Shape Matching

Michael Kazhdan

(601.457/657)

Overview

Intro

General Approach

Minimum SSD Descriptor

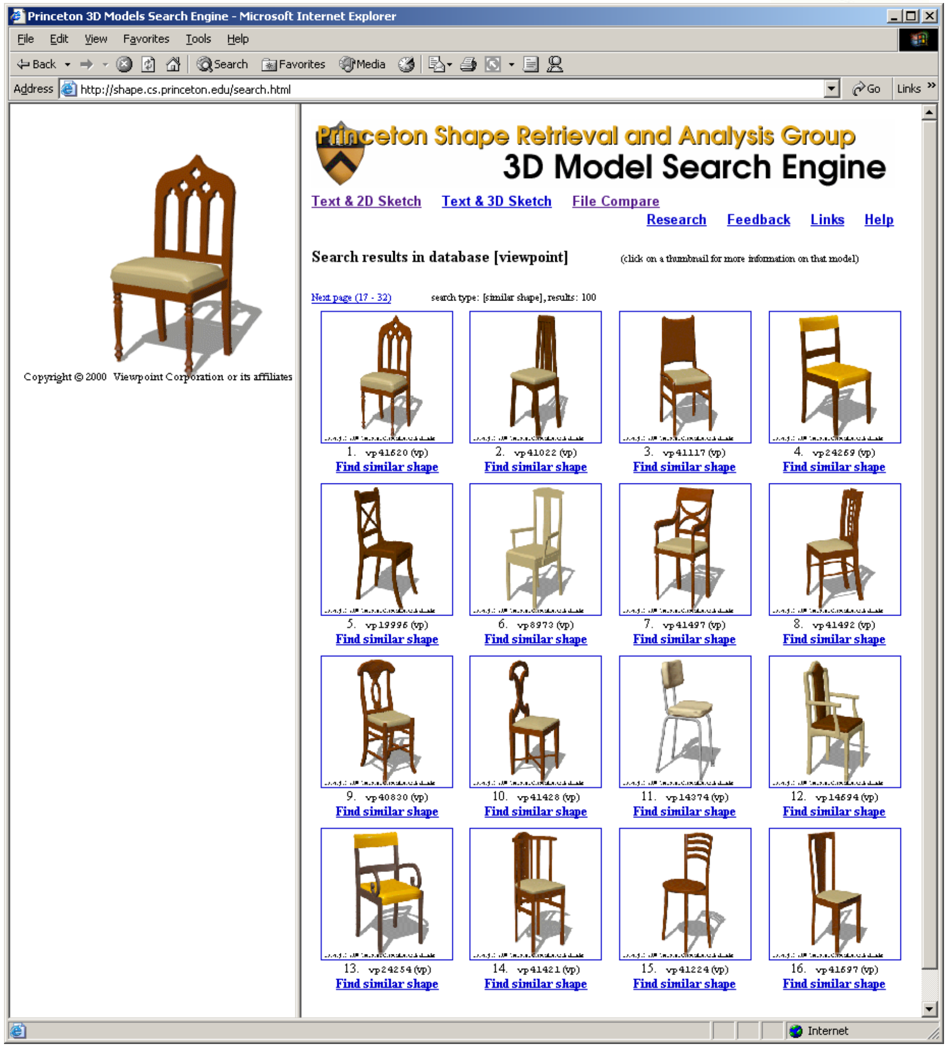
Goal

Given:

1. 3D model database
2. query shape

Find:

The database models most similar to the query.



Applications

Entertainment

Medicine

Chemistry/Biology

Archaeology

Applications

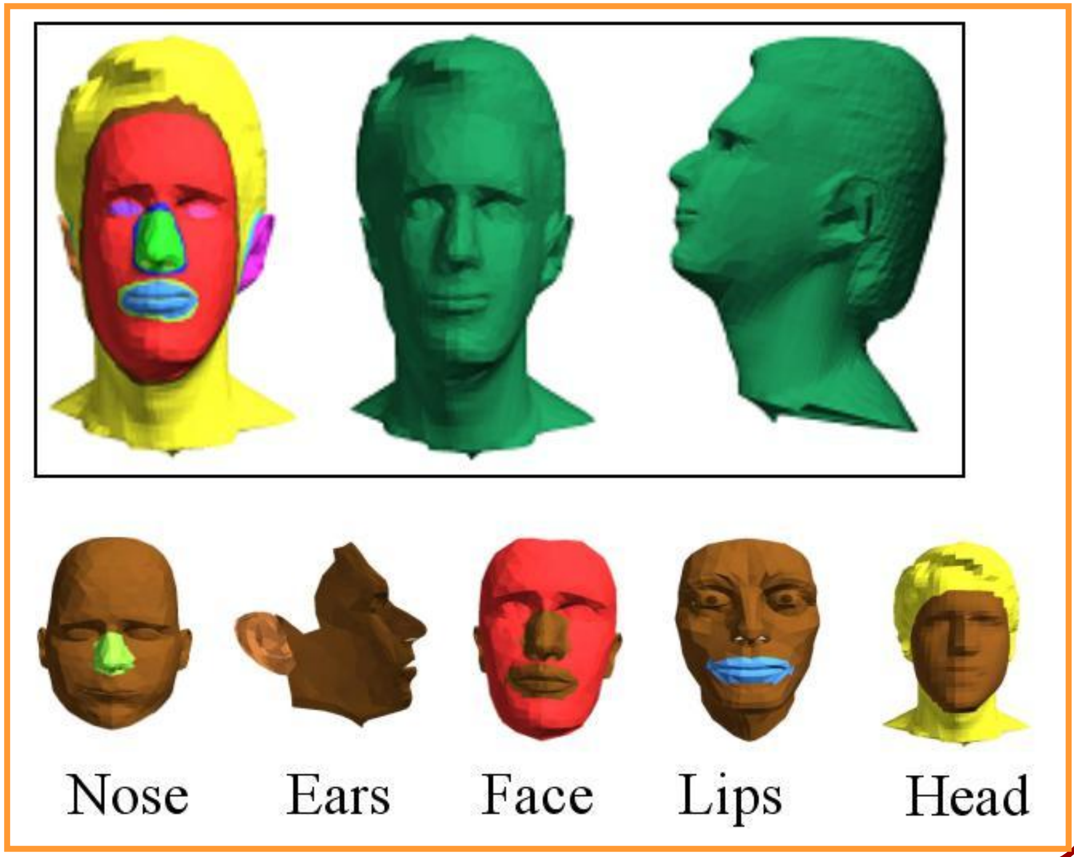
Entertainment

- Model generation

Medicine

Chemistry/Biology

Archaeology



Applications

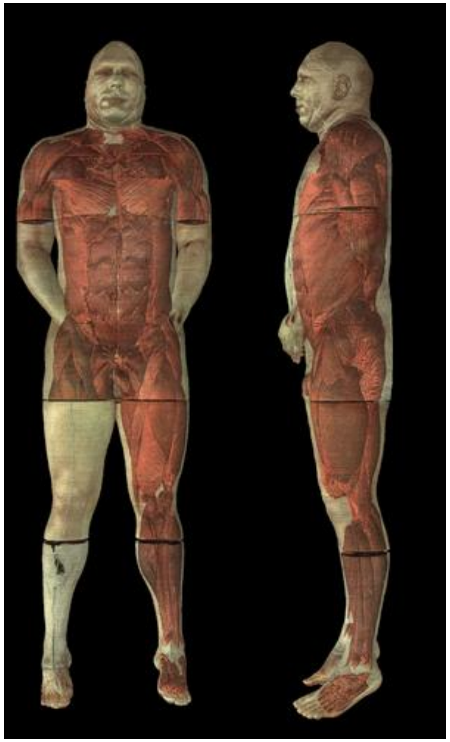
Entertainment

Medicine

- Automated diagnosis

Chemistry/Biology

Archaeology




Images courtesy of NLM

Applications

Entertainment

Medicine

Chemistry/Biology

- Docking and binding

Archaeology

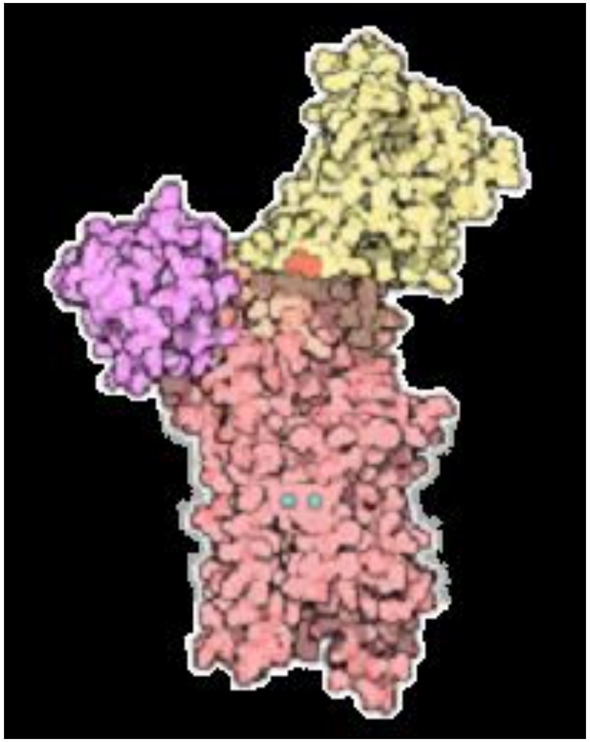


Image Courtesy of PDB

Applications

- Entertainment
- Medicine
- Chemistry/Biology
- Archaeology
  - Reconstruction

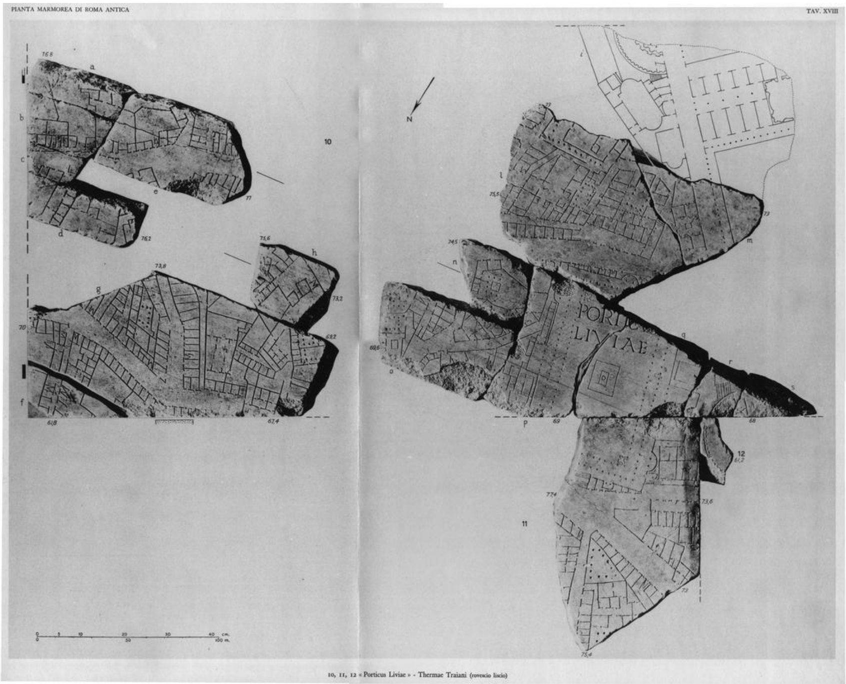


Image Courtesy of Stanford

Overview

Motivation

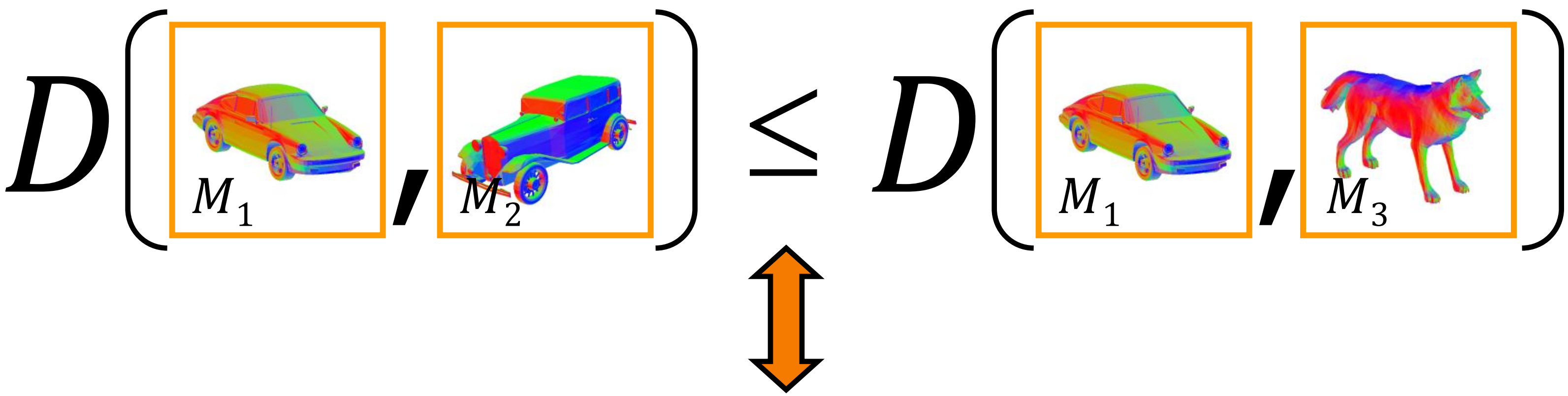
General Approach

Minimum SSD Descriptor

Shape Matching

General approach:

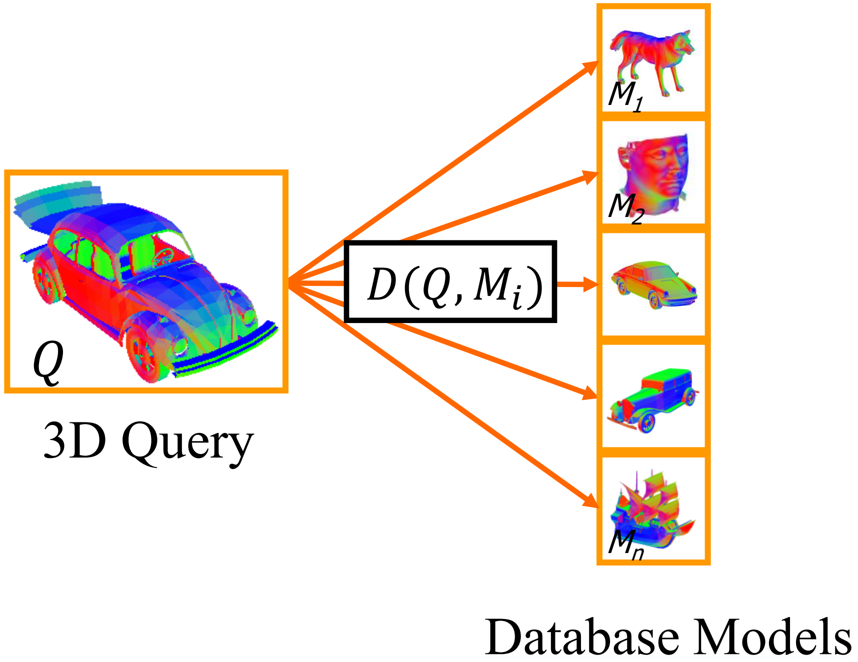
Define a function taking two models and returning the measure of their proximity.



M_1 is closer to M_2 than it is to M_3

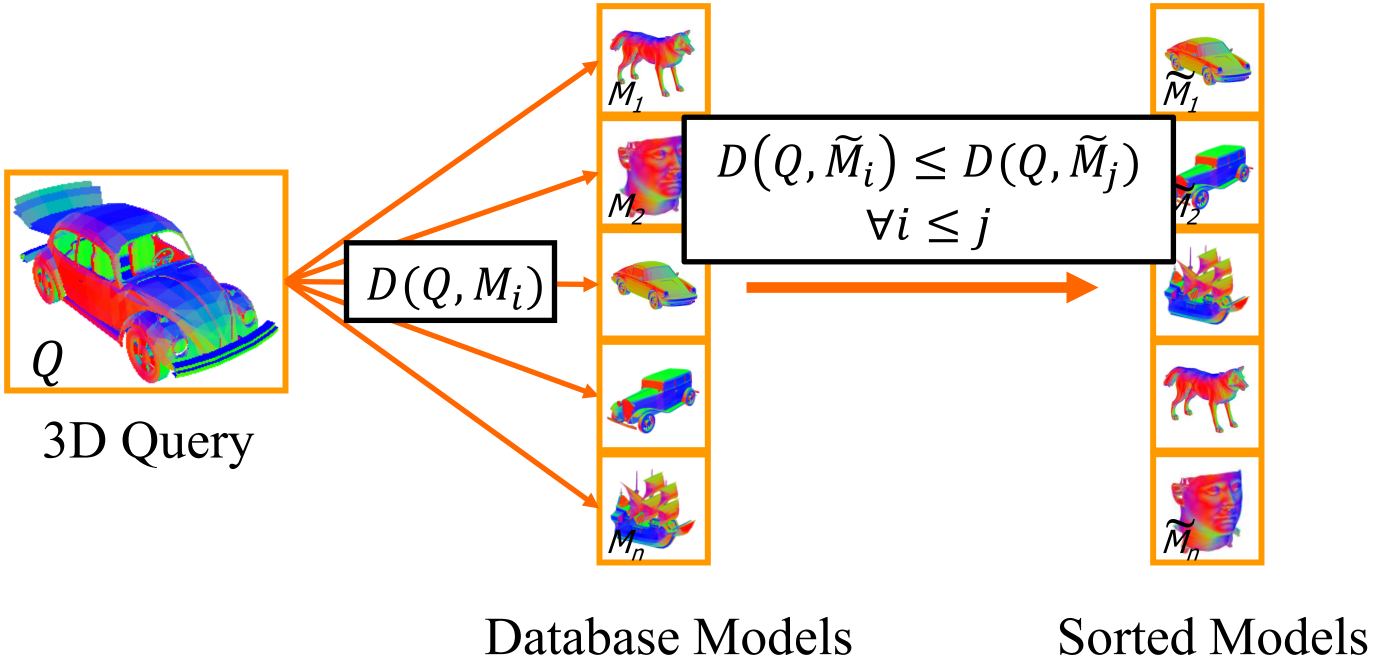
Database Retrieval

Compute the distance from the query to each database model



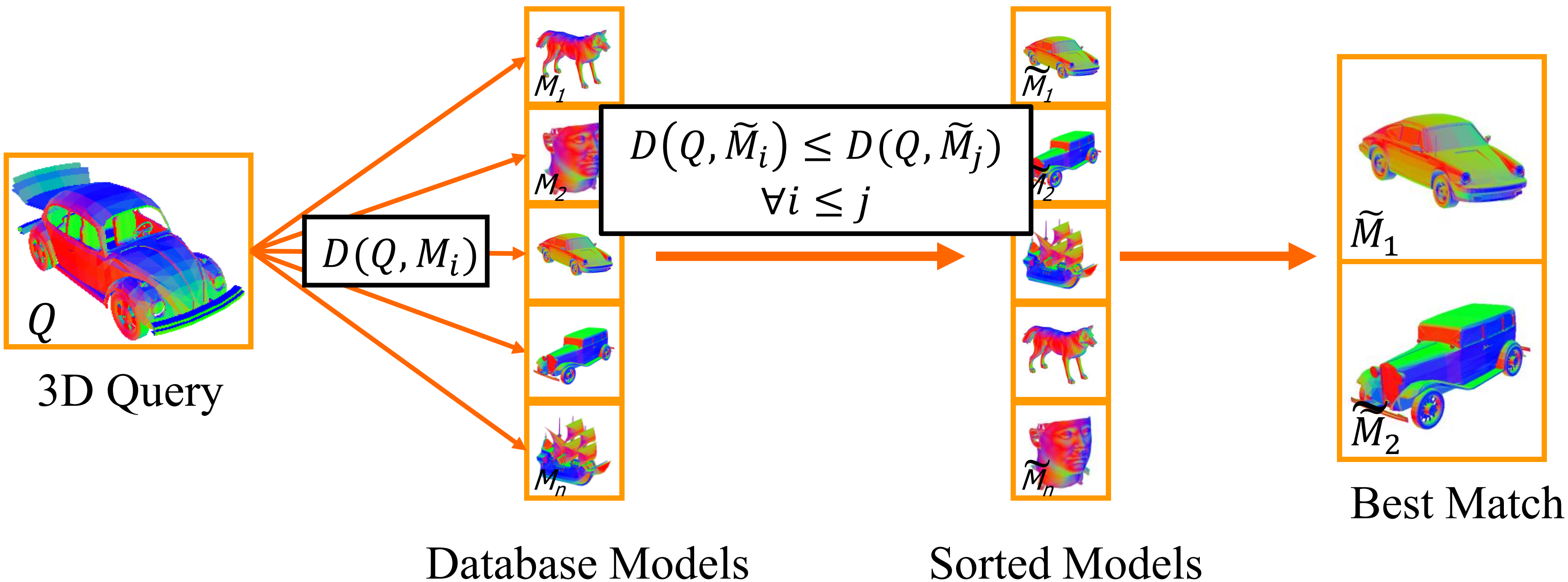
Database Retrieval

Sort the database models by proximity



Database Retrieval

Return the closest matches



Overview

Motivation

General Approach

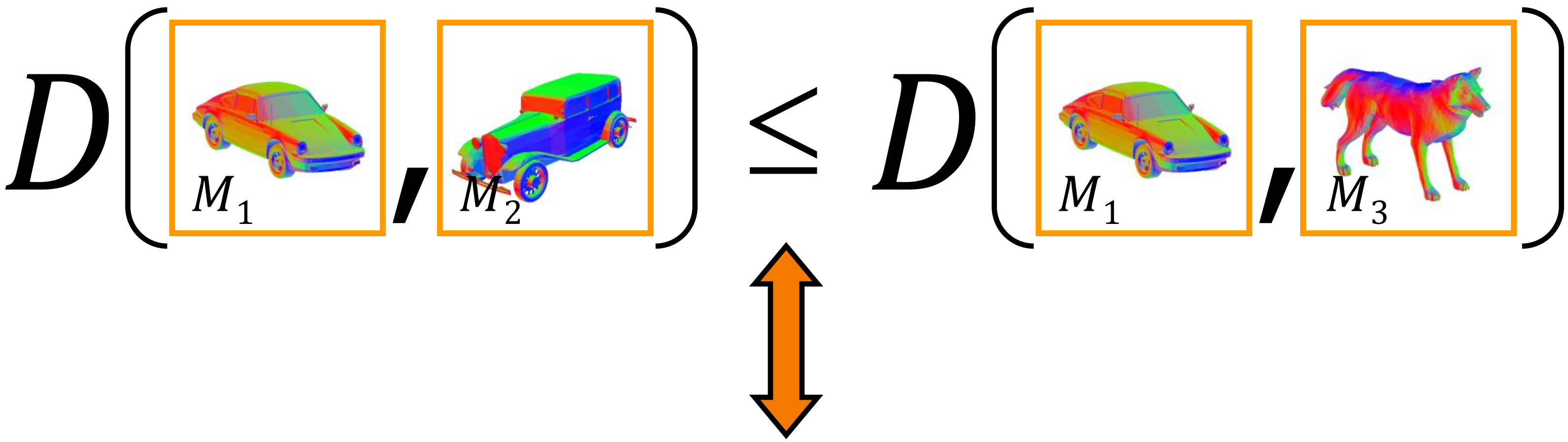
- Shape Descriptors

Minimum SSD Descriptor

Shape Matching

General approach:

Define a function that takes two models and returns a measure of their proximity.

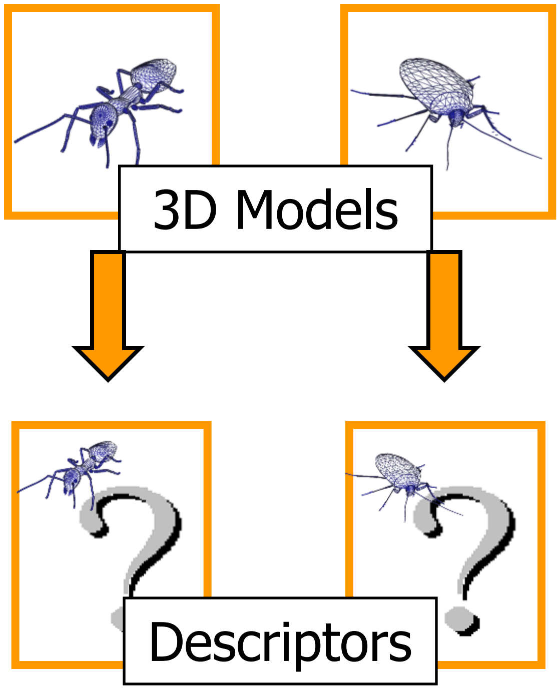


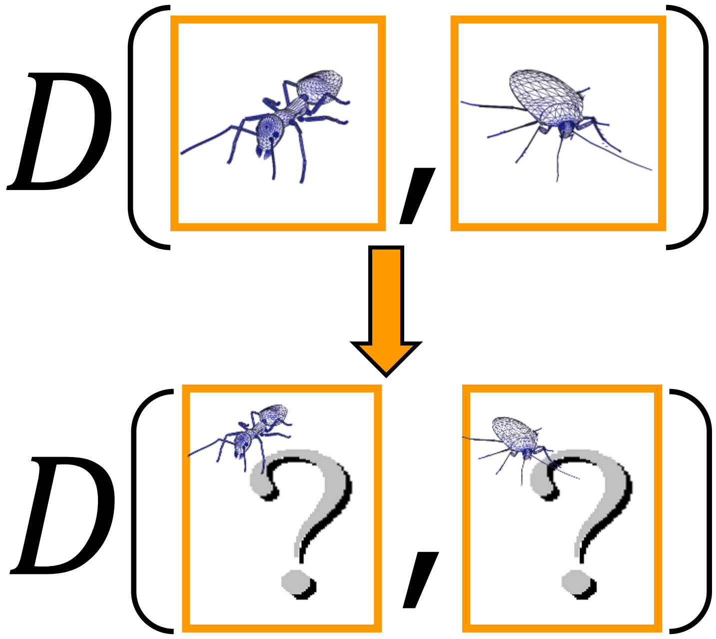
M_1 is closer to M_2 than it is to M_3

Shape Descriptors

<u>Shape Descriptor</u>:

A structured abstraction of a 3D model that is well suited to the challenges of shape matching



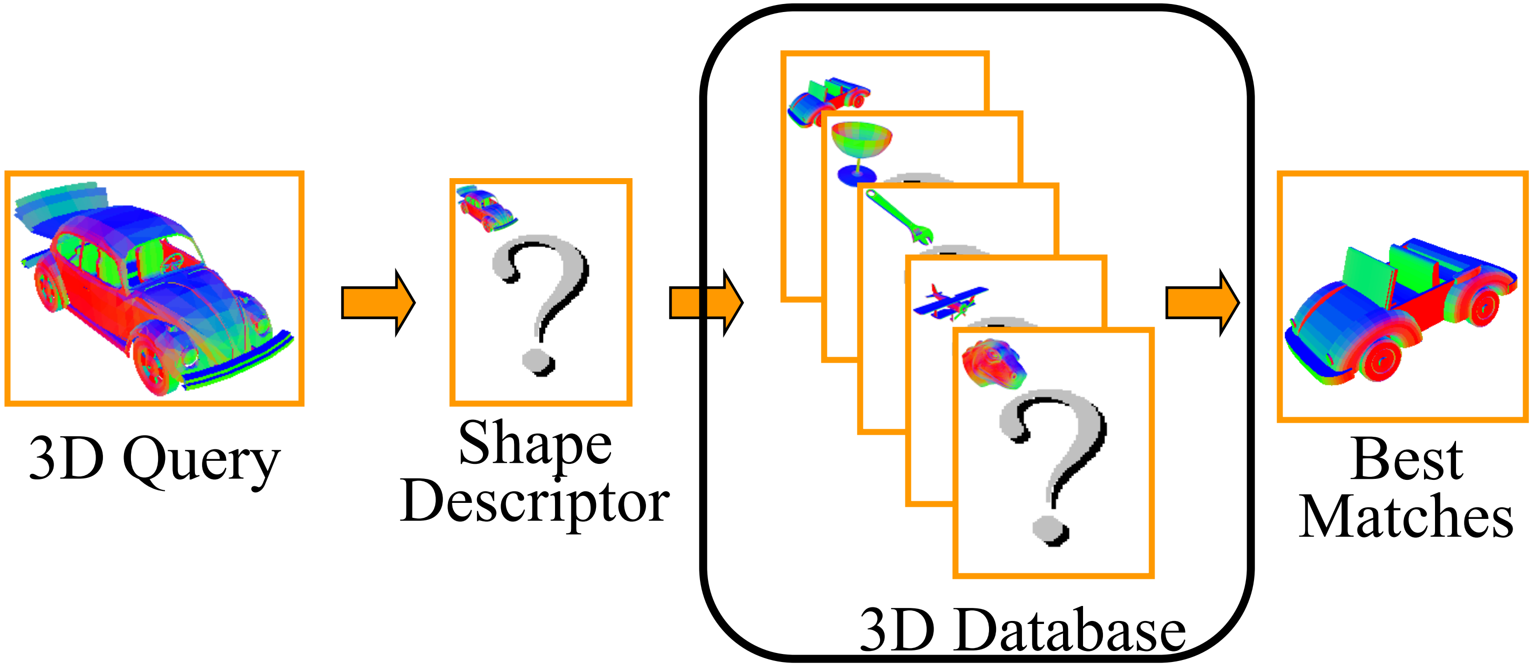


Matching with Descriptors

Preprocessing

- Compute database descriptors

Run-Time



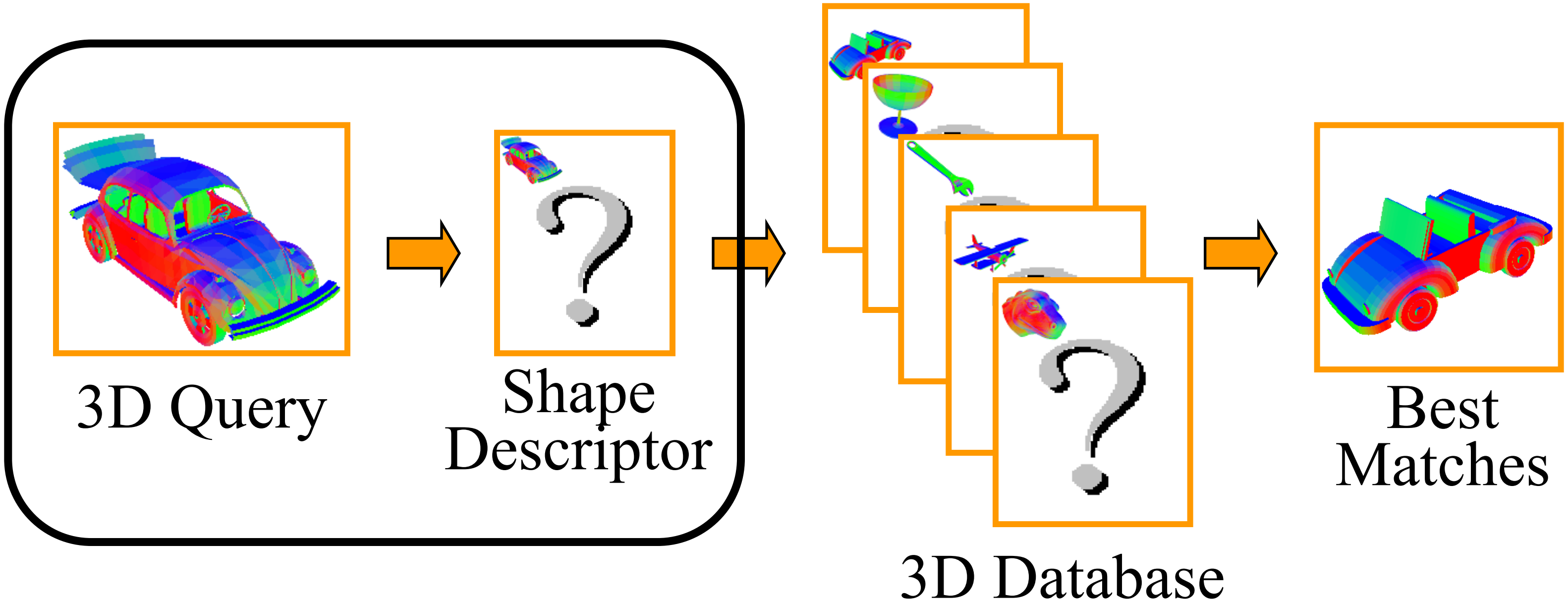
Matching with Descriptors

Preprocessing

- Compute database descriptors

Run-Time

- Compute query descriptor



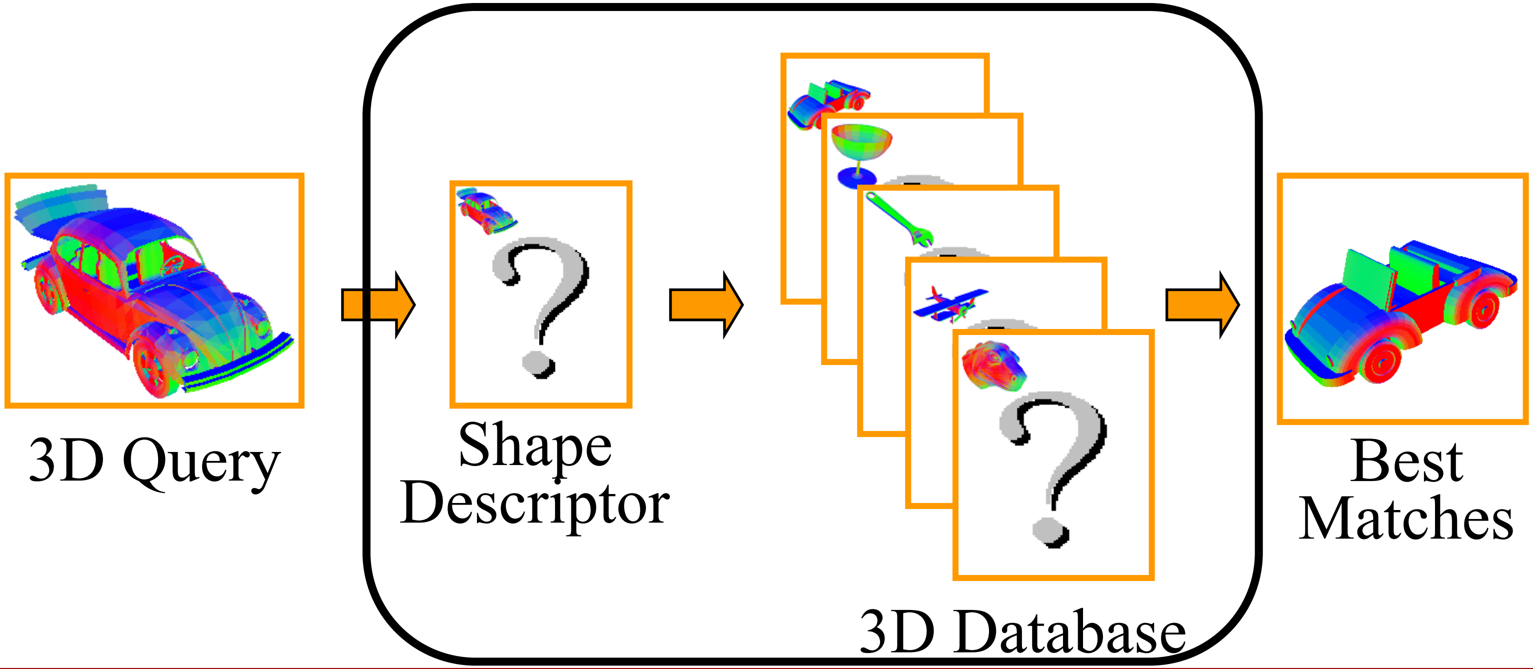
Matching with Descriptors

Preprocessing

- Compute database descriptors

Run-Time

- Compute query descriptor
- Compare query descriptor to database descriptors



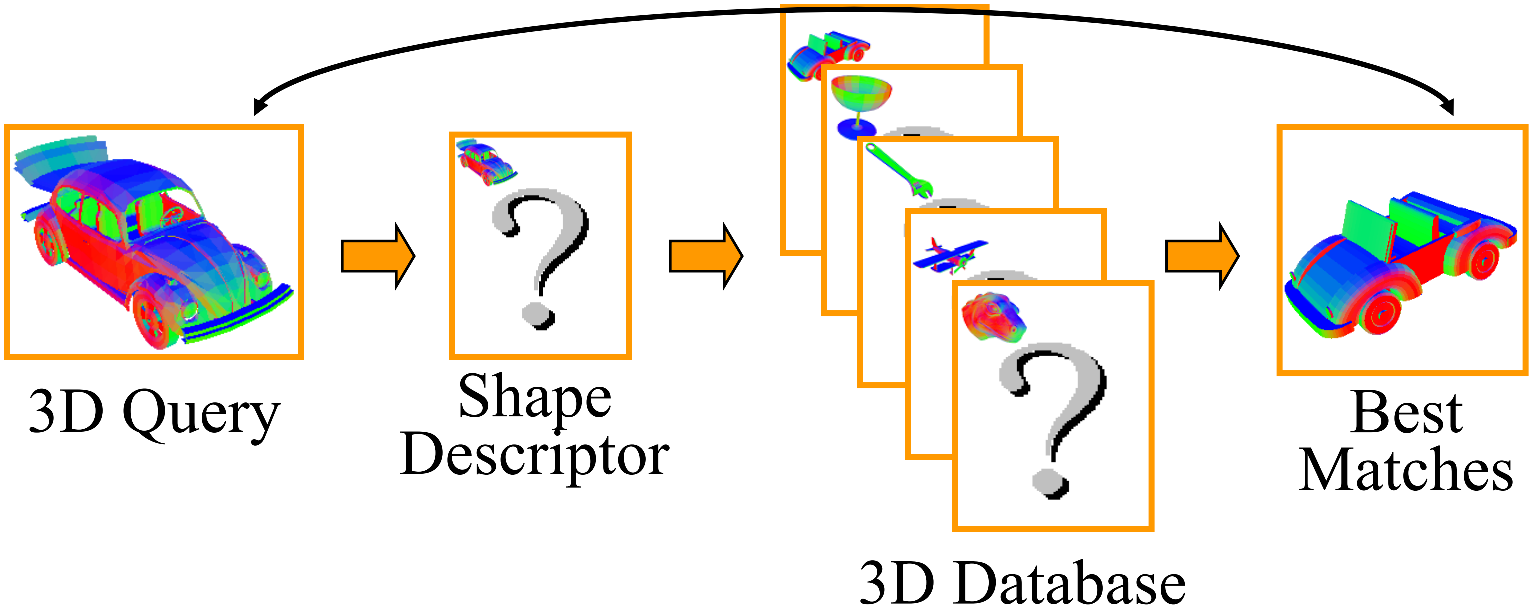
Matching with Descriptors

Preprocessing

- Compute database descriptors

Run-Time

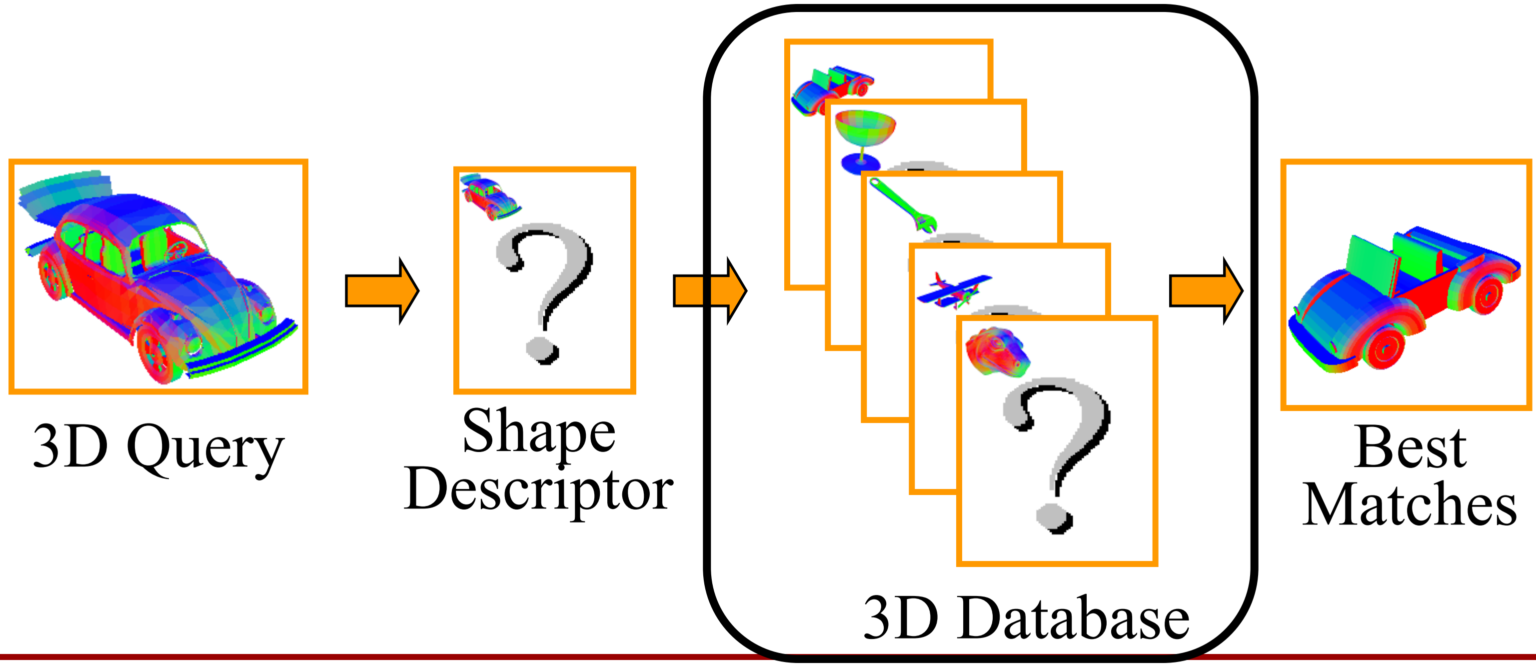
- Compute query descriptor
- Compare query descriptor to database descriptors
- Return best Match(es)



Shape Matching Challenge

Need shape descriptor that is:

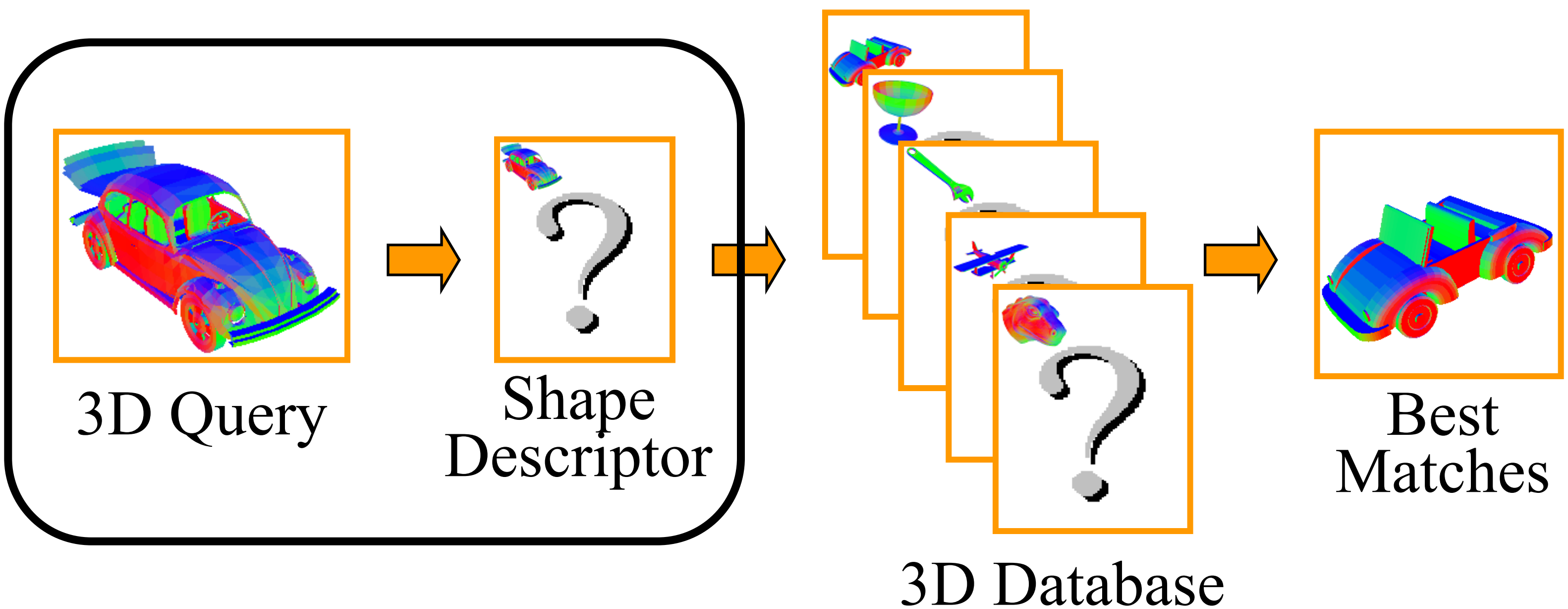
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



Shape Matching Challenge

Need shape descriptor that is:

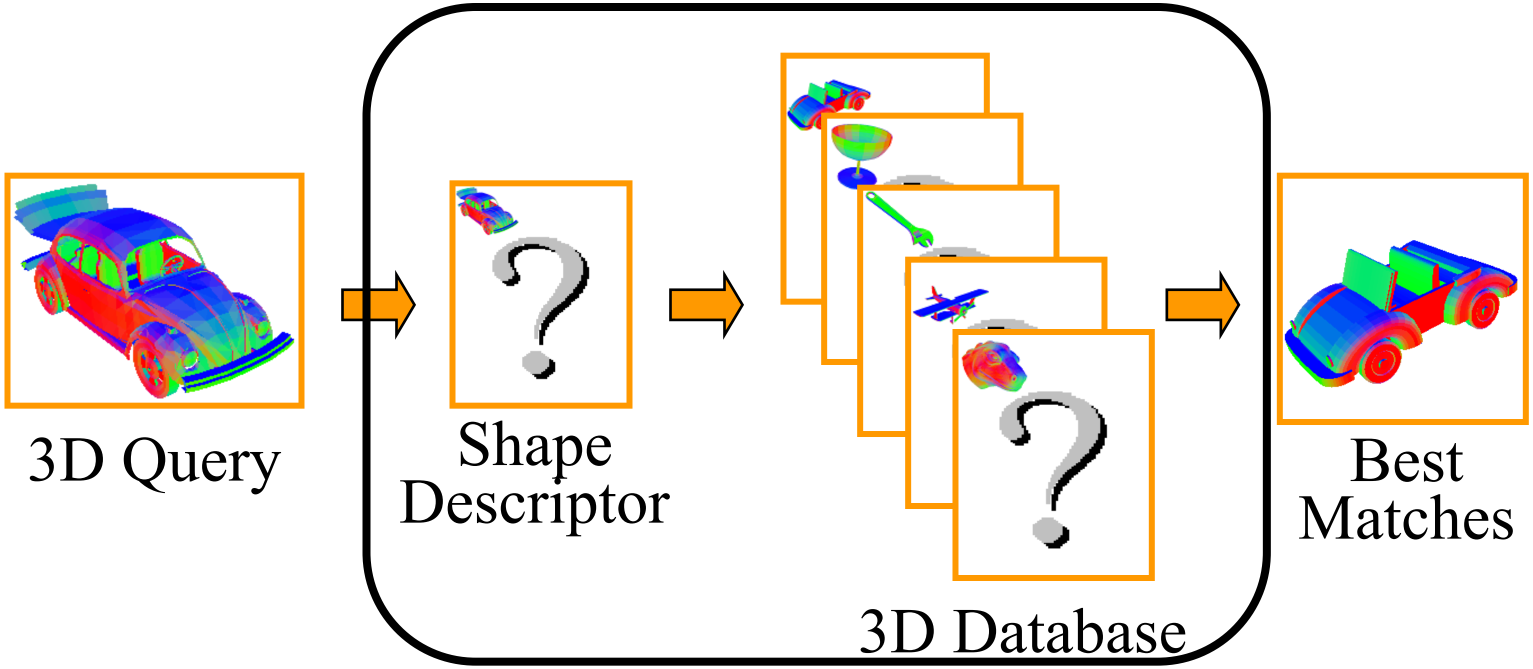
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



Shape Matching Challenge

Need shape descriptor that is:

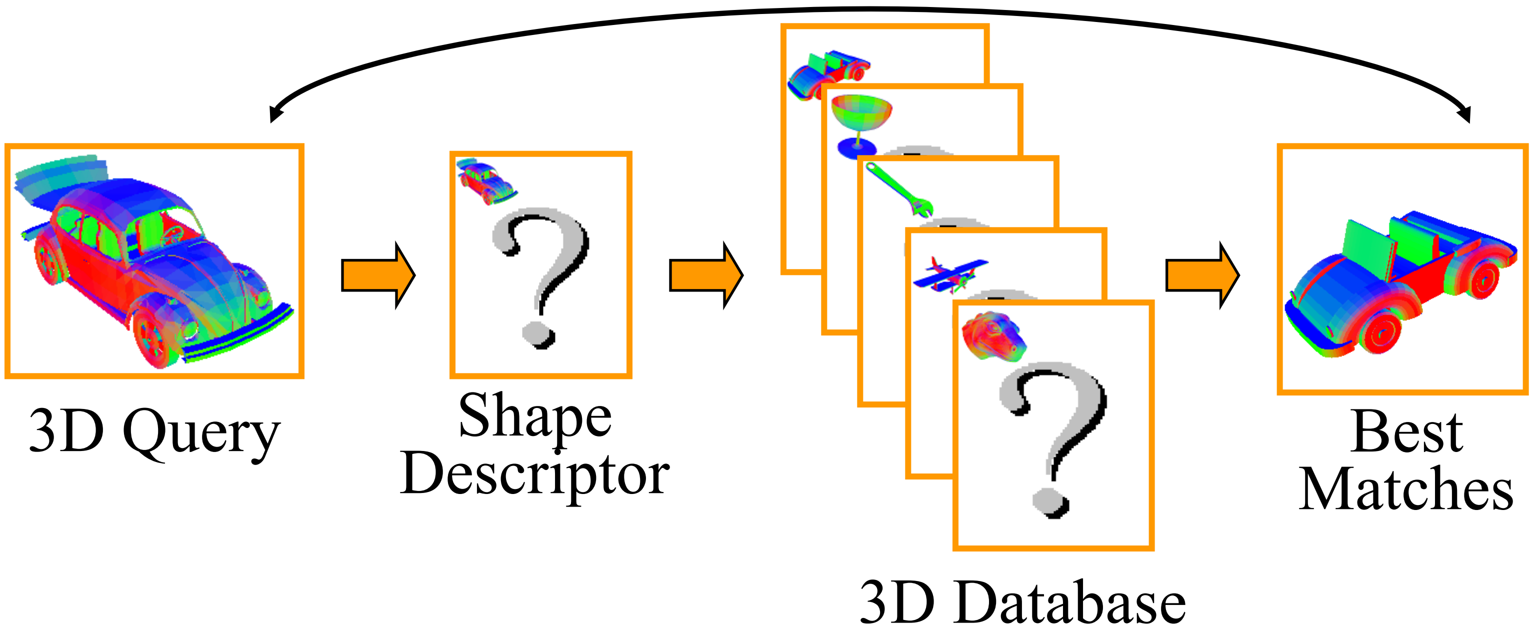
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



Shape Matching Challenge

Need shape descriptor that is:

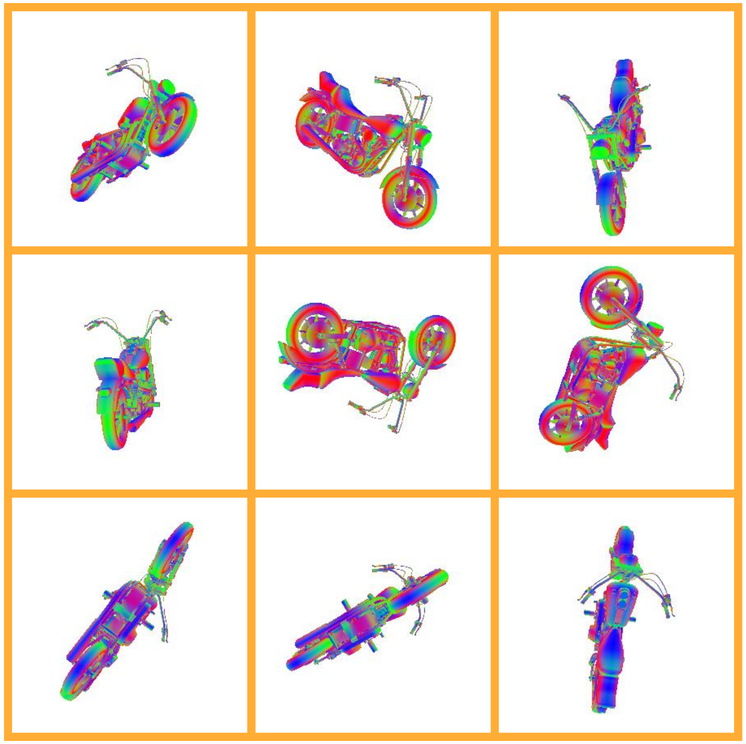
- Concise to store
- Quick to compute
- Efficient to match
- Discriminating



Shape Matching Challenge

Need shape descriptor that is:

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

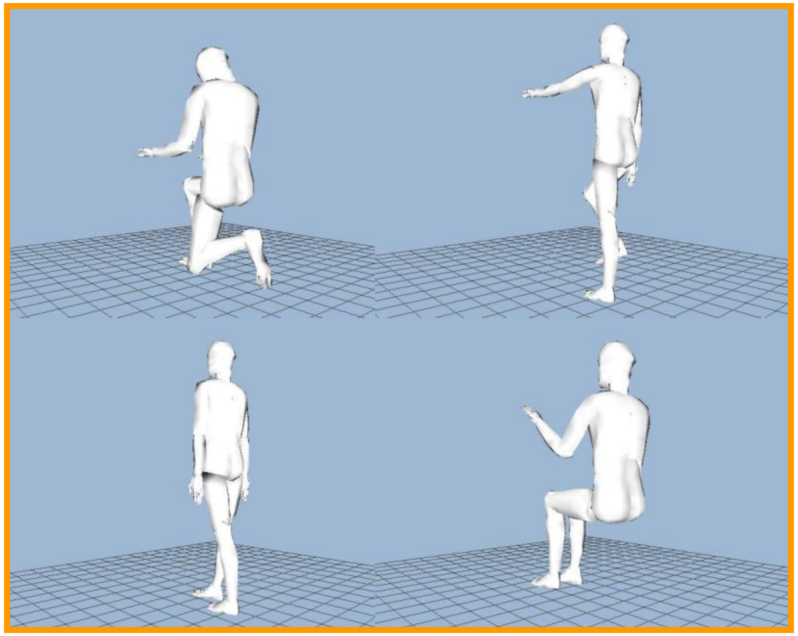


Different Transformations
(translation, scale, rotation, mirror)

Shape Matching Challenge

Need shape descriptor that is:

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



Different Articulated Poses

Shape Matching Challenge

Need shape descriptor that is:

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

Scanned Surface

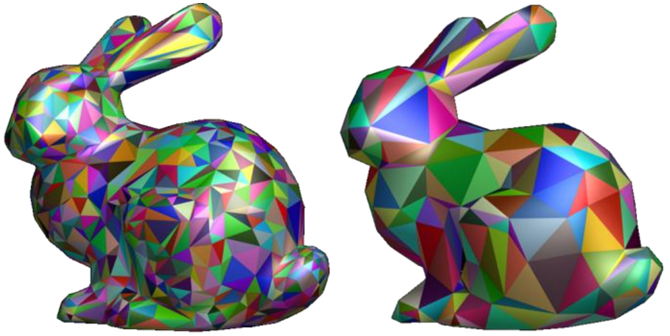
Image courtesy of Ramamoorthi et al.

Shape Matching Challenge

Need shape descriptor that is:

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

Different Genus



Different Tessellations

Images courtesy of
Viewpoint & Stanford

Overview

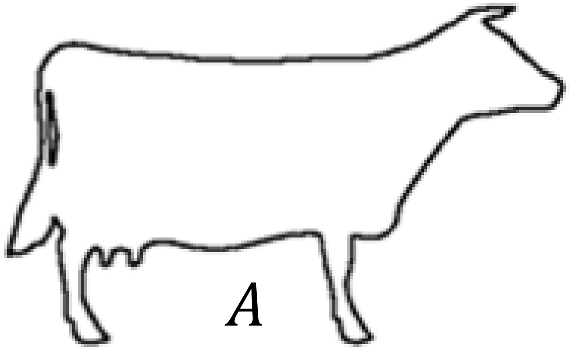
Applications

General Approach

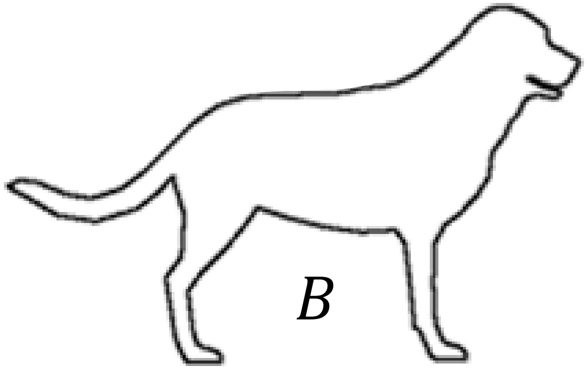
Minimum SSD Descriptor

Shape Matching Approach

Q: How should we measure the similarity between two shapes?



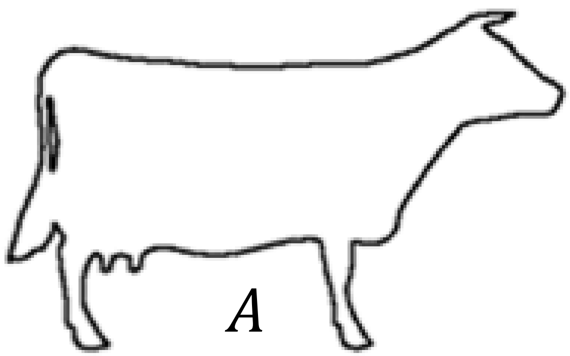
A

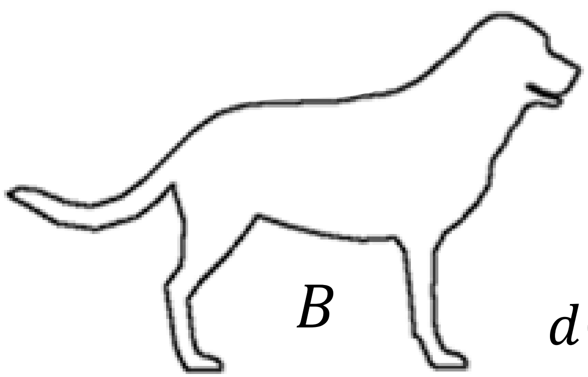


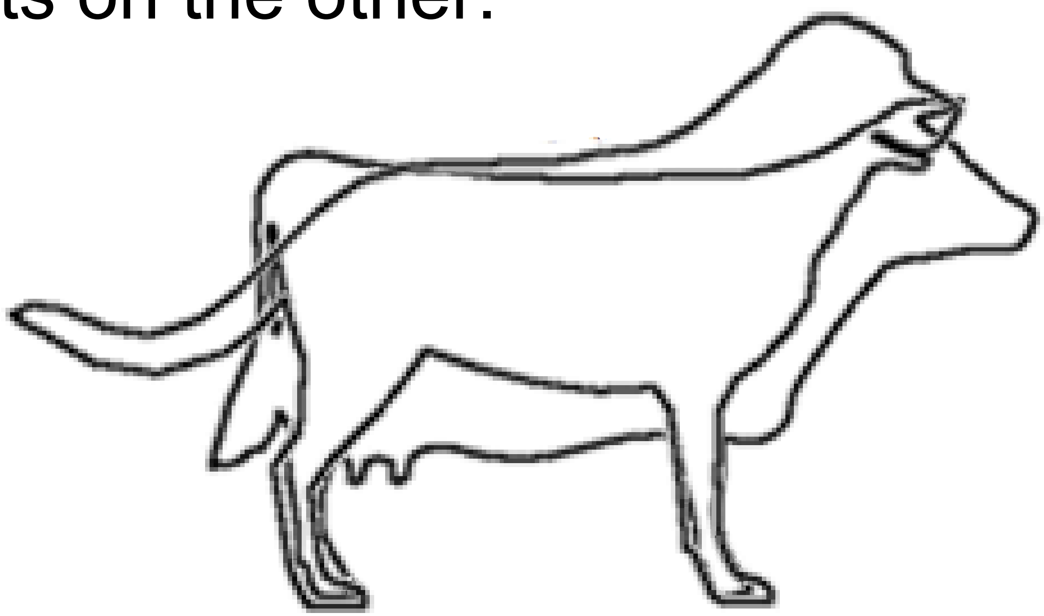
B

Shape Matching Approach

A: Define shape (dis)similarity as the sum of squared distances from points on one surface to the closest points on the other.



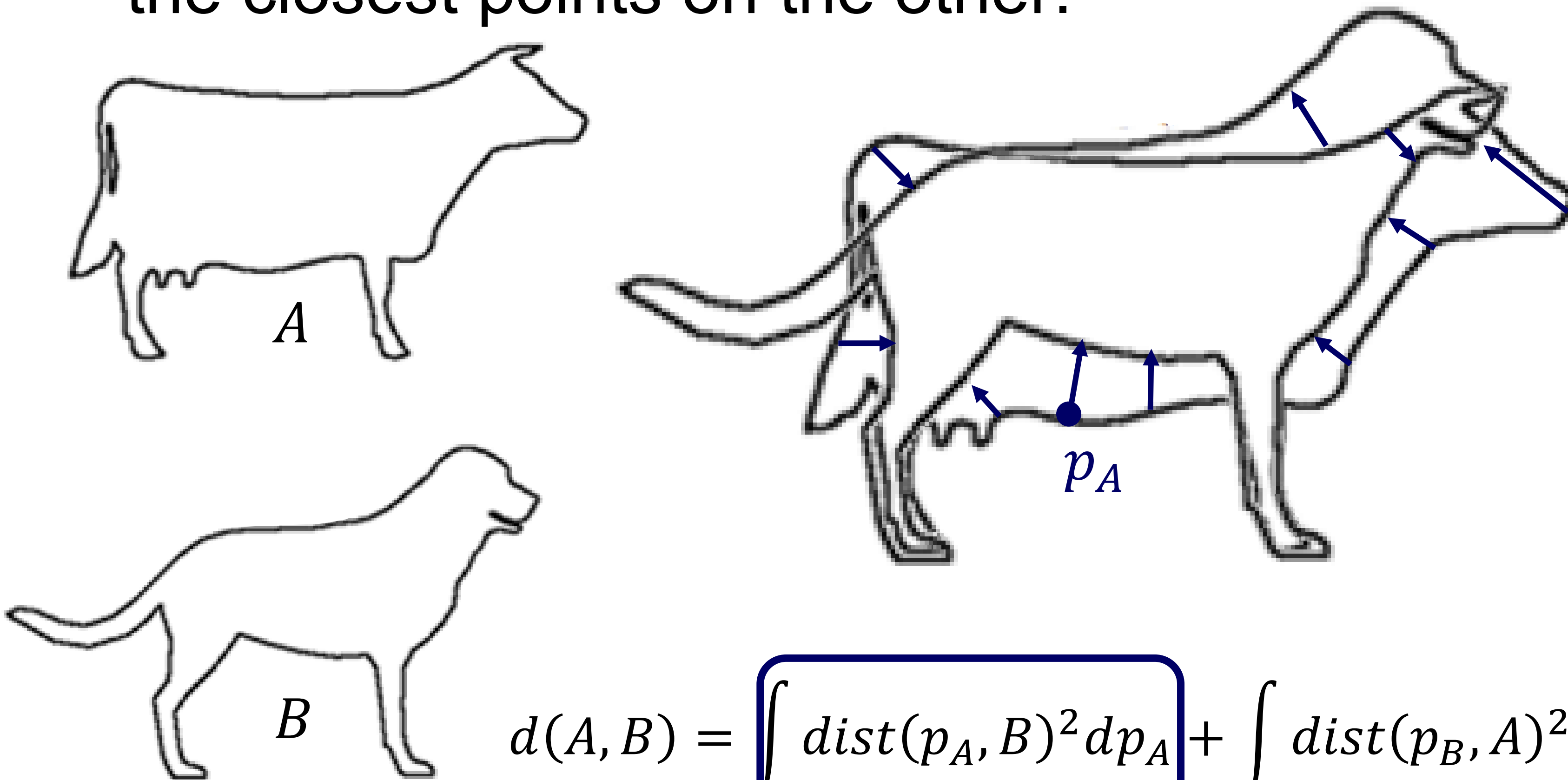




$$d(A,B)=\int_A dist(p_A,B)^2\,dp_A+\int_B dist(p_B,A)^2\,dp_B$$

Shape Matching Approach

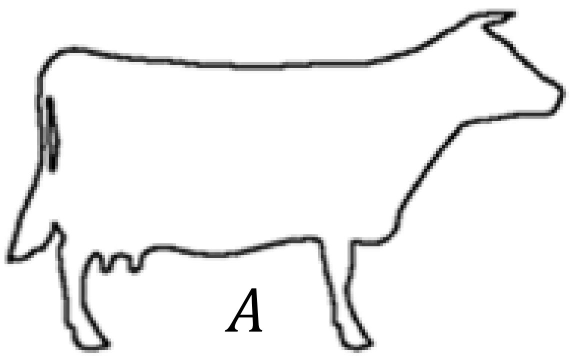
A: Define shape (dis)similarity as the sum of squared distances from points on one surface to the closest points on the other.

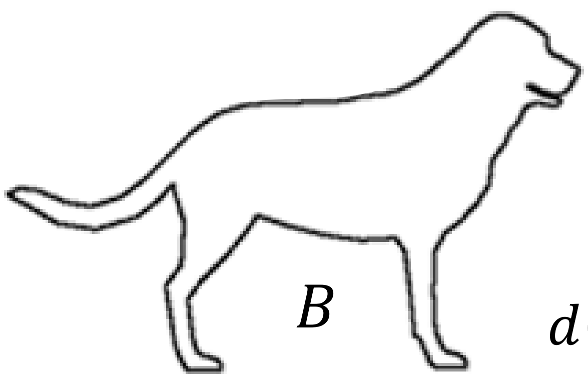


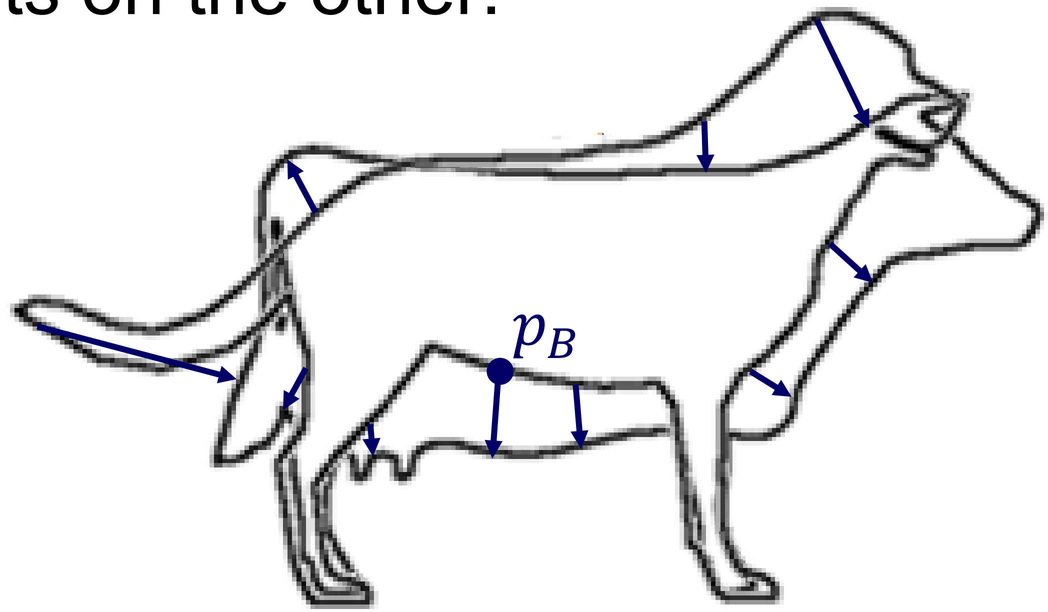
$$d(A,B) = \int_A dist(p_A, B)^2 dp_A + \int_B dist(p_B, A)^2 dp_B$$

Shape Matching Approach

A: Define shape (dis)similarity as the sum of squared distances from points on one surface to the closest points on the other.







$$d(A,B) = \int_A dist(p_A, B)^2 dp_A + \int_B dist(p_B, A)^2 dp_B$$

Overview

Applications

General Approach

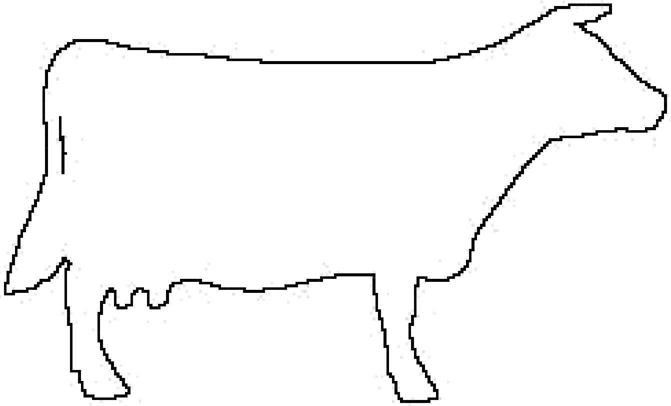
Minimum SSD Descriptor

- (Euclidean) Distance Transform

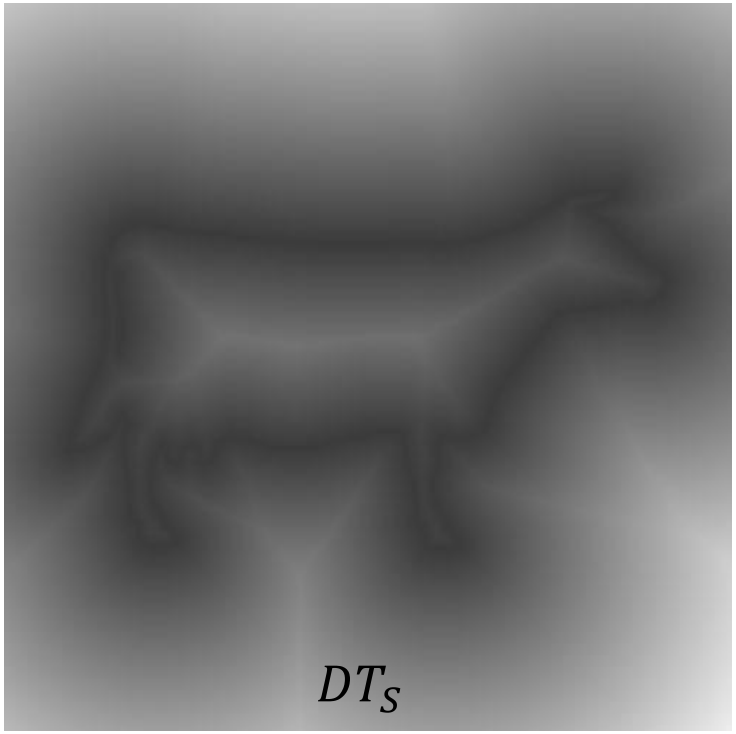
(Euclidean) Distance Transform

The (*Euclidean*) *Distance Transform* (DT) of a surface is a function (defined in 3D) returning the distance to the nearest surface point.

$$DT_S(p) = \min_{q \in S} \|p - q\|$$



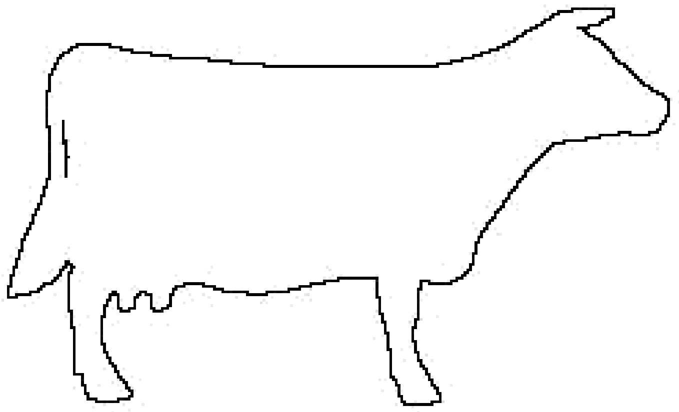
S



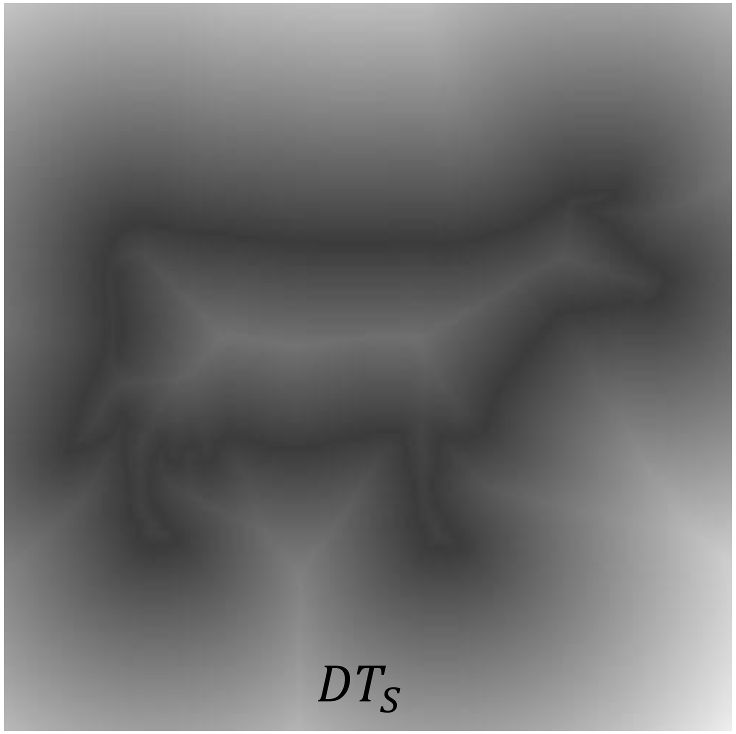
(Euclidean) Distance Transform

Grass-Fire Algorithm:

- Treat space as a field of dry grass.
- Set fire to the boundary and measure the time for the fire to reach each point.



S



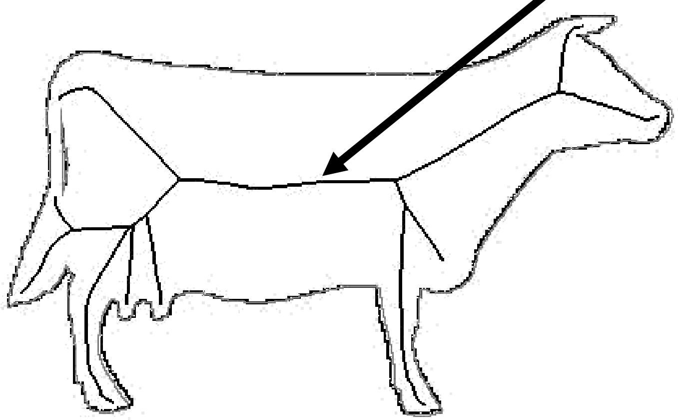
DT_S

(Euclidean) Distance Transform

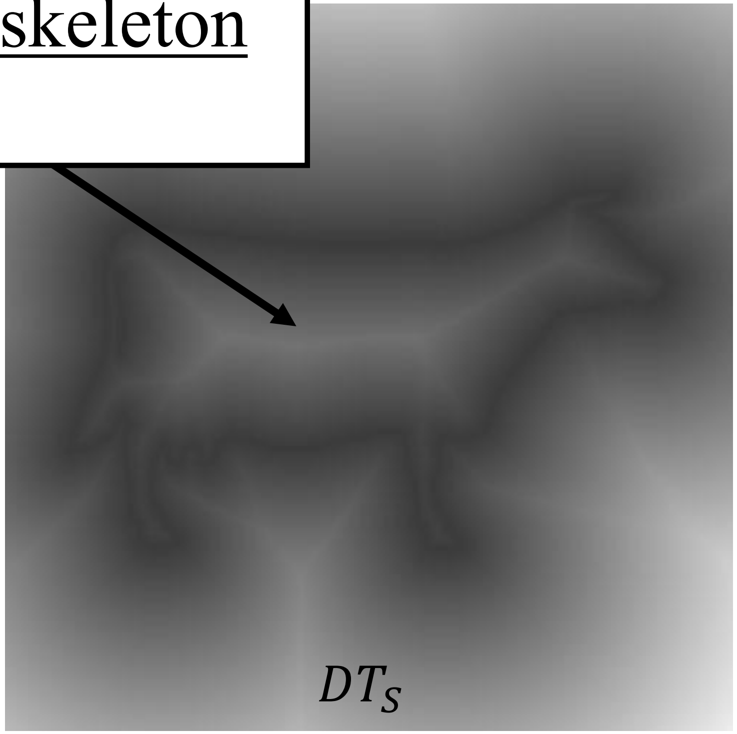
Grass-Fire Algorithm:

- Treat space as a field of dry grass.
- Set fire to the boundary and measure the time for the fire to

The points where the fire gets quenched define the skeleton of the shape.



S



Computing DT_S

Naïve:

Compute the distance to each surface point and store the minimum.

Complexity:

If there are m surface points and we want the values on a grid of resolution R:

» $O(R^2 m) \approx O(R^2 \cdot R)$ for a 2D grid

» $O(R^3 m) \approx O(R^3 \cdot R^2)$ for a 3D grid

Computing DT_S

<u>Graphics Hardware (2D)</u>:

1. For each surface point (x, y), draw a 3D right-cone with apex at $(x, y, 0)$ and axis aligned with the positive z-axis.
2. Render with orthographic projection, looking down the positive the z-axis.
3. Read the values of the depth-buffer to get the values of DT_S.

Computing DT_S

General Problem:

Given a set of points, $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ and given a point $p \in \mathbb{R}^2$ we would like to compute the distance to the closest point in P:

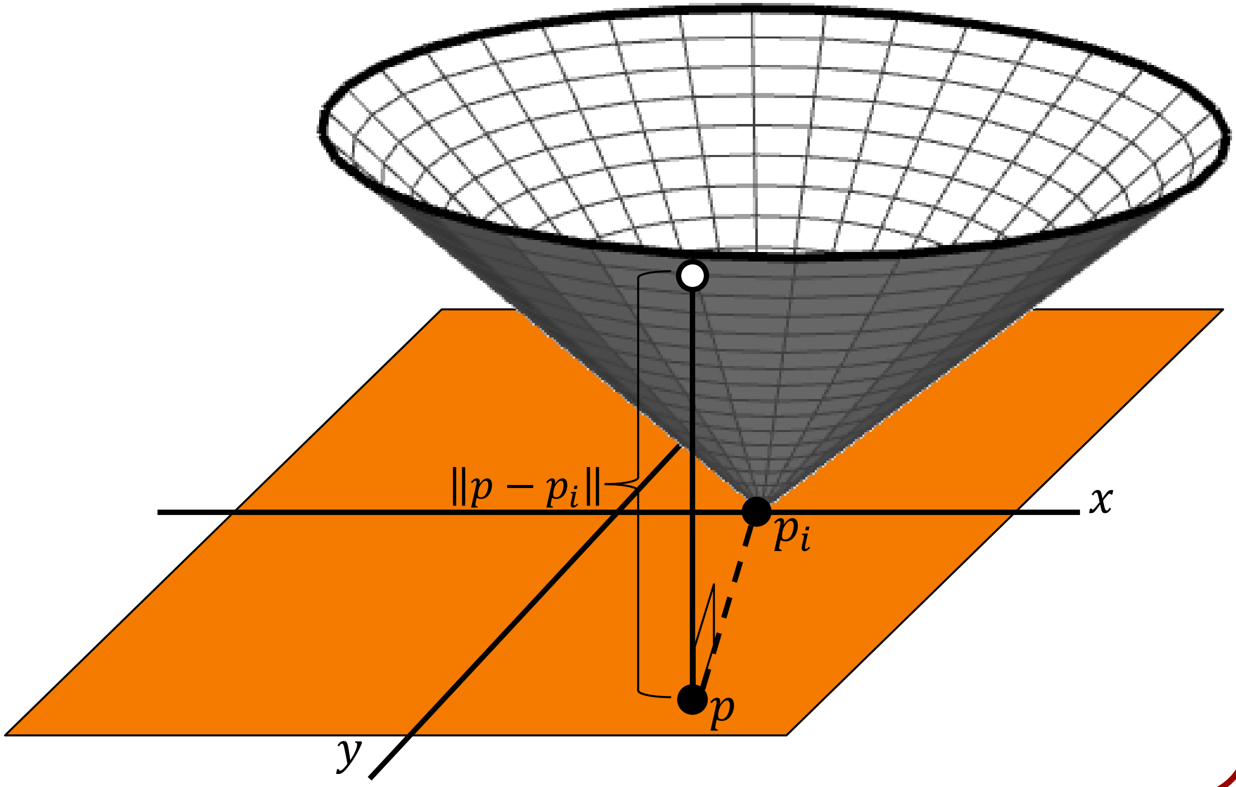
$$d(p, P) = \min_i \|p - p_i\|$$

Start by considering how we can compute the distance from the point p to a single point p_i.

Computing DT_S

<u>Graphics Hardware (2D)</u>:

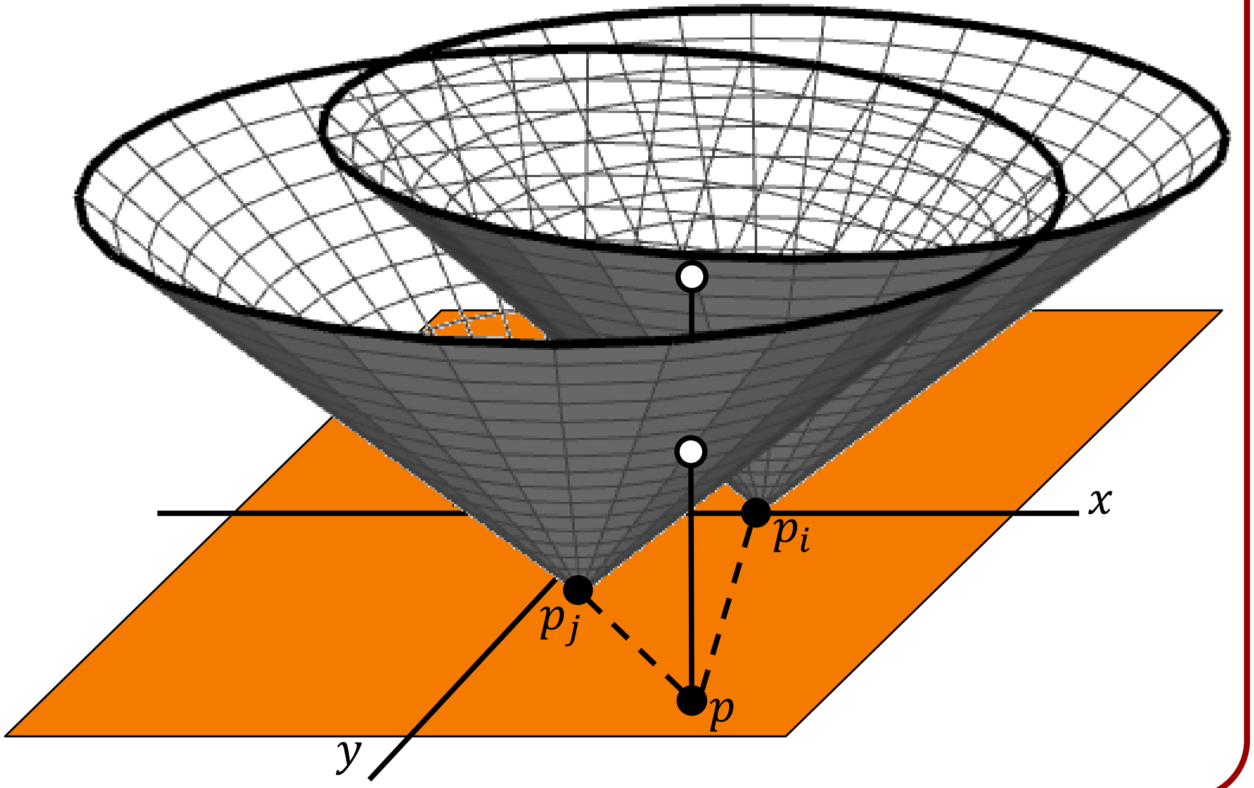
At p, the height of a **right**-cone with apex at p_i is the distance from p to p_i.



Computing DT_S

Graphics Hardware (2D):

For points p_i and p_j the distance from p to the closer of the two is the minimum of the two heights.



Computing DT_S

Graphics Hardware (2D):

For points p_i and p_j the distance from p to the closer of the two is the minimum of the two heights.

Given a collection of points in the xy-plane:

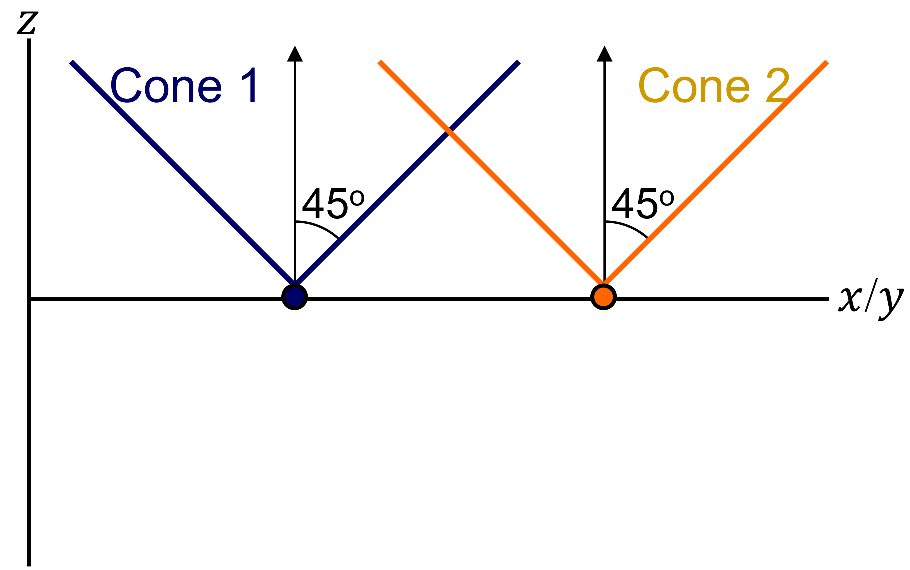
Computing DT_S

<u>Graphics Hardware (2D)</u>:

For points p_i and p_j the distance from p to the closer of the two is the minimum of the two heights.

Given a collection of points in the xy-plane:

Draw right-cones at each point



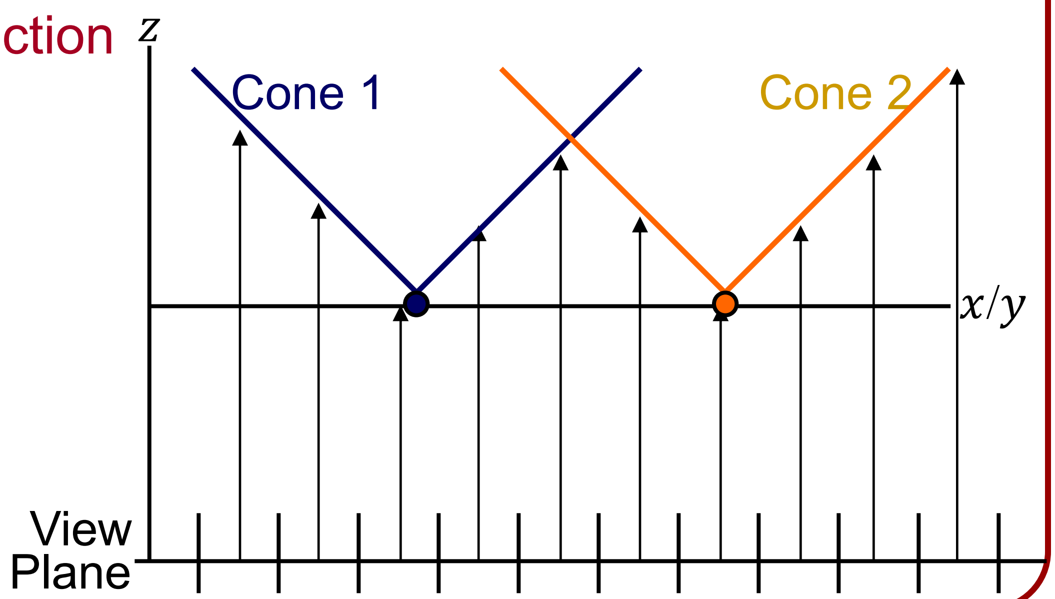
Computing DT_S

Graphics Hardware (2D):

For points p_i and p_j the distance from p to the closer of the two is the minimum of the two heights.

Given a collection of points in the xy-plane :

- Draw right-cones at each point
- View along the z-direction



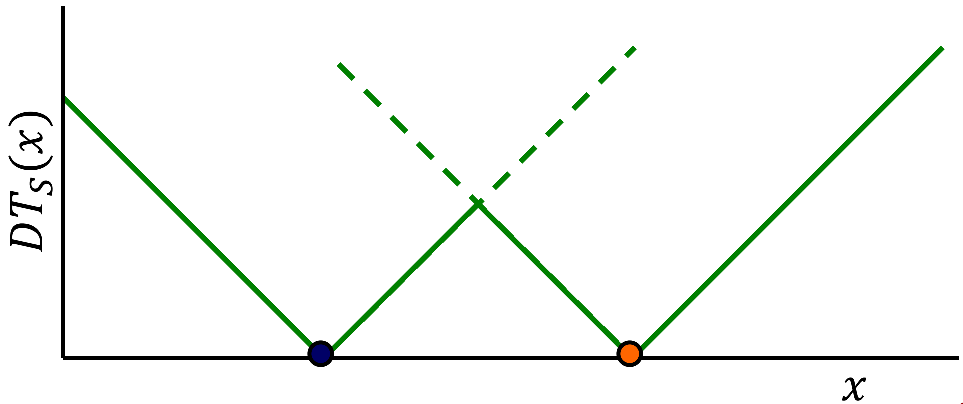
Computing DT_S

Graphics Hardware (2D):

For points p_i and p_j the distance from p to the closer of the two is the minimum of the two heights.

Given a collection of points:

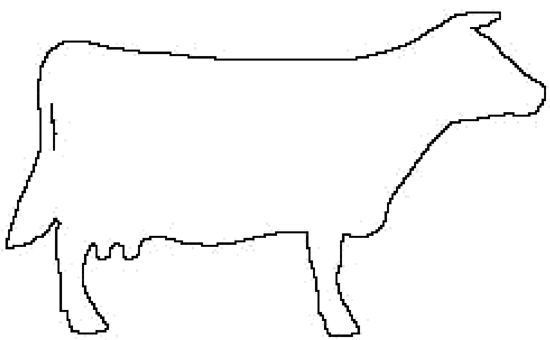
- Draw right-cones at each point
- View along the z-direction
- Read back the depth-buffer



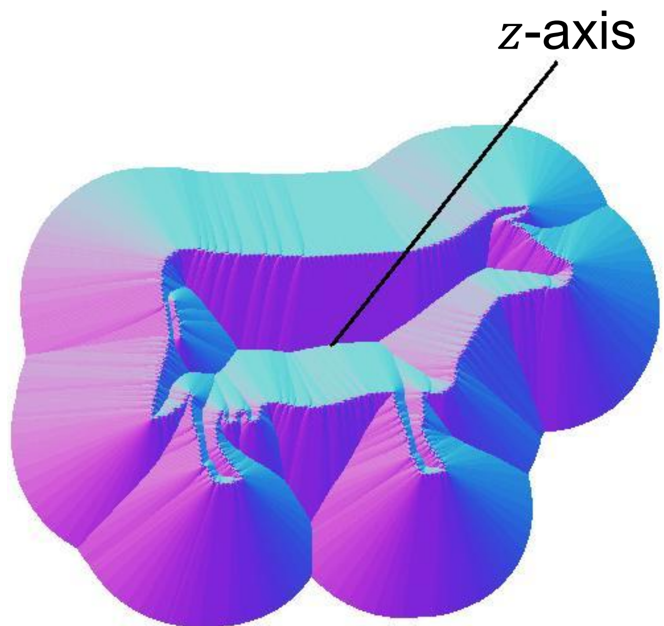
Computing DT_S

<u>Graphics Hardware (2D)</u>:

- Draw right-cones at each point
- View along the z-direction
- Read back the depth-buffer



Surface



Right-Cones

Visualization

Overview

Applications

General Approach

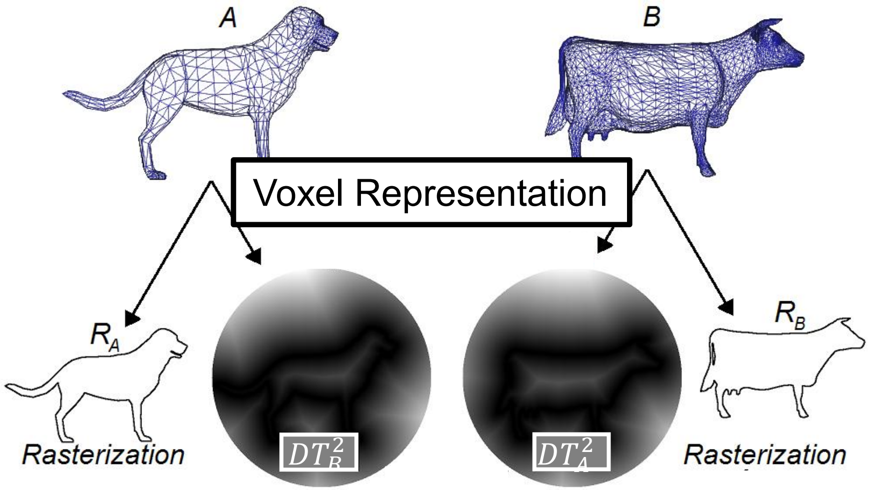
Minimum SSD Descriptor

- (Euclidean) Distance Transform

Shape Matching Implementation

<u>Preprocessing</u>:

Compute **rasterization** and **squared distance transforms**



- The value of the rasterization at a 3D point (voxel) is:

$$R_A(p) = \begin{cases} 1 & \text{if} \quad p \in A \\ 0 & \text{otherwise} \end{cases}$$

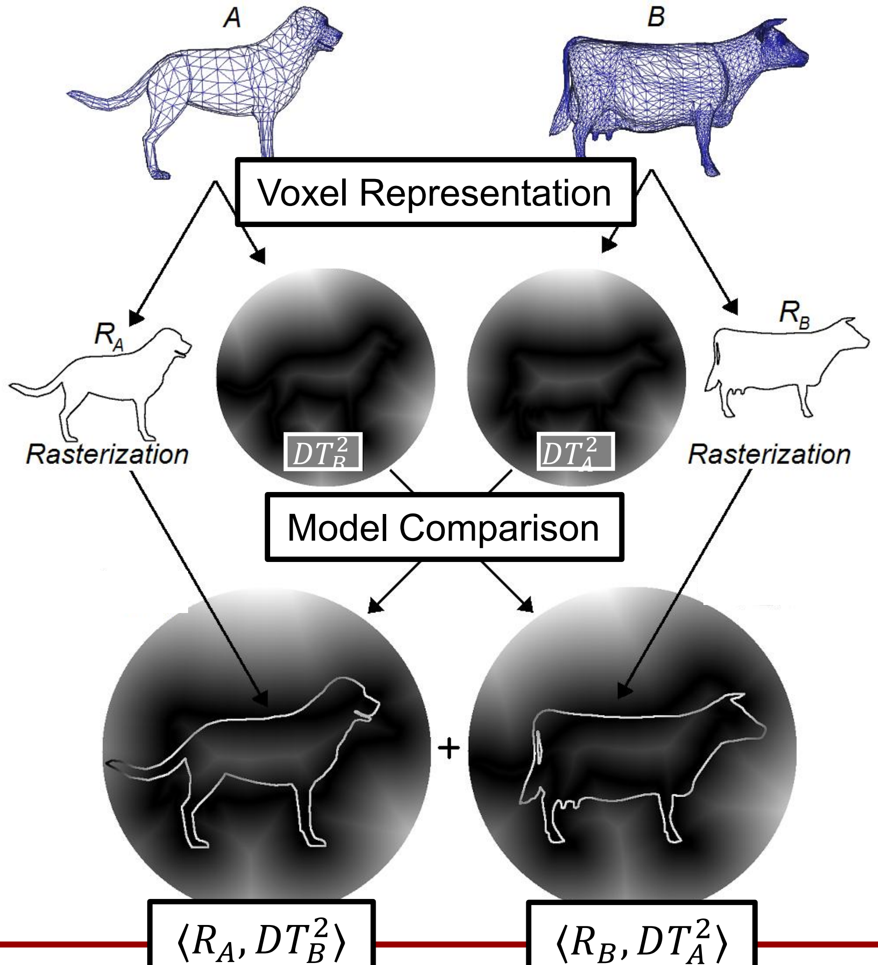
- The value of the distance transform at a 3D point is:

$$DT_A^2(p) = \min_{q \in A} \|p - q\|^2$$

Shape Matching Implementation

Run-Time:

Compute mesh similarity with two dot-products/integrals

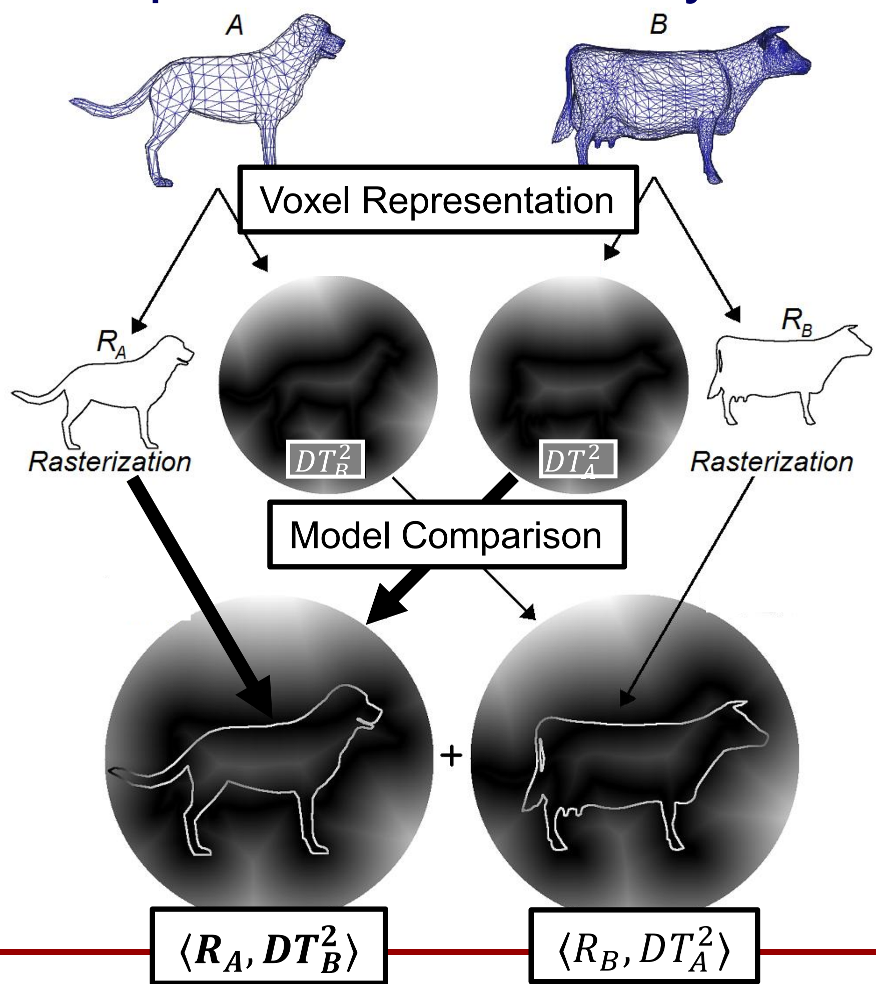


$$d(A,B) = \langle R_A, DT_B^2 \rangle + \langle DT_A^2, R_B \rangle$$

Shape Matching Implementation

Run-Time:

Compute mesh similarity with two dot-products/integrals



The dot product of R_A with DT_B^2 is the sum of the product of the two functions:

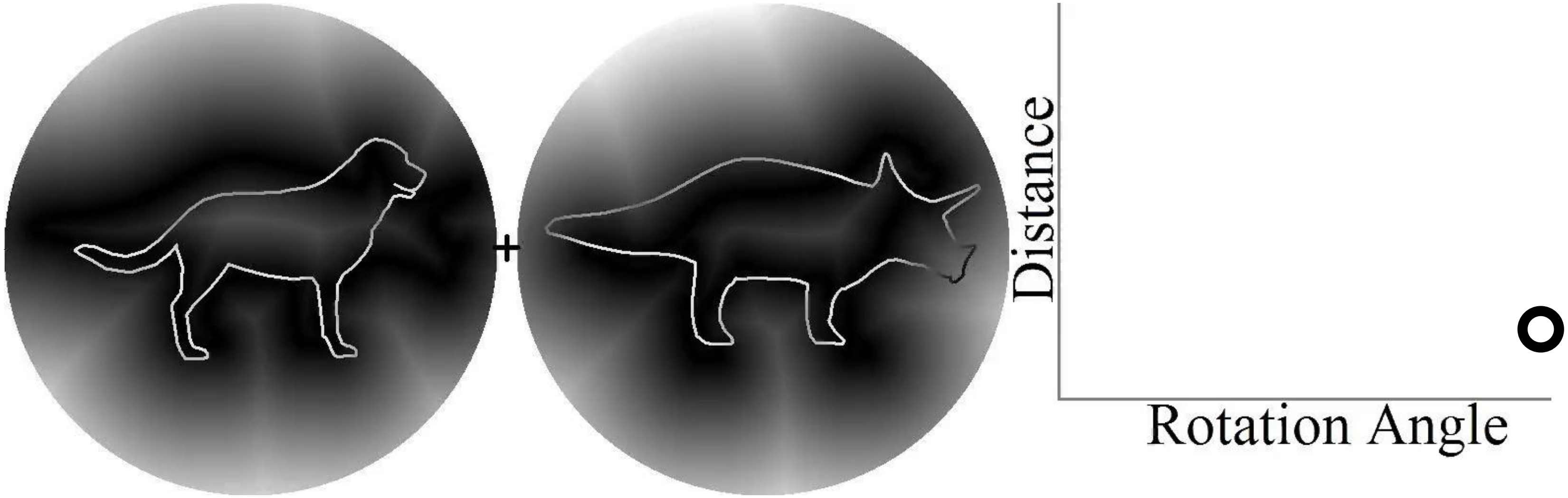
$$\begin{aligned}\langle R_A, DT_B^2 \rangle &\equiv \int_{\mathbb{R}^3} R_A(p) \cdot DT_B^2(p)dp \\ &= \int_A DT_B^2(p)dp \\ &= \int_A \min_{q \in B} \|p-q\|^2 dp\end{aligned}$$

because the rasterization R_A is equal to zero off of A and is equal to one on it.

Shape Matching Implementation

Advantages:

- ✓ Squared EDT is quick to compute
- ✓ Match surfaces without correspondences
- ✓ Can use compression techniques to reduce storage.
- ✓ Can solve for the optimal rigid-body alignment using fast signal processing techniques.



Summary

Minimum sum of squared distances descriptor:

Advantages:

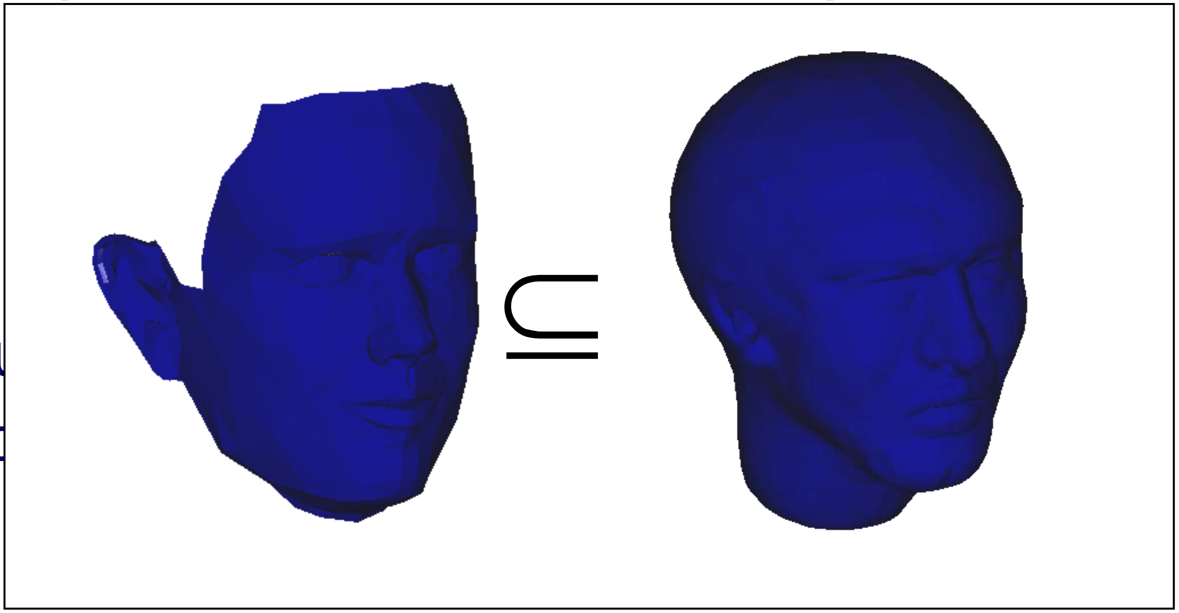
- ✓ Compact
- ✓ Discriminating
- ✓ Quick to compute
- ✓ Allows for matching over rigid body transformations

Summary

Minimum sum of squared distances descriptor:

Advantages:

- ✓ Compact
- ✓ Discriminating
- ✓ Quick to compu
- ✓ Allows for matc



Limitations:

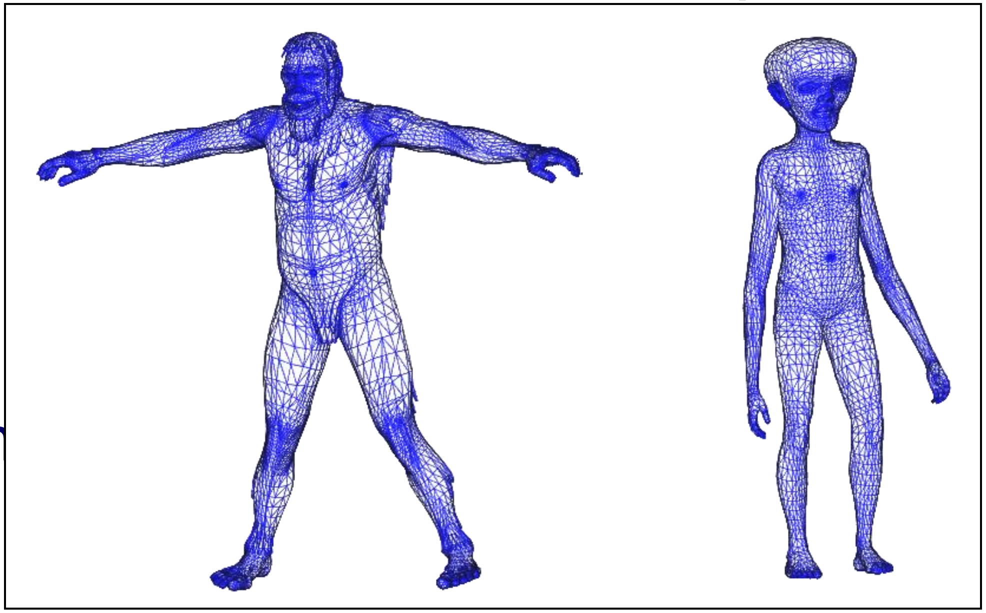
- ✗ Difficult to use for partial object matching

Summary

Minimum sum of squared distances descriptor:

Advantages:

- ✓ Compact
- ✓ Discriminating
- ✓ Quick to compute
- ✓ Allows for matchin



Limitations:

- ✗ Difficult to use for partial object matching
- ✗ Difficult to use for articulated figures

Midterm 2 Review

Michael Kazhdan

(601.457/657)

Midterm

Content:

Everything that we have covered since the first midterm:

- Radiosity
- Subdivision Surfaces
- Spline Curves/Surfaces
- Procedural Models
- Solid Models
- 3D Scanning
- Surface Reconstruction
- Animation
- Image Stitching
- Shape Matching

Midterm

Format:

- Short answer questions only
- No essays
- No True/False
- No multiple choice