Titel: Barcode Detection and Recognition in Non-constrained Systems

Ausarbeitung

Veranstaltung:
Begleitendes Praktikum zu Computer Vision WS 2016/17

Themensteller:	**Prof. Dr. Xiaoyi Jiang**
	Dimitri Berh
	Andreas Nienkötter
Betreuer:	Aaron Scherzinger
Verfasser:	**Aleksej Matis**
	Alexander Schlüter
	Kjeld Schmidt

Titel: Barcode Detection and Recognition in Non-constrained Systems

Aleksej Matis, Alexander Schlüter, Kjeld Schmidt

24. März 2017

Contents

1 Einleitung

- Motivation
- Barcodes und EAN13
- Aufteilung in Lokalisation+Boundary+Reading

2 Lokalisierung

In order to read a barcode, we first have to localize it in a given image. We implemented two approaches. Gradient+Blur is a simple algorithm, with which we had some initial success. We later discarded it in favor of the much more successful Line Segment Detector approach. Both are detailed in this chapter.

2.1 Gradient+Blur

Our initial localization attempt showed limited success, but is conceptually very simple. The Gradient+Blur approach seeks to exploit the strong boundaries between the (white) background area and black bars in a standard barcode. In a few words; Detect edges on the image, then blur the result to remove noise and apply a threshold to the result. Open and close the resulting shapes, then select the biggest one. We will now go trough the steps in more detail.

1. Edge Detection

   We begin by detecting edges with the Sobel operator. We calculate the absolute gradient, discarding direction information. This leads to slightly worse results for barcodes rotated by about 45 degrees. The resulting image will usually contain large amounts of noise. This is fixed in the next step.

2. Blur+Threshold

   Next, we blur the previous result to smooth out the noise and apply a threshold afterwards. The blur is a normalized box filter of size 9. Note that 'noise' here refers not only to image imperfections, but also noise in the absolute gradient which might be a result of complex textures or other sources. The threshold is not set as a constant brightness of 127. Instead, we calculate the mean brightness of the image, add that value to 255 and divide by 2. This simple extra step gives slightly better results when dealing with very noisy images.

3. Close Forms

   The next two steps are very similar operations. First we *close* the remaining white areas. This means we first dilate the image, then erode it by the same amount. Dilation creates a new image in which a pixel is set to white if there is at least one white pixel within a given radius white in the original image. More intuitively, white pixels grow outwards. Erosion is the opposite operation; A pixel is set to black, unless all pixels in the radius are white. This *closes* empty spaces between neighboring and inside of structures, while structures with no neighbors will remain unchanged. At this point, the barcode will ideally have merged into a solid shape,

4. Open Forms

   Opening the forms is similar to closing, but in reverse order; First eroding, then dilating by the same amount. This will eliminate small structures (which is not actually impotant at this point), while straightening edges on larger structures.

5. Selection

   Finally, we select the largest structure to be the most likely location of the barcode in the original image. In most cases, our result will be a jagged shape, only roughly tracing a rectangle. To simplify matters going forward, we now find the minimal area rectangle around the selected structure. This will get passed into the boundary detection step.

2.2 LSD

This localization method is based on the LineSegmentDetector algorithm [**GromponevonGioi2012**], which is implemented in OpenCV 3 in the cv::LineSegmentDetector class [**Bradski2017**]. The algorithm detects line segments in an image, such as the segments formed by the bars in a barcode. On a high level, it works by first calculating the image gradient and assigning to each pixel a unit vector perpendicular to the gradient, resulting in a *level line field*. Pixels with a gradient magnitude under a certain tolerance are discarded. Connected pixels with similar level line angles are clustered together and form a *line support region*. To each line support region, a rectangle is fit which covers the whole region. To limit the number of false detections, the rectangle is only accepted as a line segment if the number of covered pixels with aligned level lines is high enough relative to the total number of covered pixels. The exact threshold is based on a statistical model (*a contrario* method), so that the rectangle is only accepted if, in a purely random level line field, the found ratio of aligned level lines is unlikely. Before a rectangle is rejected, some variations of the rectangle's parameters are tried.

The barcode localization method is taken from **Creusot2016** with minor modifications. First we run the LineSegmentDetector provided by OpenCV on the image. We use default parameters, except for the first parameter LSD_REFINE_NONE, which disables refinement of line segments. We found that line refinement tends to break up barcode bars into multiple segments, e.g. if the bars in the image are not quite straight due to kinks in the material on which the barcode is printed, or if there are glare spots on the barcode. This is undesirable, since in the following steps we rely on the length of line segments in the barcode to be similar.

Next we want to find a line segment which belongs to a barcode bar approximately in the middle of the barcode. To each detected line segment, a score is assigned based on how many other line segments might belong to bars of the same barcode as the first one. Two line segments are regarded as possibly belonging to the same barcode, if

1. their centers are not too far apart,
2. they have similar length,
3. they have similar angles and
4. their projected intersection covers most of the smaller segment.

Calculation of the first three distance measures is straightforward: Let c_i be the center positions, l_i the segment lengths and α_i the segment angles, where the index $i = 1, 2$ denotes the first or the second line segment, respectively. The used criteria are

$$d_{\text{center}} := \frac{\|c_1 - c_2\|}{l_1} < 1, \qquad \frac{|l_1 - l_2|}{l_1} < 0.3, \qquad |\alpha_1 - \alpha_2| < 0.1\,.$$

As BIIIIIIILD shows, two lines fulfilling the first three criteria might still not be arranged like barcode bars, if they are displaced along the line segment direction. To catch these cases, we project the second line onto the first and check that the intersection is large enough relative to the length of the smaller line, see BIIIIIIILD2.

For each line segment which fulfills the criteria, **Creusot2016** increment the score of the first line by one. Instead, we increment the score of the first line segment by $1 - d_{\text{center}}$. This effectively weights the score by the distance between the line segments, which favors line segments which lie in the middle of the barcode.

Finally, we select the line segment with the highest score. This line segment should correspond to a barcode bar in the middle of the barcode.

3 Boundary Detection

To complement the LSD localizer, which returns a line segment in the middle of the barcode (hereafter called *best line*), we need to find the barcode's boundaries before we can attempt to read it.

3.1 Variation

The variation boundary finder is the original boundary detection method proposed in [**Creusot2016**] in conjunction with the LSD localization. Let $L_\perp$ denote the sequence of intensity values along the line perpendicular to the best line, going through the best line's center (hereafter called *bisector*). Since barcodes consist of alternating black and white bars, the variation of intensity values along a line crossing the barcode is expected to be high. The boundary points are thus expected to have the following property: Extend a “probing” line (*inner probe*) from the boundary point along the bisector, pointing inside the barcode, i.e. in direction of best line's center. Extend a second “probing” line (*outer probe*) from the boundary point in the opposite direction, away from the barcode. The variation along the inner probe is now expected to be much higher than the variation along the outer probe.

To find the boundary points, for each index k along the bisector, the signed variation difference

$$\phi_{L_\perp}(k) = \sum_{j=-R}^{0} |L_\perp(k+j) - L_\perp(k+j-1)| - \sum_{j=0}^{R} |L_\perp(k+j) - L_\perp(k+j+1)|$$

is computed. Here, R is the probing distance, which we chose as $R = l_*/2$, half of the length of the best line.

The right boundary point should lie at the bisector index with maximal $\phi_{L_\perp}$, while the left boundary point should have minimal $\phi_{L_\perp}$.

We deviate slightly from **Creusot2016** by instead using the variation measure

$$\phi^*_{L_\perp}(k) = \sum_{j=-R}^{0} |L_\perp(k+j) - L_\perp(k+j-1)| \cdot (R+j) - \sum_{j=0}^{R} |L_\perp(k+j) - L_\perp(k+j+1)| \cdot (R-j)\,,$$

giving higher weight to pixel values near the current boundary candidate at index k.

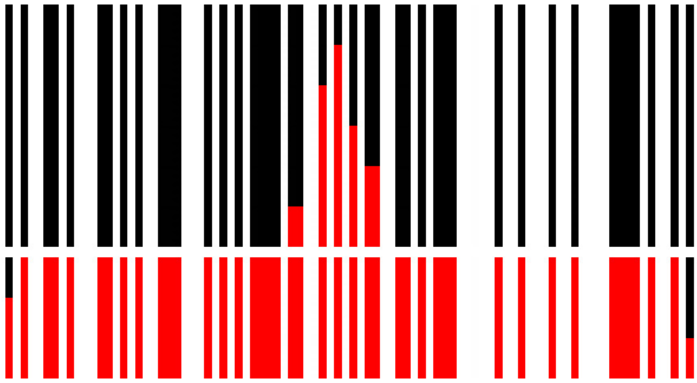

Figure 1: Wachenfeld

3.2 LSD Bound

Motivated by the high quality of line segments returned by LineSegmentDetector, we constructed a new boundary finder, using ideas from the variation method described above. The goal is to determine the line segments corresponding to the two outer barcode bars, the *boundary segments*.

We first filter the line segments, so that only segments parallel to and located next to the best line are kept. To that end, we use the angle criterion and the criterion of sufficient projected intersection as described in section 2.2. The remaining segments are sorted based on the position of the projection of their centers onto the bisector line. The boundary segments should have the following property: As before, extend an inner probe from the projected center of the boundary segment along the bisector, pointing inside the barcode. Extend an outer probe in the opposite direction, away from the barcode. The inner probe is expected to cross a large number of line segments parallel to and located next to the best line (i.e. segments which have not been filtered out in the previous step). The outer probe is expected to cross fewer such lines. ...

3.3 Wachenfeld [wachenfeld2008robust]

Abbildung 1

4 Lesen

Hier nur Template Matching

	Fehler	Prozent Korrekt	Zeit in Sekunden
Gradient+Blur	831	22%	266
LSD+Wachenfeld	452	56%	319
LSD+LSDBounds	41	97%	1760
LSD+Variation	279	74%	1595

Figure 2: Vergleich der Laufzeit und Genauigkeit der Methoden

5 Vergleich

Einleitun...

5.1 Datasets

übersicht über die Datasets

- Generiert
- Wachenfeld [**wachenfeld2008robust**]
- ArteLab [**zamberletti2010neural**] [**zamberletti2013robust**]

5.2 Laufzeit

Geprüft auf dem Wachenfeld Dataset mit 1055 Bildern [**wachenfeld2008robust**]
Laufzeit gemessen auf Debian 8 mit 4Gb Ram, 2.2 Ghz CPU, 4 Kerne und 4 Threads
Siehe Abbildung 2

6 Verbesserung

7 Fazit

8 Meta: Wie zitiere ich?

1. Titel des Papers bei `https://scholar.google.de/` Google Scholar suchen.
2. Bei dem Eintrag zu dem Paper unten auf zitieren klicken, dann auf BibTex.
3. Den BibTex string kopieren in die LITERATUR.BIB
4. Zitat hinzufügen durch \CITE{*name*}
5. Übersicht BibTex: `https://de.wikibooks.org/wiki/LaTeX-Kompendium:_Zitieren_mit_BibTeX`

Beispiel Templatematching [**chen2014scanning**]

Einleitung Lokalisierung Rand Lesen Testdaten Vergleich der Verfahren Verbesserung