multi-view stereo; voxel coloring

Jason Lawrence (some slides by Szymon Rusinkiewicz & Steve Seitz)
CS 651, Spring 2007: Computer Vision
why more than 2 views?

- camera baseline:
  - too short => low accuracy
  - too long => matching becomes hard
why more than 2 views?

- ambiguity with only two views
trinocular stereo

• straightforward approach to eliminate bad correspondences:
  - pick 2 views, find correspondences
  - for each matching pair, reconstruct 3D point
  - project point into 3rd image
  - can’t find correspondence near predicted location then reject
trinocular stereo

- trifocal geometry: relations between points in three camera views
- trifocal tensor: analogue of essential matrix
- 3x3x3 trilinear tensor (3D cube of numbers)
- given lines in 2 views, predict lines in 3rd
multibaseline stereo

- slightly different algorithm for n cameras:
  - pick one reference view
  - for each candidate depth
    - compute sum of squared differences to all other views, assuming correct disparity for view
  - resolve ambiguities: only correct depths will constructively interfere
multibaseline stereo

Fig. 5. SSD values versus inverse distance: (a) $B = h$; (b) $B = 2h$; (c) $B = 3h$; (d) $B = 4h$; (e) $B = 5h$; (f) $B = 6h$; (g) $B = 7h$; (h) $B = 8h$. The horizontal axis is normalized such that $\delta d F = 1$. 
multibaseline stereo

[Okutami & Kanade]
multibaseline stereo reconstruction
multibaseline stereo

Figure 7: The CMU Video-Rate Stereo Machine Prototype System: (a) camera head; (b) processor boards
problems with multibaseline stereo

- have to pick a reference view
- occlusion
  - with many cameras / large baseline, occlusion becomes likely
  - contributes incorrect values to error function
volumetric multiview approaches

- goal: find a model consistent with images
- “model-centric” (vs. image-centric)
- typically use discretized volume (voxel grid)
- for each voxel, compute occupied / free (for some algorithms, also color, etc.)
photo consistency

- result: guaranteed to be **consistent** with images, but not necessarily the correct scene
- i.e., many scenes produce the same images
silhouette carving

- find silhouettes in all images
- exact version: back-project all silhouettes, find intersections

binary images
silhouette carving

- find silhouettes in all images
- exact version: back-project all silhouettes, find intersections
silhouette carving

- limit of silhouette carving is visual hull or line hull
- complement of lines that don’t intersect object
- in general not the same as object
  - can’t recover “pits” in object
- not the same as convex hull
silhouette carving

- discrete version:
  - loop over all voxels in some volume
  - if projection into images lies inside silhouettes, mark as occupied
  - else mark as free
silhouette carving
voxel coloring

- Seitz and Dyer, 1997
- in addition to free / occupied, store color at each voxel
- explicitly accounts for occlusion
voxel coloring

- basic idea: sweep through a voxel grid
  - project each voxel into each image in which it is visible
  - if colors in images agree, mark voxel with color
  - else, mark voxel as empty
- agreement of colors based on comparing standard deviation of colors to threshold
voxel coloring and occlusion

- problem: which voxels are visible?
- solution, part 1: constrain camera views
  - when a voxel is considered, necessary occlusion information must be available
  - sweep occluders before occludees
  - constrain camera positions to allow this sweep
voxel coloring sweep order

Layers

Scene Traversal
voxel coloring camera positions

inward-looking cameras above scene

outward-looking cameras inside scene
panoramic depth ordering

• cameras oriented in many different directions
• planar depth ordering does not apply
panoramic depth ordering

* layers radiate outward from cameras
panoramic depth ordering

- layers radiate outward from cameras
Panoramic depth ordering

- layers radiate outward from cameras

[Seitz]
voxel color and occlusion

- solution, part 2: per-image mask of which pixels have been used
  - each pixel only used once
  - mask filled in as sweep progresses
data acquisition

- calibrated turntable
- 360 rotation (21 images)

selected dinosaur images

selected flower images
voxel coloring results

Dinosaur Reconstruction
72 K voxels colored
7.6 M voxels tested
7 min. to compute
on a 250MHz SGI

Flower Reconstruction
70 K voxels colored
7.6 M voxels tested
7 min. to compute
on a 250MHz SGI
voxel coloring results

- with texture: good results
- without texture: regions tend to “bulge out”
  - voxels colored at earliest time at which projection into images is consistent
  - model good for re-rendering: image will look correct for viewpoints near the original ones
limitations of voxel coloring

- view-independent depth order may not exist
- need more powerful general-case algorithms
  - unconstrained camera positions
  - unconstrained scene geometry/topology
space carving

* initialize to a volume $V$ containing true scene
* choose a voxel on the current surface
* project to visible input images
* carve if not photo-consistent
multi-pass plane sweep

- faster alternative:
  - sweep plane in each of 6 principal directions
  - consider cameras on only one side of plane
  - repeat until convergence
multi-pass plane sweep

true scene

reconstruction
multi-pass plane sweep
multi-pass plane sweep
multi-pass plane sweep
multi-pass plane sweep
multi-pass plane sweep
space carving results: african violet
space carving results: hand

Input Image
(1 of 100)

Views of Reconstruction