

Projektdokumentation im Modul Digitale Bildverarbeitung **Detektion und Erkennung von QR-Verkehrszeichen**

Tina Neumann und Minh Pham

21. Juni 2017

Mit dem vorliegenden ImageJ-PlugIn ist es möglich, QR-Codes in einem RGB Bild zu erkennen.

1 Ansatz

Das Erkennen von QR-Codes erfolgt mit einem horizontalen Linescan. Dabei wird das Bild zunächst binarisiert und im darauffolgenden mit einem horizontalen Linescan untersucht.

2 Verwendung von Parametern

binMode: Unterscheidung zwischen 0 und 1. Bei der Auswahl von 0, wählt man die Otsu-Methode. Bei der Auswahl von 1 wählt man die HSB Methode.

Boxgröße: Hierbei werden die Boxgrößen mit minimalen und maximalen Boxbreite und -höhe in Pixeln bestimmt.

scanColDist: Mit scanDolDist werden die vertikalen ScanLine pixelweise festgesetzt.

Bildvariablen: Dabei werden das Originalbild und das Binärbild jeweils als ImagePlus gespeichert. In dem Originalbild werden die gefundenen BoundingBoxen gezeichnet und die Bearbeitung erfolgt im Binärbild.

3 run-Methode

3.1 Öffnen & Binarisieren des Bildes

Im ersten Schritt wird das PlugIn im Makro aufgerufen, danach wird das Bild des übergebenen Pfad geöffnet und hinzu binarisiert. Desweiteren gibt es die Möglichkeit das Bild im ImageJ zu öffnen, falls das Bild im Makro nicht gefunden wird.

Es existiert zwei Möglichkeiten für das Binarisieren des Bildes. Mit binmode kann festgelegt werden, ob die Ostu-Threshold-Methode (Standard ImageJ) oder die selbstimplementierte colorThresholdBinary verwendet werden.

3.2 Auffinden von weißen Liniensegmenten

Hierbei werden die Segmente im Linescan untersucht. Ziel ist es hierbei weißen Segmente zu finden. Im ersten Schritt erfolgt das Auffinden von Segmenten. Die Spaltenprofile werden untersucht, ob diese mehr weiße Pixelanteile enthalten sind. Diese werden in der Hash-Map hinzugefügt.

3.3 Ausschluss von Segmenten

Der folgenden Hash-Map wird iteriert und die Segmente untersucht. Dabei erfolgt das Ausschließen von Segmenten, die schwarze Pixelanteile enthalten. Desweiteren werden Linien, die keine Kanten sind und die über dem Standardwert von 350 sind, entfernt.

3.4 Vergleich von Segmenten

In zwei verschachtelten For Schleifen werden die Linien der segmentMap miteinander verglichen. Dafür wird zunächst geprüft, ob sich die Segment Nummern (keys der Hashmap) und die Linien unterscheiden. Desweiteren wird untersucht, ob die Linien unterschiedliche Spalten (x Position) aufweisen. Als nächstes wird überprüft, ob der horizontale (x) Abstand der beiden Linien, den Begrenzungen von minBoxWidth und maxBoxWidth entspricht. Trifft all dies zu, werden zwei horizontale Linien erzeugt, die oberes-(rot) und unteres-(blau) Ende der beiden vertikalen Linien (magenta) verbindet. Danach wird betrachtet, ob das Linienprofil der oberen und unteren Linie ausschließlich weiße Pixel enthält. Sind die vertikalen Linien verbunden, wird noch der Winkel von oberer und unterer Linie überprüft. Ist er innerhalb der von maxAngle vorgegebenen Grenze, wird ein Rectangle erzeugt und zur ArrayList boundingBoxes hinzugefügt. Anschließend wird die innere schwarze Box mit der Methode innerBlackBox() gesucht und zu einer weiteren ArrayListe blackBoxes hinzugefügt. Danach werden die Boxen beider Listen in das Originalbild gezeichnet. Die äußeren Bounding Boxen werden in gelb, die inneren schwarzen in cyan gekennzeichnet.

4 `colorThresholdBinary`

Das Original RGB Bild wird zunächst kopiert und der Kontrast erhöht. Die Arrays `min`, `max` und `filter` enthalten Einstellungen für die akzeptierten Grenzwerte. Das Bild wird in drei Grauwertbilder zerlegt, jeweils für Farbton, Sättigung und Helligkeit. Diese werden umbenannt, um sie nacheinander mit For Schleife zu bearbeiten. Innerhalb der For Schleife werden die Grauwertbilder mit minimalem und maximalem Grenzwert binarisiert. Nach dem die Schleife durchlaufen wurde, werden die drei binären Ergebnisbilder mit der `ADD` Funktion des `ImageCalculators` zusammengeführt und geschlossen. Rückgabewert ist das zusammengesetzte Ergebnisbild (Binär).

5 `innerBlackbox`

Mit der Eingabe: `awt.Rectangle` werden die äußeren Grenzen des weißen Rahmens festgelegt. Die Methode sucht innerhalb des äußeren weißen Rahmens das schwarze Quadrat, das die kodierten Informationen enthält. In einer geschachtelten For Schleife, wird mit einem gewissen Abstand (`int padding`) zum äußeren Rand nach schwarzen Pixeln gesucht. Dabei wird das `padding` verwendet, um die Erkennung der inneren Box auch bei perspektivisch verzerrten Bildern zu verbessern. Innerhalb der Schleifen werden die äußeren Grenzen der schwarzen Box ermittelt. Am Ende der Methode wird ein neues `Rectangle` mit den gefundenen Grenzen konstruiert und zurückgegeben.

6 verworfener Ansatz

Der erste Ansatz basiert auf das Auffinden von Linien, dabei benutzen wir die Funktionen `Find Edges` und `MakeBinary`. Mit dem Wand (Zauberstab) werden die Pixel ausgewählt und überprüft, ob sie der Vordergrundfarbe entsprechen, und den festgelegten Größenbeschränkungen entspricht. Aufgrund der Verwendung von Zufallszahlen und `FloodFiller`, welches zur Markierung diente, war dieser Ansatz jedoch nicht zuverlässig und performant genug.