A Parametric Space Approach to the Computation of Multi-Scale Geometric Features

Anthousis Andreadis    Georgios Papaioannou    Pavlos Mavridis
Introduction

• Geometric Features are central in a wide range of applications
  – Example Features:
    Curvature, Shape Index
  – Example Applications:
    Object Retrieval, Registration, Stylized Rendering

• Static geometry: Pre-compute

• Dynamic/Animated: Fast-computation is challenging
Feature Computation

• We focus on the general case of features with finite local support

• Key Element
  – Vertex Adjacencies/Point Neighbors
  – N-ring, Euclidean or Geodesic Distance
Related Work

• Existing methods can be classified based on the sampling method of the geometry
  – Object space
  – Volumetric
  – Screen Space
  – Parametric space
Object space

• Data structure encoding the adjacency is required (half-edge, kD-tree etc)

• These methods do not scale well as computational complexity is directly linked to
  – Geometric density
  – Area of support

• GPU mapping is non-trivial. Existing approaches do not generalize the sampling neighborhood. [Griffin et al., 2011]
Screen space

- Sample geometric information from a 2D pixel buffer.
- Adjacencies are implied by the pixel grid
  - Trivial sampling, efficient mapping to GPU’s
- **Disadvantage:** Computations and area of support area limited to the visible point set
  - Inaccuracies, near occlusion points and at screen-space silhouettes

[Mellado et al., 2013]
Volumetric

• A volumetric representation is used (ex. level-set)
• Computational complexity now depends on the representation
• **Disadvantages**
  – Volumetric discretization is far more rough than the original surface
  – Incompatible results (ex. non-manifold surfaces)

[Museth, 2013]
Parametric Space

• Methods of this category rely on the unwrapped surface of the model on a 2D plane

• Computational complexity decoupled from the geometry

• Disadvantages
  – Neighbor discovery is not trivial
  – Cannot be performed directly on point clouds

• Existing methods are not generic
  – [Novatnack and Nishino, 2007] focus on image space techniques
  – [Hua et al.] Require specific unwrapping methodologies.
Motivation

• Design a method that is efficient, accurate and generic
  – **Efficiency**: Close to real-time even for large area of support for animated/deformable objects
    – Excludes Object Space
  – **Accuracy**: Similar results to a reference Object Space method
    – Excludes Screen Space & Volumetric
  – **Generality**: Not restricted to a specific feature, or parameterization
Method Overview

• Operates in parametric-space, but is agnostic to the actual mapping of the surface

• **Vertex Adjacencies** → **Pixel Adjacencies**
  - Not a perfect world: Chart boundaries create **discontinuities** of geometric adjacencies
Method Overview

- **Pre-Process**
  - Locate affected edges and store extra information

- **Real-Time**
  - Create Data Buffers
    - Geometry, Normal, Adjacency
  - Recreate Adjacencies and perform Computations
Data Buffers

- Information is stored in Textures
  - Geometry Buffer
    - Object space positions, Chart id
  - Normal Buffer
  - Adjacency Buffers
    - Discontinued Edges
      - Adjacent chart id
      - Corresponding chart coordinates
      - Relative Scale & Rotation
    - Local metric distortion (LMD)
      - Angular distortion
      - $u, v$ stretch factors
Data Buffer Generation

• Rasterize object triangles
  – Chart boundary edges are rasterized separately to avoid disconnected regions

• LMD factors computed using eigen-decomposition of the *first fundamental form matrix*
  – Used for the *anisotropic adjustment of scale and sampling directions*

\[
J_P^T J_P = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \quad E = \left( \frac{\partial P(u,v)}{\partial u} \right)^2 \quad G = \left( \frac{\partial P(u,v)}{\partial v} \right)^2 \\
F = \left( \frac{\partial P(u,v)}{\partial u} \right) \cdot \left( \frac{\partial P(u,v)}{\partial v} \right) \]
Sampling the Neighborhood of a Point

\[ t' = b' + R_{\theta(b\rightarrow b')} S_{s(b\rightarrow b')} (t - b) \]

\[ s(b \rightarrow b') = \left( \frac{\sigma_u(b')}{\sigma_u(b)}, \frac{\sigma_v(b')}{\sigma_v(b)} \right) \]
Sampling the Neighborhood of a Point
Monte Carlo Integration

• Geometric feature computation is usually performed with surface and volume integrals

• We estimate by Monte Carlo integration

• Generate random samples using a stratification scheme on a grid and transform them to disk using concentric mapping

• Disk samples are anisotropically scaled and rotated according to $LMD$ factors.

• We sample $A(s)$ ellipse using sample rejection based on the criterion of neighborhood $S(p)$

[Shirley and Chiu, 1997]
Monte Carlo Integration

\[ \langle I \rangle(p) = \frac{A'(s)}{N} \sum_{i=1}^{N} g(P(t_i)) \]
Adaptive Sampling

• Smooth surface areas converge faster than areas with high variance
• We use simplified two-step adaptive sampling

<table>
<thead>
<tr>
<th>Samples</th>
<th>Full</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>% AE</td>
</tr>
<tr>
<td>64</td>
<td>(a)</td>
<td>17.57ms</td>
</tr>
<tr>
<td>100</td>
<td>(b)</td>
<td>22.17ms</td>
</tr>
<tr>
<td>256</td>
<td>(c)</td>
<td>50.54ms</td>
</tr>
</tbody>
</table>
Results

• Implemented Geometric Features
  – Mean Curvature
  – Local Bending Energy
  – Normalized Sphere Volume
  – Shape Index

• Comparison with multi-core CPU object space approach using Half-Edge data structure
Results (Mean Curvature)

Reference

Embrasure
200K Triangles
340x334x330mm
10mm Radius

624ms

%AE: 1.18

Armadillo
345K Triangles
126x115x152mm
3mm Radius

1420ms

%AE: 1.41

Our Method

~22x

~25x
Results (Local Bending Energy)

Reference Our Method

Embrasure
200K Triangles
340x334x330mm
3mm Radius

113ms 21ms

%AE: 0.31

Lucy
200K Triangles
345x134x400mm
6mm Radius

360ms 57ms

%AE: 1.08

~5x

~6x
Results (Sphere Volume)

Reference

Arc
900K Triangles
250x170x136mm
6mm Radius

9410ms

Our Method

47ms

~200x

XYZ RGB Dragon
200K Triangles
200x132x90mm
3mm Radius

397ms

%AE: 0.81

52ms

%AE: 1.89

~8x
Results (Shape Index)

<table>
<thead>
<tr>
<th>Reference</th>
<th>Our Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arc</td>
<td></td>
</tr>
<tr>
<td>900K Triangles</td>
<td>5mm Radius</td>
</tr>
<tr>
<td>250x170x136mm</td>
<td></td>
</tr>
<tr>
<td>13600ms</td>
<td>129ms</td>
</tr>
<tr>
<td>~105x</td>
<td></td>
</tr>
<tr>
<td>Armadillo</td>
<td></td>
</tr>
<tr>
<td>345K Triangles</td>
<td>2mm Radius</td>
</tr>
<tr>
<td>126x115x152mm</td>
<td></td>
</tr>
<tr>
<td>2130ms</td>
<td>134ms</td>
</tr>
<tr>
<td>~16x</td>
<td></td>
</tr>
</tbody>
</table>
Results (Different Parameterizations)
Results (Scalability)

Scalability over Geometric Density

Scalability over Neighborhood Size
Thank you!

• Questions ?
• More info:
  – http://presious.eu
  – http://graphics.cs.aueb.gr