Automatic Test Grader Using Image Processing And Machine Learning Techniques

Andries Petrus Smit
18183085

Department of Electrical and Electronic Engineering

University of Stellenbosch

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 the University of Stellenbosch

Study leader: JA du Preez

B.Eng Electrical and Electronic

December 2017

Acknowledgements

I would like to acknowledge my skripsie leader, JA du Preez, parents and the engineering class of 2017 for their kind contributing to this thesis.

Declaration

I, the undersigned, hereby declare that the work contained in this final year project is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

....

Signature Date

ECSA Exit Level Outcomes Reference

Outcome	**Reference**	
	Sections	Pages
1. Problem solving: Demonstrate competence to identify, assess, formulate and solve convergent and divergent engineering problems creatively and innovatively.	*All*	*All*
5. Engineering methods, skills and tools, including information technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	*2, 3, 4, 5, 6 & 7*	*10 – 62*
6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	*All*	*All*
9. Independent learning ability: Demonstrate competence to engage in independent learning through well developed learning skills.	*2, 3, 4 & 5*	*10 – 37*
10. Engineering professionalism: Demonstrate critical awareness of the need to act professionally and ethically and to exercise judgment and take responsibility within own limits of competence.	*8.3 & 8.4*	*66 – 67*

Is die ecsa outcome reg en EE sin?

For a more detailed review on how the respective outcomes were met in this thesis, please refer to Appendix D.

Abstract

English abstract here.

Uittreksel

Afrikaanse opsomming hier. Alle dokumente moet beide Afrikaans en Engelse opsommings bevat.

Contents

ECSA Outcomes Reference iii

Abstract iv

Uittreksel v

List of Figures x

List of Tables xi

Nomenclature xii

1 Introduction 1

1.1 Problem background . . . 1
1.2 Problem statement . . . 2
1.3 Project scope and assumptions . . . 2
1.4 Project objectives . . . 4
1.5 Research methodology . . . 4
1.6 Graphical overview of system . . . 5
1.6.1 System overview . . . 5
1.6.2 Graphical representation . . . 5

2 Literature Study 7

2.1 Existing systems . . . 7
2.1.1 Standard OMR techniques . . . 7
2.2 Additional techniques . . . 10
2.2.1 Contour detection . . . 10
2.2.2 Character recognition . . . 10

2.3 Conclusion: System requirements 12

3 Image Processing **13**

3.1 Orientation Detection 13

3.1.1 Initial filtering and orientation detection 14

3.1.2 Radon Transform 17

3.1.3 Finding the template 18

3.2 Bubble detection and processing 21

3.3 Data processing and grading 21

3.4 Conclusion 22

4 Machine learning approach **23**

4.1 Character recognition using a neural network 23

4.1.1 Summary 23

4.1.2 Preprocessing to find individual digit image 24

4.1.3 Neural Network Classification 27

4.1.3.1 Introduction 27

4.1.3.2 The artificial neuron 28

4.1.3.3 Generating output from an neural network 29

4.1.3.4 Training of neural network 30

4.2 Probabilistic Graphical Models 31

4.2.1 Overview 31

4.2.2 Intended digit model 31

4.2.2.1 Structure of the graph 31

4.2.2.2 Estimating the intended digit 33

4.2.3 Intended answer model 33

4.2.3.1 Structure of the graph 33

4.2.3.2 Estimating the intended answer 34

4.2.4 Student number model 35

4.2.4.1 Structure of the graph 35

4.2.4.2 Estimating the student number 35

4.2.5 Training the model 36

4.3 Conclusion 36

5 Analysis of results **37**
5.1 Future improvements 37

6 Summary and conclusions **38**
6.1 Project summary 38
6.2 How this final year project benefits society 38
6.3 What the student learned 38
6.4 Conclusion: Summary and conclusions 38

References **39**

A Overview of system **40**
A.1 System as a whole 40
A.2 Deriving the intended student entry 41
A.2.1 Student answer 41
A.2.2 Student number 42
A.3 Deriving the estimated digit 44

B Implementation/Algorithms **46**
B.1 Deep Convolutional Neural Network 46

C Validation and results **47**

D Outcome compliance **48**

E Project plan **50**

List of Figures

1.1 Automatic grading template 1. 3
1.2 Graphical overview of the system as a whole. 6

2.1 Standard OMR template, from VijayaForm (2017) 9
2.2 Corrected answer by crossing out bubbles. 11

3.1 Four markers found from applying Radon transforms. 15
3.2 Reduced contours in image. 16
3.3 Radon transform applied on a 2 dimensional area, from Edoras (2017) . 17
3.4 Result in rotation after applying radon transform. 18
3.5 Detection of template in image and estimation of bubble locations. . . . 20

4.1 Image showing found contours for boxes used for character recognition. 24
4.2 Example image found by doing image processing in main test sheet. . . 25
4.3 The box contour found is normalized to form a rectangular shape. . . . 25
4.4 Box after black lines gets filtered out, found using a Radon transform. . 26
4.5 Custom segmentation algorithm used to find the main cluster in the remaining image. 26
4.6 Area block drawn around segment most probable to belong to the digit. 27
4.7 Image after final translation and normalization is applied. 27
4.8 Basic structure of a neural network, from Karpathy (2017) 28
4.9 Example image used as input to the neural network, from Tensorflow (2017) 29
4.10 Graphical setup for determining the intended digit written by a student. 32
4.11 Column with the evidence that gets considered for the calculation of an intended digit. 32

4.12 Graphical setup of determining student answer. 34
4.13 Graphical setup of determining a student answer. 35

A.1 System overview. 40
A.2 Graphical setup of determining student answer. 42
A.3 Graphical setup of determining student number. 44
A.4 Graphical setup of determining intended digit. 45

D.1 Project plan for the final year project. 49

E.1 Project plan for the final year project. 51

List of Tables

Nomenclature

ANN	Artificial neural network
$DCNN$	Deep convolutional neural network
OCR	Optical character recognition
OMR	Optical mark recognition
PGM	Probabilistic graphical model

Chapter 1

Introduction

As modern technology and machine learning techniques advances, it is important for the educational sector to also continuously advance their learning environment. This allows for an ever improving learning experience in and outside the classroom.

1.1 Problem background

In the recent years the Applied Mathematics Department of Stellenbosch Engineering, started observing a decrease of accuracy in grading of tutorial tests, done by a teaching assistants or demi. Students complain on a regular basis about correct answers being marked wrong or even that their answers were totally ignored. To address this problem the Applied Mathematics department proposes to automate the process of grading these tutorial tests.

The head of the department wants a system that can analyse and grade tests written on a specific template. These answer sheets are handed out for the students to fill in their respective answers. The answer sheets are then scanned-in to create a digital copy. The system is tasked with automatically grading all these digital copies and transferring the graded results to a database.

The department will send weekly scanned in sheets, which needs to be graded and the results then sent back to them. This is done in parallel with the development and expansion of the test grading software. For these reasons an agile development methodology is used with a weekly validation test, as the system gets used.

1.2 Problem statement

Given the problem background and stakeholders discussed in the previous section the problem to be solved can be formulated as

Develop and **implement** a *automatic test grading system* that will *reduce marking time* and *increasing accuracy* on the *grading* of Applied Mathematics tutorial tests, written by students.

1.3 Project scope and assumptions

Initial discussions with the department head revealed that a specific answer sheet template can be used. This allows the custom image processing software to more accurately determine what the student filled in. The template will consist of bubbles that can be coloured in as well as blocks for handwritten digits, as can be seen in Figure 1.1. This template is constructed by the Applied Mathematics Department. Thus the focus of this project is on processing that scanned in document written on the specific template. To use the template the student must fill in his/her student number and answers in the blocks above each category. They are also required to fill in the bubble underneath each digit, corresponding to that specific digit. Additionally a bubble next to each question is provided if a negative sign is required.

Any additional assumptions are specified at the appropriate times throughout this project. Two more templates will also be implemented in later chapters, which will allow for multiple choice questions.

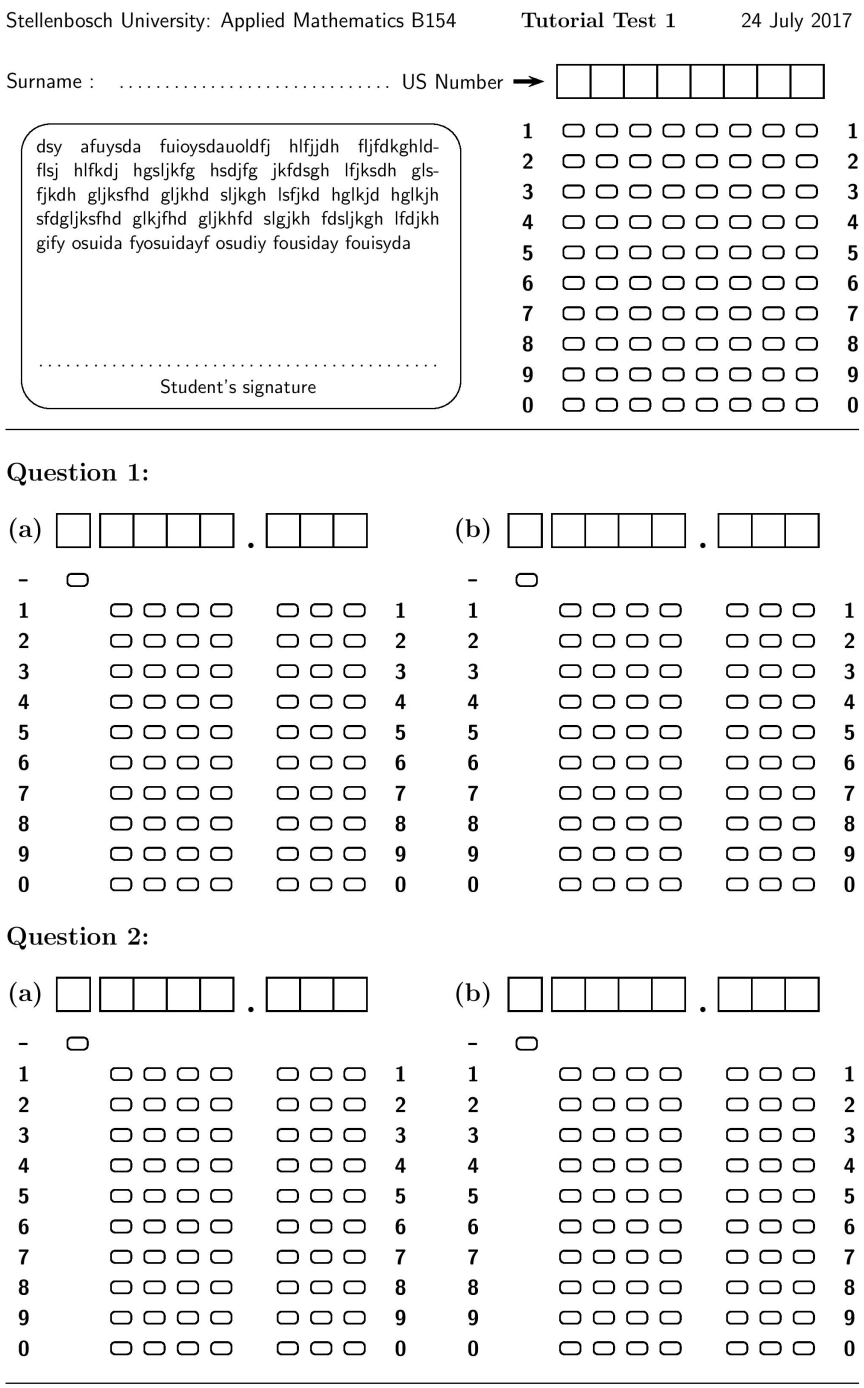

Figure 1.1: Automatic grading template 1.

1.4 Project objectives

The problem as stated in section 1.2, is addressed by pursuing the following objectives:

1. Do a *literature study* on the topics of *Image Processing*,*Computer Vision* and *Character Recognition*.

2. Develop a *software application* to enable a *user* to grade a large number (approximately 1000) scanned in student tests automatically.

3. Do a weekly *validation experiment* with the software, by grading tutorial test for the department. The results are then used as the student's grade for that test.

4. Use an *agile development methodology* to improving the software in parallel with the grading of weekly tests.

5. Add additional software to allow grading of new *templates* and to increase the speed and accuracy of the software.

The objectives are covered in different chapters in the report. Note that each objective (1–5) build on the objective(s) previously listed.

The software must also ease the use of the template for students while also providing precise and useful feedback. To do this every graded result will also include feedback on what answers the student got wrong and what the correct answers were. To increase ease of use, software is developed that provides the luxury to students to only writing down their student numbers. Thus time does not get wasted filling in the student number bubbles. This result is achieved in Chapter 4.2.4.

1.5 Research methodology

To complete the objectives listed in Section 1.4, a agile development methodology is used. The methodology followed comprises of five different phases:

1. Identify a new feature or update that needs to be implemented into the software package.

2. Do a study on the existing methods to implement this new software.

3. Implement and integrate the new software with the current knowledge of the solution.

4. Test the software and observe if it is working as planned.

5. If the software is not working as planned, revisit steps 2-4 until the new software is working.

6. Use version control, for example Git, to save the latest version of the software.

The structure and graphical overview of the software, is presented next.
(Vra prof of my wiskude reg is oor hoe die PGM stelsel werk.)

1.6 Graphical overview of system

1.6.1 System overview

When thinking about the system (for template 1), from a philosophical sense, it can fundamentally be represented with 6 information nodes. The student has certain information he/she wants to portray on the paper. This includes the 4 answers and student number he/she wants to write down. Thus those 5 nodes gives rise to the image, representing the last node. This has to be represented in a probabilistic way. This is, because the process of writing answers down and scanning the test sheet in is going to produce a different image every time a test is written, even though the same answers are intended. This graphical model is illustrated in Figure A.1. The unnamed blocks indicate that information processing happens in these steps. For the complete visual overview of the system, again refer to Appendix A.

For a more detailed mathematical derivation of the entire system, refer to Appendix A.

1.6.2 Graphical representation

For this project a graphical approach is followed in representing the software developed. A graphical model in essence allows software to be represented as information (nodes or circles) and relationships (directed arrows). The directions of the arrows represents what information causes other information to be created, thus given a parent to offspring interpretation). An example of such a graph can again be seen in Figure A.1.

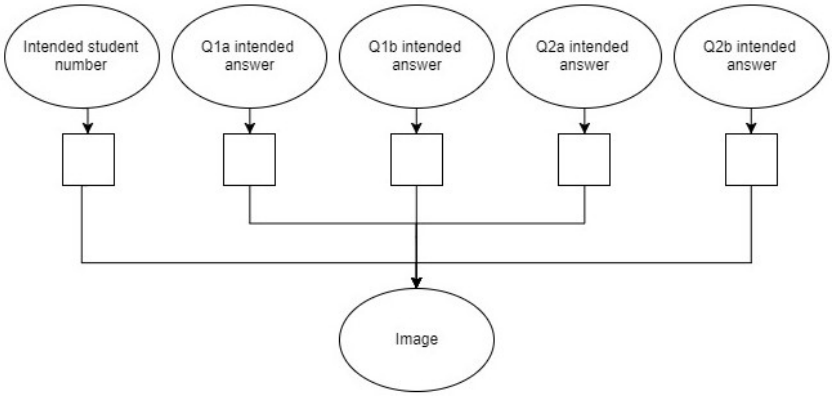

Figure 1.2: Graphical overview of the system as a whole.

This allows intuitive reasoning about how the system should operate. An observation is always made in the form of an image and then the system is tasked with predicting what the written answers are. Due to the probabilistic nature of the system a probabilistic component is also needed in this graphical models. Thus a probabilistic graphical model (PGM) is used.

A PGM is in essence the same as a general graphical model. The only difference is that now the information and relationships between information bubbles are probabilistic and not always certain as with general logic graphical models.

A literature study is presented next on methods currently used by existing automatic test grading systems. This will describe in the next chapter.

Chapter 2

Literature Study

In the previous chapter the problem statement and objectives of this project was laid out. Further a brief system overview was also provided. This chapter will provide information about already existing systems and the techniques they use in doing image processing and character recognition on images. Additionally, research on existing libraries, to aid in this project, will also be done.

2.1 Existing systems

A Optical Mark Recognition(OMR) system is a piece of software that is used to extracted hand written information from a filled in form. Each system normally has a specific template that it can extract information from. An template for one of these forms can be seen in Figure 2.1. These systems are generally used when fast and accurate grading of tests are needed. The biggest drawback of these systems are that information can only be portrayed in a very limited manner. On OMR templates there are a bubbles that allows a user to chose between different options to answer. OCR systems are thus excellent for the grading multiple choice type questions.At the moment most of these systems only grade colour-in bubbles and do not interpret characters on the page.

2.1.1 Standard OMR techniques

As can be seen in Figure 2.1, there is normally specific reference blocks on a OMR template. These blocks are included to allow the computer vision and image processing

algorithms find the orientation of the image more easily.

In an OMR system there are normally two phases to marking an test, as stated in Ivetic & Dragan (2003). The first step is to determine the region within where the answers are located in the image. In this process the system finds the orientation of the template in the image and thus can approximate the location of the bubbles. Normally some preprocessing on a blank template is done beforehand too aid in locating the bubbles. Once the bubbles are found their locations gets stored and processed. The final step is then to calculate the average pixel concentration in the bubbles and estimate whether they have been filled in.

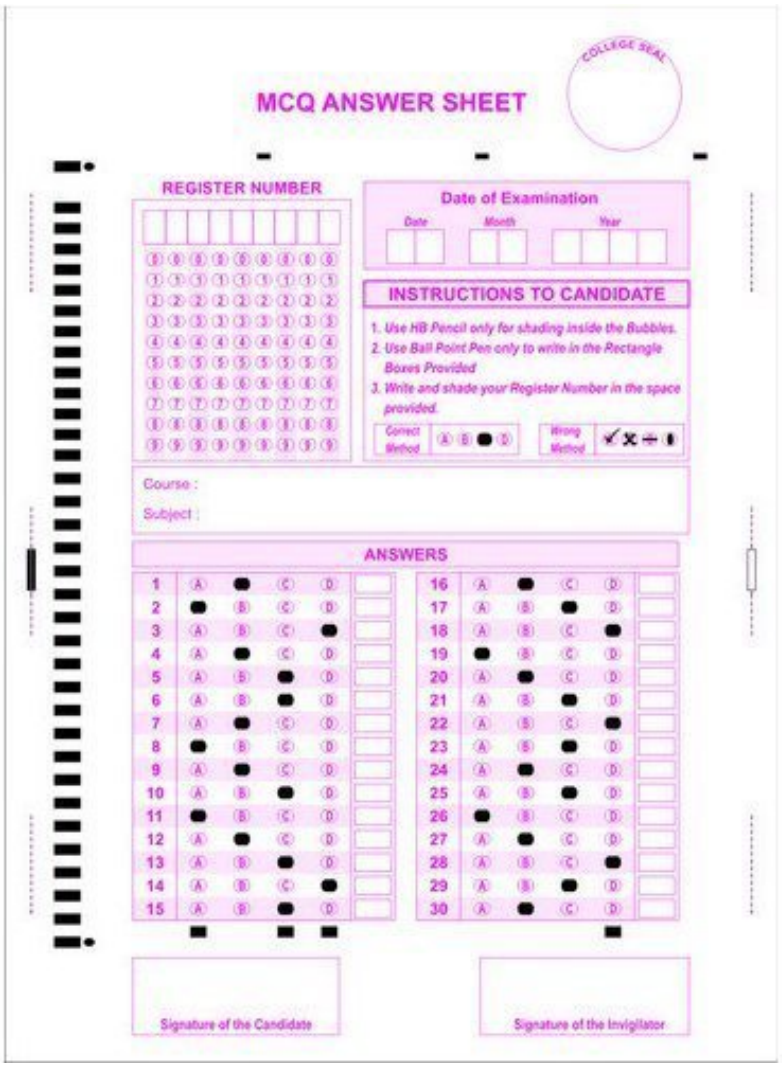

Figure 2.1: Standard OMR template, from VijayaForm (2017)

2.2 Additional techniques

The above method allows for grading of tests using a simple OMR system. For the test grader used for the Applied Mathematics tests, there is a need to go beyond such a basic system. In standard OCR, when a student makes a mistake, he/she needs to erase the existing answer and write a new answer in. This can be time consuming for a template with 8 bubble row choices per answer. This also increases the probability that the student will make a mistake in the process. Addressing this problem it is determined that two additional options need to be made available. Firstly the student is allowed to cross out answers instead of totally erasing them. An example of this can be seen in Figure 2.2. Then secondly the system needs to be accurate enough to determine the student's student number only by filling in the character blocks above the bubbles. This requirement is addressed by using a PGM in Section 4.2.4. Using optical character recognition (OCR), the bubble information and character information can thus be cross-referenced in an intelligence way, as seen in Section 4.2.

2.2.1 Contour detection

To determine if an answer in a bubble is truly coloured in and not just crossed out, contour detection is needed. This means that the contour around the bubble needs to be detected and used in analysis. In python this can be implemented using the freely available OpenCV library, as referenced in Rosebrock (2016). OpenCV is an image processing library has has highly optimized techniques to find contours in a given image. Given this new contour information the bubbles is can now be assessed by the pixels inside it, as well as its shape. (Verander die image dalk na een wat 'n contour om het)

2.2.2 Character recognition

To further increase the accuracy of the system, Optical Character Recognition (OCR) software will also be need and applied on the characters blocks, present on the templates. Research shows that one preferred way of doing OCR is using TensorFlow. This method is described in Google (2017). TensorFlow is also a python library, but allows the build of instructions to be implemented in efficient c++ code. For this test grader,

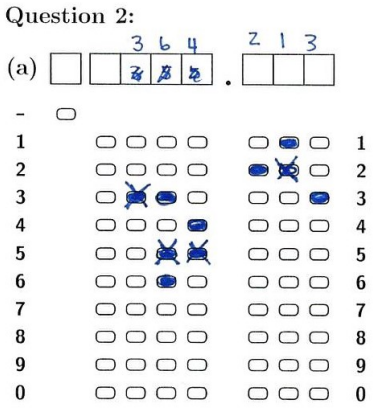

Figure 2.2: Corrected answer by crossing out bubbles.

TensorFlow is used to construct a convolutional neural network. This network will read an image, containing the digit, and predict the probability of each digit.

2.3 Conclusion: System requirements

In conclusion it can be seen that the system will need to incorporate a combination of image processing(OMR) on the bubbles and character recognition on the digits, handwritten by the students. When these different evidience is the considered in combination, a more accurate student answer can be estimated. This will also allow for more convenience for students writing these test, hoekom?. Kyk na oom Johan se raad. Die litriture study moet oor huidige metodes gaan en hul resultate.

Chapter 3

Image Processing

The previous chapter focused on existing methods of grading tests automatically. It was found that most systems only use image processing, without a machine learning component, to grade these test. In this chapter the core techniques behind processing these answer sheets, using image processing, is described. By using only these image processing techniques a reasonably accurate system can already be constructed.

For further improvements in accuracy we will investigate and implement two machine learning approaches. These approaches are discussed in Chapter 4.

3.1 Orientation Detection

As mentioned in Section 2.1.1 there are two main steps in OMR grading. The first challenge with grading a scanned in answer sheet, is finding the orientation of the template in the image. To do this there 2 or more references points must be found on the page. These reference points then allows for the calculation of the template's rotation, offset and size inside the image. In Chapter 2 it was determined that the traditional way to to find this reference points was to include them on the page, in forms of black blocks. This is an effective method of finding the template, but might look a bit less attractive, due to the black blocks on the page.

For this project the markers,reference points, that the software uses are already present on the template paper. These are the two longest horizontal lines as well as the two vertical lines on the comment box, as can be seen in Figure 3.1. Together these lines have enough information to determine 4 reference points, shown in red in

the figure. The reason these lines are chosen as references, are due to the fact that a Radon transform can easily be applied to find them, as seen in Section 3.1.2. Thus the software can successfully find the template even though one of the 4 lines are identified incorrectly. Using the reference points the rotation, offset and size of the template is calculated. But before the orientation of the image can be determined it is a good idea to quickly check if the image might be upside down. This is done to make it easier to find the orientation afterwards. To do this some initial image filtering is firstly required and is discussed in the next section.

3.1.1 Initial filtering and orientation detection

To check if and image is upside down the software first needs to find relevant contours on the page. The contours are then filtered out if it does not loosely match the characteristics of a bubble or character block. This is done in 5 steps:

1. Threshold the image by making all the pixel values either white (lower that the mean) or black (higher that the mean)
2. Do contour analysis on the image to find all the contours, using the python library OpenCV.
3. Filter through the contour array to filter out all contours that are not approximately the size and aspect ratios desired.
4. Save these contours for later use.
5. Determine if more contours lie above the middle of the image. This is true if the image is the right way around. Rotate the image by 180° otherwise.

It is important to note that there are still unwanted contours in the list, but for now this reduced list is sufficient. Once the list is found the software counts the number of contour centrepoints below and above the image center. Figure 3.2 shows the resulting contours found in the image. As can be seen in the figure more bubbles should be below the horizontal center line, for the image to be the right side up. The next step will now be to determine the coordinates of the answers the student wrote down. To do this the template is first located in the image. This process is described in Section 3.1.2.

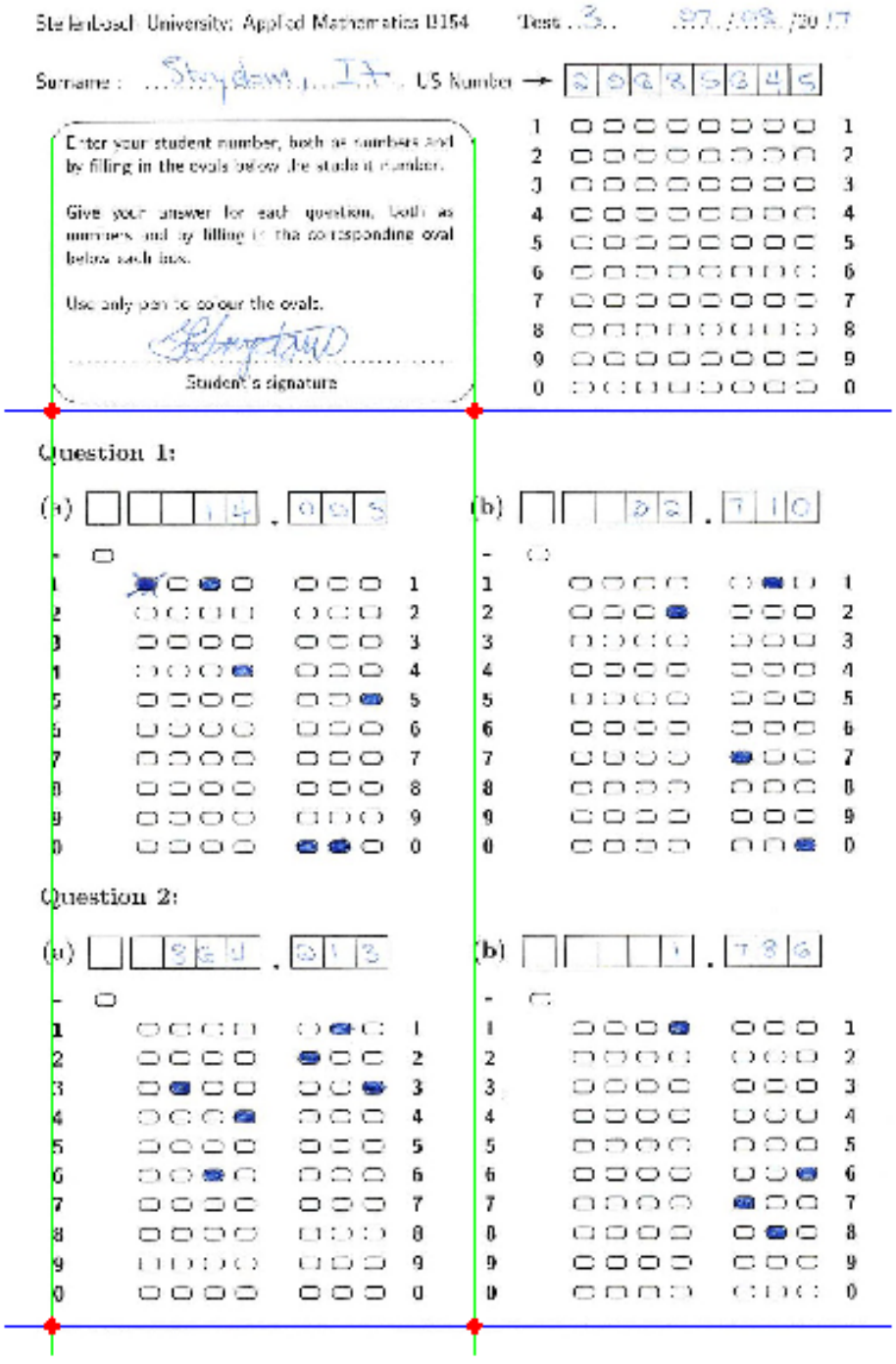

Figure 3.1: Four markers found from applying Radon transforms.

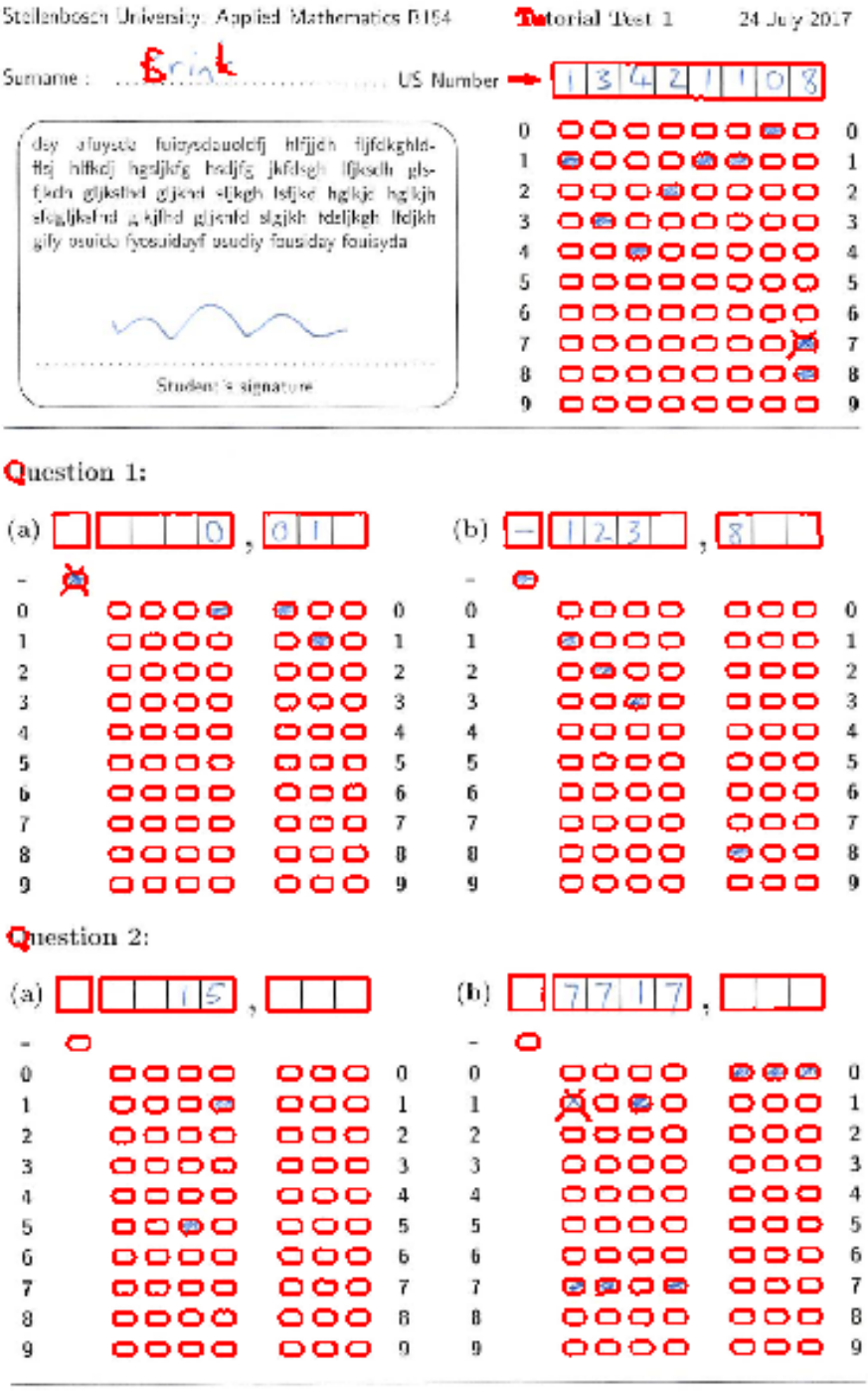

Figure 3.2: Reduced contours in image.

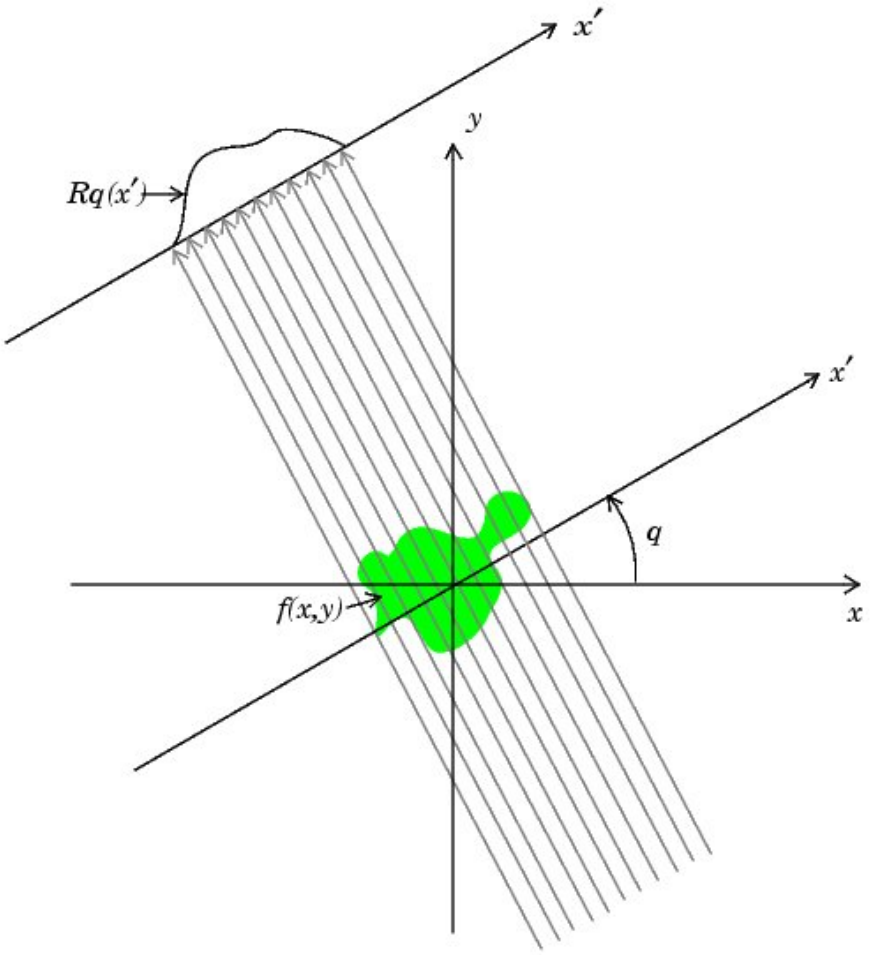

Figure 3.3: Radon transform applied on a 2 dimensional area, from Edoras (2017)

3.1.2 Radon Transform

(Maak 'n symbols page) The Radon transform is an integral transform that can be represented by a series of line integrals over a function $f(x, y)$. These transforms are commonly used in CT scans. Mathematically this transform is defined as

$$G(r, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x\cos(\theta) + y\sin(\theta) - r)dxdy.$$

Where r represents the perpendicular offset of the line and θ the angle of the line. x and y represents a 2D space within which the function $f(x, y)$ is defined. A visual interpretation of this transform can be seen in Figure 3.3.

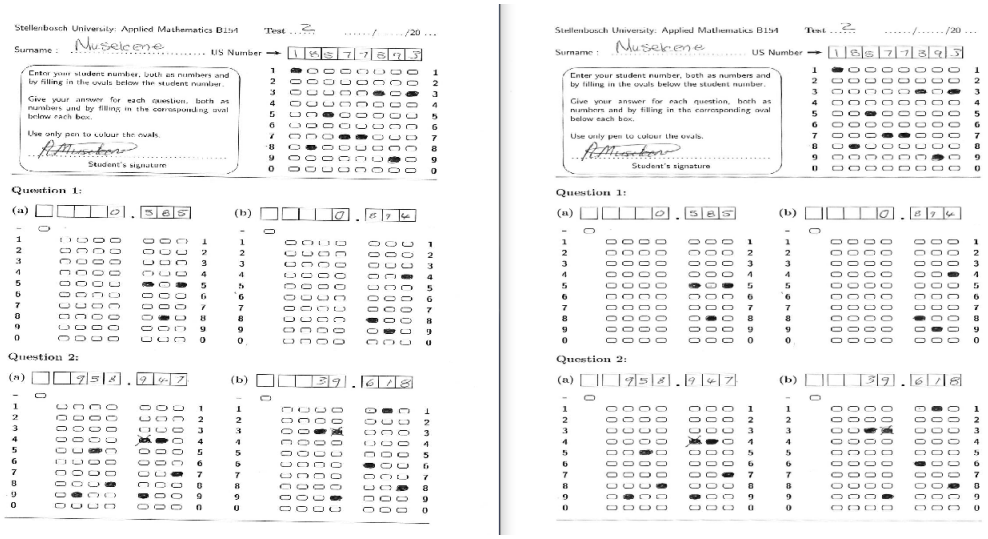

Figure 3.4: Result in rotation after applying radon transform.

3.1.3 Finding the template

In this project a Radon transform is calculated at specific intervals of the gradient. Each gradient interval will thus generate a 1 dimensional array of values corresponding with the pixel intensities along the lines that are being summed, as seen in Figure 3.3. This method is used determine where the 2 horizontal and 2 vertical lines are located, as mentioned in Section 3.1.

To find the first two horizontal lines the Radon transform is applied at a angle of 90° and then incremented by a small amounts for the next transform. Black lines will cause a spike to appear on the Radon transform when it is summing parallel with that black line. Thus by finding the transform angle value that present this maximum value. The rotation of the template can be found. The two maximum values that is recorded at this angle is can then be interpreted as the locations of the two horizontal lines. After the correct angle is found the two vertical lines can be found by applying a Radon transform at a angle 90° clockwise from the previously calculated angle. Those two values then provides the relative locations of the two vertical lines. Now the four reference point shown in Figure 3.1 can be calculated. Once the reference points are found the image is rescaled and orientated to the original template size and orientation for further processing. In Figure 3.4 the corrected image is shown, before it gets processed.

Once the template is found the bubble values and digit blocks can be determined,

using reference location. These reference locations were calculated in preprocessing done on an empty template. Figure 3.5 illustrates the final estimation of all the bubbles in the template. The estimated bubble centres are coloured red, while the green points represent the centres of all the remaining contours.

Now that all a list of possible bubble contours are found, the next challenge is to find one contour for each bubble. This is discussed in the next section.

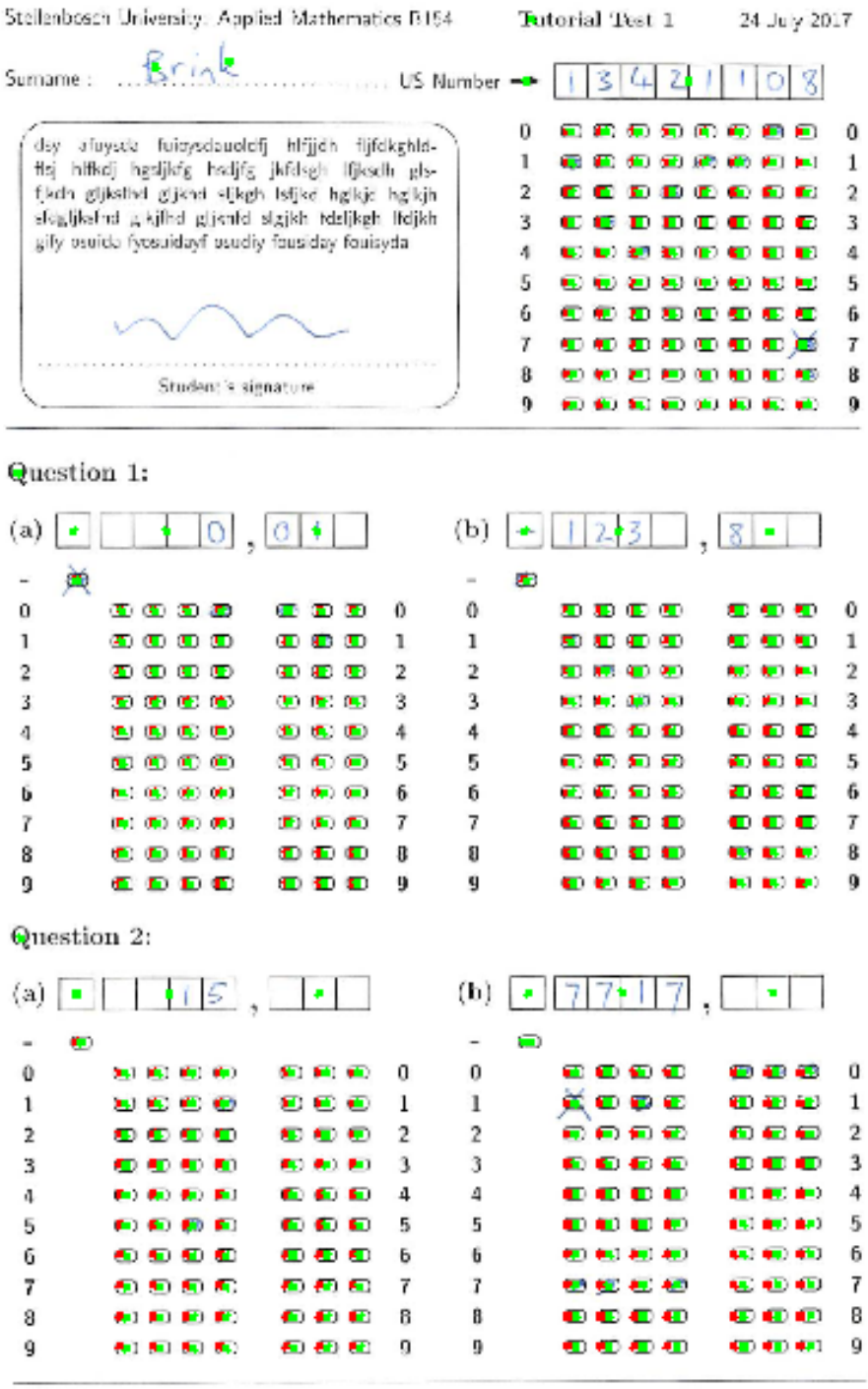

Figure 3.5: Detection of template in image and estimation of bubble locations.

3.2 Bubble detection and processing

To find the location of each bubble in the image, the system simply takes the contour closest to the estimated bubble location. This can be done in an efficient manner by sorting the contours by their locations. Searching through the contours now becomes linear and of complexity O(n), where n is the number of bubbles. This means that the processing time to find these bubbles is linearly related to the number of bubbles in the template image.

Next the data in each contour needs to be processed and stored. The first type of evidence is calculating the average pixel intensity inside the contours. If this value is high the bubble is most likely coloured in or crossed out. The advantage of using the closest contour as the bubble's estimated location, over conventional methods now becomes apparent. In conventional methods an estimated area is calculated where the bubble is most likely to be located. This becomes problematic if the system needs to know if the bubble has been crossed out, as only data about pixel intensities are present.

Using contour analysis, information about the shape of the bubble entry is also provided. Thus by drawing the smallest possible block around the contour, that still covers every value inside the contour, an area can be calculated. This area becomes large when a answer is crossed out, due to the lines stretching outside the initial bubble. By inspecting the area value, the system can successfully determine if the bubble is filled in and crossed out.

3.3 Data processing and grading

The previous section now allows each bubble to be classified into 3 categories namely, empty, completely filled in and crossed out. An additional category of partially filled in will also be introduced, as it aids grading of tests where students write lightly. An algorithm to determine what bubble was chosen can now be described as follows:

1. Count the number of completely filled in answers in each column. Store the position of that entry for later use.

2. If there are no completely filled in answers, count the amount of partially filled in answers and override the previous values.

3. If the previous result is 0, set the output value for that column to 0.

4. If step 2 or 3 presents a value greater that 1, save the answer sheet to a clashlist to be evaluated manually once the automatic grading of the test are completed.

3.4 Conclusion

This chapter provided an overview of a basic automatic test grading system using image processing and computer vision. The system can achieve acceptable results using only these techniques.

The following chapter will focus on applying additional machine learning techniques to further improve the accuracy of grading these test. Two new test templates will also be introduced. (maak seker jy het daaroor gepraat).

Chapter 4

Machine learning approach

The previous chapter briefly described the basic workings of the optical mark recognition (OCR) system inside the automatic test grader. There is still one critical piece of evidence that has not been used with the basic system. This information is the characters that the student writes in the designated boxes.

This next chapter will provide two machine learning approaches to significantly improve the accuracy in identifying digits over the previous standalone basic system, discussed in Chapter 3. An approach to locate and classify hand written characters, provided by the students, using a deep convolutional neural network (DCNN), is firstly described. Next a more accurate method in determining the true digit, student answers and student number, using probabilistic graphical models (PGM), is discussed. This PGM method is found to be accurate enough to determine the student number by only using the character recognition information provided.

4.1 Character recognition using a neural network

4.1.1 Summary

In Chapter 3 only the bubble evidence is considered when estimating what digits the student wrote down. In this section another piece of information is going to be processed to further increase the accuracy of this system. This information is the characters that the students write down in the designated boxes, as shown in Figure 4.1. To process these digits a machine learning approach, called a neural network, is implemented. Once the test sheet is processed all the character digits can be found. A 28 by 28 digit

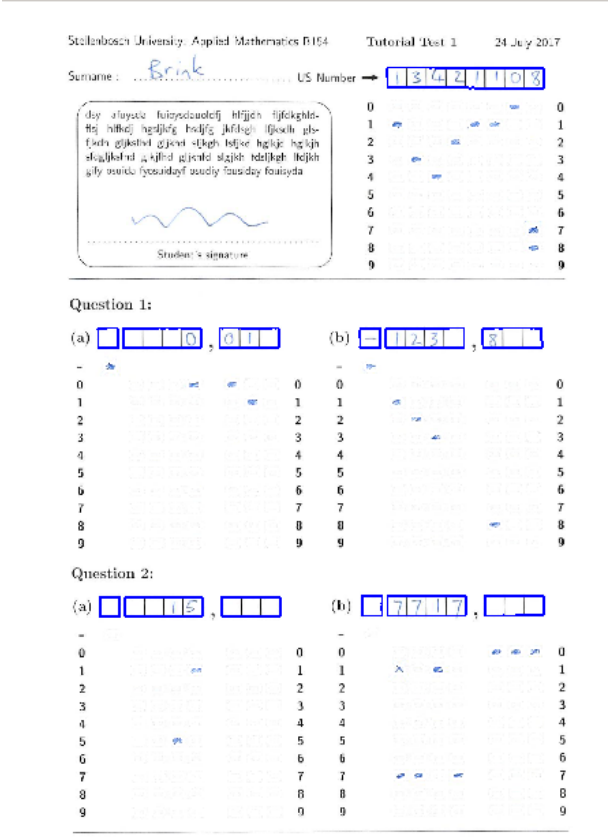

Figure 4.1: Image showing found contours for boxes used for character recognition.

image is then created for each character found on the page. An example of this is shown in Figure 4.2.

A neural network then processes these digit images and produces an estimate to what digits are most likely to be present in each image.

4.1.2 Preprocessing to find individual digit image

Before each digit can be classified using a neural network, the individual 28 by 28 pixel digits must first be found inside the test sheet. To do this image processing is required. This is done in 6 steps as seen below:

1. Find the contour closest to the expected location of the block, calculated in Section 3.1.3. This is ilustrated in Figure 4.1. The bubbles was already extracted out of the image where they were processed.

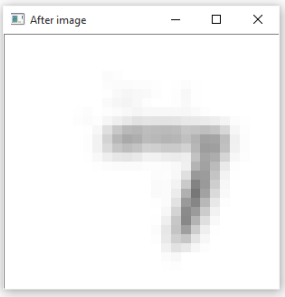

Figure 4.2: Example image found by doing image processing in main test sheet.

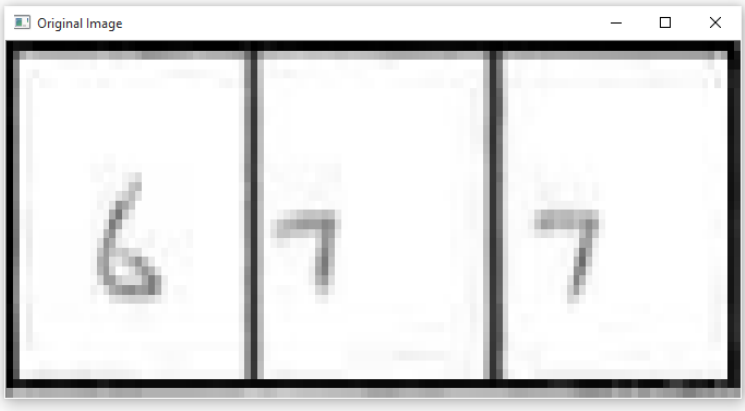

Figure 4.3: The box contour found is normalized to form a rectangular shape.

2. Transform the image to become fully rectangular using OpenCv's $four_point_transform$ method. This method applies a four point perspective transform on the image to reshape it into a rectangular form. An example of the final product can be seen in Figure 4.3.

3. Do a horizontal and vertical Radon transform to find and remove the dark box lines on the image, as seen in Figure 4.4. The mathematics behind a Radon transform is described in Section 3.1.2.

4. Use the values received from the Radon transform to segment the image into the different boxes.

5. Using a custom segmentation algorithm to find the pixels most likely to belong to the digit. This algorithm is developed and implemented using breath first search technique to cluster the image into different segments. The algorithm works by

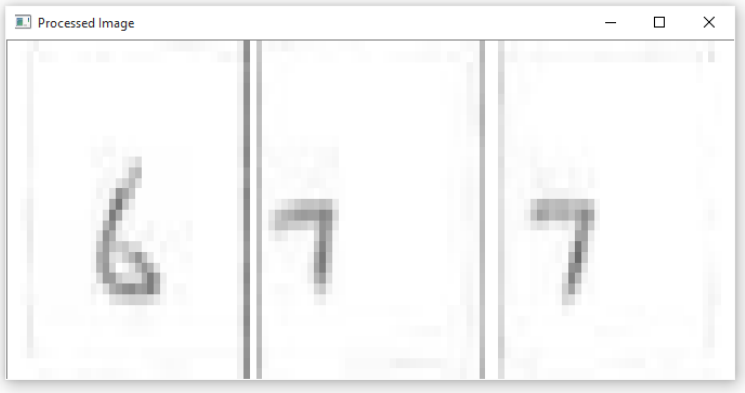

Figure 4.4: Box after black lines gets filtered out, found using a Radon transform.

Figure 4.5: Custom segmentation algorithm used to find the main cluster in the remaining image.

first searching the image for pixels higher that a threshold value. This value specifies if an pixel is background or belongs to an image segment. Then all the pixels higher that the threshold value gets assigned to a segment. A segment is thus classified as a local region that does not connect to any other segment through pixels higher that a threshold. The segment that most likely represents the digit is then extracted, as seen in Figure 4.5.

6. The area that the segment occupies then gets calculated, as seen in Figure 4.6

7. Next the image is centred and normalized using the image area as reference. This is seen in Figure 4.7

8. The image is then reshaped into a 28 by 28 greyscalled image to be processed by the neural network.

Figure 4.6: Area block drawn around segment most probable to belong to the digit.

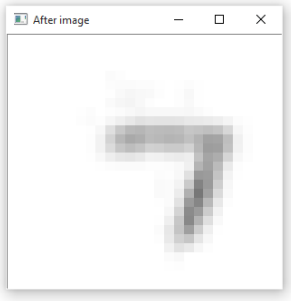

Figure 4.7: Image after final translation and normalization is applied.

Each digit on the test sheet is now separated and can be used as input to an neural network. An overview of this neural network will be given next.

4.1.3 Neural Network Classification

4.1.3.1 Introduction

A neural network is a powerful machine learning tool for approximating complex functions. The basic architecture of a neural network can be seen in Figure 4.8. A artificial neural network (ANN) consist of an input, hidden and output layer, as described in Nielsen (2015). A neural network is a simplified approximation of how neurons in the brain works. Each neuron in the network acts as a small processing unit. With a initial input, an output is determined by measuring the last output node, as seen in the Figure. The output can only be determined once each layer have been calculated one after the other.

For this project, a neural network is trained to estimate the probability of which of the 10 digits, from 0 to 9, are most likely present in the input image. The neural

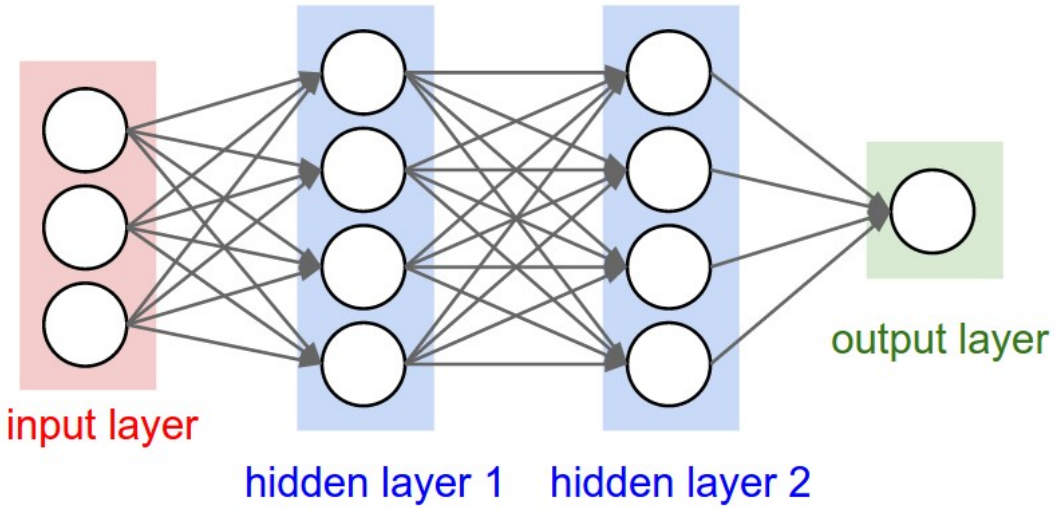

Figure 4.8: Basic structure of a neural network, from Karpathy (2017)

network will take a 2 dimensional array, representing a greyscaled image, as input to the network. Figure 4.9 illustrates an example input of a neural network using and 14 by 14 example image.

For this project a 28 by 28 greyscalled image is used as input. Thus if each pixel represents one value, ranging from 0.0 to 1.0, there is total of 748 input values. The basic workings of a neural network is described in the next section.

4.1.3.2 The artificial neuron

The neuron unit is the small processing unit used to build a neural network. The first step in calculating the output of a neural network is to find the weighted sum of the inputs. ALl the variables in this network are floating point values. A bias variable is added to allow a neuron to have an offset in its output. The weighted sum of these inputs are added with a bias variable to give,

$$z = \sum_{i=0}^{c} x_i * w_i + b. \tag{4.1}$$

Where c is the number of inputs. x_i and w_i respectively are the input and weight values at index i. b is the biased that enables and offset in the output(z) of the neuron.

The summed value then gets normalize using an sigmoid function, seen in Equation 4.2.

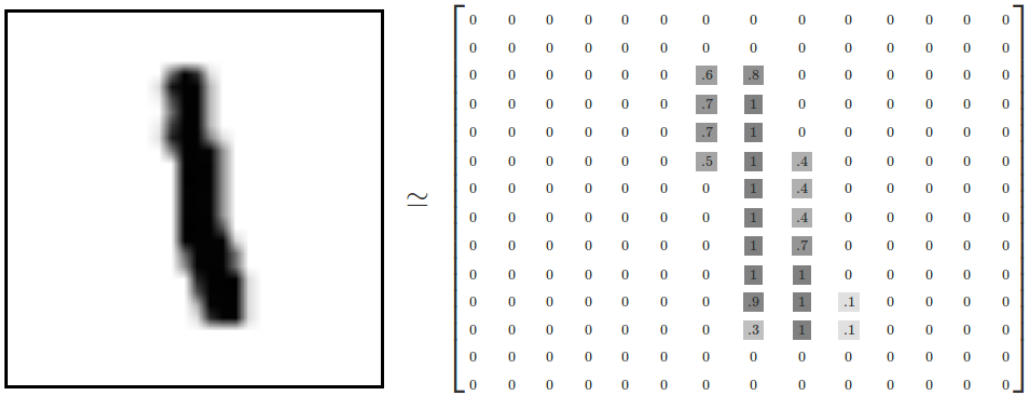

Figure 4.9: Example image used as input to the neural network, from Tensorflow (2017)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

This artificial neuron thus basically takes in a weighted input with a bais and produces a normalized output $sigma(z)$. By adjusting the weights and bais variable, each neuron can learn to exhibit certain characteristics. This process thus allows different functions to be approximated by changing these weights. If a network of these neurons is used together, as shown in Figure 4.8, complex functions can be trained onto the network. IN the case of this project these weights and biases needs to be set to specific values, to allow the network to best categorize digits from an image. To do this these weights and biases will be trained form training data, as seen in Section 4.1.3.4.

4.1.3.3 Generating output from an neural network

After the 748 input values have been assigned to the network inputs, each of the network's layers can be calculated consecutively one after another. The first coulomb in the hidden layer will thus use the 748 input values and produce a normalize output for each of the neurons in that coulomb. This is done using using the previously mentioned equations 4.1 and 4.2. Once the first layers's outputs are calculated the next layer can be calculated. This process is repeated until all the layers are calculated in the hidden section. The output layer is then calculated using the same method. For the character recognition in the test grader, 10 output neurons are used, corresponding

to the probability of the 10 digits being present. The values observed on the output neurons, gets normalized and used as the probability of each digit being present, using

$$p(i) = \frac{\sigma(z_i)}{\sum_{k=0}^{10} \sigma(z_k)}. \tag{4.3}$$

Where $p(i)$ is the probability of digit i being the character in the image. The value $\sigma(z_i)$ is the output of the output neuron at index i.

For a more detailed explanation on the DCNN used in this project, refer to Appendix B.1. (Maak dit dalk dat die neural network in overview hier beskryf word. Soos hoe dit werk as a black box. En dan Verder word die wiskunde agter NN and DCNN agter in die appedix beskryf)

Bv: For a mathematical description on neural networks and DCNN, refere to Appendix B.1.

4.1.3.4 Training of neural network

To train a neural network the MNIST dataset is used. This is a database that has a labelled training set of 60,000 images, and a labelled test set of 10,000 images. Each image has an accompanied label that specifies what digit is in the picture. The neural network is trained by using this training set, to adjust its internal weights and bias variables to more closely represent the labelled training data. The basic idea behind the training method used in a neural network is described in the following steps. For a more detailed description refer to Appendix B.

1. Calculate the network's output for each of the training images used in this training round.

2. Get the error margin of the network, using a formula that compares the true label of the training image, with the estimated label of the network.

3. Calculate the value with which each weight should be changed to reduce the error margin. One method of doing this is using the gradient decent with back propagation algorithms.

4. Repeat steps 1-3 until a time or accuracy criteria is met.

classified by the neural network. The new character recognition evidence can now be used in combination with the bubble evidence to make a significantly more accurate estimate for the intended answers from the student. The next step is thus to combine all these individual evidence, in a intelligence way, to make a accurate prediction. A method to achieve this result is described the next section.

4.2 Probabilistic Graphical Models

The final step in determining the students answers is to probabilistically determining the most likely answers, given the bubble and character evidence presented. To do this 3 probabilistic graphical models (PGM) are implemented.

4.2.1 Overview

A probabilistic graphical model (PGM) is a probabilistic graph containing random variables, where the graph expresses the conditional independence structure between these variables. The type of PGM used for this project is a Bayesian network. A Bayesian network models a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). As seen in Figure 4.10, arrows are used to indicate which variables are conditionally dependent on others.

4.2.2 Intended digit model

4.2.2.1 Structure of the graph

The figure should be interpreted in the direction which information flows. Originally a student has a certain digit that he/she wants to portray on the page. This is given by the 'Digit user intended bubble'. There are 10 possible digits to consider and thus the bubble has 10 possible states. The intended digit gives rise to the intended bubbles and character that the student wants to write down. The student might sometimes mistakenly think that the first bubble represents 0 and thus even if the intended digit is 0 the intended bubble might be 1. Thus, this must also be done in a probabilistic manner to compensate for this uncertainty. The intended bubbles and character then produces evidence, as seen in Figure 4.10.

When looking at Figure 4.11, it is observed that there are 11 evidence areas to consider. They are the the 10 bubbles and the character block. The process of writing

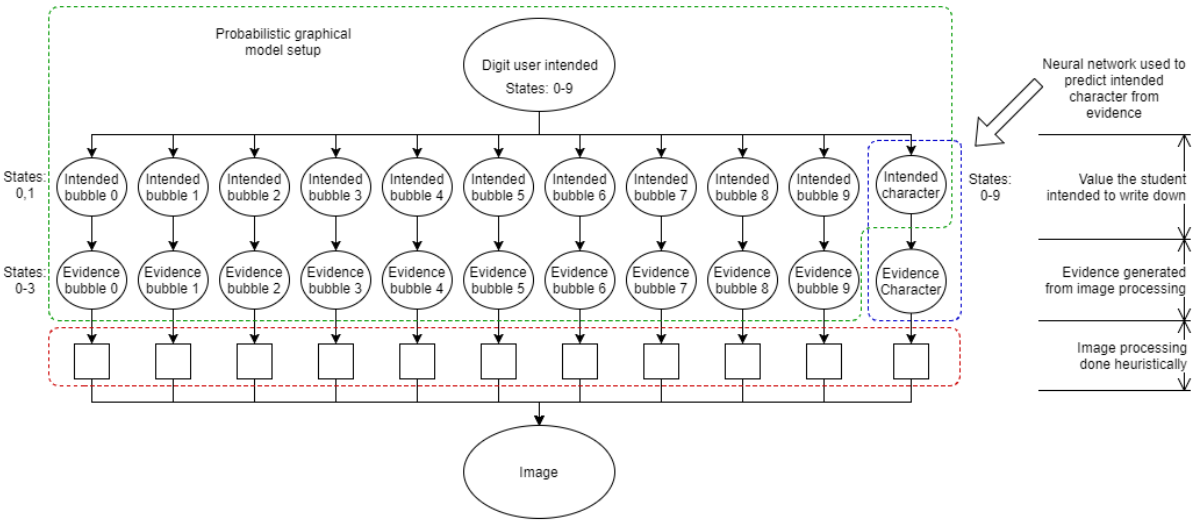

Figure 4.10: Graphical setup for determining the intended digit written by a student.

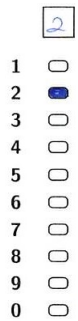

Figure 4.11: Column with the evidence that gets considered for the calculation of an intended digit.

down this digit introduces so noise into the system, due to the fact the student is not always going to write down the digit in exactly the same way. Thus the evidence is also probabilistically linked to the 'intended bubble' and 'intended character' parent distribution. This evidence then gets written down on the paper and is what ultimately influences how the image looks. Each of the bubbles can take one of 4 states as evidence. These states are blank, crossed-out, partially coloured in and fully coloured in. The character block evidence is a 28 by 28 greyscale image. Thus it can have 28*28*256 possible states, where 256 is the possible pixel intensities of each pixel. This value with true range 0 to 256, gets converted into a normalized value between 0.0 and 1.0 before it gets assigned as inputs to the neural network.

4.2.2.2 Estimating the intended digit

After the model is constructed the intended digit needs to be estimated, with the image as evidence. This can be done by reasoning from the bottom (image evidence) upwards to the intended digit. The first step is to process the image, using of image processing, to produce more tractable evidence. Producing the bubble evidence from the image is described in Chapter 3. In Section 4.1.2, the process to extract the character evidence from the image is also described. Using the neural network the probability of intended character, given the character box, can be determined. After these steps is completed the intended digit can be estimated using the PGM. This is done by first setting the evidence node to the values calculated when image processing were done on them. The 'intended character' node then gets set to prior calculated through the use of the neural network. After this is done inference is done on the PGM and the resulting probability of each digit is calculated for the 'intended digit node'. The PGM structure can again be seen in Figure 4.10.

4.2.3 Intended answer model

4.2.3.1 Structure of the graph

There are 8 possible columns to use for an answer. The first column represents the sign of an answer. Thus two signs are possible. For each of the remaining 7 columns a number from 0 - 9 can be represented. Thus there are ten possible values for each column. This gives a possible number of values that a answer can take to be $2 * 10^7 =$

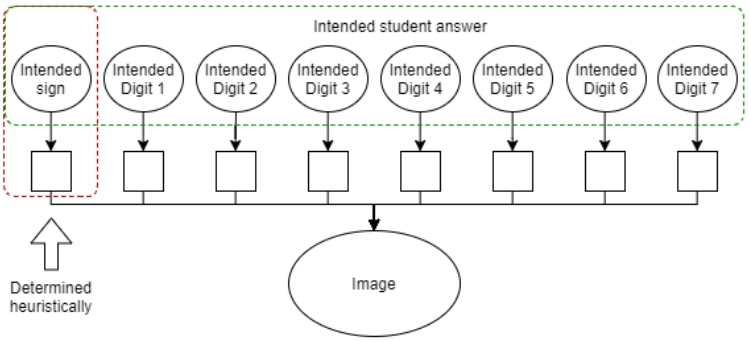

Figure 4.12: Graphical setup of determining student answer.

20000000. Each of these states needs to be calculated and becomes computationally intractable. Thus a assumption is needed to reduce the number of possible states. A fair assumption to make, is that all 20000000 possible values are equally likely to be written down. Thus each column digit becomes independent of the another columns values. This means that if a value in one column is known, it does not influence the probabilities of the other columns being a certain value. Thus the number of states to calculate now becomes $2 + 10 * 7 = 72$, because each column's states can now be calculated independently. Thus knowing this independent property the answer model can be described as a combination of 7 digit models with a heuristic calculation of the intended sign. This model is illustrated in Figure 4.12.

4.2.3.2 Estimating the intended answer

In the previous section it was found that determining the intended sign and digit for each column the intended answer can be found. Thus the intended digit for each column is calculated separately using the digit model, as described in Section 4.2.3.2. The intended sign is determined heuristically by using the image processing methods described in Chapter 3 to find if the bubble underneath the sign is coloured in. Once this is done the intended answer is determined by combining the estimated sign and digits in the order seen in Figure 4.12.

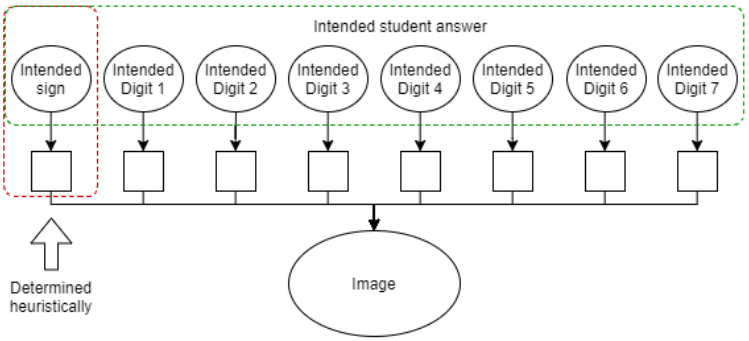

Figure 4.13: Graphical setup of determining a student answer.

4.2.4 Student number model

4.2.4.1 Structure of the graph

For a student number additional information is present. Knowing the digit value of one column changes the probability of each other column taking a certain value. The reason for this is, because there is only a limit number of student numbers to consider. Thus if the first digit is a 2, only student numbers starting with 2 still have to be considered. To account for this fact an additional node is added above the individual digit probabilities. This node represents the probability of each student number being present in the image and is related to the intended digits. The student number thus produces the intended digits. This node will have $+-900$ states, depending on the number of student numbers. The student number graph can be seen in Figure A.3.

4.2.4.2 Estimating the student number

Now that the graph has been setup the model can be used to infer the most probable student number. By setting all the bubble evidence and character priors the student number probabilities can be inferred. This model allows for a accurate result, because most student numbers are quite different to one another. It is found that if the student provides only character information and no bubbles are coloured in, the student number can still be estimated reliably. These results is states in Section 5.

4.2.5 Training the model

For this project the conditional probabilities for the intended digit model was found by using training data. From these distributions the answer model can also be constructed due to it only consisting of independent digit models. The probability of a student number given the intended digits was estimated manually.

As seen in Figure 4.10,the digit PGM will include the bubble evidence as well as a prior probability which the neural network provides. Once these values are assigned, the PGM model will infer the intended digit using the conditional probability distributions specified. To do this the conditional probabilities of the PGM, must first be determined from data.

The process of training a PGM is a simple process once enough training set is present. This was done by allowing the initial software, without the PGM model, classify the digits. Thus the training set contains the bubble and neural network outputs as evidence. The training set then also includes the intended bubbles and intended digit, as estimated by the image processing unit. Once all these test data are gathered, the conditional probabilities can be calculated. These probabilities then gets stored and used as prior probabilities in the PGM.

4.3 Conclusion

This chapter looked at two machine learning techniques to improve the accuracy with with the system infers the answers written on each scanned test sheet. A method was shown, using a neural network, to estimate the probability of each digit given only the character box as input. Additionally a approach was discussed, using a PGM, to allow the system to make a final prediction of what the student intended to write down given the bubble and character boxes as evidence.

The following chapter will cover the validation and results of the system from weekly grading done for the Applied mathematics department.

Chapter 5

Analysis of results

This still needs to be written up.

Document the accuracy of the system before and after the machine learning components have been added. Compare the results.

5.1 Future improvements

Increase speed of test. Use the Radon transform to locate circles. Then do image processing only on that areas. No need for contour finding on the whole image.

Chapter 6

Summary and conclusions

This section still needs to be written up.

6.1 Project summary

6.2 How this final year project benefits society

Must say something here. Also say something about ethics, how you realised that your work may have an impact on the environment (or not), *etc.*

6.3 What the student learned

During the execution of this project, the student learned that time management is important, because ... Also, using LaTeX is of cardinal importance to professionally typeset a report.

6.4 Conclusion: Summary and conclusions

This chapter provided a summary of the objectives, methodology and results of the final year project....

References

Edoras (2017). The inverse radon transform. Image Processing Toolbox. ix, 17

Google (2017). Deep mnist for experts. 10

Ivetic, D. & Dragan, D. (2003). Projections based omr algorithm. 8

Karpathy (2017). Cs231n convolutional neural networks for visual recognition. ix, 28

Nielsen, M.A. (2015). *Neural Networks and Deep Learning*, vol. 1. Determination Press. 27

Rosebrock, A. (2016). Bubble sheet multiple choice scanner and test grader using omr, python and opencv. 10

Tensorflow (2017). Mnist for ml beginners. ix, 29

VijayaForm (2017). Omr sheet. ix, 9

Appendix A

Overview of system

In this chapter a graphical and mathematical derivation of the entire system is given.

A.1 System as a whole

As described in Chapter 1, the system can fundamentally be represented with 6 information nodes. These nodes are shown in Figure A.1. The student has certain information he/she wants to portray on the paper. This includes the 4 answers and student number the student wants to write down. Thus those 5 nodes gives rise to the image, representing the last node. Thus fundamentally the system is tasked with working out these 5 conditional probabilities: $P(StudentNumber/Image)$, $P(Answer1/Image)$, $P(Answer2/Image)$, $P(Answer3/Image)$ and $P(Answer4/Image)$. The random variables StudentNumber and Answer(1 to 3) thus represent all the possible values that

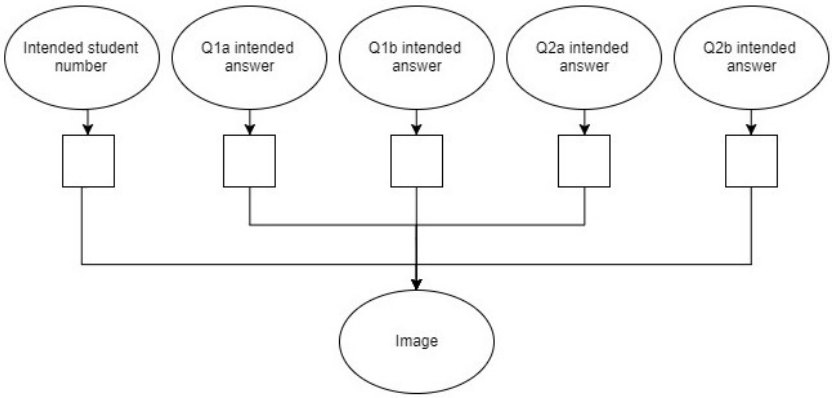

Figure A.1: System overview.

the student number and answers possibly can be. The image is a random variable representing the total number of possible states the image can represent. This value is in the order of 1240*1754*256 possible states. To practically represent this further assumptions are made in the following sections.

The aim of the software is to maximize the likelihood of those probability distributions indicating the correct answers. The reason the problem is represented in a probabilistic way is due to the fact that different images are going to be generated every time a test is written. This is true even when the same information is going to be portrayed. Everytime a student writes a test he/she is going to write in different ways. A probabilistic graphical model (PGM) is thus used to describe this system and its inner operations. For a more detailed explanation on PGM's, refer to Section 4.2. The blocks in Figure A.1 represent additional processing that is described in the following sections.

To calculate those 5 probabilities, some more detailed derivations are necessary. These derivations are described in the next section.

A.2 Deriving the intended student entry

A.2.1 Student answer

In Section 4.2.4 it was determined that the student answer can be calculated by combining the intended sign and digits of each column to form an answer. The reason for this could be attributed to the fact that these digits where independent of one another. This means that the intended digit in a certain column is not influenced by what the values in other columns are. In Equation A.1 this independence property can now be seen. To find the most probable answer only $P(sign/Image)$ and $P(digit(1-7)/Image)$ still needs to be calculated. Using the image processing techniques described in Section 3 the $P(sign/Image)$ can simply be determined heuristically by determining the probability of the bubble being coloured in, underneath the sign. Thus the only values yet to calculate is $P(digit/Image)$. This is described in Section A.3.

$$P(Answer/Image) = P(sign/Image) * P(digit1/Image) * ... * P(digit7/Image) \tag{A.1}$$

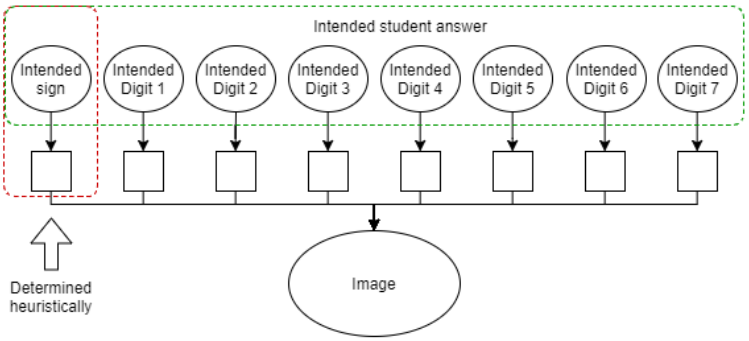

Figure A.2: Graphical setup of determining student answer.

A.2.2 Student number

As state in Section 4.2.4 the assumption of independence does not hold in the case of a student number. The reason for this is, because every 8 digit number is not equally likely to be a student number. Only student numbers that are valid will be considered as a possible state that the student number node can take on. This node will have $+-900$ states, depending on the number of student numbers. The student number graph can be seen in Figure A.3. Next a derivation for $P(studentNumber/Image)$ is needed.

In the below equations, 'Digits' represents the 8 digit random variables. The first step is to apply Bayes rule as seen in Equation A.2.

$$P(StudentNumber, Digits/Image) = \frac{P(Image/StudentNumber, Digits) * P(StudentNumber, Dig}{P(Image)} \tag{A.2}$$

When all the digits are known, the image becomes independent of the student number. This can be seen in Equation A.2. Also the chain rule states that Equation A.4 must be true.

$$P(Image/StudentNumber, Digits) = P(Image/Digits) \tag{A.3}$$

$$P(StudentNumber, Digit) = P(Digit/StudentNumber) * P(StudentNumber) \tag{A.4}$$

Next the Digit random variable is summed out to produce $P(studentNumber/Image)$, as seen in Equation A.5.

$$P(StudentNumber/Image) = \sum_{D}^{All} \frac{P(Image/Digits) * P(Digit/StudentNumber) * P(StudentNum}{P(Image)} \tag{A.5}$$

Applying Bayes rule again the following equality can be presented, as seen in Equation A.6.

$$P(StudentNumber/Image) = P(Image/Digits) = \frac{P(Digits/Image) * P(Image)}{P(Digits)} \tag{A.6}$$

Thus the result can be seen in Equation A.7. $P(StudentNumber)$ can be taken out of the sum, due to it being independant of $Digits$.

$$P(StudentNumber/Image) = P(StudentNumber) * \sum_{D}^{All} \frac{P(Digits/Image) * P(Digits/StudentNur}{P(Digits)} \tag{A.7}$$

$P(StudentNumber)$ can be initialized as a equal distribution, because every student number is equal as likely to be in a given test. $P(Digits/StudentNumber)$ is a value that is trained from data. This value symbolizes the probability that the user intended to write down a given digit given that student's student number. This number is thus strongly correlated with the student number. If the first digit of the student number is 1, Digit1 will have a high probability of being 1. The only values that still needs to be calculated are thus $P(Digits/Image) = P(Digit1/Image) * ...*$. This derivation is covered in the next section.

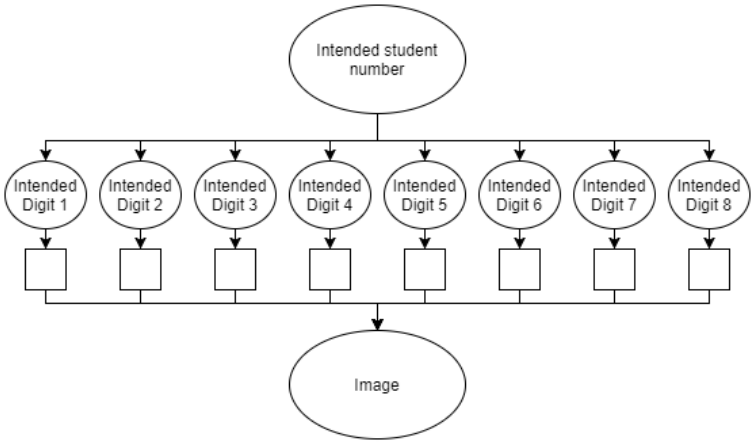

Figure A.3: Graphical setup of determining student number.

A.3 Deriving the estimated digit

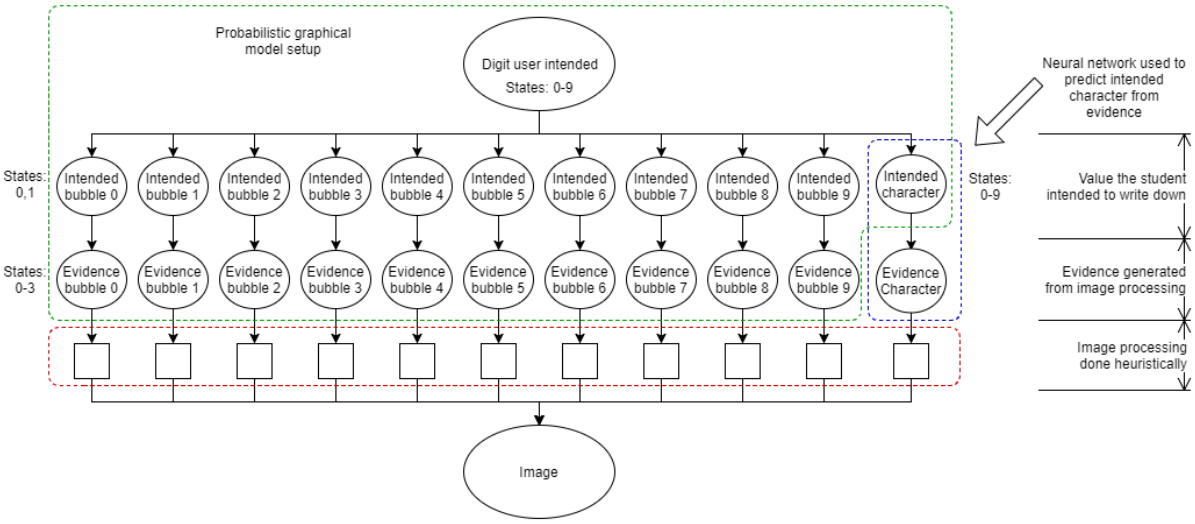

Figure A.4: Graphical setup of determining intended digit.

Appendix B

Implementation/Algorithms

Algorithms used in this project.

B.1 Deep Convolutional Neural Network

Appendix C

Validation and results

Validation and results of this system.

Appendix D

Outcome compliance

ID	Task Name	Duration	Start	Finish	Predecessors	% Complete
1	Write Problem Statement	1 day	Wed 11/03/16	Wed 11/03/16		99%
2	Draw Up Skripsie Project Plan	1 day	Wed 11/03/16	Wed 11/03/16		100%
3	Submit Deliverable One to Lecturer	0 days	Wed 11/03/16	Wed 11/03/16	1,2	100%
4	Receive Feedback on Deliverable One from Lectu	0 days	Fri 11/03/18	Fri 11/03/18	3	100%
5	Read Through Eden User Guide	0 days	Sun 11/03/20	Sun 11/03/20		50%
6	Determine Project Scope	5 days	Mon 11/03/21	Fri 11/03/25	1	70%
7	Finalise Deliverable One	1 day?	Mon 11/03/28	Mon 11/03/28	6	100%
8	**Submit Deliverable One**	**0 days**	**Wed 11/03/30**	**Wed 11/03/30**	**4,5,6,7**	**100%**
9	**Literature Study**	**102 days?**	**Thu 11/03/31**	**Fri 11/08/19**	**5**	**100%**
10	Read Thesis on ATM Cash Management for a South African Retail Bank	1 day?	Thu 11/03/31	Thu 11/03/31	8	100%
11	Read Related Work done by Adendorff and Wagner	15 days?	Mon 11/04/04	Fri 11/04/22	8	100%
12	Read Up on VRP's	65 days	Mon 11/04/04	Fri 11/07/01	8	100%
13	Read Up on Integer Programming	40 days	Mon 11/04/04	Fri 11/05/27	8	100%
14	Read Up on Inventory Management Models	45 days	Mon 11/04/04	Fri 11/06/03	8	100%
15	Read Up on Latex	95 days	Mon 11/04/04	Fri 11/08/12	8	100%
16	Read Up on Simulation	49 days	Tue 11/04/26	Fri 11/07/01	8	100%
17	Related Reading	100 days	Mon 11/04/04	Fri 11/08/19	8	100%
18	Write Literature Study	28 days	Mon 11/04/04	Wed 11/05/11		100%
19	Develop Framework of Target Report	25 days	Thu 11/04/07	Wed 11/05/11	5	100%
20	Develop Solution Strategy	25 days	Thu 11/04/07	Wed 11/05/11	5	100%
21	Update Project Plan	0 days	Wed 11/05/11	Wed 11/05/11	19,20,18	100%
22	Submit Final Queries about Deliverable Two to Lecturer	0 days	Wed 11/05/11	Wed 11/05/11	19,20,21,18	100%
23	Receive Feedback on Queries about Deliverable Two from Lecturer	0 days	Fri 11/05/13	Fri 11/05/13	22	100%
24	Incorporate Lecturer's Feedback	2 days	Mon 11/05/16	Tue 11/05/17	23	100%
25	May/June Exams	15 days?	Mon 11/05/16	Fri 11/06/03		100%
26	**Submit Deliverable Two to Lecturer**	**0 days**	**Wed 11/05/18**	**Wed 11/05/18**	**23**	**100%**
27	Receive Feedback on Deliverable Two	0 days	Fri 11/05/20	Fri 11/05/20	26	100%
28	June Holidays	10 days	Mon 11/06/06	Fri 11/06/17	25	100%
29	Develop Operational Research Models	40 days	Mon 11/06/20	Fri 11/08/12	20,10,11,28	100%
30	Run Simulations to Test Models Developed	20 days	Mon 11/07/18	Fri 11/08/12	20,16	100%
31	Write Report	20 days	Mon 11/08/15	Fri 11/09/09	1,6,19,29,30	100%
32	Draft Report Submitted at the Writing Institute for Proofreading	3 days?	Mon 11/09/12	Wed 11/09/14	31	0%
33	**Formal Submission of 1st Draft**	**0 days**	**Mon 11/10/03**	**Mon 11/10/03**	**32**	**100%**
34	Receive Feedback on 1st Draft	0 days	Fri 11/10/21	Fri 11/10/21	33	100%
35	Incorporate Final Comments on 1st Draft	4 days	Fri 11/10/21	Wed 11/10/26	34	100%
36	**Submit Final Report**	**0 days**	**Wed 11/10/26**	**Wed 11/10/26**	**35**	**100%**
37	Prepare for Oral Assessment	7 days	Thu 11/10/27	Fri 11/11/04	36	0%
38	October/November Exams	11 days	Fri 11/10/28	Fri 11/11/11		0%

Figure D.1: Project plan for the final year project.

Appendix E

Project plan

The final year project plan is shown in Figure E.1. This idea of this Gantt chart plan was to give the student an indication of the progress of the final year project. It was last updated as a final adjustment to the project report.

Figure E.1: Project plan for the final year project.