10 Multimedia-Retrieval

Ziel:
Ähnlichkeitssuche auf multimedialen Datenobjektmengen (Bildern, Videos, Audios, etc.) aufgrund spezifischer Features (z.B. Farben, Konturen, Texturen in Bildern)

Problemkreise:
- Feature-Auswahl und Feature-Extraktion
- Effiziente Suche
  - Ähnlichkeitsfilter
  - Mehrdimensionale Indexstrukturen

Feature-Auswahl und Ähnlichkeitsmaße: einfache Beispiele

Audiosignale sind im einfachsten Fall diskret abgetastete Zeitreihen der Signalamplituden $\vec{x}$

$\rightarrow$ $dist(\vec{x},\vec{y}) = (\vec{x}-\vec{y})\cdot(\vec{x}-\vec{y})^T = \sum_{i=1}^{m}(x_i - y_i)^2$

Farbbilder können im einfachsten Fall durch die Häufigkeitsverteilung der Farben unter den Pixeln (von Bildteilen) charakterisiert werden (Farbhistogramme)

$\rightarrow$ $dist(\vec{x},\vec{y}) = (\vec{x}-\vec{y})\,A\,(\vec{x}-\vec{y})^T = \sum_{i=1}^{m}\sum_{j=1}^{m} A_{ij}\,(x_i - y_i)\,(x_j - y_j)$

mit einer m×m-Farb-Farb-Ähnlichkeitsmatrix A

Die Ähnlichkeitsmaße müssen keine Vektorraumnormen sein (z.B. Algorithmen für Ähnlichkeit von Fingerabdrücken), sollten aber Metriken sein mit $dist(x,y) \le dis(x,z) + dist(z,y)$

Effizienzprobleme

Berechnung der Ähnlichkeit zwischen Query und einem Kandidaten hat u.U. sehr großen Rechenaufwand
(Beispiel: Farbverteilungsähnlichkeit mit 65536 Farben)
→ Ähnlichkeitsfilter

A priori sind alle Dokumente Kandidaten, so daß bei n Dokumenten n Ähnlichkeitsberechnungen durchgeführt werden müssen
(I/O- und Berechnungsaufwand eines sequentiellen Scans)
→ Mehrdimensionale Indexstrukturen
oder Signaturen o.ä.

Ähnlichkeitsfilter

Für Feature-Raum F (z.B. $(R_0^+)^m$) finde
effizient berechenbare Funktion f: F×F → [0,1], so daß
für alle x, y ∈ F und alle ε gilt:
dist (x, y) < ε ⇒ f(x,y) < ε,
also: f(x,y) ≤ dist(x,y)

Algorithmus:
1) Für Schwellwert ε bestimme diejenigen Datenobjekte c aus der gesamten Datenkollektion, für die f(q,c) < ε
2) Für alle in Schritt 1 ausgewählten Kandidaten c teste dist(q,c) < ε

Ähnlichkeitsfilter für Signalvektoren-Ähnlichkeit

Für Signalvektoren $\vec{x},\vec{y} \in R^m$
mit $dist(\vec{x},\vec{y}) = (\vec{x}-\vec{y})\cdot(\vec{x}-\vec{y})^T = \sum_{i=1}^{m}(x_i - y_i)^2$

wende **Diskrete Fourier-Transformation** (DFT) an:

$$\hat{x}_\nu = \frac{1}{\sqrt{m}}\sum_{k=1}^{m} x_\nu\, e^{-i2\pi k\nu/m} = \frac{1}{\sqrt{m}}\sum_{k=1}^{m} x_\nu \cos(-2\pi k\nu/m) + i\sin(-2\pi k\nu/m)$$

$$mit\ i = \sqrt{-1},\ \nu = 1,2,\ldots,m$$

mit der Umkehrtransformation $x_\nu = \frac{1}{\sqrt{m}}\sum_{k=1}^{m}\hat{x}_\nu\, e^{-i2\pi k\nu/m}$

Theorem von Parseval: $\|\vec{x}\|^2 = \sum_{k=1}^{m}|x_\nu|^2 = \sum_{k=1}^{m}|\hat{x}_\nu|^2 = \|\vec{\hat{x}}\|^2$

Es folgt für alle h≤m: $f(\vec{x},\vec{y}) = \sum_{k=1}^{h}|\hat{x}_k - \hat{y}_k|^2 \le \sum_{k=1}^{m}|x_k - y_k|^2 = dist(\vec{x},\vec{y})$

Ähnlichkeitsfilter für Farbhistogramm-Ähnlichkeit

Anstelle kompletter Farbhistogramme kann man die mittleren Rot-, Grün- und Blauanteile eines Bildes (mit N Pixeln) in einer Filterfunktion f verwenden:

$$R(\vec{x}) = \frac{1}{N}\sum_{p=1}^{N}R(p) \qquad G(\vec{x}) = \frac{1}{N}\sum_{p=1}^{N}G(p) \qquad B(\vec{x}) = \frac{1}{N}\sum_{p=1}^{N}B(p)$$

$$f(\vec{x},\vec{y}) = (R(\vec{x}) - R(\vec{y}))^2 + (G(\vec{x}) - G(\vec{y}))^2 + (B(\vec{x}) - B(\vec{y}))^2$$

Es gilt: $f(\vec{x},\vec{y}) \le a * dist(\vec{x},\vec{y}) = a(\vec{x}-\vec{y})\,A\,(\vec{x}-\vec{y})^T$
mit einer von A abhängigen Konstanten a

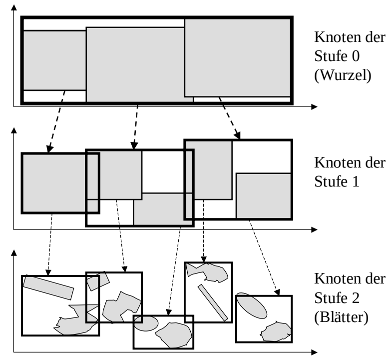
Mehrdimensionale Indexstrukturen für Ähnlichkeitssuche: R-Bäume (1)

Ein R-Baum ist ein balancierter, hohler Mehrwegebaum, der mehrdimensionale Datenobjekte (Punktdaten wie z.B. Feature-Vektoren oder „ausgedehnte“ Objekte wie Polygone) und Wegweiser als achsenparallele, umschreibende Rechtecke (**MBRs**: minimum bounding rectangles, bounding box) repräsentiert.

Ein MBR kann durch die Koordinaten des „linken unteren“ Punktes und des „rechten oberen“ Punktes beschrieben werden.

Invarianten des R-Baums sind:
- Ein Wegweiser in einem Nichtblattknoten ist das MBR des Teilbaums, auf den der Wegweiser zeigt.
- Das MBR eines Teilbaums ist das MBR aller MBRs in dem Teilbaum.

Mehrdimensionale Indexstrukturen für Ähnlichkeitssuche: R-Bäume (2)

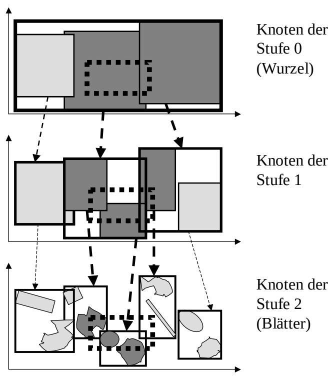


Suchen auf R-Bäumen (1)

Mehrdimensionale Bereichsanfrage („Fenstersuche“) mit Anfrage-MBR q:
Finde alle Datenobjekte x, die sich mit q schneiden oder in q enthalten sind.

```
Algorithmus:
  t := Wurzel des R-Baums;
  search (q, t);
mit Prozedur
search (q, n):
  if n ist ein Blatt ist then
     Gebe alle x in n aus,
     die sich mit q schneiden oder in q enthalten sind
  else
     T:= die Menge aller Wegweiser-MBRs in Knoten n,
          die sich mit q schneiden;
     for each t in T do search (q, t) od;
  fi
```


Suchen auf R-Bäumen (2)



Bottom-Up-Aufbau eines R-Baums (1)

Gegeben: n Datenpunkte x1, ..., xn $\in [0,1]^m$
(z.B. die Mittelpunkte der MBR der Datenobjekte)

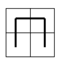
Betrachte ein m-dimensionales Gitter R = $\{i/k \mid i=0, ..., k-1\}^m$ mit k Zellen pro Dimension, wobei k eine Zweierpotenz 2^d ist, und eine raumfüllende Kurve ψ: R $\rightarrow \{0, 1, ..., k^m\}$, so daß ψ bijektiv und soweit wie möglich distanzbewahrend ist

Ladealgorithmus:
1) Sortiere x1, ..., xn aufsteigend nach ihren Positionen ψ(x1), ..., ψ(xn)
2) Fasse eine passende Anzahl bzgl. dieser Sortierung aufeinanderfolgender Datenpunkte zu einem Blattknoten zusammen.
3) Konstruiere die Baumknoten oberhalb der Blattebene in Bottom-up-Weise.

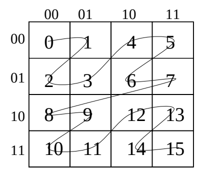
Bottom-Up-Aufbau eines R-Baums (2)

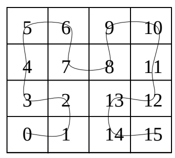
Geeignete raumfüllende Kurven (aus der Familie der Fraktale):

Peano-Kurve (Z-Kurve):
Für einen Punkt x mit binär codierten Gitterkoordinaten x11, ..., x1d (in der 1. Dimension), ... xm1, ..., xmd (in der m. Dimension) ist ψ(x) = x11 x21 ... xm1 x12 ... xm2 ... xm1 ... xmd
(bitweise Verschränkung)

Hilbert-Kurve:
H1 auf 2x2-Gitter: 

Hi auf 2^ix2^i-Gitter:
H1-Kurve auf oberster Stufe mit geeignet rotierter oder gespiegelter H(i-1)-Kurve in jedem Viertel

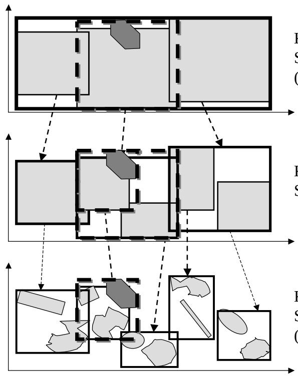




Einfügen in R-Bäume (1)

Einfügen eines MBRs b (eines neuen Datenobjekts):
  t := Wurzel des R-Baums;
  insert (b, t);
mit Prozedur
insert (b, n):
  if n ist ein Blatt then
    Füge b in n ein, berechne das MBR von b neu und
    aktualisiere den Wegweiser im Vater von n;
    Falls n übergelaufen ist, teile n in zwei Knoten auf (Split);
  else
    Bestimme unter allen Wegweiser-MBRs in n
    das günstigste MBR t (z.B. bezüglich der Differenz des
    Hypervolumens oder Umfangs des MBRs von t ∪ b gegenüber t);
    insert (b, t);
    Aktualisiere ggf. das MBR von n;
  fi

Einfügen in R-Bäume (2)



Knoten der Stufe 0 (Wurzel)

Knoten der Stufe 1

Knoten der Stufe 2 (Blätter)

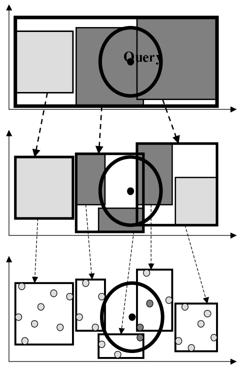
Split eines R-Baum-Knotens

Teile MBRs eines Knotens n (Datenobjekte oder Wegweiser) so auf zwei Knoten n und n‘ auf, daß
1) die Summe der Hypervolumen oder Umfänge von n und n‘ möglichst klein wird und
2) die Platzauslastung von n und n‘ nicht unter einen Mindestschwellwert sinkt.

Heuristiken:
Berechne 2 Cluster für die MBRs des ursprünglichen Knotens n
oder:
Bestimme unter den MBRs in n zwei Saat-MBRs s und s‘
(z.B. diejenigen mit maximaler Distanz unter allen Paaren) und
ordne MBR x in n je nach Distanz s oder s‘ zu
Speichere alle s zugeordneten MBRs in n und
alle s‘ zugeordneten MBRs in n‘

ε-Ähnlichkeitssuche auf R-Bäumen

Top-Down-Suche nach allen Teilbäumen, die sich mit einer Hypersphäre mit Mittelpunkt q und Radius ε schneiden (ggf. approximiert durch Suche nach MBR der Hypersphäre)



Knoten der Stufe 0 (Wurzel)

Knoten der Stufe 1

Knoten der Stufe 2 (Blätter)

N-Nearest-Neighbor-Suche auf R-Bäumen (1)

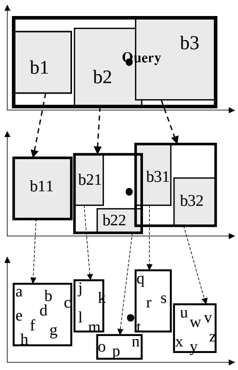
Finde die N nächsten Nachbarn zu einem Punkt q

Algorithmus:
NN: array [1..N] of record point: pointtype; dist: real end;
for i:=1 to N do NN[i].dist := ∞ od;
repeat
  priority queue Q := Wurzel t;
  Knoten n := first(Q);
  if n ist ein Blatt then
    for each p in n do if dist(p,q) < max(NN[1..N].dist)
                then nehme p in NN auf fi od;
  else
    for each Wegweiser-MBR b in n do
      lowerbound := dist (q, nächster Punkt von MBR(n));
      if lowerbound < max(NN[1..N].dist)
      then insert(Q, b) fi
    od;
until Q ist leer or dist(q, first(Q)) > max(NN[1..N].dist)

N-Nearest-Neighbor-Suche auf R-Bäumen (2)

N = 4

NN: ---	Q: b2 b3 b1
NN: ---	Q: b3 b22 b21 b1
NN: ---	Q: b31 b22 b21 b32 b1
NN: t r s q	Q: b22 b21 b32 b1
NN: t n r p	Q: b21 b32 b1
NN: t n r p	Q: b32 b1
NN: t n r p	Q: ---



Knoten der Stufe 0 (Wurzel)

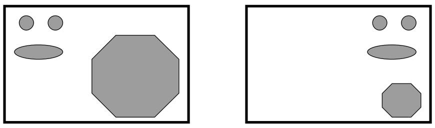
Knoten der Stufe 1

Knoten der Stufe 2 (Blätter)

Das WALRUS-System (Wavelet-based Retrieval of User-defined Scenes)

Ziel: Ähnlichkeitssuche auf Bildern, bei der Ähnlichkeit weitgehend skalierungs- und translationsinvariant ist

Beispiel:



sind ähnlich

Ansatz: 1) Bestimme Features mittels Multiskalen-Transformation, konkret: Wavelets
2) Bestimme Features für Teilbilder (sliding windows)

Wavelet-Transformation

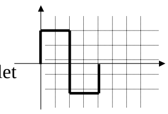
Wavelets sind Familien von Funktionen, die zur Repräsentation von Signalen in verschiedenen Skalierungsstufen geeignet sind

Mathematisch: ein Wavelet ist eine Funktion ψ mit der Eigenschaft $0 < 2\pi \int_R \frac{\|\hat{\psi}(x)\|^2}{\|x\|} dx < \infty$

wobei $\hat{\psi}$ die Fourier-Tranformierte von ψ ist.

Diese Eigenschaft impliziert $\frac{1}{\sqrt{2\pi}} \int_R \psi(t)\, dt = 0$

Beispiel: Haar-Wavelet



Haar-Wavelet-Transformation auf eindimensionalen Signalen

Die Haar-Wavelet-Transformation der Stufe k bildet Mittelwerte benachbarter Elementen der Stufe k-1 und die (zweifache) Abweichung der Elemente zum jeweiligen Mittelwert als sogenannte Detailkoeffizienten (Zerlegung des Signals in dominanten „glatten“ Anteil und untergeordnete, „rauhe“ Details)

Beispiel x = (0 0 1 2 4 4 6 10)

Haar-Wavelet-Transformierte von x:

Stufe	Mittelwerte	Detailkoeffizienten
1	(0 1.5 4 8)	(0 1 0 4)
2	(0.75 6)	(1.5 4)
3	(3.375)	(5.25)

Normierung der Detailkoeffizienten der Stufe k mit Faktor $/\sqrt{2^{max(k)-k}}$

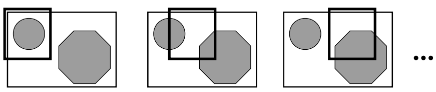
Transformierte von x besteht aus Mittelwert der höchsten Stufe und normierten Detailkoeffizienten aller Stufen
$\rightarrow$ x‘ = (3.375 5.25 1.06 2.83 0 0.5 0 2)

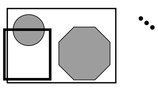
Haar-Wavelet-Transformation auf zweidimensionalen Bildern

Ein Ansatz:
Zerlege 2D-Bild in 2x2-Boxen von Pixeln mit linker oberer Ecke (2i-1,2j-1) und berechne Mittelwert der 4 Pixel an der Positioin (2i-1,2j-1) und Detailkoeffizienten für die anderen drei Positionen

```
procedure ComputeWavelet (w×w image I, transform W, w):
for i:=1 to w/2 do
for j=1 to w/2 do
  A[i,j] := ( I[2i-1,2j-1] + I[2i,2j-1] + I[2i-1,2j] + I[2i,2j] )/4;
  W[w/2+i,j] := ( -I[2i-1,2j-1] + I[2i,2j-1] - I[2i-1,2j] + I[2i,2j] )/4;
  W[i,w/2+j] := ( -I[2i-1,2j-1] - I[2i,2j-1] + I[2i-1,2j] + I[2i,2j] )/4;
  W[w/2+i,w/2+j] := (I[2i-1,2j-1] - I[2i,2j-1] - I[2i-1,2j] + I[2i,2j] )/4;
od; od;
if w>2 then ComputeWavelet (A, W, w/2)
       else W[1,1] := A[1,1] fi;
```


Bildähnlichkeit in WALRUS





Berechne Wavelet-Signaturen für alle Bildregionen, die sich durch Sliding Windows ergeben

Zum Vergleich von Bildern d und q mit Regionsmengen D und Q:
1) Finde ähnliche Regionspaare x $\in$ D, y $\in$ Q mit sim(x,y) $\geq$ ε
2) Berechne Gesamtähnlichkeit von d und q mit ähnlichen Regionspaaren (x1, y1), ..., (xm, ym) auf der Basis von

$$sim(d,q) = \frac{area(\bigcup_{i=1}^{m} x_i) + area(\bigcup_{i=1}^{m} y_i)}{area(d) + area(q)}$$

plus Besonderheiten zur Effizienzsteigerung

Features zur Klassifikation und Ähnlichkeitssuche auf Videos

Viele Ansätze und Techniken:
- Bestimmung von Kanten (Konturen) in einzelnen Frames durch Analyse der Intensitätsänderung von Pixeln
- Bestimmung von Kameraschnitten (Shots) durch Analyse der Kantenbewegungsrate (Motion Rate) in aufeinanderfolgenden Frames
- Analyse der zeitlichen Verteilung von Kameraschnitten (z.B. nützlich für Erkennung von Videopiraterie)
- Repräsentation eines Videos durch Schlüssel-Frames (z.B. je ein Frame pro Shot oder Scene)
- Erkennen von Untertiteln, Gesichtern, Stimmung (z.B. mit Lernen entsprechender Farbverteilungen anhand von Trainingsdaten) etc.
- Analyse der gesprochenen Sprache
- Ausnutzen von Annotationen (z.B. MPEG-7-Metadaten)