Lecture Outline

• Multi-view Stereo part II

• Sources:
  – Slides by Y. Furukawa, G. Vogiatzis, L. Zhang
Representations

- Depth maps
- Point clouds
- Surface patches
- Level sets
- Voxels grids
- Meshes
- ...

[Images of different representations]
Depth Maps

• Compact representation
  – 3D quantities (points) can be indexed via pixel coordinates
  – Easy to determine neighborhood relationships and connectivity
  – Plane sweeping can be done very efficiently on GPUs

• Enable straightforward visibility estimation

• Viewpoint dependent
  – Does not allow more than one layer (2 ½ D representation)
  – Viewpoint cannot be altered without revealing holes

• Depth maps are no consistent after fusion
  – They are redundant when they are consistent...
The Visibility Problem

• Which points are visible in which images?

Known Scene

Forward Visibility

Unknown Scene

Inverse Visibility
Patch-based MVS (PMVS)

Y. Furukawa and J. Ponce (PAMI 2010)

Code available at
http://grail.cs.washington.edu/software/pmvs/
What is a Patch?

- Patch consists of
  - Position \((x, y, z)\)
  - Normal \((n_x, n_y, n_z)\)
  - Extent \((\text{radius})\)

- Tangent plane approximation
What is a Patch?

- Patch consists of
  - Position \((x, y, z)\)
  - Normal \((nx, ny, nz)\)
  - Extent \((radius)\)

- Tangent plane approximation

![Diagram of a Patch](image)
Why Patches?

• Flexible
Why Patches?

• **Flexible** ↔ **Hard to enforce regularization**
Why Patches?

- **Flexible** ↔ **Hard to enforce regularization**

![Diagram showing matches and non-matches between 9x9 pixel patches.](image-url)
Why Patches?

• Flexible ↔ Hard to enforce regularization

Regularization not really necessary because

Local image patch is descriptive enough
Why Patches?

• Pure 3d data without interpolation
Why Patches?

- Extracts pure 3d data w/o interpolation

Meshing w/ standard interpolation

Scene analysis from pure 3d data

Meshing w/ smart interpolation
By the way...

• Depthmap-fusion
  – is also “flexible”
  – can also extract pure 3d data
Patches vs. Multiple Depth Maps
(according to Y. Furukawa)

• Patches $\rightarrow$ **Single global 3D model**
  Depthmaps $\rightarrow$ **Multiple redundant 3D models**

• Patches $\rightarrow$ **Clean 3D points**
  Depthmaps $\rightarrow$ **Noisy without merging**

• Patches $\rightarrow$ **Hard to make it fast**
  Depthmaps $\rightarrow$ **Easy to make it fast**
Patch-based MVS

[Lhuillier and Quan, PAMI 05]

[Furukawa and Ponce, CVPR 07 and PAMI 2010]

[Habbecke and Kobbelt, CVPR 07]
Patch Definition

- Patch $p$ is defined by
  - Position $c(p)$
  - Normal $n(p)$
  - Visible images $V(p)$

- Extent is set so that $p$ is roughly 9x9 pixels in $V(p)$
Photo-consistency

• Photo-consistency $\mathcal{N}(I, J, p)$ of $p$ between two images $I$ and $J$

$I_{xy}$: pixel color in image $I$
Photo-consistency

- Photo-consistency $N(I, J, p)$ of $p$ between two images $I$ and $J$

  $I_{xy}$: pixel color in image $I$
  $J_{xy}$: pixel color in image $J$

  \[
  N(I, J, p) = \frac{\sum (I_{xy} - \overline{I_{xy}}) \cdot (J_{xy} - \overline{J_{xy}})}{\sqrt{(I_{xy} - \overline{I_{xy}})^2 \cdot (J_{xy} - \overline{J_{xy}})^2}}
  \]
Photo-consistency

- Photo-consistency $N(I, J, p)$ of $p$ between two images $I$ and $J$

Photo-consistency $N(p)$ of $p$ with visible images $V(p) = \{I_1, I_2, \ldots, I_n\}$

$$N(p) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} N(I_i, I_j, p)}{(n+1)n/2}$$
Reconstruct Patch $p$

- Given initial estimates of
  - Position $c(p)$
  - Normal $n(p)$
  - Visible images $V(p)$
- $\{c(p), n(p)\} = \arg\max_{\{c(p), n(p)\}} N(p)$
Reconstruct Patch $p$

- Given initial estimates of:
  - Position $c(p)$
  - Normal $n(p)$
  - Visible images $V(p)$

- $\{c(p), n(p)\} = \text{arg max } N(p)$ for $\{c(p), n(p)\}$
Reconstruct Patch $p$

- Given initial estimates of
  - Position $c(p)$
  - Normal $n(p)$
  - Visible images $V(p)$

- $\{c(p), n(p)\} = \arg\max N(p)$

2 DOF
Verify a Patch

• Textures may match by accident
• Photo-consistency must be reasonably high

• Verification process
  – Keep only high photo-consistency images in $\mathcal{V}(\rho)$
  – Accept if $|\mathcal{V}(\rho)| \geq 3$
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image4} \}$
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$
Update $V(p)$

$V(p)=\{\text{Image1, Image2, Image3, Image4}\}$

$N(\text{Image1, Image2, } p)=0.75$

$N(\text{Image2, Image3, } p)=0.83$

$N(\text{Image3, Image4, } p)=0.58$

Sum $= 2.16$
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$

$N(\text{Image1, Image2, } p) = 0.75$

Sum = 2.16

Sum = 1.98

Specular Highlights!
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$

![Diagram showing relationships between images with sums and weights]
Update $V(p)$

$V(p) = \{\text{Image1}, \text{Image2}, \text{Image3}, \text{Image4}\}$

Sum = 2.16
Sum = 1.98
Sum = 1.99
Sum = 1.55
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$

Sum = 2.16

Specular Highlights!
Update $V(p)$

$V(p) = \{\text{Image1}, \text{Image2}, \text{Image3}, \text{Image4}\}$

Sum = 2.16

$N(\text{Image1}, \text{Image2}, p) = 0.75$

Add Image2, because > 0.7

Specular Highlights!
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$

Sum = 2.16

Remove Image3, because < 0.7

Specular Highlights!
Update $V(p)$

$V(p) = \{\text{Image1, Image2, Image3, Image4}\}$

Add Image4, because $> 0.7$

Sum = 2.16

0.83
Algorithm Overview

#1. Feature detection
#2. Initial feature matching
#3. Patch expansion and filtering
Feature Detection

• Extract local maxima of
  – Harris corner detector (corners)
  – Difference of Gaussian (blobs)
Algorithm Overview

#1. Feature detection
#2. Initial feature matching
#3. Patch expansion and filtering
Initial feature matching

\( c(p): \text{triangulation} \)

\( n(p): \)

\( V(p): \)
Initial feature matching

\( c(p) \): triangulation
\( n(p) \): parallel to \textit{Image1}
\( V(p) \): \{\textit{Image1, Image2}\}
Initial feature matching

\( c(p) \): triangulation
\( n(p) \): parallel to \( \text{Image} 1 \)
\( V(p) \): \{\( \text{Image} 1, \text{Image} 2 \}, \text{Image} 3 \}\n
Add visible images
If \( N(\text{Image} 1, \text{Image} 3, p) > 0.5 \)
Initial feature matching

c(p): refine
n(p): refine
V(p): \{Image1, Image2, Image3\}
\{c(p), n(p)\} = \arg\max N(p)_{\{c(p), n(p)\}}
Initial feature matching

$c(p)$: refine
$n(p)$: refine
$V(p)$: \{Image1, Image2, Image3\}

$$\{c(p), n(p)\} = \arg \max_{\{c(p), n(p)\}} N(p)$$
Initial feature matching

Verification
(update $V(p)$ and check $|V(p)| \geq 3$)
Initial feature matching
Initial feature matching
Initial feature matching

Image 1

Occupied

Image 2

Epipolar line

Image 3
Initial feature matching

• Repeat for all image features
Algorithm Overview

#1. Feature detection
#2. Initial feature matching
#3. Patch expansion and filtering
Patch expansion
Patch expansion

Pick a patch
Patch expansion

Pick a patch

Look for neighboring empty pixels

All occupied — Do nothing

Image 1

Image 2

Image 3
Patch expansion

Pick a patch

$p$
Patch expansion

Pick a patch

Identify neighboring empty pixels

Image 1

Image 2

Image 3
Patch expansion

Reconstruct a patch visible in an empty pixel

\[ p, q \]

\[ c(q): \{ \text{tangent plane of } p \text{ intersects w/ ray} \} \]

\[ n(q): \]

\[ V(q): \]
Patch expansion

Reconstruct a patch visible in an empty pixel

\[ c(q) : \{ \text{tangent plane of } p \text{ intersects w/ ray} \} \]
\[ n(q) : n(p) \]
\[ V(q) : V(p) \]
Patch expansion

Reconstruct a patch visible in an empty pixel

\[ c(q) : \text{refine} \]
\[ n(q) : \text{refine} \]
\[ V(q) : V(p) \]
Patch expansion

Reconstruct a patch visible in an empty pixel

\[ p \xrightarrow{c(q)} q \] : refine

\[ n(q) : \text{refine} \]

\[ V(q) : V(p) \]

Patch verification!
Patch expansion

Reconstruct a patch visible in an empty pixel

$c(q)$: refine

$n(q)$: refine

$V(q): V(p)$

Patch verification!
Patch expansion

Repeat
• for every patch
• for every neighboring empty pixel
Patch filtering

- Visibility consistency

Filter out $p_i$ if

$$| V(p_1) | N(p_1) < \sum_{i=2}^{6} N(p_i)$$
Limitations of MVS

- Works well for various objects and scenes
- Surfaces must be Lambertian and well-textured
- Problematic for architectural scenes
Recent Literature

[Zebedin et al., ECCV 2008]

[Furukawa et al., CVPR 2009]

[Sinha et al., ICCV 2009]
Poisson Surface Reconstruction

- Input: points with oriented normals (pointing outwards)
- Output: dense, connected, triangle mesh

http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version5.5/
Extracting a Surface from Photo-consistency

• Vogiatzis et al. (PAMI 2007)
• Divide the space in voxels
• Compute the photo-consistency of each voxel
  – By robustly combining all pairwise NCC scores
• Problem: find a minimum cost surface that separates interior from exterior of the object
• Add term that favors large volume, otherwise solution collapses to a point
How to Solve?
Minimum Cut

Can be computed in polynomial time with Ford-Fulkerson (1956) algorithm
Three Equivalent Representations

Continuous functional

\[ E[S] = \iint_S \rho(x) dS + \iiint_{V(S)} \sigma(x) dV \]
Extracting the Surface

- Marching cubes algorithm can extract isosurfaces
  - Matlab: [tri, pts] = isosurface(V)
  - Where V is a binary volume of 0s and 1s
Results
Results
Volumetric Stereo

- Determine occupancy, “color” of points in $V$
- Slides by L. Zhang
Discrete Formulation: Voxel Coloring

Goal: Assign RGBA values to voxels in V \textit{photo-consistent} with images

Discretized Scene Volume

Input Images (Calibrated)
Complexity and Computability

Discretized Scene Volume

\(N^3\) voxels

\(C\) colors

Photo-Consistent Scenes

True Scene

All Scenes \((C^{N^3})\)
Reconstruction from Silhouettes (C=2)

• Approach:
  • *Back-project* each silhouette
  • Intersect back-projected volumes
Volume Intersection

Reconstruction Contains the True Scene

• But is generally not the same
• In the limit (all views) we get *visual hull*
  > Complement of all lines that do not intersect S
Voxel-based Algorithm

Color voxel black if on silhouette in every image

- $O(\ ? \ )$, for $M$ images, $N^3$ voxels
- Don’t have to search $2^{N^3}$ possible scenes!
Results (Franco and Boyer, PAMI 2009)
Properties of Volume Intersection

Pros
– Easy to implement, fast
– Accelerated via octrees

Cons
– No concavities
– Reconstruction is not photo-consistent
– Requires identification of silhouettes
Space Carving

Space Carving Algorithm

- Initialize to a volume $V$ containing the true scene
  - Choose a voxel on the current surface
  - Project to visible input images
  - Carve if not photo-consistent
  - Repeat until convergence
Which Shape do You Get?

The **Photo Hull** is the **UNION of all photo-consistent scenes in** $V$

- It is a photo-consistent scene reconstruction
- Tightest possible bound on the true scene
Results (Kutulakos and Seitz, IJCV 2000)