Geometric representations of 3D scenes

Géraldine Morin
IRIT-VORTEX / ENSEEIHT

University of Toulouse
3D models for dedicated contexts

Catia CAD system, Dassault
CMU dome
Aachen Univ., immersive room
Geneva’s hospital, assisted surgery
Geri’s game, Pixar
Michaelangelo project, M. Levoy

MMSYS 2017
3D models for everyone

Reconstructing your environment

Kinect-based device for visually impaired [Choppin, 2012]

3D Visualization

Pompei en 3D, Uni. Lund

RGB-D camera [ScenesNN, SHREC 17]

Virtual try on:

La Redoute 3D website

Try glasses online, FittingBox

3D Glasses, Microsoft Hololens

3D factory: 3D printer, and printed letters

MMSYS 2017
Which 3D models?

What you should get out of this talk

Which model can / should you use

• Different models (advantages / drawbacks)
  • Ex: 3D scanned object should to be tracked in a video

Can you adapt this model to your needs / application

• Ex: Transmit (compress): adaptative resolution

...or adapt existing tools for this model
How to choose your 3D representation

IN: 3D data

Geometric 3D model

OUT: MM application

Other (non geometric) needs on the 3D model:
- Attributes
- Texture / appearance
- Structure
- Semantic
3D models on the shelves

Discrete vs. Continuous
With or without topology
Volume vs. Surface
Static vs. Dynamic
3D models on the shelves

Example 1: modeling fire

- Discrete vs. Continuous
- With or without topology
- Volume vs. Surface
- Static vs. Dynamic

Particle systems
3D models on the shelves

Example 2: modeling your brand new black motocycle

- Discrete vs. Continuous
- With or without topology
- Volume vs. Surface
- Static vs. Dynamic

Parametric model (NURBS based, standard in CAD)

MMSYS 2017
Overview of existing models

- **Discrete, no topology**
  - Point based models
  - Particle systems

- **Continuous, with topology**
  - Meshes
    - 2D
    - 3D

- **Parametric models**
- **Implicit models**
- **NURBS based**

1. Which 3D model?
Particle systems

Discrete, Volume, No topology, Dynamic

- Particularly adapted for modeling fluids
  - Image from *HAL 9000 Animations* with Blender Fluids

Relatively easy to implement

Also works for deformable objects
Point based models

Discrete, No topology, Surface, Static

Images from [Real time point set refinement, Guennebaud et al. Eurographics Workshop, 2004 ]
Point based models

Raw data out of scanners

Density adapted to curvature

MMSYS 2017

I. Which 3D model?
Point based models

Or even use ellipses

- Very compact
- Rendering is done via splatting: mix of neighboring splats (on the GPU)

Point based models

Access to neighbors

no topology => complexity $O(#points)$

HENCE the need to ease the access to the neighbors

• Use a space partitioning structure
  • Octree
  • Kd-tree
  • BSP (binary space partitioning) trees
Space partitionning tree

**Regular octree**
Access to a cell $O(1)$
Complexity of neighbors request $O(\#\text{neighbors})$

*Drawback: may split empty cells (wasting storage space)*
Space partitioning tree

Adaptative octree
Access to a cell – tree traversal
• when balanced $O(\log(\#\text{points}))$
• If unbalanced $O(\#\text{points})$

Advantage: split cells when needed
Space partitionning tree

**Adaptative octree**
An example
Space partitionning tree

**Kd tree**
Access to a cell -- tree traversal
Insures a balanced tree so $O(\log(\#\text{points}))$

1. Image from [CGAL reference manual]

MMSYS 2017
Space partitionning tree

Kd tree
An example
3D Meshes

Discrete, with topology, volume, static or dynamic

Image from POINTWISE® CFD solver (Computationnal Fluid Dynamics)
2D Meshes

Discrete, with topology, surface, static

The most popular representation

Most often faces are triangles, but maybe quads
Meshes coding

Coding geometry

Coordinates of the vertices

\[(v_0) \ x_0 \ y_0 \ z_0\]
\[(v_1) \ x_1 \ y_1 \ z_1\]
\[(v_2) \ x_2 \ y_2 \ z_2\]
...

Coding topology

Faces: vertex indices

\[(f_0) \ i_0 \ i_1 \ i_2\]
\[(f_1) \ i_1 \ i_2 \ i_3\]
\[(f_2) \ i_5 \ i_6 \ i_7\]
...

Half edge structure to manipulate meshes
Continuous models

Fonctional analysis:

• Study continuity

Function
Analytic continuity

parameterization

Geometric continuity
of the model

I. Which 3D model?
Continuous models: in 2D

Parametric

In the affine plane

The function

\[ t \mapsto f(t) = (x(t), y(t)) \]

defines a curve

\[ \{ f(t) \mid t \in \text{domain} \} \]

Ex: \((\cos t, \sin t)\) on \([0,2\pi]\)

Implicit

A potential function

\[ (x,y) \mapsto p(x,y) \]

defines a curve

\[ \{(x,y) \mid p(x,y) = 0\} \]

Ex: \(p(x,y) = x^2 + y^2 - 4\)
Continuous models: in 2D

**Parametric**

Advantages:

Easy to sample points on the curve

- Sample the domain, and evaluate

Ex: \( t_i \) on \([0, 2\pi]\) give points \( f(t_i)\) on the circle

**Implicit**

Advantages:

Given a point \((x,y)\) easy to check if it is

- On the curve \( p(x,y) = 0 \)
- Inside the curve \( p(x,y) < 0 \)
- Outside the curve \( p(x,y) > 0 \)
Continuous models: in 2D

**Parametric**

Drawbacks:

Hard to know if a given point \( A (x_A,y_A) \) lies on the curve

Solve \( f(t) = (x_A,y_A) \)

**Implicit**

Drawbacks:

Hard to sample points on the curve

Solve \( p(x,y) = 0 \)

MMSYS 2017
Continuous models: in 3D

Parametric
Advantages
Sampling points: discretizing the surface

Implicit
Advantages
Easy to compute intersection
- Collision detection
- Ray tracing

Guarantee on the smoothness of the surface
Continuous models: in 3D

**Parametric**

- Advantages
- Sampling

**Implicit**

- Advantages
- Easy to compute intersection
  - Collision detection
  - Ray tracing

Images from [Implicit modeling revisited, Bernhardt et al., Comp. Graphics Forum, 2010]
Continuous models: in 3D

Parametric

Bézier curves

Spline curves

Tensor product splines

MMSYS 2017
Continuous models: in 3D

Parametric

Bézier curves

Spline curves

Tensor product splines

**NURBS** parametric models are **standard** in CAD-CAM softwares

- intuitive control
- few parameters
- controlled continuity
Subdivision surfaces

Link between parametric and discrete

openSUBDIV release from Pixar in 2012
Subdivision surfaces

Are used for 3D compression

Decimate…

1. Which 3D model?
Subdivision surfaces

Are used for 3D compression
And re-encode details

[MeanSquare error approx. For wavelet based semi regular mesh compression Payan, Antonini, Trans. on Vis. And CG, 2005]
Overview of existing models

- **discrete**
  - no topology
  - Point based models
  - Particle systems

- **continuous**
  - Implicit models
  - Parametric models
  - NURBS based
  - Generalized Cylinders
  - Smoothness

- with topology
  - Meshes
  - 2D, 3D

I. Which 3D model?
5 MM applications using 3D models

1. Streaming 3D meshes
2. 3D virtual plants for NVEs (networked virtual env.)
3. Track a 3D model in a video, and augment
4. Structure a 3D model for edition
5. Teaser: Do we really need 3D?
1. Streaming 3D meshes

Progressive resolution in images
1. Streaming 3D meshes

Multi-resolution in 3D
I. Streaming 3D meshes

Multi-resolution within an object
1. Streaming 3D meshes

Different resolution in 3D

[MeanSquare error approx. For wavelet based semi regular mesh compression Payan, Antonini, Trans. on Vis. And CG, 2005]
I. Streaming 3D meshes

Progressive meshes

Dependency model (DAG)

• Meshes [Hoppe 96]
I. Streaming 3D meshes

Progressive models (DAG)

- Vertex splits

Specific dependencies

- Local support
- One connected component

[CW, WTO, SM, RG, GM. An analytical model for progressive mesh streaming. ACM MM, 2007]
1. Streaming 3D meshes

3D Streaming: consider network losses

- Propose a greedy strategy to improve the expected quality at the client

[FIFO, Greedy]

1. Streaming 3D meshes

Progressive meshes: navigation

1. Streaming friendly camera path
2. Preview streaming algorithm
3. Adaptation to bandwidth variation

Server: streaming

Client: rendering
User: previewing

transmit the visible parts of a mesh along a virtual camera path connecting the keyviews

[SZ, WTO, AC, GM, VC, 3D Mesh Preview Streaming. MMSys, 2013]
[SZ, WTO, AC, GM, VC, Bandwidth Adaptation for 3D Mesh Preview Streaming. ACM TOMCCAP (accepted)]
I. Streaming 3D meshes

Conclusion

Choose a model able to support

• multiresolution
• Progressivity

Compress through relative coding

• Handle dependencies
5 MM applications using 3D models

1. Streaming 3D meshes
2. 3D virtual plants for NVEs (networked virtual env.)
3. Track a 3D model in a video, and augment
4. Structure a 3D model for edition
5. Teaser: Do we really need 3D?
2. plants in NVEs

Nut tree: original 279K triangles

Meshes are not adapted

Mesh decimation leads to topology incoherences
2. plants in NVEs

A compact and progressive plant model
2. plants in NVEs

A compact and progressive plant model
2. plants in NVEs

A compact plant model: lossless compression rate

<table>
<thead>
<tr>
<th>Tree name</th>
<th>Size (Bytes) and compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic</td>
</tr>
<tr>
<td>Walnut</td>
<td>143608</td>
</tr>
<tr>
<td>Apple tree</td>
<td>28404</td>
</tr>
<tr>
<td>L-System (fir)</td>
<td>2666968</td>
</tr>
</tbody>
</table>

2. plants in NVEs

Progressive models (DAG)

- Plant models

5%  15%  40%  85%

2. plants in NVEs

A skeleton based model (parametric curve + implicit radius)

able to support

• A compact representation

• A progressive representation
5 MM applications using 3D models

1. Streaming 3D meshes
2. 3D virtual plants for NVEs (networked virtual env.)
3. Track a 3D model in a video, and augment
4. Structure a 3D model for edition
5. Teaser: Do we really need 3D?
3. 3D visual tracking

We have a 3D model generated by

- A scan
- Multiview reconstruction

Output: set of points

A video of this model

Goal: estimate the pose of the model in the video
3. 3D visual tracking

Recovering 3D pose:

• Known 3D object (point based model)

3D model and a rendering (using splatting)
3. 3D visual tracking

Method:

• Keyframes
• Iterative
• Splatting motion vectors

3. 3D visual tracking

Recovering 3D pose:

• Well known 3D object (point based model)
• Video of this object
3. 3D visual tracking

Represent an objet with ‘raw’ point cloud

- Adapted to vision point matching
- Robust

Possible extensions

- No topology could help considering deformable objects
5 MM applications using 3D models

1. Streaming 3D meshes
2. 3D virtual plants for NVEs (networked virtual env.)
3. Track a 3D model in a video, and augment
4. Structure a 3D model for edition
5. Teaser: Do we really need 3D?
4. Consistent 3D editing

work on similarities on standard CAD models

NURBS based representations
4. Consistent 3D editing

NURBS based surfaces

- are standards in CAD
- offer a compact representation: control points

Tensor product NURBS surface of bi-degree \((k, l)\) is defined by

\[
S(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} P_{ij} N_i^k(u) N_j^l(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} N_i^k(u) N_j^l(v)}
\]

- \(P_{ij}\): control points with associated weights \(w_{ij}\).
- \(N_i^k, N_j^l\): B-spline basis functions.
4. Consistent 3D editing

**Goal:** find part of objects similar up to an isometry

Previous works

- Global features, or graph based [Hidalga01, Ma10]
- Symmetry descriptors [Kazdan04, Podolak06]
- Vote transformation space (meshes) [Mitra06, Lipman09]
- NURBS based models with primitives [Li11, Cuillère11]
4. Consistent 3D editing

Point pairing

Isometry classification

Validation

Recover spatial coherence
- Within a NURBS patch
- Within the B-rep

21st WSCG conference, 2013 ]

MMSYS 2017
4. Consistent 3D editing

Face pairing

Isometry from corners to corners?

+ validation on control polygon

Isometry classification

Validation

Extend from pairs of faces to single faces

2. Model adaptation
4. Consistent 3D editing

- Another application: 3D Indexation
  1. Identifying the dominant symmetry
  2. Use Fourier descriptors on canonical boundary projections
  3. Compute similarity scores

4. Consistent 3D editing

Proposition

• Consistent editing
• Indexing

Other applications

• Compress
• Add semantic
• Structure
5 MM applications using 3D models

1. Streaming 3D meshes
2. 3D virtual plants for NVEs (networked virtual env.)
3. Track a 3D model in a video, and augment
4. Structure a 3D model for edition

5. Teaser: Do we really need 3D?
5. do we really need 3D?

Fixing the viewpoint

- Image based OK!
5. do we really need 3D?

Mesh + textures

Alternative: pre-render and send video

3D Data server

Server Rendering

3D content

video

viewpoint

interactions

Not scalable!
Comverse+intel’s server: up to 14 users
Vollee’s up to 40 hansets
5. do we really need 3D?

Heterogeneous resources

Benefits from others' rendering:
- Lighter 3D content
- More efficient rendering

Ex.: Second life: 26,000 grid regions, 60,000 concurrent users
5. do we really need 3D?

Image based representations

[Marciel, Shirley ‘95] planar textured based billboards

[Decoret et al ‘03] Billboard clouds

TDM

Textured Depth Mesh (TDM) [Sillion et al ‘97]
• incremental creation of TDM [Wilson, Manocha ‘03]
• Real-time creation of TDM [Ghiletiuc et al. ‘13]

[Chang et al ‘99] LDI (layered depth images) tree
5. do we really need 3D?

Rendering from depth images:

3D warping

Known:
Viewpoint 1 ($C_1$)
Pixel $x_1$ and depth $d_1$

Find: $X$ in 3D

Render from $C_2$


5. do we really need 3D?

Create a sprite tree

Use image + depth impostors

• Generated by other clients (recycle!)

Store in an octree

• Insert sprite
  From relevant locations
  Sprite selection

Render from sprite tree
5. do we really need 3D?

Create a sprite tree
5. do we really need 3D?

Create a sprite tree
Sprites and sprite tree

Create a sprite tree

A sprite
Comparison to a LDI tree node
Sprites and sprite tree

A sprite (same viewpoint, same depth)

- is stored in the tree according to its location
- is associated to a single viewpoint
- is inserted at a level of the octree corresponding to its depth (resolution)

A sprite tree node contains only sprites whose

- Depth is smaller than the maximum depth associated with the corresponding level
- Location is in its bounding box
5. do we really need 3D?

Create a sprite tree
5. do we really need 3D?

Render from a sprite tree
5. do we really need 3D?

Rendered from *sprites*

**Render from a sprite tree**

Frame rendering 30ms faster
From sprite tree than from 3D
Geometry (on the GPU)

[Zhu, Morin, Charvillat, Ooi. *Sprite tree: an efficient image-based representation for NVEs. The Visual Computer, 2017*]
5. do we really need 3D?

Rendered from sprites

Ground truth

(a) (b)
5. do we really need 3D?

How much 3D do you need

Interactions / Prediction

Coding, Storing

Transmission

Viewpoint

3D model

RGB-D
- Depth
- Sprites

Textured Meshes

Geometry Vs. Texture

Usage Contexte

- Evaluate model quality
- Multi-resolution
- Multi-models

Heterogeneous contexte

Navigation

MMSYS 2017
Conclusions and more...

You can **choose your 3D model** on the shelf

You may need to **adapt** it

New trends in 3D models

- More 3D points (new sensors LiDAR, ToF, RGB-D)
- New reconstruction primitives (segments, planes)
- More topology (simplices)
- Volume: good for printing

MMSYS 2017
Thank you!
谢谢