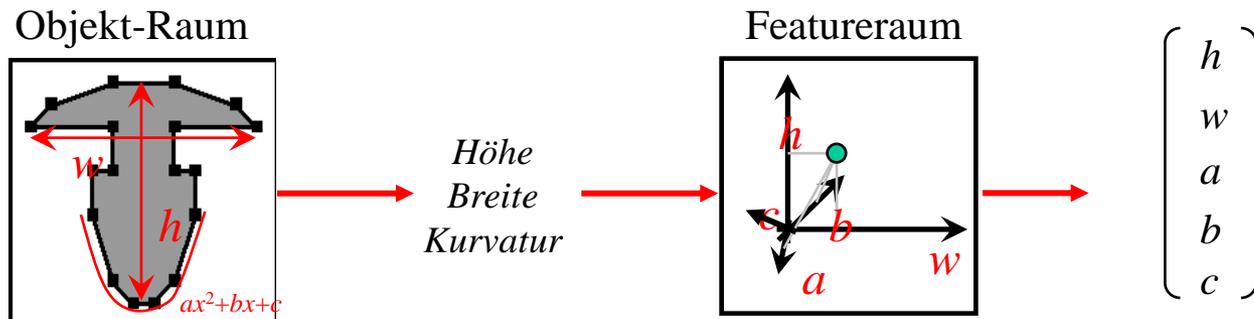
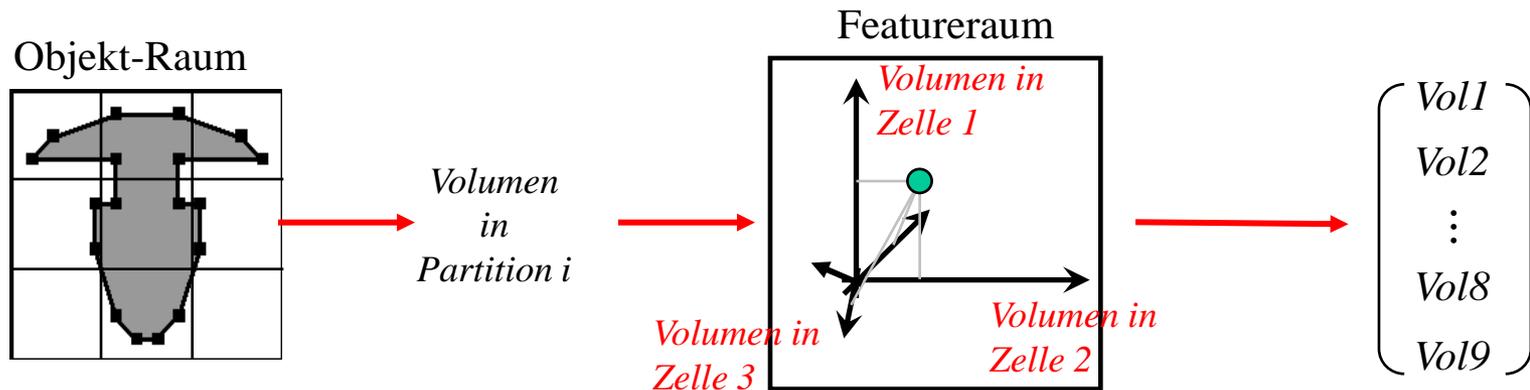


## 3.2 Ähnlichkeitsmodelle für räumliche Objekte

- Feature Transformation für räumliche Objekte
  - Ziel: „gute“ Beschreibung der realen Objekte als Featurevektoren (metrisch oder besser: Euklidisch)
    - Ähnlichkeit im Objektraum  $\approx$  Ähnlichkeit im Featureraum
    - D.h. Merkmale sollten „sinnvoll“ / „aussagekräftig“ sein
  - Möglichkeit 1:
    - Extrahiere Merkmale für das gesamte Objekt

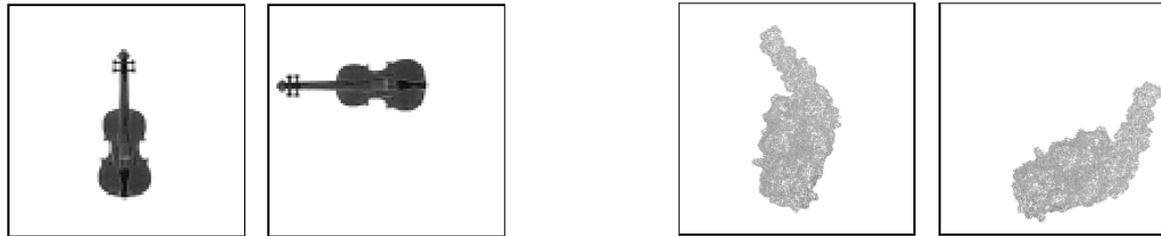


- Möglichkeit 2:
  - Partitioniere Objektraum
    - » **Objekt-spezifische Partitionierung:** das Objekt wird zerlegt, unabhängig davon, wie es im Datenraum liegt
    - » **Datenraum-spezifische Partitionierung:** der Datenraum wird zerlegt, unabhängig davon, wie das Objekt darin liegt
  - Extrahiere Merkmale aus einzelnen Partitionen
    - » Z.B. Volumen des Objekts in jeder Partition



## – Invarianzen

- Gleichheit (oder Ähnlichkeit) von Formen unabhängig von Lage und Orientierung im Raum
- Beispiele gleicher Formen im 2D und im 3D:

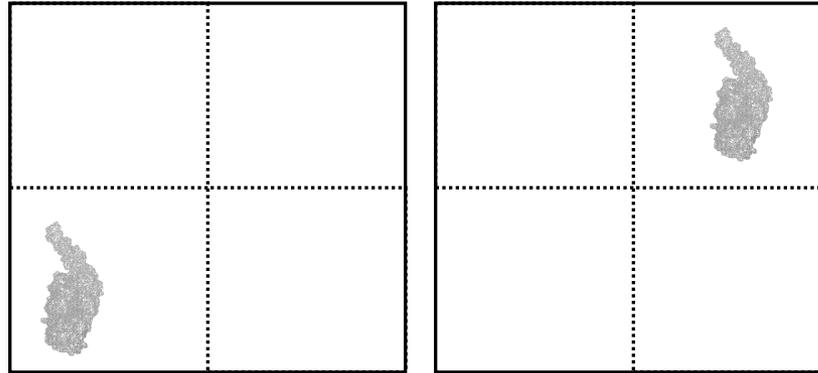


- Meist erwünscht (je nach Anwendung):
  - Kanonische Darstellung, d.h. ohne Lage- und Orientierungsinformation
  - Verallgemeinerung auf andere Objekteigenschaften
- Formal
  - Sei  $F: \mathbf{OBJ} \rightarrow \mathbf{DOM}$  eine Featuretransformation und  $F(O) \in \mathbf{DOM}$  die Featurerepräsentation von  $O \in \mathbf{OBJ}$  im Featureraum
  - Sei  $K$  eine Klasse von Transformationen auf  $\mathbf{OBJ}$
  - $F$  ist invariant gegenüber  $K$ , wenn für alle  $T \in K$  und  $O \in \mathbf{OBJ}$  gilt:

$$F(T(O)) = F(O)$$

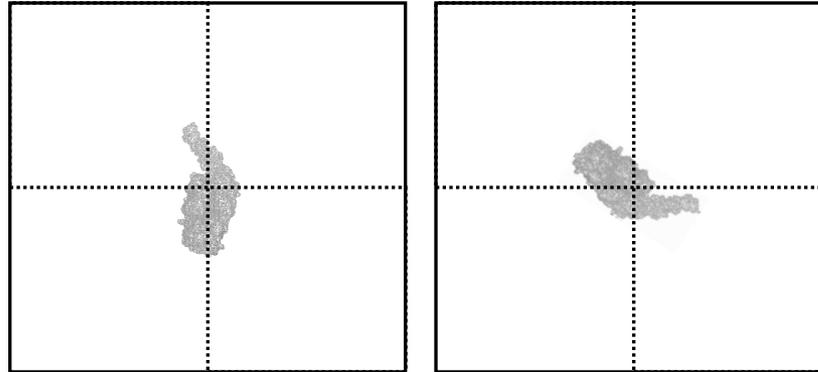
## – Die wichtigsten Invarianzen

- Translation

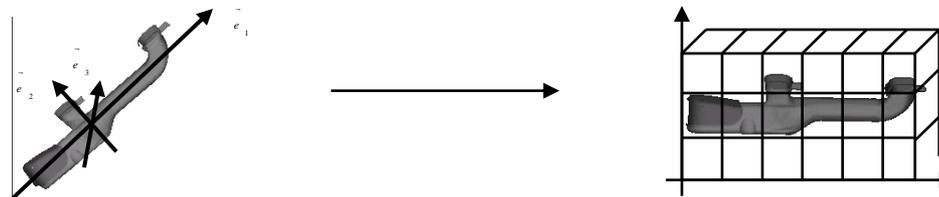


- Falls Ähnlichkeitsmodelle NICHT translationsinvariant
  - » Verschiebung des Schwerpunkts eines Objektes in den Ursprung bevor die Featuretransformation berechnet wird

- Rotation

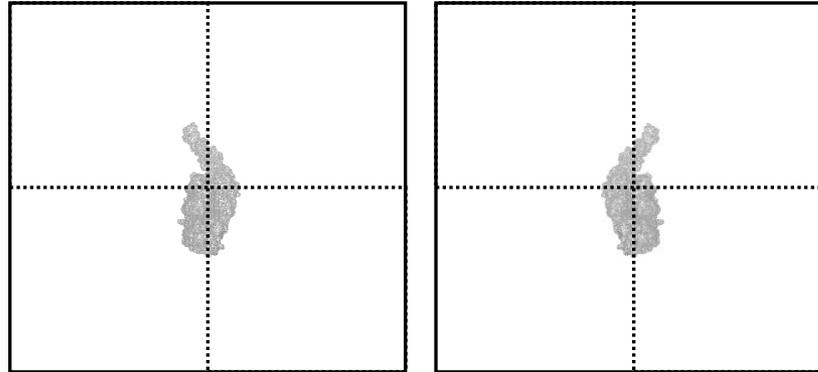


- In manchen Anwendungen reicht Invarianz bzgl. gewisser Rotationswinkel
  - » CAD: Konstrukteure speichern Objekte meist in „vernünftiger“ Lage; dann reicht 90-Grad-Rotationsinvarianz
- Falls Ähnlichkeitsmodelle NICHT rotationsinvariant
  - » **Hauptachsentransformation**: Drehung der Objekte, so dass die Hauptachsen auf den Koordinatenachsen liegen.



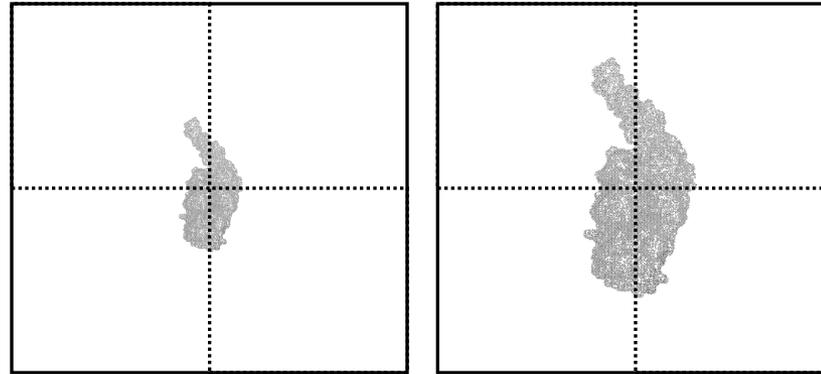
- » **Implizite Permutation**: Berechne alle möglichen Drehungen der Objekte (z.B. alle 90-Grad Drehungen) vorab oder zur Laufzeit

- Spiegelung (Reflexion)

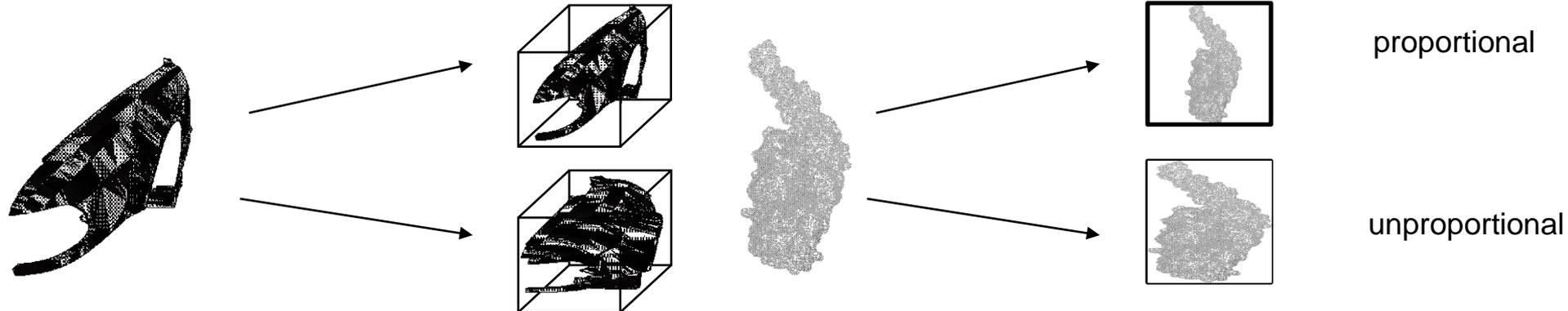


- Falls Ähnlichkeitsmodelle NICHT reflexionsinvariant
  - » **Implizite Permutation:** Berechne alle möglichen Spiegelungen der Objekte (z.B. bzgl. aller räumlichen Achsen) vorab oder zur Laufzeit

- Skalierung



- Die meisten Ähnlichkeitsmodelle sind NICHT skalierungsinvariant
- Skalierungsinvarianz wird meist durch Größen-Normierung des Objektraums
  - » Proportional: globale Skalierung
  - » Unproportional: separat entlang jeder räumlichen Achse



- Die wichtigsten geometrischen Transformationen
  - Frage: Wie können Objekte transformiert werden?
  - Lösung (siehe auch: Graphische Datenverarbeitung):
    - Darstellung der einzelnen Transformationen Translation, Spiegelung, Rotation, Skalierung als Abbildung
    - Wende die Abbildung auf alle Teile eines räumlichen Objekts an (Pixel, Voxel, (Oberflächen-)Punkte, etc.)
  - Formal:
    - Sei  $T \in \{\text{Translation, Reflexion, Skalierung, Rotation}\}$
    - Sei  $\text{obj} \in \text{OBJ}$  gegeben als Menge von Punkten (z.B. Mittelpunkte der Voxel oder Oberflächensegmente, etc.)  
d.h.  $\text{obj} = \{x \mid x \text{ ist } k\text{-dimensionaler Punkt}\}$
    - $T(\text{obj}) = \text{obj}' := \{T(x) \mid x \in \text{obj}\}$

- Basis-Transformationen im 2D

– transformiere 2D Punkt  $p = \begin{bmatrix} x \\ y \end{bmatrix}$  in  $p' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

– Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} dx \\ dy \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \end{bmatrix}$$

– Skalierung

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \end{bmatrix}$$

– Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cdot \cos \varphi - y \cdot \sin \varphi \\ x \cdot \sin \varphi + y \cdot \cos \varphi \end{bmatrix}$$

– Spiegelung (x-Achse)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix}$$

- Problem:
  - Häufig hat man eine Folge aufeinanderfolgender Transformationen
  - Anstatt jede einzelne Transformation auf jeden Punkt eines Objekts anzuwenden, fasse alle Transformation zu einer Transformationsmatrix zusammen.
  - Aber: Translation ist keine Matrixmultiplikation und lässt sich nicht zu einer einzigen Matrix zusammenfassen.
- Lösung:
  - Stelle alle Abbildungen als Matrix-Multiplikation dar
  - Dazu: 3D -Repräsentation der 2D Punkte („homogene Koordinaten“)

– Stelle  $p = \begin{bmatrix} x \\ \\ y \end{bmatrix}$  als 3D Vektor  $\hat{p} = \begin{bmatrix} X \\ Y \\ w \end{bmatrix}$  dar

- Kartesische Koordinaten Punktes  $p$  ergeben sich aus den homogenen Koordinaten:  $x = X/w$  und  $y = Y/w$

– Homogenisierung:  $\begin{bmatrix} X \\ Y \\ w \end{bmatrix} \rightarrow \begin{bmatrix} X/w \\ Y/w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

## – Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ 1 \end{bmatrix}$$

## – Skalierung

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \\ 1 \end{bmatrix}$$

## – Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot \cos \varphi - y \cdot \sin \varphi \\ x \cdot \sin \varphi + y \cdot \cos \varphi \\ 1 \end{bmatrix}$$

## – Spiegelung (x-Achse)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ y \\ 1 \end{bmatrix}$$

- Matrizen der wichtigsten Basis-Transformationen im 3D  
(homogenisiert => 4D Matrizen)

Translation

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Skalierung

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Spiegelung (x-Achse)

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

90-Grad Rotation um x-Achse

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

90-Grad Rotation um y-Achse

$$\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

90-Grad Rotation um z-Achse

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## – Übersicht

- Große Anzahl an Ähnlichkeitsmodellen für verschiedene Anwendungen und Objektrepräsentationen  
(Pixel/Voxel, Polygone, Triangulierte Oberflächen, Oberflächenpunkte, etc.)
- **Beispiele** [Iyer, Lou, Jayanti, Kalyanaraman, Ramani. Computer Aided Design, 37(5), 2005]
  - Geometrisches Hashing
  - Algebraische Moment-Invarianten
  - Iterative Closest Points (ICP)
  - Partielle Ähnlichkeitssuche mit Fourier-Transformation
  - Angular Profile, LWL-Codierung (Länge-Winkel-Länge)
  - Section Coding
  - Spherical Harmonics
  - etc.
- Im folgenden: kleine Auswahl

## 3.2.1 Formhistogramme für 2D und 3D Objekte

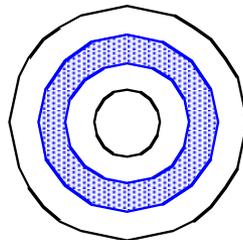
[Ankerst, Kastenmüller, Kriegel, Seidl. Proc. Int. Symp. Large Spatial Databases (SSD), 1999]

### – Anwendung:

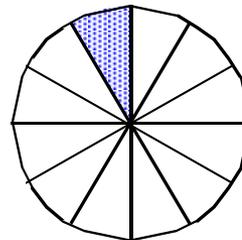
- Objekte sind Mengen von Oberflächenpunkten
- CAD-Bausteine, Moleküle, etc.

### – Grundidee: Formhistogramme

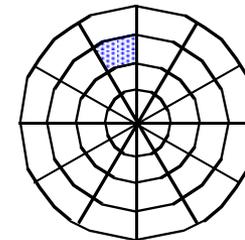
- Partitioniere den Objektraum (2D/3D)
- Bestimme Anzahl von Oberflächenpunkten des Objekts pro Zelle (normiertes Histogramm; unabhängig von Punktdichte)
- Verschiedene Raumpartitionierungen



Schalenmodell

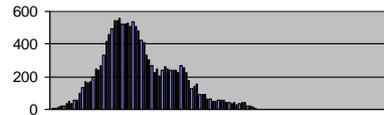


Sektorenmodell

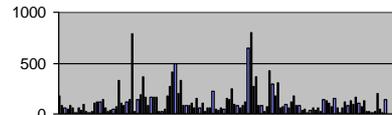


Kombiniertes Modell

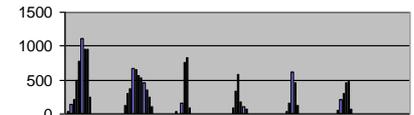
- Beispiel: Protein-Oberfläche



Schalenmodell  
(120 Schalen)



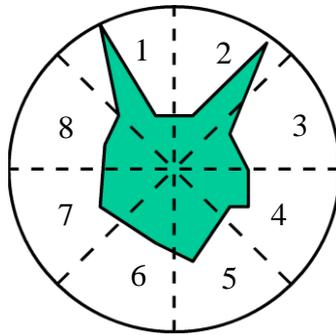
Sektorenmodell  
(122 Sektoren)



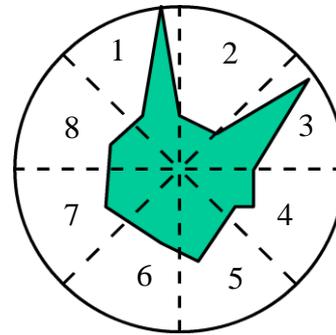
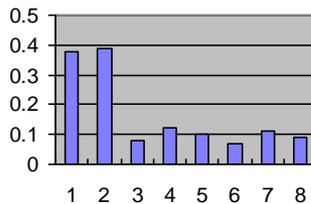
Kombiniertes Modell  
(20 Schalen, 6 Sektoren)

- Histogramm-Definition modell-spezifisch
  - **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
  - **Rotationsinvarianz** beim Schalenmodell
  - **Skalierungsinvarianz** beim Sektorenmodell

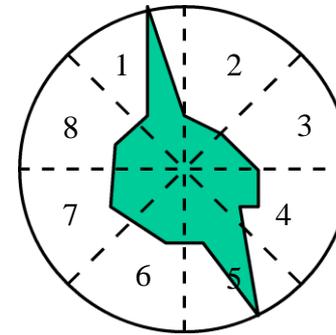
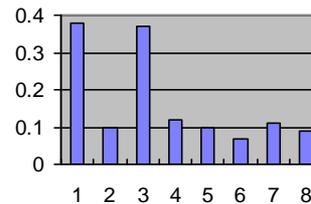
- Problem mit der euklidische Distanz auf Histogrammen



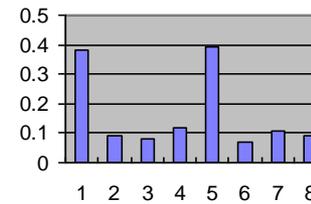
Objekt A



Objekt B



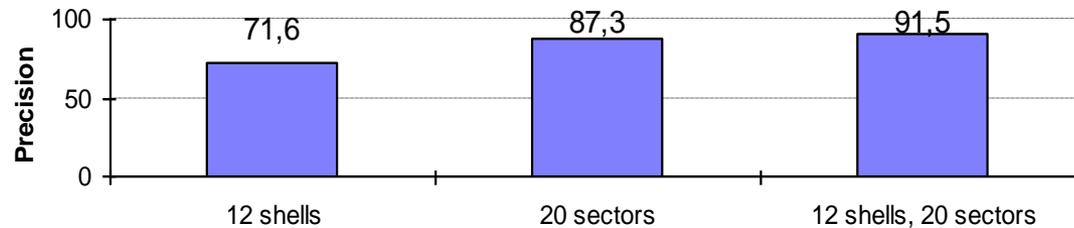
Objekt C



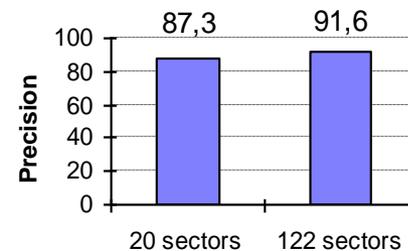
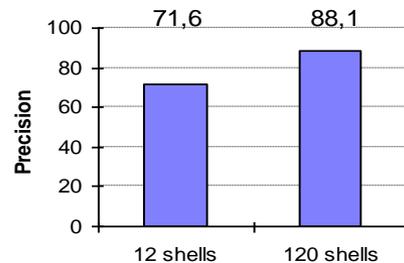
- Objekt C ist genauso ähnlich zu Objekt A wie Objekt B zu Objekt A
- Lage der Histogramm-Bins wird nicht berücksichtigt
- Lösung:
  - Quadratische Formdistanz als Distanzfunktion verwenden

$$dist(p, q) = \sqrt{(p - q) \cdot A \cdot (p - q)^T} = \sqrt{\sum_{i=1}^d \sum_{j=1}^d a_{i,j} (p_i - q_i)(p_j - q_j)}$$

- Ähnlichkeitsmatrix  $A = [a_{i,j}]$  enthält die Ähnlichkeit von Einträgen in den Zellen  $i$  und  $j$  der Raumpartitionierung
- Eintrag  $a_{i,j}$  aus Abstand  $d_{i,j}$  der Zellen  $i,j$  berechnen:  $a_{i,j} = e^{-\sigma (d_{i,j} / d_{\max})^2}$
- Verwende z.B. Euklidische Distanz als Abstand  $d_{i,j}$
- Experimentelle Untersuchung zur Wahl der Partitionierung
  - Datenbank mit Protein-Molekülen, K-NN (k=1) Anfragen mit jedem einzelnen Protein, Precision als Gütemaß



- Experimentelle Untersuchung: Granularität der Partitionierung

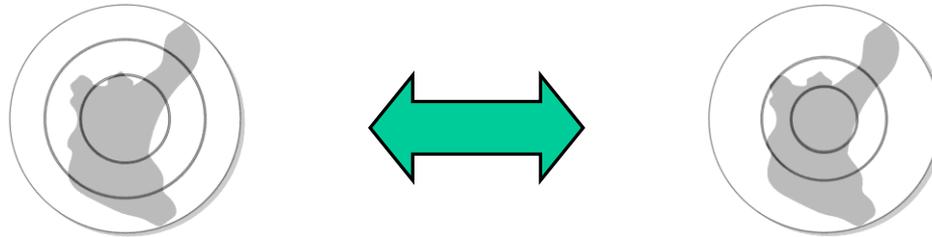


## 3.2.2 Erweiterungen der Formhistogramme

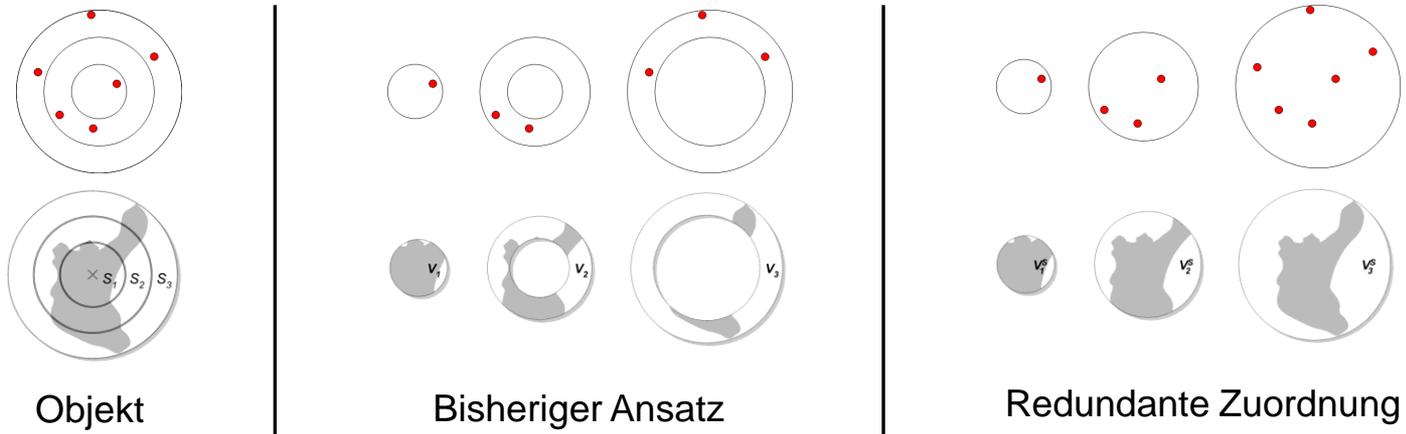
### – Verbesserung der Formhistogramme

[Aßfalg, Kriegel, Kröger, Pötke. Proc. Int. Symp. on Spatial and Temporal Databases (SSTD), 2005]

- Proportionale Aufteilung



- Redundante Zuordnung zu den Bins

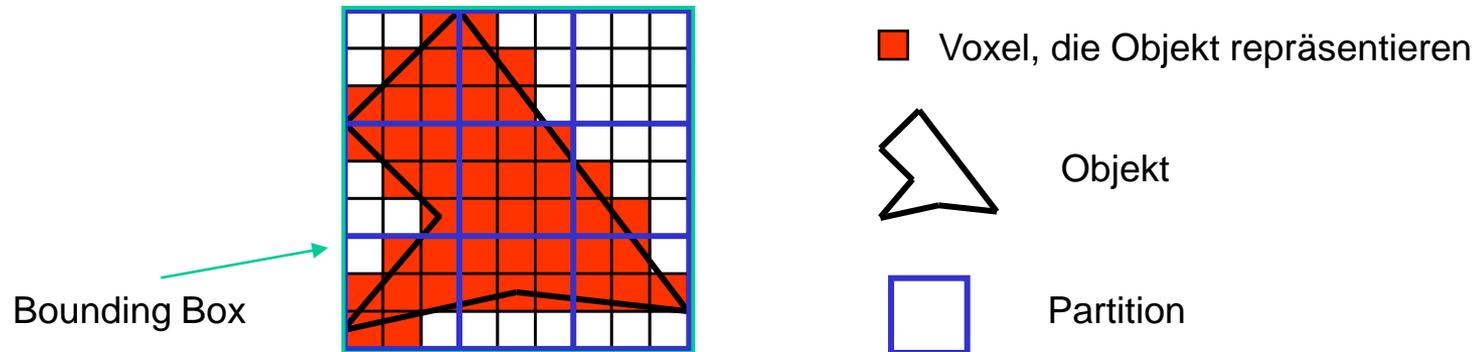


## – Erweiterung für Voxelisierte Objekte

[Kriegel, Kröger, Mashaël, Pfeifle, Pötke, Seidl. Proc. Int. Conf. Database Systems for Advanced Applications (DASFAA), 2003]

### • Partitionierung

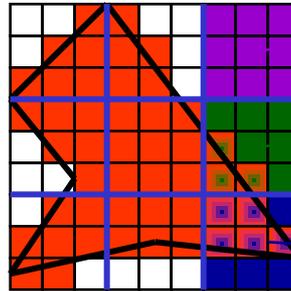
- Kugelförmige Partitionierung ist bei Voxelmengen nicht sinnvoll
  - » Voxel können auf Schnittfläche zwei oder mehr Partitionen liegen
  - » Zu welchen Partitionen sollen diese Voxel hinzugezählt werden?
  - » Sollen Voxel zu in mehreren Partitionen hinzu gezählt werden?
- Daher: würfelförmige Partitionierung der Bounding Box eines Objekts
  - » Jedes Voxel liegt in genau einer Zelle
  - » ACHTUNG: Partitionierung nicht mehr rotationsinvariant



- Räumliche Features

- Volumen Modell (ursprünglicher Ansatz)
  - » Anzahl der Objekt-Voxel pro Partition
  - » Normiert mit der Kapazität einer Partition (# aller Voxel pro Zelle)

Kapazität einer Partition: 9 Voxel



$$\begin{pmatrix} \vdots \\ 0/9 \\ \vdots \\ 3/9 \\ \vdots \\ 5/9 \\ \vdots \end{pmatrix}$$

- Bewertung:

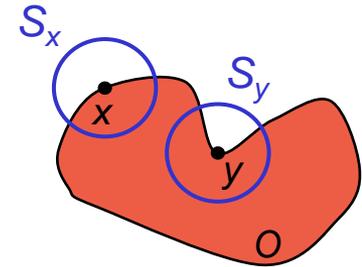
- » Einfaches Modell
- » Erweiterung: extrahiere andere Features, die die Form des Objekts innerhalb der Zellen beschreibt („Shape Descriptor“):
  - Solid Angle Wert: beschreibt die Konvexität/Konkavität
  - Eigenwerte der Hauptachsen: beschreiben die Varianz entlang der Hauptachsen (Objektausrichtung)

– Solid Angle Modell

- » Voxelisierte Referenzsphäre  $S_c$  um Zentrums-Voxel  $c$
- » Berechnen für jedes Oberflächen-Voxel  $v$  von Objekt  $o$  den SA-Wert:

$$SA(v) = \frac{|S_v \cap V^o|}{|S_v|}$$

$V^o =$  Voxelmenge, die Objekt  $o$  repräsentiert

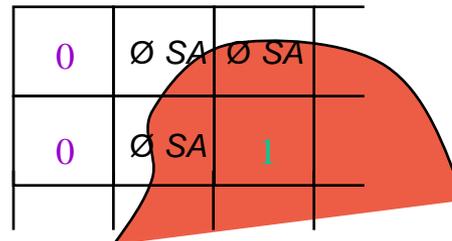


- » Berechne für jede Zelle  $z$  ein Feature  $f(z)$ :

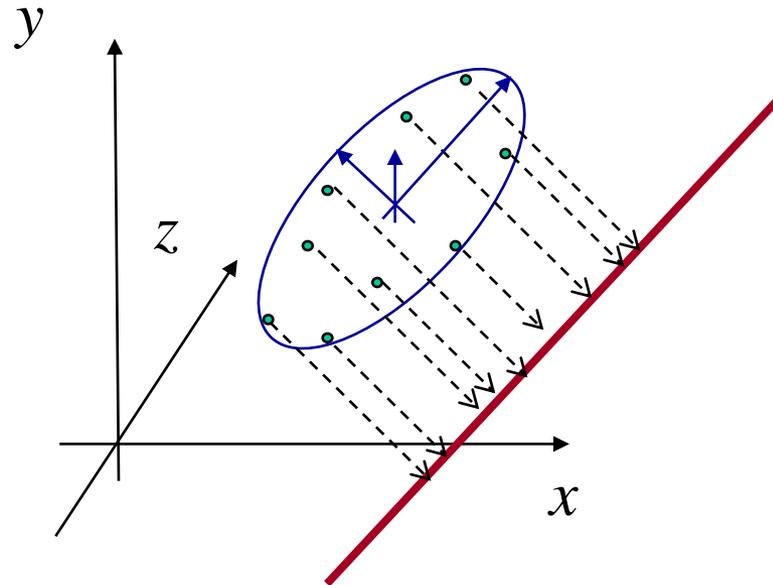
$f(z) = 0$  falls  $z$  keine Objekt-Voxel enthält

$f(z) = 1$  falls  $z$  nur Voxel aus dem Inneren des Objekts enthält

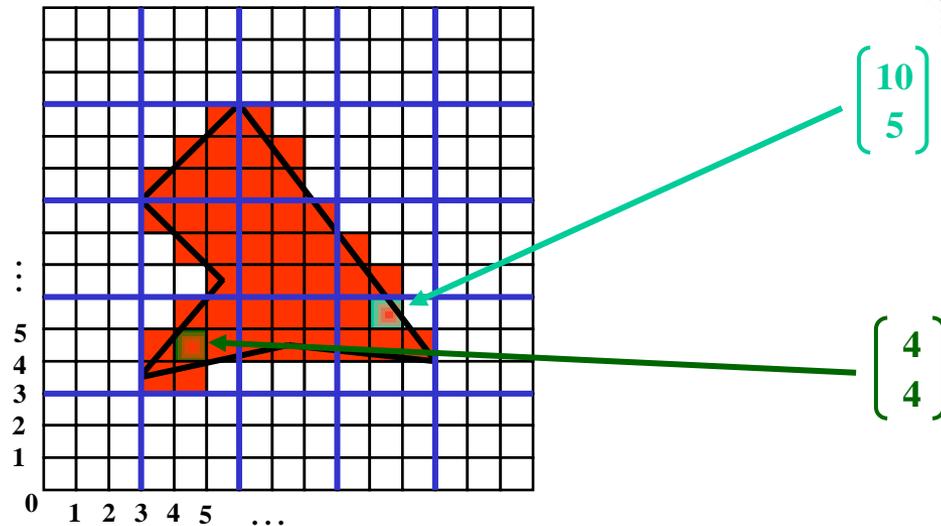
$f(z) = \frac{1}{m} \sum_{i=1}^m SA(v_i)$  falls  $z$   $m$  Oberflächen-Voxel  $v_i$  des Objekts enthält (durchschnittlicher SA-Wert aller Oberflächenvoxel in  $z$ )



- Eigenwert Modell
  - » Hauptachsenanalyse (PCA)
  - » Eigenvektoren und Eigenwerte spannen das minimal umgebende Ellipsoid einer Punktmenge auf
  - » Eigenvektoren repräsentieren die Hauptachsen (Hauptausdehnungen) der Punktmenge; stehen senkrecht aufeinander
  - » Eigenwerte modellieren die Streuung der Punktmenge entlang dieser Hauptachsen
  - » Extrahiere diese Streuung der Voxelmenge innerhalb einer Zelle als Feature



- » Modelliere jedes Voxel  $v$  des Objekts  $o$  als Vektor  $\vec{v}^o = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$



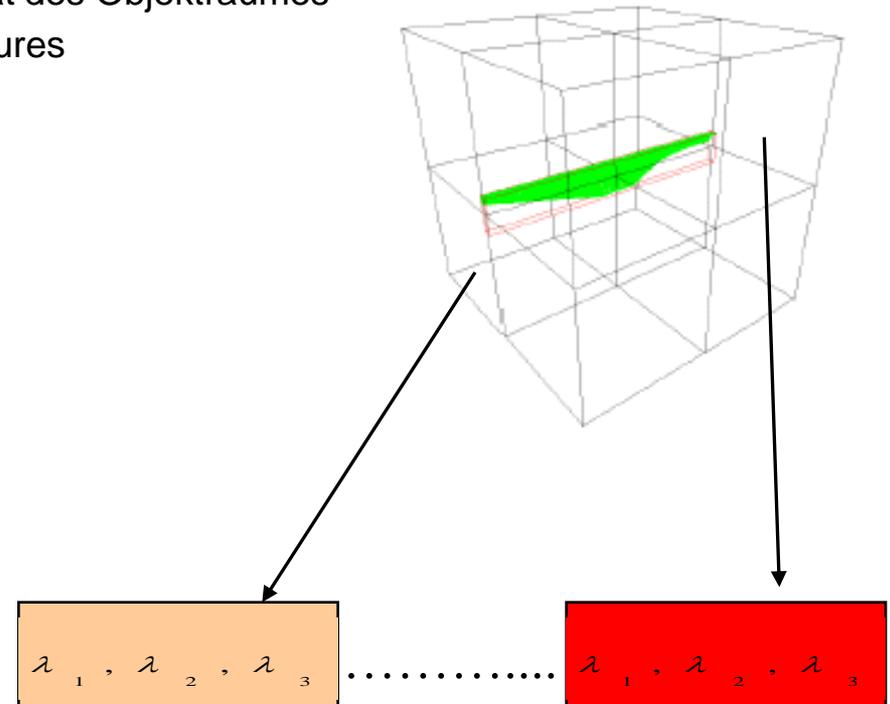
- » PCA: Kovarianzmatrix von Voxeln des Objekts  $o$  in Zelle  $i$

$$\text{Cov}_i^o = \frac{1}{|V_i^o| - 1} \begin{pmatrix} \sum_{j=1}^{|V_i^o|} x_j^2 & \sum_{j=1}^{|V_i^o|} x_j y_j & \sum_{j=1}^{|V_i^o|} x_j z_j \\ \sum_{j=1}^{|V_i^o|} x_j y_j & \sum_{j=1}^{|V_i^o|} y_j^2 & \sum_{j=1}^{|V_i^o|} y_j z_j \\ \sum_{j=1}^{|V_i^o|} x_j z_j & \sum_{j=1}^{|V_i^o|} y_j z_j & \sum_{j=1}^{|V_i^o|} z_j^2 \end{pmatrix}$$

$V_i^o$  = Voxelmenge in Zelle  $i$ , die Objekt  $o$  repräsentiert

- » Bestimmung der Eigenwerte  $\lambda_1, \dots, \lambda_d$  der Kovarianzmatrix
- » Für 3D Objekte 3 Eigenwerte
- » Allgemein:  $d$  = Dimensionalität des Objektraumes
- » Pro Zelle z. extrahiere  $d$  Features

$$f(z) = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_d \end{pmatrix}$$



– Vergleich: Bei  $k$  Zellen

- » Volumen Modell:  $d$ -dimensionales Objekt entspricht  $k$ -dimensionalem Vektor
- » Solid Angle Modell:  $d$ -dimensionales Objekt entspricht  $k$ -dimensionalem Vektor
- » Eigenwert Modell:  $d$ -dimensionales Objekt entspricht  $(d \cdot k)$ -dimensionalem Vektor