

CMPUT 428: 3D Modeling

Roderick Lan

Contents

1	Lecture - Mar 12	2
1.1	Incremental Free Space Carving	2
1.2	3D modeling system	2
1.3	3 Tier Model	2
1.4	Multiscale Model	2
1.5	Capgui	3
1.5.1	SFS methods	3
1.5.2	Common Text. Coord. Mappings	3
1.5.3	Advanced Texture Splitting and Mapping	4
1.6	Performance	4
1.7	Meso Struct	4
1.7.1	Computing Meso	4
1.7.2	Rendering Meso	4
1.8	Micro Struct	5
1.8.1	Dynamic Textures	5
1.9	Spatial Basis Intro	5
1.10	Linear basis for spatio-temporal variation	5
1.11	Geometric spatial temporal variability	6
1.12	Variability due to a planar projective warp (homography)	6
1.13	Out of Plane Variability	6
1.14	Photometric Variation	6
1.15	Composite Variability	6
1.16	How to Compute	6
1.17	Dyntex	6

1 Lecture - Mar 12

Lec09DynTextimpColl12

Structure from Silhouette

Get cone ray from silhouette

1.1 Incremental Free Space Carving

Triangulate sparse point cloud: remove tetrahedrons/triangles + remake w/ points

1.2 3D modeling system

online, incremental handling of new info events

works with sparse point clouds (good for vision/feature based methods)

models coarse

1.3 3 Tier Model

Macro, Meso, Micro model

refine geometry w/ coarse model as prior Multi Tiered Models:

- Commonly:
 - 2 Tiers: 3D geom and appearance (texture mapping)
 - Used in graphics applications, recovered from vision applications
- 3 Tier:
 - Macro - scene geometry (triangulation map)
 - Meso - fine scale geometric detail (displacement map)
 - * Micro - fine scale geometry/reflectance (texture map)
- Captured via sequential refinement

1.4 Multiscale Model

Geometry alone doesnt solve modeling, need multiscale model

Need

1. Geometry
2. Depth
3. Dynamic Texture

→ Rendering

Use image derivatives (know lighting changes, position of view, etc.) in forward way to render a diff. img (helps get photorealism)

1.5 Capgui

Step 1 - Calibration

Step 2 - Segmentation

Get rid of background

Step 3 - Shape From Silhouette

8-60 imgs

multiple views of same object → intersect **generalized cones** generated by each img to build a volume (guaranteed to contain object)

limiting smallest vol. obtainable in this way is known as the **visual hull** of the object

1.5.1 SFS methods

Voxel based (use voxel grid rep.)

inaccurate

triangulate w/ marching cubes algo

Image ray based (use image rays)

accurate

Axis aligned (use rectlinear rays (instead of camera rays), mark 'cut' points of image rays)

moderately accurate

fast

marching intersections algo

(mix of img ray and voxel based)

Step 4 - Phototextures + Texture Mapping

For each triangle in model, establish corresponding region in the phototextures

Difficulties:

- Tedious to sepcify texture coords. for every triangle

1.5.2 Common Text. Coord. Mappings

Orthogonal

Cylindrical

Spherical

Perspective Projection

Texture Chart (ie. text. split + flatten; cut object into pieces and map textures to each piece (piecwise planner))

1.5.3 Advanced Texture Splitting and Mapping

Floating Planes Method

- split into dozen - several dozen perspective mappings
- union of persp. planes accurately represent obj

LCSM (Least Squares Conformal Mapping)

- least square (locally) preserve orthogonality

Step 6 - Texture Basis Computation

1.6 Performance

Can have many gb of texture memory

Key issue: efficient memory access and processing

1. Macro - conventional geom processing
2. Meso - pixel shader; fixed code and variable data access
3. Micro - Shader/Registration comb.; fixed code and fixed data access

1.7 Meso Struct

Depth with respect to plane, doesn't work well with just one image (flat texture)

1.7.1 Computing Meso

Variational shape and reflectance Per point cost func:

$$\phi(\mathbf{X}, \mathbf{n}) = \sum_i h(\mathbf{X}, P_i) \|I_i(P_i(\mathbf{X})) - R(\mathbf{X}, \mathbf{n}, \mathbf{L})\|$$

$h \rightarrow$ visibility + sampling

$R \rightarrow$ reflectance

1.7.2 Rendering Meso

> 100 fps for consumer GPU

1.8 Micro Struct

Spatial texture basis

Render temporally varying dynamic texture by modulating a linear basis

Basis contains spatial derivs of img

Rendered by linear blending (?)

fixed execution and data access pattern

very fast implementation in graphics hardware

Can be done quickly in assembly (register extr.)

1.8.1 Dynamic Textures

3D geom and texture warp map b/w views and texture imgs

Diff texture img for each view;

A number of different misalignments

Planar error - incorrect texture coords

Out of plane error - object surface \neq texture plane

1.9 Spatial Basis Intro

Moving sine wave can be modeled

$$\begin{aligned} I(t) &= \sin(u + at) \\ &= \sin(u) \cos(at) + \cos(u) \sin(at) \\ &= \sin(u)y_1(t) + \cos(u)y_2(t) \end{aligned}$$

u spatially fixed basis

Small image motion

$$I = I_0 + \frac{\partial I}{\partial u} \Delta u + \frac{\partial I}{\partial v} \Delta v$$

Spatial fixed basis


1.10 Linear basis for spatio-temporal variation


On the obj./texture plane:

variation resulting from small warp perturbation

Taylor expansion

$$\begin{aligned} \text{[Diagram: A blue triangle with a green line segment labeled } \Delta\mu \text{ inside it]} &= \text{[Diagram: A blue triangle]} + \frac{\partial}{\partial \mu} (\text{[Diagram: A blue triangle]}) \Delta\mu + h.o.t. \\ T(\text{view}) &= T_0 + \frac{\partial}{\partial \mu} T_0 \Delta\mu + h.o.t. \end{aligned}$$



 Small if $\Delta\mu$ small
and T_0 smooth

Similarly: Can derive linear basis for out of plane and light variation!

1.11 Geometric spatial temporal variability

Image 'warp'

$$T(\mathbf{x}) = I(W(\mathbf{x}, \mu))$$

Image variability caused by imperfect warp

$$\Delta T = I(W(\mathbf{x}, \mu + \Delta\mu)) - T_w$$

First order approx.

$$\Delta T = I(W(\mathbf{x}, \mu)) + \nabla T \frac{\partial W}{\partial \mu} - T_w = \nabla T \frac{\partial W}{\partial \mu}$$

Concrete examples: img plane; out of plane

1.12 Variability due to a planar projective warp (homography)

1.13 Out of Plane Variability

1.14 Photometric Variation

light changes how obj looks (?)

don't need to raytrace

1.15 Composite Variability

composite texture intensity variability

$$\Delta \mathbf{T} = \Delta \mathbf{T}_s + \Delta \mathbf{T}_d + \Delta \mathbf{T}_l + \Delta \mathbf{T}_e$$

planar + depth + light + res. err.

Can be modeled as sum of basis

$$\begin{aligned} \Delta \mathbf{T} &= \mathbf{B}_s \mathbf{y}_s + \mathbf{B}_d \mathbf{y}_d + \mathbf{B}_l \mathbf{y}_l + \Delta(T_e) \\ &= \mathbf{B} \mathbf{y} + \Delta \mathbf{T}_e \end{aligned}$$

1.16 How to Compute

Slide 31 - 32

1.17 Dyntex