest kernel will not have an area smaller that 2200 pixels. A size check using this threshold is added to eliminate all small unwanted particles.

The next step involves using a matching principle for the top and bottom image. What this entails is as follows: The contours are used to create a square cutout of each of the particles. The centres of these square images are then determined as coordinates. The top side center is then compared to the bottom side center. If the euclidean distance measured between the top and bottom centres are less than 10 pixels then a match is found and the kernel pair is saved. A length of 10 pixels may seem like little, but after image registration the kernels are almost exactly aligned. An assumption is made that if the image segmentation process runs on two images, then normally the same mistake won't occur on both sides of an image. This step effectively filters out almost all discrepancies.

Figure 4.3: A snippet showing how pairs of kernel pairs are saved. Where "#_T" is top and "#_B" bottom

4.1.5. Dual image concatenation

After a pair have been correctly identified, the final step is to convert the images into a format suitable for the neural network. To do this one needs to somehow combine the two images. The decision was made to have the input space being 6 layers deep, consisting of two images each with dimension 125x125 pixels and in RGB format. By using a python script with OpenCV functionality we perform horizontal axis concatenation to create the final input shape of (125x125x6).

4.2. Training

This section provides detail on the data we were provided, how the model had been trained and the decisions made during this phase.

4.2.1. Data balance and split

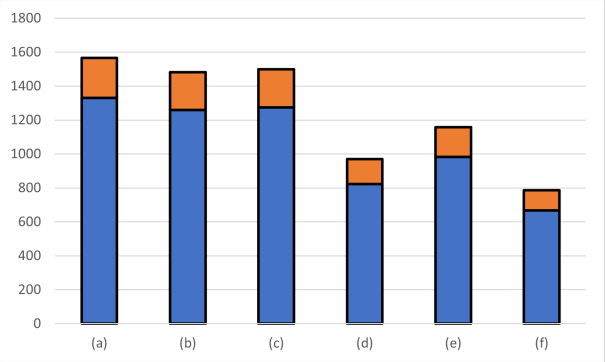
We initially have 7 categories of maize, among which there is normal and severe fusarium as labeled by the SAGL. It was decided to remove the “severe“ category in training process. The reason for this is due to the fact that fusarium in practice of grading isn’t classified into separate classes. Additionally, according to self inspection, (keep in mind I am no expert in grading) severe fusarium and normal fusarium in the training batches seemed to overlap. Another worry is that by having both classes, potential bias would be induced. Since these defects don’t occur that often in the real world it was important not create an overemphasis on these defects in the training data set.

Furthermore it was decided to split the data into split for training and validation sets with 85:15 ratio respectively. There is no test set allocated since the SAGL has provided separate graded batches on which final testing is done.

Table 4.1: Amount of kernels of each category, as provided by SAGL

Training Data		
Category		**# Kernels**
(a)	White Maize	1566
(b)	Discolored White Maize	1481
(c)	Yellow Maize	1500
(d)	Insect Damaged	969
(e)	Heat Damaged	1157
(f)	Fusarium	786
Total		**7459**

Figure 4.4: Visualisation of class balance



4.2.2. Model architecture

A well designed structure that has been used in the past for image recognition is LeNet-5. This architecture was used as a baseline to build our model off. LeNet-5 used input images

of size 32x32 with 3 convolutional layers each with kernels of size 5x5. An good visual example can be found at [17]. After numerous experimentations with different structures and parameters the model had been finalised with the structure shown in Table 4.2.

The structure consists of 5 convolutional layers as well as a global-pooling and dropout layer. Smaller kernels of size 3x3 were used as they performed better than larger kernels. This could be due to the fact that the larger kernels detected too much unnecessary details in each image.

Table 4.2: Model Architecture

Layer (type)	**Output Shape**	**# Parameters**
`Original Image`	(125, 125, 6)	0
`Conv2D`	(125, 125, 16)	880
`BatchNormalization`	(...)	64
`MaxPooling2D`	(62, 62, 16)	0
`Conv2D`	(62, 62, 32)	4640
`BatchNormalization`	(...)	128
`MaxPooling2D`	(31, 31, 32)	0
`Conv2D`	(31, 31, 64)	18496
`BatchNormalization`	(...)	256
`MaxPooling2D`	(15, 15, 64)	0
`Conv2D`	(15, 15, 128)	73856
`BatchNormalization`	(...)	512
`MaxPooling2D`	(7, 7, 128)	0
`Conv2D`	(7, 7, 256)	295168
`BatchNormalization`	(...)	1024
`MaxPooling2D`	(3, 3, 256)	0
`GlobalAveragePooling2D`	(256)	0
`Dense`	(128)	32896
`Dropout`	(50%)	0
`Dense`	(6)	774
Total params		428694 (1.64 MB)
Trainable params		427702 (1.63 MB)
Non-trainable params		992 (3.88 KB)

Loss function

The loss function used in training is the Sparse Categorical Cross-Entropy loss. This allowed us to provide labelled data of type [0,1,2,3,4,5] instead of having to one hot encode all of our data.

$$L(y, \hat{y}) = -\sum_{i=0}^{5} y_i \log(\hat{y}_i)$$

Optimizer

For the training process the Adam optimiser was chosen. The optimizer is popular since it combines two other optimizers namely; RMSprop and Momentum. Momentum takes into account the past gradients so that the "jump" rate is more smooth in each iteration. This accelerates the convergence rate and prevents the model from getting stuck at local optima. RMSprop uses a moving average of squared gradients to normalize the gradient itself. This can make the learning process more adaptive and resilient to issues like exploding or vanishing gradients. To calculate the adaptive learning rates for each parameter the and second moments of the gradients are calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Where:

- g_t is the gradient of the loss to the parameters at iteration t
- m_t is the first moment/mean
- v_t is the second moment
- β_1 and β_2 are hyperparameters that control the decay rates of the moment estimates

The parameter updates are calculated as:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where ϵ is a small constant to prevent division by zero.

Batch Normalisation

When the batches are not normalised, the network might update one weight significantly more than the other. What happens then, is that the gradient descent trajectory will fluctuate back and forth along one dimension, resulting in a longer time to reach minima. By implementing batch normalization each weight update becomes more balanced across the dimensions. With a training size of 7459 particles and batch size of 32, we have 233 batches. Each normalized as

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Where:

- $x^{(k)}$ is the activation of the k^{th} input.

- μ_B is the mean of the activations in the batch.
- σ_B^2 is the variance of the activations in the batch.
- ϵ is a small constant added to prevent dividing by zero.

Dropout

A regularization technique used in training is Dropout. At each batch iteration it sets a random fraction of input units to zero. These units are "ignored" in a sense. The whole idea of doing this is to promote independence among neurons and prevent co-adapting. (Making sure that neurons don't become overly reliant on neighboring ones - that is preventing these units to fix up the mistakes of other units). In our case we use a dropout rate of 50% and this is used after the dense layer consisting of 128 neurons. Thus at each iteration/batch 64 neurons are randomly dropped.

4.3. Evaluation

This section describes the testing data provided as well as the output produced by the evaluation script.

4.3.1. Testing data

For the testing phase five batches of graded samples were provide by courtesy of the SAGL. These samples contain a mixture of defective and normal kernels in both yellow and white maize. There was however a slight miscommunication in terms of the type of batches to expect. This lead to the following twp obstacles which were initially unaccounted for in development:

Each of the 5 batches that arrived were 250g in size, which is 100g larger than what the system was designed for. To address this the batches had simply been split in half and required two sets of images to be taken.

As previously mentioned one of the limitations of the system is that small defective kernels below 6.35mm are not accounted for. The batches that arrived had much of these small shavings in. These particles had a significant impact on the image segmentation process and in some cases were the direct cause for a batch to be classified as a lower grade. Furthermore, some of the batches had other types of defective kernels that have not been included in the training stage. Diplodia and YM insect damage are two examples of this. To address this, these particles were manually removed in the testing process. Additionally the weights of the given grades were also adjusted accordingly. This is further explained in Chapter 5

Below are the examples of two batches. The batch on the right contains the small particles below 6.35mm whereas the one on the left does not. Accompanying this are the resulting masks which illustrates the significant impact these shavings have on the segmentation process.

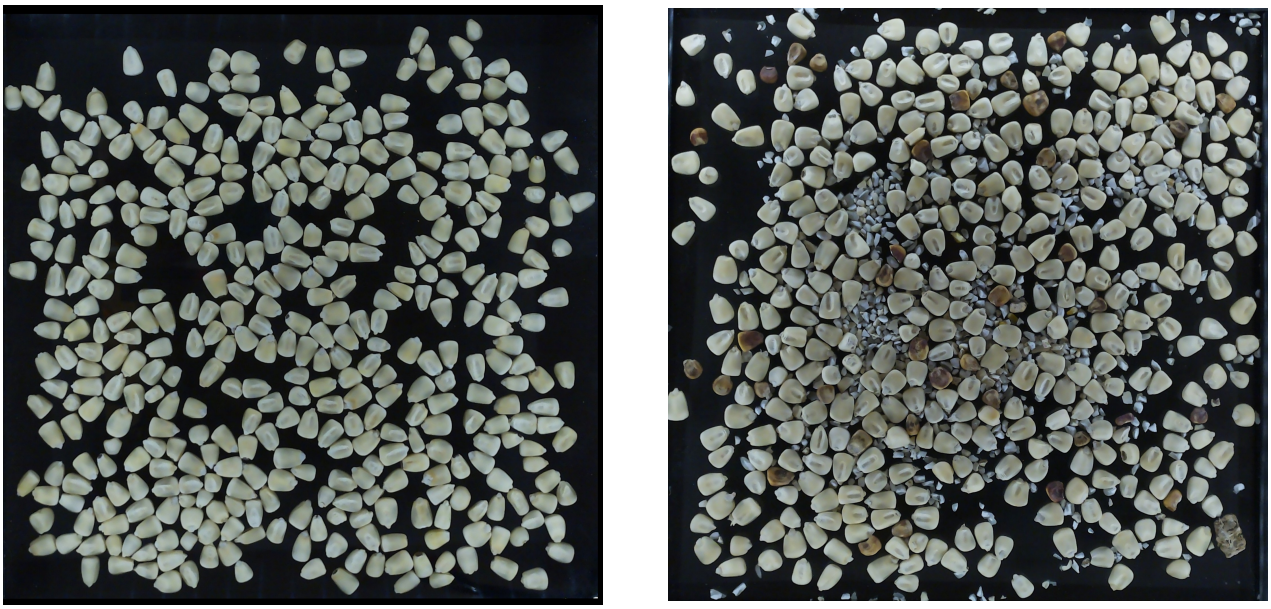


Figure 4.5: Original images of a “clean” and “dirty” batch

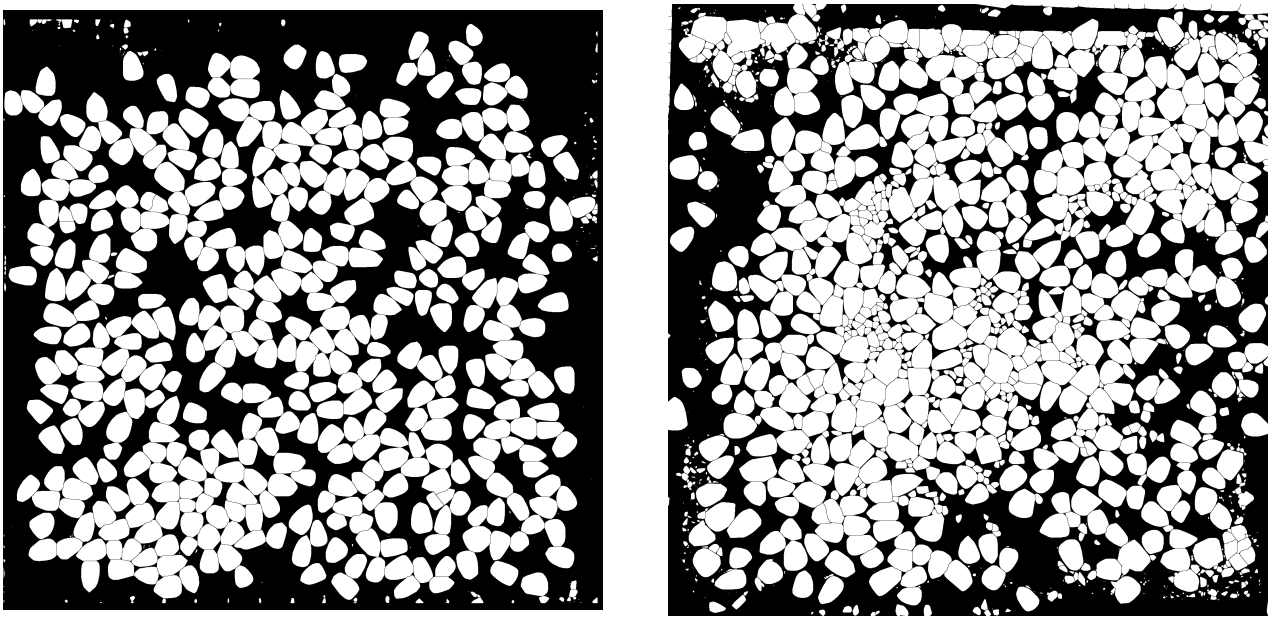


Figure 4.6: Final masks of a “clean” and “dirty” batch

4.3.2. Output

After processing the original images, the evaluation script is run which inputs the particles into the model. A report is then generated by using the regulations Table 2.1 as reference. The report shows the proportions of each class and displays the final grade of the batch in the terminal.

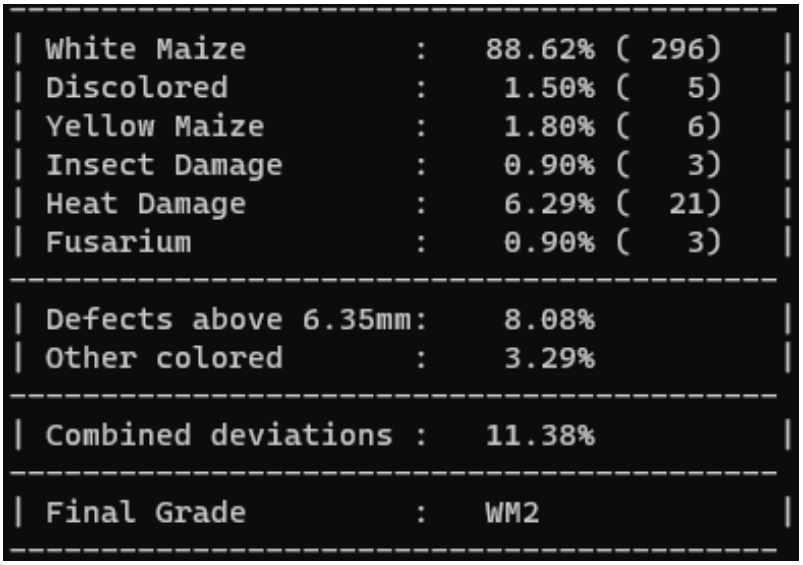


Figure 4.7: Terminal output of a sample batch

In the following chapter we will discuss some of the results obtained by our system.

Chapter 5

Results

This chapter describes the performance of the image extraction process, training, execution time and tests on unseen samples.

5.1. Image extraction

Running image segmentation on batches of kernels without proper kernel dividers will seldom yield perfect results. This is due to the nature of particles being small in size and bunching up. As previously mentioned in the prior chapter, the batches that are evaluated are those with small screenings and shavings removed. Doing image processing with these small particles included are beyond the scope of the project.

In general the script performs reasonably well but never perfectly. On average around 3% of kernels are missed with a varying range of 1% to 5% depending on the spread. This is unfortunate and has a rather big impact on the grading. When the small shavings are included in the batch this number jumps up to 15% to 20%, which essentially makes the entire system obsolete.

The kernels that are missed seem to be completely random. In some cases the algorithm decides to slice the kernels straight though the middle. Figure 5.1 is by all mean distinguishable to the naked eye, but for some unexpected reason misclassified by the algorithm. In other instances, kernels lie slightly more than tolerable over each other causing the algorithm to group them as one. Luckily both these cases are flagged as “non” kernels by the center and size checks in software.

Another thing that seldomly happens is that a little bits of glare still seem to pop up on the edges of the glass. By chance the algorithm splits this into two squarish shapes on both sides. This bypasses the center and size checking algorithm and is seen as a kernel. Figure 5.2 illustrates this. Luckily this does not happen very often and in future can be solved by using better anti reflective materials or more advanced color checks.

Figure 5.1: Example of a kernel that has been unexpectedly cut through middle. Mask (left); Original (right)

Figure 5.2: Example of a squarish shape glare which is misclassified as a kernel

The impact the glass has on the quality of the images can be best shown in Figure 5.3. There is overall not much of a difference but looking carefully, one can notice a slight shade of blur on the bottom image, making finer details less apparent. However for the purposes of this model these slight differences have little effect overall performance.

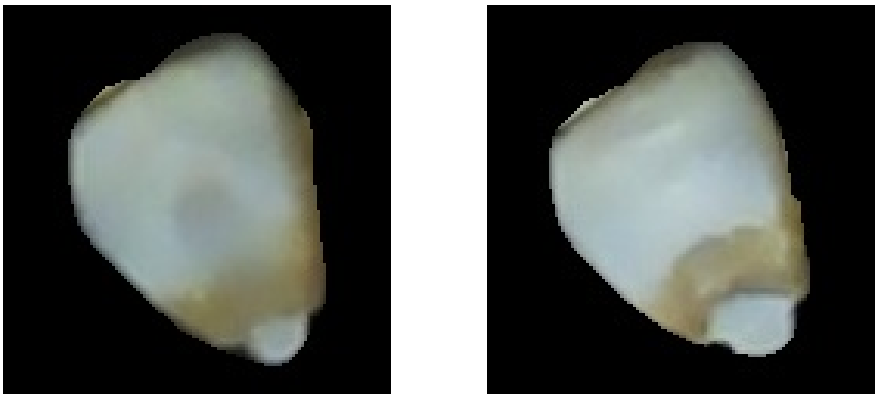


Figure 5.3: Image quality through glass comparison. Glass: (Left / bottom side). Direct: (Right / topside).

5.2. Training results

The model achieved a accuracy of 0.9777 on the training set indicating a very good fit on the training data. For the validation set, the accuracy achieved was: 0.947. This is relatively speaking quite high and indicates that model should generalize well at unseen images. Throughout the training process it was observed that the accuracies plateaued at around 50 epochs. The Figure 5.4 shows the training trends. With the validation accuracy experiencing significant fluctuations up to that point.

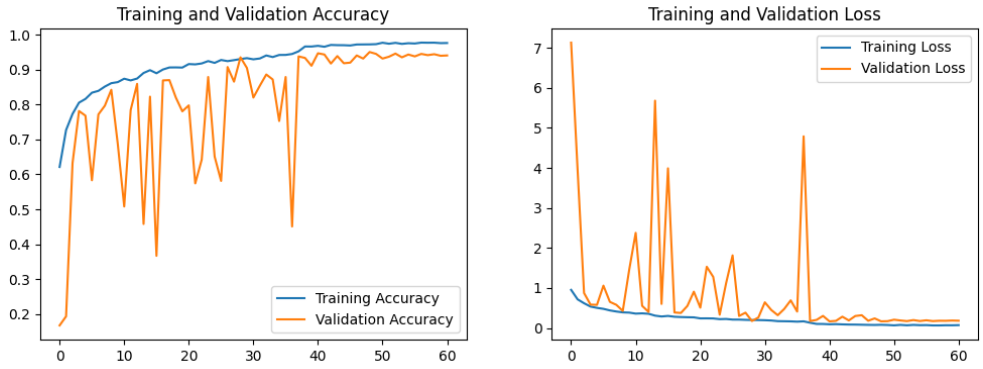


Figure 5.4: Training results of accuracy and loss per epoch

Validation set performance

To demonstrate the performance by class of the validation data the confusion matrix is show below in Figure 5.5. The validation set consisted of 1117 particles with the accuracies of each class as:

0 White Maize = 98.3471

1 Discolored Maize = 96.9565

2 Yellow Maize = 100

3 Insect Damage = 93.1298

4 Heat Damage = 95.7831

5 Fursarium = 70.0935

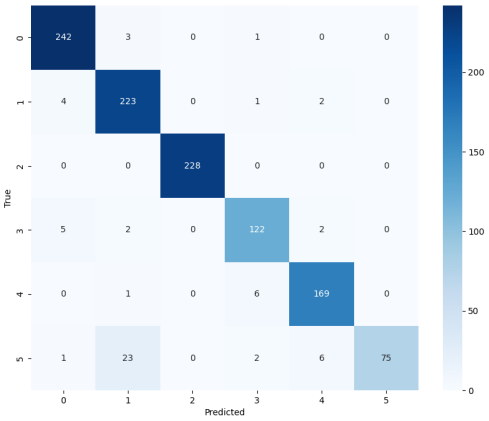


Figure 5.5: Confusion matrix

Separating yellow maize from white maize is 100% accurate, which is understandable given the complete color difference. Classifying insect and heat damage also performed reasonably well.

However, the biggest concern when looking at the above results is the poor performance of the fusarium category compared to all the others. . Upon further inspection into the images that were misclassified, I believe that the performance value can be misleading.

Many of the fusarium kernels had been predicted to be discolored by the model. Figure 5.6 shows a snippet of these miclassified kernels. Personally I, (although I am no expert grader) would have struggled and most likely made the same mistake of misclassifying these kernels as they look very similar to the discolored ones.

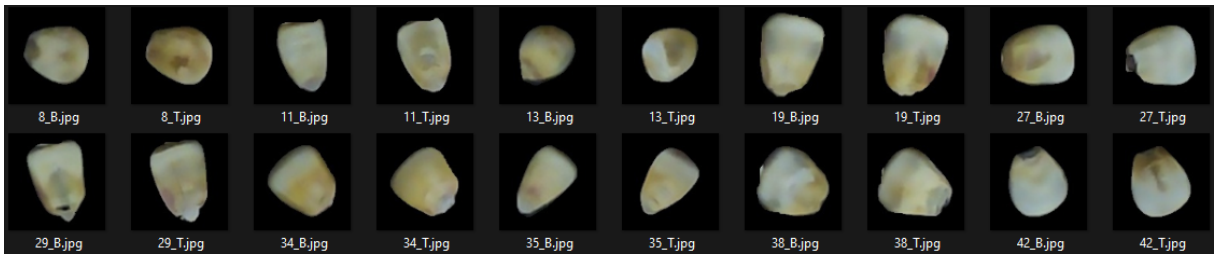


Figure 5.6: Fusarium kernels that were misclassified as Discolored

Training with the "severe" fusarium batch instead would have most definitely provided better results in training. However I still believe that using the normal set provides better results on unseen data.

Compared other tech

While directly comparing our results to that obtained in a different environment may not be be entirely justifiable, such a comparison may still provide some valuable insights. Table 5.1 below shows the validation accuracies compared to the initial benchmark accuracies shown in Chapter 2.

The detection of pure white maize performed well compared to that of the hyperspectural system. For yellow maize the model had a perfect prediction, which is on par with the multispectural system. Both insect damage and heat damage were too on par. However in the case of fusarium the CNN-model underperformed with a 70% accuracy.

These results should be taken with grain of salt since the systems were not designed to do the exact same things and the testing enviroments differed dramatically. Nevertheless what this does indicate is that one can achieve a relatively high degree of classification accuracy by using ordinary cameras and examining purely the outside shell of a kernel.

Available categories	Benchmark Accuracy %		
	Hyperspectral Camera	**Multispectral Camera**	**CNN-Model (On validation set)**
White Maize	88.3	-	98.35
Discolored White Maize	-	-	96.96
Yellow Maize	75.0	100	100
Insect Damaged	95.8	88.57	93.13
Heat Damaged	95.0	100	95.78
Fusarium	100	88.57	70.09

Table 5.1: Benchmark accuracy from previous studies

5.3. Run time

The whole idea of the system is to be designed in such a sense that the grading process is as quick and efficient. Taking photos shouldn't take longer than around two minutes. After the images have been captured the program scripts runs and the entire process from start to finish takes around 90 seconds on the Raspberry Pi. Figure 5.7 shows each step and their relevant times taken. The image segmentation process is by far the longest process to run, thereafter the particle matching and concatenation process. The final cleaning stage is just a step that removes all the temporary files created. With further optimisation in code, one might speed the process up by a few seconds. For now 90 seconds is reasonably fast compared to the time it would take to manually grade the particles.

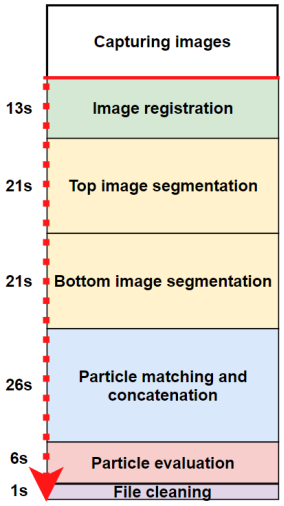


Figure 5.7

5.4. Testing on graded batches

As previously mentioned, the the SAGL have provided five 250g samples of graded maize batches. These include a mixture of different types of kernels and in most cases, the proportions of each kernel type are provided as percentages. The following section will discus the models performance on each of the batches. There are two tables for each batch. One of which is based on a direct report from the SAGL, which includes foreign matter and small shavings. This original table can be found in the Appendix B. The other table contains these percentages adjusted by the removal of the two above mentioned

categories. The conclusions are drawn from the adjusted table since this is a more accurate representation of the models performance.

5.4.1. Batch 1

Batch 1 consists mainly of yellow kernels. The biggest challenge encountered is the presence of yellow maize kernels with insect damage. The model had not been trained to recognise insect damage on yellow kernels and thus unfortunately the dominant yellow hue seems to ”overpower” the insect damaged features when the image was processed by the model. As a result almost every ID instance was seen as a plain yellow maize kernel. In addition heat damage performed relatively well, whereas some slight amounts of insect damaged kernels were seen as fursarium. Exact percentages of discolored and WM were unknown with the final combined deviation error being 3.59%.

Batch 1	**SAGL Grade Adjusted(%)**	**Model Grade (%)**	**Error**
White Maize	*	0.77	*
Discolored Maize	*	0.46	*
Yellow Maize	93.02	96.62	3.6
Insect Damage	3.28	0.31	2.97
Heat Damage	1.27	1.23	0.04
Fusarium	0	0.62	0.62
Defective above 6.35mm	4.55	2.16	2.39
Other coloured	2.43	1.23	1.2
Combined deviations	**6.98**	**3.39**	**3.59**
Class and Grade	**YM1**	**YM1**	-

Table 5.2: Batch 1 Results Adjusting SAGL grade by removing Foreign Matter and Small particles

Figure 5.8: An example of a insect damaged kernel that was mistakenly misclassified as a yellow maize kernel

5.4.2. Batch 2

In comparison to the other batches, batch 2's classification is subpar. The batch consisted of mainly white maize with lots of smaller pieces and shavings. Before taking the images these pieces had been manually removed. It is believed that some of the pieces that were discarded had been initially classified as valid maize kernels by the SAGL. This contributed significantly to the low classification accuracy on this batch. For the white maize category the model was around 5% off. Many of the white maize kernels were incorrectly identified as being heat damaged or discolored. Upon reviewing the model's results, the inconsistencies weren't immediately apparent to myself and finding kernels that were clearly misclassified in these two categories was challenging. I am lead to believe that this again comes back to the above mentioned issue. Additionally, diploda was present in the batch and unfortunately is one of the categories not been present in the training phase. The final combined deviation error was 5.14%.

Batch 2	**SAGL Grade Adjusted(%)**	**Model Grade (%)**	**Error**
White Maize	95.44	90.30	5.14
Discolored Maize	0	1.34	1.34
Yellow Maize	0	0	0
Insect Damage	0	0.45	0.45
Heat Damage	0.21	3.73	3.52
Fusarium	4.14	4.18	0.04
Diplodia	0.21	-	-
Defective above 6.35mm	4.56	8.36	3.8
Other coloured	0	1.34	1.34
Combined deviations	**4.56**	**9.7**	**5.14**
Class and Grade	**WM1**	**WM2**	-

Table 5.3: Batch 2 Results Adjusting SAGL grade by removing Foreign Matter and Small particles

5.4.3. Batch 3

Overall this batch performed well, especially in the WM category. However the biggest challenge from this set is the models under representation of yellow maize kernels. Another point of concern is tha HD and ID had been detected where the maize was in fact not

faulty. This batch contained many broken particles, which the model often interpreted as insect damage. Final combined deviation error was 2.35%.

Batch 3	SAGL Grade Adjusted (%)	Model Grade (%)	Error
White Maize	87.39	89.74	2.35
Discolored Maize	0.25	1.65	1.4
Yellow Maize	12.24	4.95	7.29
Insect Damage	*	2.01	*
Heat Damage	*	1.65	*
Fusarium	*	0	*
Defective above 6.35mm	0.12	3.66	3.54
Other coloured	12.49	6.6	5.89
Combined deviations	**12.61**	**10.26**	**2.35**
Class and Grade	**OTHER**	**WM3**	**-**

Table 5.4: Batch 3 Results Adjusting SAGL grade by removing Foreign Matter and Small particles

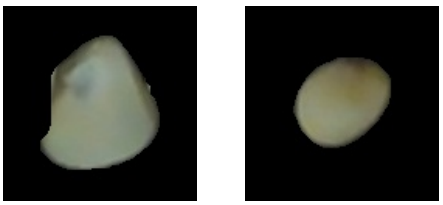


Figure 5.9: Example of kernels that have been misclassified. Left: Oddly shaped white maize kernel that is classified as insect damaged. Right: Circular and slightly discolored White maize kernel classified as heat damaged.

5.4.4. Batch 4

For this batch ID classification and YM was reasonably close. A recurring issue was again the over classification of heat damage. A significant numbet of fusarium and white maize kernels where classified as heat damaged. The final combined deviation error was 2.47%.

Batch 4	SAGL Grade Adjusted (%)	Model Grade (%)	Error
White Maize	*	0	*
Discolored Maize	*	0.54	*
Yellow Maize	83.98	86.44	2.46
Insect Damage	0.23	0.18	0.05
Heat Damage	3.18	9.22	6.04
Fusarium	4.55	3.62	0.93
Defective above 6.35mm	7.96	13.02	5.06
Other coloured	8.07	0.54	7.53
Combined deviations	**16.03**	**13.56**	**2.47**
Class and Grade	**YM3**	**YM3**	**-**

Table 5.5: Batch 4 Results Adjusting SAGL grade by removing Foreign Matter and Small particles

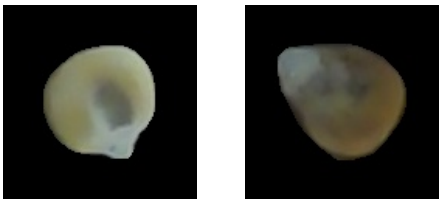


Figure 5.10: Example of kernels that have been misclassified. Left: White maize kernel that had been classified as being heat damagd. Right: Fusarium kernel that had also been classified as heat damaged.

5.4.5. Batch 5

The classification for this batch was arguably the best among all the others. With a ”defective above 6.35mm” error of 2% and ”other colored” error of 0.97%. Unfortunately the exact percentages of discolored maize and yellow maize were unknown, and additionally the class ”sprouted” had not been trained with model and was thus unrecognised. But in almost all of the classes the model got the proportions mostly correct. The final combined deviation error was 2.82%

Batch 5	SAGL Grade Adjusted (%)	Model Grade (%)	Error
White Maize	92.65	89.68	2.97
Discolored Maize	*	2.10	*
Yellow Maize	*	1.61	*
Insect Damage	0	0.97	0.97
Heat Damage	4.39	5.48	1.09
Fusarium	0	0.16	0.16
Sprouted	0,22	-	-
Defective above 6.35mm	4.61	6.61	2
Other coloured	2.74	3.71	0.97
Combined deviations	**7.35**	**10.32**	**2.82**
Class and Grade	**WM1**	**WM2**	**-**

Table 5.6: Batch 5 Results Adjusting SAGL grade by removing Foreign Matter and Small particles

5.5. Summary of test batches

The following section provides tables summarizing the results of the five batches.

Table 5.7 describes the error rate among ”defective kernels above 6.35mm”, (category [#2]). What this is referring to is the insect damage, heat damage and fusarium that occurs in every batch. On average the error between all the batches amounts to 3.04% which is reasonable in normal circumstances. Referring back to the regulations Table 2.1, when classifying a batch as either WM1 or WM2 based on category [#2], there is only a 6% margin of error. Thus this 3.04% obtained is in fact quite high in comparison.

[#2] Defectives above 6.35mm			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error (%)**
[1]	4.55	2.16	2.39
[2]	4.56	8.36	3.8
[3]	0.12	3.66	3.54
[4]	7.96	13.02	5.06
[5]	4.61	6.61	2
Average Error:			**3.04**

Table 5.7: Defectives test

For the class "other colored kernels" (category [#3]), the average error between the batches amounted to 3.39, as per Table 5.8. With reference again to the regulations Table 2.1. The difference here between grades is 3%. The error rate is thus quite high again, suggesting clear room for improvement.

[#3] Other colored kernels			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
[1]	2.43	1.23	1.2
[2]	0	1.34	1.34
[3]	12.49	6.6	5.89
[4]	8.07	0.54	7.53
[5]	2.71	3.71	0.97
Average Error:			**3.386**

Table 5.8: Other color test

When looking at the "combined deviations" summary Table 5.9 (category [#4]), the total average error is 3.27%. There is more leeway from this regulation category as seen in Table 2.1. All in all this error value is not too bad if this was the only metric used to classify maize batches. However the challenges associated with the previous two categories remain.

[#4] Combination of deviations			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
[1]	6.98	3.39	3.59
[2]	4.56	9.7	5.14
[3]	12.61	10.26	2.35
[4]	16.03	13.56	2.47
[5]	7.35	10.32	2.82
Average Error:			**3.27**

Table 5.9: Combined deviations

Chapter 6

Conclusion

This chapter discuses the successes and shortcomings of the system, as well as potential future recommendations.

6.1. Summary

The primary objective of the project is to develop an accurate and cost-effective method for grading maize kernels using image processing techniques. While the system designed largely does accomplish this goal, it is in several areas not perfect.

Image capture

The overall box design performs well and creates a consistent environment for the images to be taken. Images through the glass also remain high quality. Some drawbacks of this system is that the glass platform tends to get dirty when used frequently. The size of platform limited to 150g samples of maize at a time. Glare on the corners of the image pose a impact on the overall quality of the system and furthermore the box is quite big and thus not very portable.

Image processing and training

The image segmentation section of the project performs relatively well, missing only a few kernels out of every batch. These misses should however be addressed in further research. In terms of speed, the system runs reasonably fast compared to having to manually grade kernels by hand. The high validation accuracies in training process is promising and it proves is that it is very possible to achieve similar levels of accuracy using images from ordinary cameras.

Unseen batches

When testing on the unseen batches the system was met with some challenges. Primarily involving issues to shavings, misclassifying ID on yellow maize and tendency for the over-classification of heat damage. The model had an average error rate of around 3.04%

for category [#2], 3.39% for category [#3] and 3.27% for category [#4]. This error rate was higher than expected. When comparing these results with the regulations, one can see the effect it has on the overall grade. In this case only one out of the five batches had been given the correct final grade.

6.2. Future suggestions

Image capture

- For capturing images, it would be a great addition to have some system that automatically sifts out smaller particles. Possibly even doing separate evaluation on those particles.
- For industrial use, one might would want to also consider using a smaller box with a more suitable camera as the current one is not very portable.
- To remove glare, one could use better anti glare materials. Such as anti reflective glass.
- To create a perfectly working image segmentation system one can build in sieves or dividers in the container platform.
- One can setup more one camera to make image capturing process even faster.

Image processing and training

- To prevent kernels from being missed one can use more advance software techniques for image segmentation.
- To improve model performance one can expanding the training data-set. Especially by adding extra categories. This includes testing with “severe fusarium”.

Other

For a direct comparison having multi-spectral or hyper-spectral camera with which the same kernels can be taken would be ideal. Additionally it would be nice to the use of other spectrum filter cameras such as NoIR to test the difference in performance.

Final thoughts

The model can be fine tuned slightly more and possibly trained with more types of data. However, I for one don’t believe that this will drastically change the performance on unseen cases. With different experimental batches (without shavings) and better image segmentation techniques, I believe this error rate would be a lot lower and that, that is where the bottleneck lies. The current system works well if one would like to have a quick overview of the batch, but in terms of grading regulations, the system is still far from industrial type use and has a lot of room for improvement.

Bibliography

[1] "Grading regulations for maize, government notice no. r.473," Department of argiculture, Tech. Rep., 2009.

[2] T. S. A. G. Laboratory, "Quality report 2020/2021 season," South African Maize Crop, Tech. Rep., 2021.

[3] K. Sendin, M. Manley, and P. J. Williams, "Classification of white maize defects with multispectral imaging," *Food Chemistry*, vol. 243, pp. 311–318, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0308814617316096

[4] K. Sendin, M. Manley, F. Marini, and P. J. Williams, "Hierarchical classification pathway for white maize, defect and foreign material classification using spectral imaging," *Microchemical Journal*, vol. 162, p. 105824, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026265X20337644

[5] D. Tyagi, "Introduction to sift(scale invariant feature transform)," *Medium*, 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[6] M. Inc, "What is image registration?" *Mathworks*, 2023. [Online]. Available: https://www.mathworks.com/discovery/image-registration.html

[7] I. Arganda-Carreras, "Classic watershed," *ImageJ Wiki*, 2019.

[8] A. W. R. Fisher, S. Perkins and E. Wolfart, "Erosion," *Hypermedia Image Processing Reference*, 2003.

[9] scikit-image team, "Convex hull," *scikit-image/Examples*, 2023. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/edges/plot_convex_hull.html#sphx-glr-download-auto-examples-edges-plot-convex-hull-py

[10] M. Mishra, "Convolutional neural networks, explained," *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[11] A. Burkov, *The Hundred-page Machine Learning Book*. Andriy Burkov, 2019.

[12] S. University, "Cs231n convolutional neural networks for visual recognition," *CS231n: Deep Learning for Computer Vision*, 2023.

[13] R. P. Ltd, "Raspberry pi zero 2 w," Tech. Rep., october, 2021. [Online]. Available: https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf

[14] ——, "Raspberry pi 4 model b datasheet," Tech. Rep., January, 2021. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf

[15] ——, "Raspberry pi 3 camera datasheet," Tech. Rep., Accessed: August 28, 2023. [Online]. Available: https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf

[16] SPAZIO, "Technical dats sheet, led tapelight," *Printfriendly.com*. [Online]. Available: https://www.printfriendly.com/p/g/BhxsKG

[17] S. Saxena, "The architecture of lenet-5," *Analytics Vidhya*, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/

Appendix A

Additional images

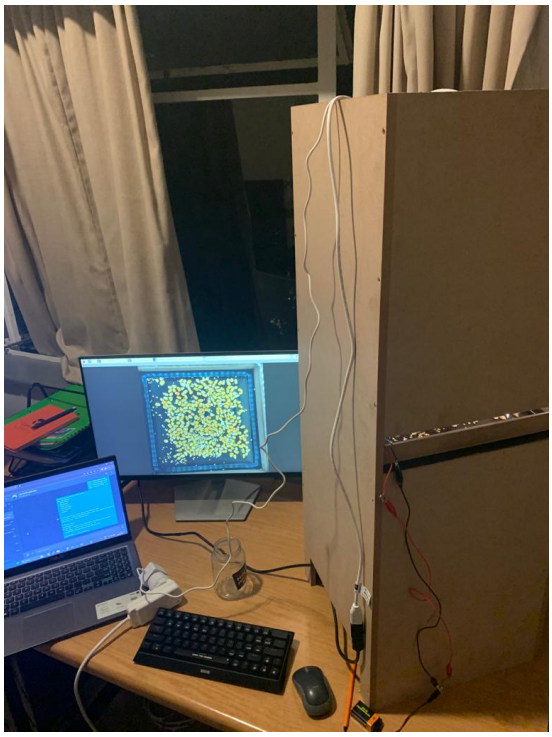


Figure A.1

Figure A.2

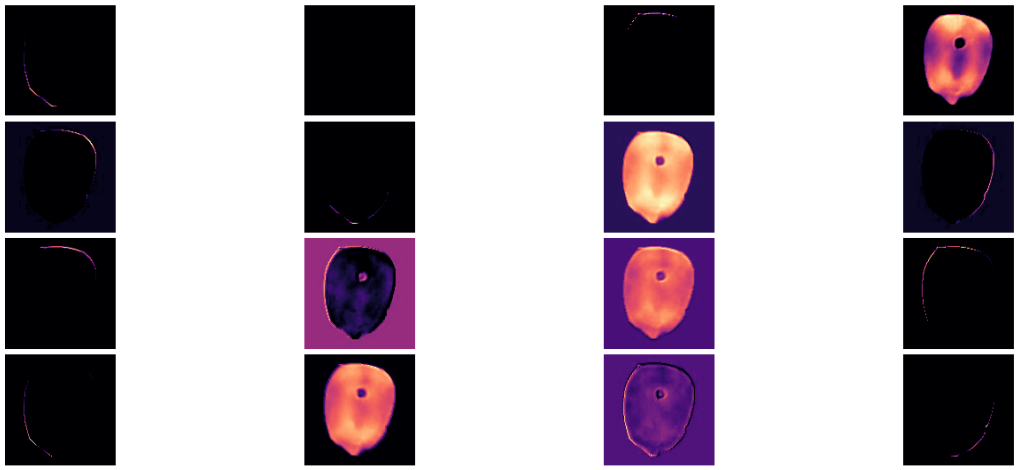


Figure A.3

Appendix B

Batch results

Batch 1	SAGL Grade (%)	Model Grade (%)	Error
White Maize	*	0.77	*
Discolored Maize	*	0.46	*
Yellow Maize	88.00	96.62	8.62
Insect Damage	3.10	0.31	2.79
Heat Damage	1.20	1.23	0.03
Fusarium	0	0.62	0.62
Defective above 6.35mm	4.30	2.16	2.14
Defective below 6.35mm	5.10	-	-
Defective kernels (above and below 6.35mm)	9.4	2.16	7.24
Other coloured	2.30	1.23	1.07
Foreign Matter	0.3	-	-
Combined deviations	**12**	**3.39**	**8.61**
Class and Grade	**YM2**	**YM1**	-

Table B.1: Batch 1 Results

Batch 2	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
White Maize	90.0	90.30	0.3
Discolored Maize	0	1.34	1.34
Yellow Maize	0	0	0
Insect Damage	0	0.45	0.45
Heat Damage	0.20	3.73	3.53
Fusarium	3.90	4.18	0.28
Diplodia	0.2	-	-
Defective above 6.35mm	4.3	8.36	4.06
Defective below 6.35mm	5.6	-	-
Defective (above and below 6.35mm)	9.9	8.36	1.54
Other coloured	0	1.34	1.34
Foreign Matter	0.1	-	-
Combined deviations	**10.0**	**9.7**	**0.3**
Class and Grade	**WM2**	**WM2**	-

Table B.2: Batch 2 Results

Batch 3	SAGL Grade (%)	Model Grade (%)	Error
White Maize	70.7	89.74	19.04
Discolored Maize	0.2	1.65	1.45
Yellow Maize	9.9	4.95	4.95
Insect Damage	*	2.01	*
Heat Damage	*	1.65	*
Fusarium	*	0	*
Defective above 6.35mm	0.1	3.66	3.56
Defective below 6.35mm	16.2	-	-
Defective kernels (above and below 6.35mm)	16.3	3.66	12.64
Other coloured	10.1	6.6	3.5
Foreign Matter	2.9	-	-
Combined deviations	**29.3**	**10.26**	**19.04**
Class and Grade	**OTHER**	**WM3**	**-**

Table B.3: Batch 3 Results

Batch 4	SAGL Grade (%)	Model Grade (%)	Error
White Maize	*	0	*
Discolored Maize	*	0.54	*
Yellow Maize	73.9	86.44	12.54
Insect Damage	0.2	0.18	0.02
Heat Damage	2.8	9.22	6.42
Fusarium	4.0	3.62	0.38
Defective above 6.35mm	7	13.02	6.02
Defective below 6.35mm	11.6	-	-
Defective kernels (above and below 6.35mm)	18.6	13.02	5.58
Other coloured	7.1	0.54	6.56
Foreign Matter	0.4	-	-
Combined deviations	**26.1**	**13.56**	**12.54**
Class and Grade	**OTHER**	**YM3**	**-**

Table B.4: Batch 4 Results

Batch 5	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
White Maize	84.4	89.68	5.28
Discolored Maize	*	2.10	*
Yellow Maize	*	1.61	*
Insect Damage	0	0.97	0.97
Heat Damage	4.0	5.48	1.48
Fusarium	0	0.16	0.16
Sprouted	0.2	-	-
Defective above 6.35mm	4.2	6.61	2.41
Defective below 6.35mm	8.0	-	-
Defective kernels (above and below 6.35mm)	12.2	6.61	5.59
Other coloured	2.5	3.71	1.21
Foreign Matter	0.9	-	-
Combined deviations	**15.6**	**10.32**	**5.28**
Class and Grade	**OTHER**	**WM2**	**-**

Table B.5: Batch 5 Results

Preprocessing Stage	Frameworks	Key Functions
Image Registration	OpenCV NumPY	SIFT_create() BFMatcher() detectAndCompute() knnMatch() findHomography() warpPerspective()
Image Segmentation	ImageJ Fiji Java plugin	IJ.run(”HSB Stack”) ImagePlus(”Hue_Mask”) ImagePlus(”Saturation_Mask”) ImagePlus(”Brightness_Mask”) ImageCalculator.run(”AND create”)
	ImageJ Fiji Java plugin	IJ.run(”8-bit”) IJ.run(”Erode”)
	ImageJ Fiji Java plugin	IJ.run(”Watershed”)
	OpenCV NumPY	cv2.convexHull() cv2.findContours() cv2.drawContours()
Particle Matching	OpenCV NumPY	get_centers_and_contours() pad_image_to_size() np.linalg.norm() cv2.boundingRect() cv2.bitwise_and()
Image Concatenation	NumPY	np.concatenate()

Table B.6: Preprocessing software