Elektronische Bildverarbeitung
Dozent/Tutor: Udo Frese

SoSe17

Lukas Bertram
Sebastian Bliefert
Johannes Hochbein
Pascal Pieper

Übungsblatt 1

Lösungsvorschlag
Abgabe: 30.05.2017

Aufgabe 1

1 Gesamtkonzept

Um die Würfel im Versuchsaufbau zu zählen, wird das Eingabebild zunächst geeignet vorprozessiert und anhand der Farben aufgeteilt. Die resultierenden Einzelbilder für jede Farbe werden binarisiert und anschließend einzeln der Würfelerkennung zugeführt. Diese ermittelt annähernd die geometrischen Mittelpunkte der segmentierten (Würfel-)flächen und nutzt diese Informationen zur Trennung der einzelnen Würfel mithilfe des „watershed"-Algorithmus. Anschließend werden die innerhalb der separierten Würfelflächen vorhandenen Augen mithilfe eines „blob detectors"gezählt und in der Statistik der Augenverteilung erfasst.

2 Teilalgorithmen

2.1 void preprocessColors(Mat&img, double highestValue))

-

tbd

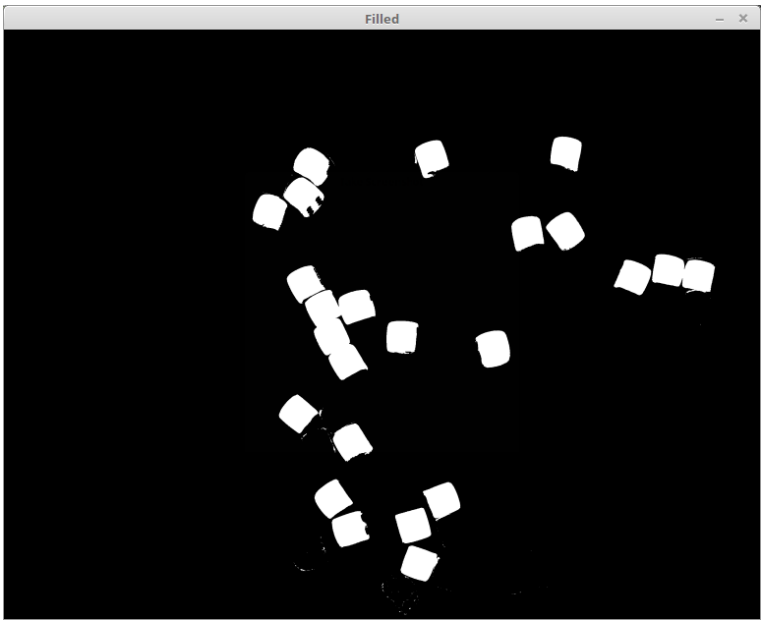
2.2 void seperateDiceColors(Mat& image, Mat& display, vector <Dice>& dices)

- **Mat& image:** Verweis auf das Eingabebild
- **Mat& display**: Verweis auf das zur Anzeige verwendete Bild
- **vector<Dice>& dices**: Verweis auf den mit den gefundenen Würfeln zu füllenden Vektor

Die Funktion *seperateDiceColors(...)* extrahiert die für die einzelnen Würfelfarben relevanten Pixel aus dem Eingabebild und ruft *segmentAndRecognizeFromBinImage(...)* für jedes der Ergebnisbilder auf. Für die Extraktion wird das Eingabebild zunächst in den HSV-Farbraum konvertiert und anschließend mithilfe der openCV Funktion „inRange()"auf das für die einzelnen Würfelfarben relevante Spektrum begrenzt und binarisiert.

2.3 void segmentAndRecognizeFromBinImage(Mat& image, vector<Dice>& dices)

- **Mat& image**: Verweis auf das binäre Eingabebild
- **vector<Dice>& dices**: Verweis auf den mit den gefundenen Würfeln zu füllenden Vektor



segmentAndRecognizeFromBinImage(...) segmentiert einzelne Würfel aus dem Eingabebild. Dazu werden auf einer Arbeitskopie des Bildes zunächst die Würfelflächen vollständig mit Weiß aufgefüllt (Hierfür Hintergrund per floodfill(...) auffüllen, Ergebnis invertieren und auf das Eingabebild addieren), sodass die schwarzen Augen von den Würfelflächen verschwinden.

Mithilfe dieses Bildes werden die „Schwerpunkte“der Würfelflächen unter Verwendung des durch den distanceTransform-Algorithmus generierten Bilds bestimmt: Das Ergebnisbild (8 Bit Graustufen) wird durch Anwendung eines Schwellwerts, Dilatation und anschließender Konturfindung überführt in einen Vektor aus Markern, die die „Schwerpunkte“der Würfel repräsentieren. Mit „Schwerpunkt“ist hier das Maximum der Entfernungen eines jeden Pixels innerhalb der Würfelfläche zum Rand der Fläche gemeint.

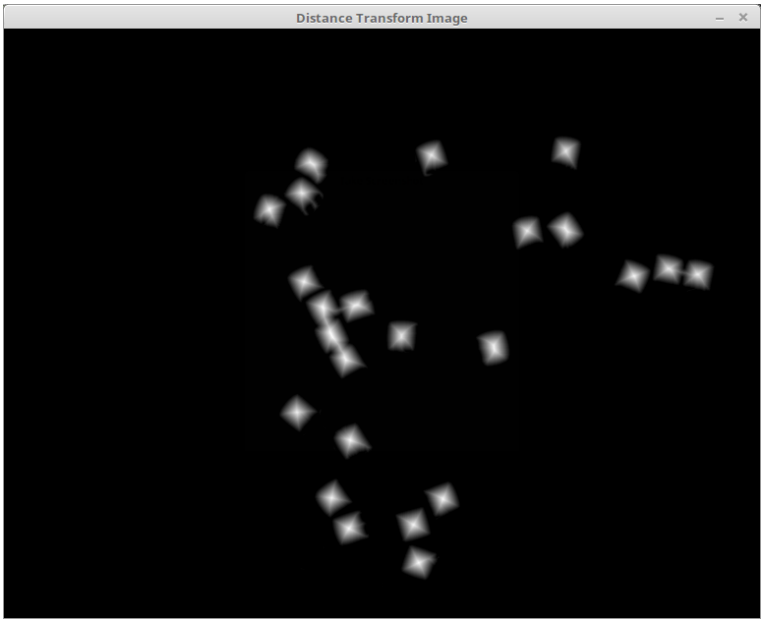
Die so gewonnen Marker werden als Startpunkte für den Watershed-Algorithmus gebraucht, welcher die eigentliche Segmentierung der einzelnen Würfel vornimmt. Dazu füllt der Algorithmus das Eingabebild anhand der Marker in alle Richtungen auf, bis sich die einzelnen Füllgebiete entweder treffen oder der Hintergrund den Füllvorgang begrenzt (genauere Informationen zu watershed(...) unter [**?**]) und stellt die Konturen der so gefundenen Gebiete zur Verfügung.

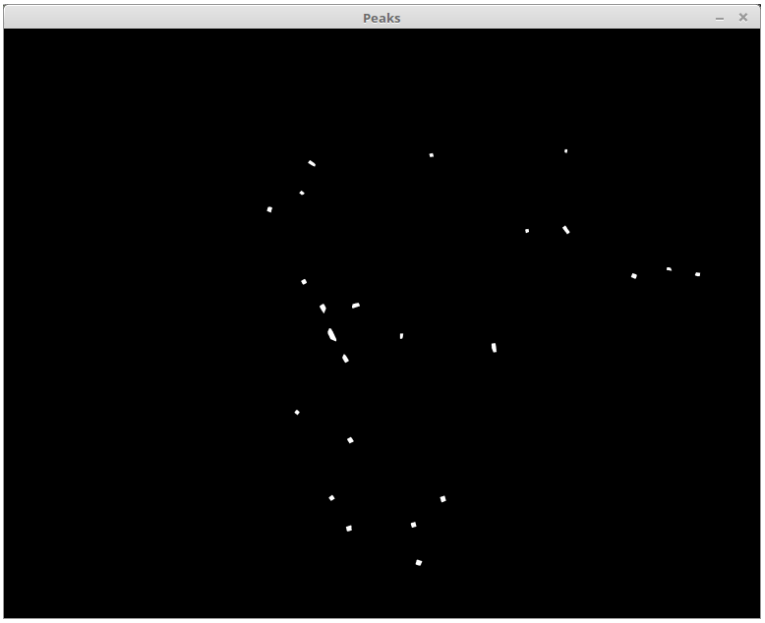
Aus jeder dieser Konturen wird mit minAreaRect(...) das für die Augenzählung relevante Gebiet bestimmt und zusammen mit weiteren Parametern an *countBlobs(...)* übergeben. Zusätzlich wird jeder gefundene Würfel im Vektor *dices* für spätere Operationen abgelegt.

2.4 Dice countBlobs(SimpleBlobDetector& d, Mat& orig, RotatedRect& elem, vector<Point>& approx)

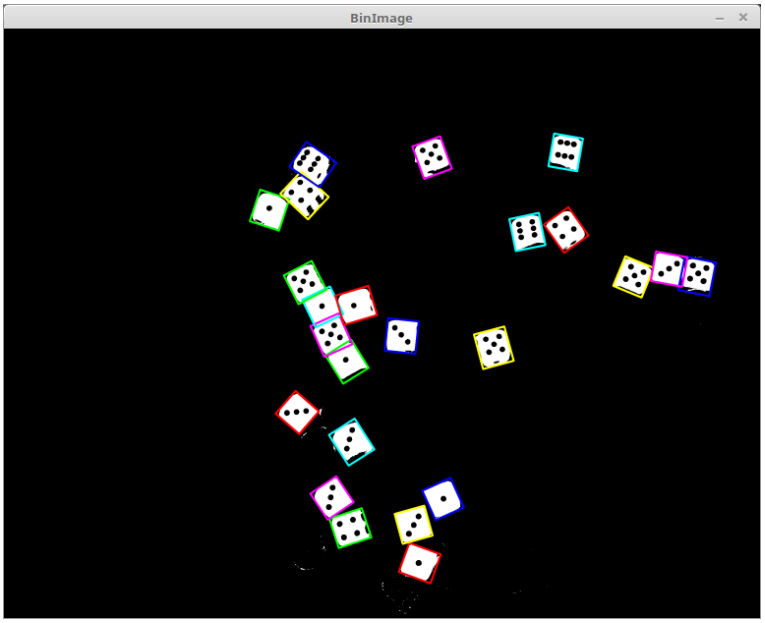
-

tbd


Distance Transform Image


Peaks


Final Result


BinImage

2.5 void addToStatistics (vector<int>& statistics, const vector<Dice>& dices)

-

tbd

2.6 bool passed (const vector<int>& statistics)

-

tbd

Aufgabe 2

tbd.

[CODE]

Aufgabe 3

3 Konzept:

Der Keratograf sendet ein definiertes Muster aus konzentrischen, abwechselnd schwarzen und weißen Kreisen in Richtung Auge. Je nach Position des Auges vor dem Keratografen ergibt sich eine entsprechende Spiegelung der Ringe auf der Oberfläche der Hornhaut. Die Kamera ist im Zentrum des Keratografen platziert und nimmt das Bild der Spiegelung auf. Nach Bestimmung des gemeinsamen Mittelpunkts der Kreise wird der Abstand der Übergänge von Schwarz zu Weiß und von Weiß zu Schwarz entlang eines radialen Schnitts vermessen. Dazu wird eine Halbgerade festgelegt, die zunächst horizontal durch den Mittelpunkt verläuft und in diesem beginnt. Mit definierter Schrittweite wird der Schnittpunkt der Geraden mit einem fiktiven, zum Mittelpunkt konzentrischen Kreis auf diesem Kreis umlaufend verschoben. In jedem Zyklus werden nach einem "Weiterdrehenälle Kanten in zur Halbgeraden senkrechter Richtung zunächst gefunden und dann in ihrem Abstand zum Mittelpunkt (Beginn der Halbgeraden und damit Urpsrung des zur Vermessung genutzten Koordinatensystems) vermessen.

4 Bewertung der Messwerte:

Der theoretisch perfekte Abstand zwischen den Kanten ist bekannt für eine perfekt kugelfömige Honrhaut. Die Differenzen zwischen den theoretischen und den tatsächlich gemessenen Abständen gibt Aufschluss über die Form der Hornhaut: - Ist der gemessene Abstand kleiner als der theoretisch erwartete Wert, so ist die Hornhaut an dieser Stelle konkaver als die perfekte Kugel. - Ist größer als der theoretisch erwartete Wert, so ist die Hornhaut an dieser Stelle konvexer als die perfekte Kugel.

Literatur

[1] OpenCV Documentation, *Image Segmentation with Watershed Algorithm*, `http://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html`, abgerufen am 30.05.17, 12:00Uhr.