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

Check of dit reg is??

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*

Abstract

English abstract here.

Opsomming

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

Contents

List of Figures

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 a 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 analyze and grade tests written on a specific template. These answer sheets will be 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 will be done in parallel with the development and expansion of the test grading software. For these reasons an agile development mythology will be used with a weekly validation test, as the system gets used.

1.2 Problem statement

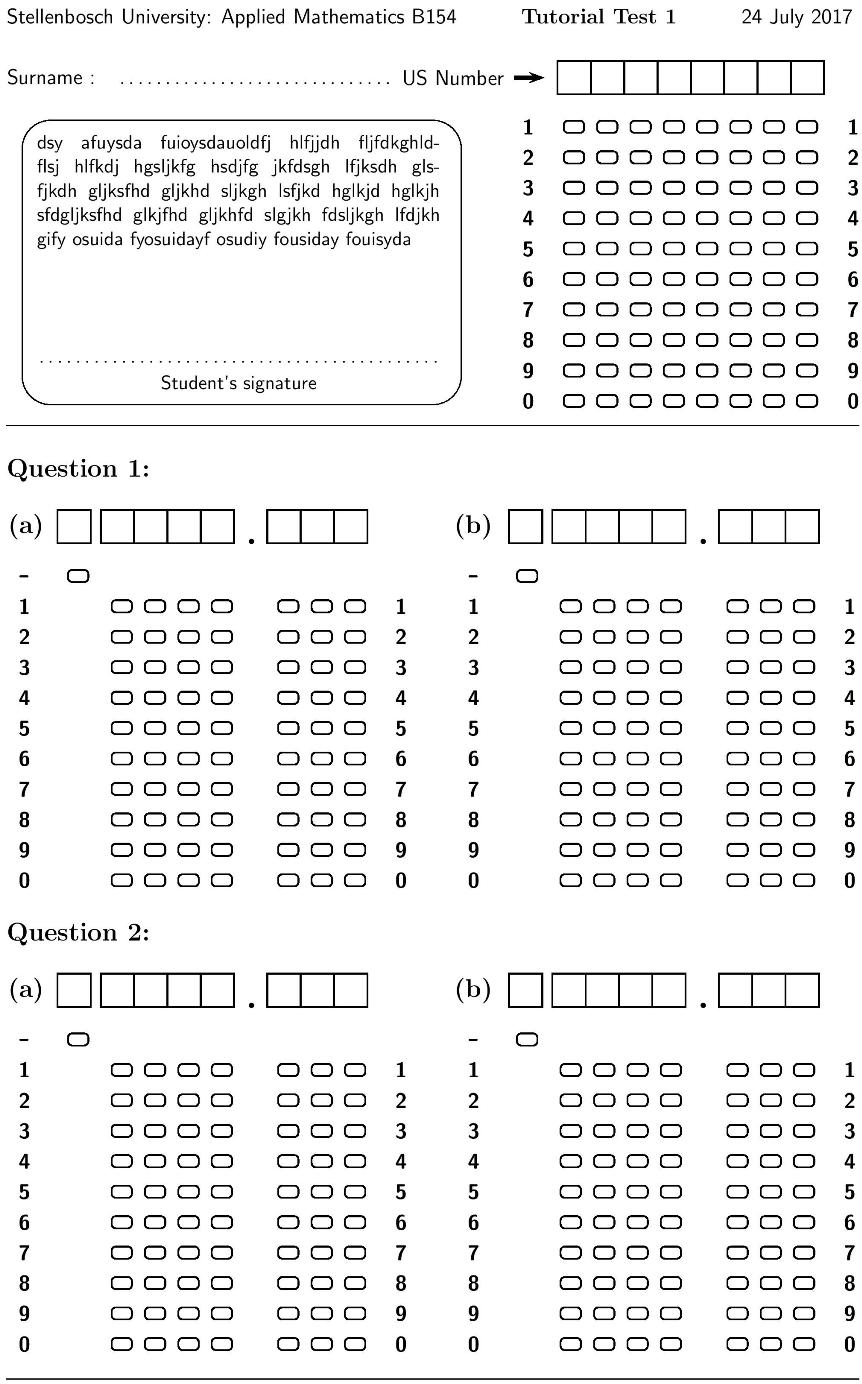
Given the problem background and stakeholders discussed in the previous section the problem to be solved can be formulated as

> **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 colored 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 thesis will be on processing that scanned in document written on the specific template. Too use the template the student must fill in their 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 n bubble next to each question is provided if a negative sign is required.

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


Stellenbosch University: Applied Mathematics B154 **Tutorial Test 1** 24 July 2017

Surname : .... US Number →

dsy afuysda fuioysdauoldfj hlfjjdh fljfdkghld-flsj hlfkdj hgsljkfg hsdjfg jkfdsgh lfjksdh gls-fjkdh gljksfhd gljkhd sljkgh lsfjkd hglkjd hglkjh sfdgljksfhd glkjfhd gljkhfd slgjkh fdsljkgh lfdjkh gify osuida fyosuidayf osudiy fousiday fouisyda

....
Student's signature

1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
0 0

Question 1:

(a) .

(b) .

\-

1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
0 0

Question 2:

(a) .

(b) .

\-

1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
0 0

Figure 1.1: Automatic grading template 1.

1.4 Project objectives

The problem as stated in section ??, will be 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 is then used as the student's grade for that test.
4. Use an *agile development mythology* 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 will be covered in different chapters in the report. Note that each objective (1–5) built 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 will be 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 mythology will be 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 currently 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 recurs steps 2-4 until the new software is working.

6. Backup the new working software using Git or another backup utility.

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

For a detailed derivation of the entire system, refer to Appendix B.

1.6.1 System overview

When thinking about the system(for template 1), from a philosophical sense, it can be represented by 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 1.2. The unnamed blocks indicate that information processing happens in these steps. For a more detailed visual overview of the system, please refer to Appendix A.

1.6.2 Graphical representation

For this thesis a graphical approach is followed in representing the software developed. A graphical model in essence allows software to be represented as information (represented by circles) and relationships(represented by directed arrows). The directions of the arrows represents what information causes other information to be created(From

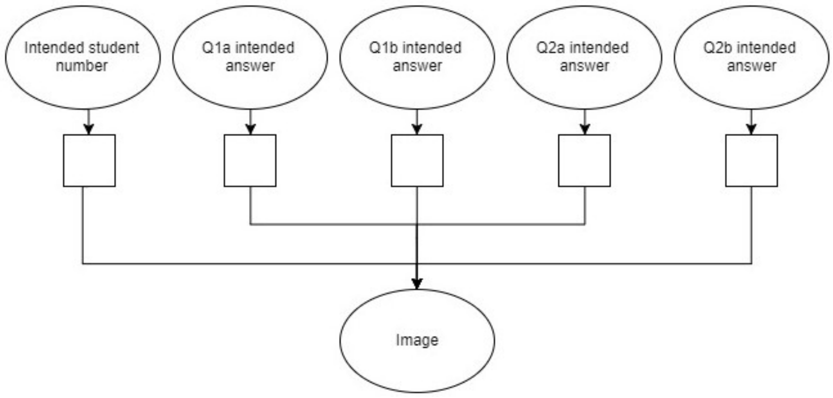


Figure 1.2: System overview.

parent to offspring). This can again be seen in Figure 1.2. 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 modeling(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 will be 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 thesis was laid out. Further a brief system overview was also provided. This chapter will provide information about already excising systems and the techniques they use in doing image processing and character recognition on images. There will also be looked at already excising libraries to aid in the implementation of these techniques.

2.1 Existing systems

The types of systems used to grade test automatically are commonly known as a Optical mark Recognition(OMR) systems. OMR systems are used to extracted answers from filled in forms, using a special template. These test are normally used when fast and accurate grading of tests are needed, where test only have specific answer choices. OCR systems are thus excellent for the grading multiple choice type questions. The drawback of most of these systems are still that they normally can only mark colored in bubbles and not interpret characters on the paper.

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

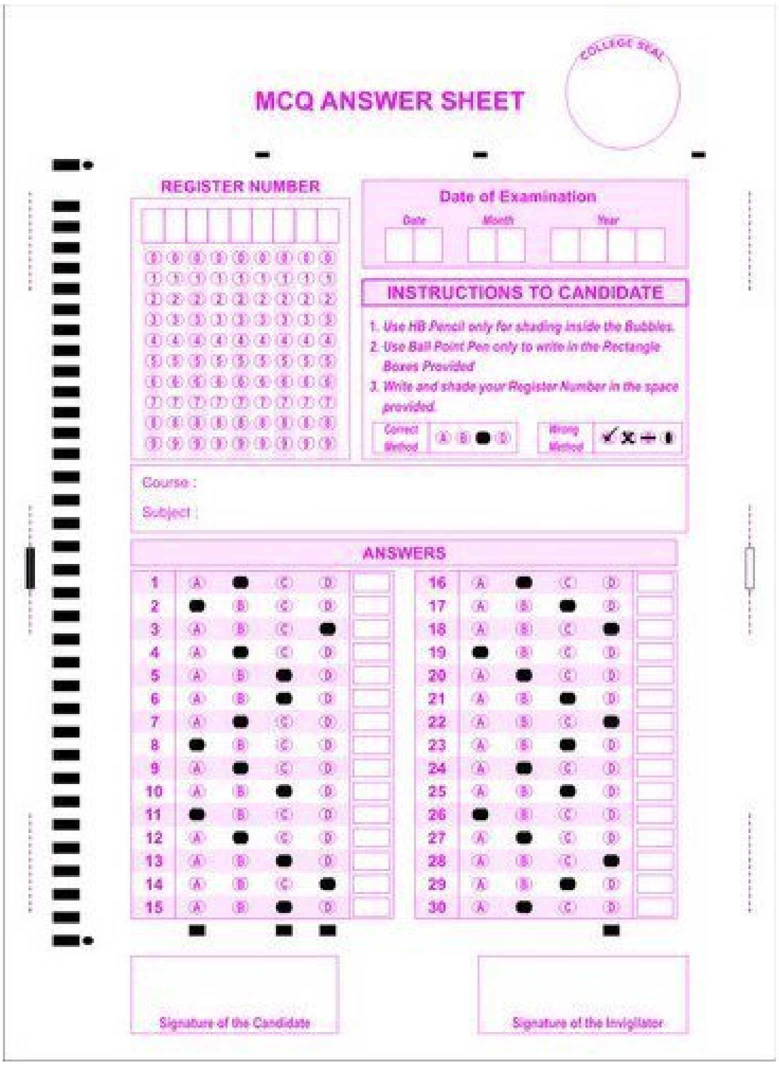


Figure 2.1: Standard OMR template, VijayaForm (2017)

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 weather that has been filled in.

2.2 Additional techniques

The above method allows for a simple OMR system to grade tests. For the test grader used for Applied Mathematics there is needed 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 needs 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 will be addressed by using a PGM, as seen in Section 4.2 (Sit in). 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 colored 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. With this information each bubble can be assessed not only by the pixels inside it, but also its shape. (Verander die image dalk na een wat 'n contour om het)

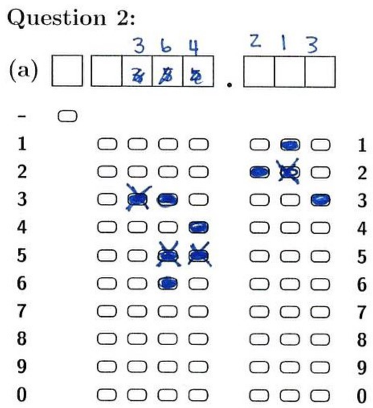


Figure 2.2: Example answer from student

2.2.2 Character recognition

To further increase the accuracy of the system optical character recognition (OCR) software will also need to be applied on the blocks with characters in. Doing some research, one preferred way of doing OCR is using TensorFlow as shown in Google (2017). This is also a python library, but allows the build of instructions to be implemented in efficient c++ code. For this test grader, TensorFlow will be used to setup a convolutional neural network to read a image, containing the digit, and predict the probability of each digit.

2.3 Conclusion: System requirements

I conclusion it can be seen that the system will need to incorporate a combination of image processing on the bubbles and OCR on the digits handwritten by the students. Taking this combined evidence an more accurate result can be estimate, while also improving the convenience for students writing these test.

Chapter 3

Image Processing

The previous chapter focused on already 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, will be described. By using only these image processing techniques a reasonably accurate system can already be constructed.

For further improvements in accuracy two machine learning approaches will be investigate and implemented. These approaches will be discussed in Chapter 4.

3.1 Orientation Detection

As mentioned in Section 2.1.1 there are two main parts in OMR grading. The first challenge with grading a scanned in answer sheet, is finding the orientation of the test sheet in the image. This can be done by finding five values, including the rotation, offset(xOff,yOff) and a reference point(xRef,yRef). In Chapter 2 it was found that the common way to implement this is to include specific reference markers on the page. a Disadvantage with this method is that if their is to little markers on the page, the student might accidentally write over them or write something that resembles a marker and confuse the system. To compensate for this the markers that will be chose, are already present on the template paper. They are the two longest horizontal lines as well as the two vertical lines on the comment box. Together these lines have enough information to determine the size, offset and rotation of the template. The reason these lines are chosen as references is due to the fact that a Radon transform can easily be applied to determine where they are, as seen in Section 3.1.2. 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 will be required and will be discussed next.

3.1.1 Initial filtering and orientation detection

To check if and image is upside down the software will first find relevant contours on the page and then filter out those it thinks probably does not contain bubbles or character information. This is done in 5 steps:

1. Threshold the image by making all the pixel values either white(lower that mean) or black(higher that 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 list will be sufficient. Once the list is found the software counts the number of contour centerpoints below and above the image center. Looking at Figure 3.4 it can easily be seen that more bubbles should be below the middle, if the image is not upside down. The next step will now be to determine the location of the answers the student wrote down. To do this the template must first be found in the image. This will now be done in Section 3.1.2.

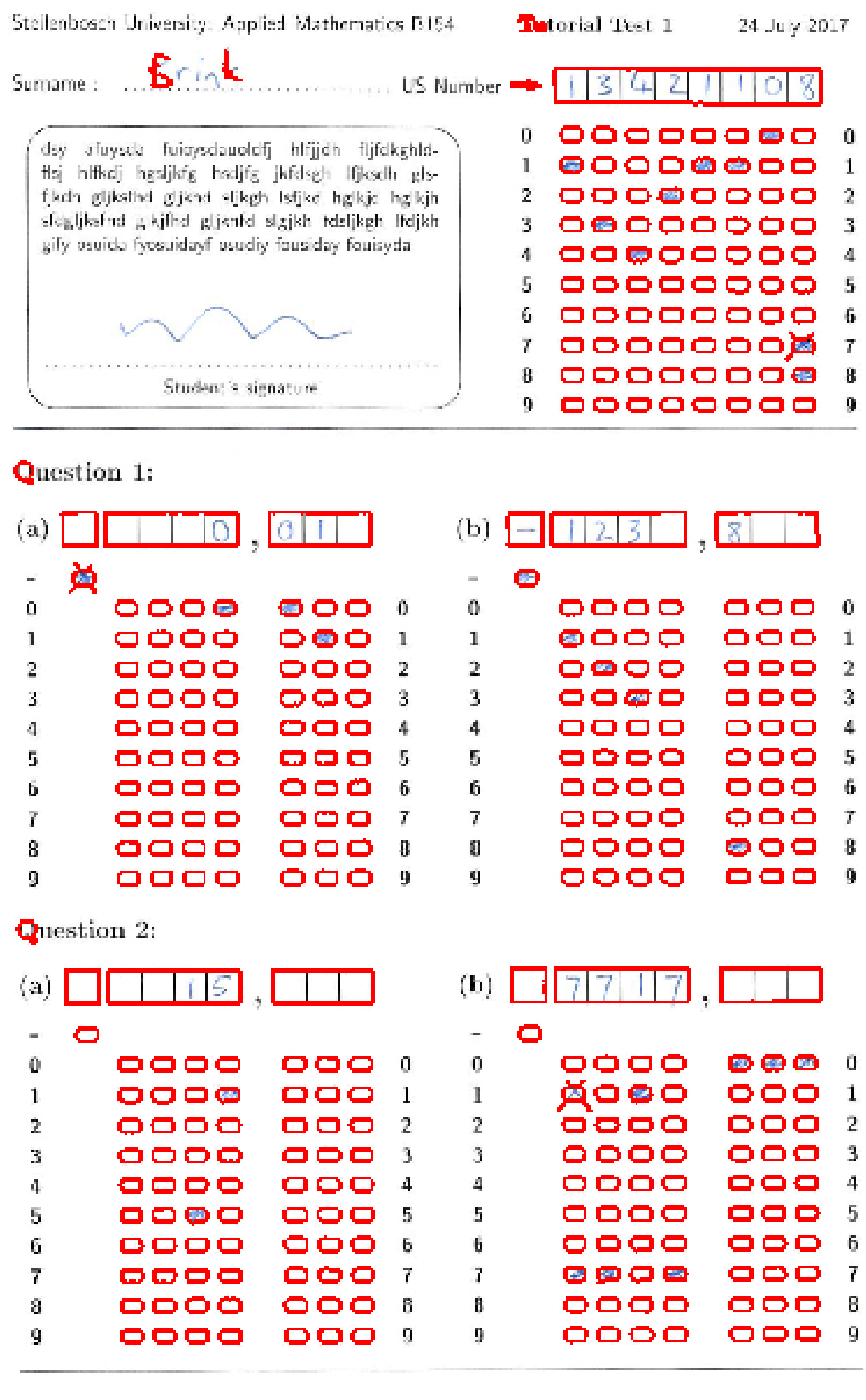


Figure 3.1: Reduced contours in image.

3.1.2 Radon Transform

And Radon transform is an integral transform that converts an 2 dimensional image into a 1 dimensional array. An example of this transform can be seen in Figure 3.2. This transform is always done over a chosen axis. In this case we will chose the horizontal axis to sum over. By counting up all the pixel values corresponding to the same horizontal row a single value is observed. By repeating this process for each row an resulting 1 dimensional array is created. This array provides information about the total concentration of that line. In the case of the test grader, this can be used to identify where the lines on the pages are, by looking at the maximum values. By transforming the image one can also look at the maximum values of each Radon transform, as this will almost always occur when the template is rotated to be aligned with the horizontal axis.

To find the template inside the image, the image needs to be rotated correctly first. To do this we apply a Radon transform at different angles until convergence, where the Radon transform has the highest maximum. Once the image is rotate correctly the two maximum values of the radon transform will indicate the two horizontal lines present on the image. Using those two lines the relative size of the template in the image can be determined and a y offset value. The last step is thus to determine the x-offset of the image. To do this a vertical Radon transform is applied to detect the two vertical lines of the comment box. This will provide the last x-offset value needed to find the template inside the image as well as additional size evidence to confirm the previous estimate. Once the template is found the bubble values can be determined, using preprocessing done on an empty template. Figure 3.3 illustrates the final estimation of all the bubbles in the template. The estimated bubbles are colored red while the green points represent the centers of all the remaining contours.

In Figure 3.4 The final rotation after applying consecutive Radon transforms can be seen.

In the next step a contour will be assigned to each bubble and then stored.

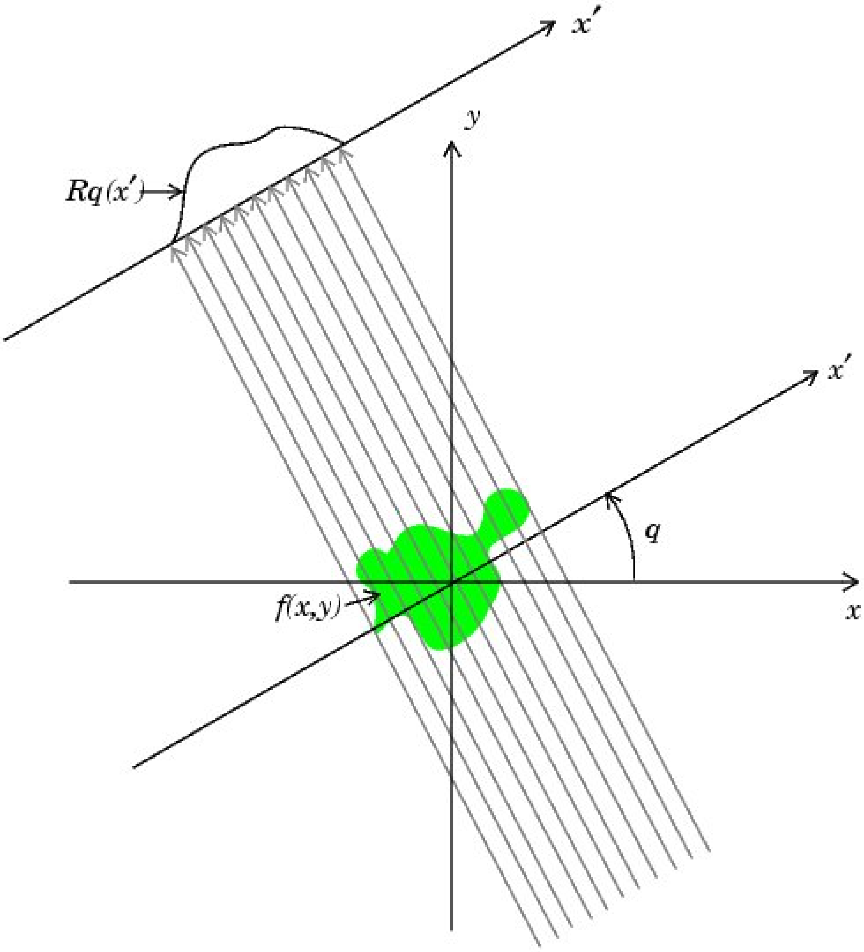


Figure 3.2: Radon transform applied on a 2 dimensional area.

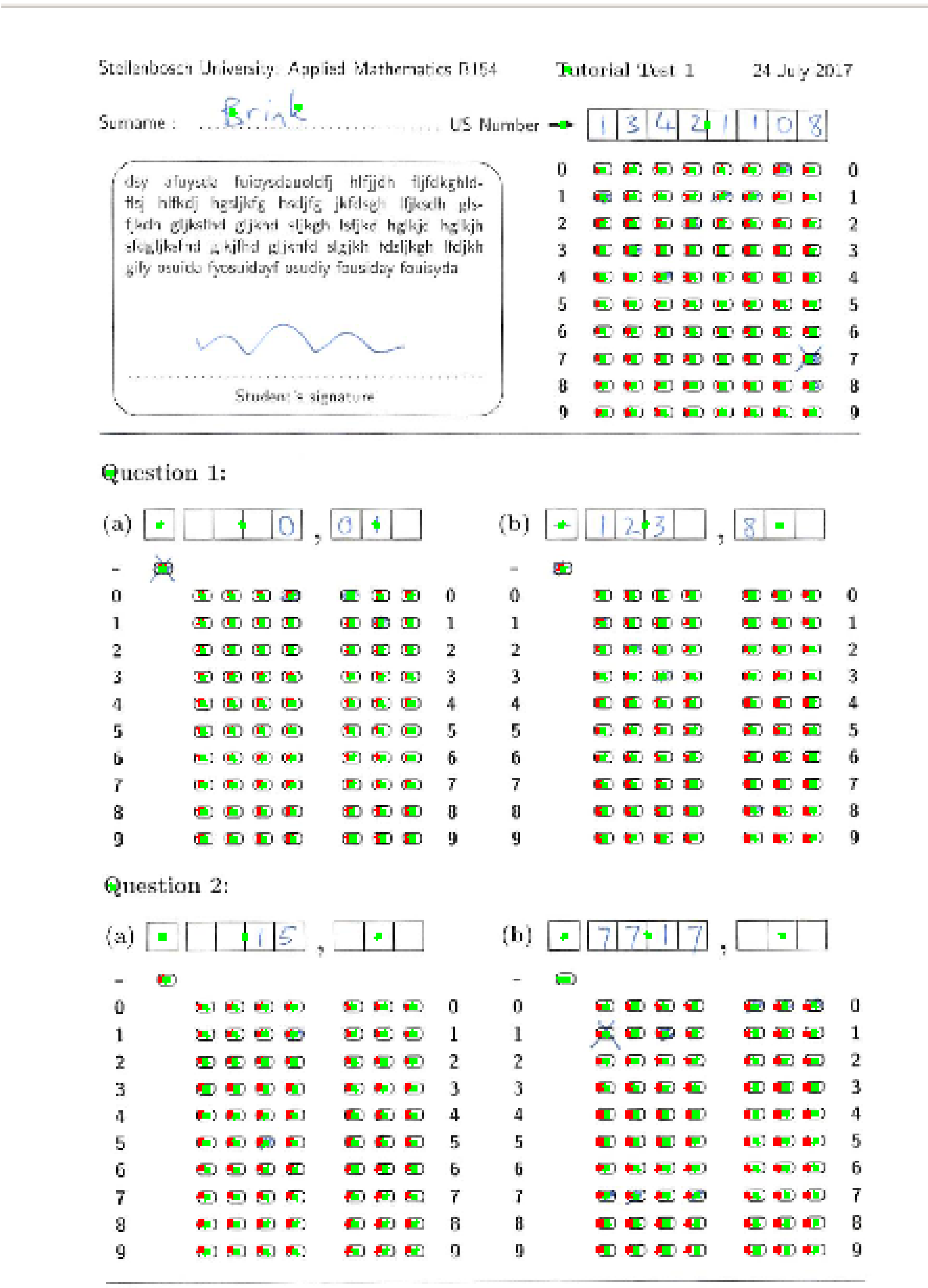


Figure 3.3: Detection of template in image and estimation of bubble locations.

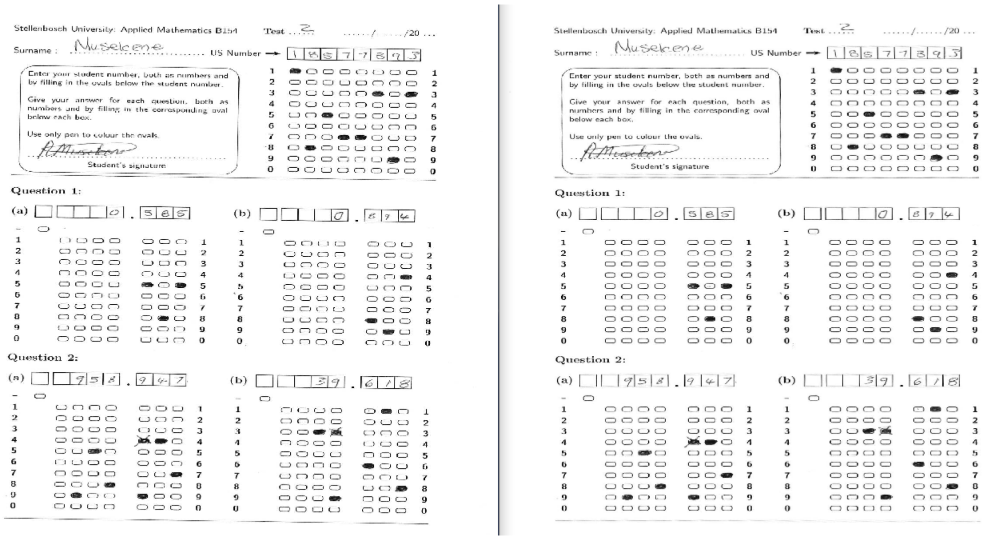


Figure 3.4: Result in rotation after applying radon transform.

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 there locations. Searching through the contours now becomes linear and of rank O(n), where n is the number of bubbles. 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 colored in or crossed out. The advantage of using the closest contour in the bubble's estimated location, over conventional methods of just using a area where the bubble is probably located now becomes apparent. By drawing the smallest block around the contour, that still covers every value inside the contour, an area can be calculated. This area will become large when a answer is crossed out, due to the lines stretching outside the initial bubble. By applying a threshold to this area value the system can successfully determine between filled-in and crossed out answers.

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 added 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. Detect the number of completely filled-in answers in each column.
2. If there are no completely filled-in answers, check the amount of partially filled-in answers and override the previous value.
3. If the previous result is 0, set the output value for that column to 0.
4. If step 2 or 3 presents more that 1 answer 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 information that has not been observed in the system. This information is the characters that the student writes into the designated boxes.

This next chapter will provide two machine learning approaches to significantly improve the accuracy of the system over the previous basic model alone. Firstly an approach to locate and classify hand written characters, provided by the students, using a deep convolutional neural network(DCNN), will be described. A more accurate method in determining the true digit represented by the bubble evidence, using a probabilistic graphical model(PGM), will also be implemented. This method will then allow for an integrated probabilistic approach to determine the true answer using the character and bubble evidence.

For a more detailed explanation on the DCNN used in this thesis please refer to Appendix C.1.

4.1 Character recognition using a neural network

4.1.1 Introduction

An neural network is a powerful machine learning tool for approximating complex functions. The basic architecture of an neural network can be seen in Figure 4.2. The structure of an feed-forward neural network consist of an input, hidden and output layer, as described in Nielsen (2015). A artificial neural network(ANN) is a simplified

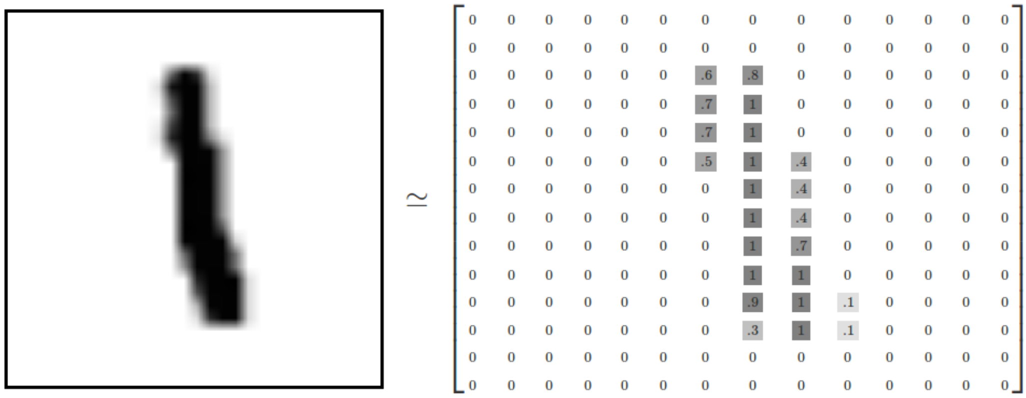


Figure 4.1: Example image used as input to the neural network, Tensorflow (2017)

approximation of how neurons in the brain works. Each neuron in the network acts as a small processing unit. The final output can thus be collected by reading the value at the output layer after it has been passed each layer of the network.

For this thesis a neural network will be trained to predict what digit(0-9) are most likely present in a image. Figure 4.1 illustrates the input of of the neural network using and 14 by 14 example image. For this thesis a 28 by 28 greyscalled image will be used as input. Thus if each pixel is 1 value(0.0-1.0) there will be a total of 748 input values.

4.1.2 Neural Network Basics

4.1.2.1 The artificial neuron

The neuron unit takes in decimal values(Inputs) and first calculates the weighted sum as seen in, Equation 4.1. Where n is the number of inputs. x_i and w_i respectively are the input and weight values at index i. The summed value then gets normalize using an sigmoid function, seen in Equation 4.2. This artificial neuron thus basically takes in a weighted input and produces a normalized out. By adjusting the weights certain inputs will be excite and inhibit the output of the neuron more. This process thus allows different functions to be approximated by changing these weights. These weights are then the variables that needs to be learned from data provided to it. If a network of these neurons gets place together, as seen in Figure 4.2, complex functions can be trained onto the network.

4.1.2.2 Getting a output from the network

After the 748 input values have been set each of the network coulombs can be calculated one at a time. 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, using Equations 4.1 and 4.2. Once the first column's outputs are calculated the next column can be calculated. This is done until all the columns are calculated in the hidden layer. The output layer is then calculated using the same method. For this thesis 10 output neurons will be used to correspond 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 Equation 4.3. Where $prob(i)$ is the probability of digit i being the character in the image. The value $Out(z_i)$ us the output of the output neuron at index i.

$$z = \sum_{i=0}^{n} x_i * w_i \tag{4.1}$$

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

$$prob(i) = \frac{Out(z_i)}{\sum_{k=0}^{10} Out(z_k)} \tag{4.3}$$

4.1.2.3 Training of neural network

To train a neural network the MNIST dataset will be used. This is a database that has a labeled training set of 60,000 images, and a labeled test set of 10,000 images. The neural network will be trained using the training set.The basic idea behind the training method used in a neural network will be described in the following steps. For a more detailed description refer to Appendix ...

1. Calculate the neural network output for each of the training images used for this training round.

2. Get the error margin of the network using a formula that compares the true labels of the training set with the estimated labels generated by the neural network.

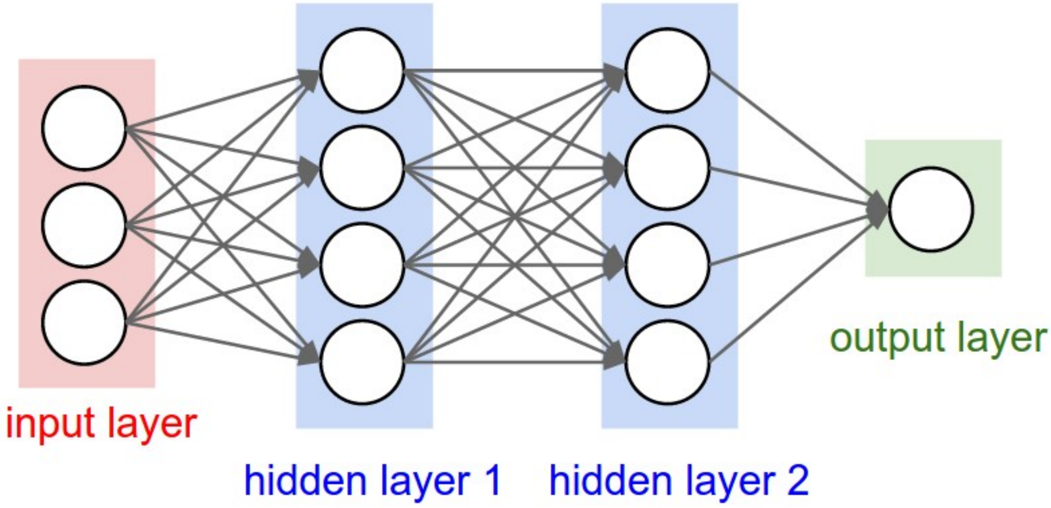


Figure 4.2: Basic structure of an neural network, Karpathy (2017)

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

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

4.1.3 Preprocessing on image

For the neural network to be able to classify the digits from the images, those images must first be found inside the image. 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 3.1.2. This is illustrated in Figure 4.3. The bubbles have already been filter out of the image in a previous process.

2. Transform the image to become fully rectangle using OpenCv's $four_{p}oint_{t}ransform$ 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 4.4.

3. Do a horizontal and vertical Radon transform, Section 3.1.2, to find and remove the dark box lines on the image, as seen in Figure 4.5.
4. Use the values received from the radon transform to segment the image into the different boxes.
5. Using a custom segmentation algorithm find the pixels most likely to belong to the digit, as seen in Figure 4.6.
6. Locate the area the pixels are located in, as seen in Figure 4.7
7. Finally calculate the center of the pixels, recenter and normalize the image area, as seen in Figure 4.8
8. Reshape the image into an 28 by 28 greyscalled image to be processed by the neural network.

Each box on the image can now be classified by the neural network and saved. The new character recognition evidence can now be used in combination with the bubble evidence to make a significantly more accurate estimate of the intended answers of the student. Next all these individual evidence must be combined in a intelligence way to make a accurate prediction. This method will be covered in the next section.

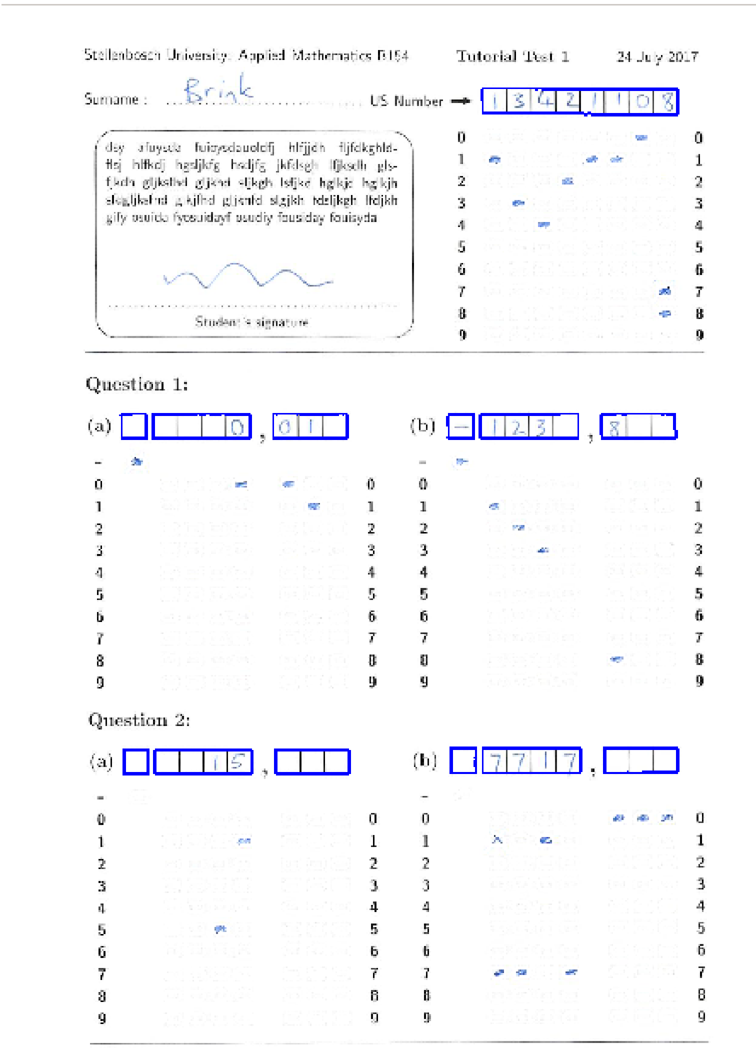


Figure 4.3: Image scanned for character areas.

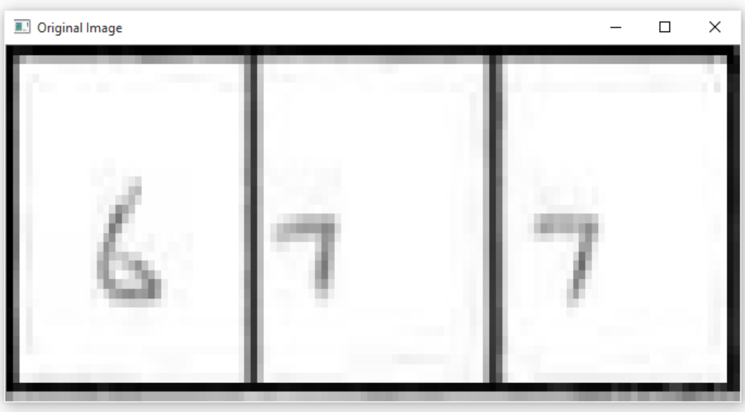


Figure 4.4: The found box is then normalized to rectangular shape.

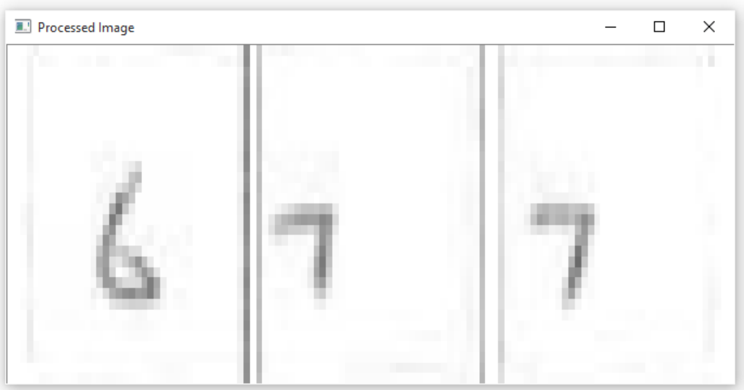


Figure 4.5: a Randon transform then gets applied.

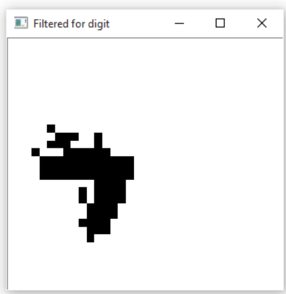


Figure 4.6: An clustering algorithm is used to find the main cluster in the remaining image.

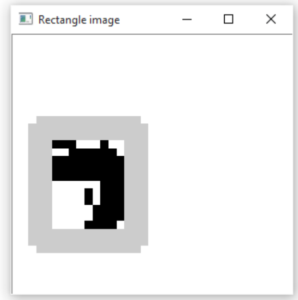


Figure 4.7: Area of found cluster.

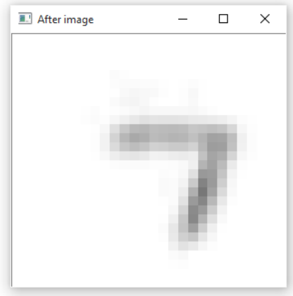


Figure 4.8: Final translation and normalization of image.

4.2 Probabilistic Graphical Models

The last step in this process is to probabilistic determining the true values the student, given all the evidence presented. To do this a probabilistic graphical model(PGM) will be developed.

4.2.1 Structure of the graph

a Probabilistic graphical model(PGM) is a probabilistic model of random variables, where the graph expresses the conditional independence structure between these variables. The type of PGM used in this thesis 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 another. The Figure should be interpreted in the direction which information flows. Firstly 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. This true 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. The intended bubbles and character then produces evidence, as seen in Figure 4.10. When looking at Figure 4.9, 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 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' or '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 colored in and fully colored in. The character block evidence is an 28 by 28 greyscale image. Thus it can have 28*28*256 possible states, where 256 is the possible pixel intensities of each pixel.

4.2.2 Determining the intended digit

Now that the model constructed the intended digit needs to be estimated, given the image. Thus is done by reasoning from the bottom(image evidence) and upwards to the intended digit. The first step is to process the image that produces the evidence using of image processing. Producing the bubble evidence from the image is described in Chapter 3. In 4.1.3, the process to extract the character evidence from the image was also described. Using the neural network the probability of intended character can be determined. To determine the intended digit of the student a PGM is constructed.

4.2.3 Training the model

This subsection still needs to be completed.

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

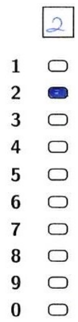


Figure 4.9: Column representing one digit.

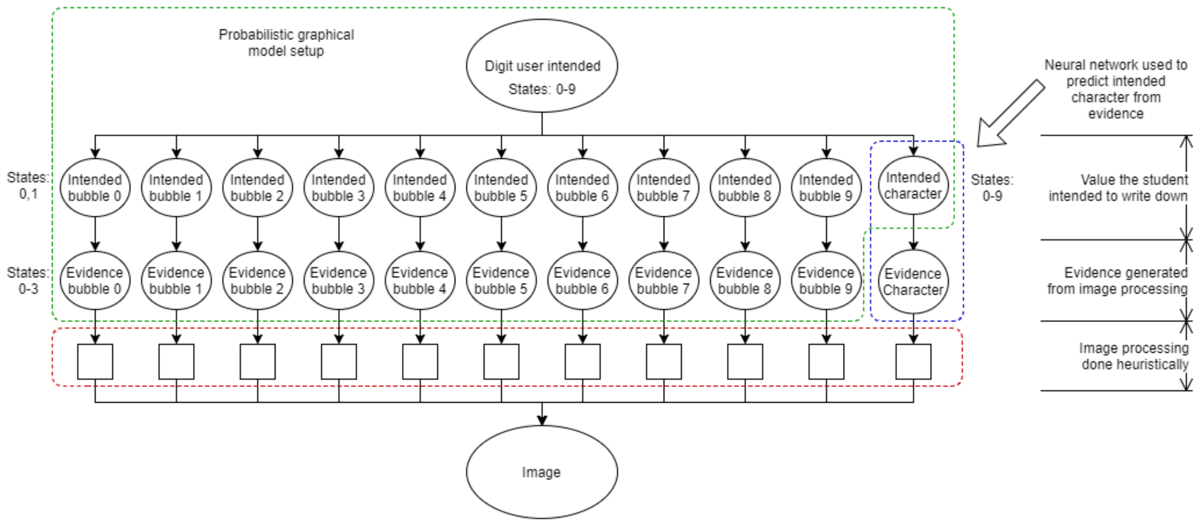


Figure 4.10: Graphical setup of digit generation.

4.2.4 Student number identification

To be continued..

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.

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

Google (2017). Deep mnist for experts. 10

Ivetic, D. & Dragan, D. (2003). Projections based omr algorithm. 7

Karpathy (2017). Cs231n convolutional neural networks for visual recognition. x, 22

Nielsen, M.A. (2015). *Neural Networks and Deep Learning*, vol. 1. Determination Press. 19

Rosebrock, A. (2016). Bubble sheet multiple choice scanner and test grader using omr, python and opencv. 9

Tensorflow (2017). Mnist for ml beginners. x, 20

VijayaForm (2017). Omr sheet. x, 8

Appendix A

Graphical overview of system as a whole

This is a graphical overview of the entire system.

Appendix B

Mathematical derivation of system as a whole

This appendix introduces and contains a probabilistic mathematical derivation that describes the overall workings of this system.

Appendix C

Implementation/Algorithms

Algorithms used in this project.

C.1 Deep Convolutional Neural Network

Appendix D

Validation and results

Validation and results of this system.

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.

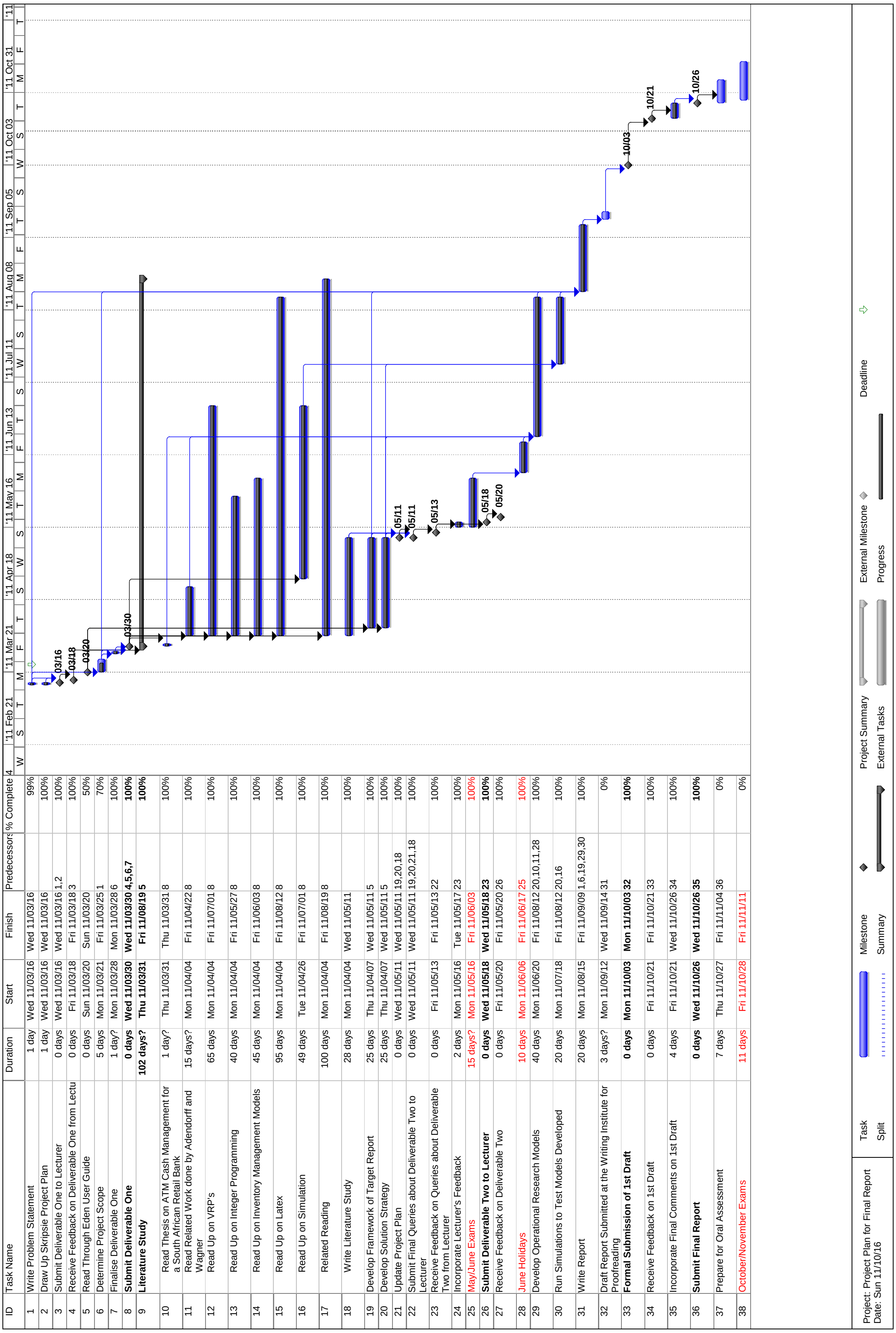


Figure E.1: Project plan for the final year project.