Shape Matching

Michael Kazhdan
(601.457/657)
Overview

• Intro
• General Approach
• Minimum SSD Descriptor
Goal

Given a database of 3D models, and given a query shape, find the database models that are most similar to the query.
Applications

- Entertainment
- Medicine
- Chemistry/Biology
- Archaeology
- Etc.
Applications

- Entertainment
  - Model generation
- Medicine
- Chemistry/Biology
- Archaeology
- Etc.
Applications

- Entertainment
- Medicine
  - Automated diagnosis
- Chemistry/Biology
- Archaeology
- Etc.

Images courtesy of NLM
Applications

• Entertainment
• Medicine
• Chemistry/Biology
  ◦ Docking and binding
• Archaeology
• Etc.

Image Courtesy of PDB
Applications

• Entertainment
• Medicine
• Chemistry/Biology
• Archaeology
  o Reconstruction
• Etc.

Image Courtesy of Stanford
Overview

• Motivation
• General Approach
• Minimum SSD Descriptor
Shape Matching

General approach:
Define a function that takes in two models and returns a measure of their proximity.

\[ D(M_1, M_2) \leq D(M_1, M_3) \]

\( M_1 \) 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

\[ D(Q, \tilde{M}_i) \leq D(Q, \tilde{M}_j) \quad \forall i \leq j \]
Database Retrieval

- Return the closest matches

\[ D(Q, \tilde{M}_i) \leq D(Q, \tilde{M}_j) \quad \forall i \leq j \]
Overview

• Motivation

• General Approach
  ◦ Shape Descriptors

• Minimum SSD Descriptor
Shape Matching

**General approach:** Define a function that takes in two models and returns a measure of their proximity.

\[ D(M_1, M_2) \leq D(M_1, M_3) \]

\( M_1 \) is closer to \( M_2 \) than it is to \( M_3 \).
Shape Descriptors

**Shape Descriptor:**
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

3D Query → Shape Descriptor → 3D Database → Best Matches
Matching with Descriptors

Preprocessing
- Compute database descriptors

Run-Time
- Compute query descriptor
- Compare query descriptor to database descriptors

3D Query → Shape Descriptor → 3D Database → Best Matches
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

3D Query → Shape Descriptor → 3D Database → Best Matches
Shape Matching Challenge

Need shape descriptor 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 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
  - Insensitive 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
  - Insensitive 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
-Insensitive 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
- Insensitive to noise
- Insensitive to topology

Different Genus

Different Tessellations

Images courtesy of Viewpoint & Stanford
Overview

- Applications
- General Approach
- Minimum SSD Descriptor
Shape Matching Approach

Define shape dissimilarity as sum of squared distances from points on one surface to closest points on other

\[ d(A, B) = \int_A \text{dist}(p_A, B)^2 \, dp_A + \int_B \text{dist}(p_B, A)^2 \, dp_B \]
Define shape dissimilarity as sum of squared distances from points on one surface to closest points on other

\[
d(A, B) = \int_A \text{dist}(p_A, B)^2 dp_A + \int_B \text{dist}(p_B, A)^2 dp_B
\]
Shape Matching Approach

Define shape dissimilarity as sum of squared distances from points on one surface to closest points on other

\[ d(A, B) = \int_A \text{dist}(p_A, B)^2 \, dp_A + \int_B \text{dist}(p_B, A)^2 \, dp_B \]
Shape Matching Intuition

Measures “amount of work” required to deform one surface onto the other

\[ d(A, B) = \int_A \text{dist}(P_A, B)^2 dP_A + \int_B \text{dist}(P_B, A)^2 dP_B \]
Overview

• Applications

• General Approach

• Minimum SSD Descriptor
  ◦ (Euclidean) Distance Transform
(Euclidean) Distance Transform

The (Euclidean) Distance Transform (DT) of a surface is a function giving the distance of every point in space to the boundary.

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

Surface

\[ DT \]
(Euclidean) Distance Transform

Grass-Fire Algorithm:

• Think of space as a field of dry grass.

• Set fire to the boundary and measure the amount of time for the fire to reach each point.
(Euclidean) Distance Transform

Grass-Fire Algorithm:

• Think of space as a field of dry grass.

• Set fire to the boundary and measure the amount of time for the fire to reach each point. The points where the fire gets quenched define the skeleton of the shape.
Computing the EDT

Brute Force:

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

If there are $m$ surface points and we want the values on a grid of resolution $n$, the overall complexity becomes:

- $O(n^2m) \approx O(n^2 \cdot n)$ for a 2D grid
- $O(n^3m) \approx O(n^3 \cdot n^2)$ for a 3D grid
Computing the EDT

Graphics Hardware (2D):

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. Draw the cones with orthogonal projection, looking down the positive \(z\)-axis.

3. The values of the depth-buffer are the values of the EDT.

Visualization
Computing the EDT

Graphics Hardware (2D):

At the point $p_0$, the height of a right-cone with apex at $p$ is equal to the distance from $p$ to $p_0$. 

\[ \| p - p_0 \| \]
Computing the EDT

Graphics Hardware (2D):

The height of a right-cone with apex at \( p \) is equal to the distance from \( p \).

Given a collection of points:

- Draw right-cones at each point
Computing the EDT

Graphics Hardware (2D):

The height of a right-cone with apex at \( p \) is equal to the distance from \( p \).

Given a collection of points:

- Draw right-cones at each point
- View along the \( z \)-direction
Computing the EDT

Graphics Hardware (2D):

The height of a right-cone with apex at $p$ is equal to the distance from $p$.

Given a collection of points:
- Draw right-cones at each point
- View along the $z$-direction
- Read back the depth-buffer
Computing the Distance Transform

Graphics Hardware (2D):
- Draw right-cones at each point
- View along the $z$-direction
- Read back the depth-buffer
Overview

• Applications

• General Approach

• Minimum SSD Descriptor
  ◦ (Euclidean) Distance Transform
Shape Matching Implementation

Preprocessing:

- Rasterize and compute the squared distance transforms

\[ D{T}^2 \]
Shape Matching Implementation

Run-Time:

- Compute mesh similarity with two dot products

\[
\begin{align*}
&\langle R_A, DT_B^2 \rangle + \langle DT_A^2, R_B \rangle \\
&d(A, B) = \langle R_A, DT_B^2 \rangle + \langle DT_A^2, R_B \rangle
\end{align*}
\]
Shape Matching Implementation

Run-Time:
- Compute mesh similarity with two dot products

The value of the rasterization at a 3D point is:

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

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

The dot product of $R_A$ with $DT^2_B$ is the sum of the product of the two functions:

\[
\langle R_A, DT^2_B \rangle = \int_{\mathbb{R}^3} R_A(p) \cdot DT^2_B(p) dp
\]

\[
= \int_A DT^2_B(p) dp
\]

\[
= \int_A \min_{q \in B} \|p - q\|^2 dp
\]

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:
  ○ Discriminating
  ○ Quick to compute
  ○ Allows for matching over rigid body transformations

• Limitations:
  ○ Difficult to use for partial object matching.
  ○ Difficult to use for articulated figures and deformable models.
Summary

Minimum sum of squared distances descriptor:

- **Advantages:**
  - Compact
  - Discriminating
  - Quick to compute
  - Allows for matching over rigid body transformations

- **Limitations:**
  - Difficult to use for partial object matching.
  - Difficult to use for articulated figures and deformable models.
Summary

Minimum sum of squared distances descriptor:

• Advantages:
  ◦ Compact
  ◦ Discriminating
  ◦ Quick to compute
  ◦ Allows for matching over rigid body transformations

• Limitations:
  ◦ Difficult to use for partial object matching.
  ◦ Difficult to use for articulated figures and deformable models.