Department of Informatics
Technical University of Munich

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

BACHELOR'S THESIS IN INFORMATICS

TUMexam: Digitalizing Attendee Control for Examinations – Recognition of Student Card and Registration Number Box

Julian Villing

Technical University of Munich

Department of Informatics

Bachelor's Thesis in Informatics

TUMexam: Digitalizing Attendee Control for Examinations – Recognition of Student Card and Registration Number Box

TUMexam: Digitalisierung der Anwesenheitskontrolle in Klausuren – Erkennung der Student Card und Matrikelnummberbox

Author:	Julian Villing
Supervisor:	Prof. Dr.-Ing. Georg Carle
Advisor:	Dr.-Ing. Stephan Günther
	Maurice Leclaire
	Benedikt Jaeger
	Johannes Naab
Date:	December 16, 2019

I confirm that this Bachelor's Thesis is my own work and I have documented all sources and material used.

Garching, December 16, 2019

Location, Date

Signature

Abstract

Attendee control is a very important part of exams. It is usually done by filling out a form on the exam and signing on an attendee control list during examination. Instead of filling out a form on the exam, personalization with TUMexam is done using a sticker which is put onto the coverpage during attendance control.

While this approach works resonably well, it can be improved. Instead of the attendance control list procedure, a photo of the coverpage is taken. To make this work, it is necessary to have a registration number box on the exam cover which allows to identify the student later on. Additionally, a student card needs to be in the image to verify that the registration numbers on the coverpage and on the student card match.

The attendee control lists are either sorted by the students' last names or by a predefined seating order. If no seating lists are created in advance, the latter option is not possible. To parallelize attendee control it is required that these lists are ordered according to the seating lists. This would allow multiple examiners to start the procedure at different rows of the lecture hall simultaneously.

With our new approach, there is no need for attendee control lists or stickers. Instead, all the work is performed by the app on the examiners' mobile devices. This difference makes parallelization very easy as there are no predetermined orders in which the students need to be checked. Furthermore, reducing the error rate compared to the previous solution also reduces the necessary work to be done during the exam as fewer exams need special handling.

In this thesis we investigate how it is technically feasible to extract the registration number box as well as the student card and get the registration number from both of them. It is required that the new approach used must be better than the sticker-based solution in terms of accuracy. The current method correctly retrieves the aformentioned information in 99 % of all cases. That means that the new approach must have a higher precision than its counterpart and also should not consume more time than before.

Zusammenfassung

Die Anwesenheitskontrolle ist ein elementarer Bestandteil jeder Klausur. Standardmäßig wird dies über das Ausfüllen der Felder auf der Klausur und dem Unterschreiben auf einer Anwesenheitsliste umgesetzt. TUMexam verwendet auf dem Deckblatt einen Sticker zur Personalisierung, der während der Anwesenheitskontrolle aufgeklebt wird, anstatt dass der Student die Felder selbst ausfüllen muss.

Der Ansatz funktioniert zwar gut, man könnte ihn aber verbessern indem man statt der Anwesenheitsliste ein Foto vom Deckblatt der Klausur macht. Damit die Klausur auch im Nachhinein noch dem richtigen Studenten zugeordnet werden kann, wird ein Deckblatt mit einer Matrikelnummerbox benötigt. Um sicherstellen zu können dass die Matrikelnummer korrekt angekreuzt wurde, muss zusätzlich ein Studentenausweis im Bild enthalten sein. Mit diesem kann die angekreuzte Nummer verglichen werden.

Die Anwesenheitslisten sind entweder nach dem Nachnamen oder der Sitzplatzliste sortiert. Bei freier Wahl des Sitzplatzes ist die letztere Reihenfolge nicht möglich. Um die Anwesenheitskontrolle zu parallelisieren muss die Anwesenheitsliste nach der Sitzplatzliste sortiert sein. Durch Auftrennen der Liste wird es der Klausuraufsicht ermöglicht, die Kontrolle bei verschiedenen Reihen im Hörsaal zu beginnen.

Mit unserem neuen Ansatz werden keine Anwesenheitslisten oder Sticker benötigt. Dies wird stattdessen durch die Anwendung auf den Smartphones der Aufsicht übernommen. Dieser Unterschied macht die Parallelisierung sehr einfach weil es keine vorgegebene Reihenfolge gibt, in der die Anwesenheit überprüft werden muss. Eine Reduzierung der Fehlerrate gegenüber der aktuellen Situation verringert zudem den Aufwand zusätzlich, da weniger Klausuren gesondert behandelt werden müssen.

In Rahmen dieser Bachelorarbeit untersuchen wir, inwiefern es technisch möglich ist, die Matrikelnummer und den Studierendenausweis zu extrahieren. Unser neuer Ansatz muss besser als der auf Stickern basierende Ansatz sein. Aktuell werden die zuvor genannten Informationen in 99 % der Fälle korrekt erkannt. Daraus folgt, dass die neue Methode verlässlicher sein muss als die aktuelle und bestenfalls auch weniger Zeit in Anspruch nimmt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

TUMexam assists during the preparation, execution, correction as well as review of anonymized digital exams. The exam preparation includes the creation of seating lists for the different lecture halls the exam is written in. By using stickers on the coverpage, the exam is anonymized and the student cannot be identified without looking up the registration number.

This solution has several advantages over a conventional exam such as automated scanning and cross evaluation. It is also possible for students as well as examiners to review one's exam online. Using TUMexam, each page has a pagecode which can be used during scanning to associate the different pages to a specific exam and the respective student. These features allow to correct the exam anonymously while still being able to map it to the individual student afterwards. Furthermore, crosses used to grade each problem as well as the solutions to multiple choice problems are evaluated automatically.

Currently, attendance control is implemented using stickers which need to be put on the exam and signed by the student on the attendance list during examination. These stickers need to be printed in advance and contain all the information necessary to link the student to the exam.

The sticker-based attendance control mechanism can be improved by a digitalized solution. It is important that the exam can be matched reliably to the correct student while taking into account human errors. To achieve this, we add a registration number box and a placeholder for the student card to the coverpage. The only personalized information on the coversheet, the registration number, can be extracted from the registration number box. It identifies the student afterwards. The placeholder is necessary to ensure that the student correctly entered the registration number. Figure 1.1 visualizes how

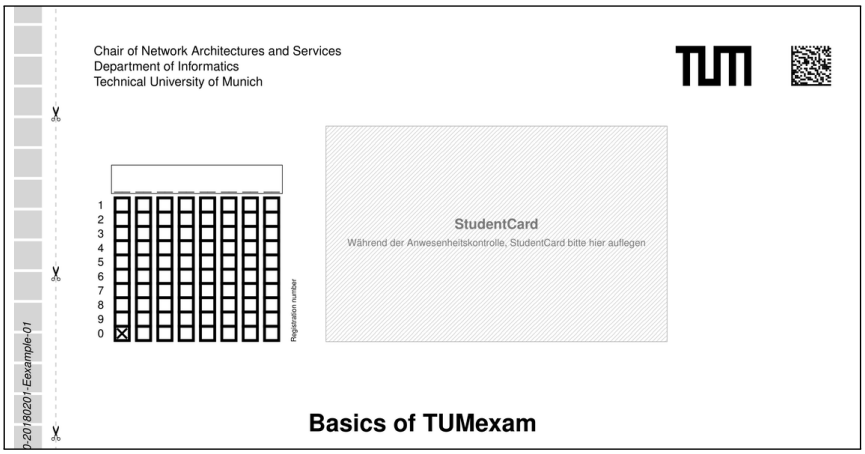

Figure 1.1: A template for the exam coverpage with a registration number box on the left and a student card placeholder on the right

the coverpage could look. By taking a photo of the exam's coverpage, it is possible to identify both objects and extract the registration numbers from both of them. Verifying that both numbers are identical ensure that the connection between the student and the correct exam can be established afterwards.

In this thesis, the future solution is tested for feasibility by creating a proof of concept. This program is able to extract the registration numbers from the image. Chapter 2 contains the relevant background information. Work which is related to this thesis is summarized in Chapter 3. The different approaches used to implement the future solution are presented in Chapter 4. Chapter 5 explains how the necessary images were aquired. The results from each of the different implementations are presented in Chapter 6. Chapter 7 contains information about how the registration number is extracted from a cropped image. The approaches are evaluated in Chapter 8. A comparison of the different technologies as well as future work is shown in Chapter 9.

Chapter 2

Background

This chapter explains the different techniques which are used during this thesis and how they work. Neural networks are explained in Section 2.1. The theory behind edge detection is covered in Section 2.2.

2.1 Neural Networks

As stated by Francis in [3], neural networks are designed to incorporate key features of neurons in the brain and process data in a similar manner. The purpose of neural networks is to perform various tasks which they are trained for. Examples for tasks involving images include recognizing handwritten digits, categorizing images as well as detecting objects within images. Given the example of identifying handwritten digits, the programmer previously had to come up with an algorithm can classify numbers reliably. The developer would also have to take into account that the program must cope with the many different ways a single digit can be written. By using those networks, figuring out the correct answer does not need to be done by the programmer anymore. During training, the network itself identifies the features which are necessary to precisely get the correct solution.

Neural networks consist of neurons (nodes) and connections (edges) which connect two nodes. Nodes are grouped into layers and are connected to nodes in the predecessing and successing layers. Edges have a weight which is multiplied to the value when it is passed from one node to another. Data can be fed into input layers and extracted from output layers. All other layers are known as hidden layers.

A very simple graphical example for explanation purposes can be seen in Figure 2.1. In this network the input data consists of the three values I1, I2, and I3, and the output data consists of the two values O1 and O2. Labels of some edges are left out for brevity. This adds up to seven nodes and ten edges which are part of this network. The neural networks used in this thesis are many magnitudes more complex.

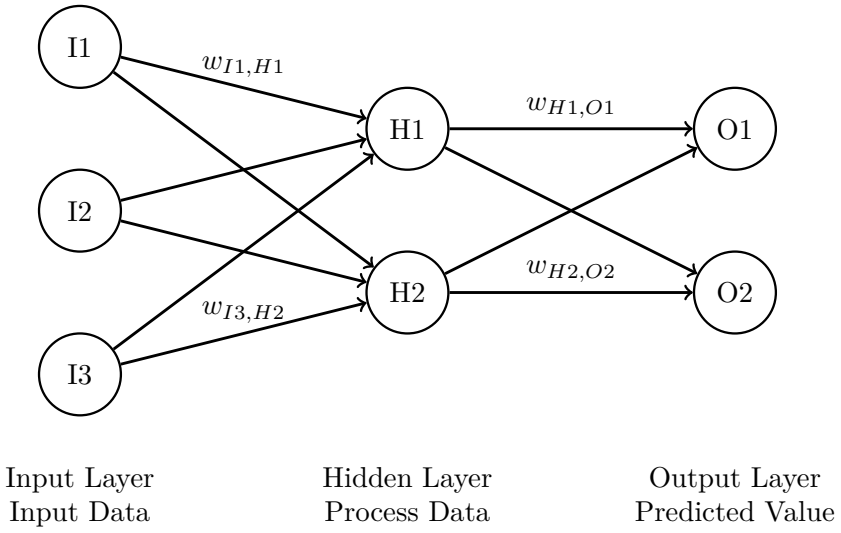

Figure 2.1: A general schema for a neural network with one hidden layer

The values of the nodes in the input layer are the values of the input data. Each succeessing node's value is computed by summing up all the connected nodes multiplied by the connections' weights. This correlation can be expressed as a series of matrix multiplications. The values for the hidden layer are calculated as shown in (2.1) and used in (2.2) in order to get the output layer's values. Training is used to adjust the weights so that the neural network reliably returns the expected values.

$$\begin{pmatrix} H1 \\ H2 \end{pmatrix} = \begin{pmatrix} w_{I1,H1} & w_{I2,H1} & w_{I3,H1} \\ w_{I1,H2} & w_{I2,H2} & w_{I3,H2} \end{pmatrix} \cdot \begin{pmatrix} I1 \\ I2 \\ I3 \end{pmatrix} \tag{2.1}$$

$$\begin{pmatrix} O1 \\ O2 \end{pmatrix} = \begin{pmatrix} w_{H1,O1} & w_{H2,O1} \\ w_{H1,O2} & w_{H2,O2} \end{pmatrix} \cdot \begin{pmatrix} H1 \\ H2 \end{pmatrix} \tag{2.2}$$

In order to train a neural network, two sufficiently large datasets are needed. The datasets map the input data fed into the network to the expected output. The first mapping is used to actually train the neural network and contains almost all information. The remaining information is provided by the second dataset, which is used to check

the accuracy on unknown input data. To ensure that the validation is performed with new inputs, it is absolutly necessary that those two datasets are distinct.

By feeding data into the network the actual output can be observed. The deviation from the expected output is calculated by a loss function. It is used by the optimizer to adjust the weights within the net. These adjustments aim to decrease the loss and therefore increase the probabilty of predicting the correct output. By optimizing for an extended period of time, the outputs should get more precise until a sufficiently high accuracy is achieved.

After a large amount of training iterations have passed, the validation dataset is used to check how well the neural network performs on unknown data. A lower loss on validation data implies a higher accuracy.

A simple real-world application for neural networks is the detection of handwritten digits. The inputs are grayscale 28×28 images. The outputs are probabilities for each of the digits zero to nine. The highest probability corresponds to the digits predicted to be in the image. MNIST has a large database with 70,000 images which can be used to train the network as provided by Y. LeCun, C. Cortes, and C. J. Burges in [7]. A very simple network which is capable to solve this task is explained in the TensorFlow Tutorial at [1]. It consists of three sequential layers which are made up of 784, 128 and ten nodes each. The networks consists of a total of 922 nodes and 101632 edges because the nodes are fully meshed together. This network is already far bigger that the schema presented above while still being very small compared to neural networks using within this thesis.

The Faster R-CNN is a region-based convolutional neural network as explained by Ren et al. in [11]. Compared to this neural network, the previously explained ones compute their prediction based on the whole supplied image. R-CNNs propose many different areas within the image on which the predictions are performed. Furthermore instead of returning a single result such as the single identified digit, these neural networks are able to identify more than one object in the image. For each object, the rectangular bounding box containing the object in the source image is retrieved as well.

The inputs for the neural network are images and the outputs are bounding boxes with their corresponding classes. The images can be of any size as they are resized automatically before being passed into the network. This is done to ensure that the size of the photo matches the amount of nodes in the first layer. Bounding boxes are defined using four integers `x, y, width, height`. The first two values denote its top left corner's coordinate, the other two values represent the width and height. Since only those four values are used, the denoted rectangle is always aligned on both the `x` and

the y axis. This means that the rectangle cannot be rotated to better fit the identified object. The returned classes, which are integers, are associated with the different objects the network is able to detect.

Mask R-CNNs extend the existing Faster R-CNN by additionally outputting a mask together with each bounding box and class as described by He et al. in [5]. This mask contains all pixels in the image which are inside the bounding box and part of the identified object. From those pixels it should be possible to retrieve further information such as the orientation which is not preserved in simple bounding boxes.

2.2 Edge Detection

The purpose of edge detection is to recognize objects in an image by identifying regions enclosed within adjacent edges. This can be done by grouping them into contours which are further processed and filtered based on their individual properties such as width and height.

This technique detects edges based on the gradients in the image as explained by the OpenCV Development Team in [2] and is now summarized. A lower and an upper threshold are required which determine whether or not an edge is included in the result. After filtering out noise by blurring the image, the intensity and directions of the gradients within the image are calculated. From all the pixels which could make up the resulting edges, only pixels with an intensity greather than the lower threshold are preserved. Pixels are directly accepted as an edge if their intensity is additionally greather than the upper threshold. The remaining pixels are only considered to be part of an edge, if they are connected to an already accepted edge otherwise they are discarded.

Compared to using a neural network backed object detection, edge detection has the advantage of detecting objects by their edges and properties, such as their aspect ratio, whereas networks can only detect objects it is trained on.

Chapter 3

Related Work

This chapter presents existing state-of-the-art research used for the recognition and extraction of necessary information from images by means of commercial products. Section 3.1 covers the results from current research. The different currently existing commercial products are presented in Section 3.2. At last, Section 3.3 presents other related theses written at the Chair of Network Architectures and Services at TUM enhancing TUMexam.

3.1 Current Research

The topic of this thesis, extraction of id cards and the included information, has been covered in literature before. The extraction of information within id cards is addressed in multiple United States Patents: *Government ID Card Validation Systems* by Jain, Fox, and Boyle in [6] and *Method and System for Recognizing Information on a Card* by Li and Chen in [8].

The first patent explains how id cards can be recognized using neural networks on a mobile device by analyzing features. The patent's purpose is to only transmit good images to the remote server by filtering them locally in order to reduce the required computational effort. This allows to prevent transmission of images which cannot be processed, such as blurry images. The neural network's training is performed on images that were previously either accepted or rejected by the remote server.

The second patent covers an approach to detect and extract information from id cards. Selected regions from which text should be extracted need to be known beforehand. Each region is split into one or more character areas containing containing a single

letter or digit. Each area is deblurred and then analyzed for multiple images. Once the detection succeeds, the detected value is preserved. Recognition works by identifing each character and joining the results to get the whole text for all areas.

3.2 Commercial Products

This section covers commercial products achieving similar things our thesis aims to do. It also justifies why they cannot be used to reach our goal. Section 3.2.1 explains in detail what Google Lens [4] can do. Office Lens [9] by Microsoft is covered in Section 3.2.2.

3.2.1 Google Lens

Google Lens is a mobile application using artificial intelligence, which is available since 2017. It is developed by Google, and aimed at extracting information from data sources such as images or a camera streams in realtime. On the one hand, it is used to identify and decode texts in images with support for foreign languages such as Chinese. On the other hand, it is capable of recognizing known objects in a picture such as monuments and also provide additional related information.

One of the main advantages of Google Lens is the decoding speed and accuracy of the text found in an image even under bad conditions, which is an essential requirement as neither the images taken nor the student cards can be ensured to be in perfect condition.

Nonetheless, the use of Google Lens also comes with disadvantages. The major disadvantage is that the program requires a steady network connection to provide its services. This also rises privacy concerns since these images are handled by a closed-source application and they are not processed locally on the device but need to be transferred to Google. Furthermore, the necessary actions, such as evaluating the registration number box or identifying and extracting the registration number from the student card, are not supported by the application. Another major drawback is the missing application programming interface (API) making it difficult to interact with the application.

When taking these limitations into consideration, this product is not suitable to the very problem addressed in this thesis.

3.2.2 Office Lens

Office Lens is another application from Microsoft aimed at image recognition. It is available for Windows, Android, and iOS operating systems. Compared to Google Lens, the purpose of Office Lens is to provide image recognition for document scans. As such, the main goals are to identify the document in the image and process it so that

the resulting image only contains the identified document. Additionally, it is possible to select different presets, such as “whiteboard” or “business card”, which affect the filters applied on the resulting image.

To extract the identified document from the image, several steps are performed in order to obtain the desired result. The first step is to deskew the image. This step is necessary because images taken with a smartphone can be skewed. The next steps are to rotate and crop the image. All these steps can be implemented using a perspective transform which is based on matrix calculations.

One of Office Lenses main advantages is that since the processing is performed on the device, no internet connection is required. Furthermore if the document is identified correctly, the resulting image can be deskewed and cropped so that only the actual document is contained in the output.

Nonetheless, Office Lens also has certain limitations. One of them is that the application does not support text recognition, which is a major drawback as we are required to extract that information from the document. Similar to Google Lens, it is not possible to interface with the application from code due to missing API, which limits its usefulness in our use case.

3.3 Related Theses

There are multiple theses at the Chair of Network Architectures and Services at TUM that also cover the attendee control digitalization. In this thesis, we extract the registration number box and the student card from the exam’s coverpage. We then identify and extract the registration number from the registration number box. Further work is implemented by the following theses: “TUMexam: Digitalizing Attendee Control for Examinations” by Nissen in [10] and “TUMexam: Digitalizing Attendee Control for Examinations – Image and Text Recognition on Mobile Devices” by Schrimper in [12].

The first thesis by Nissen in [10] covers the detection and extraction of the registration number from the cropped student card that is provided by the tool implemented in this thesis. Within the scope of the thesis, the different approaches to reliably find the registration number in the image and identifying the eight digits from the registration number are evaluated and compared. A major challenge is the physical condition of the text printed on the student card which might have been subject to damage. Due to the damaged digits on the student card, degraded data is included in the image which leads to an increased complexity for the reliable detection of the registration number. The missing information makes it difficult to correctly recognize the number.

The second thesis by Schrimper in [12] compares and evaluates alternative implementations to detect and extract the information from images on mobile devices using different methods. This covers in particular the detection of the registration number box as well as the student card on a mobile device. Furthermore, the extraction of the registration number encoded in the registration number box as well as the one printed on the student card is presented.

Chapter 4

Approaches

This chapter covers the different approaches used to extract the registration number box and the student card from the TUMexam coverpage. Section 4.1 describes the different approaches making use of neural networks. A completly different approach, edge detection, is described in Section 4.2. Section 4.3 covers approaches which focus on the identification of markers instead of directly identifying the different objects. After the markers have been found, the objects can be extracted because their location relative to the markers is known.

4.1 Neural Networks

The neural networks used in this thesis are trained to identify the objects contained in the supplied images. The recognition is performed using features the networks have determined on their own and as such it is not necessary to implement the detection on our own. In Section 4.1.1, the detection using the Faster R-CNN is explained. Section 4.1.2 covers the detection using the Mask R-CNN. The approach of extending the Faster R-CNN is introduced in Section 4.1.3.

4.1.1 Faster R-CNN

The Faster R-CNN is a region-based convolutional neural network as explained by Ren et al. in [11]. Its inputs are images and its outputs are bounding boxes, a class, and a probability for each object found. Before being able to train, the bounding boxes and associated classes for all objects in the images need to be aquired. The bounding box is the smallest possible rectangle which fully surrounds the object. It is made up of four integers: `x, y, width, height`. The first two numbers correspond to the minimum

`x` and `y` coordinate the object is located on. The other two values are determined by the difference between the maximum and minimum value for each axis. As a matter of such, the exact position of the object within the rectangle is unknown.

4.1.2 Mask R-CNN

Mask R-CNNs extend the existing Faster R-CNN by additionally outputting a mask together with each bounding box and class as described by He et al. in [5]. Images are passed into the Mask R-CNN and the outputs are the results from the Faster R-CNN as well as a mask extracted per object. The mask marks all pixels which are associated with the identified object and must be available for the training. The bounding boxes which are also necessary for the training can be dynamically calculated from the masks as they directly correspond to the objects location. As all types of objects we want to extract are rectangular, collecting the four corner points for each object is sufficient. Connecting these four points yields the bounding box. The area surrounded by the corner points is equal to the mask. Using this knowledge it is now possible to dynamically generate the correct bounding boxes and masks just from the previously aquired cornerpoints. The masks returned from the neural network could be used to get the objects' exact position which is in turn used to extract it. By having the position, there is no additional step needed to identify the object inside the bounding box output by the Faster R-CNN. While the object is identified immediately, its orientation is still unknown.

4.1.3 Extending the Faster R-CNN

As previously explained, the Faster R-CNN returns bounding boxes for each object. This results consists of four values which define the area of the image in which the object is located. The values correspond to `x`, `y`, `width` and `height`. We want to alter this neural network so that it returns the four corners of our rectangular objects instead of an approximate bounding box. For this to work, the neural network needs to return four coordinates consisting of an `x` and `y` value resulting in a total of eight values. By extending the neural network according to the new requirements, it should be possible to get the exact locations for each object. This will be used to not only detect the position of the objects of interest but also to extract the objects without the need for additional processing steps. As the resulting values are ordered, we can identify which values correspond to which edge. This allows us to correctly rotate the object, regardless of how the object is rotated in the image.

Our goal could be achieved by making some adjustments to the code creating the neural network's model: we need to rewrite methods that implicitely expect the current four values to support our eight values. Affected by this changes is for example the method

`FasterRcnnBoxCoder._encode()` as well as the decoding counterpart. Furthermore, some methods such as `FasterRcnnBoxCoder.code_size()` return the amount of expected values (four) which we adjust as well. Performing these changes througout the source code should make this possible. The necessary training data is already available from the data we aquired for the Mask R-CNNs training.

4.2 Edge Detection

In the general procedure, the extracted edges are used to identify contours. Contours are defined as one ore more connected edges. If they do not match given criteria (minimum width / height or relative size) are filtered and not processed in the rest of the evaluation. From the contours, the lines are extracted. If there are multiple lines nearby which only differ slightly from each other, they are replaced by a line being the average of the lines. Boxes are retrieved by intersecting the results with each other. Each set of overlapping boxes is reduced to a single contour, thus eliminating duplicates. The resulting boxes need to be mapped to the objects of interest. We comparing key attributes such as the aspect ratio to get this association. Not every box necessarily represents an object and there might also be another box fitting better that is not selected.

We make use of multiple approaches to solve the problem above. The necessary preprocessing steps which are always needed before edge detection can be applied are described in Section 4.2.1. Section 4.2.2 covers the general edge detection algorithm. Section 4.2.3 explains how detection works if we limit an object to a single contour. A different approach, removing this limitation by intersecting individual lines, is shown in Section 4.2.4. Section 4.2.5 presents the last approach which identifies the boxes after the image has been rotated.

4.2.1 Preprocessing the Image

Preprocessing is necessary to ensure the objects are detected in every image regardless of varying factors such as the student card color or the lighting conditions. As each image is different, the preprocessing must cope with all possible inputs. Depending on the image, some processing steps might not yield a great improvement but might very well be needed in another image. Figure 4.1 visualizes the effect the preprocesing steps have on the resulting edges.

Histogram Equalization: Contrast-limited adaptive histogram equalization (CLAHE) increases the contrast of the image. It aims to increase the intensity of shadows since white student cards are hard to detect on an white exam. The lines surrounding the

Figure 4.1: Comparison of detected edges of a selected image without and with preprocessing on the left and right, respectively

student cards are therefore more present. The effect histogram equalization has can be seen in Figure 4.2.

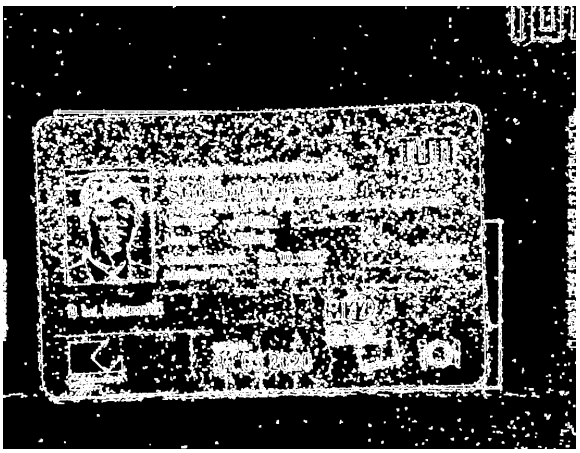

Figure 4.2: Detected edges after applying histogram equalization

Blurring: Blurring removes noise from the image which would otherwise result in unnecessary edges. Prominent edges are preserved as the huge color difference remains present after this step. Less prominent edges and noise are eliminated because the smaller color difference drops below the edge detection threshold. The outcome from applying blurring is visualized in Figure 4.3.

Bilateral Filtering: Bilateral filtering, similar to blurring, aims to reduce unwanted edges from the image. It is a more complex filter which blurs the image while preserving edges and therefore reducing the noise while not getting rid of the edges we want. Figure 4.4 highlights the difference bilateral filtering has on the output.

Figure 4.3: Detected edges of a selected image after additionally applying blurring

Figure 4.4: Detected edges of a selected image after further applying bilateral filtering

4.2.2 Performing Edge Detection

The process of edge detection and its postprocessing consists of seveal steps. Each of those serves a different purpose with the common goal to clearly identify the objects afterwards. The main goals of the postprocessing are to remove unwanted contours and precisely get each object's contours. Unwanted contours do not provide any benefit towards the common goal. Additionally, more contours relate to more computational effort and a longer execution time and is to be avoided. The exact identification is necessary so that the images resulting from the extraction procedure can then be used to further extract or evaluate the registration number contained or encoded within the results.

Identify and Filter Contours: The first step is to identify the contours and filter them given their properties. We want to exclude contours which do not represent relevant

information so that the amount of unnecessary comparisons and other operations on the data is reduced to a minimum.

During identification, only outer contours are selected and contours inside of them are dismissed. We achieve this by looking at the hierarchy returned by OpenCV's `findContours()`, which contains information about the contour surrounding every contour. If no parent contour is identified we want to preserve the contours as this exactly represents the kind of contours we are looking for. This reduces the total number of contours and therefore decreases the duration of the following operations. The difference is presented in Figure 4.5.

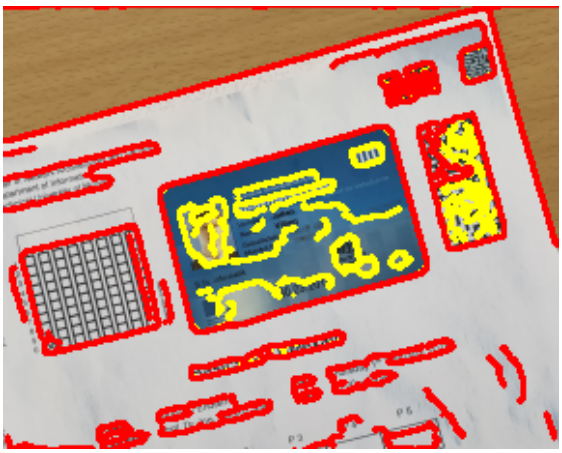

Figure 4.5: Comparison of extracted contours of a selected image with and without inner contours

The removal of inner contours can be clearly seen in the student card. This contour does not have any other contours inside because they are enclosed by outer contours surrounding the card. These inner contours are removed since they just make the extraction harder without providing any useful information.

The next step is to remove all contours which cannot represent our objects. To do so, we filter the extracted contours and only retain the contours which cover a minimum area and also have a minimum width and height. The latter constaint eliminates single lines which are long enough so that they encloses more space than specified by the minimum area threshold. The necessary raw data is provided by OpenCV's `minAreaRect()`. Figure 4.6 shows the remaining contours after filtering them.

The last step performed on the contours is to reduce them by only retaining the largest contour for each set of overlapping ones. It is under some conditions possible that some inner contours are not remove by the previous steps. This reduction is performed in two parts. In the first part, overlapping contours are identified and each contours area is stored. We get this information using OpenCV's `drawContours()` and `countNonZero()`

Figure 4.6: Extracted contours of a selected image after filtering

methods. The second part consists of keeping all contours that do not overlap and the largest contour for each group of contours that do overlap.

Calculate Average and Group Lines: Using OpenCV's `HoughLinesP()`, we get the lines for each contour. We do this in a way so that the lines are still associated with the contour they were extracted from. At first, we compute the average of nearby lines which have nearly the same angle because the lines returned by OpenCV might not be continuous. To do so, we look at all pairs of lines and calculate their distance. The distance is derived from the intersections of a line orthogonal to the others. If we find two or more lines which are nearby and have nearly the same angle, we compute the average line from each of the similar lines' angle and center. Figure 4.7 shows the result of this operation.

Figure 4.7: Comparison of the lines of a selected image before and after averaging

Now, we need to combine the lines in groups of four. The intersections of these lines within a group form boxes which are used to detect the objects later on. This is done by finding the parallel and orthogonal lines with the greatest distance. We can do this

because we know that the lines were extracted from a singe contour and are therefore all related. If we intersect those lines, we can create a box by connecting the intersections which should surround our object very precisely. The result from grouping the lines is presented in Figure 4.8.

Figure 4.8: Resulting lines and boxes of a selected image after grouping

4.2.3 Extracting Objects From a Single Contour

The first approach uses all the different sets of parameters for the edge detection to get all possible boxes. As a result, most of the contours found in the image are duplicated for each parameter set. For each group of overlapping boxes, the biggest box is chosen and the procedure is resumed normally.

An improvement is to only use as much sets of parameters as necessary. This should produce roughly the same output while being faster. In the worst case it should behave identically as not having used the approach in the first place. For each set of parameters, the procedure is executed the same way it is executed without the optimization until all objects of interest are identified.

4.2.4 Extracting Objects From Multiple Contours

After all lines have been identified by the procedure described above, they are joined together as if they all came from a single contour. Whereas in the previous approaches all lines are grouped by their respective contour, this is not the case in this approach. Removing this association enables lines from different contours to form a box later on which is important if an object's edges are split into multiple contours for any reason. From all overlapping boxes, the smallest one is interpreted as the correct one. This is necessary because there are larger rectangles enclosing the rectangles we want to identify as the correct ones. Selecting the largest contour would result in a single contour somewhat resembling a bounding box containing all lines which is not what we want.

4.2.5 Extracting Objects From the Rotated Image

We use `libdmtx` to find the position of data matrices within images. For this approach the rectangular matrix on the right of the student card in the correctly rotated image is used. If we supply an image to `libdmtx`, we get, amongst other information, the corner points for each data matrix found by the library. These corner points are listed in counter clockwise oriententation starting with the bottom right corner.

With these corner points, it is possible to calculate relevant properties of each matrix such as their width, height, and rotation. We can determine the aspect ratio of each matrix' width and height and use that information to find the only matrix on the exam's coverpage with a rectangular shape. Using this rotation, we can rotate the image and therefore realign the image.

After rotating the image, we know that the registration number box must be on the left side of the image and the student card on the right of the box. At this point the we execute the common edge detection process. We can limit the search to smaller areas of the image as we have rough know where the object is located.

4.3 Marker-Backed Detection

In contrast to the former approaches which attempt to identify the objects directly, marker-backed approaches focus on detecting distinctive features. In this case these features are markers which can be used to identify the objects we want. The advantage over other approaches is that while other approaches have to work solely with the features from the objects of interest, using this approach it is possible to choose characteristic aspects which are ideal for identification. Furthermore, we know where the objects locations relative to the known markers positions. The existing template needs to be altered to contain four bold markers as additional features. The adapted template can be seen in Figure 4.9.

Section 4.3.1 explains how template matching is achieved. Feature detection is demonstrated in Section 4.3.2. Section 4.3.3 shows how optical mark recognition is done.

4.3.1 Template Matching

Template matching works by moving the template image over the source image. This requires that we have an image containing the small markers which will be used as the template. In each possible combination, the difference between the area of the source image and the template image is calculated and stored. The template is found at the location where the difference is minimal. To prevent false positives, it is advised to filter

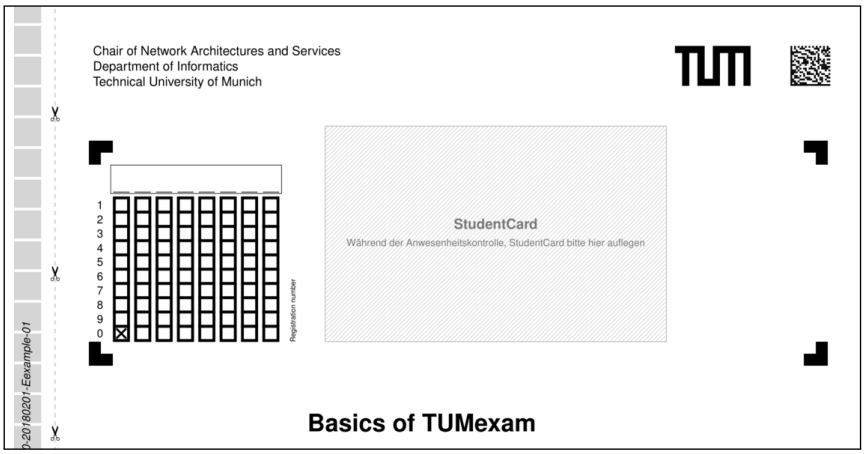

Figure 4.9: The adapted exam template from Figure 1.1 with four dominant markers

the stored deviations with a threshold ensuring that only locations with a reasonably low difference are taken into consideration.

Given the template image has the dimensions $w \times h$ and the source image has the dimensions $W \times H$, the resulting image has the dimensions $W - w + 1 \times H - h + 1$. Each coordinate (x_i, y_i) corresponds to the difference between the template image and the area in the source image within the rectangle defined by (x_i, y_i) and $(x_i + w, y_i + h)$. This algorithm is provided by OpenCVs `matchTemplate()`. A visualization of these probabilities for the top-right marker can be seen in Figure 4.10. The results for the other markers yield similar results but the markers in the other positions are identified. Executing this process for all four markers gives us the necessary information needed to realign the image. The pixels in Figure 4.10 correspond to the probability of a match at the pixels location. The lighter a pixel is, the higher is the probability and the greatest value is interpreted as a match. Together with the markers' dimensions, the markers' exact position can be determined. It is easy to see that there are only few positions for the marker to be as there are only few bright spots in the image.

Since this technique is based upon the difference between the template and areas within the source image, exact matches are absolutely necessary. Specifically, the size of the marker template must match the size of the marker in the image, and the image must not be rotated. To fulfill these preconditions, the image must be aligned and the markers must be scaled before template matching can work. We identify the rotation and size of the rectangular data matrix using `libdmtx` in a similar manner as explained in Section 4.2.5.

We can realign the input image as we know its rotation from the matrix' rotation. From the known ratio between the marker template size and matrix size, we can resize the

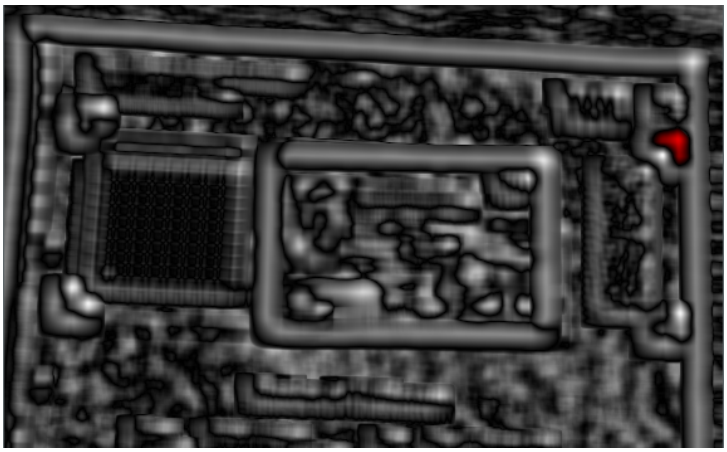

Figure 4.10: Visualized output from the template matching algorithm for the top-right marker with additional highlighting in red

markers so that they match the size in the image. After fixing the rotation and size difference, we can now use template matching to identify the markers.

Performing template matching for all four markers should give us four positions. Since we now know where the markers are located, we can create a box which exactly matches the area surrounded by the markers. We can now use a perspective transform to align the box to known coordinates and therefore deskew and crop the image accordingly. In the resulting picture, we can extract the objects using their known position relative to the markers. To retrieve the student cards, it is required to perform another step such as edge detection to get the exact position. A second step is required because it cannot be ensured that the student card is perfectly aligned on the paper. The extraction must be able to cope with this obstacle.

If some locations are missing and it is not possible to predict them or we do not want to predict them for some reason, we cannot proceed further. At this point we have to start over with different parameters or fall back to another approach.

4.3.2 Feature Detection

Instead of manually finding the markers we can use existing algorithms to automatically perform the detection for us. For this, we need a template of the whole image we want to have extracted and the image in which we want to find this template. This template contains the whole area of interest and not just the markers which is the case in the previous method. One such approach is feature detection.

For this approach we need two images: the template image we want to extract (not just the marker as it was needed for template matching, as well as the image in which

we want to find the template. We supply both images separately to a feature detector which then detects features in both images. The outputs from the feature detector are passed to the feature matcher which associates corresponding features.

By looking at all pairs of coordinates from all matching features, it is possible to calculate a homography. This is in turn used to compute a matrix for use in a perspective transform. With the matrix, we can get the rectangle surrounding our template in the image that we want to locate in the first place. The actual extraction is identical to the template matching approach described above, since the exact location of the target area is now known.

4.3.3 Optical Mark Recognition

Another possibility is to manually detect the markers using edge detection. Optical mark recognition aims to extract information by identifying known features from contours. With these features recognized, we can then perform operations such as deskewing, cropping or rotation.

In our case, we attempt to identify the markers. To achieve this, we first apply a threshold to the image to only get edges corresponding to dark contours in the image. After that, we filter the contours detected by edge detection on the thresholded image. In the first step, we remove contours which do not have an aspect ratio close to a square. We can do this, because we know that the marker resembles a rectangle with one fourth missing. In the next step, we remove contours which have less than approximately three thirds filled using the same knowledge. If we decide to change the template later on, it is required to adjust the filters according to the new properties of the markers in the template. At this point, we have reduced the contours so far that there are only very few contours left. A visualization is shown in Figure 4.11.

We group the remaining contours by their size to join the markers together and move the remaining smaller / larger ones into other groups. From these groups, we only consider groups containing four contours as we are looking for our four markers. We combine the contours by wrapping them in one large convex hull and get the corners of the box by intersecting the lines connecting the four contours. If there is more than one group, we use the group that encloses the biggest area. We can do this because we know that the markers must be represented by the contours which are the furthest apart. The result can be seen in Figure 4.12.

This box now surrounds the same region as the other two marker-backed approaches. To extract the objects, we need to apply a perspective transform to deskew and rotate the image so that it is aligned. From this image, we can extract the objects from their

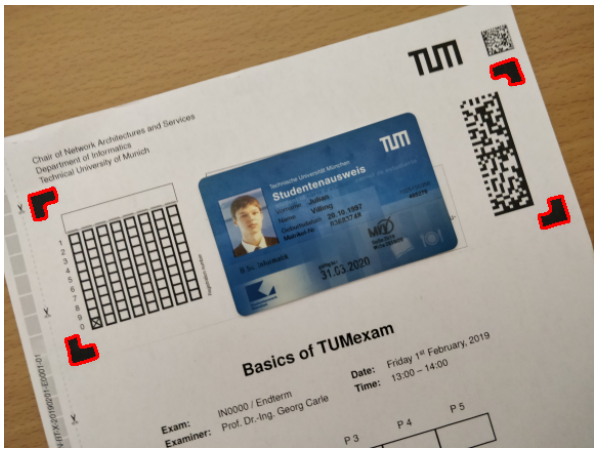

Figure 4.11: Marker contours detected with optical mark recognition

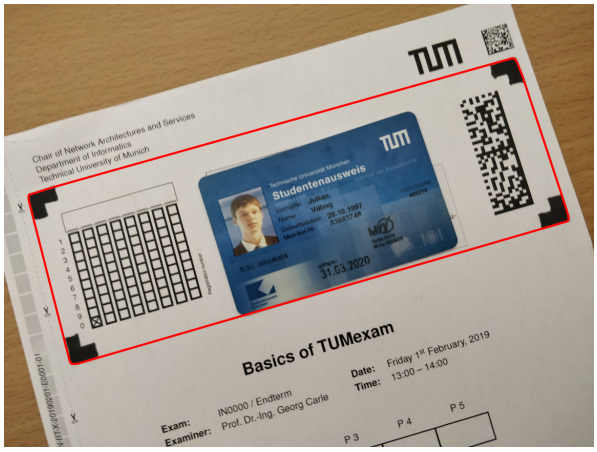

Figure 4.12: Box detected with optical mark recognition

known position. The registration number box is extracted using its known position in the aligned image. The student card is extracted using edge detection as thoroughly described in Section 4.2.

Chapter 5

Getting the Necessary Images

We need a large set of images in order to be able to successfully train the neural network and test the other algorithms. While we are able to take photos of the coverpage with our student card on our own, more data is needed for multiple reasons.

The first reason is that there are white and blue student cards, both of which are needed to ensure that the neural networks are able to recognize them correctly. Furthermore, the pictures are needed to ensure that edge detection still works if the student card's color is very similar to the coversheet's color, on which the card is placed. The second reason is that traces of use begin to show on the student card, which greatly reduce the readability of the registration number. This damage can come in different shapes. To ensure that the detection still works reliable on damaged student cards, various different student cards and digits are needed. Since the registration number is made up of eight digits, not all different digits can be present on a single student card. To still be able to ensure that the detection works for every digit, a large set of digits is necessary. We need them especially due to the great variations of their quality. With enough images we can test our tools and ensure that they are accurate and reliable.

As previously stated, taking many images of our own student cards is not sufficient. It is therefore necessary to gather a large amount of images containing the student card on top of the exam's coverpage. We organized a photo booth at TUM to aquire these necessary photos. We created a poster informing other students about our goal and the reasoning behind the photo booth. It also contains information about privacy as they provide sensitive information if they lend us their student cards. After printing many copies of the coversheet, we headed to the booth we organized and took photos and

videos. We took these photos from various angles and tilts to ensure our tools work with a lot of different images.

With these photos it is now possible to train our neural network as well as testing edge detection. After we switched to marker-based extraction approaches we organized another photo booth as the template used previously did not contains those new features.

The first booth yielded 553 images from which 29 were unusable because the images were blurry or cut-off. This left us with 524 usable images. In the second booth, we gained 52 videos with an average duration of 30 seconds. This translates to about 46800 images gained from those videos. The actual number of usable images is a bit lower as not all videos are usable because they do not contain our features or are blurry.

Chapter 6

Analysis and Results

In this chapter we analyze the different approaches. This includes whether they work and which challenges cause issues with a specific approach. Section 6.1 presents the results of neural networks. The success from using edge detection is documented in Section 6.2. Section 6.3 provides information about the output from using marker-backed detection.

6.1 Neural Networks

The outcome from using neural networks is presented in the following sections. The major disadvantage that neural networks have, is that they do try to solve a different problem than we have. While we want to get the exact position of our objects, the neural networks are engineered to identify which objects are present and where they are located. Therefore we need to adapt the outcome in such a way that it helps us to reach our goal. Section 6.1.1 covers the Faster R-CNN. The results from Mask R-CNN are shown in Section 6.1.2. Section 6.1.3 explains the outcome from extending the Faster R-CNN.

6.1.1 Faster R-CNN

The bounding boxes output from the Faster R-CNN have an accuracy of 99 %. This is a good result and the objects are identified very reliably. Unfortunately, there is no further information regarding the rotation or exact shape and position, which would be useful. As a result, using a Faster R-CNN for our goal is not useful as the precise detection still needs to be done. However, if we would just want to detect objects and their approximate location is sufficient, this approach would we ideal. Cropping the source image to a smaller, not 100 % accurate image is not viable given the amount of

time the evaluation takes. An examplary output is visualized in Figure 6.1 and more information is presented in Figure A.1.

Figure 6.1: Output from the Faster R-CNN

6.1.2 Mask R-CNN

The accuracy of the Mask R-CNN decreased to 95 % compared to the accuracy of the Faster R-CNN. This is only a slight worsening and still relatively good. Unfortunately, the bounding boxes output by the Mask R-CNN are not as accurate as the output from the underlying Faster R-CNN. The additional output, masks, are not very precise as well and limited to the area enclosed by the bounding boxes. This renders masks inaccurate even if they just miss some pixels located on the outside of the bounding box. More problems are caused by the fact that the masks do not form a rectangle with straight lines. This issues makes reconstructing the object's location much harder. The advantage of the masks containing information about the objects shape and position is therefore virtually nonexistent. Furthermore, the detection is very slow and therefore not useful in a potential mobile application. This issue is visualized in Figure 6.2, further information is shown in Figure A.2.

6.1.3 Extending the Faster R-CNN

To be able to fully understand the issues with this approach, magic numbers need to be introduced first. A magic number is a number whose purpose is unknown as the meaning cannot be determined from the context. An example for the missing context are compiler optimizations. When a program gets compiled, the compiler replaces constants defined in the code with their (fixed) value. After this optimization it is not possible to trace this number back to the constant. This is especially problematic if multiple constants have the same value as the value could have multiple origins. It is also possible that

Figure 6.2: Output from the Mask R-CNN

programmers do not use such variables in the first place and just directly inserts the numbers in their source code without giving an explanation. This results in the same problem, that the purpose of the number is unknown. While this is completely fine when executing the program, problems are caused when one wants to adjust the magic number. As the only remaining part is the number without any context, it is very difficult to impossible to determine its origin. We therefore cannot be sure that we are actually looking at the number we want to adjust. It might also be a completly different number which must not be changed.

Now we come back to the flaws this attempt has. First of all, the whole network is engineered and optimized in a way to output bounding boxes. The whole source code is adjusted to solve this very goal and it is not easily extendable. Secondly, the source code contains the previously explained magic numbers whose origin is not always deducible. This makes it very difficult for us to apply changes as we are never sure if we are changing the correct number. A similar problem occurs in methods which implicitly use the magic number four, such as expecting exactly four arguments or returning four values. The methods' assumption is not correct anymore and as such the code now contains errors caused by the falsification. Further problems arise from the size of the codebase. As it is not possible to search for a specific feature in the code, we have to identify the necessary changes manually.

Since the program was not designed for this use case, there are no precautions in place to help altering the neural network's structure. Possible measures which could have been taken include comments or annotations in the code. They would indicate which parts of the code need to be adjusted or where a magic number is used. As a result, it is not possible to reliably identify all pieces which need to be fixed. Furthermore,

it might also be impossible to resolve every issue without rewriting parts of the neural networks' strucute. This can be the case because only bounding boxes are calculated and necessary information about the objects' corners might be missing entirely.

6.2 Edge Detection

The results from edge detection are presented in the following sections. Generally, with an accuracy of about 70 %, the results are worse than the neural networks and more importantly, worse than the target accuracy of 99 %. This is related to the problem that it is not always possible to reliably determine the box corresponding to the objects. Nonetheless, these approaches still manage to extract a great percentage of the total images possible. Section 6.2.1 covers the approach which limits objects to a single contour. Extracting the objects from multiple contours is presented in Section 6.2.2. Section 6.2.3 explains the outcome of rotating the image before retrieving the objects.

6.2.1 Extracting Objects From a Single Contour

This approach has an approxmiate accuracy of 74 %, a runtime of 1 s and an even better performance in the improved version. Furthermore, the boxes are very precise and reliable given that the edges are detected correctly. This is a requirement as the relevant information is included only in these contours. Furthermore, no other contours are necessary for the object to be identified, since objects are limited to single contour. In case additional, incorrect, lines were detected inside the outer contour, only the lines on the outside are considered. We can do this because we know that this the only contour that encloses our object and the outer-most lines must enclose our object.

6.2.2 Extracting Objects From Multiple Contours

Compared to the previous approach, finding the smallest box did not provide consistent results. This is caused by the now missing relationships between lines and contours because all lines are merged as if they came from a big contour. A clear advantage of this approach is that objects can be found if the edge detection splits the objects' edges into two previously separate contours. This problem does occur occasionally. A major disadvantage is that much more potential boxes have to be checked. Furthermore it is possible that rectangles are incorrectly selected. Incorrect rectangles that better match the distinctive features of the objects can be selected instead of the correct ones'. Due to the missing relationship between lines and contours, it is not possible anymore to select the outer lines as resulting lines. The assumption used previously is that an object is identified by, and limited to, a single contour. Therefore the outer lines are not known to originate from the same object. To cope with the problem, the smallest box is chosen.

The new choice is problematic if a contour has inner lines as they incorrectly shrink the resulting box. The difference this rule makes and the new boxes created from that are presented in Figure 6.3 and more images are shown in Figure A.3.

Figure 6.3: Additional boxes after removal of the contour association, the yellow boxes represent the resulting box candidates

6.2.3 Extracting Objects From the Rotated Image

This approach is based upon the previous approach of finding objects from a single contour while reducing the area in which the objects are searched for. The simplification reduces the amount of contours which need to be compared. Additionally the approximate location of the object is known. A small disadvantage is that this approach requires the pagecode to be found using `libdmtx`. Otherwise we would need to fall back to the underlying approach without using rotation. Therefore, this approach is only a slight improvement from 74 % to 88 % as the chance of choosing the wrong box is only decreased and an additional step needs to be performed prior to detecting the objects.

6.3 Marker-Backed Detection

Marker-backed detection algorithms produced mixed results. Some approaches reached an accuracy close to 100 % while other approaches failed to get consistent results. The advantage of using markers is that it is not necessary to precisely detect the objects. Instead, the additional features can be used to gain the necessary information and perform the extraction. The following sections cover different methods to detect the markers. After finding them, they can be used to realign the image. In this resulting image, the position of our objects is known and they can be extracted. Template matchings' results are shown in Section 6.3.1. Section 6.3.2 covers the outcome from using feature detection. Information about the success of optical mark recognition can be found in Section 6.3.3.

6.3.1 Template Matching

This approach works by finding the position where the template matches the area of the image the most. The marker template must match with the marker found the source image. Specifically, they must have an identical size and rotation. This means that before template matching can be applied, the image must be rotated and the markers must be resized. These two preconditions can be fulfilled after finding the pagecode and using its location data to calculate the marker's target size and the image's rotation. The disadvantage of this approach is that is is required to find the pagecode. Otherwise it is not possible to apply template matching because we cannot ensure that the marker template match the markers in the image. While this approach works sufficiently with an accuracy of up to 97 %, requiring the pagecode to be found makes it very inflexible. The extraction procedure is visualited in Figure A.4 and identical for all marker-backed approaches. Another approch which works equally well while not having this limitation much more favorable.

6.3.2 Feature Detection

Feature detection works in two steps: in the first step, a feature detector detects features. This step needs to be performed on both images separately. In the second step, a feature matcher groups related features using the extracted information from both images. Using the identified correlation between both pictures, it should be possible to locate and extract the location of the template within the image. Unfortunately, the feature detection finds only few features which are not enough to guarantee that the template is found in the image at all times. Without more related features it is not possible to calculate the homography which is a prerequisite to extracting the template area from the image. As such, the approach fails to produce consistent results and since it is unreliable, it is not suitable to solve the problem. The problems of using this technique is shown in Figure 6.4. The green lines indicate the related keypoints in both images and the red box denotes where the template was identified. In this case, the result does not include the left markers. Additional boxes are visualized in Figure A.5 in the appendix to demonstrate the huge variations.

6.3.3 Optical Mark Recognition

This approach focuses on using edge detection to identify the markers' locations which in turn are used to extract the area enclosed by the markers. The detection and identification of the contours which correspond to the markers has an accuracy close to 100 %. After thresholding and filtering the contours, many of the inital contours are already eliminated. Grouping the contours and keeping the groups which contain four contours

Figure 6.4: Visualized matches and identified box after applying feature detection

further reduces the amount of contours. The correct markers are identified by selecting the largest area surrounded by the contours. It is then extracted as a fixed-size image in the same manner as for other marker-backed approaches. It is shown in Figure A.4.

Chapter 7

Registration Number Prediction

This chapter explains the features an image used for prediction must have and how the registration number is extracted from the input data. Section 7.1 explains how the neural network was trained and how the prediction works. The procedure necessary to retrieve the correct registration number from the registration number box is presented in Section 7.2.

7.1 Overview

The Chair of Network Architectures and Services has trained a neural network which is able to correctly predict whether a box is crossed or not. For each box it can either be not crossed, crossed, or filled. The latter is to be interpreted as not crossed in order to give students and examiners the possibility to correct errors they might have made. Prediction the state of all 80 boxes of a registration number box takes around 1.5 s on our laptop. Beside the time needed for evaluation, the time necessary to initialize the neural network is included as well.

Using this network, it is possible to determine which boxes in the registration number box are crossed, which is the only case we are interested in, and we can calculate the predicted registration number using this information. In case a student crossed the wrong box, we can detect this as the resulting number from the registration number box deviates from the student card's registration number. Additionally, we can detect whether a student ticked none or multiple boxes in a column and as such making the correct number ambiguous.

7.2 Prediction Procedure

To be able to reliably use the neural network, it is necessary that the registration number box is extracted very precisely. This implies that the box is rotated correctly. It is otherwise not possible to correctly cut out the crosses. We extract the crosses per column so that we have eight sets consisting of ten crosses, each corresponding to a digit of the registration number.

These are then passed to the neural network which predicts the state of the ten boxes. Having those boxes we identify all crossed boxes and determine which digit is selected. This only works if exactly one box is crossed. If zero or more than one boxes in the current column are crossed, an error value is returned.

This procedure is visualized in Figure 7.1. The columns in which only a single cross is detected are highlighted in green. Columns with multiple crosses are colored in yellow. If no box was detected as crossed at all, the respective column is colored red. These colors can be applied only after the detection is completed and are solely used for visualization purposes.

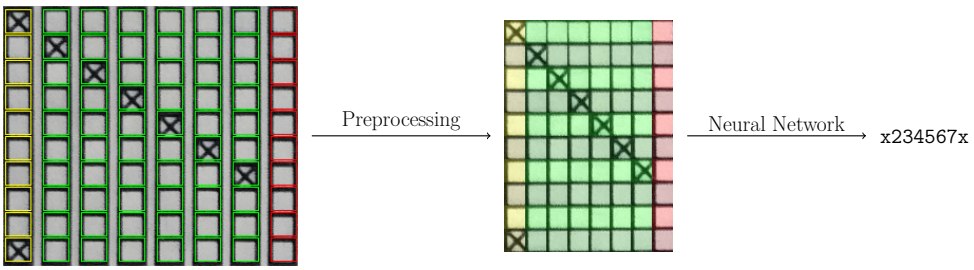

Figure 7.1: Visualization of the prediction procedure

Chapter 8

Evaluation

This chapter shows the evaluation of the techniques applied during the thesis. The results for the different approaches are presented and classified. Section 8.1 evaluates the images taken at various occasions. The accuracy of the different approaches is compared in Section 8.2. Section 8.3 presents the runtime of the different algorithms. We compare the approaches in Section 8.4 to determine which of these help us to reach our goal.

8.1 Images

The amount of images gained from taking photos at different occasions are presented here. They are split amongst the total and usable amount of images taken. An image is usable if it is not blurred and contains the necessary information. The results can be seen in Table 8.1. The ratio between images taken and good images is also added to the table as an insight of how many images were removed due to flaws and therefore cannot be used by the different approaches.

Image Batch	Total Image Count	Good Image Count	Ratio
First image batch	384	357	93.0 %
Photo booth 1, without a DM	280	251	89.6 %
Photo booth 1, with a DM	273	273	100 %
Photo booth 2	513	450	87.7 %

Table 8.1: The amount of images in the different batches

For the first set of images, we took photos on our own. The photos aquired at photo booth 1 are split into two disjoint sets. The first set lists the images which do not contain

a Data Matrix (DM) code. The second set lists the remaining images. The videos taken at photo booth 2 result in around 46800 images. The statistics are extrapolated from a smaller sample due to the huge amount of images.

8.2 Accuracy

The first metric is the accuracy of the different approaches which determines how well the approach works for the given dataset. For applicable approaches, the accuracy after supplying two images is evaluated as well. This is an option to increase the precision if the the execution time is low enough. This is the most important metric as it decides whether the approach is actually better than the current approach to handle attendee control which makes use of stickers.

Table 8.2 shows the accuracy of the neural networks and is separated intentionally. Whereas the accuracy for other approaches is computed by the amount of successful extractions, the neural networks compute this value themselves. It is an indicator how well the neural networks perform their task. As such the accuracy is not comparable to the other approaches' accuracies. The accuracies for the edge detection and marker-backed algorithms are presented in Table 8.3 and Table 8.4 for the registration number boxes and student cards, respectively.

Approach	Accuracy
Faster R-CNN	99 %
Mask R-CNN	95 %

Table 8.2: The accuracies of the different neural networks used

The accuracy for the extension of the Faster R-CNN is not present as it was not possible to get this approach to work. While the accuracies are very high, they do not correspond to the accuracies of our extraction. This is the case because the neural networks perform a slightly different task: they identify the region in which the objects are located which is the user case they were designed for. Accuracy information, which is necessary to perform the extraction is not available. As such, they are unfortunately not applicable to our scenario.

Feature detection and the second edge detection approach are missing from Table 8.3 and Table 8.4 for the same reason extending the Faster R-CNN is missing from Table 8.2. Neither of those approaches are able to produce (reliable) results, either because we are not able to get the approach to work or because the results are inconsistent.

Approach	One Image	Two Images
Edge Detection (Single Contour)	77 %	86 %
Edge Detection (Rotated)	88 %	97 %
Template Matching	97 %	99 %
Optical Mark Recognition	100 %	100 %

Table 8.3: The accuracy of the different approaches tested regarding the extracted registration number boxes

Approach	One Image	Two Images
Edge Detection (Single Contour)	70 %	77 %
Edge Detection (Rotated)	90 %	97 %
Template Matching	83 %	98 %
Optical Mark Recognition	90 %	99 %

Table 8.4: The accuracy of the different approaches tested regarding the extracted student cards

Compared to the other approaches, plain edge detection yields a relatively low accuracy but it can be improved by taking and processing another image. The neural networks yield a high accuracy but they do not provide the necessary information relevant for further extraction. Template matching and optical mark recognition deliver good results while requiring markers to be present in the image.

8.3 Runtime

The second relevant metric is the runtime, i. e., determining how long the image processing takes. This, together with the accuracy, determines whether processing a second image is actually useful. It can also give a rough idea which approaches could work on a mobile device as they are usually more limited in processing power than a fully-blown desktop computer. Furthermore, this metric, which is presented in Table 8.5, helps to determine if the approach is actually able to outperform the sticker-based approach it should improve on.

Taking our goal into account, a huge disadvantage of the neural networks used in this thesis is their runtime. While this might not be an issue when using them to solve other problems, in our case, in which the detection and evaluation of the registation number box and the student card is to be performend on a mobile device with usually limited computing capabilities, high-performance approaches are necessary.

Approach	Min. Runtime	Avg. Runtime	Max. Runtime
Faster R-CNN	3.76 s	4.23 s	5.65 s
Mask R-CNN	6.43 s	8.77 s	9.63 s
Edge Detection (Single Contour)	0.6 s	0.99 s	1.4 s
Edge Detection (Rotated)	0.22s	1.86s	8.64s
Template Matching	0.32 s	0.75 s	1.43 s
Optical Mark Recognition	0.29 s	0.4 s	0.7 s

Table 8.5: The runtime of the different approaches tested

The pure edge detection based approaches are faster but have a high worst-case runtime. From a time perspective, it is possible to evaluate a second photo in order to improve the accuracy. Unfortunately, the increased accuracy does not match the target accuracy of 99 %.

The best approaches regarding their speed are the marker-backed approaches: template matching and optical mark recognition.

8.4 Comparison of the Different Approaches

We can identify usable approaches by checking if they satisfy our needs. We compare the previously collected metrics accuracy and runtime and if they need the image to be rotated correctly. An approach suitable to our problem fulfills the accuracy requirement and should also be rotation-independent. A method with a good runtime has an advantage over the other approaches. The comparison is shown in Table 8.6.

Approach	Accuracy	Runtime	Rotation-Independent
Faster R-CNN	✓	×	✓
Mask R-CNN	✓	×	✓
Edge Detection (Single Contour)	×	✓	✓
Edge Detection (Rotated)	×	○	×
Template Matching	○	✓	×
Optical Mark Recognition	✓	✓	✓

Table 8.6: Comparison of the different metrices for our approaches

Our results show that the neural networks used in this thesis are not suitable since they do not solve our problem directly. Furthermore, they require too much time for the slight improvement they yield. We can also conclude that plain edge detection is not able to always extract our objects. Their accuracy, which is less than 99 %, does

therefore not fulfill this major requirement. Both marker-based approaches fulfill the runtime constraint. The accuracy constraint is satisfied by optical mark recognition and by template matching if two photos per student are taken. Unfortunately, template matching requires the image to be rotated and scaled correctly. As a result, optical mark recognition is the approach we use to reliably detect and extract the objects from the supplied images.

Chapter 9

Conclusion

This chapter summarizes the thesis and explains what needs to be done in the future. Our neural network approaches are recapped in Section 9.1. Section 9.2 lists the results from using edge detection and Section 9.3 covers the outcome from using marker-backed approaches. Future work and improvements are listed in Section 9.4. In Section 9.5 summarizes the important points of the thesis.

9.1 Using Neural Networks

Our first attempt to extract the student card and registration number box was to use neural networks to identify the objects in our image. At first we use the already existing Faster R-CNN. It returns a bounding box for each identified image together with a classification and a probability that the result is correct. This neural network did recognize the objects in the image reliably but it is neither fast nor has it a high precision. While this technique delivered good results in checking where the object is located, no information about its rotation, orientation, or exact position is available. As such, this approach does not have any benefit except shrinking down the search area for each object.

To resolve this problem, we used the Mask R-CNN, another existing neural network. This extends the previous network by additionally returning a mask which marks all pixels related to the current identified object within the bounding box. Adding this mask greatly increased the processing time. Unfortunately, these improvements to the neural network are not usable. The masks are very imprecise and do neither yield information on the object's orientation nor rotation. Therefore, the additional processing

only increases the runtime but does not help to solve the problem in this very case. As such, there is no problem-related improvement over the previous network.

Since both approaches did not solve our problem, we tried to rewrite the Faster R-CNN so that instead of a bounding box the four cornerpoints for each of our objects are returned. With these points we have all necessary information about the identified objects such as rotation, orientation and exact location. To achieve this goal, we edit the source code in the appropriate places according to the changes we want to make. For this, we need to rewrite methods which do not work together with the new requirements, and adjust values inside the source code. The difficult challenge is that it is not clear which parts of the code need to be adjusted and some parts are ambiguous. This technique did not work out since it was not possible to identify the necessary changes and to resolve this issue.

9.2 Using Edge Detection

Our first attempt failed, so we continued with edge detection. We start by detecting the contours in the image and after retrieving the lines for each contour, we calculate the boxes defined by the lines' intersections. From all overlapping boxes in the image, we take the largest box. Smaller boxes correspond to features inside the actual object such as the face inside the student card. We filter the resulting boxes by their aspect ratio and identify the different objects we want to extract according to that. Now, we extract the objects from the appropriate boxes and align the registration number box from its relative position to the student card's position. Using the width and height of the cropped student card, we rotate it such that the width of the resulting image is greater than the height. While we are able to extract the objects in about 77 of 100 images, we have no information about the student card's orientation and try to improve the accuracy with the following approach.

We adapt our edge detection algorithm by joining all the lines as if they came from a single contour, enabling us to identify objects which are distributed over multiple, not connected contours. We then combine the lines to create various boxes and attempt to find our objects. By merging all lines, we cannot select the largest box and have to fall back to filtering out all boxes which overlap with another, smaller box. As there are no overlapping boxes left, we continue the same way we did before by identifying the boxes corresponding to our objects based on their aspect ratio. This approach did not work out because the association between lines and the corresponding contour is necessary to reliably identify objects. It is not clear whether nearby lines are part of the same contour or if they are from two different contours which are located very close to each

other. For this reason, we are not able to clearly identify the correct boxes and this approach does not help solve the problem.

To improve the original edge detection approach, we align the image by rotating it prior to detecting edges according to the rotation of the pagecode found on the coversheet. We can now search for the object in a smaller area because we know that the registration number box is on the left of the correctly aligned image and the student card is located to the right of the registration number box. If we fail to identify the box in the first step, we can still look for the student card in the whole image. To be able to rotate the image, it is required that we are able to reliably identify the pagecode. After the rotation, the procedure is similar to the first edge detection approach except that the procedure is executed twice: first for the registration number box and after that for the student card, each operating on a cropped source image. The actual detection is identical to the first approach. The approach has an accuracy of up to 90 %. This is an improvement compared to the other approaches but it is unfortunately still not as good as the sticker-backed approach.

9.3 Using Marker-Backed Approaches

As it turns out, detecting and identifying the objects directly does not work reliably. To solve these problems we add new markers on the coversheet with the sole purpose to improve detection accuracy. The first marker-backed approach is template matching. It works by comparing a template, in our case a cropped image of the marker, with the image of the coversheet. By finding the position where the difference is the smallest, the position of the template is located in the image. For this detection to work, a minimum threshold must be passed to ensure the marker is actually identified. This approach requires that the template matches the size its contents have in the image. Additionally, the image must not be rotated and only slight skew can be compensated by this technique to work. As such, before we can perform template matching, we need to rotate and scale the image. We do the preparation with the information about size and angle we get from finding the pagecode in the image. Using the results from template matching, we can calculate the box enclosed by the markers and apply a perspective transform which aligns the box to the size of a known box. We know the position and size of the registration number box in this fixed-size image and can easily extract it. The student card is identified using the approximate area in which it must be located which we also know in advance. While this attempt yielded good results on registration number boxes, the accuracy of student card is only 83 %. Additionally it is necessary that the pagecode is found as template matching does not work otherwise.

To solve this limitation we test other approaches. The next approach is feature detection, which can identify corresponding features using our template image and the source image from the camera. At first, a feature detector detects keypoints on its own. A feature matcher identifies related keypoints afterwards. From many relationships between the two images it is possible to locate the template within the image and compute the transform needed to extract the relevant part of the image. Unfortunately, this technique did not produce reliable results and it is therefore not possible to extract our target image.

As the previous approach does not work, we try optical mark recognition. For this approach, we need to know about the unique properties of our markers to be able to differentiate them from the other contours returned from edge detection. The first step is to apply a threshold leaving only dark parts of the image, such as our markers, on which we apply edge detection and in turn detect the contours. The next step is to identify the markers we need by eliminating all the contours not corresponding to a marker. We filter them by the aspect ratio of their bounding rectangle as we know that the markers are enclosed by an approximate square. After that, we keep contours only if their area is greater than the minimum area threshold. This removes contours which do not cover an area similar to the markers, such as small lines. Now we group the contours which have approximately the same size and only keep groups which contain exactly four contours as this is the amount of markers we need construct the box. Because the markers are the features forming the largest rectangle and have a distinct aspect ratio, we filter the remaining groups so that we identify the group containing the markers we are searching for. We apply a perspective transformation to receive the deskewed and aligned image in which we already know the positions of the objects. This attempt has a high accuracy close to 100 % and a low duration and is therefore the approach we choose to solve our problem.

9.4 Future Work

After we created the proof of concept, the next big step is to create a mobile application for Android and iOS. It should be able to perform the same extraction the proof of concept in this thesis does. Specifically, it should work well on same input of pretaken photos. Furthermore, it is necessary to ensure that the implementation does not run into performance or memory bottlenecks. The following task would be to design an application able to extract the objects containing the necessary information in realtime. After that, the results from the other thesis “TUMexam: Digitalizing Attendee

Control for Examinations" by Nissen in [10] should be included. This would enable the application to additionally retrieve the registration number from the extracted objects.

The second big step is to identify and resolve potential legal issues. This includes the attendee control list that is not required anymore with our approach including the signature which previously was on said list. Further concerns may arise from the use of the examiners' personal phones.

The next part comes down to optimizing the application. This includes minimizing the marker size as well as using the smallest possible pagecode while maintaining the high detection rate. Additional changes may be incorporated into our tool. This can be a different coversheet layout for which the tools' detection and extraction parameters must be adjusted. Another reason for changing the tool could be more efficient or more accurate detection algorithms.

9.5 Summary

Using neural networks or edge detection approaches that directly identify and extract the objects in the image did not yield reliable results. Their accuracies cannot compete with the target accuracy accomplished by the sticker-based approach. Therefore these techniques are not applicable to our problem and are therefore not recommended.

Based on the good results from using optical mark recognition, this approach should be used to detect and extract the objects of interest from a photo taken. Not only does this approach have a high accuracy and precision, it also has a very low execution time compared to the other approaches.

Chapter A

Appendix

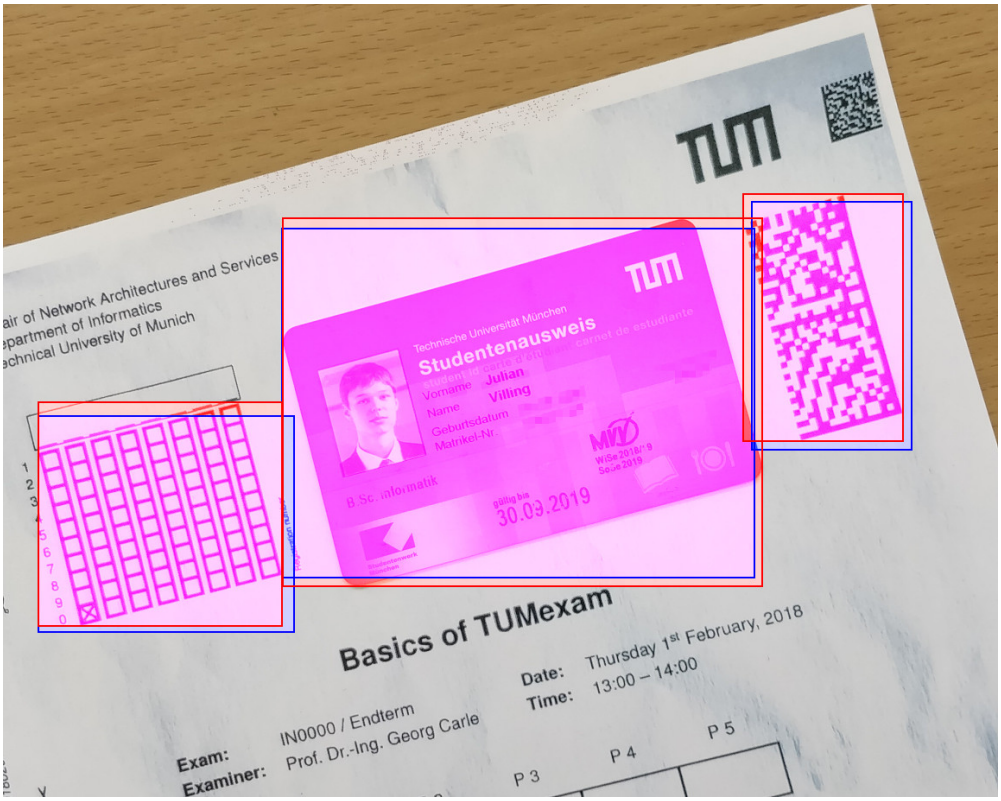

Figure A.1: Visualized deviation of the actual bounding boxes output by the Faster R-CNN from the expected results. The actual and expected areas are marked in blue and red, respectively. Overlapping areas are colored in magenta. The red background highlights areas which are not included in the actual bounding boxes but are expected to be covered.

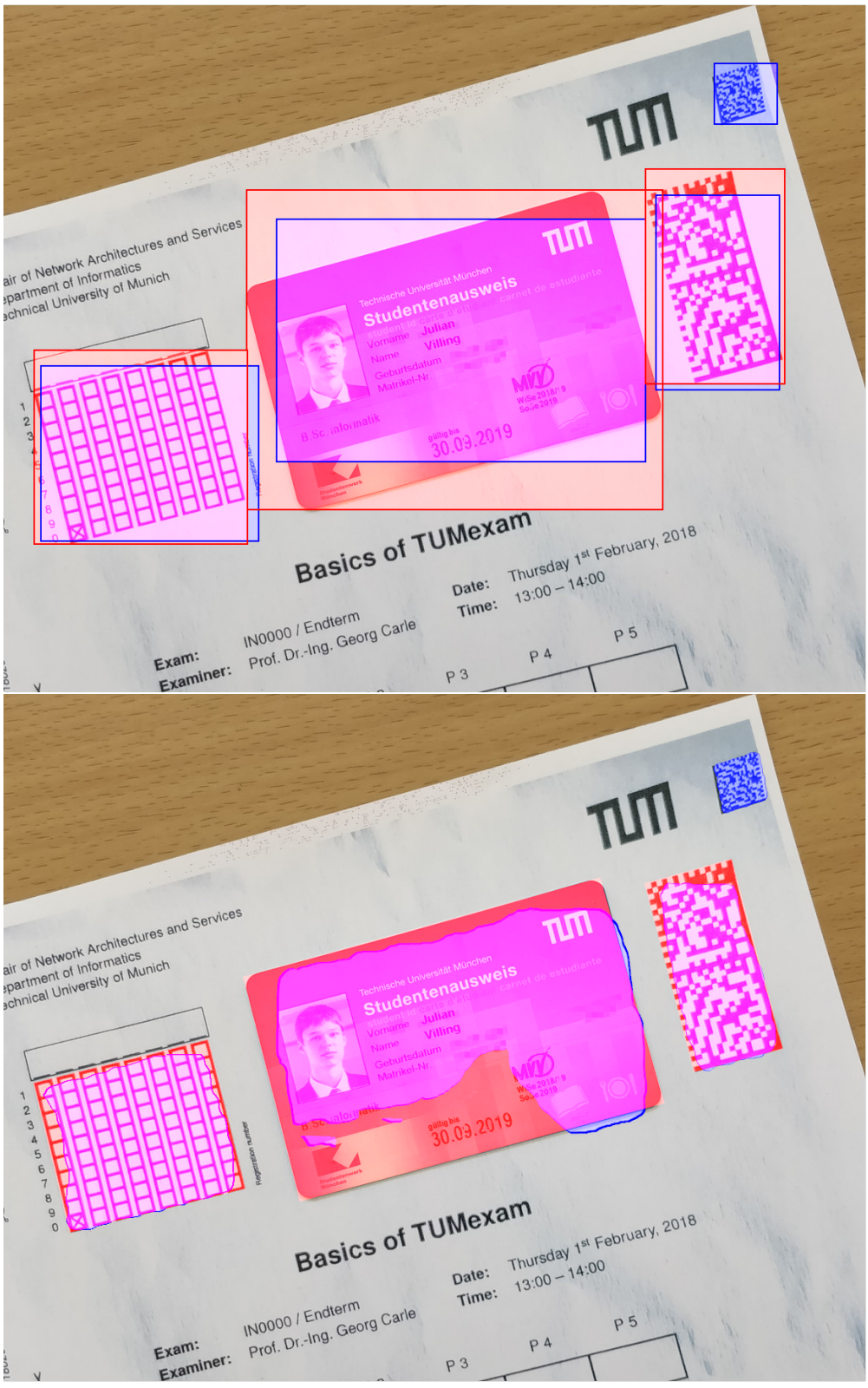

FIGURE A.2: Visualized deviation of the actual bounding boxes and masks outputted by the Mask R-CNN from the expected results. The boxes are shown at the top and the masks are visualized at the bottom. The actual and expected areas are marked in blue and red, respectively. Overlapping areas are colored in magenta. The read background highlights areas which are not included in the actual masks/bounding boxes but are expected to be covered.

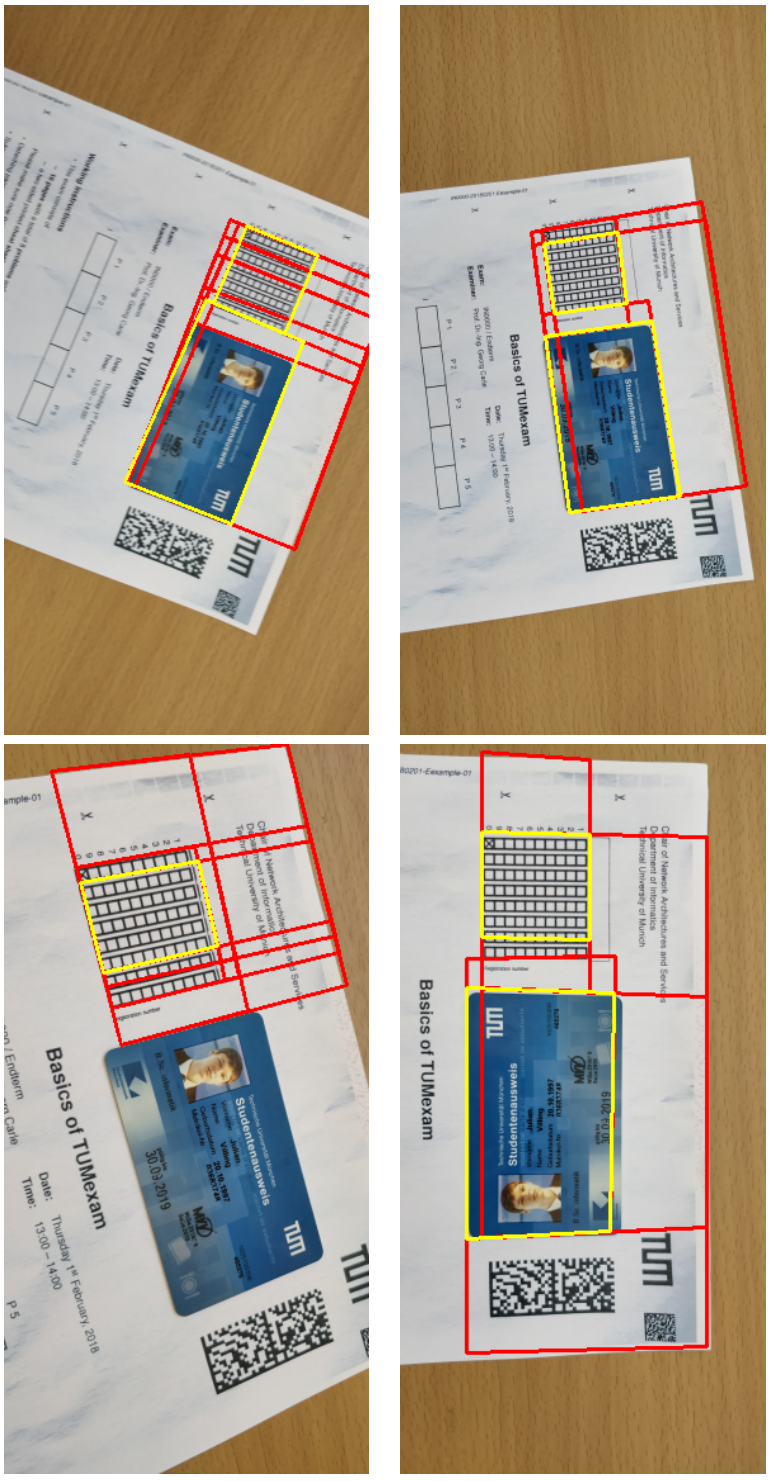

FIGURE A.3: More edge detection results after joining the lines highlighting the increased number of boxes as well as the problem of identifying the smallest one. The yellow box marks the resulting box candidates. Note how the boxes do not match the actual objects which we want to extract.

FIGURE A.4: Visualized results from the extraction using marker-backed approaches. The image at the top shows the regions from which the objects are extracted. The image at the bottom shows the objects after they have been extracted. The registration number box is extracted very precisely, the student card is extracted with a slight skew.

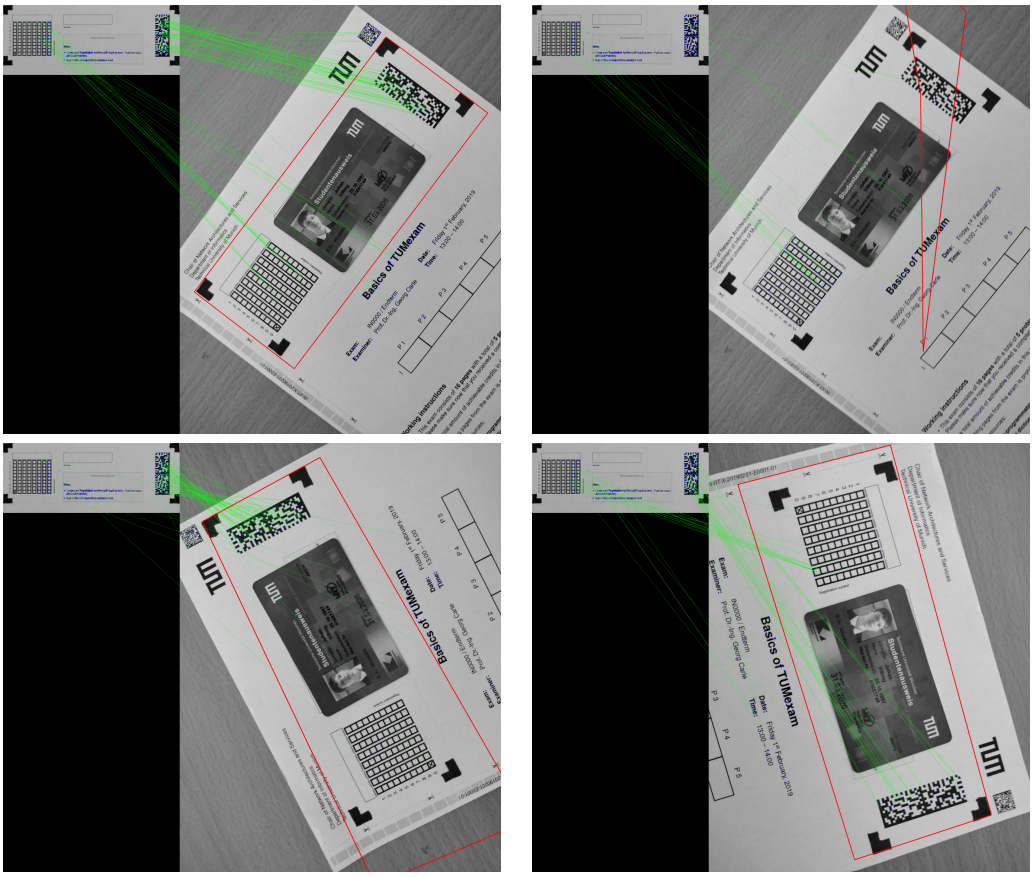

FIGURE A.5: More feature detection results demonstrating the huge variations. Note how the result is not always a rectangle. This makes it impossible to use this approach in an use case which requires precise results.

Bibliography

[1] *Basic Classification: Classify Images of Clothing*. URL: https://web.archive.org/web/20191017093049/https://www.tensorflow.org/tutorials/keras/classification (visited on 11/11/2019).

[2] *Canny Edge Detector – OpenCV 2.4.13.7 Documentation*. URL: http://web.archive.org/web/20190519161302/https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html (visited on 11/26/2019).

[3] Louis Francis. "The Basics of Neural Networks Demystified". In: *Contingencies* 11.12 (2001), pp. 56–61.

[4] *Google Lens*. URL: https://web.archive.org/web/20190728015825/https://lens.google.com/ (visited on 12/09/2019).

[5] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[6] Chintan Jain, Ryan Fox, and Monika Valiramani Boyle. *Government ID Card Validation Systems*. US Patent 10,242,283. Mar. 2019.

[7] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST Handwritten Digit Database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[8] Yang Li and Guo Chen. *Method and System for Recognizing Information on a Card*. US Patent 10,210,415. Feb. 2019.

[9] *Microsoft Office Lens*. URL: http://web.archive.org/web/20191117161504/https://www.microsoft.com/en-us/p/office-lens/9wzdncrfj3t8 (visited on 12/09/2019).

[10] Leon Nissen. "TUMexam: Digitalizing Attendee Control for Examinations". Bachelor's Thesis. Technical University of Munich, 2019.

[11] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[12] Felix Schrimper. “TUMexam: Digitalizing Attendee Control for Examinations – Image and Text Recognition on Mobile Devices”. Bachelor’s Thesis. Technical University of Munich, 2019.