Effiziente Fingerabdrucksuche mit Hilfe von Database Clustering

Robert Meiche

31. Mai 2007

1 Vorwort

Diese Ausarbeitung bezieht sich zum größten Teil auf den Artikel „Efficient fingerprint search based on database clustering“ von M. Liu, X. Jiang und A. Chichung Kot. Alle zusätzlich benutzten Quellen werden direkt im Text angegeben. Alle anderen Ergebnisse, Formeln etc., die nicht explizit mit einer Quelle angegeben sind, beziehen sich bzw. stammen vom oben genannten Artikel.
Des Weiteren wird in dieser Ausarbeitung mehrmals das „fine matching“ erwähnt. Ich konnte weder in dem genannten Artikel noch irgendwo anders eine genaue Definition finden. Ich gehe davon aus, dass es sich dabei um den direkten Bildvergleich handelt. Es werden dann zwei Bilder direkt durch Extratkion von z. Bsp Minutienpunkten verglichen.

2 Einleitung

Bei Systemen zur Fingerabdruckerkennung unterscheidet man generell zwischen zwei verschiedenen Modi. Zum einen gibt es die Authentifizierung und zum anderen die Identifikation. Bei der Authentifizierung wird der Fingerabdruck vom System nur mit dem entsprechenden Abdruck aus der Datenbank verglichen. Dieser kann direkt in der Datenbank gefunden werden, da zusätzlich zum Fingerabdruck noch eine weitere Identifikationsinformation (z. Bsp. Name und Geburtsdatum auf einer Chipkarte) zur Verfügung steht.
Bei der Identifikation hingegen muss zu einem Fingerabdruck die weitere Identitätsinformation gefunden werden, indem mit allen Fingerabdrücken der Datenbank verglichen werden muss. Um dieser Komplexität gewachsen zu sein (bei Datenbanken mit mehr als 50 Mio. Abdrücken), muss man sehr effiziente Verfahren zur Repräsentation und Aufbereitung der Daten verwenden.
Ein solches Verfahren ist die Fingerabdrucksuche mit Hilfe von Database Clustering, welches in den nächsten Kapiteln vorgestellt wird.

3 Klassifizierung der Daten

Um eine große Datenmenge effizient zu durchsuchen, muss man versuchen, die Anzahl der Vergleiche zu minimieren. Hierzu teilt man die Daten in Klassen ein, um dann nachher über entsprechende Kriterien die Suche auf möglichst wenige Klassen einzuschränken. Es gibt verschiedene Möglichkeiten der Klassifikation, die nachfolgend erläutert werden.

3.1 Exklusive Klassifikation

Bei der exklusiven Klassifikation wird der Fingerabdruck genau einer Klasse zugeordnet. Ein Beispiel hierfür sind die so genannten Henry-Klassen. Diese unterteilen die Daten exklusiv anhand bestimmter menschlicher Merkmale. Ein Problem dieser Methode ist, dass die meisten automatisierten Klassifikationsalgorithmen die Fingerabdrücke lediglich in vier oder fünf Klassen einteilen. Weiterhin ergibt sich im Allgemeinen eine sehr schlechte Verteilung der Daten auf diese Klassen (teilweise nur 3% in einer Klasse), wodurch im Durchschnitt auf 29,48% der Fingerabdrücke ein so genanntes *fine-matching* durchgeführt werden muss.
Vor allem wegen der ungleichmäßigen Verteilung auf sehr wenige Klassen eignet sich diese Klassifikation nicht dafür, die Zahl der Vergleiche auf großen Datenbeständen zufrieden stellend zu vermindern.

3.2 Kontinuierliche Klassifikation

Bei dieser Art der Klassifikation versucht man das Problem der exklusiven Klassifikation zu beheben, indem man mehr Klassen erstellt. Dies wird erreicht durch Repräsentation der Fingerabdrücke durch numerische Vektoren. Diese Vektoren können z. Bsp. Minutienpunkte oder andere Informationen, mit denen man durch Vergleiche mit anderen numerischen Vektoren Fingerabdrücke identifizieren kann, enthalten. Da der Vergleich von solchen Vektoren wesentlich schneller geht wie das *fine-matching*, führt dies bei der Suche mit Hilfe der Kontinuierlichen Klassifikation zu einer besseren Performance. Allerdings eignet sich das immer noch nicht, um effektiv große Datenbanken zu durchsuchen. Ein weiterer Nachteil dieser Klassifikation, der sich negativ auf die Such-Performance auswirkt, ist die Tatsache, dass die Vektoren der Datenbank nur an Hand der Gemeinsamkeiten mit dem Suchvektor eingeteilt werden. Es werden nicht die Gemeinsamkeiten unterhalb der Daten herausgestellt um diese dann mit der Sucheigenschaft abzugleichen.

3.3 Data-Clustering

Bei dieser Methode versucht man die Vorteile beider vorherigen Methoden zu vereinen. Zum einen möchte man einen Abdruck exklusiv einer Klasse zuordnen und zum anderen möchte man die Datenmenge in möglichst viele Klassen gleichmäßig verteilen. Ein großer Vorteil des *Data-Clustering* ist es, dass alle Klassen (bei diesem Verfahren *Cluster* genannt) einen Repräsentanten haben. Dies bedeutet, dass nicht mit allen Mustern der Datenbank verglichen werden muss, sondern zuerst werden nur die Repräsentanten mit dem entsprechenden Fingerabdruck verglichen. Daraufhin wird die Anzahl der Vergleiche schon sehr stark vermindert. Des Weiteren lassen sich dann die einzelnen Cluster noch mal unterteilen um eine weitere Verringerung der Vergleiche zu erzielen. Wie diese Art der Aufteilung funktioniert und welche Eigenschaften eines Fingerabdrucks zur Suche herangezogen werden bei der Suche mittels *Database-Clustering,* wird im weiteren noch genauer erläutert.

4 Fingerabdrucksuche mittels Database-Clustering

Wie in Abschnitt 3.3 schon erwähnt, wird versucht die Anzahl der Vergleiche zu vermindern indem zuerst einmal nur mit Repräsentanten der einzelnen Cluster verglichen wird. Um eine solche Unterteilung der Daten zu erreichen, nutzt das hier vorgestellte Verfahren Ähnlichkeiten zwischen den Eigenschaften der Fingerabdrücke. Es werden zwei verschiedene Eigenschaften von einem Fingerabdruck extrahiert. Zum einen wird der Abdruck kreisförmig partitioniert, um daraus einen Orientierungsvektor zu generieren und zum anderen wird ein Vektor erstellt, der den durchschnittlichen Kantenabstand (Average-Ridge-Distance) darstellt. Einen Einblick in die sogenannte *feature-extraction* bietet das nächste Kapitel. Der Orientierungsvektor dient der Einteilung in Cluster, während der Average-Ridge-Distance-Vector (kurz ARD-V) die einzelnen Cluster noch einmal unterteilt. Diese zusätzliche Unterteilung dient noch einmal der Effizienzsteigerung, da sich mit dem ARD-V schneller vergleichen lässt, da es sich hierbei nur um einen ein-dimensionalen Vektor handelt. Der Orientierungsvektor hingegen hat in diesem Verfahren eine Größe von 156 Dimensionen. Das Verfahren lässt sich allgemein in zwei Phasen einteilen: die offline-clustering-phase und die online-query-phase. Doch zunächst soll erstmal in den nächsten beiden Abschnitten kurz auf die Gewinnung der Eigenschaften eingegangen werden.

4.1 Feature-Extraction

Es gibt zwei Arten von Merkmalen bei Fingerabdrücken: die lokalen und die globalen Merkmale. Während die globalen Merkmale die Flussstruktur der Kanten erfassen und einzelne Punkte (so genannte singular points, siehe Abb. 1a), werden bei den lokalen Eigenschaften die minuziösen Details der Kanten erfasst. Die beiden Vektoren dieses Verfahrens erfassen jeweils die globalen Merkmale und zudem korrelieren diese beiden Features sehr wenig miteinander, wodurch Redundanz durch doppelte Eigenschaftsrepräsentation vermieden wird.

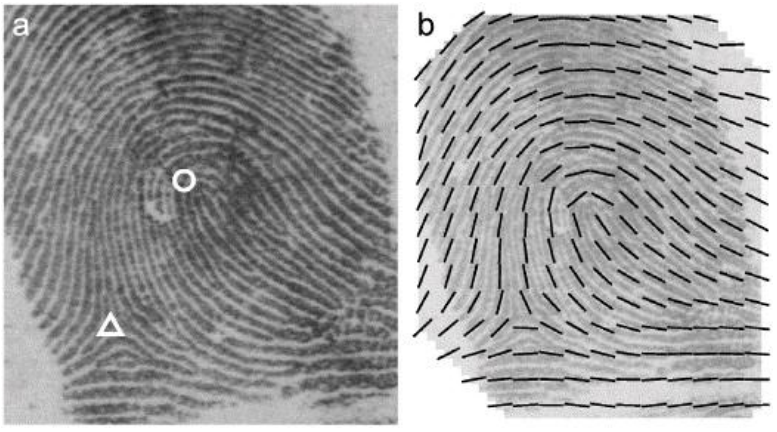

Abbildung 1: a: Core- und Delta-Points b: Orientierungsfeld

4.1.1 Orientierungsvektor

Um einen Orientierungsvektor aus einem Fingerabdruck zu extrahieren, gibt es zwei Möglichkeiten. Uniform-Spacing und Non-Uniform-Spacing. Beide Verfahren teilen das Bild in Sek-

toren ein, wobei das Erstere nur eine Einteilung in Blöcke vorsieht, während das Andere eine Kreisförmige Sektorenbildung benutzt. Das Letztere hat sich als effizienter erwiesen, da beim Non-Uniform-Spacing die Orientierungsmessungen sich mehr an den Regionen, welche die *singular points* enthalten, konzentrieren. Um nun die Orientierungen dieser Sektoren zu berechnen, werden ein Referenzpunkt und eine Referenzrichtung berechnet. Dies ist notwendig um zwei Fingerabdrücke mit verschiedenen Ausrichtungen miteinander vergleichen zu können. Der Referenzpunkt ist definiert an der Stelle mit maximaler Krümmung. Das verwendete Verfahren [2] ermittelt auf jedem Fingerabdruck einen einzigartigen Referenzpunkt. Mit Hilfe dieses Punktes wird auch die Referenzrichtung berechnet. Die unterschiedlichen Eigenschaften der einzelnen Orientierungen werden ausgewertet, indem die Inkonsistenzen der jeweiligen Orientierungswerte berechnet werden. Ein solches Orientierungs-Inkonsistenzen-Feld ist auf Abbildung 2a zu sehen.

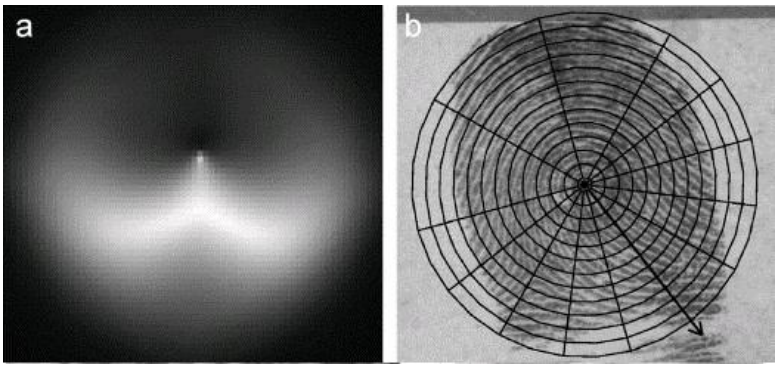

Abbildung 2: a:Orientierungs-Inkonsistenzen-Feld b:kreisförmige Segmentierung

Auf Abb. 2b sieht man die kreisförmige Segmentierung eines Abdrucks. Um nun einen Orientierungsvektor konstruieren zu können, werden die Orientierungen aller Sektoren konkateniert. Die Berechnung eines Sektors funktioniert folgendermaßen (der j-te Sektor des i-ten Rings):
$S_{i,j} = \{(x,y)|(i-1)b + b_0 \leq r < i * b + b_0, \varphi_{j-1} \leq \varphi < \varphi_j, 1 \leq x \leq X, 1 \leq y \leq Y\}$

wobei $(1 \leq i \leq E, 1 \leq j \leq F)$ und $r = \sqrt{(x-x_r)^2 + (y-y_r)^2}$
$\varphi = arctan\left(\frac{y-y_r}{x-x_r}\right) - \theta_r mod 2\pi$
(x_r, yr) ist dabei der Referenzpunkt und θ_r die Referenzrichtung
Die Parameter φ_j,b,E und F wurden empirisch festgelegt um die Performance zu erhöhen. Es sind 13 Sekroren (F=13) bei 12 Ringen (E=12), wodurch sich ein 156 dimensionaler Vektor ergibt. $\varphi_j = j * \frac{\pi}{8}$ wenn $1 \leq j \leq 5$. Wenn $6 \leq j \leq 8$ oder $9 \leq j \leq 13$ dann entspricht $\varphi_j = (2j-5)\frac{\pi}{8}$ bzw. $\varphi_j = (j+3)\frac{\pi}{8}$. Des Weiteren wird der Parameter b(der von der Auflösung abhängig ist) auf 18 Pixel gesetzt, da eine Auflösung von 500 dpi verwendet wird. Die Anzahl der Ringe(E=12) hängt von der größe der Berührungsfläche ab(in diesem Fall 512x480 Pixel). Da auf einem Fingerabdruckbild nicht nur der Abdruck an sich, sondern ebenfalls ein Hintergrund vorhanden ist, wird ein zusätzlicher Vektor bei der Berechnung benutzt: $S_q = \{s_{q,1}...s_{q,M}\}$ M ist die Anzahl der Sektoren und $s_{q,k} \in (0,1), k = 1...M$ gibt an ob sich der jeweilige Sektor im Hintergrund befindet oder sich zur Orientierungsbestimmung eignet.

4.1.2 Average-Distance-Vector (durchschnittlicher Kantenabstand)

Bei Fingerabdrücken ist der lokale Kantenabstand definiert als der Abstand der Mittelpunkte zweier benachbarter Kanten. Bei Abdrücken, die bei einer Auflösung von 500 dpi gemacht

werden, variiert die Länge solcher Distanzen zwischen drei und 25 Pixeln. Diese Eigenschaft eignet sich sehr gut, um Fingerabdrücke zu unterscheiden. Allerdings kann die ermittelte Distanz von ein und demselben Finger unterschiedlich sein, je nachdem wie stark der Finger auf den Sensor aufgedrückt wird, weshalb ein Vektor von Kantenabständen nicht zur Identifikation ausreichen kann. Dies ist aber auch Sensorabhängig und kann dementsprechend auch je nach Sensor ein immer besseres Identifikationsmerkmal werden. Alles in allem stellt der durchschnittliche Kantenabstand eine sehr gute zusätzliche Sucheigenschaft dar, die innerhalb dieses Verfahrens genutzt wird, um die Cluster nochmals zu Klassifizieren.
Es gibt verschiedene Verfahren zur Berechnung, wie z.Bsp die Spektral-Analyse, die statistische Methode und eine Kombination dieser beiden [3]. Diese Verfahren funktionieren alle in soweit, indem sie zuerst die lokalen Orientierungen berechnen und dann jeweils eine eigene Methode darauf anwenden. Die von den Autoren benutzte Variante zur Ermittlung der Average-Ridge-Distance ist ein Algorithmus zur Bildverbesserung eines Abdrucks [4]. Hierbei wird eine Region Mask generiert, von der dann der durchschnittliche Abstand extrahiert wird, wodurch dann ein ein-dimensionaler ARD-Vector entsteht.

4.2 Offline-phase: Database-Clustering

Das erste Ziel dieser Phase ist es, die vorhandenen Daten in Cluster einzuteilen und den für jeden Cluster entsprechenden Repräsentanten zu finden. Um dies zu erreichen wird eine modifizierte Version des K-Means-Algorithmus verwendet. K-Means wird sehr häufig zur Einteilung von Daten benutzt, da er eine relativ gute Performance bei wenig Speicherverbrauch liefert. Im Allgemeinen läuft dieser Algorithmus folgendermaßen ab [5]:

1. Es werden k sogenannte *group centroids* (die Repräsentanten) ausgewählt
2. Danach werden alle Daten entsprechend den *group centroids* zugeordnet
3. Wenn alle Daten verteilt sind, werden die Repräsentanten für jeden Cluster neu berechnet
4. Wenn sich die Zuordnung der Daten ändert, weiter mit Schritt 2 ansonsten Abbruch

Beim K-Means-Algorithmus wird der Parameter k beim Start festgelegt. Um die Fingerabdrucksuche zu beschleunigen und die Genauigkeit zu erhöhen, wurde dieser Parameter approximativ festgelegt. Es hat sich herausgestellt, dass bei einer Datenbank mit M Mustern sich die Anzahl der Cluster (K) am besten folgendermaßen berechnet: $K = \sqrt{M}$
Da man bei der Suche nach Fingerabdrücken eine möglichst hohe Genauigkeit haben möchte, hat man sich entschieden, mehrere Cluster nach dem ersten Suchvorgang weiter zu durchsuchen. So verhindert man, dass unter Umständen ein Fingerabdruck, der zufällig zwei Repräsentanten sehr ähnlich ist, dem falschen Cluster zugeordnet wird. Um dadurch die Performance nicht allzu sehr zu beeinflussen, ändert sich dann der Parameter entsprechend: $K = \tau\sqrt{M}$ wobei $(1 \leq \tau \leq 3)$.
Im Folgenden noch ein Beispiel: Wenn man bei der Datenbank von einer Größenordnung von 50.000.000 Mustern ausgeht, erhält man $K = 2 * \sqrt{M} \approx 14142$ Cluster, wenn M=50 Mio. und $\tau = 2$ entsprechen. Dadurch würde man in der ersten Phase des Suchens nach einem Abdruck max. 14142 Vergleiche benötigen, um mehrere ähnliche Cluster zu erhalten. Die Entwickler dieses Verfahrens geben dazu an, dass sich die durchschnittliche Anzahl an Vergleichen (in der ersten Suchphase) so berechnen lässt: $\frac{M+K}{K}$ =durchschn. Anzahl Vergleiche bis entsprechender/-en Cluster gefunden.

In dem Bsp. wären das dann: $\frac{50.000.000+14142}{14142} \approx 3537$ Vergleiche.
Wie zu Anfang erwähnt, wird eine modifizierte K-Means-Variante benutzt. Da beim K-Means-Verfahren generell die Euklidische Distanz benutzt wird, gilt es genau dort eine Änderung vorzunehmen. Denn wenn man zwei Orientierungsvektoren hat, die bis auf ihre unterschiedliche Ausrichtung sehr ähnlich sind. Ihre Distanz könnte 0 sein, aber die Euklidische Distanz würde eine große Abweichung aufweisen. Dies umgeht man, indem man die Distanz zweier Orientierungsvektoren folgendermaßen berechnet:
$d_C(\theta_q, \theta_p) = 1 - \frac{\left|\sum_{k=1}^{M} v_k e^{j2\left(\theta_{p,k}-\theta_{q,k}\right)}\right|}{\sum_{k=1}^{M} v_k}$, wobei θ_q, θ_p zwei Orientierungsvektoren darstellen. $j = \sqrt{-1}$, $v_k \in \{0,1\}$ und $v_k = 1$ wenn Element k für beide Abdrücke p,q gültig ist.
$d_C(\theta_q, \theta_p) \in [0,1]$ und $d_C(\theta_q, \theta_p) = 0$, wenn alle Differenzen der beiden Vektoren gleich sind.
Die folgende Berechnung zeigt, dass dieses Distanzmaß invariant gegenüber einer konstanten Menge von Orientierungsdifferenzen(verursacht durch eine leichte Drehung zweier ausgerichteter Fingerabdrücke) ist:
Seien a,c Skalare und I ein Vektor derselben Größe wie θ.
$\left|\sum_{k=1}^{M} v_k e^{j2\left(\theta_{p,k}+a-\theta_{q,k}-c\right)}\right| = \left|e^{j2(a-c)}\right| \left|\sum_{k=1}^{M} v_k e^{j2\left(\theta_{p,k}-\theta_{q,k}\right)}\right| = \left|\sum_{k=1}^{M} v_k e^{j2\left(\theta_{p,k}-\theta_{q,k}\right)}\right|$

$\Rightarrow d_C(\theta_q + a * I, \theta_p + c * I) = d_C(\theta_q, \theta_p)$
Genau diese Berechnung stellt eine Modifikation dar. Mit Hilfe von diesem Distanzmaß ordnet der K-Means-Algorithmus die Orientierungsvektoren in die entsprechenden Cluster ein.
In der dritten Phase des K-Means-Algo. werden die Repräsentanten neu berechnet. Dies wird normalerweise durch Berechnung des Mittelwertes gemacht. Allerdings ist dies bei den Orientierungsvektoren auf Grund der Periodizität nicht möglich, da z. Bsp. die durchschnittliche Orientierung von 0 und π gleich 0 ist und nicht wie der arithmetische Mittelwert $\frac{\pi}{2}$. Aus diesem Grund werden die Repräsentanten für jeden Cluster wie folgt berechnet:
$Z_l = \frac{1}{2} arctan \frac{\sum_{p \in C_l} S_p sin 2\theta_p}{\sum_{p \in C_l} S_p cos 2\theta_p}$, Z_l entspricht dabei dem Repräsentant von Cluster C_l. S_p entspricht dabei den in Kapitel 3.1.1 erläuterten Vektor, der angibt ob der Sektor den Vorder- bzw. Hintergrund darstellt.
Um die Effektivität bei der Fingerabdrucksuche nicht durch zu große oder zu kleine Cluster zu gefährden, wird als letzte Modifikation nochmal auf Größe untersucht und dabei werden zu kleine Cluster eliminiert und zu große werden aufgeteilt und ein entsprechender Repräsentant wird zufällig aus der Menge des neu entstandenen Cluster gewählt.
Nach diesen Modifikationen läuft der K-Means folgendermaßen ab:

1. Es werden k sogenannte *group centroids* (die Repräsentanten) ausgewählt
2. Zuordnung zu den Clustern entsprechend des eingeführten Distanzmaßes
3. Repräsentanten der Cluster Z_l berechnen
4. Die Distanzen zwischen den neuen Repräsentanten und den alten berechnen. Wenn die der maximale Repräsentant größer als ein vorher festgelegter Parameter ist, gehe zu 2.
5. Wenn keine zu kleinen oder zu großen Cluster vorhanden sind dann gebe die Cluster aus, ansonsten eliminere bzw. teile die Cluster. und gehe zu Schritt 2

Nach diesem Verfahren werden die erstellten Cluster noch jeweils in B verschiedene, vordefinierte Gruppen eingeteilt. Als Kriterium über das den entsprechenden Gruppen zugeteilt

wird ist der Average-Distance-Vector. Die Fingerabdrücke werden einer Gruppe zugeordnet wenn der ARD-wert nahe dem Mittelwert einer Gruppe ist. Der Mittelwert der k-ten Gruppe lässt sich mit $ARD_{min} + \frac{k*(ARD_{max}-ARD_{min})}{B}$ berechnen. Da im Durchschnitt bei einer Suche per ARD-V drei Gruppen zurückgegeben werden, sollte der Parameter B größer wie 20 sein.

4.3 Online-Phase: Fingerabdrucksuche (Query processing)

Die Suche nach einem Fingerabdruck gliedert sich in drei Stufen (siehe Abb. 3). In der ersten Stufe wird der entspechende Orientierungsvektor (sei er hier θ_q) mit den Repräsentanten der Cluster ($Z_l, 1 \leq l \leq K$) verglichen. Dazu wird wie beim Clustern das Distanzmaß des modifizierten K-Means-Algo.($d_C(\theta_q, Z_l)$) benutzt. Um nun festzustellen, welcher Cluster entsprechend zum Orientierungsvektor passt, wird ein Grenzwert errechnet. Für jede Distanz $d_C(\theta_q, Z_l)$ die unter diesem Wert liegt wird der entsprechende Cluster zur Ergebnismenge hinzugefügt. Der Grenzwert wurde in dem Verfahren folgendermaßen ermittelt: $min_{l=1}^{K} d_C(\theta_q, Z_l) + \sigma$.
σ ist hierbei ein zusätzlicher Optimierungsfaktor um den näheren Umkreis der Abfrage zu justieren.

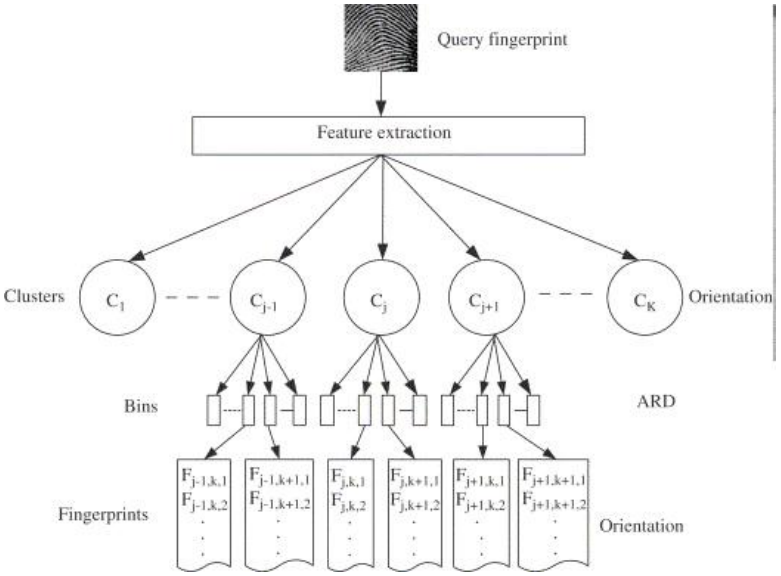

Abbildung 3:

Nachdem die erste Ergebnismenge feststeht, wird diese nun mit Hilfe des ARD-V durchsucht. Dies geschieht, indem die Distanz des ARD-V mit den Mittelwerten der einzelnen Gruppen der jeweiligen Cluster verglichen wird. Auch hier gibt es einen Grenzwert, der bei Experimenten mit diesem Verfahren bei einem Pixel lag. Nach dieser zweiten Stufe erhalten wir eine Ergebnismenge von Gruppen (die wie vorhin Abschnitt 4.2 genannt im Durchschnitt bei 3 liegt) zurückgegeben. Diese Menge wird dann in der dritten Stufe mit Hilfe des Orientierungsvektors und dem entsprechendem Distanzmaß durchsucht. Der Grenzwert wird mit $min_{l=1}^{n} d_C(\theta_q, \theta_n) + \sigma$ angegeben, wobei n die Anzahl der Fingerabdrücke in den entsprechenden Gruppen der Ergebnismenge ist und σ ein Optimierungsparameter, um die Anzahl der Fingerabdrücke aus dieser Suche zu justieren.

5 Ergebnisse

Zum Abschluss sollen hier noch einmal ein paar Ergebnisse bezüglich der Effektivität des in dieser Ausarbeitung vorgestellten Verfahrens. Als Testdatenbank wurde die NIST DB4 benutzt. Sie enthält 2000 Paare von Fingerabdrücken mit einer Auflösung von je 480x512 Pixel. Von dieser Datenbank wurden einige Fingerabdrücke entfernt, so dass eine zweite Datenbank mit 1204 Paaren von Abdrücken entsteht. Diese Daten werden als templates verwendet, während die Daten der NIST DB4 als Such-templates benutzt werden.

5.1 Ergebnisse: Feature-Extraction

In Abbildung 4 erkennt man, dass das Uniform-Spacing Verfahren, welches einen Abdruck in Blöcke einteilt (siehe Abschnitt 4.1.1), generell eine geringere Treffsicherheit hat (siehe Y-Achse) bei allen Penetrationsraten (diese Rate bezeichnet die durchschnittlichen Prozentsatz an Daten der aus der Datenbank geholt wird). Wie zu erkennen ist, reicht allein schon die Umstellung auf die Non-Uniform-Spacing Variante um eine bessere Erkennungsrate zu bekommen. Weiterhin ist der Vorteil des Average-Distance-Vector zu sehen, der nicht nur die Treffsicherheit erhöht sondern auch dafür sorgt, dass 38 Prozent weniger Orientierungsvergleiche gemacht werden.
Daraus lässt sich schließen, dass die beiden Sucheigenschaften (Orientierungsvektor und ARD-V) sehr gute Kriterien darstellen und zusätzlich auch zur Performancesteigerung beitragen.

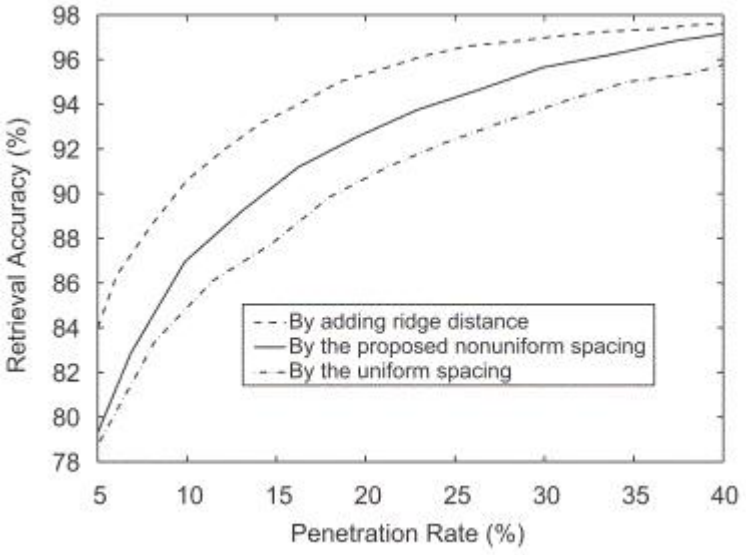

Abbildung 4:

5.2 Ergebnisse: Vergleich Clustering mit full fingerprint search

Bei diesem Ergebnis soll noch mal drauf aufmerksam gemacht werden, dass die Fingerabdrucksuche per Database-Clustering hauptsächlich der Performanz dient. Wie man an der folgenden Abbildung sieht, erhöht sich die Treffsicherheit nur minimal bei Verwendung des hier beschriebenen Verfahrens im Gegensatz zum full fingerprint search. Mit letzterem ist gemeint, wenn der Suchabdruck einfach sukzessive mit jedem Abdruck der Datenbank verglichen wird.

In Abbildung 5a sieht man, wie stark sich die Anzahl der Vergleiche ändert: Bei höchster Penetrationsrate sind es circa ein drittel weniger Vergleiche.

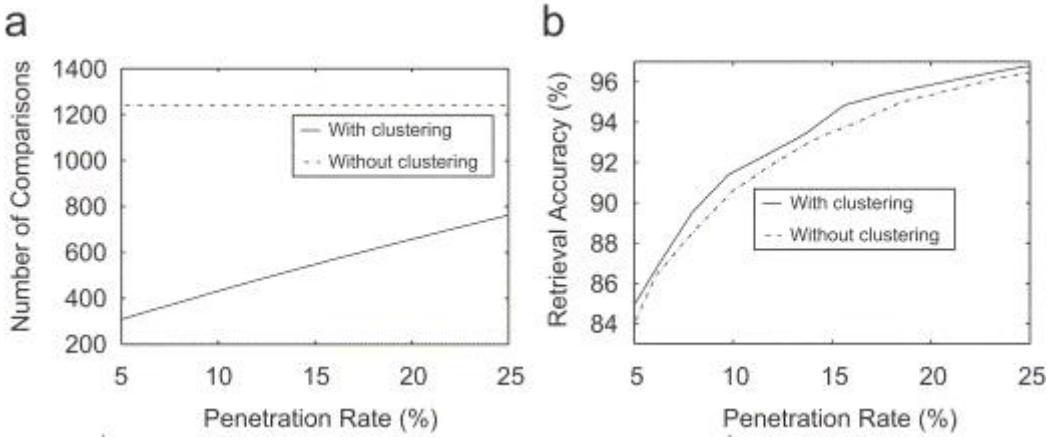

Abbildung 5:

5.3 Ergebnisse: Vergleich mit einem anderen Verfahren

Zum Schluss wird hier noch mal der Vergleich mit einem anderen Verfahren gezeigt. Zu Abbildung 6 ist zu sagen: data set 2 entspricht der reduzierten NIST DB4 Datenbank (siehe Abschnitt 5); die Referenz zum anderen Verfahren ist [6].
Das Vergleichsverfahren benutzt die kontinuierliche Klassifikation zur Einteilung der Daten. Man erkennt deutlich eine verbesserte Treffsicherheit und vor allem die signifikante Performanzsteigerung bei niedrigen Penetrationsraten.

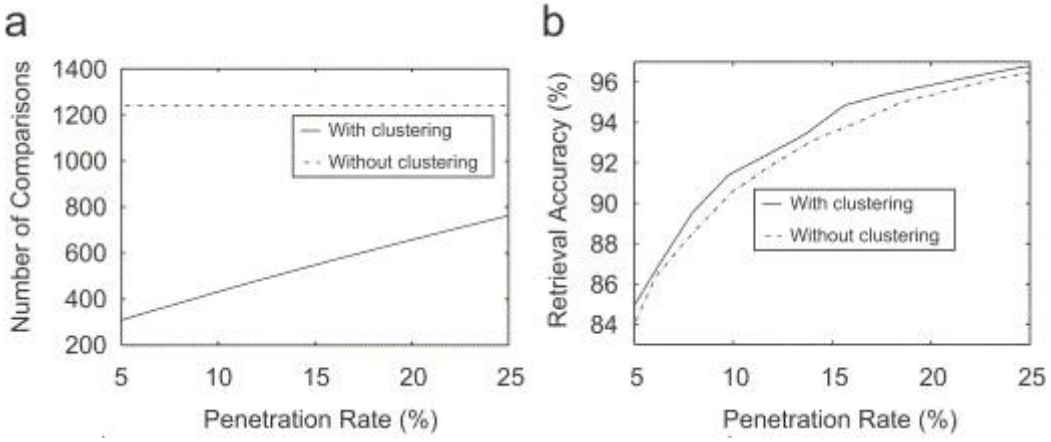

Abbildung 6:

6 Zusammenfassung

Zusammenfassend lässt sich sagen, dass das vorgestellte Verfahren sehr viele interessante Aspekte hat. Es wird mit Hilfe globaler Merkmale, die sich in zwei Vektoren zusammenfassen lassen, eine sehr beeindruckende Performanz erzielt. Gleichzeitig erhöht man gegenüber anderen Verfahren die Treffsicherheit und reduziert durch den eindimensionalen Average-Distance-Vector erheblich die Anzahl der Vergleiche mit dem größeren Orientierungsvektor. Wenn man das Beispiel aus Kapitel 4.2 noch mal betrachtet: In der ersten Stufe der Suche würde man bei 50 Mio. Einträgen in der Datenbank maximal 14142 Vergleiche benötigen, um die Cluster zu finden, in denen sich der Abdruck befinden könnte. Diese durchsucht man nun mit dem ARD-V. Wenn man nun davon ausgeht, dass sich die Abdrücke gleich über die Cluster verteilen, würde jeder Cluster 3536 Abdrücke enthalten. Bei den Experimenten zu diesem Verfahren wurde nur auf einer Datenbank mit 1204 Fingerabdrücken operiert. Da die Entwickler eine Einteilung der Cluster in mehr als 20 Gruppen empfehlen, dürfte man annehmen, dass im Falle von 50 Mio. eine Einteilung in 200 Gruppen realistisch wäre. Dadurch ergäbe sich eine Verteilung von durchschnittlich max. 18 Fingerabdrücken pro Gruppe. Da durchschnittlich 3 Gruppen als Ergebnismenge zurückgegeben werden, würde man in diesem Beispiel noch max. $3 * 18 = 54$ Vergleiche mit Hilfe des Orientierungsvektors benötigen (in Stufe drei).
Für Stufe zwei ergeben sich folgende Werte: da wir mehrere Cluster nach der ersten Stufe zurückbekommen müssen diese natürlich alle durchsucht werden. Da in diesem Beispiel davon ausgegangen wird, dass ein Cluster 200 Gruppen enthält und der ARD-V des Suchabdrucks nur mit Mittelwerten der Gruppen verglichen wird, beläuft sich die Anzahl der max. zusätzlichen Vergleiche auf: (Anzahl Cluster aus Stufe eins der Suche) * 200.
Wenn man nun diese Werte betrachtet, erkennt man, wie durch Database-Clustering signifikant die Anzahl der Vergleiche verringert werden kann. Aus diesem Grund denke ich, dass dies ein sehr praktikables Verfahren ist um auf sehr großen Datenbanken zu suchen.

Literatur

[1] M.Liu, X. Jiang, A. Chichung Kot, Efficient fingerprint search on database clustering, *Pattern Recognition*

[2] M. Liu, X.D. Jiang and A.C. Kot, Fingerprint reference point detection, EURASIP J. Appl. Signal Process. 2005 (2005) (4), pp. 498-509

[3] Y. Yin, J. Tian and X. Yang, Ridge Distance Estimation in Fingerprint Images:Algorithm and Performance Evaluation,EURASIP Journal on Applied Signal Processing Volume 2004 (2004), Issue 4, Pages 495-502

[4] L. Hong, Y. Wan and A.K. Jain, Fingerprint image enhancement: algorithm and performance evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) (8), pp. 777-789.

[5] Quelle: $http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/kmeans.html$, Abrufdatum 29.05.2007

[6] R. Cappelli, A. Lumini, D. Maio and D. Maltoni, Fingerprint classification by directional image partitioning, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999)(5), pp.402-421