**Schnapskönig**

**EDBV WS 2017/2018: AG A2**

Jan Michael Lajarno (01425799)

Andreas Brunner (01429369)

Miran Jank (01526438)

Thorsten Korpitsch (01529243)

Aleksandar Marinkovic (01634028)

4. Januar 2018

**1 Gewählte Problemstellung**

**1.1 Ziel**

Beim Schnappsen treten 2 Spieler gegeneinander an. Zu Beginn bekommt jeder 5 Karten vom Stapel und eine Karte wird aufgedeckt, welche das Adut darrstellt, danach wird jede Runde von beiden Spielern eine Karte gezogen. Jede Runde wird jeweils 1 Karte von jedem Spieler auf den Tisch gelegt wobei beide Spieler versuchen einen Stich zu machen. Stechen bedeutet, dass die eigene Karte entweder ein Adut ist und die Karte des Gegenübers keines ist, beziehungsweise, dass die eigene Karte einen höheren Wert hat als die des Gegenübers. Es wird solange gespielt bis der Stapel leer ist und beide Spieler keine Karten mehr auf der Hand haben, beziehungsweise einer der Spieler Karten im Wert von 66 Punkten gestochen hat. Sind alle Karten ausgespielt worden, wird der Gewinner ermittelt indem die gewonnenen Punkte beider Spieler gezählt werden und danach der mit der höheren Punktezahl gewinnt.

**1.2 Eingabe**

Der Benutzer muss pro Spielzug ein Farbbild, in einem gangigen Format(.PNG/.JPG/.JPEG), der Karten in das Programm laden.

**1.3 Ausgabe**

In der Konsole wird ein Zwischenstand nach jeder Runde ausgegeben, am Ende wird der Gewinner ausgegeben und der Endstand.

**1.4 Voraussetzungen und Bedingungen**

Ein Farbbild der Karten. Der Hintergrund sollte möglichst Einfarbig sein (nicht wei , texturarm). Die Kamera soll sich in einem Winkel von 45 bis 135 Grad befinden. Die Karten müssen mit einem dünnen schwarzen Rand preperiert sein. Die obere Karte darf maximal 45% der anderen Karte überdecken.

**1.5 Methodik**

1. Threshold nach Otsu: Um in Kombination mit Zusammenhangskomponenten die Karten zu trennen
2. Geometrische Transformation: Um das Eingabebild vorzubereiten wird aus dem, bis zu 45 Grad schragen• Bild, ein Bild aus der Vogelperspektive (90 Grad) trans-formiert, die hochkant steht
3. Template-Matching: Um die Spielkarte zu identi zieren

**1.6 Evaluierung**

* Werden beide Karten erkannt?
* Wird das Bild richtig transformiert, oder werden Buchstaben/Symbole verzerrt?
* Wird die Karte richtig identi ziert?

**1.7 Datenbeispiel**



Figure - Karo-Ass sticht Kreuz-König (Farbzwang), 16 Punkte gewonnen.



Figure - Karo-König sticht Karo-Dame, 7 Punkte gewonnen.

**1.8 Zeitplan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Meilenstein | abgeschlossen am | | Arbeitsaufwand in h | |
|  |  |  |  |  |
|  | geplant | tatsächlich | geplant | tatsächlich |
| Vorarbeit Projekt | 20.10.2017 | 21.10.2017 | 20 | 23 |
| Prototyp erstellen | 10.11.2017 | 01.12.2017 | 40 | 80 |
| Geometrische Transformation | 22.12.2017 | 27.12.2017 | 105 | 92 |
| Threshold nach Otsu | 18.12.2017 | 25.12.2017 | 67 | 60 |
| Pattern-Matching | 18.12.2017 | 30.12.2017 | 43 | 30 |
| GT-Test | 31.12.2017 | 31.12.2017 | 10 | 12 |
| TH-Test | 27.12.2017 | 29.12.2017 | 5 | 5 |
| PM-Test | 27.12.2017 | 29.12.2017 | 5 | 3 |
| Unit-Test | 05.01.2017 | 04.01.2017 | 5 | 2 |
|  |  |  |  |  |

**2 Arbeitsteilung**

|  |  |
| --- | --- |
| Name | Tätigkeiten |
| Jan Michael Lajarno | Matlab: Prototyp, ausführbares Programm, |
|  | Bericht: Methodik/Implementierung(Template Matching), |
|  | Gewählte Problemstellung |
| Andreas Brunner | Matlab: Prototyp, tmc.m |
|  | Bericht: Arbeitsteilung, Evaluierung |
|  |  |
| Miran Jank | Matlab: Prototyp, geom\_tranfs\_lowercard.m |
|  | Bericht: Methodik(Geometrische Transformation), Implementierung (Geometrische Transformation) |
|  |  |
| Thorsten Korpitsch | Matlab: Prototyp, splitCards.m, thresholdOtsu.m |
|  | Bericht: Evaluierung, Methodik/Implementierung (Vorverar- |
|  | beitung, Trennen der Karten), Schlusswort |
|  |  |
| Aleksandar Marinkovic | Matlab: Prototyp, gettform.m, geotransform.m, geom\_tranfs\_uppercard.m |
|  | Bericht: Implementierung(Geometrische Transformation) |
|  |  |
|  |  |

**3 Methodik**

*(2-3 Seiten)*

*Hier wird die verwendete Methodik in der Theorie vorgestellt:*

*Welche Methodik wurde verwendet? Warum eignet sich diese Methodik für die gewählte Problemstellung? Habt ihr Methoden verändert (Einschränkungen, Abwandlungen), wenn ja wie? etc.*

*Die erwähnten Methoden werden zum größten Teil auf Beschreibungen in Büchern oder wissenschaftlichen Artikeln beruhen. Daher ist hier auch der richtige Platz für Zitate. Die hier zitierten Publikationen sollten mittels Abkürzung bzw. Nummer referenziert sein und sich in der Referenzliste am Ende des Berichts über diese Bezeichnung finden lassen.*

*Ein Beispielsatz (inkl. entsprechender Literaturangabe am Ende des Berichts): Interest Points wurden mittels Scale Invariant Feature Transform detektiert.*

*Bei der Verwendung von Latex gestaltet sich das Zitieren besonders einfach - siehe Beispielssatz im Source der Latex-Vorlage.*

*Wichtig in diesem Abschnitt ist, dass sich der Leser eures Berichts mit den verwendeten Methodiken auskennen und wissen, weshalb ihr diese Methodiken verwendet habt und keine anderen. Es soll dem Leser helfen den nächsten Abschnitt des Berichts besser zu verstehen.*

**3.1 Threshold nach Otsu**

Die Aufgabe für die erste verwendete Methode ist das Trennen der Karten vom Hintergrund. Hier haben wir uns für ein Schwellenwertverfahren entschieden. Bei einem Schwellenwertvefahren wird ein Graustufenbild in ein Binärbild umgewandelt. Wir haben uns für den Threshold nach Otsu entschieden, da es in der Kombination mit der Zusammenhangskomponente besonders dazu eignet um Objekte vom Hintergrund zu trennen.

Beim Threshold nach Otsu wird versucht, die beiden Segmente (Vordergrund/Hintergrund) so kompakt wie möglich zu machen und die Überschneidung gering zu halten.

Um das Trennen der Karten zu vereinfachen schränken wir die Eingabe dahingehend ein, dass beim Bild der Karten der Hintergrund nicht weiß sein darf und möglichst texturarm. Diese Einschränkungen machen unseren Threshold nach Otsu stabiler und sehr erfolgssicher beim Trennen der Karten vom Hintergrund.

**3.2 Geometrische Transformation**

Um die Karten im finalen Schritt wirklich erkennen zu können, müssen die Bilder vom Blickwinkel der Perspektive in die Orthogonale gebracht werden. Für diese Aufgabe haben wir die Geometrische Transformation gewählt, da wir sowohl perspektivisch verzerren, als auch normale Translationsmatrizen verwenden müssen.

Als Einschränkung wurde hier zwei wesentliche Aspekte getroffen: Das Bild muss in einem Winkel von 65° bis 135° aufgenommen werden, und es darf maximal die Hälfte einer Karte verdeckt werden. Die erste Einschränkung dient der Qualität und Stabilität des Verfahrens, denn wenn eine zu starke Verzerrung durch die Perspektive hervorgerufen wird, so kann sich das orthogonale Bild nur geringer Qualität erfreuen. Dies lässt sich dadurch erklären, dass bei der Transformation der Bildinhalte jeder einzelne Pixel gelesen wird, und an einen neuen Ort übersetzt wird – durch eine große Verzerrung hätten wir somit fehlende Information. Die zweite Einschränkung gilt der unteren Karte und dem Template Matching. Wäre mehr wie die Hälfte bedeckt, so könnte nicht mehr zuverlässig nach übereinstimmenden Bildbereichen gesucht werden, und auch für die Transformation der unteren Karte würden wesentliche Anhaltspunkte fehlen. Somit wurde beschlossen, dass zumindest eine horizontale Hälfte der Spielkarte immer ersichtlich sein muss.

**3.3 Template-Matching**

Um die Karte schlussendlich zu identifizieren haben wir uns für das Template Matching entschieden. Das Template-Matching bietet genau die Lösung zu unserer letzten Problemstellung. Mithilfe des Template-Matching versuchen wir herauszu nden ob es sich bei der Karte um Herz, Pik, Kreuz oder Karo handelt und ob es sich um Ass, König Dame, Bube oder Zehn handelt. Aus der Kombination dieser beiden Symbolen können wir die Karte eindeutig identifizieren.

Beim Template-Matching wird versucht ein Bild beziehungsweise einen Bildausschnitt in einem anderen Bild wiederzufinden. User Template-Matching wird mittels einer Correlation-Matrix realisiert, es wird druch alle Möglichkeiten durchiteriert und das passenste wird der Karte zugewiesen.

**4 Implementierung**

Hier gebt ihr einen Überblick über eure Implementierung:

Wie habt ihr die im vorhergehenden Abschnitt vorgestellte Methodik praktisch umgesetzt? Wie werden die einzelnen Methoden kombiniert (zB. Implementierungspipeline)? Hier ist Platz für Implementierungsdetails wie zum Beispiel gewählte Parameter.

Wie startet der User das Programm? Welche Parameter hat der User zu setzen? Auch in diesem Abschnitt können Referenzen und Zitate notwendig sein.

Wichtig in diesem Abschnitt ist, dass der Leser eures Berichts versteht wie ihr euer Projekt in MATLAB umgesetzt habt um sich auch im Quelltext leichter zurecht finden zu können.

**4.1 Vorverarbeitung**

Zuerst wird der Datensatz geladen und in ein Graustufenbild umgewandelt.

**4.2 Karten trennen**

Der Thershold ist die erste Methodik in der Methodik-Pipeline, in Kombination mit der Zusammenhangskomponente. Der Funktion splitCards.m wird das Eingabebild übergeben. Als erstes wird dieses in ein Graustufenbild umgewandelt und an die Funktion thresholdotsu.m übergeben.

Dort werden zuerst die aufkommenden Grauwerte gezählt mittles der histcounts-Funktion. Danach wird die gewichtete Summe aller Pixel berechnet, indem man die Anzahl der Pixel mit ihrem jeweiligen Grauwert multipliziert. Danach wird in einer for-Schleife durch alle Grauwerte durchiteriert. In jedem Schleifendurchgang werden die Pixel, die den aktuellen Grauwert besitzen, zum Hintergrund dazugerechnet. Der Vorder-grund wird neu berechnet indem von der Summe der Pixel, die Pixel vom Hintergrund abzieht. Danach wird die Gewichtete Summe des Hintergrunds berechnet, analog da-zu, wie am Beginn die gewichtete Summe aller Pixel berechnet wurde. Jetzt können die Durchschnittswerte des Hinter- und Vordergrunds berechnet werden. Als vorletzten Schritt berechnen wir die Between Class Variance.

Als letztes wird überprüft ob die Between Class Variance, die bisher größte ist. Ist das der

Fall, wird der Threshold auf den aktuell iterierten Grauwert gesetzt und die maximale

Between Class Variance auf die gerade berechnete gesetzt. Nach dem durchlaufen der

Schleife, wird das Bild mittels des optimalen Thresholds in ein Binärbild umgewandelt.



Figure - Binärbild mittels Threshold nach Otsu

Auf dieses Binärbild wird die Zusammenhangskomponente angewendet um die Karten zu trennen. Aus den 2 größten Zusammenhangskomponenten werden 2 Bilder erstellt, die die Karten repräsentieren. Bei diesen werden noch die Löcher gefüllt.

Dies passiert, da wir nicht nur die Karten trennen wollen, sondern auch gleichzeitig herausfinden wollen, welche Karte die Obere und welche die Untere ist. Das Ganze berechnen wir ganz einfach indem wir die Fläche beider Karten vergleichen und die Karte mit der größeren Fläche die sein muss, die oben liegt, da sie ja einen Teil der unteren Karte verdeckt. Das Füllen der Löcher funktioniert vor allem bei Bildkarten unterschiedlich gut, wie man im nachfolgenden Bild erkennen kann, das kann zu Problemen führen bei der Erkennung welche Karte die Obere und welche die Untere ist.

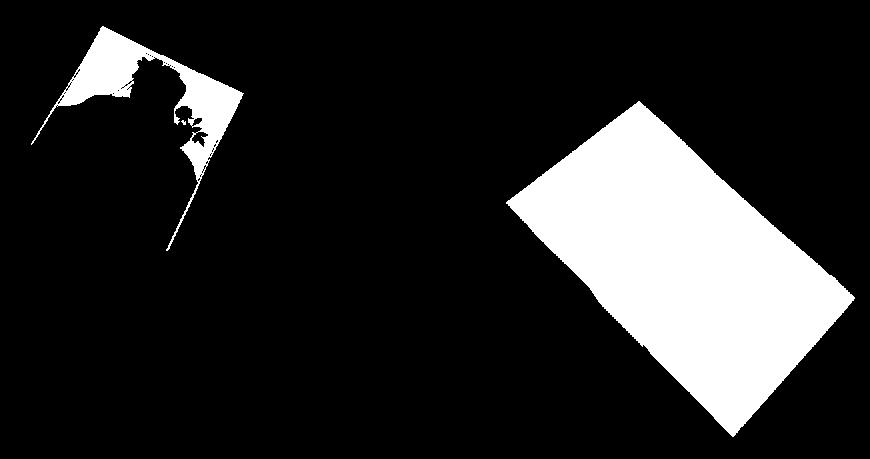


Figure - Binärbilder der getrennten Karten als Montage. Dame(links), König(rechts)

Danach werden beide Karten an die 2. Methodik weitergegeben, die geometrische Transformation.

**4.3 Geometrische Transformation**

Bei der geometrischen Transformation müssen die Karten separat voneinander transformiert werden. Bei der oberen Karte werden zuerst die Ecken mithilfe einer Boundingbox bestimmt. Ein Algorithmus geht die Seiten der Boundingbox entlang und bestimmt die Koordinaten der Ecken.

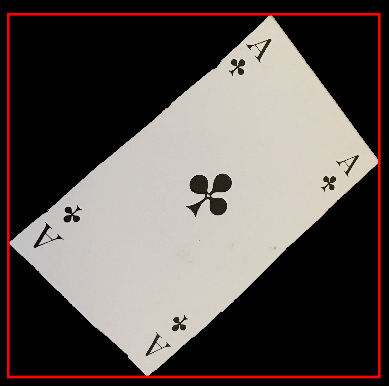
Zusätzlich zu den Eckpunkten werden vier weitere Zielkoordinaten benötigt, eine Spielkarte hat ein Verhältnis von 5:8 daraus bilden wir die Basis für die Zielkoordinaten. Bei der Bestimmung der Zielkoordinaten wird zwischen zwei Fällen unterschieden und zwar ob die obere Ecke in die linke oder rechte Hälfte hin transformiert werden muss.

Figure 5 - rechte Ecke oben

Figure – rechte Ecke oben

Figure 6 - linke Ecke oben

Figure – linke Ecke oben

Um die richtigen Zielkoordinaten zu bestimmen, wird als nächstes die Distanz zwischen der oberen und der linken bzw. der rechten Ecke bestimmt. Da die Karten ein Seitenverhältnis von 5:8 besitzen, lässt sich damit die Orientierung der Karte bestimmen. Wenn die Distanz zwischen der oberen und linken Ecke kürzer als die Distanz zwischen der oberen und rechten Ecke ist (Figure 5) muss die linke Ecke an der Position (0, 0) transformiert werden. Wenn die Distanz länger ist (Figure 6) muss die obere Ecke an die Position

(0, 0) transformiert werden.

Als nächstes werden die Eckpunkte und Zielkoordinaten als Parameter an die Funktion gettform.m übergeben um die Transformationsmatrix zu bestimmen. Die Matrix wird mithilfe des DLT-Algorithmus (Direct Linear Transformation) ermittelt.

Bevor das Bild transformiert wird muss es in einem double Typ umgewandelt werden. Dies ist notwendig da später beim Interpolieren kein uint8 Typ akzeptiert wird.

Der Funktion geotransform.m werden als Parameter das Bild und die Transformationsmatrix übergeben. Falls es sich beim Bild um ein Graustufenbild handelt wird das Bild ohne Probleme transformiert. Da wir jedoch RGB-Bilder als Input haben müssen alle drei Kanäle einzeln transformiert werden.

In geotransform.m werden als erstes die x- und y-Werte festgelegt welche gleich der Anzahl der Spalten / Reihen ist. Diese Werte werden der meshgrid Funktion übergeben, um zwei Matrizen xi und yi zu erstellen. Die xi-Matrix ist eine Kopie der x-Werte welche y-mal wiederholt werden und yi sind die y-Werte x-Mal kopiert. Nun können die Daten mit der Transformationsmatrix multipliziert werden, danach werden die homogenen x- und y-Werte normalisiert. Diese werden als Query-Points für die Interpolation verwendet. Die Query-Points und das Bild werden der interp2-Funktion übergeben und diese führt die Interpolation aus.

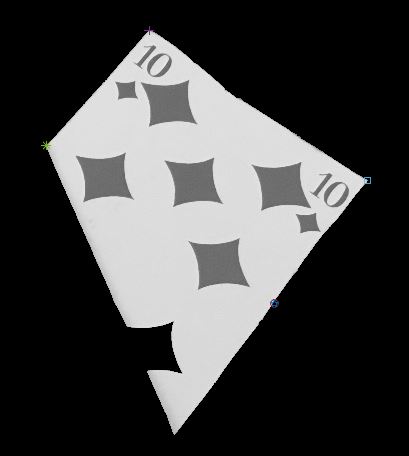


Figure - Transformation vom Kreuz König

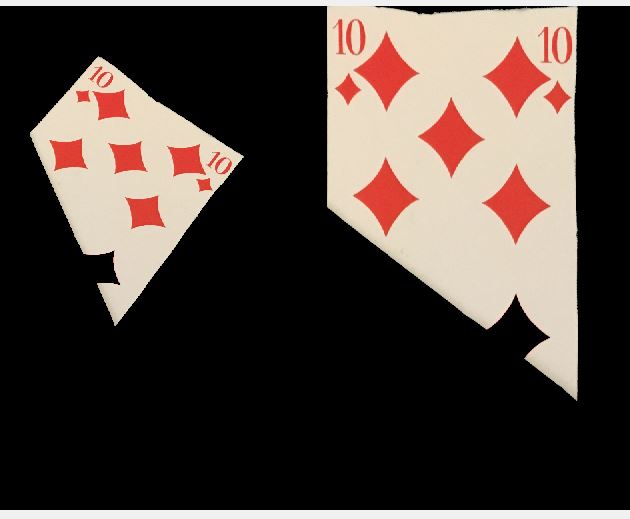
In geom\_transf\_lowercard.m wird genau diese Transformations-Pipeline eingehalten, und somit die untere Karte von Perspektive auf Orthogonal transformiert – Die Bounding Box wird erstellt, Eckpunkte werden berechnet, diese der Transformationsmatrix übergeben, und somit wird ein korrigiertes Bild berechnet.

Ein großer Unterschied besteht allerdings in der Berechnung der Eckpunkte, sowie beim finalen Mapping. Dadurch, dass die untere Karte von der oberen überlappt wird, geht die Information für die vierte (und eventuell sogar dritte) Ecke verloren. Um eine neue Ecke zu erstellen, wird auf das natürliche Seitenverhätnis von Karten zugegriffen.

Da eine volle Karte in der Ratio 8:5 steht, und wir immer (durch unsere vorher getroffene Beschränkung) eine horizontale Hälfte der Karte sehen, erhalten wir somit effektiv sichtbare 4:5. Also können wir die gleiche Transformation anwenden, wenn wir die Zielmatrix mit dem originalen Punkten ([0,0];[0,8];[5,0];[5,8]) auf die neuen Verhältnisse anpassen ([0,0];[0,4];[5,0];[5,4]).



Nun müssen also nur noch die zu transformierenden Ecken gefunden werden – Drei dieser Ecken sollten schon bestehen. Die vierte wurde abgeschnitten, und somit wird sie zum unteren Schnittpunkt der Karte. Somit verbleibt nur noch eine Fehlerhafte Ecke, der obere Schnittpunkt der Karte, angrenzend an die Längste gerade. Diese Gerade wird gekürzt, bis sie gleich lang wie die kürzeste ist. Somit entsteht an diesem Punkt die neue vierte Ecke, mit Hilfe welcher Transformiert wird, um ein orthogonales Bild zu erhalten.



**5 Evaluierung**

*(2-X Seiten)*

*Hier stellt ihr euren Datensatz vor und beantwortet Evaluierungsfragen: z.B. Fakten zum Datensatz: Anzahl der Bilder, Größe der Bilder, Quelle des Datensatzes (falls selbst aufgenommen: Aufnahmegerät, Einstellungen,... / falls nicht selbst erstellt: Datenbank vostellen... ! Referenzen!)*

*Diskussion der Evaluierungsfragen: Beantwortung der Fragen, Diskussion anhand von Beispielen, Diskussion von Grenzfällen: für welche Bilder funktioniert die Implementierung für welche nicht? Worin unterscheiden sich diese Bilder? Warum funktionieren sie nicht? etc.*

*Evaluiert wird der ganze Datensatz, nicht nur einzelne Bilder. Einzelne Bilder können zum Aufzeigen von Fehlern/Problemen/besonders guten Ergebnissen... genutzt werden. Zur Evaluierung gehört auch das Testen der einzelnen Methodiken (separat), mit Erwähnung eventueller Einschränkungen.*

Die Datensätze für unser Projekt lassen sich grundsätzlich in zwei Kategorien unterteilen. ~~Auf der einen Seite gibt es die Templates, die wir für das Template-Matching benötigen, damit wir entscheiden können, um welche Karte es sich handelt. Dieser Datensatz umfasst genau 20 Bilder, da es beim Schnapsen insgesamt 20 Karten im Deck gibt.~~

Der erste Art der Datensätze sind die Templates, die wir für das Template-Matching einsetzen. Sie umfassen genau 4 Symbole und 5 Alphanumerische Zeichen, welche die Kartentypen Pik, Herz, Karo, Kreuz und den Kartenwert Ass, König, Dame, Bub, 10 beschreiben. Es wäre hier möglich für ein anderes Kartenset, welches sich in der Symbolik unterscheidet, neue Templates zu erstellen um unser Programm auch bei diesem Kartenset einzusetzen.

Die zweite Kategorie der Datensätze bilden dann die Spielverläufe selbst. Grundsätzlich wird nach jedem gespielten Zug ein Foto gemacht, welches dann als Input für das Programm dient. Zu dem verwendeten Aufnahmegerät und den Eigenschaften der Bilder kommen wir später. Der Datensatz der Spielverläufe hat keine feste Anzahl an Bildern und kann somit beliebig groß werden.

Wir haben für das Testen unseres Programms insgesamt drei Datensätze zur Verfügung gestellt. Jeder Datensatz unterscheidet sich vom anderen durch die Anzahl der gespielten Züge und auch in den gespielten Karten beider Spieler, um möglichst viele verschiedene Karten-Kombinationen abzudecken. Der erste Datensatz umfasst 7 Bilder, der zweite 8 Bilder und der dritte Datensatz umfasst 10 Bilder.



Figure - Beispiel eines gespielten Zugs - König sticht Dame

Für die Bilder wird kein spezielles Equipment benötigt. Alle Bilder unserer Datensätze wurden mit einer herkömmlichen Smartphone-Kamera aufgenommen. Die Bilder werden anschließend noch skaliert, um die Laufzeit des Programms etwas zu verbessern.

Die Bilder haben eine Auflösung von 2016x1512 und wurden genau um die Hälfte skaliert, um die Bildgröße unter 1MB zu halten. Es werden je Durchlauf des Programms relativ viele Bildoperationen angewendet, die durchaus rechenintensiv sind. Mit dieser Bildgröße kann die Laufzeit des Programms relativ gut in einen annehmbaren Bereich gebracht werden.

Der Hintergrund muss einfärbig und so dunkel wie möglich sein, damit man mit unseren angewendeten Algorithmen die Karten verarbeiten kann. In unseren Datensätzen wird ein dunkler grüner Hintergrund verwendet, ~~da die Karten nichts Grünes enthalten und der Hintergrund bei der Umwandlung der Bilder zu Binärbilder zu Schwarz wird. Somit können die Karten perfekt vom Hintergrund getrennt werden.~~ dass ermöglicht eine fehlerfrei Trennung der beiden Karten vom Hintergrund. Hier sei angemerkt, dass der dunkle Hintergrund eine Vorveraussetzung ist.

* 1. **Evaluierungsfragen**
     1. **Werden beide Karten erkannt?**

Nach zahlreichen Testdurchläufen werden beide Karten - also obere und untere Karte - in allen Fällen erkannt und voneinander getrennt. Die beiden Karten werden nach dem Trennen als separate Bilder gespeichert. Dies kann beispielsweise wie folgt aussehen.

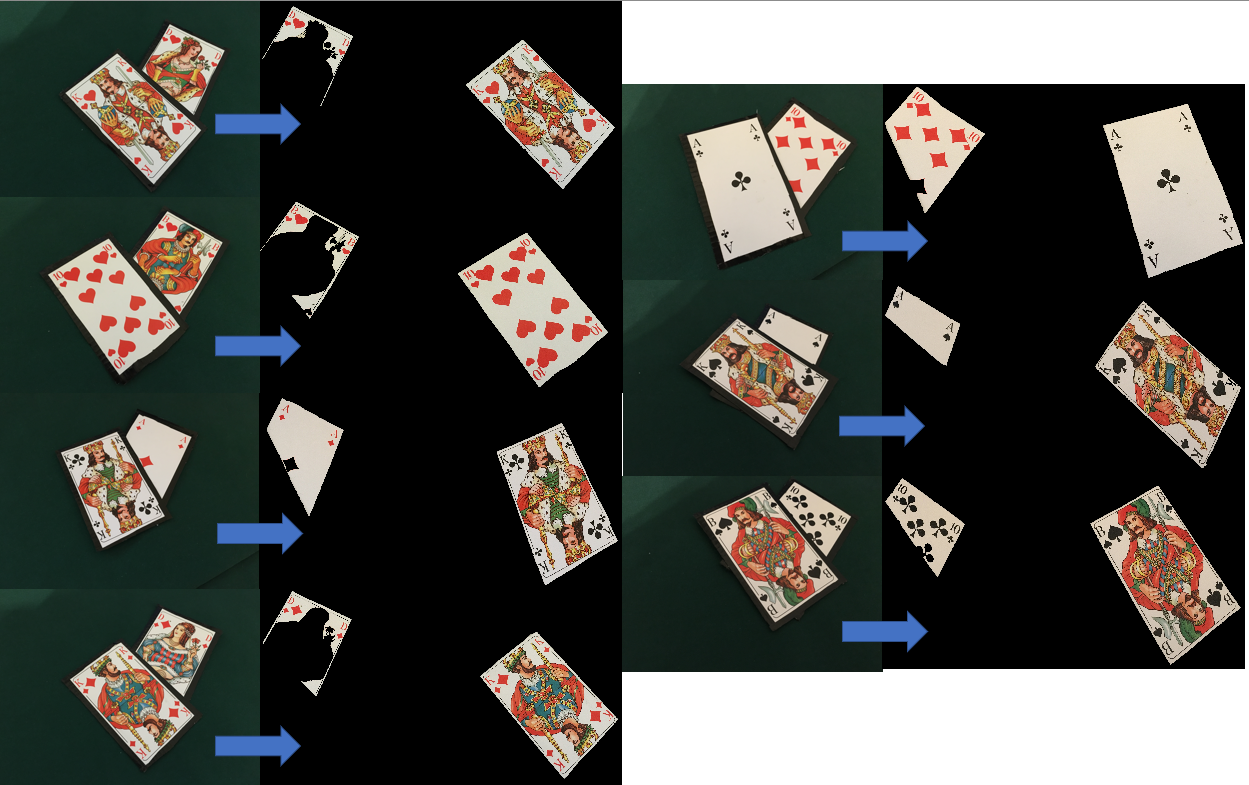


Figure - Kartentrennung - Datensatz 1

Bei diesem Beispielbild ist der Weißanteil der unteren Karte größer und wird somit als obere Karte erkannt. Diesem Fehler kann zwar entgegengewirkt werden

In Abbildung 4 ist der Output nach der Kartentrennung eines kompletten Spiels zu sehen. Auf dem Output-Bild befindet sich auf der rechten Seite immer die vordere Karte und auf der linken Seite die untere Karte. Einige Probleme gab es mit den Karten "10" und "Ass". Diese enthalten keine Texturen, daher überwiegt der Weißanteil und wird auch dann als vordere Karte erkannt, wenn sie unten liegt. Da die Karten nach dem Weißanteil getrennt werden (obere Karte hat im Normalfall immer eine größere weiße Fläche als die untere), mussten hier einige Änderungen vorgenommen werden. Wenn eine der beiden Karten "10" oder "Ass" unten liegen, so muss mehr als 45 Prozent der Karte verdeckt sein, damit sie als untere erkannt wird. Ein gescheitertes Beispiel schaut folgendermaßen aus.



Figure - Fehlerhafte Kartentrennung

* + 1. **Wird das Bild richtig transformiert, oder werden Buchstaben/Symbole verzerrt?**
    2. **Wird die Karte richtig identifiziert?**

Mit Hilfe einer Korrelationsberechnung werden 100 Prozent der Karten, die oben liegen, identifiziert. Der Template-Datensatz sieht wie folgt aus.



Figure - Templates für die Identifizierung der Karten

In Abbildung 8 sind alle Templates zu sehen, die für die Identifizierung der Karte benötigt werden. Die Identifizierung wird in zwei Schritte aufgesplittet. Im ersten Schritt wird das Symbol erkannt (untere Zeile Abbildung 7) und im zweiten Schritt wird anschließend der Buchstabe beziehungsweise die Ziffer (obere Zeile Abbildung 7) erkannt.

Die identifizierten Eigenschaften der Karten werden anschließend auf der Konsole ausgegeben. Ein Beispiel-Output für den ersten Datensatz sieht wie folgt aus.



Figure - Erfolgreiche Identifizierung aller Karten

Anfangs gab es große Schwierigkeiten bei der Unterscheidung zwischen Bube und Dame beziehungsweise Herz und Pik, da beide Templates jeweils sehr ident sind. Dieses Problem wurde mit einer Skalierung und mit verbesserten Templates, die direkt aus dem Output der geometrischen Transformation extrahiert wurde, gelöst.



Figure - Idente Templates (Bube, Dame / Herz/Pik)

**6 Schlusswort**

Unsere Schlussfolgerung ist, dass der Anfang des Projekts, also das nden geeigneter Methoden um unsere Problem zu lösen, beziehungsweise einen ersten Prototypen zu entwickeln der schwierigste Schritt für uns waren. Insbesondere das nden der Karten auf dem Bild, bzw. die Findung der Kanten und Ecken um eine Geometrische Trans-formation im naechsten Schritt zu ermoeglichen. Sobald das erledigt war und wir beim Prototyping eine Methode (Canny-Edge-Detection) gegen eine andere ausgetauscht hatten (Threshold nach Otsu) ging es sehr zuegig voran und wir konnten die Zeit die wir aufholen, welche wir beim Prototyping verloren hatten.

Zurzeit gibt es noch Probleme, wenn der Hintergrund zu hell ist, werden die Karten teil-weise nicht getrennt, da unser Programm die Karten mittels Threshold nach Otsu und der Zusammenhangskomponente trennt. Wird dann der Hintergrund als Vordergrund erkannt und ist eine größere Zusammenhangskomponente als die kleinere Karte, wird diese nicht mehr erkannt und stattdessen wird der Hintergrund als 2. Karte erkannt.  
  
Ein weiteres Problem bildet das Template-Matching bei der unteren Karte. Bei der oberen Karte, machen wir uns die Eigenschaften einer Spielkarte zu Nutze, da sie Achsensymetrisch ist und beschränken uns beim Matching auf das erste ¼ der Karte. Dies hält die Laufzeit sehr niedrig, schwächt aber nicht die Erfolgsrate. Bei der unteren Karte haben wir es nicht geschafft sicherzustellen, dass eine „nicht-abgeschnittene“ Ecke immer an der selben Stelle ist, und müssen daher beim Matching die ¾ der Karte überprüfen, dass die Laufzeit erhöht.

Eine grosse Verbesserung unserer Loesung wäre, wenn die Karten nicht mit einem schwarzen Rand präpariert werden müssten. Beziehungsweise wenn man einen Schritt weiter geht, dass nicht ein bestimmtes Kartendeck verwendet werden muss, sondern ein beliebiges (Kartedecks unterscheiden sich oft in der Symbolik, was bei uns dazu fuehrt, dass die Karten nicht mehr richtig identi ziert werden).

**Literatur**