MAEG 5720: Computer Vision in Practice

Lecture 13:

Reconstruction

Dr. Terry Chang

2021-2022

Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

Summary of Fundamental Matrix

- Fundamental Matrix $\mathrm{F}=(\mathrm{K}')^{-T} R'\ S_b\ (R'')^T (\mathrm{K}'')^{-1}$
- F is Singular (det(F)=0)
- F consist of the *Relative Orientation* of image pair from uncalibrated cameras
- F has 7 DoF
- Given the projection matrices P'=[A'|a'] and P'' =[A''|a''] of the two cameras, F can be computed directly

$$\mathrm{F} = \mathrm{A}'^{-\mathrm{T}} \mathrm{S}_{b_{12}} \mathrm{A}''^{-1}$$

- Coplanarity Constraint

$$\mathbf{x}'^{T} \mathrm{F}\, \mathbf{x}'' = 0$$

Summary of Essential Matrix

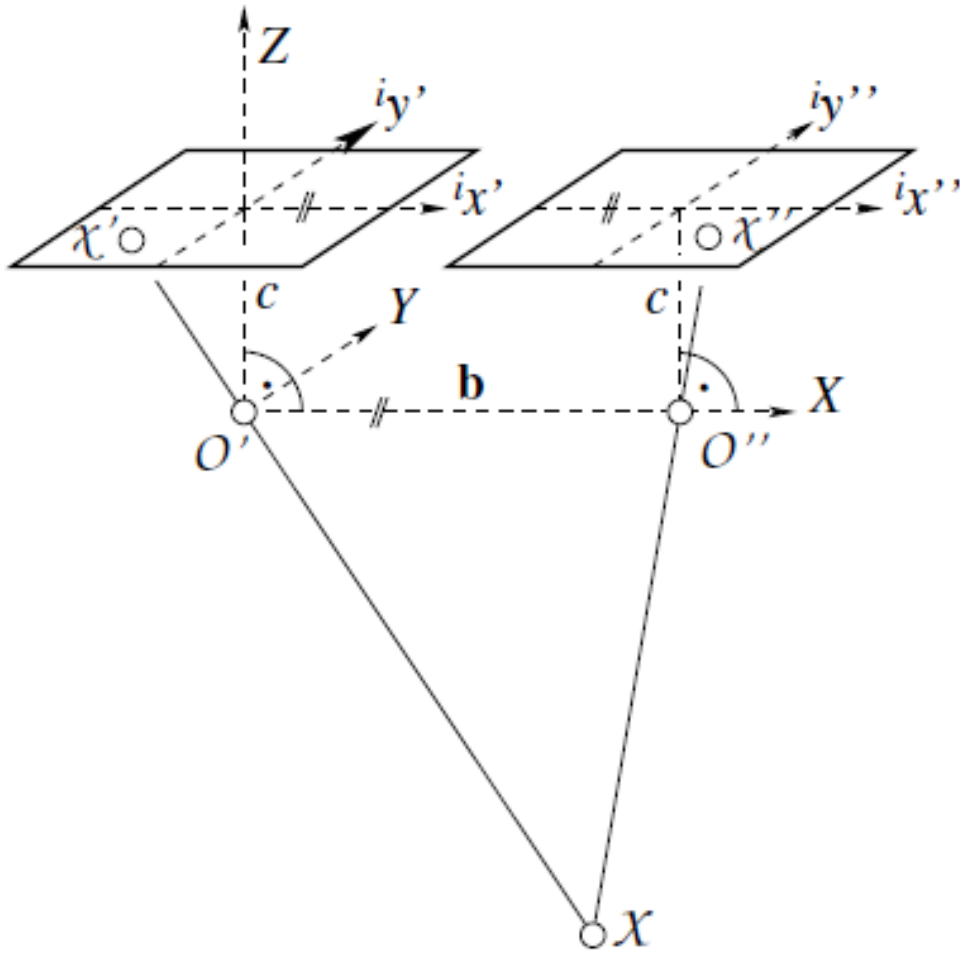
- The essential matrix $\mathrm{E} = R'\ S_b\ (R'')^T$ express the coplanarity constraint of two calibrated camera
- $ {}^{\mathrm{c}}\mathrm{x}'^{T}\,\mathrm{E}\ {}^{\mathrm{c}}\mathrm{x}'' = 0$
- E express the R.O. $\mathrm{E} = S_b\ R^T$
- E has 5-DoF

Today’s Agenda

- Stereo Normal View
- Disparity Calculation
- Stereo Normal View Reconstruction
- Triangulation of calibrated camera

The Normal Case of Image Pair

- Close to configuration of human stereo vision system
- More computational efficient
- Both Cameras are *ideal* and *identical*
- The viewing direction are *parallel* and *orthogonal* to the base vector
- x' and x'' axis are parallel to the basis

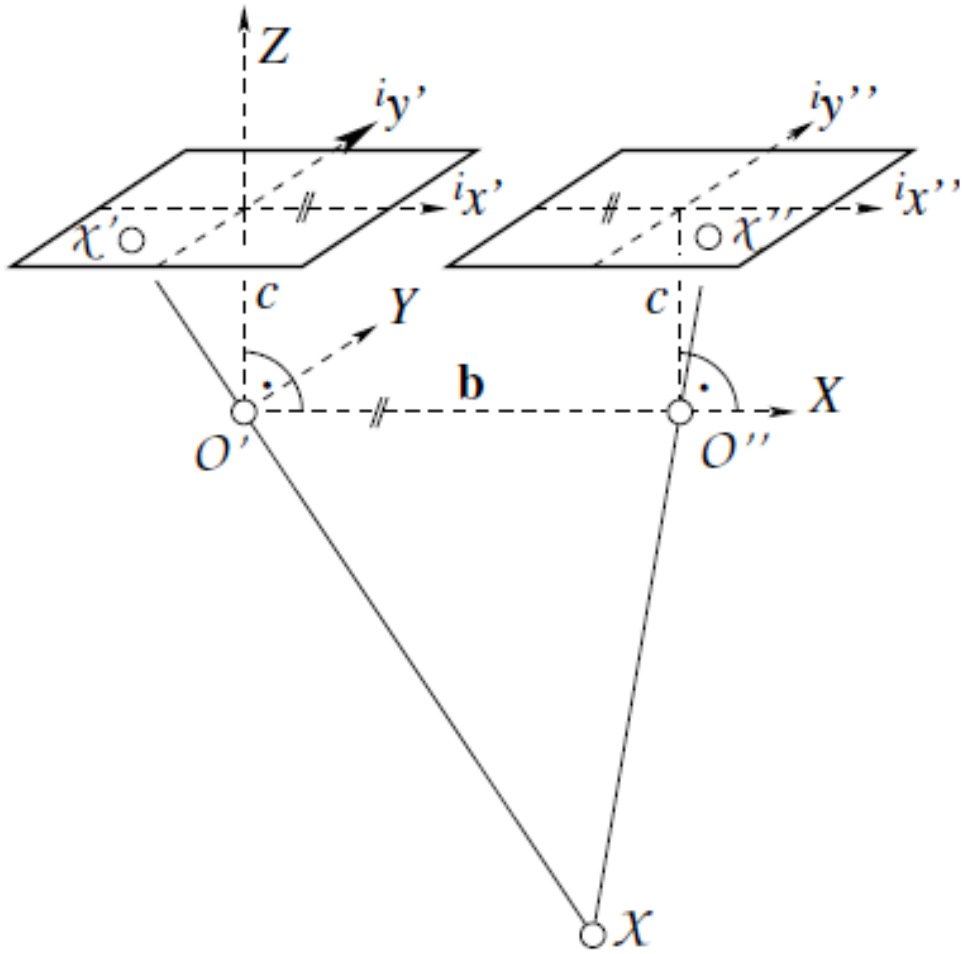


The Normal Case of Image Pair

- Assume error-free image coordinate X' and X'' on both images.

- The constraints for rays $O'X'$ and $O''X''$ intercept at X in the scene is y-parallax

$$p_y = {}^{i}y'' - {}^{i}y' = 0$$



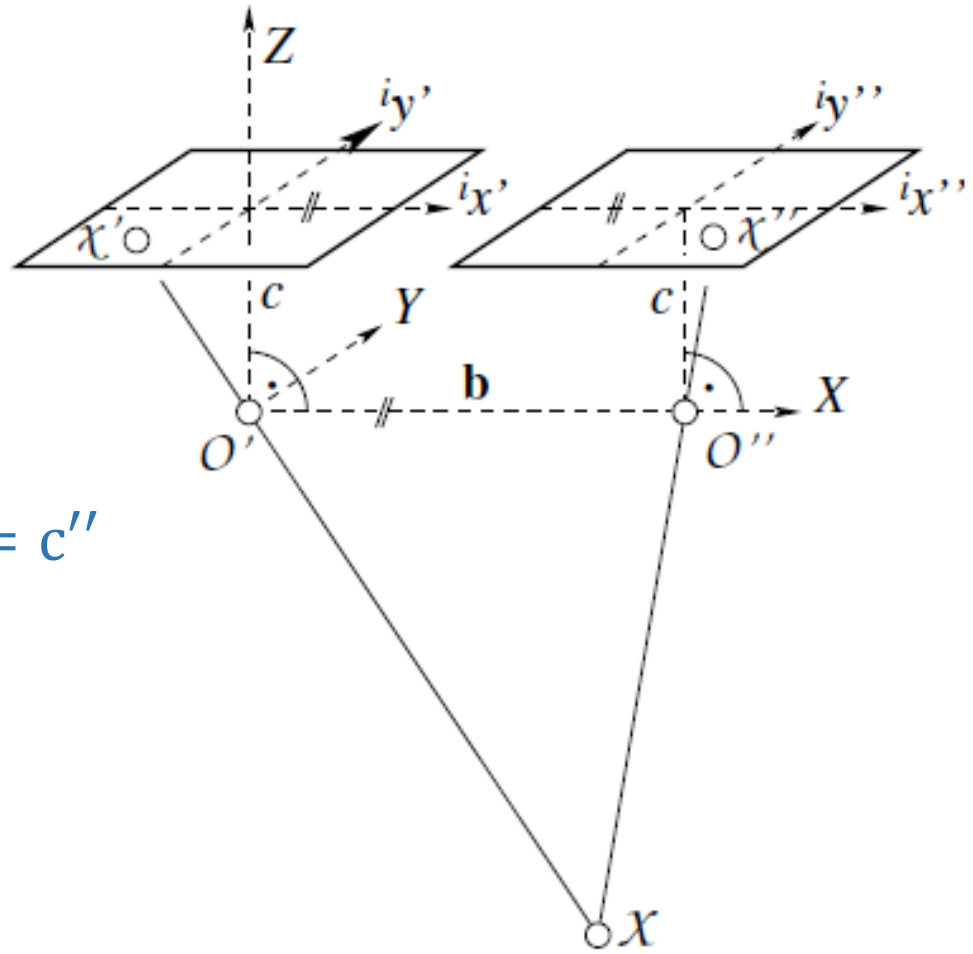
The Normal Case of Image Pair

- The special configuration is

$$R' = R'' = I_3$$

$$K = K' = K'' = \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ with } c = c' = c''$$

$$\mathbf{b} = \begin{bmatrix} B_x \\ 0 \\ 0 \end{bmatrix}$$



The Normal Case of Image Pair

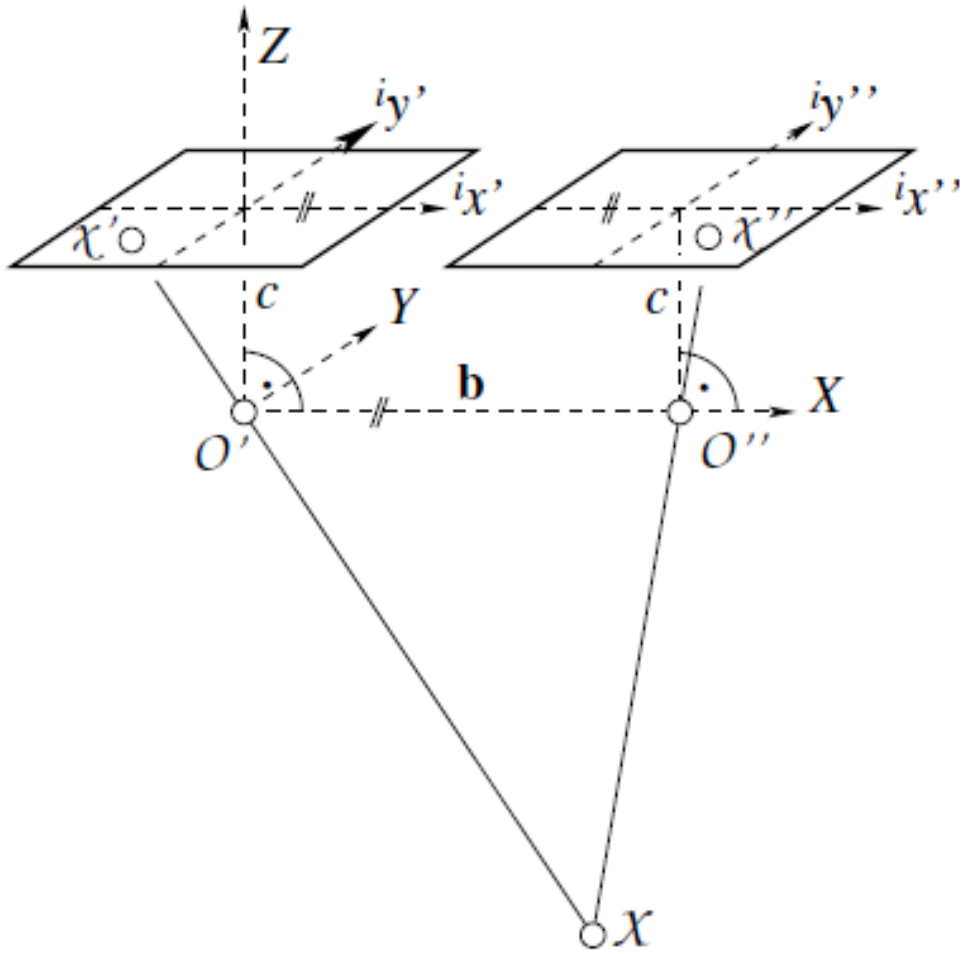
- For Stereo normal case, essential matrix E

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -B_X \\ 0 & B_X & 0 \end{bmatrix}$$

- The coplanarity constraint:

$$\begin{bmatrix} {}^{i}x' & {}^{i}y' & c \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -B_X \\ 0 & B_X & 0 \end{bmatrix} \begin{bmatrix} {}^{i}x'' \\ {}^{i}y'' \\ c \end{bmatrix} = 0$$

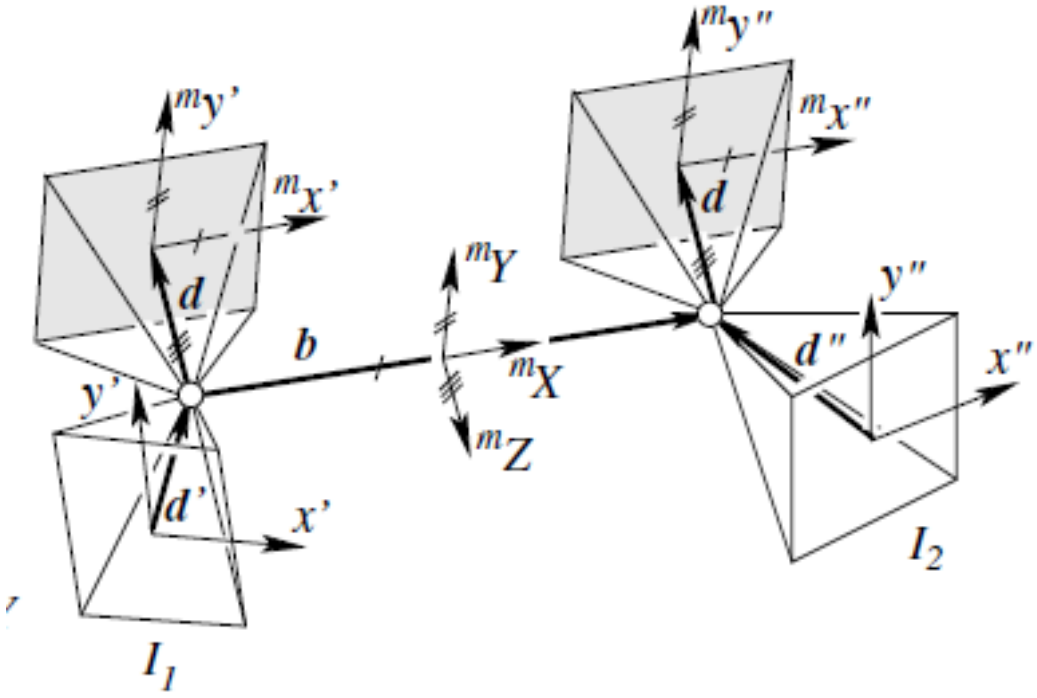
$$cB_X\left({}^{i}y'' - {}^{i}y'\right) = 0$$



Can we generate Stereo Normal Image Pair?

Can Stereo-normal case be generated?

- Given: image pair I_1 and I_2
- Aim: To general normalized image pair where
  - y-parallax $p_y = 0$
  - Same calibration $K = K' = K''$
  - Same rotation matrices $R = R' = R''$
  - Common viewing direction $d = avg(d_1, d_2)$
  - The common principal distance $c < 0$, normalized stereo image in viewing direction.

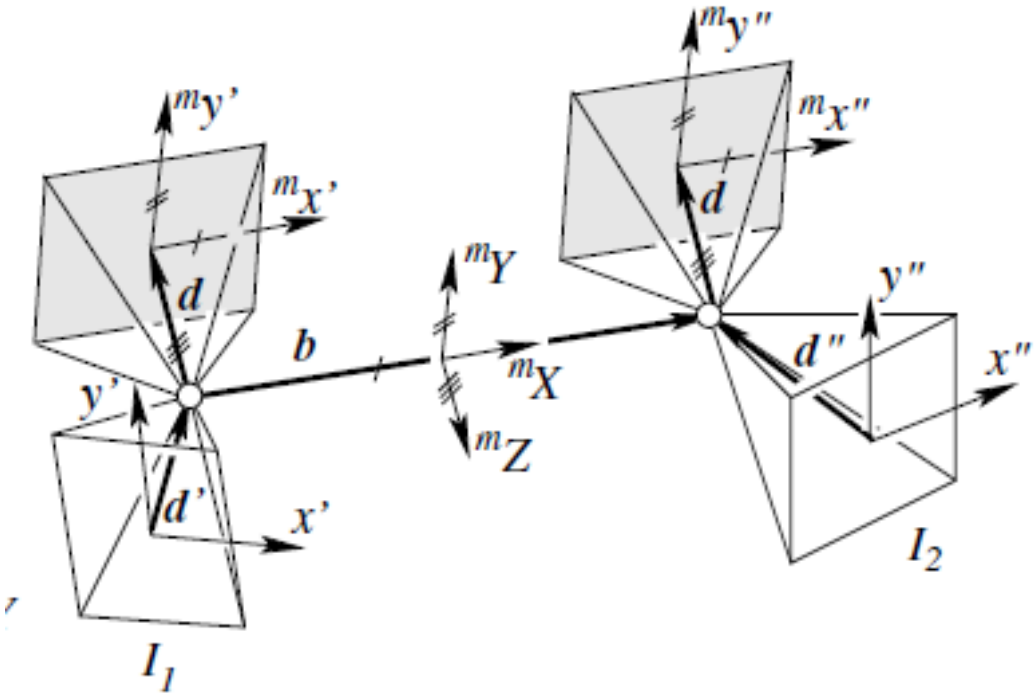


Creating Stereo Normal Case Image Pair

- Keep two bundles of ray
- Replace two image planes.
- Define two homographies transformation

$$x' \rightarrow {}^{m}x' \quad \& \quad x'' \rightarrow {}^{m}x''$$

$$x' = H'_m {}^{m}x' \quad \& \quad x'' = H''_m {}^{m}x''$$



Formulating the homography

Given: the two projection matrices P' and P'' with

- different intrinsic parameters $\mathrm{K}'$and K'',
- different rotation matrices R' and R'',
- different projection centres Z' and Z''

- **Goal:** find H'_m and H''_m

$$\mathrm{x}' = \mathrm{P}'\mathrm{X} = \mathrm{K}'R'\boxed{[I|-\mathrm{Z}']}\mathrm{X} \qquad \mathrm{x}'' = \mathrm{P}''\mathrm{X} = \mathrm{K}''R''\boxed{[I|-\mathrm{Z}'']}\mathrm{X}$$

$$^{m}\mathrm{x}' = {}^{m}\mathrm{P}'\mathrm{X} = \mathrm{K}R\boxed{[I|-\mathrm{Z}']}\mathrm{X} \qquad {}^{m}\mathrm{x}'' = {}^{m}\mathrm{P}''\mathrm{X} = \mathrm{K}R\boxed{[I|-\mathrm{Z}'']}\mathrm{X}$$

Formulating the homography

- Since $\mathrm{x}' = \mathrm{H}'_m {}^m\mathrm{x}'$ & $\mathrm{x}'' = \mathrm{H}''_m {}^m\mathrm{x}''$

- We obtain the homography

$$\mathrm{H}'_m = \mathrm{K}' R' R^T K^{-1} \qquad \mathrm{H}''_m = \mathrm{K}'' R'' R^T K^{-1}$$

- The inverse homography is given

$${}^m\mathrm{x}' = \mathrm{K} R \; R'^T \mathrm{K}'^T \mathrm{x}' \qquad {}^m\mathrm{x}'' = \mathrm{K} R \; R''^T \mathrm{K}''^T \mathrm{x}''$$

Parameters for homographies

- The homographies

$$\mathrm{H}'_m = \mathrm{K}' R' R^T K^{-1} \qquad \mathrm{H}''_m = \mathrm{K}'' R'' R^T K^{-1}$$

How to choose parameter K?

$K = \mathrm{Diag}([\mathrm{c}, \mathrm{c}, 1])$ where $\mathrm{c} < 0$

We have two transformed images in the viewing position

Parameters for homographies

- How about R ?
- Let $R = [r_1, r_2, r_3]^T$

- ${}^{m}\mathrm{x}'$ *and* ${}^{m}\mathrm{x}''$ *coordinate parallel to base vector B,*

  r_1=N(**b**) *where* N is normalized vector

- The viewing directions (Z-axes) are orthogonal to the base and average of viewing direction

  $$\mathbf{d}^* = \mathrm{N}\left(\frac{\mathrm{d}'}{|\mathrm{d}'|} + \frac{\mathrm{d}''}{|\mathrm{d}''|}\right)$$

  $$r_2 = \mathrm{N}(\mathbf{b} \times \mathbf{d}^*)$$

- The y-axes must perpendicular to N(**b**) and $\mathrm{N}(\mathbf{b} \times \mathbf{d}^*)$

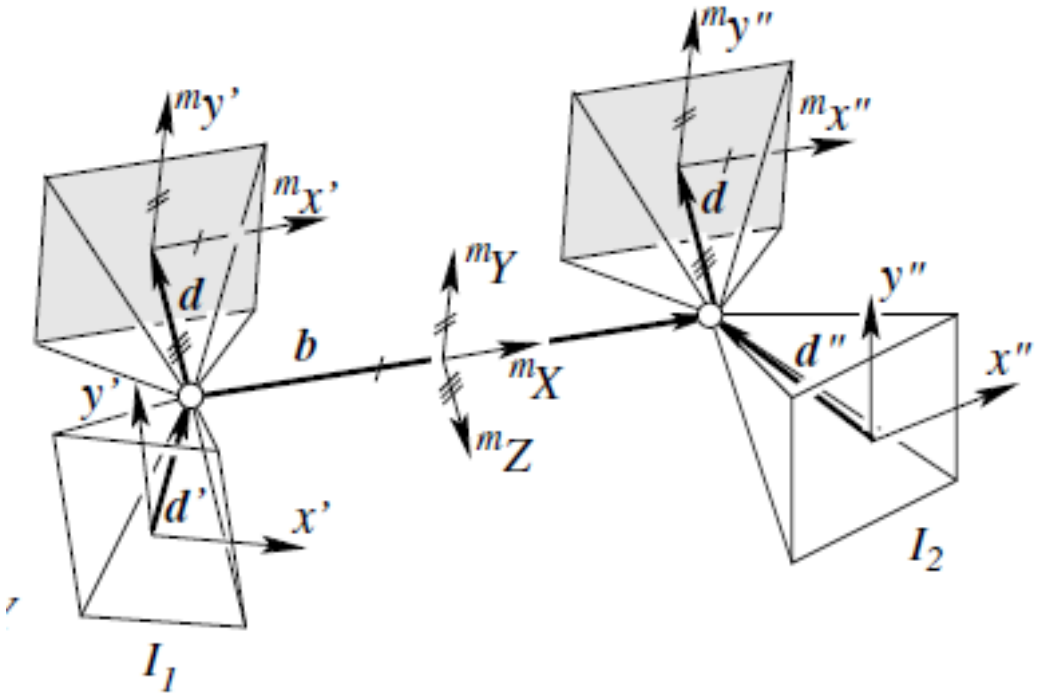
$$r_3 = \mathrm{N}(\mathbf{b} \times (\mathbf{b} \times \mathbf{d}^*))$$

- Finally we have

$$R = [\mathrm{N}(\mathbf{b}), \mathrm{N}(\mathbf{b} \times \mathbf{d}^*), \mathrm{N}(\mathbf{b} \times (\mathbf{b} \times \mathbf{d}^*))]^T$$

- The common viewing direction is

$$-\mathrm{N}(\mathbf{b} \times (\mathbf{b} \times \mathbf{d}^*))$$

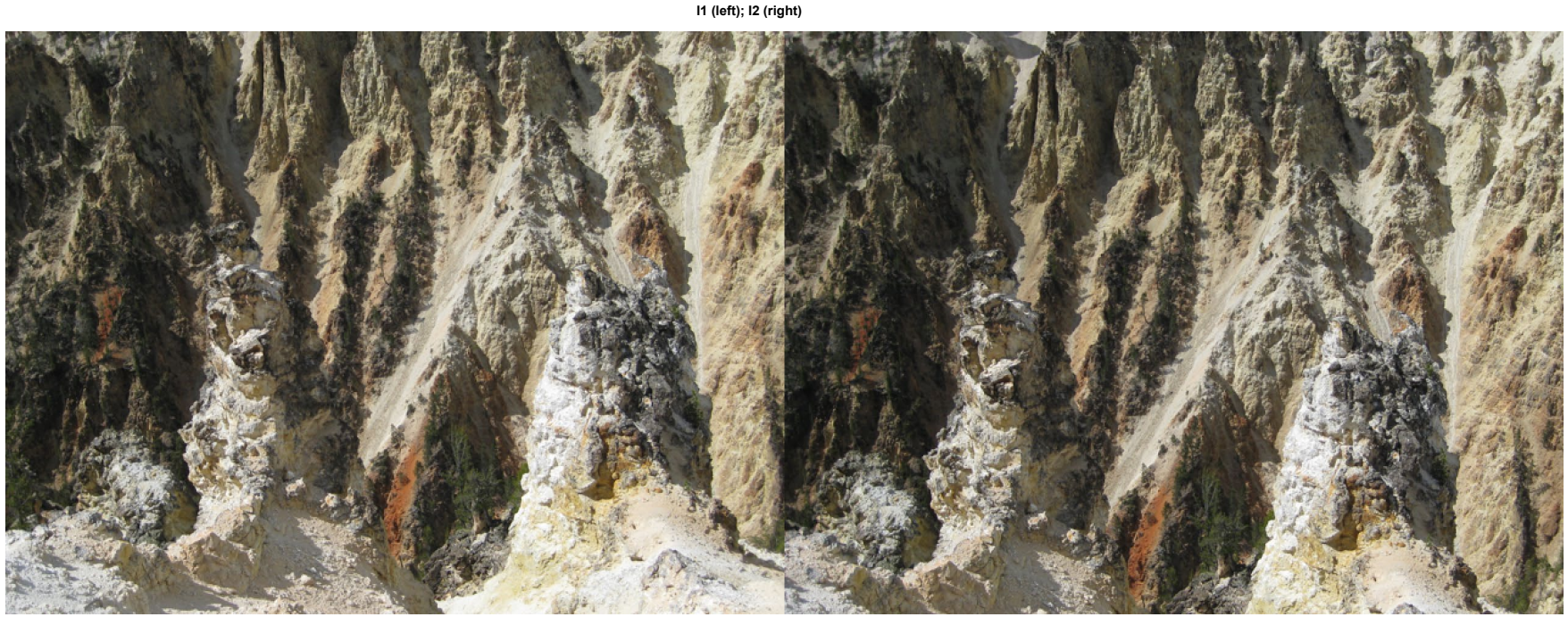


Summary of Steps for generating Stereo Normal image pair

- Given an image pair
- Define the two homographies between image pair and generated normal pair $\mathrm{x}' = \mathrm{H}'_m {}^m\mathrm{x}'$ & $\mathrm{x}'' = \mathrm{H}''_m {}^m\mathrm{x}''$
- Compute the homographies $\mathrm{H}'_m = \mathrm{K}'R'R^T K^{-1}$ & $\mathrm{H}''_m = \mathrm{K}''R''R^T K^{-1}$
- Solve for K and R
- The projection is given by

$$ {}^m\mathrm{x}' = {}^m\mathrm{P}'\mathrm{X} = \mathrm{K}R[I|-\mathrm{Z}']\mathrm{X} \qquad {}^m\mathrm{x}'' = {}^m\mathrm{P}''\mathrm{X} = \mathrm{K}R[I|-\mathrm{Z}'']\mathrm{X} $$

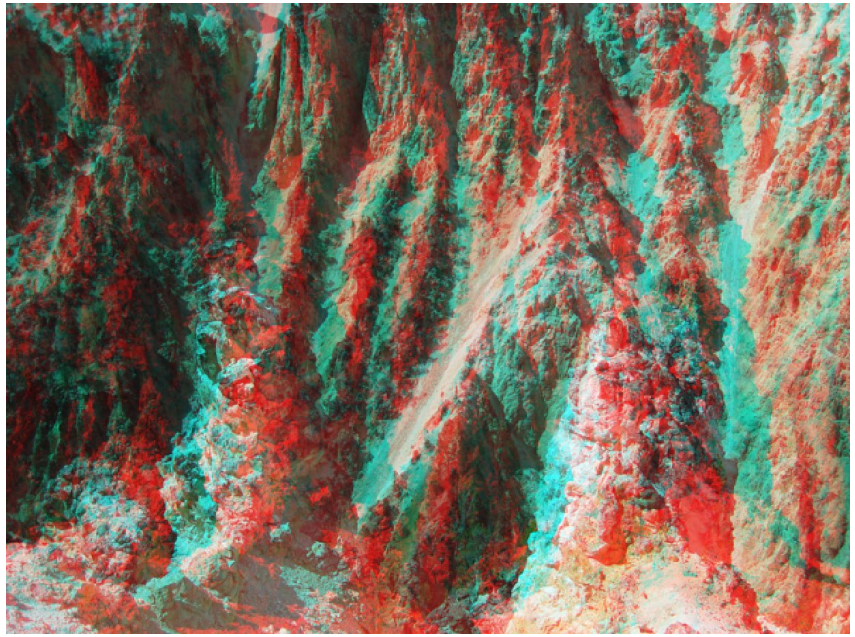
Example



Source: MATLAB

Rectification – Stereo Normal Pair

Composite Image (Red - Left Image, Cyan - Right Image)



Obvious offset between the images in orientation and position

Rectified Stereo Images (Red - Left Image, Cyan - Right Image)

Rectify Images

```
[t1, t2] = estimateUncalibratedRectification(fMatrix, ...
  inlierPoints1.Location, inlierPoints2.Location, size(I2));
tform1 = projective2d(t1);
tform2 = projective2d(t2);
```

```
[I1Rect, I2Rect] = rectifyStereoImages(I1, I2, tform1, tform2);
figure;
imshow(stereoAnaglyph(I1Rect, I2Rect));
title('Rectified Stereo Images (Red - Left Image, Cyan - Right Image)');
```

Today’s Agenda

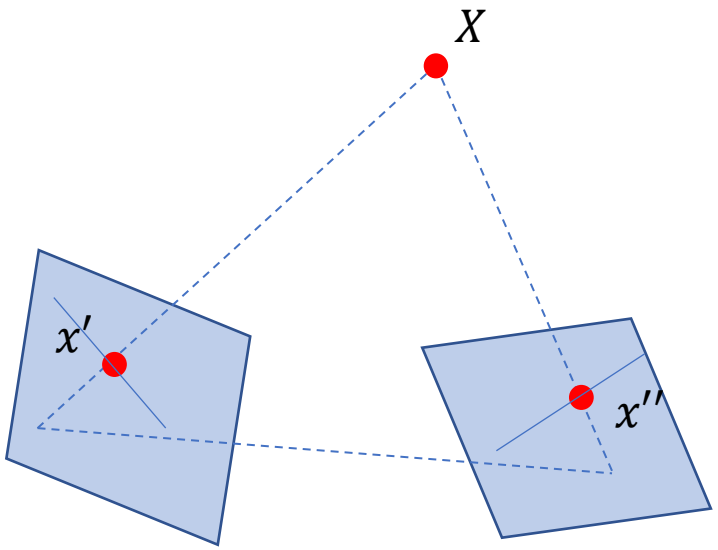
- Stereo Normal View
- Stereo Correspondence
- Disparity Calculation
- Stereo Normal View Reconstruction
- Triangulation of calibrated camera

Stereo Correspondence Matching

Correspondence Problem

- Given a point in 3D, we need to find the corresponding points in left and right images

- x' belongs to $l' = Fx''$
- x' belongs to $l'' = F^T x'$



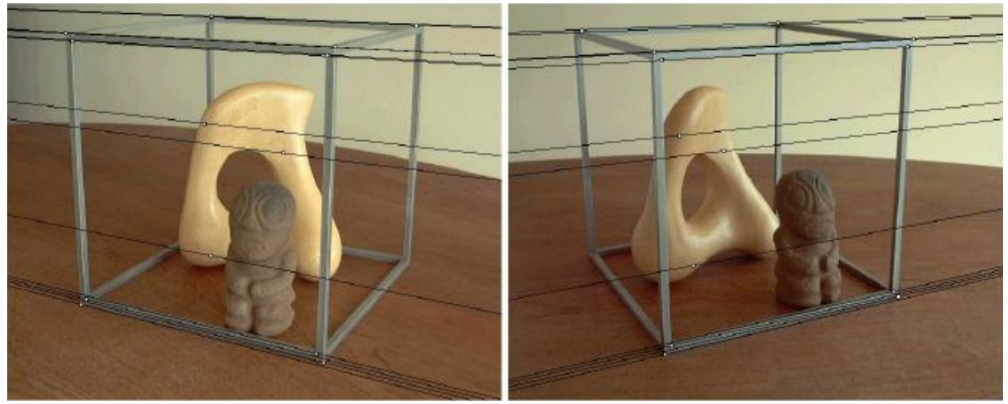
Difference between Stereo and optical flow

- 1-D VS 2D: Stereo search is 1-D while optical flow search for (u,v) is a 2-D problem.
- Disparities are quantized in integer unit of pixel
- Occlusion are modeled explicitly in stereo vision
- In stereo case, you must either calibrate the camera or estimate the epipolar lines
- In stereo, the images must be taken simultaneously.
- In stereo, monotonicity constraints can be imposed.

Why are parallel images useful?

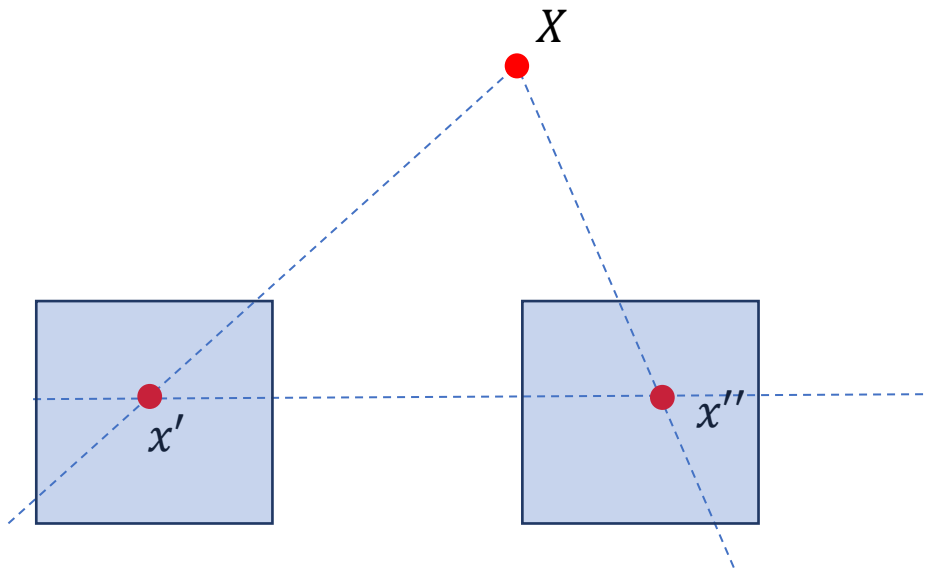
- Make the Correspondence problem easier

- Makes the triangulation process easy.



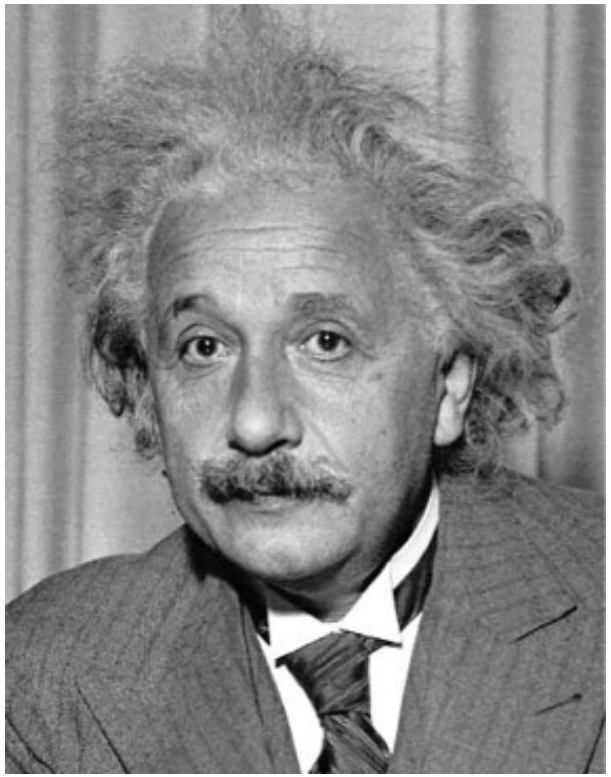
In stereo normal case

- When image are rectified, this problem becomes mush easier
- How to search for correspondence?



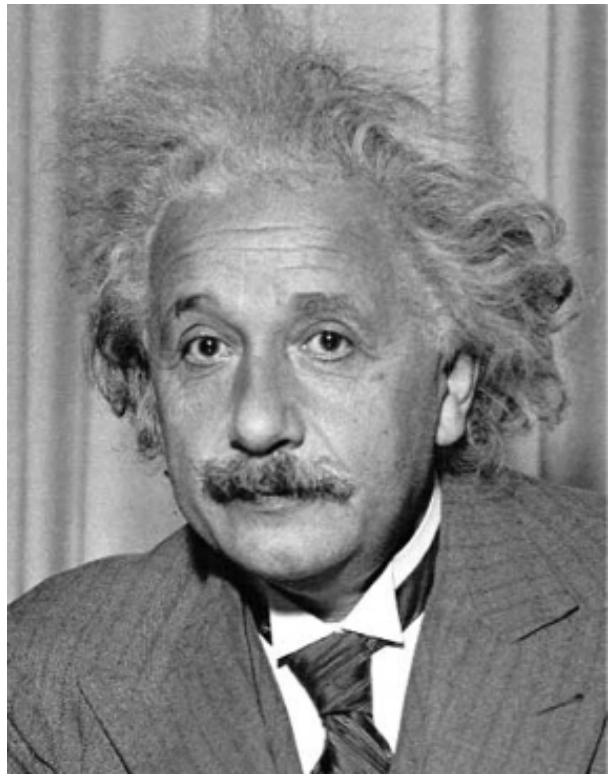
Find this template

How do we detect the template 👁 in he following image?



Find this template

How do we detect the template 👁 in he following image?



filter 👁

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

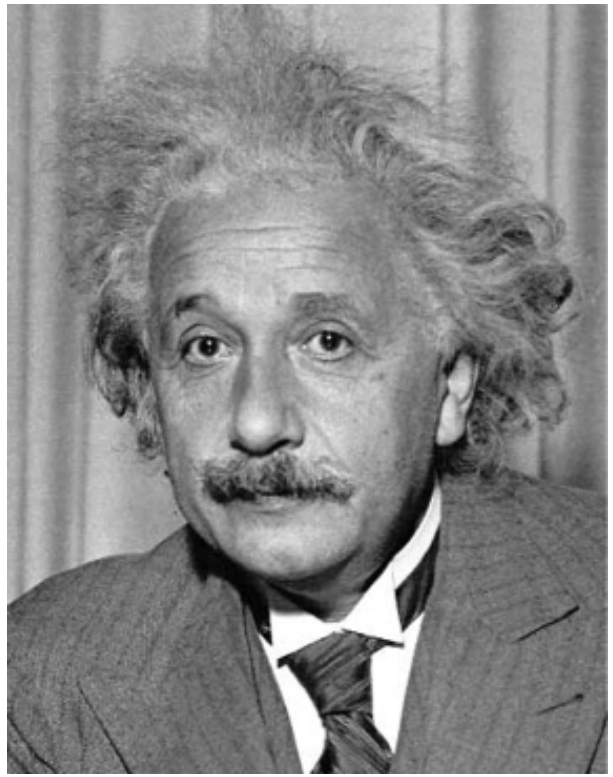
image

What will the output look like?

Solution 1: Filter the image using the template as filter kernel.

Find this template

How do we detect the template 👁 in he following image?

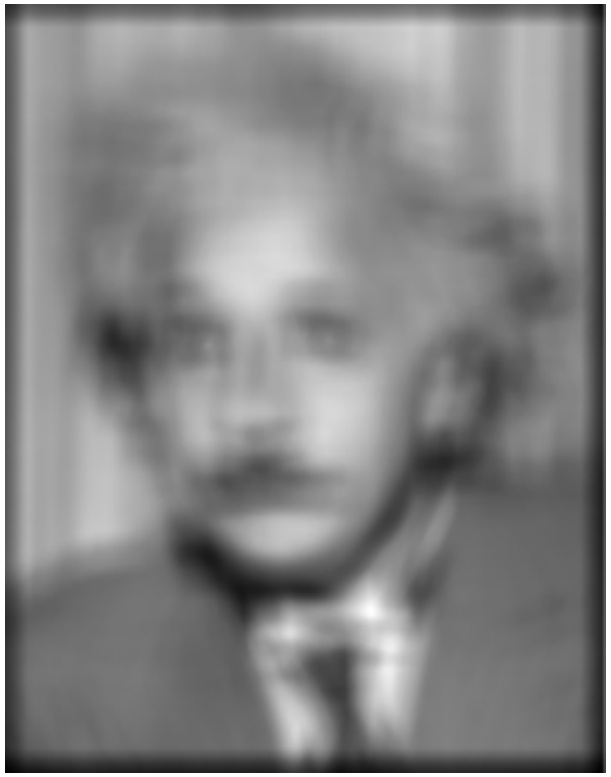


filter 👁

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

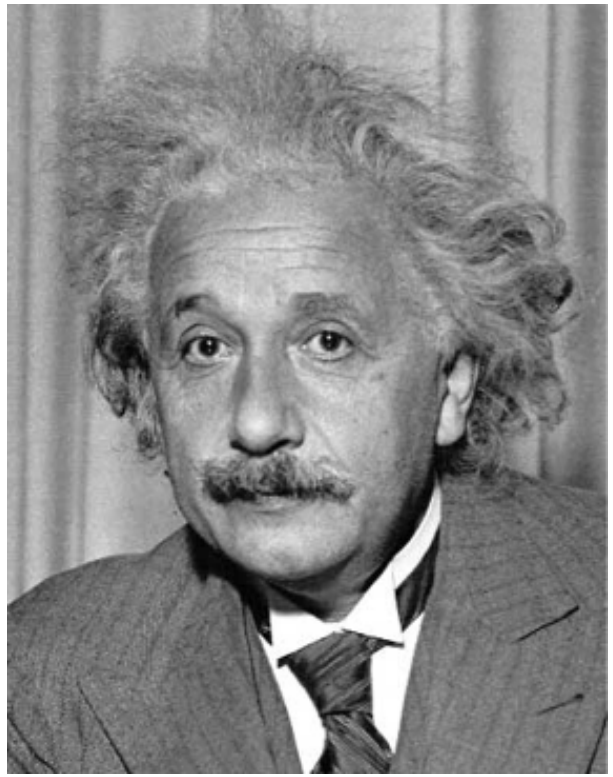


Solution 1: Filter the image using the template as filter kernel.

What went wrong?

Find this template

How do we detect the template 👁 in he following image?

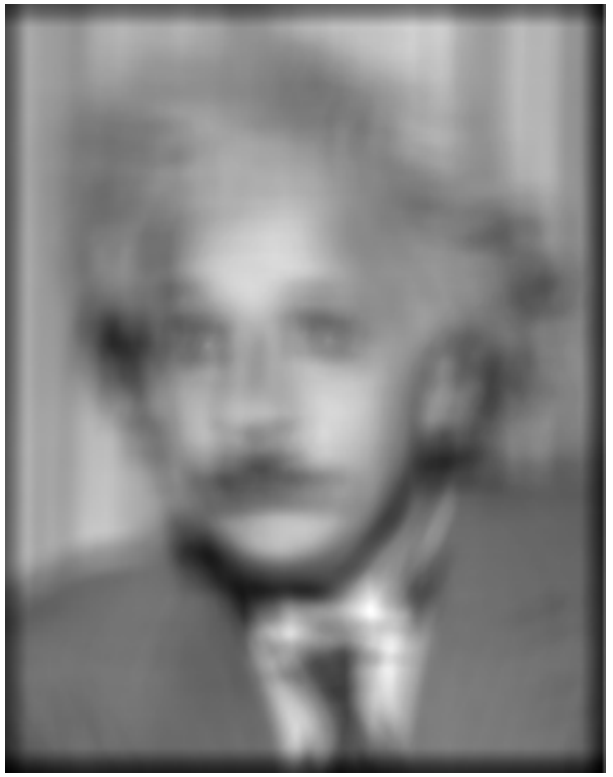


filter 👁

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

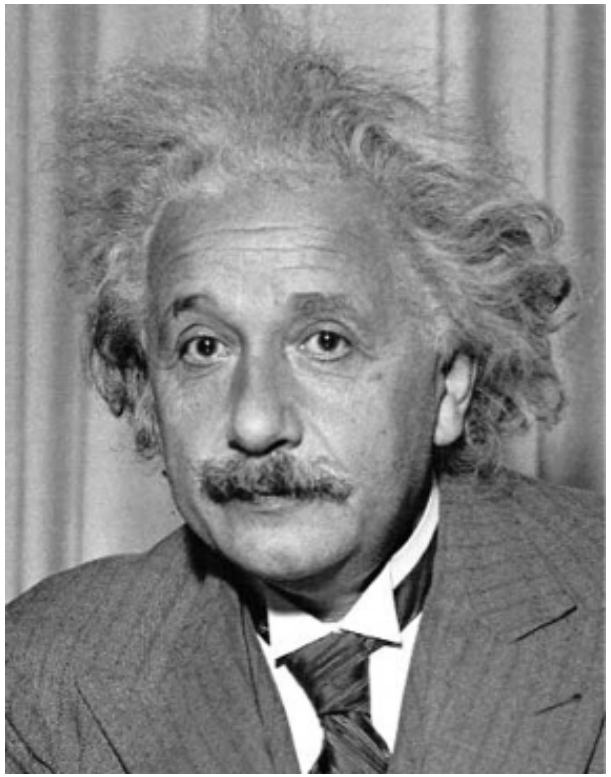


Increases for higher local intensities.

Solution 1: Filter the image using the template as filter kernel.

Find this template

How do we detect the template  in he following image?



filter 

template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

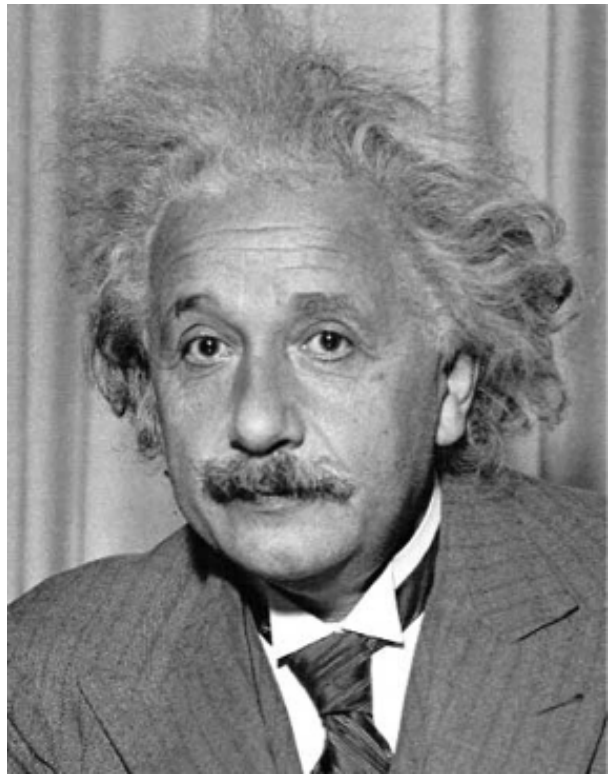
image

What will the output look like?

Solution 2: Filter the image using a *zero-mean* template.

Find this template

How do we detect the template in he following image?



output

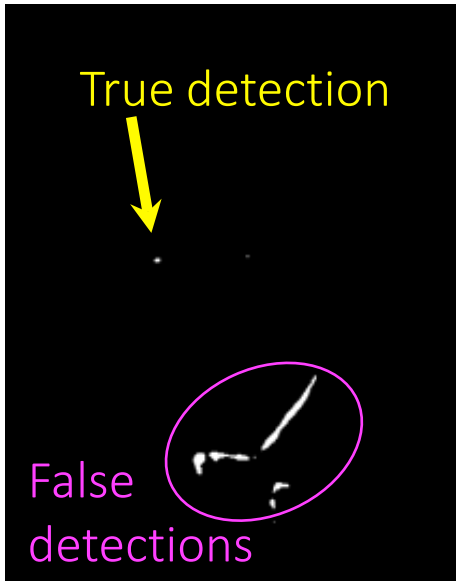
filter

template mean

output

$$h[m,n] = \sum_{k,l}(g[k,l] - \bar{g})f[m+k, n+l]$$

thresholding



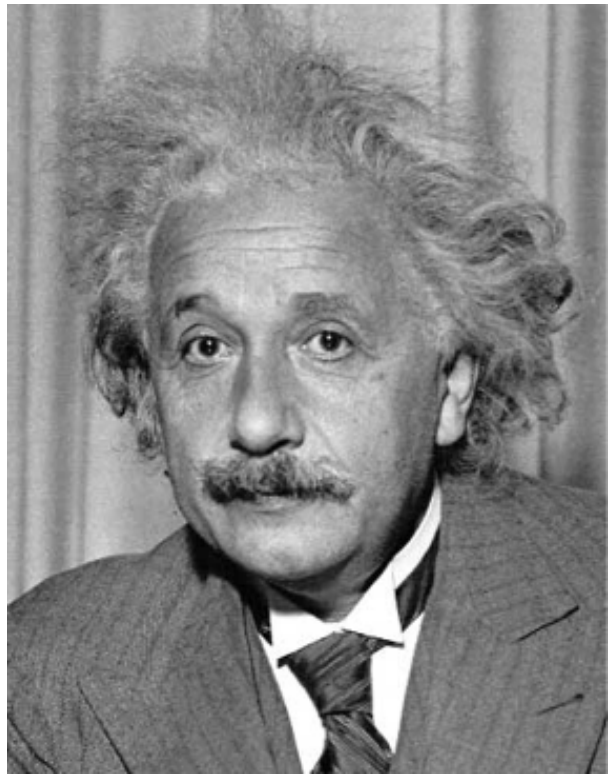
image

Solution 2: Filter the image using a *zero-mean* template.

What went wrong?

Find this template

How do we detect the template in he following image?



output

filter

template mean

output

$$h[m,n] = \sum_{k,l}(g[k,l] - \bar{g})f[m+k, n+l]$$

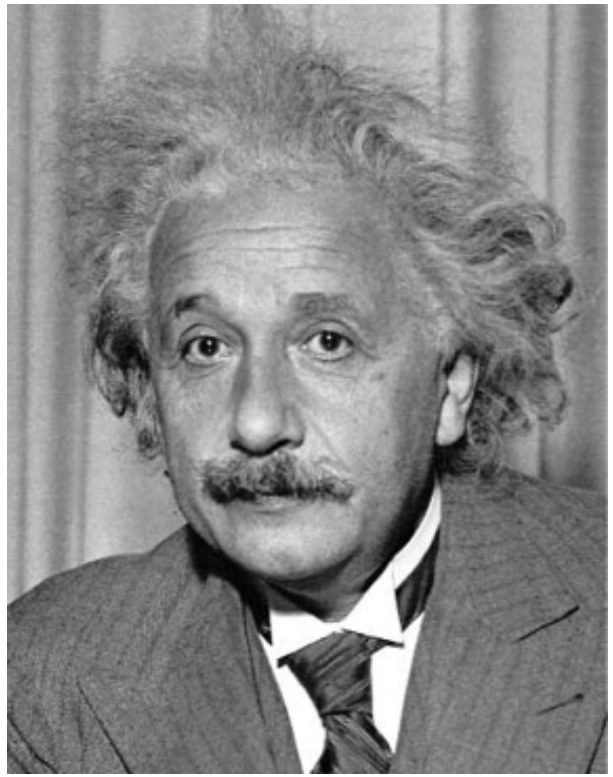
image

Not robust to high-contrast areas

Solution 2: Filter the image using a *zero-mean* template.

Find this template

How do we detect the template 👁 in he following image?



filter 👁

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

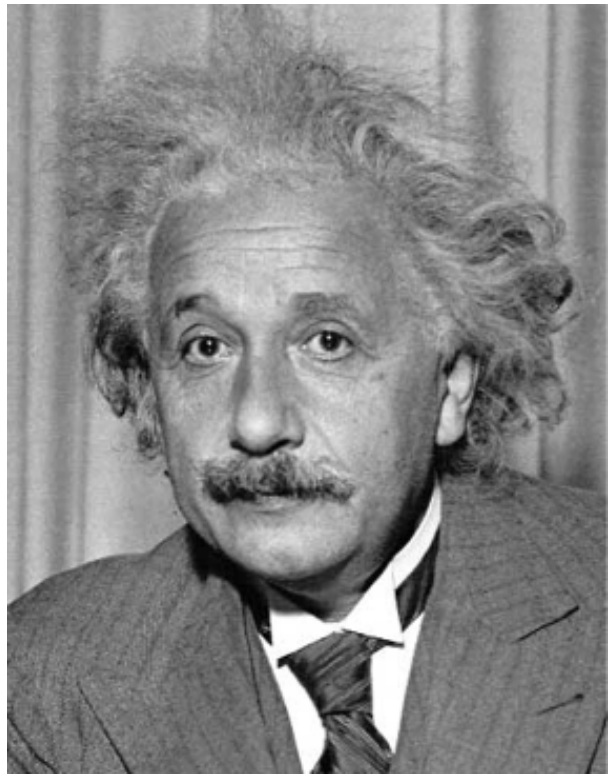
image

What will the output look like?

Solution 3: Use sum of squared differences (SSD).

Find this template

How do we detect the template in he following image?



filter

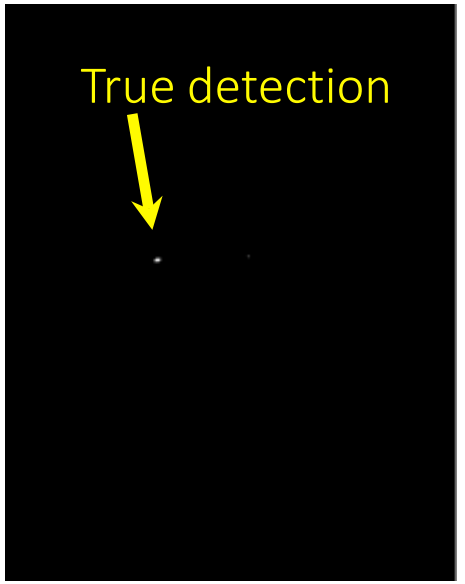
output

$$h[m,n] = \sum_{k,l}(g[k,l] - f[m+k, n+l])^2$$

image

1-output

thresholding

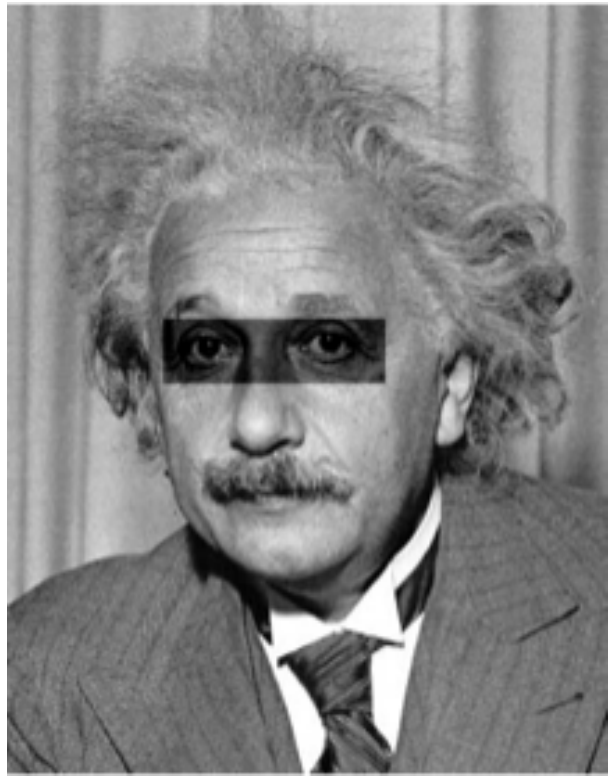


Solution 3: Use sum of squared differences (SSD).

What could go wrong?

Find this template

How do we detect the template 👁 in he following image?



1-output

filter 👁

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

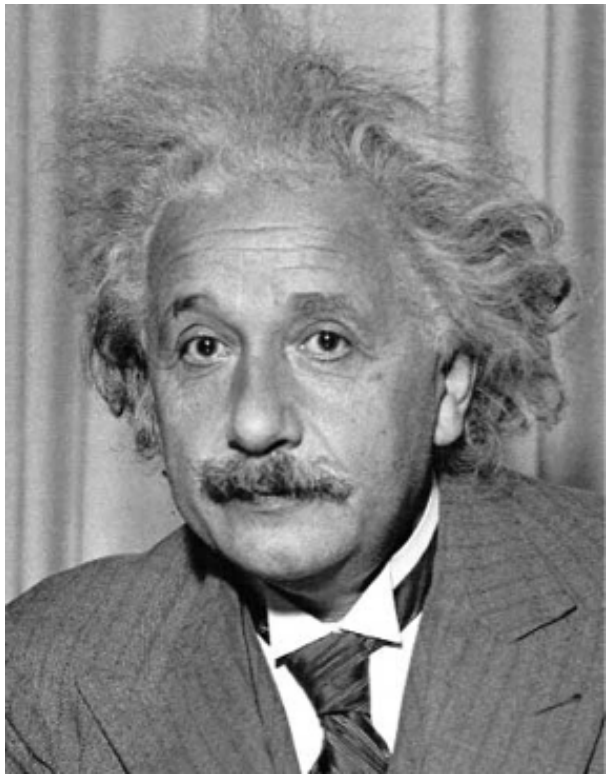
image

Not robust to local intensity changes

Solution 3: Use sum of squared differences (SSD).

Find this template

How do we detect the template 👁 in he following image?



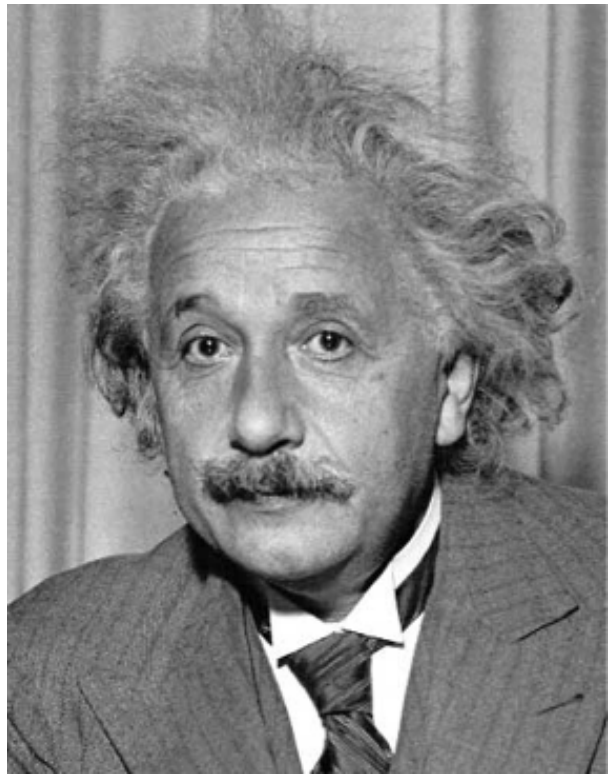
Observations so far:

- subtracting mean deals with brightness bias
- dividing by standard deviation removes contrast bias

Can we combine the two effects?

Find this template

How do we detect the template 👁 in he following image?



What will the output look like?

filter 👁

template mean

output

$$h[m,n] = \frac{\sum_{k,l}(g[k,l] - \bar{g})(f[m+k,n+l] - \bar{f}_{m,n})}{\sqrt{(\sum_{k,l}(g[k,l] - \bar{g})^2 \sum_{k,l}(f[m+k,n+l] - \bar{f}_{m,n})^2)}}$$

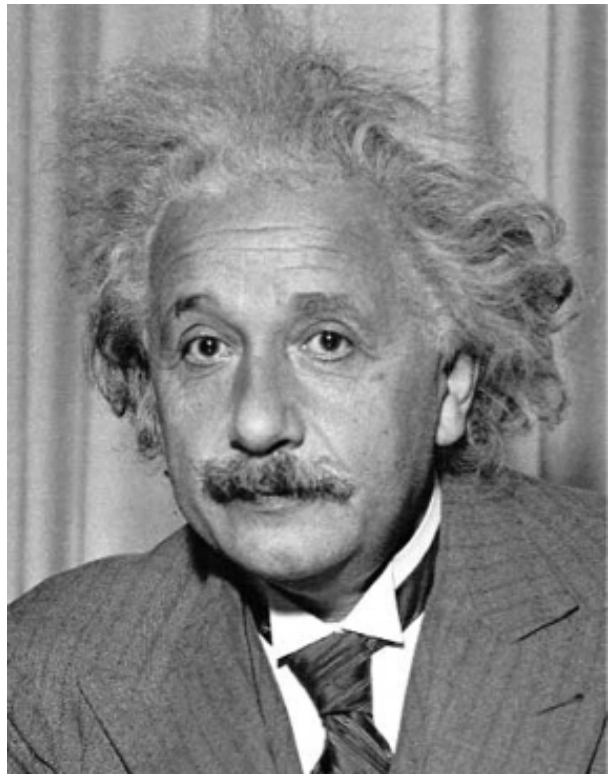
image

local patch mean

Solution 4: Normalized cross-correlation (NCC).

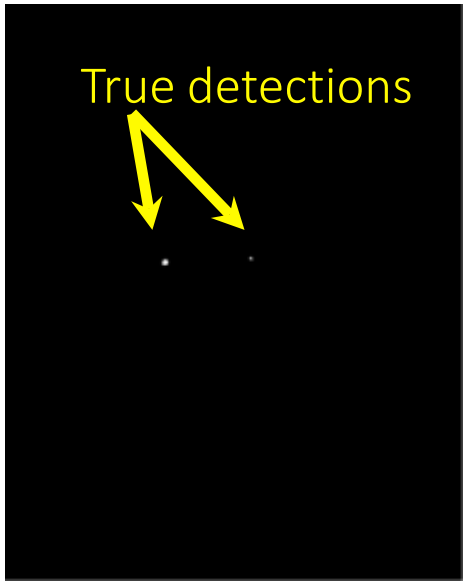
Find this template

How do we detect the template 👁 in he following image?



1-output

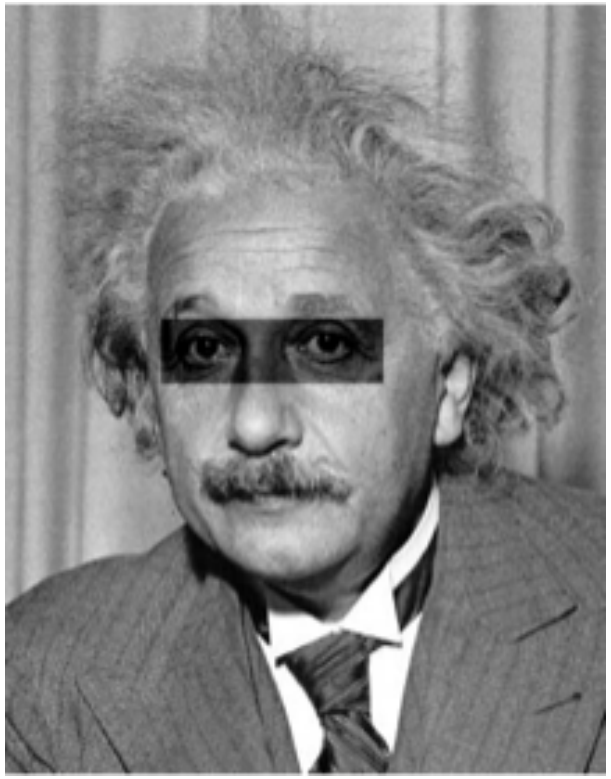
thresholding



Solution 4: Normalized cross-correlation (NCC).

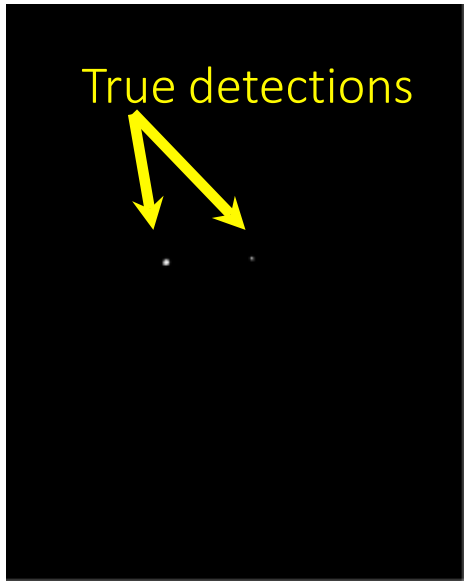
Find this template

How do we detect the template  in he following image?



1-output

thresholding

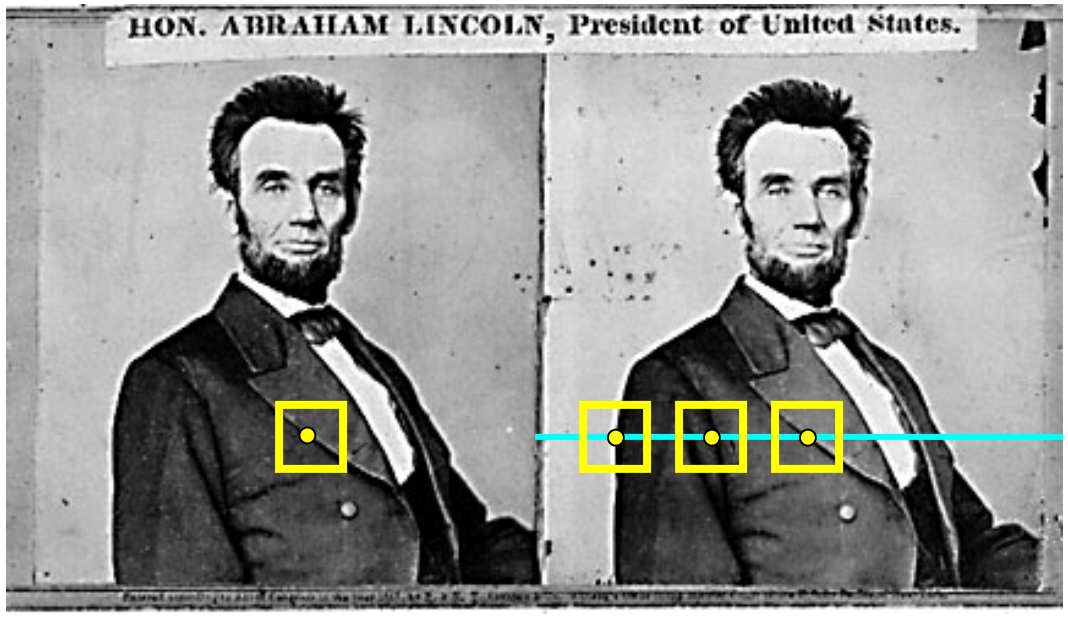


Solution 4: Normalized cross-correlation (NCC).

Basic stereo matching algorithm

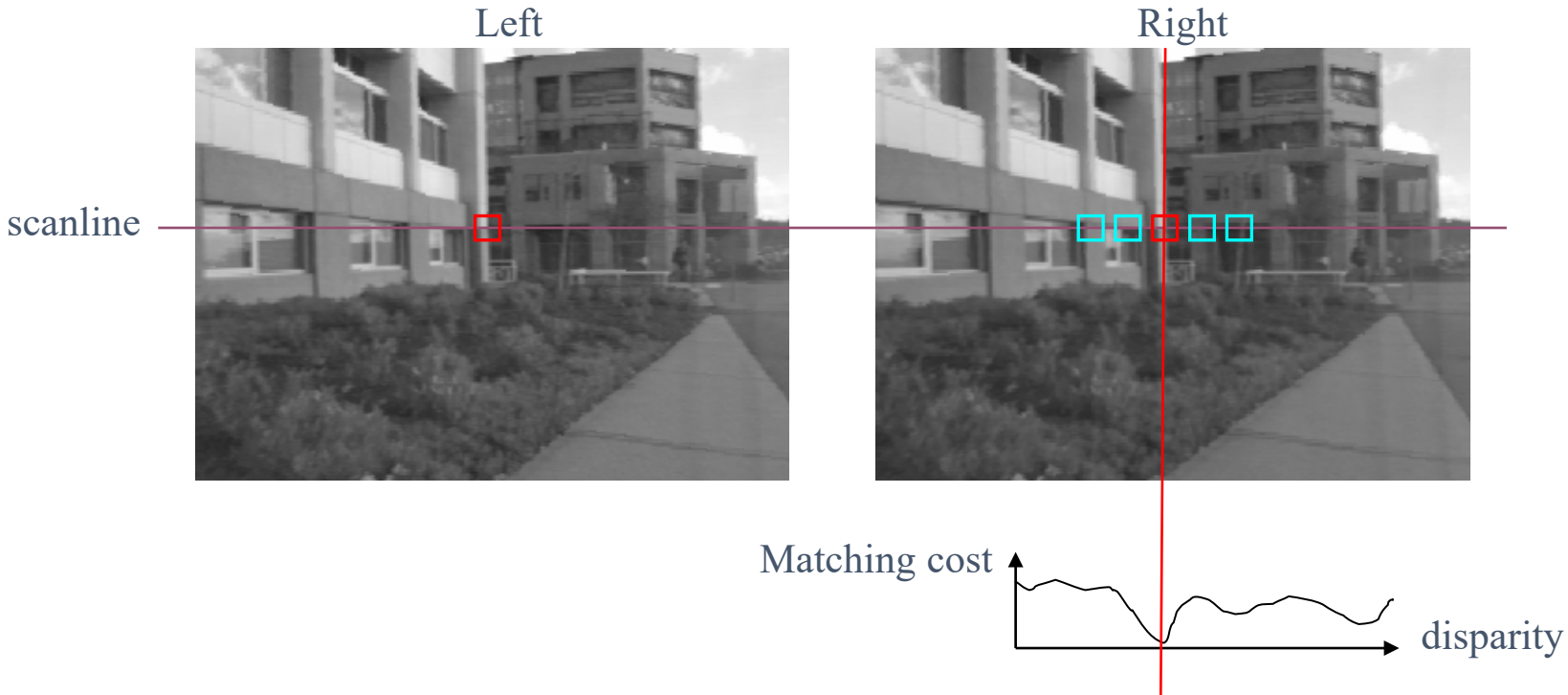
- For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Search along epipolar line and pick the best match
  - Triangulate the matches to get depth information

- Simplest case: epipolar lines are scanlines
  - In Stereo Normal Case

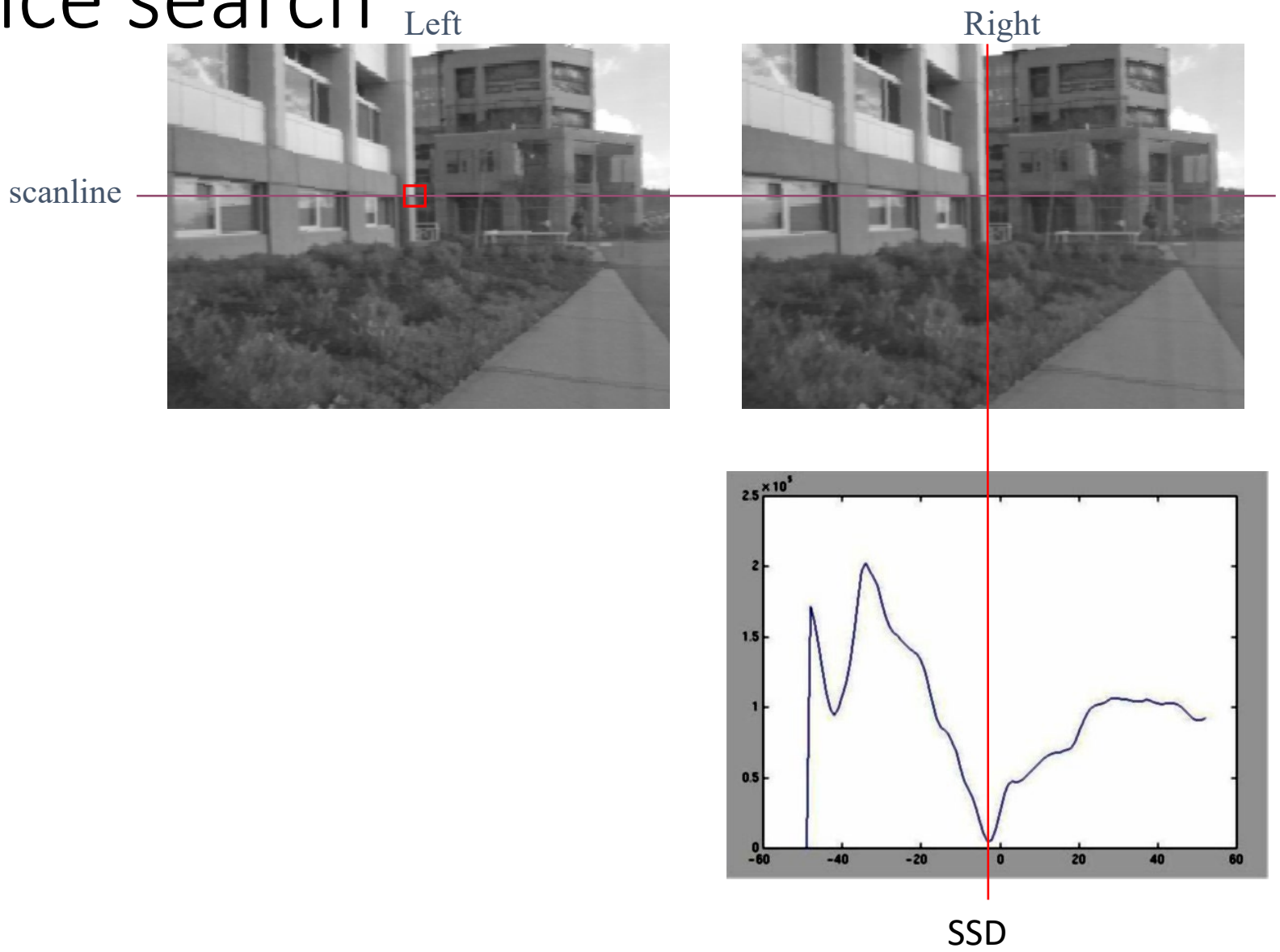


Correspondence search

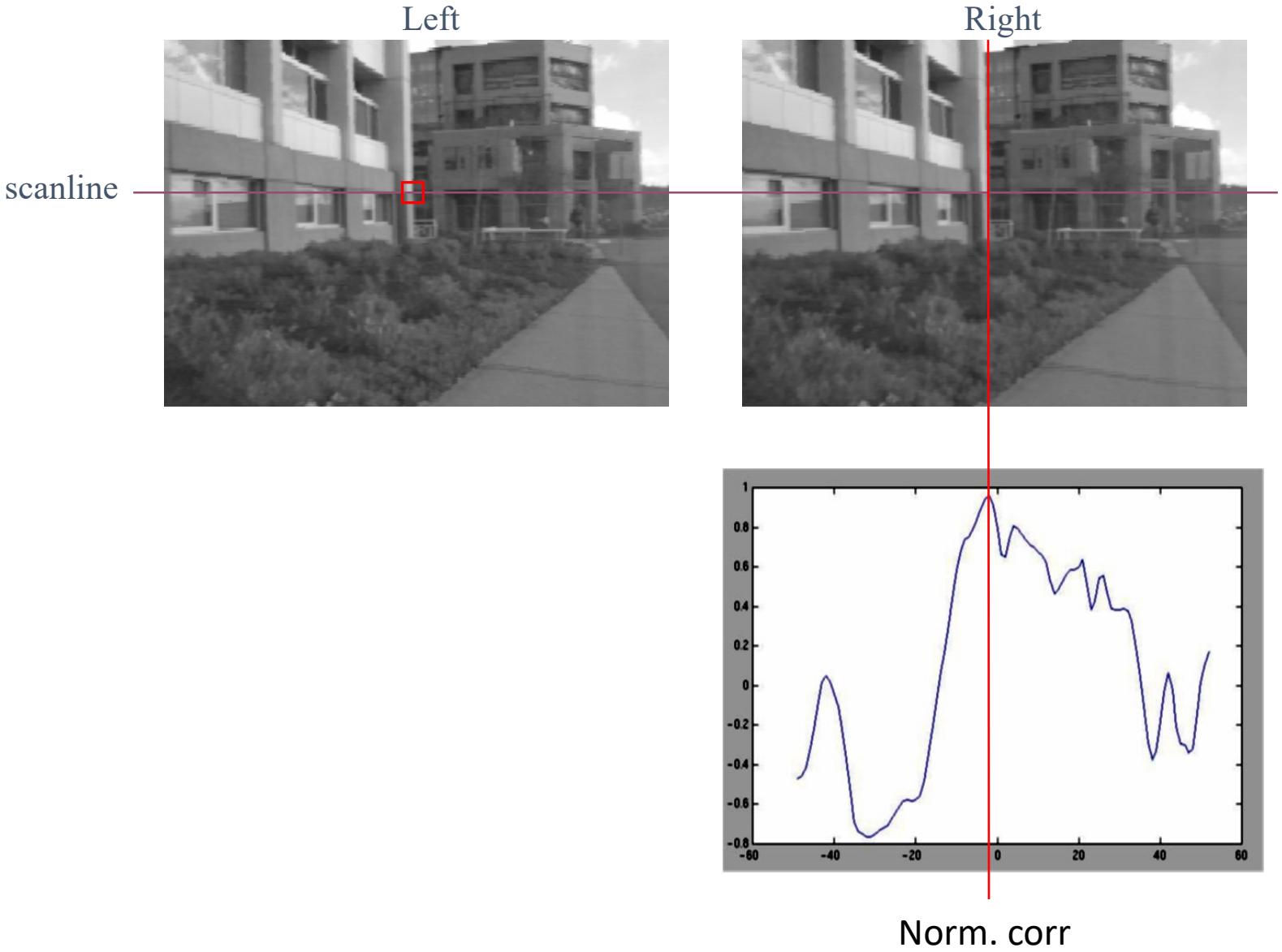
- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation



Correspondence search

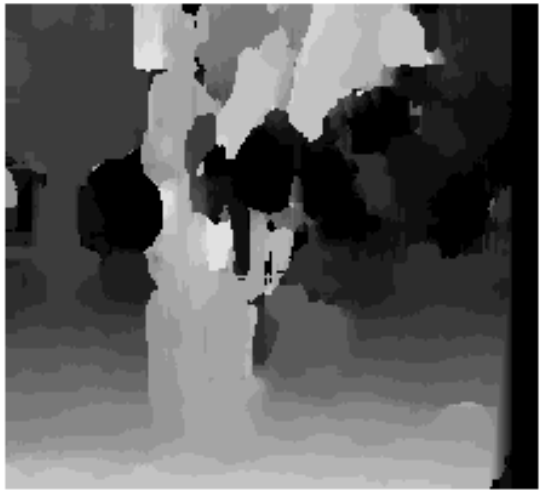


Correspondence search



Effect of window size

W = 3



W = 20

- Smaller window
  - \+ More detail
  - – More noise
- Larger window
  - \+ Smoother disparity maps
  - – Less detail
  - – Fails near boundaries

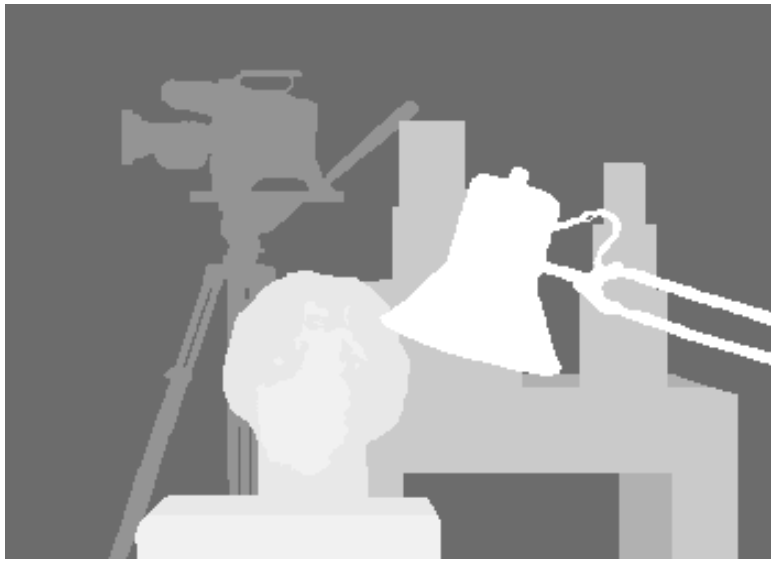
Better results with *adaptive window*

- T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*,, Proc. International Conference on Robotics and Automation, 1991.
- D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2):155-174, July 1998

Stereo Results

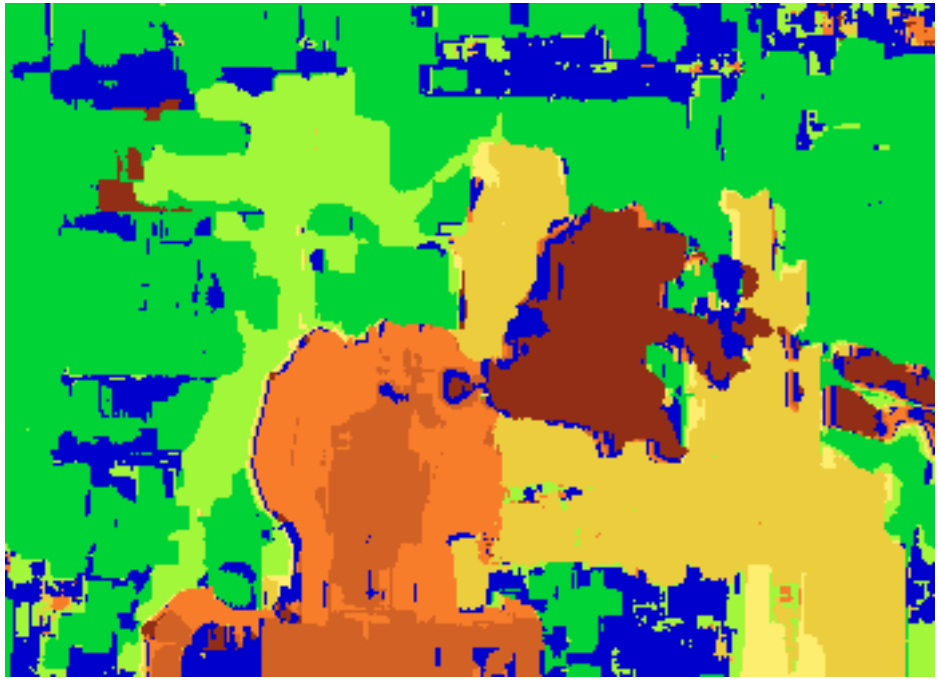
- Data from University of Tsukuba

Scene



Ground truth

Results with Window Search

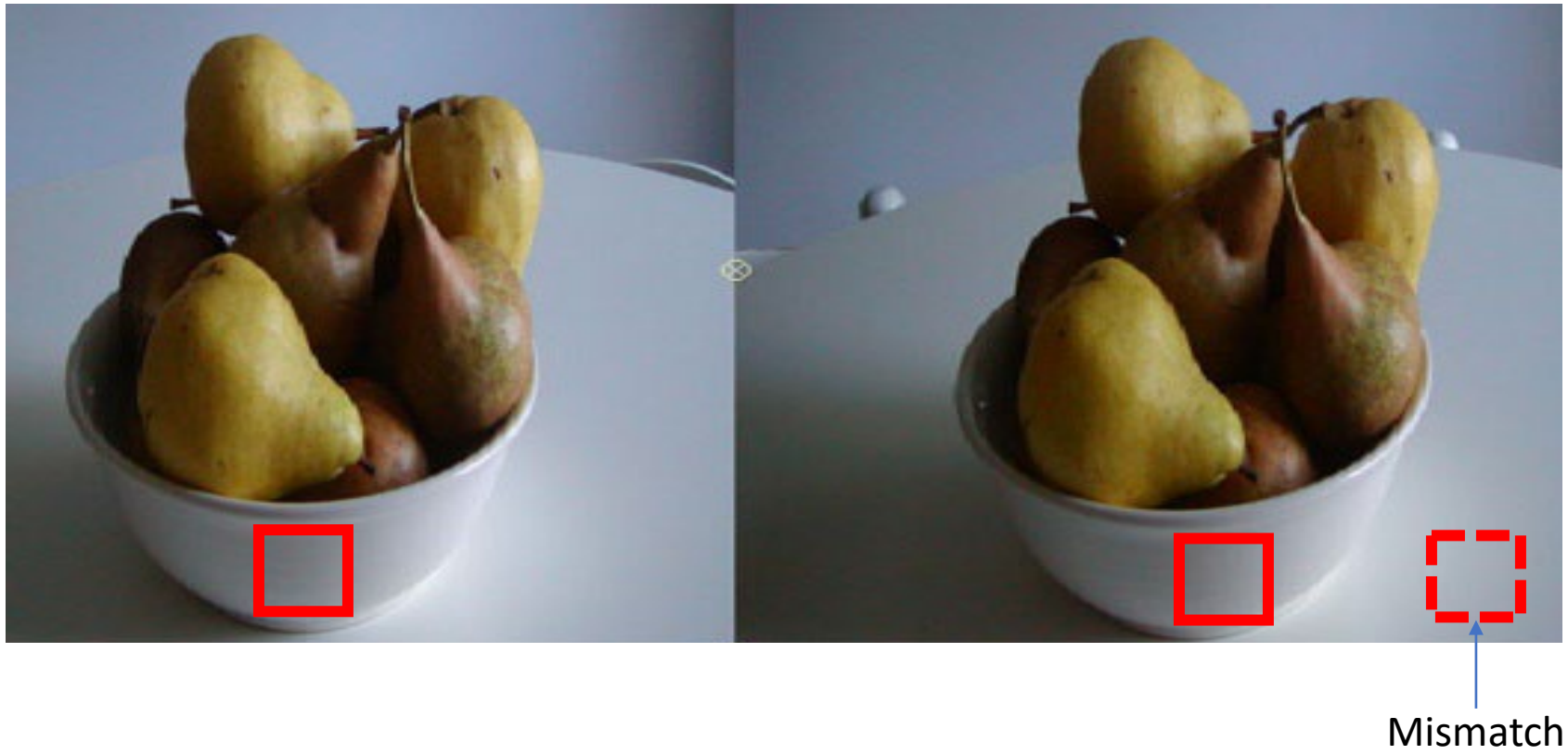


Window-based matching
(best window size)

Ground truth

Issues

- Mismatch at flat regions



Issues

- Repetitive Patterns

Better methods exist…

State of the art method

Ground truth

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

Difference between Stereo and optical flow

- 1-D VS 2D: Stereo search is 1-D while optical flow search for (u,v) is a 2-D problem.
- Disparities are quantized in integer unit of pixel
- Occlusion are modeled explicitly in stereo vision
- In stereo case, you must either calibrate the camera or estimate the epipolar lines
- In stereo, the images must be taken simultaneously.
- In stereo, monotonicity constraints can be imposed.

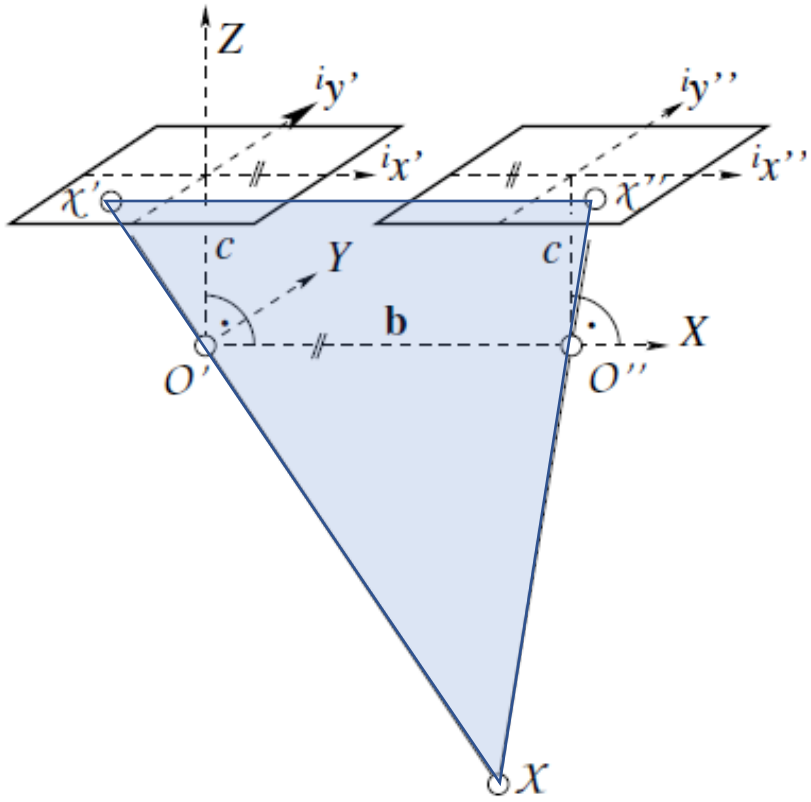
Stereo Correspondence

Today's Agenda

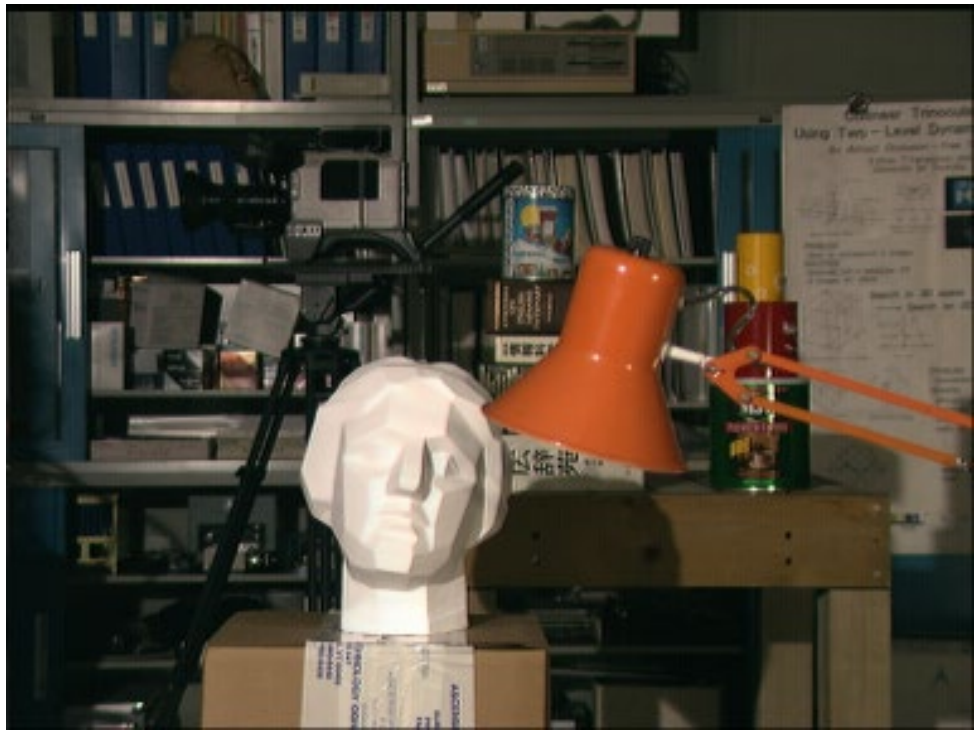
- Stereo Normal View
- Stereo Correspondence
- Disparity Calculation
- Stereo Normal View Reconstruction
- Triangulation of calibrated camera

Reconstruction - Stereo Normal Case

- Images in the same plane
- Identically oriented (R=I)
- Basis Vector B in X-direction



Stereo Image



Left Image

Right Image

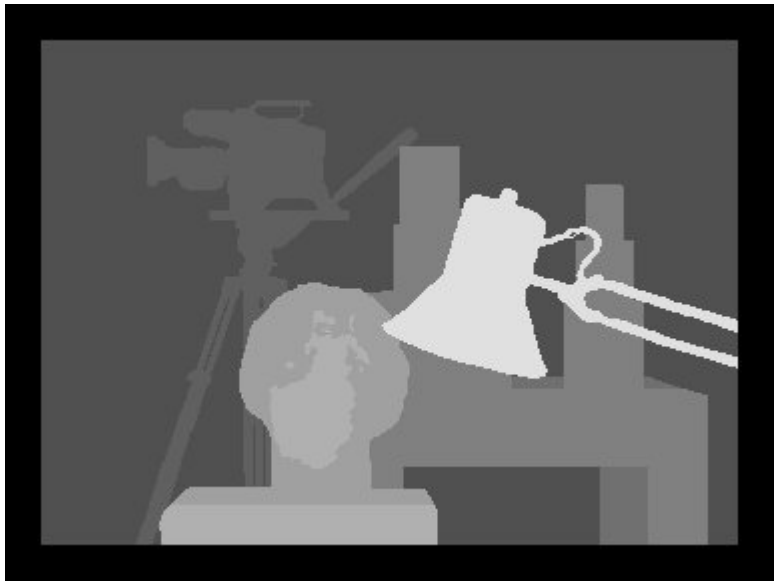
Left Image

Right Image

Stereo Image

Left Image

Right Image



Disparity Map

Closer object move more or less?

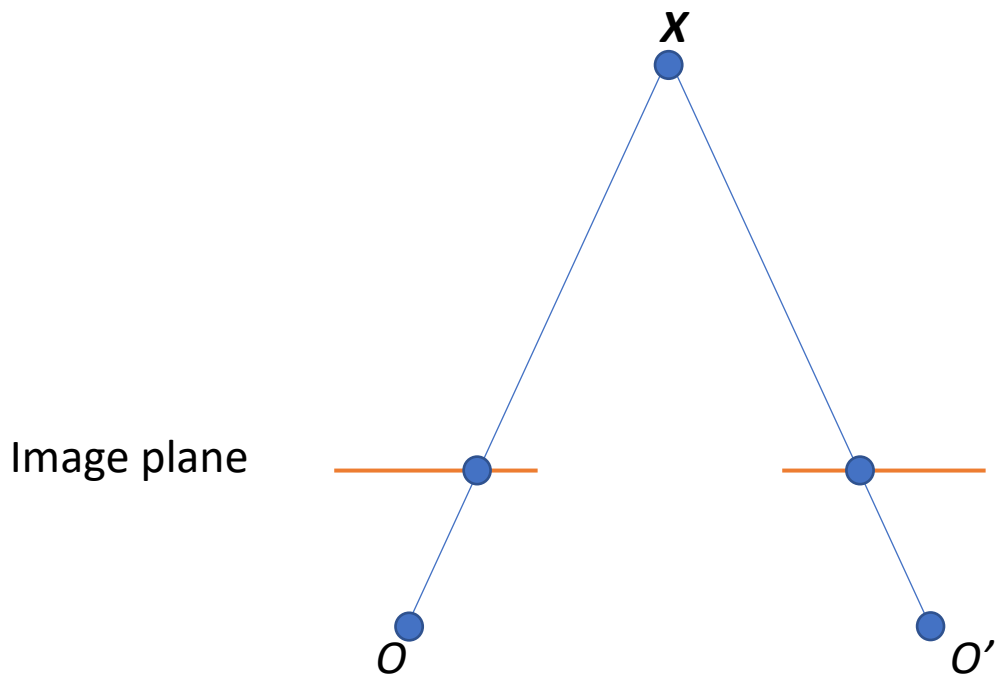
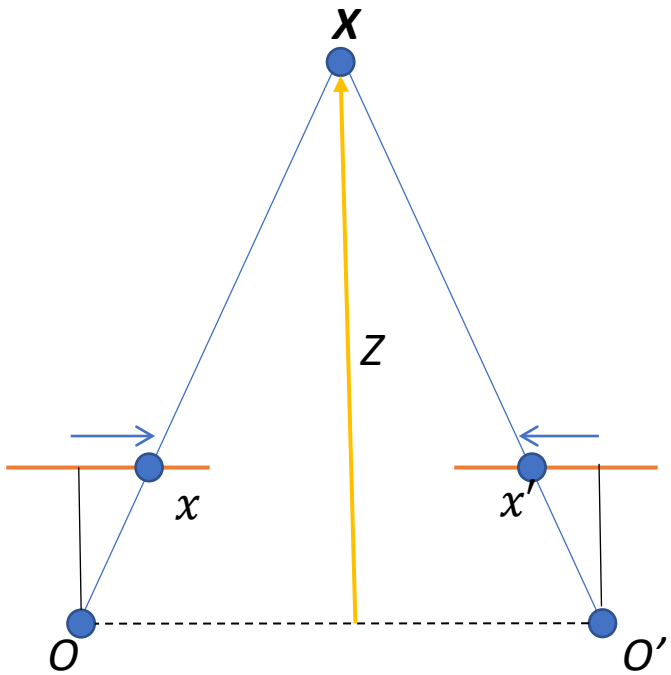
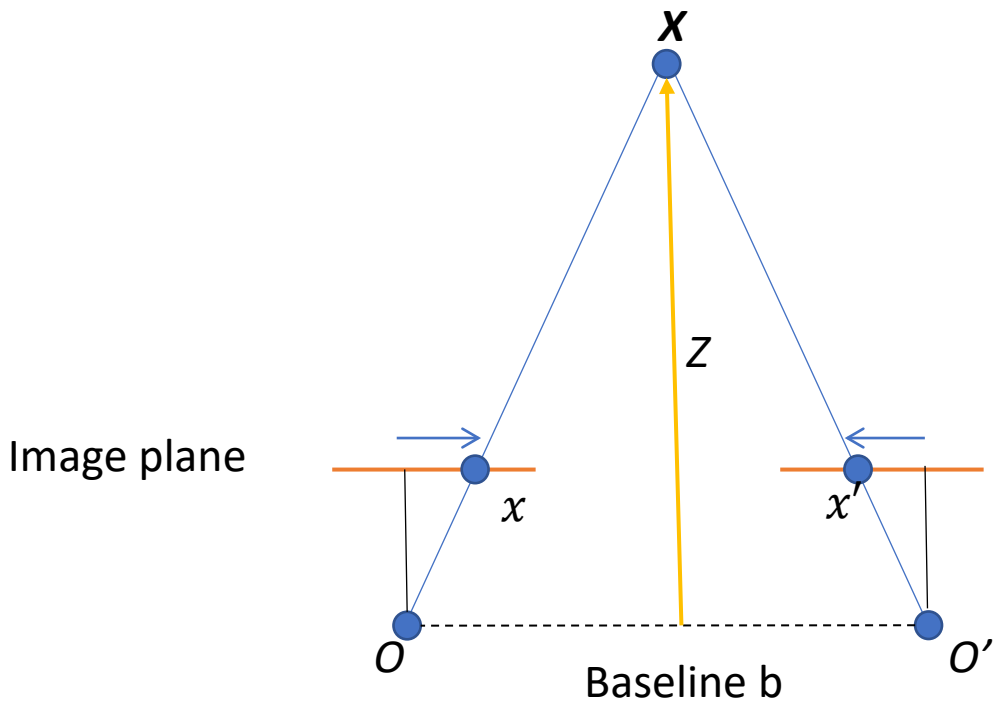
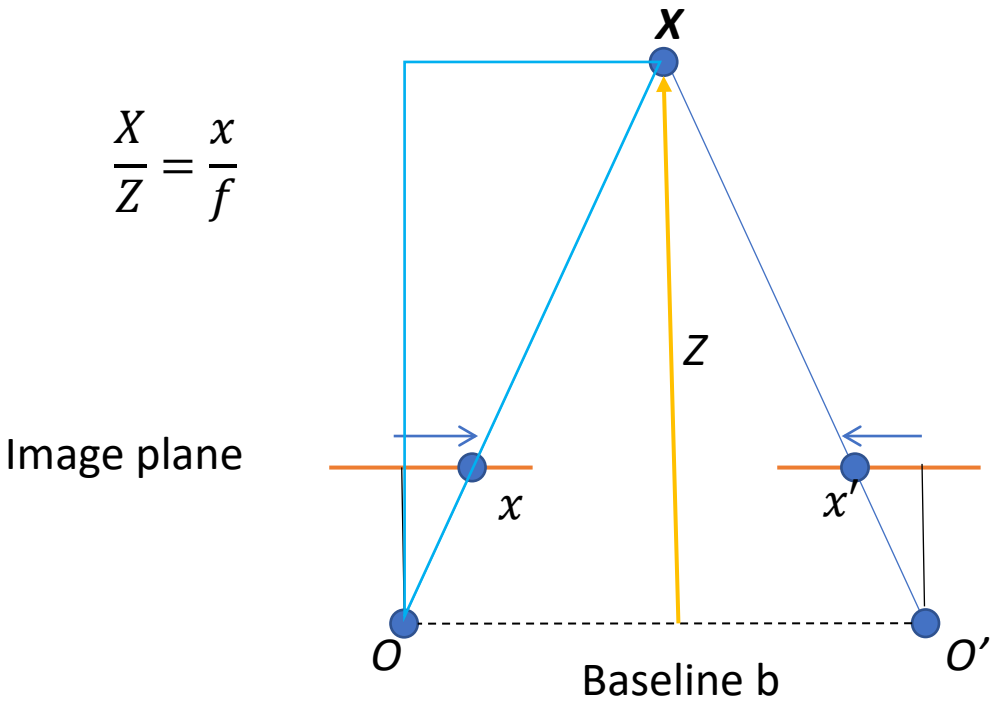

X
Image plane
O
O’

Image plane



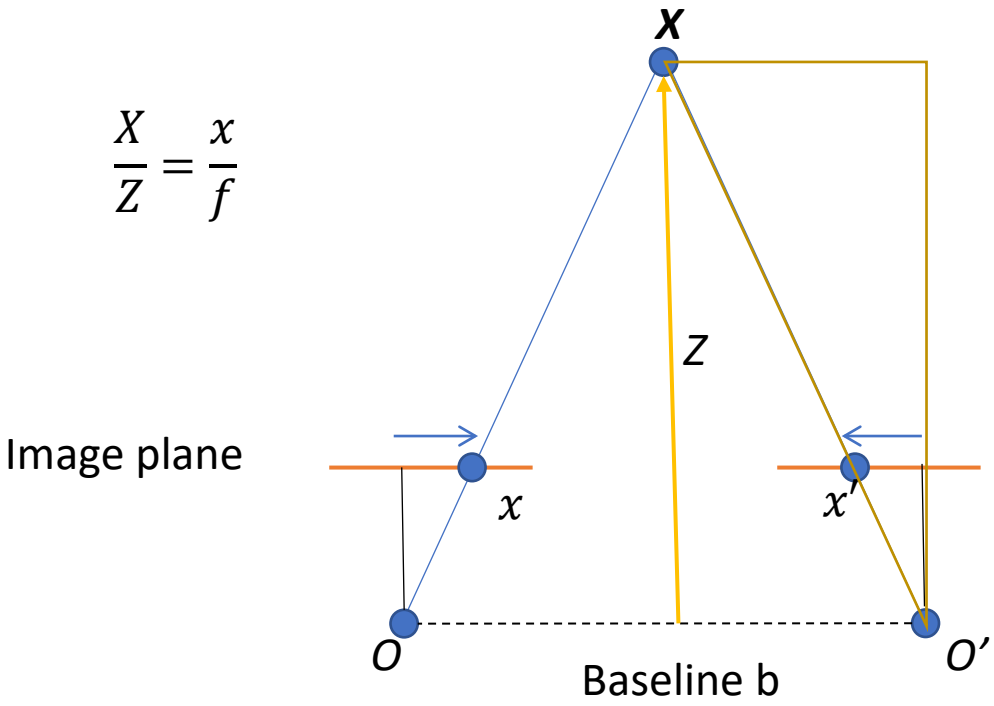

X
Z
Image plane
x
x'
O
O'
Baseline b

$$\frac{X}{Z} = \frac{x}{f}$$



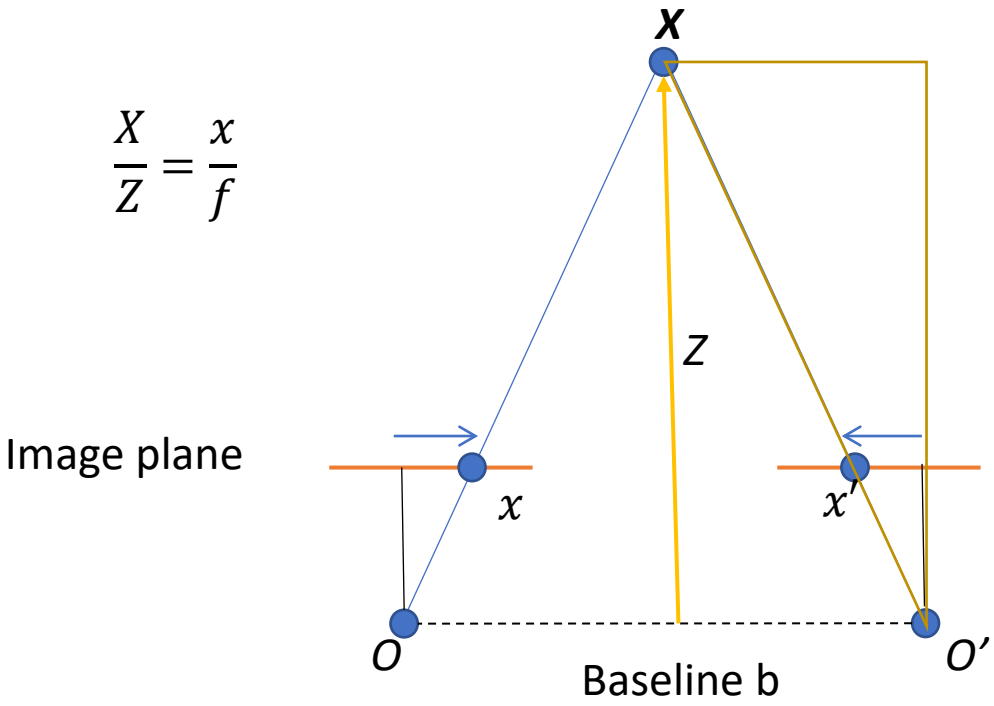
$$\frac{X}{Z} = \frac{x}{f}$$

$$\frac{b - X}{Z} = \frac{x'}{f}$$



$$\frac{X}{Z} = \frac{x}{f}$$

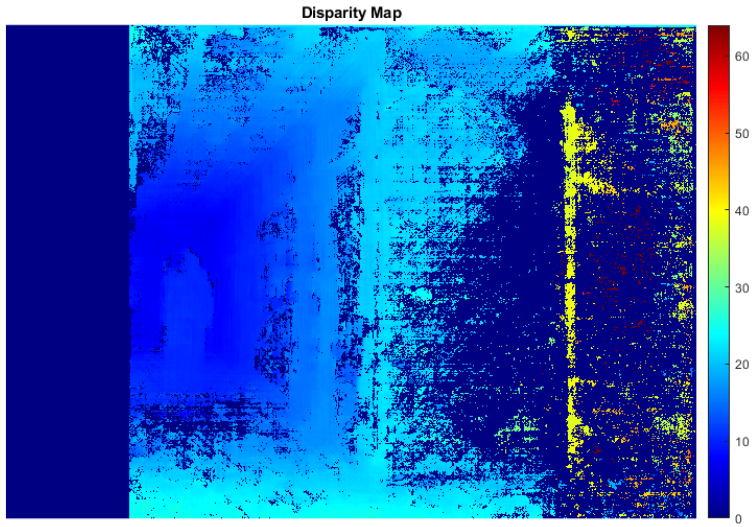
$$\frac{b - X}{Z} = \frac{x'}{f}$$



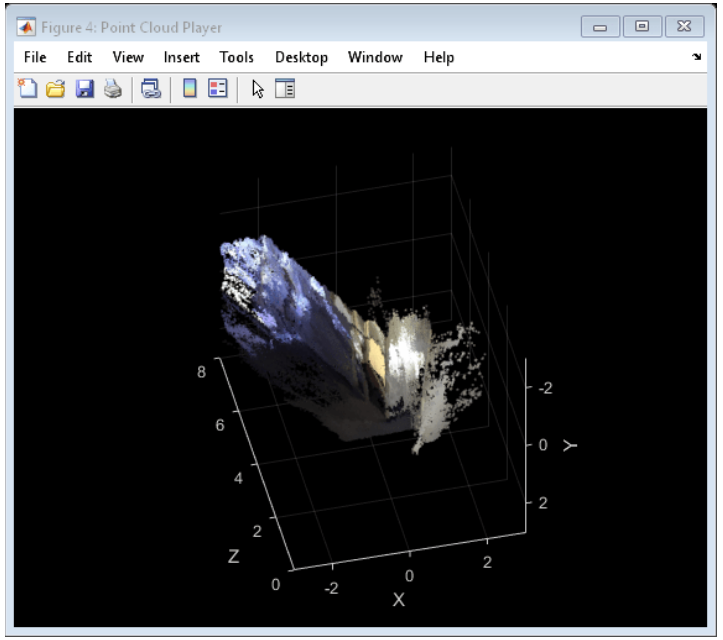
Disparity $d = x - x' = \frac{bf}{Z}$

Example

Rectified Images



Disparity Map



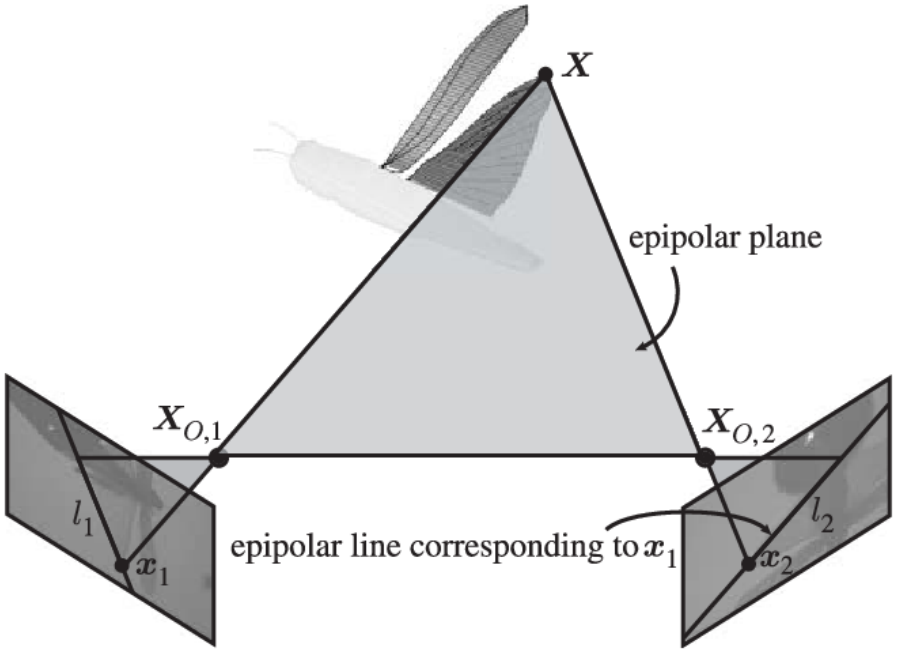
Reconstructed Points

Today’s Agenda

- Stereo Normal View
- Disparity Calculation
- Stereo Normal View Reconstruction
- Triangulation of calibrated camera

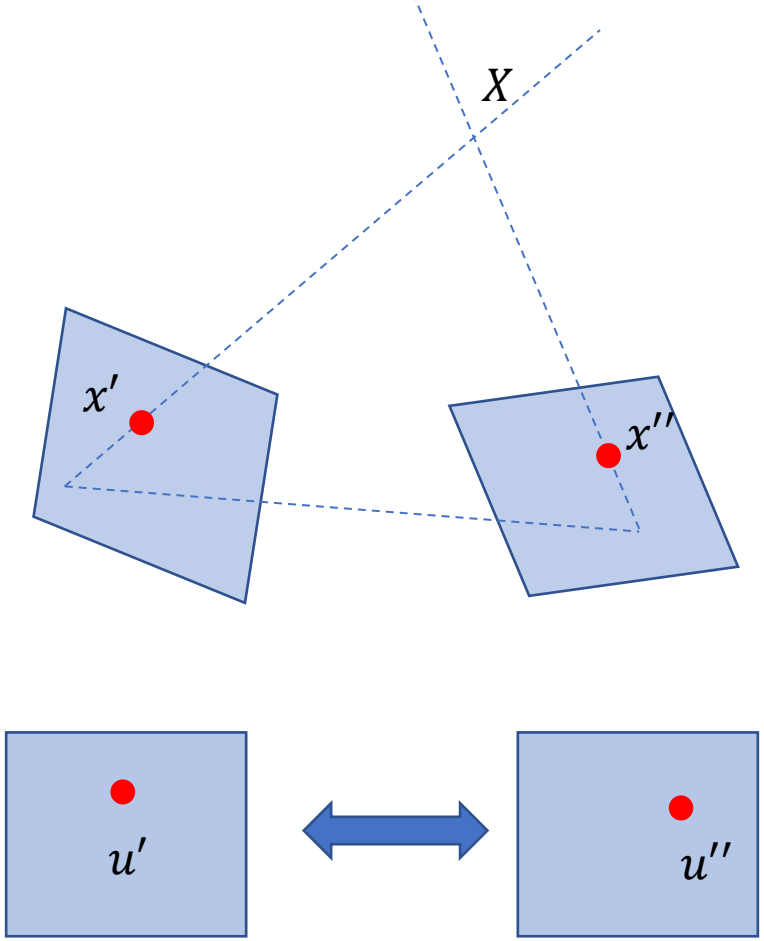
Reconstruction by Triangulation

- **Given**: A pair of cameras with known relative orientation
- **Aim**: To compute the points in 3D



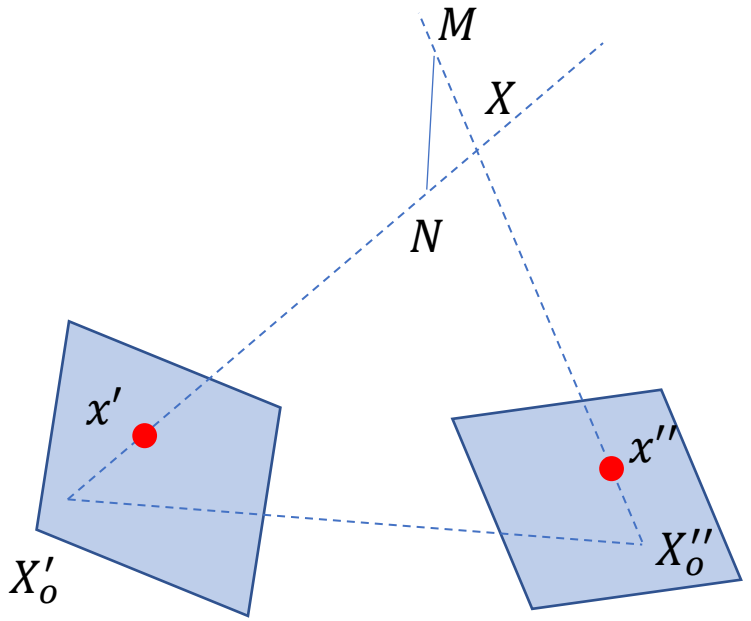
The 3D point Reconstruction

- Assume we know the camera matrices P’ and P’’ and the 2D correspondence between the image points from **X**
- To determine the 3D point **X**, rays are back-projected from the image points to determine their intersection.
- Problem ?



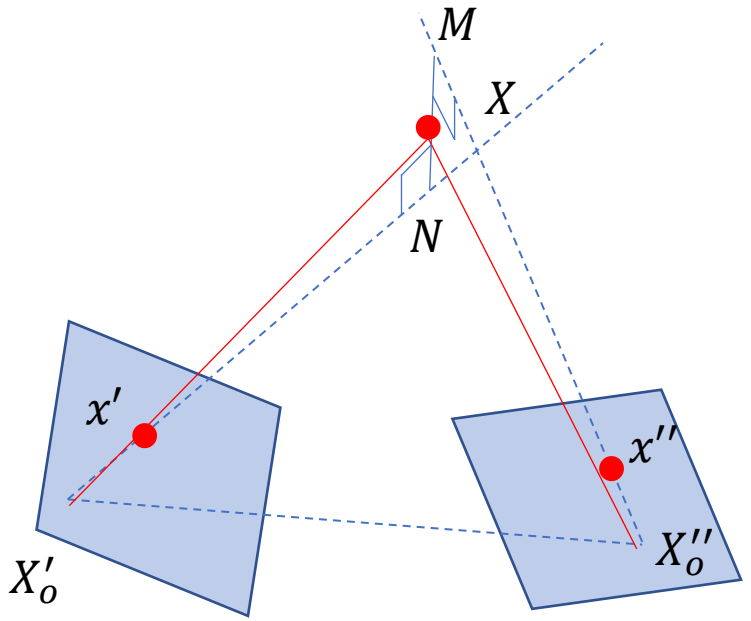
Finding the intersection in 3D

- Due to noise, the two ways in 3D will not intersect.
- We need to estimate the best solution
- We can express the back projection to 3D as two lines
  - $n(\lambda) = X'_o + \lambda \mathrm{R}'^T \mathrm{x}'$
  - $m(\mu) = X''_o + \mu \mathrm{R}''^T \mathrm{x}''$



Minimization 3D error

- The 2 lines
  - $n(\lambda) = X'_o + \lambda \mathrm{R}'^T \mathrm{x}'$
  - $m(\mu) = X''_o + \mu \mathrm{R}''^T \mathrm{x}''$
- Shortest distance is
  - $mn \perp MX''_o$ & $mn \perp NX'_o$
- Therefore we have
  - $(m-n) \cdot \mathrm{R}''^T \mathrm{x}'' = 0$
  - $(m-n) \cdot \mathrm{R}'^T \mathrm{x}' = 0$
- Two equations and two known.
- Solve for λ and μ
- Then mid-point of mn



3D Reconstruction from uncalibrated camera

- Linear Triangulation Method - This algorithm use two equations for the projection to solve for 3D points that has least squares error.

- In each image we have the projection

$$\mathrm{x} = \alpha \mathrm{PX} \qquad \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \alpha \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- use cross product $\mathrm{x} \times \mathrm{PX} = 0$ to remove the unknown factor α

Reconstruction from uncalibrated camera

Since

$$\mathrm{x}' \times \mathrm{P}'\mathrm{X} = 0$$

We have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathrm{p}^{1\mathrm{T}} \\ \mathrm{p}^{2\mathrm{T}} \\ \mathrm{p}^{3\mathrm{T}} \end{bmatrix} \mathrm{X} = 0$$

As the third term is a linear combination of the first two terms, we can eliminate it

$$\begin{bmatrix} \mathrm{y}'\mathrm{p}^{3\mathrm{T}} - \mathrm{p}^{2\mathrm{T}} \\ \mathrm{p}^{1\mathrm{T}} - x'\mathrm{p}^{3\mathrm{T}} \\ x'\mathrm{p}^{2\mathrm{T}} - \mathrm{y}'\mathrm{p}^{1\mathrm{T}} \end{bmatrix} \mathrm{X} = 0 \quad \Rightarrow \quad \begin{bmatrix} \mathrm{y}'\mathrm{p}^{3\mathrm{T}} - \mathrm{p}^{2\mathrm{T}} \\ \mathrm{p}^{1\mathrm{T}} - x'\mathrm{p}^{3\mathrm{T}} \end{bmatrix} \mathrm{X} = 0$$

Each point gives you ***two equations***

Minimizing the algebraic error

- Similarly we have for another image

$$\begin{bmatrix} \mathrm{y''p^{3T}} - \mathrm{p^{2T}} \\ \mathrm{p^{1T}} - x''\mathrm{p^{3T}} \end{bmatrix} \mathrm{X} = 0$$

Two points give you 4 equations

$$\begin{bmatrix} \mathrm{y'p^{3T}} - \mathrm{p^{2T}} \\ \mathrm{p^{1T}} - x'\mathrm{p^{3T}} \\ \mathrm{y''p^{3T}} - \mathrm{p^{2T}} \\ \mathrm{p^{1T}} - x''\mathrm{p^{3T}} \end{bmatrix} \mathrm{X} = 0$$

$$\begin{bmatrix} y'P_{31} - P_{21} & y'P_{32} - P_{22} & y'P_{33} - P_{23} & y'P_{34} - P_{24} \\ P_{11} - x'P_{31} & P_{12} - x'P_{32} & P_{13} - x'P_{33} & P_{14} - x'P_{34} \\ y''P_{31} - P_{21} & y''P_{32} - P_{22} & y''P_{33} - P_{23} & y''P_{34} - P_{24} \\ P_{11} - x''P_{31} & P_{12} - x''P_{32} & P_{13} - x''P_{33} & P_{14} - x''P_{34} \end{bmatrix} \mathrm{X} = 0$$

$$\mathrm{X} = [X \quad Y \quad Z \quad 1]^T$$

Solving $\mathrm{AX} = 0$ by SVD, $\mathbf{x} = \mathbf{v_4}/\mathrm{v_{44}}$ where $\mathrm{v_{44}}$ is the last element of $\mathbf{v_4}$

Representing in Screw Symmetric Matrix

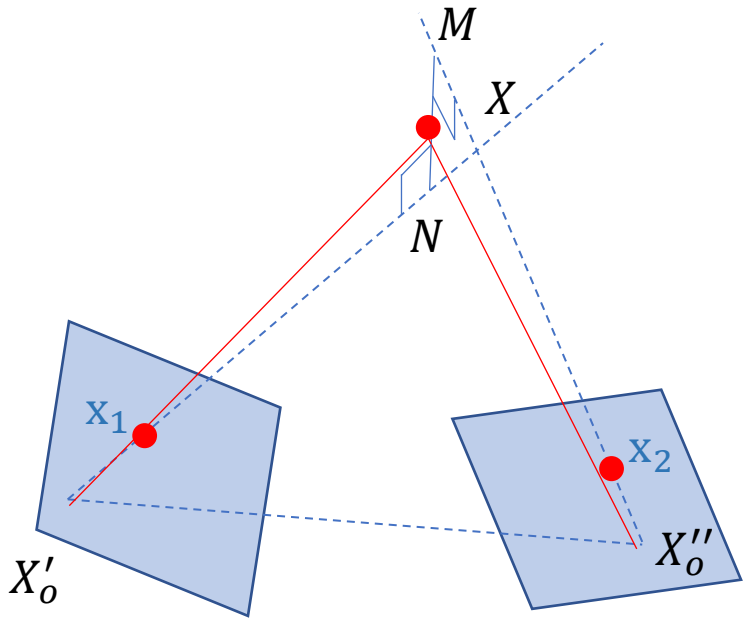
Minimization 3D error

$$x_1 \times P_1 X = 0$$
$$x_2 \times P_2 X = 0$$

- We can rewrite

$$\begin{pmatrix} [(x_1)_\times] P_1 \\ [(x_2)_\times] P_2 \end{pmatrix} X = 0$$

Solve the System by SVD



Reconstruction Ambiguity (Calibrated Camera)

- For calibrated camera (i.e. known $E = S_B R^T$, K, K')

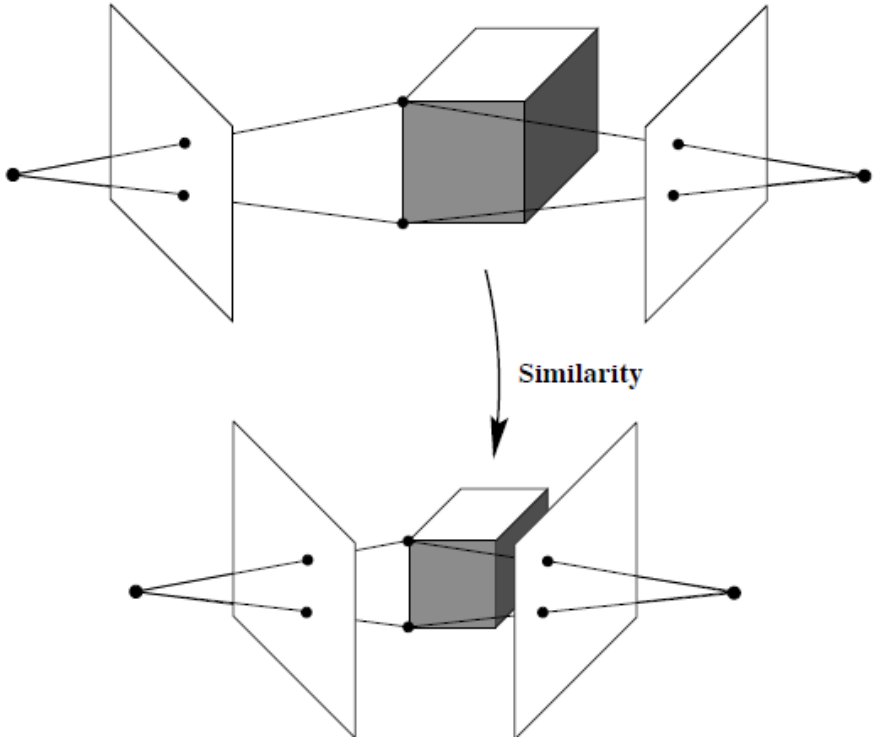


Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision’

Reconstruction Ambiguity (Calibrated Camera)

- Let $\mathrm{H_s}$ be similarity transform

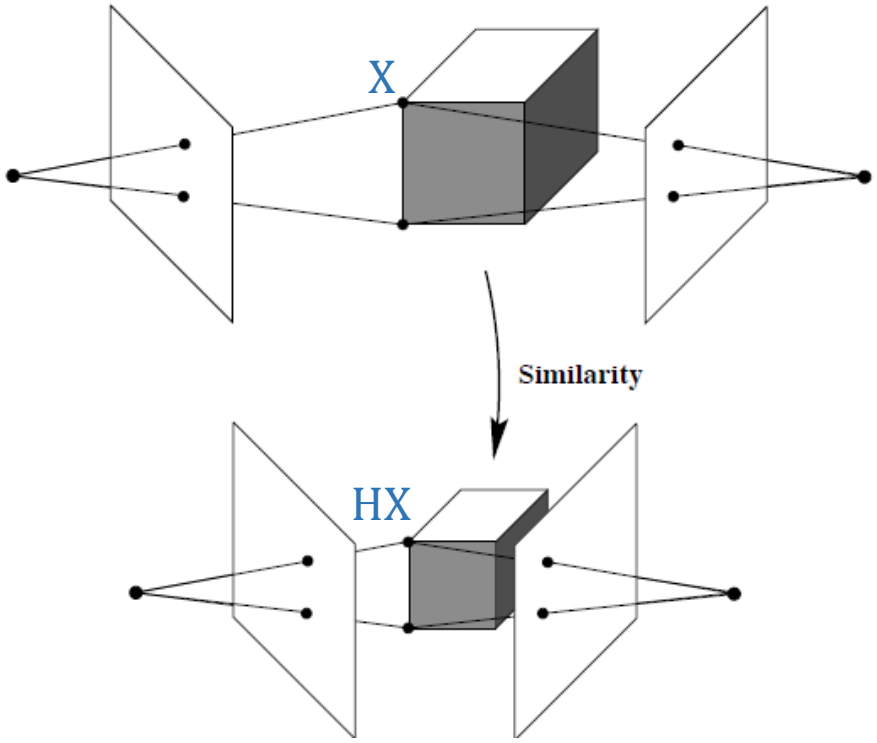
$$\mathrm{H_s} = \begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0^T & \lambda \end{bmatrix}$$

- The projection can be written

$$\mathrm{PX}_i = (\mathrm{PH_s}^{-1})(\mathrm{H_s X}_i)$$

- If P is decomposed in $\mathrm{P} = \mathrm{K}[\mathrm{R_p}|\mathrm{t_p}]$

$$\mathrm{PH_s}^{-1} = [\mathrm{R_p R}^{-1}|\mathrm{t}']$$



It is a valid projection matrix, it is also shown in (Longuet Higgins-81)

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision’

Example

- Once you have the *essential matrix* $\mathrm{E} = S_b\, R^T$, the *relative orientation* of two cameras can be recovered.

- *Translation* and *rotation* are as follow:

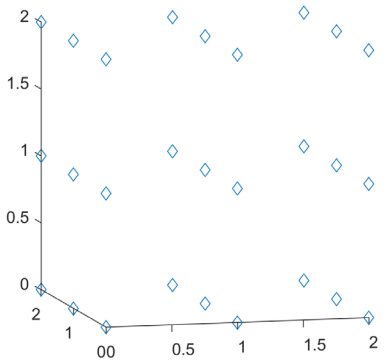
$$S_B = UZU^T \quad \text{or} \quad S_B = UZ^TU^T$$
$$R^T = UWV^T \quad \text{or} \quad R^T = UW^TV^T$$

- Since $S \times t = 0$, we can rewrite $S_B t = 0$, $t = u(3)\ or\ -u(3)$

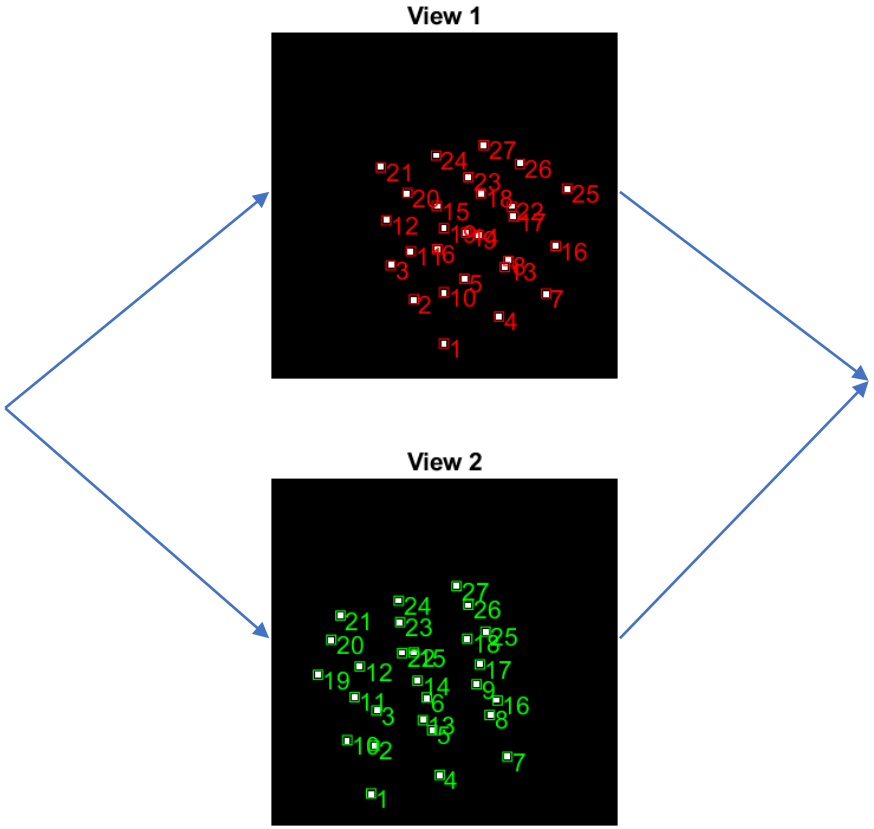
- The camera matrices are

$$\mathrm{P}_1 = [\mathrm{I}|0] \qquad \mathrm{P}_2 = [VWU^T|\mathrm{u}(3)\mathrm{or}-\mathrm{u}(3)]$$

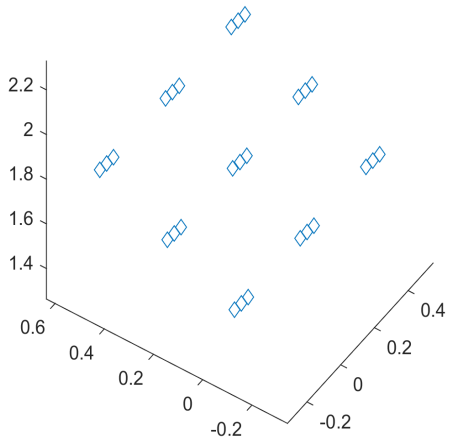
Example



Original Points



Projected Images



Reconstructed Points

Reconstruction Ambiguity (Uncalibrated Camera)

- For uncalibrated (unkwon K, K' and know F)
- Suppose $\mathrm{x}_i \leftrightarrow \mathrm{x}_i'$ is a set of correspondence, we have

$$\mathrm{x}_i'\mathrm{F}\mathrm{x}_i = 0$$

Decomposition of $\mathrm{F} \rightarrow (\mathrm{P}, \mathrm{P}')$is not unique

$$\mathrm{F} \rightarrow (\mathrm{P}, \mathrm{P}') \text{ or } \mathrm{F} \rightarrow (\mathrm{P}\mathrm{H}^{-1}, \mathrm{P}'\mathrm{H}^{-1})$$

The point X is reconstructed as HX

$$\mathrm{PX} = \mathrm{PH}^{-1}\mathrm{HX}$$

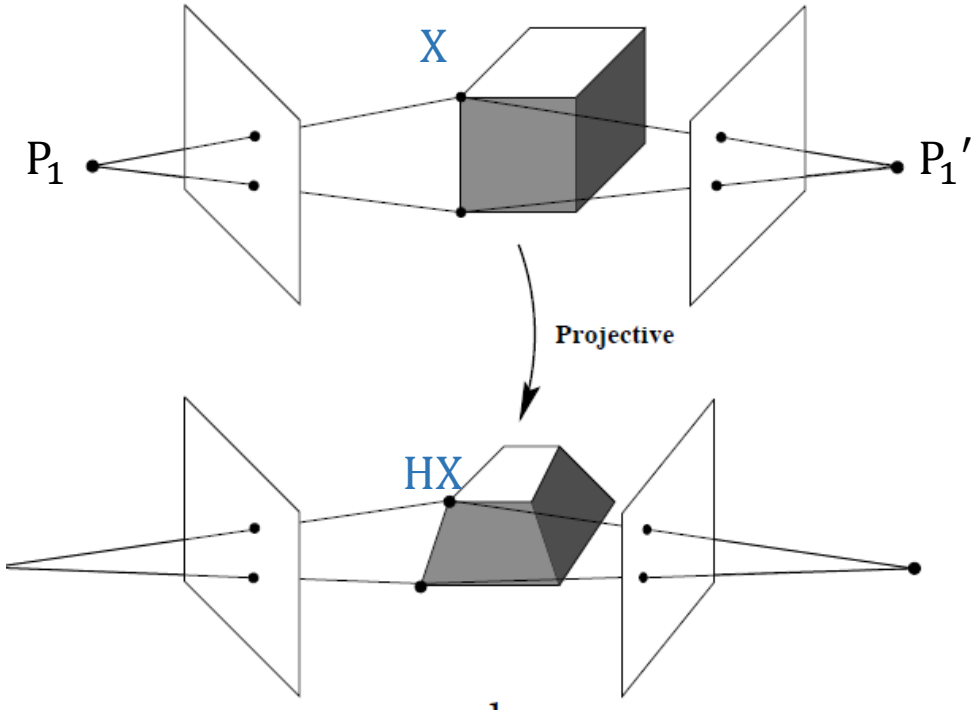
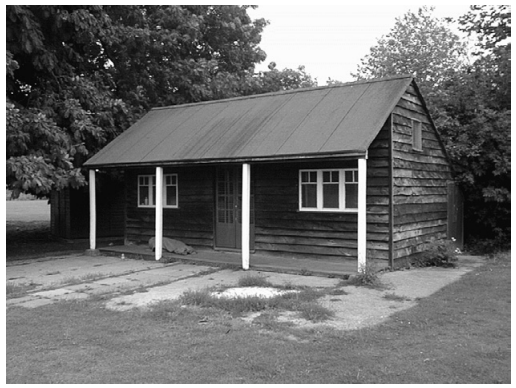
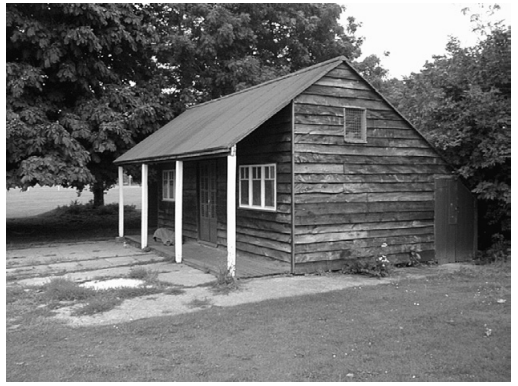


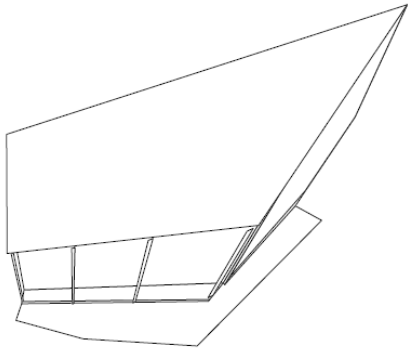
Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

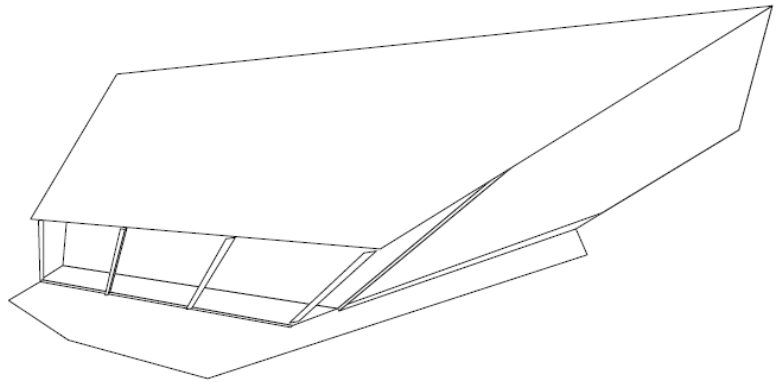
Reconstruction Ambiguity (Uncalibrated Camera)





Original Image Pair





Two views of 3D reconstructed Scene

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Stratified Reconstruction

- Aim: To upgrade the ***projective reconstruction*** to ***metric reconstruction***

- 2 steps
  - Start with projective reconstruction
  - Upgrade to affine reconstruction
  - Finally to metric reconstruction

- The information from the scene is required for the upgrade

From Projective to Affine Reconstruction

- The essence of affine reconstruction is to locate the ***plane at infinity*** by some means.

- Suppose we have determined the true ***plane at infinity*** π expressed as 4-vector in ***projective reconstruction***. In true reconstruction, $\pi = (0,0,0,1)^T$.

- We want to find the transformation $\mathrm{H} = \begin{bmatrix} \mathrm{I} & 0 \\ \pi^{\mathrm{T}} & \end{bmatrix}$ where

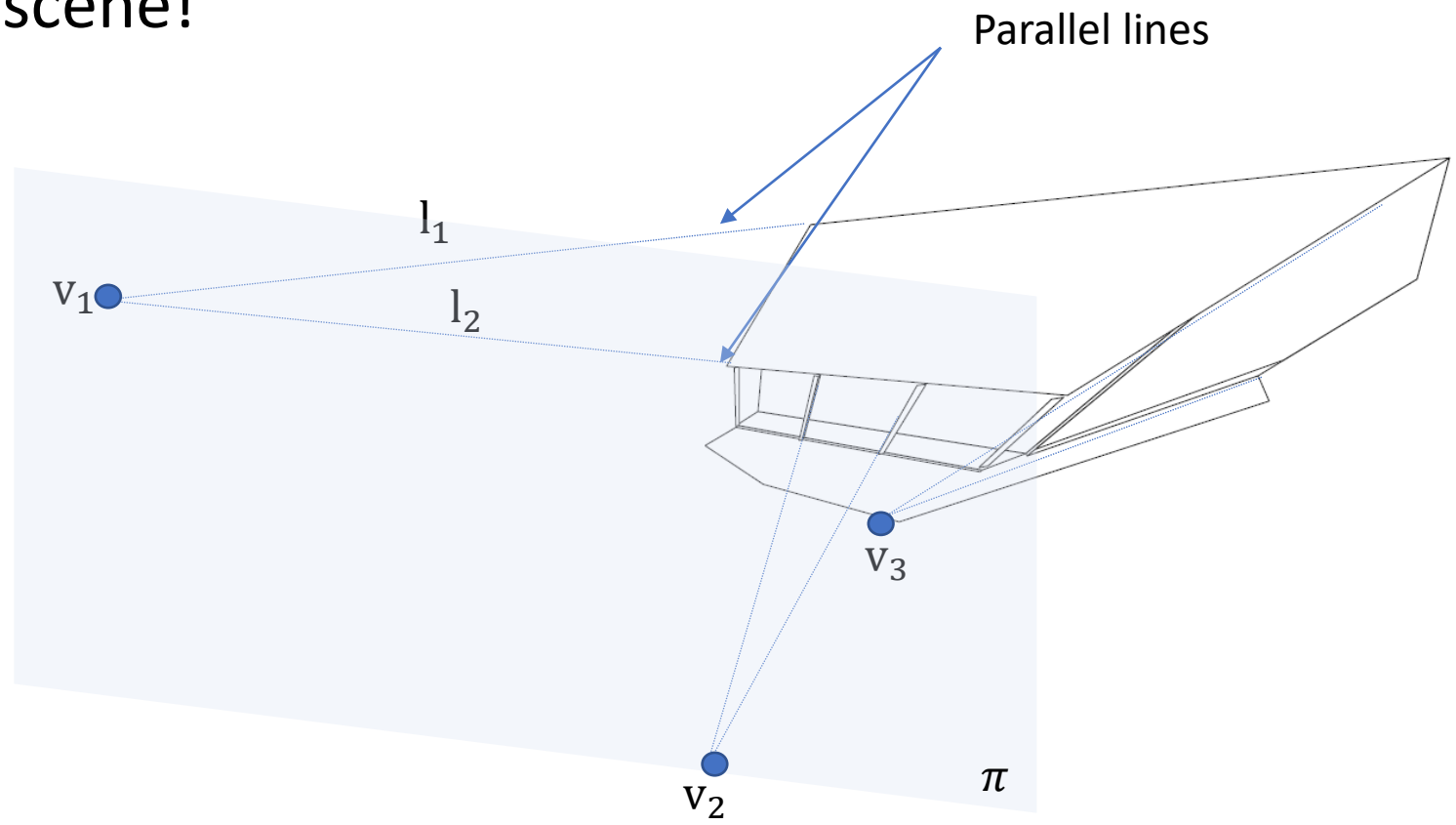
$$\mathrm{H}^{\mathrm{T}}\ (0,0,0,1)^T = \pi$$

Applying the transformation H to all point X will remove the projective distortion.

Point/Line/Plane at infinity

From Projective to Affine Reconstruction

- How to locate the ***plane at infinity*** ? Require knowledge from the scene!



Vanishing point $v_1 = l_1 \times l_2$

Plane π=$(v_1 \times v_2) \times (v_2 \times v_3)$

π is the plane containing v_1, v_3 and v_3

π=$(v_1 \times v_2) \times (v_2 \times v_3)$

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Transform π to π_∞

- i.e. $H^{-T}\pi = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \pi_\infty$

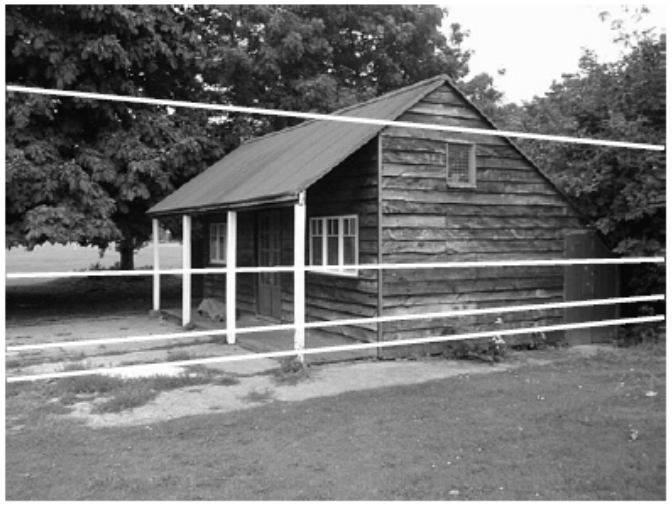
- $H = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\\pi_1 & \pi_2 & \pi_3 & \pi_4\end{bmatrix}$

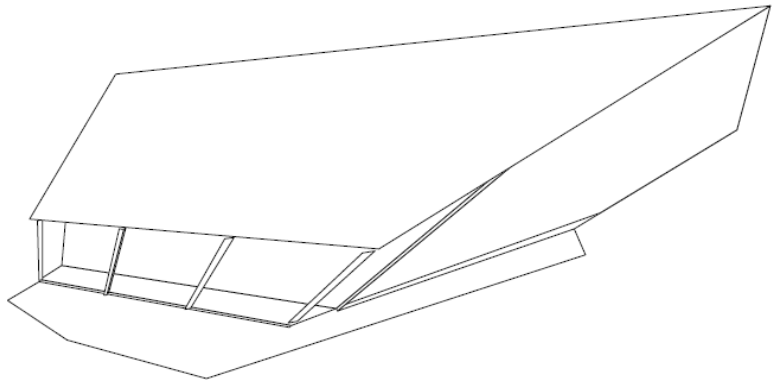
- Verification

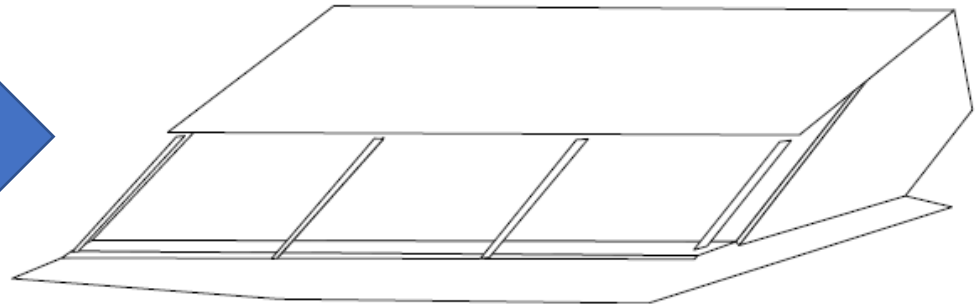
- $H^T\pi = \begin{bmatrix}1 & 0 & 0 & \pi_1\\0 & 1 & 0 & \pi_2\\0 & 0 & 1 & \pi_3\\0 & 0 & 0 & \pi_4\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}\pi_1\\\pi_2\\\pi_3\\\pi_4\end{bmatrix}$

- Use H to transform all points

From Projective to Affine Reconstruction







From Affine to Metric Reconstruction

- The key to metric reconstruction is the identification of ***absolute conic*** ω
- Suppose we have affine reconstruction from the camera $\mathrm{P} = [\mathrm{M} \mid \mathrm{m}]$
- Once ω is known, the ***affine reconstruction*** of the image can be upgraded to ***metric reconstruction*** by 3D transformation

$$\mathrm{H} = \begin{bmatrix} \mathrm{A}^{-1} & \\ & 1 \end{bmatrix} \qquad \text{where} \qquad \mathrm{X_m} = \mathrm{H}\mathrm{X_A}$$

Where A is obtained by Cholesky factorization from $\mathrm{A}\mathrm{A}^{\mathrm{T}} = (\mathrm{M}^{\mathrm{T}}\omega\mathrm{M})^{-1}$

From Affine to Metric Reconstruction

- For the metric camera, the camera matrix is defined as $\mathrm{P_M} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]$

- The affine camera matrix $\mathrm{P_A}[\mathrm{M} \mid \mathrm{m}]$ is transformed to $\mathrm{P_M} = \mathrm{P_A H^{-1}} = [\mathrm{M_m} | \mathrm{m_m}]$ where $\mathrm{H^{-1}}$ is of the form

$$\mathrm{H} = \begin{bmatrix} \mathrm{A^{-1}} & 0 \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$$

$$\mathrm{P_M} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}] = [\mathrm{MA} \mid \mathrm{m}]$$

From Affine to Metric Reconstruction

- By the relationship,

$$\mathrm{P_M} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}] = [\mathrm{MA} \mid \mathrm{m}]$$

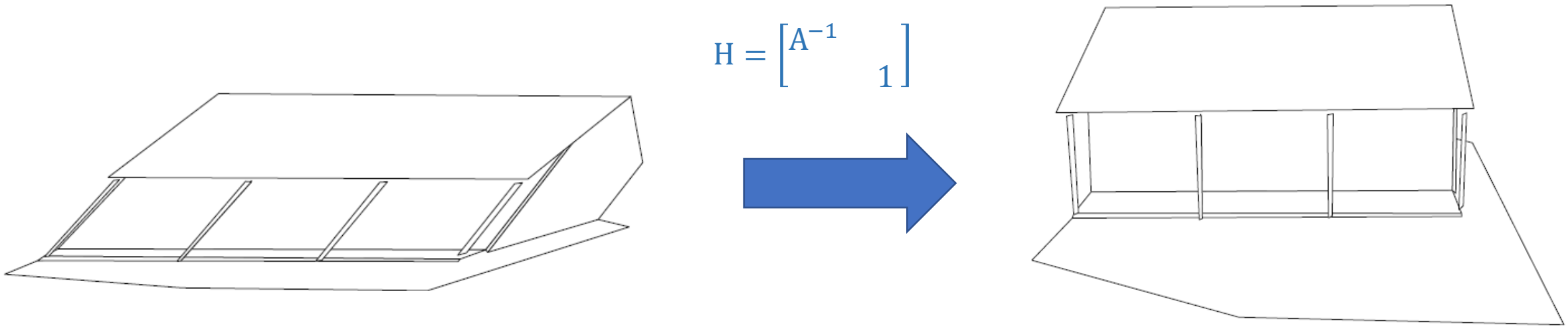
- We have
  - $\mathrm{MA}(\mathrm{MA})^T = \mathrm{KR}(\mathrm{KR})^T = \mathrm{KK}^T$
  - $\mathrm{MAA^TM}^T$ = KK^T Let $\omega^{-1} = \mathrm{KK}^T$

- Therefore
  - $\mathrm{AA^T} = (\mathrm{M^T}\omega\mathrm{M})^{-1}$

This approach to *metric reconstruction* relies on identifying the *image of the absolute conic*.

Image of Absolute Conic ω section 10.4.2 PP 273

Metric Reconstruction



Metric Reconstruction Result.

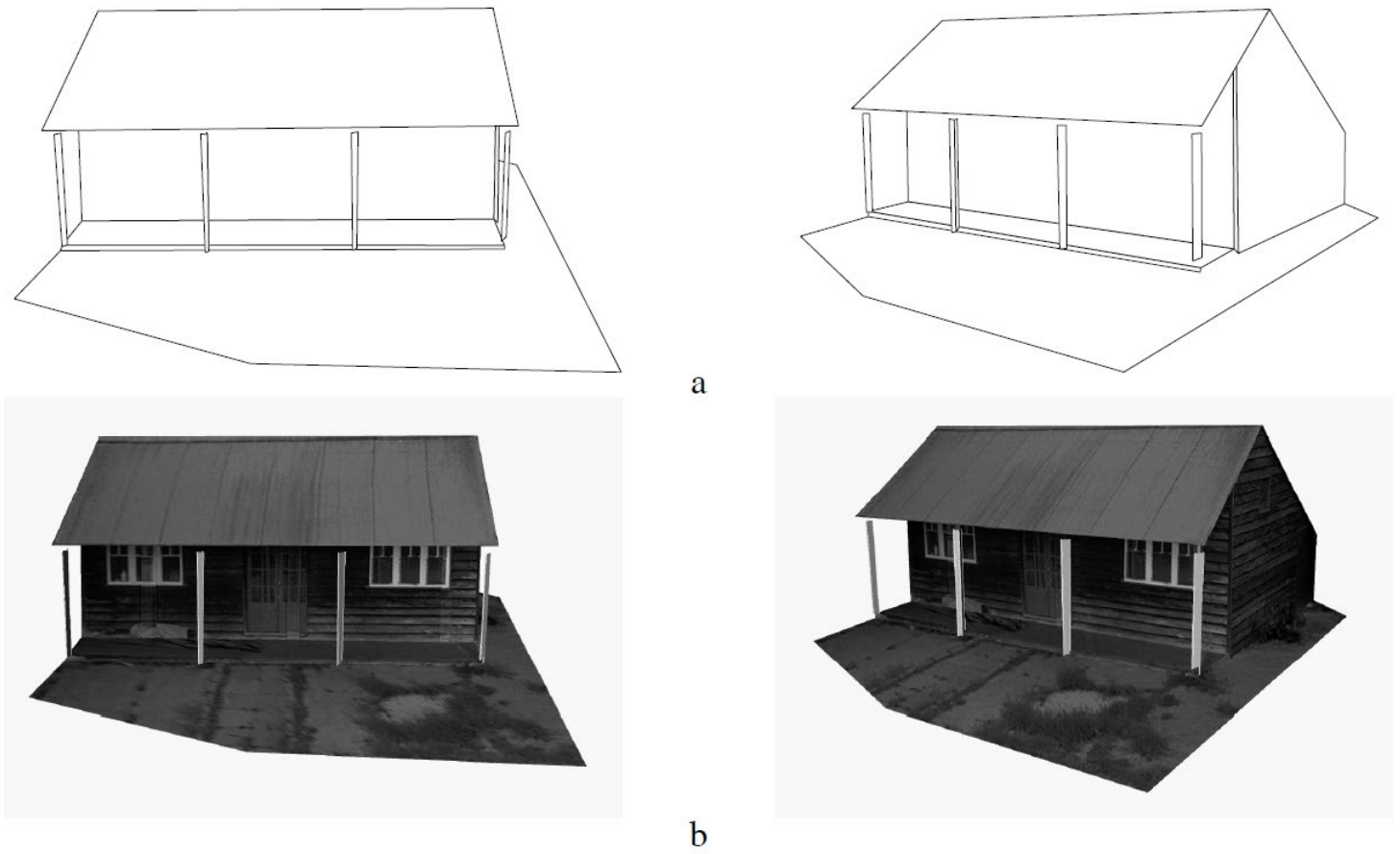


Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Summary of Stratifield Reconstruction

- Given: *2 images* from *uncalibrated cameras*
- Goal: *Metric Reconstruction*

1. Search for the correspondence (for instance, SIFT)
2. Estimate the *Fundamental Matrix* F (By Longuet-Higgins)
3. Estimate a pair of *camera matrix* P and P’
4. Make use of the scene information such as parallel lines to find the transformation of $\pi \rightarrow \pi_\infty$ to upgrade from *projective* to *affine ambiguity*
5. Further upgrade to metric reconstruction using *image of absolute conic* (IAC)

Direct Reconstruction – Using Ground Truth

- It is possible to jump directly from a *projective reconstruction* to a *metric reconstruction* if *"ground control points"* (that is points with known 3D locations in a Euclidean world frame) are given.

- Suppose we have a set of *n* ground control points $\{\mathrm{X}_{\mathrm{E}i}\}$ which are imaged at $\mathrm{x}_i \leftrightarrow \mathrm{x}_i'$. We will use this points to transform the projective reconstruction to metric

- The 3D location $\{\mathrm{X}_i\}$ of the control point in the projective reconstruction may be computed from their image correspondences $\mathrm{x}_i \leftrightarrow \mathrm{x}_i'$. We have.

$$\mathrm{X}_{\mathrm{E}i} = \mathrm{H}\mathrm{X}_i, i = 1, \dots, n.$$

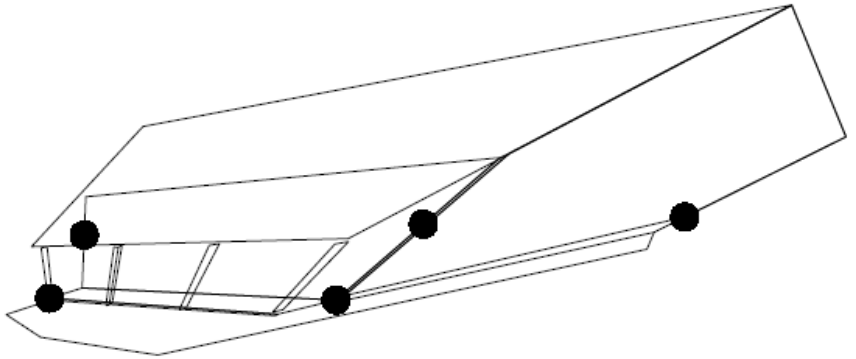
- Each point correspondence provides 3 linearly independent equation. H has 15 degree of freedom. We can compute H *for* $\mathrm{n} > 5$

Direct Reconstruction – Using Ground Truth

- Alternatively, one may bypass the computation of the X_i and compute H by DLT

$$\mathrm{x}_i = PH^{-1}\mathrm{X}_{\mathrm{E}i}$$

- Once H has been computed it may be used to transform the cameras P, P' of the projective reconstruction to true Euclidean counterparts.



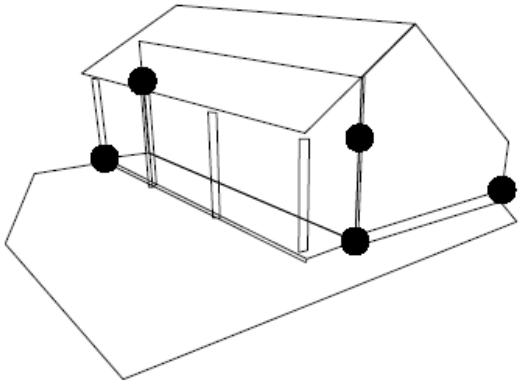
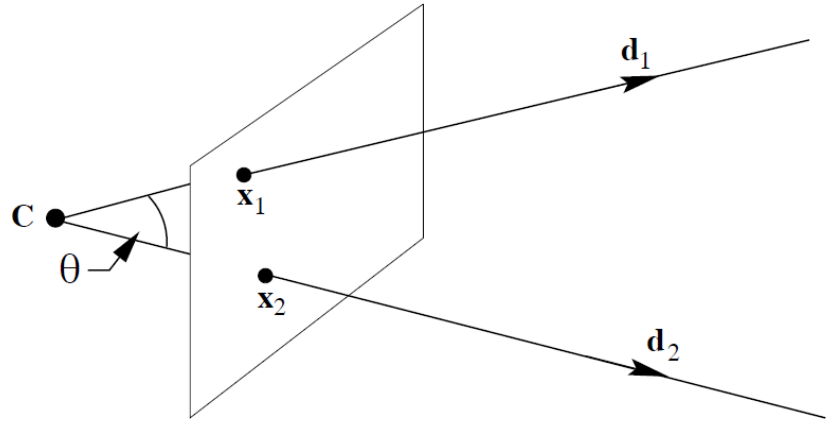


Image Credit: R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision"

Appendix – Image of Absolute Conic ω

- *Image of Absolute Conic (IAC)* $\omega = K^{-T}K^{-1} = (KK^T)^{-1}$
- Angle between two rays

$$\cos\theta = \frac{(\mathbf{x}_1^{\mathrm{T}}\omega\mathbf{x}_2)}{\sqrt{(\mathbf{x}_1^{\mathrm{T}}\omega\mathbf{x}_1)(\mathbf{x}_2^{\mathrm{T}}\omega\mathbf{x}_2)}}$$



- ω is a *symmetric matrix* $\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{23} & \omega_{33} \end{bmatrix}$
- f k is *zero skew* and *square pixels*. We have $\begin{bmatrix} \omega_{11} & 0 & \omega_{13} \\ 0 & \omega_{11} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix}$

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Finding ω

- Select sets of orthogonal parallel lines
- Find the vanishing points for all the parallel lines $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$
- We can solve for ω by the orthogonal relationship.

$$\mathrm{v}_i{}^T \omega \mathrm{v}_j$$

back

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Reference

- Wolfgang Forstner and Bernhard P. Wrobel, “Photogrammetric Computer Vision”, Chapter 13

- R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”, Chapter 7 & 8