

# Automatisierte Aufbereitung archäologischer Grabungsfotos mittels Computer Vision

Simon Metzger

## **Masterarbeit**

zur Erlangung des akademischen Grades Master of Arts im  
Studiengang Digitale Methodik der Geistes- und  
Kulturwissenschaften

Johannes-Gutenberg-Universität Mainz und Hochschule Mainz

## **Zusammenfassung**

In der Archäologie, aber auch in anderen grabenden Wissenschaften werden Funde und Befunde fotografiert und mittels einer daneben platzierten Tafel mit Metadaten versehen. Selbst wenn diese Fotografie schon digital erfolgt ist, fehlt es oft an weiteren Metadaten. Praktisch wäre daher, die Metadaten von den Tafeln zu entnehmen und den Fotos zuzuordnen. Im Rahmen des Projektes erfolgt das in zwei Schritten: Der Erkennung der Tafel, um die Textregion zu finden und die Texterkennung selbst. Beides soll in einer Python-Pipeline abgehandelt werden. Für Ersteres wurden CNNs erprobt, was nicht funktionierte, und schließlich auf klassische Computervision zurückgegriffen. Die Erkennung basiert auf findcontours, was zunächst eine Binarisierung notwendig macht. Dafür gibt es zwei verschiedene Ansätze, die sich bewährt haben: Der adaptive und der iterative. Bei ersterem werden Pixel im Kontext des sie umgebenden Bildausschnittes binarisiert, bei zweiterem werden verschiedene Grenzwerte für die Binarisierung genutzt. Die so erhaltenen Konturen werden über den Flächeninhalt mit dem sie umgebenden kleinstmöglichen Rechteck verglichen. Nähern sich die Flächen ausreichend einander an, gilt auch die Kontur als Rechteck und somit als mögliche Tafel. Es folgt das Ausschneiden der gefundenen Bildregionen in zwei möglichen Verfahren. Das erste entnimmt den Bereich des kleinstmöglichen Rechtecks aus dem Bild und eignet sich nur bedingt, um die Rotation der Tafeln auszugleichen. Das zweite dient dazu, die tatsächlichen Eckpunkte der Tafeln zu ermitteln und anhand dieser eine Projektion auf ein Rechteck vorzunehmen. Anschließend werden die so erzeugten Bildausschnitte mittels OCR auf ihre Beschriftung untersucht. Dabei müssen die Problematiken der Kreidebeschriftung ausgeglichen werden. Die Ergebnisse werden anschließen evaluiert und ausgegeben.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>2</b>
1.1 Grabung Kapitol . . . . .	2
1.2 Datensatz vorstellen . . . . .	2
<b>2 Material und Methoden</b>	<b>3</b>
2.1 Fotos . . . . .	3
2.2 Software . . . . .	5
2.3 Tafeldetektion . . . . .	5
2.3.1 CNN-basierter Ansatz . . . . .	5
2.3.2 Kontur-basierter Ansatz . . . . .	5
2.4 Cropverfahren . . . . .	12
2.4.1 Simple Crop . . . . .	12
2.4.2 Hough Crop . . . . .	13
2.5 Texterkennung . . . . .	17
2.5.1 Preprocessing . . . . .	18
2.5.2 OCR . . . . .	18
2.6 Evaluation . . . . .	19
<b>3 Ergebnisse</b>	<b>20</b>
3.1 Gesamtergebnis . . . . .	20
3.2 Tafelerkennung . . . . .	20
3.3 Texterkennung . . . . .	21
3.4 Weitere Tafeln . . . . .	22
<b>4 Diskussion</b>	<b>24</b>

# **1 Einleitung**

Einleitung und Fragestellung  
[Hough, 1962]

## **1.1 Grabung Kapitol**

Grabungsverlauf bis 2014 (recherchieren)  
Übernahme durch DAI (recherchieren)

## **1.2 Datensatz vorstellen**

Herkunft  
Umfang  
Fragestellungen des Projektes

## 2 Material und Methoden

In diesem Kapitel werden das zu Grunde liegende Material – die Grabungsfotos – und die darauf angewendete Software vorgestellt. Im Anschluss wird die Verarbeitung der Fotos im Detail beschrieben. Diese gliedert sich in zwei Teilbereiche: Die Tafeldetektion und die Texterkennung (OCR<sup>1</sup>). Die Tafeldetektion wiederum ist in die eigentliche Erkennung sowie das Ausschneiden der Tafeln aus dem Gesamtbild unterteilt. Für beide Arbeitsschritte gibt es je zwei Varianten.

### 2.1 Fotos

Der Gegenstand dieser Arbeit sind 1.453 Fotos, die während der Grabungen auf dem Kapitol in Rom in den Jahren 2011-2014 entstanden sind. Die Fotos haben eine Auflösung von 3264 x 2448 Pixeln und liegen im JPG-Format vor. Teilweise sind die exif-Daten<sup>2</sup> vorhanden, die von der Kamera erzeugt wurden. Sie beinhalten das Datum, das Kameramodell (Nikon Coolpix L19), die Belichtungsdauer und den ISO-Wert für die Lichtempfindlichkeit des Fotosensor. Weitere Metadaten – Datum, Ort, das Kürzel für die Grabung sowie weitere Anmerkungen – befinden sich auf Tafeln, die auf einigen der Fotos zu sehen sind. Das Datum auf der Tafel und das in den exif-Daten weichen teilweise um mehrere Tage voneinander ab. Bei den Tafeln, die auf den Grabungsfotos zu sehen sind, handelt es sich um Schiefertafeln mit einem Holzrahmen, die mit Kreide beschriftet wurden (Vgl. Abb 1). Die Detektion der Tafeln beruht auf folgenden Faktoren: (1) Die Tafeln haben grundsätzlich eine rechteckige Form. (2) Durch die Breite des Rahmens können bis zu zwei Rechtecke erkannt werden, ein inneres und ein äußeres. (3) Der Rahmen ist hell und die Schreibfläche dunkel, es besteht ein starker Kontrast. Die im Beispielbild gezeigte Tafel stellt ein Idealbild dar: Die Tafel nimmt einen relativ großen Teil des Originalbildes ein. Sie ist frontal vor der Kamera positioniert. Die Beleuchtung ist gut und indirekt. Keines der weiteren Bildelemente verdeckt die Tafel. Diese Beschreibung impliziert schon die Problemfelder, die bei der Detektion beachtet werden müssen:

1. Die Tafel ist unter Umständen rotiert (Abb 2).
2. Die Distanz der Tafel zur Kamera und damit ihre Größe im Bild kann stark variieren.
3. Der Rahmen der Tafel kann teilweise verdeckt oder anderweitig durch Gegenstände überlagert sein (Vgl. Abb 2).

---

<sup>1</sup>Optical Character Recognition

<sup>2</sup>Exchangeable Image File Format, das Standardformat für Metadaten in Bilddateien [CIPA Standardization Committee, 2020]



Abbildung 1: Beispiel eines Fotos der verwendeten Tafel. GOT bezeichnet die Kampagne, darunter folgt das Datum. US (*unità stratigrafica*) bezeichnet die stratigrafische Einheit.

4. Ein geringer Kontrast des Hintergrunds zum Tafelrahmen kann die Detektion erschweren.
5. Unregelmäßigkeiten im Rahmen, die auf grobe Verarbeitung oder Abnutzung zurückzuführen sind, können die Detektion erschweren.
6. Die Beleuchtung kann zu Problemen führen. Grundsätzlich sind alle Fotos hell und gut ausgeleuchtet, direktes Licht kann sich aber negativ auf die Kontraste auswirken.
7. Weitere Gegenstände, die den Spezifika der Tafeln entsprechen, können im Bild vorhanden sein.



Abbildung 2: Problematische Tafeln: Rotation und teilweise verdeckter Rahmen.

## 2.2 Software

Der Programmcode ist in Python (Version 3.8) geschrieben. Verwendet werden die *packages* Numpy (1.19.5) und vor allem OpenCV(4.4.0.44). Die Texterkennung arbeitet mit PyTesseract (0.3.7).

## 2.3 Tafeldetektion

Der folgende Abschnitt befasst sich mit der ersten Teilaufgabe dieser Arbeit: Der Detektion der Tafeln auf den Grabungsfotos. Hier werden mehrere Ansätze vorgestellt, die im Laufe der Auseinandersetzung mit dem Thema erprobt wurden: die Tafeldetektion mittels CNNs und mittels klassischer Computer Vision über die Detektion von Konturen.

### 2.3.1 CNN-basierter Ansatz

### 2.3.2 Kontur-basierter Ansatz

`Cv2.Contours` basiert auf einem Algorithmus, der Punkte gleicher Farbe und Intensität umrandet [Suzuki and Abe, 1985]. Das Resultat ist eine Liste von Punkten, die ein geschlossenes Polygon ergeben. Die gefundenen Konturen können mit einer Hierarchie versehen werden, bei der Konturen, die sich innerhalb anderer Konturen befinden, als deren „Kinder“ gelten. Das Verfahren ist darauf ausgelegt, auf binäre Bilder angewendet zu werden. Die Implementierung in OpenCV unterscheidet in Nullwerte und Nicht-Nullwerte [OpenCV-Documentation, 2021b]. Nicht-binäre Bilder werden dadurch automatisch binarisiert. Für die Erzielung optimaler Ergebnisse kommt dem Binarisierungsverfahren eine große Bedeutung zu. Auf Basis dieses Algorithmus lässt sich Detektionsverfahren aufbauen, das Objekte, die sich vom Hintergrund der Bilder abheben, zu erkennen. Die Herausforderung besteht, nach diesem Ansatz, in zwei Punkten: Erstens muss die Binarisierung so erfolgen, dass eine möglichst saubere Trennung von Vordergrund (Objekten) und Hintergrund (vor allem Erde) der Bilder stattfindet und zweitens müssen aus den gefundenen Vordergrundobjekten diejenigen ausgewählt werden, die als Tafeln in Frage kommen. Bei der Binarisierung haben sich zwei Wege als praktikabel herausgestellt, die im Folgenden beide präsentiert werden sollen. Diese Ansätze werden im Folgenden als adaptiver und als iterativer Ansatz bezeichnet. Im Flowchart sind die Abläufe der Rechtecksdetektion dargestellt (Vgl. Abb.3).

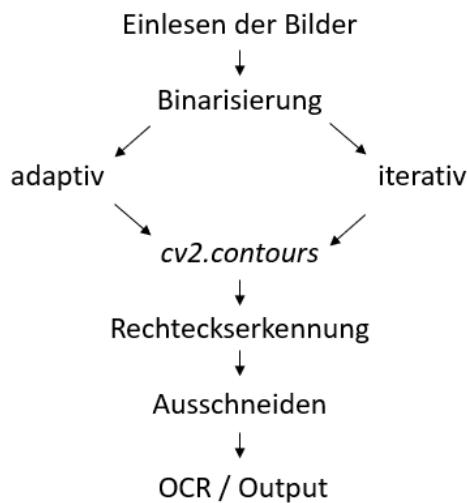


Abbildung 3: Flowchart Rechteckserkennung: Die Binarisierung erfolgt durch einen der beiden möglichen Ansätze. Auf dieser Basis wird die eine Konturerkennung durchgeführt. Aus diesen Konturen werden die Rechtecke ausgewählt und aus dem Gesamtbild ausgeschnitten. Es folgt die weitere Verarbeitung.

### Adaptive Binarisierung

Das einfachste Verfahren der Binarisierung eines Bildes besteht darin, einen Grenzwert festzulegen. Dieser muss auf der Spanne der Farbwerte, also zwischen 0 und 255 liegen. Farbwerte unterhalb dieses Grenzwertes werden zu Nullen, Farbwerte darüber zu Einsen. Das Ergebnis ist ein Schwarz-Weiß-Bild. Bei komplexen Szenarien, wie den Grabungsfotos, ist dieses Verfahren jedoch zu einfach. So kann ein Foto beispielsweise stark unterschiedliche Beleuchtung, wie direktes Sonnenlicht und Schatten, enthalten. Eine Differenzierung innerhalb dieser Zonen ist so nicht möglich (Vgl. Abb. 4).

Daher wird für den ersten Ansatz ein adaptiven Thresholds [?] gewählt: Statt global, über das gesamte Bild, einen Grenzwert festzulegen, können lokale Grenzwerte errechnet werden. Die Größe des lokalen Ausschnittes sowie das exakte Verfahren können dabei frei gewählt werden. In diesem Fall wird für die Binarisierung ein Gauss-Verfahren auf einen Kernel von 11 x 11 Pixeln angewendet. Die Beleuchtung oder Farbunterschiede innerhalb des Bildes können so ausgeglichen werden (Vgl. Abb. 5). Vor der Binarisierung wird das Bild in ein Graustufenbild umgewandelt und verkleinert<sup>3</sup>. Das verbessert die Genauigkeit der Detektion und

<sup>3</sup>Genauer: Der längere Bildrand wird auf 1000 Pixel reduziert, der kürzere entsprechend angepasst. Die Grabungsfotos haben, wie bereits erwähnt, eine Auflösung von 3264 x 2448 Pixeln. Es

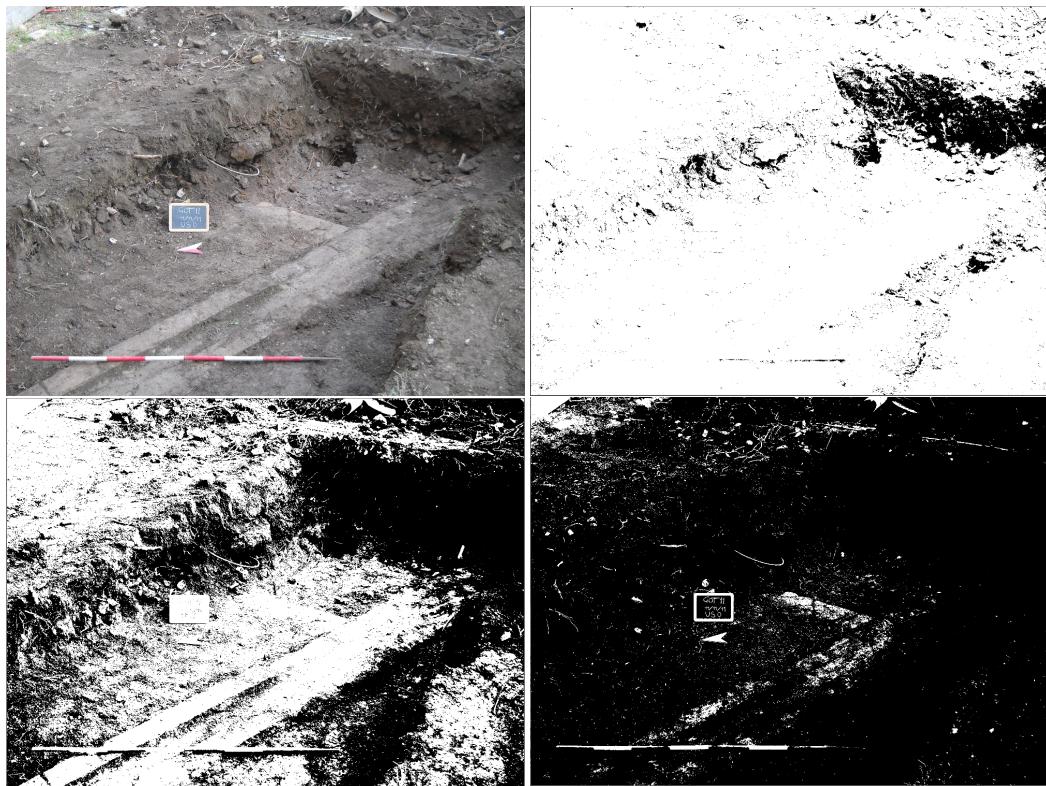


Abbildung 4: Grabungsfoto im Original (o.l.), mit niedrigem (60, o.r.), mittlerem (125, u.l.) und hohem (185, u.r.) Grenzwert.

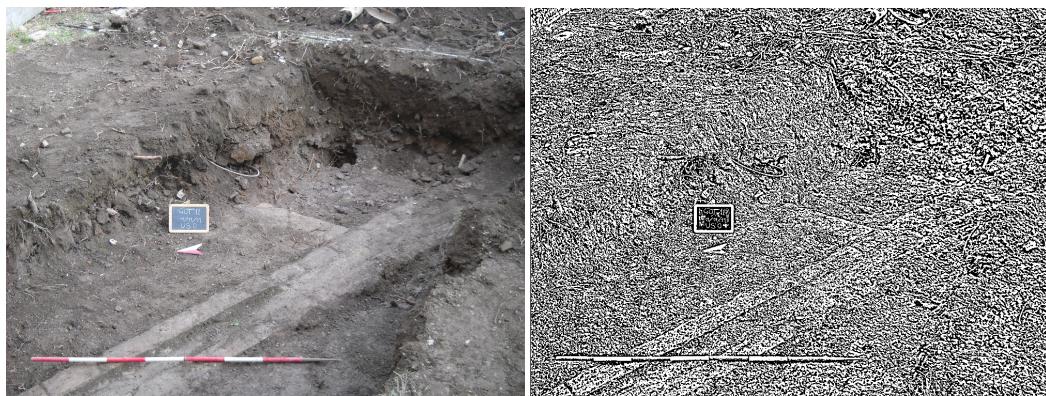


Abbildung 5: Grabungsfoto im Original sowie mit adaptivem Threshold.

findet also eine Reduktion auf ca.  $\frac{1}{10}$  der Fläche statt. Bilder kleiner als 1000 Pixel würden theoretisch auf 1000 Pixel vergrößert werden. Da für die hier beschriebenen Prozesse und vor allem für die Texterkennung später aber eine gewisse Bildqualität erforderlich ist, ist von diesem Fall nicht

verringert die zu verarbeitende Datenmenge, wodurch eine Beschleunigung des Prozesses zu erwarten ist.

### Iterative Binarisierung

Die Grundidee der iterativen Binarisierung besteht darin, das Bild mit einem globalen Grenzwert zu binarisieren. Die Problematik davon besteht darin, dass bei großen Beleuchtungsunterschieden oder sehr hellen oder dunklen Objekten auf dem Bild ganze Bereiche durch den Grenzwert von der weiteren Bearbeitung ausgeschlossen werden. Der iterative Ansatz sieht daher vor, den Grenzwert von 20 auf 200 in Fünferschritten zu erhöhen. Dadurch entstehen pro Foto 37 binäre Bilder, auf die die Konturenerkennung angewendet werden kann. Auf jedem dieser Bilder können mehrere mögliche Tafeln erkannt werden<sup>4</sup>. Der nächste Schritt besteht also darin, aus diesen möglichen Tafeln die auszuwählen, die am wahrscheinlichsten tatsächlich eine ist. Dazu wird eine Grundannahme getroffen: In dem Bereich, in dem auf den meisten der 37 Bilder eine Tafel vermutet wird, befindet sich tatsächlich eine Tafel. Alle anderen werden als Falsch-Positive betrachtet. Diese Annahme beruht auf zwei Faktoren: Erstens hat sich gezeigt, dass Objekte, die keine Tafeln sind, aber als solche erkannt werden können – z.B. Fenster, Türen oder Plakate – nur unter wenigen Grenzwertes als solche eingeordnet werden. Das liegt unter anderem daran, dass die Tafeln einen hellen Holzrahmen und eine dunkle Innenfläche haben, was zu einem starkem Kontrast führt, der auf vielen Stufen des Grenzwertes erhalten bleibt<sup>5</sup>. Außerdem werden durch eben diesen Rahmen in mittleren Grenzwert-Bereichen die Tafeln oft zweimal erkannt: Einmal an der Außenkante und einmal an der Innenkante des Rahmens. Dadurch häuft sich die Detektion möglicher Tafeln in diesem Bereich. Basierend auf dieser Annahme wird auf alle möglichen Tafeln immer paarweise die intersection over union angewendet. Dieser Algorithmus basiert auf dem Jaccard-Koeffizienten zur Berechnung der Ähnlichkeit zweier Mengen [Jaccard, 1902]. Der Koeffizienten wird errechnet, indem die Schnittmenge durch die Vereinigungsmenge geteilt wird. Das Ergebnis liegt zwischen 0 und 1. Je mehr es sich der 1 annähert, desto ähnlicher sind die Mengen. Diese Berechnung lässt sich auch auf die Rechtecke, mit denen die möglichen Tafeln verortet werden, anwenden (Vgl. Abb. 6).

---

auszugehen.

<sup>4</sup>Die eigentliche Tafelerkennung wird erst im folgenden Abschnitt beschrieben. Da die dort gewonnenen Informationen nicht, wie beim adaptiven Ansatz, an das Hauptprogramm übergeben, sondern innerhalb der Funktion des iterativen Ansatzes weiter verarbeitet werden, ist hier ein Vorgriff nötig.

<sup>5</sup>Die Tafeln verfügen damit über eine Eigenschaft, die auch beim Einsatz von AR-Markern genutzt wird: Starke hell-dunkel Kontraste beschleunigen und vereinfachen die Detektion

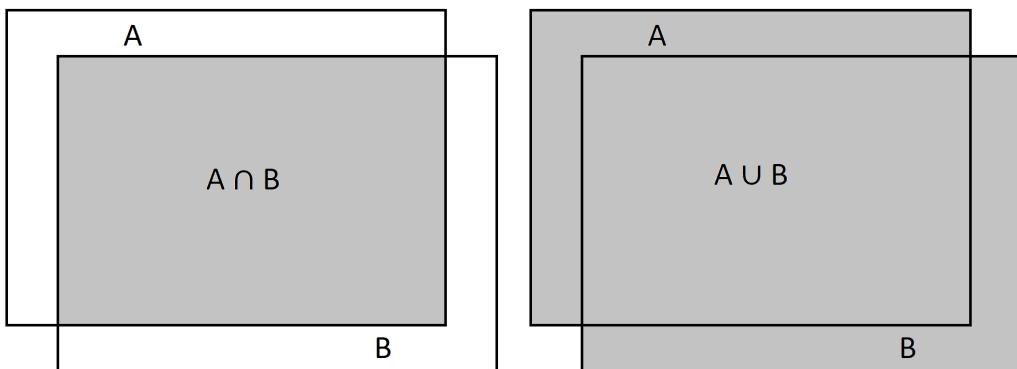


Abbildung 6: Schnittmenge und Vereinigungsmenge von Rechtecken. Der Jaccard-Koeffizient wird durch die Division der beiden Flächen bestimmt.

Zwei Rechtecke galten dann als hinreichend ähnlich, wenn der Jaccard-Koeffizient über 0.9 lag. Jedes Mal, wenn ein Rechteck im Vergleich mit einem anderen diesen Wert erzielt, wird für das Rechteck ein Zähler erhöht. Das Rechteck, welches am Ende der Vergleiche den höchsten Wert in diesem Zähler erzielt, wird als tatsächliche Tafel angenommen. Eine Ausnahme ergibt sich, wenn dieser Wert unter 6 liegt. In diesem Falle wird von Falsch-Positiven und somit einem Foto ohne Tafel ausgegangen.

### Rechteckserkennung

Ist die Binarisierung der Bilder nach einer der beiden vorgestellten Methoden erfolgt, können mittels `cv2.findContours` die Konturen der Objekte darauf gefunden werden (Vgl. Abb. ??). Die Hierarchie der Konturen wird dabei außer Acht gelassen, da sich daraus keine verlässlichen Informationen gewinnen lassen. Als nächstes müssen unter diesen Konturen die ausgewählt werden, die als Tafel in Frage kommen. Dazu wird die Tatsache genutzt, dass Tafeln auf den Fotos als Rechtecke abgebildet werden. Durch ihre Lage zur Kamera können sie zu einem gewissen Grad davon abweichen, grundsätzlich bleibt diese Form aber erhalten. Andere Rechtecke kommen zwar vor, wie Plakate, Fenster, Türen, Ziegel und Fliesen, unterscheiden sich jedoch meist in der tatsächlichen Form oder können im Zweifelsfall bei der späteren Texterkennung ausgeschlossen werden. Dementsprechend geht es in der weiteren Tafelerkennung darum, Rechtecke zu finden und diese durch verschiedene weitere Kriterien so zu sortieren, dass alle Tafeln und möglichst nur Tafeln erkannt werden. Das wird in mehreren Schritten erreicht: Zunächst wird die Fläche der Konturen berechnet. Genutzt wird hierfür

[Siltanen, 2012, p. 45]



Abbildung 7: Grabungsfoto im Original sowie nach der Anwendung der Konturerkennung.

die OpenCV-Funktion `cv2.contourArea`. Ist die Fläche zu klein, wird die Kontur aussortiert. Als Grenzwert wird hier die Kantenlänge der längsten Kante des Bildes genommen, damit bei höherer Auflösung weiterhin korrekt sortiert werden kann(Vgl. Abb. 8). Eine Größenbegrenzung ist sinnvoll, da die Tafeln in der Regeln eher prominent im Bild zu sehen sind. Sollte tatsächlich eine Tafel durch dieses Kriterium aussortiert werden, wäre der Text darauf nicht mehr lesbar und somit ohnehin nicht interessant für die weitere Verarbeitung.

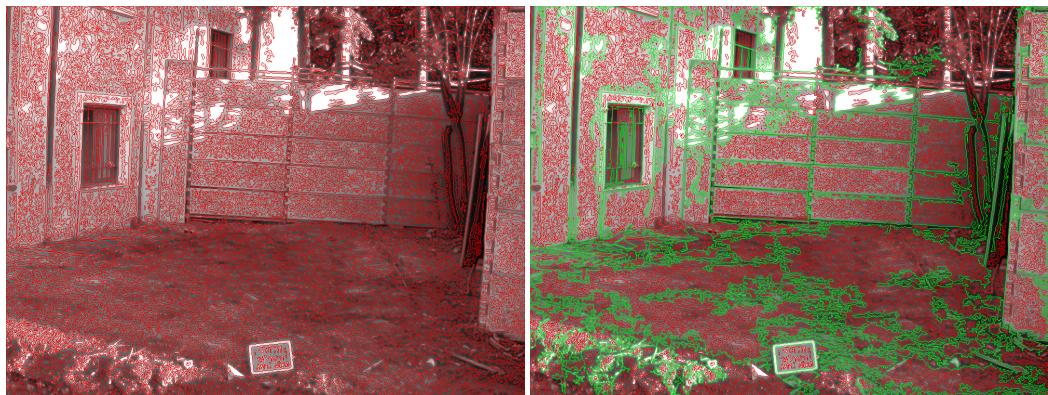


Abbildung 8: Konturen vor und nach der Selektion nach Fläche.

Der nächste Schritt besteht darin, durch `cv2.minAreaRect` das kleinstmögliche Rechteck um die Kontur zum bilden (Vgl. Abb. 9).

Anschließend werden die bereits berechneten Flächen der Konturen mit denen der kleinstmöglichen Rechtecke verglichen. Die Annahme: Nähert sich der Flächeninhalt der Kontur dem des Rechtecks an, handelt es sich bei der Kontur selbst wahrscheinlich um ein Rechteck. Dabei hat sich als Grenzwert bewährt,

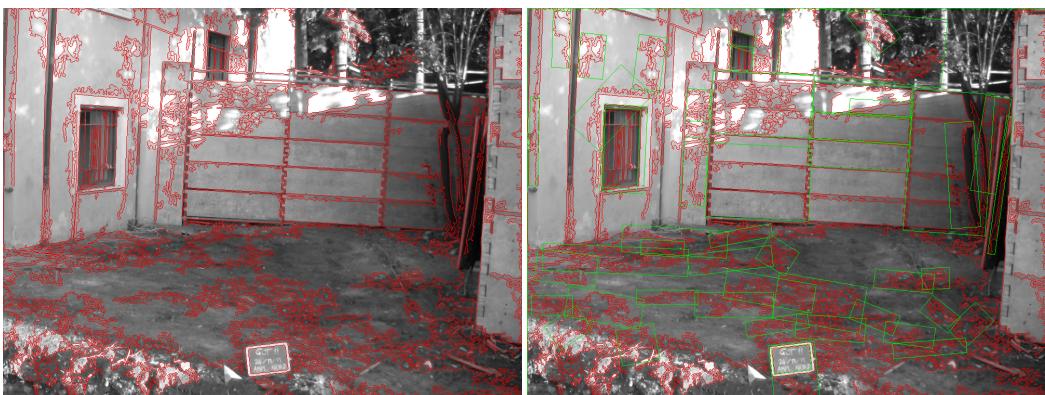


Abbildung 9: Konturen ohne und mit Rechtecken.

dass die Fläche der Kontur 85% der Fläche des Rechtecks betragen sollte. Ebenfalls ausgeschlossen wird ein Rechteck, wenn es die Fläche des gesamten Bildes ausmacht, da teilweise, wie auch bei diesem Beispiel, der Rand des Bildes als Kontur erkannt werden kann (Vgl. Abb. 10).



Abbildung 10: Rechtecke aller Konturen und jene derer mit annähernd rechteckiger Fläche.

Als letztes Kriterium wird das Seitenverhältnis herangezogen. Im Programm besteht die Möglichkeit, vor der Untersuchung aller Bilder ein Template zu hinterlegen, auf dem eine Tafel gut und eindeutig zu sehen ist. Aus diesem Template wird das Seitenverhältnis der Tafel berechnet. Findet dieser Schritt nicht statt, wird ein maximales Seitenverhältnis von 2:1 angenommen, da Tafeln selten in länglichen Formaten zu finden sind<sup>6</sup>. Mit einer deutlichen, Toleranz von 30% in jede

<sup>6</sup>Sollte das Format einer Tafel doch einmal von dieser Annahme abweichen, lassen sich hier problemlos Anpassungen vornehmen.

Richtung wird das Seitenverhältnis der verbliebenen Rechtecke mit dem des Templates verglichen. Sind die Werte sich ähnlich genug, wird das Rechteck als Tafel interpretiert und gilt als das Ergebnis des Algorithmus (Vgl. Abb. 11).



Abbildung 11: Die bisher verbliebenen Rechtecke und die mit dem richtigen Seitenverhältnis.

## 2.4 Cropverfahren

Durch die vorhergehende Rechtecksdetektion sind die Positionen der Tafeln bekannt. Der nächste Schritt besteht darin, die Tafeln aus dem Gesamtbild auszuschneiden, damit diese Ausschnitte für die Texterkennung genutzt werden können. Auch die perspektivischen Verzerrungen der Tafeln ausgeglichen werden müssen. Für diesen Ausgleich müssen die Eckpunkte der Tafeln bestimmt und das durch sie definierte Viereck auf die Fläche eines Rechtecks übertragen werden. Auch hier haben sich wieder zwei Verfahren ergeben, von denen eines effizient arbeitet, während das zweite stärker auf das Entzerren der Bilder abzielt (Vgl. Abb. 12).

### 2.4.1 Simple Crop

Ein Weg, die Tafeln aus den Bildern auszuschneiden, besteht darin, die kleinstmöglichen Rechtecke aus dem Detektionsverfahren als Basis für den Ausschnitt zu nutzen. Basierte die Detektion auf dem adaptiven Ansatz, musste zunächst das Rechteck auf die ursprüngliche Bildgröße skaliert werden, da für die weitere Verarbeitung die Originalbilder mit der höheren Auflösung verwendet wurden. Zu diesem Zweck wurde das Verhältnis aus der Größe des Originalbildes und der des skalierten Bildes gebildet. Multipliziert man dieses Verhältnis mit den vier Kennzahlen der Rechtecke (X- und Y-Koordinate des Mittelpunktes sowie Breite und Höhe) wird das Rechteck passend skaliert. Als letzter Schritt wurden aus diesen Werten, unter Einbezug der Rotation des Rechtecks, die Eckpunkte berechnet.

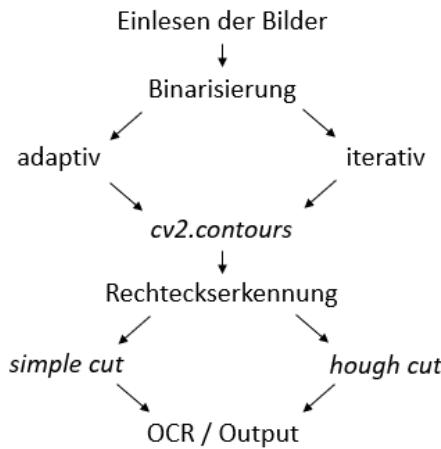


Abbildung 12: Flowchart Crop-Verfahren.

Neben den so erhaltenen Ausgangskoordinaten mussten aber auch Zielkoordinaten erzeugt werden, auf die das Rechteck projiziert wird. Diese wurden ebenfalls dem Detektionsrechteck entnommen, indem Höhe und Breite als Maße für das Projektionsziel genutzt wurden. Für die Transformation des Ausgangsrechtecks auf das Zielrechteck wurde mittels `cv2.tafelcrop` die Transformationsmatrix erzeugt ([OpenCV-Documentation, 2021a]). Die eigentliche Transformation erfolgte dann durch `cv2.warpPerspective` (Vgl. Abb. 11). Der so erzeugte Tafelausschnitt wurde anschließend auf 1000 Pixel skaliert, was in der Regel eine Vergrößerung darstellt, um eine einheitliche Größe zu gewährleisten und zu kleinen Ausschnitten zu vermeiden. Außerdem wurde das Bild um 90 °rotiert, wenn die Höhe die Breite übertraf, um nur Ausschnitte im Querformat zu erhalten.

#### 2.4.2 Hough Crop

Durch unterschiedlichen Perspektiven, aus denen die Fotos aufgenommen wurden, ist die beschriftete Seite der Tafeln nicht immer frontal der Kamera zugewendet. Simple Crop ist nicht geeignet, um die daraus resultierende Verzerrung auszugleichen (Vgl. Abb. 14).

Daher wurde ein zweiter Ansatz entwickelt, um dieses Problem zu beheben. Ziel war es hier, statt der Eckpunkte der gefundenen Rechtecke die Eckpunkte der Tafel für das Cropverfahren zu nutzen. Diese mussten also zunächst gefunden werden. Das kann erreicht werden, indem die Seiten der Tafeln durch Kan-tendetektion ermittelt und im Anschluss deren Schnittpunkte berechnet werden [Ye et al., 2007]. Zunächst musste dazu ein passender Bildausschnitt gewählt wer-

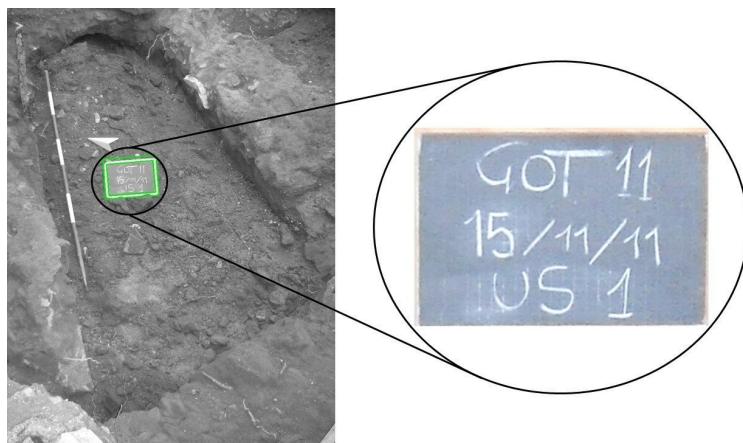


Abbildung 13: Eine Tafel vor und nach dem Ausschneiden.

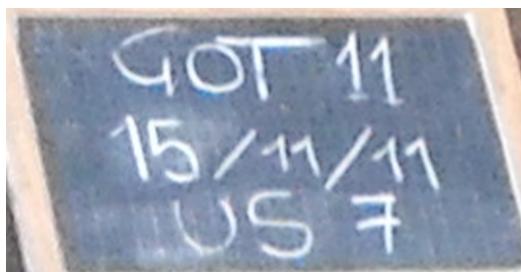


Abbildung 14: Diskrepanz zwischen den Eckpunkten der gefundenen Rechtecke und den Eckpunkten der Tafel.

den. Auf den Tafeln befinden sich viele Objekte, die lange gerade Kanten aufweisen. Nach der Detektion zu entscheiden, welche davon Teil einer Tafel sind und welche nicht ist komplex und würde die gesamte Problematik der Detektion neu aufwerfen. Gleichzeitig konnten auch die detektierten Rechtecke nicht verwendet werden, da hier Teile des Rahmens abgeschnitten sein können. Daher wurden die Rechtecke um den Faktor 1,3 vergrößert und mittels `simple crop` ausgeschnitten (15).

Diese Bildausschnitte wurden mit mehreren Filtern bearbeitet, um das meist stark vorhandene Rauschen zu reduzieren. Es folgte die Umwandlung in ein Graustufenbild, das normalisiert wurde. Beim Prozess der Normalisierung werden Farbskalen, in diesem Fall also die Graustufen, auf ihre maximalen Werte gestreckt. Der niedrigste wird also auf 0, der höchste auf 255 gesetzt und alle Werte dazwischen entsprechend angepasst (16).

Auf die normalisierten Bilder wurde die *Canny Edge Detection* (Kantenerkennung) angewendet [Canny, 1986]. Dieser Algorithmus berechnet die Gradi-

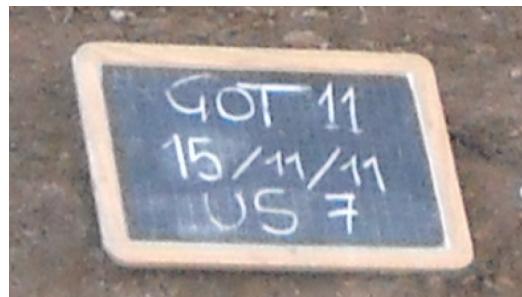


Abbildung 15: Bildausschnitt für Hough Crop, basierend auf dem vergrößerten Tafelrechteck.

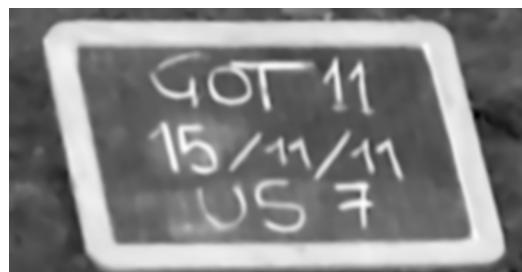


Abbildung 16: Graustufenbild, gefiltert und normalisiert.

enten der einzelnen Pixel, wodurch Kantenverläufe im Bild erkennbar werden. Auf diese wird eine *non-maximum suppression* angewendet, d.h., die Gradienten der Pixel werden mit ihren Nachbarn verglichen und jeweils nur das Maximum weiterverwendet. Die Kantenstärke wird somit auf die Breite von einem Pixel reduziert. Aus den so erhaltenen Gradienten werden mittels zweier Schwellwerte durch Hysterese die gewünschten Kanten bestimmt: Enthält ein Pixel einer Kante den zweiten, höheren Schwellwert, gilt sie als eine der gesuchten Kanten. Von dort ausgehend werden alle Pixel der Kante, die den niedrigeren Schwellwert übertreffen, ebenfalls als Teil dieser Kante (17).

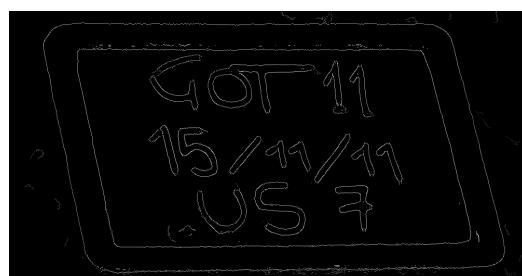


Abbildung 17: Kantendetektion mit dem Canny-Algorithmus.

Dieses Gradientenbild wiederum wurde mittels Hough<sup>7</sup>-Transformation [Hough, 1962] einer Linienerkennung unterzogen. Die vorher detektierten Kanten, die einen beliebigen Verlauf nehmen können, wurden also jetzt darauf geprüft, ob sie eine gerade Linie bilden. Das erfolgt, indem mit der hesseschen Normalform jede mögliche Gerade durch jeden Kantenpunkt berechnet wird. Jeder Kantenpixel, durch den die Geraden verlaufen, erhält einen *upvote*, es wird also ein Zähler inkrementiert. Durch die Bereiche mit den meisten *Upvotes* laufen die gesuchten Linien auf dem Bild. Ist eine Linie gefunden, ist sie wahrscheinlich Teil des Rahmens der Tafel und somit für die Erkennung der Eckpunkte relevant (18).

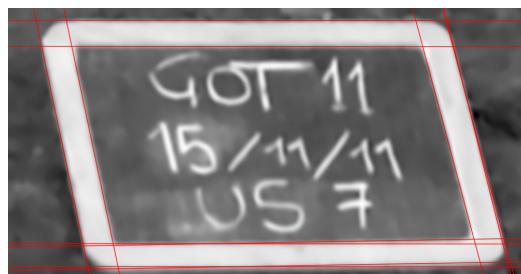


Abbildung 18: Linienerkennung mit dem Hough-Algorithmus.

Im nächsten Schritt wurden die Geraden mit dem Bildrahmen geschnitten. Die Schnittpunkte wurden als neue Endpunkte des Liniensegments für die weiteren Berechnungen genutzt, um etwaige Schnittpunkte außerhalb des Bildes ausschließen zu können. Die Endpunkte wurden, entsprechend ihrer Position im Bild, den vier Ecken zugeteilt. Anschließend wurden die Schnittpunkte der Linien untereinander berechnet, aber nur, wenn sie 1) nicht in die gleiche Richtung und 2) nicht diagonal durchs Bild verliefen. Beide Kriterien konnte durch die Einordnung der Eckpunkte bestimmt werden. Damit war sichergestellt, dass nur Linien miteinander geschnitten wurden, die annähernd senkrecht zueinander verlaufen, wie es bei den Tafelrändern der Fall ist. Schnittpunkte am Rand wurden ausgeschlossen, um die Detektion des Bildrandes oder Störobjekte zu auszuschließen. Die Gruppe der Schnittpunkte wurde wiederum in die vier Ecken aufgeteilt. In jeder Ecke wurde jetzt der Punkt bestimmt, der am weitesten Richtung Bildmitte liegt. Dieser wurde als der wahrscheinlichste Eckpunkt angenommen (19). Das weitere Verfahren entsprach dem des simple crop-Ansatzes: Mit den Eckpunkten wurde eine Transformationsmatrix berechnet, durch die der Bildausschnitt innerhalb der vier Eckpunkte auf eine Rechtecksform übertragen werden konnte (20).

<sup>7</sup>Gesprochen: [hʌf]

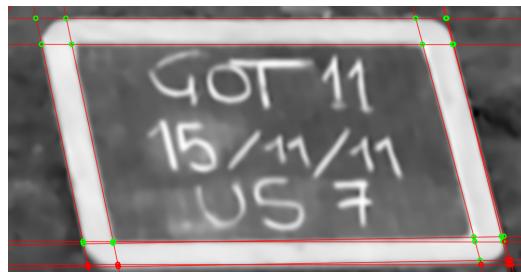


Abbildung 19: Die Schnittpunkte der Linien nach der Berechnung. Die roten Punkte liegen zu nahe am Rand und wurden daher ausgeschlossen.



Abbildung 20: Die Tafel, in ein Rechteck transformiert.

## 2.5 Texterkennung

Im folgenden Abschnitt wird der zweite Aspekt der Verarbeitung von Grabungsfotos vorgestellt: Die Texterkennung oder *optical character recognition* (OCR). Da die Vorbereitung der Bilder für das Gesamtergebnis von großer Bedeutung ist, wird zunächst das *Preprocessing* behandelt. Im Anschluss wird die eigentliche Texterkennung mit Tesseract vorgestellt. Folgende Probleme müssen bei der Texterkennung adressiert werden: (1) Komplexität der Szene, (2) Beleuchtungsverhältnisse, (3) Rotation des Textes relativ zur Kamera, (4) Unschärfe, (5) Größe und Format der Textfelder, (6) Neigungswinkel des Textes relativ zur Kamera, (7) Schriftart, (8) Mehrsprachigkeit, (9) perspektivische Verzerrungen [Hamad and Kaya, 2016]. In diesem Falle wurde allen Aspekten, die der Textdetektion gelten (1, 5), geringeres Gewicht beigemessen, da die Tafeln bereits als die Grenzen des beschriebenen Areals betrachtet werden konnten. Die Neigung und die Schiefe der Buchstaben relativ zur Kamera (3,6,9) wurden bereits beim Cropverfahren entsprechend der Möglichkeiten ausgeglichen. Beleuchtung, Unschärfe und Rauschen waren Teil des Preprocessings (2.4), die sprachbezogenen Punkte (7,8) wurden im Rahmen der eigentlichen Texterkennung adressiert.

### 2.5.1 Preprocessing

Das Preprocessing für die Texterkennung folgte gängigen Verfahren<sup>8</sup> [Shah and Gokani, 2014], [Hallale and Salunke, 2013]. Zunächst mussten die Bilder in Graustufen umgewandelt werden. Nacheinander wurden zwei Filter angewendet, `cv2.GaussianBlur` und `cv2.fastNlMeansDenoising`. Damit sollten die Störeffekte durch verwischte Kreide auf der Tafel ausgeglichen werden. Das Bild wurde invertiert, so dass die eigentliche weiße Kreide schwarz dargestellt und der Hintergrund hell wurde, was den Empfehlungen für Tesseract entspricht [Tesseract Documentation, 2021]. Der letzte Schritt bestand in der Normalisierung der Bilder. Alle diese Schritte wurden auf das Grundbild sowie die daraus extrahierten 3 Farbkanäle angewendet [Chen et al., 2004].

### 2.5.2 OCR

Pro gefundener Tafel waren jetzt vier Bilder vorhanden, die der Texterkennung unterzogen werden konnten. Für jedes dieser Bilder wurde diese zweifach ausgeführt: Einmal mit `image_to_string` und einmal mit `image_to_boxes`. Ersteres liest den Text Zeilenweise ein, letzteres jeden Buchstaben einzeln, wodurch die Ergebnisse variieren. `Image_to_string` wurde zusätzlich auf das um 180° rotierte Bild angewendet, um die Orientierung zu ermitteln: Die Ergebnisse der Texterkennung beider Orientierung wurden, nach der Bereinigung von Leerzeichen und Zeilenumbrüchen, anhand der Länge miteinander verglichen. Die Version, bei der mehr Text gefunden wurde, wurde als korrekt orientiert angenommen. Eine solche Funktion ist zwar in Tesseract bereits grundsätzlich implementiert, da die Texterkennung hier aber ohnehin unter erschwerten Bedingungen und ohne erkennbare Wörter stattfand, wurde auf diesen Ansatz zurückgegriffen<sup>9</sup>. Die Ergebnisse der beiden Pytesseract-Funktionen aus den je vier Bildern wurden im Anschluss untereinander verglichen. Der längste gefundene Text wurde als Ergebnis übernommen. Beide Befehle wurden mit folgender Konfiguration ausgeführt: (1) Es wurde eine Whitelist verwendet, die auf das Schema der Tafeln zugeschnitten wurde. Gelistet wurden die Ziffern 0-9, die Buchstaben *G,O,T,U* und *S* sowie das Sonderzeichen */*. (2) Als Sprache wurde ein eigenes Wörterbuch angegeben, das entsprechend der Tafeln nur wenige Worte enthält (*US, GOT*). (3) Der *OCR Engine Mode* (*oem*) wurde auf ein Neurales Netzwerk mit *Long Short-term Memory*[Hochreiter and Schmidhuber, 1997] festgelegt. (4) Für die *Page Segmentation* (*psm*) wurde ein vollautomatischer Modus, ohne wei-

<sup>8</sup>Warum übliche Schritte im Preprocessing, wie das Thresholding, hier unterblieben, wird im weiteren Verlauf dieser Arbeit ausführlich diskutiert werden.

<sup>9</sup>U.a. liefert die Funktion `image_to_osd` von Pytesseract die Orientierung des gefundenen Textes. Für die Anwendung derselben befindet sich auf den Tafeln zu wenig Text.

tere Vorgaben, gewählt.

## 2.6 Evaluation

Die Ergebnisse sowohl bei Tafelerkennung und -ausschnitt als auch bei der Texterkennung wurden durch statistische Analyse des erkannten Textes evaluiert. Dazu wurden die Texte auf einer Auswahl von 79 Tafeln transkribiert und in einer Liste abgelegt. Diese Liste wurde dann als Referenz (Grundwahrheit) für die Auswertung der erkannten Texte eingelesen. Als Maße für die Genauigkeit der Erkennung wurden die Maße *recall*, *precision* und *f-score* verwendet [Klinke, 2017] [Ye et al., 2007]. *Recall* gibt dabei an, wie viele der Buchstaben auf der Tafel tatsächlich erkannt wurden ( $Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$ ). Die *Precision* gibt an, wie viele der erkannten Buchstaben tatsächlich Teil der Grundwahrheit sind ( $Precision = \frac{TruePositives}{TruePositives + FalsePositives}$ ). Der F-Score erzeugt einen gewichteten, harmonischen Mittelwert aus diesen beiden Maßen, um die Qualität der Texterkennung in einem Wert zusammen zu fassen ( $F = 2 \times \frac{precision \times recall}{precision + recall}$ ). Der F-Score wurde dabei als bevorzugtes Maß verwendet, die beiden anderen können optional mit ausgegeben werden. Aus den F-Scores wurden mehrere Durchschnittswerte gebildet, um das Gesamtergebnis zu evaluieren: (1) Der Mittelwert der zeilenweisen Erkennung, (2) der Mittelwert der buchstabenweisen Erkennung, (3) der Mittelwert aus diesen beiden Werten, (4) der Mittelwert der als bestes Ergebnis eines Bildes ermittelten Textzeilen.

## 3 Ergebnisse

In diesem Kapitel werden die Ergebnisse der zuvor vorgestellten Methoden präsentiert. Dabei wird zunächst das Gesamtergebnis präsentiert. Im Anschluss wird auf Besonderheiten bei Tafel- und Texterkennung eingegangen. Basis der Auswertung ist dabei ein Testdatensatz aus 292 Bildern, worunter sich 79 Bildern mit Tafeln befinden. Im Anschluss werden Versuche mit den Datensätzen anderer Projekte präsentiert.

### 3.1 Gesamtergebnis

Für das Gesamtergebnis wurden die Tafelausschnitte in den vier Varianten gemeinsam der Texterkennung unterzogen (Vgl. collection.csv). Der durchschnittliche *F-Score (average)* betrug im Ergebnis 0.49. Die Durchschnittswerte der zeilenweisen (*textaverage*) und buchstabenweisen (*boxaverage*) Texterkennung lagen bei 0.51 bzw. 0.47. Die durch das Programm als bestes Ergebnis ausgewählten Zeichenketten (*bestaverage*) erzielten einen *F-Score* von 0.61. Das theoretische Optimum (*optimum*), also der Durchschnitt der tatsächlichen besten Ergebnisse pro Bild, lag bei 0.7. Einige der Tafelausschnitte konnten nicht ausgelesen werden. Das schlechteste Ergebnis wurde bei catacom\_1110 erzielt, auf dem Text nur bei einem von 5 Ausschnitten erkannt wurde (*F-Score* 0.15). Das beste Ergebnis war ein *F-Score* von 1 bei catacom\_1152.

Verfahren	boxaverage	textaverage	average	bestguess	optimum
Gesamt	0.47	0.51	0.49	0.61	0.7
Adaptive Hough	0.43	0.47	0.45	0.49	0.56
Adaptive Simple	0.52	0.57	0.54	0.61	0.64
Iterative Hough	0.44	0.47	0.45	0.47	0.52
Iterative Simple	0.51	0.56	0.54	0.57	0.6

### 3.2 Tafelerkennung

Bei der Tafelerkennung zeigten beide Ansätze gute Ergebnisse, die nur in Nuancen voneinander abwichen. So erkannte der adaptive Ansatz aus 292 Bildern 133 mit Tafeln. Dass diese Zahl über der der 79 Tafeln lag, ist darin begründet, dass der äußere und der innere Tafelrand Tafel identifiziert werden können, es also bis zu zwei richtige Erkennungen pro Bild geben kann<sup>10</sup>. Zudem wurden 8 Falsch-Positive erkannt, Falsch-Negative gab es keine. Der *Recall* lag damit, wie gewünscht, bei 1. Die *Precision* betrug 0.926, der *F-Score* damit 0.961. Beim

<sup>10</sup>Der Einfluss dieses Umstandes auf die Kennwerte ist so gering, dass er vernachlässigbar ist.

iterativen Ansatz wurden 85 Tafeln erkannt. Da dieser Ansatz nur das wahrscheinlichste Ergebnis pro Bild ausgibt und hier keine Falsch-Negativen auftraten, lag die Zahl der Falsch-Positiven bei 6. Der *Recall* war also auch hier 1, die *Precision* 0.929. Der *F-Score* betrug 0.963. Die relativ hohe Zahl der Falsch-Positiven resultierte aus vier Fotos, auf denen Plakate abgebildet waren. Da Plakate alle Kriterien der Tafeln erfüllen – rechteckig, passendes Seitenverhältnis, mit Text beschrieben – konnte der Algorithmus weder in diesem Schritt noch später, bei der Texterkennung, diese Bilder aussortieren. Eine manuelle Entfernung der Bilder aus dem Datensatz, die sich auch in der Praxis als erster Schritt empfehlen würde, steigerte den *F-Score* auf 0.992 beim adaptiven und auf 0.987 beim iterativen Ansatz. Die Unterschiede zwischen beiden Ansätzen sind also gering, wobei die Anfälligkeit für Falsch-Positive im iterativen Ansatz etwas niedriger ist.

Bei der Auswertung der Texterkennung nach Detektionsverfahren ergaben sich Unterschiede: Beim adaptiven Verfahren (ausgewertet mit beiden Schnittverfahren) lag das *bestaverage* des *F-Scores* bei 0.59, das theoretische Optimum bei 0.67 (Vgl. adaptive only.csv). Das iterative Verfahren kam auf einen *F-Score* von 0.56 bei einem Optimum von 0.65 (Vgl. iterative only.csv). Beim nächsten Schritt, dem Schnittverfahren, variierten die Ergebnisse deutlicher. So erzielte das *bestaverage* des Hough Crop-Verfahrens (iterativer und adaptiver Ansatz parallel) einen *F-Score* von 0.49, das theoretische Optimum betrug 0.59 (Vgl. hough only.csv). Im Gegensatz dazu lag das *bestaverage* des Simple Crop bei 0.62 und somit noch über der Gesamtauswertung. Das theoretische Optimum lag mit 0.67 etwas unter der Gesamtauswertung (Vgl. simple only.csv). Als Einzelverfahren war die Kombination Adaptive Simple Crop am besten, mit einem *bestaverage* von 0.61 und einem theoretischen Optimum von 0.64 (Vgl. crop adaptive simple.csv).

Verfahren	boxaverage	textaverage	average	bestguess	optimum
Gesamt	0.47	0.51	0.49	0.61	0.7
Adaptive (Hough + Simple)	0.47	0.51	0.49	0.59	0.67
Iterative (Hough + Simple)	0.48	0.51	0.5	0.56	0.65
Hough (Adaptive + Iterative)	0.43	0.47	0.45	0.49	0.59
Simple (Adaptive + Iterative)	0.52	0.56	0.54	0.62	0.67

### 3.3 Texterkennung

Im Bereich der Texterkennung ließ sich ein Unterschied zwischen Image to Box, also der Buchstabenerkennung, und Image to Text, der Zeilenerkennung, feststellen. So war der mittlere *F-Score* bei der Buchstabenerkennung um 0.03 bis 0.05 niedriger als bei der Zeilenerkennung (Vgl. collection.csv u.a.). Im Einzelfall variierten die Ergebnisse jedoch stark, sodass die Buchstabenerkennung

deutlich bessere Ergebnisse als die Zeilenerkennung liefern kann, was sich positiv auf den *bestguess* auswirkt. Wie bereits vorgestellt wurde für die Texterkennung ein Wörterbuch und eine Whitelist verwendet. In der Auswertung zeigte sich, dass vor allem die Whitelist das Ergebnis signifikant verbessert. Die Anwendung der Texterkennung ohne Wörterbuch, aber mit Whitelist, auf die mit Adaptive Simple Crop erzeugten Tafelausschnitte erzielte die gleichen Ergebnisse wie der Durchlauf mit Wörterbuch, also ein *bestaverage* von 0.61 (Vgl. no dic.csv). Wurden sowohl Whitelist als auch Wörterbuch entfernt, sank der durchschnittliche *F-Score* auf 0.26, wobei die Buchstabenerkennung 0.25 erzielte, die Zeilenerkennung nur 0.01 (Vgl. no config no dic.csv). Schaltete man nur das Wörterbuch dazu, verbesserte sich der Wert der Zeilenerkennung auf 0.26 (Vgl. dic only.csv). Festzustellen war auch, dass die Auswahl des *bestguess* nie zu Ergebnissen unterhalb es Durchschnitts führte, andererseits aber für deutliche Verbesserungen sorgen konnte. Deutlich wurde das in der Gesamtauswertung aller Ansätze, bei der die Differenz zwischen dem durchschnittlichen *F-Score* der beiden Texterkennungsverfahren (0.49) und dem durchschnittlichen *F-Score* des *bestguess* (0.61) 0.12 Punkte betrug. Allerdings bestand auch ein Unterschied von 0.9 zum theoretischen Optimum (0.7) (Vgl. collection.csv).

### 3.4 Weitere Tafeln

Bei der Anwendung des Algorithmus auf die projektfremden Tafeln der Gruppe Terrestrische Ökohydrologie der FSU Jena sowie der späteren Grabungen am Kapitol durch das Deutsche Archäologische Institut ergaben sich folgende Ergebnisse: Bei den Tafeln der Ökohydrologie Jena wurde eine Stichprobe aus 40 Fotos mit Tafeln verwendet. Von diesen wurden 9 korrekt erkannt. Auf 16 Fotos wurde fälschlicherweise ein beiliegendes Maßband erkannt<sup>11</sup>. Der *Recall* beträgt hier 0.225, die *Precision* 0.36. Der *F-Score* liegt bei 0.277. Diese Ergebnisse wurden mit dem iterativen Verfahren erzielt. Mittels adaptivem Ansatz konnten keine Tafeln ermittelt werden. Die Texterkennung verlief – abgesehen von Zahlen auf dem Maßband – ergebnislos. Aus den Fotografien der Grabungen des DAI wurden 35 Bilder mit Tafeln darauf ausgewählt. Mit dem adaptiven Verfahren wurden 26 Tafeln erkannt, dazu gab es eine Falsch-Positive. Daraus ergab sich ein *Recall* von 0.743, eine *Precision* von 0.963 und ein *F-Score* von 0.838. Das iterative Verfahren erkannte 25 Tafeln ohne Falsch-Positive. Der *Recall* betrug hier 0.714, die *Precision* 1 und der *F-Score* 0.833. Für die Texterkennung wurde die Whitelist modifiziert, sodass alle Ziffern und alle Großbuchstaben enthalten waren, wie es der Beschriftung der Tafeln entspricht. Der *F-Score* der Zeilenerkennung lag bei

<sup>11</sup>Hier ist technisch gesehen kein unerwünschtes Ergebnis erzielt worden: Auch das Maßband enthält rechteckige Felder mit passendem Seitenverhältnis und Zahlen darauf.

0.86, der der Buchstabenerkennung bei 0.9. Das theoretische Optimum lag bei 0.91. Der *bestguess* lag mit 0.83 deutlich unter diesen Werten.

## **4 Diskussion**

## Literatur

- [Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8:679 – 698.
- [Chen et al., 2004] Chen, X., Yang, J., Zhang, J., and Waibel, A. (2004). Automatic detection and recognition of signs from natural scenes. *IEEE Transactions on Image Processing*, 13(1):87–99.
- [CIPA Standardization Committee, 2020] CIPA Standardization Committee (2020). Exif 2.32 metadata for xmp.
- [Hallale and Salunke, 2013] Hallale, S. B. and Salunke, G. D. (2013). Offline handwritten digit recognition using neural network. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2:4373–4377.
- [Hamad and Kaya, 2016] Hamad, K. and Kaya, M. (2016). A detailed analysis of optical character recognition technology. *International Journal of Applied Mathematics, Electronics and Computers*, 4:244–249.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.
- [Hough, 1962] Hough, P. V. C. (1962). Method and Means for Recognizing Complex Patterns. *U.S. Patent*, (3,069,654).
- [Jaccard, 1902] Jaccard, P. (1902). Tois de distribution florale dans la zone alpine. *Bulletin de la Société Vaudoise des Sciences Naturelles*, (38):72.
- [Klinke, 2017] Klinke, H. (2017). Information retrieval. In Jannidis, F., Kohle, H., and Rehbein, M., editors, *Digital Humanities. Eine Einführung*, pages 268–278. J.B. Metzler Verlag, Stuttgart.
- [OpenCV-Documentation, 2021a] OpenCV-Documentation (2021a). Geometric image transformations.
- [OpenCV-Documentation, 2021b] OpenCV-Documentation (2021b). Structural analysis and shape descriptors.
- [Shah and Gokani, 2014] Shah, J. and Gokani, V. (2014). A simple and effective optical character recognition system for digits recognition using the pixel-contour features and mathematical parameters. *International Journal of Computer Science and Information Technologies*, 5:6827–6830.

- [Siltanen, 2012] Siltanen, S. (2012). *Theory and applications of marker based augmented reality*. PhD thesis.
- [Suzuki and Abe, 1985] Suzuki, S. and Abe, K. (1985). Topological Structural Analysis of Digitized Binary Images by Border Following. *COMPUTER VISI-ON, GRAPHICS, AND IMAGE PROCESSING*, (30):32–46.
- [Tesseract Documentation, 2021] Tesseract Documentation (2021). Improving the quality of the output.
- [Ye et al., 2007] Ye, Q., Jiao, J., Huang, J., and Yu, H. (2007). Text detection and restoration in natural scene images. *Journal Of Visual Communi-cation and Image Representation*, 18:504–513.