Höhere Technische Bundes-Lehr- und Versuchsanstalt Villach

Abteilung für Informatik

**Diplomarbeit**

Nadelkarte-Bauteilerkennung





Eingereicht am d. m. yyyy von

Betreut und beurteilt von

Prof. Dipl.-Ing. Help Me Technikus

**Judth Marcel 5AHIFS**

# Eidesstattliche Erklärung

Wir versichern, dass wir diese Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt haben. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen haben wir alle gekennzeichnet und im Literaturverzeichnis angeführt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Diplomandin 1, geb. am 32.12.1999

Villach, am d. m. yyyy

# Danksagung

Ich bedanke mich ...

# Abstract

Our college thesis …

# Kurzfassung

Diese Diplomarbeit beschreibt ... (mindestens 2 Abschnitte über zirka eine ¾ Seite)

# Inhaltsverzeichnis

[I. Eidesstattliche Erklärung 3](#_Toc523892741)

[II. Danksagung 4](#_Toc523892742)

[III. Abstract 5](#_Toc523892743)

[IV. Kurzfassung 6](#_Toc523892744)

[V. Inhaltsverzeichnis 7](#_Toc523892745)

[VI. Abbildungsverzeichnis 8](#_Toc523892746)

[VII. Quellcodeverzeichnis 9](#_Toc523892747)

[1. Einleitung und Überblick 10](#_Toc523892748)

[2. Technologien 11](#_Toc523892749)

[1.1 MATLAB 11](#_Toc523892750)

[1.1.1 Allgemein 11](#_Toc523892751)

[1.1.1.1 Erstellen einfacher Variablen: 11](#_Toc523892752)

[1.1.1.2 Vektor 11](#_Toc523892753)

[1.1.1.3 Matrizen 11](#_Toc523892754)

[1.1.1.4 Multiplizieren von Matrizen: 12](#_Toc523892755)

[1.1.1.5 Zusammenführen von 2 Arrays: 13](#_Toc523892756)

[1.1.1.6 Einzelne Werte auslesen: 13](#_Toc523892757)

[1.1.1.7 Speichern und Laden: 14](#_Toc523892758)

[1.1.1.8 Array-Funktionen 14](#_Toc523892759)

[1.1.1.9 2D-Plot 15](#_Toc523892760)

[1.1.1.10 3D-Plots 16](#_Toc523892761)

[1.1.1.11 Programm: 17](#_Toc523892762)

[1.1.1.12 Schleifen 17](#_Toc523892763)

[1.1.1.12.1 For-Schleife: 17](#_Toc523892764)

[1.1.1.13 Funktionen 17](#_Toc523892765)

[1.1.1.14 Bildverarbeitung 17](#_Toc523892766)

[VIII. Nadelkarte Bildvergleich 19](#_Toc523892767)

[1.2 Umwandeln in einen geeigneten Farbbereich 19](#_Toc523892768)

[1.3 Rotation der Nadelkarte 21](#_Toc523892769)

[1.4 Überlagerung von Bauteilplan 22](#_Toc523892770)

[1.5 Differenz der Nadelkarten-Bilder: 24](#_Toc523892771)

[1.6 Erstellung einer Vorlage 25](#_Toc523892772)

[1.7 Erkennung der Bauteile 26](#_Toc523892773)

[3. Probleme 27](#_Toc523892774)

[4. Ablauf 27](#_Toc523892775)

[IX. Quellen: 30](#_Toc523892776)

[5. Resümee 31](#_Toc523892777)

[6. Anhang 32](#_Toc523892778)

[1.8 Literaturverzeichnis 32](#_Toc523892779)

[1.9 Anleitungen / HowTos 33](#_Toc523892780)

[1.10 Beispiele / Sources 34](#_Toc523892781)

[1.11 Projekt- / Source-Verweise 35](#_Toc523892782)

# Abbildungsverzeichnis

1. MATLAB default Layout………………………………………………x
2. MATLAB workspace whos…………………………………………….x
3. MATLAB 2d-Plot normal…………………………………………….x
4. MATLAB 2d-Plot personalisiert……………………………………….x
5. MATLAB 3d-Plot……………………………………………………..x
6. MATLAB Trasholder LAB Beispiel………………………………….x
7. Bauteilplan Beispiel…………………………………………………..x
8. Überlappung: Ausrichtung der Nadelkarte…………………………..x
9. Differenz-Vergleich RGB-LAB………………………………………x
10. Vorlage für NK………………………………………………………..x

# Quellcodeverzeichnis

Für Quellcodes, Tabellen etc. lassen sich jeweils eigene Verzeichnisse anlegen.

# Einleitung und Überblick

Hier steht ein einleitender Text zur Arbeit (wahlweise persönlicher Zugang, Begriffserklärungen, These dieser Arbeit, Geschichte des Themas, Problemstellung, aktueller Stand).

Der zweite Teil dieses Kapitels verschafft dem Leser[[1]](#footnote-1) einen Überblick, was welches Kapitel beschreibt. Damit findet er rascher jene Kapitel, die ihn interessieren oder die er überspringen kann.

# Technologien

## MATLAB

### Allgemein

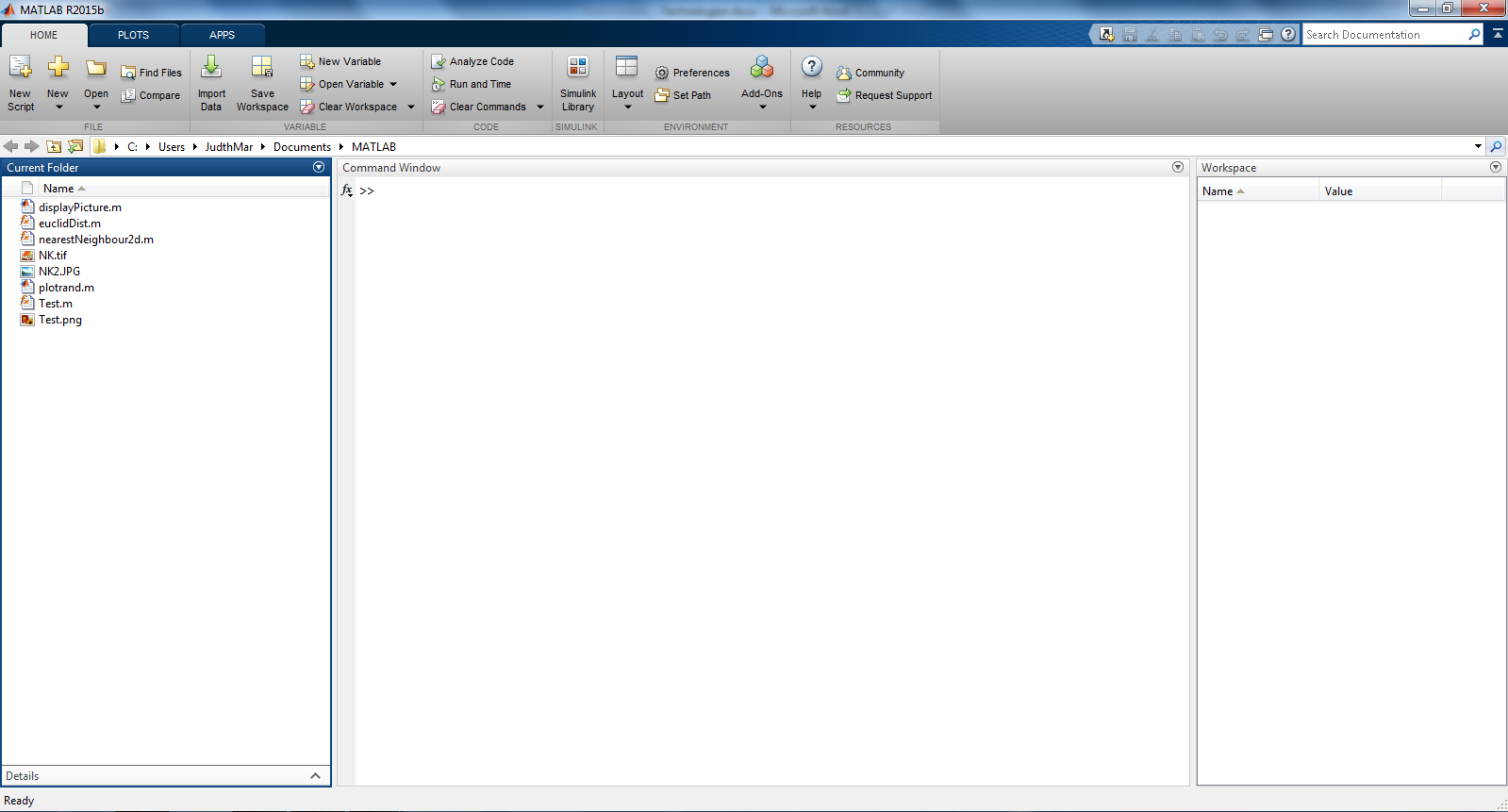


Abbildung 1: MATLAB default Layout

MATLAB enthält ein Command Window wo alle Befehle eingegeben werden und einen Workspace wo alle Variablen angezeigt werden. Außerdem befindet sich auf der linken Seit der Current Folder, wo alle Dateien angezeigt werden.

#### Erstellen einfacher Variablen:

>>a = 3

Es wird automatisch a = 3 ausgegeben und um die automatische Ausgabe zu verhindern muss man ein Semikolon (‚;‘) einfügen.

Beispiel:

>>a=3;

🡪 Keine Ausgabe

>>b=4;

>>c=a+b

Ausgabe: c=7

#### Vektor

Ein Vektor kann folgendermaßen realisiert werden:

>>arr=[1 2 3 4 5]

arr = a = *1×4*

1 2 3 4

#### Matrizen

>>mat=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]

mat = 4×4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Sinus berechnen in einer Matrix:

>>sin(mat)

ans =

0.8415 0.9093 0.1411 -0.7568

-0.9589 -0.2794 0.6570 0.9894

0.4121 -0.5440 -1.0000 -0.5366

0.4202 0.9906 0.6503 -0.2879

Ausgabe einer Matrix mit „‘ “:

>> mat'

ans =

1 5 9 13

2 6 10 14

3 7 11 15

4 8 12 16

Jeden Wert der Matrix um eines erhöhen

>> mat + 10

ans =

11 12 13 14

15 16 17 18

19 20 21 22

23 24 25 26

#### Multiplizieren von Matrizen:

>> erg = mat.\*mat

erg =

1 4 9 16

25 36 49 64

81 100 121 144

169 196 225 256

>> erg = mat \* inv(mat)

erg =

2 0 0 0

0 -4 0 8

0 -8 -4 0

8 -8 0 16

Diese Matrix wird statt Integer als floating-point value gespeichert. Mit „format“ kann der gewünschte Datentyp ausgewählt werden.

>> format long

>> P=mat \* inv(mat)

P =

2 0 0 0

0 -4 0 8

0 -8 -4 0

8 -8 0 16

#### Zusammenführen von 2 Arrays:

* Vertikal anhängen:

>>mat2=[mat,mat]

* Senkrecht anhängen:

>>mat2=[mat;mat]

Mit magic(n) wird eine Matrix erstellt die die Werte 1 bis n2 beinhaltet und so angeordnet sind das jede Reihe, jede Spalte und die ersten Diagonalen dieselbe Summe bilden.

>>magic(3)

Mat=3x3

8 1 6

3 5 7

4 9 2

#### Einzelne Werte auslesen:

>>Mat(8)

ans = 4

>>Mat(3,2)

ans = 9

>>Mat(1:2,2)

ans = 2x1

1

5

>>Mat(: , 2)

ans = 3x1

1

5

9

Mit „whos“ kann der Inhalt vom Workspace angezeigt werden und dadurch wird es im „Workspace-Pane“ angezeigt.

>> whos

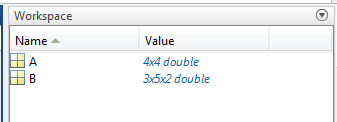


Abbildung 2: MATLAB Workspace whos

#### Speichern und Laden:

Mit „save“ wird der Workspace im aktuellen Ordner gespeichert und kann wieder mit „load“ geladen werden.

>>save myFile.date

>>load myFile.date

Characters:

>>text = ‚Hello‘

text = ‚Hello‘

>>text2 = ‚World‘

text2 = ‘World’

>>comText = [text, '-', text2]

comText = ‘Hello-World’

Mit num2str(number) werden Nummern in Characters umgewandelt.

>> infoText = ['I am ', num2str(17), ' years old']

infoText = ‘I am 17 years old’

#### Array-Funktionen

Das Maximum eines Vektors wird mit max(vektor) gezeigt.

>>arr1 = [2 4 5 1]

>>arr2 = [1 6 3 2]

>>max(arr1)

ans = 5

>>max(arr1, arr2)

ans = 2 6 5 2

>> [max,location] = max(arr1)

max = 5

Location = 3

#### 2D-Plot

Plots werden mit der „plot“ Fuktion gezeichnet. Dieser wird der X-Wert und der Y-Wert übergeben. Dieses Plot wird danach grafisch dargestellt.

>> x = 0:pi/100:2\*pi;

>> y = sin(x);

>>plot(x,y)

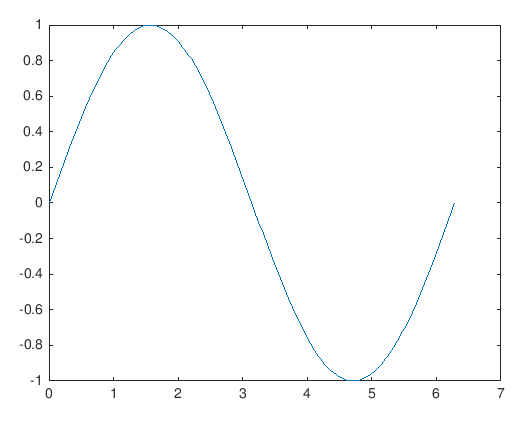


Abbildung 3: MATLAB 2d-Plot Normal

Außerdem kann die Linienart mit einem dritten Parameter verändert werden.

>> plot(x,y, 'r--')

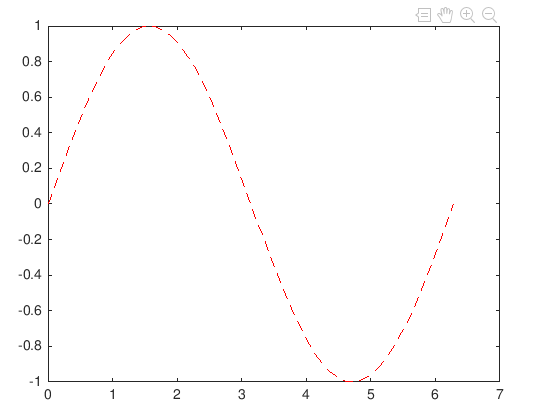


Abbildung 4: MATLAB 2d-Plot Normal

#### 3D-Plots

>> [X,Y] = meshgrid(-2:.2:2);

>>Z = X .\* exp(-X.^2 - Y.^2);

>>surf(X, Y, Z)

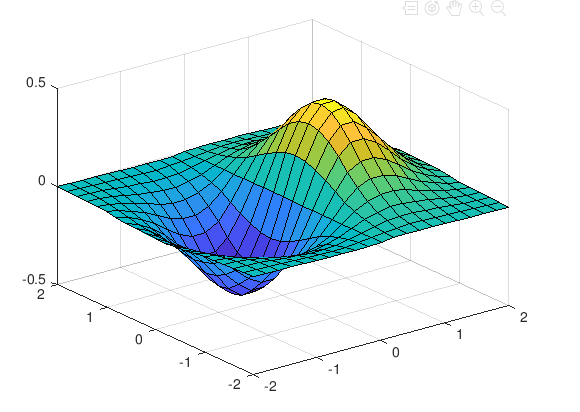


Abbildung 5: MATLAB 3d-Plot

#### Programm:

Um ein neues Script zu erstellen wird der Befehl „edit“ verwendet. Dieses Script kann über den Run-Button oder über die Kommando-Zeile ausgeführt werden.

#### Schleifen

##### For-Schleife:

Beispiel:

for i = 1:10

X = [num2str(i),'. Durchgang'];

disp(X);

end

#### Funktionen

Neue Funktionen werden durch einen Rechtsklick im „Current Folder“ 🡪 New File 🡪 Function erstellt.

function y = Test(x)

y = x + 100;

end

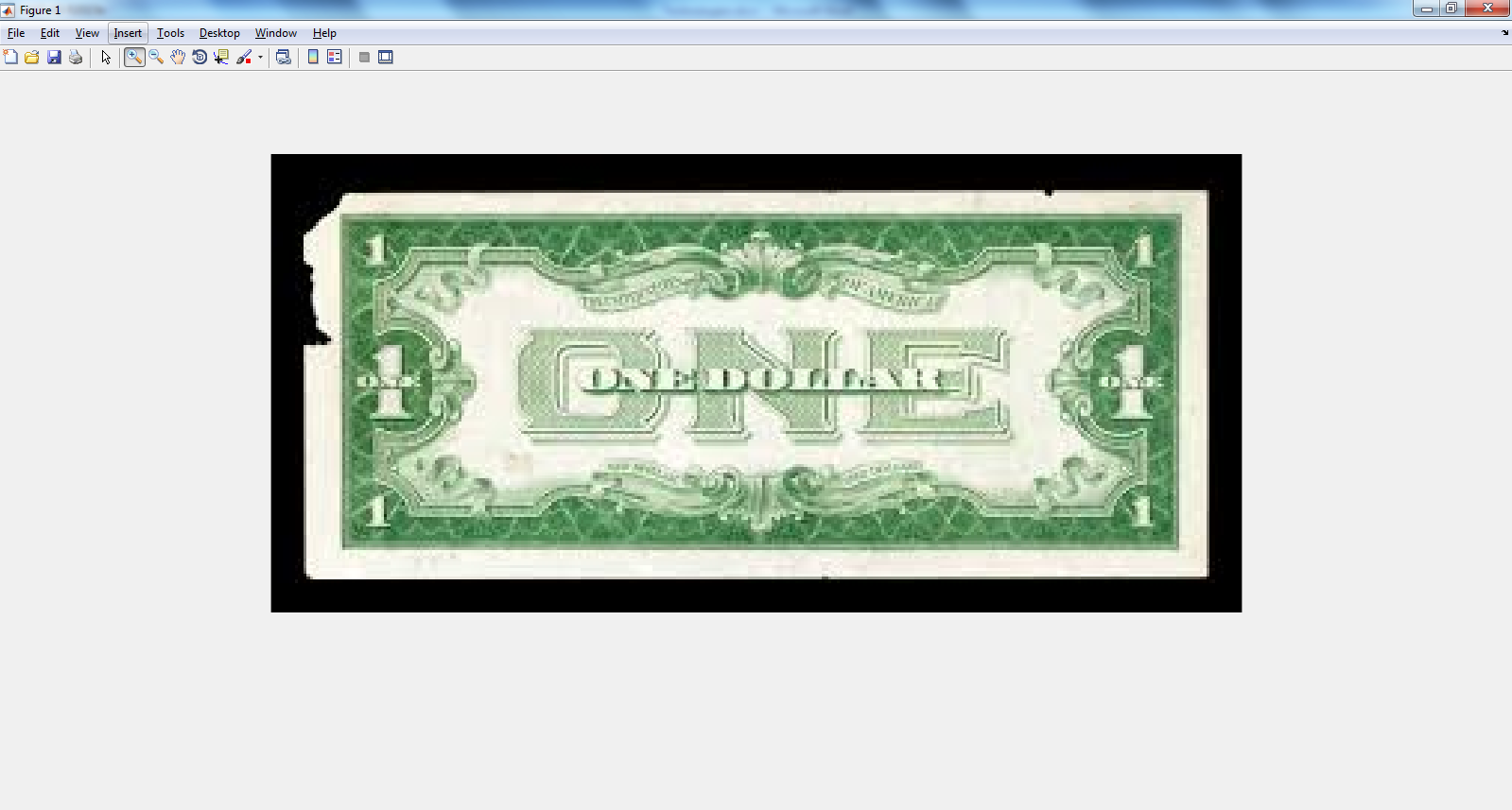
#### Bildverarbeitung

Um ein Bild einzulesen wird die Funktion „imread“ verwendet und um dieses Bild danach auszugeben, wird „imshow“ verwendet. Beispiel:

img = imread('Test.png');

imshow(img);

Ausgabe:



Dadurch dass das Bild in eine Matrix gespeichert wird, kann man mit der Funktion „rgb2lab“ in verschiedene Graustufen gelapped werden. Dadurch könne n die Bilder besser verglichen werden.

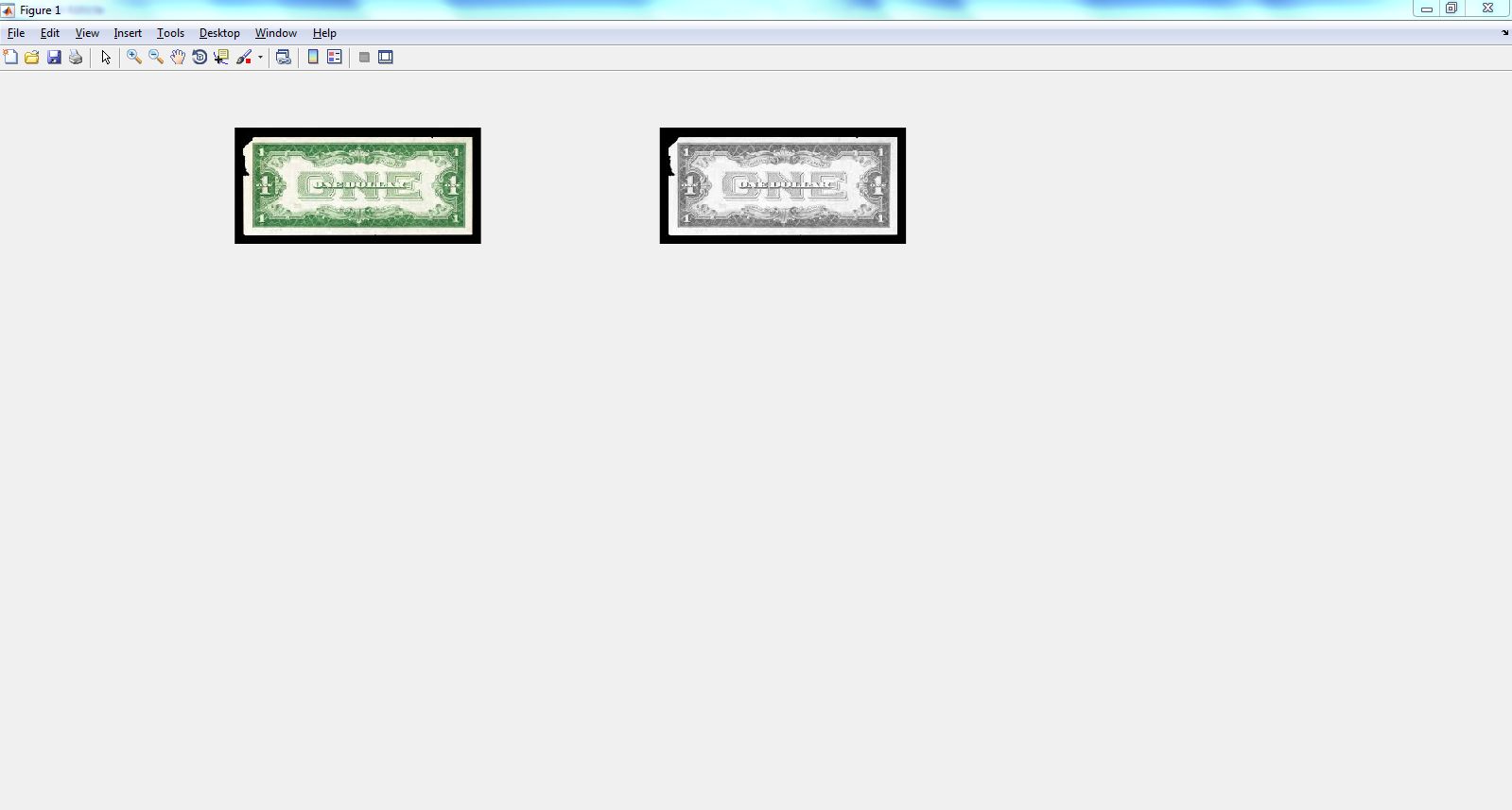
labimg = rgb2lab(img);

subplot(4,3,2);

imshow(labimg(:, :, 1), []);

Mit subplot können mehrere Bilder nebeneinander angezeigt werden. Der erste Parameter beschreibt die Anzahl der Bilder in einer Reihe und der zweite Parameter ist die Anzahl der Reihen. Dadurch können Bilder schnell verglichen werden.

Ausgabe:



Nach diesem wichtigen ersten Schritt wird das Bild klassifiziert und dafür werden die Koordinaten für die einzelnen Klassen benötigt. In diesem Beispiel gibt es einen Hintergrund, den grünen Farbbereich und den weißen Farbbereich. Durch die Koordinaten dieser Bereiche werden die Pixelwerte aus dem neu erstelltem Bild entnommen und in einem mehrdimensionalen Array abgespeichert.

white = labimg(72:82, 51:62, :);

gray = labimg(30:31, 88:91, :);

bgrd = labimg(5:10, 49:52, :);

classes = [[mean2(white(:, :, 2)), mean2(white(:, :, 3))]', [mean2(gray(:, :, 2)), mean2(gray(:, :, 3))]', [mean2(bgrd(:, :, 2)), mean2(bgrd(:, :, 3))]'];

Mit diesem Array „classes” wird im labimg überprüft, zu welcher Klasse jedes Pixel gehört und daraufhin werden die zugeteilten Klassen in ein neues Array gespeichert.

[height, width, channels] = size(labimg);

for i = 1:height

for j = 1:width

x = labimg(i, j, 2:3);

classimg(i, j) = nearestNeighbour2d(labimg(i, j, 2:3), classes);

end

end

function y = nearestNeighbour2d(x, classes)

dists = zeros(size(classes(1, :)));

for i = 1: size(classes(1, :), 2)

dists(i) = euclidDist(x(1), classes(1, i) + euclidDist(x(2), classes(2, i)));

end

[~, y] = ismember(min(dists), dists);

end

function d = euclidDist( x, y )

d = (x - y) ^2;

end

# Nadelkarte Bildvergleich

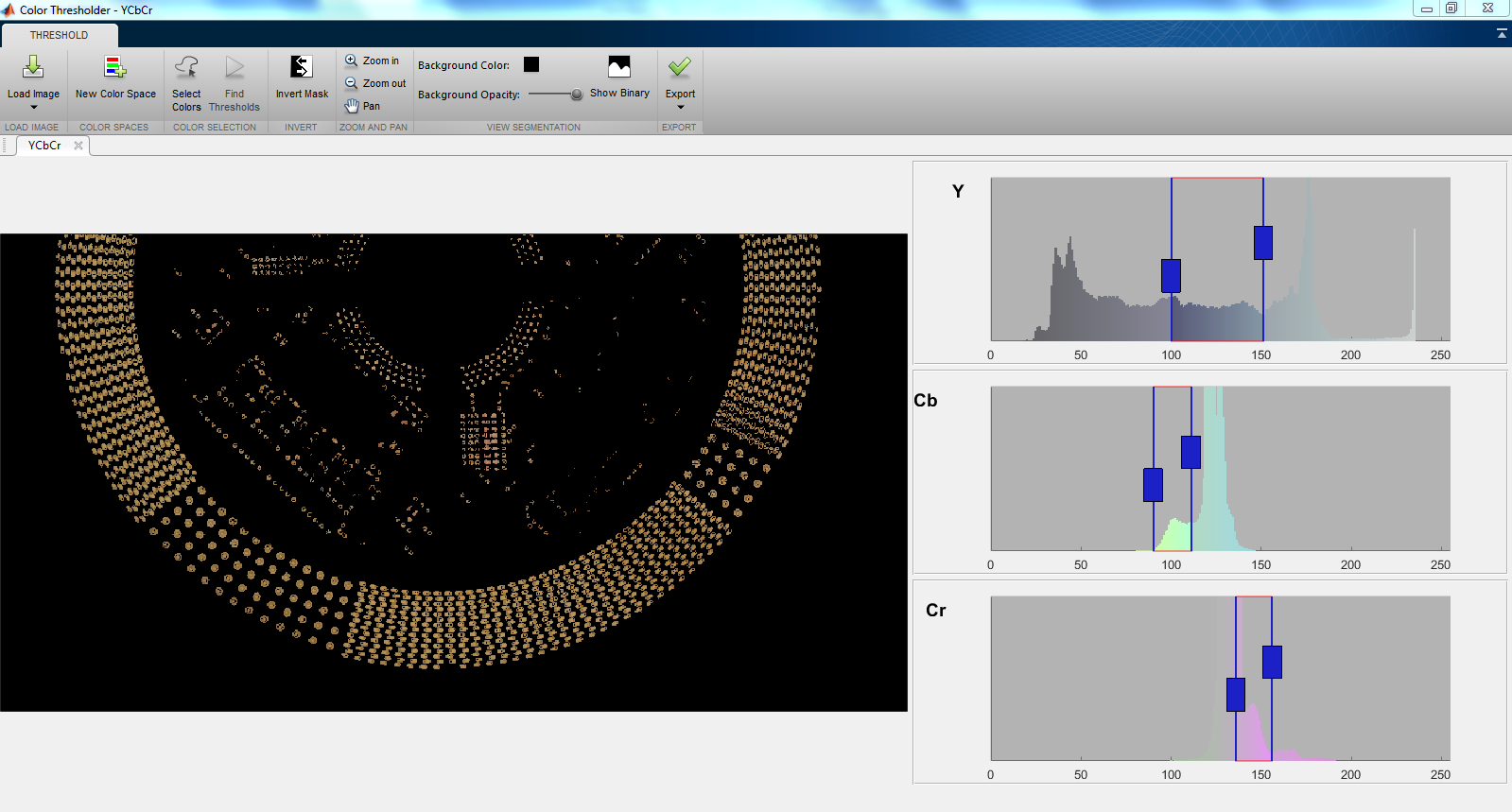
Nach dem Erstellen eines Bildes einer Nadelkarte, soll durch ein Referenzbild und einer Vorlage bestimmt werden, ob auf dieser Nadelkarte Bauteile fehlen und wenn Bauteile fehlen, soll der Name des Bauteiles/der Bauteile bestimmt werden.

## Umwandeln in einen geeigneten Farbbereich

Auf einem normalen Bild einer Nadelkarte, sind die Bauteile mit bloßem Auge gut erkennbar, aber schwer zu erkennen aus der Sicht von Matlab. Darum ist es von Vorteil, das Nadelkarten-Bild mit einem anderen Farbraum zu bearbeiten, sodass nur mehr die Anschlüsse ersichtlich sind.

MATLAB besitzt ein Tool namens “Color Tresholder”, in dem Bilder in verschiedenen Farbbereiche bearbeitet werden können. Außerdem kann der Code vom bearbeiteten Bild exportiert werden. Die Farben „Y“, „Cb“ und „Cr“ müssen so eingestellt werden, dass die restlichen Bauteile verschwinden und die Anschlüsse noch zu sehen sind.

Beispiel für exportierten Code:



Durch das Klicken auf das Export-Symbol, kann dieser Code generiert werden:

function [BW,maskedRGBImage] = createMask(RGB)

%createMask Threshold RGB image using auto-generated code from colorThresholder app.

% [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using

% auto-generated code from the colorThresholder App. The colorspace and

% minimum/maximum values for each channel of the colorspace were set in the

% App and result in a binary mask BW and a composite image maskedRGBImage,

% which shows the original RGB image values under the mask BW.

% Auto-generated by colorThresholder app on 20-Jul-2018

%------------------------------------------------------

% Convert RGB image to chosen color space

I = rgb2ycbcr(RGB);

% Define thresholds for channel 1 based on histogram settings

channel1Min = 100.000;

channel1Max = 151.000;

% Define thresholds for channel 2 based on histogram settings

channel2Min = 90.000;

channel2Max = 111.000;

% Define thresholds for channel 3 based on histogram settings

channel3Min = 136.000;

channel3Max = 156.000;

% Create mask based on chosen histogram thresholds

BW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...

(I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...

(I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);

% Initialize output masked image based on input image.

maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.

maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

Dadurch dass nun nur mehr die Stecker erkennbar sind, kann das Referenzbild viel einfacher mit dem aktuellem Zustandsbild verglichen werden.

## Rotation der Nadelkarte

Da nicht alle Nadelkarten gleich fotografiert werden und die Bilder leicht rotiert sein können, muss dies über MATLAB ausgeglichen werden. Es werden beide Bilder verglichen und daraufhin gemeinsame Punkte erkannt und eingezeichnet. Durch diese Punkte kann darauf berechnet werden, um wie viel Grad die Nadelkarte zu drehen ist. Daraufhin wird das Bild gedreht und wieder dem Main Script zurückgegeben.

function [ rotatedImage ] = rotateImage( original, distorted )

ptsOriginal = detectSURFFeatures(original);

ptsDistorted = detectSURFFeatures(distorted);

[featuresOriginal, validPtsOriginal] = extractFeatures(original, ptsOriginal);

[featuresDistorted, validPtsDistorted] = extractFeatures(distorted, ptsDistorted);

clear ptsOriginal;

clear ptsDistorted;

indexPairs = matchFeatures(featuresOriginal, featuresDistorted);

matchedOriginal = validPtsOriginal(indexPairs(:,1));

matchedDistorted = validPtsDistorted(indexPairs(:,2));

[tform, ~, ~] = estimateGeometricTransform(matchedDistorted, matchedOriginal, 'similarity');

outputView = imref2d(size(distorted));

recovered = imwarp(distorted,tform,'OutputView',outputView);

rotatedImage = recovered;

end

## Überlagerung von Bauteilplan

Ein Bauteilplan wird mit den Koordinaten, die in einer CSV-Datei abgespeichert sind, gezeichnet und sieht wie folgt aus:

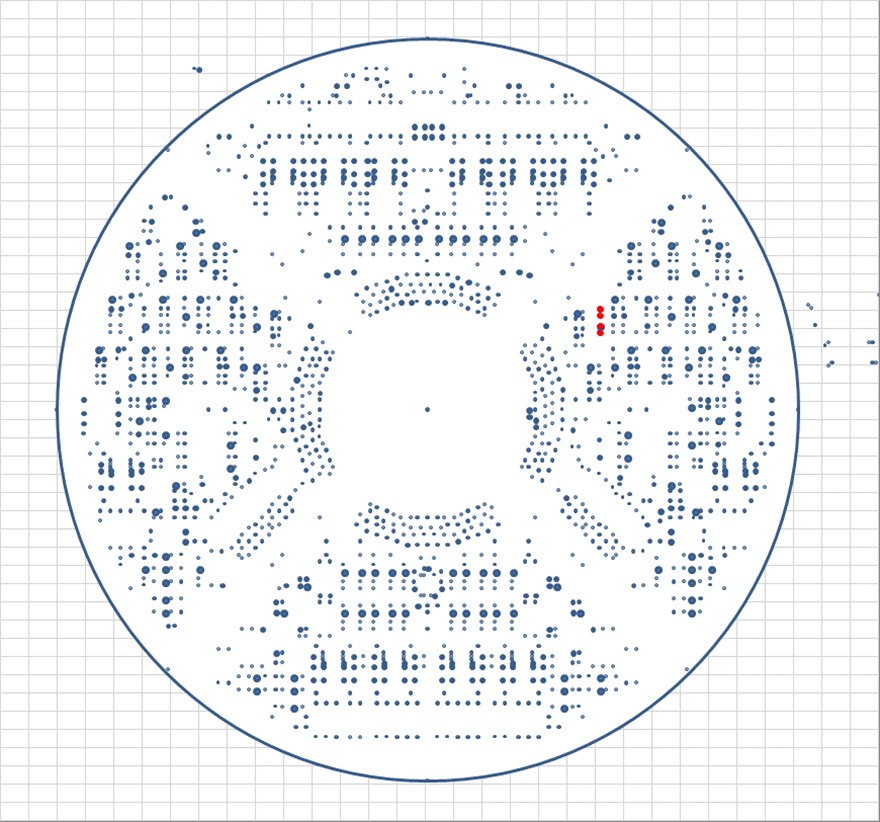


Abbildung x: Bauteilplan Beispiel

Dieser Bauteilplan muss so über das Bild der Nadelkarte gelegt werden, dass jeder Anschluss genau beim jeweiligen Bauteil liegt. Bei den rot eingezeichneten Punkten fehlt ein Bauteil.

Voraussetzung für die sogenannte „Überlappung“ ist die 45° Ausrichtung, welche durch die Querträger ersichtlich ist. Außerdem soll das Nadelkartenbild genau mit dem Rahmen der Nadelkarte beginnen und mit dem Rahmen enden.

Beispiel für eine richtig ausgerichtete Nadelkarte:

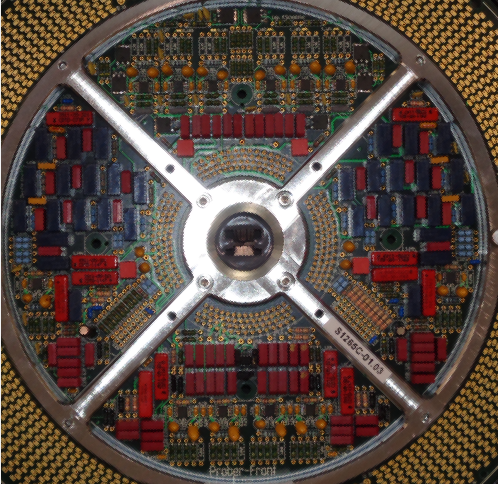


Abbildung x: Überlappung: Ausrichtung der Nadelkarte

Zuallererst muss das Bauteilplan-Bild in die richtige Größe gebracht werden und transparent werden. Um die Größe der Nadelkarte zu erreichen beginnt man mit der Umwandlung des Bauteil-Plans in eine Grauskala. Dies wird über die Funktion „rgb2gray“ erreicht. In weiterer Folge wird „mat2gray“ angewendet, um die Intensität des Bildes.

Ablauf:

* Bilder einlesen
* Bild rotieren da die Vorlage um 45° gedreht ist
* Auf die richtige Größe bringen
* In Graustufe darstellen
* Mit 1 – tmp das Gegenteil bilden
* Übertragen in ein Binäres Bild
* Die Löcher im Bild mit imfill füllen
* Die Eigenschaften des Bildes auswerten
* Den Radius berechnen und somit das sichtbare des Layouts ausschneiden
* Das Bild auf die richtige Größe bringen
* Zusammenführen der Vorlage und der Nadelkarte

## Differenz der Nadelkarten-Bilder:

Die Grundüberlegung war, die Nadelkarten im HSV, YCbcr oder LAB zu vergleichen. Dort werden die Anschlüsse gut ersichtlich und die Differenz theoretisch einfach. Praktisch aber wird nach einigen Tests ersichtlich, dass bei geringen Unterschieden der Bilder, sprich Rotation oder minimale Verschiebung entlang der X- bzw. Y-Achse eine automatische Differenz fast unmöglich ist. Nach der Differenz sind immer noch sehr viele Anschlüsse ersichtlich und auch nach der Bearbeitung mit einem Filter sieht man eine Verbesserung, doch der Anschluss, wo das Bauteil fehlt ist nicht mehr zu sehen.

Dadurch ist das RGB-Modell eine bessere Wahl und es ist sofort ein riesiger Unterschied erkennbar:

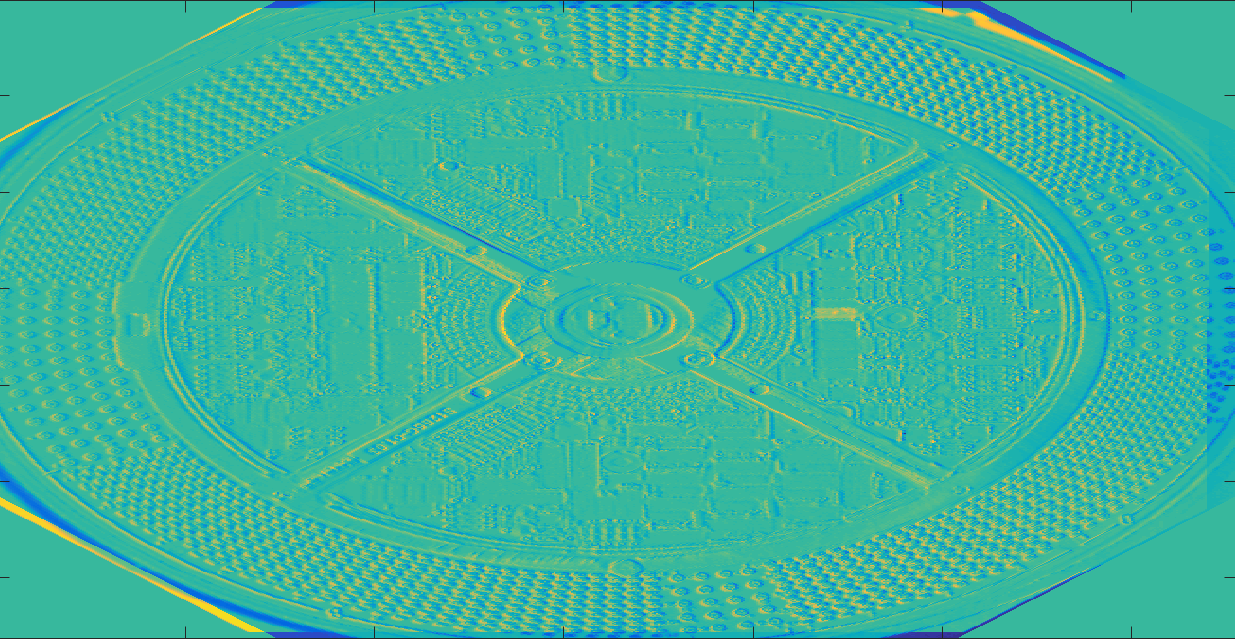


Abbildung x: Differenz-Vergleich RGB-LAB

Die gelb markierten Flächen beim RGB-Modell, ist die Differenz der beiden Bilder. Natürlich ist auch dieses Farbmodell nicht perfekt, aber mit diesem Ergebnis kann deutlich besser weitergearbeitet werden.

Programmablauf:

* Einlesen der Bilder
* Rotieren der Bilder
* Auf die gleiche Größe bringen
* Roten, Grünen und Blauen Farbkanal extrahieren
* Differenz bilden
* Differenz in eine Graustufe umwandeln
* Mit dem Medianfilter kleinere Objekt entfernen
* Pixel die einen größeren Wert als 0.6 haben in eine neue Matrix speichern
* Mit bwareopen Objekte die kleiner als 50 Pixel sind entfernen
* ein Komponenten-Bild aus den Koordinaten vom CSV-File bilden
* mit einem Maximum-Filter die Objekte am Komponenten-Bild vergrößern
* Differenz mit Komponenten-Bild multiplizeren
* Übrig gebliebenen Komponenten finden
* CSV/Datenbank speichern

## Erstellung einer Vorlage

Problem dabei 🡪 Koordinaten sind nicht so genau. Das bedeutet beim Vergleich von Differenzbild und Koordinaten sind Ungenauigkeiten vorhanden und es werden statt nur das richtige Bauteil, ein falsches bzw. mehrere falsche Bauteile gespeichert. Dadurch muss eine andere Strategie entwickelt werden.

1. Ansatz

Es soll ein Bild erstellt werden mit Hilfe der Koordinaten. Diese Koordinaten sind in einer Excel Tabelle gespeichert. Danach werden diese Koordinaten gezeichnet und zwar bekommt jedes Bauteil einen Pixel und eine entsprechende Farbe. Die Farbe entsteht das dass 1. Element den Index 1 hat und das n. Element den Index n hat. Somit entsteht ein Farbmuster welches aber nicht relevant ist sondern der Index.

So kann bestimmt werden dass der Pixel mit dem Index y genau das Bauteil aus der Excel Tabelle ist.

Die Koordinaten etwas zu vergrößern, sprich statt nur 1 Pixel zum Beispiel 10 Pixel verwenden. Fazit: War auch nicht wirklich genau.

1. Ansatz

Auf eine andere Vorlage gestoßen:

Abbildung 10: Vorlage NK

Die Überlegung bei diesem Ansatz war alle Bauteile auszufüllen und diese zu labeln um wieder den Index dieser Bauteile hat. Problem dabei ist dass es nicht so einfach ist alle Bauteile zu erwischen. Zu allererst müssen alle Linien als Linien erkannt werden und danach ausgefüllt werden. Dies wird mit der Funktion „edge“ erledigt doch dadurch das diese Vorlage keine allzu große Auflösung hat, sind manche Bauteile offen. Diese kann man natürlich schließen, doch manche Bauteile, wie zum Beispiel die runden Bauteile, werden nicht erkannt und gehen verloren.

1. Ansatz

Die Nadelkarte halb Automatisch mit der Vorlage auszurichten. Dadurch muss man einige Punkte definieren und die Funktion richtet die Vorlage danach aus.

Tatsächlich werden von allen 3 Ansätzen ein Teil verwendet. Es wird eine Vorlage erstellt die mit den Koordinaten gezeichnet wird und gelabelt wird, damit jedes Bauteil eine Farbe hat. Anschließend werden die Punkte verbunden und in der gleichen Farbe ausgefüllt, um eine einheitliche Fläche zu erhalten. Danach wird diese Vorlage mit der in der Abbildung 10 gezeigten Vorlage multipliziert um diese Fläche genau auf die richtigen Punkte einschränken zu können. Daraufhin werden die kleineren Punkte wieder miteinander verbunden und ausgefüllt. Nach diesem Schritt ist die Vorlage fertig und wird in eine MAT-Datei abgespeichert um sie danach zu laden. Es ist nicht notwendig, bei jeder Nadelkarte eine neue Vorlage zu erstellen, es kann beim gleichen Typ immer dieselbe verwendet werden.

## Erkennung der Bauteile

Dadurch das alle Komponenten vorhanden sind, sprich Differenz, Bauteilliste und Vorlage, können nun die Bauteile identifiziert werden, indem die Differenz mit der Vorlage multipliziert wird und somit gelabelte Farbflecken übrig bleiben. Diese Flecken haben verschieden Nummern mit welcher der Name des Bauteiles herausgefunden werden kann. Es können am Differenz-Bild viele Fehler auftreten wie zum Beispiel verschiedene Belichtungen führen zu unterschiedliche Referenzen. Durch die Multiplikation mit der Vorlage werden nur noch die wirklich betroffenen Bauteile dargestellt und können zu einer bestimmten Wahrscheinlichkeit fehlerhaft sein. Diese Wahrscheinlichkeit wird aus der Anzahl der Pixel die im Ergebnis der Multiplikation diese Nummer haben und der maximal möglichen Pixel.

# Probleme

* Richtige Ausrichtung der Kamera
* Größe der Bilder
* Falscher Farbraum
* Richtige Kamera finden
* Falscher Farbraum
* Koordinaten der Bauteilliste
* Genaues Ausrichten nach dem rotieren (oftmals wenige Pixel unterschiede, evtl wegen Anpassung der Größe)
* Vorlage erstellen ( Koordinaten passen nicht zu 100%)
* Vorlage Objekte erkennen (nicht alle werden erkannt)
* Vorlage und NK auf richtige Größe zu bringen
* Ungenauigkeit bei der Erkennung der Bauteile. (Graues Bild der Differenz wo Pixel größer einem bestimmten wert sind)
* Ausrichten der Nadelkarte beim erstellen eines Referenzbildes (mit cbselect)

# Ablauf

* First\_RGB\_Test (Dollarschein)
* Color Treshholder einfache Differenz gebildet
* Erster Test für mehre Farbräume
* **Resize**

Es ist sehr wichtig alle Nadelkarten auf dieselbe Größe zu bekommen, um sie optimal vergleichen zu können. Der einheitliche Anfang einer Nadelkarte ist der Beginn der Stecker am Rand der Nadelkarte. Um diese ersichtlich zu bekommen wird das Nadelkarten-Bild in ein anderes Farbmodell umgewandelt, wie vorhin schon und danach bis zu den Steckern zugeschnitten. Daraufhin muss die Größe noch zum Referenzbild angepasst werden, wobei eine Skalierung ausgerechnet werden muss.

scale = size(componentImage, 1) / size(rgbNKImg, 1);

rgbNKImg = imresize(rgbNKImg, scale);

Dadurch sind beide Bilder in der selben Größe und können perfekt übereinander gelegt warden und somit auch verglichen werden.

* **Rotate**

Für die Rotation der Nadelkarte wurde eine Funktion geschrieben die zwei Bilder übergeben bekommt. Das erste Bild ist das Originalbild, sprich Referenzbild und das zweite Bild die verdrehte Nadelkarte. Zuallererst werden mit der Funktion „detectSURFFeatures()“ die Surf-Points der Bilder gespeichert.

* Vorlage überlagern
* Differenz rgb
* Bauteile moseover
* Vorlage
* Vorlage mit vorlage multiplizert
* Zusammengesezt
* Vorlage automatisch erstellen

Zuerst über Punkte probiert, nicht funktioniert und darauf mit den Anschlüssen

* Vorlage mit Nadelkarte zusammenfügen

Problem 🡪 Größe passt nicht zusammen und Ausrichtung

Größe des Referenzbildes verändern da sonst Schwierigkeiten mit der Vorlage entstehen

**13.08.2018**

* Fertig ausrichten der Nadelkarten für differenzbild
* Verbesserung des Vergleich-Algorithmus und drawComponentImage Algorithmus

**14.08.2018**

* Differenz verbessert
* GUI erstellt
* Vorlage verbessert
* Neue Fotos
* testen

**16.08.2018**

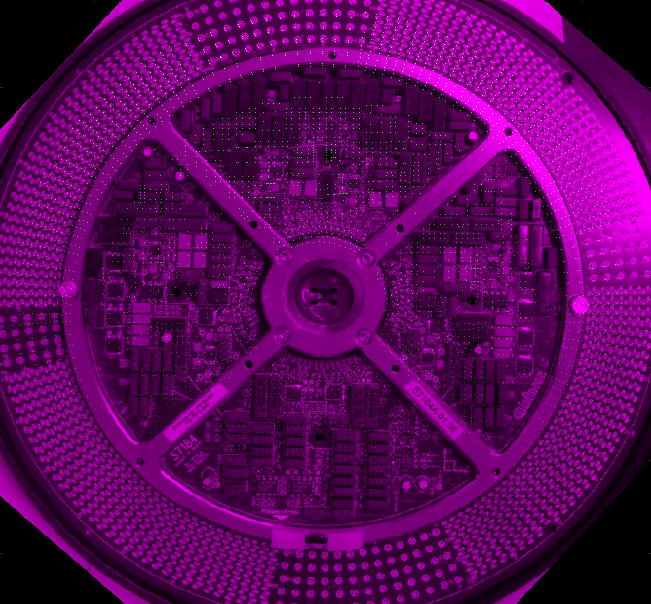
* Layout verbessern
* GUI style verändert
* Vorlagen erstellt

**8.17.2018**

* Vorlage richtig ausrichten
* Cb Select eingebaut bei vorlage erstellen

**27.08.2018**

* Mouseover von NK fertigstellen
* Problem: Koordinaten haben immer einen unterschiedlichen Rand am end, was dazu führt das die Koordinaten nicht mehr auf das Bild passen.



Dadurch muss es anders gelöst werden, sprich mit der Vorlage. Dadurch sind nicht alle Bauteile ersichtlich aufgrund der Ungenauigkeit der Vorlage.

**28.08.2018-30.08.2018**

Präsentation Vorbereitung

Gui erstellt

**03.09.2018**

Excel-Dateien formatiert

Koordinaten Tests

Anpassung des Programms auf die Excel Listen

**04.09.2018**

# Quellen:

1. „MATLAB A PRACTICAL INTRODUCTION TO PROGRAMMING AND PROBLEM SOLVING – SECOND EDITION“ von Stormy Attaway
2. Mathworks, <https://de.mathworks.com> 13.07.2018

# Resümee

Hier fassen wir unsere Ergebnisse zusammen ... (persönliche Meinung, was ist das Ergebnis dieser Arbeit [auf den Punkt gebracht], Vergleich zur These im Kapitel Einleitung und Überblick, Ausblick in die Zukunft, offene Fragen, ..., *optional*, mindestens eine A4 Seite)

# Anhang

Dieser Anhang ...

## Literaturverzeichnis

1. M. Björkander: „Graphical Programming Using UML and SDL”, in IEEE Computer, IEEE Computer Society, Vol. 33, No. 12, Dez. 2000, pp. 30-35   
   *allgemein für eine Zeitschrift: Autor: „Titel“, Name der Zeitschrift, Jahrgang, Ausgabenummer, Monat (optional), Seiten (dt. S. x-y, engl. pp. x-y)*
2. OMG: „Action Semantics for the UML”, <http://www.omg.org>, 31. Dez. 2015  
   *allgemein für eine Internetseite: Titel der Webpage: „Titel des Abschnitts in der Webpage (optional)“, vollständige URL, Datum des Zugriffs   
   Sinnvollerweise speichert man eine Kopie davon, zB als pdf Dokument.*
3. Y. Haimes: „Risk Modeling, Assessment and Management”, John Wiley & Sons, New York, 1998, 2nd edition   
   *allgemein für ein Buch: Autor(en): „Titel des Buchs“, Verlag, Erscheinungsjahr, Auflage (optional)*

## Anleitungen / HowTos

Folgendes HowTo ...

## Beispiele / Sources

Folgende Beispiele ...

## Projekt- / Source-Verweise

Programm-Sources oder Projekte sind in Analogie zu den Grafiken mit ./src/kapitel/file über eine Tabelle zu verbinden:

|  |  |
| --- | --- |
| Kurze Beschreibung | link to source or project file |
| Demonstriert OLE | ./src/kapitel/Demo.java |
|  |  |

1. Auf das korrekte Gendern sollte nicht vergessen werden. [↑](#footnote-ref-1)