Image Retrieval

Seminar mit Themen zum Bereich
Multimediale Informationssysteme (MMIS) bei der
Arbeitsgruppe Datenbanken und Informationssysteme

Betreut von Markus Bon

Ausarbeitung von Björn Sanders

Matrikelnummer : 340539

Sommersemester 2002

Abgabetermin: 26.06.2002

Inhalt

1 Einleitung..... 3

2 Grundlagen..... 3
2.0 Definition: Bildgitter..... 4
2.1 Definition: Zelle..... 4
2.2 Definition: Objektstruktur..... 5
2.3 Definition: Rechteck..... 5
2.4 Definition: Image Database..... 5

3 Bildkomprimierung..... 6
3.0 Diskrete Fourier Transformation..... 7
3.1 Diskrete Cosinus Transformation……..... 8

4 Segmentation..... 8
4.0 Connected region……..... 8
4.1 Homogeneity predicate……...… 9

5 Ähnlichkeitsbasiertes Suchen..... 11
5.0 Distanzfunktion..... 12
5.1 Transformation..... 14

6 Image Database..... 15

7 Aussicht..... 21

8 Schaubildverzeichnis..... 21

9 Literaturverzeichnis..... 22

1. Einleitung:

Image Retrieval, die Suche nach bestimmten Bildern in Datenbanken, ist Thema dieser Seminararbeit. Dies beinhaltet die Klärung nach der Frage, was denn überhaupt Image Retrieval ist und wieso es in Zukunft immer mehr an Wichtigkeit gewinnt.
Über die letzten Jahre hat sich ein Trend entwickelt, mehr und mehr Photographien oder Bilddateien (Images) in Datenbanken zu speichern. Es gibt verschiedenste Bereiche in der Wirtschaft und Forschung, bei denen solche Bilddateien eine zentrale Rolle spielen. Zum Beispiel sammelte die NASA in ihrem EOSDIS Projekt Daten über die Erde; jeder amerikanische Bürger, der einen Pass besitzt, ist durch sein Photo registriert. Auch kann man sich vorstellen, dass Krankenhäuser und Ärzte eine Menge an Bildern, Beispielsweise Röntgenbildern, aufbewahren müssen. Zwar werde Grossteile dieser Bilder noch in nichtdigitaler Form aufbewahrt, aber mit dem Rückgang der Preise in der Computerbranche werden immer mehr Bilder in digitaler Form gespeichert. Wo man sich früher noch per Hand durch die Ordner wühlen musste, soll dies in Zukunft der Computer erledigen. Daher braucht man effiziente Methoden um Bilder in Datenbanken zu verwalten und Algorithmen um eine effiziente Suche nach ihnen zu ermöglichen.

Im relationalen Datenmodell finden wir im allgemeinen textuelle Anfragen. Bei diesen war es so, dass ein Benutzer seine Anfrage eingetippt hat und darauf bezogen eine Antwort erhielt. Wenn man sich eine moderne Suchmaschine fürs Internet vorstellt, so werden dort die verschiedenen Quellen nach einem Stichwort durchsucht.
Die Frage die sich aufdrängt, ist doch, ob so etwas ähnliches bei Bildern auch möglich ist ?

Man kann sich folgendes Szenario vorstellen, in dem ein Polizist ein Photo von einem Verdächtigen gefunden hat, aber leider nicht weiß, wer die Person auf dem Photo ist. Nach einem Stichwort kann hier nicht gesucht werden. Der Inhalt des Bildes ist entscheidend. Nun könnte der Polizist eine Anfrage an die Datenbank stellen, die in etwa so oder ähnlich lauten könnte:

„Finde alle **Bilder**, die **ähnlich** zu diesem hier sind und gib mir die Namen der Personen zurück, die darauf zu sehen sind“.

Bei dieser Anfrage kann man gut erkennen, was die zwei wesentlichen Unterschiede zu den bisherigen Anfragen sind:

1. Die Anfrage enthält ein Bild.
2. Die Anfrage ist ungenau, was hier durch das Wort ähnlich dargestellt werden soll, was dazu führt, dass man eine neue, „ungenaue“ Art der Suche benötigt.

Im Folgenden werden wir zunächst einen Blick auf Bilddateien im allgemeinen werfen, um uns anschließend mit Bildkomprimierungen zu beschäftigen wie zum Beispiel GIFs, JPEGs oder TIFFs. Danach betrachten wir verschiedene Segmentierungstechniken, um dann näher auf die ähnlichkeitsbasierte Suche einzugehen. Abschließend werde ich noch ein Beispiel angeben, wie man Images in ein Datenbank-Management-System (DBMS) integrieren kann.

2. Grundlagen:

Da im Laufe des Aufsatzes verschiedene Begriffe wie Zelle, Objektstruktur, Rechteck oder Image Database verwendet werden, werden hier einige Definitionen eingeführt, um den Umgang mit diesen Begriffen zu erleichtern.

2.0 Definition: Bildgitter

Jedes Bild I hat eine zugehöriges Paar von geraden Zahlen (m, n), die Gitterauflösung des Bildes genannt werden. Dies teilt das Bild in (m x n) Zellen gleicher Größe, die man Bildgitter nennt. Jede Zelle in einem gegebenen Gitter (m x n) besteht aus Pixeln.

2.1 Definition: *Zelle*

Eine Zelle dient der Unterteilung eines Bildes.
Jede Zelle hat bestimmte Eigenschaften, die durch ein Tripel (*Name, Werte, Methode*) beschrieben werden, wobei *Name* einen String enthält, der die Eigenschaft wiedergibt, *Werte* ist eine Menge von Werten, die diese Eigenschaft annehmen kann und *Methode* ist der Algorithmus, der die Eigenschaften verarbeitet.

Stellt man sich nun ein schwarz-weiß Bild vor, so wäre eine mögliche Eigenschaft einer Zelle:

(swfarbe, {s, w}, swalgo)

Der Name der Eigenschaft ist hier swfarbe, die Werte, die die Eigenschaft annehmen kann, sind s für Schwarz und w für Weiß und swalgo ist zum Beispiel ein Algorithmus, der eine Zelle als Eingabe nimmt, über die Anzahl der schwarzen bzw. weißen Pixel die „Farbe“ bestimmt und dann schließlich entweder Schwarz oder Weiß zurückliefert.
Bei einem Bild, dass Grauabstufungen als Eigenschaft hat, könnte eine Zelle folgendermaßen dargestellt werden:

(graustufe, [0,1], graualgo)

Die Eigenschaft heißt Graustufe und es können Werte in dem Intervall [0,1] angenommen werden, wobei 0 = weiß und 1 = schwarz bedeutet.

Eine Zelle wird außerdem noch durch die Kanten beschrieben, die sie abgrenzen.
Die Kanten wiederum werden durch vier integer Werte dargestellt:

(XLB, XUB, YLB und YUB)

Alle Pixel einer Zelle liegen somit innerhalb dieser Kanten:

$$\{(i,j) \mid XLB \leq i \leq XUB \;\&\; YLB \leq j \leq YUB\}$$

Somit könnte ein Beispiel für den obigen graualgo folgendermaßen aussehen:

$$graua\lg o(zelle) = \frac{\sum_{XLB \leq i < XUB} \sum_{YLB \leq j < YUB} fin\deg rau(i,\ j)}{(XUB - XLB) \times (YUB - YLB)}$$

findegrau(i,j) gibt den Grau-Wert vom Pixel (i, j) wieder

Das RGB-Farbmodell verwendet die Grundfarben Rot, Grün und Blau zur additiven Farbmischung.
Eine Farbe wird durch ein Tripel (r, g, b) beschrieben, so dass z.B. rot 3, grün 6 und blau 1 für eine ganz bestimmte Farbe steht.
Bei einem farbigen Image erhält man somit die folgenden drei Eigenschaften:
(Rot,{0,...,7}), (Grün{0,...,7}), (Blau{0,...,7})

Jedes Objekt besitzt eine bestimmte Struktur, die man verwenden kann, um dieses Objekt zu charakterisieren.

2.2 Definition: *object shape (Objektstruktur)*

Als object shape bezeichnet man jede Anzahl von Punkten P, so dass gilt,
wenn p, $q \in P$, dann existiert eine Folge von Punkten $p_1,..., p_n$ die alle in P sind,
so dass:

1. $p = p_1$ und $q = p_n$

2. für alle $1 \leq i < n$, ist p_{i+1} ein Nachbar von p_i
   dies ist dann der Fall, wenn:
   - $p_i = (X_i, Y_i)$ und $p_{i+1} = (X_{i+1}, Y_{i+1})$ und
   - (X_{i+1}, Y_{i+1}) eine der der folgenden Bedingungen erfüllt:

$(X_{i+1}, Y_{i+1}) = (X_i + 1, Y_i)$	$(X_{i+1}, Y_{i+1}) = (X_i - 1, Y_i)$
$(X_{i+1}, Y_{i+1}) = (X_i, Y_i + 1)$	$(X_{i+1}, Y_{i+1}) = (X_i, Y_i - 1)$
$(X_{i+1}, Y_{i+1}) = (X_i + 1, Y_i + 1)$	$(X_{i+1}, Y_{i+1}) = (X_i + 1, Y_i - 1)$
$(X_{i+1}, Y_{i+1}) = (X_i - 1, Y_i + 1)$	$(X_{i+1}, Y_{i+1}) = (X_i - 1, Y_i - 1)$

Die Definition 2.2 besagt also nichts anderes, als dass für eine Anzahl von Punkten eine gültige Struktur existiert, wenn es zu zwei beliebigen Punkten ein Pfad gibt, der vollständig in der Struktur liegt.

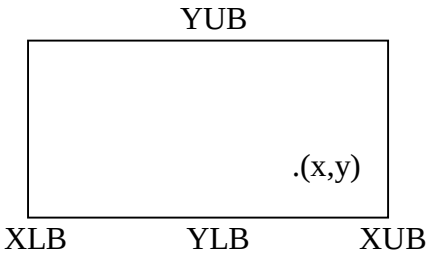
Zur Vereinfachung werden wir im Folgenden nur Objekte mit einer rechteckigen Struktur betrachten.

2.3 Definition: Rechteck

Ein Rechteck ist eine Objektstruktur P, so dass XLB, XUB, YLB und $YUB \in$ Int existieren, mit

$P = \{(x, y) \mid XLB \leq x < XUB \ \& \ YLB \leq y < YUB\}$

<u>Schaubild 2.0:</u>



Nun fehlt nur noch der Begriff der Image Datenbank, in die später die Bilder gespeichert werden.

2.4 Definition: *Image Database (IDB)*

Eine IDB besteht aus einem Tripel(GI, Prop, Rec)

1. GI ist eine Menge von gridded images (Raster/Gitter) von der Form (Image, m, n)
2. Prop ist eine Menge von Zellen Eigenschaften
3. Rec ist eine Abbildung, die jedem Bild eine Menge von Rechtecken beschreibende Objekten zuordnet.

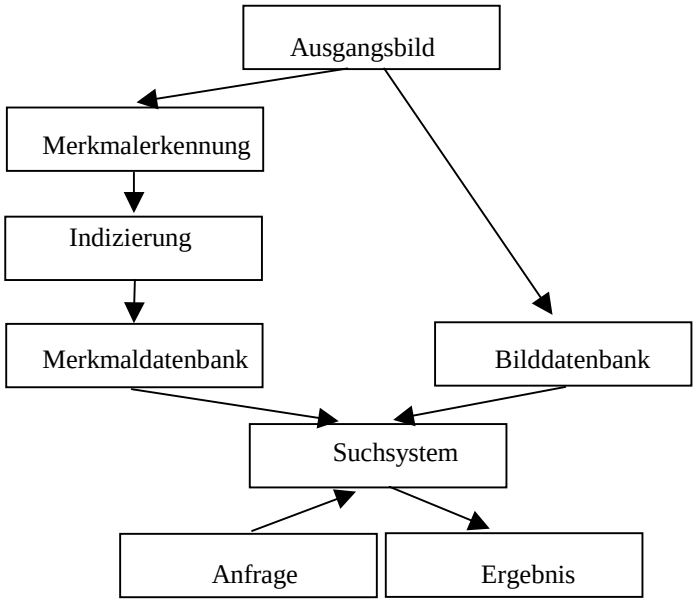
Bei dieser Art der Bilddarstellung muss man zwei wichtige Faktoren beachten:

1. Images sind meist große Objekte, die aus einem Array (p1 x p2) von Pixeln bestehen. Das explizite Speichern von Eigenschaften auf Pixelbasis ist nur schwer zu bewerkstelligen, was zu einer Gruppe von image compression (Bildkomprimierungs-) Algorithmen geführt hat, die in Abschnitt 3 beschrieben werden.

2. Wenn ein Bild gegeben ist, ob nun in komprimierter oder unkomprimierter Form, muss festgelegt sein, welche Merkmale (Features) in dem Bild erscheinen, nach dem es später indiziert wird. Der Prozess zu Einteilung in Segmente wird als Segmentation bezeichnet, auf den wir in Abschnitt 4 näher eingehen.

Nach der Einteilung eines Bildes in Segmente, braucht man nun noch geeignete Suchoperationen, die ein gesamtes Bild oder nur eine Anzahl von Segmenten eines Bildes mit einem anderen Bild oder eine Anzahl von Segmenten vergleicht. Diese Techniken werden in Abschnitt 5 beschrieben.

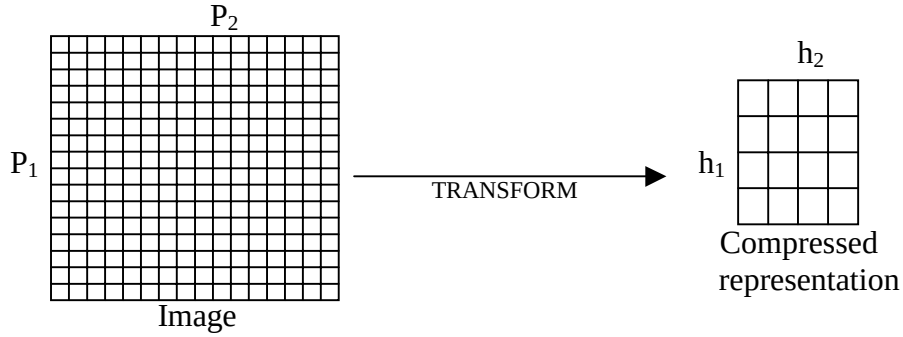
Schaubild 2.1:



In diesem Schaubild gehen wir von der Ausgangssituation aus, dass wir ein Bild haben, dass in eine Datenbank gespeichert wird. Aus dem Bild werden die Merkmale extrahiert und indiziert. Diese Merkmale werden anschließend ebenfalls in eine Datenbank gespeichert. Das Suchsystem muss nun nur eine Verbindung zwischen der Merkmaldatenbank und der Bilddatenbank herstellen. Tätigen wir nun eine Anfrage auf dem Suchsystem, so durchsucht es die Merkmaldatenbank mit unserer Anfrage und durchsucht anschließend mit den Treffern aus der Merkmalsdatenbank die Bilddatenbank und liefert uns dann Bilder, die zu unserer Anfrage passen, als Ergebnis zurück.

3. Bildkomprimierung:

3.0 Schaubild:



Nehmen wir ein zweidimensionales Bild I, dass aus den Pixeln (p_1 x p_2) besteht, dann ist es typisch, dass I(x, y) eine Zahl ist, die ein oder mehrere Attribute eines Pixels beschreibt. Zum Beispiel könnte I(x, y) eine Nummer zwischen 0 und 256 sein, was dann den zugehörigen RGB Anteil beschreibt. Wenn man sich nun überlegt, wie viele Informationen gespeichert werden müssen, kann man sich leicht vorstellen, dass es zu extrem vielen Einträgen (bis zu mehreren Millionen) in der zugehörigen Image Matrix kommen kann.
Daher hat man schnell nach einer Möglichkeit gesucht, diese Matrix zu komprimieren, wie es in Schaubild 3.0 vereinfacht dargestellt wird.

Die Erstellung einer komprimierten Repräsentation des ursprünglichen Bildes besteht aus zwei Schritten:

1. Size selection: Die Größe h des komprimierten Bildes wird durch den Bilddatenbank Designer gewählt. Das Verhältnis von p_1 zu h_1 und p_2 zu h_2 gibt die „Stärke/Größe“ der Komprimierung an. Um so größer h ist, um so genauer ist die Repräsentation, aber mit zunehmender Größe nimmt auch die Komplexität des Erstellens eines Indizes zu. Wir nehmen an, dass die komprimierte Größe des Bildes aus einem Paar (h_1, h_2) , mit $h_1, h_2 \in \mathbf{N}^+$ (positiver natürliche Zahlen) besteht.

2. Transform selection: Der Benutzer muss eine Transformation auswählen, die geeignet ist, das Bild zu komprimieren.

Im Folgenden werde ich kurz auf zwei bekannte Transformationen eingehen.

3.0 The Discrete Fourier Transformation (DFT)

$$DFT\ (x, y) = \frac{1}{p_1 p_2} \sum_{a=0}^{p_1-1} \sum_{a=0}^{p_2-1} (I(a,b) \times e^{-j(\frac{2\pi xa}{p_1} + \frac{2\pi yb}{p_2})})$$

p_1, p_2 Höhe und Breite des ursprünglichen Bildes.
a = 0... p_1 -1
b = 0... p_2 -1

wobei j die Komplexe Zahl $\sqrt{-1}$ ist.

Die DFT hat einige gute Eigenschaften. Man kann das original Image aus der DFT Repräsentation wiedergewinnen, indem man die inverse Transformation DFT^{-1} auf DTF(I) anwendet. Nicht alle Komprimierungstechniken unterstützen eine 100% Invertierbarkeit. Während DFT in der Theorie invertierbar ist, sieht die Praxis meist anders aus. Da man DFT

nicht alleine verwendet, sondern mit anderen nicht invertierbaren Operationen verwendet, verliert man die Eigenschaft der Invertierbarkeit.
DTF hat außerdem die Eigenschaft den euklidischen Abstand zu bewahren, den man zum Beispiel bei der Bildsuche als Maß für Ähnlichkeit verwenden kann.

3.1 The Discrete Cosine Transform (DCT)

$$DCT(i,j) = \frac{2}{\sqrt{p_1 \times p_2}} \alpha(i) \times \alpha(j) \sum_{r=0}^{p_1-1} \sum_{s=0}^{p_2-1} (\cos(\frac{(2r+1) \times \pi i}{2r}) \times \cos(\frac{(2s+1) \times \pi i}{2s}))$$

$\alpha(i), \alpha(j) = \frac{1}{\sqrt{2}}$ wenn u,v = 0
sonst $\alpha(i), \alpha(j) = 1$

DCT hat den Vorteil, dass sie schnell berechenbar ist.

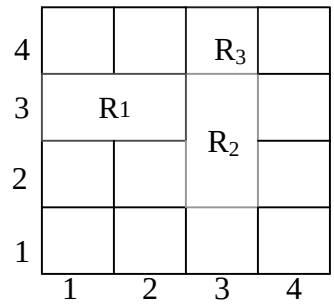
Sowohl DCT als auch DFT wurden verschiedenen Performanzansprüchen angepasst, wie zum Beispiel Rechenzeit oder Qualität der Komprimierung.

4. Segmentation:

Bisher sind wir davon ausgegangen, dass es in einem Bild bestimmte Merkmale, wie Farbe oder Texturen, gibt, die von besonderem Interesse sind, dass man diese Bereiche finden kann und dass man das Bild nach diesen Merkmalen in homogene Segmente aufteilen kann.

Nehmen wir an, I ist ein Bild, dass (m x n) Zellen enthält. Im Worst-Case ist eine Zelle, wie bereits in der Definition genannt, ein Pixel, aber in den meisten Fällen ist eine Zelle ein Rechteck bestehend aus einer Menge von Pixeln.

4.0 Schaubild:



4.0 Definition: connected region

Eine connected region R in einem Bild I ist eine Menge von Zellen, so dass, wenn (x_1, y_1), $(x_2, y_2) \in R$ sind, eine Folge von Zellen $C_1,..., C_n$ aus R existiert, so dass:

1. $C_1 = (x_1, y_1)$ und
2. $C_n = (x_2, y_2)$ und
3. der Euklidische Abstand zwischen den Zellen C_i und C_{i+1} für alle $1 \leq i < n$ 1 beträgt

Schaubild 4.0 zeigt drei connected regions R_1, R_2, und R_3
Außerdem werden in diesem Schaubild folgende Dinge verdeutlicht:

1. ($R_1 \cup R_2$) ist eine connected region und
2. ($R_2 \cup R_3$) ist eine connected region und
3. ($R_1 \cup R_2 \cup R_3$) ist eine connected region, aber
4. ($R_1 \cup R_3$) ist eine keine connected region. Der Grund dafür liegt in dem euklidischen Abstand zwischen den Zellen (2,3), die zu R_1 gehört und (3,4), die zu R_3 gehört, da der Abstand Wurzel 2 beträgt und somit größer als 1 ist.

Nun kehren wir zur eigentlichen Segmentierung zurück und es stellt sich die Frage, wonach man sich bei der Segmentierung eines Bildes richten kann ?
Um diese Frage zu beantworten betrachten wir zunächst den Begriff der Homogenitäts-prädikate, die eine zentrale Rolle bei der Segmentierung einnehmen.

4.1 Definition: homogeneity predicate
Ein Homogenitätsprädikat in Bezug auf ein Bild I ist eine Funktion H die eine beliebige connected region R als Eingabe nimmt und entscheidet, ob eine Region homogen oder nicht homogen ist.

Da diese Definition doch sehr allgemein gehalten ist, sollen zur Verdeutlichung ein paar Beispiele für solche Prädikate folgen:

1. Sei δ eine reale Zahl mit $0 < \delta \leq 1$ und wir betrachten schwarz-weiß Bilder. Ein einfaches Homogenitätsprädikat wäre zum Beispiel H_{δ}^{bw} :

   H_{δ}^{bw} liefert „true“ zurück, wenn über (100 * δ) % der Zellen in Region R die selbe Farbe haben.

   Betrachten wir folgendes Beispiel:

   4.0 Tabelle

Region	Zahl der schwarzen Pixel	Zahl der weißen Pixel
R_1	800	200
R_2	900	100
R_3	100	900

   Gegeben seien folgende Prädikate $H_{0,8}^{bw}$, $H_{0,89}^{bw}$ und $H_{0,92}^{bw}$
   Als Ergebnisse dieser drei Prädikate erhält man folgende

   4.1 Tabelle

Region	$H_{0,8}^{bw}$,	$H_{0,89}^{bw}$	$H_{0,92}^{bw}$
R_1	true	false	false
R_2	true	true	false
R_3	true	true	false

   Man muss aber beachten, dass bei diesem Prädikat nur die Anzahl der Pixel mit gleicher Farbe entscheidend ist, nicht welche Farbe am häufigsten vorkommt.

2. Ein anderes Beispiel wäre, wenn jedes Pixel des Bildes durch eine reale Zahl zwischen 0 und 1 inklusive dargestellt wird. Diese Zahl wird bw-level genannt, denn 0 bedeutet „weiß“, 1 bedeutet „schwarz“ und jeder Wert dazwischen beschreibt eine Graustufe. Man kann sich nun ein Homogenitätsprädikat vorstellen, bei dem f eine Funktion ist, die jeder Zelle eine Zahl zwischen 0 und 1 inklusive zuordnet. Hinzukommt ein Abweichungsfaktor (Ungenauigkeitsfaktor) $0 \leq \eta \leq 1$ und einem Schwellwert (Schrankenwert) δ, wie oben beschrieben.
   $H^{f,\eta,\delta}(R)$ ist „true“, wenn

$$\frac{\{(x,y) \,\|\, bw-level(x,y) - f(x,y)| < \eta \;\}}{(m \times n)} > \delta$$

   Dabei ist der entscheidende Teil der Formel

   | bw-level(x,y)-f(x,y) | < η

   Hier wird überprüft, wie weit die Vorhersage von f von dem tatsächlichen bw-level in (x, y) abweicht. Ist dies kleiner als der Abweichungsfaktor, wird anschließend überprüft, ob mehr Zellen als δ dies erfüllen, wenn dies der Fall ist, liefert $H^{f, \eta, \delta}(R)$ „true“ zurück, ansonsten „false“.

3. Eine andere Methode ist, dass man zuerst alle bw-levels in einer gewissen Spanne klassifiziert. Zum Beispiel bekommen alle bw-levels zwischen 0 und 0.1 die 1, zwischen 0.1 und 0.2 die 2 usw. bis zwischen 0.9 und 1 die 10 zugewiesen. δ ist wieder wie oben definiert. $H^{class}(R)$ liefert „true“ zurück, falls (100 * δ) % der Zellen aus der Region R in die selbe Klassifizierung fallen.

4. Eine weiter Methode ist eine dynamische Version von H^{class}. Das Prädikat $H_{\delta}^{dyn,\eta}$ Liefert „true“ zurück, wenn (100 * δ) % der Zellen der Region R innerhalb eines Bereiches η einer realen Zahl r liegen, wobei r zur Laufzeit festgelegt wird.

Trotz der genannten Beispiele muss man beachten, dass die Homogenitätsprädikate nicht unbedingt mit bw-levels arbeiten müssen. Sie können auch auf Farbskalen, Texturen oder Höhen/Tiefen basieren.
Sei nun ein Image I durch die Pixel (m x n) gegeben und eine Segmentierung von I in Bezug auf ein Homogenitätsprädikat P in eine Menge von Regionen $R_1,..., R_k$, so dass gilt:

1. $R_i \cap R_j = \emptyset$ für alle $1 \leq i \neq j \leq k$
2. $I = R_1 \cup \ldots \cup R_k$
3. $H(R_i)$ = „true“ für alle $1 \leq i \leq k$
4. für alle unterschiedlichen i, j, $1 \leq i, j \leq n$, für die $R_i \cup R_j$ eine connected region ist, liefert $H(R_i \cup R_j)$ = „false“

Wir nehmen eine einfache (4 x 4) Region an, die folgende bw-levels enthält:
4.2 Tabelle

Zeile/Spalte	1	2	3	4
1	0.1	0.25	0.5	0.5
2	0.05	0.30	0.6	0.6
3	0.35	0.30	0.55	0.8
4	0.6	0.63	0.85	0.90

Wir wenden nun das Homogenitätsprädikat $H_1^{dyn,0,05}$ an, das besagt, dass eine Region homogen ist, wenn ein r existiert, so dass jede Zelle in der Region ein bw-level v hat, so dass $|v - r| \leq 0.05$

Schaubild 4.1:

4	0.6	0.63	0.85	0.90
3	0.35	0.30	0.55	0.8
2	0.05	0.30	0.6	0.6
1	0.1	0.25	0.5	0.5
	1	2	3	4

Man kann leicht sehen, dass es folgende 5 Regionen gibt, die eine gültige Segmentierung des obigen Images bilden.

$R_1 = \{(1,1),(1,2)\}$
$R_2 = \{(1,3),(2,1),(2,2),(2,3)\}$
$R_3 = \{(3,1),(3,2),(3,3),(4,1),(4,2)\}$
$R_4 = \{(3,4),(4,3),(4,4)\}$
$R_5 = \{(1,4),(2,4)\}$

Man kann sich leicht vorstellen wie in etwa ein Algorithmus ablaufen müsste, der ein Bild in Bezug auf ein Homogenitätsprädikat segmentiert:

1. Aufteilung:
   Der Algorithmus nimmt als Input das gesamte Bild. Wenn es homogen ist, ist der Algorithmus fertig. Ist dies aber nicht der Fall, so teilt der Algorithmus das Bild in 2 Teile und führt rekursiv diesen Prozess fort, bis er eine Menge von Regionen $R_1,...,R_n$ gefunden hat, die homogen sind und die ersten drei Bedingungen aus der Definition von Homogenitätsprädikate erfüllen.

2. Zusammenfassen:
   Der Algorithmus überprüft, welche R_is zusammengefasst werden können. Am Ende dieses Schrittes, erhält man eine zulässige Segmentierung $R'_1,..., R'_k$ des Bildes, wobei $k \leq n$ ist und die R'_i eine Vereinigung von R_js sind.

5. Ähnlichkeitsbasiertes Suchen:

Bisher haben wir gesehen, wie man Techniken zur Bildkomprimierung verwendet, wie zum Beispiel DFT oder DCT. Wir haben außerdem gesehen, wie man ein Bild segmentieren kann. Aber eine wichtige Frage ist bisher noch nicht geklärt worden, nämlich wie man überprüft, ob ein Segment eines Bildes ähnlich zu einem anderen Bild ist.

Schaubild 5.0:

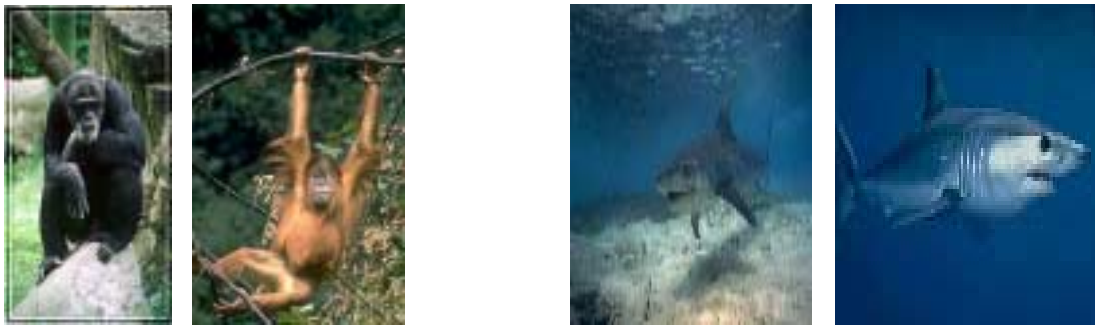


Bild a Bild b Bild c Bild d

Wenn man nun die Bilder von Schaubild 5.0 betrachtet, so kann man zu dem Schluss kommen, dass die Bilder a und b ähnlich sind, da sie beide Affen zeigen. Doch für einen Experten für Affen, sind die beiden Bilder absolut verschieden. Genauso ist es bei den Bildern c und d. Für einen Laien sind beide Bilder ähnlich, da sie Haie zeigen, aber für jemanden, der sich mit Haien beschäftigt, sind es zwei total verschiedene Bilder. Die Schwierigkeit liegt also in der Definition des Begriffes „*Ähnlich*“ und seiner Verwendung bei der Suche.

Es gibt zwei große Näherungsverfahren zum ähnlichkeitsbasiertem Suchen:

1. ***Metrische Näherung****:*
Bei dieser Näherung wird angenommen, dass es eine Distanzmetrik d gibt, die es ermöglicht zwei Bilder miteinander zu vergleichen. Um so kürzer die Distanz zwischen zwei Bildern ist, um so ähnlicher sind sie sich. Bei dieser Art der Näherung würde eine Suche dann folgendermaßen aussehen:
„Gegeben sei ein Image I, finde den nächsten Nachbar von I im Bildarchiv.“

Die metrische Näherung ist die am meisten verbreitete Näherung in der Datenbankwelt.
Im Detail funktioniert das Verfahren wie folgt:

Gegeben sei eine Menge von Objekten obj, wobei ein Objekt entweder ein Bild oder ein Segment eines Bildes sein kann, mit den Pixeleigenschaften $p_1,...,p_n$, so dass jedes Objekt o als eine Menge S(o) bestehend aus (n+2)-Tupel der Form (xcoord, ycoord, $v_1,....,v_n$) beschrieben werden kann, wobei v_i die Eigenschaft p_i verknüpft mit der Pixelkoordinate (x,y) beschreibt. Das Objekt ist in Rechtecke aufgeteilt , so dass S(o) aus w x h (n+2)-Tupeln besteht, wobei h die Höhe und w die Breite eines Rechtecks beschreibt.

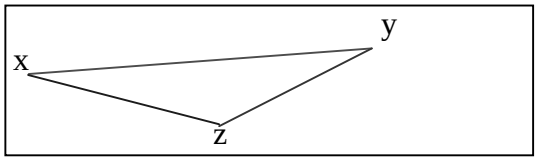
Um die Ähnlichkeit zwischen zwei Objekten zu bestimmen, verwendet man eine Distanzfunktion d.

5.0 Definition: Distanzfunktion

Eine Funktion mit einer Menge X auf dem Intervall [0,1] wird als Distanzfunktion bezeichnet, wenn folgende Axiome für alle $x, y, z \in X$ gelten:

1. $d(x, y) = d(y, x)$ (Symmetrie)
2. $d(x, y) \leq d(x, z) + d(z, y)$ (Transitivität)
3. $d(x, x) = 0$ (Reflexivität)

Schaubild 5.1:



Dieses einfach Schaubild soll noch mal die Definition einer Distanzfunktion veranschaulichen.

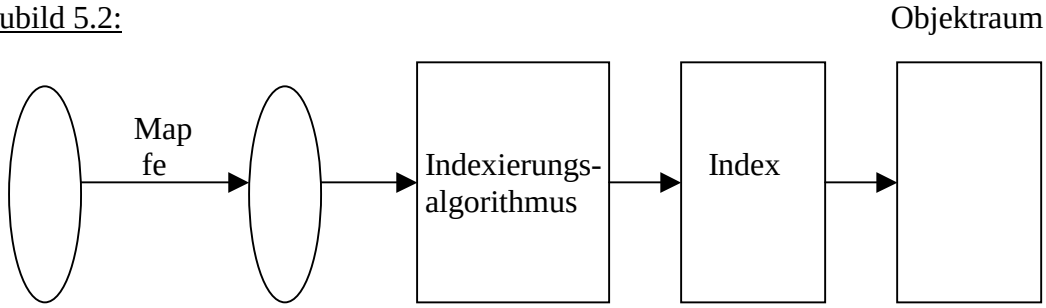
Zu 1) Punkt 1 sagt nichts anderes aus, als dass es egal ist, ob man die Distanz von x nach y oder von y nach x berechnet.

Zu 2) Punkt 2 besagt, dass die Distanz von x nach y die kürzeste Distanz ist und es keinen anderen Weg zu y gibt, der kürzer ist.

Zu 3) Schließlich sagt Punkt 3 aus, dass die Distanz zu sich selber 0 sein muss.

Da ein Objekt durch eine Menge von Attributen bestimmt werden kann, kann die Berechnung von d_{obj} sehr aufwendig werden.
Um dieses Problem zu umgehen versucht man eine Funktion fe zu benutzen, die ein Objekt nur noch auf einen einzelnen Punkt in einem s-dimensionalen Raum abbildet, wobei s normalerweise sehr klein ist (n+2).
Vorher wurde ein Objekt o auf eine Menge von Punkten in einem (n+2)-dimensionalen Raum abgebildet.

Schaubild 5.2:



Das Schaubild stellt dar, welche Schritte ausgeführt werden.

Bei der Abbildung der Objekte in den s-dimensionalen Raum, muss zusätzlich gewährleistet bleiben, dass die ursprüngliche Distanz $d(o_1,o_2) \leq d(o_1,o_3)$, wobei d eine Metrik auf dem ursprünglichen (n + 2)-dimensionalen Raum ist, erhalten bleibt. Die neue Distanz sähe dann folgendermaßen aus: $d'(fe(o_1), fe(o_2)) \leq d'(fe(o_1), fe(o_3))$

Startet man nun eine Anfrage mit einem Objekt $o_{Anfrage}$, so würde man das Bild (den Index i) zurückliefert bekommen, welches die kleinste Distanz $d'(o_{Anfrage}, x_i)$ zu dem Objekt $o_{Anfrage}$ hat, wobei x_i alle Objekte in dem s-dimensionalen Raum sind.

2. ***Transformationsnäherung***:

Dieser Ansatz ist allgemeiner als der metrische Ansatz und er geht von der Grundlage aus, dass der Unterschied zwischen 2 Objekten o_1 und o_2 proportional zu den (minimalen) Kosten für eine Transformation von o_1 in o_2 oder andersherum ist.
Bei diesem Ansatz gibt es eine Anzahl von Transformationsoperatoren $to_1,...,to_n$. Im Falle von Bildern beinhalten diese Operationen zum Beispiel Translation, Rotation und Skalierung. Erweiterungs- und Verkleinerungsoperatoren sollten außerdem vorhanden sein, um die Struktur eines Objektes verändern zu können. Dies sollen natürlich nur ein paar Beispiele von Operatoren sein und es soll nicht bedeuten, dass dies die einzigen zulässigen Operatoren sind. Jedem Operator werden bestimmte Kosten zugeteilt. Um so höher die Kosten für einen Operator sind, um so schwieriger ist es ihn anzuwenden.

5.1 Definition: Transformation

Eine Transformation von einem Objekt o in ein Objekt o' ist eine Folge von Transformationsoperatoren $to_1,...,to_r$ und eine Folge von Objekten $o_1,...,o_{r,}$, so dass:

1. $o_1(o) = o_1$,
2. $to_i(o_{i-1}) = o_i$ und
3. $to(o_r) = o'$.

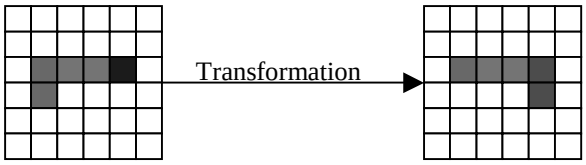
Die Kosten für die obige Transformationssequenz TS lautet:

$$\text{cost (TS)} = \sum_{i=1}^{r} \text{cost}(to_i)$$

Man muss beachten, dass es eventuell keine oder sogar eine Menge von Transformationssequenzen gibt, die Objekt o in ein Objekt o' transformieren. TSeq(o, o') ist die Menge aller Transformationssequenzen, die o in o' transformieren. Die Unähnlichkeit zwischen o und o' wird durch dis(o, o') beschrieben, mit Bezug auf Transformationsoperatoren TR und einer Menge von Kostenfunktionen CF

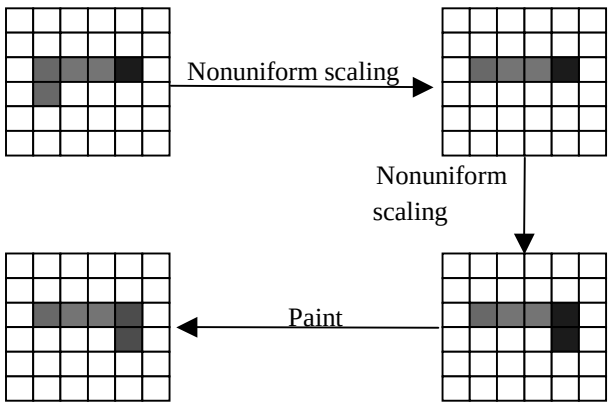
$$dis(o, o') = \min\{ cost(TS) \mid TS \in Tseq(o, o') \cup Tseq(o', o) \}$$

Schaubild 5.3:



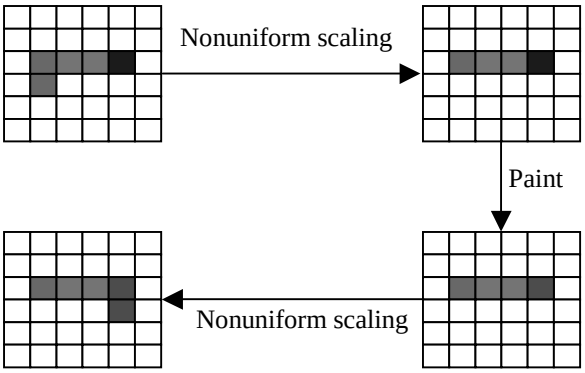
Ein Objekt o_1 soll in ein Objekt o_2 überführt werde.

Schaubild 5.4:



Transformationssequenz TS_1, die o_1 in o_2 transformiert

Schaubild 5.5:



Eine zweite Transformationssequenz TS_2,
die ebenfalls o_1 in o_2 transformiert

Nun da wir zwei Möglichkeiten der ähnlichkeitsbasierten Suche gesehen haben, soll nun auf den Bereich eingegangen werden, den wir bisher noch außen vor gelassen haben, nämlich das Speichern von Bildern, so dass Image Retrieval möglichst gut unterstützt wird.

6. Image Database:

Es gibt mehrere Möglichkeiten um eine Image Database darzustellen:

1. Die Darstellung einer IDB in Form von Relationen
2. Die Nutzung von räumlichen Strukturen wie zum Beispiel R-Trees
3. Die Darstellung einer IDB durch image transformations (Komprimierung)

Wir werden hier beispielhaft genauer auf den ersten Punkt eingehen, die Darstellung einer Image DB durch Relationen:

(Siehe *Definition 2.4*: *Image Database (IDB))*

IDB = (GI, Prop, Rec)

Die folgenden Schritte werden durchgeführt:

1. Generierung einer Relation *Image* mit folgendem Schema:
   (Image, ObjID, XLB, XUB, YLB, YUB)
   Wobei Image der Name eine Imagedatei ist. Die ObjId steht für ein Objekt, das in dem Image enthalten ist und XLB, XUB, YLB, YUB beschreiben das betreffende Rechteck R. Wenn R in Rec(I) enthalten ist, existiert ein Tupel
   (I, newId, XLB, XUB, YLB, YUB) in der Relation images.
   Das Schaubild 6.0 zeigt die Relation images, die mit einer Gesichtsdatenbank verknüpft ist (Schaubild 6.1).

Schaubild 6.0: (Image Relation)

Image	ObjId	XLB	XUB	YLB	YUB
pic1.gif	o_1	10	60	5	50
pic1.gif	o_2	80	120	20	55
pic2.gif	o_3	20	65	20	75
pic3.gif	o_4	25	75	10	60
pic4.gif	o_5	20	60	30	80

Image	ObjId	XLB	XUB	YLB	YUB
pic5.gif	o_6	0	40	15	50
pic6.gif	o_7	20	75	15	80
pic6.gif	o_8	20	70	130	185
pic7.gif	o_9	15	70	15	75

Schaubild 6.1:

Bild 1

Bild 2

Bild 3

Bild 4

Bild 5

Bild 6

Bild 7

2. Für jede Eigenschaft p ∈ Prop erstellt man eine Relation R_p mit dem Schema:

   (Image, XLB, XUB, YLB, YUB, Value)

   Wieder ist Image der Name einer Bilddatei, wobei XLB, XUB, YLB, YUB wieder eine Zelle des Images beschreiben und Value gibt den Wert der Eigenschaft p an.

   Meistens gibt es folgende drei Eigenschaften eines Bildes:

   1. Pixel-level-Eigenschaften:
      Diese Eigenschaften können zum Beispiel
      der Rot/Grün/Blau Anteil eines Pixels sein.

   2. Object/region level-Eigenschaften:
      Einige Objekte in einem Bild können eigene Eigenschaften haben, wie zum Beispiel das Objekt o_1 des pic1.gif den Namen und das Alter als Eigenschaft haben kann, mit den zugehörigen Werten „Sanders, Bjoern“ und 23.

   3. Image-level-Eigenschaften:
      Dies können Eigenschaften sein, die das Bild als ganzes betreffen, wie zum Beispiel das Datum, der Ort oder die Person von der es gemacht wurde,

Zwar müsste eigentlich theoretisch für jede Eigenschaft p in Prop eine eigene Relation angelegt werden, aber in der Praxis ist es nicht praktikabel jeden Pixelwert als Tupel in einer Relation zu repräsentieren. Statt dessen werden nur die Eigenschaften 2. und 3. explizit in einer Relation angegeben.

Kommen wir nun zu der Frage, wie überhaupt eine Anfrage auf solch einer Image DB auszusehen hat ?
Man könnte sich folgende Anfrage vorstellen:
„Finde alle Bilder, auf denen Bjoern Sanders ist“

In SQL würde diese Anfrage etwa so lauten:

```
Select  image
From    images I, name N
Where I.ObjId = n.ObjId AND
        N.name = “Bjoern Sanders”
```

Würde diese Anfrage auf dem nachfolgenden Schema angewendet werden, so würde uns die Anfrage durch die eindeutige Zuordnung die richtigen Bilder zurückliefern.

Schaubild 6.2:

ObjId	Name
o_1	Bjoern Sanders
o_2	Dirk Niebuhr
o_3	Dirk Niebuhr
o_4	Bjoern Sanders
o_5	Judith Werner
o_6	Sabine Queckbörner
o_7	Judith Werner
o_8	Sybille Hardt
o_9	Sybille Hardt

Ausgangssituation

Anfrage →

ObjId	Name
o_1	Bjoern Sanders
o_2	Bjoern Sanders

Ergebnis

Wenn man nun jedoch berücksichtigt, dass man für die Zuordnung der Namen zu den Bildern ein Bildbearbeitungsprogramm verwendet, so erhält man Wahrscheinlichkeitswerte zu den jeweiligen Zuordnungen.

Schaubild 6.3:

ObjId	Name	Prob
o_1	Bjoern Sanders	0.8
o_1	Jens Wrede	0.2
o_2	Dirk Niebuhr	0.75
o_2	Judith	0.25
o_3	Dirk Niebuhr	1
o_4	Bjoern Sanders	1
o_5	Judith Werner	1
o_6	Sabine Queckbörner	1
o_7	Judith Werner	0.7
o_7	Sabine Queckbörner	0.3
o_8	Sybille Hardt	0,65
o_8	Sabine Queckbörner	0,30
o_8	Bjoern Sanders	0,05
o_9	Sybille Hardt	1

Was diese Schaubild ausdrücken soll, ist, dass man nun in dieser Relation noch zusätzlich ein Attribut mit der Wahrscheinlichkeit hat, so dass zum Beispiel eine Wahrscheinlichkeit von 0.8 besteht, dass Bjoern Sanders das richtige Namensattribut von o_1 ist.

Doch durch diese neue Zuordnung, werden die Anfragen noch komplexer. Sucht man nämlich nun zum Beispiel nach den Bildern, auf denen Bjoern Sanders mit Judith Werner zu sehen ist, erhält man zwei mögliche Kandidaten, nämlich Bild 1 und Bild 6. Dies kommt zu stande, weil Bild 1 zwei Objekte hat, bei denen die Wahrscheinlichkeit bei 80% liegt, dass Bjoern Sanders das Objekt o_1 ist und eine 25% Wahrscheinlichkeit besteht, dass Judith Werner o_2 ist.
Genauso ist es mit Bild 6, bei dem mit 70% Judith Werner o_7 ist und Bjoern Sanders mit 5 % o_8 ist.
Nun stellt sich doch die Frage, wie hoch die Wahrscheinlichkeit ist, dass Bild 1 oder das Bild 6 beide enthält.
Man könnte nun annehmen, dass die Wahrscheinlichkeit für Bild 1 (0.8 x 0,.5) = 0.2 und die Wahrscheinlichkeit für Bild 6 (0.7 x 0.05) = 0.035 beträgt.

Leider ist dies nicht die richtige Antwort. Um zu verdeutlichen, wo die Probleme liegen, nehmen wir ein zusätzliches fiktives Bild 8 an, dass zwei Objekten o_{10} und o_{11} enthält. Zusätzlich wird die obige Tabelle um folgende Einträge erweitert:

Schaubild 6.4:

ObjId	Name	Prob
o_{10}	Judith Werner	0.5
o_{10}	Sabine Queckbörner	0.5
o_{11}	Bjoern Sanders	0.8
o_{11}	Jens Wrede	0.2

Es gibt für das Bild 8 diese vier Möglichkeiten:

Fall 1: o_{10} ist Judith Werner und o_{11} ist Bjoern Sanders
Fall 2: o_{10} ist Judith Werner und o_{11} ist nicht Bjoern Sanders
Fall 3: o_{10} ist nicht Judith Werner aber o_{11} ist Bjoern Sanders
Fall 4: o_{10} ist nicht Judith Werner und o_{11} ist nicht Bjoern Sanders

Nun müssen wir für diese vier Möglichkeiten die Wahrscheinlichkeiten bestimmen, denn die Wahrscheinlichkeit für den ersten Fall ist schließlich die Wahrscheinlichkeit, dass Bild 8 eine Antwort auf unsere Anfrage ist.

Nehmen wir an, dass p_i die Wahrscheinlichkeit für den Fall i ($1 \leq i \leq 4$) ist:

$p_1 + p_2 = 0.5$
$p_3 + p_4 = 0.5$
$p_1 + p_3 = 0.8$
$p_2 + p_4 = 0.2$

$p_1 + p_2 + p_3 + p_4 = 1$

Die Wahrscheinlichkeit für die erste Gleichung ergibt sich aus dem Wissen, dass o_{10} Judith Werner ist (in Bezug auf den ersten und zweiten Fall) und wir wissen aus der Tabelle, dass die Wahrscheinlichkeit 0.5 beträgt.
Die Wahrscheinlichkeit für die zweite Gleichung folgt aus der Tatsache, dass o_{10} nicht Judith Werner ist (in Bezug auf den dritten und vierten Fall) und aus der Tabelle wissen wir, dass die Wahrscheinlichkeit, dass o_{10} nicht Judith Werner ist, 0.5 beträgt.

Die Wahrscheinlichkeit für die dritte Gleichung ergibt sich auf ähnliche Weise. Sie ergibt sich aus der Tatsache, dass o_{11} Bjoern Sanders ist (in Bezug auf die Fälle eins und drei) und wir wissen aus der Tabelle, dass die Wahrscheinlichkeit 0.8 beträgt.
Schließlich ergibt sie die Wahrscheinlichkeit für die letzte Gleichung (in Bezug auf die Fälle zwei und vier) dadurch, dass die Wahrscheinlichkeit 0.2 beträgt, dass o_{11} nicht Bjoern Sanders ist.

Um jedoch zu klären, wie groß die Wahrscheinlichkeit ist, dass Bjoern Sanders und Judith Werner auf dem Bild acht gemeinsam zu sehen sind, müssen wir das obige Gleichungssystem nach p_1 hin auflösen. Man erhält als Ergebnis nicht einen festen Wert, sondern ein Intervall von Wahrscheinlichkeiten. Hier läge p_1 zum Beispiel in dem Intervall [0.3, 0.5]. Wenn man die Wahrscheinlichkeit, dass Judith Werner o_{10} und Bjoern Sanders o_{11} ist, miteinander multipliziert, so erhält man eine Wahrscheinlichkeit von 0.4. Zwar liegt das innerhalb des Intervalls, aber kann nicht alle vier Fälle wiedergeben.
Das Problem ist im Moment, dass wir nur einen Wahrscheinlichkeitswert in unserer Relation haben, doch durch die Anfrage erhalten wir ein Intervall von Wahrscheinlichkeiten.
Daher müssen wir die bisherige Wahrscheinlichkeit durch Intervalle ersetzen, um dieses Problem zu umgehen. Wir lassen einfach ein Bilderkennungsprogramm die Wahrscheinlichkeiten p_i ermitteln und beziehen dann einen Ungenauigkeitswert ε mit ein. So erhalten wir ein Intervall $[p_i - \varepsilon, p_i + \varepsilon]$. Kehren wir nun zu unserem Beispiel zurück und beziehen einen Ungenauigkeitswert von 3% ein, wobei die Wahrscheinlichkeiten aus $[\max(0, p_i - 0.03), \min(1, p_i + 0.03)]$:

Schaubild 6.5:

ObjId	Name	Prob(Lower)	Prop(Upper)
o_1	Bjoern Sanders	0.77	0.83
o_1	Jens Wrede	0.17	0.23
o_2	Dirk Niebuhr	0.72	0.78
o_2	Judith	0.22	0.28
o_3	Dirk Niebuhr	0.97	1
o_4	Bjoern Sanders	0.97	1
o_5	Judith Werner	0.97	1
o_6	Sabine Queckbörner	0.97	1
o_7	Judith Werner	0.67	0.73
o_7	Sabine Queckbörner	0.27	0.33
o_8	Sybille Hardt	0.62	0.68
o_8	Sabine Queckbörner	0.27	0.33
o_8	Bjoern Sanders	0.02	0.08
o_9	Sybille Hardt	0.97	1
o_{10}	Judith Werner	0.47	0.53
o_{10}	Sabine Queckbörner	0.47	0.53
o_{11}	Bjoern Sanders	0.77	0.83
o_{11}	Jens Wrede	0.17	0.23

Kehren wir nun zu unserer Anfrage zurück und betrachten die Wahrscheinlichkeiten, dass sowohl Bjoern Sanders als auch Judith Werner auf dem Bild acht zu sehen sind. In den folgenden Fällen wurden wieder die selben Begründungen wie oben verwendet:
$0.47 \leq p_1 + p_2 \leq 0.53$
$0.47 \leq p_3 + p_4 \leq 0.53$
$0.77 \leq p_3 + p_4 \leq 0.83$
$0.17 \leq p_3 + p_4 \leq 0.23$

$p_1 + p_2 + p_3 + p_4 = 1$

Die Wahrscheinlichkeiten der Ungleichungen ergeben sich wie oben, nur dass dieses Mal nicht ein Wahrscheinlichkeitswert gegeben ist, sondern ein Intervall.
Lösen wir das obige System, so erhalten wir für p_1 als minimal Wert 0.24 und als maximal Wert 0.53.

Welche Auswirkungen hat das oben beschriebene Problem auf die Datenbankwelt
und wie fließen diese Lösungen in die image Datenbank ein ?

Wir schließen hier mal unser Beispiel und suchen einen allgemeineren Ansatz:

Wir definieren eine Wahrscheinlichkeitsrelation über dem Schema (A_1,..., A_n), so dass sich folgende Relation ergibt (A_1,..., A_n, LB, UB), wobei LB und UB innerhalb dem Intervall [0, 1].
Die Namensrelation ist Wahrscheinlichkeitsrelation die drei Attribute hat:
(ImageId, ObjId, Name)
Die Namensrelation besitzt einige Einschränkungen:
$(\forall t_1, t_2)\ t_1.\text{ObjId} = t_2.\text{ObjId} \rightarrow t_1.\text{ImageId} = t_2.\text{ImageId}$

Dies bedeutet, dass einer ObjectId nur genau ein Bild zugeordnet werden kann.
Die folgende Bedingung sagt, dass LB immer kleiner gleich UB sein muss:

$(\forall t)\ t.\text{LB} \leq t.\text{UB}$

Nehmen wir nun an, dass eine Image Datenbank aus einer Relation *name*, wie oben beschrieben, besteht und weiteren „normalen“ Relationen R_1,...,R_n die Bildeigenschaften darstellen.
Eine Anfrage der Form „Finde alle Bilder in der Datenbank, die die Objekt mit den Namen s_1,...,s_n enthalten“
könnte also in SQL lauten:

SELECT ImageId
FROM *name* T_1,…,T_n
WHERE T_1.Name = s_1 AND … AND T_n.Name = s_n AND
T_1.ImageId = T_2.ImageId AND … AND T_1.ImageId = T_n.ImageId

Als Ergebnis dieser Anfrage erhält man eine Tabelle mit drei Spalten, die ImageID, LB und UB
(im, l, u) ist ein Ergebnis, falls für jedes $1 \leq j \leq n$ ein Tupel $T_j \in$ *name* existiert, so dass

1. t.ImageID = im
2. t.LB = l_i und t.UB = u_i und
3. $[l, u] = [l_1, u_1] \otimes [l_2, u_2] \otimes ... \otimes [l_n, u_n]$
   wobei
   $[x, y] \otimes [x', y'] = [\max(0, x + x' - 1), \min(y, y')]$

Man hat bewiesen, dass der $\otimes$ Operator genau die selben Ergebnisse liefert, wie die Lösung des linearen Gleichungssystems im Abschnitt zuvor. Dadurch können Image Datenbanken effektiv implementiert werden, indem sie den einfachen $\otimes$, wie oben beschrieben, verwenden.

In SQL gibt es einen dafür extra den speziellen Operator HAS.
Eine Anfrage der Form „Finde alle Bilder in der Datenbank, die die Objekt mit den Namen $s_1,...,s_n$ enthalten“, müsste also in SQL folgendermaßen aussehen:

name HAS $s_1,...,s_n$

In unserem Fall müsste die Anfrage, die unser „Enthalten sein“-Problem löst, dann folgendermaßen in SQL lauten:

SELECT ImageId
FROM *name* $T_1,\ldots,T_n$
WHERE T_1.Name HAS s_1 AND … AND T_n.Name HAS s_n AND
T_1.ImageId = T_2.ImageId AND … AND T_1.ImageId = T_n.ImageId

7. Aussicht:

Wir haben die verschiedenen Aspekte, wie zum Beispiel Bildkomprimierung, Merkmalerkennung, ähnlichkeitsbasiertes Suchen und die Darstellung von Bildern in einer IDB, des Image Retrieval kennengelernt. Image Retrieval wird in Zukunft immer wichtiger, denn in der Wirtschaft und in der Forschung werden die Einsatzmöglichkeiten und die Nutzung in diesem Bereich nach und nach erkannt und ausgebaut. Daher ist die Entwicklung im Bereich Image Retrieval auch noch lange nicht abgeschlossen, sondern erst am Anfang. Durch das Eindringen in immer neue Bereiche, wird die Suche nach effizienteren und genaueren Algorithmen und Methoden vorangetrieben. Image Retrieval wird in Zukunft ein wichtiger Bereich der Informatik sein.

8. Schaubildverzeichnis:

1 Schaubild 2.0..... 5
2 Schaubild 3.0..... 7
3 Schaubild 4.0..... 8
4 Schaubild 5.0..... 12
5 Schaubild 5.1..... 13
6 Schaubild 5.2..... 13
7 Schaubild 5.3..... 14
8 Schaubild 5.4..... 14
9 Schaubild 5.5..... 15
10 Schaubild 6.0..... 15
11 Schaubild 6.1..... 16
12 Schaubild 6.2..... 17
13 Schaubild 6.3..... 17
14 Schaubild 6.4..... 18

9. Literaturverzeichnis:

Quelltexte: Suchmodellbasiertes Content-based Image Retrieval von Horst Eidenberger
Quelltexte: Content-based Image Retrieval von John Eakins & Margaret Graham
Quelltexte: Multimedia retrieval von Jilali Raki & David Squire, Zoran Pecenovic
Quelltexte: Fast algorithms for image retrieval von Zoran Pecenovic
Quelltexte: Content-Based Image Retrieval Systems von Remco C. Veltkamp, Mirela Tanase
Quelltexte: Integrated Region-Based Image Retrieval von James Z. Wang
Quelltexte: Feature Extraction and Selection for Image Retrieval von Xiang Sean Zhou, Ira Cohen, Qi Tian, Thomas S. Huang
Quelltexte: Apers, Blanken, Houtsma (Eds), Multimedia Databases in Perspective, Springer, 1997