Kompakte Graphmodelle handgezeichneter Bilder
Einbeziehung in Authentifizierung und Bilderkennung

Seminar Mustererkennung WS 2006/07
Autor: Stefan Lohs

Inhaltsverzeichnis

1 Einleitung

Handgeschriebene Texte und Bilder werden heutzutage in vielen Bereichen eingesetzt, zum Beispiel in sicherheitsrelevanten Anwendungen, bei Banken zum Vergleich von Unterschriften oder bei Texterkennungsprogrammen. Meist stehen die Anwendungen in Zusammenhang mit portablen Geräten wie PDAs oder Handys. Da eingescannte Bilder meist als Bitmap vorliegen, ist es wichtig, einen Kompressionsalgorithmus zu finden, der die wichtigen Daten erhält, aber die Größe stark reduziert, um Datentransmissionen und Datenbanken möglichst klein zu halten. Die hier beschriebene Vorgehensweise erzeugt ein mathematisches Modell, dass diese Bedingungen erfüllt.

Um eine Unterschrift zu überprüfen muss ein System aus folgenden Komponenten bestehen:
◦ Einem Scanner und einem Programm um das Bild zu empfangen.
◦ Einem Programm zum komprimieren, dass mit einer Datenbank verbunden ist.
◦ Einem Programm, um das Bild wiederherzustellen.

Zur Automatisierung sind folgende Schritte notwendig. Scanne das Bild von einem Dokument und erzeuge ein mathematisches Modell. Speichere das Bild in einer Datenbank. Wird das Bild für einen Vergleich benötigt wird es über den Schlüssel aus der Datenbank extrahiert. Aus dem Graph wird nun das Bild wiederhergestellt und kann verglichen werden.

2 Das graphische Modell

Die hier vorgestellte Transformation besteht aus fünf Teilen:
1. Image Thinning - Bildverdünnung
2. Bitmap zu Graph Transformation
3. Shape Smoothing - Konturenglättung
4. Graphminimierung
5. Konturenkompression

2.1 Image Thinning

Die „Skelettierung“ überführt die Konturlinien in einzelne Pixellinien mithilfe des „modifizierten Naccache - Shinghal Algorithmus. Dabei wird die durchschnittliche Dicke der Linien verringert.

2.2 Bitmap zu Graph Transformation

Wir führen einen Pixel-Index ein, welcher der Anzahl an benachbarten Pixeln in einem 3x3 Feld um ein Pixel entspricht. Ein Pixel mit Index 1 ist ein Endpunkt. Ein Linien-Punkt hat den Index 2 und mit 3 oder 4 ist es ein Knoten. Der erzeugte Graph wird in 2 Listen gespeichert. Die Liste der Knoten enthält die X,Y-Koordinaten des Bitmap-Pixels und einen Zeiger auf den dazugehörigen Konturzweig. Die Konturenzweigliste ist eine Kette von 8-bit Codezeichen (Freeman code) die die nächste Konturrichtung beschreiben. Nach diesem Schritt bleiben nur noch Knoten mit dem Index 1, 3 oder 4 übrig, da Pixel mit Index 2 durch die Konturzweige beschrieben werden (Abb. 1).

Abbildung 1: 3x3 Feld um ein Pixel,Quelle:[1]

2.3 Shape Smoothing

Durch das Einscannen kann es bei dem Bild zu Verzerrungen oder Rauschen kommen, was bei der Verdünnung zu falschen Knoten führt. Diese Defekte haben zwar keinen wirklichen Einfluss auf die Qualität des Bildes, verbrauchen aber unnötigen Speicherplatz. Deshalb wird der Graph geglättet und diese Knoten beseitigt. Dabei werden falsche Knoten gelöscht (a,b), unterbrochene Konturen geglättet (c), alle Knoten mit dem Pixelindex 4 gelöscht (d) und einige von den Knoten mit dem Pixelindex 3 entfernt (e) (Abb. 2). Wenn man den Algorithmus verlustfrei halten möchte, kann man das Glätten überspringen, aber das Löschen aller Knoten mit dem Index 4 kann immer verlustfrei gemacht

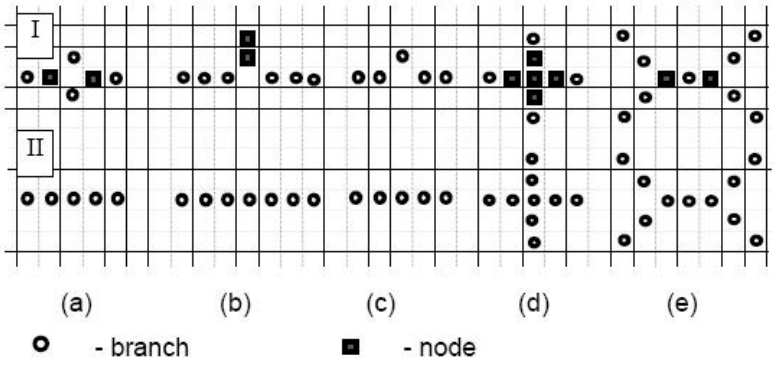

Abbildung 2: Löschen von unnötigen Knoten, Quelle:[1]

werden. Somit haben wir nach dem Smoothing nur noch Knoten mit dem Index 1 oder 3.

2.4 Graphminimierung

Der Graph, der im 2. Schritt entstanden ist, ist redundant, daher ist es möglich, den Graph zu minimieren, indem man Knoten löscht und Zweige verbindet. Zum Schluss hat jeder Zweig nur noch einen Knoten als Startpunkt für die Pixelkette.

Lemma: Vor der Minimierung ist jeder Zweig des Graphen mit 2 Knoten mit Pixel-Index 1,3 oder 4 verbunden.

Theorem: Die Anzahl B der Zweige des Graphen ohne Schleifen kann folgendermaßen berechnet werden:

$$B = \frac{I_1 + I_2 + .. + I_N}{2}$$

mit: N ist Anzahl der Knoten des Quellgraphen, I_j ist der Index des j-ten Knoten (außer Knoten mit Index 4)

Corollar 1: Aus dem Theorem folgt, dass die Anzahl der Zweige mit Schleifen folgendermaßen berechnet werden kann:

$$B = \frac{N_1 + 3 \cdot N_3}{2} + N_{loops}$$

mit: N_1 ist Anzahl der Knoten mit Index 1,
N_3 ist Anzahl der Knoten mit Index 3,
N_{loops} ist die Anzahl der beliebig wählbaren Knoten mit Index 1 um alle Schleifen zu erreichen.

Die Graphmimierung kann Knoten mit dem Index 1 und 3 eliminieren und minimiert die Anzahl der Zweige durch Reduzierung der Knoten mit Index 3. Daher kann Corollar

1 für die minimale Anzahl an Knoten und Zweigen umformuliert werden.

Corrolar 2: Die Anzahl der Zweige im minimalen Graph ist:

$$B^{min} = \frac{N_1^{min} + 3 \cdot N_3^{min}}{2} + N_{loops}$$

mit: N_1^{min} und N_3^{min} ist die minimale Anzahl an Knoten mit Index 1 und 3.

Die Graphminimierung wird durch das Spannbaum-Problem mit folgenden Schritten gelöst.

Schritt 1: Wähle einen Knoten mit dem Index 3. Verfolge 3 unverbundene Pfade, bis man einen bereits verbundenen Knoten findet oder einen mit Index 1. Markiere den Knoten und alle Zweige mit „verbunden".

Schritt 2: Wiederhole Schritt 1, bis alle Knoten mit Index 3 verbunden sind.

Schritt 3: Wähle einen Knoten mit Index 1 und verfolge den Zweig. Markiere den Knoten und den Zweig als „verbunden".

Schritt 4: Wiederhole Schritt 3, bis alle Knoten mit Index 1 verbunden sind.

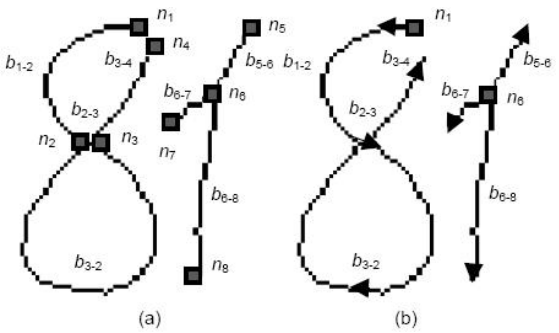

Abbildung 3: Minimierung, Quelle:[1]

2.5 Konturkompression

Nach dem Verdünnen und dem Glätten hat der Graph keine scharfen Richtungsänderungen mehr, und somit kann der Nachfolger im Zweig nur noch in 3 Richtungen liegen. Aus diesem Grund können wir die Kette des Zweiges mit relativen Koordinaten beschreiben (-1,0,1). Daher brauchen wir nur $N \log_2 3$ bits statt $8N$ bits für eine Kette der Länge N. Dazu werden noch alle Zweige mit dem modifizierten „Run-Length Encoding algorithm" komprimiert.

3 Wiederherstellung des Bildes

Die Wiederherstellung wird auf Basis eines Vektormodells durchgeführt. Man erzeugt das Gerüst, indem man durch die Knoten des Graphen läuft und die Zweige wieder in ein

Bitmap transformiert. Durch diese Methode kann man zwar das ursprüngliche Bild nicht wiederherstellen, aber das hat keinen Einfluss auf das Vergleichen von zwei Bildern oder Unterschriften. Um eine exaktere Wiederherstellung machen zu können, müsste man sich die Liniendicke für jeden Punkt mit abspeichern(Abb. 4).

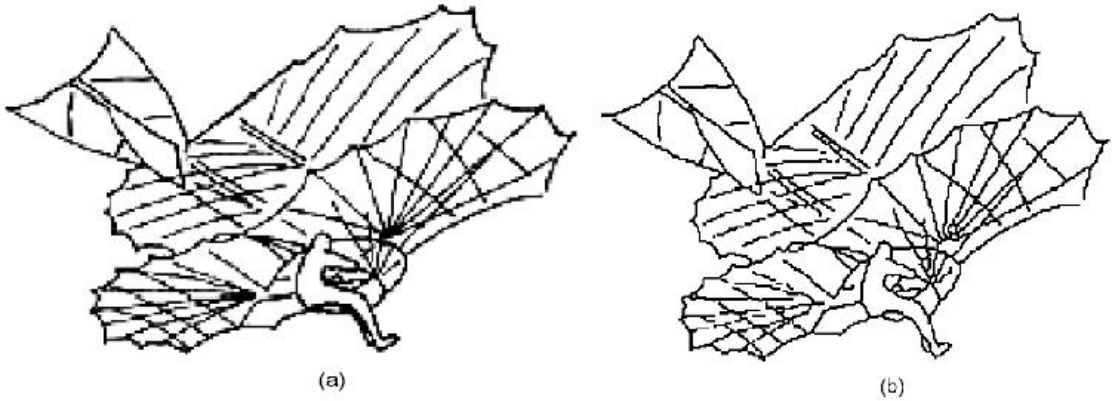

Abbildung 4: Vergleich Original - Modell, Quelle:[1]

4 Ergebnis

Experimente haben ergeben, dass der Algorithmus beim Speichern von Unterschriften 10-20 fach, bei handgeschriebenem Text 5-12 fach und bei Bogenformen 8-17 fach komprimiert hat. Bei Konturen ist er mit 11.9 facher Kompression ungefähr 4 mal besser als Rasterkompression. Vergleich mit „CorelTrace“ (Abb. 5): Bei einem zweiten Expe-

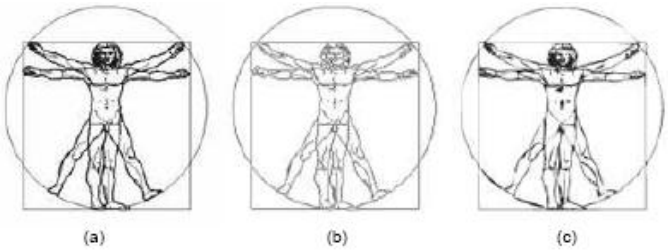

Abbildung 5: a)Origninal b)vorgestellter Algorithmus c)CorelTrace, Quelle:[1]

riment wurde die Erkennung von handgeschriebenen Zahlen untersucht. Dabei lag der maximaler Fehler bei der Erkennung bei 5,9 Prozent.

Literatur

[1] Denis V. Popel, „Compact Graph Model of Handwritten Images: Integration into Authentification and Recognition, LNCS 2396 (2002), S. 272-280“