



Skript zur Vorlesung  
**Datenbanksysteme II**  
Sommersemester 2005

# Kapitel 6: Ähnlichkeitsmodelle für Multimediatdaten

Vorlesung: Christian Böhm  
Übungen: Elke Achtert, Peter Kunath

Skript © 2005 Christian Böhm

<http://www.dbs.informatik.uni-muenchen.de/Lehre/DBSII>



## Inhalt

1. Sequenzen
2. Bilder
3. Polygone, 3D



# Inhalt

## 1. Sequenzen

## 2. Bilder

## 3. Polygone, 3D



# Ähnlichkeitsmodelle für Sequenzdaten

- Eine Sequenz der Länge  $n$  ist eine Abbildung der Indexmenge  $I_n = \{1, \dots, n\}$  in einen Wertebereich  $W: I_n \rightarrow W$
- Sequenzen lassen sich anhand ihres Wertebereichs klassifizieren:
  - nominale Werte (Kategorien, Alphabete, allgemein: Aufzählungstypen)  
Beispiele:  
Texte:  $I_n \rightarrow$  Buchstaben  
DNA-Sequenzen:  $I_n \rightarrow$  Nucleinsäuren  $\{C, G, A, T\}$   
Proteinsequenzen:  $I_n \rightarrow$  Aminosäuren  $\{LEU, ARG, \dots\}$
  - kontinuierliche Werte (reelle Zahlen)  
Beispiele (allg. Zeitreihen):  
Aktienkurse:  $I_n \rightarrow$  Kurswerte  
Fieberkurven:  $I_n \rightarrow$  Temperaturwerte



# Ähnlichkeitsmodelle für Sequenzdaten

Datenbanksysteme II  
Kapitel 6: Ähnlichkeitsmodelle für Multimediadaten

5

- Klassen von Anfragen
  - Vollsequenzsuche: Sequenzen sind über ihre gesamte Länge ähnlich
  - Teilsequenzsuche: Suche nach Vorkommen von kurzen Anfragesequenzen in (längeren) Datenbanksequenzen



# Ähnlichkeitsmodelle für Zeitreihen fester Länge

Datenbanksysteme II  
Kapitel 6: Ähnlichkeitsmodelle für Multimediadaten

6

- Anfragebeispiele
  - “Identifiziere Unternehmen mit einem ähnlichen Umsatzverlauf.”
  - “Bestimme Produkte mit einer ähnlichen Entwicklung der Verkaufszahlen.”
  - “Ermittle Aktien mit einem ähnlichen Kursverlauf.”
  - “Stelle fest, ob ein Musikstück zu einem der geschützten Werke ähnlich ist.”
- Charakterisierung des Verfahrens
  - Beschränkung auf Vollsequenzsuche (Teilsequenzen werden nicht betrachtet).
  - Sequenzen  $I_n \rightarrow IR$  werden als n-dimensionale Vektoren aus  $IR^n$  betrachtet.



# Diskrete Fourier-Transformation (1)

## Diskrete Fourier-Transformation (DFT)

- Gegeben sei ein Signal  $x = [x_t], t = 0, \dots, n - 1$
- Die DFT von  $x$  ist eine Sequenz  $X = [X_f]$  von  $n$  komplexen Zahlen,  $f = 0, \dots, n - 1$  mit

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \exp(-i2\pi ft / n) \quad f = 0, \dots, n - 1 \quad (f: \text{Frequenzen})$$

wobei  $i$  die komplexe Einheit bezeichnet, d.h.  $i^2 = -1$ .

- Durch die inverse DFT wird das ursprüngliche Signal  $x$  wiederhergestellt:

$$x_t = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} X_f \exp(i2\pi ft / n) \quad t = 0, \dots, n - 1 \quad (t: \text{Zeitpunkte})$$

- $[x_t] \leftrightarrow [X_f]$  bezeichne ein Fourier-Paar, d.h.  $\text{DFT}([x_t]) = [X_f]$  und  $\text{DFT}^{-1}([X_f]) = [x_t]$ .



# Diskrete Fourier-Transformation (2)

- Die DFT ist eine *lineare Abbildung*, d.h. mit  $[x_t] \leftrightarrow [X_f]$  und  $[y_t] \leftrightarrow [Y_f]$  gilt auch:
  - (i)  $[x_t + y_t] \leftrightarrow [X_f + Y_f]$  und
  - (ii)  $[ax_t] \leftrightarrow [aX_f]$  für ein Skalar  $a \in \mathbb{R}$
- Verschiebungsinvarianz
  - $A = |c|$  heißt die Amplitude und  $\varphi$  die Phase einer komplexen Zahl  $c = a + ib = A \cdot e^{i\varphi}$ .
  - Eine Verschiebung im Zeitbereich verschiebt nur die Phase, nicht jedoch die Amplitude der Fourierkoeffizienten:  
 $[x_{t-t_0}] \leftrightarrow [X_f \exp(i2\pi ft_0/n)]$



## Satz von Parseval (1)

### Energie einer Sequenz

- Die *Energie*  $E(c)$  von  $c$  ist das Quadrat der Amplitude:  $E(c) = |c|^2$ .
- Die *Energie*  $E(x)$  einer Sequenz  $x$  ist die Summe aller Energien über die Sequenz:

$$E(x) = \|x\|^2 = \sum_{t=0}^{n-1} |x_t|^2$$

### Satz von Parseval

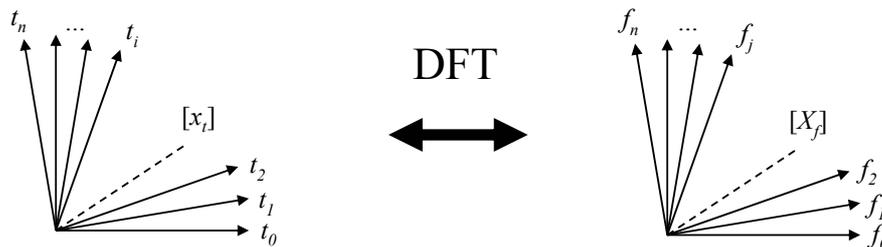
- Die Energie eines Signals im Zeitbereich ist gleich der Energie im Frequenzbereich.

Formal: Sei  $X$  die DFT von  $x$ , dann gilt:

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2$$



## Satz von Parseval (2)



- Mit anderen Worten:  
Die euklidische Distanz zweier Signale  $x$  und  $y$  stimmt im Zeit- und im Frequenzbereich überein:

$$\|x - y\|^2 = \|X - Y\|^2$$



## Satz von Parseval (3)

### Grundidee der Technik

- Als Ähnlichkeitsfunktion für Sequenzen wird die euklidische Distanz verwendet:

$$D(x, y) = \|x - y\| = \sqrt{\sum_{t=0}^{n-1} |x_t - y_t|^2}$$

- Der Satz von Parseval ermöglicht nun, die Distanzen im Frequenzbereich statt im Zeitbereich zu berechnen:  
 $D(x, y) = D(X, Y)$



## Kürzen der Sequenzen

### Kürzen der Sequenzen für die Indexierung

- In praktischen Beispielen haben die tiefsten Frequenzen die größte Bedeutung.
- Die ersten Frequenz-Koeffizienten enthalten also die wichtigste Information.
- Für den Aufbau eines Index werden die transformierten Sequenzen gekürzt, d.h. von  $[X_f], f = 0, 1, \dots, n - 1$  werden nur die ersten  $c$  Koeffizienten  $[X_{f < c}]$ ,  $c < n$ , indexiert.
- Im Index kann dann eine untere Schranke der echten Distanz berechnet werden:

$$D_c(x, y) = \sqrt{\sum_{f=0}^{c-1} |x_f - y_f|^2} \leq \sqrt{\sum_{f=0}^{n-1} |x_f - y_f|^2} = D(x, y)$$

- Diese Eigenschaft ist wichtig für die Suche, weil sie die Vollständigkeit der Ergebnisse des Index garantiert, d.h. die Ergebnisse aus dem Index bilden eine Obermenge der tatsächlichen Ergebnisse.



## Anfragebearbeitung (1)

Es ergibt sich also folgender Anfrageablauf:

- Ausgegeben werden sollen alle Objekte  $o$  in der Datenbank, die sich höchstens um  $\varepsilon$  vom Anfrageobjekt  $q$  ("query") unterscheiden:  
 $\{o \in DB \mid D(o, q) \leq \varepsilon\}$

- *Vollständigkeit* der Anfragebearbeitung  
Ein Filterschritt basierend auf einem Index mit  $c < n$  Koeffizienten kann nun eine Obermenge der Ergebnisse finden (Kandidaten), d.h. der Filterschritt ist *vollständig*:

$$\{o \in DB \mid D(o, q) \leq \varepsilon\} \subseteq \{o \in DB \mid D_c(o, q) \leq \varepsilon\}$$

(tatsächliche Ergebnisse)      (Kandidaten aus Index)

Beweis: Wegen  $D_c(o, q) \leq D(o, q)$  gilt für jedes  $o$  mit  $D(o, q) \leq \varepsilon$  auch  $D_c(o, q) \leq \varepsilon$ .

Es gibt also keinen Treffer  $o'$  mit  $D(o', q) \leq \varepsilon < D_c(o', q)$ , der im Index verloren geht.



## Anfragebearbeitung (2)

- *Korrektheit* der Anfragebearbeitung  
Ein nachgeschalteter Verfeinerungsschritt berechnet die exakten Distanzwerte  $D(o, q)$  auf den vollständigen Sequenzen und gewährleistet so, daß die ausgegebenen Resultate tatsächlich das Ähnlichkeitskriterium  $D(o, q) \leq \varepsilon$  erfüllen.



# Zusammenfassung

## Zusammenfassung

- Zeitreihen werden mit Hilfe der Diskreten Fourier-Transformation (DFT) vom Zeitbereich in den Frequenzbereich abgebildet.
- Satz von Parseval: Die euklidische Distanz ist invariant gegenüber der Fourier-Transformation, d.h. sie hat den gleichen Wert im Zeit- wie im Frequenzbereich.
- Beobachtung: nur die tiefsten Frequenzen haben eine praktische Bedeutung. Experimente zeigen, dass die ersten 1 bis 3 Koeffizienten genügen.
- Insgesamt: Abbildung von hochdimensionalen Zeitreihen in niedrigerdimensionale Sequenzen von Frequenzwerten.
- Mehrstufige Anfragebearbeitung mit  $R^*$ -Baum im Filterschritt.

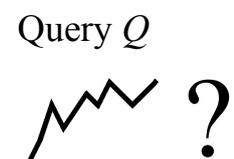
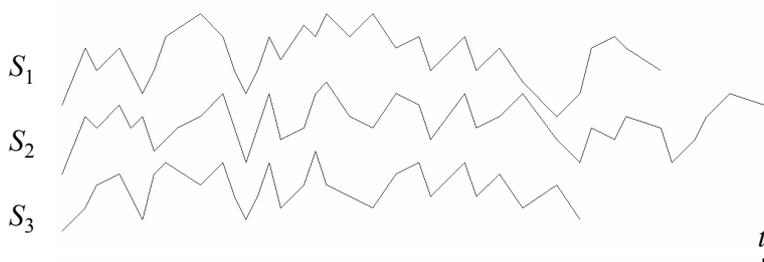


# Suche nach Teilsequenzen (1)

## Modell für die Suche nach Teilsequenzen

Hier: Suche nach Teilsequenzen (nicht mehr nur vollständige Sequenzen)

- Gegeben seien  $N$  Sequenzen  $S_1, S_2, \dots, S_N$  beliebiger Längen, eine Anfragesequenz  $Q$ , eine Ähnlichkeitstoleranz  $\varepsilon$  sowie eine Distanzfunktion  $D$  für Sequenzen.
- Gesucht sind diejenigen  $S_i$ , die Teilsequenzen  $s = S_i[k, k+\text{len}(Q)-1]$  beinhalten, deren Abstand  $D(s, Q)$  höchstens  $\varepsilon$  beträgt. Zu jedem  $S_i$  soll auch die Position  $k$  der entsprechenden Teilsequenz  $s$  ausgegeben werden.





## Suche nach Teilsequenzen (2)

### Anwendungsbeispiele

- Finanz- und betriebswirtschaftliche Datenbanken: “Finde Beispiele aus der Vergangenheit, bei denen sich die Verkaufsentwicklung ähnlich entwickelt hat wie bei unserem Produkt in den vergangenen drei Monaten.”
- Technisch-wissenschaftliche Datenbanken: “Wann konnte ein ähnliches Verhalten des Sonnenwindes gemessen werden wie heute vormittag von 10.00 bis 12.00 Uhr?”

### Mindestlänge $w$ für Anfragesequenzen

- Im weiteren wird eine Mindestlänge  $w$  für Anfragesequenzen angenommen.
- Beispiel: Bei Aktienkursen ist man an wöchentlichen oder monatlichen Mustern der Kursverläufe interessiert, da sie weniger rauschanfällig sind.
- Kürzere Anfragen werden nach wie vor unterstützt (durch sequentielle Suche).

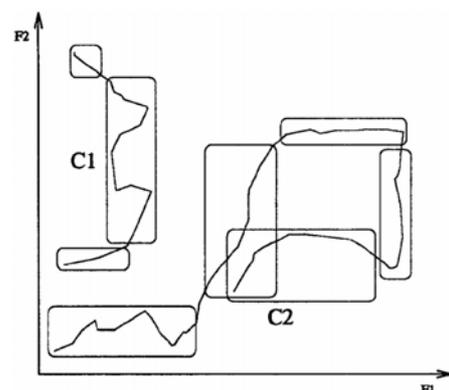


## Suche nach Teilsequenzen (3)

### Abbildung der Zeitreihen

- Über jede der Sequenzen wird ein Fenster der Länge  $w$  geschoben.
- An jeder Fensterposition wird die sichtbare Teilsequenz (wie oben) durch DFT codiert und stellt dadurch einen Punkt im  $w$ -dimensionalen Frequenzraum dar.
- Eine Sequenz  $S$  wird also durch eine Folge von  $\text{len}(S) - w + 1$  vielen Punkten der Dimension  $w$  repräsentiert, d.h. ein (Feature-)Punkt für jede Fensterposition.

Beispiel (Quelle: [FRM 84]):  
Zwei Sequenzen  $S_1$  und  $S_2$  mit den DFT-Folgen  $C_1$  und  $C_2$  (hier im 2D)





## Suche nach Teilsequenzen (4)

### Suche über den Featurepunkten

- Sequentieller Scan: Durchlaufe die Menge aller Punkte in der Datenbank
- Verwendung mehrdimensionaler Indexstrukturen (z.B. R\*-Baum, X-Baum, ...)

### Index für die einzelnen Punkte

- Bereichsanfrage:  $\epsilon$ -Bereich um Anfragepunkt im Index anfragen, dann verfeinern.
- Dieser Suchalgorithmus ist vollständig (Beweis wie vorher).
- Experimente zeigen, dass diese Methode etwa doppelt so langsam wie eine sequentielle Suche ist, eine bessere Lösung ist also dringend erwünscht.



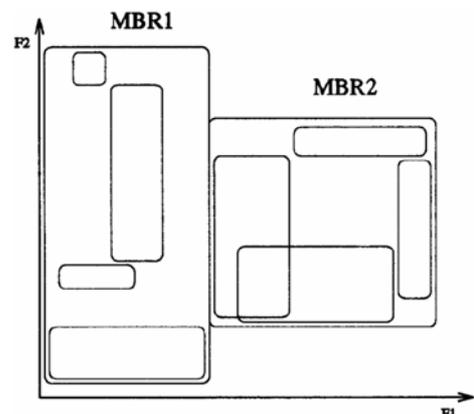
## Suche nach Teilsequenzen (5)

### Zusammenfassen von Punkten zu Bereichen.

- Beobachtung: Aufeinanderfolgende Punkte liegen oft nahe beieinander, da die Inhalte zweier stark überlappender Fenster sehr ähnlich sind.
- Idee: Aufspaltung der Punktesfolgen in Teilfolgen, dann Repräsentation der Teilfolgen durch ihre minimal umgebenden (Hyper-)Rechtecke.

Im Beispiel (Quelle: [FRM 84]):

- (i) Speicherung weniger Rechtecke statt vieler Punkte und
- (ii) (hierarchische) Zusammenfassung der Rechtecke im Index (MBR = minimum bounding rectangle)

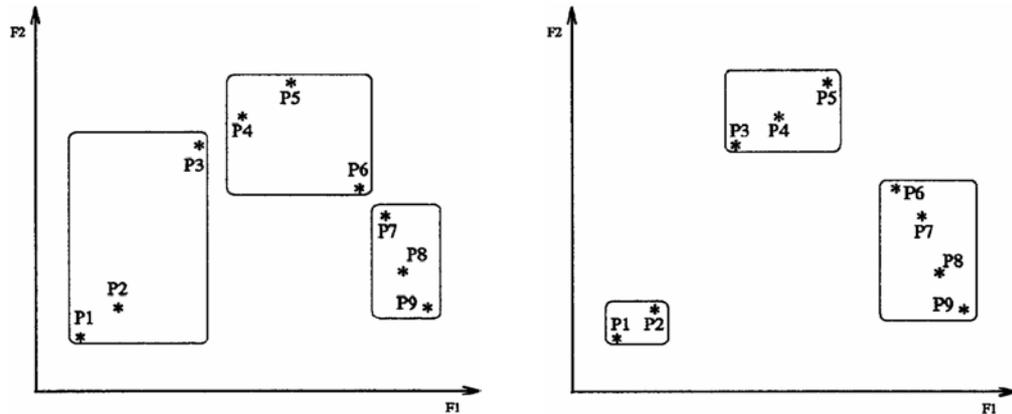




## Suche nach Teilsequenzen (6)

### Einfügen neuer Daten

- wesentliche Aufgabe: Unterteilung von Punktfolgen  $C$  in "gute" Teilfolgen.
- Punktezahl kann fix sein (z.B. 50) oder von Gesamtlänge abhängen (z.B.  $\sqrt{\text{len}(C)}$ )
- Aufteilung mit fester Punktanzahl pro Teilfolge kann schlecht sein (Quelle: [FRM 84]):



## Suche nach Teilsequenzen (7)

- Idee: Heuristik zur Aufteilung bei Minimierung der erwarteten Plattenzugriffe
- Algorithmus zur Zerlegung einer Punktfolge in Teilfolgen:

Ordne den ersten Punkt einer (trivialen) Teilfolge zu.  
**Für jeden** nachfolgenden Punkt  $p$ :  
**Falls**  $p$  die Grenzkosten der aktuellen Teilfolge erhöht,  
**dann** starte eine neue Teilfolge mit  $p$ ,  
**sonst** füge  $p$  in die aktuelle Teilfolge ein.

- Grenzkosten für  $k$  Punkte in einer entsprechenden Teilfolge  $L$ :  
 $\text{Zugriffe}(L) / k$
- *Erwartete Zugriffe für ein  $n$ -dim. Rechteck  $L$  aus  $[0, 1]^n$  mit den Seitenlängen  $L_1, L_2, \dots, L_n$ :*

$$\text{Zugriffe}(L) = \prod_{i=1}^n (L_i + 0,5)$$



## Suche nach Teilsequenzen (8)

### Anfragebearbeitung mit Anfragesequenzen der Minimallänge $w$

- Anfrage: “Suche Teilsequenzen, die zur Sequenz  $q$  höchstens den Abstand  $\varepsilon$  haben.”
- Algorithmus:
  1. Bilde die Anfragesequenz  $q$  auf den Punkt  $q_f$  im Featureraum ab.
  2. Identifiziere mit Hilfe des Index diejenigen Teilfolgen, deren minimal umgebende Rechtecke die Kugel um  $q_f$  mit dem Radius  $\varepsilon$  schneiden.
  3. Untersuche die zugehörigen Teilsequenzen und verwirf falsche Antworten.
- Die *Korrektheit* der Antworten wird im Schritt 3 sichergestellt (“Verfeinerung”).
- Die *Vollständigkeit* der Antworten wird dadurch garantiert, dass im Schritt 2 (“Filterschritt”) umgebende Rechtecke verwendet werden; es können also keine Resultate verloren gehen.



## Suche nach Teilsequenzen (9)

### Anfragebearbeitung mit längeren Anfragesequenzen $q$ , d.h. $\text{len}(q) > w$

- Problem: Index kennt nur Sequenzen der Länge  $w$
- *Idee Präfixsuche*: Suche mit einer Teilsequenz von  $q$  der Länge  $w$ , z.B. dem Präfix.
- Die Präfixsuche stellt die Vollständigkeit sicher, da auf jeden Fall eine Obermenge der tatsächlichen Resultate ermittelt wird:

Lemma: Falls zwei Sequenzen  $s$  und  $q$  derselben Länge  $l$  sich (bezüglich des euklidischen Abstands  $D$ ) um nicht mehr als  $\varepsilon$  unterscheiden, dann stimmen auch jeweils zwei entsprechende Teilsequenzen  $s[i:j]$  und  $q[i:j]$  im selben Rahmen  $\varepsilon$  überein:

$$D(s, q) \leq \varepsilon \Rightarrow D(s[i:j], q[i:j]) \leq \varepsilon \quad \text{für } 1 \leq i \leq j \leq l$$

Beweis: Die Behauptung folgt aus folgender Eigenschaft:

$$D(s[i:j], q[i:j]) = \sqrt{\sum_{k=i}^j |s_k - q_k|^2} \leq \sqrt{\sum_{k=1}^l |s_k - q_k|^2} = D(s, q)$$



## Suche nach Teilsequenzen (10)

**Anfragebearbeitung für sehr lange Anfragesequenzen  $q$ ,  
d.h.  $\text{len}(q) \gg w$**

- Problem: Volumen der Anfragekugel im Featureraum ist sehr groß
- Idee: Zerlege die Anfragesequenz in mehrere Teilsequenzen der Länge  $w$
- *Annahme: Die Länge der Anfragesequenz  $q$  ist ein Vielfaches von  $w$ :  $\text{len}(q) = p \cdot w$  (andernfalls kann die obige Präfixlösung angewandt werden).*
- Algorithmus *Zerlegungssuche*:

1. Zerlege die Anfragesequenz  $q$  in  $p$  Teilsequenzen der Länge  $w$ , die  $p$  Kugeln mit Radius  $\varepsilon/\sqrt{p}$  im Featureraum entsprechen.
2. Hole mit Hilfe des Index alle Teilfolgen aus der Datenbank, deren umgebendes Rechteck eine der  $p$  Anfragekugeln schneidet.
3. Untersuche die zugehörigen Teilsequenzen, um falsche Resultate zu verwerfen.



## Suche nach Teilsequenzen (11)

**Vollständigkeit des Algorithmus**

- Seien zwei Sequenzen  $s$  und  $q$  gleich lang, d.h.  $\text{len}(s) = \text{len}(q) = p \cdot w$ .
- Für das folgende Lemma betrachten wir die  $p$  disjunkten Teilsequenzen

$$s_i = s[i \cdot w + 1 : (i+1) \cdot w] \text{ und } q_i = q[i \cdot w + 1 : (i+1) \cdot w] \text{ für } i = 0, \dots, p-1.$$

Lemma: Falls zwei Sequenzen  $s$  und  $q$  derselben Länge  $l$  sich höchstens um  $\varepsilon$  unterscheiden, dann gibt es mindestens ein Paar  $s_i$  und  $q_i$  von sich entsprechenden Teilsequenzen mit einem Abstand kleiner oder gleich  $\varepsilon / \sqrt{p}$  :

$$D(s, q) \leq \varepsilon \Rightarrow \exists i = 0, \dots, p-1 : D(s_i, q_i) \leq \varepsilon / \sqrt{p}$$



## Suche nach Teilsequenzen (12)

Beweis (durch Widerspruch):

Sei  $D(s, q) \leq \varepsilon$ . Falls alle  $p$  Teilsequenzen einen Abstand größer als  $\varepsilon / \sqrt{p}$  hätten, wäre der Gesamtabstand größer als  $\varepsilon$ :

$$\text{Aus } D(s_i, q_i) = \sqrt{\sum_{j=i \cdot w + 1}^{(i+1) \cdot w} (s_i[j] - q_i[j])^2} > \varepsilon / \sqrt{p} \quad \text{für alle } i = 0, \dots, p-1$$

folgt  $D^2(s_i, q_i) = \sum_{j=i \cdot w + 1}^{(i+1) \cdot w} (s_i[j] - q_i[j])^2 > \varepsilon^2 / p$ , und damit

$$D^2(s, q) = \sum_{j=1}^{p \cdot w} (s[j] - q[j])^2 > p \cdot \varepsilon^2 / p = \varepsilon^2, \text{ also } D(s, q) > \varepsilon$$



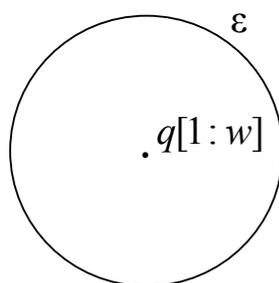
## Suche nach Teilsequenzen (13)

**Vergleich von Präfixsuche und Zerlegungssuche:**

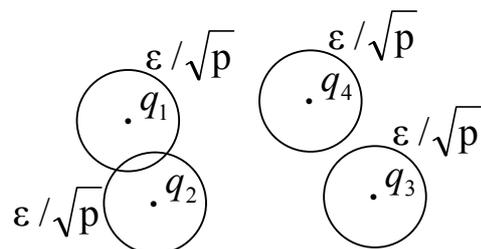
Betrachte das Volumen  $V$  der Anfragekugeln im  $r$ -dimensionalen Featureraum:

- *Präfixsuche*:  $V(\varepsilon) = K \cdot \varepsilon^r$   
( $K$  bezeichne das Volumen der  $r$ -dim. Einheitskugel).
- *Zerlegungssuche*:  $p$  Kugeln á  $K \cdot (\varepsilon / \sqrt{p})^r$ ,  
also  $V(\varepsilon) = K p \varepsilon^r / \sqrt{p}^r = K \varepsilon^r p^{1-r/2}$

Ergebnis: Zerlegungssuche ist effizienter, falls  $p^{1-r/2} < 1$ , d.h. falls  $r > 2$ .



Präfixsuche



Zerlegungssuche



# Inhalt

1. Sequenzen

2. Bilder

3. CAD/3D



# Ähnlichkeitsmodelle für Bilder

## Inhaltsbezogene Suche in Bilddatenbanken

- Suche über Standardattribute
  - Primärschlüssel (z.B. Dateiname): Keine “Suche”, da Identifikator bereits bekannt.
  - Sekundäre Merkmale (Kontextinformationen) wie Entstehungsdatum, Entstehungsort, Rechteinhaber sind nur begrenzt hilfreich.



## Inhaltsbezogene Suche

- Inhaltsbasierte Suche über Schlüsselwörter
  - Manuelle Verschlagwortung bedeutet großen Aufwand.
  - Schlagwörter müssen normiert sein (Abhilfe durch Dictionaries möglich).
  - Schlagwörter decken immer nur bestimmte ausgewählte Aspekte ab (z.B. abgebildete Gegenstände, Indoor/Outdoor/...-Klassifikation)
  - Schlagwortsuche versagt, wenn betrachteter Aspekt nicht als Schlagwort aufgenommen wurde (z.B. "Suche alle Bilder mit hohem Grünanteil am unteren Rand").
- Suche über den eigentlichen Bildinhalt
  - Konzept: Inhalt aus der internen Bildrepräsentation (Pixel) ableiten.
  - Aufwand und Probleme der manuellen Verschlagwortung entfallen.
  - Möglichkeiten: Farben, Texturen, Formen



## Merkmale von Bildern

### Merkmale von Bildern

- Farbe
  - Farbhistogramme (QBIC) [HSE+ 95]
- Textur
  - Beschaffenheit von Bildsegmenten (z.B. Holzmaserung, Kieselsteine, Karomuster)
  - Evaluierung verschiedener Distanzfunktionen [PBRT 99]
- Formen (Konturen)
  - Algebraische Moment-Invarianten [TC 91] [FBF+ 94]
  - Pixelbasierte Ähnlichkeitsmodelle [WJ 96] [AKS 98]
  - Morphologisches Ähnlichkeitsmodell [KSF+ 98]



## Systeme zur Inhaltsbasierten Suche

### Systeme zur Inhaltsbasierten Suche

- *QBIC*: Query By Image (and Video) Content. IBM Almaden Research Center
- *ImageMiner*. Technologie-Zentrum Informatik, Uni Bremen
- *VisualSeek*. Center for Telecom Research, Columbia Univ., NY
- *MARS: Multimedia Analysis and Retrieval System*. U. Illinois/Urbana-Champaign
- *Surfimage*. INRIA Recquencourt, France
- ... und viele mehr!



## Farbhistogramme

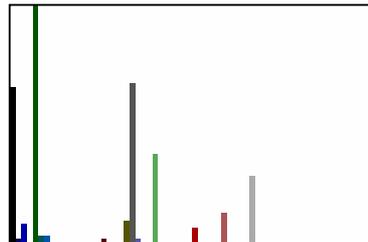
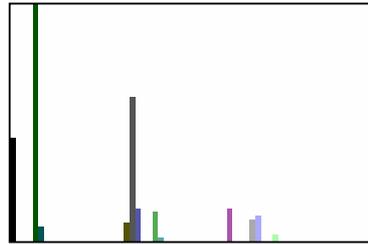
### Farbhistogramme

- Repräsentation der Farbverteilung in einem Bild (auf Pixelbasis)
- Definition der Farbhistogramme
  - Farbraum festlegen (z.B. RGB, HSV, HLS, Munsell, ...)
  - Menge von Repräsentanten im Farbraum auswählen (sample points)
  - z.B. Gitter im Farbraum mit  $4 \times 4 \times 4 = 64$  Farben oder  $8 \times 8 \times 8 = 512$  Farben
- Berechnung der Farbhistogramme
  - Für jedes Pixel, erhöhe den Zähler des nächstgelegenen Repräsentanten um eins.
  - Evtl. Normierung, um Histogramm von der Bildgröße unabhängig zu machen.



# Farbhistogramme

Beispiel für Farbhistogramme (64 Repräsentanten):



# Distanzfunktionen

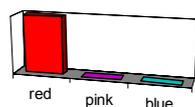
## Distanzfunktionen

- Beispiel euklidische Distanz:  
Seien  $H^P$  und  $H^Q$  die Farbhistogramme der Bilder  $P$  und  $Q$ .

$$D(P, Q) = \sqrt{(H^P - H^Q) \cdot (H^P - H^Q)^T}$$



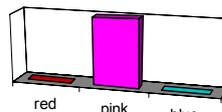
'RED'



(1, 0, 0, ...)



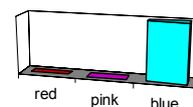
'PINK'



(0, 1, 0, ...)



'BLUE'



(0, 0, 1, ...)

- Es gilt:  
 $D(\text{'RED'}, \text{'PINK'}) = \sqrt{2}$ ,  $D(\text{'RED'}, \text{'BLUE'}) = \sqrt{2}$ ,  
 $D(\text{'PINK'}, \text{'BLUE'}) = \sqrt{2}$



# Quadratische Formen

## Quadratische Formen als Distanzfunktionen

- Definition: Sei  $A$  eine Ähnlichkeitsmatrix, dann gilt:

$$D_A(P, Q) = \sqrt{(H^P - H^Q) \cdot A \cdot (H^P - H^Q)^T} = \sqrt{\sum_i \sum_j a_{ij} (H_i^P - H_i^Q)(H_j^P - H_j^Q)}$$

- Die Einträge  $a_{ij}$  einer Ähnlichkeitsmatrix  $A = [a_{ij}]$  beschreiben die Ähnlichkeit der Dimensionen  $i$  und  $j$  in den Vektoren (Bins  $i$  und  $j$  in den Histogrammen)

$$A = \begin{bmatrix} 1 & & & \\ & \dots & & \\ & & a_{ij} & \dots \\ & & & & 1 \end{bmatrix}$$

- Im obigen Beispiel erhalten wir für die Matrix  $A' = \begin{bmatrix} 1,0 & 0,9 & 0,0 \\ 0,9 & 1,0 & 0,0 \\ 0,0 & 0,0 & 1,0 \end{bmatrix}$  die Abstandswerte:

$$D(\text{'RED'}, \text{'PINK'}) = \sqrt{0,2}, \quad D(\text{'RED'}, \text{'BLUE'}) = \sqrt{2}, \\ D(\text{'PINK'}, \text{'BLUE'}) = \sqrt{2}$$



# Ähnlichkeitsmatrizen (1)

## Beispiele für Ähnlichkeitsmatrizen (vgl. [HSE+ 95])

(im folgenden ist  $d_{ij}$  der Abstand der Bins  $i$  und  $j$ )

- $a_{ij} = (1 - d_{ij} / d_{max})$
- $a_{ij} = \exp(-\sigma (d_{ij} / d_{max})^2)$   
(für  $\sigma \rightarrow \infty$  erhält man die Einheitsmatrix)
- QBIC verwendet eine aus Ergebnissen der Perzeptionsforschung abgeleitete Matrix



## Ähnlichkeitsmatrizen (2)

### Eigenschaften von Ähnlichkeitsmatrizen

- Symmetrie

Wir dürfen annehmen, dass Ähnlichkeitsmatrizen immer symmetrisch sind, denn:

Lemma: zu jeder Matrix  $A'$  gibt es eine symmetrische Matrix

$$A = (A' + A'^T)/2, \text{ so dass gilt: } D_A(P, Q) = D_{A'}(P, Q)$$

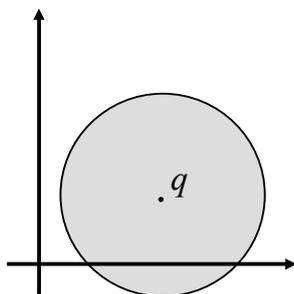
Beweis: Sei  $\Delta = H^P - H^Q$ , dann gilt:

$$\begin{aligned} D_A(P, Q) &= \sqrt{\Delta \cdot A \cdot \Delta^T} = \sqrt{\Delta \cdot \frac{A' + A'^T}{2} \cdot \Delta^T} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n \left( \frac{a_{ij} + a_{ji}}{2} \right) \Delta_i \Delta_j} \\ &= \sqrt{\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n a_{ij} \Delta_i \Delta_j + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n a_{ji} \Delta_i \Delta_j} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij} \Delta_i \Delta_j} = D_{A'}(P, Q) \end{aligned}$$

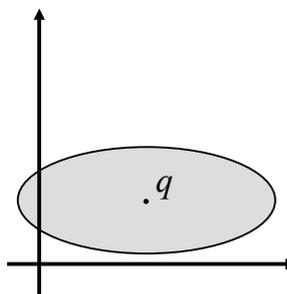


## Ähnlichkeitsmatrizen (3)

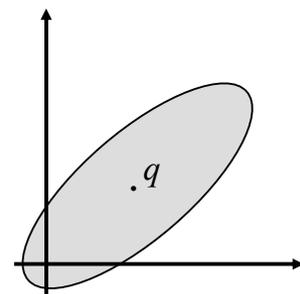
### Geometrie von $\epsilon$ -Anfragen



euklidische Distanz



gewichtete  
euklidische Distanz



positiv-definite  
quadratische Form



## Ähnlichkeitsmatrizen (4)

### Positiv definite Matrizen (PD)

- Ähnlichkeitsmatrizen müssen positiv definit sein
- Definition (aus der linearen Algebra):  
 $A$  ist positiv definit gdw.  $x A x^T > 0$  für alle  $x \neq 0$
- D.h. Matrix ist PD gdw. Distanzfunktion ist PD (nötig für Metrik!)
- Test, ob eine Matrix PD ist z.B. durch Berechnung der Cholesky-Zerlegung

### Positiv semidefinite Matrizen (PSD)

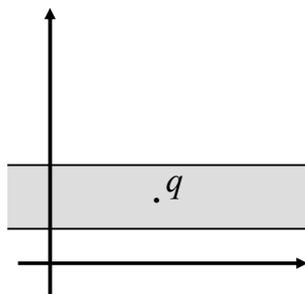
- Für bestimmte Anwendungen kann man sich semi-definite Matrizen vorstellen
- Definition:  $A$  ist positiv semi-definit gdw.  $x A x^T \geq 0$  für alle  $x \neq 0$
- D.h. auch für  $x \neq 0$  (Histogramme:  $H^P \neq H^Q$ ) kann der Distanzwert verschwinden
- Geometrische Deutung für  $\varepsilon$ -Anfragen: der Anfragebereich ist unbeschränkt



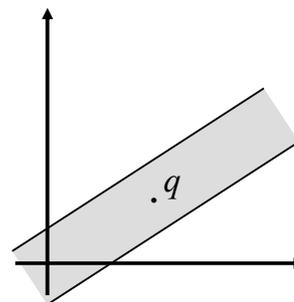
## Ähnlichkeitsmatrizen (5)

### Lemma:

falls  $x A x^T = 0$  für ein  $x \neq 0$ , dann auch  $\lambda x A \lambda x^T = 0$  für alle  $\lambda \in \mathbb{R}$



$\varepsilon$ -Anfragebereich einer PSD gewichteten euklid. Distanz



$\varepsilon$ -Anfragebereich einer PSD quadratischen Form



## Anfragebearbeitung (1)

### Anfragebearbeitung für Quadratische Formen

Problem:

- Die Auswertung einer quadratischen Form in  $d$  Dimensionen benötigt  $O(d^2)$  viele arithmetische Operationen
- Die Laufzeit einer sequentiellen Auswertung für eine Datenbank mit  $n$  Objekten ist  $O(n \cdot d^2)$ .

⇒ Gesucht sind schnellere Verfahren.

Diagonalisierung der Ähnlichkeitsmatrix

- Idee: Quadratische Form in gewichteten euklidischen Abstand überführen.
- Grundlage: Jede PD-Matrix  $A$  läßt sich diagonalisieren.
- Das bedeutet: Es gibt eine Diagonalmatrix  $W = \text{diag}(w_1, \dots, w_n)$  sowie eine orthonormale Matrix  $V$ , d.h.  $V V^T = V^T V = Id$ , so dass gilt:  
 $A = V W V^T$



## Anfragebearbeitung (2)

- Damit gilt für alle Vektoren (Histogramme)  $p$  und  $q$ :

$$\begin{aligned} D_A(p, q) &= \sqrt{(p - q) \cdot V W V^T \cdot (p - q)^T} \\ &= \sqrt{(pV - qV) \cdot W \cdot (pV - qV)^T} = D_W(pV, qV) \end{aligned}$$

- $D_A$  ist also äquivalent zum gewichteten euklidischen Abstand  $D_W$ , nachdem alle beteiligten Vektoren der Basistransformation  $V$  unterworfen wurden.
- Die Gewichte  $w_1, \dots, w_n$  sind die *Eigenwerte* der Matrix  $A$ , die zugehörigen Spalten in  $V$  sind die *Eigenvektoren* von  $A$ .



## Anfragebearbeitung (3)

- Dimensionsreduktion
  - Da  $A$  positiv definit ist, sind alle Eigenwerte  $w_i$  positiv, d.h.  $w_i > 0$ , und ein “Abschneiden” der  $d$ -dimensionalen Vektoren  $pV$  und  $qV$  auf  $r < d$  Dimensionen liefert eine garantierte untere Schranke für  $D_W = D_A$ :
$$D_{W,r}(pV, qV) = \sqrt{\sum_{i=1}^r w_i (pV_i - qV_i)^2} \leq \sqrt{\sum_{i=1}^d w_i (pV_i - qV_i)^2} = D_W(pV, qV)$$
  - Die untere Schranke-Eigenschaft stellt die Vollständigkeit der Anfragebearbeitung sicher.
  - Beobachtung: Die Transformation hängt von der Ähnlichkeitsmatrix  $A$  ab!



## Anfragebearbeitung (4)

- Anpassbarkeit durch den Benutzer
  - Ähnlichkeit hat einen stark subjektiven Charakter.
  - Ein Benutzer ist mit der vorgegebenen Matrix möglicherweise nicht zufrieden.
  - Neuaufbau eines Indexes zur effizienten Datenbanksuche bzgl.  $D_A$  ist sehr teuer; erwünscht sind deshalb flexiblere Techniken, die eine Modifikation der Ähnlichkeitsmatrix zur Anfragezeit ermöglichen.



## Texturen (1)

### Texturen in Bildern

Die Textur beschreibt die Beschaffenheit von Bildsegmenten (dargestellte Oberflächen)

### Texturmodell in QBIC

([FBF+ 94] Faloutsos C., Barber R., Flickner M., Hafner J., et al.: *Efficient and Effective Querying by Image Content*. Journal of Intelligent Information Systems 3, 231-262, 1994.)

- **Gerichtetheit, Orientiertheit (Directionality)**



- Vorhandensein von (Vorzugs-)Richtungen
- Beispiel: Mauerfugen versus Kieselsteine
- aus Verteilung der Gradientenrichtungen in den Bildern



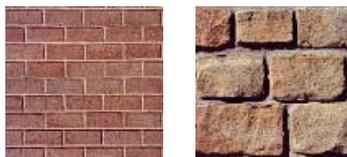
## Texturen (2)

- **Kontrast (Contrast)**



- Lebendigkeit (Unruhe) eines Musters
- Beispiel: weiße Wand versus Sand
- Berechnung aus der Varianz im Grauwert histogramm

- **Granularität (Coarseness)**



- Größenordnung der Textur
- Beispiele: Sand vs. Kieselsteine; feine vs. grobe Mauer
- Berechnung durch über das Bild verschobene Fenster unterschiedlicher Größe

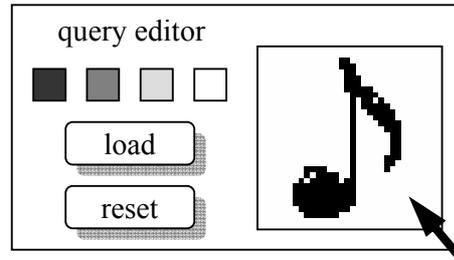


# Formen in Bildern (1)

## Pixelbasiertes Modell für Formen in Bildern

([AKS 98] Ankerst M., Kriegel H.-P., Seidl T.: A Multistep Approach for Shape Similarity Search in Image Databases. IEEE Transactions on Knowledge and Data Engineering (TKDE) 10(6), 1998, 996-1004.)

- Anwendungen für formbasierte Ähnlichkeitssuche
  - Query By Sketch (z.B. mausgesteuerter Editor)



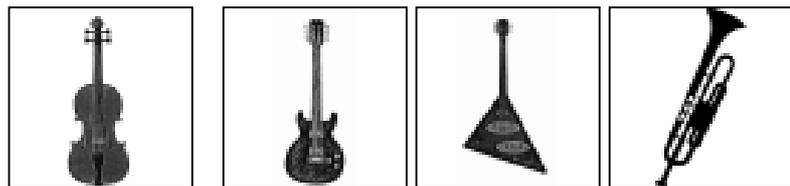
- Vorgegebene Bilder
  - Grafikarchive
  - Patentrecherche
  - Medizinbilder



# Formen in Bildern (2)

- Konzept der Differenzbilder

Anfrage- und  
Ergebnisbilder  
(64×64 Pixel)



Differenzbilder und  
euklidische Distanz



d = 128.2

d = 223.9

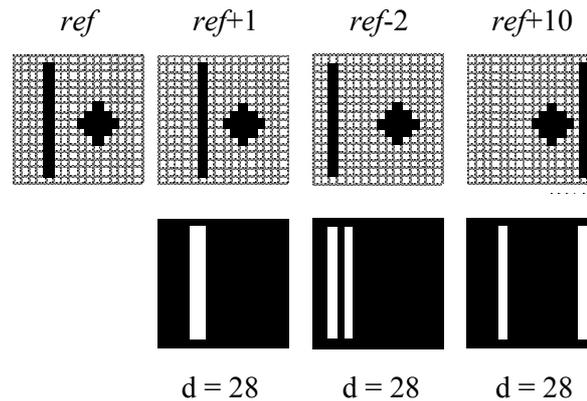
d = 424.7



## Formen in Bildern (3)

### Probleme mit der euklidischen Distanz

- Beispiel “*ref*”
  - Referenzbild: Balken links von einem Punkt.
  - Bei den Vergleichsbildern ist der Balken um +1, -2 bzw. +10 Pixel horizontal verschoben.
  - Euklidischer Abstand zu *ref* bleibt jedoch immer derselbe.



## Formen in Bildern (4)

- Problembeschreibung
  - Leichte Verschiebungen sind von starken Veränderungen nicht unterscheidbar.
  - Invarianz gegenüber globalen Translationen stellt keine Lösung dar.
  - Erwünscht ist die Robustheit gegenüber kleinen, lokalen Veränderungen.



## Formen in Bildern (5)

### Umgebungsbasierte Distanzfunktion

Lösungsidee: Betrachte die Nachbarschaften der Pixel

Statt nur die direkt übereinanderliegenden Pixel wie im Differenzbild zu betrachten, werden nun auch benachbarte Pixel zur Ähnlichkeitsbewertung herangezogen.

- Beispiel für unterschiedliche Gewichtungen benachbarter Pixel

0	0	0	0	0	0	0
0	0	0.06	0.11	0.06	0	0
0	0.06	0.28	0.44	0.28	0.06	0
0	0.11	0.44	1	0.44	0.11	0
0	0.06	0.28	0.44	0.28	0.06	0
0	0	0.06	0.11	0.06	0	0
0	0	0	0	0	0	0

$w_{3,3}$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0.01	0.09	0.19	0.25	0.19	0.09	0.01	0
0	0.06	0.25	0.56	1	0.56	0.25	0.06	0
0	0.01	0.09	0.19	0.25	0.19	0.09	0.01	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$w_{4,2}$

- Die Gewichtung kann für alle Pixel gleich gewählt werden, sie kann aber auch über die Bildfläche hinweg variieren.
- Durch die Gewichtung  $w_{1,1}$  wird der euklidische Abstand beschrieben.



## Formen in Bildern (6)

- Formale Definition

- Lokale Nachbarschaften: zu jedem Pixel  $p$  zweier Bilder  $F, G$  werden die Pixel  $p'$  in der Umgebung betrachtet:

$$d_w(F, G) |_p = \sum_{p' \text{ Pixel}} w(p - p') \cdot (F(p') - G(p'))$$

Im obigen Modell sind nur wenige Gewichte  $w(\Delta) = w(p - p')$  ungleich Null.

- Gesamtdistanz: die mit dem lokalen Umgebungsabstand gewichteten Pixeldifferenzen werden über alle Pixel der gesamten Bildfläche addiert:

$$\begin{aligned} d_w(F, G)^2 &= \sum_{p \text{ Pixel}} (F(p) - G(p)) \cdot d_w(F, G) |_p = \\ &= \sum_{p \text{ Pixel}} (F(p) - G(p)) \cdot \sum_{p' \text{ Pixel}} w(p - p') \cdot (F(p') - G(p')) = \\ &= \sum_{p \text{ Pixel}} \sum_{p' \text{ Pixel}} (F(p) - G(p)) \cdot w(p - p') \cdot (F(p') - G(p')) = \\ &= (F - G) \cdot W \cdot (F - G)^T \end{aligned}$$



## Formen in Bildern (7)

- Beobachtung:
  - Die Distanzfunktion  $d_w(F, G)$  ist eine quadratische Form.
  - Die Ähnlichkeitsmatrix  $W$  enthält für jedes Pixelpaar  $p, p'$  den Eintrag  $w(p - p')$ .
- Übertragung auf Farbbilder
  - Bisher werden Pixeldifferenzen  $F(p) - G(p)$  bezüglich der Grauwerte benutzt.
  - Bei Farbbildern berechnet man nicht Grauwertdifferenzen  $F(p) - G(p)$ , sondern Farbabstandswerte  $d_C(F(p), G(p))$  bzgl. einer Farbdistanzfunktion  $d_C$ .

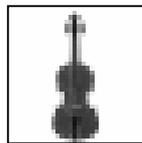


## Formen in Bildern (8)

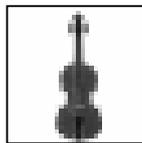
### Experimentelle Ergebnisse

- Testdatenbank mit 10.000 Clip Arts der Auflösung  $32 \times 32 = 1.024D$

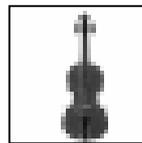
Referenzbild



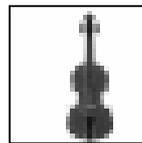
+1 Pixel



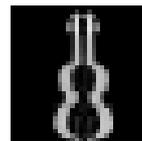
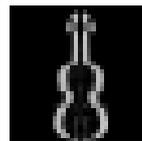
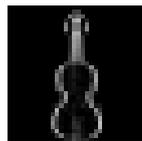
+2 Pixel



+3 Pixel



← verschobene  
Anfragebilder



← Differenzbilder

Nachbarschafts-

bereiche

$w_{1,1}$ :

1

9

328

$w_{5,1}$ :

1

3

92

$w_{12,1}$ :

1

3

10

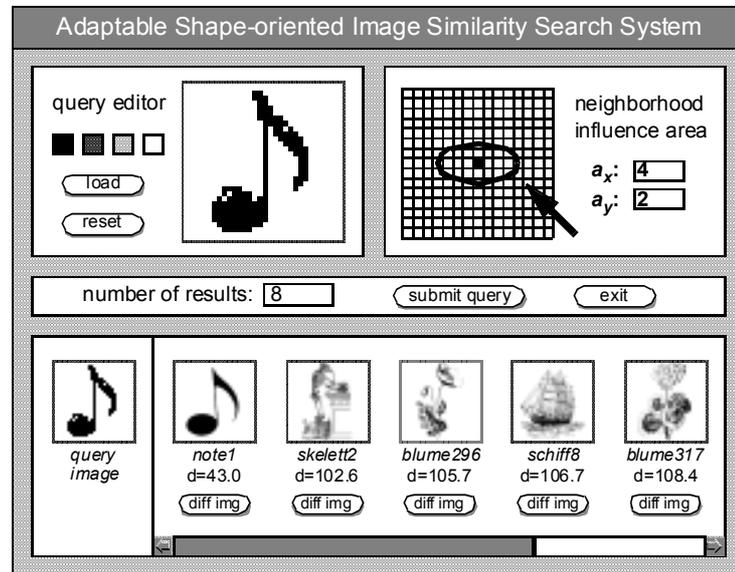
- Beobachtung: Für den Nachbarschaftsbereich  $w_{12,1}$  wird die um drei Pixel nach rechts verschobene Violine unter die “Top-Ten” eingeordnet.



## Formen in Bildern (9)

### Beispiel für eine Benutzeroberfläche

- Anfragespezifikation: Sketch Editor, Ellipsoid Editor, Parameter  $k$ .
- Ergebnisausgabe: Anfragebild, Ergebnisbilder, Distanzwerte, Differenzbilder



## Inhalt

1. Sequenzen

2. Bilder

3. Polygone, 3D



# Ähnlichkeitsmodelle für Polygone

- Approximation von Formen durch größensortierte Rechtecke [Jag 91]
- Ähnlichkeit von Kantenzügen [GM 93] [MG 93] [MG 95]
- Partielle Ähnlichkeitssuche per Fourier-Transformation [BKK 97a] [BK 97]
- Angular Profile, LWL-Codierung (Länge-Winkel-Länge) [BMH 92] [BK 97]
- Section Coding [BKK 97b] [BK 97]

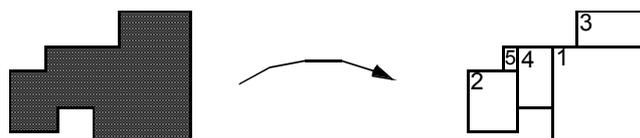


# Überdeckungsmodell für 2D Formen

([Jag 91] Jagadish H. V.: A Retrieval Technique for Similar Shapes. Proc. ACM Int. Conf. on Management of Data (SIGMOD) 1991, 208-217.)

## Grundidee

- Ähnlichkeitsmodell für 2D-Formen, hier: leicht erweiterbar auf 3D-Formen.
- Distanzfunktion: Flächeninhalt der symmetrischen Differenz zweier Formen.
- Hier: Translations- und skalierungsinvariant, nicht jedoch rotationsinvariant.
- Vorgehen: Repräsentation der Formen durch rechteckige Überdeckungen.
- Speicherung der Rechtecksflächenmaßzahlen z.B. der Größe nach geordnet.





# Überdeckungsmodell für 2D Formen

## Objektmodell

- Formen sind als konturierte Objekte gegeben (d.h. Polygone).
- Extraktion von Formen aus Grauwertbildern möglich, solange klare Konturen bestimmt werden können (Probleme z.B. bei teilweise verdeckten Objekten).
- Die Polygone müssen nicht konvex sein (d.h. Einbuchtungen möglich).



# Rechtecksüberdeckungen (1)

## • Additive Überdeckung

Durch eine Folge von Rechtecken  $\langle R_1, R_2, \dots, R_k \rangle$  ist eine Folge von additiven Überdeckungen  $\langle C_0, C_1, \dots \rangle$  wie folgt definiert:

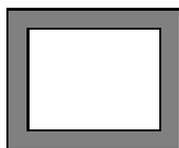
$$C_0 = \emptyset, \quad C_{i+1} = C_i \cup R_{i+1}$$

## • Allgemeine Überdeckung

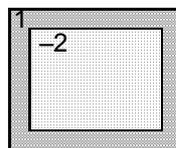
Neben dem Hinzufügen von Rechtecksflächen ( $\cup$ ) ist auch das Entfernen von Rechtecksflächen ( $-$ ) möglich:

$$C_0 = \emptyset, \quad C_{i+1} = C_i \cup R_{i+1} \quad \text{oder} \quad C_{i+1} = C_i - R_{i+1}$$

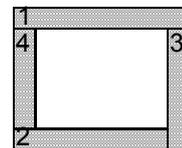
- Für endliche Formen  $S$  konvergieren (additive) Überdeckungssequenzen schon im Endlichen, d.h. es gibt ein  $K$ , so dass  $C_K = S$ , und wir definieren  $C_j = C_K$  für  $j \geq K$ .
- Überlappungen sind erlaubt, sollen aber möglichst gering ausfallen



gegebene Form



allg. Überdeckung



additive Überdeckung



## Rechtecksüberdeckungen (2)

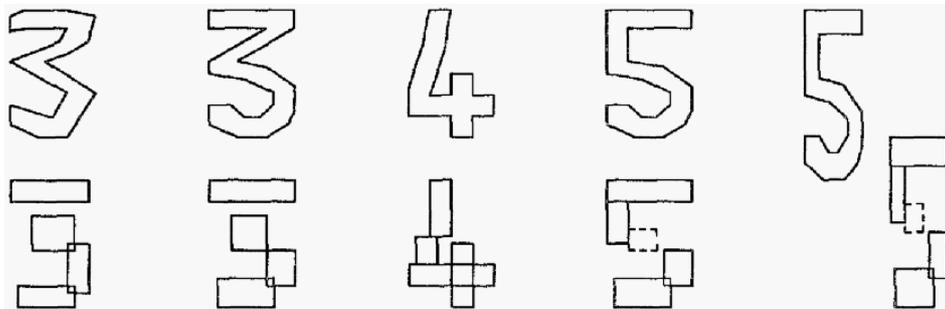
- **Approximative Rechtecksüberdeckungen**
  - Anstatt aller Rechtecke einer Überdeckung werden nur wenige gespeichert.
  - Das Entfernen kleiner Rechtecke entspricht dem Beseitigen hochfrequenter Fehler wie Schmutzflecken oder Diskretisierungsfehlern (z.B. bei eingescannten Bildern).
- **Approximationsqualität**
  - Die ersten Rechtecke einer Überdeckung sollen schon eine möglichst gute Approximation der ursprünglichen Form liefern.
  - Kumulatives Fehlerkriterium: Die Approximationsfehler der Überdeckungssequenz  $\langle C_0, C_1, \dots, S \rangle$  werden sukzessive aufsummiert, die Gesamtsumme zählt:

$$\text{kumulativer Fehler} = \sum_{i=1..n} |S - C_i|$$



## Rechtecksüberdeckungen (3)

- Minimierung der Gesamtsumme führt zu Minimierung der “frühen” Fehler  $|S - C_i|$  für kleine  $i$ , da diese mehrfach gewertet werden.
- Beispiel: Fünf Ziffern und zugehörige erste Rechtecke:



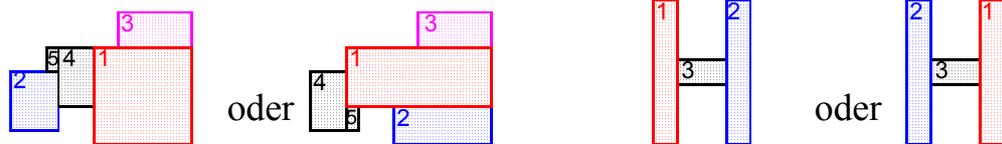
Quelle: [Jag 91]



## Probleme der Rechtecksüberdeckungen (1)

- **Nicht-eindeutige Repräsentation**

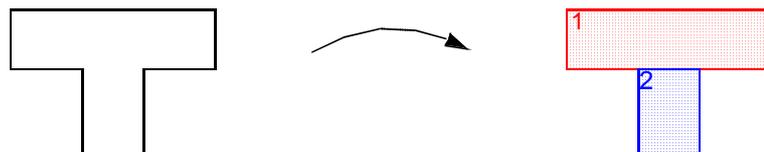
- Es kann unterschiedliche optimale Zerlegungen eines Objektes geben.
- Insbesondere bei Symmetrie ist die Reihenfolge der Rechtecke nicht eindeutig.
- Lösung: Objekt mehrfach speichern oder mehrfache Anfragen für eine Form.



## Probleme der Rechtecksüberdeckungen (2)

- **Rechteckige Formen**

- Wird eine Form schon durch wenige Rechtecke exakt beschrieben, besteht die Überdeckungssequenz ggf. aus weniger Elementen, als im Index gespeichert werden.
- Lösung: Parameter der weiteren Rechtecke nicht als Punkte, sondern als Bereiche im Featureraum speichern → räumliche Indexe (statt Punktindexe) verwenden.





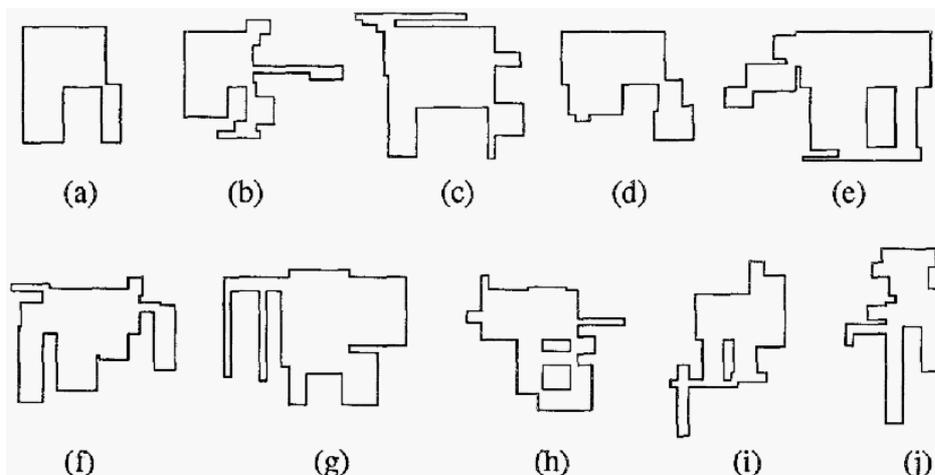
## Ähnlichkeitsanfragen (1)

- **Experimentelles Umfeld**
  - Datenbank: 16.000 synthetische Formen.
  - Jede Form wurde aus 10 zufällig erzeugten Rechtecken zusammengesetzt, danach wurden jeweils additive Überdeckungen berechnet.
  - Im Index wurden jeweils die größten drei Rechtecke der Überdeckung gespeichert.
  - Anfragen: Bereichsanfragen um zufällig ausgewählte Formen der Datenbank



## Ähnlichkeitsanfragen (2)

- Beispiel für das Ergebnis einer Ähnlichkeitsanfrage:  
(a: Anfrageform; b – j: Ergebnisformen)



Quelle: [Jag 91]

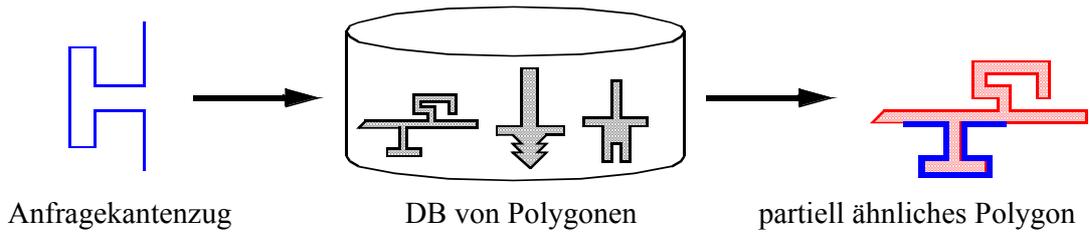


# Partielle Ähnlichkeitssuche (1)

([BKK 97] Berchtold S., Keim D. A., Kriegel H.-P.: *Using Extended Feature Objects for Partial Similarity Retrieval*. VLDB Journal 6(4), 1997, 333-348.)

- **Ziel**

- Translations-, rotations- und skalierungsinvariante Ähnlichkeit von Polygonen.
- Unterstützung von partieller Ähnlichkeitssuche.

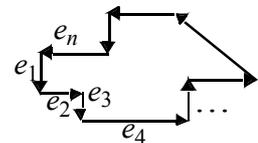


# Partielle Ähnlichkeitssuche (2)

- **Polygondarstellung**

- Ein Polygon ist ein geschlossener Kantenzug  $\langle e_1, \dots, e_n \rangle$ .
- Polygon wird als parametrische Kurve  $p(t)$  dargestellt, wobei  $t$  von 0 bis  $2\pi$  läuft.
- Da der Kantenzug geschlossen ist, gilt:

$$p(2\pi) = \int_{0 \dots 2\pi} p(t) dt = \sum_{i=1}^n e_i = p(0) = 0$$



- Die Länge des Kantenzuges ist auf  $2\pi$  normiert:  $\sum_{i=1}^n |e_i| = 2\pi$
- Über zwei Parameter  $a$  und  $b$  lassen sich Ausschnitte aus Polygonen beschreiben.

- **Invariante Darstellungen**

- *Skalierungsinvarianz*: Durch Normierung auf Länge  $2\pi$  erreicht.
- *Translationsinvarianz*: Die Polygone sind nicht absolut positioniert.
- *Rotationsinvarianz*: Betrachte statt der konkreten Richtungen die Krümmungen!
- Invarianz gegenüber gewähltem Startpunkt: In partieller Ähnlichkeit enthalten



## Partielle Ähnlichkeitssuche (3)

- **Invariante Darstellungen**

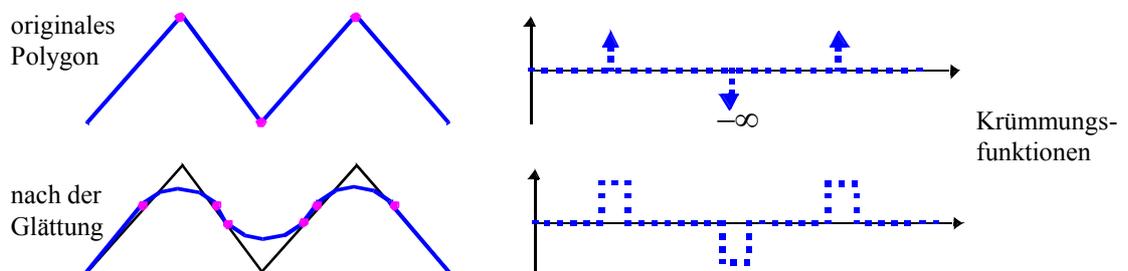
- *Skalierungsinvarianz*: Durch Normierung auf Länge  $2\pi$  erreicht.
- *Translationsinvarianz*: Die Polygone sind nicht absolut positioniert.
- *Rotationsinvarianz*: Betrachte statt der konkreten Richtungen die Krümmungen!
- Invarianz gegenüber gewähltem Startpunkt: In partieller Ähnlichkeit enthalten



## Partielle Ähnlichkeitssuche (4)

- **Repräsentation der Krümmungen**

- Problem: Kantenzug ist nicht stetig differenzierbar (scharfe Knicke an den Ecken).
- Lösung: Glättung durch Approximation der Ecken mit Kreisausschnitten.



- Die Amplituden der Krümmungsfunktion hängen vom gewählten Radius  $r$  ab.
- Die Breite eines Ausschlages in der Krümmungsfunktion hängt vom Winkel  $\alpha_i$  ab.
- Für die Krümmungsfunktion eines Polygons werden die Fourier-Koeffizienten gespeichert (analytische Berechnung)



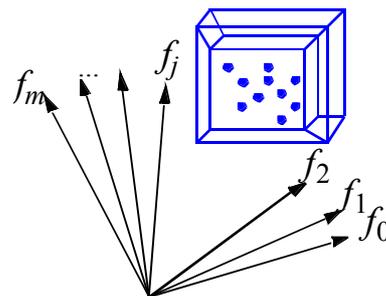
## Behandlung der partiellen Ähnlichkeit (1)

- **Repräsentation von Polygonausschnitten.**
  - Ausschnitte  $(a, b)$  aus Kantenzügen  $p(t)$ , d.h.  $a \leq t \leq b$ , werden selbst wieder auf den Bereich  $0, \dots, 2\pi$  skaliert. Der Datenraum ist zyklisch für den Fall  $b < a$ .
  - Für die Polygonausschnitte werden ebenfalls die Krümmungsfunktionen berechnet und deren Fourier-Koeffizienten ermittelt.
  - Für den Featureraum werden einige der Koeffizienten als Dimensionen ausgewählt.
  - Die Ähnlichkeit von Polygonausschnitten wird über eine geeignete Distanzfunktion im hochdimensionalen Featureraum definiert (z.B.  $p$ -Norm).



## Behandlung der partiellen Ähnlichkeit (2)

- **Unendlichkeitsproblem**
  - Es gibt unendlich viele Ausschnitte  $(a, b)$  und damit Punkte im hochdimensionalen Featureraum, die weder alle berechnet noch alle gespeichert werden können.
  - Lösung: Speichere nicht die einzelnen Punkte, sondern jeweils das minimal umgebende Hyperrechteck mehrerer Featurepunkte.
  - Dazu verschiedene Strategien, wie viele Featurepunkte zusammengefasst werden.





## Behandlung der partiellen Ähnlichkeit (3)

- **Zusammenfassen von Featurepunkten**
  - Erster Schritt: Zusammenfassen der Featurepunkte für Abschnitte, die auf derselben Kante des Polygons beginnen und auf einer bestimmten anderen Kante enden (d.h.  $n \cdot n$  viele Hyperrechtecke).
  - Beobachtung: Manche dieser Boxen sind sehr klein, andere sehr groß.
  - Kleine Boxen können weiter zusammengefasst werden:
    - Die Hyperrechtecke für benachbarte Kanten im Polygon werden zusammengefasst.
  - Große Boxen können weiter zerlegt werden:
    - Zerlegung im Featureraum:
      - Auswahl bestimmter Achsen oder
      - Auswahl aller Achsen: dann gibt es  $2^m$  viele Zerlegungsprodukte.
    - Alternative: Zerlege Boxen im zweidimensionalen Parameterraum, d.h. fasse andere Polygonausschnitte zusammen.



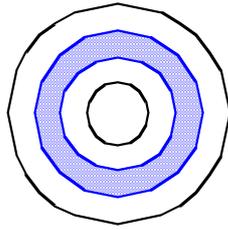
## Formhistogramme für 3D-Objekte

- ([AKKS 99] Ankerst M., Kastenmüller G., Kriegel H.-P., Seidl T.: 3D Shape Histograms for Similarity Search and Classification in Spatial Databases. Proc. Int. Symposium on Large Spatial Databases (SSD) 1999 (LNCS 1651), 207-226.)
- **Ziel**
    - Translations- und rotationsinvariante Suche nach ähnlichen Formen im 3D.
    - Objekte sind als Mengen von Oberflächenpunkten gegeben.
    - Beispielanwendungen: Moleküle, CAD-Bauteile.
  - **Grundidee: Formhistogramme**
    - Partitioniere den 3D-Raum in Zellen (Histogramm-Bins).
    - Bestimme den Anteil an Punkten des Objektes pro Zelle (normiertes Histogramm).
    - Durch die Normierung werden die Histogramme unabhängig von der Punktedichte.

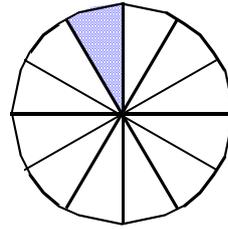


# Formhistogramme, Beispiele

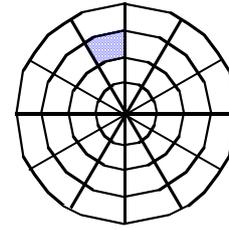
- Verschiedene Raumpartitionierungen



Schalenmodell



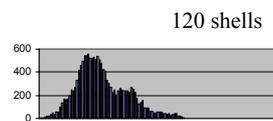
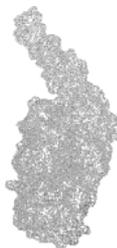
Sektorenmodell



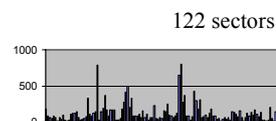
kombiniertes Modell

Quelle: [AKKS 99]

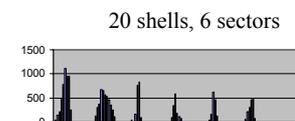
- Beispielobjekt Seryl-tRNA Synthetase (PDB-Code: 1SER-B)



Schalenmodell



Sektorenmodell



kombiniertes Modell



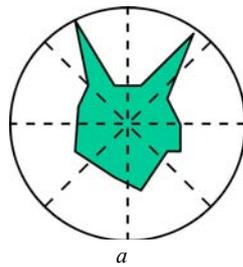
# Formhistogramme, Definition

- **Formale Definition der Histogramme**
  - *Schalenmodell*: Definiere die Bins über den Abstand zum Mittelpunkt, d.h. Anzahl der Punkte auf der jeweiligen Schale.
  - *Sektorenmodell*: Anzahl der Punkte im jeweiligen Sektor.
  - *Kombiniertes Modell*: Synthese aus Schalen- und Sektorenmodell.
- **Invarianzen**
  - Translationsinvarianz durch Lagenormierung:  
Verschiebung des Schwerpunkts eines Objektes in den Ursprung.
  - Rotationsinvarianz durch Hauptachsentransformation:
    - Drehung der Objekte, so dass die Hauptachsen auf den Koordinatenachsen liegen.
    - unnötig beim Schalenmodell, dieses ist inhärent rotationsinvariant.

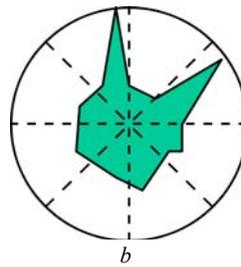
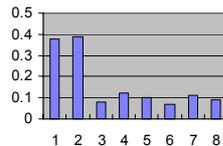


# Formhistogramme, Distanzfunktion (1)

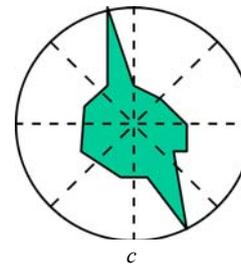
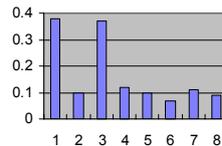
- Probleme mit dem euklidischen Abstand (Beispiel im 2D)



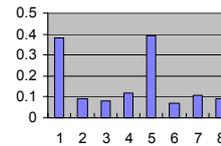
a



b



c



Quelle: [AKKS 99]

- Die Form *c* gilt als genauso ähnlich zu *a* wie zu *b*.
- Die Ähnlichkeit räumlich benachbarter Histogramm-Bins wird nicht berücksichtigt.



# Formhistogramme, Distanzfunktion (2)

- Quadratische Formen als Distanzfunktionen

$$d_A(p, q) = \sqrt{(p - q) \cdot A \cdot (p - q)^T} = \sqrt{\sum_i \sum_j a_{ij} \cdot (p_i - q_i) \cdot (p_j - q_j)}$$

- Für die Formhistogramme enthält die Ähnlichkeitsmatrix  $A = [a_{ij}]$  die Ähnlichkeit von Einträgen in den Zellen *i* und *j* der Raumpartitionierung
- Diese Ähnlichkeit läßt sich aus dem Abstand  $d_{ij}$  der Zellen *i* und *j* berechnen, z.B.:  
$$a_{ij} = \exp(-\sigma (d_{ij} / d_{max})^2)$$
- Als Abstand  $d_{ij}$  eignet sich beispielsweise der euklidische Abstand



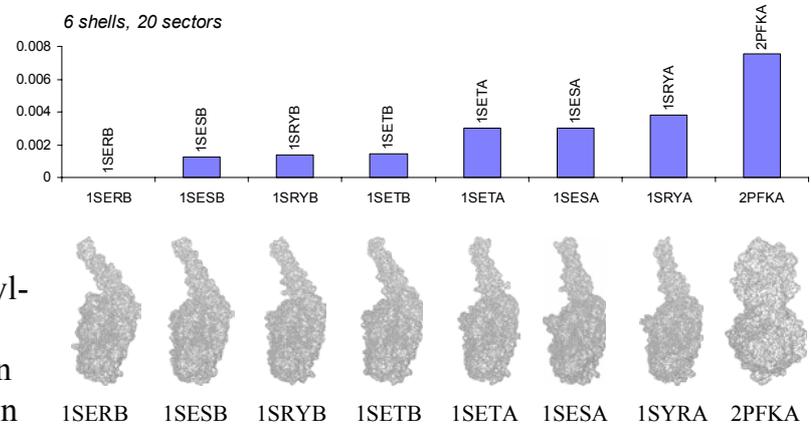
# Formhistogramme, Ergebnisse (1)

- **Einfache Suche nach ähnlichen Molekülen**

Anfrage: 1SER-B

Die erwarteten Ergebnisse (Seryl-Proteasen) treten auf den ersten Positionen auf.

Das erste nicht-Seryl-Protein (2PFKA) unterscheidet sich in seiner Form sowie in seinem Distanzwert zur Anfrage deutlich von den Seryl-Proteinen.



Quelle: [AKKS 99]



# Formhistogramme, Ergebnisse (2)

- **Klassifikation auf der gesamten Datenbank**

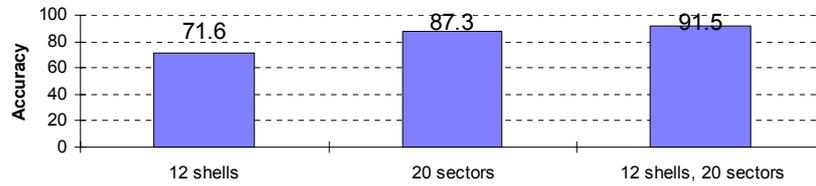
- Das Ähnlichkeitsmodell wird in einem nächsten-Nachbar-Klassifikator verwendet.
- D.h. ein Anfrageobjekt bekommt das Klassenlabel des ähnlichsten Objektes aus der Datenbank zugeordnet.
- Klassifikationsgenauigkeit: Wie oft wird die Klassenentscheidung richtig getroffen?
- “Leave-One-Out”-Experiment: jedes Objekt wird gegen die restliche DB angefragt.



# Formhistogramme, Ergebnisse (3)

- **Ergebnisse**

- Vergleich verschiedener Histogrammmodelle (12 Schalen, 20 Sektoren, 12×20 Zellen).



- Vergleich verschieden-granularer Raumpartitionierungen (12 bzw. 120 Schalen bzw. 20 bzw. 122 Sektoren).

