Point-Based Computer Graphics

Eurographics 2002 Tutorial T6

Organizers

Markus Gross
ETH Zürich

Hanspeter Pfister
MERL, Cambridge

Presenters

Marc Alexa
TU Darmstadt

Markus Gross
ETH Zürich

Mark Pauly
ETH Zürich

Hanspeter Pfister
MERL, Cambridge

Marc Stamminger
Bauhaus-Universität Weimar

Matthias Zwicker
ETH Zürich

Contents

Tutorial Schedule

8:30-8:45	Introduction (M. Gross)
8:45-9:45	Point Rendering (M. Zwicker)
9:45-10:00	Acquisition of Point-Sampled Geometry and Appearance I (H. Pfister)
10:00-10:30	Coffee Break
10:30-11:15	Acquisition of Point-Sampled Geometry and Appearance II (H. Pfister)
11:15-12:00	Dynamic Point Sampling (M. Stamminger)
12:00-14:00	Lunch
14:00-15:00	Point-Based Surface Representations (M. Alexa)
15:00-15:30	Spectral Processing of Point-Sampled Geometry (M. Gross)
15:30-16:00	Coffee Break
16:00-16:30	Efficient Simplification of Point-Sampled Geometry (M. Pauly)
16:30-17:15	Pointshop3D: An Interactive System for Point-Based Surface Editing (M. Pauly)
17:15-17:30	Discussion (all)

Presenters Biographies

Dr. Markus Gross is a professor of computer science and the director of the computer graphics laboratory of the Swiss Federal Institute of Technology (ETH) in Zürich. He received a degree in electrical and computer engineering and a Ph.D. on computer graphics and image analysis, both from the University of Saarbrucken, Germany. From 1990 to 1994 Dr. Gross was with the Computer Graphics Center in Darmstadt, where he established and directed the Visual Computing Group. His research interests include physics-based modeling, point based methods and multiresolution analysis. He has widely published and lectured on computer graphics and scientific visualization and he authored the book "Visual Computing", Springer, 1994. Dr. Gross has taught courses at major graphics conferences including SIGGRAPH, IEEE Visualization, and Eurographics. He is associate editor of the IEEE Computer Graphics and Applications and has served as a member of international program committees of major graphics conferences. Dr. Gross was a papers co-chair of the IEEE Visualization '99 and Eurographics 2000 conferences.

Dr. Hanspeter Pfister is Associate Director and Senior Research Scientist at MERL - Mitsubishi Electric Research Laboratories - in Cambridge, MA. He is the chief architect of VolumePro, Mitsubishi Electric's real-time volume rendering hardware for PCs. His research interests include computer graphics, scientific visualization, and computer architecture. His work spans a range of topics, including point-based rendering and modeling, 3D scanning, and computer graphics hardware. Hanspeter Pfister received his Ph.D. in Computer Science in 1996 from the State University of New York at Stony Brook. He received his M.S. in Electrical Engineering from the Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, in 1991. He is Associate Editor of the IEEE Transactions on Visualization and Computer Graphics (TVCG), member of the Executive Committee of the IEEE Technical Committee on Graphics and Visualization (TCVG), and member of the ACM, ACM SIGGRAPH, IEEE, the IEEE Computer Society, and the Eurographics Association.

Mark Pauly is currently a PhD student at the Computer Graphics Lab at ETH Zurich, Switzerland. He is working on point-based surface representations for 3D digital geometry processing, focusing on spectral methods for surface filtering and resampling. Further research activities are directed towards multiresolution modeling, geometry compression and texture synthesis of point-sampled objects.

Dr. Marc Stamminger received his PhD in computer graphics in 1999 from the University of Erlangen, Germany, for his work about finite element methods for global illumination computations. After that he worked at the Max-Planck-Institut for Computer Science (MPII) in Saarbrücken, Germany, where he headed the global illumination group. As a PostDoc in Sophia-Antipolis in France he worked on the interactive rendering and modeling of natural environments. Since 2001 he is an assistant professor at the Bauhaus-University in Weimar. His current research interests are point-based methods for complex, dynamic scenes, and interactive global illumination methods.

Matthias Zwicker is in his last year of the PhD program at the Computer Graphics Lab at ETH Zurich, Switzerland. He has developed rendering algorithms and data structures for point-based surface representations, which he presented in the papers sessions of SIGGRAPH 2000 and 2001. He has also extended this work towards high quality volume rendering. Other research interests concern compression of point-based data structures, acquisition of real world objects, and texturing of point-sampled surfaces.

Dr. Marc Alexa leads the project group "3d Graphics Computing" within the Interactive Graphics System Group, TU Darmstadt. He received his PhD and MS degrees in Computer Science with honors from TU Darmstadt. His research interests include shape modeling, transformation and animation as well as conversational user interfaces and information visualization.

Presenters Contact Information

Dr. Markus Gross
Professor
Department of Computer Science
Swiss Federal Institute of Technology (ETH)
CH 8092 Zürich
Switzerland
Phone: +41 1 632 7114
FAX: +41 1 632 1596
grossm@inf.ethz.ch
http://graphics.ethz.ch

Dr. Hanspeter Pfister
Associate Director
MERL - A Mitsubishi Electric Research Lab
201 Broadway
Cambridge, MA 02139
USA
Phone: (617) 621-7566
Fax: (617) 621-7550
pfister@merl.com
http://www.merl.com/people/pfister/

Matthias Zwicker
Department of Computer Science
Swiss Federal Institute of Technology (ETH)
CH 8092 Zürich
Switzerland
Phone: +41 1 632 7437
FAX: +41 1 632 1596
zwicker@inf.ethz.ch
http://graphics.ethz.ch

Mark Pauly
Department of Computer Science
Swiss Federal Institute of Technology (ETH)
CH 8092 Zürich

Switzerland
Phone: +41 1 632 0906
FAX: +41 1 632 1596
pauly@inf.ethz.ch
http://graphics.ethz.ch

Dr. Marc Stamminger
Bauhaus-Universität Weimar
Bauhausstr. 11
99423 Weimar
Germany
Phone: +49 3643 583733
FAX: +49 3643 583709
Marc.Stamminger@medien.uni-weimar.de

Dr. Marc Alexa
Interactive Graphics Systems Group
Technische Universität Darmstadt
Fraunhoferstr. 5
64283 Darmstadt
Germany
Phone: +49 6151 155 674
FAX: +49 6151 155 669
alexa@gris.informatik.tu-darmstadt.de
http://www.igd.fhg.de/~alexa

References

M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. Silva.
Point set surfaces. Proceedings of IEEE Visualization 2001, p. 21-28, San Diego, CA, October 2001.

O. Deussen, C. Colditz, M. Stamminger, G. Drettakis, Interactive visualization of complex plant ecosystems. Proceedings of IEEE Visualization 2002, *to appear*, Boston, MA, October 2002.

W. Matusik, H. Pfister, P. Beardsley, A. Ngan, R. Ziegler, L. McMillan, Image-based 3D photography using opacity hulls. Proceedings of SIGGRAPH 2002, *to appear*, San Antonio, TX, July 2002.

W. Matusik, H. Pfister, A. Ngan, R. Ziegler, L. McMillan, Acquisition and rendering of transparent and refractive objects. Thirteenth Eurographics Workshop on Rendering, *to appear*, Pisa, Italy, June 2002.

M. Pauly, M. Gross, Spectral processing of point-sampled geometry. Proceedings of SIGGRAPH 2001, p. 379-386, Los Angeles, CA, August 2001.

M. Pauly, M. Gross, Efficient Simplification of Point-Sampled Surfaces. IEEE Proceedings of Visualization 2002, *to appear*, Boston, MA, October 2002.

H. Pfister, M. Zwicker, J. van Baar, M. Gross, Surfels - surface elements as rendering primitives. Proceedings of SIGGRAPH 2000, p. 335-342, New Orleans, LS, July 2000.

M. Stamminger, G. Drettakis, Interactive sampling and rendering for complex and procedural geometry, Rendering Techniques 2001, Proceedings of the Eurographics Workshop on Rendering 2001, June 2001.

L. Ren, H. Pfister, M. Zwicker, Object space EWA splatting: a hardware accelerated approach to high quality point rendering. Proceedings of the Eurographics 2002, to appear, Saarbrücken, Germany, September 2002.

M. Zwicker, H. Pfister, J. van Baar, M. Gross, EWA volume splatting. Proceedings of IEEE Visualization 2001, p. 29-36, San Diego, CA, October 2001.

M. Zwicker, H. Pfister, J. van Baar, M. Gross, Surface splatting. Proceedings of SIGGRAPH 2001, p. 371-378, Los Angeles, CA, August 2001.

M. Zwicker, H. Pfister, J. van Baar, M. Gross, EWA splatting. IEEE Transactions on Visualization and Computer Graphics, *to appear.*

M. Zwicker, M. Pauly, O. Knoll, M. Gross, Pointshop 3D: an interactive system for point-based surface editing. Proceedings of SIGGRAPH 2002, *to appear*, San Antonio, TX, July 2002

Project Pages

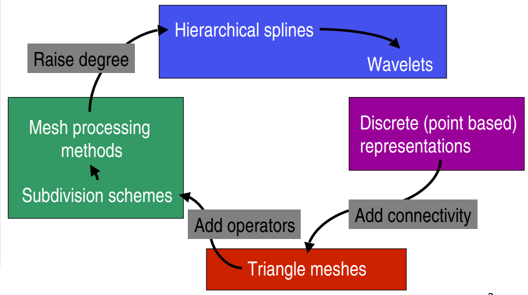
- Rendering
  http://graphics.ethz.ch/surfels
- Acquisition
  http://www.merl.com/projects/3Dimages/
- Dynamic sampling
  http://www-sop.inria.fr/reves/personnel/Marc.Stamminger/pbr.html
- Processing, sampling and filtering
  http://graphics.ethz.ch/points
- Pointshop3D
  http://www.pointshop3d.com

Point-Based Computer Graphics

Eurographics 2002 Tutorial T6

Marc Alexa, Markus Gross,
Mark Pauly, Hanspeter Pfister,
Marc Stamminger, Matthias Zwicker

Surf. Reps. for Graphics



Polynomials...

- ✓ Rigorous mathematical concept
- ✓ Robust evaluation of geometric entities
- ✓ Shape control for smooth shapes
- ✓ Advanced physically-based modeling

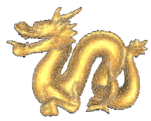
- × Require parameterization
- × Discontinuity modeling
- × Topological flexibility

Refine h rather than p !

Polynomials -> Triangles

- Piecewise linear approximations
- Irregular sampling of the surface
- Forget about parameterization

Triangle meshes



- Multiresolution modeling
- Compression
- Geometric signal processing

Triangles...

- ✓ Simple and efficient representation
- ✓ Hardware pipelines support Δ
- ✓ Advanced geometric processing is being in sight
- ✓ The widely accepted queen of graphics primitives

- × Sophisticated modeling is difficult
- × (Local) parameterizations still needed
- × Complex LOD management
- × Compression and streaming is highly non-trivial

Remove connectivity !

Triangles -> Points

- From piecewise linear functions to Delta distributions
- Forget about connectivity

Point clouds

- Points are natural representations within 3D acquisition systems
- Meshes provide an articificial enhancement of the acquired point samples

History of Points in Graphics

- Particle systems [Reeves 1983]
- Points as a display primitive [Whitted, Levoy 1985]
- Oriented particles [Szeliski, Tonnesen 1992]
- Particles and implicit surfaces [Witkin, Heckbert 1994]
- Digital Michelangelo [Levoy et al. 2000]
- Image based visual hulls [Matusik 2000]
- Surfels [Pfister et al. 2000]
- QSplat [Rusinkiewicz, Levoy 2000]
- Point set surfaces [Alexa et al. 2001]
- Radial basis functions [Carr et al. 2001]
- Surface splatting [Zwicker et al. 2001]
- Randomized z-buffer [Wand et al. 2001]
- Sampling [Stamminger, Drettakis 2001]
- Opacity hulls [Matusik et al. 2002]
- Pointshop3D [Zwicker, Pauly, Knoll, Gross 2002]...?

7

The Purpose of our Course is …

I) …to introduce points as a versatile and powerful graphics primitive

II) …to present state of the art concepts for acquisition, representation, processing and rendering of point sampled geometry

III) …to stimulate YOU to help us to further develop Point Based Graphics

8

Taxonomy

Rendering (Zwicker)

Acquisition (Pfister, Stamminger)

Point-Based Graphics

Representation (Alexa)

Processing & Editing (Gross, Pauly)

9

Morning Schedule

8:30-8:45	Introduction (M. Gross)
8:45-9:45	Point Rendering (M. Zwicker)
9:45-10:00	Acquisition of Point-Sampled Geometry and Appearance I (H. Pfister)
10:00-10:30	Coffee Break
10:30-11:15	Acquisition of Point-Sampled Geometry and Appearance II (H. Pfister)
11:15-12:00	Dynamic Point Sampling (M. Stamminger)

10

Afternoon Schedule

14:00-15:00	Point-Based Surface Representations (M. Alexa)
15:00-15:30	Spectral Processing of Point-Sampled Geometry (M. Gross)
15:30-16:00	Coffee Break
16:00-16:30	Efficient Simplification of Point-Sampled Geometry (M. Pauly)
16:30-17:15	Pointshop3D: An Interactive System for Point-Based Surface Editing (M. Pauly)
17:15-17:30	Discussion (all)

11

Point-Based Rendering

Matthias Zwicker
Computer Graphics Lab
ETH Zürich

Point-Based Rendering

- Introduction and motivation
- Surface elements
- Rendering
- Antialiasing
- Hardware Acceleration
- Conclusions

Motivation 1

Quake 2
1998

Nvidia GeForce4
2002

Motivation 1

- Performance of 3D hardware has exploded (e.g., GeForce4: 136 million vertices per second)
- Projected triangles are very small (i.e., cover only a few pixels)
- Overhead for triangle setup increases (initialization of texture filtering, rasterization)

➡ A simpler, more efficient rendering primitive than triangles?

Motivation 2

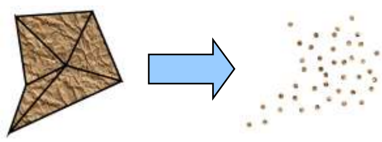
- Modern 3D scanning devices (e.g., laser range scanners) acquire huge point clouds
- Generating consistent triangle meshes is time consuming and difficult

➡ A rendering primitive for direct visualization of point clouds, without the need to generate triangle meshes?

4 million pts.
[Levoy et al. 2000]

Points as Rendering Primitives

- Point clouds instead of triangle meshes [Levoy and Whitted 1985, Grossman and Dally 1998, Pfister et al. 2000]



triangle mesh (with textures)

point cloud

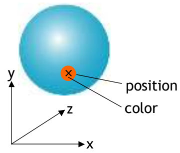
Point-Based Surface Representation

- Points are ***samples*** of the surface
- The point cloud describes:
  - 3D geometry of the surface
  - Surface reflectance properties (e.g., diffuse color, etc.)
- There is no additional information, such as
  - connectivity (i.e., explicit neighborhood information between points)
  - texture maps, bump maps, etc.

Surface Elements - Surfels

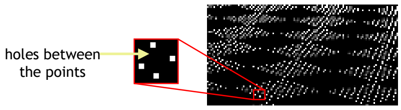
- Each point corresponds to a surface element, or ***surfel***, describing the surface in a small neighborhood
- Basic surfels:

```
BasicSurfel {
  position;
  color;
}
```



Surfels

- How to represent the surface between the points?

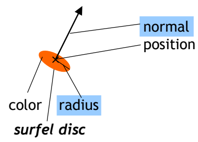


- Surfels need to ***interpolate*** the surface between the points
- A certain ***surface area*** is associated with each surfel

Surfels

- Surfels can be extended by storing additional attributes
- This allows for higher quality rendering or advanced shading effects

```
ExtendedSurfel {
  position;
  color;
  normal;
  radius;
  etc...
}
```



Surfels

- Surfels store essential information for ***rendering***
- Surfels are primarily designed as a ***point rendering primitive***
- They do not provide a mathematically smooth surface definition (see [Alexa 2001], point set surfaces)

Model Acquisition

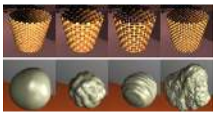
- 3D scanning of physical objects
  - See Pfister, acquisition
  - Direct rendering of acquired point clouds
  - No mesh reconstruction necessary


[Matusik et al. 2002]

Model Acquisition

- Sampling synthetic objects
  - Efficient rendering of complex models
  - Dynamic sampling of procedural objects and animated scenes (see Stamminger, dynamic sampling)

[Zwicker et al. 2001]

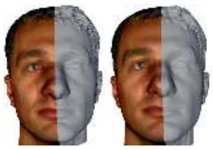


[Stamminger et al. 2001]

Point-Based Computer Graphics Your Name 13

Model Acquisition

- Processing and editing of point-sampled geometry



spectral processing
[Pauly, Gross 2002]
(see Gross, spectral processing)

point-based surface editing
[Zwicker et al. 2002]
(see Pauly, Pointshop3D)

Point-Based Computer Graphics Your Name 14

Point Rendering Pipeline

Point Cloud → Forward Warping → Filtering and Shading → Visibility → Image Reconstruction → Frame-buffer

- Simple, pure forward mapping pipeline
- Surfels carry all information through the pipeline („surfel stream")
- No texture look-ups
- Framebuffer stores RGB, alpha, and Z

Point-Based Computer Graphics Your Name 15

Point Rendering Pipeline

Forward Warping → Filtering and Shading → Visibility → Image Reconstruction

- Perspective projection of each point in the point cloud
- Analogous to projection of triangle vertices
  - homogeneous matrix-vector product
  - perspective division

Point-Based Computer Graphics Your Name 16

Point Rendering Pipeline

Forward Warping → **Filtering and Shading** → Visibility → Image Reconstruction

- Per-point shading
- Conventional models for shading (Phong, Torrance-Sparrow, reflections, etc.)
- High quality antialiasing is an advanced topic discussed later in the course

Point-Based Computer Graphics Your Name 17

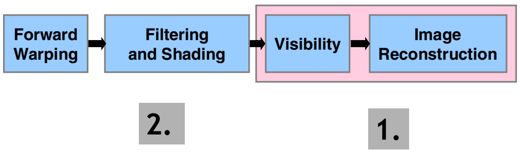
Point Rendering Pipeline

Forward Warping → Filtering and Shading → **Visibility → Image Reconstruction**

- Visibility and image reconstruction is performed simultaneously
  - Discard points that are occluded from the current viewpoint
  - Reconstruct continuous surfaces from projected points

Point-Based Computer Graphics Your Name 18

Overview



Visibility and Image Reconstruction

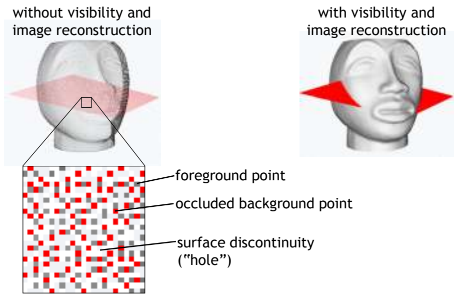
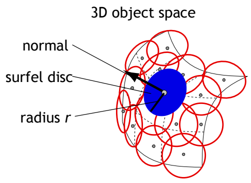


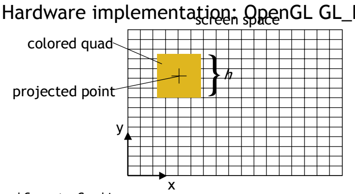
Image Reconstruction

- Goal: avoid holes
- Use surfel disc radius *r* to cover surface completely



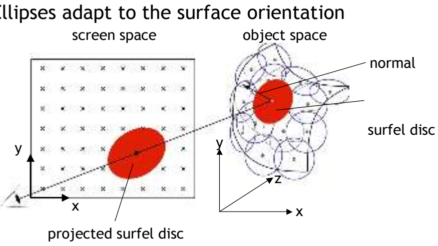
Quad Rendering Primitive

- Draw a colored quad centered at the projected point
- The quad side length is *h*, where *h* = 2 * *r* * *s*
- The scaling factor *s* given by perspective projection and viewport transformation
- Hardware implementation: OpenGL GL_POINTS



Projected Disc Rendering Primitive

- Project surfel discs from object to screen space
- Projecting discs results in ellipses in screen space
- Ellipses adapt to the surface orientation

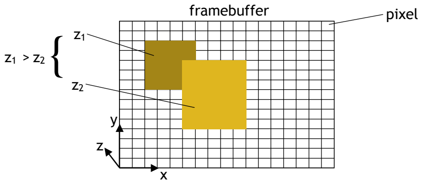


Comparison

- Quad primitive
  - Low image quality (primitives do not adapt to surface orientation)
  - Efficient rendering
  - Supported by conventional 3D accelerator hardware (OpenGL GL_POINTS)
- Projected disc primitive
  - Higher image quality (primitives adapt to surface orientation)
  - Not directly supported by graphics hardware
  - Higher computational cost

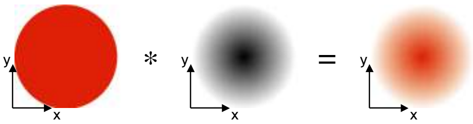
Visibility: Z-Buffering

- ***No blending*** of rendering primitives



Splatting

- A splat primitive consists of a colored point primitive and an alpha mask



colored point primitive *c*

alpha mask *w(x,y)* (often a 2D Gauss function)

splat primitive *c * w(x,y)*

Splatting

- The final color *c(x,y)* is computed by ***additive alpha blending***, i.e., by computing the weighted sum

color of splat *i* alpha of splat *i* at position *(x,y)*

$$c(x,y) = \frac{\sum_i c_i w_i(x,y)}{\sum_i w_i(x,y)}$$

- Normalization is necessary, because the weights do not sum up to one with irregular point distributions

$$\sum_i w_i(x,y) \neq 1$$

Splatting

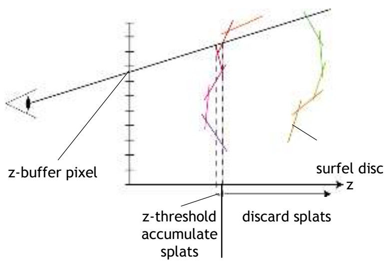
without normalization with normalization

varying brightness because of irregular point distribution

no artifacts

Splatting

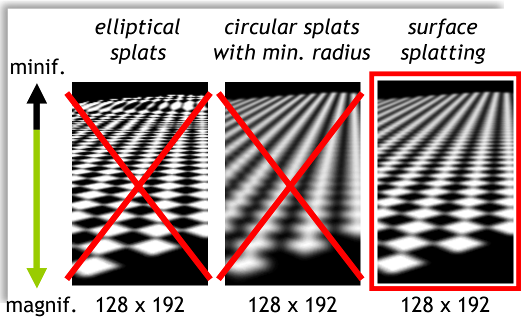
- Extended z-buffering



Extended Z-Buffering

```
DepthTest(x,y) {
  if (abs(splat z - z(x,y)) < threshold) {
    c(x,y) = c(x,y) + splat color
    w(x,y) = w(x,y) + splat w(x,y)
  } else if (splat z < z(x,y)) {
    z(x,y) = splat z
    c(x,y) = splat color
    w(x,y) = splat w(x,y)
  }
}
```

Splatting Comparison



High Quality Splatting

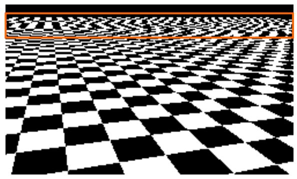
- High quality splatting requires careful analysis of ***aliasing*** issues
  - Review of signal processing theory
  - Application to point rendering
  - Surface splatting [Zwicker et al. 2001]

Aliasing in Computer Graphics

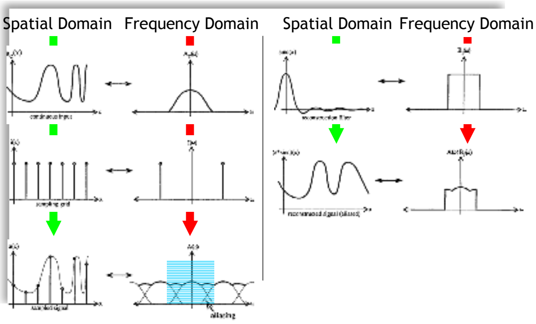
- Aliasing = Sampling of continuous functions below the ***Nyquist frequency***
  - To avoid aliasing, sampling rate must be twice as high as the maximum frequency in the signal
- Aliasing effects:
  - Loss of detail
  - Moire patterns, jagged edges
  - Disintegration of objects or patterns
- Aliasing in Computer Graphics
  - Texture Mapping
  - Scan conversion of geometry

Aliasing in Computer Graphics

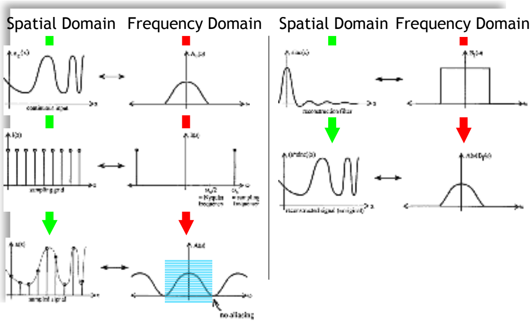
- Aliasing: high frequencies in the input signal appear as low frequencies in the reconstructed signal



Occurrence of Aliasing



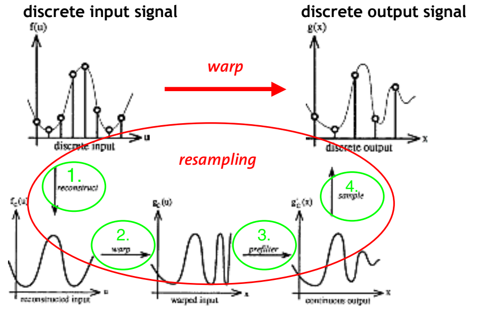
Aliasing-Free Reconstruction



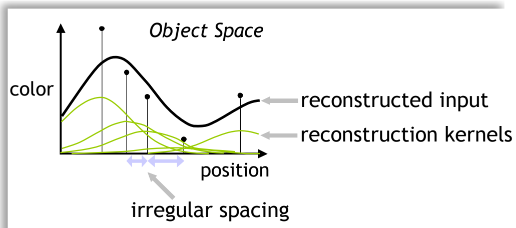
Antialiasing

- Prefiltering
  - Band-limit the continuous signal before sampling
  - Eliminates all aliasing (with an ideal low-pass filter)
  - Closed form solution not available in general
- Supersampling
  - Raise sampling rate
  - Reduces, but does not eliminate all aliasing artifacts (in practice, many signals have infinite frequencies)
  - Simple implementation (hardware)

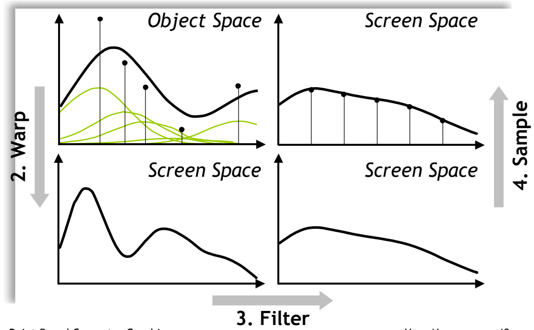
Resampling



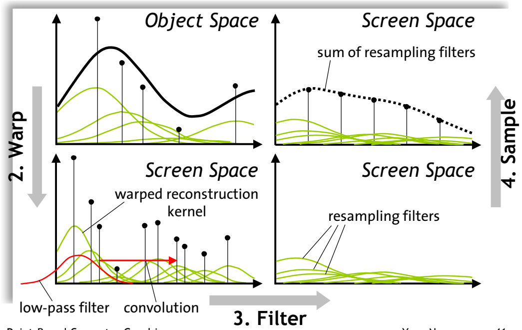
Resampling Filters



Resampling Filters



Resampling Filters

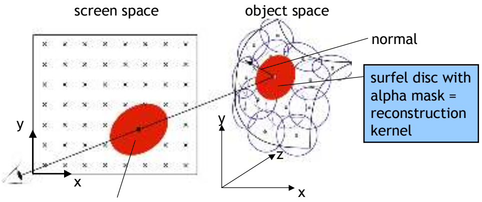


Resampling

- Resampling in the context of surface rendering
  - Discrete input function = surface texture (discrete 2D function)
  - Warping = projecting surfaces to the image plane (2D to 2D projective mapping)

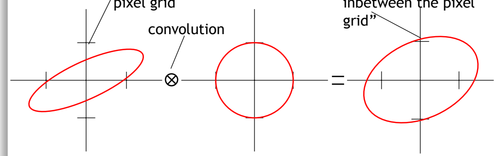
2D Reconstruction Kernels

- Warping a 2D reconstruction kernel is equivalent to projecting a surfel disc with alpha mask



Resampling Filters

- A resampling filter is a convolution of a warped reconstruction filter and a low-pass filter



Mathematical Formulation

$$c(x,y) = \sum_k c_k r_k(m^{-1}(x,y)) \otimes h(x,y)$$

pixel color

warping function

low pass filter

reconstruction kernel

reconstruction kernel color

Gaussian Resampling Filters

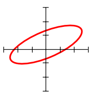
- Gaussians are closed under linear warping and convolution
- With Gaussian reconstruction kernels and low-pass filters, the resampling filter is a Gaussian, too
- Efficient rendering algorithms (***surface splatting*** [Zwicker et al. 2001])

Mathematical Formulation

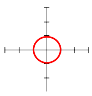
$$c(x,y) = \sum_k c_k r_k(m^{-1}(x,y)) \otimes h(x,y)$$

Gaussian reconstruction kernel

Gaussian low-pass filter



screen space



screen space

Mathematical Formulation

$$c(x,y) = \sum_k c_k r_k(m^{-1}(x,y)) \otimes h(x,y)$$

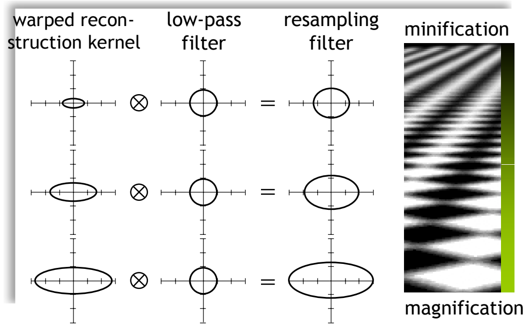
$$= \sum_k c_k G_k(x,y)$$

Gaussian resampling filter

Algorithm

```
for each point P {
  project P to screen space;
  shade P;
  determine resampling kernel G;
  splat G;
}
for each pixel {
  normalize;
}
```


Properties of 2D Resampling Filters



Hardware Implementation

- Based on the object space formulation of EWA filtering
- Implemented using textured triangles
- All calculations are performed in the programmable hardware (extensive use of vertex shaders)
- Presented at EG 2002 ([Ren et al. 2002])

Surface Splatting Performance

- Software implementation
  - 500 000 splats/sec on 866 MHz PIII
  - 1 000 000 splats/sec on 2 GHz P4
- Hardware implementation [Ren et al. 2002]
  - Uses texture mapping and vertex shaders
  - 3 000 000 splats/sec on GeForce4 Ti 4400

Conclusions

- Points are an efficient rendering primitive for highly complex surfaces
- Points allow the direct visualization of real world data acquired with 3D scanning devices
- High performance, low quality point rendering is supported by 3D hardware (tens of millions points per second)
- High quality point rendering with anisotropic texture filtering is available
  - 3 million points per second with hardware support
  - 1 million points per second in software
- Antialiasing technique has been extended to volume rendering

Applications

- Direct visualization of point clouds
- Real-time 3D reconstruction and rendering for virtual reality applications
- Hybrid point and polygon rendering systems
- Rendering animated scenes
- Interactive display of huge meshes
- On the fly sampling and rendering of procedural objects

Future Work

- Dedicated rendering hardware
- Efficient approximations of exact EWA splatting
- Rendering architecture for on the fly sampling and rendering

References

- [Levoy and Whitted 1985] The use of points as a display primitive, technical report, University of North Carolina at Chapel Hill, 1985
- [Heckbert 1986] Fundamentals of texture mapping and image warping, Master's Thesis, 1986
- [Grossman and Dally 1998] Point sample rendering, Eurographics workshop on rendering, 1998
- [Levoy et al. 2000] The digital Michelangelo project, SIGGRAPH 2000
- [Rusinkiewicz et al. 2000] Qsplat, SIGGRAPH 2000
- [Pfister et al. 2000] Surfels: Surface elements as rendering primitives, SIGGRAPH 2000
- [Zwicker et al. 2001] Surface splatting, SIGGRAPH 2001
- [Zwicker et al. 2002] EWA Splatting, to appear, IEEE TVCG 2002
- [Ren et al. 2002] Object space EWA splatting: A hardware accelerated approach to high quality point rendering, Eurographics 2002

Acquisition of Point-Sampled Geometry and Appearance

Hanspeter Pfister, MERL
pfister@merl.com

Wojciech Matusik, MIT
Addy Ngan, MIT
Paul Beardsley, MERL
Remo Ziegler, MERL
Leonard McMillan, MIT

The Goal: To Capture Reality

- Fully-automated 3D model creation of real objects.
- Faithful representation of appearance for these objects.

Image-Based 3D Photography

- An image-based 3D scanning system.
  - Handles fuzzy, refractive, transparent objects.
  - Robust, automatic
  - Point-sampled geometry based on the *visual hull*.
  - Objects can be rendered in novel environments.

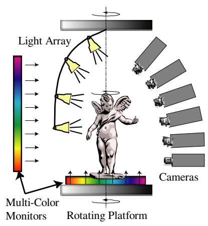
Previous Work

- Active and passive 3D scanners
  - Work best for diffuse materials.
  - Fuzzy, transparent, and refractive objects are difficult.
- BRDF estimation, inverse rendering
- Image based modeling and rendering
  - Reflectance fields [Debevec et al. 00]
    - Light Stage system to capture reflectance fields
    - Fixed viewpoint, no geometry
  - Environment matting [Zongker et al. 99, Chuang et al. 00]
    - Capture reflections and refractions
    - Fixed viewpoint, no geometry

Outline

- Overview
- ➢ System
- Geometry
- Reflectance
- Rendering
- Results

The System



Outline

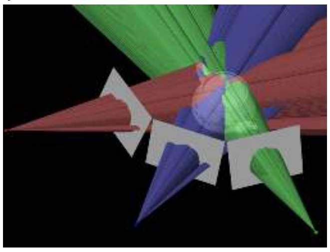
- Overview
- System
- ➢ Geometry
- Reflectance
- Rendering
- Results

Acquisition

- For each viewpoint (6 cameras x 72 positions)
  - Alpha mattes
    - Use multiple backgrounds [Smith and Blinn 96]
  - Reflectance images
    - Pictures of the object under different lighting
      (4 lights x 11 positions)
  - Environment mattes
    - Use similar techniques as [Chuang et al. 2000]

Geometry – Opacity Hull

- Visual hull augmented with view-dependent opacity.



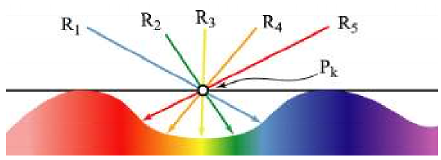
Approximate Geometry

- The approximate visual hull is augmented by radiance data to render concavities, reflections, and transparency.

Geometry Example

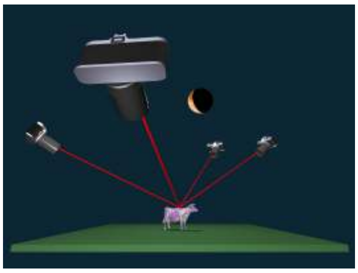
Surface Light Fields

- A surface light field is a function that assigns a color to each ray originating on a surface. [Wood *et al.*, 2000]



Shading Algorithm

- A view-dependent strategy.



Color Blending

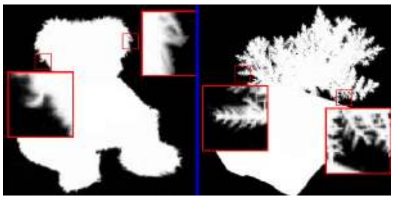
- Blend colors based on angle between virtual camera and stored colors.
- Unstructured Lumigraph Rendering
  [Buehler et al., SIGGRAPH 2001]
- View-Dependent Texture Mapping
  [Debevec, EGRW 98]

Point-Based Rendering

- Point-based rendering using LDC tree, visibility splatting, and view-dependent shading.

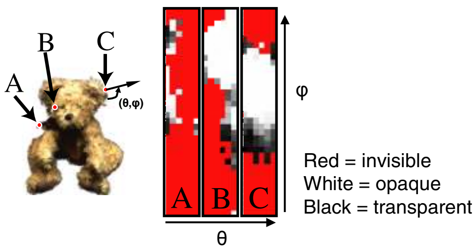
Geometry – Opacity Hull

- Store the opacity of each observation at each point on the visual hull [Matusik et al. SIG2002].



Geometry – Opacity Hull

- Assign view-dependent opacity to each ray originating on a point of the visual hull.



Example

Results

- Point-based rendering using EWA splatting, A-buffer blending, and edge antialiasing.

Opacity Hull – Discussion

- View dependent opacity vs. geometry trade-off.
  - Similar to radiance vs. geometry trade-off.
- Sometimes acquiring the geometry is not possible (e.g. resolution of the acquisition device is not adequate).
- Sometimes representing true geometry would be very inefficient (e.g. hair, trees).
- Opacity hull stores the "macro" effect.

Point-Based Models

- No need to establish topology or connectivity.
- No need for a consistent surface parameterization for texture mapping.
- Represent organic models (feather, tree) much more readily than polygon models.
- Easy to represent view-dependent opacity and radiance per surface point.

Outline

- Overview
- Previous Works
- Geometry
- ➢ Reflectance
- Rendering
- Results

Light Transport Model

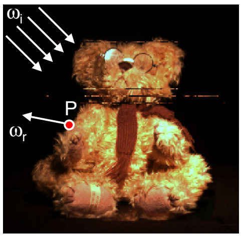
- Assume illumination originates from infinity.
- The light arriving at a camera pixel can be described as:

$$C(x,y) = \int_{\Omega} W(\omega) E(\omega) d\omega$$

C(x,y) - the pixel value
E - the environment
W - the *reflectance field*

Surface Reflectance Fields

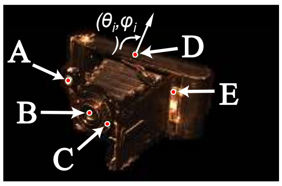
- 6D function: $W(P,\omega_i,\omega_r) = W(u_r,v_r;\theta_i,\Phi_i;\theta_r,\Phi_r)$

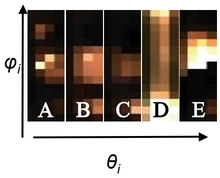


Reflectance Functions

- For each viewpoint, 4D function:

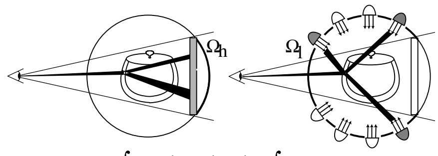
$$W_{xy}(\omega_i) = W(x, y; \theta_i, \Phi_i)$$





Reflectance Field Acquisition

- We separate the hemisphere into high resolution Ω_h and low resolution Ω_l [Matusik et al., EGRW2002].



$$C(x,y) = \int_{\Omega_h} W_h(\xi)T(\xi)d\xi + \int_{\Omega_l} W_l(\omega_i)L(\omega_i)d\omega$$

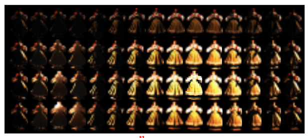
Acquisition

- For each viewpoint (6 cameras x 72 positions)
  - Alpha mattes
    - Use multiple backgrounds [Smith and Blinn 96]
  - Reflectance images ⟵ Low resolution
    - Pictures of the object under different lighting (4 lights x 11 positions)
  - Environment mattes ⟵ High resolution
    - Use similar techniques as [Chuang et al. 2000]

Low-Resolution Reflectance Field

$$C(x,y) = \int_{\Omega_h} W_h(\xi)T(\xi)d\xi + \int_{\Omega_l} W_l(\omega_i)L(\omega_i)d\omega$$

- W_l sampled by taking pictures with each light turned on at a time [Debevec et al 00].



$$\int_{\Omega_l} W_l(\omega_i)L(\omega_i)d\omega \approx \sum_{i=1}^{n} W_i L_i \quad \text{for n lights}$$

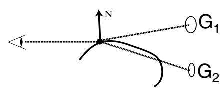
Compression

- Subdivide images into 8 x 8 pixel blocks.
- Keep blocks containing the object (avg. compression 1:7)
- PCA compression (avg. compression 1:10)

High-Resolution Reflectance Field

$$C(x,y) = \int_{\Omega_h} W_h(\xi)T(\xi)d\xi + \int_{\Omega_l} W_l(\omega_i)L(\omega_i)d\omega$$

- Use techniques of environment matting [Chuang et al., SIGGRAPH 00].
- Approximate W_h by a sum of up to two Gaussians:
  - Reflective G_1.
  - Refractive G_2.



$$W_h(\xi) = a_1 G_1 + a_2 G_2$$

Surface Reflectance Fields

- Work without accurate geometry.
- Surface normals are not necessary.
- Capture more than reflectance:
  - Inter-reflections
  - Subsurface scattering
  - Refraction
  - Dispersion
  - Non-uniform material variations
- Simplified version of the BSSRDF [Debevec et al., 00].

Outline

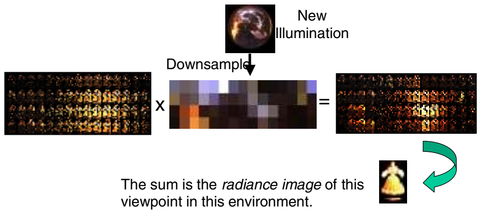
- Overview
- Previous Works
- Geometry
- Reflectance
- ➢ Rendering
- Results

Rendering

- Input: Opacity hull, reflectance data, new environment
- Create *radiance* images from environment and low-resolution reflectance field.
- Reparameterize environment mattes.
- Interpolate data to new viewpoint.

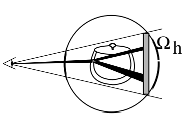
1st Step: Relighting Ω_l

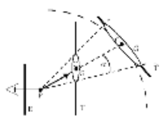
- Compute *radiance image* for each viewpoint.



2nd Step: Reproject Ω_h

- Project environment mattes onto the new environment.
  - Environment mattes acquired was parameterized on plane T (the plasma display).
  - We need to project the Gaussians to the new environment map, producing new Gaussians.





3rd Step: Interpolation

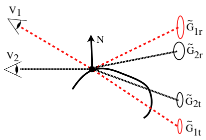
- From new viewpoint, for each surface point, find four nearest acquired viewpoints.
  - Store visibility vector per surface point.
- Interpolate using unstructured lumigraph interpolation [Buehler et al., SIGGRAPH 01] or view-dependent texture mapping [Debevec 96].
  - Opacity.
  - Contribution from low-res reflectance field (in the form of radiance images).
  - Contribution from high-res reflectance field.

3rd Step: Interpolation

- For low-res reflectance field, we interpolate the RGB color from the radiance images.

For high-resolution reflectance field:

Interpolate *direction* of reflection/refraction.
Interpolate other parameters of the Gaussians.
Convolve with the environment.



Outline

- Overview
- Previous Works
- Geometry
- Reflectance
- Rendering
- ➢ Results

Results

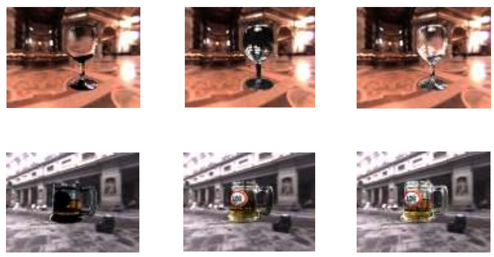
- Performance for 6x72 = 432 viewpoints
- 337,824 images taken in total !!
  - Acquisition (47 hours)
    - Alpha mattes – 1 hour
    - Environment mattes – 18 hours
    - Reflectance images – 28 hours
  - Processing
    - Opacity hull ~ 30 minutes
    - PCA Compression ~ 20 hours (MATLAB, unoptimized)
  - Rendering ~ 5 minutes per frame
  - Size
    - Opacity hull ~ 30 - 50 MB
    - Environment mattes ~ 0.5 - 2 GB
    - Reflectance images ~ Raw 370 GB / Compressed 2 - 4 GB

Results

Results

Results

High-resolution Ω_h Low-resolution Ω_l Combined



Results

Results – Ω_h

Results – Ω_l

Results – Combined

Results

Results

Conclusions

- A fully automatic system that is able to capture and render any type of object.
- Opacity hulls combined with lightfields / surface reflectance fields provide realistic 3D graphics models.
- Point-based rendering offers easy surface parameterization of acquired models.
- Separation of surface reflectance fields into high- and low-resolution areas is practical.
- New rendering algorithm for environment matte interpolation.

Future Directions

- Use more than 2 Gaussians for the environment mattes.
- Better compression.
- Real-time rendering.

Acknowledgements

- Colleagues:
  - MIT: Chris Buehler, Tom Buehler.
  - MERL: Bill Yerazunis, Darren Leigh, Michael Stern.
- Thanks to:
  - David Tames, Jennifer Roderick Pfister.
- NSF grants CCR-9975859 and EIA-9802220.
- Papers available at:
  - http://www.merl.com/people/pfister/

dynamic point sampling

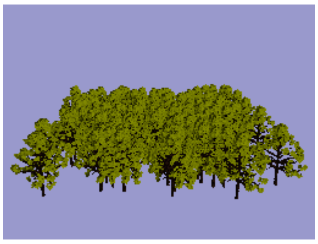
Marc Stamminger

motivation

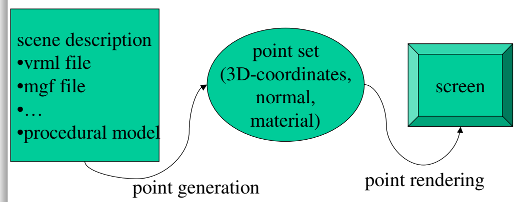
- tree created by AMAP
- 150,000 triangles
- 8 fps

motivation

- level of detail
- 100 trees
- 270,000 points
- 20 fps



point rendering pipeline



point generation

- Surfels
  (Pfister et al., SIG2000)
  - (orthographic) views
- Q-Splat
  (Rusinkiewicz et al.,SIG2000)
  - filtered triangle mesh hierarchy
- Randomized z-Buffer
  (Wand et al., SIG2001)
  - random points

point rendering

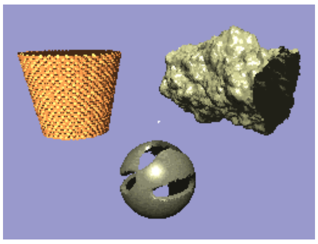
- in software
  - filtering
  - texturing
  - hole filling
- in hardware
  - as points
  - as polygonal disks
  - as splats

our approach

- dynamic point generation for
  - procedural objects
  - terrains
  - complex dynamic objects
  - point rendering with OpenGL's GL_POINT
  - very fast (> 10^7 points per second)
  - OpenGL does lighting

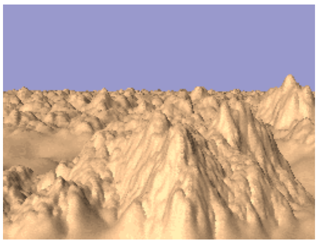
results

- points are well suited for
- procedural geometry



results

- points are well suited for
- procedural geometry
- terrains



results

- points are well suited for
- procedural geometry
- terrains
- complex geometry

results

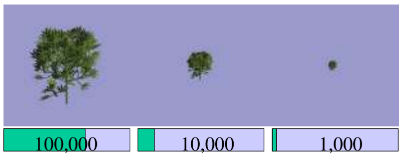
- points are well suited for
- procedural geometry
- terrains
- complex geometry
- combinations

results

- points are well suited for
- procedural geometry
- terrains
- complex geometry
- combinations
- eco systems

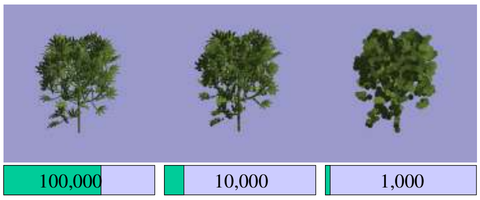
complex polygonal geometry

- generate list of randomly distributed samples
- for every frame: compute *n*, render the first *n*



complex polygonal geometry

- easy speed / quality trade off
- frame rate control



sample densities

- adapt point densities to **image space (2D)**
- or: adapt to **post-perspective space (3D)**

densities complex geometry

- world space -> post-perspective:
  - area decreases by squared distance
  - goal: uniform post-perspective point density
  - point number ~ area/d^2

modified complex geometry

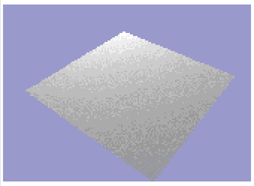
- simple modifications on the fly

30 fps

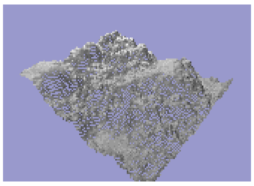
complex geometry

- video „complex geometry"
- download at
  `http://www-sop.inria.fr/reves/research`

displaced geometry

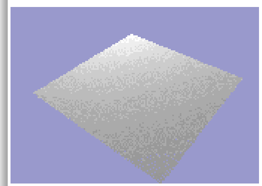


25,000 points

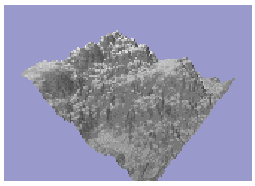


25,000 points

displaced geometry

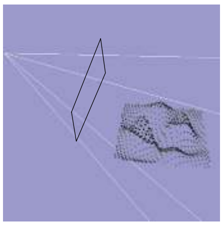


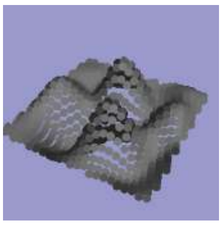
25,000 points



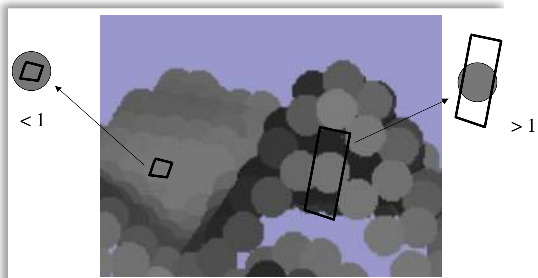
100,000 points

adaptive sampling



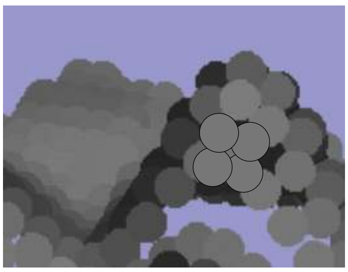


undersampling factor

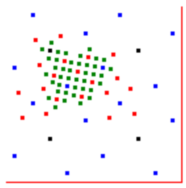


undersampling in 2D image space, not post-perspective !

undersampling factor

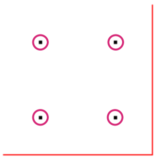


adaptive point generation

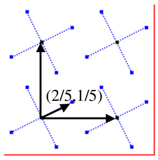


adaptive sample pattern

√5 sampling

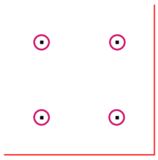


initial samples,
all undersampled

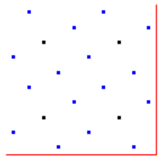


newly inserted
samples

√5 sampling

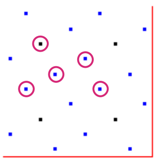


initial samples,
all undersampled

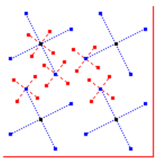


newly inserted
samples

√5 sampling

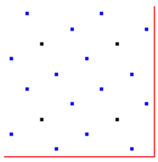


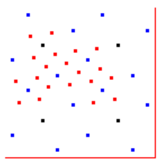
undersampled
samples



newly inserted
samples

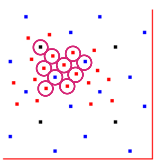
√5 sampling



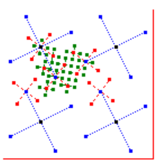


newly inserted
samples

√5 sampling

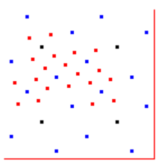


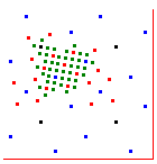
undersampled
samples



newly inserted
samples

√5 sampling

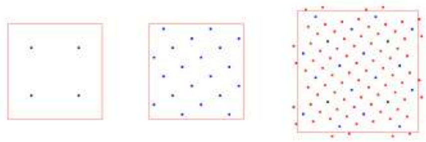




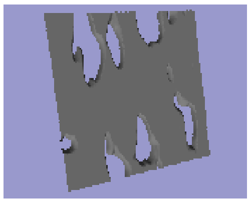
newly inserted
samples

√5 sampling

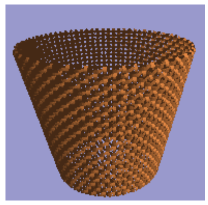
- rotated, nested grids
  - grid distance decreases by 1/sqrt(5)
  - rotation angle ≈ 27º
- special attention to boundaries



procedural modifiers



original geometry: square

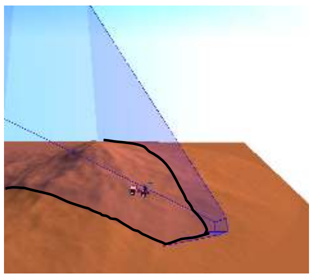


original geometry: truncated cone

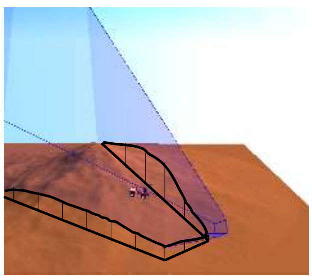
video

- video „√5 sampling"
- download at
  `http://www-sop.inria.fr/reves/research`

terrains

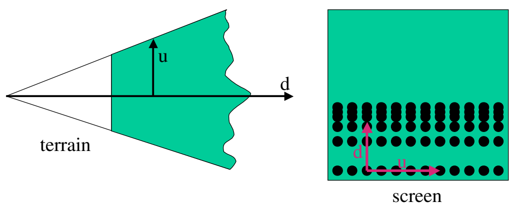


terrains



terrain parameterization

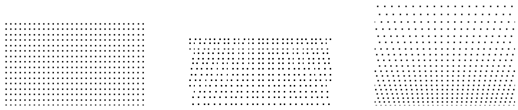
- parameterize sector by (d,u)



terrain parameterization

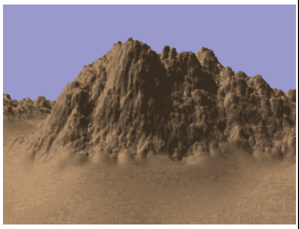
$$d(v) = \frac{1}{\frac{1}{d_{\min}} - v\left(\frac{1}{d_{\max}} - \frac{1}{d_{\min}}\right)}$$

looking straight ahead | looking up | looking down



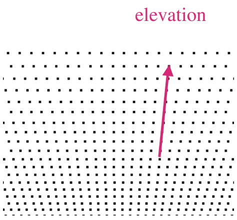
terrain algorithm

- √5 sampling scheme
- undersampling factor
  - parameterization distortions
  - perspective distortions
  - displacement

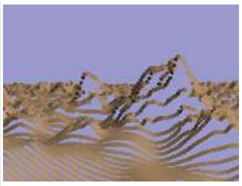


terrain occlusion culling

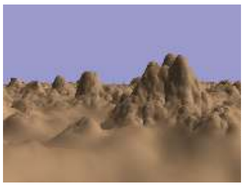
- elevation direction in image space along v
- simplifies occlusion culling



terrain occlusion culling



occlusion culling,
regular sampling



occlusion culling,
with adaptive sampling

video

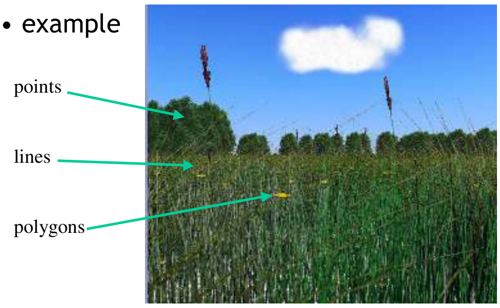
- video „terrain rendering"
- download at
`http://www-sop.inria.fr/reves/research`

eco systems

- level of detail:
  - polygonal model
  - replace polygons by points and lines
  - reduce number of points and lines

eco systems

- example

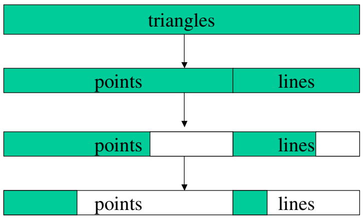


eco systems

- modeller (xfrog) delivers:
  - triangle set $\mathbf{T_p}$
  - random point set representing $\mathbf{T_p}$
  - triangle set $\mathbf{T_l}$
  - random line set **L** representing $\mathbf{T_l}$ ($|\mathbf{L}| < \mathbf{T_l}$)

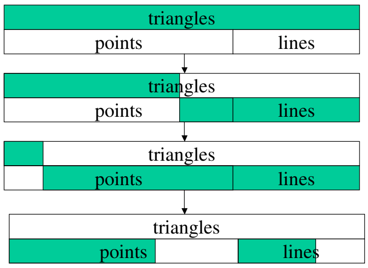
eco systems

- level-of-detail 1



eco systems

- level-of-detail 2



eco systems

- criterion for point / line number (per object)
  - user parameter:
    point size d_p / line width d_l
  - approximate screen space area of object:
    $A' = A * 0.5 / d^2$
  - #points ~ A' / d_p^2
  - #lines ~ A' / d_p

eco systems

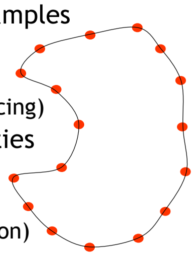
- video „eco system rendering"
- download at
  `http://www-sop.inria.fr/reves/research`

Surfaces from Point Samples

Marc Alexa
TU Darmstadt

Motivation

- Many applications need definition of surface based on point samples
  - Reduction
  - Up-sampling
  - Interrogation (e.g. ray tracing)
- Desirable surface properties
  - Manifold
  - Smooth
  - Local (efficient computation)



Overview

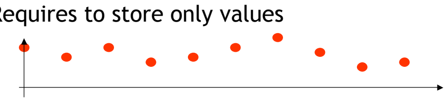
- Introduction & Basics
- Fitting Implicit Surfaces
- Projection-based Surfaces

Introduction & Basics

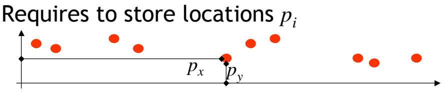
- Regular/Irregular
- Approximation/Interpolation
- Global/Local
- Standard techniques
  - LS, RBF, MLS
- Problems
  - Sharp edges, feature size/noise
- Functional/Manifold

Regular/Irregular

- Regular
  - Requires to store only values

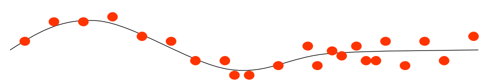


- Irregular
  - Requires to store locations p_i

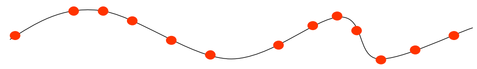


Approximation/Interpolation

- Noisy data -> Approximation

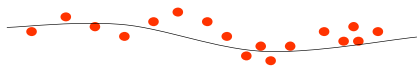


- Perfect data -> Interpolation

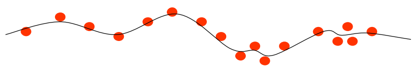


Global/Local

- Global approximation



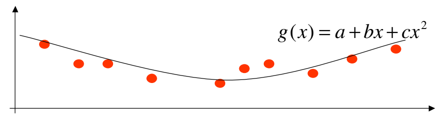
- Local approximation



- Locality comes at the expense of smoothness

Least Squares

- Fits a primitive to the data
- Minimizes squared distances between the p_i's and primitive g



$$\min_g \sum_i \left(p_{i_y} - g\left(p_{i_x}\right)\right)^2$$

Least Squares - Example

- Primitive is a polynomial

$$g(x) = \left(1, x, x^2, \ldots\right) \cdot \mathbf{c}^T$$

- $\min \sum_i \left(p_{i_y} - \left(1, p_{i_x}, p_{i_x}^2, \ldots\right)\mathbf{c}^T\right)^2 \Rightarrow$

$$0 = \sum_i 2 p_{i_x}^j \left(p_{i_y} - \left(1, p_{i_x}, p_{i_x}^2, \ldots\right)\mathbf{c}^T\right)$$

- Linear system of equations

Least Squares - Example

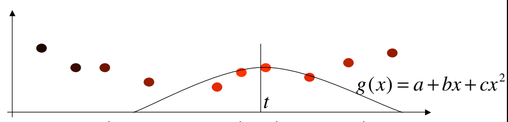
- Resulting system

$$0 = \sum_i 2 p_{i_x}^j \left(p_{i_y} - \left(1, p_{i_x}, p_{i_x}^2, \ldots\right)\mathbf{c}^T\right) \Leftrightarrow$$

$$\begin{pmatrix} 1 & x & x^2 & \cdots \\ x & x^2 & x^3 & \\ x^2 & x^3 & x^4 & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} y \\ yx \\ yx^2 \\ \vdots \end{pmatrix}$$

Moving Least Squares

- Compute a local LS approximation at t
- Weight data points based on distance to t



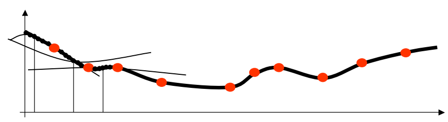
$$\min \sum_i \left(p_{i_y} - g\left(p_{i_x}\right)\right)^2 \theta\left(\left\| t - p_{i_x} \right\|\right)$$

Moving Least Squares

- The set

$$f(t) = g_t(t), g_t : \min_g \sum_i \left(p_{i_y} - g\left(p_{i_x}\right)\right)^2 \theta\left(\left\| t - p_{i_x} \right\|\right)$$

is a smooth curve, iff θ is smooth



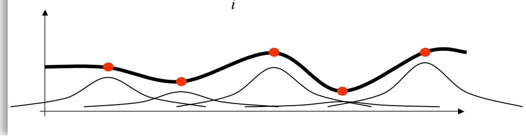
Moving Least Squares

- Typical choices for θ:
  - $\theta(d) = d^{-r}$
  - $\theta(d) = e^{-d^2/h^2}$
- Note: $\theta_i = \theta\left(\left\|t - p_{i_x}\right\|\right)$ is fixed
- For each t
  - Standard weighted LS problem
  - Linear iff corresponding LS is linear

Radial Basis Functions

- Represent interpolant as
  - Sum of radial functions r
  - Centered at the data points p_i

$$f(x) = \sum_i w_i r\left(\left\|p_i - x\right\|\right)$$



Radial Basis Functions

- Solve $p_{j_y} = \sum_i w_i r\left(\left\|p_{i_x} - p_{j_x}\right\|\right)$

  to compute weights w_i
- Linear system of equations

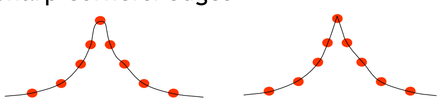
$$\begin{pmatrix} r(0) & r\left(\left\|p_{0_x} - p_{1_x}\right\|\right) & r\left(\left\|p_{0_x} - p_{2_x}\right\|\right) & \cdots \\ r\left(\left\|p_{1_x} - p_{0_x}\right\|\right) & r(0) & r\left(\left\|p_{1_x} - p_{2_x}\right\|\right) & \\ r\left(\left\|p_{2_x} - p_{0_x}\right\|\right) & r\left(\left\|p_{2_x} - p_{1_x}\right\|\right) & r(0) & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} p_{0_y} \\ p_{1_y} \\ p_{2_y} \\ \vdots \end{pmatrix}$$

Radial Basis Functions

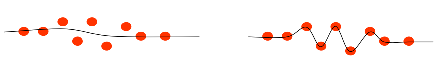
- Solvability depends on radial function
- Several choices assure solvability
  - $r(d) = d^2 \log d$ (thin plate spline)
  - $r(d) = e^{-d^2/h^2}$ (Gaussian)
    - h is a data parameter
    - h reflects the feature size or anticipated spacing among points

Typical Problems

- Sharp corners/edges

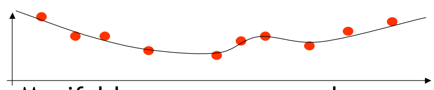


- Noise vs. feature size

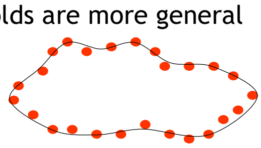


Functional/Manifold

- Standard techniques are applicable if data represents a function



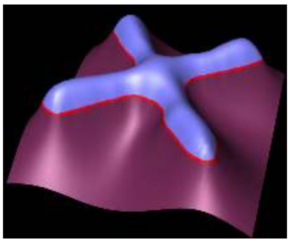
- Manifolds are more general



Implicits

- Each orientable n-manifold can be embedded in n+1 - space
- Idea: Represent n-manifold as zero-set of a scalar function in n+1 - space
  - Inside: $f(\mathbf{x}) < 0$
  - On the manifold: $f(\mathbf{x}) = 0$
  - Outside: $f(\mathbf{x}) > 0$

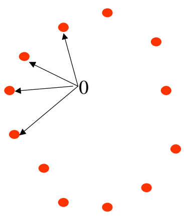
Implicits - Illustration



- Image courtesy Greg Turk

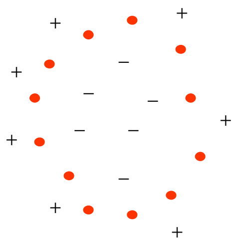
Implicits from point samples

- Function should be zero in data points
  - $f(p_i) = 0$
- Use standard approximation techniques to find f
- Trivial solution: $f = 0$
- Additional constraints are needed



Implicits from point samples

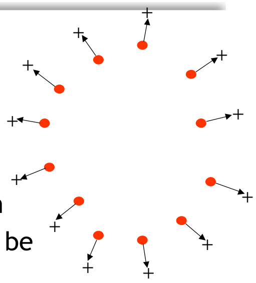
- Constraints define inside and outside
- Simple approach (Turk, O'Brien)
  - Sprinkle additional information manually
  - Make additional information soft constraints



Implicits from point samples

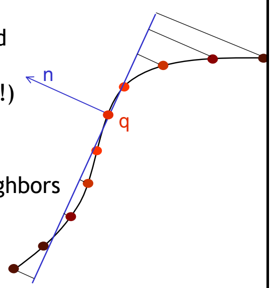
- Use normal information as constraint

  $f(p_i + n_i) = 1$
- Normals could be computed from scan
- Or, normals have to be estimated



Estimating normals

- Two problems
  - Normal direction and
  - Orientation (Implicits are signed!)
- Normal direction by fitting a tangent
  - LS fit to nearest neighbors
  - Weighted LS fit
  - MLS fit

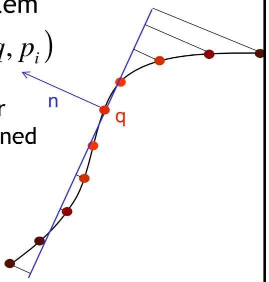


Estimating normals

- General fitting problem

$$\min_{\|n\|=1} \sum_i \langle q - p_i, n \rangle^2 \theta(q, p_i)$$

  - Problem is non-linear because n is constrained to unit sphere



Estimating normals

- The constrained minimization problem

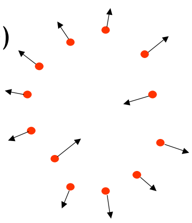
$$\min_{\|n\|=1} \sum_i \langle q - p_i, n \rangle^2 \theta_i$$

is solved by the eigenvector corresponding to the smallest eigenvalue of

$$\begin{pmatrix} \sum_i (q_x - p_{i_x})^2 \theta_i & \sum_i (q_x - p_{i_y})^2 \theta_i & \sum_i (q_x - p_{i_z})^2 \theta_i \\ \sum_i (q_y - p_{i_x})^2 \theta_i & \sum_i (q_y - p_{i_y})^2 \theta_i & \sum_i (q_y - p_{i_z})^2 \theta_i \\ \sum_i (q_z - p_{i_x})^2 \theta_i & \sum_i (q_z - p_{i_y})^2 \theta_i & \sum_i (q_z - p_{i_z})^2 \theta_i \end{pmatrix}$$

Estimating normals

- Consistent orientation
  - Problem is NP-hard
- Greedy approach (Hoppe)
  - Compute spanning tree based on graph of k-nearest neighbors
  - Orient consistently along spanning tree



Computing Implicits

- Given N points and normals p_i, n_i and constraints $f(p_i) = 0, f(p_i + n_i) = 1$
- Let $p_{i+N} = p_i + n_i$
- An RBF approximation

$$f(\mathbf{x}) = \sum_i w_i r(\|p_i - \mathbf{x}\|)$$

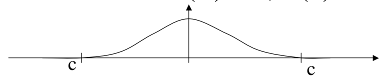
leads to $2N$ linear equations in $2N$ unknowns (a $2N \times 2N$ matrix)

Computing Implicits

- Practical problems: $N > 10000$
- Matrix solution becomes difficult
- Two solutions
  - Sparse matrices allow iterative solution
  - Smaller number of RBFs

Computing Implicits

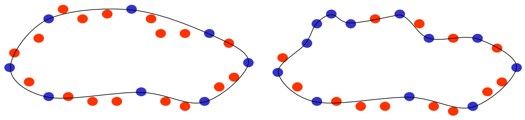
- Sparse matrices $\begin{pmatrix} r(0) & r(\|p_0 - p_1\|) & r(\|p_0 - p_2\|) & \cdots \\ r(\|p_1 - p_0\|) & r(0) & r(\|p_1 - p_2\|) & \\ r(\|p - p_0\|) & r(\|p_2 - p_1\|) & r(0) & \\ \vdots & & & \ddots \end{pmatrix}$
  - Needed: $d > c \rightarrow r(d) = 0, r'(c) = 0$



  - Compactly supported RBFs

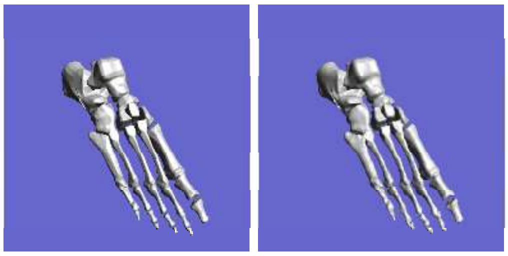
Computing Implicits

- Smaller number of RBFs
- Greedy approach (Carr et al.)
  - Start with random small subset
  - Add RBFs where approximation quality is not sufficient



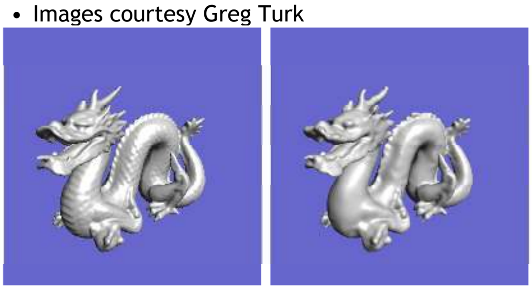
RBF Implicits - Results

- Images courtesy Greg Turk



RBF Implicits - Results

- Images courtesy Greg Turk

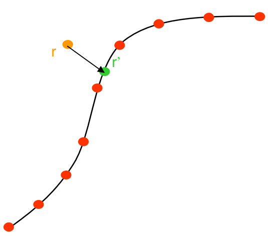


Implicits - Conclusions

- Scalar field is underconstrained
  - Constraints only define where the field is zero, not where it is non-zero
- Signed fields restrict surfaces to be unbounded
  - All implicit surfaces define solids

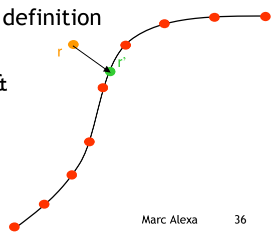
Projection

- Idea: Map space to surface
- Surface is defined as fixpoints of mapping



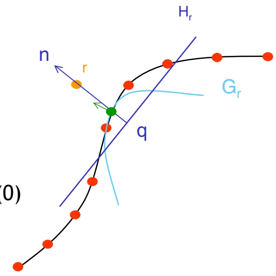
Surface definition

- Projection procedure (Levin)
  - Local polyonmial approximation
    - Inspired by differential geometry
  - "Implicit" surface definition
  - Infinitely smooth &
  - Manifold surface



Surface Definition

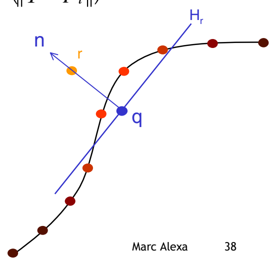
- Constructive definition
  - Input point r
  - Compute a local reference plane H_r=<q,n>
  - Compute a local polynomial over the plane G_r
  - Project point r'=G_r(0)
  - Estimate normal



Local Reference Plane

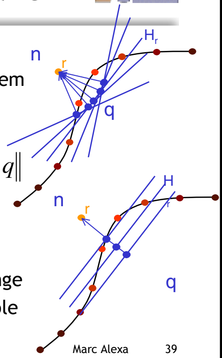
- Find plane $H_r = \langle q, n \rangle + D$
  - $\min_{q,\|n\|=1} \sum_i \langle q - p_i, n \rangle^2 \theta(\|q - p_i\|)$
  - $\theta(\mathrm{d}) = e^{d^2/h^2}$
    - h is feature size/ point spacing
  - H_r is independent of r's distance
  - Manifold property

Weight function based on distance to q, not r



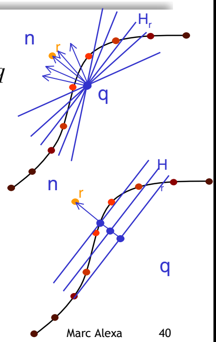
Local Reference Plane

- Computing reference plane
  - Non-linear optimization problem
- Minimize independent variables:
  - Over n for fixed distance $\|r - q\|$
  - Along n for fixed direction n
  - q changes -> the weights change
  - Only iterative solutions possible



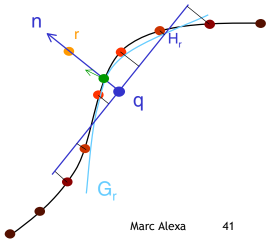
Local Reference Plane

- Practical computation
  - Minimize over n for fixed q
    - Eigenvalue problem
  - Translate q so that $r = q + \|r - q\| n$
    - Effectively changes $\|r - q\|$
  - Minimize along n for fixed direction n
    - Exploit partial derivative



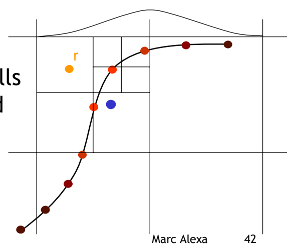
Projecting the Point

- MLS polyonomial over H_r
  - $\min_{G \in \Pi_d} \sum_i \left(\langle q - p_i, n \rangle - G\left(p_i|_{H_r}\right)\right)^2 \theta(\|q - p_i\|)$
  - LS problem
  - r'=G_r(0)
  - Estimate normal



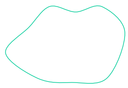
Spatial data structure

- Regular grid based on support of θ
  - Each point influences only 8 cells
- Each cell is an octree
  - Distant octree cells are approximated by one point in center of mass

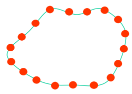


Error bounds

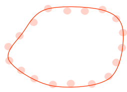
- Paradigm:
  - Given surface S



  - Point set $P = \{p_i\}$ sampled from S

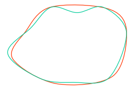


  - ($r_i \in S$) defines S_R



Error bounds

- Approximation error of S_P to S
  - MLS error approximating a function f with a polynomial g: $\|f - g\| \leq M \cdot h^{m+1}$
    - $M \in O(\|f^{(m+1)}\|)$
    - m = degree of polynomial
  - S_P is approximated by a polynomial in each point
  - $\|S - S_P\| \leq M \cdot h^{m+1}$



Error bounds

- Conclusions
  - Remark: Curvature is a useful criterion only for piecewise linear surfaces
  - Generally: Higher order derivatives are not accessible
  - Quality of representation is mainly dictated by h
  - Number of points control h
  - Increase/decrease number of points to adjust the quality of representation

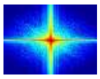
Conclusions

- Projection-based surface definition
  - Surface is smooth and manifold
  - Surface may be bounded
  - Representation error mainly depends on point density
  - Adjustable feature size h allows to smooth out noise

Some References

- Alexa, Behr, Cohen-Or, Fleishman, Levin, Silva. Point Set Surfaces. IEEE Visualization 2002, pp. 21-28, 2002
- Carr, Beatson, Cherrie, Mitchell, Fright, McCallum, Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. SIGGRAPH 2001 Proc., pp. 67-76, 2001
- Hoppe, DeRose, Duchamp, McDonald, Stuetzle. Surface Reconstruction from unorganized points. SIGGRAPH 1992 Proc., pp. 71-78, 1992
- Levin. The approximation power of moving least-squares. Math. Comp. 67(224):1517-1531, 1998
- Levin. Mesh-independent surface interpolation. Curves & Surfaces 2000
- Savchenko, Pasko, Okunev, Kunii. Function representation of solids reconstructed from scattered surface points and contours. Computer Graphics Forum, 14(4):181-188, 1995
- Turk, O'Brien. Shape transformation using variational implicit surfaces. SIGGRAPH 1999 Proc., pp. 335-342, 1999
- Turk, O'Brien. Variational implicit surfaces. Technical Report GITGVU 9915, Georgia Institute of Technology, 1999

Spectral Processing of Point-Sampled Geometry

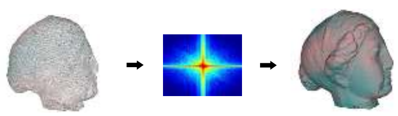


Overview

- Introduction
- Fourier transform
- Spectral processing pipeline
- Applications
  - Spectral filtering
  - Adaptive subsampling
- Summary

Introduction

- Idea: Extend the Fourier transform to manifold geometry



⇨ Spectral representation of point-based objects

⇨ Powerful methods for digital geometry processing

Introduction

- Applications:
  - Spectral filtering:
    - Noise removal
    - Microstructure analysis
    - Enhancement
  - Adaptive resampling:
    - Complexity reduction
    - Continuous LOD

Fourier Transform

- 1D example:

$$X_n = \sum_{k=1}^{N} x_k e^{-j2\pi \frac{nk}{N}}$$

output signal; input signal; spectral basis function

- Benefits:
  - Sound concept of frequency
  - Extensive theory
  - Fast algorithms

Fourier Transform

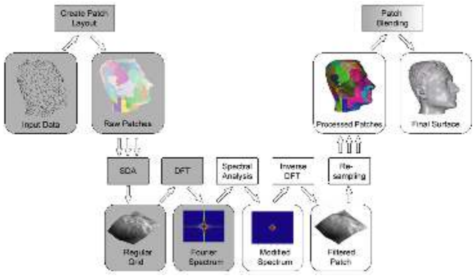
- Requirements:
  - Fourier transform defined on Euclidean domain
    ⇨ we need a global parameterization
  - Basis functions are eigenfunctions of Laplacian operator
    ⇨ requires regular sampling pattern so that basis functions can be expressed in analytical form (fast evaluation)
- Limitations:
  - Basis functions are globally defined
    ⇨ Lack of local control

Approach

- Split model into patches that:
  - are parameterized over the unit-square
    - ⇨ mapping must be continuous and should minimize distortion
  - are re-sampled onto a regular grid
    - ⇨ adjust sampling rate to minimize information loss
  - provide sufficient granularity for intended application (local analysis)

⇨ process each patch individually and blend processed patches

Spectral Pipeline



Patch Layout Creation

Clustering ⇨ Optimization

Samples ⇨ Clusters ⇨ Patches

Patch Layout Creation

- Iterative, local optimization method
- Merge patches according to quality metric:

$$\Phi = \Phi_S \cdot \Phi_{NC} \cdot \Phi_B \cdot \Phi_{Reg}$$

Φ_S ⇨ patch Size

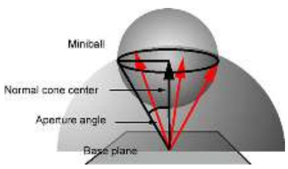
Φ_{NC} ⇨ curvature

Φ_B ⇨ patch boundary

Φ_{Reg} ⇨ spring energy regularization

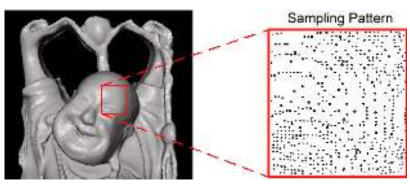
Patch Layout Creation

- Parameterize patches by orthogonal projection onto base plane
- Bound normal cone to control distortion of mapping using smallest enclosing sphere



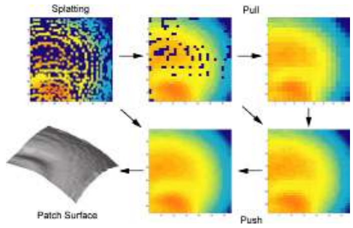
Patch Resampling

- Patches are irregularly sampled:



Patch Resampling

- Resample patch onto regular grid using hierarchical push-pull filter (scattered data approximation)



Spectral Analysis

- 2D discrete Fourier transform (DFT)
  - ⇨ Direct manipulation of spectral coefficients
- Filtering as convolution:

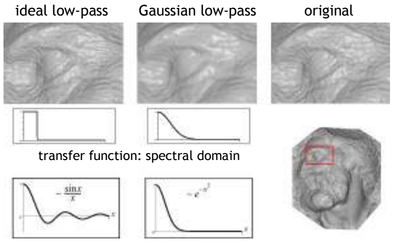
$$F(x \otimes y) = F(x) \cdot F(y)$$

  - ⇨ Convolution: $O(N^2)$ ⇨ multiplication: O(N)
- Inverse Fourier transform
  - ⇨ Filtered patch surface

Spectral Filters

- Smoothing filters

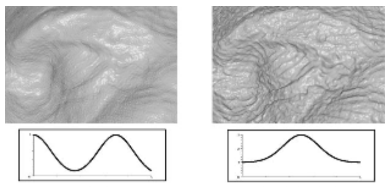
ideal low-pass | Gaussian low-pass | original



transfer function: spatial domain

Spectral Filters

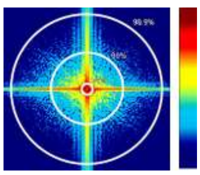
- Microstructure analysis and enhancement



band-stop | Enhancement

Spectral Resampling

- Low-pass filtering
  - ⇨ Band-limitation
- Regular Resampling
  - ⇨ Optimal sampling rate (sampling theorem)
  - ⇨ Error control (Parseval's theorem)



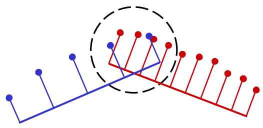
Power Spectrum

Reconstruction

- Filtering can lead to discontinuities at patch boundaries
  - ⇨ Create patch overlap, blend adjacent patches

Sampling rates → Point positions → Normals →

region of overlap

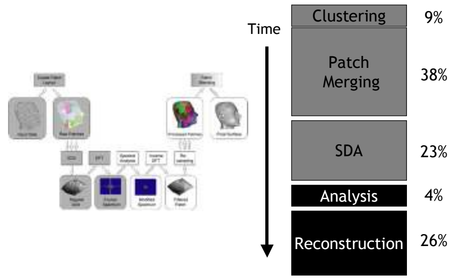


Reconstruction

- Blending the sampling rate

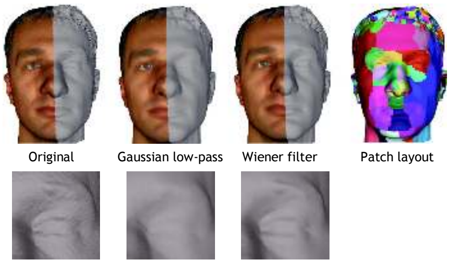
blended sampling rate in region of patch overlap → discretized sampling rate on regular grid → pre-computed sampling patterns

Timings



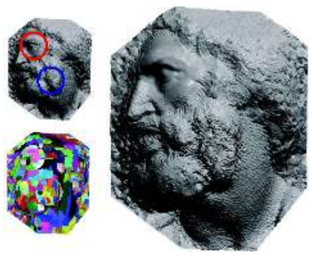
Applications

- Surface Restoration



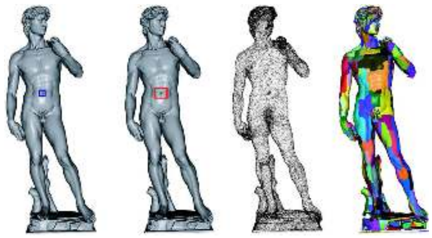
Applications

- Interactive filtering



Applications

- Adaptive Subsampling



4,128,614 pts. = 100% 287,163 pts. = 6.9%

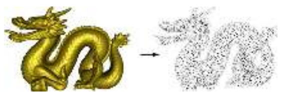
Summary

- Versatile spectral decomposition of point-based models
- Effective filtering
- Adaptive resampling
- Efficient processing of large point-sampled models

Reference

- Pauly, Gross: *Spectral Processing of Point-sampled Geometry*, SIGGRAPH 2001

Efficient Simplification of Point-sampled Surfaces



Overview

- Introduction
- Local surface analysis
- Simplification methods
- Error measurement
- Comparison

Introduction

- Point-based models are often sampled very densely
- Many applications require coarser approximations, e.g. for efficient
    - Storage
    - Transmission
    - Processing
    - Rendering

⇨ we need simplification methods for reducing the complexity of point-based surfaces

Introduction

- We transfer different simplification methods from triangle meshes to point clouds:
    - Incremental clustering
    - Hierarchical clustering
    - Iterative simplification
    - Particle simulation
- Depending on the intended use, each method has its pros and cons (see comparison)

Local Surface Analysis

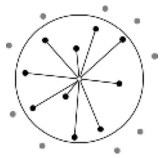
- Cloud of point samples describes underlying (manifold) surface
- We need:
    - mechanisms for locally approximating the surface ⇨ MLS approach
    - fast estimation of tangent plane and curvature ⇨ principal component analysis of local neighborhood

Neighborhood

- No explicit connectivity between samples (as with triangle meshes)
- Replace geodesic proximity with spatial proximity (requires sufficiently high sampling density!)
- Compute neighborhood according to Euclidean distance

Neighborhood

- k-nearest neighbors



- can be quickly computed using spatial data-structures (e.g. kd-tree, octree, bsp-tree)
- requires isotropic point distribution

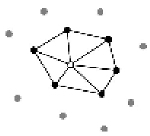
Neighborhood

- Improvement: angle criterion (Linsen)

- project points onto tangent plane
- sort neighbors according to angle
- include more points if angle between subsequent points is above some threshold

Neighborhood

- Local Delaunay triangulation (Floater)



- project points into tangent plane
- compute local Voronoi diagram

Covariance Analysis

- Covariance matrix of local neighborhood N:

$$\mathbf{C} = \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_n} - \bar{\mathbf{p}} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_n} - \bar{\mathbf{p}} \end{bmatrix}, \qquad i_j \in N$$

- with centroid $\bar{\mathbf{p}} = \frac{1}{|N|} \sum_{i \in N} \mathbf{p}_i$

Covariance Analysis

- Consider the eigenproblem:

$$\mathbf{C} \cdot \mathbf{v}_l = \lambda_l \cdot \mathbf{v}_l, \qquad l \in \{0,1,2\}$$

- C is a 3x3, positive semi-definite matrix
  - ⇨ All eigenvalues are real-valued
  - ⇨ The eigenvector with smallest eigenvalue defines the least-squares plane through the points in the neighborhood, i.e. approximates the surface normal

Covariance Analysis

- The total variation is given as:

$$\sum_{i \in N} |\mathbf{p}_i - \bar{\mathbf{p}}|^2 = \lambda_0 + \lambda_1 + \lambda_2$$

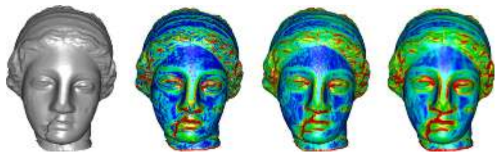
- We define surface variation as:

$$\sigma_n(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \qquad \lambda_0 \le \lambda_1 \le \lambda_2$$

  - measures the fraction of variation along the surface normal, i.e. quantifies how strong the surface deviates from the tangent plane ⇨ estimate for curvature

Covariance Analysis

- Comparison with curvature:



original | mean curvature | variation n=20 | variation n=50

Surface Simplification

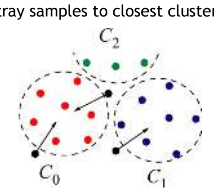
- Incremental clustering
- Hierarchical clustering
- Iterative simplification
- Particle simulation

Incremental Clustering

- Clustering by region-growing:
  - Start with random seed point
  - Successively add nearest points to cluster until cluster reaches maximum size
  - Choose new seed from remaining points
- Growth of clusters can also be bounded by surface variation
  - ⇨ Curvature adaptive clustering

Incremental Clustering

- Incremental growth leads to internal fragmentation
  - ⇨ assign stray samples to closest cluster



- Note: this can increase maximum size and variation bounds!

Incremental Clustering

- Replace each cluster by its centroid

original model with color-coded clusters (34,384 points)

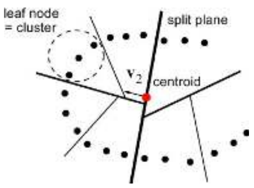
simplified model (1,000 points)

Hierarchical Clustering

- Top-down approach using binary space partition:
- Split the point cloud if:
  - Size is larger than user-specified maximum or
  - Surface variation is above maximum threshold
- Split plane defined by centroid and axis of greatest variation (= eigenvector of covariance matrix with largest associated eigenvector)
- Leaf nodes of the tree correspond to clusters

Hierarchical Clustering

- 2D example



Hierarchical Clustering

- Adaptive clustering

original model with
color-coded clusters
(34,384 points)

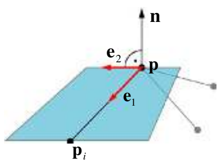
simplified model
(1,000 points)

Iterative Simplification

- Iteratively contracts point pairs
  - ⇨ Each contraction reduces the number of points by one
- Contractions are arranged in priority queue according to quadric error metric (Garland and Heckbert)
- Quadric measures cost of contraction and determines optimal position for contracted sample
- Equivalent to QSlim except for definition of approximating planes

Iterative Simplification

- Quadric measures the squared distance to a set of planes defined over *edges* of neighborhood
  - plane spanned by vectors $\mathbf{e}_1 = \mathbf{p}_i - \mathbf{p}$ and $\mathbf{e}_2 = \mathbf{e}_1 \times \mathbf{n}$



Iterative Simplification

original model
(187,664 points)

simplified model
(1,000 points)

remaining point pair
contraction candidates

Particle Simulation

- Resample surface by distributing particles on the surface
- Particles move on surface according to inter-particle repelling forces
- Particle relaxation terminates when equilibrium is reached (requires damping)
- Can also be used for up-sampling!

Particle Simulation

- Initialization
  - randomly spread particles
- Repulsion
  - linear repulsion force $F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$

  ⇨ only need to consider neighborhood of radius r
- Projection
  - keep particles on surface by projecting onto tangent plane of closest point
  - apply full MLS projection at end of simulation

Particle Simulation

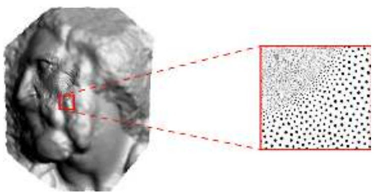
- Adaptive simulation
  - Adjust repulsion radius according to surface variation
    ⇨ more samples in regions of high variation

original model
(75,781 points)

simplified model
(6,000 points)

Particle Simulation

- User-controlled simulation
  - Adjust repulsion radius according to user input



Measuring Error

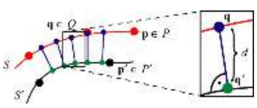
- Measure the distance between two point-sampled surfaces using a sampling approach
- Maximum error: $\Delta_{\max}(S, S') = \max_{\mathbf{q} \in Q} d(\mathbf{q}, S')$

  ⇨ Two-sided Hausdorff distance
- Mean error: $\Delta_{\text{avg}}(S, S') = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} d(\mathbf{q}, S')$

  ⇨ Area-weighted integral of point-to-surface distances
- Q is an up-sampled version of the point cloud that describes the surface S

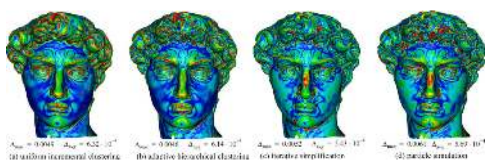
Measuring Error

- $d(\mathbf{q}, S')$ measures the distance of point $\mathbf{q}$ to surface S' using the MLS projection operator with linear basis functions



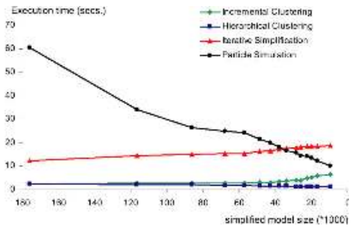
Comparison

- Error estimate for Michelangelo's David simplified from 2,000,000 points to 5,000 points



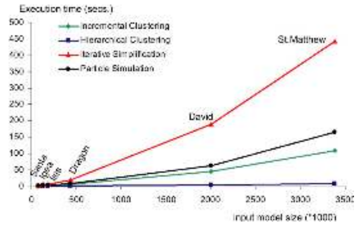
Comparison

- Execution time as a function of target model size (input: dragon, 535,545 points)



Comparison

- Execution time as a function of input model size (reduction to 1%)

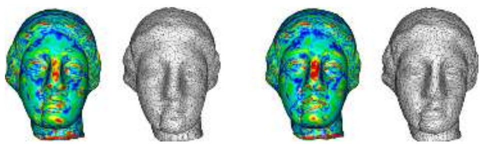


Comparison

- Summary

	Efficiency	Surface Error	Control	Implementation
Incremental Clustering	+	-	-	+
Hierarchical Clustering	+	-	-	+
Iterative Simplification	-	+	o	o
Particle Simulation	o	+	+	-

Point-based vs. Mesh Simplification



point-based simplification with subsequent mesh reconstruction

mesh reconstruction with subsequent mesh simplification (QSlim)

⇨ point-based simplification saves an expensive surface reconstruction on the dense point cloud!

References

- Pauly, Gross: *Efficient Simplification of Point-sampled Surfaces*, IEEE Visualization 2002
- Shaffer, Garland: *Efficient Adaptive Simplification of Massive Meshes*, IEEE Visualization 2001
- Garland, Heckbert: *Surface Simplification using Quadric Error Metrics*, SIGGRAPH 1997
- Turk: *Re-Tiling Polygonal Surfaces*, SIGGRAPH 1992
- Alexa et al. *Point Set Surfaces*, IEEE Visualization 2001

pointshop

An Interactive System for Point-based Surface Editing

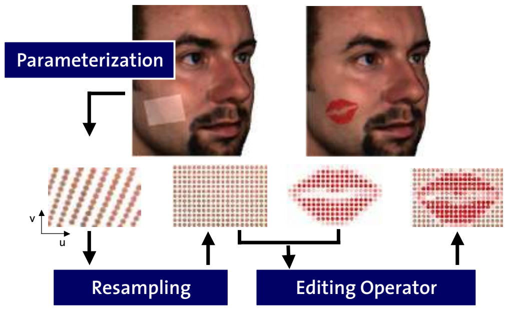
Overview

- Introduction
- Pointshop3D System Components
  - Point Cloud Parameterization
  - Resampling Scheme
  - Editing Operators
- Summary

PointShop3D

- Interactive system for point-based surface editing
- Generalizes 2D photo editing concepts and functionality to 3D point-sampled surfaces
- Uses 3D surface pixels (*surfels*) as versatile display and modeling primitive

Concept



Key Components

- Point cloud parameterization Φ
  - brings surface and brush into common reference frame
- Dynamic resampling Ψ
  - creates one-to-one correspondence of surface and brush samples
- Editing operator Ω
  - combines surface and brush samples

$$\underbrace{S'}_{\text{modified surface}} = \Omega(\Psi(\Phi(\underbrace{S}_{\text{original surface}})), \Psi(\underbrace{B}_{\text{brush}}))$$

Parameterization

- Constrained minimum distortion parameterization of point clouds

$$\mathbf{u} \in [0,1]^2 \Rightarrow X(\mathbf{u}) = \begin{bmatrix} x(\mathbf{u}) \\ y(\mathbf{u}) \\ z(\mathbf{u}) \end{bmatrix} = \mathbf{x} \in P \subset R^3$$

Parameterization

contraints = matching of feature points

minimum distortion = maximum smoothness

Parameterization

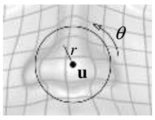
- Find mapping X that minimizes objective function:

$$C(X) = \sum_{j \in M} \underbrace{(X(\overset{\text{brush points}}{\mathbf{p}_j}) - \overset{\text{surface points}}{\mathbf{x}_j})^2}_{\text{fitting constraints}} + \varepsilon \underbrace{\int_P \gamma(\mathbf{u})\, d\mathbf{u}}_{\text{distortion}}$$

Parameterization

- Measuring distortion

$$\gamma(\mathbf{u}) = \int_\theta \left(\frac{\partial^2}{\partial r^2} X_{\mathbf{u}}(\theta, r) \right)^2 d\theta$$



- Integrates squared curvature using local polar re-parameterization

$$X_{\mathbf{u}}(\theta, r) = X\left(\mathbf{u} + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \right)$$

Parameterization

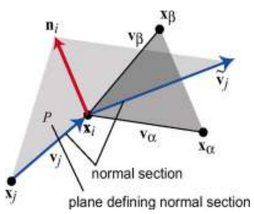
- Discrete formulation:

$$\tilde{C}(U) = \sum_{j \in M} (\mathbf{p}_j - \mathbf{u}_j)^2 + \varepsilon \sum_{i=1}^{n} \sum_{j \in N_i} \left(\frac{\partial U(\mathbf{x}_i)}{\partial \mathbf{v}_j} - \frac{\partial U(\mathbf{x}_i)}{\partial \tilde{\mathbf{v}}_j} \right)^2$$

- Approximation: mapping is piecewise linear

Parameterization

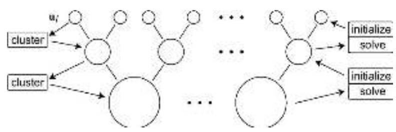
- Directional derivatives as extension of divided differences based on k-nearest neighbors



Parameterization

- Multigrid solver for efficient computation of resulting sparse linear least squares problem

$$\tilde{C}(U) = \sum_j \left(\mathbf{b}_j - \sum_{i=1}^{n} a_{j,i} \mathbf{u}_i \right)^2 = \|\mathbf{b} - A\mathbf{u}\|^2$$



Reconstruction

- Parameterized scattered data approximation

fitting functions

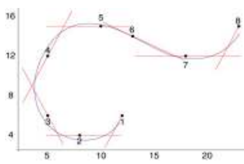
weight functions

$$X(\mathbf{u}) = \frac{\sum_i \Phi_i(\mathbf{u}) r_i(\mathbf{u})}{\sum_i r_i(\mathbf{u})}$$

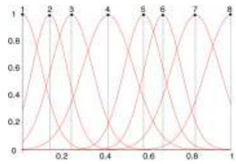
normalization factor

- Fitting functions
  - Compute local fitting functions using local parameterizations
  - Map to global parameterization using global parameter coordinates of neighboring points

Reconstruction



reconstruction with linear fitting functions



weight functions in parameter space

Reconstruction

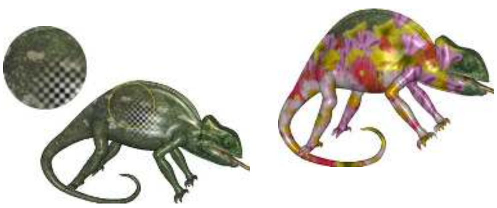
- Reconstruction with linear fitting functions is equivalent to surface splatting!
  - ⇨ we can use the surface splatting renderer to reconstruct our surface function (see chapter on rendering)
  - This provides:
    - Fast evaluation
    - Anti-aliasing (Band-limit the weight functions before sampling using Gaussian low-pass filter)
  - Distortions of splats due to parameterization can be computed efficiently using local affine mappings

Sampling

- Three sampling strategies:
  - Resample the brush, i.e., sample at the original surface points
  - Resample the surface, i.e., sample at the brush points
  - Adaptive resampling, i.e., sample at surface or brush points depending on the respective sampling density

Editing Operators

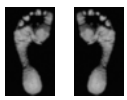
- Painting
  - Texture, material properties, transparency



Editing Operators

- Sculpting
  - Carving, normal displacement

texture map



displacement maps

carved and texture mapped point-sampled surface

Editing Operators

- Filtering
  - Scalar attributes, geometry

Summary

- Pointshop3D provides sophisticated editing operations on point-sampled surfaces
  ⇨ points are a versatile and powerful modeling primitive
- Limitation: only works on "clean" models
  - sufficiently high sampling density
  - no outliers
  - little noise
  ⇨ requires model cleaning (integrated or as pre-process)

Reference

- Zwicker, Pauly, Knoll, Gross: *Pointshop3D: An interactive system for Point-based Surface Editing*, SIGGRAPH 2002

pointshop

- check out:

www.pointshop3D.com