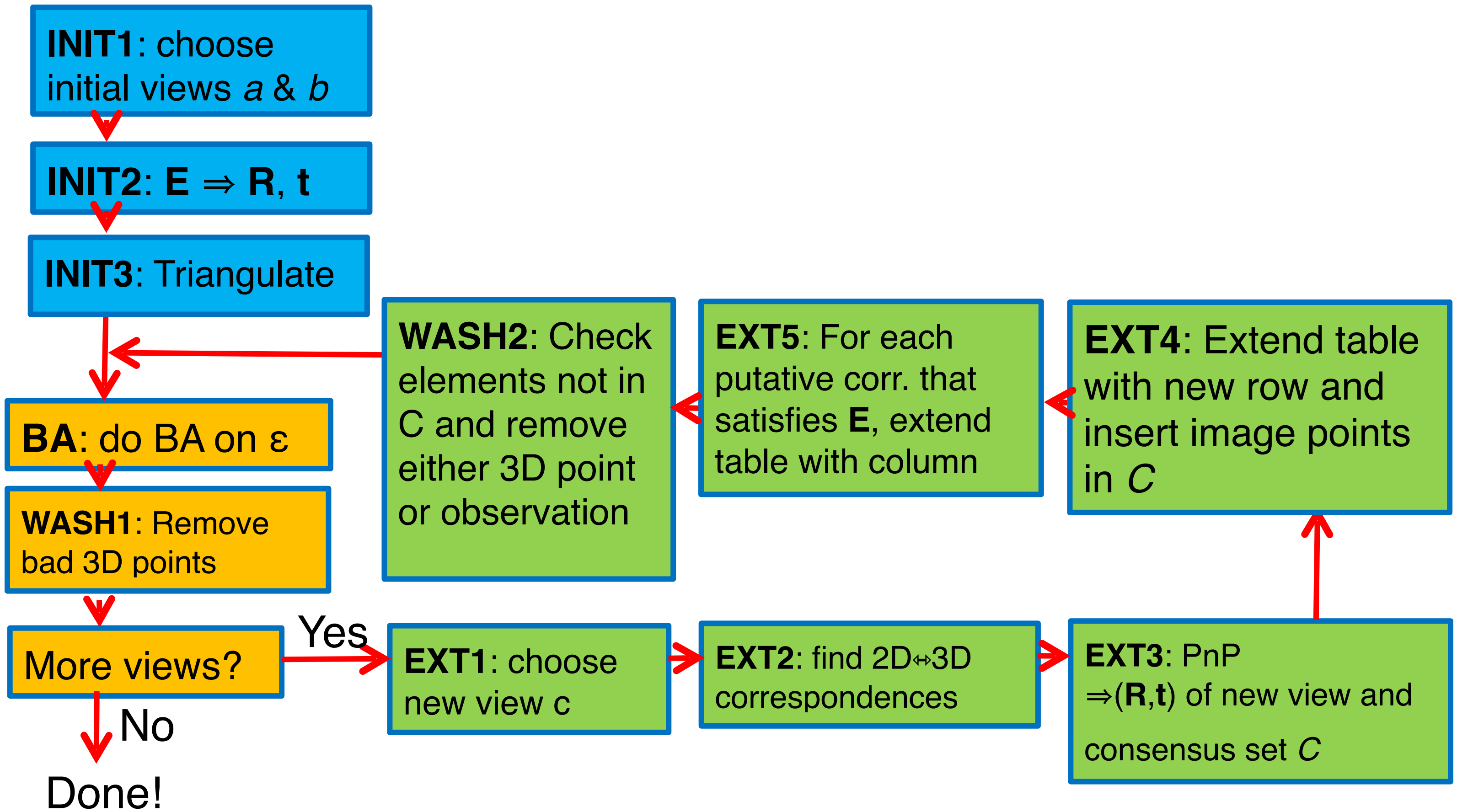
TSBB15
Computer Vision

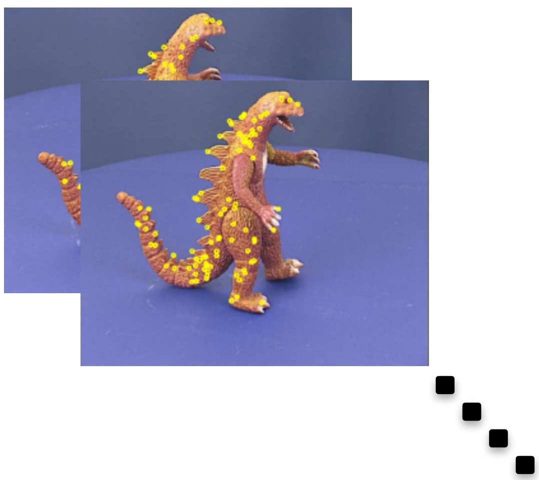
Lecture 14
Multi-view stereo

Per-Erik Forssén

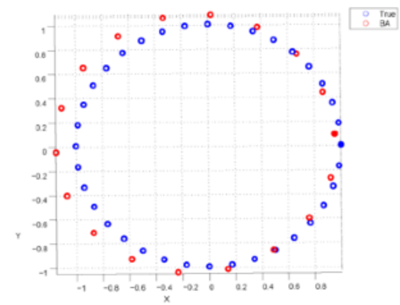
Recap: Incremental SfM pipeline from Lecture 12

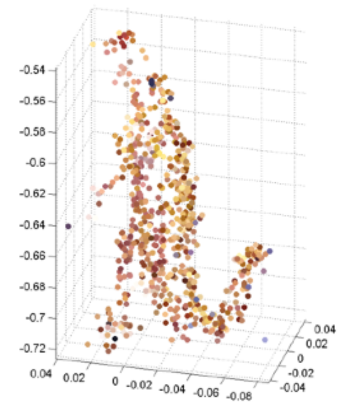


Incremental SfM



⇒





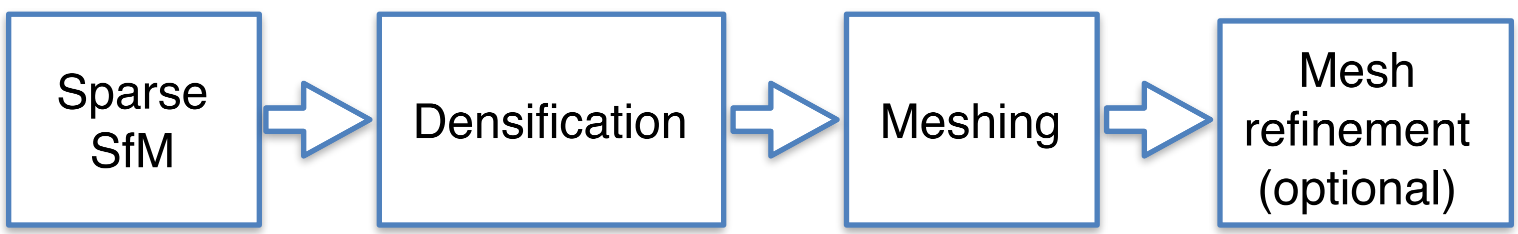
Results from 2011 project by Bertil Grelsson and Freddie Åström

The output of incremental SfM is a sparse 3D model and a set of camera poses.

Dense 3D models

The output of incremental SfM is a sparse 3D model and a set of camera poses.

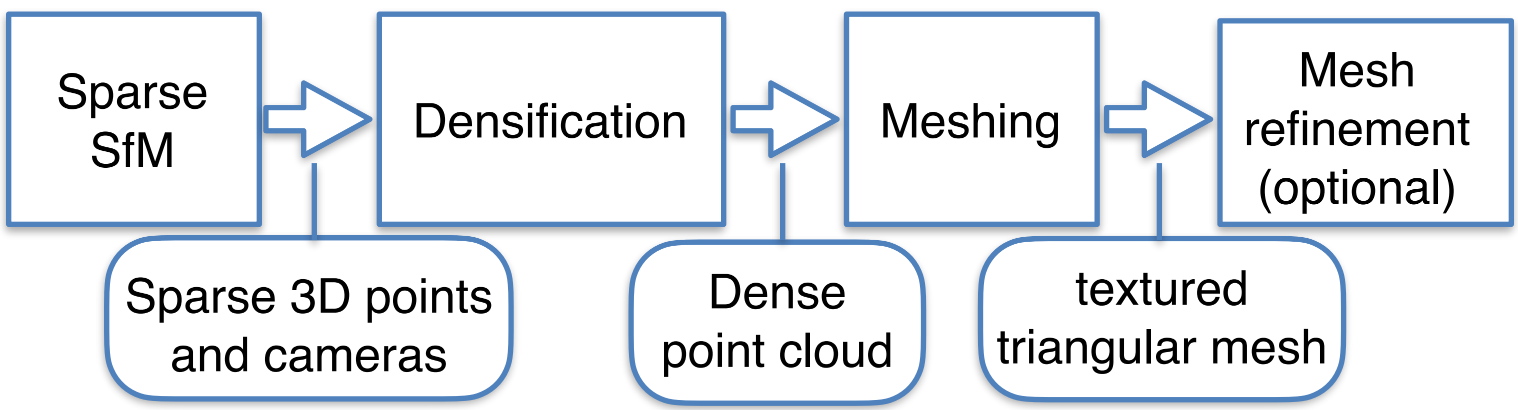
In commercial 3D modelling systems, sparse SfM is followed by two or three additional steps:



Dense 3D models

The output of incremental SfM is a sparse 3D model and a set of camera poses.

In commercial 3D modelling systems, sparse SfM is followed by two or three additional steps:



Densification Approaches

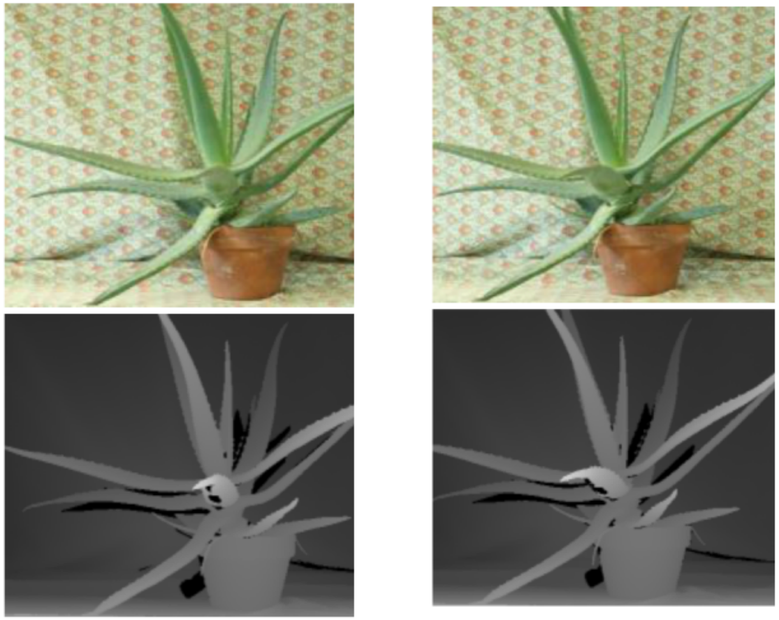
Densification needs at least two views.

- **Two-view stereo** methods need **view selection**
  - we want a wide baseline
  - but also many correspondences
  - same criteria as for the initial pair in incremental SfM

  See e.g. Schönberger&Frahm, *Structure from Motion Revisited*, **CVPR16** (linked on the project 2 page)
- **Multi-view stereo** methods are in general more accurate, but also much more expensive.

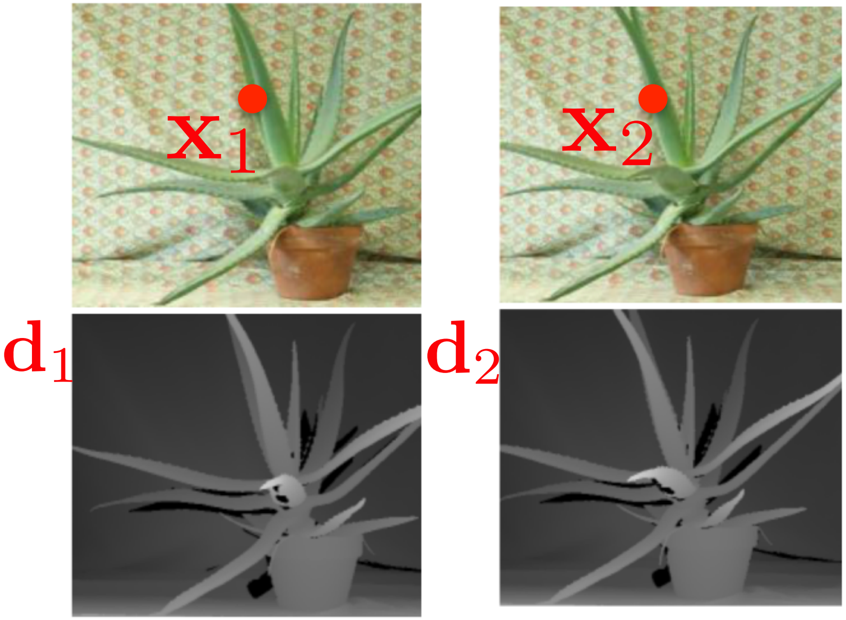
Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf



Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf



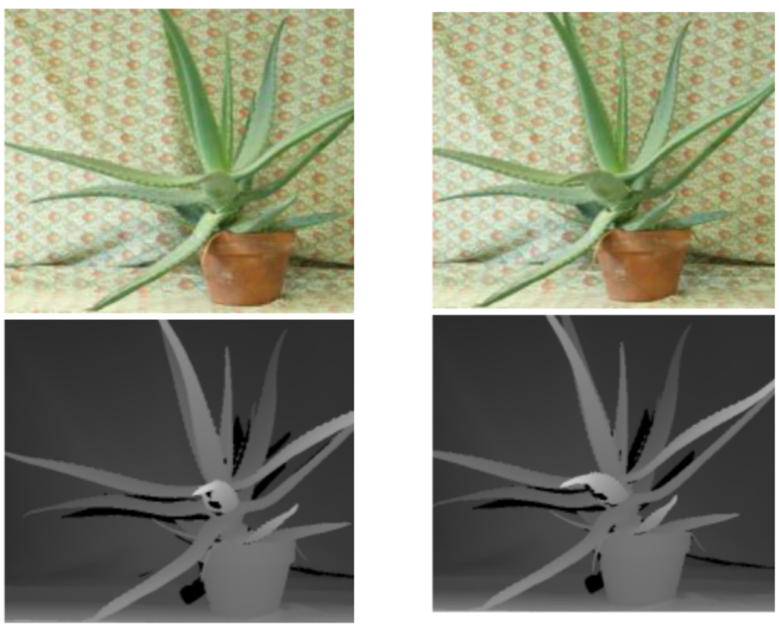
These **disparity maps** **d**(**x**) encode correspondences ($\mathbf{x}_1$,$\mathbf{x}_2$) as:

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{d}_1(\mathbf{x}_1)$$

$$\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{d}_2(\mathbf{x}_2)$$

Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf



These **disparity maps** $\mathbf{d}(\mathbf{x})$ encode correspondences ($\mathbf{x}_1$,$\mathbf{x}_2$) as:

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{d}_1(\mathbf{x}_1)$$
$$\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{d}_2(\mathbf{x}_2)$$

Other algorithms instead use **correspondence maps** $\mathbf{c}(\mathbf{x})$, and then we have:

$$\mathbf{x}_2 = \mathbf{c}_1(\mathbf{x}_1)$$
$$\mathbf{x}_1 = \mathbf{c}_2(\mathbf{x}_2)$$

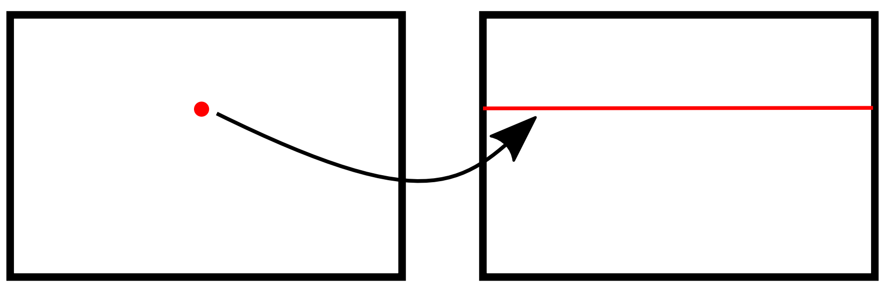
Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02]
  http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
  1. Rectify images to have horizontal epipolar lines (See TSBB06) This results in the fundamental matrix

$$\mathbf{F}_{\mathrm{R}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \sim (\mathbf{H}_1^{-1})^T \mathbf{F} \mathbf{H}_2^{-1}$$

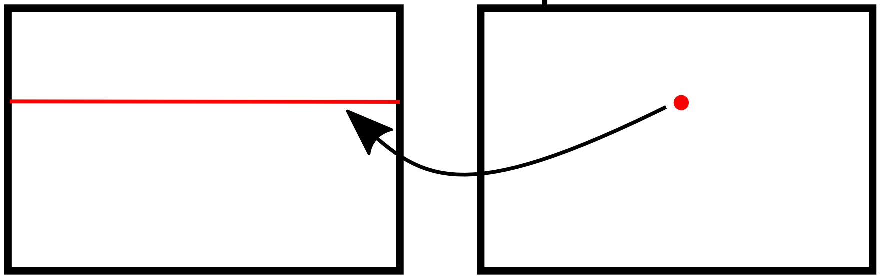
Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02]
  http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
  1. Rectify images to have horizontal epipolar lines
  2. For each point in the left image we then search for a corresponding point only on the line with the same y-coordinate. E.g. with block matching.



Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02]
  http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
  1. Rectify images to have horizontal epipolar lines
  2. For each point in the left image we then search for a corresponding point only on the line with the same y-coordinate. E.g. with block matching.
  3. Do the same in the right image, and remove inconsistencies in the correspondence maps.



Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02]
  http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
  1. Rectify images to have horizontal epipolar lines
  2. For each point in the left image we then search for a corresponding point only on the line with the same y-coordinate. E.g. with block matching.
  3. Do the same in the right image, and remove inconsistencies in the correspondence maps.
  I.e. check that these are small:

$$J_1(\mathbf{x}_1) = \|\mathbf{x}_1 - \mathbf{c}_2(\mathbf{c}_1(\mathbf{x}_1))\|$$

$$J_2(\mathbf{x}_2) = \|\mathbf{x}_2 - \mathbf{c}_1(\mathbf{c}_2(\mathbf{x}_2))\|$$

Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
- PatchMatch on two frames, followed by epipolar constraint. [Barnes et al. SIGGRAPH09] https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/

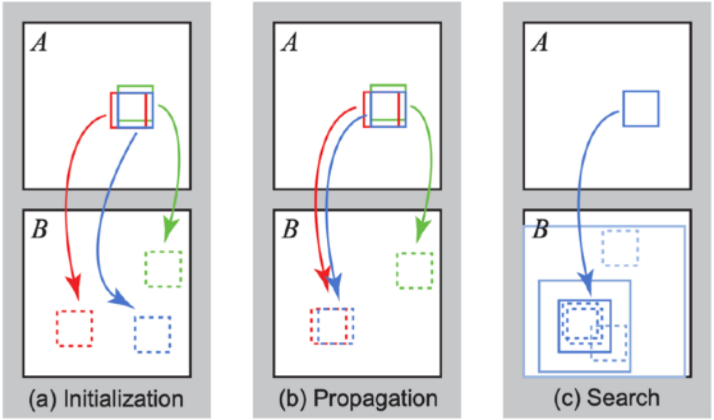


Image from Barnes et al. SIGRRAPH'09

Two-view Densification

- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
- PatchMatch on two frames, followed by epipolar constraint. [Barnes et al. SIGGRAPH09] https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/

(a) View of the scene. (b) Sparse point cloud from Kontiki (c) Result after densification.

Images from CDIO-project GoPro Trails 2018

Two-view Densification

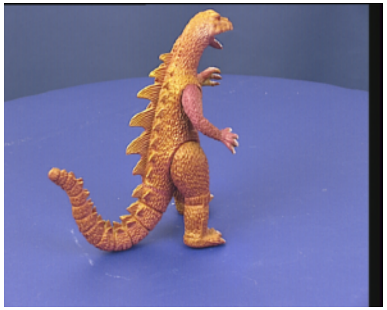
- Classic stereo, using two images and the epipolar constraint [Scharstein & Szeliski IJCV02] http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
- PatchMatch on two frames, followed by epipolar constraint. [Barnes et al. SIGGRAPH09] https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/
- Depth map search by optimization.
  Can be parallelized on GPU using the plane-sweep algorithm. [Gallup et al. CVPR07] https://inf.ethz.ch/personal/pomarc/pubs/GallupCVPR07.pdf

Multi-view Densification

- Multi-view methods, e.g. from the Furukawa&Hernández tutorial.
- Other methods on leaderboards for MVS datasets:
  Middlebury:
  https://vision.middlebury.edu/mview/
  Tanks and temples:
  https://www.tanksandtemples.org
  ETH 3D:
  https://www.eth3d.net/overview
  DTU dataset:
  http://roboimagedata.compute.dtu.dk/
  Robust vision challenge:
  http://www.robustvision.net

Meshing approaches

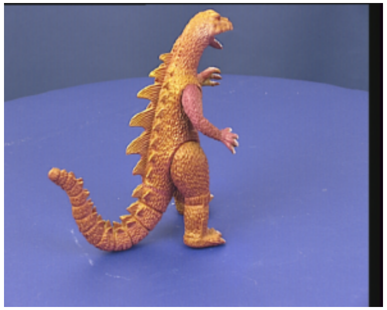
- There are two main sources of information for volumetric methods:
- **Object outlines**/silhouettes



$\Rightarrow$

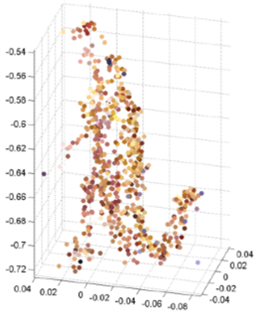
Meshing approaches

- There are two main sources of information for volumetric methods:
- **Object outlines**/silhouettes



$\Rightarrow$

- **3D points**



Meshing approaches

- There are two main sources of information for volumetric methods:
- **Object outlines**/silhouettes
  A classic silhouette method is **space carving**, see:

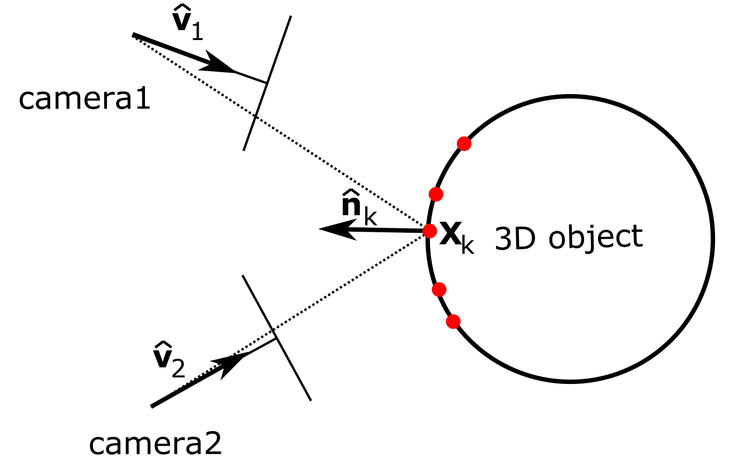
  A. Fitzgibbon, et al., *Automatic 3D Model Construction for Turn-Table Sequences*, Springer Verlag 1998
  Linked on the project webpage.

- **3D points**
  Classic methods use Delaunay tetrahedralization from convex hull of the point cloud. Or triangulation from successive projections of point cloud. See the Furukawa&Hernández tutorial.

Meshing approaches

- **Volumetric methods** compute a volume from the 3D points.
- Such methods are more robust to errors in the 3D points (both noisy points, and outliers)
- They require an **oriented point cloud** as input, i.e. each 3D point $\mathbf{X}_k$, should have a surface normal $\mathbf{n}_k$.

  - $\mathbf{n}_k$ can be determined up to sign from neighbours of $\mathbf{X}_k$
  - Sign can be determined by requiring that $\mathbf{v}^T\mathbf{n}_k<0$ for cameras that see $\mathbf{X}_k$



Meshing approaches

- **Volumetric methods** compute a volume from the 3D points.
- **Opt 1**: Define the volume as a density:

$$V(x,y,z)=\tau \quad \tau\in[0,1]$$

τ=0 means free space τ=1 means fully occupied.

E.g. M. Kazhdan, H. Hoppe, *Screened Poisson Surface Reconstruction*, **ToG** 2013

- **Opt 2**: Define the volume as a truncated signed distance to the surface

$$D(x,y,z)=d \quad d\in[-d_{\max},d_{\max}]$$

E.g. B. Curless, M. Levoy, *A Volumetric Method for Building Complex Models from Range Images*, **SIGGRAPH'96**

Voxels to Mesh

- Voxels can be converted to a mesh using
  marching cubes:
  W. Lorenzen, H. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, **SIGGRAPH'87**
  https://dl.acm.org/doi/10.1145/37401.37422
- in **ray casting**, a ray is cast from each pixel in a camera, and stopped at the first surface intersection.
  R. Newcombe et al. *KinectFusion: Real-time Dense Surface Mapping and Tracking*. **ISMAR'11**
  + This method is very fast.
  - However, a mesh generated in this way may have holes if viewed from other directions.

Mesh texture sampling

- Normally a textured mesh is desired. The texture is obtained by sampling from the input images.
- For each triangle in the mesh, a suitable frame is selected. Desirable properties include:
  - The area of the projected triangle in the image should be large
  - The texture resolution on the 3D surface should be the same in all directions.

Mesh refinement

Mesh refinement is covered in the Furukawa and Hernández tutorial.

Image source: A. Fitzgibbon, G. Cross and A. Zisserman, Automatic 3D Model Construction for Turn-Table Sequences, in 3D Structure from Multiple Images of Large-Scale Environments , Editors Koch & Van Gool, Springer Verlag 1998