School of Computing
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Final Report

Narrow Fabric Defect Detection using Computer Vision

Tom Schofield

Submitted in accordance with the requirements for the degree of
BSc Computer Science

2023/23

COMP3931 Individual Project

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Final Report	*PDF file*	*Uploaded to Minerva (DD/MM/YY)*
Digitally signed participant consent forms	*PDF file / file archive*	*Uploaded to Minerva (DD/MM/YY)*
Link to online code repository	*URL*	*Sent to supervisor and assessor (DD/MM/YY)*

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) 

Summary

This report aims to assess how computer vision could be used to identify defects in fabric. Utilising the open-source technologies of OpenCV and TensorFlow, three different inspection techniques were crated and compared. PyQt was then used to build a prototype inspection application to apply the techniques developed.

The first inspection technique attempted to statistically analyse images of fabric, looking for different factors that could determine its quality. The second utilised image morphology and contour finding to look for large objects present in the fabric, that could be treated as defects. The last method leveraged TensorFlow to build a CNN (Convolutional Neural Network) that was trained to distinguish images of defective fabric from images of non-defective fabric. This training used pre labelled defect data obtained from the aitex fabric image database.

A prototype graphical application was then created using the second and third inspection methods. The report compares the accuracies and speeds of each inspection technique to each other and human inspection, the current method used by the majority of producers.

The report concluded that the implementation of statistical analysis could not accurately identify defects, and that morphology and contour finding could be used to locate larger defects and could be used to aid human inspectors. Finally, that a CNN based approach can obtain a similar level of accuracy as human inspectors and is a promising candidate for fully autonomous inspection.

Acknowledgements

I would like to thank my assessor, Natasha Shakhlevich Chakhlevitch and both of my supervisors, Amy Lowe and David Head, who provided indispensable guidance throughout the project.

I would also like to thank my friends who participated in an experiment to gather human inspection data.

Finally, I would like to dedicate this report to my late father Charles Schofield, who inspired the idea for the project.

Table of Contents

Chapter 1
Introduction and Background Research

1.1 Introduction

The purpose of this project was to explore how computer vision could be used in fabric defect detection and what advantages it could offer over traditional methods.

This project and report will investigate the effectiveness of different computer vision techniques at locating defects in images of fabrics. To carry out this investigation three different inspection techniques were created, the first two using OpenCV and the last using TensorFlow.

Lastly, a prototype application was created that would be used to automatically inspect user supplied images using the most effective inspection methods found in the initial stages of the project.

1.2 Fabric Production

All fabric is produced through one of a number of processes: weaving, knitting, felting, bonding, or turfing. Out of these, the most common is weaving, which produces most conventional fabrics. First, fibres (synthetic or natural) are spun into yarn, which is then converted into the fabric using the process of weaving (Shaker, 2016). As this production method is the most widespread, the project's definition of "fabric" or "textile" was limited to only what is produced by weaving. To understand the problem, it is important to have a surface-level understanding of how these production techniques function as each stage can introduce distinct defects.

1.2.1 Spinning

First, fibres are harvested naturally for example flax, or are produced synthetically such as polyester. These are then aligned and collected into yarn through the process of spinning, this process varies for natural and synthetic materials. Natural fibres can be spun in many ways, but all involve them being twisted, which binds them together and forms yarn (Smith, 1969, p. 1).

Synthetic fibres are spun differently, liquid polymer is extruded through many, densely packed small holes. A cool “quench air” is passed over them to solidify the liquid polymer into long continuous fibres. Variations in the air passed over them cause the fibres to bunch in certain areas binding them together into yarn (Denn, 1983, pp. 179-180).

1.2.2 Weaving

After yarn is created, weaving can be used to convert this yarn into fabric. Weaving is the process of interlacing yarn perpendicularly to each other. The patterns this is done in determine the properties and appearance of the textile (Adanur, 2020, p. 1). Terms used later in the project are warp and weft and they are key when understanding the weaving process. All woven fabric consists of warp and weft yarns, warp yarns are the yarns that run parallel to the edge, known as the “selvage”, of the fabric and are used to form its structure. Weft yarns are those running perpendicular to the edge of the fabric and are interlaced between the warp yarns using a loom (Lord,1982).

1.3 AITEX Fabric Image Database

This project could not have been developed without the AITEX fabric image database and its accompanying article “A PUBLIC FABRIC DATABASE FOR DEFECT DETECTION METHODS AND RESULTS” (Silvestre-Blanes, 2019, pp.363-374). The database consists of 245 images ranging over 7 different types of woven fabric. It contains 140 images of defect free fabric, 20 each type of fabric. With 105 images of defective fabric spread between all 7 fabric and 12 defect types. Usefully the database also contains 105 mask images for each image of defective fabric, these mask images match the dimensions of the fabric images (4096×256 pixels). The masks contain only black pixels apart from the pixels that make up the defect, these are coloured white.

1.4 Defect Types

Now the project has a baseline understanding of how fabric is created and a database to refence, we can assess what defects are produced during the phases of fabric creation.

The twelve defect types present in the AITEX database are:

- Fuzzy balls, these are the result of warp yarns that are too small. Lowering their abrasion resistance and allowing fibres to break loose to form fuzz balls (cotton works list of defects, 2020).

- Neps, these are formed by an accumulation of fly and fluff on the machinery used and damage the fabrics appearance (Nateri, 2014). Often caused in the spinning process.
- Knots, these occur in natural or synthetic warp yarns and damage the appearance and tensile strength of the textile. Knots can also refer to dead or immature natural fibres (Lim, 2018).
- A crease is a valley or ridge present in the final fabric caused by folding or improper tension.
- A broken end appears as an untied or broken warp end and manifests as horizontal lines in the fabric. The yarn is usually broken during weaving or finishing processes (Lim, 2018).
- A broken pick defect consists of a broken weft yarn and results in a sudden discontinuity in the weave (Lim, 2018).
- Broken yarns are breaks in either the warp or weft yarn during the weaving process and result in elongated holes.
- Contamination refers to foreign objects stuck too or woven into the textile. Even if the object is later removed it may have disturbed the weave or cause uneven dyeing.
- A warp ball is caused by a single or multiple warp yarns becoming clumped or entangled.
- Selvage is the densely woven edge of a piece of fabric, a cut in the selvage can cause a separation in the weave (Lim, 2018).
- Weft curling Is produced by the use of highly twisted weft yarns and it disturbs the pattern of the weave (Textile Tutorials, 2014).
- A weft crack results in a narrow streak running in parallel with the weft. This is caused by an absence of weft yarn (Textile Sphere).

Figure 1 displays the ROI (region of interest) of 256 x 256 pixels from images of defective fabrics. The defects are: (a) broken end, (b) broken yarn, (c) broken pick, (d) weft curling, (e) fuzzy ball, (f) cut selvage, (g) crease, (h) warp ball, (i) knot, (j) contamination, (k) nep, and (l) weft craft (Silvestre-Blanes, 2019).

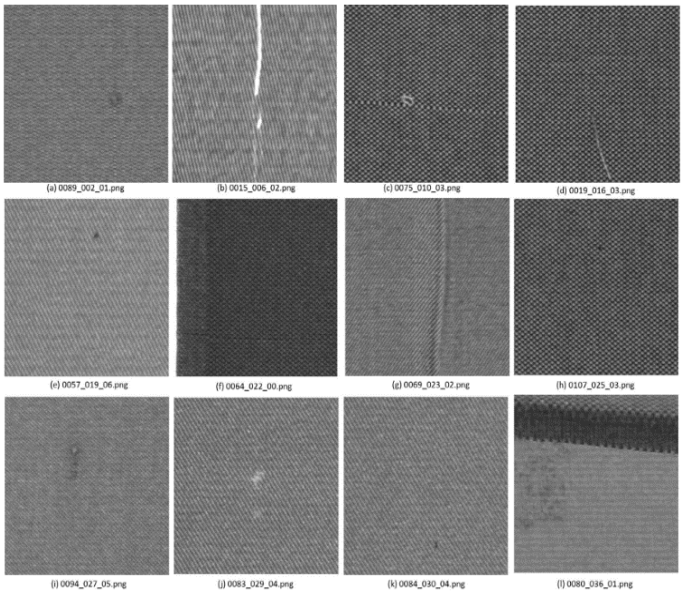


Figure 1 - Defect Images (Silvestre-Blanes, 2019)

1.5 Importance of Inspection

Fabric inspection is important for a multitude of reasons, the upmost being the prevention of inferior quality product being sold to or used by consumers (Malek, 2013). Inferior or defective product being used or sold can cause a variety of consequences for the producer.

Inspection at early stages can be financially beneficial to the producer because it stops the offending sections from being processed or being sold to those who will apply finishing processes. Defects are not only cosmetically undesirable; many also affect the structure of the weave and can degrade its tensile strength. Ultimately leading to textiles being distributed with a lower tensile strength than advertised, creating the potential for catastrophic failures. For example, it is important that seat belts or ratchet straps have no structural defects.

1.6 Current Inspection Techniques

Currently, fabric inspection techniques vary, but the majority use human inspection. Human inspection uses workers to scrutinise fabric by hand. The accuracy of human inspection declines over time due to the monotonous nature of the task. This results in slower, more expensive, and erratic inspection (Chin, 1982).

Many in the industry have paired the use of human inspection with inspection using DSP (digital signal processing). DSP analyses the waveform produced by a sensor pointed at the fabric in controlled lighting. However, most are now moving towards computer vision approaches as they are easier to implement and are more capable of inspecting a variety of fabrics.

1.7 Computer Vision

Computer vision is defined as the process of allowing computers to see the real world (Learned-Miller, 2011); it encompasses capturing, processing, and analysing images. Computer vision has proven invaluable when automating quality control. It offers higher accuracy and speed than human inspection, but more importantly, it is far more consistent, not suffering from deteriorating results as the day progresses (Brosnan, 2004).

There are a number of code libraries that enable computer vision, Google, Microsoft, and Amazon all have offerings. However, the most widely used library is OpenCV, created by Intel. Image morphology encompasses a broad set of image processing operations based on the boundaries of shapes. The most common forms are erosion, dilation, opening and closing (Haralick,1987). Erosion reduces the size of objects, particularly important when reducing noise in an image. As images of fabric are inherently noisy due to the repeating weave, image morphology may be key when emphasising larger objects and defects in an image. OpenCV offers all of these morphological transformations in optimised functions, however, each can only be applied to a greyscale image.

Conversion to grayscale, thresholding, and contour tracking functions are all offered by OpenCV. Converting an image to grayscale replaces each 3-channel pixel composed of a red, green, and blue value with a single grey value. Grayscale images are easier to process and are required for many other image processing algorithms. Thresholding converts a grayscale image into a bi-level image and is usually used to separate foreground objects from the background. Contour finding can be used to find the boundary of an object (Xie, 2013). Guobo Xie and Wen Lu used all the above methods to detect the number of copper strands in a small wire, checking to see if the correct number was present.

1.7.1 Computer Vision Quality Control Example

Alessandro Massaro, Valeria Vitti, and Angelo Galiano wrote the paper "AUTOMATIC IMAGE PROCESSING ENGINE ORIENTED ON QUALITY CONTROL OF ELECTRONIC BOARDS." It details how computer vision was used to find defects on electronic boards after welding and how the final method was calibrated and tested. The paper details a number of steps involved in these processes. The first was to segment the image; this involves splitting the image into a number of small groups of pixels, simplifying the image and making it easier to analyse. The snake contour and watershed are two processes used to segment. They then applied thresholding to separate the welds from the background of the boards, and the area of the resulting blobs was used to identify any that were defective (Massaro, 2018).

While the report did not detail exactly how effective these techniques were, it offered insight into processes that could be used in this project.

1.8 Machine Learning

The term machine learning describes a set of methods or algorithms that make predictions based on previous data they have been shown (Zhou, 2021). This project will focus on an area of machine learning called deep learning. All deep learning techniques use multiple layers to find or extract features from data. There are a number of methods that can be used when processing images, but the most popular is the convolutional neural network (CNN). They gained this popularity as they work particularly well at image classification, image recognition, and object detection tasks, so they were the clear choice when developing the project.

A CNN processes an image through multiple layers; some of these act as filters that are trained to recognise different features in the image, such as corners, edges, and patterns. These filtering layers are known as convolutional layers; two other types of layers are also present in CNN: max pooling and fully connected. Each convolutional layer is comprised of many filters and during training, the weights inside them are adjusted. The output of the convolutional layer is passed through an activation function such as ReLU to a max pooling layer, the max pooling layer reduces the output size while preserving the most important features. The last step of a CNN is the fully connected layer, which takes the output of previous layers, scaling this down and producing a set of scores that each relate to a specific class, using a final activation function (O'Shea, 2015). This is only a brief overview of how a CNN functions.

1.9 Background Research Summary

Automated systems can hold a clear advantage over manual inspection methods in both speed and accuracy, however there are many challenges. As shown in figure 1 defects can vary in size, colour and shape while this does not impact manual human inspection it makes rule based automated inspection far more complicated. There also exist the issue of when to inspect as many defects are unique to a specific step of production. Hence the project will move forward with the assumption that all inspection will be done after any finishing processes. While this is not optimum as inspection would be easier and more money and time could be saved if it occurred in earlier stages. However, for the sake of creating a universal inspection system inspection must occur on the finished fabric as each producers' methods are likely to differ.

Researching methods previously used for quality control and image processing have informed how the project will develop. The use of OpenCV, morphology and contour tracking have been successful when identifying defects in products. However, the previously discussed study ‘AUTOMATIC IMAGE PROCESSING ENGINE ORIENTED ON QUALITY CONTROL OF ELECTRONIC BOARDS’ classified defects purely based on size. It is still unclear if an approach like this will work with fabric defects, as depicted in figure 1 defects differ from normal fabric in several categories.

A CNN will likely be suited to this project due to its ability to recognise shape and texture. However, it is unlikely that the AITEX dataset consisting of 245 images will be enough to achieve a reasonable level of accuracy and so some form of image segmentation and data augmentation will be required to increase the data available.

Chapter 2
Methods

The goal of this chapter is to address the design and implementation of the three inspection techniques mentioned earlier and create the prototype inspection application. All were completed in python utilising OpenCV, TensorFlow, Skarn, Matplotlib and PyQt5.

2.1 Initial Project Decisions

This project was developed through agile methodology, using sprints to manage milestones. This method suited the project as no form of inspection guaranteed results, and so may have needed iterative adjustment. This allowed some area of the project to always be in active development, even during long model training periods or when certain avenues of thought led to dead ends.

Version control was used throughout, the project is stored in the GitLab repository: https://gitlab.com/TomSchofield/personalproject .

The decision to use the aitex image fabric defect database informed what detection methods would be implemented and how they would be evaluated. There was another choice, the University of Moratuwa fabric defect database (Belkhir, 2020). This included a far greater amount of defect data however it was split between only 5 defect types: colour, cut, hole, thread, metal contamination. All images from this dataset are also all from one type of fabric. For these reasons and the fact the type of fabric is less widespread than that found in the

aitex database. It was decided that while accuracy might, suffer the inspection methods produced would be more general if the project used the aitex database.

One of the initial ideas for the project was to include some form of defect categorisation, this would separate images already deemed as defective into individual categories. However once development commence it was decided that this was out of the scope of the project, the primary reason for this is the lack of a large enough datasets. The dataset chosen contained at most 20 images of each defect type and so does contain enough data to train a defect categorisation algorithm and achieve a reasonable accuracy.

2.2 Sprint 1 - Data Preparation and Analysis

2.2.1 Goals and Design Decisions

The goals for the first sprint were to prepare and investigate the dataset.

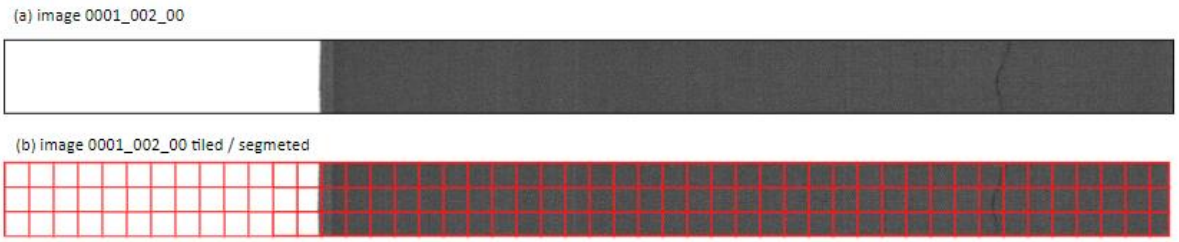


Figure 2 - (a) original image, (b) tiled image

Figure 2.a shows an example image form the dataset; all images are standardised to a size of 4096×256 pixels. This presents two main problems, the first is that the pixels that make up the defect represent a very small proportion of the overall image. The second is that any operations on such a large image would take a substantial amount of time.

Figure 2.b shows the solution to this issue, to segment the image. Once the image is segmented these tiles could be processed individually. Furthermore, defective pixels represented a much larger proportion of the tile they reside in. However, when examining figure 2 a flaw in this methodology appears, approximately the first quarter of the image is the background behind the fabric. Hence the tiles in the section contain no meaningful information and would probably be miss classified as defective. To remedy this the background must be removed.

The last goal will be to store the created tiles as either a defective or normal. However this presents some issues as not all tiles from a defective image are themselves defective.

2.2.2 Implementation

2.2.2.1 Tile Class

```
class Tile:
    x = 0
    y = 0
    width = 0
    height = 0
    imagex = 0
    imagey = 0
    roi = None
    overlap = 0
```

Code Snippet 1 – Tile class attributes

This tile class holds all the key attributes needed when describing the contents and location of the segment in relation to the master image. The attributes “x” and “y” hold the position of the tiles in relation to the other tiles. Whereas “imagex” and “imagey” hold the position of the tile in relation to the pixels of the master image. Overlap simply describes the amount the tiles overlap and is calculated from the right most corner of the previous tile.

```
def populate(self,master):
        self.roi = master[y:y+height, x:x+width]
        if (self.roi.shape[1] < width):
            new_x = x - (width-self.roi.shape[1])
            self.roi = master[y:y+height, new_x:new_x+width]
            self.imagex = new_x
        if (self.roi.shape[0] <height):
            new_y = y - (height-self.roi.shape[0])
            self.roi = master[new_y:new_y+height, x:x+width]
            self.imagey = new_y
```

Code Snippet 2 – population method

Code snippet 2 shows the implementation of the populate method of the tile class. It takes the master image as a parameter and calculates which segment of the image should be copied into the tile’s “roi” (region of interest). The rest of the snippet relaxes the overlap constraint if the tile is not the correct size. This happens at the edge of the master image where there aren’t enough pixels to fill an entire tile. The number of tiles that should be created is calculated this is done by finding the number of tiles in the x and y axis. The

number of tiles in either direction is the looped over and each tile is created and populated and placed into a 2d array.

2.2.2.2 Preparing the images

As described in the goals of this sprint the images needed to be cropped to remove the background. Unfortunately, the exact location of the edge was not in the same location for each image. Images were loaded into a NumPy array using the OpenCV “imread()” function.

Three processes were applied to determine the location of the edge. first a light blur is applied to the image, then a threshold is applied and finally Canny edge detection is used to find all vertical edges present in the image.

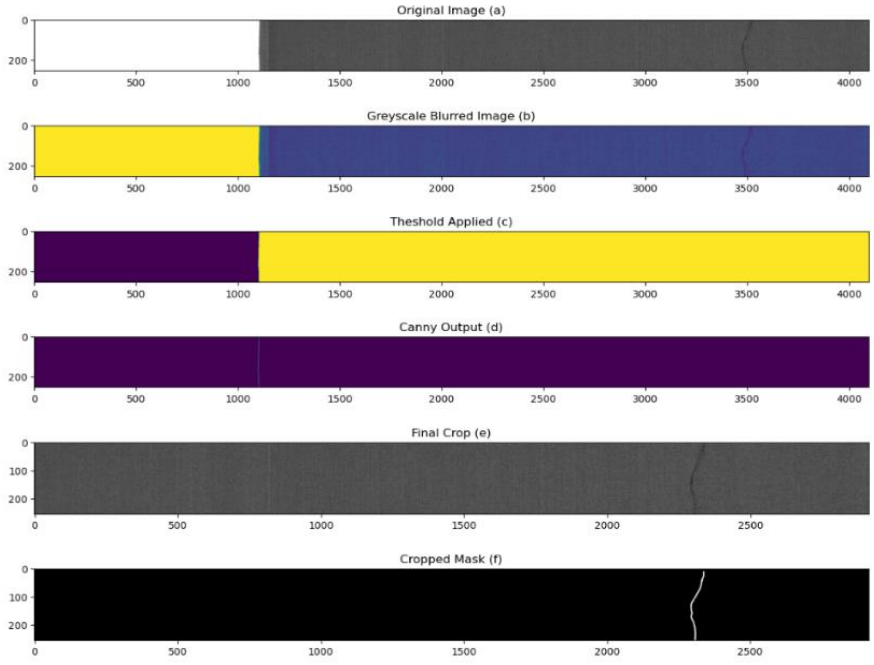


Figure 2 - edge finding steps

Figure 3 demonstrates the steps involved when finding the edge of the fabric. (b) displays the conversion of the image to greyscale, this is needed to apply thresholding. Blurring was then achieved using the OpenCV function “blur()”, this function receives the image to be blurred and the size of the kernel that should be used, it then performs the box blur algorithm. Box blur was used as it is substantially faster than other methods such as the Gaussian blur. The project used a kernel of size 3 by 3, meaning each pixel was replaced by the average of itself and the eight pixels surrounding (Szeliski, 2022). The purpose of this blur is to reduce noise and obscure the weave of the textile. This is needed as light spots can confuse the thresholding algorithm and vertical lines in the weave can confuse the canny algorithm.

(c) displays the output of the thresholding algorithm, once again this functionality is offered by an OpenCV function. The “threshold()” function again takes the parameters: the greyscale image to be processed, the threshold value, a maximum value and a tag representing the type of thresholding to be applied. The project uses binary inverse thresholding, any pixel in

the image greater than the thresh hold value is set to 0 and any value less than the threshold value is set to the maximum value. A threshold value of 240 as the background pixels appeared as an off-white. The purpose of applying a threshold is to remove any vertical lines form the weave of the fabric leaving only the line between the background and fabric, removing any noise that may confuse the canny edge detection algorithm.

(d) the Canny edge detection algorithm again is offer by an OpenCV function. Canny edge detection works in stages, it first smooths the image to remove noise then computes the edge strength and edge direction. Edge strength and direction are found through the gradient of the image, a transition from black to white is considered a positive gradient. Edges produce a steep gradient and so can be located (Xu, 2017). The OpenCV function uses further processing to clean up the edge. The “Canny()” function returns a list of pixels that constitute the edge, the innermost pixels location is then used to crop the image lengthwise. (c) The dimensions of the cropped images were then found using the “.shape” attribute of a NumPy array and the corresponding mask image was cropped to the same dimensions (f).

2.2.2.3 Saving and labelling tiles

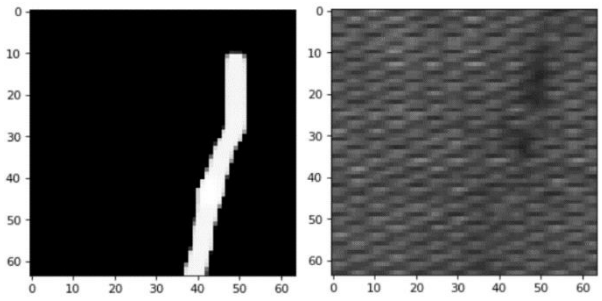


Figure 3 - example mask and image tile

Now the images had been cropped the individual tiles needed to be labelled and saved. As each image is broken down into approximately 540 individual tiles and the database consists of 245 images manual labelling would be infeasible. However, the cropped mask images can be used to determine which tiles include defects. The black pixels in the mask represent any pixel in the original image that is not part of the defect. Whereas white pixels take the place of the pixels that make up the defect. As the masks were cropped to the same dimensions as the original images the tiles line up exactly, figure 3 shows this relation.

If any mask tile contains any white pixels, its corresponding unmasked tile could safely be labelled as defective. In practice this method had one problem, even masks that contained only a single white pixel would be treated as defective, this is far too strict and could confuse later inspection methods. To remedy this only mask tiles that contained over 500 white pixels would be treated as defective. This increased the amount of data available from 245 large images to 909 defective tiles and 113,307 normal tiles. This segmentation will allow for more detailed testing and training of inspection methods in later sprints.

2.3 Sprint 2 - Beginning Inspection

2.3.1 Goals and Design Decisions

One goal of this sprint was to develop an analytical approach to inspection by comparing the properties of tiles to a known good tile. First mean and range would be calculated for each tile and compared to a known non-defective tile, while this is unlikely to work, if it does it would lead to extremely fast inspection times.

The main goal for this sprint was to develop an inspection technique using the principles of object detection. To achieve this, a way to reduce noise in the image must be developed as well as a way to find groups of similarly coloured pixels. OpenCV morphology and thresholding was chosen as the way noise would be reduced. This was decided as OpenCV's implementations of morphological transformation have become an industry standard and so have been optimised and offer GPU acceleration.

2.3.2 Implementation

2.3.2.1 Statistical Analysis

Two forms of statistical analysis were developed, the first was an attempt to use the range and standard deviation of the pixel values within a tile to deduce whether it if defective. The image data inside the tiles are stored in a NumPy array and so the standard deviation can be simply found using the NumPy function "np.std()". This was done after all tiles were converted to grayscale.

2.3.2.2 Colour Space Reduction

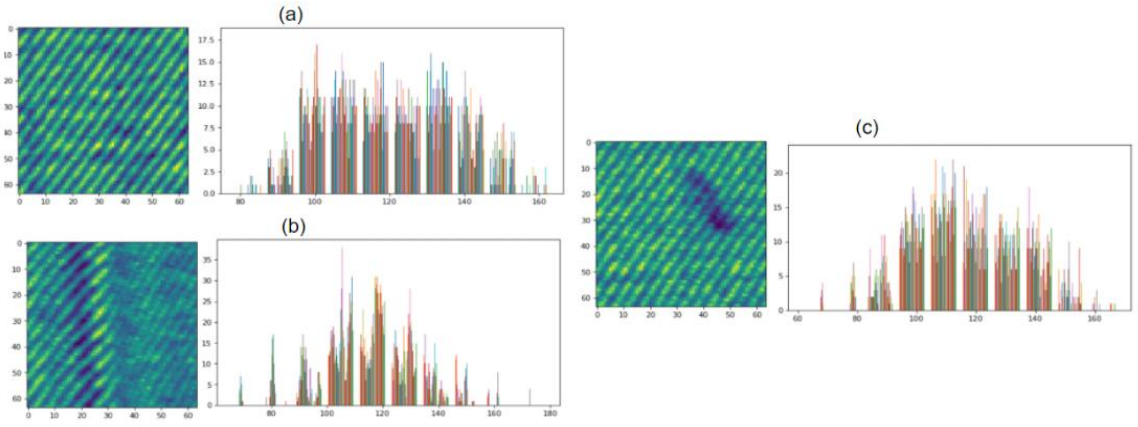


Figure 4 - tiles and histograms: (a) no defect, (b) crease, (c) knot

Figure 4 inspired the next approach, it was created using the “hist()” function from OpenCV, histogram (a) displays a more uniform spread of pixel values whereas (b) and (c) display spikes in the number of occurrences at certain pixel values. The idea was to reduce the colour space to a set of intervals and count the number of instances inside each interval. With the hope that defective tiles will have abnormally high pixel counts in certain intervals. This was achieved by rounding pixel values to the closest interval.

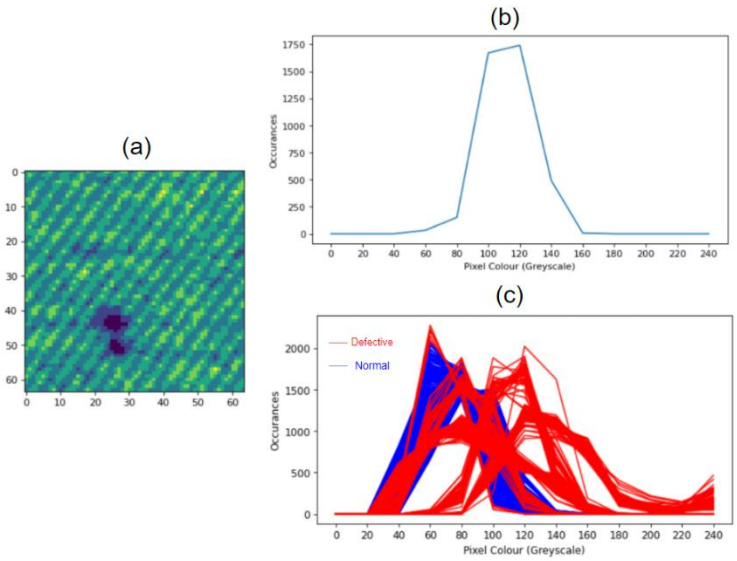


Figure 5 - (a) reduced colour space image, (b) plot of pixel occurrences at each interval, (c) occurrence plot over 500 images.

Figure 5 (b) illustrates that the majority of pixels in (a) have a value of either 100 or 120. Figure 5 (c) elaborates depicting a plot of 500 random tiles split between normal and defective. The plot clearly shows that many defective tiles differ greatly at certain pixel values. After further investigation it was decided that any reduced colour space tile that had over 2000 occurrences of any pixel value could be labelled as defective. Any tile with less than 1250 occurrence of value 80, more than 1800 occurrence of value 100, more than 1000 occurrence of value 120, or over 500 occurrences of any value over and including 140 could be labelled as defective.

2.3.2.3 Morphology and Contour Finding Inspection

Contour finding algorithms first need an image with very little noise. However as stated previously in the project, images of fabric are inherently noisy do to the weave. To reduce the noise several steps were taken.

First a light blur is applied as shown in code snippet 3 (appendix c) to shrink highlights and shadows and obscure the weave. This blur is achieved in the same way as blurring in the first sprint with the kernel size being controlled by the “lightBlur” parameter.

Highlights and shadows produced by inconsistent lighting needed to be removed. This was done by the function “setLims()”. This function is supplied with an image and a lower and upper bound. A new image array is created, and each pixel is appended, if any pixel has a lower value of than lower bound it was replaced, if any pixel has a value higher than the

upper bound it was replaced. Once every pixel had been processed the new image array is returned. The maximum and pixel threshold are both determined by the parameter “pixThresh”.

While this worked its sequential implementation proved too slow and instead OpenCV thresholding was used to remove highlights. The thresholding was done using the same “threshold()” function as sprint one, using a max and threshold value determined by “pixThresh” and the “THRESH_TRUNC” tag. This tag changes how the function worked as values lower than the threshold are now kept.

The next step in reducing noise was to apply image morphology. Several morphological transformations were used starting with erosion. Figure 6 demonstrates how erosion effects an image, the erosion algorithm attacks the boundaries of an object and leads to small objects being completely removed. It does this by moving a kernel (2d matrix) across the image and the anchor pixel, usually the centre pixel, is replaced by the minimum pixel value inside the area described by the kernel. The size of the kernel decides the amount of erosion that takes place. Dilation, also shown in figure 6, works in a very similar way however the anchor pixel is replaced by the maximum pixel value instead increasing the size of objects. A heavier blur is then applied to reduce or even eliminate any remaining background noise and increase the size of defects that were just shrunk by erosion.

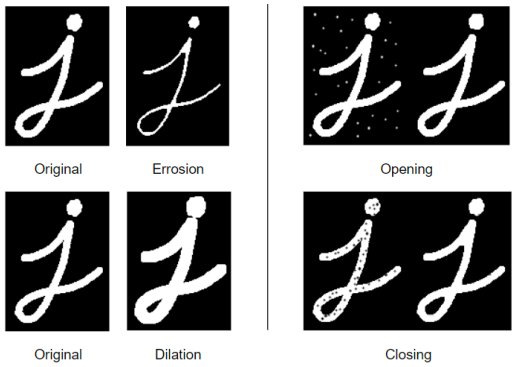


Figure 6 - morphology examples taken from OpenCV documentation.

Adaptive thresholding is then applied, this is very similar to the thresholding done previously in the project, but instead of using a global threshold value, chunks of the image are processed individually, and a local threshold value is computed for each.

Two forms of morphology are then applied consecutively: Opening and Closing. Opening applies erosion followed by dilation, again reducing noise and increasing the size of the defect. Closing applies dilation followed by erosion and fills in any gaps inside the defect that have been created from previous operations.

OpenCV contour finding is then used outline or bound the shape of the defect or defects. It works by finding all points on the boundary of an object. If these points, then form an unbroken loop a contour is found. This works best when used on binary images, where the object is white, and the background is black. This is another reason why adaptive

thresholding was applied and why the image is inverted after the morphological transformations. The end of the function simply checks if the size of any of the contours found is over a given value, if so the tile is deemed defective and an “exitCode” of 0 is returned.

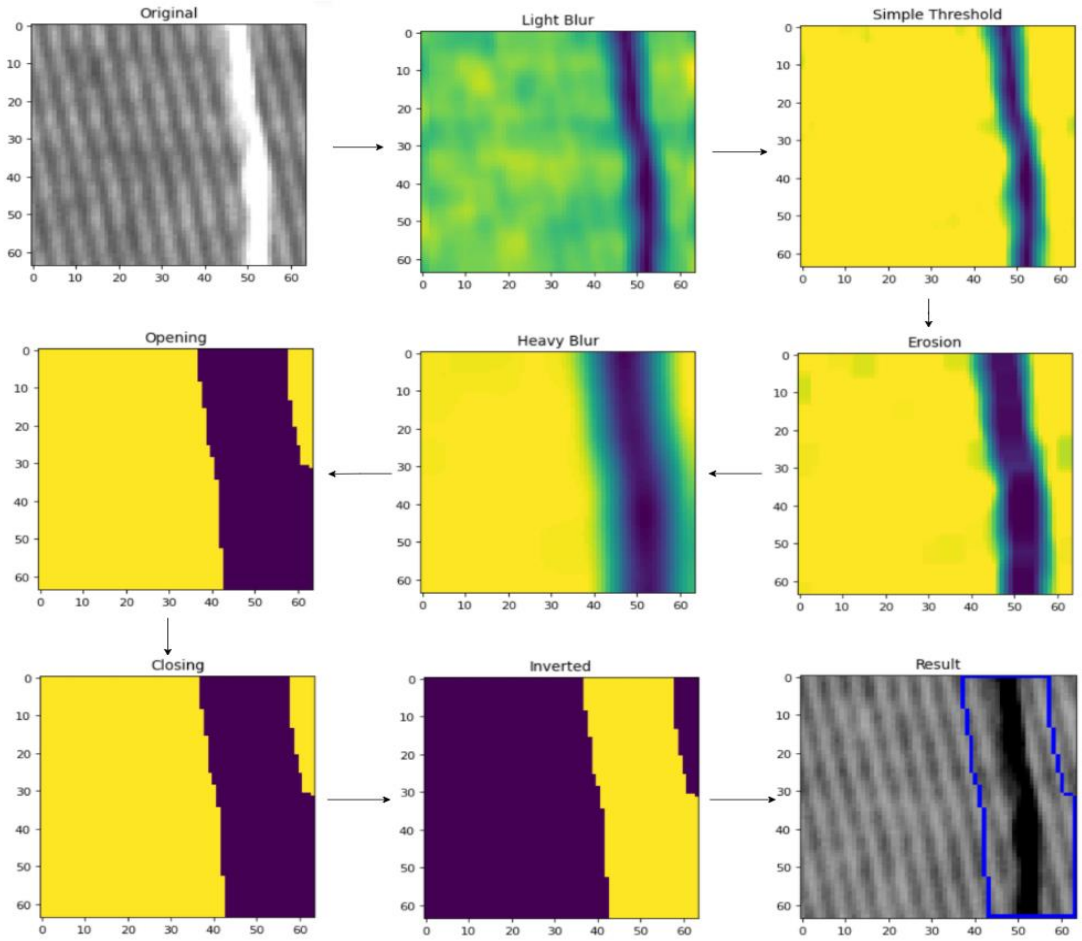


Figure 7 - morphology inspection steps

However, the was on major problem with this implementation, lighter defects would not be found. The initial thresholding was needed to remove highlights but would also remove lighter defects. This was easily fixed by running the “findDefect()” function twice each tile, inverting the tile after the first call, transforming any light defects into darker defects which could be detected. This was implemented in the “twoPassInspection()” function. Figure 7 displays the result of each stage of the inspection function, while some steps may seem unnecessary, they become more necessary in smaller or more complex defects.

2.4 Sprint 3 - Parameter Tuning and TensorFlow

2.4.1 Goals and Design Decisions

For this sprint, the goals were to tune the contour finding inspection technique to increase its accuracy and start development on an inspection technique leveraging deep learning. The tuning will be done by exhausting lists of parameters until the highest accuracy is found. While this technique will likely be slow, it will find the exact combination of parameters needed to achieve the maximum accuracy it is capable of.

For the third and last inspection technique developed in the project, it was decided that some form of neural network would be trained to categorise image tiles as either defective or normal. A convolutional neural network was the exact method chosen. Hence, the second goal for this sprint was to develop an appropriate layer structure for the CNN and then train it. The CNN would be trained on the tiles, as the defects represent a larger proportion of this smaller tile. It would also increase the number of images in each class in the training set.

Keras was used to create this CNN. While pre-built CNNs such as VGG 16 are available, many are too deep for this application. While deeper neural nets have more capacity to learn and so in some applications can reach higher accuracy, they require more training data than was available. They take longer when training and inferring and are more prone to overfitting. As the goal of the project was to create an application that could inspect several textiles, overfitting was actively avoided. Instead, a shallower network that is easier to train was chosen.

2.4.2 Implementation

2.4.2.1 Parameter Tuning

To make sure the second inspection technique was as accurate as possible multiple sets of parameters were looped through. So that this optimising could be done in a reasonable amount of time it was split up in to two sets of nested for loops. The first set consisted of two for loops that iterated over two lists [50,100,150,200,250,300] and [50,60,70,80,90,100,110,120,140]. The first list decides the minimum size of a defect and the second contains what pixel value should be used when thresholding. The upper and lower bounds of the list were found through manual testing.

The Second set of for loops consisted of three nested for loops. The first two iterated over values 1-10 and controlled the factor of the first blurring and erosion filters. The third iterated over values 1-20 and controlled the amount of the heavier blur.

For each set of parameters 1000 randomly selected normal tiles and all 900 defective tiles were put through the inspection function and the accuracy was calculated. The set of parameters that produced the highest accuracy was then saved for later use.

2.4.2.2 Creating the Convolutional Neural Network

In keras a CNN is produced by defining a number of layers. First a sequential model object was created this instructs keras to process in series, through the layers.

```
model = Sequential([

  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(32, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(64, activation='relu'),
  layers.Dense(32, activation='relu'),
  layers.Dense(1, activation='sigmoid')
])
```

Code snippet 4, model definition

Code snippet 4 demonstrates how the model was defined. First was a normalisation layer, this converts each pixel from a value ranging between 0 and 255 to between 0 and 1, helping the model converge faster. Next is a convolution layer, earlier in the project a brief definition of a convolutional layer was given. To elaborate, each filter consists of a kernel of weights, for the second layer the kernel is a 3 by 3 matrix filled with arbitrary weights.

This convolution layer contains 32 filters, same padding and using ReLU as its activation function. Each filter is moved over each pixel of the input, which as this is the first convolution layer is a normalise fabric tile, the dot product is then computed where the filter and input overlap (Wu, 2017). Same padding specifies that the output of the layer should be the same size as the input adding padding if needed, this helps to preserve features, making sure they can be processed by subsequent layers. This is then passed to a max pooling layer that reduces the size of the output while keeping the most important features, this process is repeated twice more using 64 filters.

After the third max pooling layer the output is passed to a flattening layer that converts the data from many dimensions to one. A series of fully connected layers is then used to reduce

the output until only a single score between 0 and 1 is produced. As the project will classify images in a binary fashion, either defective or normal, the sigmoid activation function was best suited for the final fully connected layer. The score represents the probability that the given tile is non defective. A score close to 0 means the tile image most likely contains a defect, a score close to 1 means the image likely does not contain a defect.

Table 4 in appendix c details the size of the data sets the model was trained on, all the defective tiles were used. Whereas only 5000 of the 113,307 non-defective images were used, to reduce bias. The model was then trained over 30 epochs during which the weights inside the filters were adjusted, each epoch was evaluated on the validation set.

In an effort to produce a second more general model with possibly increased accuracy due to a larger training set, data augmentation was used. Keras allows this done through the addition of two layers. A random flip layer and a random rotation layer as the names suggest a random flip layer randomly flips some images horizontally and the random rotation layer randomly rotates some images.

The model can then be used to predict the class of an tile by calling ‘model.predict()’ on either a single or an array of tiles. The confidence of each prediction was found by subtracting the absolute difference between the prediction and the closest class (either 0 or 1) from 1 then multiplying by 100.

2.5 Sprint 4-Inspection Application and Human Testing Application

2.5.1 Goals and Design Decisions

The goals of this sprint were to develop a prototype inspection application and an application to test the accuracy of human inspection. The inspection application will use the morphology inspection technique and the convolutional neural networks trained. It must allow the user too:

- Import a set of images to be inspected.
- Choose which inspection technique is used.
- If the neural network inspection is chosen, the user can choose which model to use.
- Control the parameters of the chosen inspection technique.
- View the defects found.

Images were imported rather than the application being connected to a live camera feed. This enables the user to quickly inspect a variety of fabric without having to reset a machine.

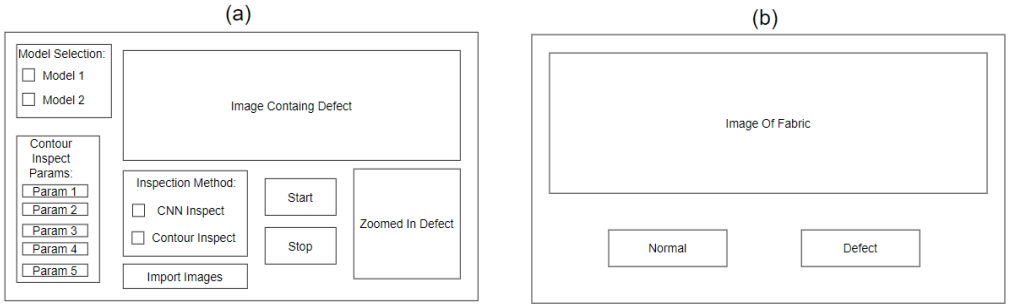


Figure 8 - (a) wireframe of inspection application, (b) wireframe of human testing application

While this would not be ideal in a production environment it is optimal for testing. The application to test human inspection should be simple, it should display several images in a random order, some with defects and some without. For each image the user should select whether they believe the image to include a defect or not. The application should display how the user performed so it can be recorded. This application should be as minimal as possible as to not take up development time that could be used form more important tasks. I used PyQt to develop both as the use of the designer tool allows for rapid development and it offers an intuitive system to connect events to functions.

2.5.2 Implementation

2.5.2.2 Inspection Application

QT designer was used to create a basic layout for the application, consisting of several Qt widgets: labels (used to display both images and text), check boxes, spin boxes (allow the users to input numerical data) and buttons. Once converted into a python file these buttons, check boxes and spin boxes could be connected to functions.

The image data is loaded after the user specifies the folder they are stored in, no background removal is incorporated into this prototype so any images must have been pre-processed beforehand. The images are moved into main memory by listing all the file names in the specified folder and calling the OpenCv “imread()” function. This loads all the images at once, enabling faster access when inspecting at the cost of high hardware utilisation. The model used for the neural network inspection technique was loaded at the applications startup.

Inspection begins once the start button is pressed, an image is tiled and each tile is run through the selected inspection technique, the array of the defective tiles found is then returned and displayed to the user. Whenever the “next” button is pressed the next defect found is displayed until all the defects in the list have been shown. “Start” can then be repeatedly pressed to inspect images in turn, each time displaying the defective tiles found.

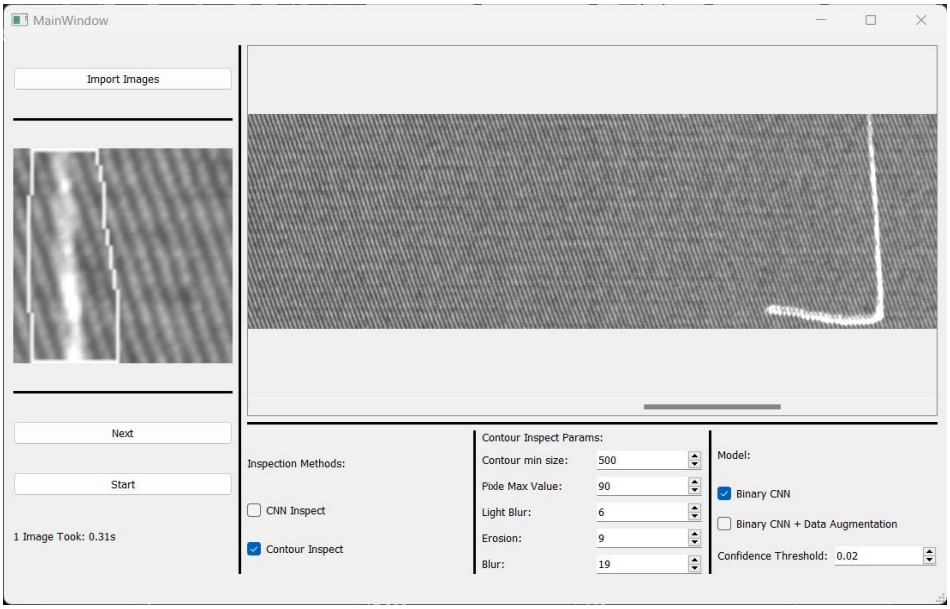


Figure 9 - Inspection Application User Interface

2.5.2.3 Human Testing Application

This application was developed very similarly to the last. The filenames from two predetermined folders are stored in a list, they contain names of 8 defective images and 8 normal images. The user is then presented with a simple screen only containing a start button. Once pressed the application uses the time library of python to obtain the current time.

The start button is then hidden, the first image to be inspected by the user is displayed and the application waits for the user to press either the “defect” or “normal” button. When pressed the application checks whether the file name of the image being displayed belongs to the folder of defective or normal images and increments the appropriate variable out of “normalCorrect”, “normalIncorrect”, “defectCorrect”, “defectIncorrect”. Once all 16 images have been inspected these variables are displayed to the user along with the accuracy they achieved.

Chapter 3
Results

The previous chapter focused on the implementation, but little thought was given to how these implementations could be analysed. This chapter will present the results gathered and explain what methods were used to gather them. All testing was conducted using functions available from Matplotlib and Sklearn.

3.1 Key Inspection Metrics

When analysing fabric inspection methods there exists three key metrics, general accuracy, catch rate and misidentification rate. A general accuracy can be calculated by the number of images containing defects correctly identified plus the number of images not containing defects correctly identified all over the total number of images. Catch rate can be described as the number of defects caught divided by the total number of defects. Similarly, misidentification rate is the fraction of non-defective images that are identified by the inspection technique as defective. False positive rate and true positive rate will be discussed throughout the section. In terms of this project the false positive rate is equivalent to misidentification rate and true positive rate equivalent to catch rate.

While general accuracy can be informative when directly comparing inspection techniques it can also be misleading. Two techniques with the same accuracy may offer vastly different value to a fabric producer. A high catch rate is usually desired to ensure few defects make it through production. However, if this is achieved by allowing a high misidentification rate then production could be slowed by unnecessary stoppages as defects will need to be recorded or removed.

3.2 Human Inspection

To evaluate the effectiveness of the inspection methods developed and weather they would be advantageous to a fabric producer to implement, the project needed a baseline accuracy of current human inspection. The previous chapter explains how a human testing application was developed and how the testers used the application. After 8 testers analyse all 16 images of defective and non-defective fabric they were gathered and placed in table 5 (appendix c). All tester names were kept anonymous, and all were shown the same 5 defective and 5 non-defective images before the test, giving them a baseline understanding. All testers were also supplied with consent forms (appendix D) and a copy of the project information form (appendix E).

The average accuracy obtained from the testers was 68.75% with a catch rate of 75% and a misidentification rate of 37.5%.

3.3 Statistical Analysis using Standard Deviation and Range

When applying statistical analysis using the standard deviation and rage of the pixel values in greyscale tiles the following tables were produced:

Normal Tile Number	Standard Deviation	Range
1	16.448	81
2	16.736	85
3	11.917	70
4	18.832	94
5	27.395	128
Average:	18.266	91.6

Defective Tile Number	Standard Deviation	Range
1	16.289	114
2	16.839	107
3	17.569	107
4	17.099	113
5	14.642	99
Average:	16.488	108

Table 1 and 2 - ranges and standard deviations of example tiles

3.4 Statistical Analysis using Reduced Colour Space

As described in the implementation chapter an attempt was made to detect defects in a tile based on the number of pixels with a certain value, once the colour space of the tile was reduced.

Figure 10 displays a confusion matrix formed by the statistical analysis detection method performed on 1000 known good tiles and all 909 defective tiles in the dataset. A confusion matrix is a particularly useful way of analysing the effectiveness of inspection methods, this is as all three of the key inspection metrics above can be ascertained from this single plot.

The data for this plot was generated by repeatedly running the “reduceInspect()” on two sets of tiles 1000 normal and 909 defect tiles. Two lists were created one consisting of targets and one of predictions. The targets represent the known state of images, that being either defective or non-defective/normal. For each image if it belongs to the set of normal images 1 is appended to the targets, 0 if it belongs to the set of defective tiles. If “reduceInspect()” function returns true for a specific image the image is predicted to be a defect and so a 0 is appended to the predictions list, a 1 is appended if the function returns false. Using the Sklearn “ConfusionMatrixDisplay.from_predictions()” function a confusion matrix can be constructed from these targets and prediction. It achieved an accuracy of 49.475%, a catch rate of 94.49% and a misidentification rate of 91.6%.

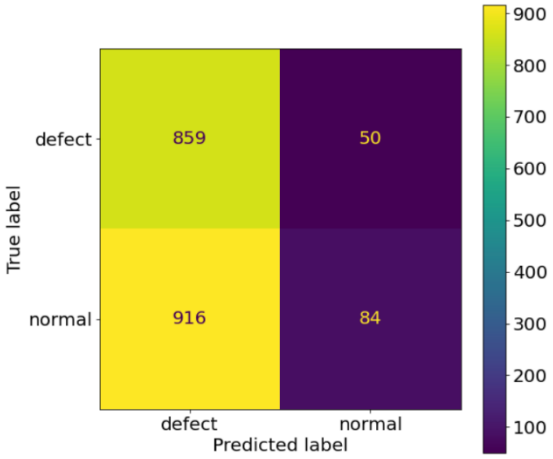


Figure 10 - reduced colour space inspection confusion matrix

The average execution time of the “reduceInspect()” function was 0.118 seconds, meaning that a master image containing on average 465 tiles would take 54.87 seconds to fully inspect. This average was found using the execution times from all 1909 tiles inspected to create the above confusion matrix.

3.4 Morphology and Contour Finding Inspection

During the project 2 forms the morphology and contour finding inspection methods were produced, one using a the ‘findDefect()’ function and one using the ‘twoPassInspection()’ which as documented in the implementation applies the former function, inverts the image and applies the function again with the hope of finding defects missed in the first pass. This section will present the results for both implementations. All results found used the same testing strategy as reduce colour space inspection.

3.4.1 Single Pass Inspection

Before parameter tuning single pass inspection the function used a light blur kernel size of 5 by 5, a global threshold value of 110, an erosion kernel size of 5 by 5, a heavy blur kernel size of 15 by 15 and a minimum defect size of 100 pixels.

Using this set parameters, it achieved an accuracy of 75.327%, with each tile taking 0.000228 seconds to inspect. Meaning in the worst case a master image would take 0.106 seconds to fully inspect. Figure 11 displays the confusion matrix obtained through this testing. With a catch rate of 72.06% and a misidentification rate of 21.7%.

While, parameter tuning was done using the two-pass inspection method, when applying the best parameters found to single pass inspection the accuracy increased to 81.0896%. Obtaining 78.77% catch rate and 16.6% misidentification rate.

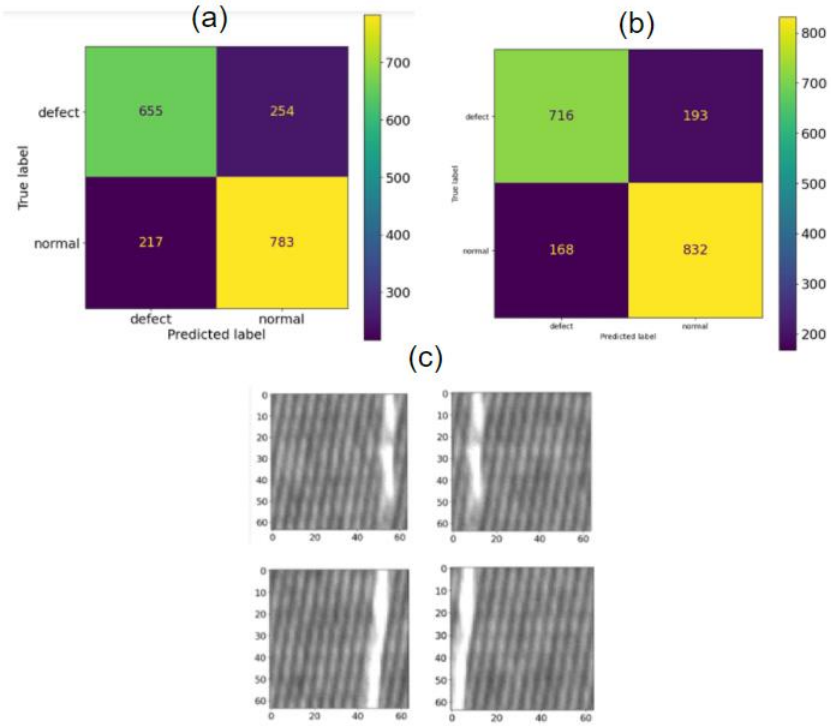


Figure 11 - (a) single pass confusion matrix using set parameters, (b) single pass confusion matrix using tuned parameters, (c) example defects not identified by single pass inspection.

3.4.2 Two Pass Inspection

Before parameter tuning using the same parameters as single pass inspection it achieved an accuracy of 76.637%, catch rate of 78.8%, misidentification rate of 27.1% and an execution time per tile of 0.000316 seconds. Meaning in the worst case an master image would take 0.147 seconds to fully inspect.

After parameter tuning was complete the best parameters were found to be: a global threshold of 100, a light blur kernel size of 6 by 6, a heavy blur kernel size of 15 by 15, an erosion kernel size of 6 by 6 and a minimum defect size of 100 pixels. With these parameters this method achieved an 84.294% accuracy, catch rate of 89.99%, misidentification rate of 21.4%

However, these accuracies cannot be directly compared to those obtained through human inspection. This is because humans inspect on complete images of fabric and not the individual tiles. To obtain an accuracy that can be fairly compared to that of human testing the images used in human testing were tiled, each individual tile was then ran through the tuned 'twoPassInspection()' function and if even a single tile returned as defective the entire image was deemed defective. The same testing scheme was then applied using targets and predictions. On the same set of images human testers inspected, this technique achieved an accuracy of 56.25%, catch rate of 87.5%, misidentification rate of 75% Figure 24. On the entire data set it achieved 49.19%, catch rate of 92.4%, misidentification rate of 83%.

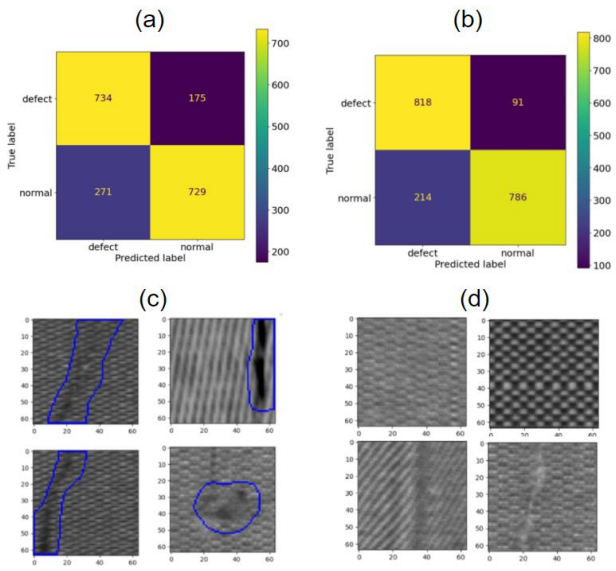


Figure 12 - two pass inspection. (a) confusion matrix with set parameter, (b) confusion matrix using tuned parameters, (c) example output of identified defects, (d) example defects not caught.

3.5 Convolutional Neural Network

During this section a number of other metrics were considered such as f1 scores, these can be used as a measure of accuracy and are less prone to bias than general accuracy.

The first CNN produced used no data augmentation and after being trained on 3,719 non-defective tiles and 673 defective tiles, figure 16 (appendix c) displays the training and validation accuracy and loss throughout training. When predicting on the test set an accuracy 98.76%, catch rate of 95.45% and a misidentification rate of 0.63%. Figure 16 displays the f1, precision and recall scores. With recall for the defect (0) class being equivalent to catch rate.

An ROC (receiver operating characteristic) curve can be used to find a classification threshold that minimises the rate of false positives while maximising true positive rate. These curves were created using sklearn 'metrics.roc_curve()', the metrics returned by this function pointed to an optimum threshold of 0.96. When using this value to determine classification, accuracy reduced to 98.0%, catch rate increased to 96.6% and a misidentification rate increased to 1.68%. The model was further tested with a threshold value of 0.15 designed to

minimize misidentification rate, lowering it too 0.42% and increasing accuracy to 98.76% but in turn decreasing catch rate to 94.4%.

However as mentioned before these measures are calculated when classifying individual tiles, when applied to full images using the rule that a single defective tile deems the entire image as defective. The highest accuracy obtained when inspecting the same image set as the testers was 62.5% and was obtained when using the threshold of 0.15. It achieved a catch rate of 75% and a misidentification rate of 50%. When applied to all images the best accuracy was gained using the threshold 0.15 at 71%.

After 30 epochs of training with data augmentation applied on the same dataset produced an accuracy of 98.23%, catch rate of 89.8% and a misidentification rate of 0.21%.

The optimum threshold produced by the roc curve was 0.928. When applied accuracy was unchanged and misidentification rate rose to 1.68%, however catch rate increased to 98%. When testing using the same full images as the testers the best result with accuracy 56%, catch rate of 38% and misidentification rate of 25% were produced with a threshold of 0.15. Strangely using the 0.928 threshold, produced a 100% misidentification rate and catch rate. This was observed on both the set of images shown to testers and all the images in the dataset. When using a more accurate 0.15 threshold on all image's accuracy increased to 66%.

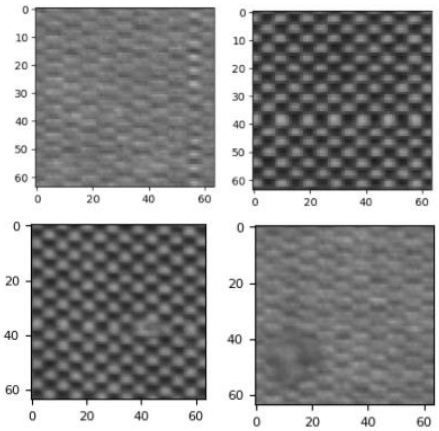


Figure 13 - exampled defects not caught by optimum CNN inspection (no data augmentation at a threshold of 0.15).

Each full image took an average of 3.7 seconds to inspect meaning all 16 tester images would have taken 59.2 seconds to inspect.

3.6 Prototype Inspection Application

As this application was developed in only a single sprint it was not tested extensively, it was only designed to offer a proof of concept of what an inspection application may look like. However, the application was shown, and a demonstration was given to all 7 individuals that took part in the human inspection test. Five agreed that operating the application was intuitive and that it presented the feedback they would expect. Two questioned why the application inspected a single image at a time, pointing out that it would be tedious to use for any extended amount of time. This was done to allow for debugging but would not be optimal in a production environment.

Chapter 4
Discussion

This chapter will discuss and evaluate the results found in the previous chapter. Here the methods will be compared both too each other and the value they could deliver if implemented in a production environment.

4.1 Evaluation

4.1.1 Human Inspection

The application created to gather the results of human inspection wasn't optimal, human inspection would not be completely visual and larger regions could be inspected at a single time. Despite this the accuracy found almost completely mimicked the real human inspection which varies from 60%-70%. While some individuals achieved higher accuracy in my application, they were only tested on 16 images, if this was expanded upon the tedious nature of the job would have likely lowered their accuracy.

4.1.2 Statistical Analysis Based Inspection

At first statistical analysis seemed promising, when comparing the histograms in figure 4.b and 4.c to the histogram in figure 4.a many, of the bins in the defect histograms have a higher number of occurrences. This seems reasonable as defects usually present as a large mass of similarly coloured pixels, for these reasons it was developed further. This was further reinforced by table 3 and 4 which showed a clear pattern that the standard deviation of a tile was lower if the tile included a defect. However, these defective tiles also had a higher range, these measures seem to contradict themselves. There also existed outliers that broke both conventions in either table hence a detection method was not produced using only these metrics. It is likely that both the histograms and tables are misleading due to their small sample size and the project should not have treated them as representative of the larger dataset. Figure 16 clearly indicates that when colour space is reduced many defective tiles differ greatly at certain pixel values. For example, many defective tiles contained over 1500 pixels with value 140 whereas no non-defective tile had over 100 pixels at the same value.

However, despite the seemingly clear distinction between defective tiles and non-defective tiles a low accuracy was produced by the reduced colour space inspection technique. A 49.4% accuracy is what would be expected from an algorithm that randomly decided a class, the key problem was the extremely high catch and rate misidentification rate. This indicates

that the rules that decided defect classification were incorrectly chosen, classifying almost every image as defective. The nature of the fabric chosen for the project may have also played a part in reducing the accuracy of this method as shadows and highlights distort monotone images far more than colour images. A similar method has been used successfully on colour fabric. This is as the red, green, and blue channels can be inspected separately or compared to each other (Rasheed, 2020).

Overall, this technique produced results that were worse than human inspection with the time to inspect each full image of 54.87 seconds due to the sequential implementation when reducing the image's colour space. It would be slower than human inspection by a factor of 7.5.

4.1.3 Morphology and Contour Finding Inspection

This inspection method proved to be far more effective than a statistical approach with both single and two pass inspection obtaining a higher accuracy. The largest improvement being the misidentification rate dropping to 21.7% while keeping a relatively high catch rate even before the parameters were tuned. Single pass inspection ignores lighter defects, while two pass inspection eradicates this issue, it does not necessarily mean it is superior in a commercial environment. If the specific fabric to be inspected is prone to neps, fuzzy balls and knots, and lighter defects such as broken yarns were rare then the faster speed of single pass inspection may be desirable. The times to inspect full images are very similar so in most cases the higher catch rate of two pass inspection will be optimal.

Unfortunately, even with parameters tuned to the misidentification rate on full images is unacceptably high leading to lower accuracy then human inspection. This is due to a single tile misidentification rate of 21.4% meaning that out of an average of 465 tiles in a single image approximately 100 tiles will be miss identified as defective. Testing further show that the method struggles with defects that have similar colour to the average pixel. Hence this current implementation could not be used for fully autonomous inspection. A key advantage this form of inspection offers over both a statistical and deep learning approach is shown by its output (figure 12) as it returns the contour surrounding a defect, this can be used as a visual aid to assist a human inspector or could be used to classify the defects found.

4.1.3 Convolutional Neural Network Inspection

A convolutional neural network inspection approach performed far better than either of the two previous methods however it was by no means perfect. As seen by the high training accuracies and slightly lower validation accuracies the first model was slightly overfit, however the almost identical validation and training accuracies and losses produced by the

second model showed that the data augmentation worked as intended eradicating this overfitting and creating a more general inspection technique. Despite this on the test set both achieved a very similar accuracy with the catch rate of the second model being over 5% lower. It is likely that this second model is underfit due to the random rotation layer included, all images of fabric must have the weave running perfectly vertically any rotation in this would be classed as a defect. By including this rotation, we are confusing the model by telling it to ignore a rotation in the weave, in turn lowering catch rate.

Unfortunately, this high individual tile accuracy is not carried into full image inspection. Here it performed as well as most human tester, it did produce a slightly higher catch rate than humans but also a higher misidentification rate. This likely due to bias in the training data, cotton incorporated splits defects into a number of classes. Three of which are line, pattern and isolated. As line and pattern defects are large, when the full images was segmented into tiles a single defect would result in a number of tiles. Unfortunately, the same could not be said for small, isolated defects such as knots. We can see in figure 13 that this bias negatively affected the ability of the model to recognise these isolated defects reducing catch rate. Another reason for the lower accuracies on full images is that even with sub 1% misidentification rate, out of a possible 465 tiles per image it is likely at least one tile will be miss identified as defective tainting the entire image.

With an average full image inspection time of 3.7 seconds and an accuracy of 71% the solution performs almost identically to human inspectors, and so this inspection technique could be a candidate for use in a fully automated system. It is worth considering that this testing was done on a system without a GPU. With the highly parallel nature of a CNN, inspection times could be reduced by up to a factor of 100 (Chaubal, 2022). This would make it more effective then human inspection.

4.1 Conclusions

In conclusion all the project’s aims were achieved, three inspection techniques were produced as well as prototype application to enable users to apply two of these methods. All were compared not only each other but to the current human inspection solution and each was found to have value in commercial use. Statistical analysis if used with a different dataset of coloured images could be used to autonomously inspect, morphological inspection while currently not achieving the accuracy needed to autonomously inspect. Its speed means could be a useful tool to aid human inspectors increasing their accuracy. Finally, CNN inspection showed promise in its ability to outperform all other explored forms of inspection, with little further development needed to reach this. All testers saw the value in

the final application produced and understood how it may be used, but pointed that inspecting multiple images at once would increase its value.

4.2 Ideas for future work

If the project were to be continued most effort should be placed into further increasing accuracy and refining the inspection application. There a many ways accuracy could be increased, through a statistical approach plotting average pixels values along the x and y axis looking for peaks and troughs would likely be more effective than the methods produced during the project, it would also allow the location of defects to be found. Morphology inspection could also be improved as size of the kernels used for opening and closing were static and not tuned, this could added and would hopefully improve accuracy. In both CNN and morphology inspection if even a single tile was seen as defective the entire image was treated as such, a condition could be placed on this needing a certain number of tiles to be defective, possibly reducing misidentification rate.

Several small tweaks like ones above could be implemented but the greatest improvement to accuracy would be made by increase size and variety of the dataset. With hundreds or even thousands more labelled images the first two inspection techniques could be tuned and optimised more effectively. This would have the largest impact when training new models. A respectable accuracy was achieved with a small data set so is not unreasonable to a expect and excellent accuracy with a much larger and less biased data set. The images used by CNNs could be reduced in size from 64 by 64 to 16 by 16, as long this does not remove defects it could lead to similar accuracy with vastly reduced inspection times. If all these changes worked well then, a combined approach could be explored. Using image morphology tuned for high catch rate as an initial screening step thanks to its high speed, a CNN model could then inspect only tiles it flags as suspicious lowering the overall time to inspect.

In the long term the prototype inspection application should be moved from a system where a batch of images loaded in then inspected to one connected to a live camera. This would enable it to inspect images in real time and would be the final step needed to create a full autonomous inspection system.

List of References

Shaker, K., Umair, M., Ashraf, W. and Nawab, Y. (2016) Fabric manufacturing. Physical Sciences Reviews, Vol. 1 (Issue 7), pp. 20160024. https://doi.org/10.1515/psr-2016-0024

P. A. Smith B.Sc., Ph.D., F.T.I. (1969) YARN PRODUCTION AND PROPERTIES, Textile Progress, 1:2, 1-117, DOI: 10.1080/00405166908688985

Adanur, S., 2020. Handbook of weaving. CRC press.

Ray, S.C. ed., 2012. Fundamentals and advances in knitting technology. CRC Press.

Perkins, W.S., 1991. A Review of Textile Dyeing Processes. Textile Chemist & Colorist, 23(8).

Silvestre-Blanes, J., Albero-Albero, T., Miralles, I., Pérez-Llorens, R. and Moreno, J., 2019. A public fabric database for defect detection methods and results. Aitex. [Online]. [Date Accessed]. Available from: https://www.aitex.es/afid/

Silvestre-Blanes, J., Albero-Albero, T., Miralles, I., Pérez-Llorens, R. and Moreno, J., 2019. A public fabric database for defect detection methods and results. Autex Research Journal, 19(4), pp.363-374

Nateri, A.S., Ebrahimi, F. and Sadeghzade, N., 2014. Evaluation of yarn defects by image processing technique. Optik, 125(20), pp.5998-6002.

Malek, A.S., Drean, J.Y., Bigue, L. and Osselin, J.F., 2013. Optimization of automated online fabric inspection by fast Fourier transform (FFT) and cross-correlation. Textile Research Journal, 83(3), pp.256-268.

2023. Cotton Works. [online]. [Accesses 20 April 2023]. Available from: https://www.cottonworks.com/en/defect/0121/

Lim, S.L. 2018. 23 Fabric Defects To Look Out For During Fabric Inspection. [online]. [Accesses 20 April 2023]. Available from: https://www.intouch-quality.com/blog/5-common-fabric-defects-prevent

Textile-tutorials. 2018. Types of Fabric Defects with Causes and Remedies. [online]. [Accesses 20 April 2023]. Available from: https://textiletutorials.com/types-of-fabric-defects-with-causes-and-remedies/#:~:text=The%20weft%20curling%20defect%20is,the%20surface%20of%20woven%20fabric

Textile Sphere. [online]. [Accesses 20 April 2023]. Available from: https://www.textilesphere.com/2021/03/weaving-yarn-processing-fabric-

defects.html#:~:text=Weft%20Crack%3A,due%20to%20absence%20of%20weft.&text=Broken%20filament%3A,the%20main%20yarn%20are%20broken

Lord, P.R., 1982. Weaving: Conversion of yarn to fabric (Vol. 12). Woodhead Publishing.

Chin, R.T. and Harlow, C.A., 1982. Automated visual inspection: A survey. IEEE transactions on pattern analysis and machine intelligence, (6), pp.557-573.

Learned-Miller, E.G., 2011. Introduction to computer vision. University of Massachusetts, Amherst.

Belkhir, N.B. 2020. Textile Defect Detection. Kaggle. [Online]. [Date Accessed]. Available from: https://www.kaggle.com/datasets/belkhirnacim/textiledefectdetection

Szeliski, R., 2022. Computer vision: algorithms and applications. Springer Nature.

Xu, Z., Baojie, X. and Guoxin, W., 2017, October. Canny edge detection based on Open CV. In 2017 13th IEEE international conference on electronic measurement & instruments (ICEMI) (pp. 53-56). IEEE.

Brosnan, T. and Sun, D.W., 2004. Improving quality inspection of food products by computer vision—a review. Journal of food engineering, 61(1), pp.3-16.

Haralick, R.M., Sternberg, S.R. and Zhuang, X., 1987. Image analysis using mathematical morphology. IEEE transactions on pattern analysis and machine intelligence, (4), pp.532-550.

Xie, G. and Lu, W., 2013. Image edge detection based on opencv. International Journal of Electronics and Electrical Engineering, 1(2), pp.104-106.

Massaro, A., Vitti, V. and Galiano, A., 2018. Automatic image processing engine oriented on quality control of electronic boards. Signal & Image Processing: An International Journal (SIPIJ), 9(2), pp.1-14.

Zhou, Z.H., 2021. Machine learning. Springer Nature.

O'Shea, K. and Nash, R., 2015. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Wu, J., 2017. Introduction to convolutional neural networks. National Key Lab for Novel Software Technology. Nanjing University. China, 5(23), p.495.

Rasheed, A., Zafar, B., Rasheed, A., Ali, N., Sajid, M., Dar, S.H., Habib, U., Shehryar, T. and Mahmood, M.T., 2020. Fabric defect detection using computer vision techniques: a comprehensive review. Mathematical Problems in Engineering, 2020, pp.1-24.

2023. Cotton Incorporated. Fabric Defects Classification. [online]. [Accesses 10th May 2023]. Available from: https://www.cottoninc.com/quality-products/textile-resources/fabric-defects-classification/

Chaubal, C.C. 2022. Choosing a Server for Deep Learning Inference. [online]. [Accesses 10th May 2023]. Available from: https://developer.nvidia.com/blog/choosing-a-server-for-deep-learning-inference/#:~:text=Results%20from%20the%20MLPerf%202.0,need%20a%20real%2Dtime%20response.

Appendix A
Self-appraisal

<This appendix must contain everything covered under the 'self-appraisal' criterion in the mark scheme. Although there is no length limit for this section, 2-4 pages will normally be sufficient. The format of this section is not prescribed, but you may like to consider the following sections and subsections.>

A.1 Critical self-evaluation

Overall, I am satisfied with how the project went as it achieved its aims. Unfortunately, accuracies were lower than anticipated, I attribute this to spreading myself too thin. Perhaps spending all my focus on one inspection technique would have afforded me enough time to develop a more comprehensive final inspection application. This would have limited my ability to compare different inspection techniques and may have led me down a dead end, so I do believe my approach to the project was correct but lacked effective time management.

While I have had experience with coding projects a report of this scale was new and posed challenges I wasn't fully prepared for. From the start of the project, I had a clear vision the steps involved and exactly what I wanted to achieve. The background research conducted reassured me that most of these goals were achievable, however when finding a dataset to use there were few options with the AITEX dataset being the most promising but still small.

After initial prototyping and the completion of deep learning tutorials I realised that my ideas to not only identify defects but then classify them was unachievable within the time span and with the data available. In the end I am glad I limited myself to defect identification as it allowed me to explore the subject in more detail and produce an effective final solution. The initial sprints when well and development was fast, this ended when testing began in April which took far longer than anticipated and place me behind when starting the report. In hindsight the report should have been written in tandem with development.

I am happy with code produced and the results gathered, I believe the code was written to a high standard, all inspection methods function as expected and the final application was intuitive to use. I would have liked enough time to perform more tuning on the inspection methods. This being said I not unsatisfied with any of the results as the project's focus was to compare the methods, not achieve the highest accuracy possible.

A.2 Personal reflection and lessons learned

I believe this project has allowed grown as a computer scientist, it has developed my ability to critically evaluate the academic work of others to help build my own ideas and avoid pitfalls. It allowed to become comfortable with a number of industry standard computer vision and deep learning tools and libraries. Cementing my interest in these areas and building a skill set I can take into employment after graduation. In particular, it has taught me how to properly assess deep learning techniques, allowing me to identify overfitting and bias and to critically think about role of data augmentation, both its benefits and disadvantages.

Since the start of university, I have been confident in my ability to write essays and reports quickly. This project has questioned whether I was overconfident in this regard and has given a new respect for those who have masted time management and can minimise procrastination, areas of myself I have now identified need work. I now understand that the importance of documenting software development, it allows you to analyses the decisions made during implementation and whether different more effective methods exist.

A.3 Legal, social, ethical and professional issues

A.3.1 Legal issues

The project extensively uses a number of libraries including matplotlib, tensorflow, OpenCV and sklearn all of which are free to use in any personal and commercial sense without limitation and so covers all content within the project.

All training, testing and general development was done using the AITEX public defect database which is free to use for research purposes. It was created by Javier Silvestre-Blanes, Teresa Albero-Albero, Ignacio Miralles, Rubén Pérez-Llorens and Jorge Moreno.

Dataset:

https://www.aitex.es/afid/

Paper:

https://www.autexrj.com/cms/zalaczone_pliki/[23000929_-_Autex_Research_Journal]_A_Public_Fabric_Database_for_Defect_Detection_Methods_and_Results.pdf

If this project was further developed for commercial use the consent would need to be gained from the creators of the dataset or a new model would need to be trained from a new set of images.

A.3.2 Social issues

As the current project is purely research focused it poses no social issues. However, if it was taken forward and used commercially it could have large social ramification. As with any form of automation there is the possibility for machines to replace human workers and increase economic inequality.

A commercial application could increase defect catch rate over current solutions increasing the amount of waste generated.

A.3.3 Ethical issues

A number of testers were used to gain an insight into the accuracy of current solution. To ensure the project remained ethical all testers were given a project information sheet (appendix E) and signed a consent forms (appendix D). They were shown an ample number of example defective and non-defective images, allowing them to make comfortable and informed decisions when being evaluated.

Testers could withdraw any time and the results gather form them were anonymised to protect their privacy.

A.3.4 Professional issues

The project was developed with goof professional practices in mind, git lab version control was used through, and most coding was done through jupyter notebook allowing it to be documented and separated under headings. All external material has been correctly credited.

Appendix B
External Materials

- The AITEX fabric image dataset was used throughout the project to test and train: https://www.aitex.es/afid/
- OpenCv morphology documentation was used to inform code decisions: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html
- OpenCv threshold documentation was used to inform code decisions: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html
- The Tensor Flow CNN tutorial was used to understand how a CNN should be implemented and tested: https://www.tensorflow.org/tutorials/images/cnn

Appendix C

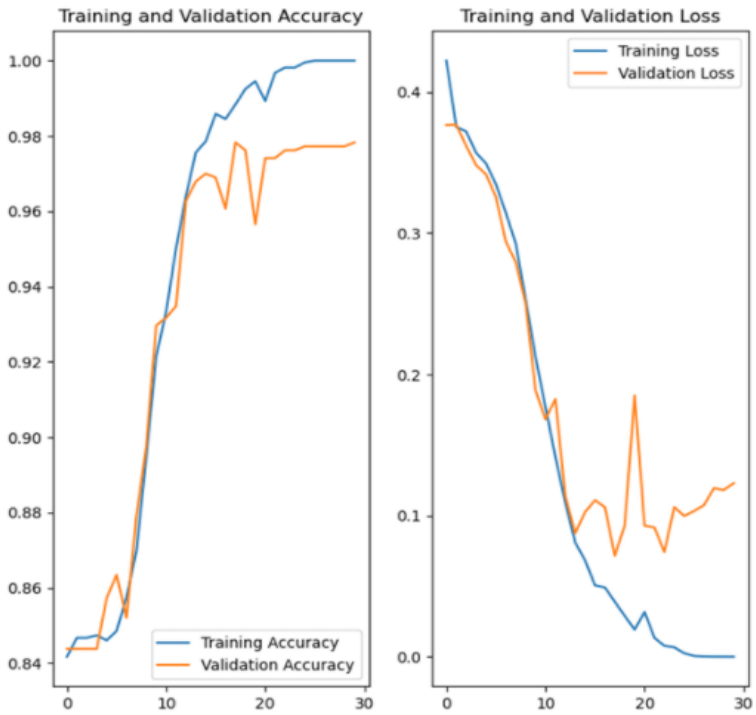
Additional Tables, Figures and Code Snippets

Set	Number of Non-Defective Images	Number of Defective Images
Training	3719	673
Validation	816	151
Test	476	88

Table 4 - Dataset Breakdown

Tester:	Accuracy (%):	Defects Correctly Identified:	Images Misidentified Defects:	Non-Defects Correctly Identified:	Images Misidentified Non-Defects:	Time Taken (secs)
1	62.5	6	4	4	2	158
2	62.5	5	3	5	3	95
3	75.0	7	3	5	1	147
4	87.5	6	0	8	2	126
5	62.5	6	4	4	2	119
6	68.75	5	2	6	3	112
7	62.5	7	5	3	1	58
Average	68.75	6	3	5	2	116.4

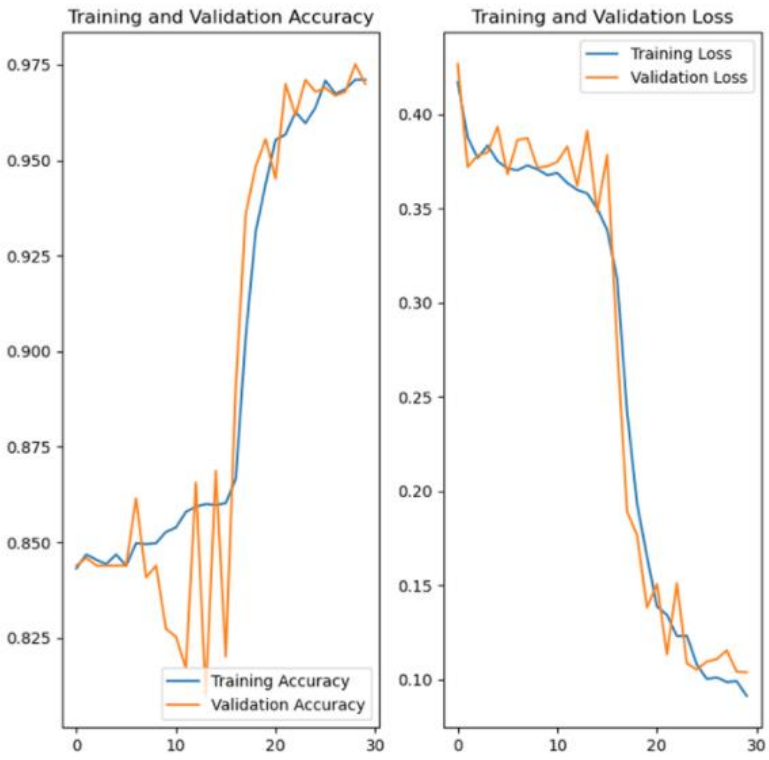
Table 5 - Human Testing Results



metrics gathered from test set:

	precision	recall	f1-score	support
0.0	0.97	0.95	0.96	88
1.0	0.99	0.99	0.99	476
accuracy			0.99	564
macro avg	0.98	0.97	0.98	564
weighted avg	0.99	0.99	0.99	564

Figure 3 - training and accuracy validation and losses, testing metrics without data augmentation.



metrics gathered from test set:

	precision	recall	f1-score	support
0.0	0.99	0.90	0.94	88
1.0	0.98	1.00	0.99	476
accuracy			0.98	564
macro avg	0.98	0.95	0.97	564
weighted avg	0.98	0.98	0.98	564

Figure 4 - training and accuracy validation and losses, testing metrics with data augmentation.

```
def findDefect(img, threshHold, pixThresh, lightBlur, errode, blur):
    exitCode = 0 # an exit code of 0 means a blob has been detected
    greyMaster = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # grey blob defect
    # Light blur to reduce size of small shadows and highlights
    lightBlur = cv2.blur(greyMaster, (lightBlur,lightBlur))
    # adjusted = setLims(lightBlur, 0 ,110) # massively increases time to run
    ret, adjusted = cv2.threshold(lightBlur, pixThresh,pixThresh,cv2.THRESH_TRUNC)
    kernel = np.ones((errode,errode),np.uint8) # forms the matrix used when eroding
    erosion = cv2.erode(adjusted,kernel,iterations = 1)
    # large blur to hide background weave and increase the size of defects
    greyBlur = cv2.blur(erosion, (blur,blur))
    #threshold on Gray image
    thresh = cv2.adaptiveThreshold(greyBlur,255,cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 101, 3)
    # Apply morphology open then close
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))
    blob = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (9,9))
    blob = cv2.morphologyEx(blob, cv2.MORPH_CLOSE, kernel)
    # invert blob
    blob = (255 - blob)
    # Get contours
    cnts = cv2.findContours(blob, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Code Snippet 3, first half of the “findDefcet()” function.

Consent Form (User Testing) (Appendix D)

Title of Project: _____

Name of Project Student: _____

Initial the box if you agree with the statement to the left

1 I confirm that I have read and understand the information sheet dated *[insert date]* explaining the above project and I have had the opportunity to ask questions about the project. ☐

2 I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason and without there being any negative consequences. In addition, should I not wish to answer any particular question or questions, I am free to decline. *Insert contact details here of project student.* ☐

3 I understand that my responses will be kept strictly confidential. I understand that my name will not be linked with the project materials, and I will not be identified or identifiable in the report or reports that result from the project. ☐

4 I agree for the data collected from me to be used in future research. ☐

5 I agree to take part in the above project and will inform the project student should my contact details change. ☐

_____ _____ _____

Name of participant Date Signature

(*or legal representative*)

_____ _____ _____

Name of person taking consent Date Signature

(*if different from project student*)

To be signed and dated in presence of the participant

_____ _____ _____

Project student Date Signature

To be signed and dated in presence of the participant

Project Information Sheet (Appendix E)

Project Title: Narrow Fabric Defect Detection using Computer Vision

You are being invited to take part in a student project. Before you decide, it is important for you to understand the aim of the project and what participation will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

Project Aim: Using computer vision to generate software capable of detecting defects in images of narrow fabric.

Why have I been chosen? Participant was select for no specific reason as I aim to enlist a variety of participants. Overall, there will be 10 participants.

Do I have to take part?
It is up to you to decide whether or not to take part. If you do decide to take part you will be given this information sheet to keep (and be asked to sign a consent form) and you can still withdraw at any time. You do not have to give a reason.

What will happen to me if I take part? The participant will be presented with a piece of software created during the project. 16 images will then be presented to the participant, for each image they will press one of two buttons indicating if they believe the fabric in the image shown contains a defect. Their results and the time it took them to complete the exteriorise will be recorded for use in the project.

Will my taking part in this project be kept confidential? All the participant’s names and personal data will remain confidential in the final report only the results gained from the task will be published.

What type of information will be sought from me and why is the collection of this information relevant for achieving the project's objectives? The results from the exercise will contain the number of images the participant identified correctly and the time it took them to decide for all 16 images. These results will be compared to the results of the software detection methods created in the project, comparing its accuracy and speed to conventional detection methods (human detection).

What will happen to the results of the project?
The results of this project will be published in a report to be submitted for assessment at the end of the undergraduate module COMP3931 Individual Project in the School of Computing at the University of Leeds.

Contact for further information: ed19ts3@leeds.ac.uk or +44 07788 262432

If you decide to participate in this project, you will be given a copy of this information sheet and a signed consent form to keep. Thank you very much for taking the time to read this information sheet.