Dense 3D Reconstruction

Christiano Gava
christiano.gava@dfki.de

Thanks to Yasutaka Furukawa

Outline

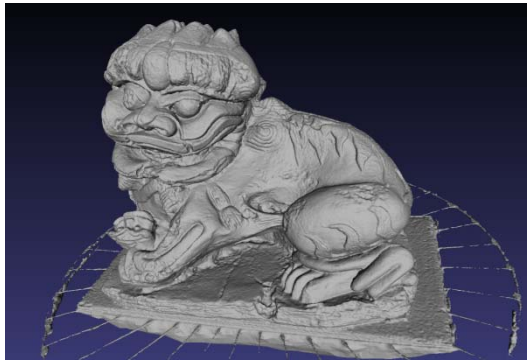
- Previous lecture: structure and motion II
  - Structure and motion loop
  - Triangulation
  - Wide baseline matching (SIFT)

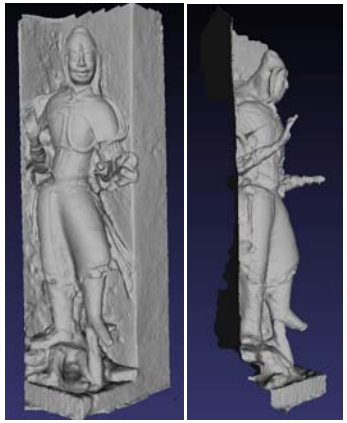
- Today: dense 3D reconstruction
  - The matching problem
  - 2-view reconstruction (disparity maps)
  - Multi-view reconstruction (point clouds)

- Next lecture: structured light

Why dense reconstruction?

- Accurate 3D models → cultural heritage!





- Reconstruction of houses, buildings, famous touristic sites…

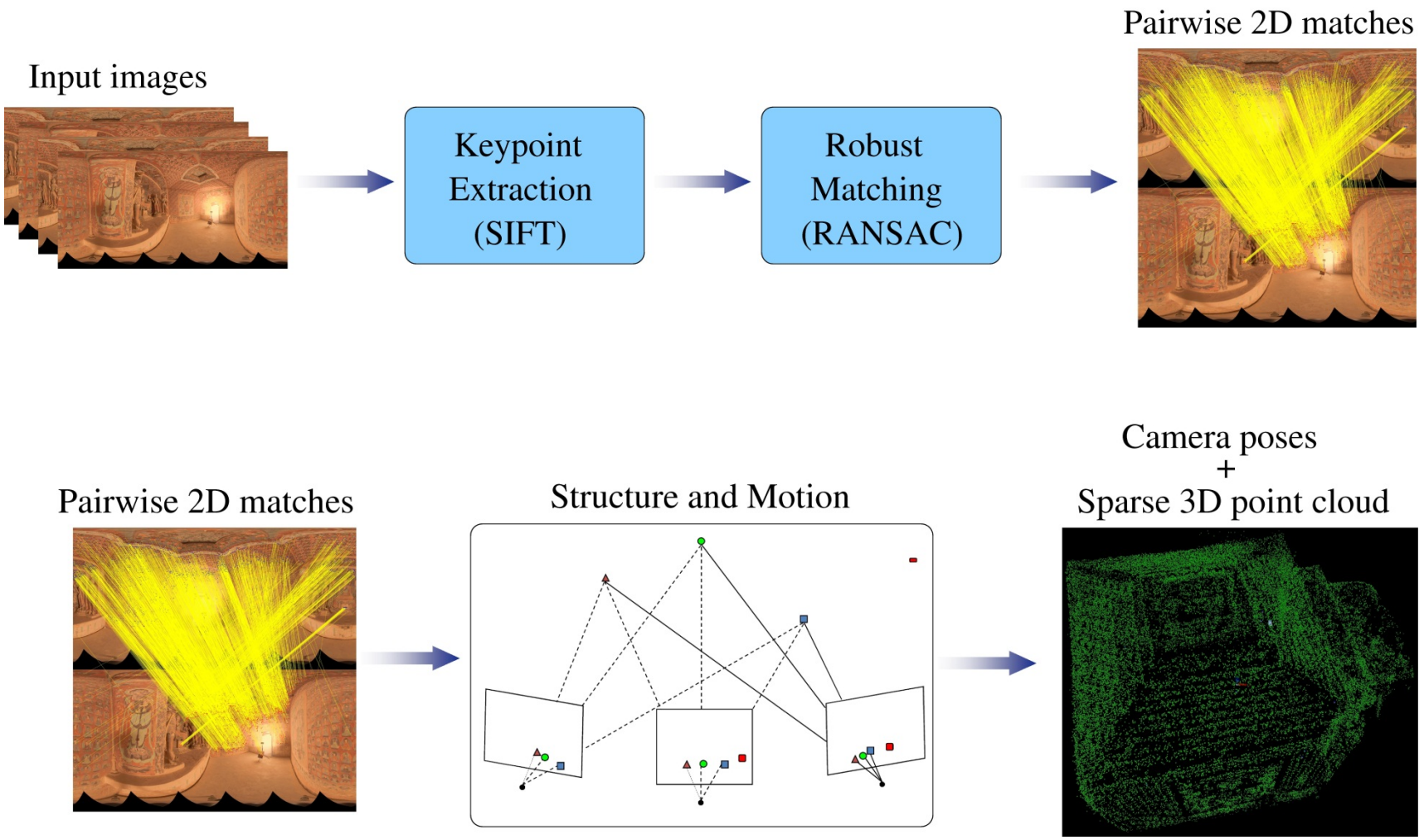
Piazza San Marco – Venice

13.703 images

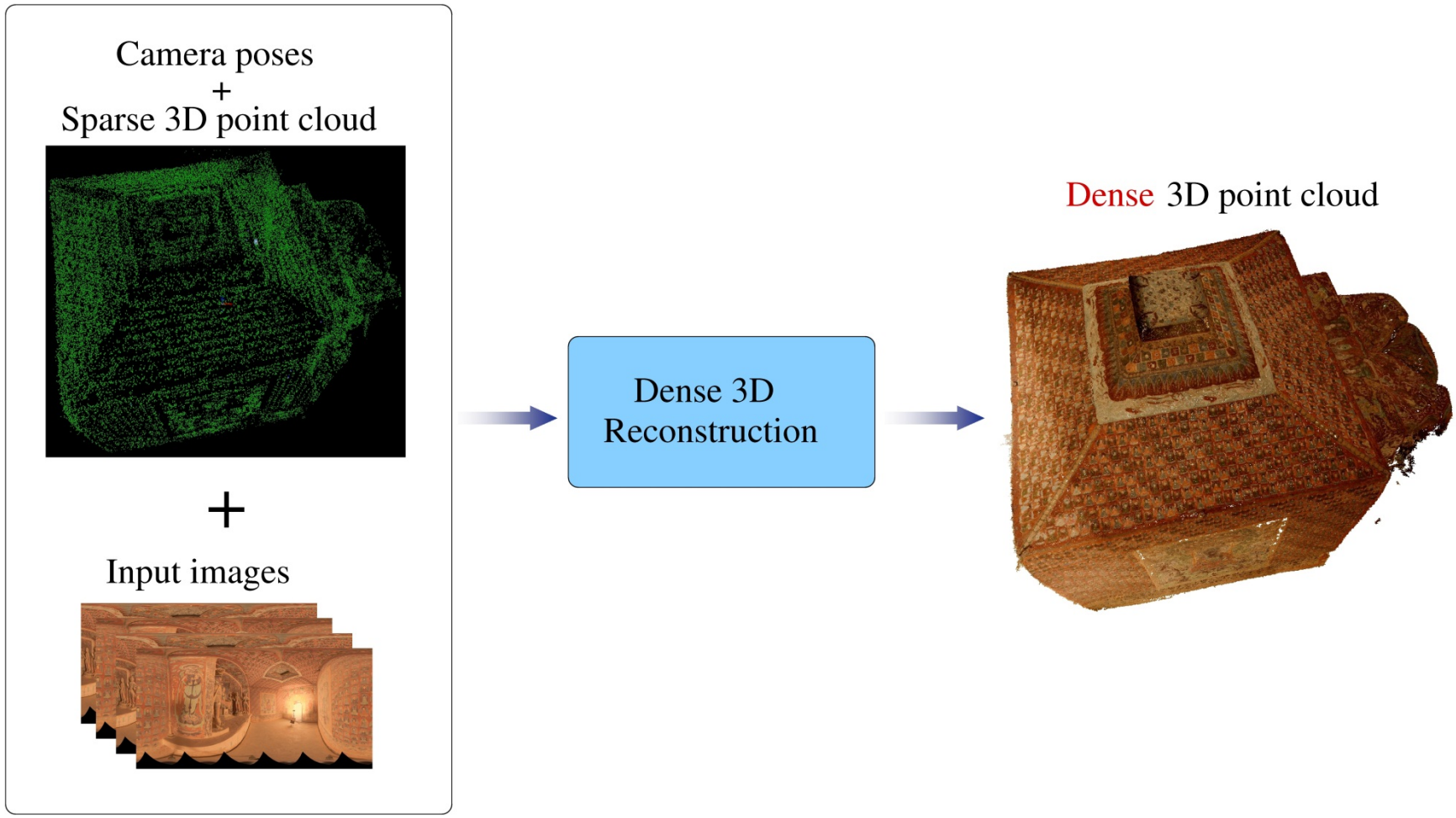
27.707.825 points

Interesting! But how can we go from pictures to 3D models?

Overview I


Input images
Keypoint Extraction (SIFT)
Robust Matching (RANSAC)
Pairwise 2D matches
Pairwise 2D matches
Structure and Motion
Camera poses
+
Sparse 3D point cloud

Overview II


Camera poses
+
Sparse 3D point cloud
+
Input images
Dense 3D Reconstruction
Dense 3D point cloud

Dense 3D reconstruction

- When we take a picture, a 3D scene is projected onto a 2D image → loss of depth information
- 3D reconstruction is the inverse process: build the 3D scene from a set of 2D images → recover depth information

How can we do that?

MATCHING (difficult) + TRIANGULATION (previous lecture)

Two Subproblems

In general, 3D reconstruction can be divided into two major problems:

- Matching
  - finding corresponding elements among the images
- Triangulation
  - establishing 3D coordinates based on the 2D image correspondences found during matching

Which image entities should we match?

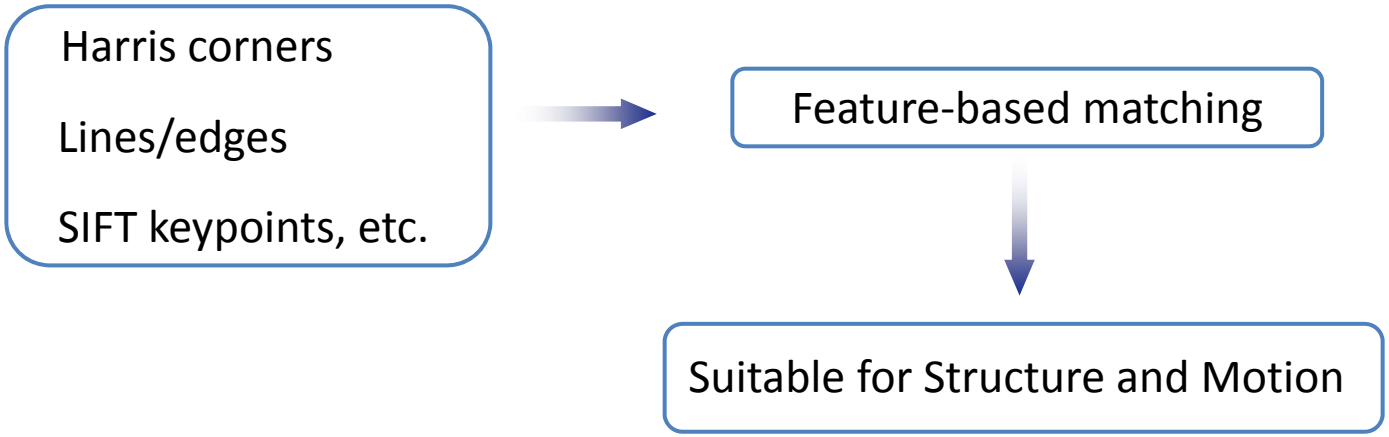
Outline

- Previous lecture: structure and motion II
  - Structure and motion loop
  - Triangulation
  - Wide baseline matching (SIFT)

- Today: dense 3D reconstruction
  - The matching problem
  - 2-view reconstruction (disparity maps)
  - Multi-view reconstruction (point clouds)

- Next lecture: structured light

Matching



But we are interested in dense reconstruction

we need dense matching!

match as many pixels as possible

Dense matching challenges

Scene elements do not always look the same in the images

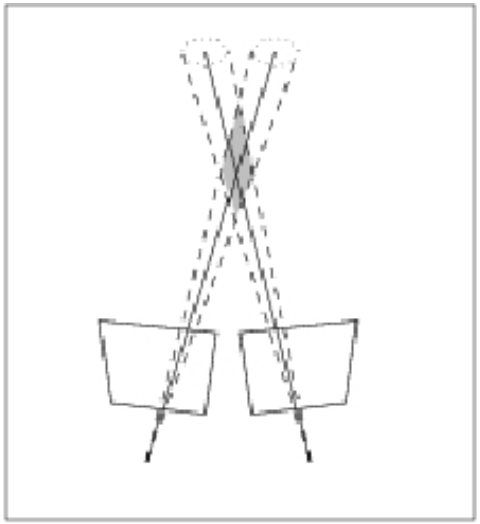
- Camera-related problems
  - Image noise
  - Lens distortion
  - Color/chromatic aberration
- Viewpoint-related problems
  - Perspective distortions
  - Occlusions
  - Specular reflections
- Scene-related problems
  - Illumination changes
  - Moving objects

Dense matching challenges

- Choice of camera setup
  - Small baseline **X** wide baseline

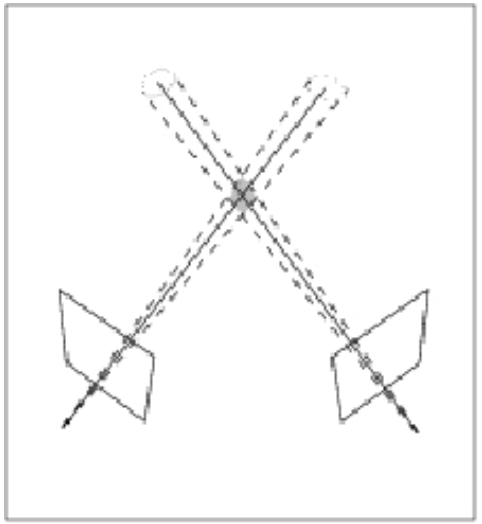
Matching is easy

Higher depth uncertainty



Matching is hard

Lower depth uncertainty

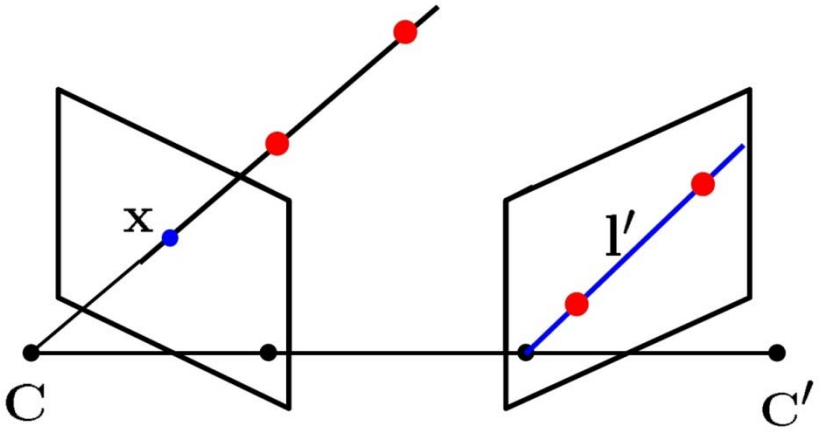


Besides that…

- For every point in an image, there are many possible matches in the other image(s)
- Locally many points look similar → high ambiguity

What can we do?

- We can use camera geometry → finding the correspondence becomes a 1D search problem thanks to the epipolar geometry

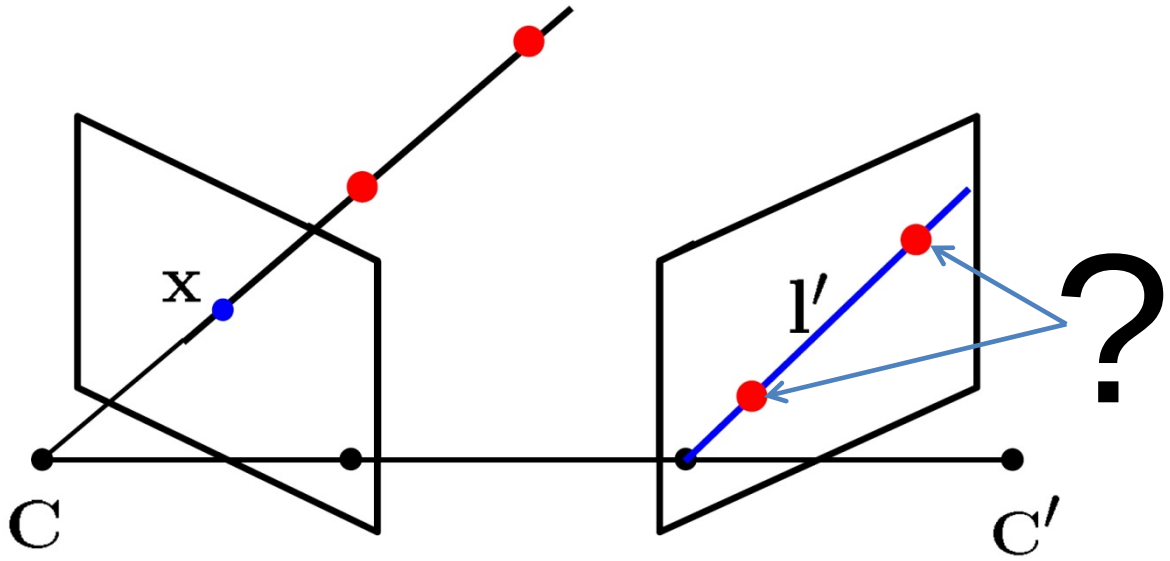


$$\mathbf{l}' = \mathbf{F}\mathbf{x} \qquad \mathbf{F} = \mathbf{K}'^{-\top}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{K}^{-1}$$

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$

The matching problem

- Even using the epipolar constraint, there are many possible matches



- Use of pixel neighborhood information → correlation-based approaches (also called window-based approaches)

Common window-based approaches

- Normalized Cross-Correlation

$$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$$

- Sum of Squared Differences

$$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i,y+j)\right)^2$$

- Sum of Absolute Differences

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i,y+j)|$$

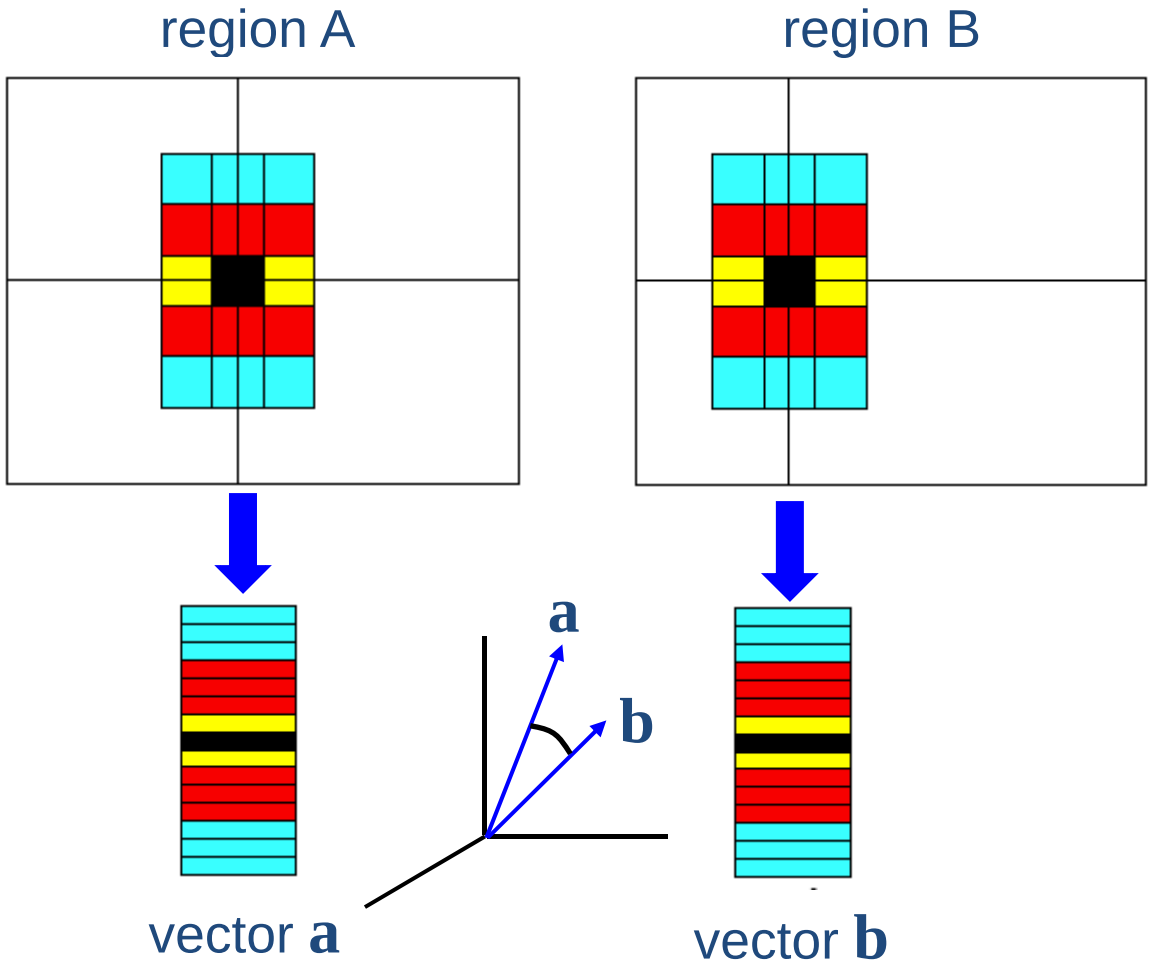
Recall NCC

write regions as vectors

$$A \rightarrow \mathbf{a}, \quad B \rightarrow \mathbf{b}$$

$$NCC = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

$$-1 \leq NCC \leq 1$$



The matching problem

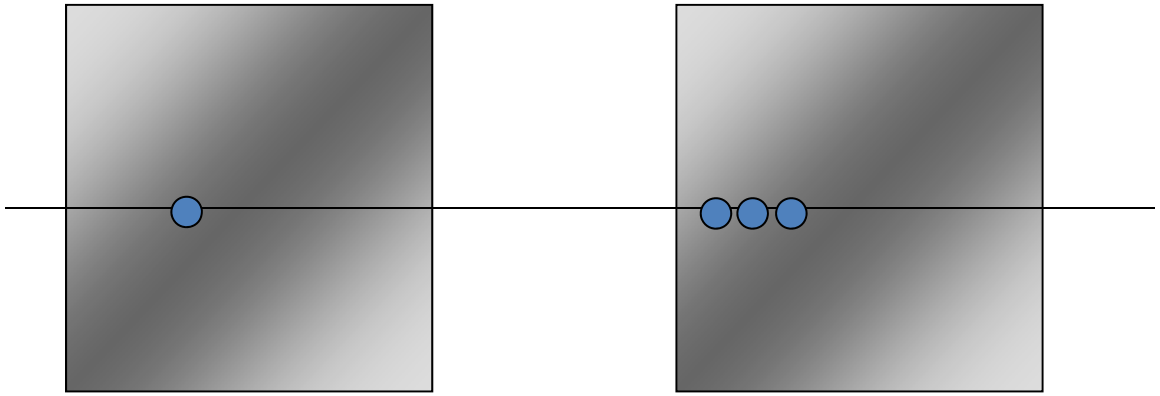
- Search along the epipolar line for the best match

Best match:

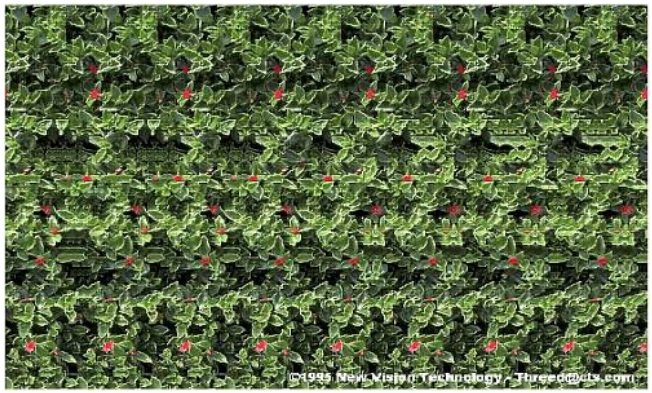
- Maximum NCC
- Minimum SSD or SAD

The matching problem

- Still, matching can be ambiguous!
  - Low textured regions



  - Repetitive texture pattern



The matching problem

Therefore some assumptions about the scene are made

- Appearance
  - The projections of a scene (3D) patch should have similar appearances in all images
- Uniqueness
  - A point in an image will have only one match in another image
- Ordering
  - Order of appearance (e.g. left to right) is not altered by camera displacement
- Smoothness
  - Depth varies smoothly (objects have smooth surfaces)

Outline

- Previous lecture: structure and motion II
  - Structure and motion loop
  - Triangulation
  - Wide baseline matching (SIFT)
- Today: dense 3D reconstruction
  - The matching problem
  - 2-view reconstruction (disparity maps)
  - Multi-view reconstruction (point clouds)
- Next lecture: structured light

2-view reconstruction

- Input: a pair of images (usually left and right)
- Output: a disparity or depth map

left image

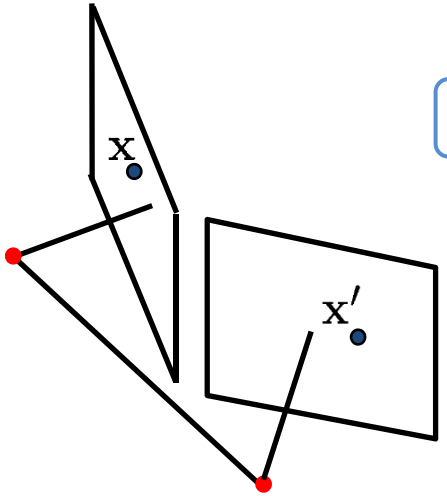
right image

depth map

$$disparity \propto \frac{1}{depth}$$

2-view reconstruction

- General case: converging cameras



Need to use epipolar geometry

- This geometry can be simplified by rectification
  - One of the images
  - Both images

Image rectification

- Rectifying one image

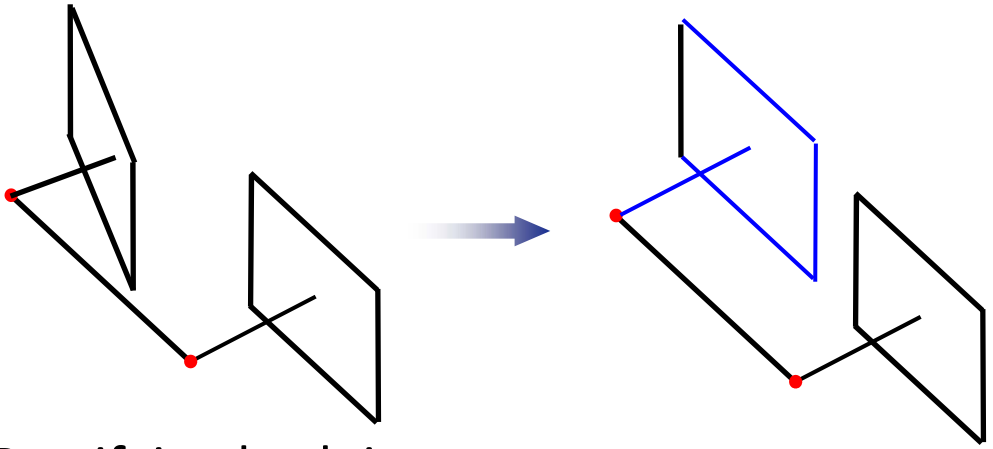
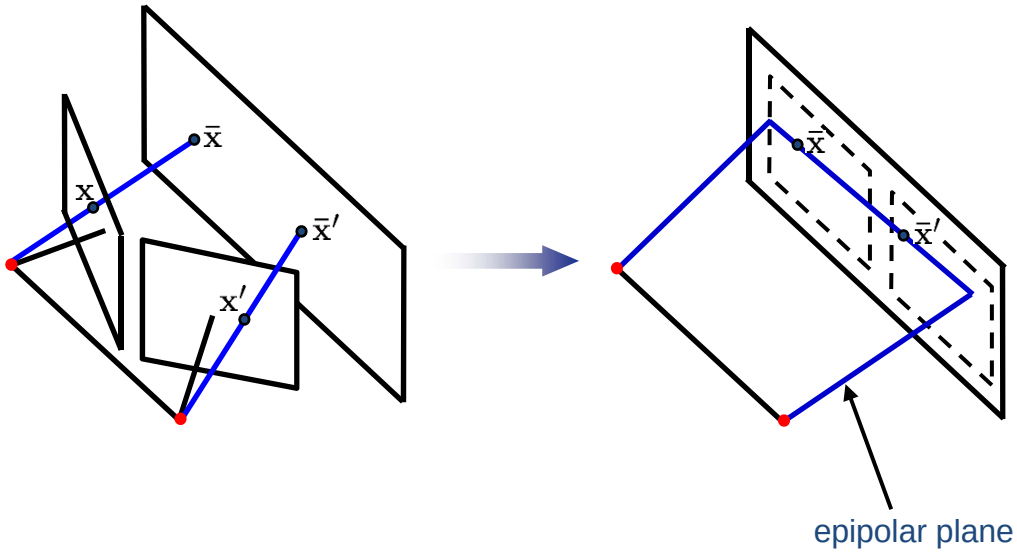


Image mapping is a 2D homography
(projective transformation)

$$\mathrm{H} = \mathrm{KRK}^{-1}$$

- Rectifying both images



Project images onto plane
parallel to baseline

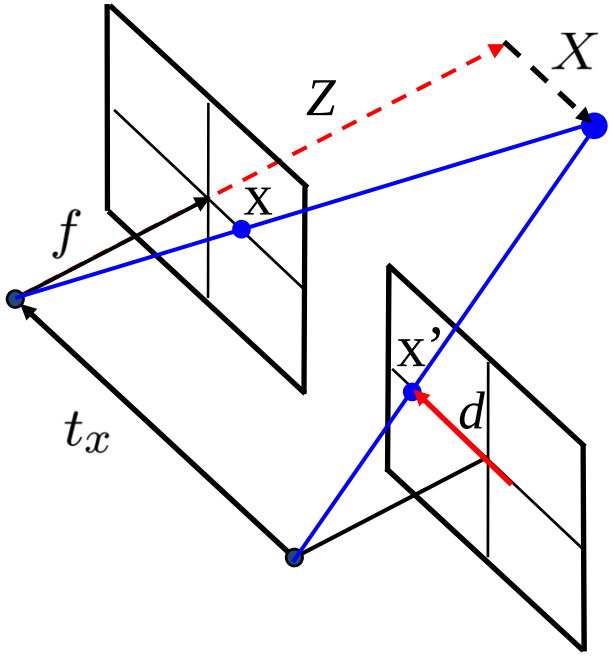
Example of rectifying both images

Parallel cameras

- Triangulation reduces to a simple geometric problem

$$\mathtt{K} = \mathtt{K}' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathtt{R} = \mathtt{I} \qquad \mathbf{t} = \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix}$$

$$Z = \frac{f t_x}{d}, d = x' - x$$



General dense correspondence algorithm

For each pixel in the left image

- compute the neighbourhood cross correlation along the corresponding epipolar line in the right image
- the corresponding pixel is the one with the highest cross correlation

Parameters

- size (scale) of neighbourhood
- search disparity

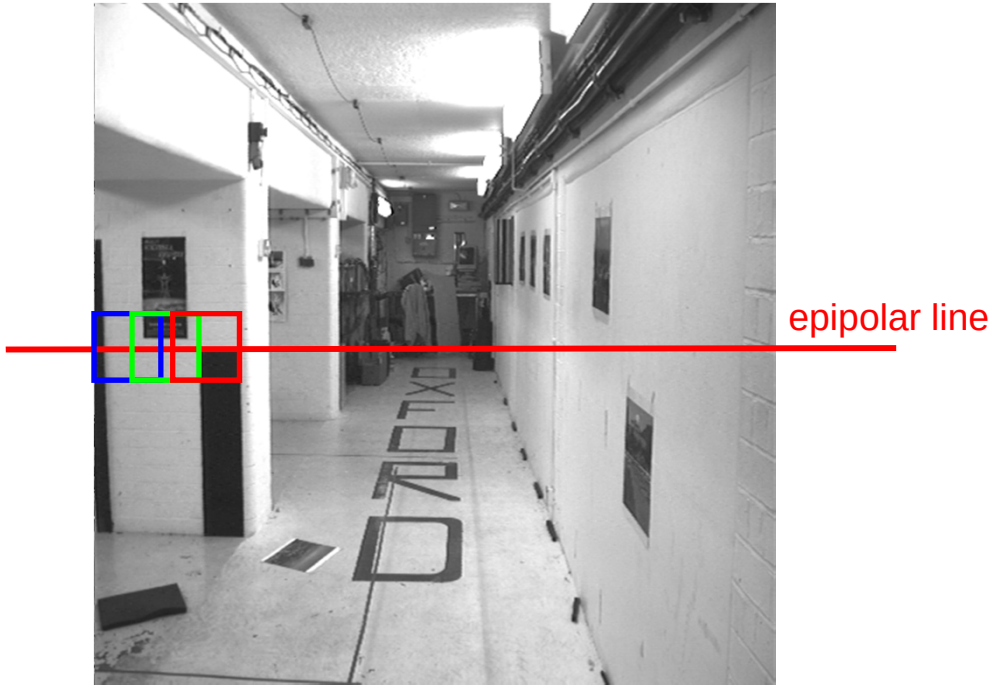
Other constraints

- uniqueness
- ordering
- smoothness of depth field

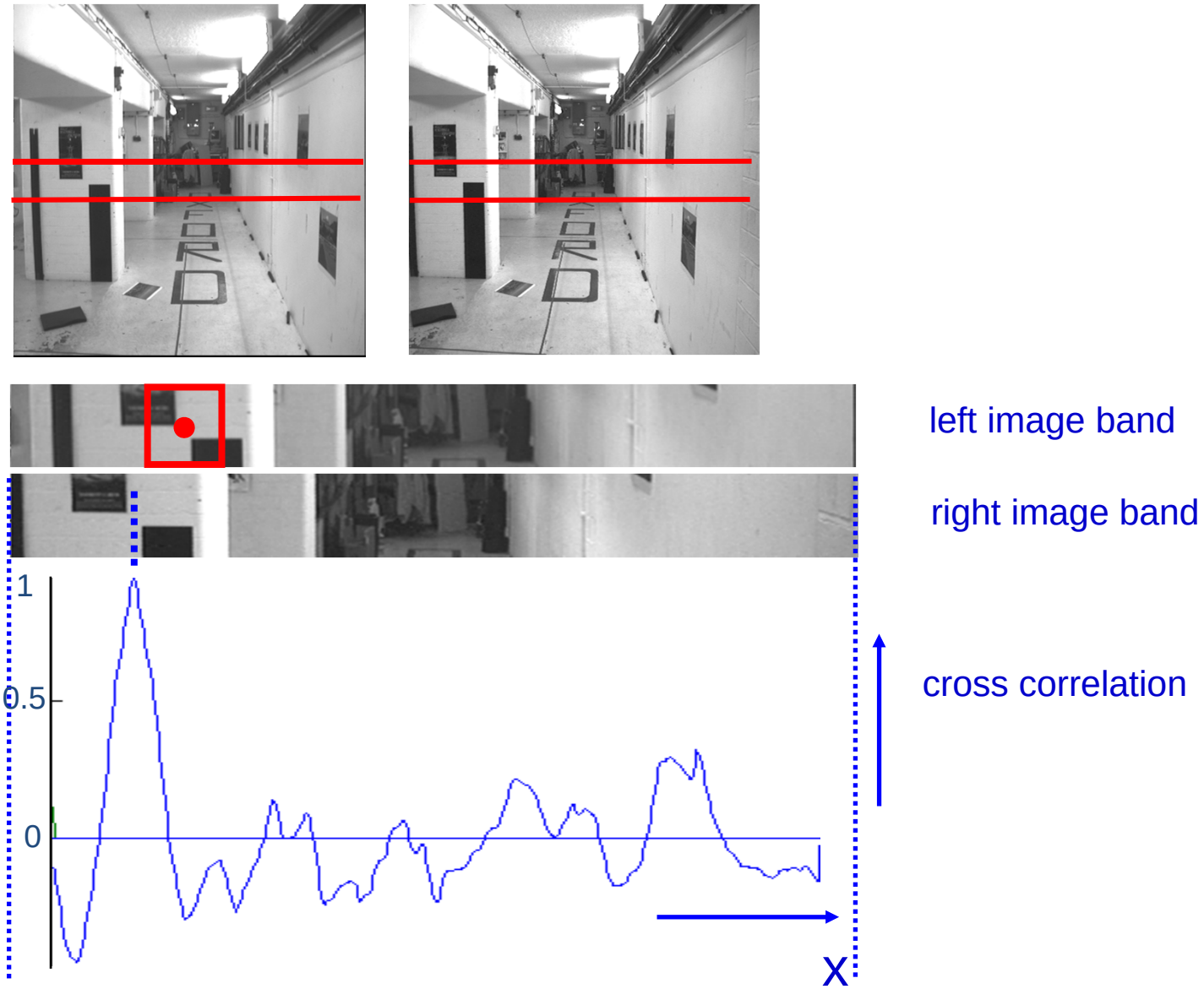
Applicability

- textured scene, largely fronto-parallel

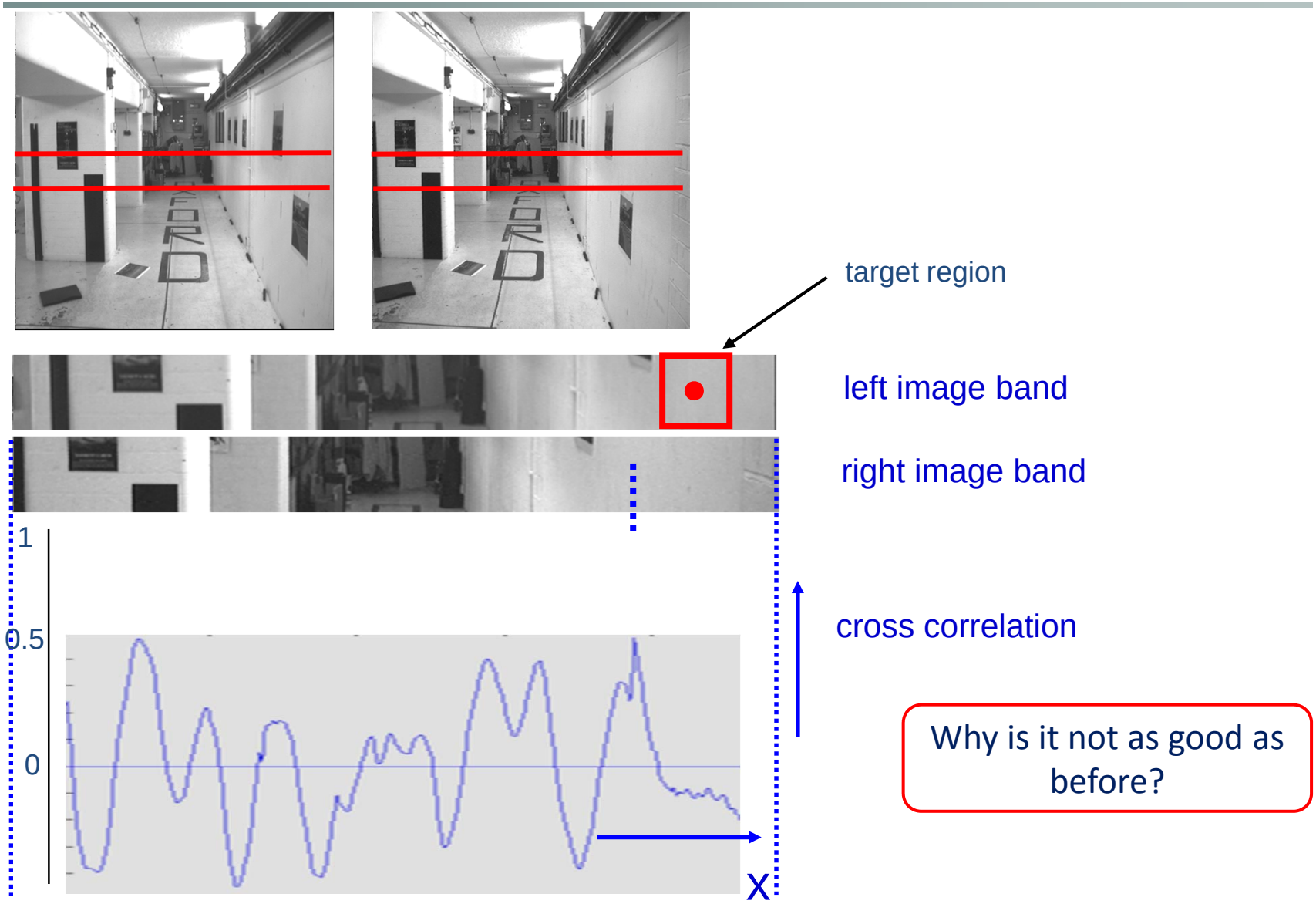
Example with NCC



Example with NCC



Example with NCC

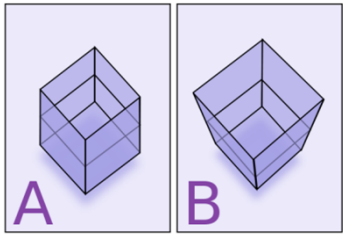


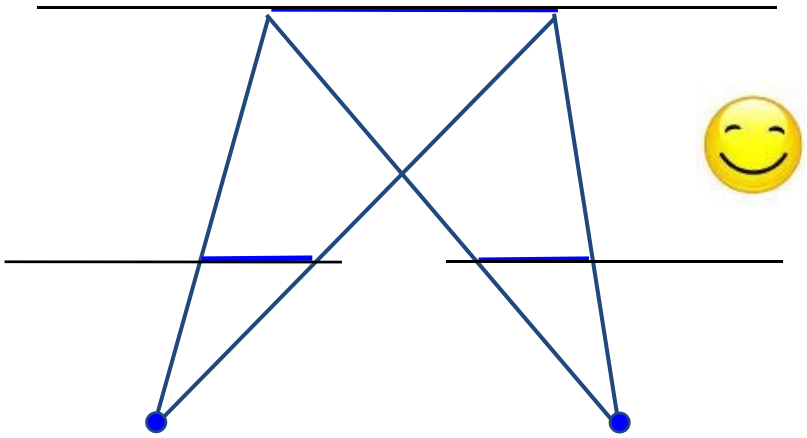
Why is it not as good as before?

2-view reconstruction

1.The neighbourhood region does not have a “distinctive” spatial intensity distribution

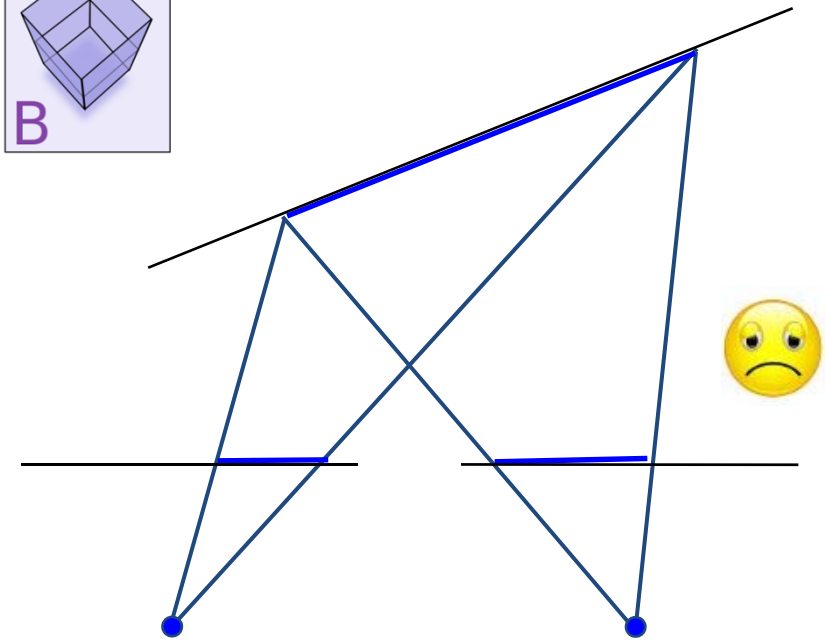
2.Foreshortening effects





fronto-parallel surface

imaged length the same

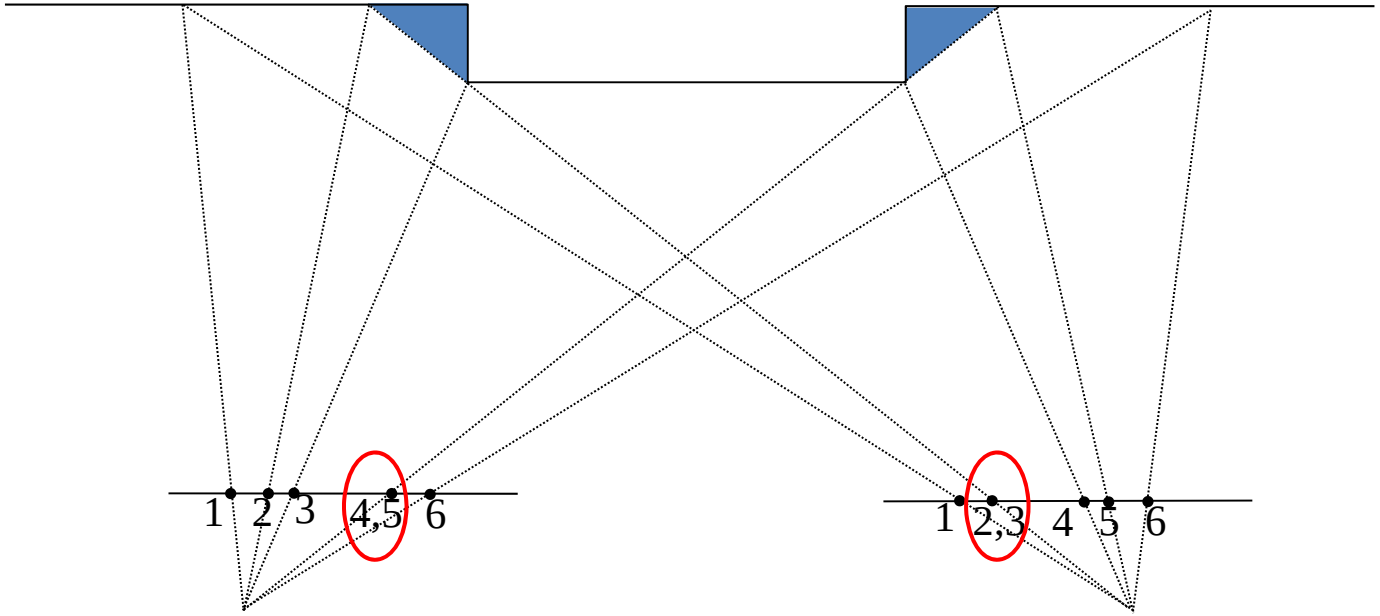


slanting surface

imaged lengths differ

Occlusion

surface slice



Outline

- Previous lecture: structure and motion II
  - Structure and motion loop
  - Triangulation
  - Wide baseline matching (SIFT)

- Today: dense 3D reconstruction
  - The matching problem
  - 2-view reconstruction (disparity maps)
  - Multi-view reconstruction (point clouds)

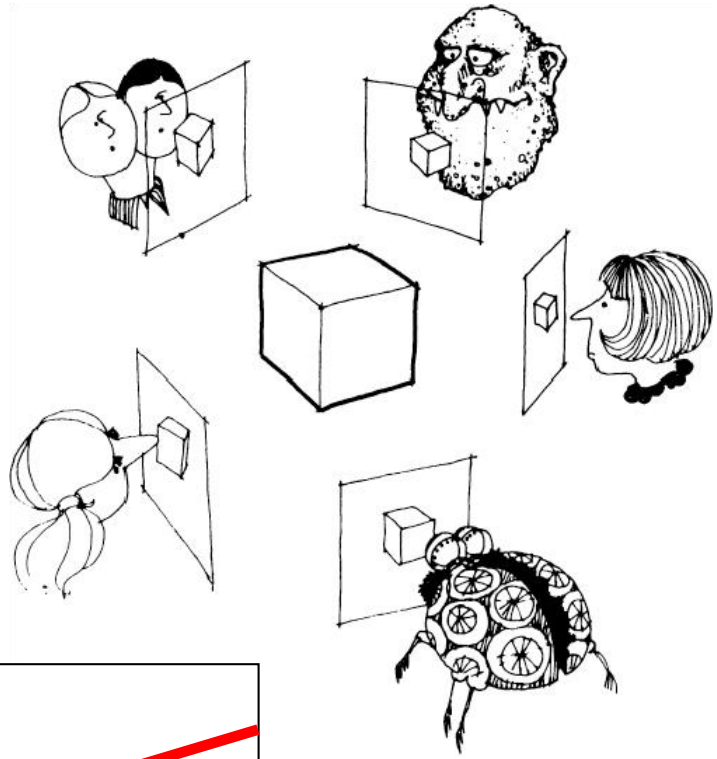
- Next lecture: structured light

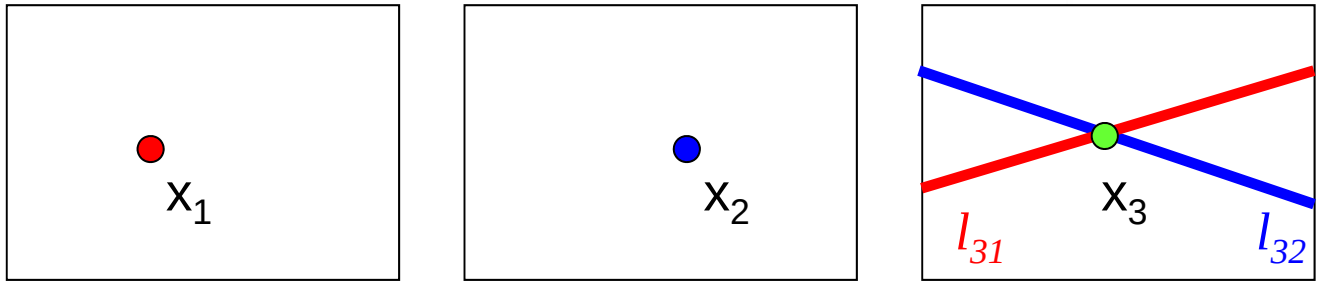
Multi-view stereo

Many views of the same scene

More images

More constraints!





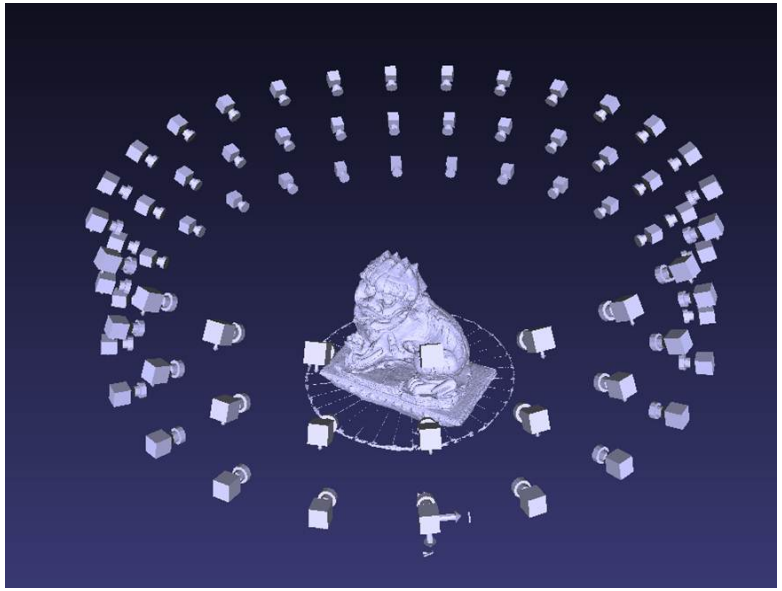
$$l_{31} = F^T{}_{13}\, x_1$$
$$l_{32} = F^T{}_{23}\, x_2$$

Multi-view stereo

- More constraints
- Less occlusions
- More robust
- Appearance constraint can be relaxed

However…

- More complex
- Strong perspective distortions
- Propagation of calibration uncertainties

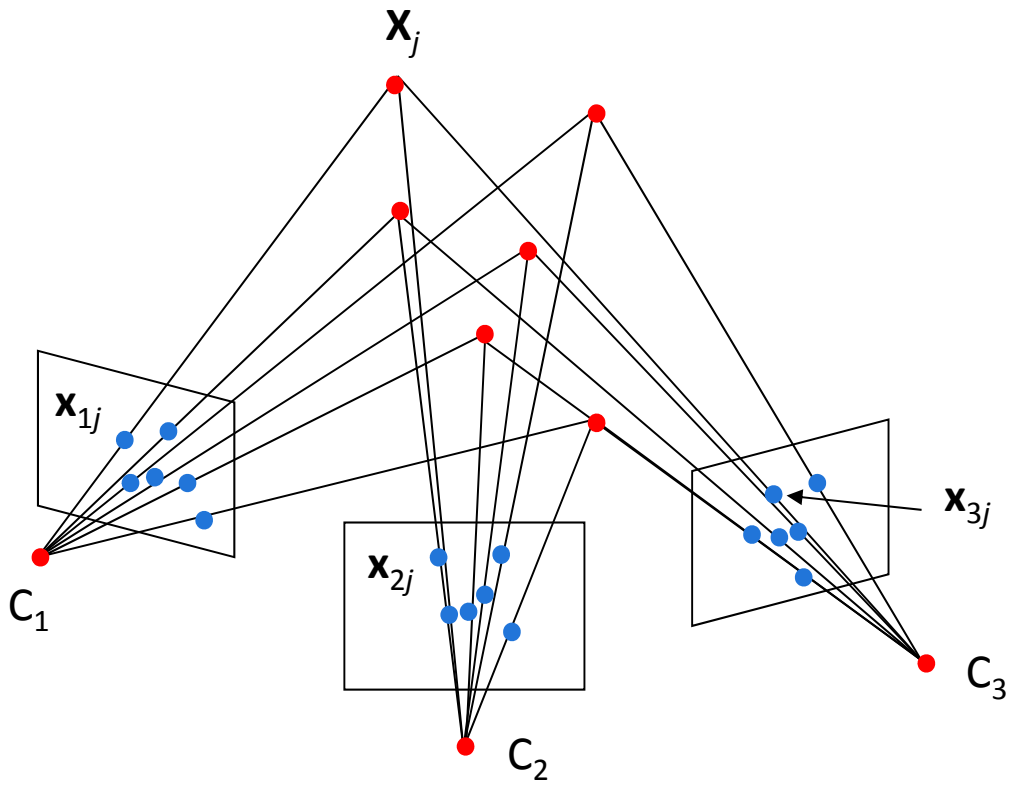


Dense matching along many images is still very hard…

Precise calibration is essential!

Multi-view reconstruction

Assuming we have found correct matches along *n* images, how do we triangulate them?



Exactly as we did before!

Remember the algebraic solution…

Stack equations for all camera views:

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} X = 0$$

The PMVS Algorithm

Overview of the PMVS algorithm

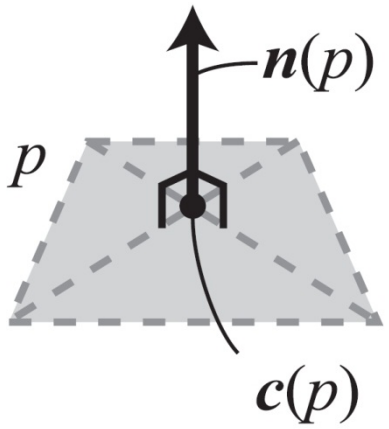
- A complete understanding is out of scope of the lecture
- Next slides give an overview of the algorithm
- The aim is to show one solution (out of many) for the question “how can we go from pictures to 3D models?“

The PMVS algorithm

- PMVS stands for Patch-based Multi-View Stereopsis
- Input: set of accurately calibrated cameras
- Output: dense 3D point cloud (from which a surface can be reconstructed)

The PMVS algorithm

- Surface is locally approximated by a small rectangle → the patch



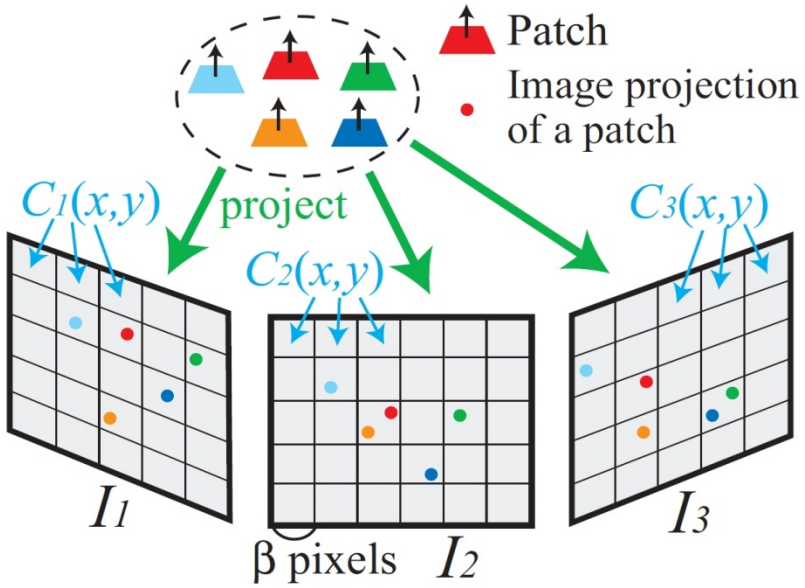
A patch is a rectangle modeled by:

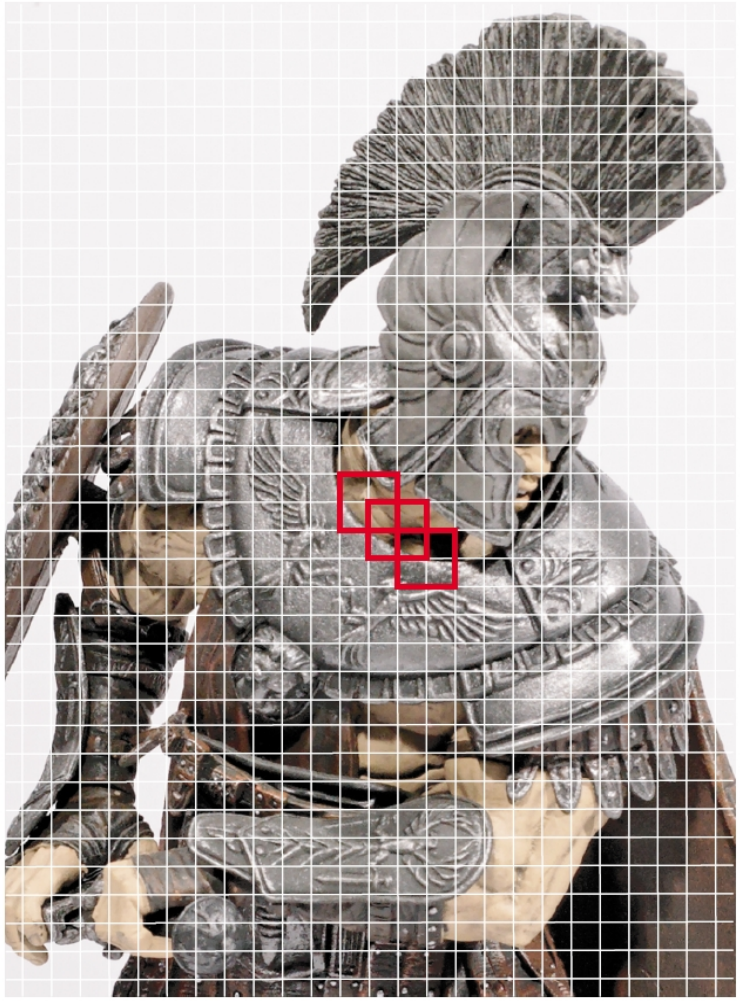
- Its 3D position c(p)
- Its 3D normal vector n(p)
- x and y axes (not shown)
- A reference image R(p)

The reference image is the one with the best view of the patch

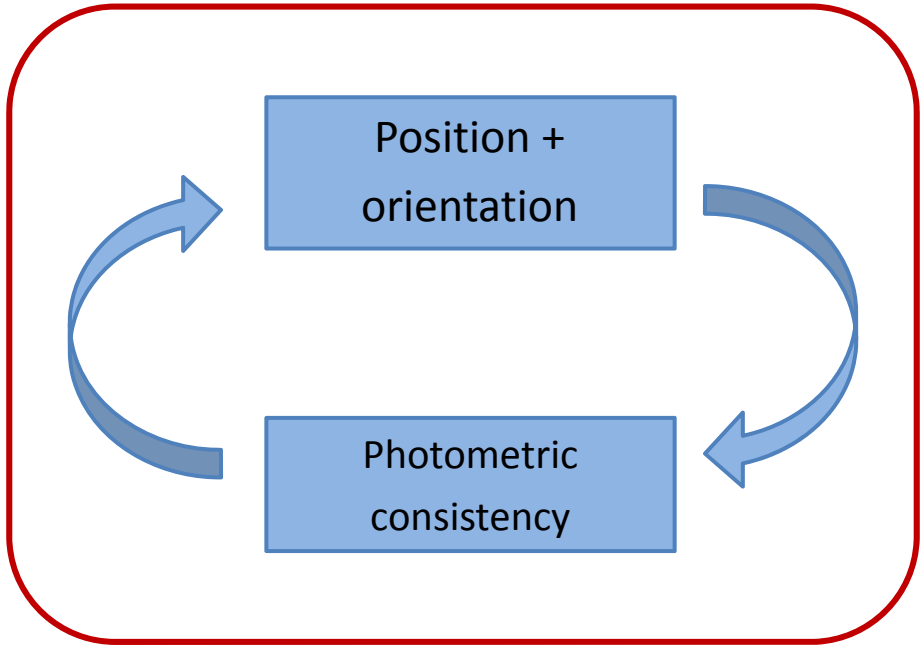
PMVS – image model

- Images are divided into cells of β×β pixels (usually 2)
- Idea is to reconstruct at least 1 patch/cell
- Small cells → high density of final point cloud





Patch refinement

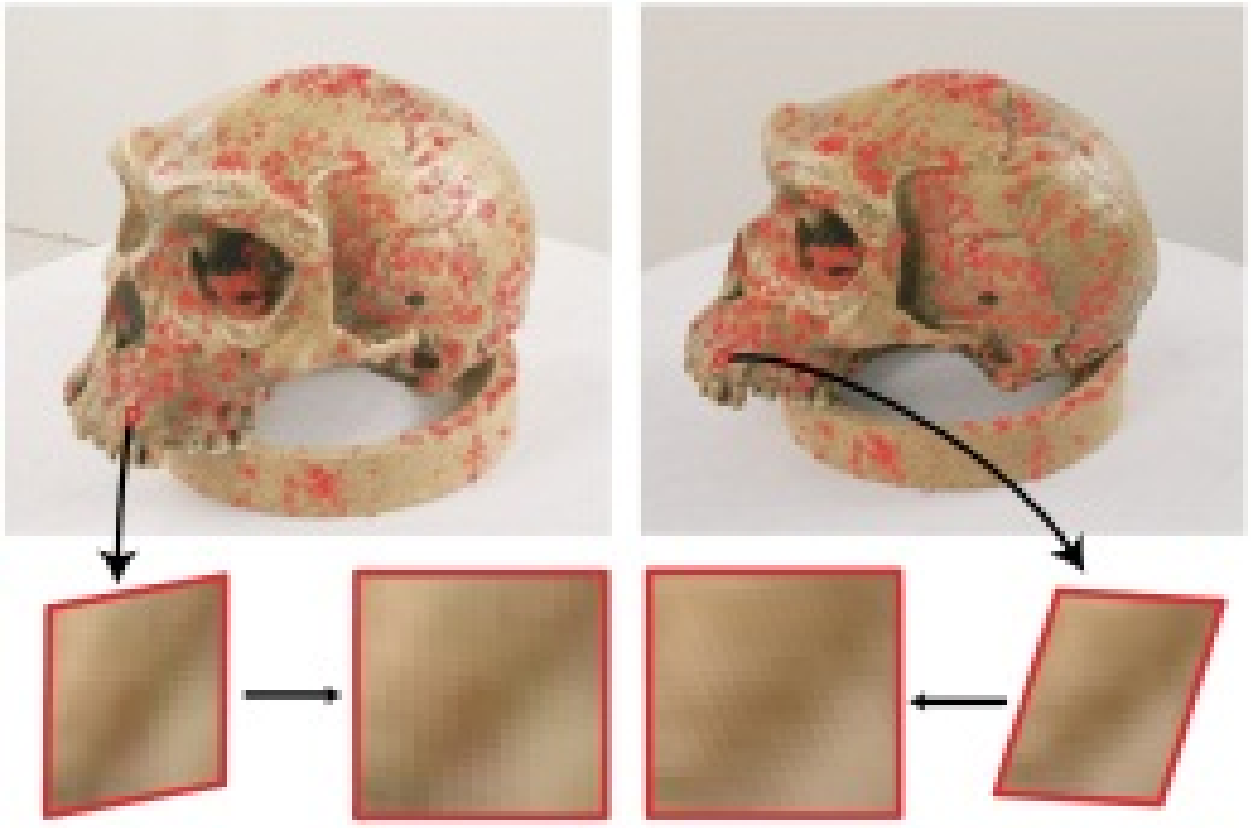


Non-linear Optimization

- Maximizes photometric consistency
- Uses NCC

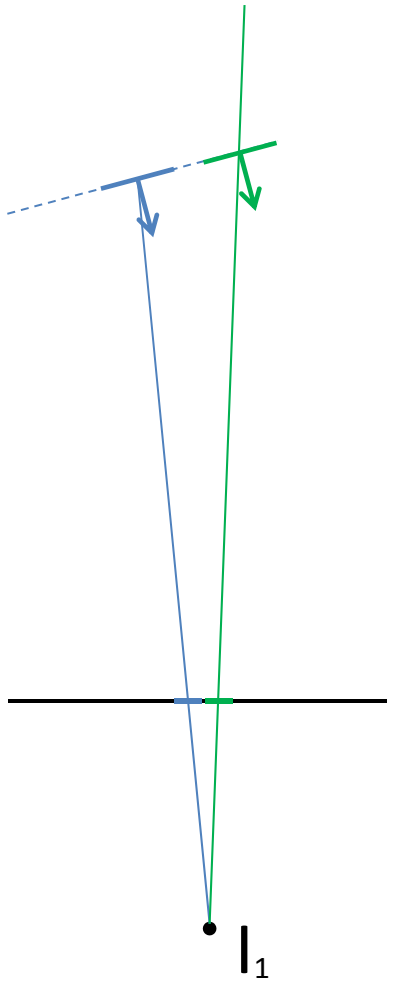
PMVS – only affine distortions

- Simplification: considers only affine distortions
- Samples patch projections and then compute NCC



PMVS - expansion

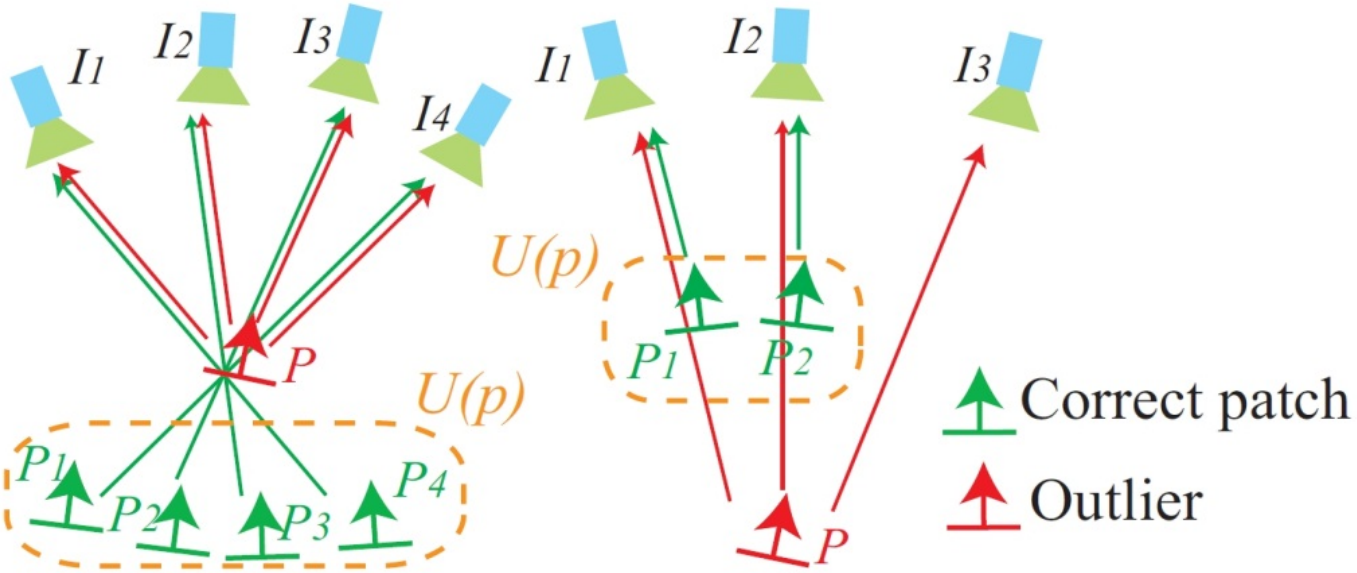
- Reconstructed (3D) points from the initialization serve as “seeds” for an expansion algorithm (region growing)
- A seed patch is expanded in the following way:
  - The new patch will project onto a neighboring cell
  - Position is set to the intersection of the back-projected ray and the plane of its parent patch
  - Same orientation (normal vector is propagated)
  - Same reference image



- Optimize position and orientation of new patch

PMVS - filtering

- Enforces global visibility consistency

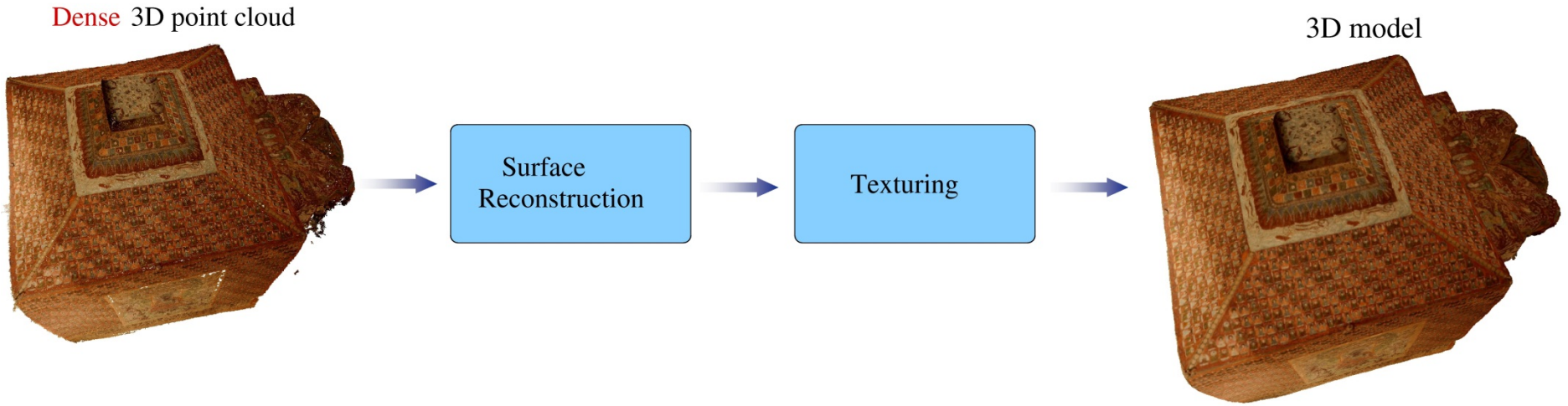


PMVS – loop expansion and filtering

- The algorithm iterates between expansion and filtering until the point cloud is dense enough (3 times are sufficient)

Final remarks

- The point cloud is not the last step in a complete dense 3D reconstruction pipeline…



Interesting links

- Middleburry stereo data set: http://vision.middlebury.edu/stereo/data/
- Middlebury multi-view data set: http://vision.middlebury.edu/mview/
- PMVS – version 2: http://grail.cs.washington.edu/software/pmvs/
- CVLab multi-view data sets:
  - http://cvlab.epfl.ch/data/strechamvs/
  - http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html

References

- Furukawa, Y. & Ponce, J. **Accurate, Dense and Robust Multi-View Stereopsis**, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 01, 1-14

- Furukawa, Y.; Curless, B.; Seitz, S. M. & Szeliski, R. **Towards Internet-Scale Multi-View Stereo**, CVPR, 2010

- Hartley, R. I. & Zisserman, A. **Multiple View Geometry in Computer Vision**, Cambridge University Press, ISBN: 0521540518, 2004

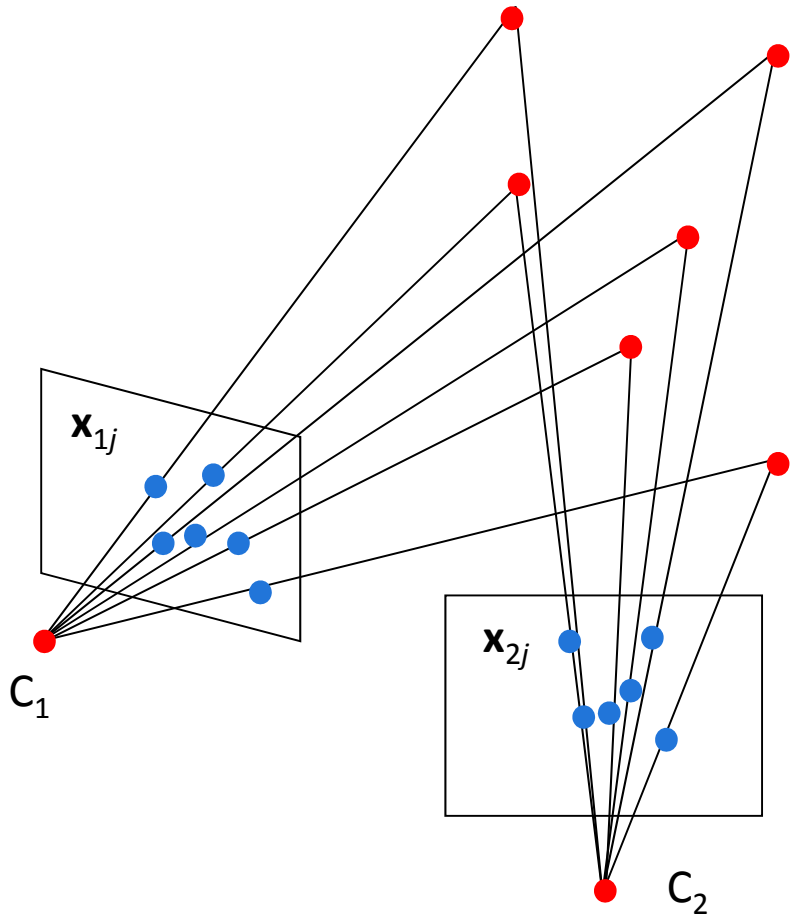
Videos

Thank you!

JUST A REMINDER

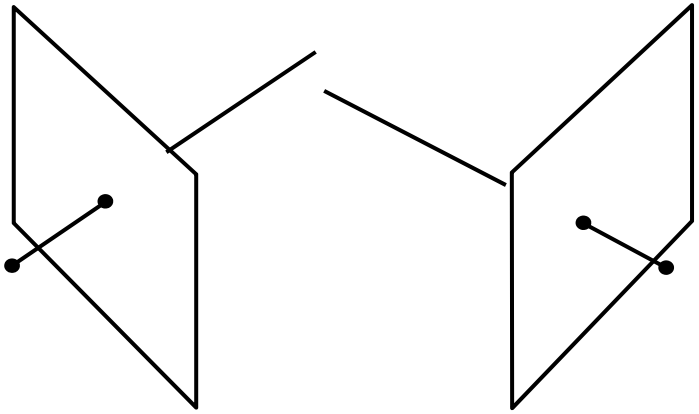
Triangulation

- If images are not rectified $\mathbf{x}_j$



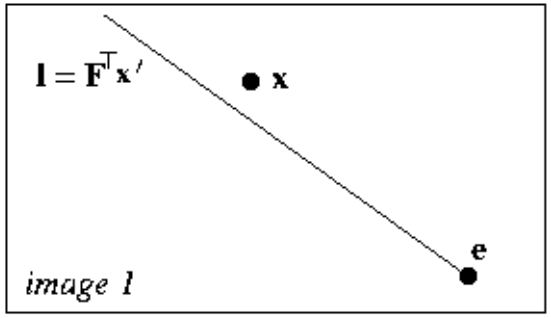
Triangulation

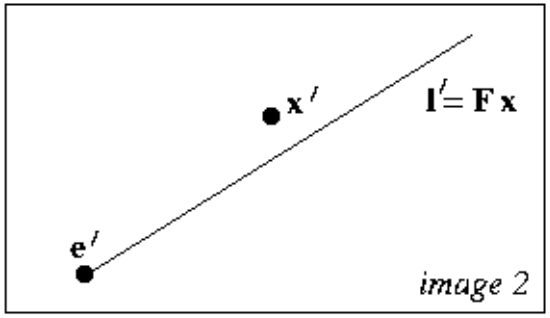
- Given: corresponding measured (i.e. noisy) feature points x and x’ and camera poses, P and P’
- Problem: in the presence of noise, back projected rays do not intersect



Rays are skew in space

and measured points do not lie on corresponding epipolar lines





Triangulation: algebraic solution

- Equation for one camera view (camera pose P and feature point x)

4 entries, 3 DOF

i[th] row of P

$$x \propto PX \Leftrightarrow \lambda x = PX \Leftrightarrow x \times \begin{bmatrix} p^{1T} \\ p^{2T} \\ p^{3T} \end{bmatrix} X = 0$$

(3x4) matrix

$$x(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{1\top}\mathbf{X}) = 0$$
$$y(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{2\top}\mathbf{X}) = 0$$
$$\cancel{x(\mathbf{p}^{2\top}\mathbf{X}) - y(\mathbf{p}^{1\top}\mathbf{X}) = 0}$$

$$\overbrace{\begin{bmatrix} x\mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} - \mathbf{p}^{2\top} \end{bmatrix}}^{A} \mathbf{X} = \mathbf{0}$$

3 equations, only 2 linearly independent
⇨drop third row

Triangulation: algebraic solution

- Stack equations for all camera views:

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} X = 0$$

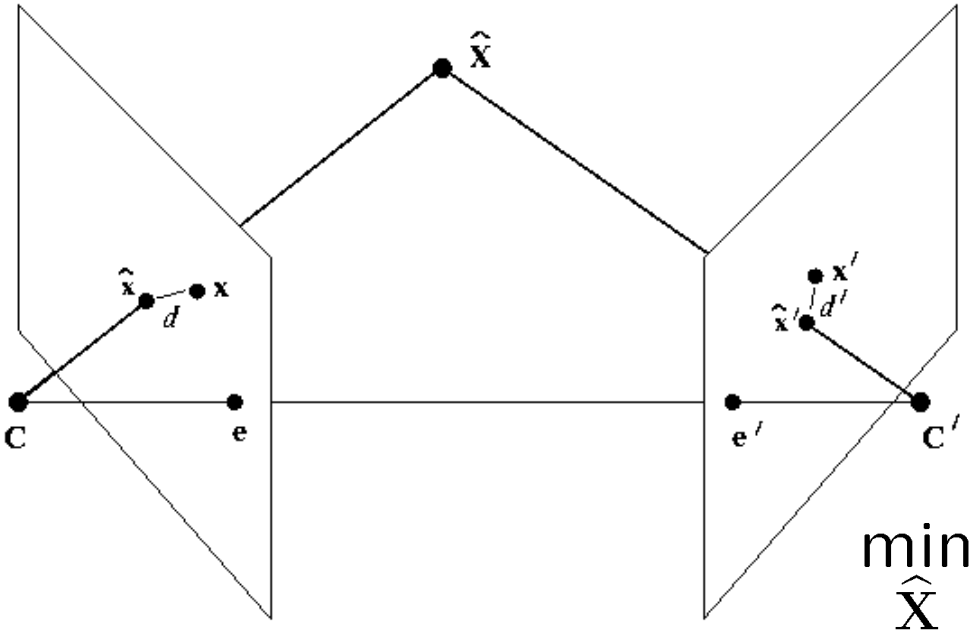
- X has 3 DOF (4 parameters due to homogeneous representation)
- One camera view gives two linearly independent equations
- Two camera views are sufficient for a minimal solution
- Solution for X is eigenvector of A^TA corresponding to smallest eigenvalue

Triangulation: minimize geometric error

- Estimate 3D point $\hat{X}$, which exactly satisfies the supplied camera geometry P and P', so it projects as

$$\hat{x} \propto P\hat{X} \qquad \hat{x}' \propto P'\hat{X}$$

where $\hat{x}$ and $\hat{x}'$ are closest to the actual image measurements



Assumes perfect camera poses!

$$\min_{\hat{\mathbf{X}}} \quad \mathcal{C}(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$$