

# Matching and Recognition in 3D

Based on slides by Tom Funkhouser and Misha Kazhdan

# From 2D to 3D: Some Things Easier



No occlusion (but sometimes missing data instead)

Segmenting objects often simpler

# From 2D to 3D: Many Things Harder



Rigid transform has 6 degrees of freedom vs. 3

- Brute-force algorithms much less practical

Rotations do not commute

- Difficult to parameterize, search over

No natural parameterization for surfaces in 3D

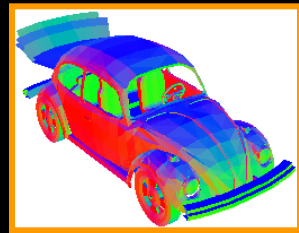
- Hard to do FFT, convolution, PCA
- Exception: range images (which are view dependent)

# Shape Matching Challenge

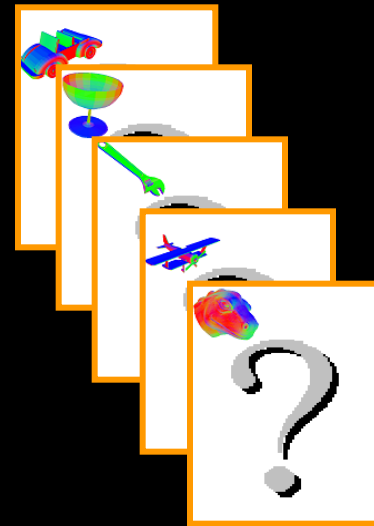


Need shape descriptor & matching method that is:

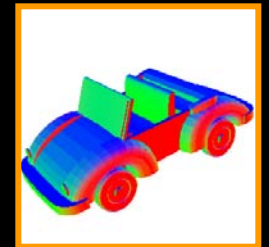
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



3D Query



3D Database



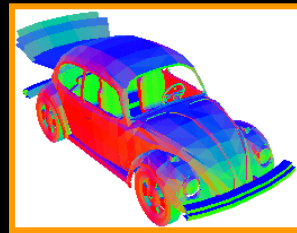
Best Matches

# Shape Matching Challenge



Need shape descriptor & matching method that is:

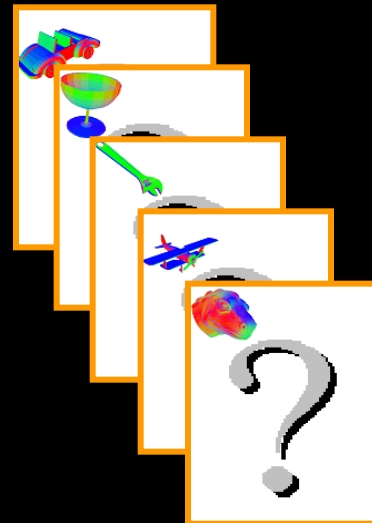
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



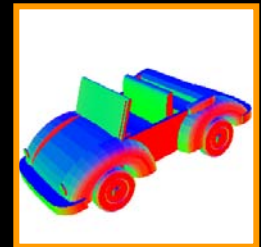
3D Query



Shape  
Descriptor



3D Database



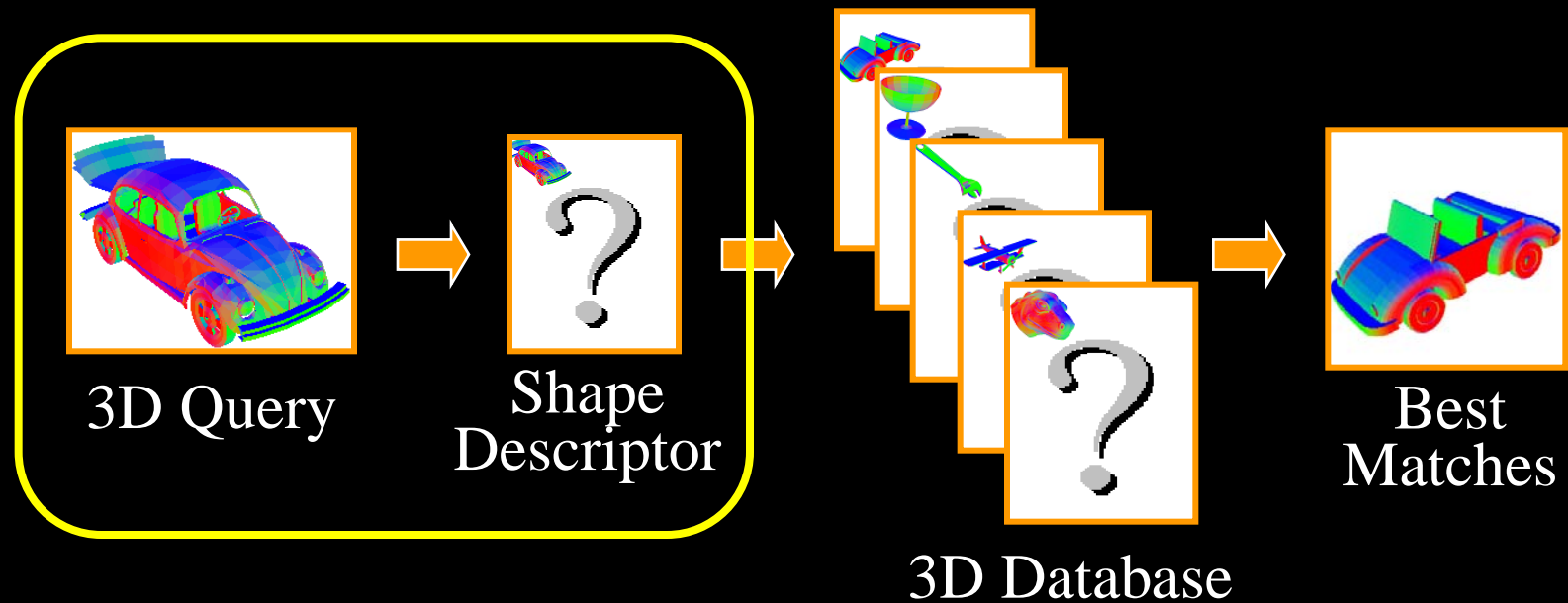
Best  
Matches



# Shape Matching Challenge

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating

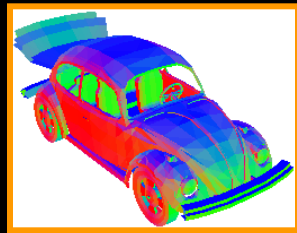




# Shape Matching Challenge

Need shape descriptor & matching method that is:

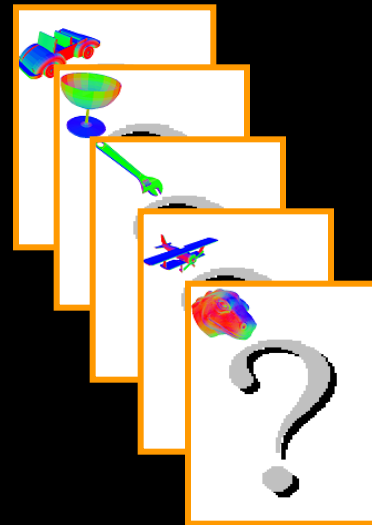
- Concise to store
- Quick to compute
- **Efficient to match**
- Discriminating



3D Query



Shape  
Descriptor



3D Database



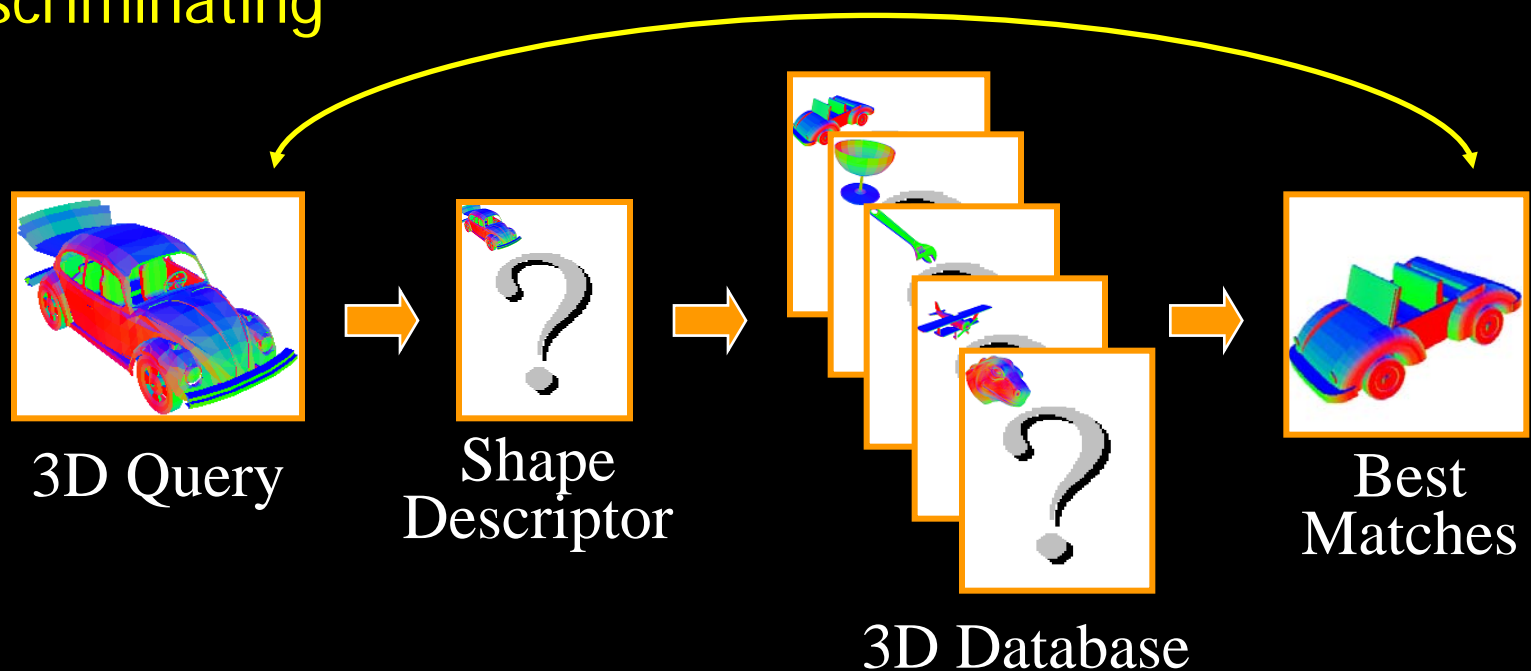
Best  
Matches



# Shape Matching Challenge

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- **Discriminating**



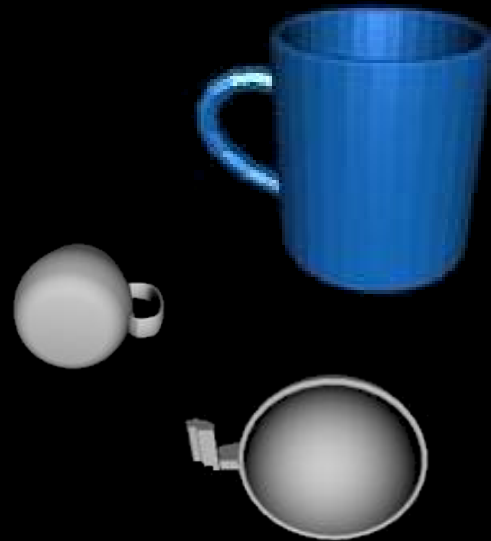




# Shape Matching Challenge

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating
- **Invariant to transformations**
  - Invariant to deformations
  - Insensitive to noise
  - Insensitive to topology
  - Robust to degeneracies



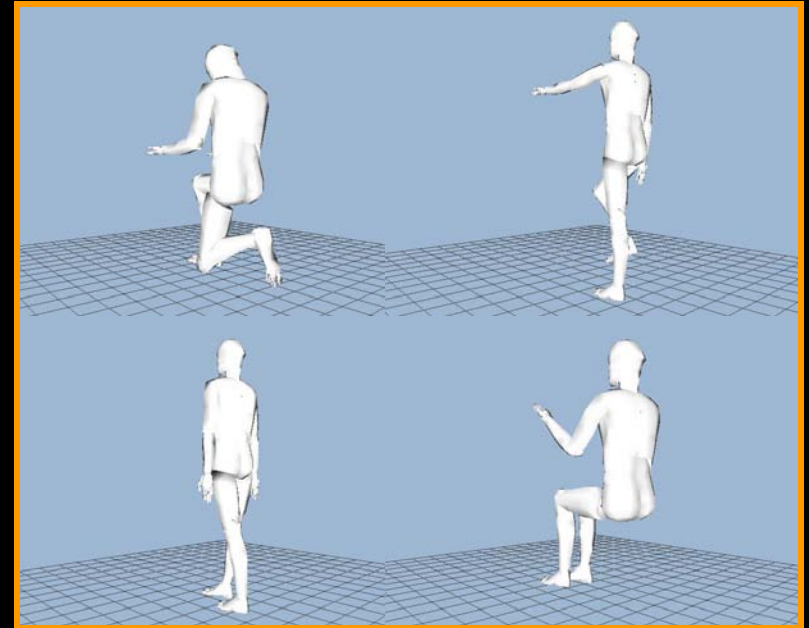
Different Transformations  
(translation, scale, rotation, mirror)



# Shape Matching Challenge

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating
- Invariant to transformations
- **Invariant to deformations**
- Insensitive to noise
- Insensitive to topology
- Robust to degeneracies



Different Articulated Poses

# Shape Matching Challenge

Image courtesy of  
Ramamoorthi et al.

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating
- Invariant to transformations
- Invariant to deformations
- **Inensitive to noise**
  - Inensitive to topology
  - Robust to degeneracies



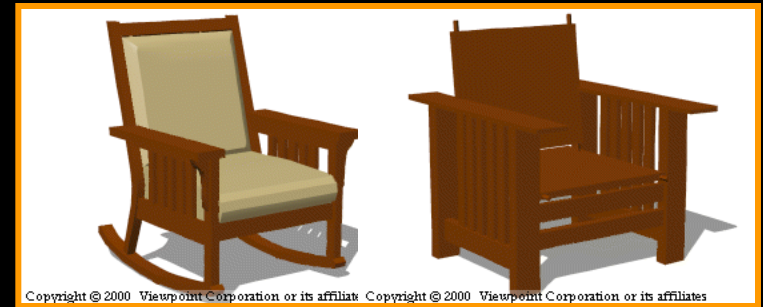
Scanned Surface

# Shape Matching Challenge

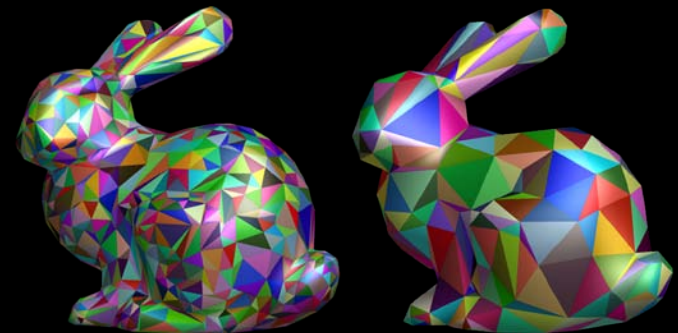
Images courtesy of  
Viewpoint & Stanford

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating
- Invariant to transformations
- Invariant to deformations
- Insensitive to noise
- **Insensitive to topology**
- Robust to degeneracies



Different Genus



Different Tessellations

# Shape Matching Challenge

Images courtesy of  
Utah & De Espona

Need shape descriptor & matching method that is:

- Concise to store
- Quick to compute
- Efficient to match
- Discriminating
- Invariant to transformations
- Invariant to deformations
- Insensitive to noise
- Insensitive to topology
- **Robust to degeneracies**



No Bottom!



&\*Q?@#A%!

# Taxonomy of 3D Matching Methods

Images courtesy of  
Amelia & Osada

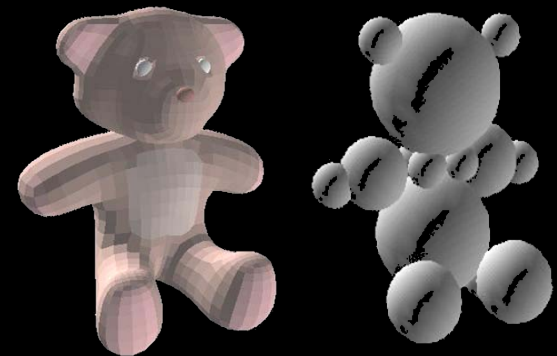
## Structural representations

- Skeletons
- Part-based methods
- Feature-based methods



## Statistical representations

- Attribute feature vectors
- Volumetric methods
- Surface-based methods
- View-based methods





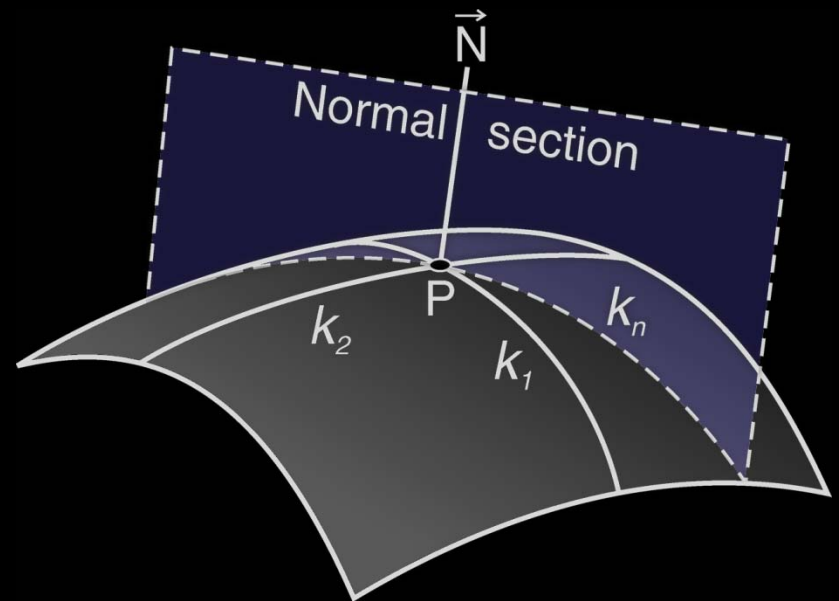
# Features on Surfaces

Can construct edge and corner detectors

Analogue of 1<sup>st</sup> derivative: surface normal

Analogue of 2<sup>nd</sup> derivative: curvature

- Curvature at each point in each direction
- Minimum and maximum: "principal curvatures"
- Can threshold or do nonmaximum suppression



# Using Curvatures for Recognition/Matching



Curvature histograms: compute  $\kappa_1$  and  $\kappa_2$  throughout surface, create 2D histograms

Invariant to translation, rotation

Alternative: use  $\kappa_2 / \kappa_1$  – also invariant to scale

- Shape index:  $S = \frac{1}{2} - \frac{1}{\pi} \tan^{-1} \frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2} \in [0..1]$

Curvatures sensitive to noise (2<sup>nd</sup> derivative...), so sometimes just use sign of curvatures





# Using Curvatures for Segmentation

Sharp creases in surface (i.e., where  $|\kappa_1|$  is large) tend to be good places to segment

Option #1: look for maxima of curvature in the first principal direction

- Much like Canny edge detection
- Nonmaximum suppression, hysteresis thresholding

Option #2: optimize for both high curvature and smoothness using graph cuts, snakes, etc.

# Taxonomy of 3D Matching Methods

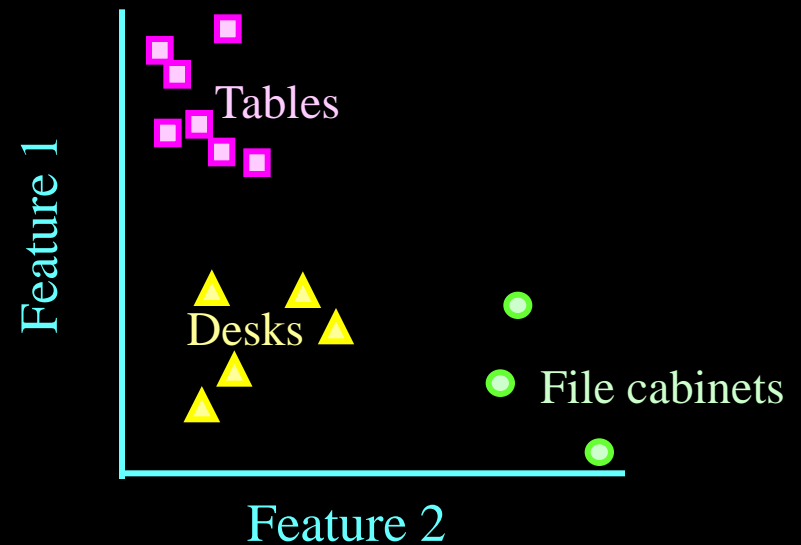
Image courtesy of  
Mao Chen

## Structural representations

- Skeletons
- Part-based methods
- Feature-based methods

## Statistical representations

- Attribute feature vectors
- Volumetric methods
- Surface-based methods
- View-based methods



# Example



## Shape distributions

- Shape representation: probability distributions
- Distance measure: difference between distributions
- Evaluation method: classification performance

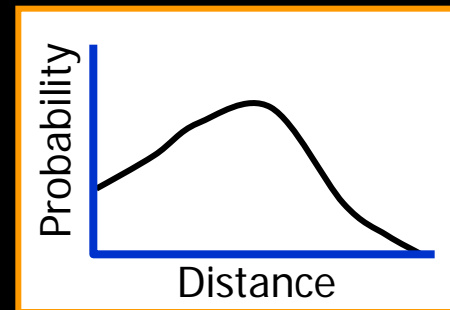
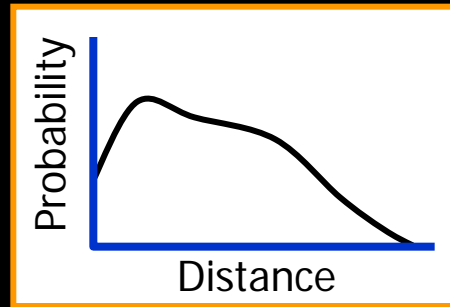
# Shape Distributions



Key idea: map 3D surfaces to common parameterization by randomly sampling shape function



Randomly  
sample  
shape  
function



Similarity  
Measure

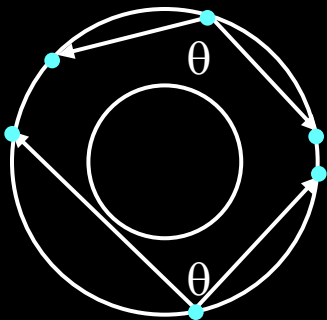
3D Models

D2 Shape Distributions

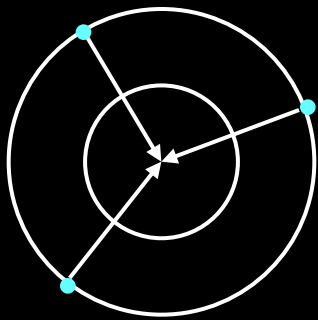


# Which Shape Function?

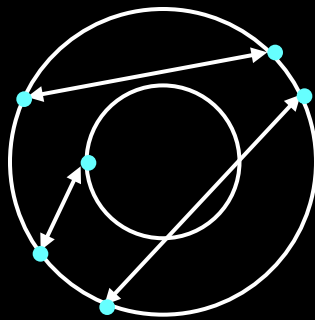
Implementation: simple shape functions based on angles, distances, areas, and volumes



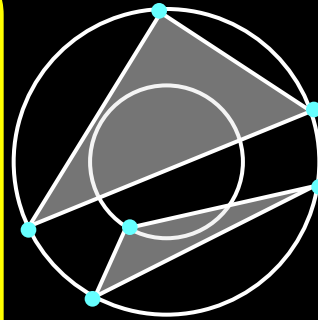
A3  
(angle)



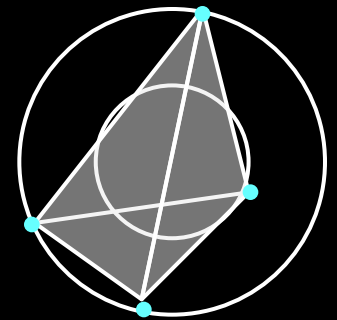
D1  
(distance)



D2  
(distance)



D3  
(area)



D4  
(volume)

[Ankerst 99]

# D2 Shape Distribution



## Properties

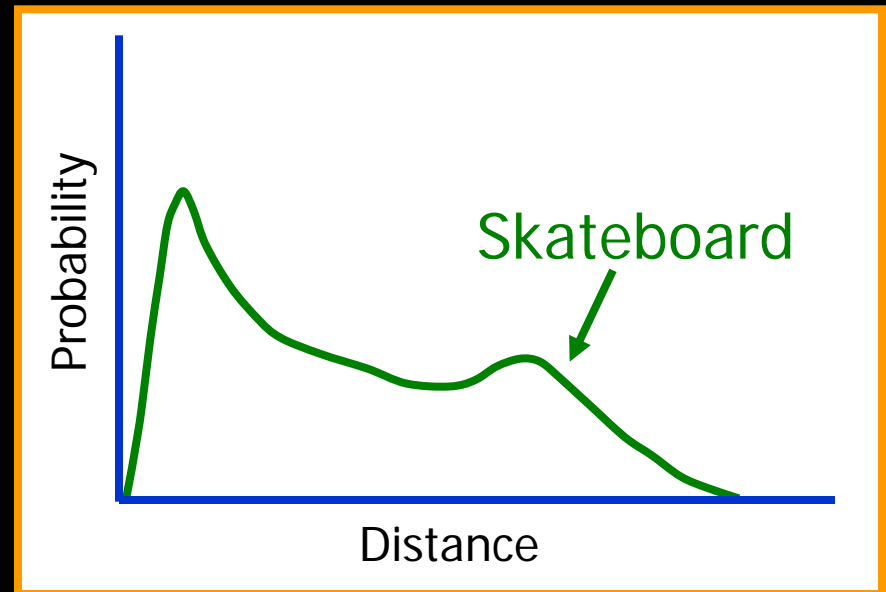
- Concise to store?
- Quick to compute?
- Invariant to transforms?
- Efficient to match?
- Insensitive to noise?
- Insensitive to topology?
- Robust to degeneracies?
- Invariant to deformations?
- Discriminating?



# D2 Shape Distribution

## Properties

- Concise to store?
- Quick to compute?
- Invariant to transforms?
- Efficient to match?
- Insensitive to noise?
- Insensitive to topology?
- Robust to degeneracies?
- Invariant to deformations?
- Discriminating?



512 bytes (64 values)

0.5 seconds ( $10^6$  samples)

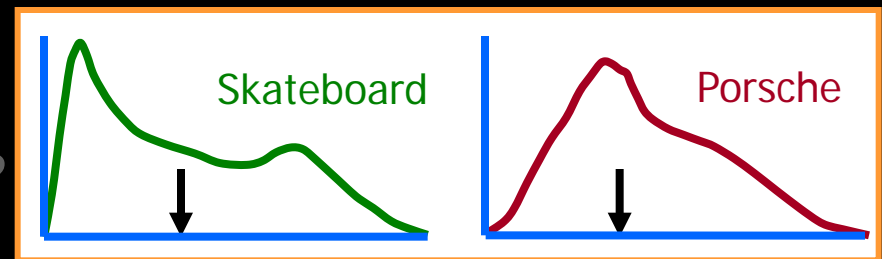


# D2 Shape Distribution

## Properties

- ✓ Concise to store
- ✓ Quick to compute
- **Invariant to transforms?**
  - Efficient to match?
  - Insensitive to noise?
  - Insensitive to topology?
  - Robust to degeneracies?
  - Invariant to deformations?
  - Discriminating?

- ✓ Translation
- ✓ Rotation
- ✓ Mirror
- ✓ Scale (w/ normalization)



Normalized Means

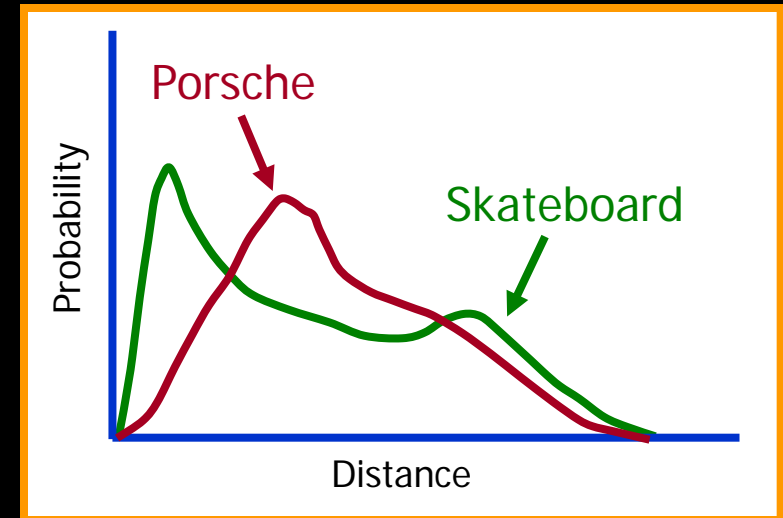


# D2 Shape Distribution



## Properties

- ✓ Concise to store
- ✓ Quick to compute
- ✓ Invariant to transforms
- **Efficient to match?**
  - Insensitive to noise?
  - Insensitive to topology?
  - Robust to degeneracies?
  - Invariant to deformations?
  - Discriminating?

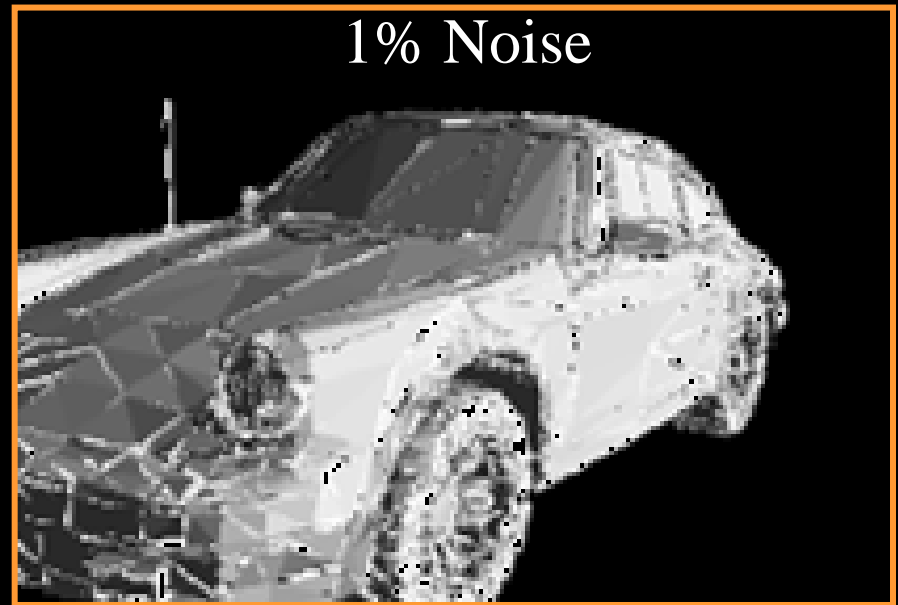




# D2 Shape Distribution

## Properties

- ✓ Concise to store
- ✓ Quick to compute
- ✓ Invariant to transforms
- ✓ Efficient to match
- **Insensitive to noise?**
- **Insensitive to topology?**
- **Robust to degeneracies?**
- Invariant to deformations?
- Discriminating?

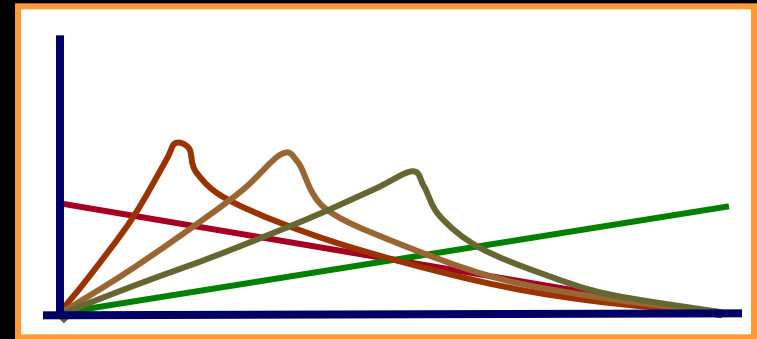




# D2 Shape Distribution

## Properties

- ✓ Concise to store
- ✓ Quick to compute
- ✓ Invariant to transforms
- ✓ Efficient to match
- ✓ Insensitive to noise
- ✓ Insensitive to topology
- ✓ Robust to degeneracies
- **Invariant to deformations?**
  - Discriminating?



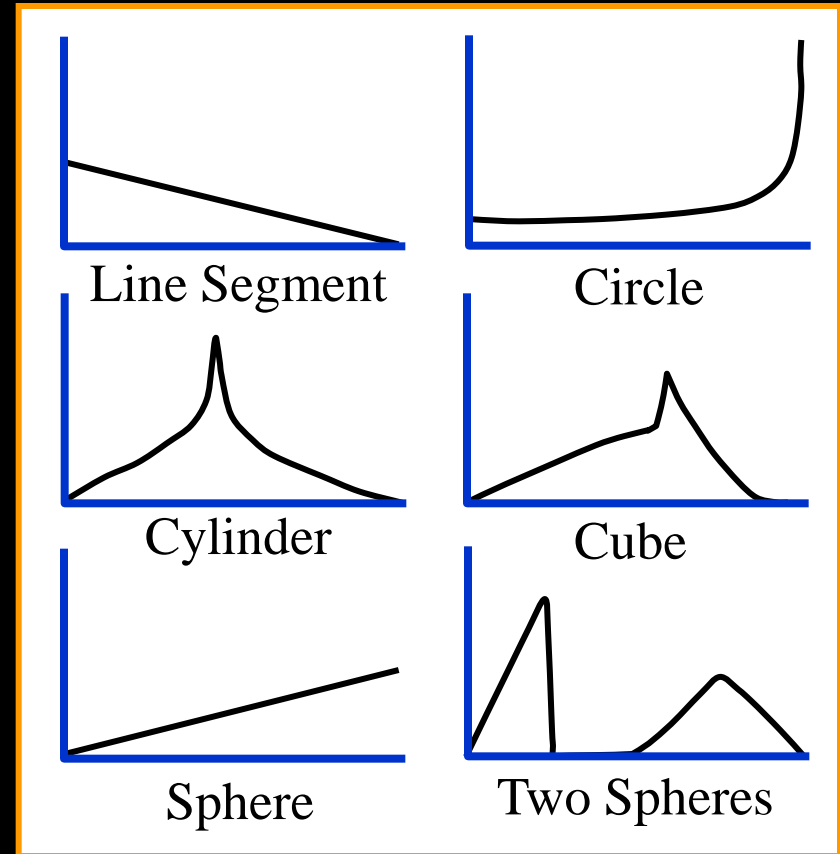
Ellipsoids with  
Different Eccentricities



# D2 Shape Distribution

## Properties

- ✓ Concise to store
- ✓ Quick to compute
- ✓ Invariant to transforms
- ✓ Efficient to match
- ✓ Insensitive to noise
- ✓ Insensitive to topology
- ✓ Robust to degeneracies
- ✗ Invariant to deformations
- **Discriminating?**



# D2 Shape Distribution Results

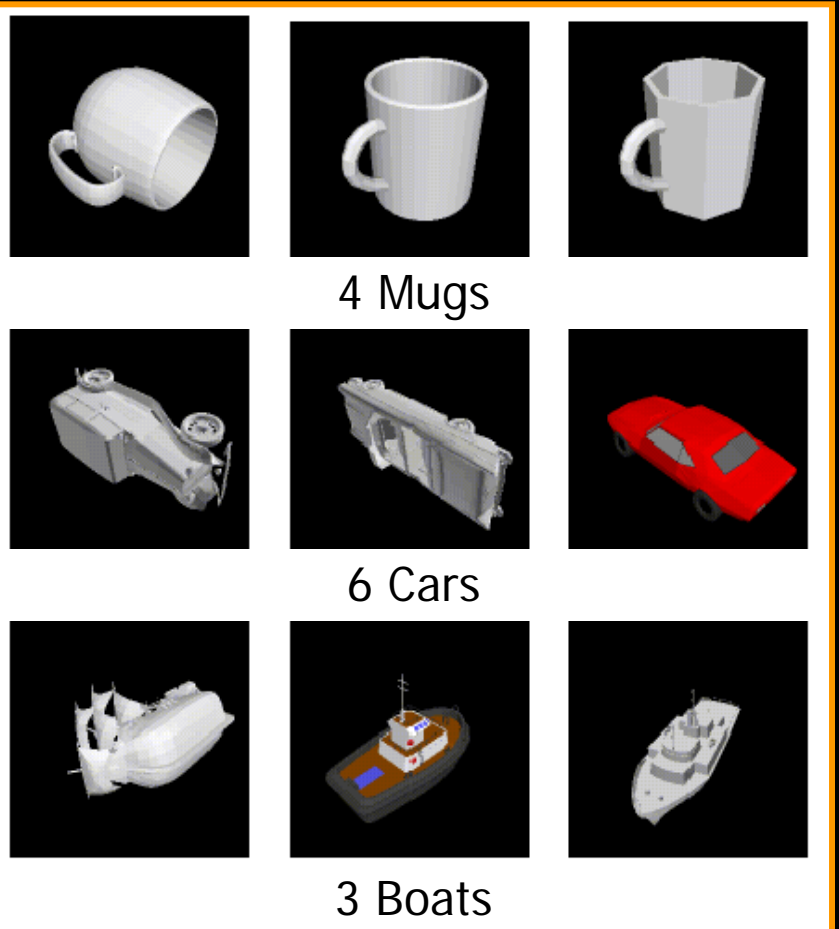


## Question

- How discriminating are D2 shape distributions?

## Test database

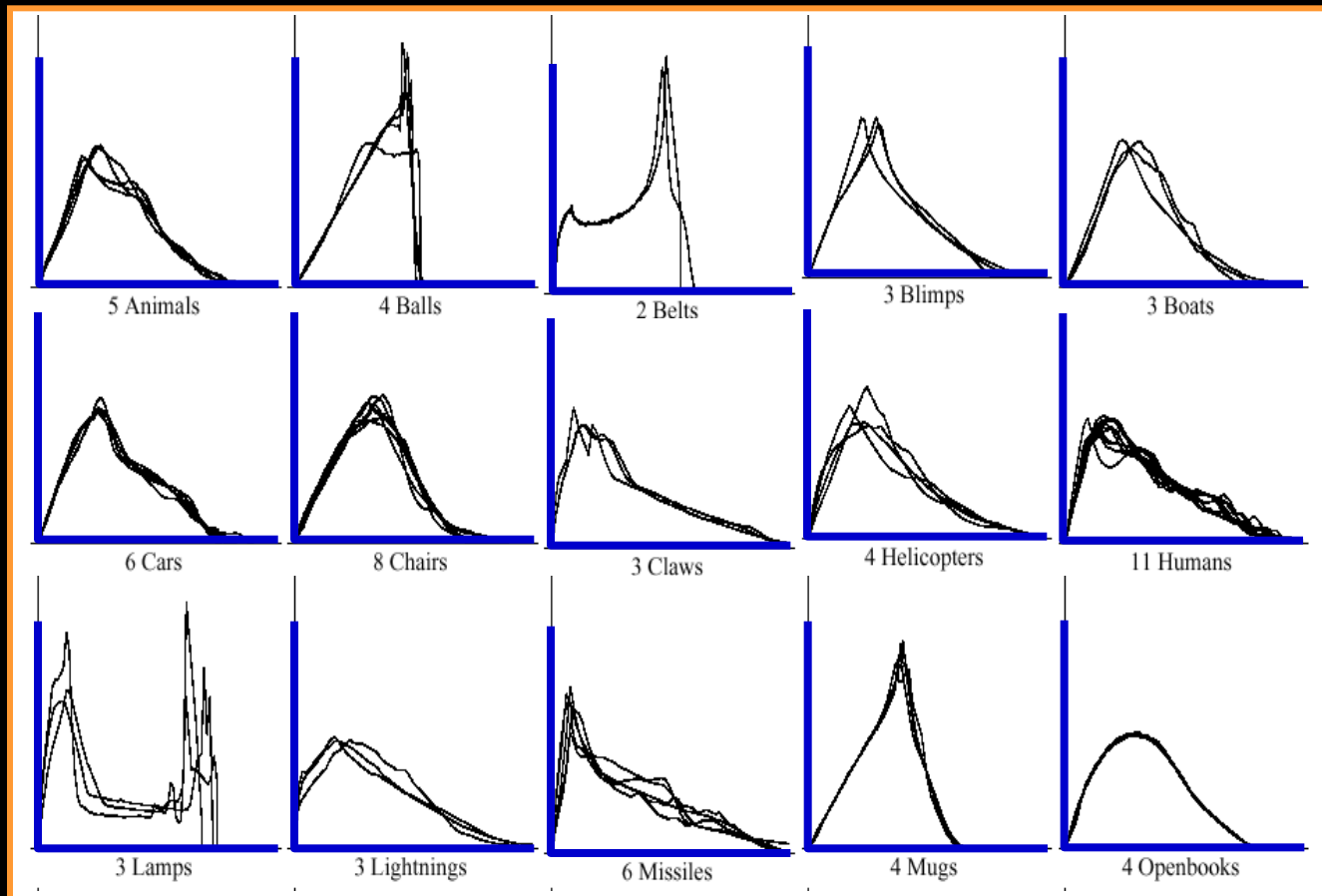
- 133 polygonal models
- 25 classes





# D2 Shape Distribution Results

D2 distributions are different across classes

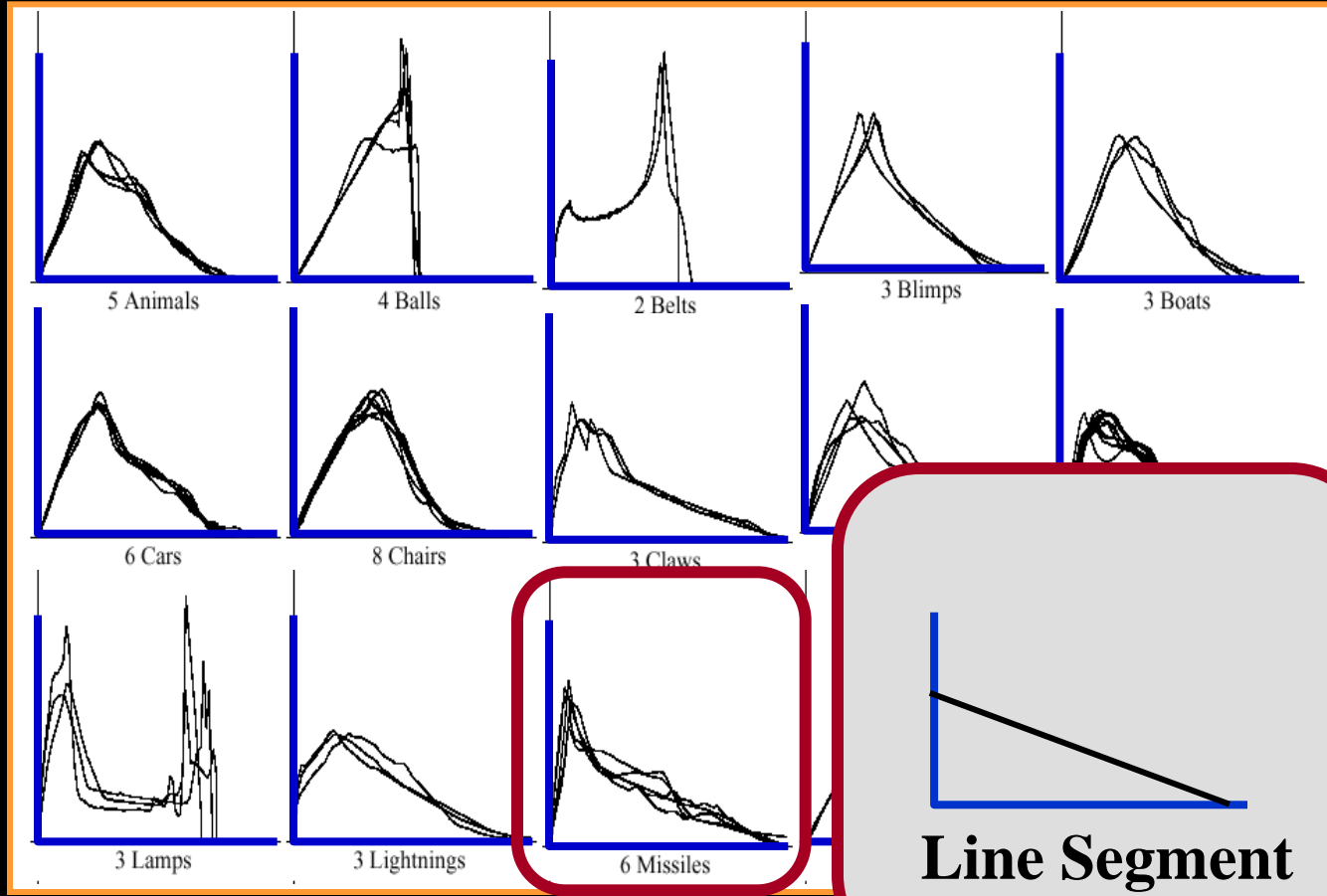


D2 shape distributions for 15 classes of objects



# D2 Shape Distribution Results

D2 distributions reveal gross shape of object

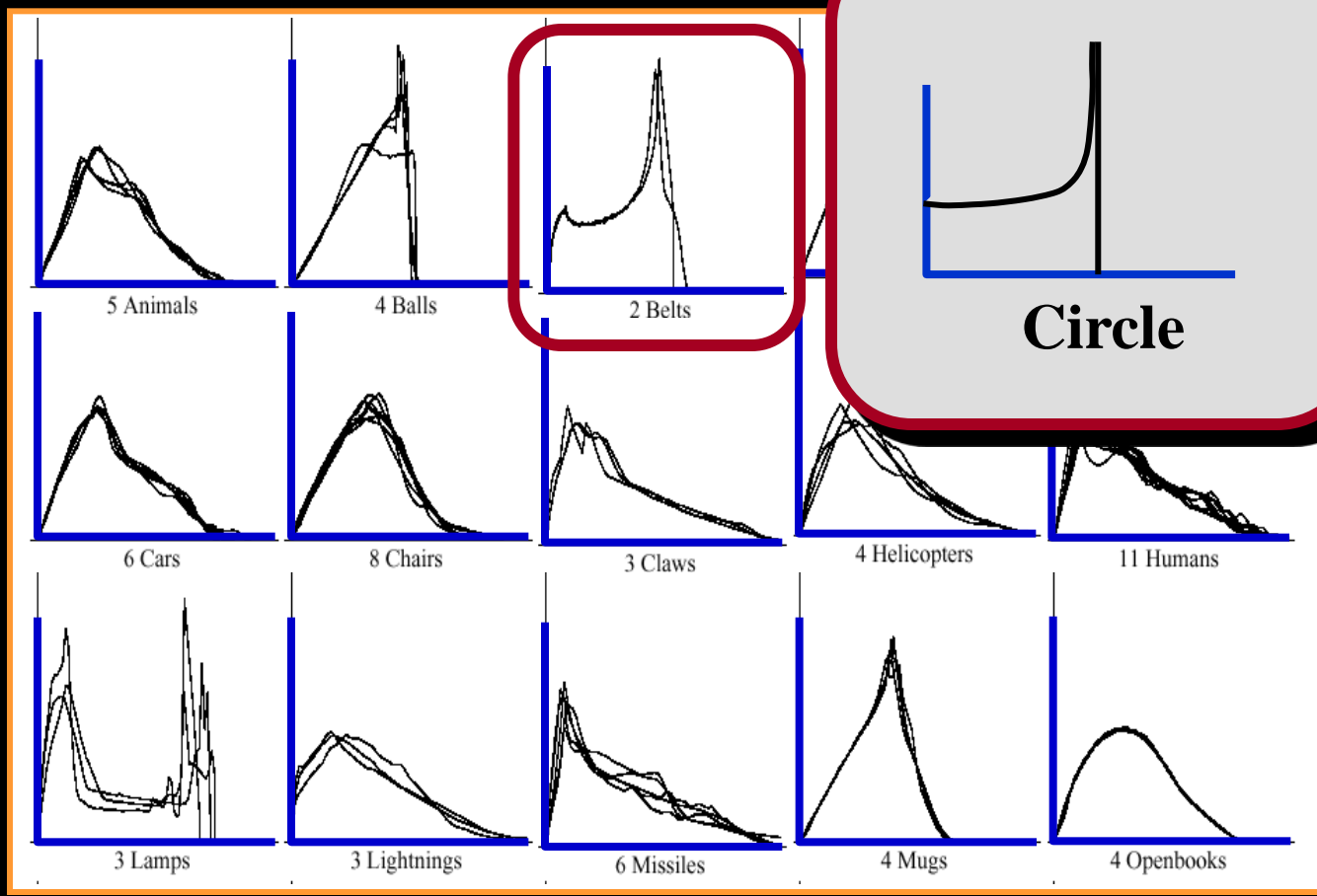


D2 shape distributions for 15 classes



# D2 Shape Distribution Results

D2 distributions reveal gross shape of object



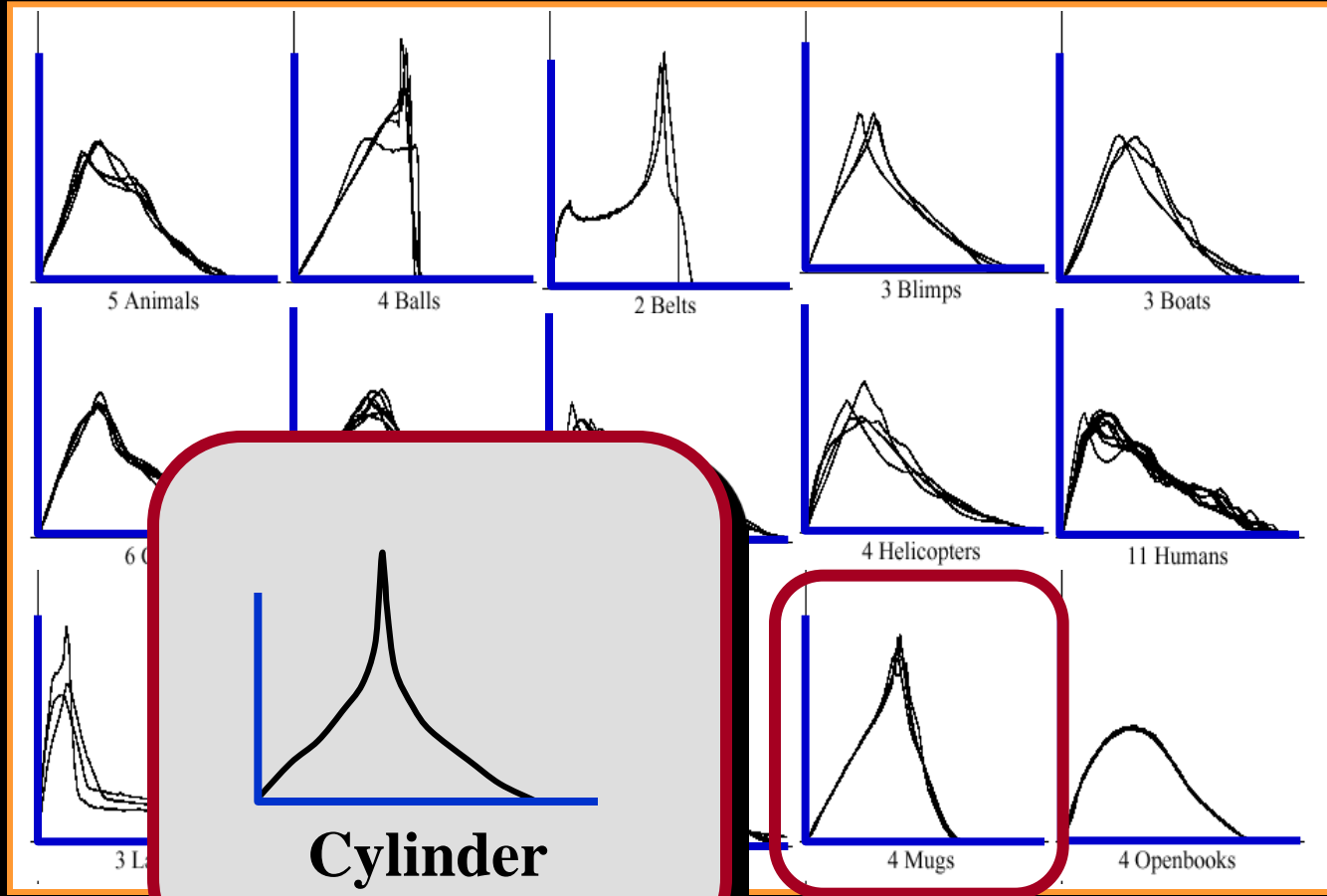
D2 shape distributions for 15 classes of objects





# D2 Shape Distribution Results

D2 distributions reveal gross shape of object

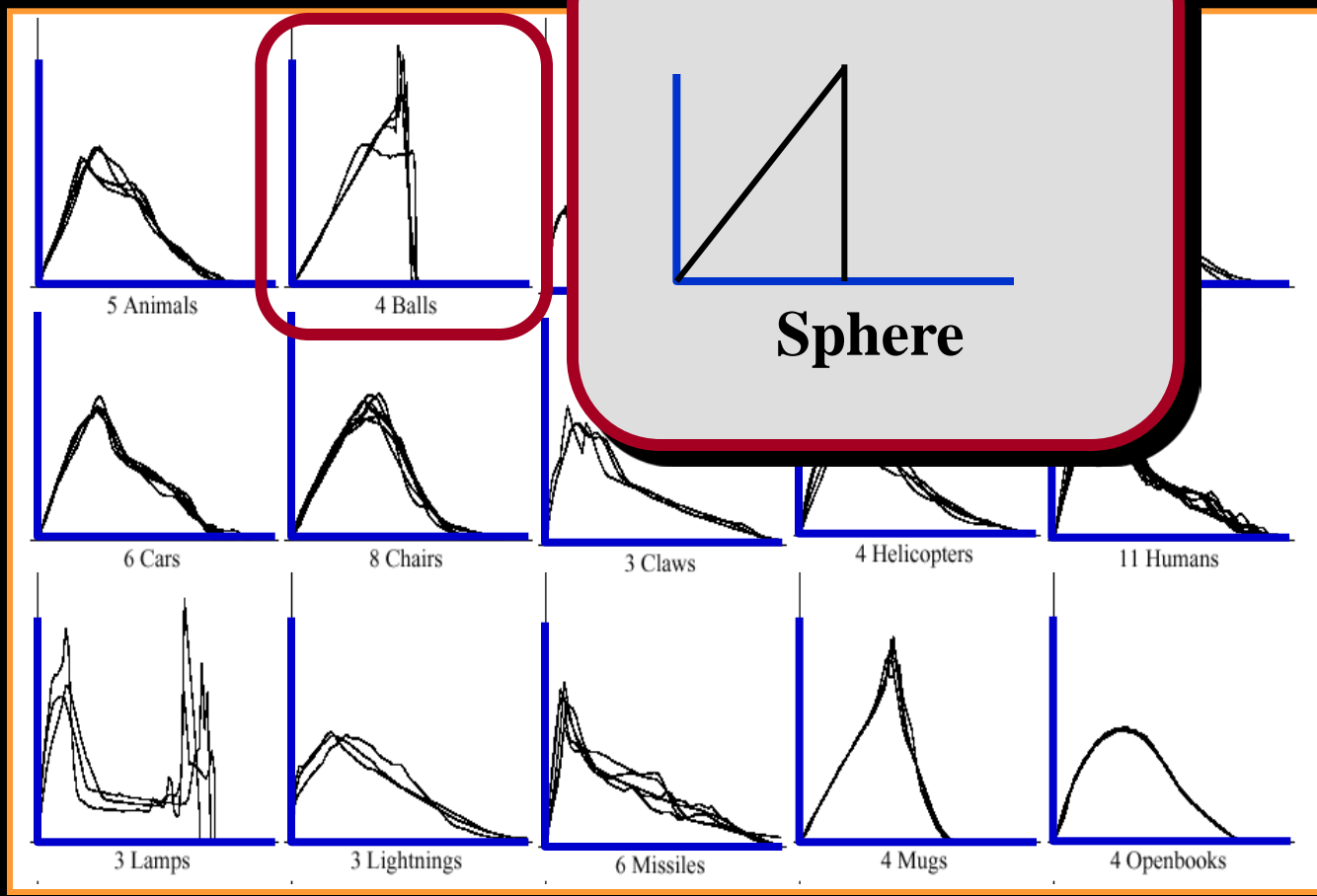


D2 distributions for 15 classes of objects



# D2 Shape Distribution Results

D2 distributions reveal gross shape information

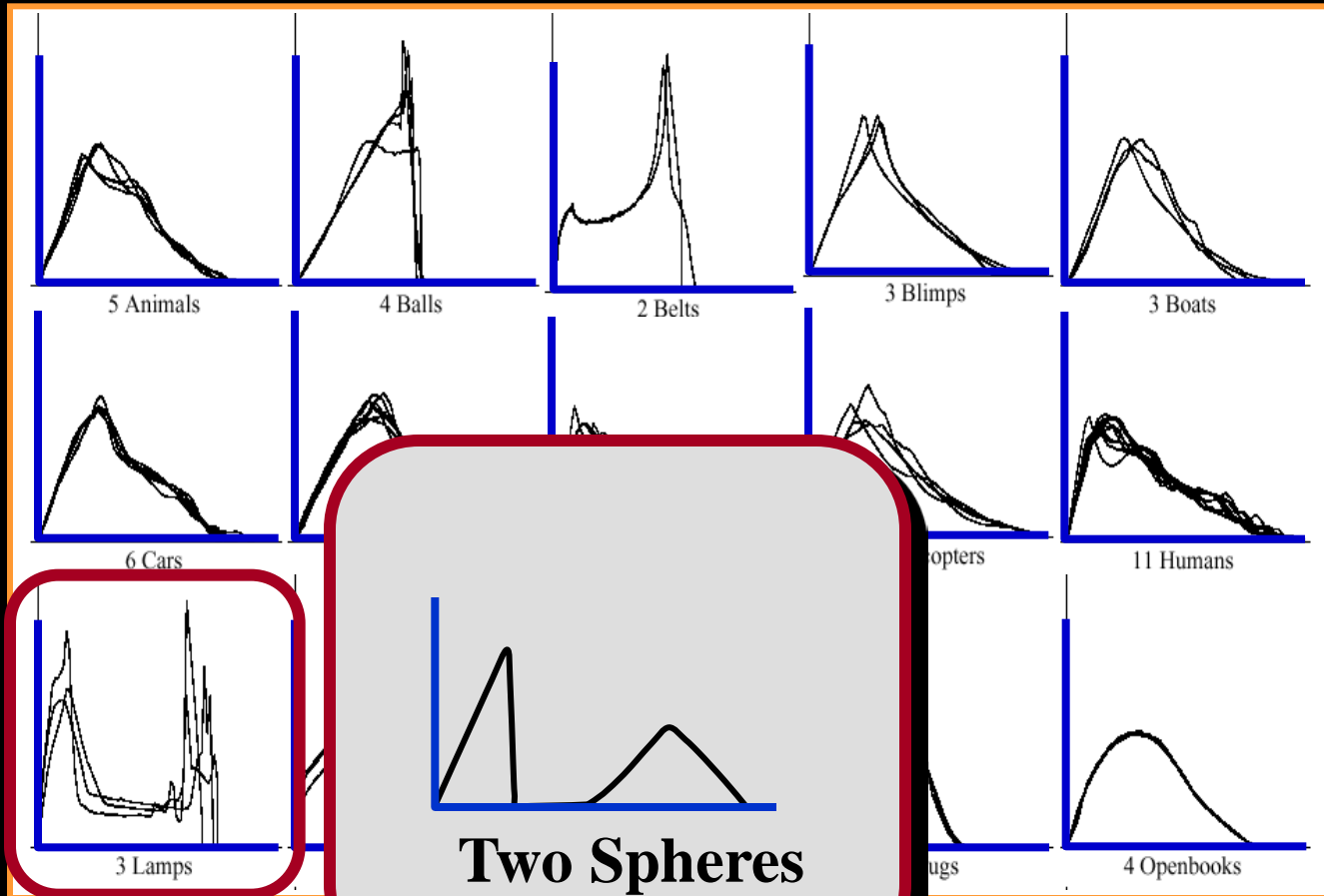


D2 shape distributions for 15 classes of objects



# D2 Shape Distribution Results

D2 distributions reveal gross shape of object

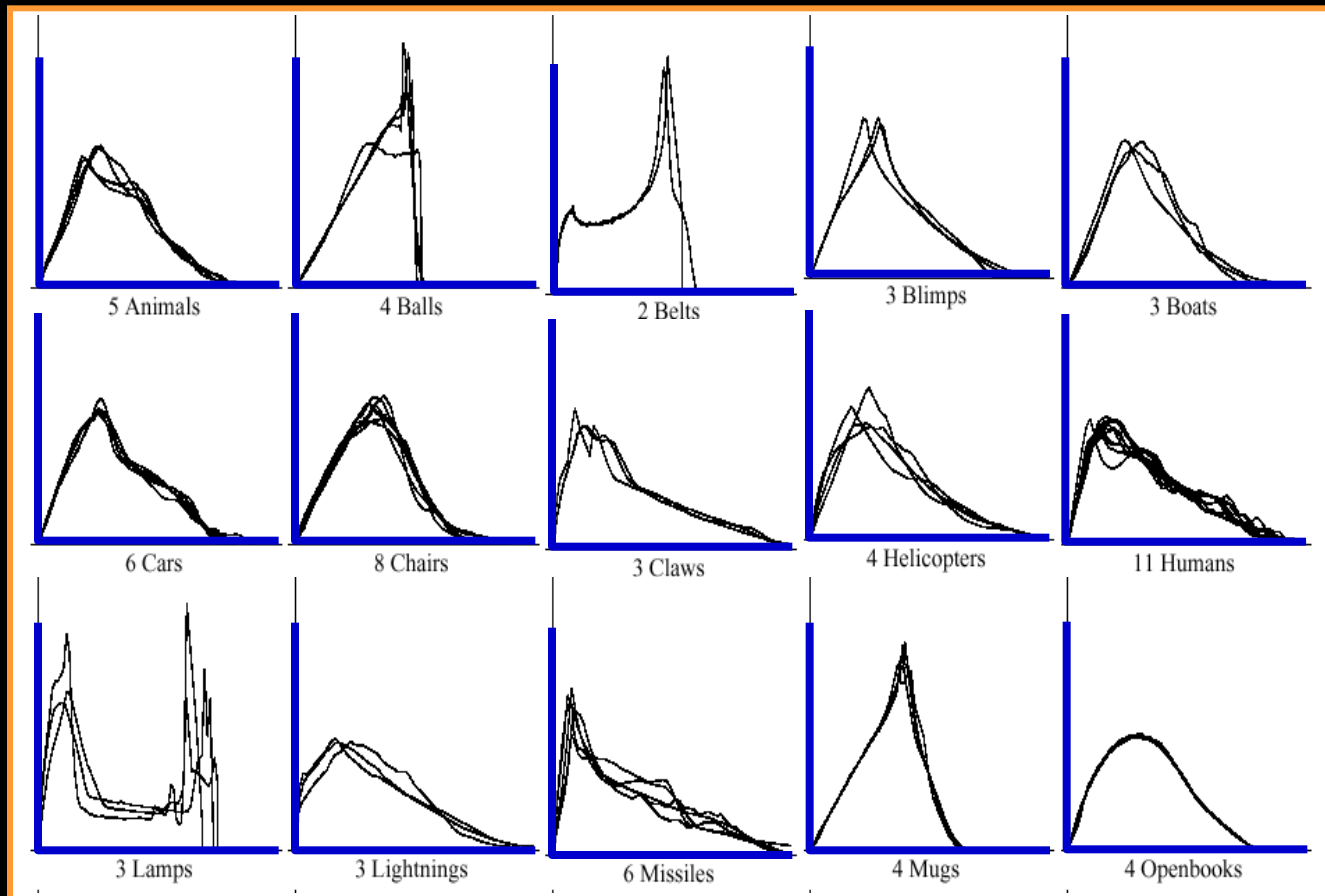


D2 shape distributions reveal gross shape of classes of objects



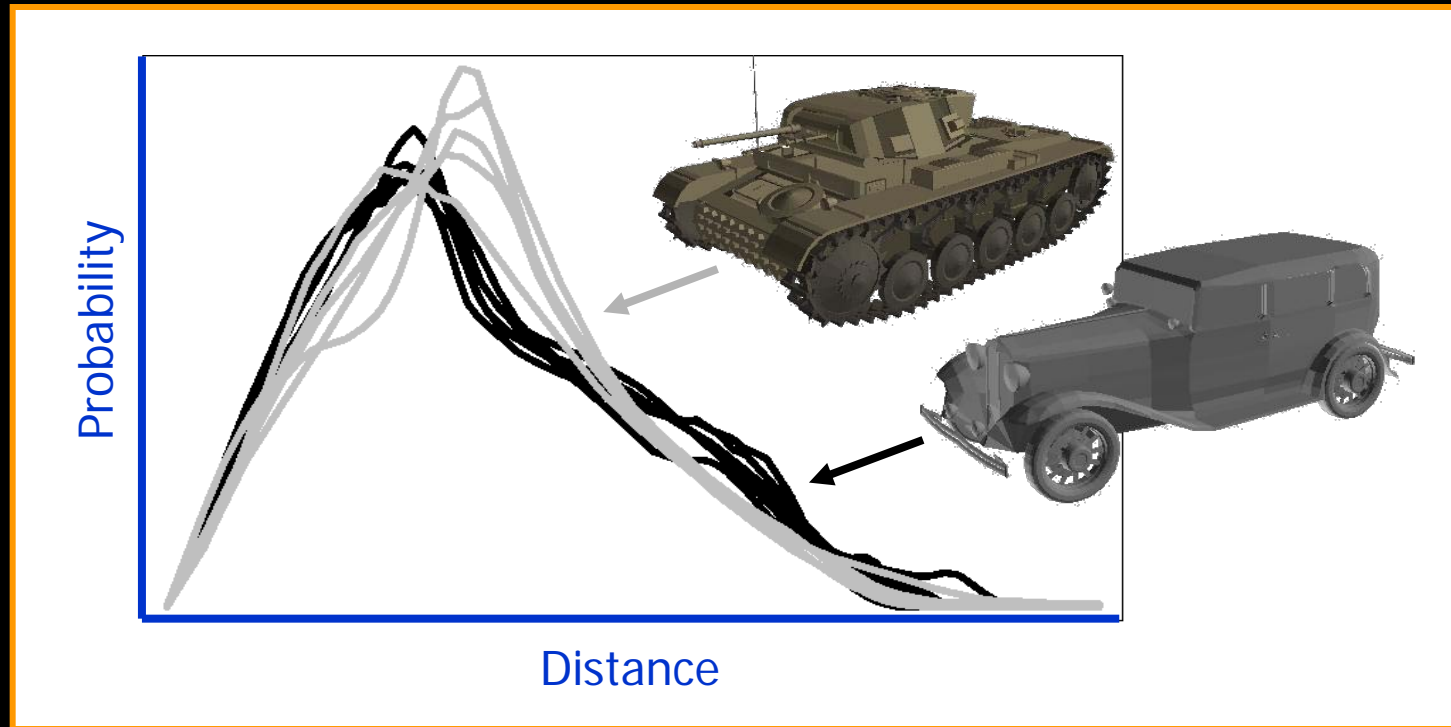
# D2 Shape Distribution Results

But ... are D2 distributions discriminating?



D2 shape distributions for 15 classes of objects

# D2 Shape Distribution Results



D2 distributions for 5 tanks (gray) and 6 cars (black)



# Evaluation Methods

For each model (the query):

- Compute match score for all models
- Rank matches from best to worst
- Measure how often models in same class as query appear near top of ranked list



Query



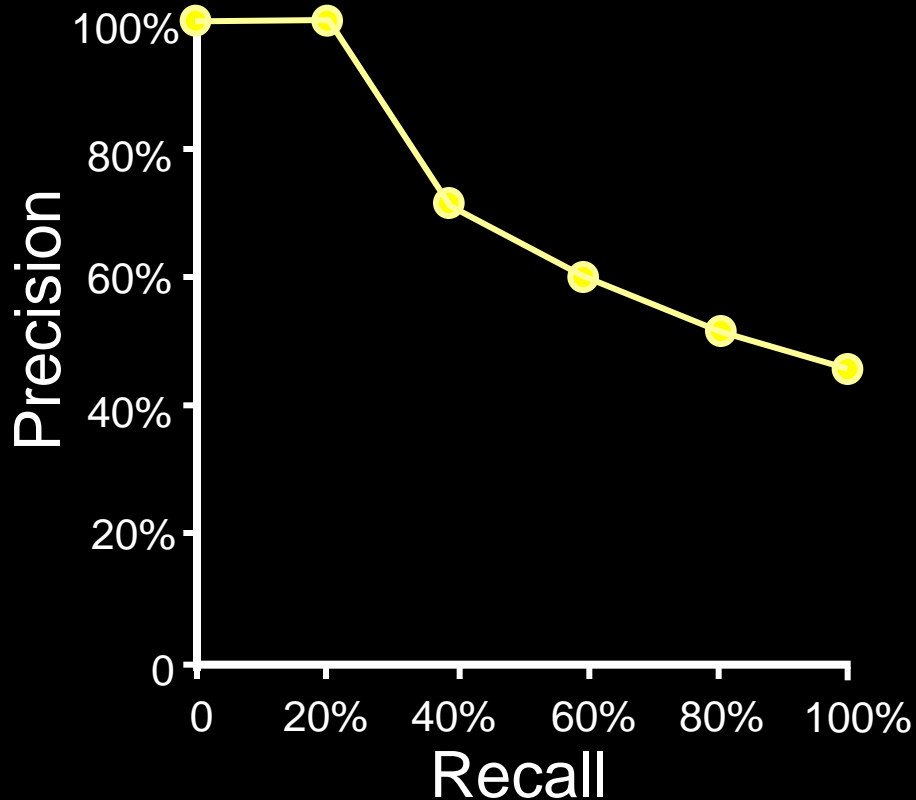
Ranked Matches



# Evaluation Methods

## Precision-recall curves

- Precision =  $\text{retrieved\_in\_class} / \text{total\_retrieved}$
- Recall =  $\text{retrieved\_in\_class} / \text{total\_in\_class}$





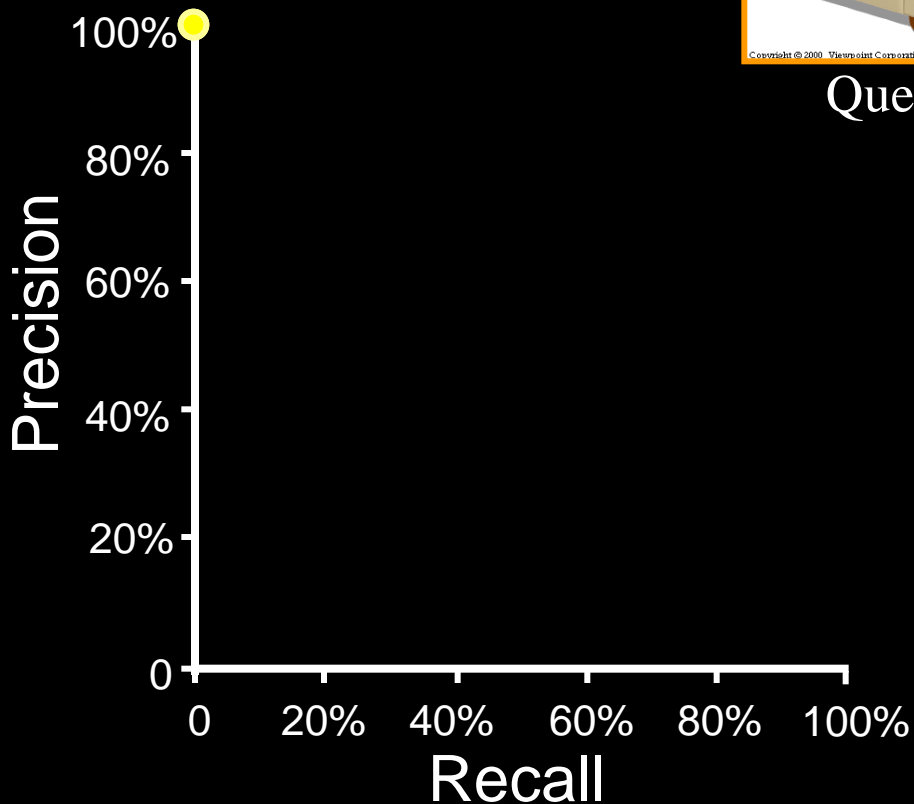
# Evaluation Methods

## Precision-recall curve example

- Precision = 0 / 0
- Recall = 0 / 5



Query



Ranked Matches





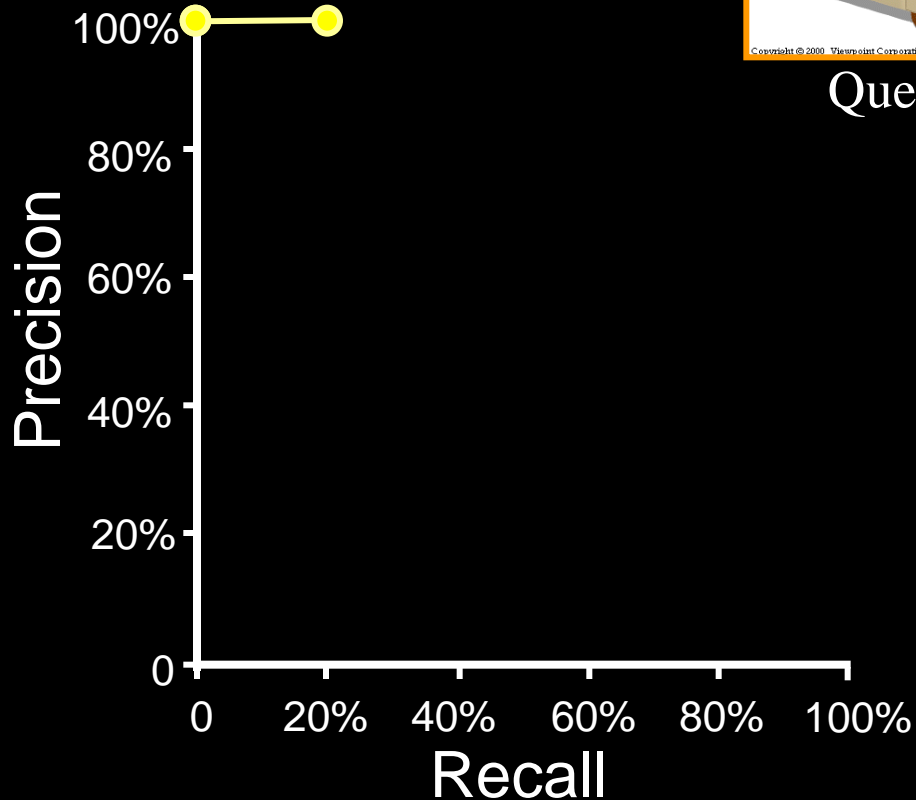
# Evaluation Methods

## Precision-recall curve example

- Precision = 1 / 1
- Recall = 1 / 5



Query



Ranked Matches



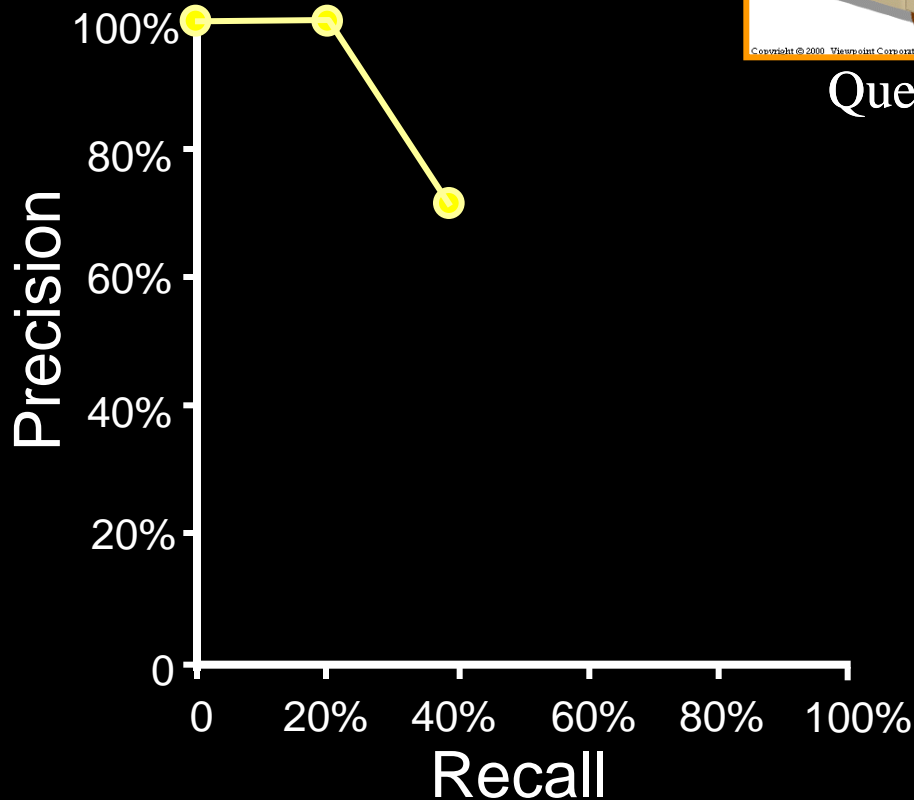
# Evaluation Methods

## Precision-recall curve example

- Precision = 2 / 3
- Recall = 2 / 5



Query



Ranked Matches



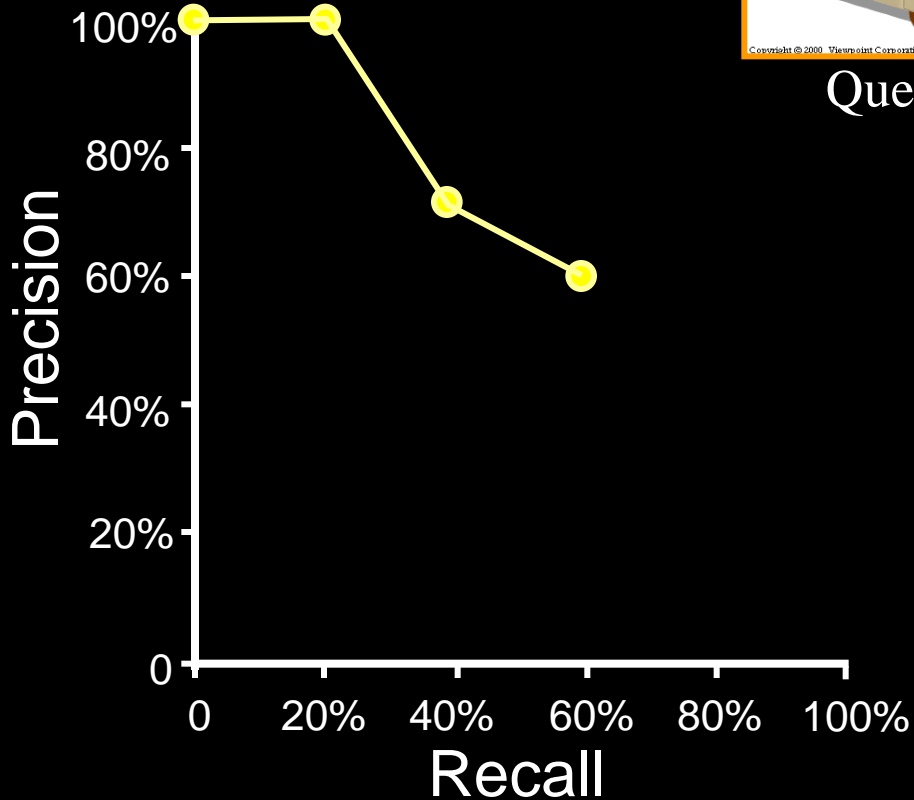
# Evaluation Methods

## Precision-recall curve example

- Precision = 3 / 5
- Recall = 3 / 5



Query



Ranked Matches



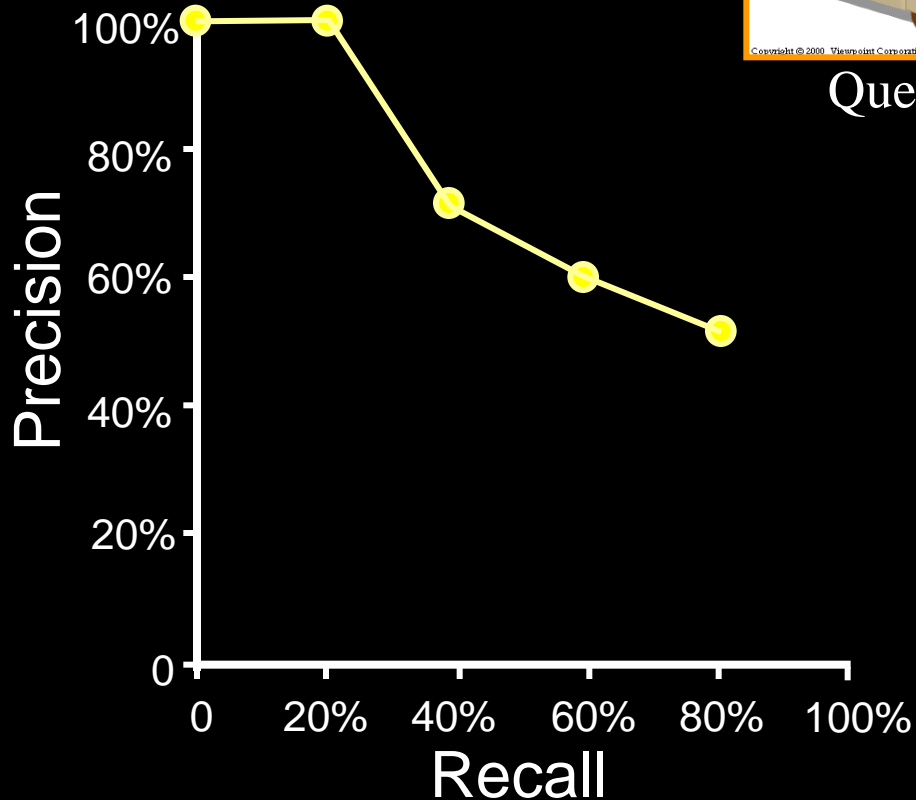
# Evaluation Methods

## Precision-recall curve example

- Precision = 4 / 7
- Recall = 4 / 5



Query



Ranked Matches



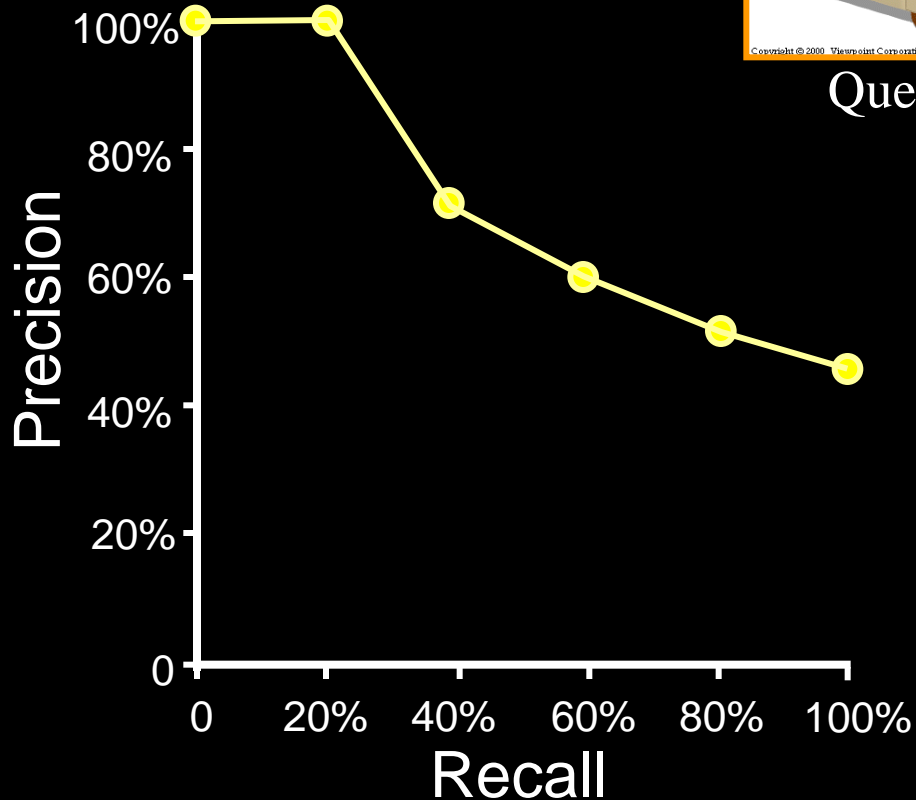
# Evaluation Methods

## Precision-recall curve example

- Precision = 5 / 9
- Recall = 5 / 5

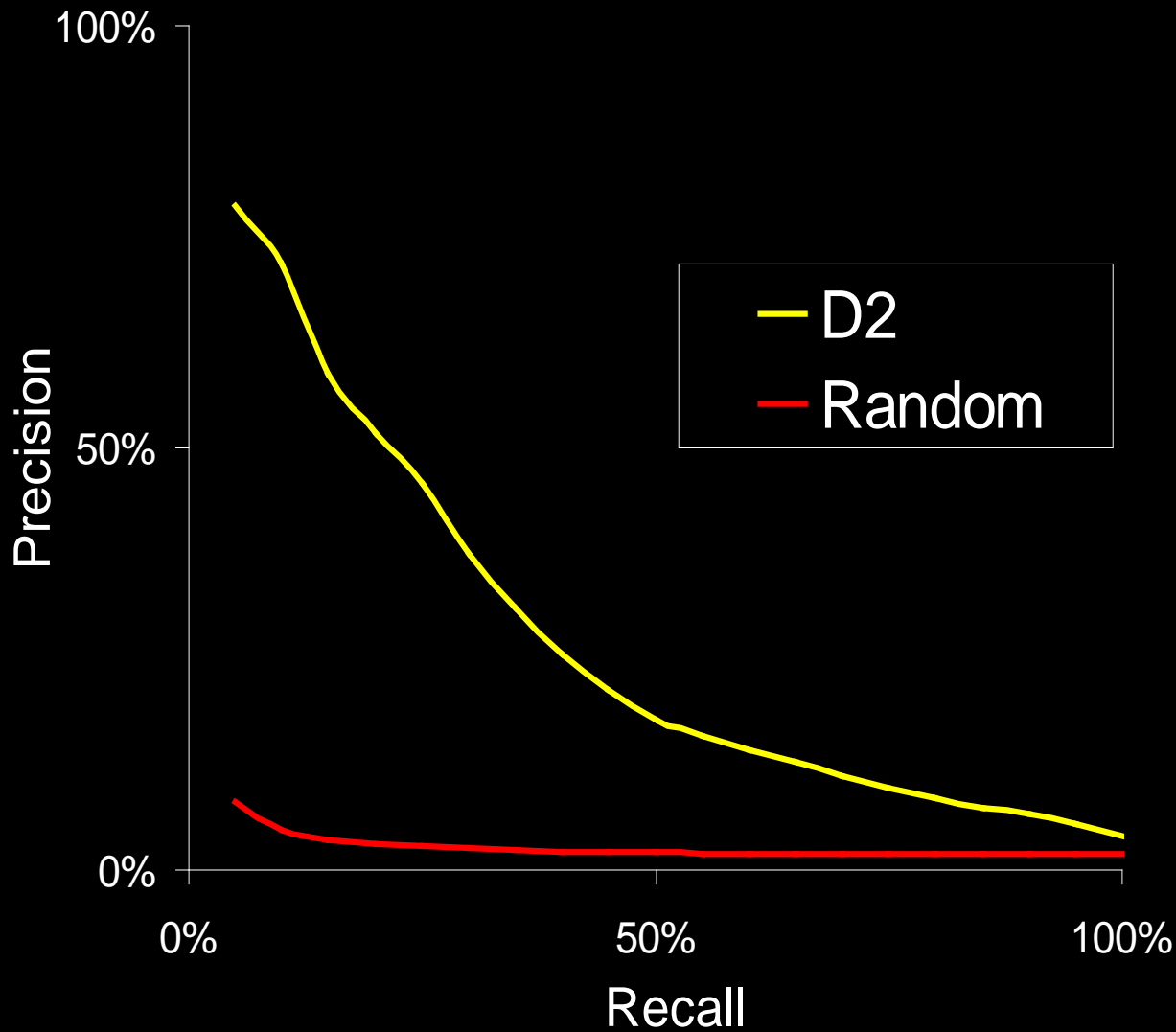


Query



Ranked Matches

# Evaluation Methods



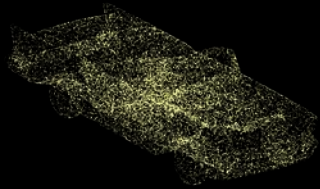


# Extended Gaussian Image

[Horn, 1984]

## Properties:

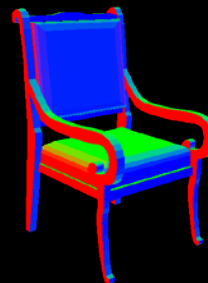
- Invertible for convex shapes
- Can be defined for most models
- 2D array of information



Point  
Clouds



Polygon  
Soups



Closed  
Meshes



Genus-0  
Meshes

Shape Spectrum



# Extended Gaussian Image

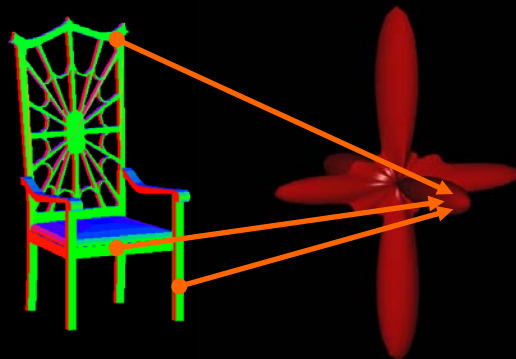
[Horn, 1984]

## Properties:

- Invertible for convex shapes
- Can be defined for most models
- 2D array of information

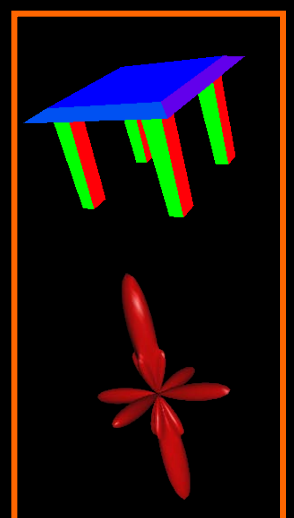
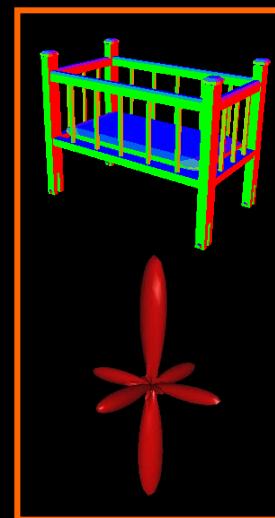
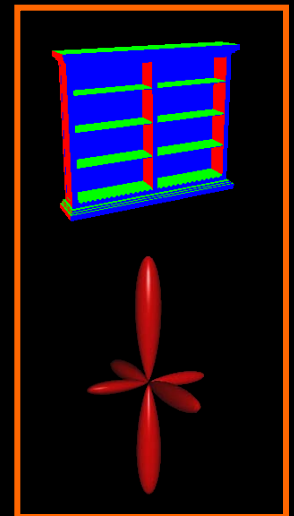
## Limitations:

- In general, shapes are not convex
- Normals are sensitive to noise



3D Model

EGI



# Extended Gaussian Image

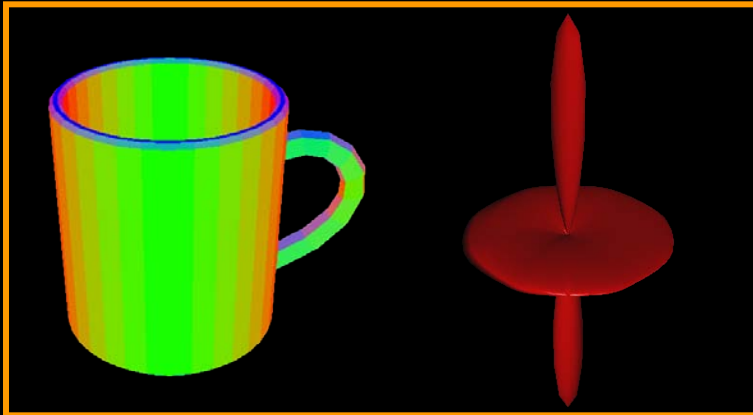
[Horn, 1984]

## Properties:

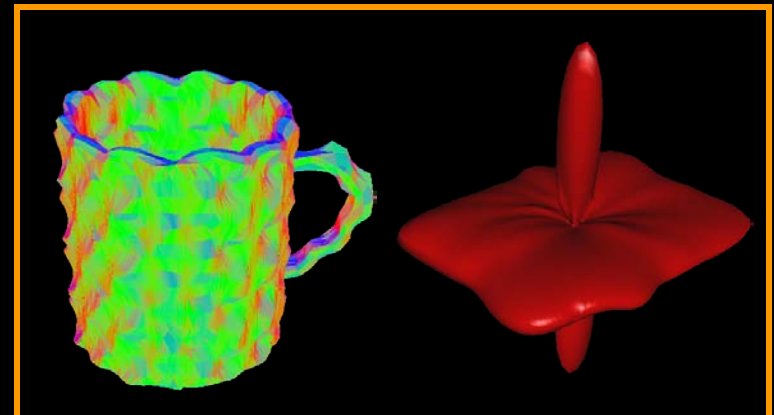
- Invertible for convex shapes
- Can be defined for most models
- 2D array of information

## Limitations:

- In general, shapes are not convex
- Normals are sensitive to noise



Initial Model

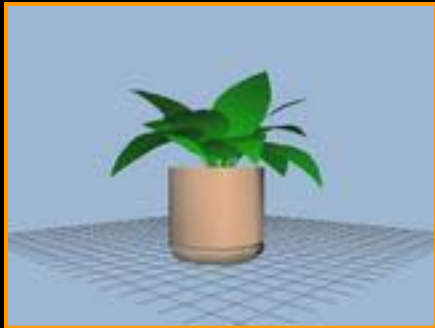


Noisy Model

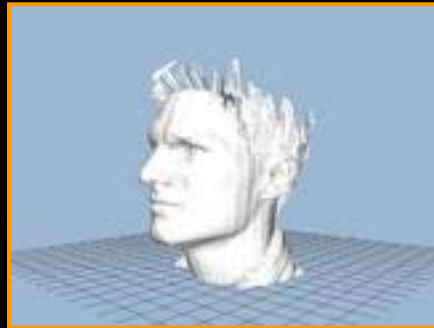
# Retrieval Results



## Princeton Shape Benchmark



51 potted plants



33 faces



15 desk chairs



22 dining chairs



100 humans



28 biplanes

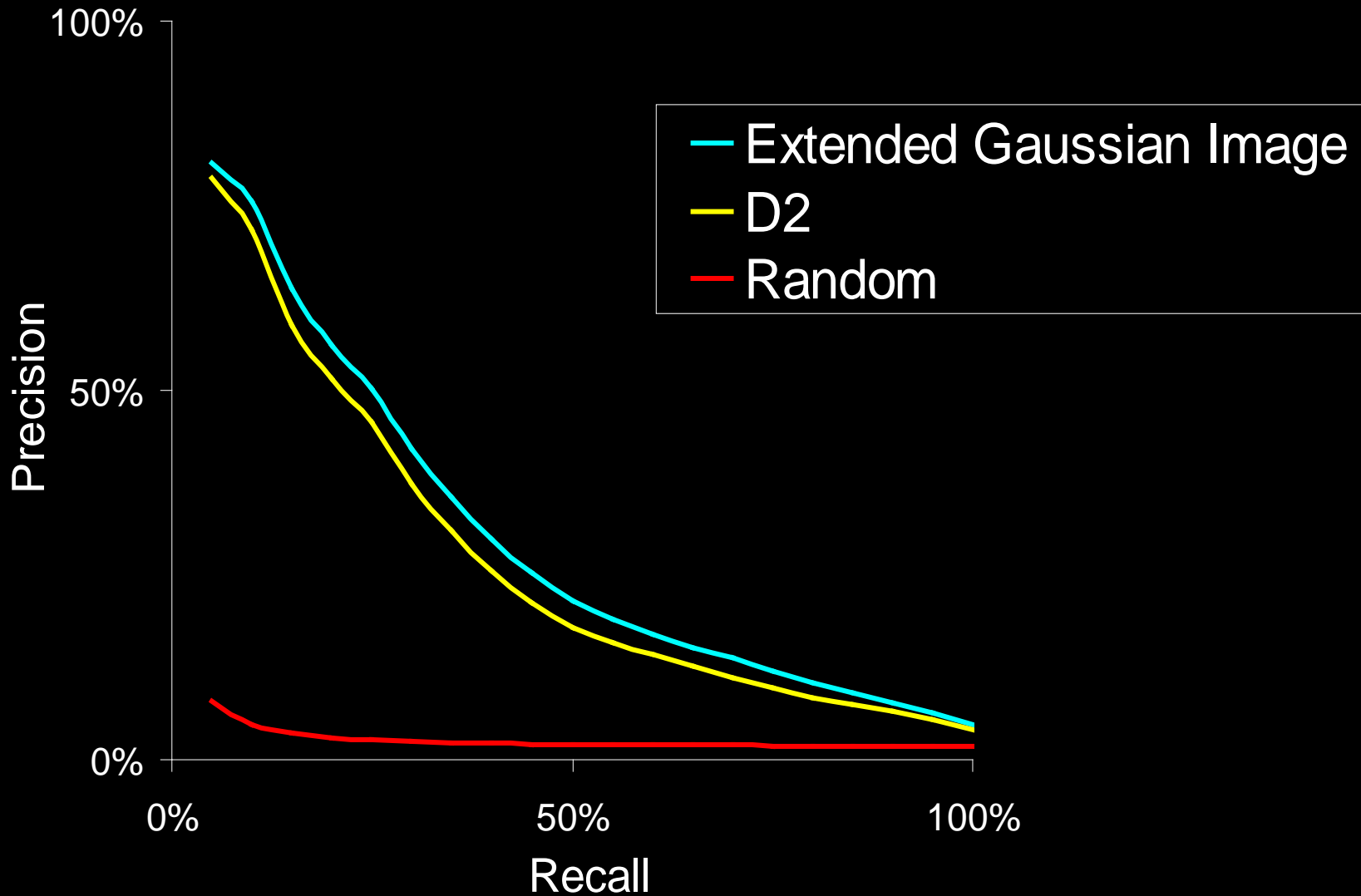


14 flying birds



11 ships

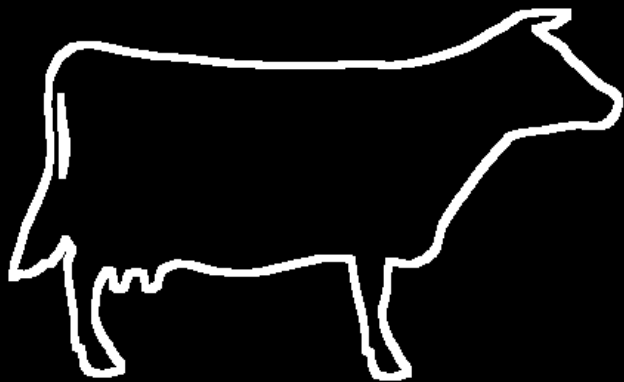
# Retrieval Results



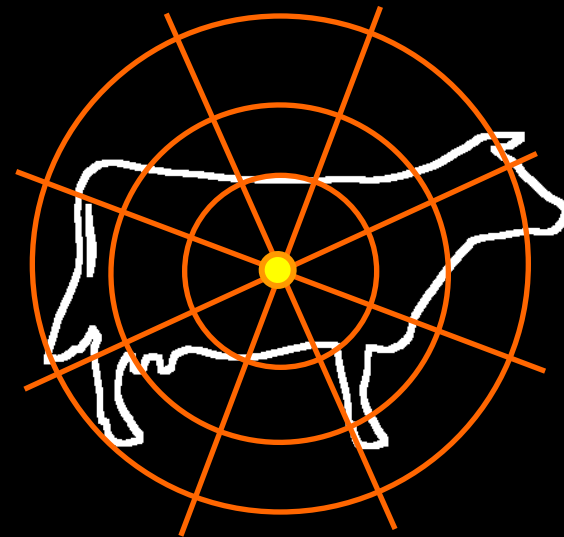
# Shape Histograms

[Ankerst *et al.*, 1999]

Shape descriptor stores a histogram of how much surface resides at different bins in space



Model



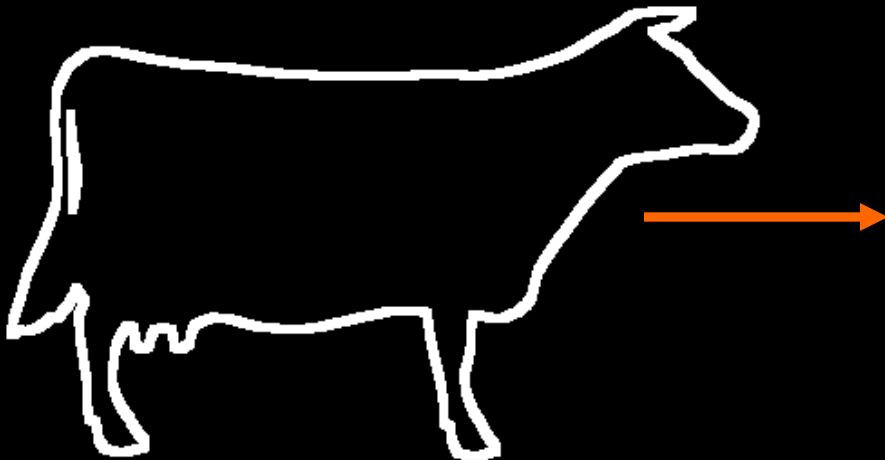
Shape Histogram  
(Sectors + Shells)



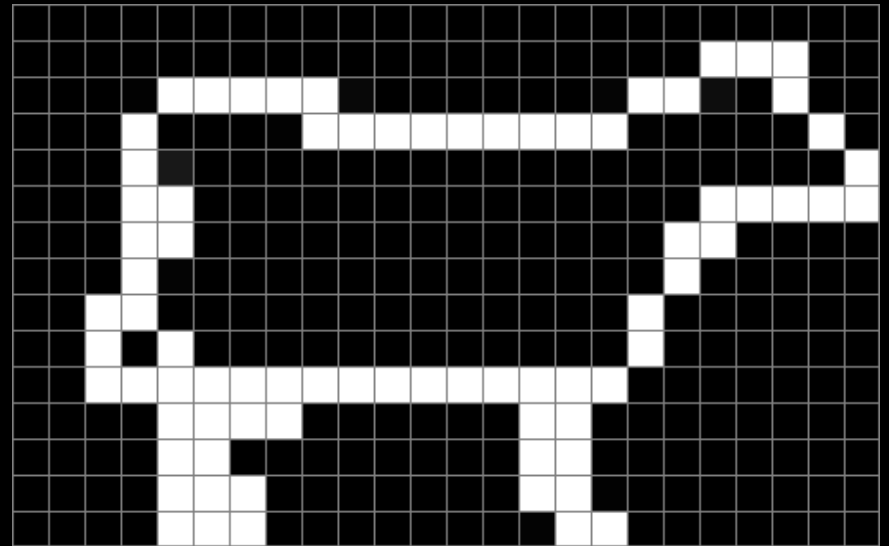
# Boundary Voxel Representation

Represent a model as the (anti-aliased) rasterization of its surface into a regular grid:

- A voxel has value 1 (or area of intersection) if it intersects the boundary
- A voxel has value 0 if it doesn't intersect



Model



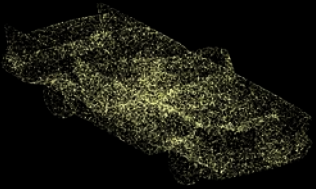
Voxel Grid

# Boundary Voxel Representation



## Properties:

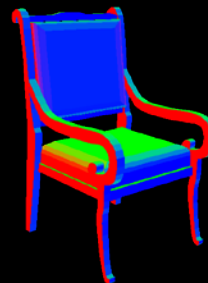
- Can be defined for any model
- Invertible
- 3D array of information



Point  
Clouds



Polygon  
Soups



Closed  
Meshes



Genus-0  
Meshes

Shape Spectrum

# Boundary Voxel Representation

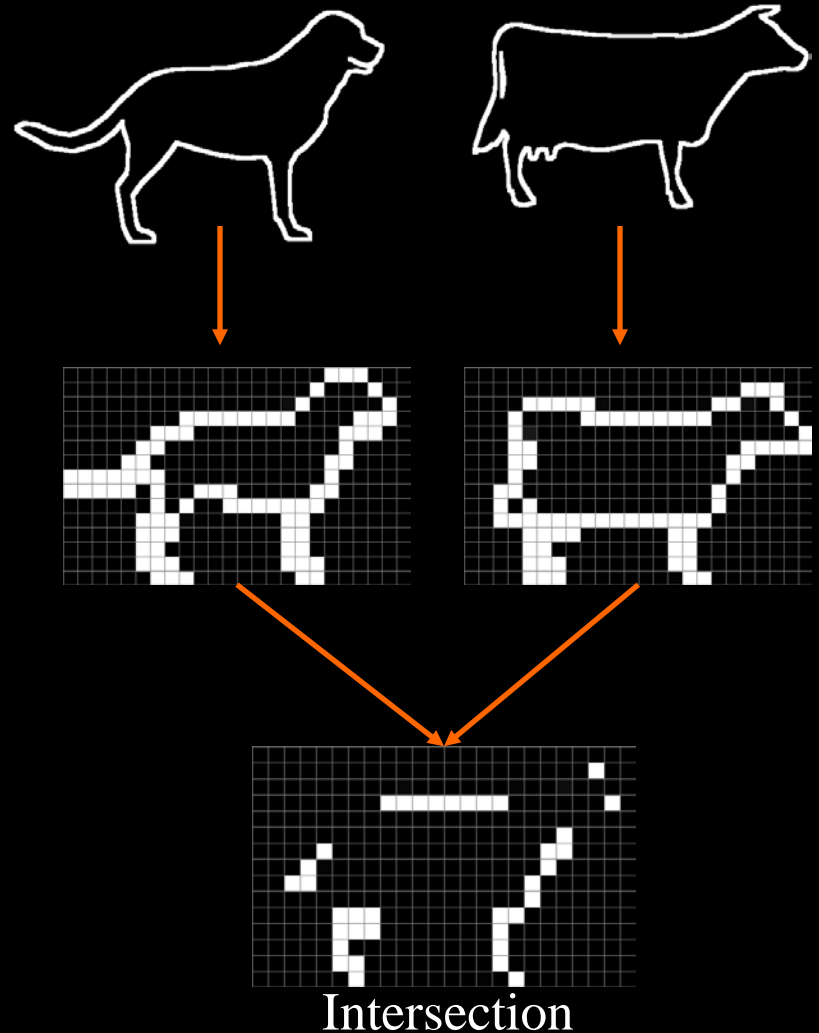


## Properties:

- Can be defined for any model
- Invertible
- 3D array of information

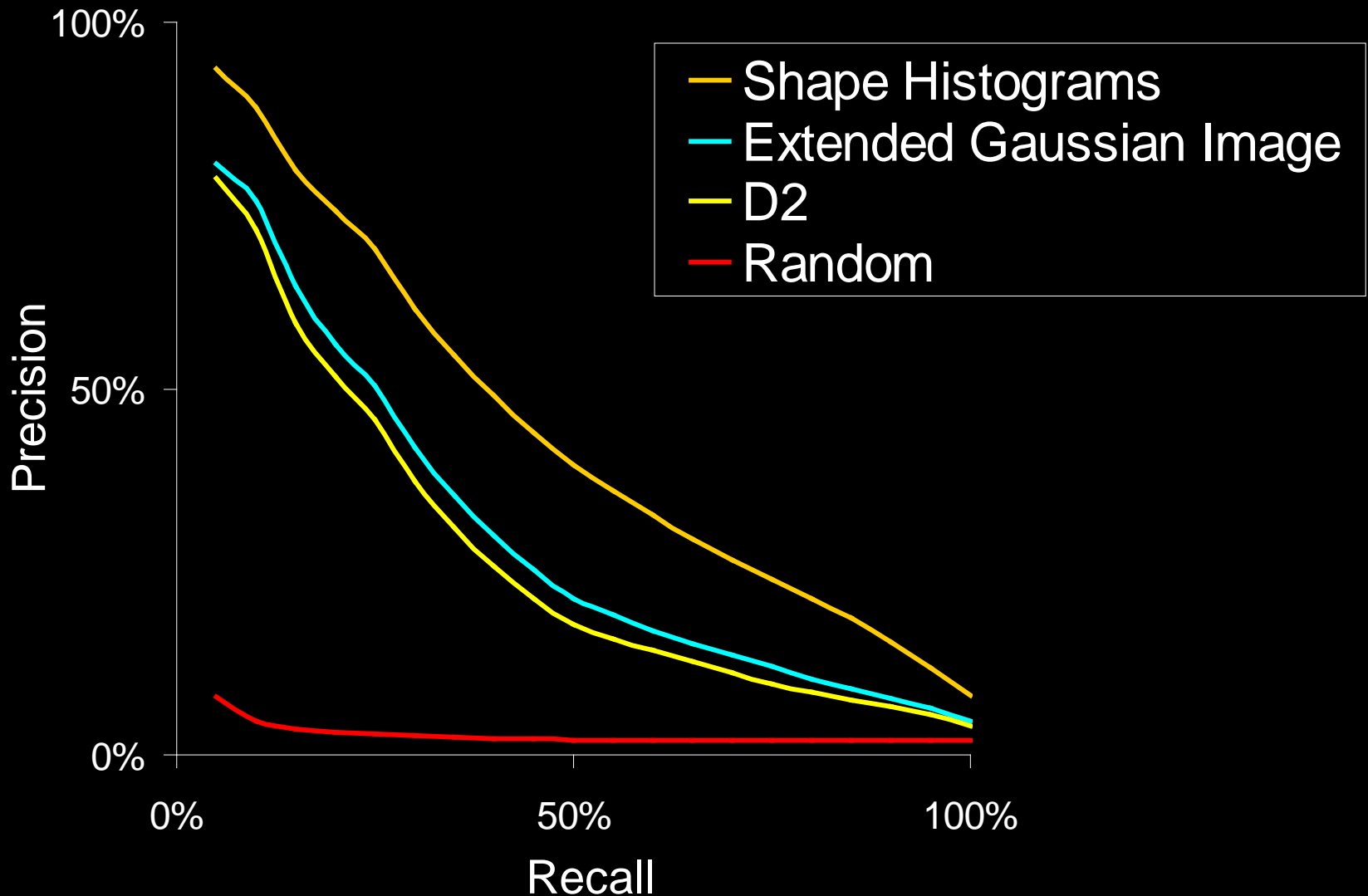
## Limitations:

- Difficult to match
  - If the resolution is too high:  
most voxels miss
  - If the resolution is too low:  
representation is too coarse





# Retrieval Results



# Histogram Representations

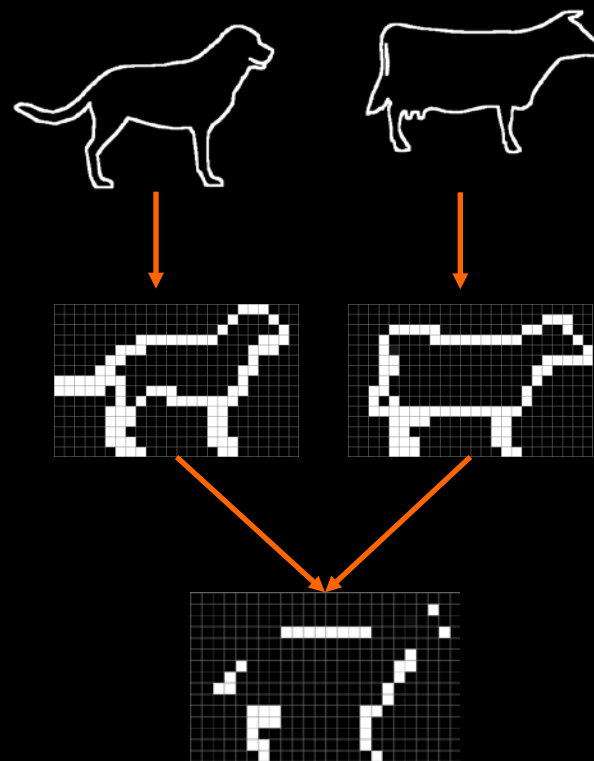


## Challenge:

- If shape properties are mapped to nearby bins, they will not be compared

## Solutions:

- Match across adjacent bins:  
Earth Mover's Distance
- Low-pass filter:  
Convolution with a Gaussian



# Earth Mover's distance

[Rubner *et al.* 1998]

Match by computing the minimal amount of work needed to transform one distribution into the other

Computing the distance:

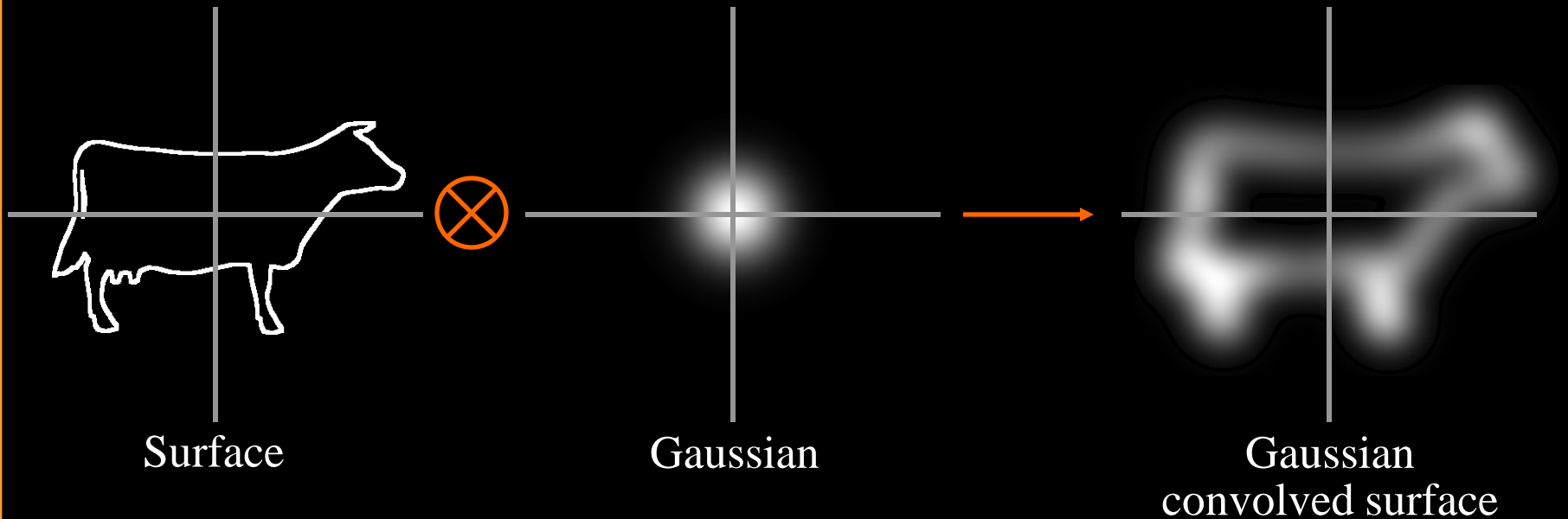
- For 1D histograms can use the CDF to compare efficiently
- In general, need to solve the transportation problem which is inefficient for large numbers of bins

# Convolving with a Gaussian



The value at a point is obtained by summing Gaussians distributed over the surface of the model

- ✓ Distributes the surface into adjacent bins
- ✗ Blurs the model, loses high frequency information

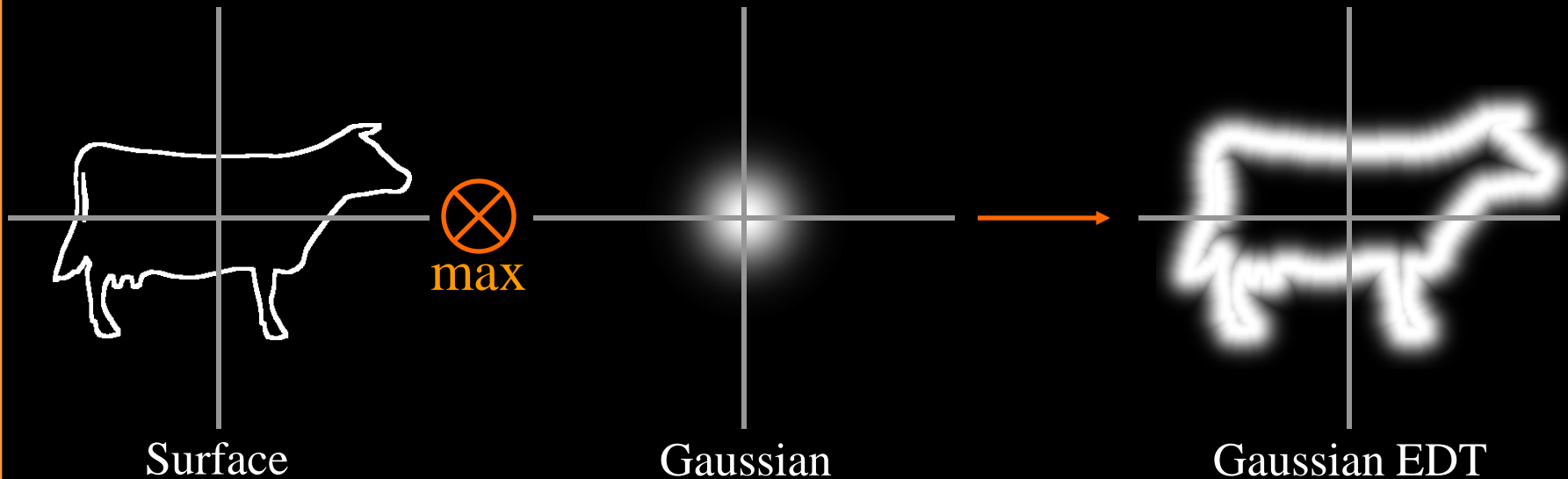


# Gaussian EDT

[Kazhdan *et al.*, 2003]

The value at a point is Gaussian applied to distance to closest point on the surface

- ✓ Distributes the surface into adjacent bins
- ✓ Maintains high-frequency information

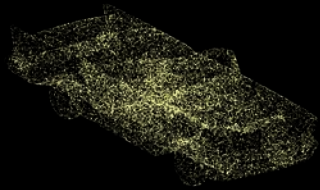


# Gaussian EDT



## Properties:

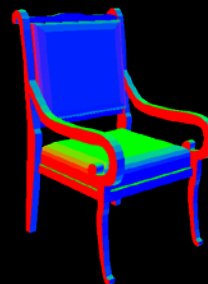
- Can be defined for any model
- Invertible
- 3D array of information
- Difference measures proximity between surfaces



Point  
Clouds



Polygon  
Soups



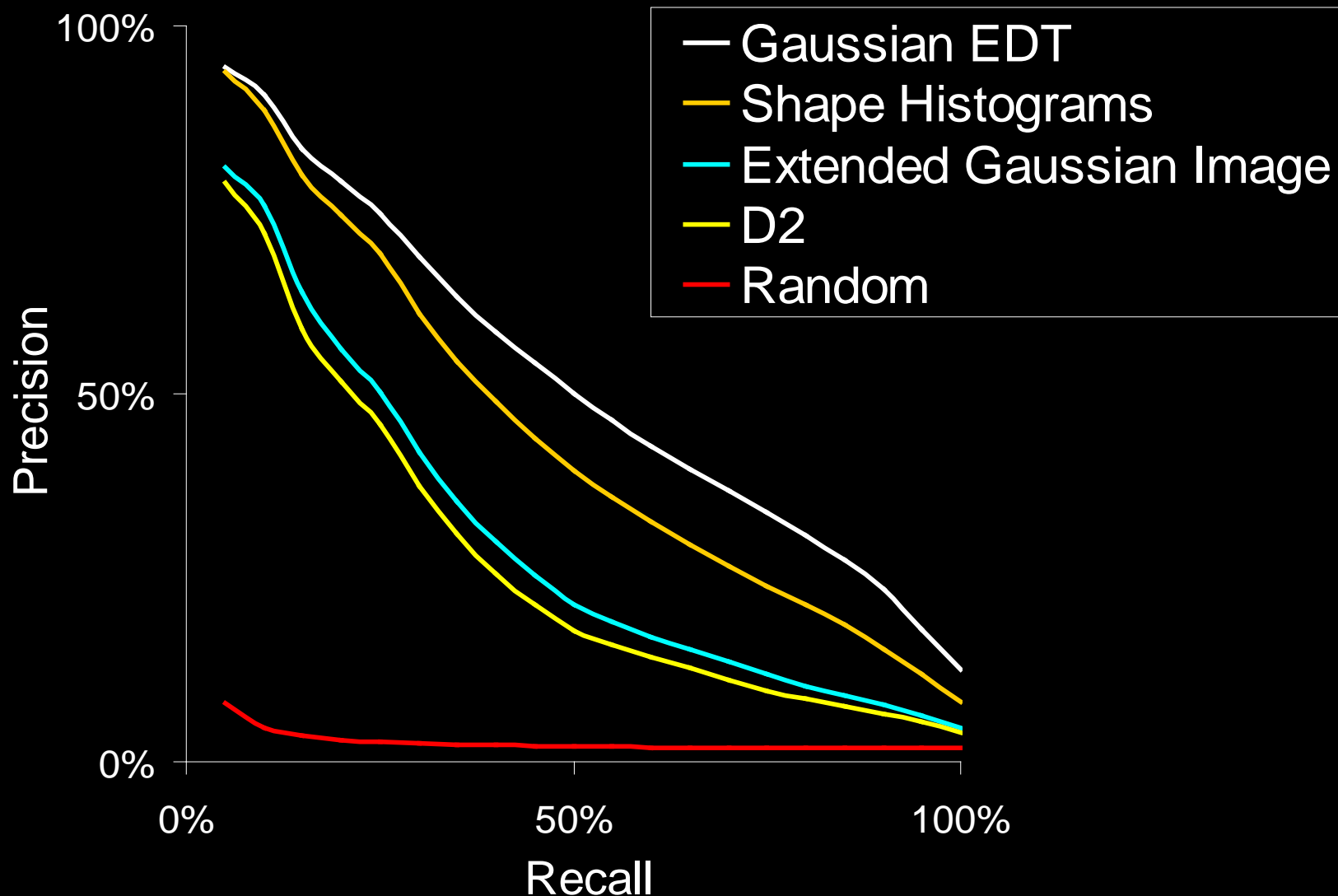
Closed  
Meshes



Genus-0  
Meshes

Shape Spectrum

# Retrieval Results



# Handling Transformations



Key difficulty:

locating objects under any rigid-body transformation

Approaches:

- **Exhaustive search:** try all possibilities
- **Invariance:** use descriptors that do not change under transformations
- **Normalization:** align objects to canonical coordinate frame





# Exhaustive Search

Search for the best aligning transformation:

- Compare at all alignments
- Match at the alignment for which models are closest



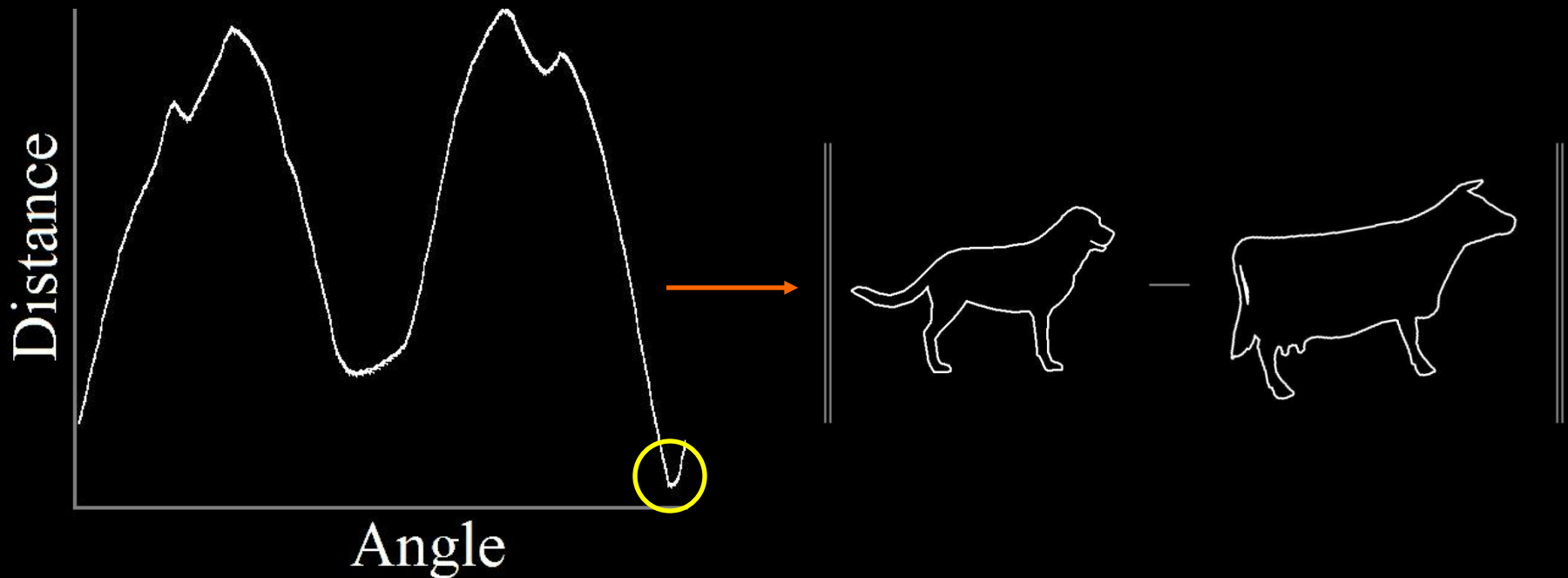
Exhaustive search for optimal rotation



# Exhaustive Search

Search for the best aligning transformation:

- Compare at all alignments
- Match at the alignment for which models are closest





# Exhaustive Search

Search for the best aligning transformation:

- Use signal processing for efficient correlation
- Represent model at many different transformations

Search for the best aligning transformation:

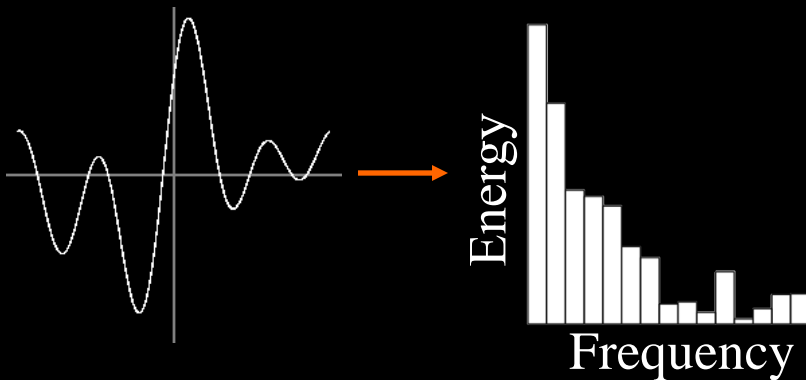
- Gives the correct answer
- Is hard to do efficiently



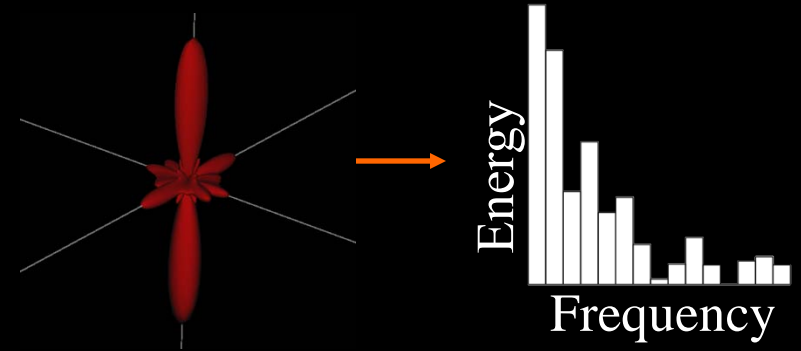
# Invariance

Represent a model with information that is independent of the transformation

- Power spectrum representation
  - Fourier Transform for translation and 2D rotations
  - Spherical Harmonic Transform for 3D rotations

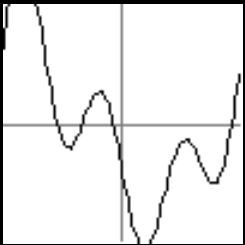


Circular Power Spectrum



Spherical Power Spectrum

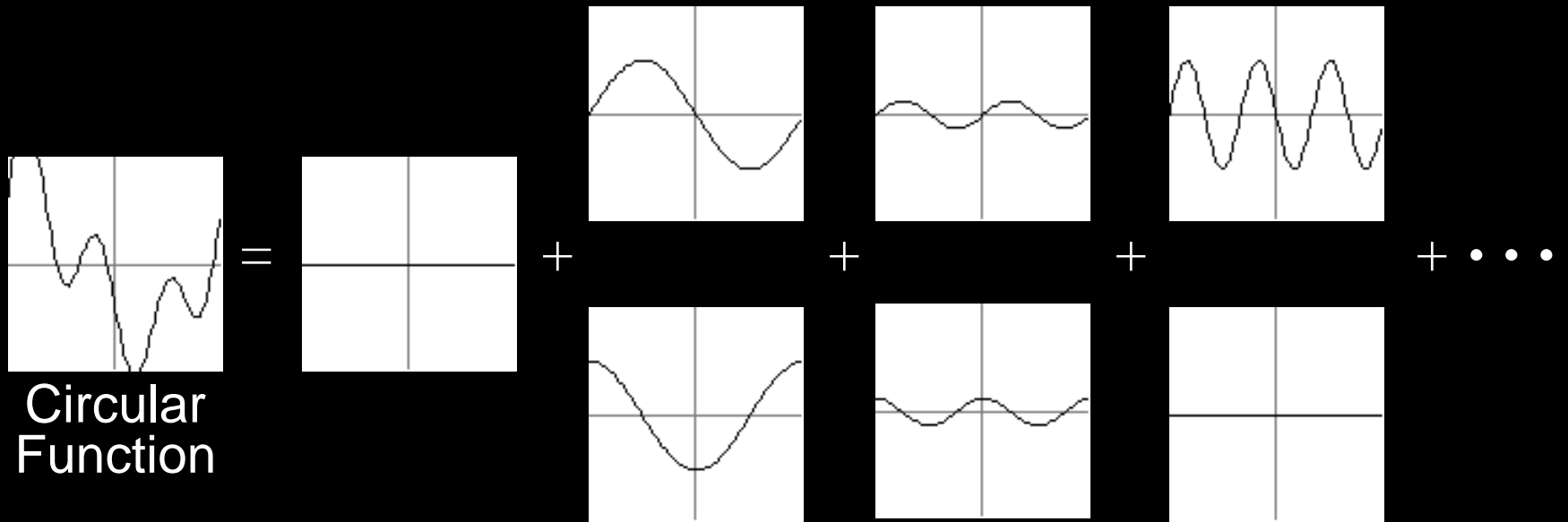
# Circular Power Spectrum



Circular  
Function



# Circular Power Spectrum

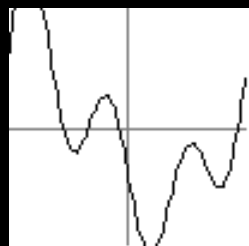


Circular Function

Cosine/Sine Decomposition

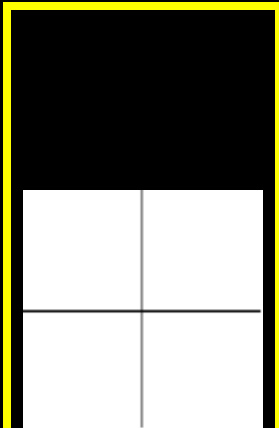


# Circular Power Spectrum

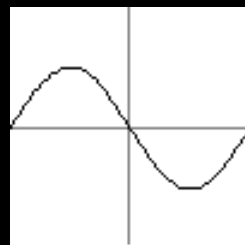


Circular Function

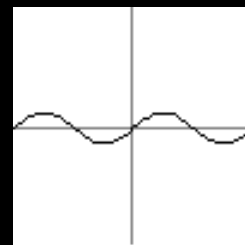
=



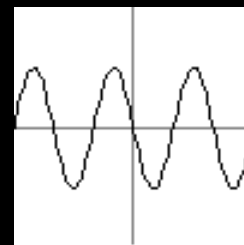
+



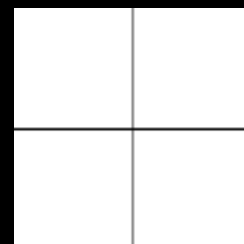
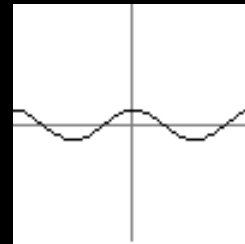
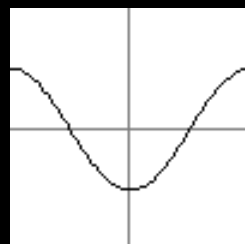
+



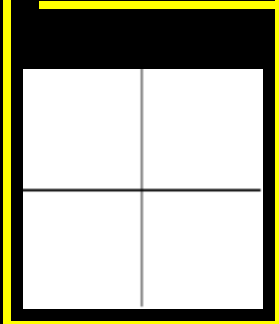
+



+ ...



=

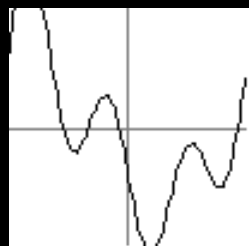


Constant

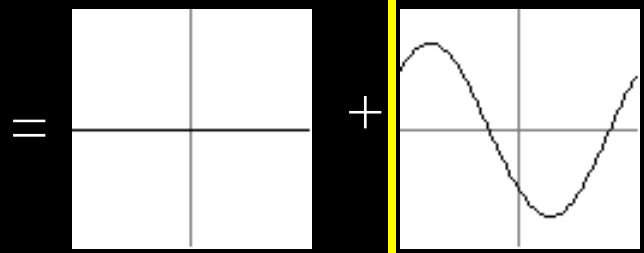
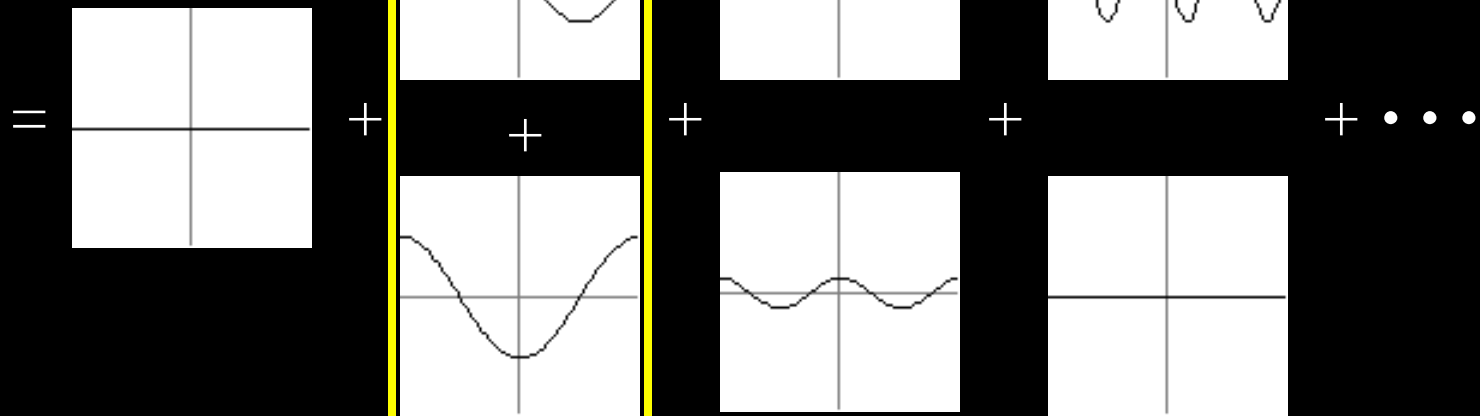
Frequency Decomposition



# Circular Power Spectrum



Circular Function



Constant

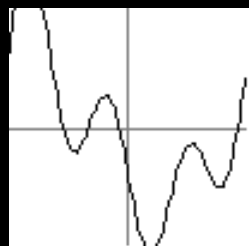
1<sup>st</sup> Order

Frequency Decomposition

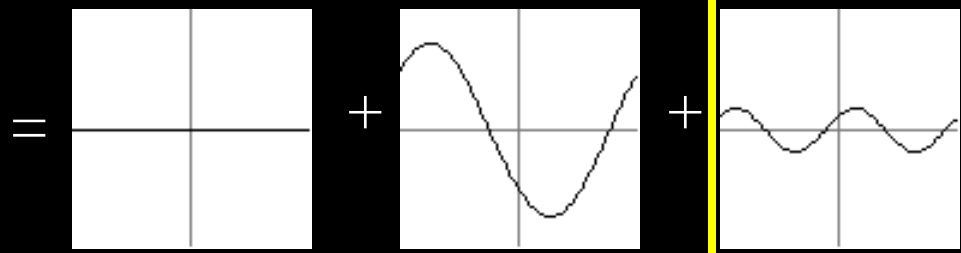
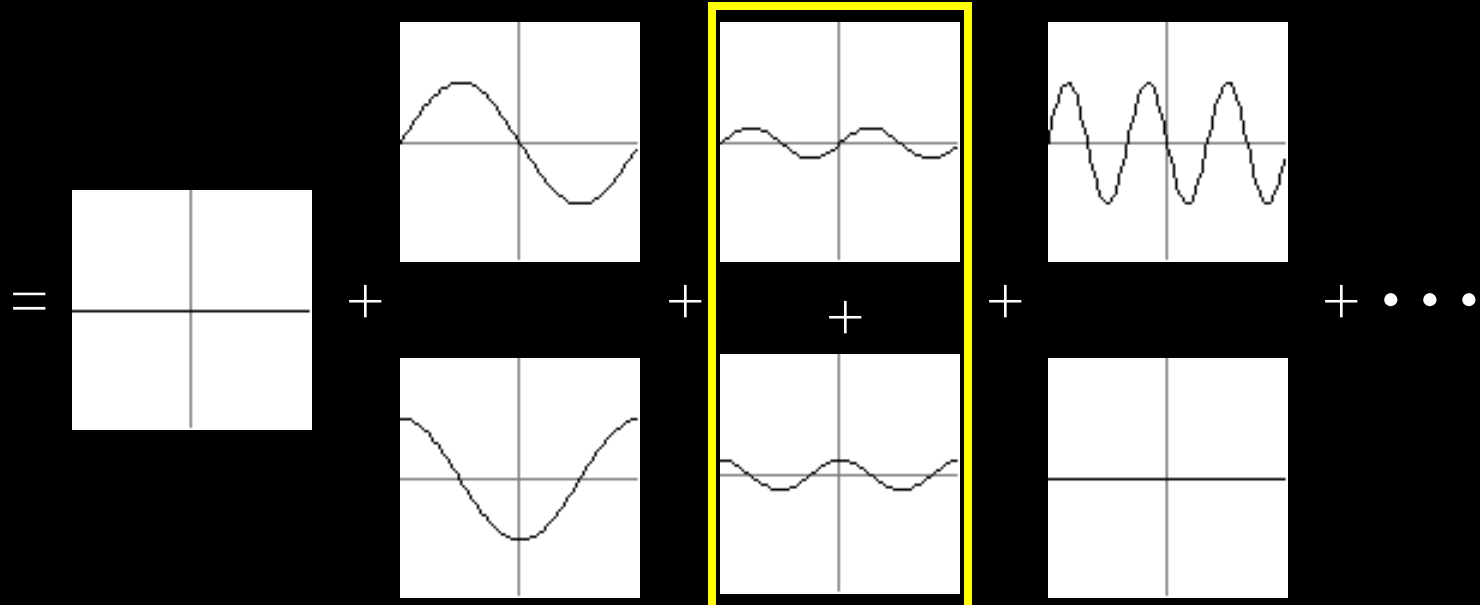




# Circular Power Spectrum



Circular Function



Constant

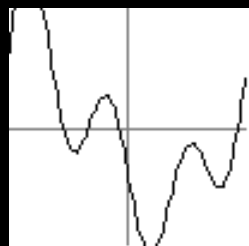
1<sup>st</sup> Order

2<sup>nd</sup> Order

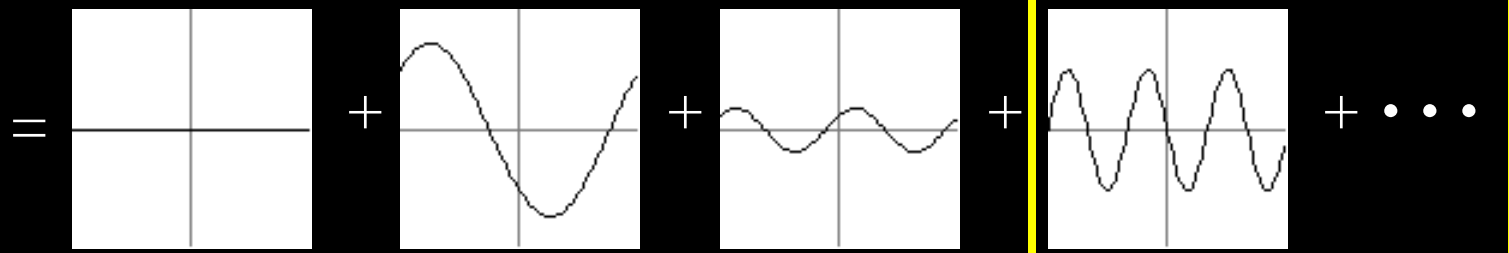
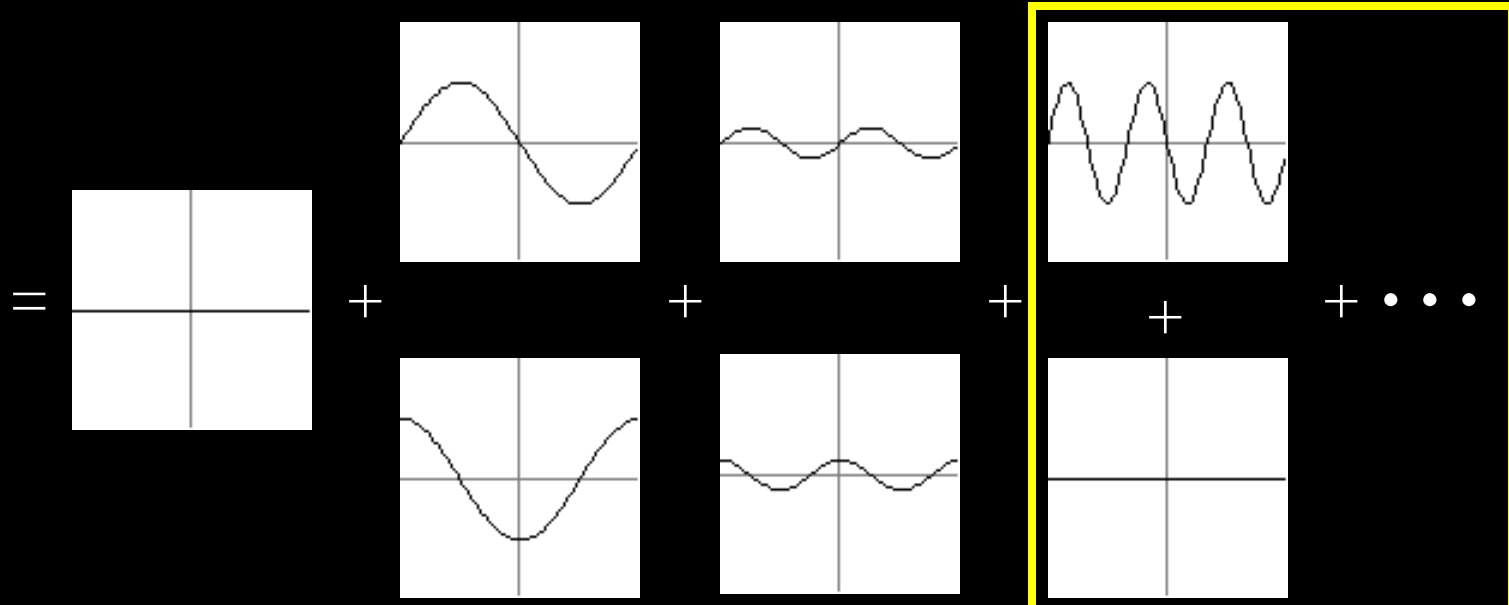
Frequency Decomposition



# Circular Power Spectrum



Circular Function



Constant

1<sup>st</sup> Order

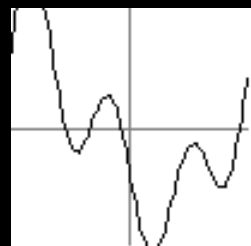
2<sup>nd</sup> Order

3<sup>rd</sup> Order

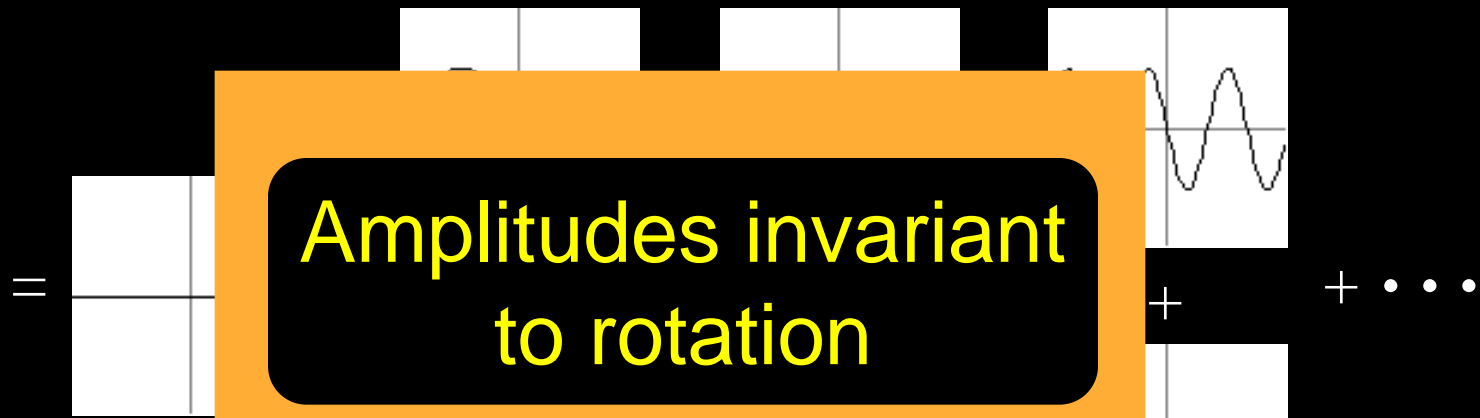
Frequency Decomposition



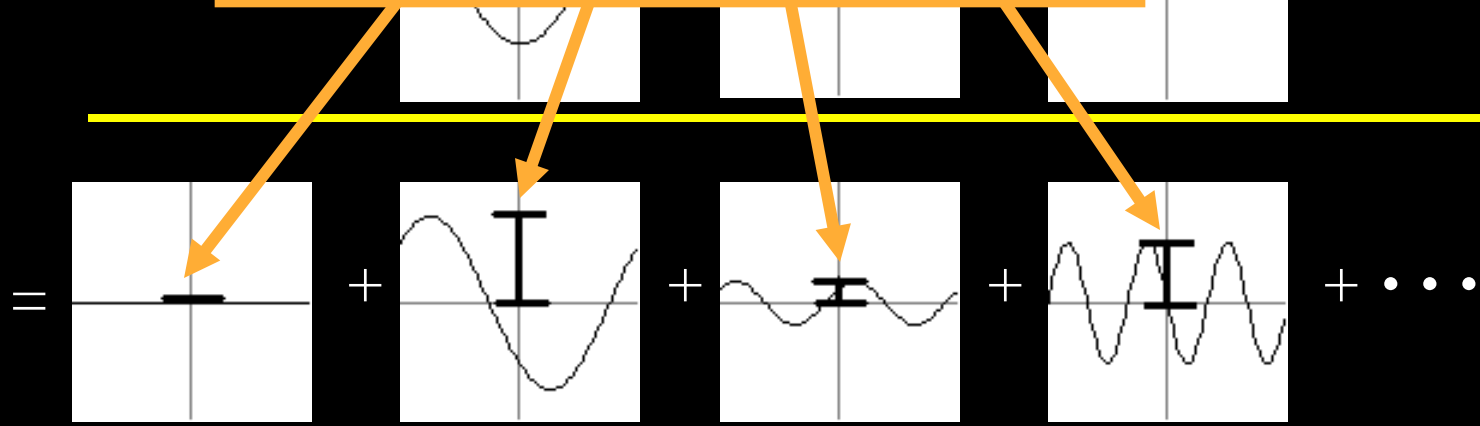
# Circular Power Spectrum



Circular Function



Amplitudes invariant to rotation



Constant

1<sup>st</sup> Order

2<sup>nd</sup> Order

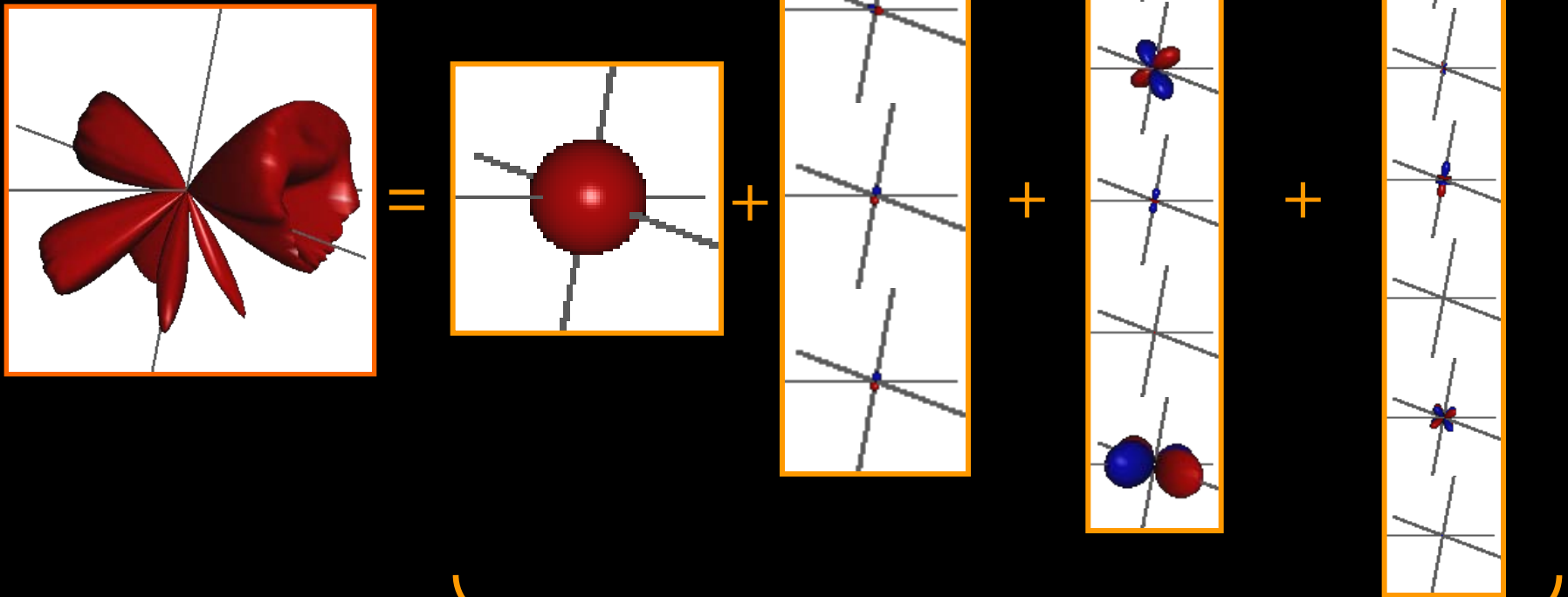
3<sup>rd</sup> Order

Frequency Decomposition

# Spherical Power Spectrum



Represent each spherical function as a sum of harmonic frequencies (orders)

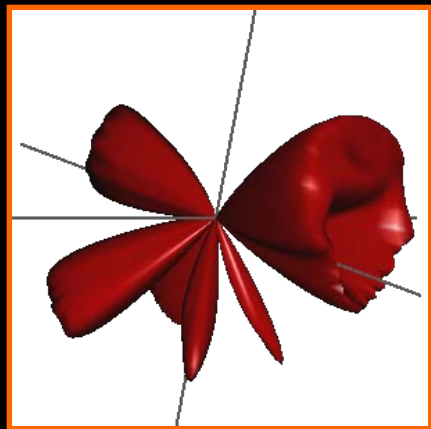


Harmonic Decomposition

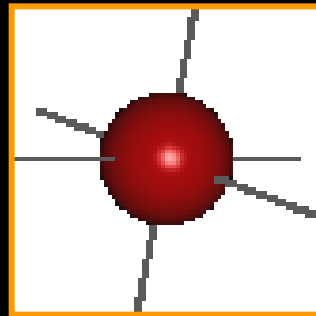
# Spherical Power Spectrum



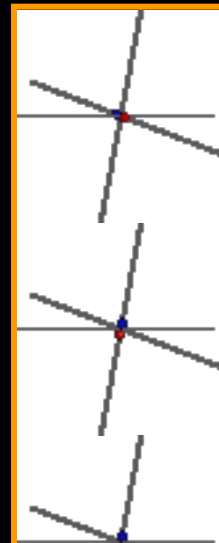
Represent each spherical function as a sum of harmonic frequencies (orders)



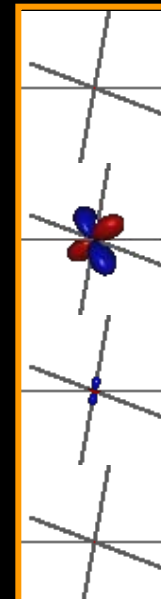
=



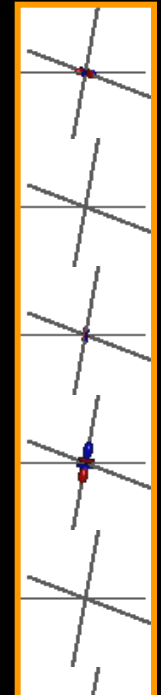
+



+



+

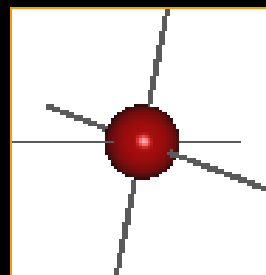


Constant

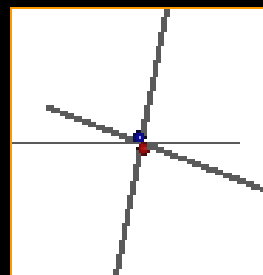
1<sup>st</sup> Order

2<sup>nd</sup> Order

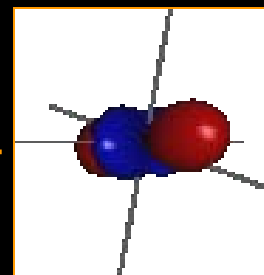
3<sup>rd</sup> Order



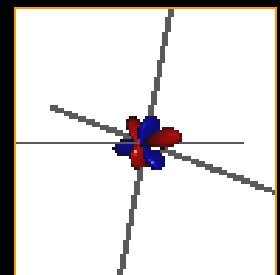
+



+



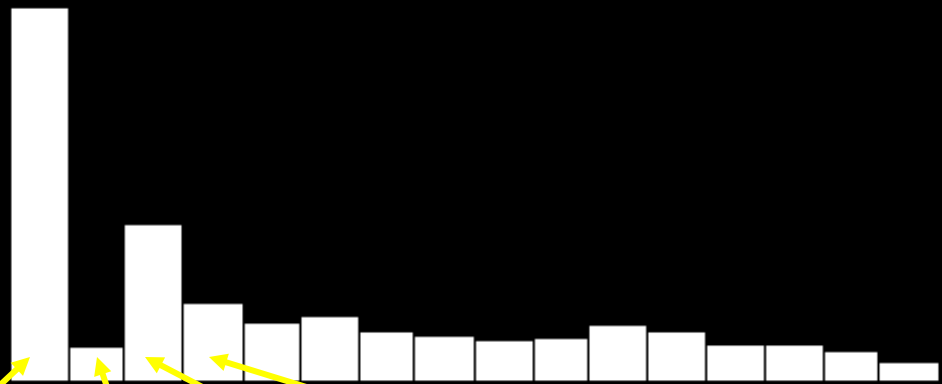
+



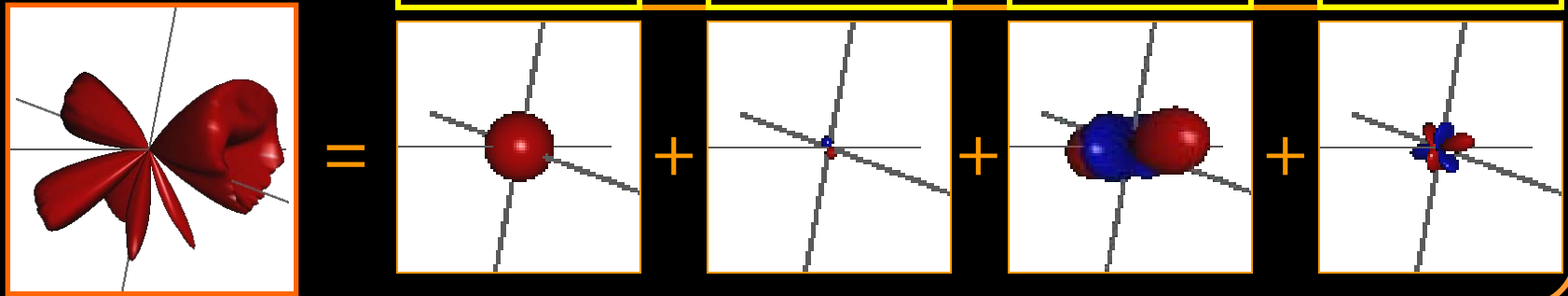


# Spherical Power Spectrum

Store "how much" ( $L_2$ -norm) of the shape resides at each frequency to get rotation invariant representation



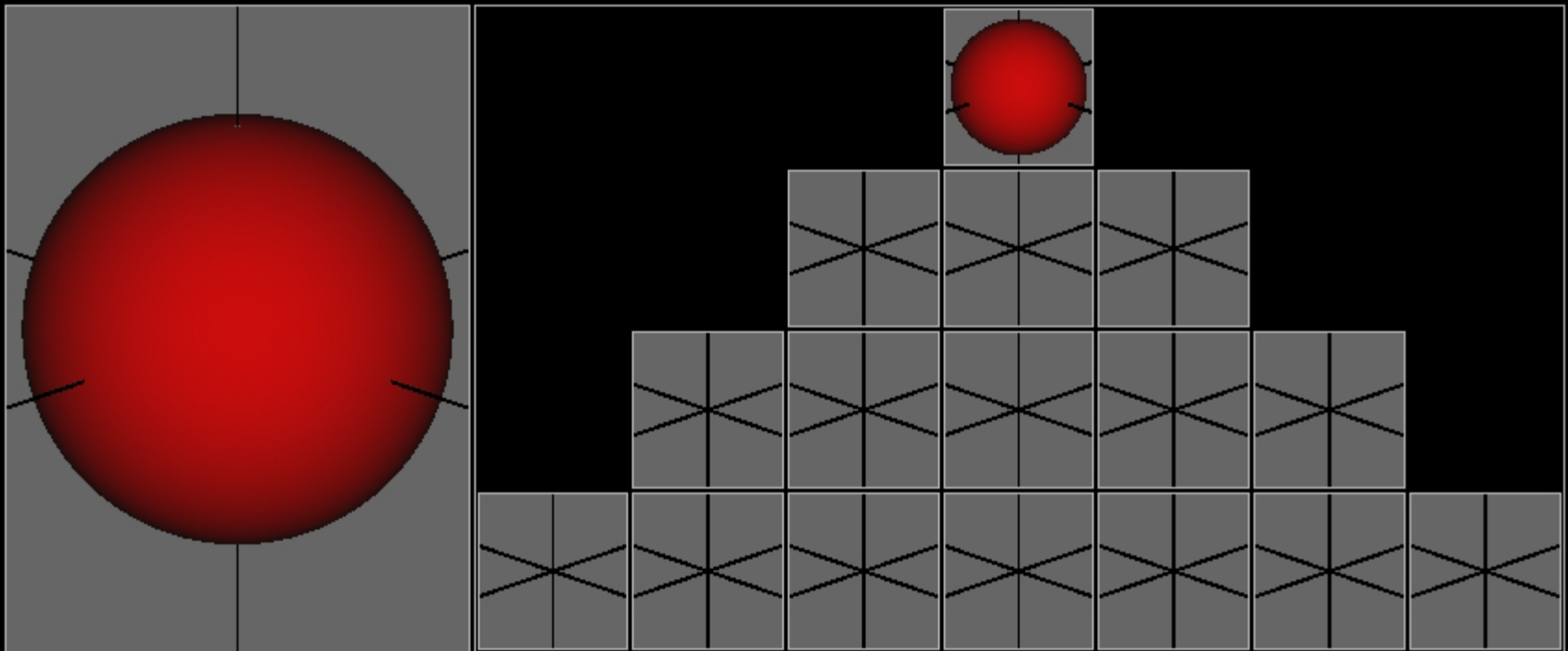
Constant      1<sup>st</sup> Order      2<sup>nd</sup> Order      3<sup>rd</sup> Order



# Invariant to transforms?



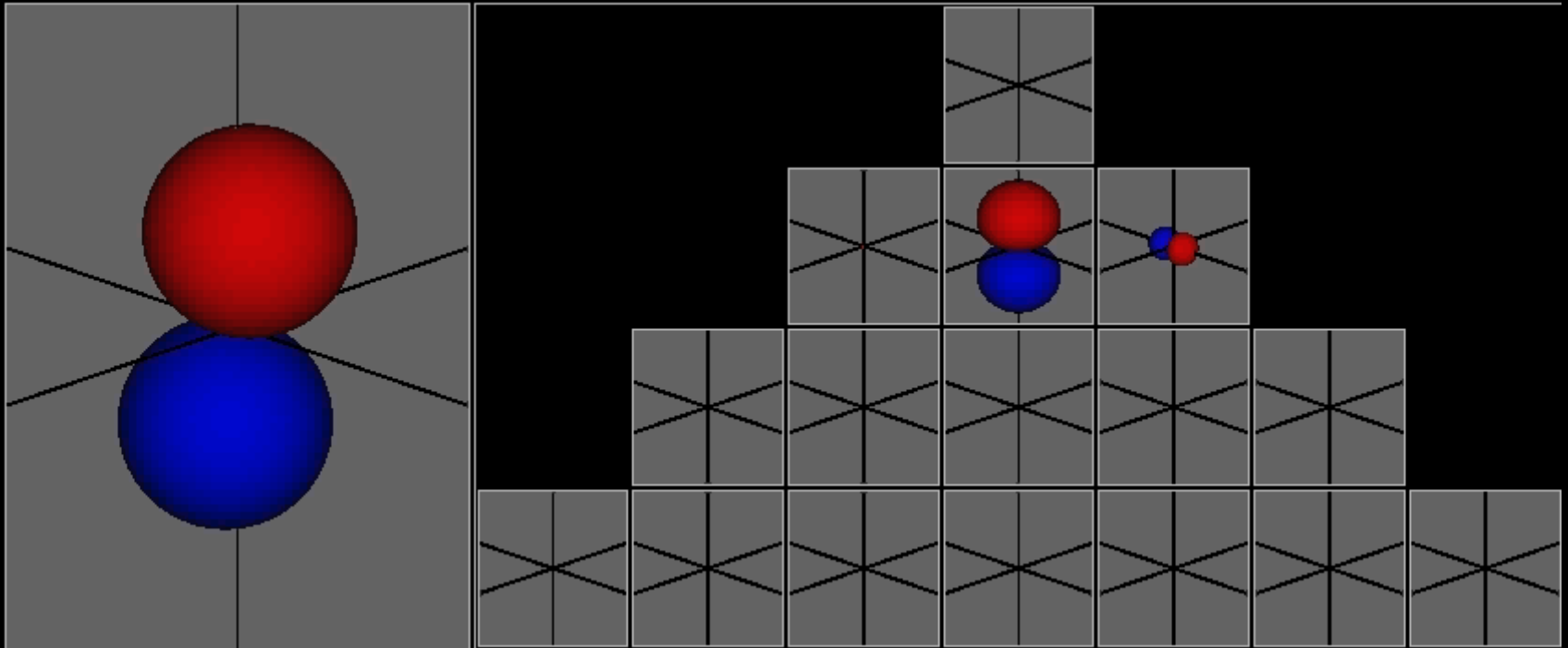
- Frequency subspaces are fixed by rotations:



# Invariant to transforms?



- Frequency subspaces are fixed by rotations:

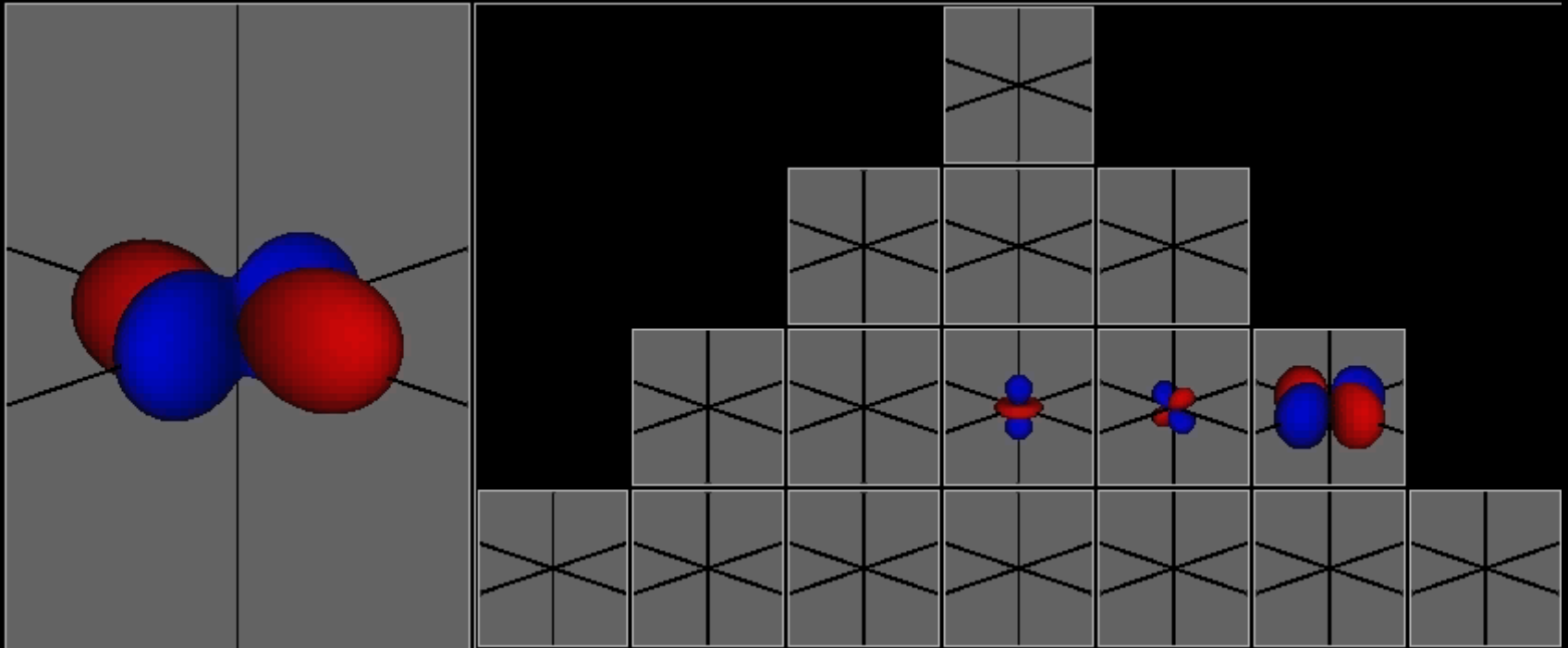






# Invariant to transforms?

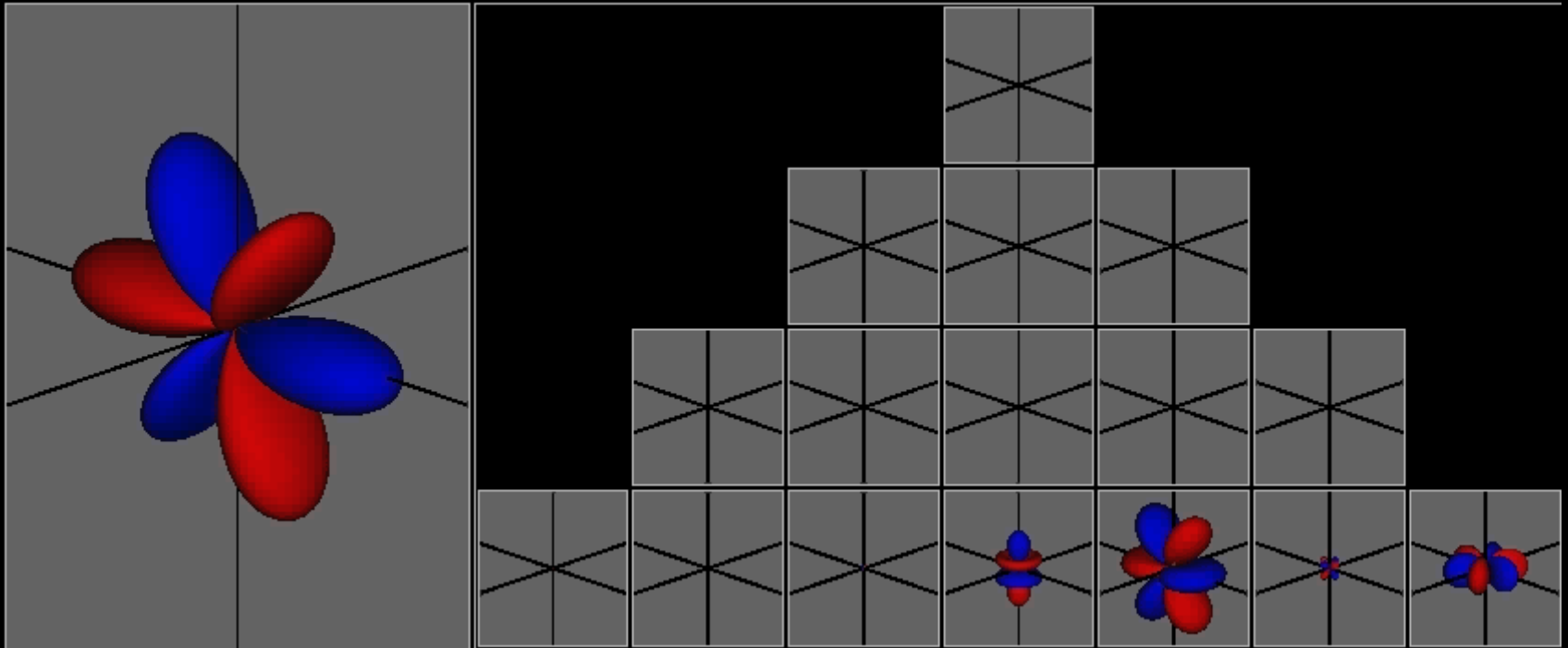
- Frequency subspaces are fixed by rotations:





# Invariant to transforms?

- Frequency subspaces are fixed by rotations:

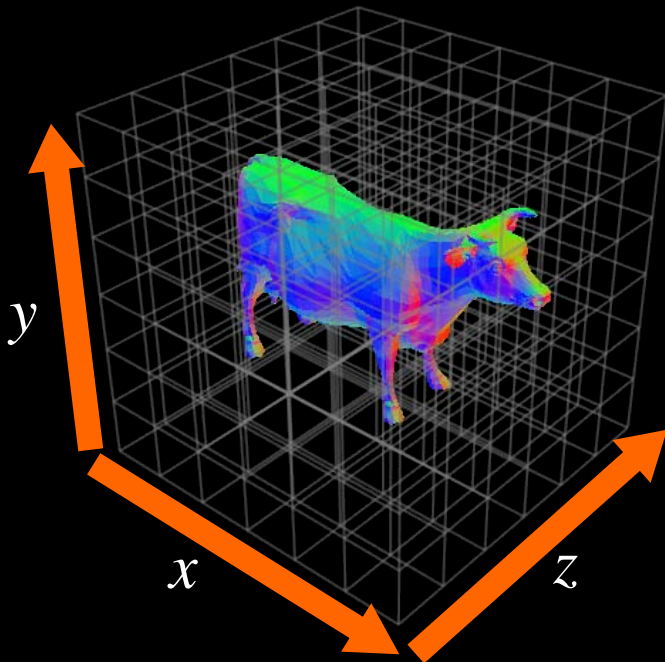




# Power Spectrum

Translation-invariance:

- Represent the model in a Cartesian coordinate system
- Compute the 3D Fourier transform
- Store the amplitudes of the frequency components



Cartesian Coordinates

$$f(x, y, z) = \sum_{l, m, n} f_{l, m, n} e^{i(lx + my + zn)}$$

$$\{ \| f_{l, m, n} \| \}_{l, m, n}$$

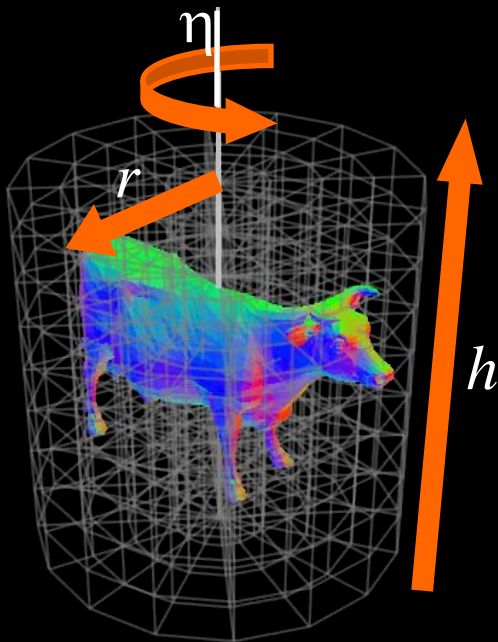
Translation Invariant  
Representation



# Power Spectrum

Single axis rotation-invariance:

- Represent the model in a cylindrical coordinate system
- Compute the Fourier transform in the angular direction
- Store the amplitudes of the frequency components



Cylindrical Coordinates

$$f(r, h, \theta) = \sum_k f_k(r, h) e^{i(k\theta)}$$

$$\{\|f_k(r, h)\|\}_k$$

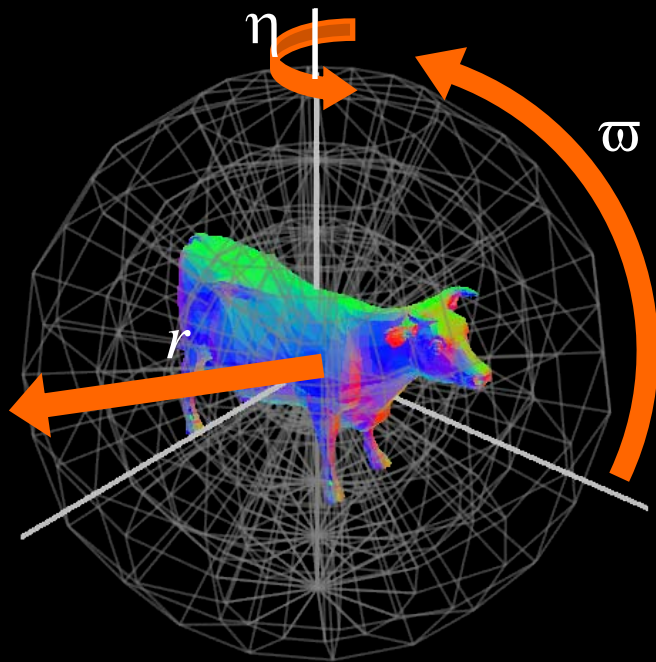
Rotation Invariant  
Representation



# Power Spectrum

Full rotation-invariance:

- Represent the model in a spherical coordinate system
- Compute the spherical harmonic transform
- Store the amplitudes of the frequency components



Spherical Coordinates

$$f(r, \theta, \phi) = \sum_l \sum_{|m| \leq l} f_{l,m}(r) Y_l^m(\theta, \phi)$$

$$\left\{ \sqrt{\sum_{|m| \leq l} \|f_l^m(r)\|^2} \right\}_l$$

Rotation Invariant  
Representation

# Power Spectrum



## Power spectrum representations

- Are invariant to transformations
- Give a lower bound for the best match
- Tend to discard too much information

Translation invariant:  $n^3$  data  $\rightarrow$   $n^3/2$  data

Single-axis rotation invariant:  $n^3$  data  $\rightarrow$   $n^3/2$  data

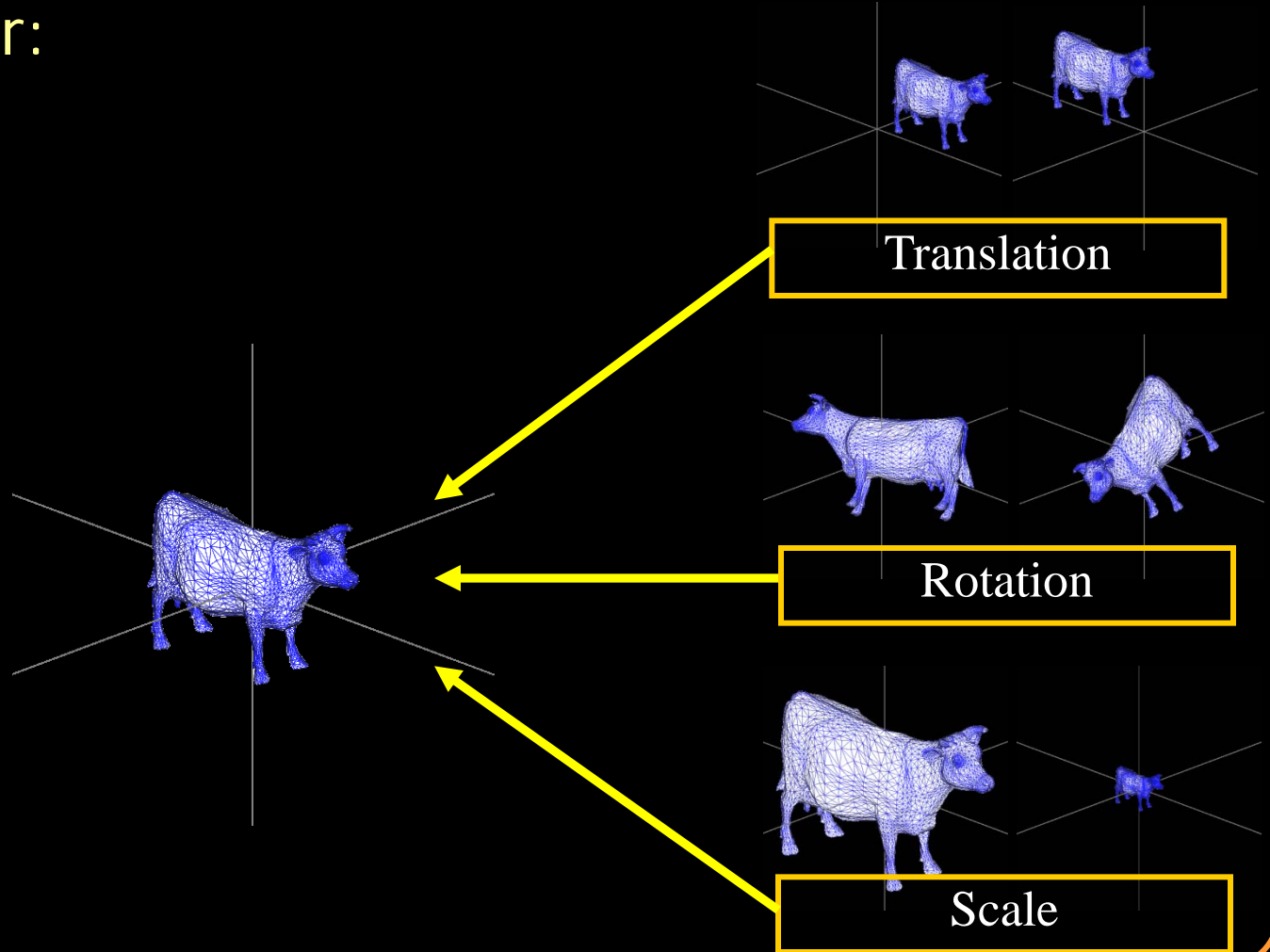
Full rotation invariant:  $n^3$  data  $\rightarrow$   $n^2$  data



# Normalization

Place a model into a canonical coordinate frame by normalizing for:

- translation
- scale
- rotation

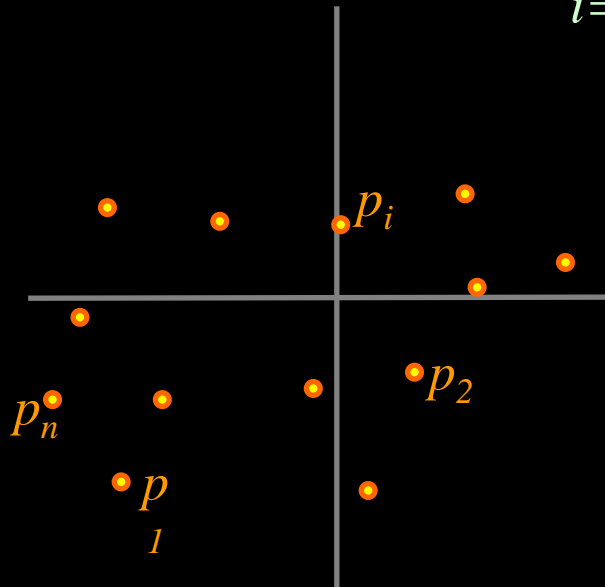


# Alignment of Point Sets

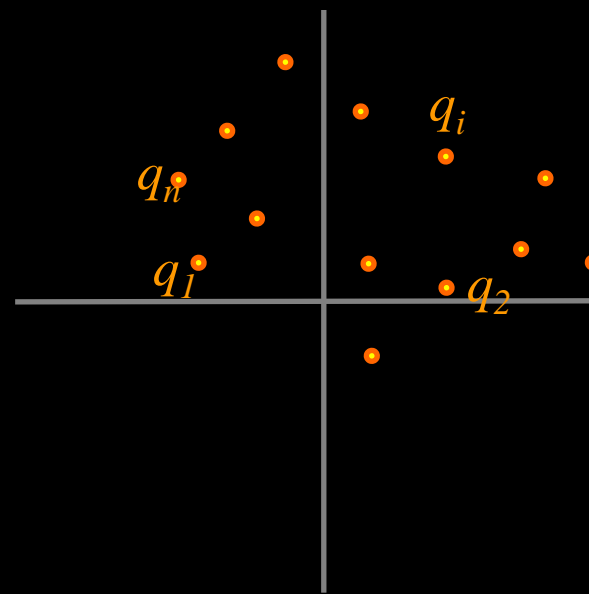
[Horn *et al.*, 1988]

Given two point sets  $P = \{p_1, \dots, p_n\}$  and  $Q = \{q_1, \dots, q_n\}$ , what is the transformation  $T$  minimizing the sum of squared distances:

$$d(P, Q) = \sum_{i=1}^n \|p_i - T(q_i)\|^2$$



Point set  $P$



Point set  $Q$



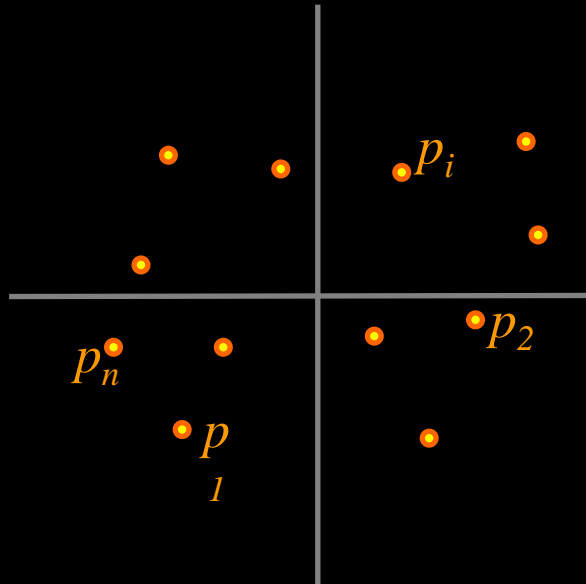
# Alignment of Point Sets

[Horn *et al.*, 1988]

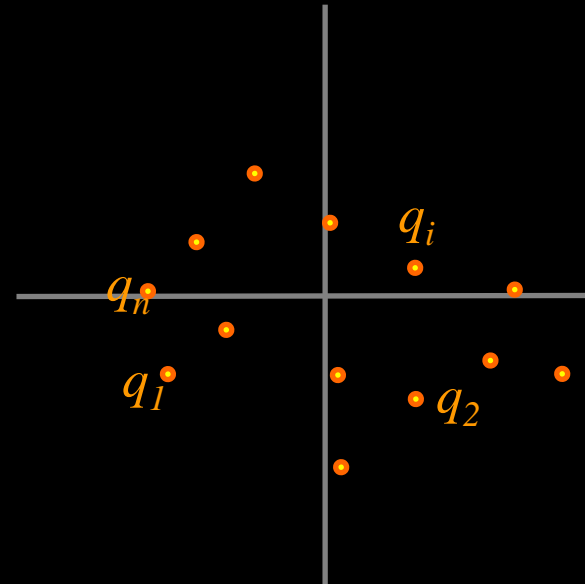
## Translation

- Align the models so that their center of mass is at the origin.

$$\sum_{i=1}^n p_i = 0 \quad \text{and} \quad \sum_{i=1}^n q_i = 0$$



Translated point set  $P$



Translated point set  $Q$

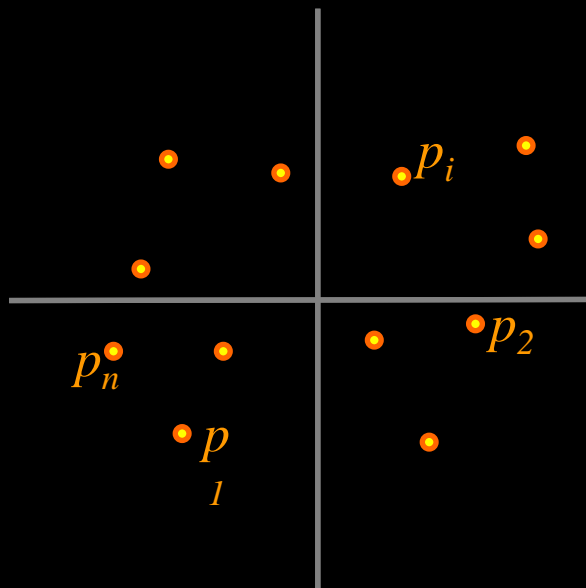
# Alignment of Point Sets

[Horn *et al.*, 1988]

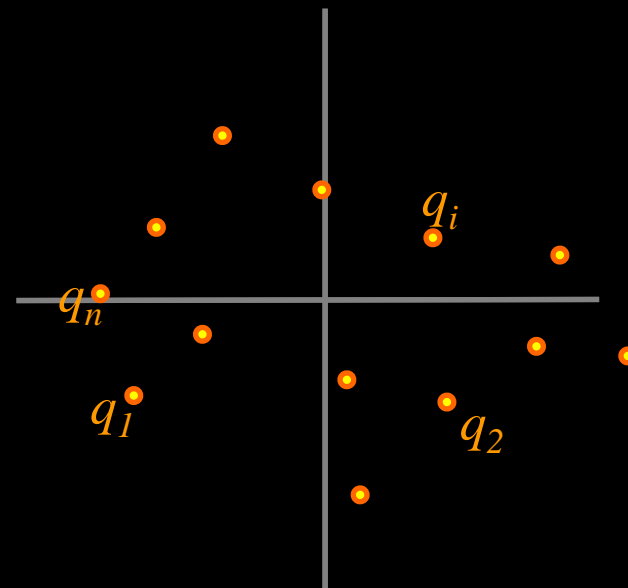
## Scale

- Align the models so that their mean variance is 1.

$$\sum_{i=1}^n \|p_i\|^2 = 1 \quad \text{and} \quad \sum_{i=1}^n \|q_i\|^2 = 1$$



Scaled point set  $P$



Scaled point set  $Q$

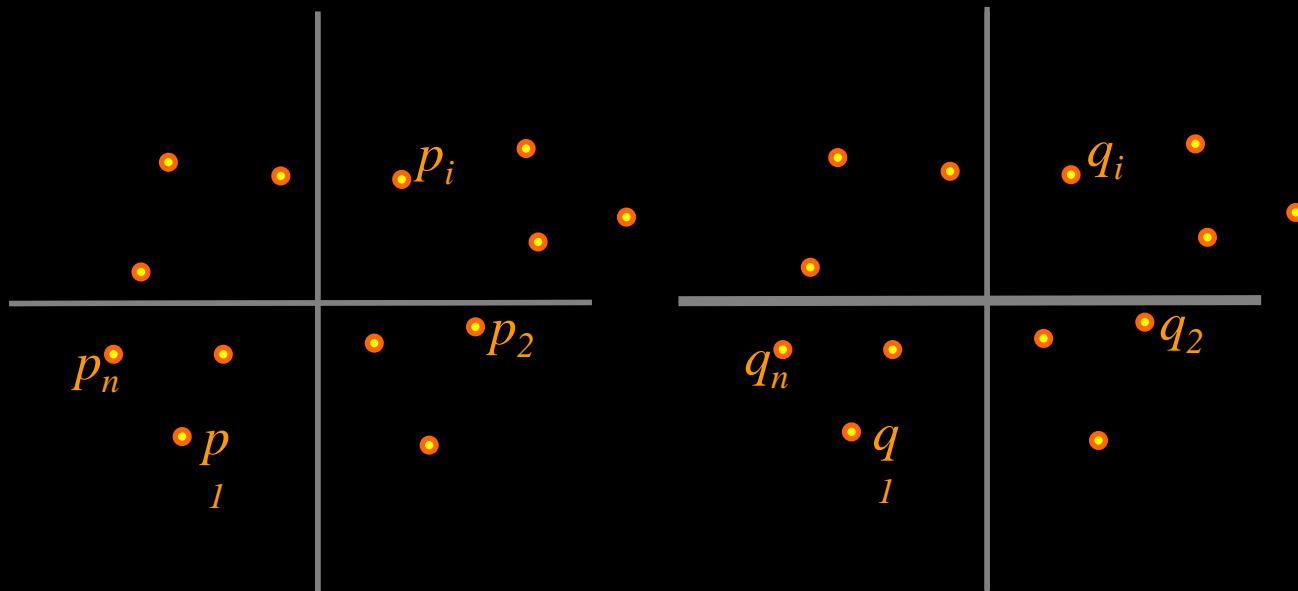
# Alignment of Point Sets

[Horn *et al.*, 1988]

## Rotation

- SVD on cross covariance matrix:

$$M = (p_1 | \dots | p_n) \cdot (q_1 | \dots | q_n)^T$$



Rotationally aligned point sets  $P$  and  $Q$



# Normalization

Place a model into a canonical coordinate frame:

- Translation: center of mass

$$\sum_{i=1}^n p_i = 0 \quad \text{and} \quad \sum_{i=1}^n q_i = 0$$

Can be done on a per-model basis

- Scale: mean variance

$$\sum_{i=1}^n \|p_i\|^2 = 1 \quad \text{and} \quad \sum_{i=1}^n \|q_i\|^2 = 1$$

Can be done on a per-model basis

- Rotation: SVD on cross covariance matrix

$$M = (p_1 | \dots | p_n) \cdot (q_1 | \dots | q_n)^t$$

Need to know the correspondences between models



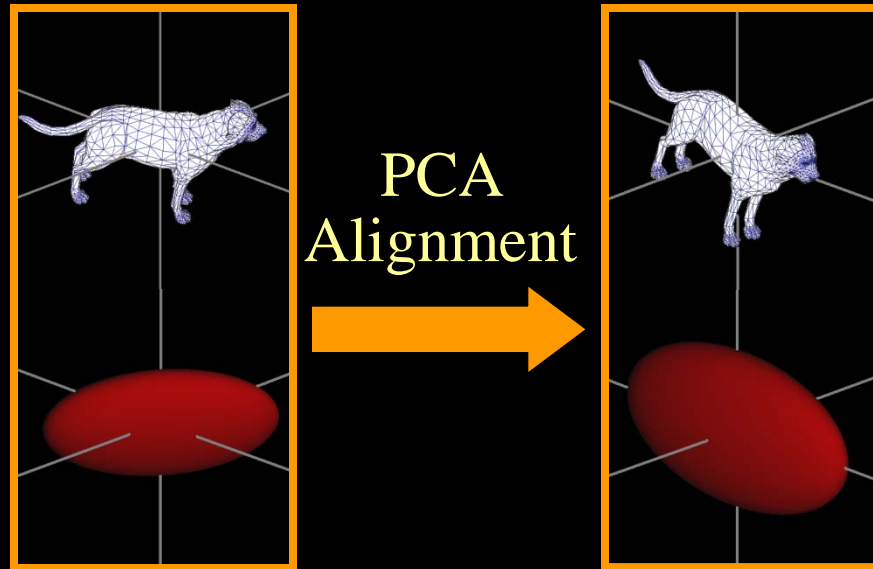
# Rotation

Challenge:

- We want to normalize for rotation on a per-model basis

Solution:

- Align the model so that the principal axes align with the coordinate axes





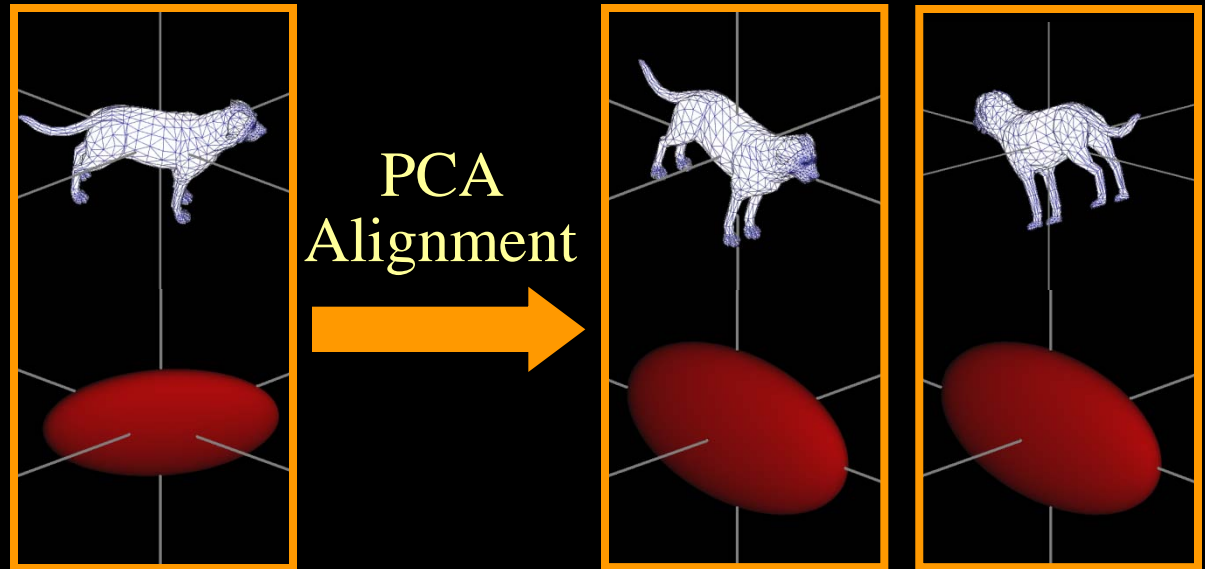
# Rotation

Challenge:

- We want to normalize for rotation on a per-model basis

Solution:

- Align the model so that the principal axes align with the coordinate axes



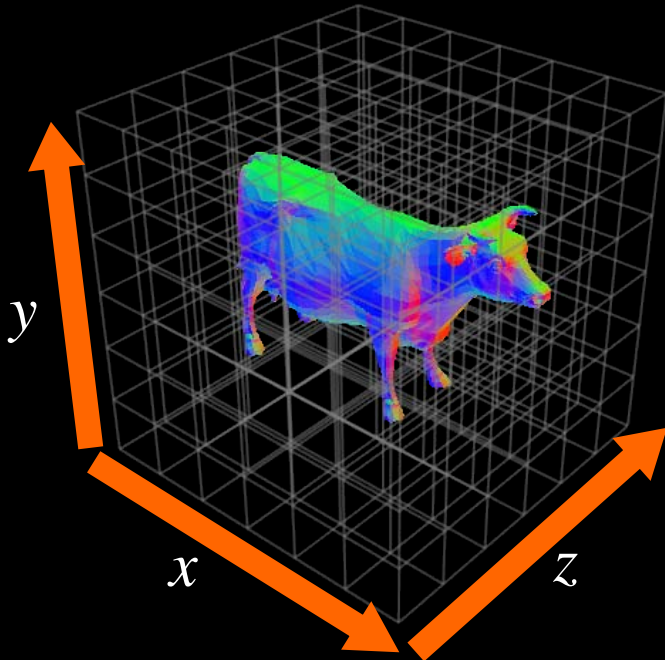
Directions of the axes are ambiguous



# Normalization (PCA)

PCA defines a coordinate frame up to reflection in the coordinate axes.

- Make descriptor invariant to the eight reflections
  - Reflections fix the cosine term
  - Reflections multiply the sine term by -1



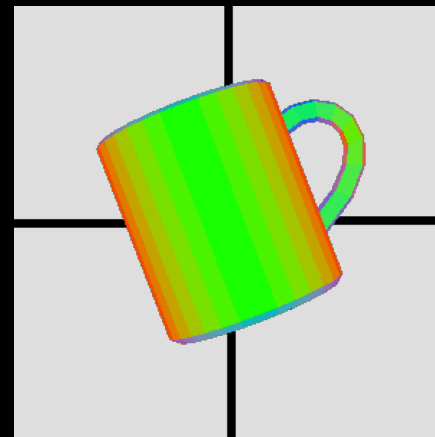
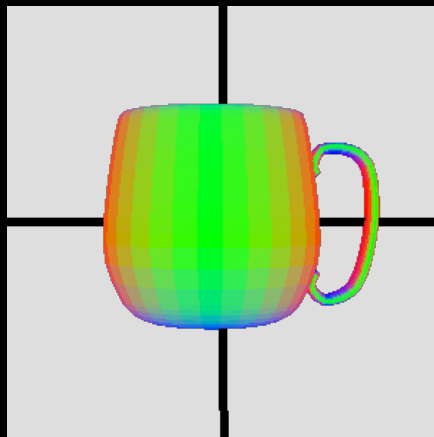
$$f(\theta) = \sum_k a_k \cos(k\theta) + b_k \sin(k\theta)$$

$\{a_k, |b_k|\}_k$   
Translation Invariant  
Representation

# Problem with PCA-Based Alignment



If singular values are close, axes unstable

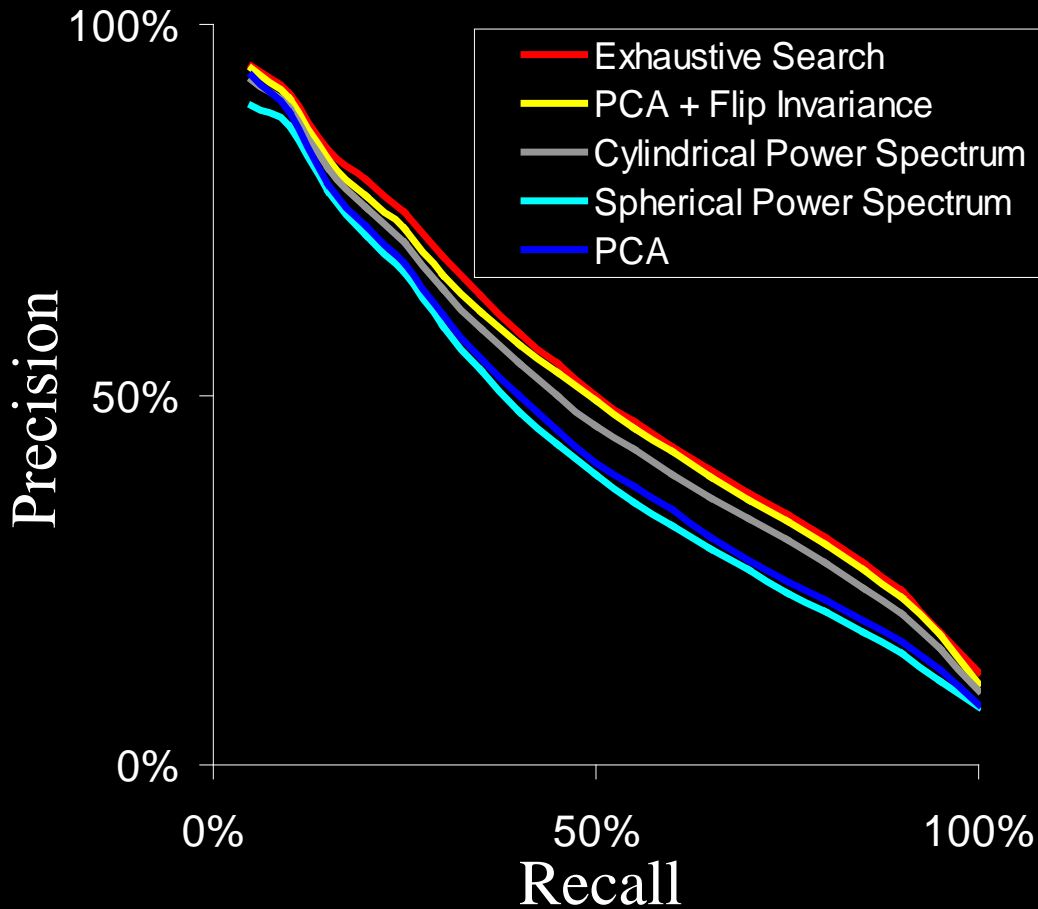




# Retrieval Results (Rotation)



## Gaussian EDT



## Size:

Method	Floats
Exhaustive Search	8192
PCA + Flip Invariance	8192
PCA	8192
Cylindrical PS	4352
Spherical PS	512

## Time:

Method	Secs.
Exhaustive Search	20.59
PCA + Flip Invariance	.67
PCA	.67
Cylindrical PS	.32
Spherical PS	.03