CMPUT 428: 3D Modeling

Roderick Lan

Contents

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

$\rightarrow$ 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 $\rightarrow$ 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 specify 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 (piecewise 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 ot plane, doesnt 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

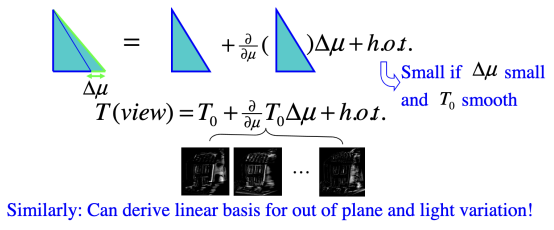
$$I = I_0 + \frac{\partial I}{\partial u}\Delta u + \frac{\partial I}{\partial v}\Delta v$$

Spatial fixed basis

1.10 Linear basis for spatio-temporal variation

On the obj./texture plane:

> variation resulting from small warp perturbation
>
> Taylor expansion


$= \quad + \frac{\partial}{\partial \mu}(\quad)\Delta\mu + h.o.t.$
$\Delta\mu$
Small if $\Delta\mu$ small and T_0 smooth
$T(view) = T_0 + \frac{\partial}{\partial \mu} T_0 \Delta\mu + h.o.t.$
...
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 (?)
dont need to raytrace

1.15 Composite Variability

composite texture intesity 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

$$\begin{aligned}\Delta\mathbf{T} &= \mathbf{B_s y_s} + \mathbf{B_d y_d} + \mathbf{B_l y_l} + \Delta(T_e) \\ &= \mathbf{By} + \mathbf{\Delta T_e}\end{aligned}$$

1.16 How to Compute

Slide 31 - 32

1.17 Dyntex

2 Lecture - Mar 19

lec07modeling18EmbMedia

2.1 Summary, 2 view rec

F independent of point correspondence, maps them to each other; same for all
can use real measurements to rectify

2.2 Multiview Stereo

Nonsequential image selections - features lost and reinitialized as new features

SOVE by matching with other *close* views

2.2.1 Relating more views

Relating nearby images
For every view i

```
    extract features
    compute 2 view geom $i-1$ / $i$ and matches
    Compute pose using robust algo
    For all close views $k$:
        compute two view geom $k$ / $i$ and matches
        infer new 2D - 3D matches and add to list
    Refine pose using all 2D - 3D matches
    Refine existing structure
    Initialize new struct
```

If viewpoints are close; most img changes can be modelled through a planar homography (can approx. map close imgs via homography)
Qualitative distance measure is obtained via residual error on best possible planar homography
Find 'best' homography (?)

2.3 Refining structure and Motion

Minimize reprojection error

$$\min_{\hat{P}_k, \hat{M}_i}$$

via MLE (if error zero mean gaussian noise); nonlinear opt.
Huge problem but can be solved efficiently (bundle adjustment)
Sometimes do this for simple scenes (usually use fundamental matrix method but can use several sets of affine models)
Quality of final doesnt depend on initial

2.4 Bundle Adjustment

Refining a captured model
Search method; change X and P until we get best match
Refine structure X_j and motion P^i
Minimize geometric error
ML solution, assuming noise is Gaussian
Tolerant to missing data

$$\min \sum_{i,j} d(PX, x)^2$$

2.5 Projective ambiguity and self-calibration

autocalibration - determine projective transformation T that upgrades the projective reconstruction to a metric one

2.6 Complete modeling system

Seq. of frame $\rightarrow$ scene struct

1. Get corresponding points (via. tracking or detect/match)

   tracking helpful for maintaining feature correspondence

2. Affine factorization (alr. computes ML estimate over all frames; no need for bundle adjustment for simple scenes)

3. Self calibration

4. If several model segments: merge, bundle adjust

2.7 Stereo rec

Sparse SFM $\rightarrow$ detailed map
Get depth map via stereo

2.8 Stereo image rect

Reproject images planes onto a common plane; planes parallel w/ line b/w optical center
All epipolar lines are **parallel** in the rectified image plane.
Rotate to make parallel $\rightarrow$ scan line matching

Take epipolar point $\rightarrow$ make infinite point $\rightarrow$ parallel epipolar lines
Do this in a way that minimizes distortion

3 Lecture - Mar 21

lec07modeling18EmbMedia

3.1 Stereo image rect cont.

Make use of epipolar geometry, move 2 separate image planes onto same plane to simplify search problem $\rightarrow$ epipolar lines go from defined by Fund matrix to being in canonical (rows corr. to rows)

Find epipolar point thats finite, get homography that maps it to point at infinity

Could choose any plane to rep. ray sets, but want planes that are similar to the imgs (dont compress x,y axis too much)

3.2 Stereo Matching Algos

Match pixels in conjugate epipolar lines

 assume brightness constancy

 many approaches

3.2.1 Basic algo

```
For each epipolar line:
    For each pixel in the left img:
        - compare with every pixel on same epipolar line in right img
        - pick pixel with minimum match cost
```

Improvement: match windows
Solve as energy minimization

3.2.2 Stereo as energy min

Find disparities d that minimize an energy func. $E(d)$
Simple pixel / window matching:

$$E(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

$C(x, y, d(x, y))$ is the SSD distance b/w windows $I(x, y)$ and $J(x, y + d(x, y))$
$C(x, y, d)$ - the disparity space image (DSI)

$$d(x, y) = \arg\min_{d'} C(x, y, d')$$

Simple pixel window matching - choose min of each col in DSI independently

3.3 Stereo Matching

Epipolar line gives 1D matching
Ordering
Uniqueness
Disparity Limit
Disparity Gradient Limit
Trade offs:

matching cost (data)

discontinuities (prior)

3.4 Disparity Map

$(x', y') = (x + D(x, y), y)$

3.5 Hierarchical stereo matching

Down sample w/ gaussian pyramid
Faster comp.
Deals w/ large disparity ranges

3.6

Sparse point cloud $\rightarrow$ can use splatting

3.7 SLAM - Online scene modeling

Triangulate points and reproject texture

4 Lecture - Mar 26

4.1 Methods of Modelling - MVS

4.1.1 Shape reconstruction

Given:

set of imgs of an obj / scene

camera calibration info

light cal. info

Find surface that best agrees w/ input imgs
Approach:

choose surface representation $\mathcal{S}, X$

define photo consistency func (+ regularization in practice) $\phi(X)$

solve minimization

$$\min_{\mathcal{S}} \int_{X \in \mathcal{S}} \phi(X) dX$$

4.1.2 Photo Consistency Function

$$\phi(X)$$

Based on img cues (shading, stereo, silhouette, ...)
Extension SFS/PS to multi view: (need cam + light calibration)

move cam $\rightarrow$ SFS

move obj. $\rightarrow$ PS

4.1.3 Surface Repr

Image-centered

depth/disparity wrt image plane

partial obj rec - limited res, viewpoint dependent

Object-centered

Voxels

Level sets (implicit - fold)

Mesh

Depth wrt a base mesh

Local patches

4.1.4 Volumetric Representation

obj: collection of voxels
normals: ?
method: carve away voxels that arent photoconsistent w imgs (discrete)
regularization: no way of ensuring smoothness
visibility: ensured by the order of traversal (in general a problem)

4.1.5 Disparity/Depth map

obj (surface): $s(x, y) = (x, y, f(x, y))$
normals:

$$\mathbf{u} = s_x \times s_y = [s_x, s_y, -1]^\top$$

method: find f tht best agrees w input imgs (min the cost functional integrated over surface)

$$\min \int_x \int_y \phi(x, y, f, \nabla f) dx dy$$

regularization: smoothness on ∇f

$$\phi_{smooth} = \nabla f^2$$

visibility: ? (fine mesh)

4.1.6 Depth wrt base mesh

obj (surface): $X = X^B + fd$ d = displacement dir. (displacement map)
normals: local (per triangle)

$$\mathbf{n} = [f_x, f_y, -1]^\top$$

transform to global cs
methods:

$$\min_f \int_{X^B} \phi(X^B, f, \nabla f) dx dy$$

regularization: smoothness on ∇f (local / global)
visibility: ? (fine mesh)

4.1.7 Mesh

obj: mesh vertices

$$X = \lambda_3 v_1 + \lambda_2 v_2 + \lambda_3 v_3$$

normals: interpolated

$$n = \lambda_1 n_1 + \lambda_2 n_2 + \lambda_3 n_3$$

$$n_j = \sum_{j,k,i \in \Delta(v_j)} (v_k - v_j) \times (v_i - v_j)$$

methods:
(finish)

4.2 voxel coloring

carve away voxels until photo consistent
depth ordering

4.3 space carving: photo hull

photo hull - union of all photo consistent shapes

4.4 graph cut

multiple possible regularizations

4.5 surface evolution