Adaptive Textzeilensegmentierung zur Lösung des Vice Versa-Problems

Marco Block, Marte Ramírez Ortegón, Manuel Siebeneicher, Alexander Seibert, Raúl Rojas
[block|marte|siebenei|seibert|rojas]@inf.fu-berlin.de

Januar 2007

I. Einführung

Segmentierung von Textzeilen ist ein notwendiges Vorverarbeitungsverfahren für Texterkennung. Um die Textzeilen segmentieren zu können, müssen diese horizontal orientiert sein. Möglicherweise ist dafür eine Verformung (*warping*) des Bildes als Vorabschritt notwenig ([3], [8]). Das Dokument kann aber noch um 180° verdreht vorliegen. Dieses Problem bezeichnen wir als das „vice versa"-Problem. Normalerweise ist es Aufgabe der Texterkennung, dieses Problem zu lösen, was zeitaufwendig sein kann. Es ist aber bereits bei der Separierung der Textzeilen möglich, die Orientierung des Textes zu erfassen.

Im folgenden beschreiben wir eine robuste Methode, um Textzeilen zu separieren. Als Nebenprodukt unserer adaptiven Segmentierung ergibt sich eine gute Heuristik für das „vice versa"-Problem. In Kombination mit den SITT-Features läßt sich diese Heuristik weiter verbessern.

II. Allgemeine Zeilensegmentierung

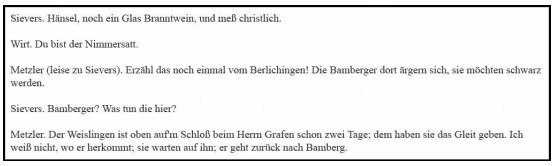
Bisher wurde der Zeilensegmentierung von digital erzeugten Texten nicht allzu viel Beachtung geschenkt, da es sich um eine sehr intuitive und rudimentäre Methode für die Vorverarbeitung einer Texterkennung handelt. In vielen Arbeiten wird sie eher am Rande erwähnt, wie beispielweise bei der Mosaikerstellung verschiedener Dokumente in [9] oder bei dem *bottom-up* Ansatz in [4].

Anders sieht es bei handgeschriebenen Texten aus. Es gibt ein sehr großes Interesse daran, schnellere und bessere Algorithmen für dieses Problem zu formulieren ([6], [5]). Eine Einführung bietet [7].

Es läßt sich viel Zeit bei der Texterkennung sparen, wenn die richtige Orientierung der Textzeilen bereits vorliegt. Daher war es unser Interesse, schon in einer frühen Vorverarbeitsphase diese Orientierung zu ermitteln.

III. Adaptive Zeilensegmentierung

Wir werden das Verfahren anhand eines Beispiels erläutern und im Abschnitt IV eine Bewertung des Systems mit schwierigen Beispielen und Ausnahmen vornehmen.


Sievers. Hänsel, noch ein Glas Branntwein, und meß christlich.

Wirt. Du bist der Nimmersatt.

Metzler (leise zu Sievers). Erzähl das noch einmal vom Berlichingen! Die Bamberger dort ärgern sich, sie möchten schwarz werden.

Sievers. Bamberger? Was tun die hier?

Metzler. Der Weislingen ist oben auf'm Schloß beim Herrn Grafen schon zwei Tage; dem haben sie das Gleit geben. Ich weiß nicht, wo er herkommt; sie warten auf ihn; er geht zurück nach Bamberg.

Fig. 1. Beispiel einer Textpassage.

Interessant sind im folgenden nur die prägnanten Teile eines Textes, dazu wird das Bild I binarisiert [2] und anschließend ein Histogramm H über alle Zeilen berechnet.

Für ein Bild I_k liefert $I(x,y)$ die Intensität an der Stelle (x,y), mit $x \in \{0,1,...,\widetilde{x}_k - 1\}$ und $y \in \{0,1,...,\widetilde{y}_k - 1\}$, wobei $\widetilde{x}_k$ und $\widetilde{y}_k$ die maximale Anzahl der Bildpunkte in x- und y-Richtung sind. Die histogrammbasierte Segmentierung heißt auch Schwellenwertsegmentierung [7]. Die Pixel werden zeilenweise aufsummiert:

$$H(y) = \Sigma_{i=0}^{\widetilde{x}-1} I(i,y) \tag{1}$$

Als Voraussetzung für eine Segmentierung der Zeilen ist die vertikale Orientierung der Textzeilen gegeben (Histogramm des Bildes aus 1):

In Abbildung 2 sind die Textzeilen bereits sehr gut erkennbar. Wenn die Zeilen nicht horizontal ausgerichtet wären, würde das Histogramm für das Beispiel in Abbildung 3 wie in Abbildung 4 gezeigt, kein gutes Ergebniss liefern.

Nach Betrachtung zahlreicher Histogramme (in über 70% aller unserer Zeilenbeispiele) konnten wir eine naheliegende Eigenschaft identifizieren: ein kleines Maximum unterhalb einer Textzeile.

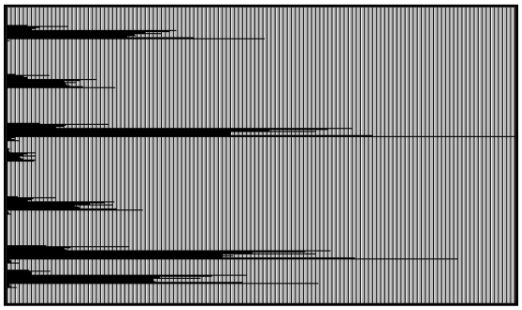


Fig. 2. Vertikales Histogramm

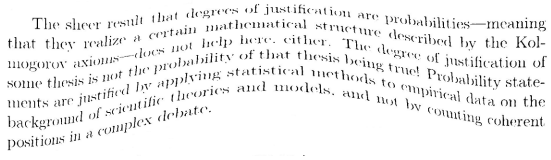


Fig. 3. Verformter Text in einem identifizierten Textblock.

Dieses Maximum wird durch die Textbögen von Kleinbuchstaben, wie 'g', 'j' usw. erzeugt.

Nach Beobachtung dieser Eigenschaft, läßt sich nun ein ideales Modell für Textzeilen (die lateinische oder z.B. kyrillische Buchstaben beinhalten) aufstellen.

A. Ideales Modell

Ein unter einer Zeile auftretendes Maximum liefert eine sehr gute Heuristik für die Textausrichtung und damit eine Lösung für das „vice versa"-Problem. Tritt ein lokales Maximum zwischen den

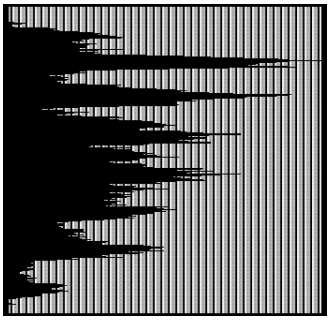


Fig. 4. Vertikales Histogramm für die Abbildung 3

Fig. 5. Textbögen unterhalb der Haupttextaxe erzeugen ein lokales Maximum im Histogramm H.

Textzeilen auf, so kann anhand der Nähe zu den Textzeilen eine erste Entscheidung getroffen werden. In den meisten Fällen werden mehrere Textzeilen aus demselben Dokument in dieser Funktion verarbeitet, sodass durch Mehrheitsentscheid eine sichere Prognose vorliegt.

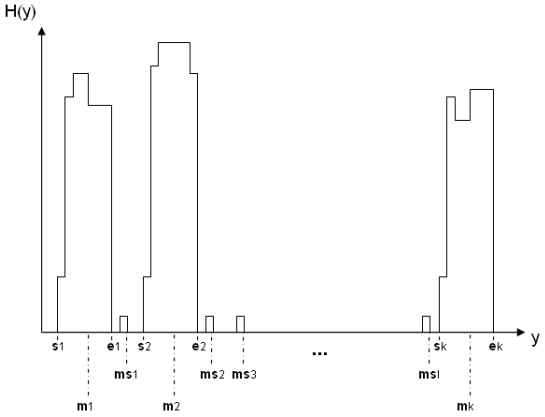


Fig. 6. Ideales Modell für das Histogramm H. Die Zeilenzentren $m_1,...,m_k$ und die dazugehörigen ms_i, falls existent sind dargestellt. Jeder Zeilenkörper um m_i im Histogramm besitzt einen Startpunkt s_i und einen Endpunkt e_i.

Wir definieren die gefundenen Textzeilen (ohne die kleinen Maxima) als (s_i, m_i, e_i), mit s_i und e_i sind jeweils der linke und rechte Rand und m_i als Gewichtszentrum der Zeile i für alle $i = 1,...,k$. Ferner gelte $s_i < m_i < e_i$. Kleine lokale Maxima, die nicht als Textzeilen identifiziert wurden, werden nur durch ihr Zentrum ms_j repräsentiert, mit $j = 1,...,l$. Anschliessend werden die kleinen Maxima ms_j mit der Funktion R den nächstgelegenden Textzeilenzentren m_i zugeordnet (siehe dazu Abbildung 6).

$$R(ms) = min|m_i - ms|, \forall i = 1,...,k$$

Dann können wir die Textrichtung bestimmen, indem wir zählen, wie oft ms_i auf derselben Seite von m_i vorkommt:

$$dir_H = \sum_i sgn(R(ms_i) - ms_i), \tag{2}$$

mit $i = 1,...,l$ und

$$sgn(x) = \begin{cases} +1 & , wenn\ x \geq 0 \\ -1 & , wenn\ x < 0 \end{cases}$$

als Vorzeichenfunktion. Sollte nun gelten, dass $dir_H < 0$ ist, so muss das Dokument um 180° gedreht werden.

B. Implementierung der adaptiven Textzeilensegmentierung

Die Implementierung ist einfach und die Zeilensegmentierung arbeitet auch auf großen Bildern sehr schnell.

Nach der Berechnung des Histogramms H, wird H in einem Verarbeitungsschritt geglättet. Anschließend wird ein Schwellwert Θ in Abhängigkeit zum Modus, d.h. zum am häufigsten vorkommenden Wert in H, ermittelt. Dieser Schwellwert ist eine Heuristik, die sich experimentell bewährt hat. Die Textboxen können nun extrahiert werden.

Bei schwellenwertbasierter Segmentierung kann es, trotz Vorverarbeitung und Glättung, durch Rauschen zu fehlerhaft getrennten Zeilen kommen. Diese Fehler werden nachträglich behoben.

```
function segmentLines(Image I) begin
  // Histogramm berechnen
  H = computeHistogram(I);
  H = smoothHistogram(H);

  // Schwellwert berechnen
  εΘ = mode(H) * 2;
  Θ  = min(H) + εΘ;

  // Liniengrenzen grob segmentieren
  // Annahme - mindestens alle Grenzen der
  // Linien werden gefunden
  linesBounds = determineLines(H, Θ);
  linesBounds = correctLines(linesBounds);

  // gefundene Liniengrenzen verbessern
  linesBounds = improveLines(linesBounds);
  linesBounds = correctLines(linesBounds);

  // Zeilen extrahieren
  textBoxes = extractLines(linesBounds, I);
end function
```

Zur Verbesserung der Textboxen werden die Zwischenräume der Textboxen näher analysiert. In diesem Vorgang werden die Grenzen weiter nach außen verschoben. Ähnlich wie bei der Bestimmung der groben Liniengrenzen, wird ein Schwellwert basierend auf dem Modus verwendet. In diesem Fall jedoch nicht auf dem Modus des vollständigen Histogramms, sondern auf dem lokalen Modus. Dadurch können zeilenübergreifende Artefakte eliminiert werden.

```
function improveLines(linesBounds) begin
  gaps = getLinesBoundsGaps(linesBounds);

  // Räume zwischen den gefundenen Zeilen
  // untersuchen
  for gaps do
    // Schwellwert für aktuellen Zwischenraum
    // bestimmen.
    Θcurr = mode(gap) + εΘ;
    // Die zwei Punkte finden, die kleiner
    // als der Threshold sind und am weitesten,
    // höchstens aber n/2, mit |gap| = n, von
    // der oberen bzw. unteren Grenze entfernt
    // sind.
    gap = relocateGap(gap, Θcurr);
  end for
  return computeNewBounds(linesBounds, gaps);
end function
```

C. Histogramm und SITT-Features

SITT-Features als markante Textmerkmale haben sich bereits zur Mosaikerstellung von Bildern bewährt [1]. Angewendet auf das Beispielbild aus 1 ergibt sich die Abbildung 7.

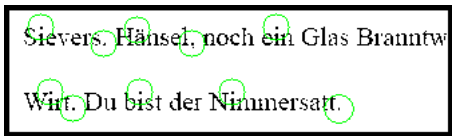


Fig. 7. Ausschnitt der Textpassage mit erkannten SITT-Features.

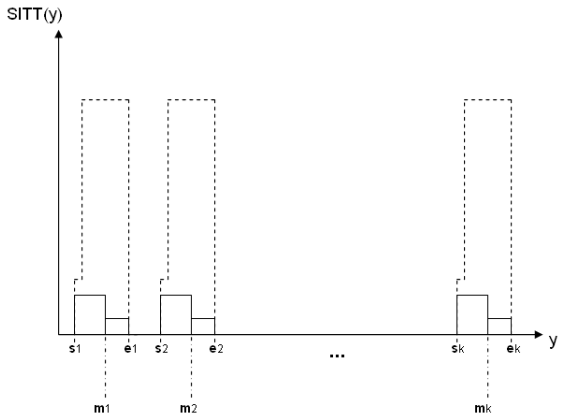


Fig. 8. Über die Abbildung 6 wurde ein weiteres Histogramm gelegt. Die SITT-Features wurden ermittelt und analog zu H zeilenweise im Bild zusammengetragen. Da hier die Y-Achse waagerecht verläuft sind die Werte senkrecht abzulesen. In diesem Histogramm sind alle Werte zwischen s_i und m_i aufsummiert und an jede Stelle zwischen s_i und m_i eingetragen. Das gleiche gilt für die Bereiche zwischen m_i und e_i.

Wenn man diese sehr schnell zu berechnenden Merkmale ebenfalls in das ideale Modell (siehe Abbildung 8) eines Histogramms einträgt, lässt sich wieder eine interessante Beobachtung machen, die eine weitere Eigenschaft zur Festigung unserer Prognose liefert.

D. Heuristik für das „vice versa“-Problem

Auch die SITT-Features haben die Eigenschaft, dass sie eine Heuristik für die Textausrichtung

liefern. Oberhalb von Textzeilen lassen sich mehr Punkte finden, als Satzzeichen.

Analog zu Abschnitt III-A läßt sich durch eine Kombination der beiden Eigenschaften dir_H und den SITT-Features eine robuste Heuristik für das „vice versa“-Problem formulieren:

$$dir_{H+SITT} = dir_H + \sum_i sgn(L(s_i, m_i) - L(m_i, e_i)), \tag{3}$$

mit

$$L(x,y) = \sum_{j=x}^{y} SITT(j)$$

ist die Summe der Werte des SITT-Histogramms zwischen x und y.

Sollte $dir_{H+SITT} < 0$ gelten, so muss das Dokument um 180° gedreht werden, ansonsten ist es richtig herum.

IV. Resultate und Disskussion

Für eine Texterkennung sind viele Vorverarbeitungsschritte notwendig. Durch die adaptive Textzeilensegmentierung ist bereits ein weiteres wichtiges Problem, das „vice versa“-Problem gelöst, dass die weitere Interpretation des Textes verbessert und die Performanz steigern kann. Die schnell zu berechnenden SITT-Features, die sich schon für die Erstellung von Dokumentmosaiken bewährt haben, in Kombination mit einem sehr häufig auftretenden kleinen Maximum unterhalb einer Textzeile, liefern eine robuste Heuristik für die Lösung.

References

[1] Block M., Ramiréz M., Seibert A., Kretzschmar J., Rojas R.: „*SITT-A Simple Robust Scaleinvariant Text Feature Detector for Document Mosaicing*“, Technical Report B-07-02, Freie Universität Berlin, Januar 2007

[2] Ramírez M., Block M., Rojas R.: “*New Robust Binarization Approach in Letters*“, Guadalajara México, CONCIBE 2006

[3] K.B.Chu et al: „*A fast and stable approach for restoration of warped document images*", ICDAR 2005

[4] Mitchell P.E., Yan H.: „Newspaper Document Analysis featuring Connected Line Segmentation“, Sydney, VIP 2001

[5] Nicolas S., Paquet T., Heutte L.: “*Text Line Segmentation in Handwritten Document Using a Production System*“, V(00):245-250, IWFHR 2004

[6] Li Y., Zheng Y., Doermann D., Jaeger S.:“*A New Algorithm for Detecting Text Line in Handwritten Documents*“, IWFHR 2006

[7] Tönnies K.D.: „*Grundlagen der Bildverarbeitung*“, Pearson Studium 2005

[8] Schneider D.: "*Perspective and Curvature Correction for Camera-based Document Processing*", Magisterarbeit Freie Universität Berlin, April 2006

[9] Zappalá A., Gee A., Taylor M.: „*Document mosaicing*“, Image and Vision Computing(17):589-595, 1999