**Optischer Münzenzähler**

****

Dokumentation

10. Dezember 2014

Julien Villiger, Daniel Inversini

Modul BZG1301: Programmierung Matlab/Octave

Inhalt

[Zusammenfassung 3](#_Toc408312020)

[Einleitung 3](#_Toc408312021)

[Vorgehen, Methoden Analysen 4](#_Toc408312022)

[Vorarbeiten 4](#_Toc408312023)

[Bilder aufnehmen 4](#_Toc408312024)

[Grunddaten ermitteln 5](#_Toc408312025)

[Optimales Verhältnis von Bildauflösung zu Abstand der Kamera 5](#_Toc408312026)

[Bildaufbereitung 6](#_Toc408312027)

[Graustufenerkennung 6](#_Toc408312028)

[Binäres Bild 6](#_Toc408312029)

[Closing 6](#_Toc408312030)

[Bildanalyse 7](#_Toc408312031)

[Kantendedektion 7](#_Toc408312032)

[Houghtransformation 7](#_Toc408312033)

[Resultate 8](#_Toc408312034)

[Diskussion 9](#_Toc408312035)

[Anhang 10](#_Toc408312036)

[Glossar\* 11](#_Toc408312037)

# Zusammenfassung

Bei dieser Arbeit wurde eine Methode entwickelt, um Münzen auf einer Fotoaufnahme zu erkennen und dessen Werte zu ermitteln. Somit sollte mit einem Knopfdruck möglich sein, die Summe sämtlicher auf dem Foto liegenden Münzen auszugeben.

Es stellte sich die Frage, ob die Bilder (meist aufgenommen mit einem Smartphone) genügend detaillierte Informationen liefern, um den Erkennungsprozess durchzuführen und ob die möglichen Fehlerquellen tolerierbar waren.

Bewahrheitet hat sich die Annahme, dass leichte Verschiebungen des Kamerawinkels und dessen Distanz zu den Münzen einen entscheidenden Einfluss auf das Resultat haben. Diese mussten mit verschiedenen Algorithmen soweit es geht ausgemerzt werden.

Der Erkennungsprozess des Bildes unterliegt etlichen Bildanalyse-Algorithmen und Verarbeitungen wie einer Kantenerkennung [1], Binarisierung [2], und weiteren (Siehe Kapitel Vorgehen)

Resultiert hat eine Matlab [3] Anwendung, welche zuverlässig die Summe der zu erkennenden Münzen ermittelt, vorausgesetzt das Foto entspricht gewissen Kriterien.

# Einleitung

Der Rahmen dieser Arbeit wurde durch uns selbst definiert. Wir steckten uns somit das Ziel, dass ein normales Foto analysiert werden kann.

Somit kann dieses Matlab Script unter Umständen sehr schnell und unabhängig eingesetzt werden. (Innerhalb weniger Minuten Münzen auf dem Tisch ausleeren, fotografieren mit dem Smartphone, per Mail/Bluetooth an den PC senden, Script ausführen und Resultat Anzeigen).

Ob dies schneller und effizienter ist, als die Münzen klassisch von Hand zu zählen, können und wollen wir nicht beantworten. Mit dieser kleinen Anwendung kann aber auch informatik-unbegeisterten gezeigt werden, wie mit einfachen Methoden, etwas Kreativität und einem Grundwissen von Bildverarbeitung ein alltägliches Problem gelöst werden kann.

Die verwendeten Methoden\* in der Übersicht (geordnet nach Anwendungsschritt):

* Graustufenerkennung [4]
* Binäres Bild
* Closing [5]
* Kantendedektion
* Hough Transformation [6]

\*Folgende kleinere subsidiäre Funktionen wurden ebenfalls verwendet, auf diese wird aber nicht speziell eingegangen:

* Grunddaten (Durchmesser und Verhältniss aus einer Config-Datei)
* Timer (zum Messen der Performance)
* Schwarzen und weisse Pixel zählen (Durchmesser ermitteln)
* Matrixoperationen

# Vorgehen, Methoden Analysen

Dieses Kapitel enthält drei Teile:  
Die Vorarbeiten, welche Beispielbilder und weitere Informationen enthält,  
die Bildaufbereitung, wo das Bild für die Analyse vorbereitet wird (ist bereits Teil des Scripts) und  
die Bildanalyse, welche die Münzen identifiziert und mit dem Grössenverhältnis der Münzen untereinander den Wert bestimmt.

## Vorarbeiten

### Bilder aufnehmen

Folgende Bilder dienen uns als Entwicklungshilfe, jeweils von uns selbst erstellt:



Abbildung : Referenzbild 1a, verschiedene Münzen und Referenzcoin (1-Räppler)



Abbildung : Referenzbild 1, verschiedene 1-Fränkler und Referenzcoin (1-Räppler)



Abbildung : Referenzbild 1b

Auf allen Referenzbildern ist der 1-Räppler erkennbar.

### Grunddaten ermitteln

Folgende Daten mussten wir uns extern beschaffen\*:

|  |  |
| --- | --- |
| Wert der Münze [SFr.] | Durchmesser in [mm] |
| 5 | 31.45 |
| 2 | 27.4 |
| 1 | 23.2 |
| 0.5 | 18.2 |
| 0.2 | 21.05 |
| 0.1 | 19.15 |
| 0.05 | 17.15 |

Tabelle : Daten Schweizer Münzen

\*Quelle: <http://de.wikipedia.org/wiki/Schweizer_Franken>

Dazu kommt noch unsere Referenzmünze, ein Schweizer 1-Räppler:

|  |  |
| --- | --- |
| Wert der Münze [SFr.] | Durchmesser in [mm] |
| 0.01 | 16 |

### Optimales Verhältnis von Bildauflösung zu Abstand der Kamera

Da die heutigen Smartphones extrem hochauflösende Photos schiessen, mussten wir einen optimalen Abstand von Kamera zum fotografierenden Objekt (Münzen) finden.

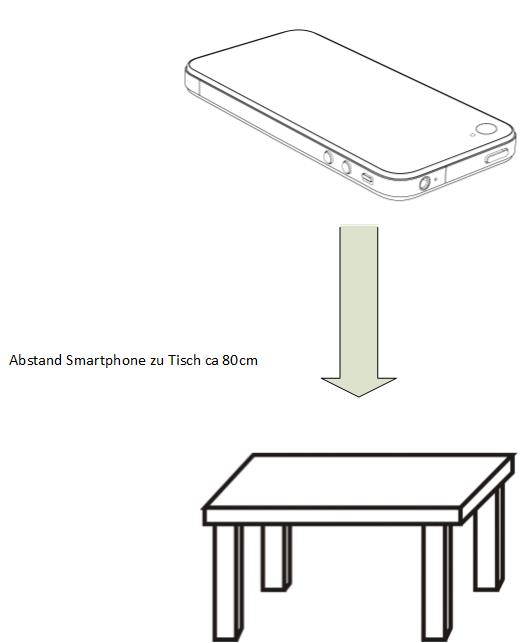
Warum ist dies so wichtig?

Abbildung : Abstand Smartphone zu Tisch

Da ein Bild schnell eine Auflösung von 8-11 Megapixel haben kann, können die Flächen der Münzen auch relativ viele Pixel haben.

Somit wird es dann später schwieriger, mit einen relativ knappen Threshhold [7], ähnlich grosse Münzen zu Unterscheiden.  
(Siehe oben, bspw. 50 Rappen und 10 Rappen)

Durch diesen selbst eruierten Abstand sind die Münzen etwas kleiner (was aber bei der grossen Auflösung nicht weiter ins Gewicht fällt), und eine perspektivische Verzerrung [8] kann auch eliminiert werden.

## Bildaufbereitung

Folgende der bereits erwähnten Methoden verwenden wir, um das Bild vorzubereiten für die Analyse:

### Graustufenerkennung

Mit der Graustufenerkennung suchen wir den Referenzcoin.



Referenzcoin

Abbildung : Graustufenbild

### Binäres Bild

Das binäre Bild erleichtert uns später die Kantendedektion.

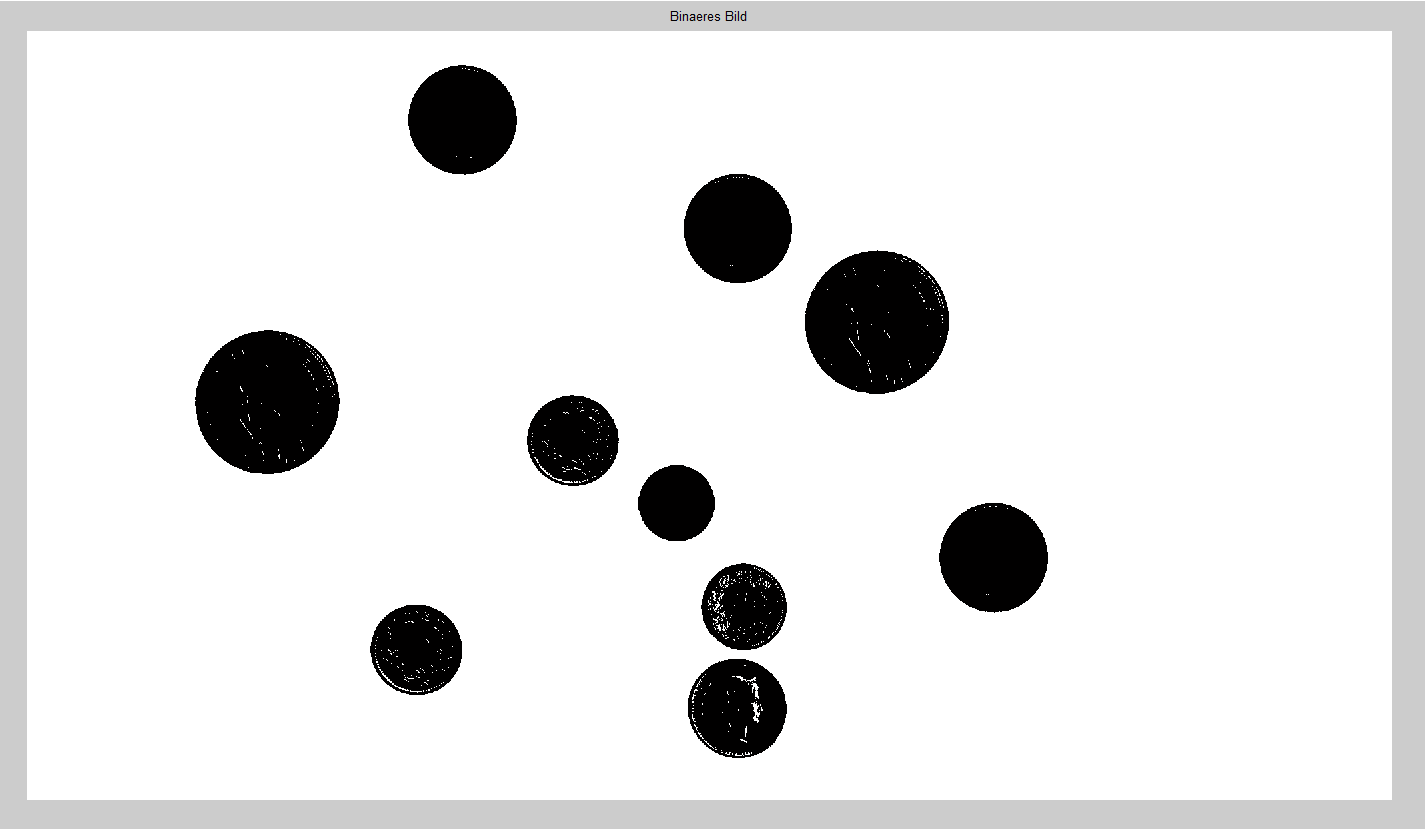


Abbildung : Binäres Bild

### Closing

Eventuelle Löcher (Weisse Punkte innerhalb einer Münze, bspw durch den Blitz erzeugt) werden hier geschlossen. Es wird das vorher erzeugte binäre Bild verwendet.

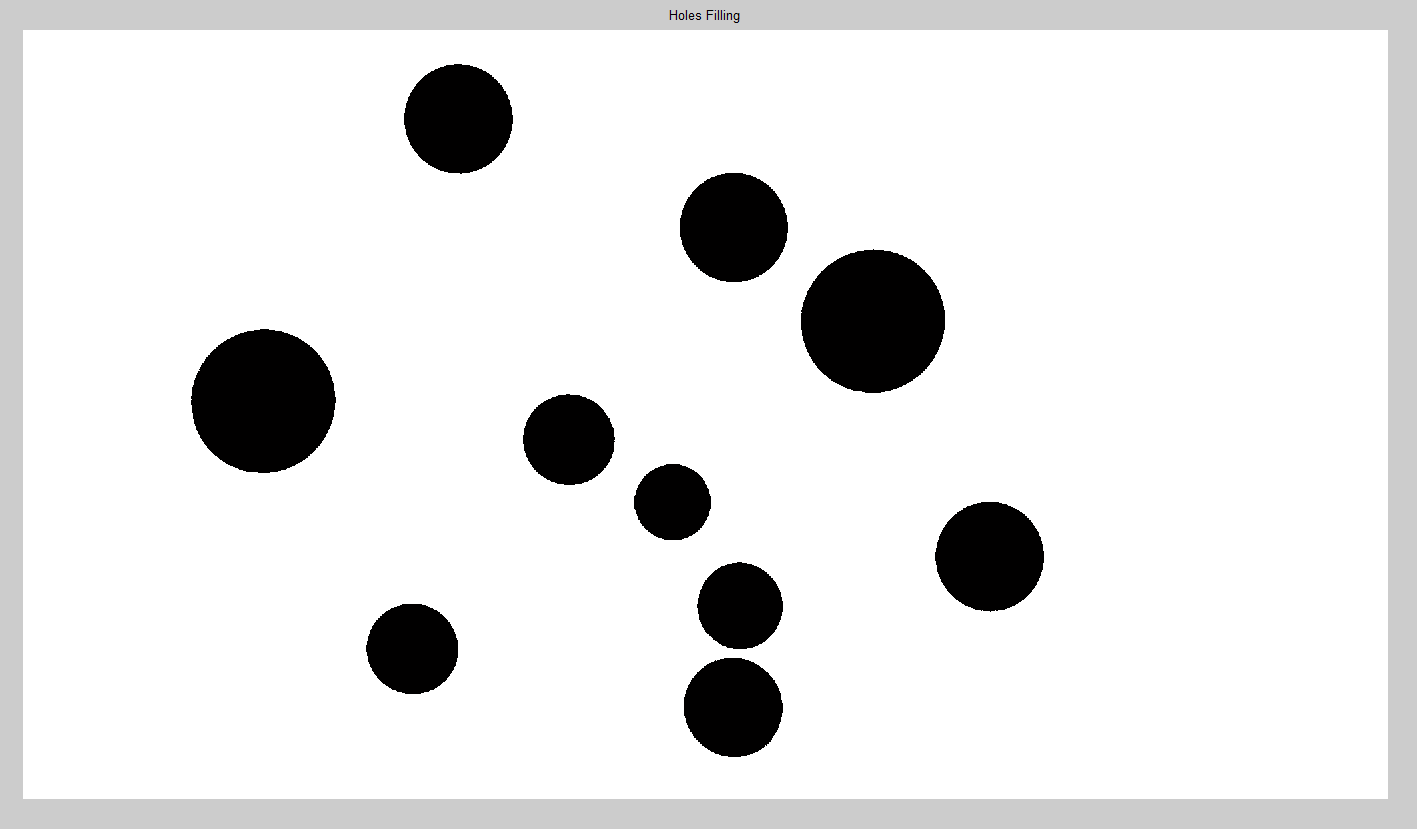


Abbildung : Closing 1

Danach werden noch die unregelmässigen schwarzen Punkte elimniert. Es wird wiederum das vorher erzeugte Bild verwendet.

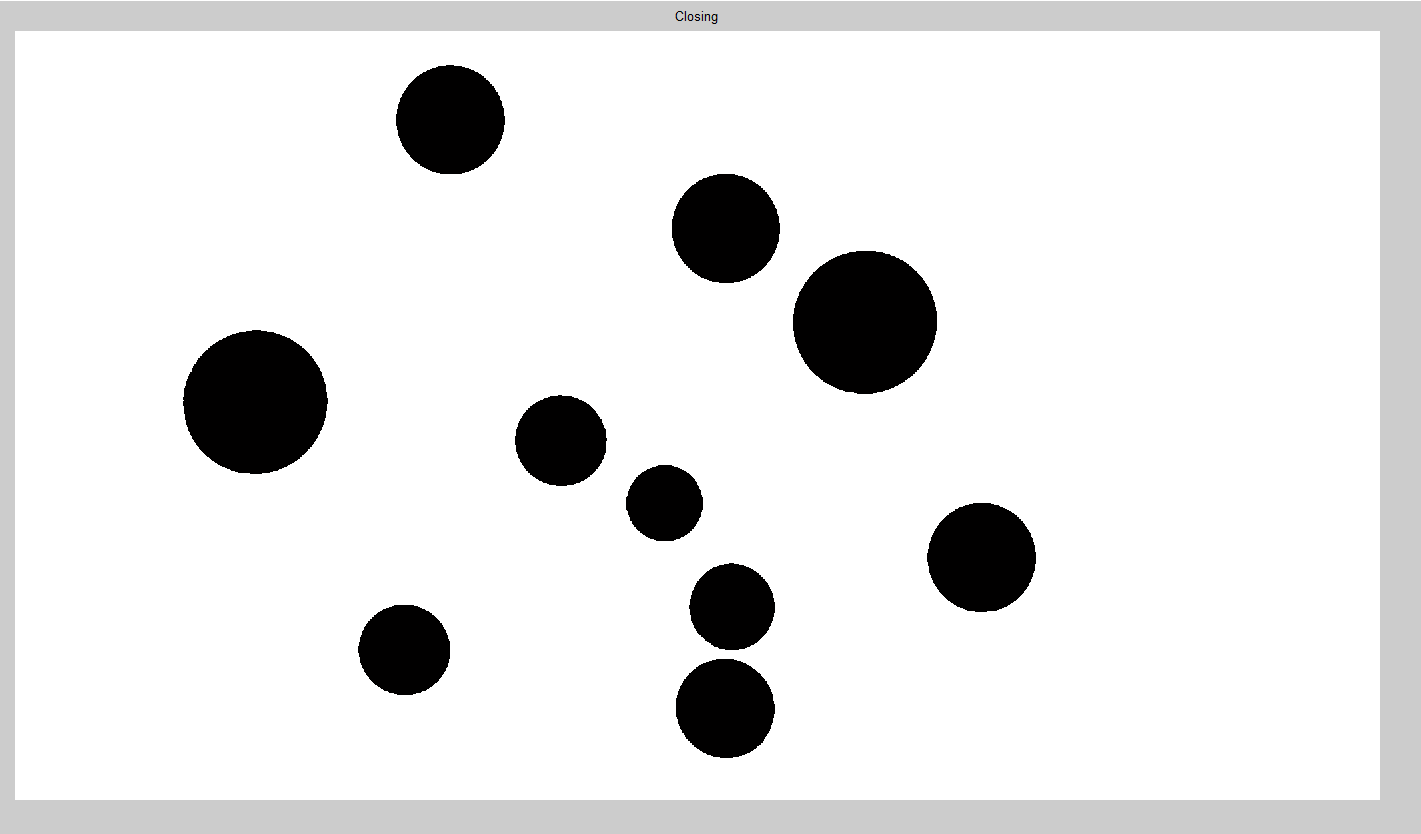


Abbildung : Closing 2

## Bildanalyse

### Kantendedektion

Wir verwendeten den Canny-Algorithmus, um die Kanten zu dedektieren. Wir haben uns für diesen entschieden, weil dieser allgemein als bester klassischer Kantenerkennungsalgorithums anerkannt ist. \*

\* http://www.mathematik.uni-ulm.de/stochastik/lehre/ws05\_06/seminar/wagner.pdf

### Houghtransformation

Blah blah blah

# Resultate

Niceness.

# Diskussion

# Anhang

Anhang

# Glossar\*

|  |  |  |
| --- | --- | --- |
| 1 | Kantenerkennung | Die Kantenerkennung ist ein digitales Bildanalyse-Verfahren. Eine Kante wird als Grenzfläche zwischen zwei unterscheidlichen Bildbereichen definiert. Kanten können durch eine Veränderung des Lichts (Farbe, Schatten) oder durch unterschiedliche Texturen (Oberflächen) bedingt sein. |
| 2 | Binarisierung | Ein Binärbild, oder eine Binarisierung ist eine digitale Rastergraphik, wo die Pixel nur aus Schwarz oder Weiss bestehen. |
| 3 | Matlab | Matlab steht für MATrix LABoratory, und ist eine komerzielle Software von „The MatWorks“. |
| 4 | Graustufenerkennung | Dies bezeichnet ein Bild nur bestehend aus Abstufungen von reinem Weiss bis zu reinem Schwarz. So kann ein RGB Bild mit bspw 16.7 Mio Farben in 255 Graustufen unterteilt werden. |
| 5 | Closing | Bezeichnet ein morphologisches Verfahren auf Binärbildern, um Löcher zu füllen (Dilatation gefolgt von Erosion) |
| 6 | Hough Transformation | Die Hough-Transformation ist ein robustes globales Verfahren zur Erkennung von Geraden, Kreisen oder beliebigen anderen parametrisierbaren geometrischen Figuren in einem binären Gradientenbild, also einem Schwarz-Weiß-Bild, nach einer Kantenerkennung. Das Verfahren wurde 1962 von Paul V. C. Hough unter dem Namen „Method and Means for Recognizing Complex Patterns“ patentiert. |
| 7 | Treshhold | Schwellenwert, auch Schwellenwertverfahren. Definiert, welche Bildpunkte das gesuchte Objekt darstellen, und welche der Umgebung angehören. (Bei Beispielsweise Graustufen) |
| 8 | Perspektivische Verzerrung | Dies ist der geometrische Abbildungsfehler, welcher entsteht wenn gleiche Objekte unterschiedlich weit vom Zentrum (Fotokamera, Auge) entfernt sind. |

\*Quelle: Wikipedia.org