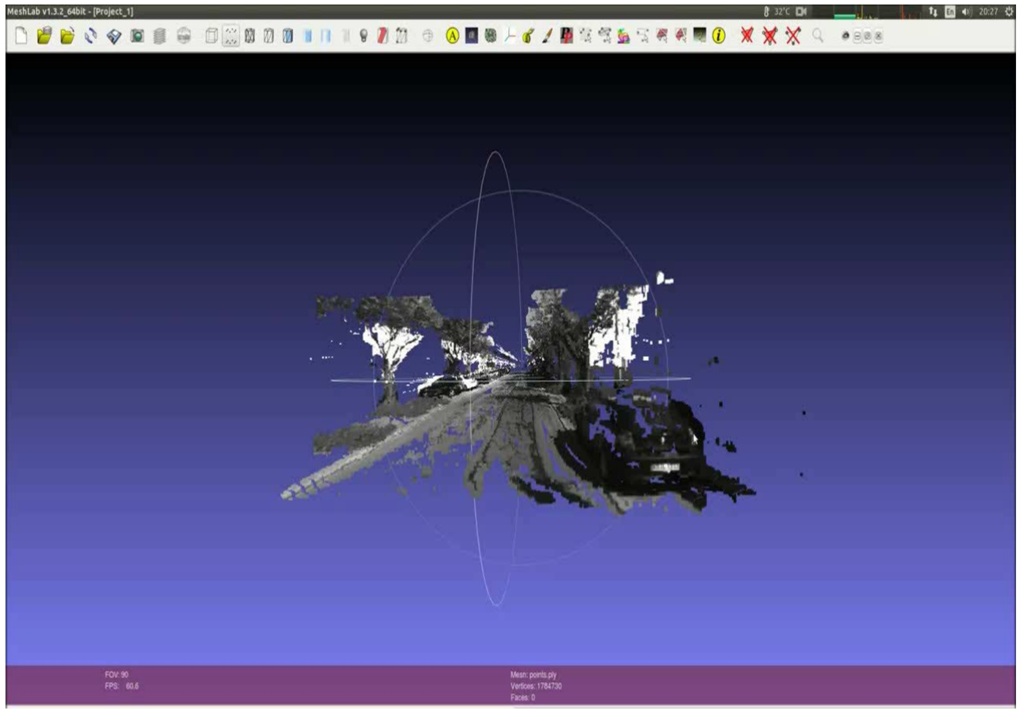
Vision Algorithms for Mobile Robotics

Lecture 07
Multiple View Geometry 1

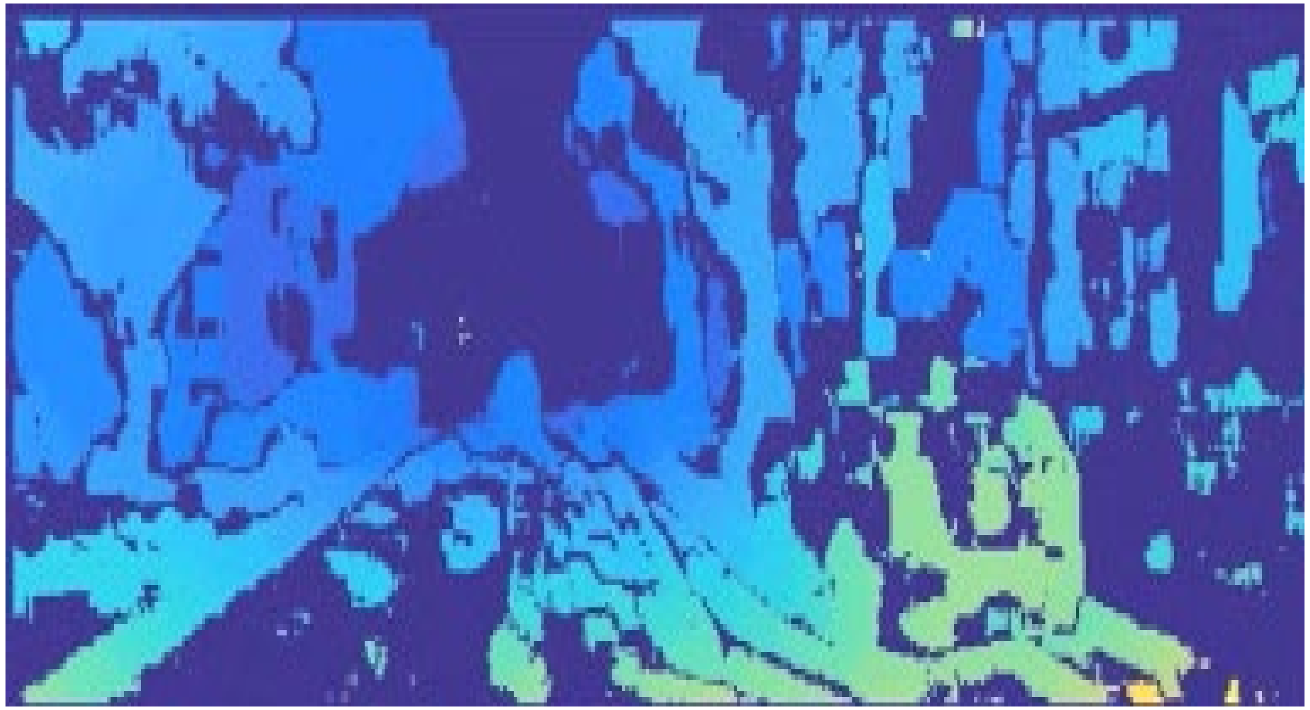
Davide Scaramuzza
http://rpg.ifi.uzh.ch

Lab Exercise 5 – Today

Stereo vision: rectification, epipolar matching, disparity, triangulation



3D point cloud



Disparity map (cold= far, hot=close)

I will be away on November 11th. The lecture will be replaced by a video recording. Q&A session at exercise.

23.09.2021	**Lecture 01 - Introduction to Computer Vision and Visual Odometry** **No Exercise today.**
30.09.2021	**Lecture 02 - Image Formation: perspective projection and camera models** **Exercise 01- Augmented reality wireframe cube**
07.10.2021	**Lecture 03 - Camera Calibration** **Exercise 02 - PnP problem**
14.10.2021	**Lecture 04 - Filtering & Edge detection** **No Exercise today.**
21.10.2021	**Lecture 05 - Point Feature Detectors, Part 1** **Exercise 03 - Harris detector + descriptor + matching**
28.10.2021	**Lecture 06 - Point Feature Detectors, Part 2 – Scaramuzza away – Video recording** **Exercise 04 - SIFT detector + descriptor + matching + Q&A session**
04.11.2021	**Lecture 07 - Multiple-view geometry 1** **Exercise 05 - Stereo vision: rectification, epipolar matching, disparity, triangulation**
11.11.2021	**Lecture 08 - Multiple-view geometry 2 – Scaramuzza away – Video recording** **Exercise 06 - Eight-Point Algorithm + Q&A session**
18.11.2021	**Lecture 09 - Multiple-view geometry 3** **Exercise 07 - P3P algorithm and RANSAC**
25.11.2021	**Lecture 10 - Multiple-view geometry 4** **Exercise session: Intermediate VO Integration**
02.12.2021	**Lecture 11** **Optical Flow and KLT Tracking** **Exercise 08 - Lucas-Kanade tracker**
09.12.2021	**Lecture 12a (1st hour) - Place recognition** **Lecture 12b (2nd hour) - Dense 3D Reconstruction and Place recognition** **Lecture 12c (3rd and 4th hour, replaces exercise) - Deep Learning Tutorial** **Optional Exercise on Place Recogntion**
16.12.2021	**Lecture 13 - Visual inertial fusion** **Exercise 09 - Bundle Adjustment**
23.12.2021	**Lecture 14 - Event based vision** **Exercise session: Final VO Integration**

Multiple View Geometry

San Marco square, Venice

14,079 images, 4,515,157 points

Agarwal, Snavely, Simon, Seitz, Szeliski, *Building Rome in a Day*, International Conference on Computer Vision (ICCV), 2009. PDF, code, datasets

Most influential paper of 2009

State of the art software: COLMAP

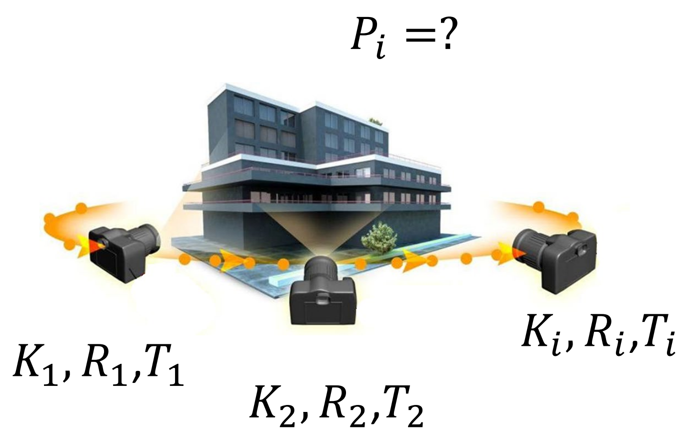
Multiple View Geometry

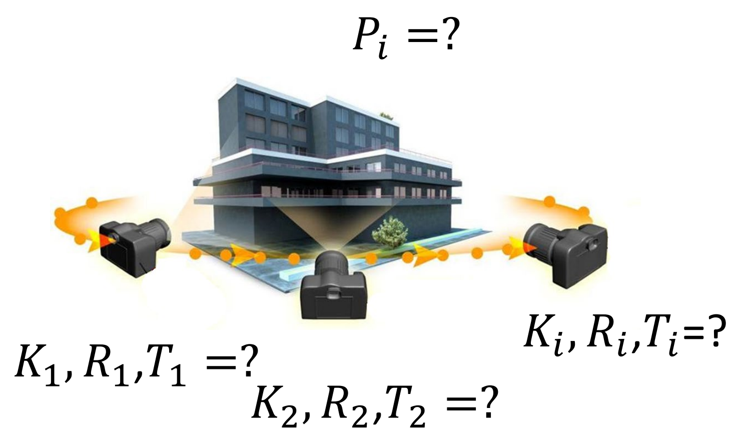
3D reconstruction from multiple views:

- **Assumptions**: K, T and R are known.
- **Goal**: Recover the 3D structure from images

Structure From Motion:

- **Assumptions**: none (K, T, and R are unknown).
- **Goal**: Recover simultaneously 3D scene structure and camera poses (up to scale) from multiple images





2-View Geometry

Depth from stereo (i.e., stereo vision):

- **Assumptions**: K, T and R are known.
- **Goal**: Recover the 3D structure from two images

2-view Structure From Motion:

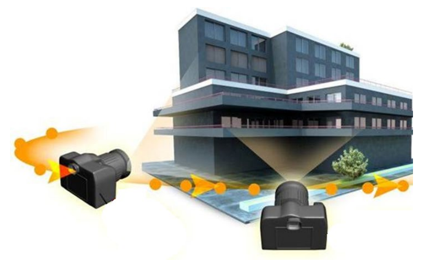
- **Assumptions**: none (K, T, and R are unknown).
- **Goal**: Recover simultaneously 3D scene structure and camera poses (up to scale) from two images

$P_i = ?$

K_1, R_1, T_1

K_2, R_2, T_2

$P_i = ?$



$K_1, R_1, T_1 = ?$

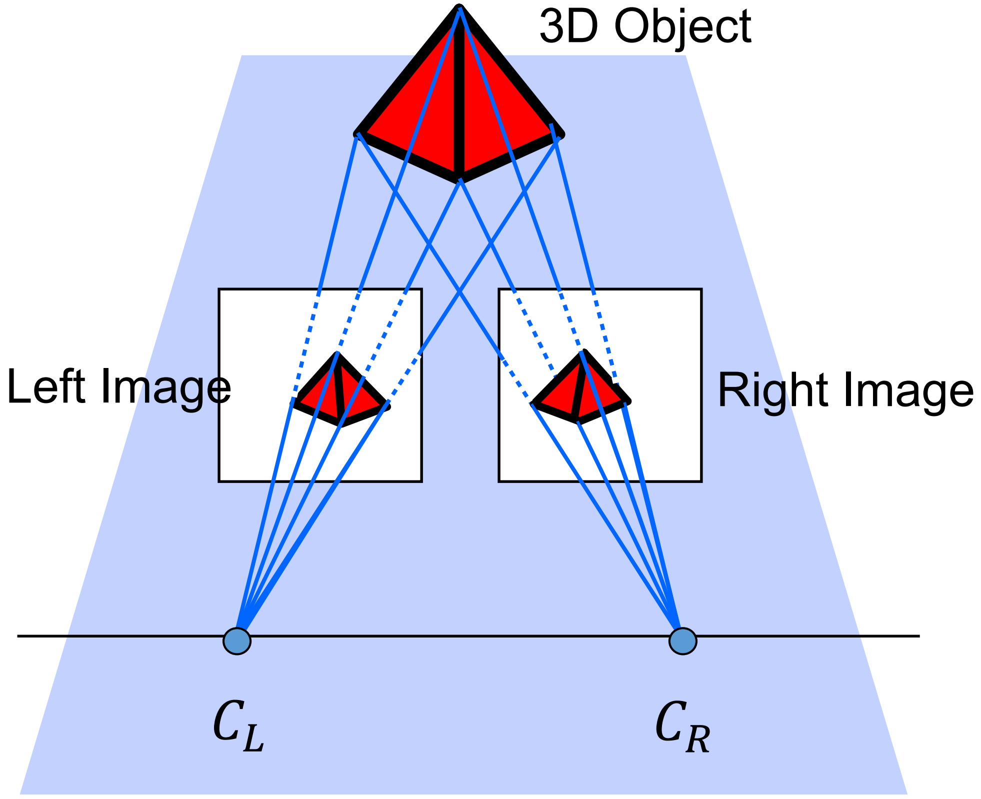
$K_2, R_2, T_2 = ?$

Today’s outline

- Stereo Vision
- Epipolar Geometry

Depth from Stereo

Goal: **recover the 3D structure** by computing **the intersection of corresponding rays**



The Human Binocular System

- **Stereopsys** is the principle by which our brain allows us to perceive depth from the left and right images
- Images project on our retinas up-side-down but our brains lets us perceive them as **straight**. Radial distortion is also removed. This process is called **rectification**

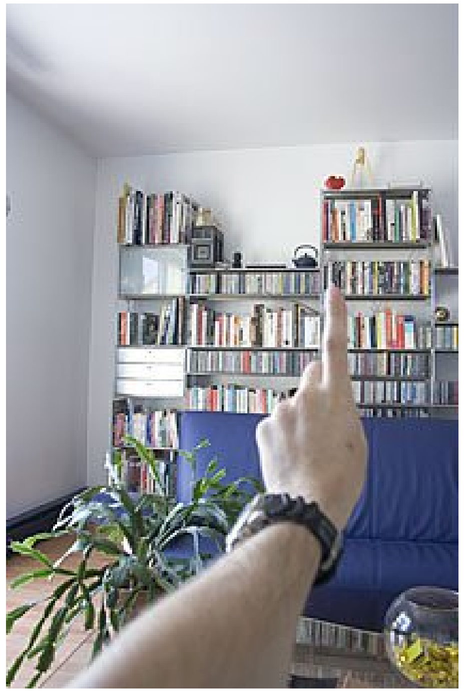


Image from the left eye

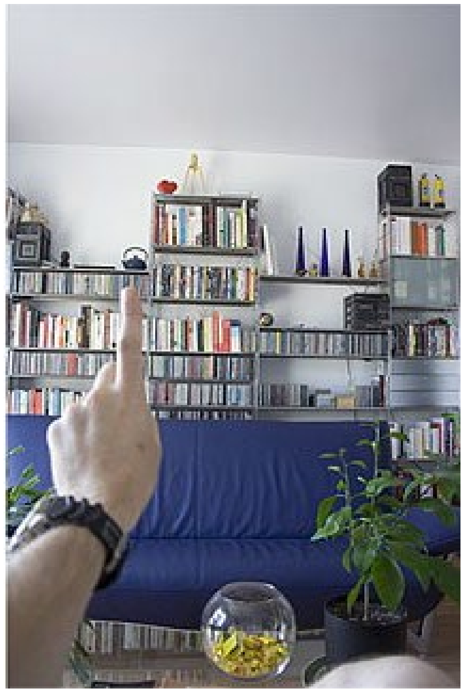
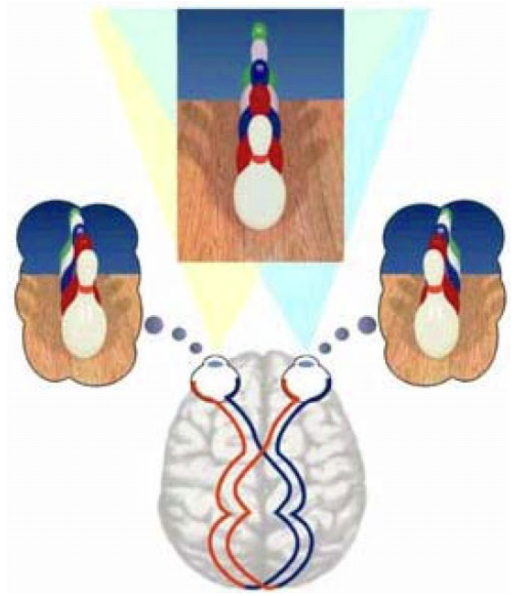
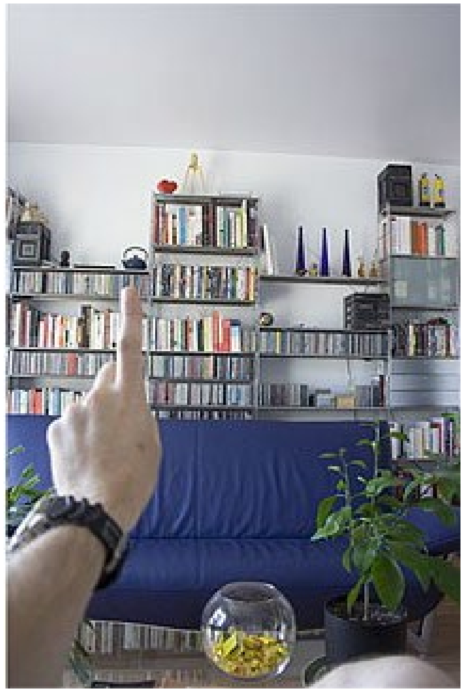


Image from the right eye



The Human Binocular System

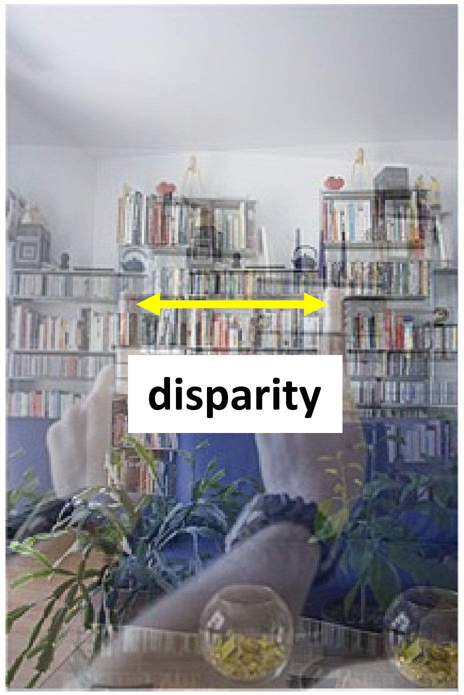
- **Stereopsys** is the principle by which our brain allows us to perceive depth from the left and right images
- Images project on our retinas up-side-down but our brains lets us perceive them as **straight**. Radial distortion is also removed. This process is called **rectification**



Make a simple test:
1. Fix an object
2. Open and close alternatively the left and right eyes.
- The horizontal displacement is called **disparity**
- The **smaller the disparity**, **the farther** the **object**

The Human Binocular System

- **Stereopsys** is the principle by which our brain allows us to perceive depth from the left and right images
- Images project on our retinas up-side-down but our brains lets us perceive them as **straight**. Radial distortion is also removed. This process is called **rectification**



Make a simple test:
1. Fix an object
2. Open and close alternatively the left and right eyes.
- The horizontal displacement is called **disparity**
- The **smaller the disparity**, **the farther** the **object**

The Human Binocular System

- **Stereopsys** is the principle by which our brain allows us to perceive depth from the left and right images
- Images project on our retinas up-side-down but our brains lets us perceive them as **straight**. Radial distortion is also removed. This process is called **rectification**
- What happens if you wear a pair of mirrors for a week?

An early experiment in “perceptual plasticity” was conducted by Psychologist George Stratton in 1896. He used his inverted vision goggles, over a period of 8 days, and over time adapted to the point where he was able to function normally.

https://en.wikipedia.org/wiki/George_M._Stratton#Wundt's_lab_and_the_inverted-glasses_experiments

Stereo Vision

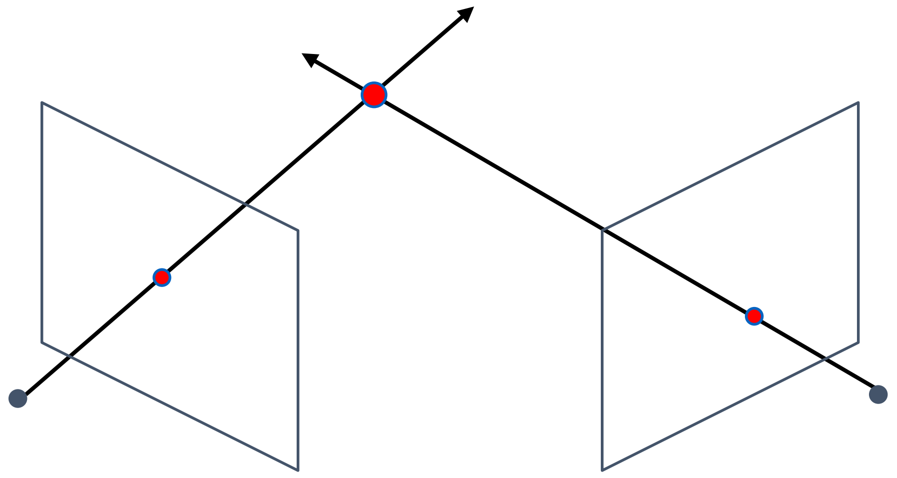
- Triangulation
  - Simplified case
  - General case
- Correspondence problem
- Stereo rectification

Intel RealSense D455 stereo camera:
uses stereo and structured infrared light for depth estimation
https://www.intelrealsense.com/stereo-depth/

Intel RealSense T265 stereo camera for visual-inertial odometry and SLAM:
https://www.intelrealsense.com/tracking-camera-t265/

Stereo Vision

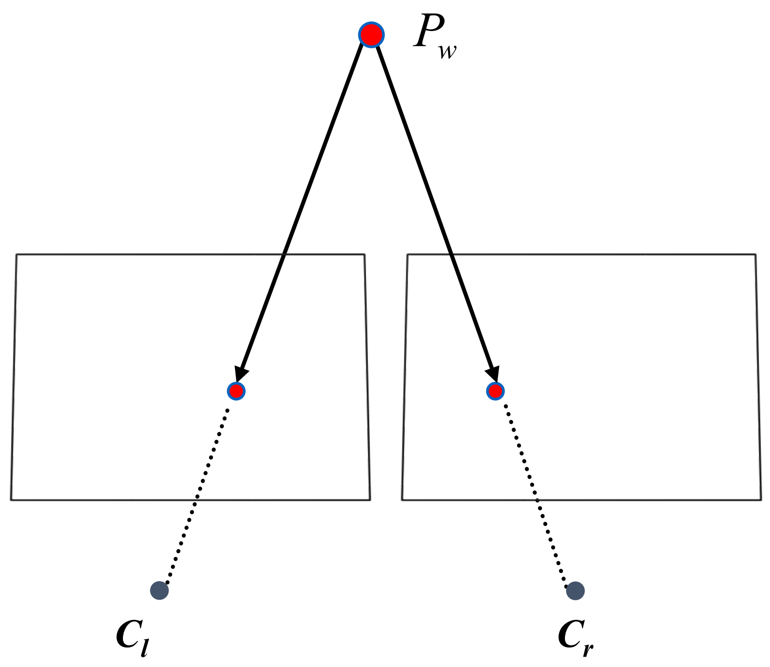
- **Goal**: find an expression of the 3D point coordinates as a function of the 2D image coordinates
- **Assumptions**:
  - **cameras are calibrated**: both intrinsic and extrinsic parameters are known
  - **point correspondences** are given



Stereo Vision

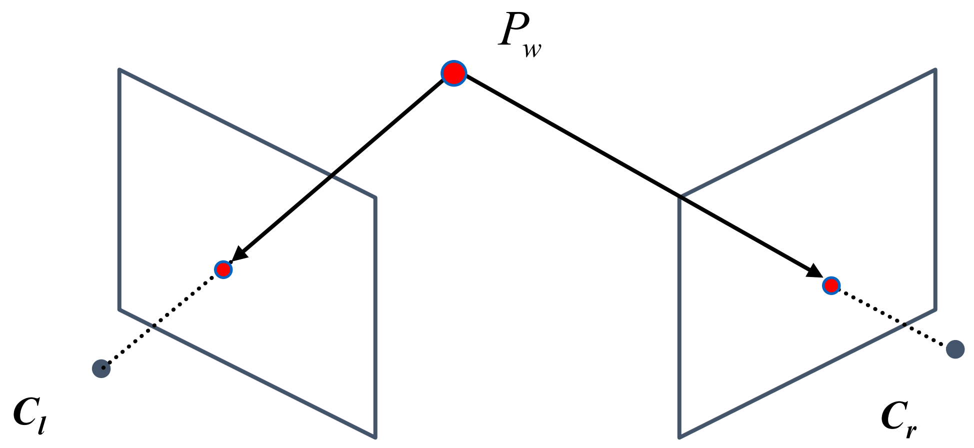
Simplified case

(identical cameras and aligned)



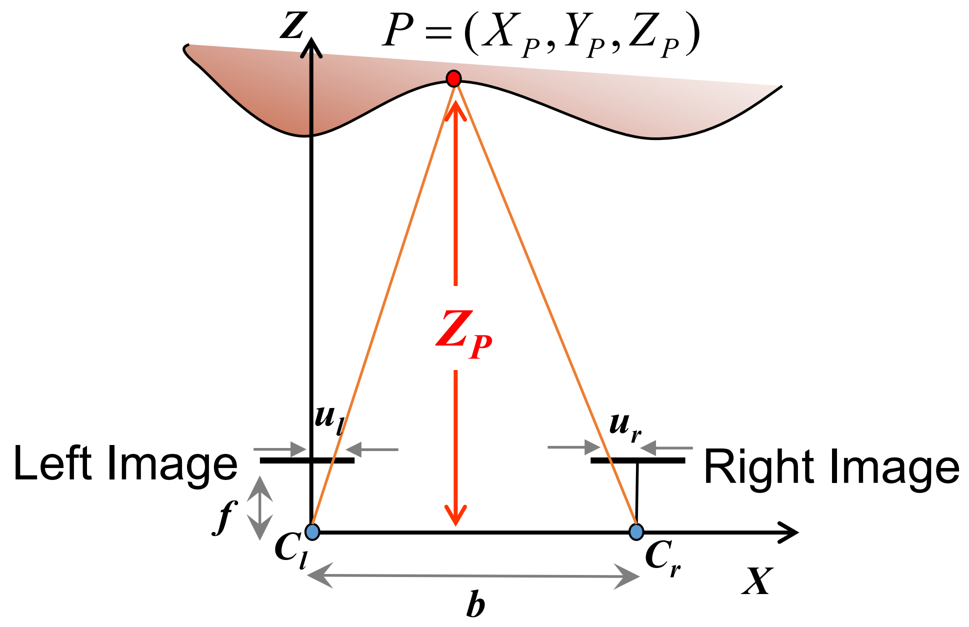
General case

(non identical cameras and not aligned)



Stereo Vision - Simplified Case

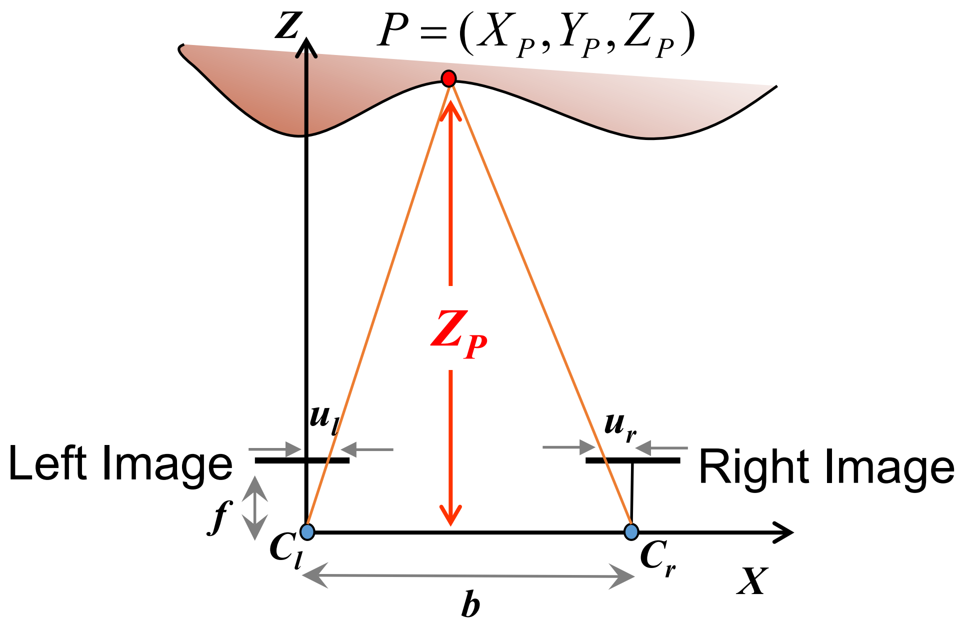
Both cameras are **identical** (i.e., same intrinsics) and are **aligned** to the x-axis



Baseline = distance between the optical centers of the two cameras

Stereo Vision - Simplified Case

Both cameras are **identical** (i.e., same intrinsics) and are **aligned** to the x-axis



Baseline = distance between the optical centers of the two cameras

From Similar Triangles:

$$\frac{f}{Z_P}=\frac{u_l}{X_P}$$

$$\frac{f}{Z_P}=\frac{-u_r}{b-X_P}$$

$$\Rightarrow \quad Z_P=\frac{bf}{\boxed{u_l-u_r}}$$

Disparity

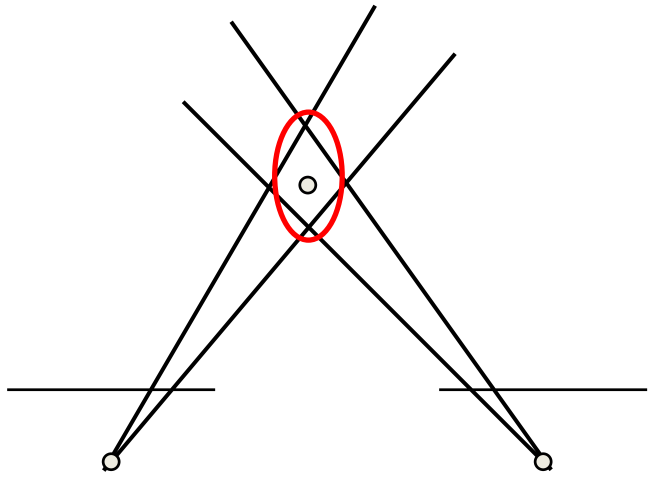
difference in image location of the projection of a 3D point on two image planes

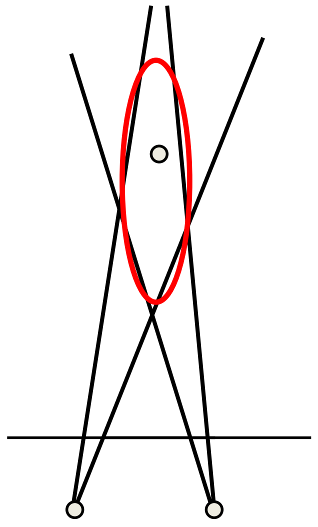
1. What's the max disparity of a stereo camera?
2. What's the disparity of a point at infinity?

Choosing the Baseline

What's the optimal baseline?

- **Large baseline:**
  - **Small depth error but…**
  - **Minimum** measurable **depth increases**
  - **Difficult search** problem for **close objects**
- **Small baseline:**
  - **Large depth error** but…
  - Minimum measureable depth decreases
  - Easier search problem for close objects

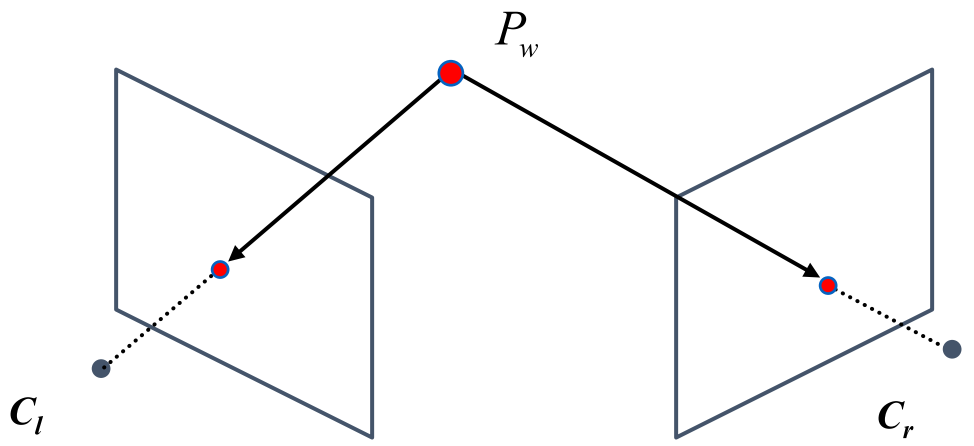

Large Baseline


Small Baseline

1. Can you compute the depth uncertainty as a function of the disparity?
2. Can you compute the depth uncertainty as a function of the distance?
3. How can we increase the accuracy of a stereo system?

Stereo Vision – General Case

- Two identical cameras do not exist in nature
- Aligning both cameras on a horizontal axis is very hard $\rightarrow$ Impossible, why?



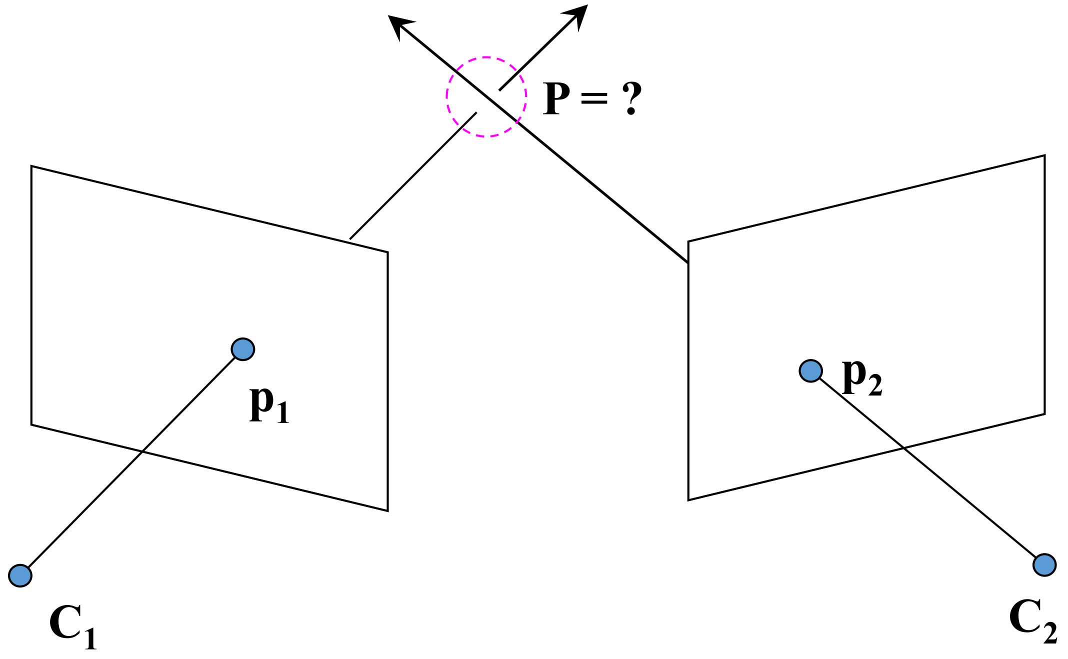
- In order to be able to use a stereo camera, we need the
  - **Extrinsic parameters** (relative rotation and translation)
  - **Instrinsic parameters** (focal length, optical center, lens distortion of each camera)

  ⇨Use a calibration method (Tsai's method (i.e., 3D object) or Zhang's method (2D grid), see Lectures 2, 3)

    ⇨ How do we compute the relative pose between the left and right cameras?

Triangulation

- **Triangulation** is the problem of determining the 3D position of a point given a set of corresponding image locations and known camera poses
- We want to intersect the two visual rays corresponding to p_1 and p_2, but, because of noise and numerical errors, they won't meet exactly, so we can only compute an approximation



Triangulation: Least Square Approximation

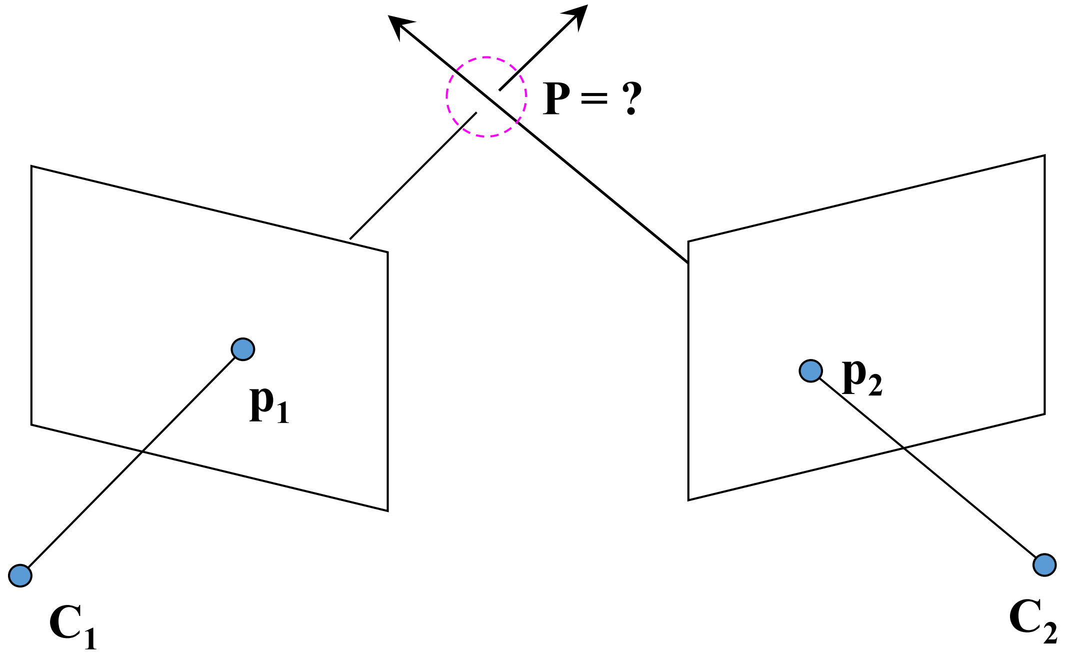
We construct the system of equations of the left and right cameras, and solve it:

Left camera (here (and often) assumed as **world frame**)

$$\lambda_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = K_1 [I|0] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

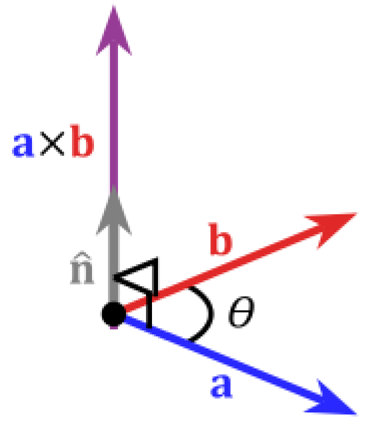
Right camera

$$\lambda_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = K_2 [R|T] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$



Review: Cross Product (or Vector Product)

$$\vec{a} \times \vec{b} = \vec{c}$$



- Vector cross product takes two vectors and returns a third vector that is perpendicular to both inputs

$$\vec{a} \cdot \vec{c} = 0$$

$$\vec{b} \cdot \vec{c} = 0$$

- So here, $\boldsymbol{c}$ is perpendicular to both $\boldsymbol{a}$ and $\boldsymbol{b}$, which means the dot product = 0
- Also, **recall that the cross product of two parallel vectors is the 0 vector**
- The vector **cross product** can also be expressed as the product of a **skew-symmetric matrix** and a vector

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_\times]\mathbf{b}$$

Triangulation: Least Square Approximation

Left camera

$$\lambda_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = K_1 [I|0] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \Rightarrow \lambda_1 p_1 = M_1 \cdot P \quad \Rightarrow p_1 \times \lambda_1 p_1 = p_1 \times M_1 \cdot P \quad \Rightarrow 0 = p_1 \times M_1 \cdot P$$

Right camera

$$\lambda_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = K_2 [R|T] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \Rightarrow \lambda_2 p_2 = M_2 \cdot P \quad \Rightarrow p_2 \times \lambda_2 p_2 = p_2 \times M_2 \cdot P \quad \Rightarrow 0 = p_2 \times M_2 \cdot P$$

Triangulation: Least Square Approximation

Left camera

$$\Rightarrow 0 = p_1 \times M_1 \cdot P \quad \Rightarrow [p_{1\times}] \cdot M_1 \cdot P = 0$$

Recall:

$$[\mathbf{a}_\times] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Right camera

$$\Rightarrow 0 = p_2 \times M_2 \cdot P \quad \Rightarrow [p_{2\times}] \cdot M_2 \cdot P = 0$$

Triangulation: Least Square Approximation

Left camera

$$\Rightarrow 0 = p_1 \times M_1 \cdot P \quad \Rightarrow [p_{1\times}] \cdot M_1 \cdot P = 0$$

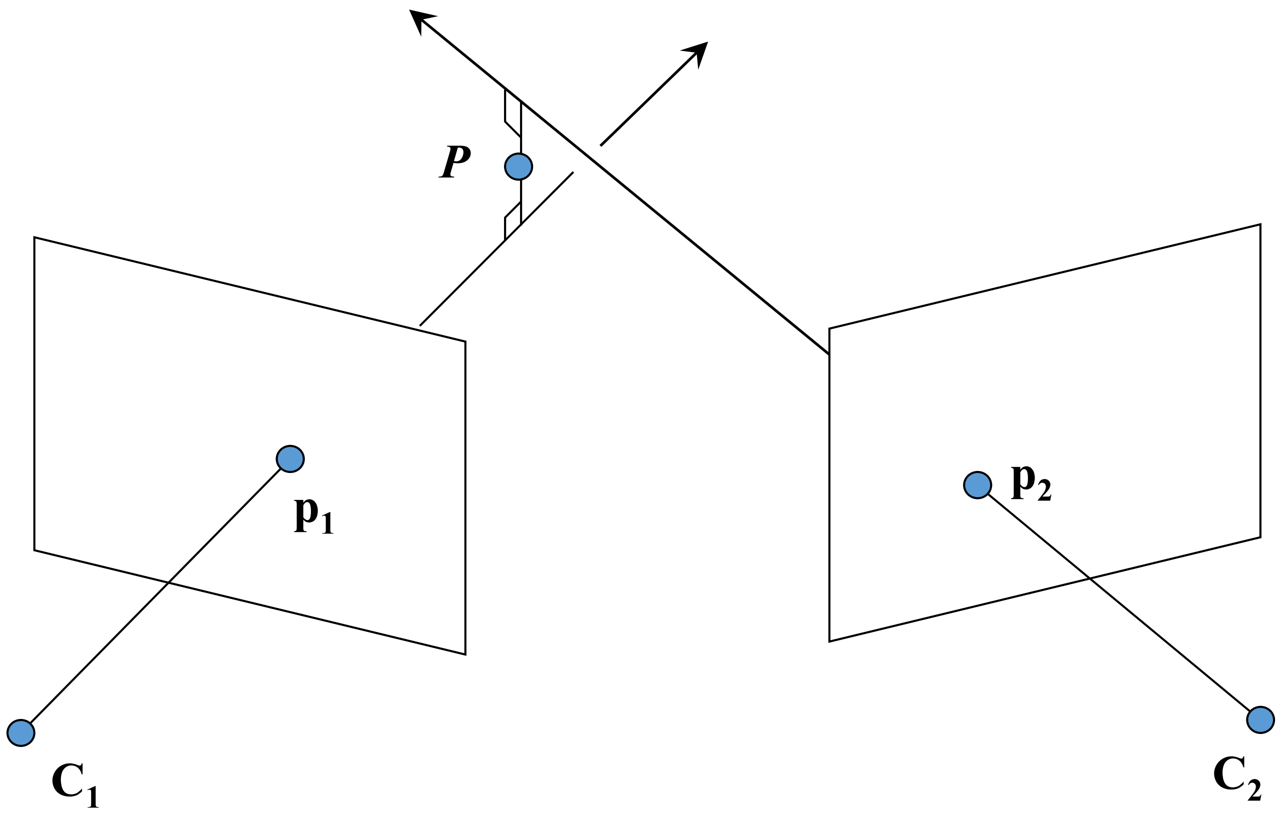
Right camera

$$\Rightarrow 0 = p_2 \times M_2 \cdot P \quad \Rightarrow [p_{2\times}] \cdot M_2 \cdot P = 0$$

- We get a homogeneous system of equations
- ***P*** can be determined using SVD, as we already did when we talked about DLT (see Lecture 03)

Geometric interpretation of Least Square Approximation

P is computed as the **midpoint of the shortest segment** connecting the two lines

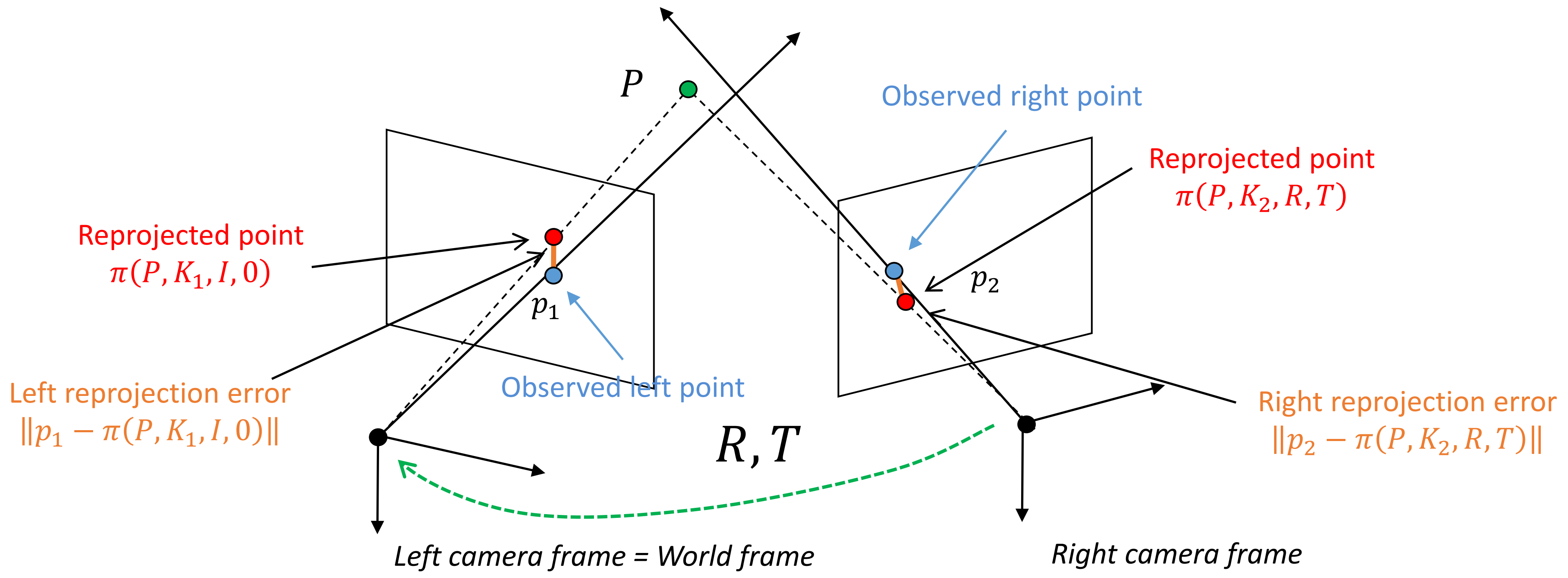


Triangulation: Nonlinear Refinement

- Initialize P using the least-square approximation; then refine P by minimizing the **sum of left and right squared reprojection errors** (for the definition of reprojection error refer to Lecture 3):

$$P = argmin_P \ \ \|p_1 - \pi(P, K_1, I, 0)\|^2 + \|p_2 - \pi(P, K_2, R, T)\|^2$$

- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



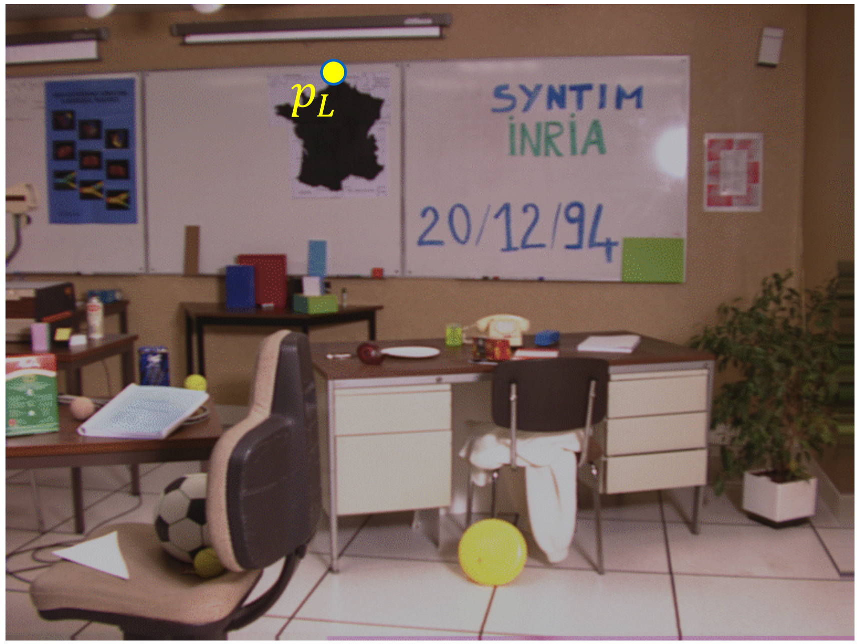
P
Observed right point
Reprojected point $\pi(P, K_2, R, T)$
Reprojected point $\pi(P, K_1, I, 0)$
p_2
p_1
Left reprojection error $\|p_1 - \pi(P, K_1, I, 0)\|$
Observed left point
R, T
Right reprojection error $\|p_2 - \pi(P, K_2, R, T)\|$
Left camera frame = World frame
Right camera frame

Stereo Vision

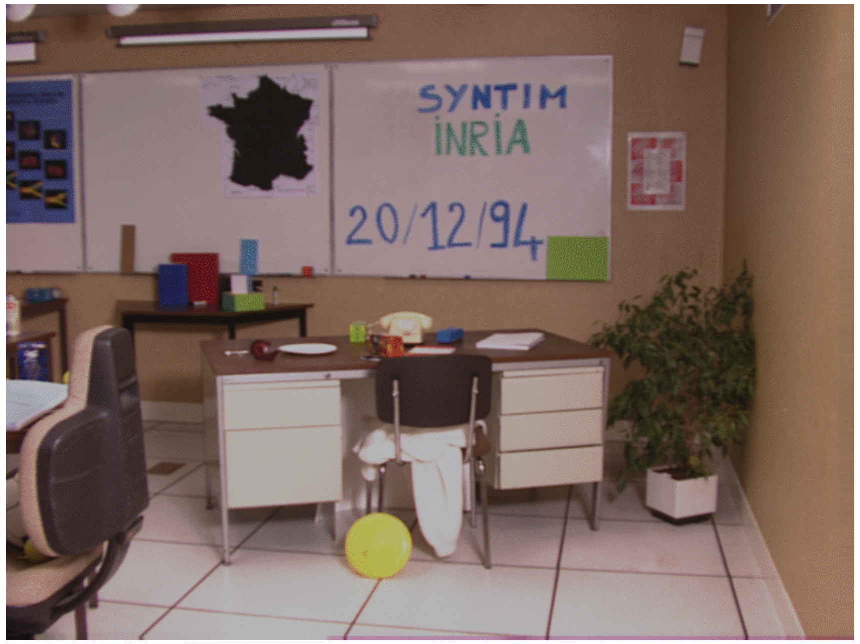
- Triangulation
  - Simplified case
  - General case
- Correspondence problem
- Stereo rectification

Correspondence Problem

Given a point, p_L, on left image, how do we **find its correspondence**, p_R, on the right image?



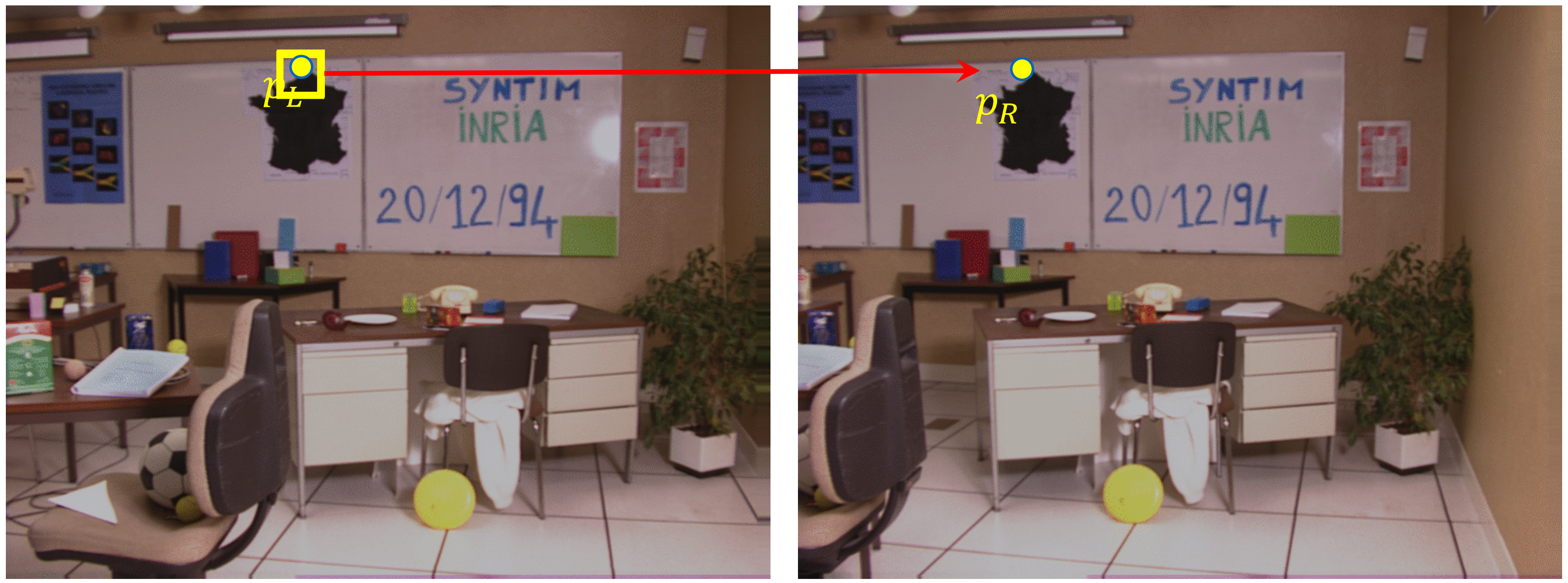
Left image



Right image

Correspondence Problem

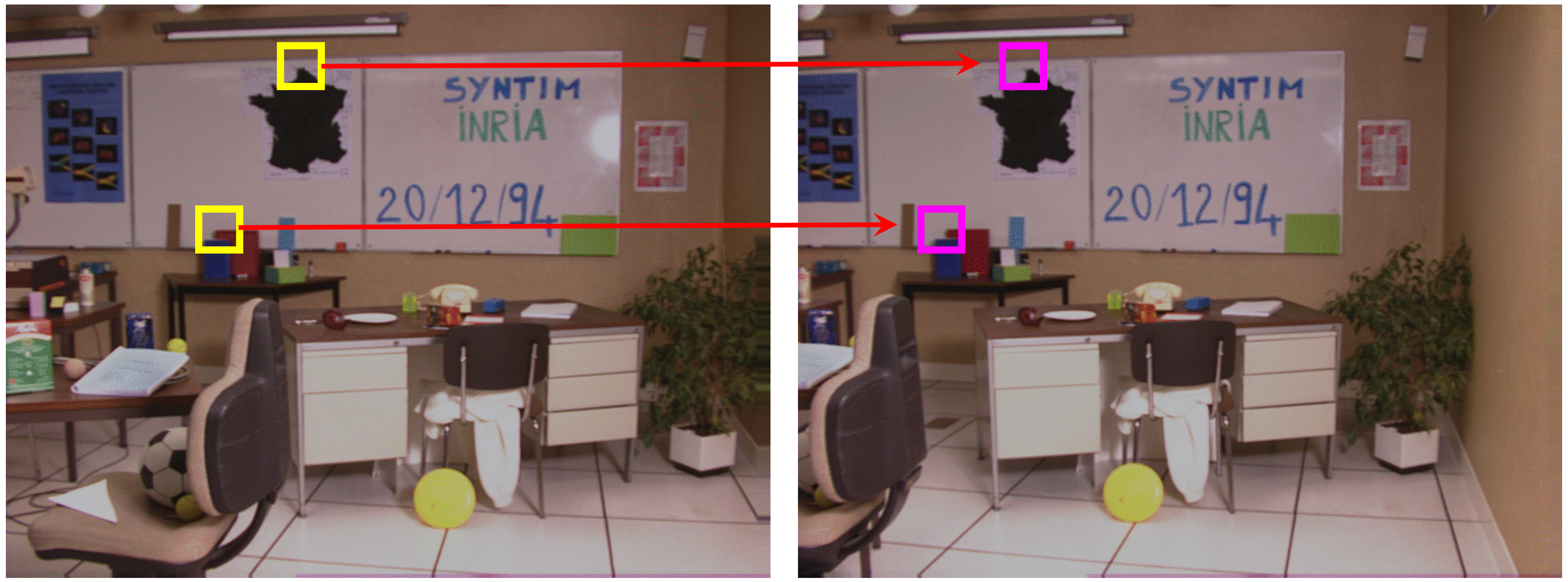
Given a point, p_L, on left image, how do we **find its correspondence**, p_R, on the right image?



Left image Right image

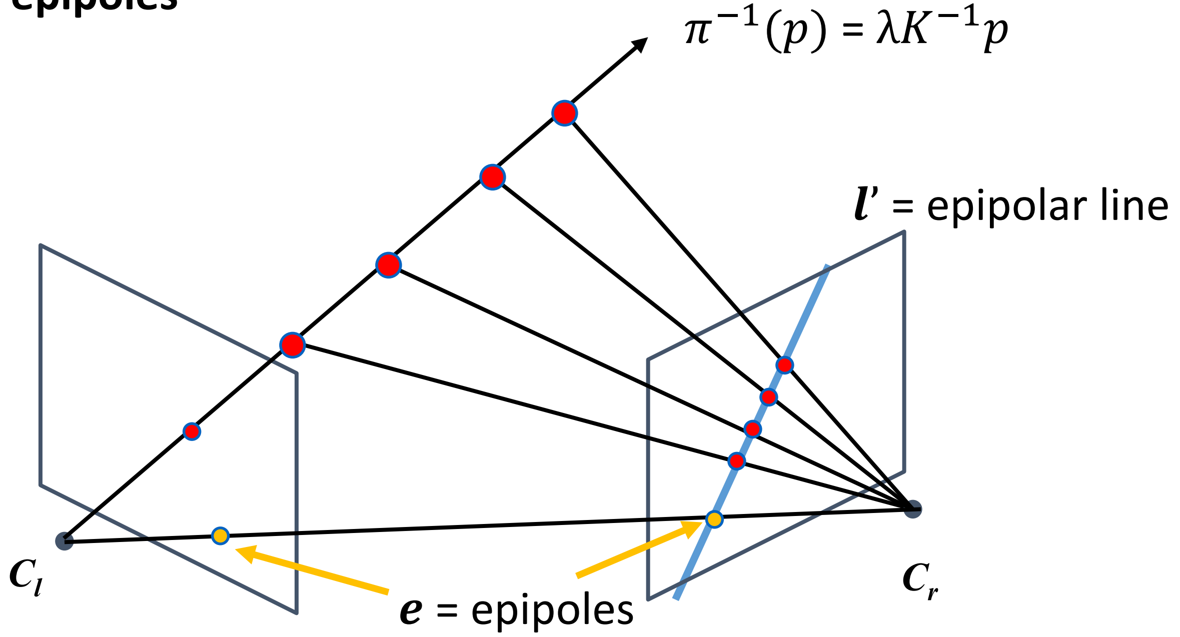
Correspondence Problem

Use one of these: **(Z)NCC, (Z)SSD, (Z)SAD, Census Transform** plus Hamming distance



Correspondence Problem

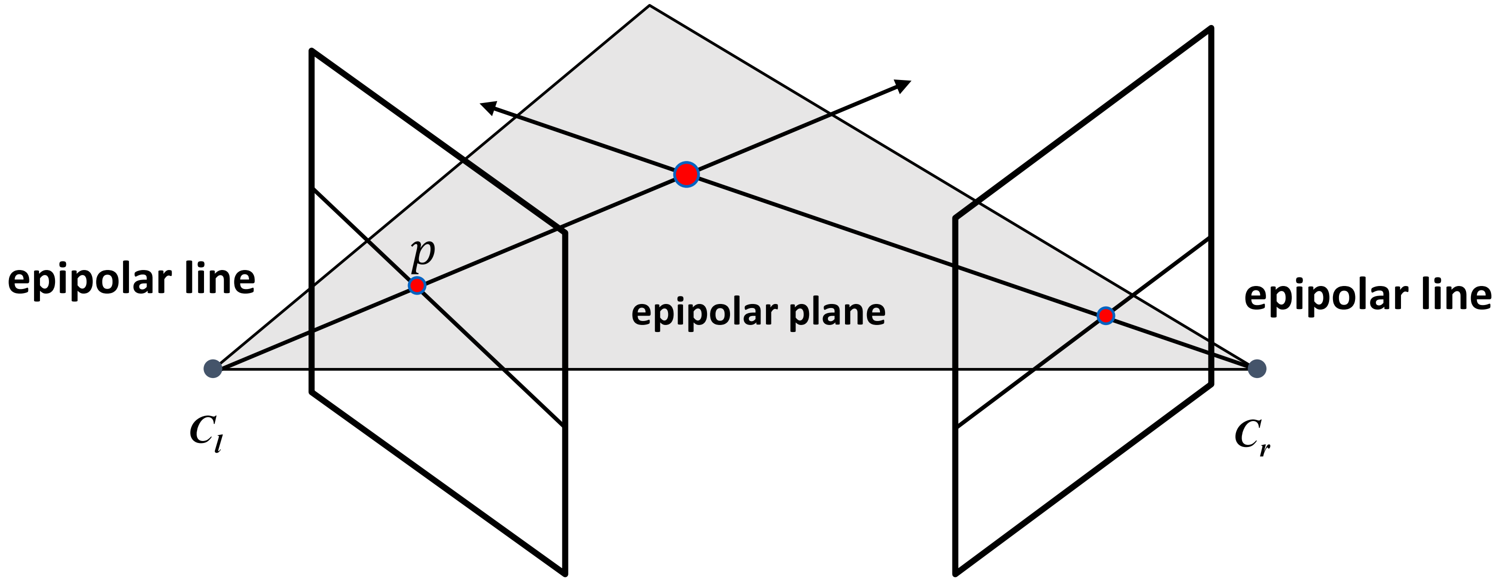
- **This 2D exhaustive search** is computationally very **expensive! How many comparisons?**
- **Can we make the correspondence search 1D**?
- Potential matches for $\boldsymbol{p}$ have to lie on the corresponding **epipolar line** $\boldsymbol{l'}$
  - The **epipolar line** is the projection of a back-projected ray $\pi^{-1}(p)$ onto the other camera image
  - The **epipole** is the projection of the optical center on the other camera image
  - A **stereo camera has two epipoles**



$\pi^{-1}(p) = \lambda K^{-1}p$
$\boldsymbol{l'}$ = epipolar line
C_l
C_r
$\boldsymbol{e}$ = epipoles

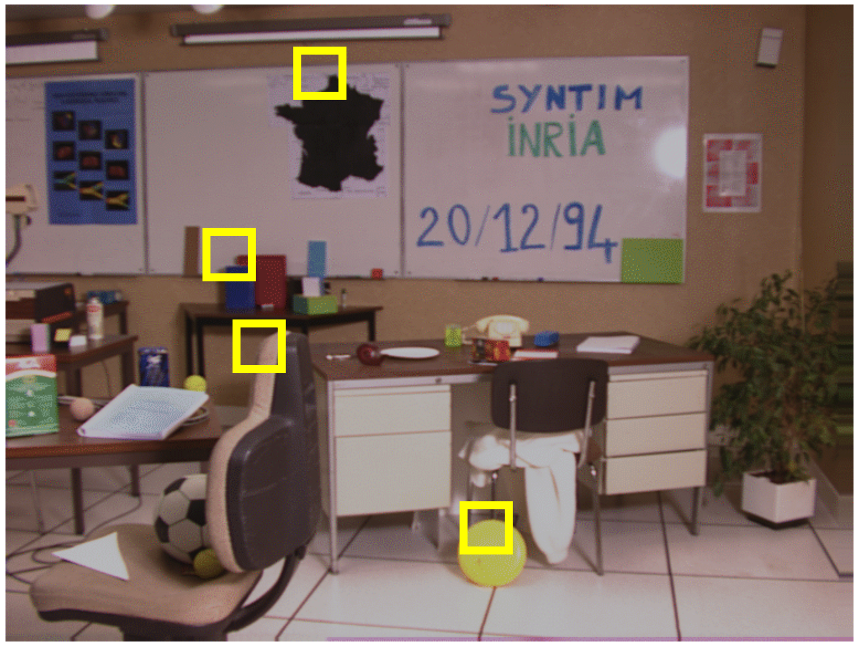
The Epipolar Constraint

- The camera centers C_l, C_r and the image point p determine the so called **epipolar plane**
- The intersections of the epipolar plane with the two image planes are called **epipolar lines**
- **Corresponding points must therefore lie along the epipolar lines**: this constraint is called **epipolar constraint**
- The epipolar constraint **reduces correspondence problem** to **1D search along the epipolar line**

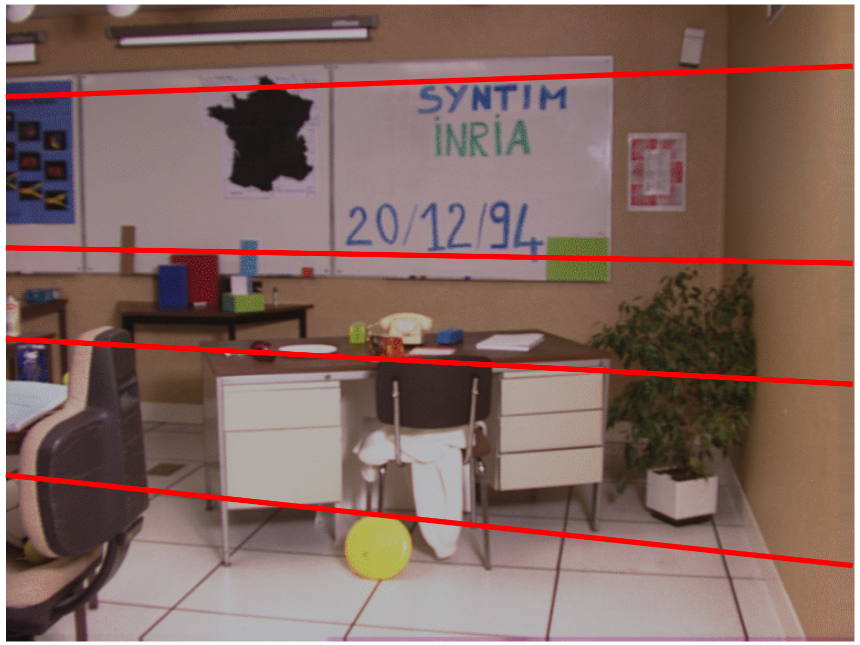


1D Correspondence Search via Epipolar Constraint

Thanks to the epipolar constraint, corresponding points can be searched for along epipolar lines: $\longrightarrow$ **computational cost reduced to 1 dimension**!



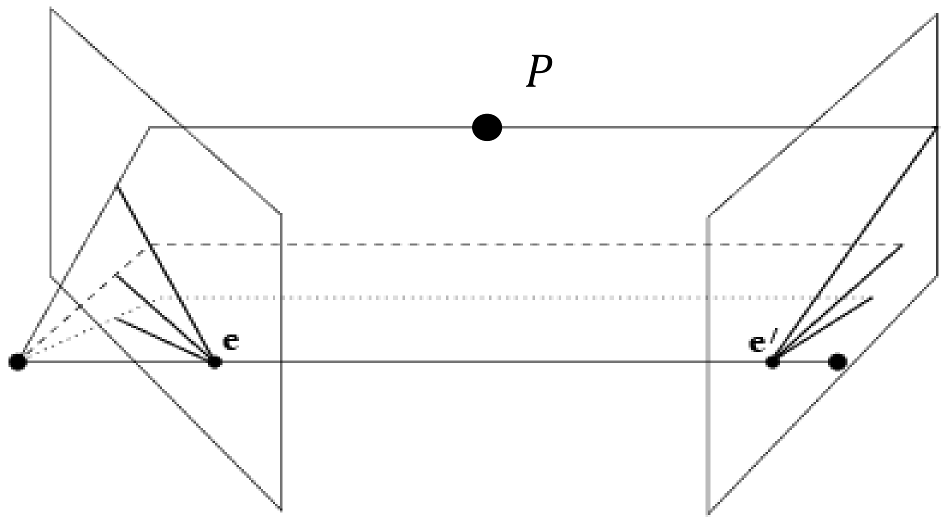
Left image

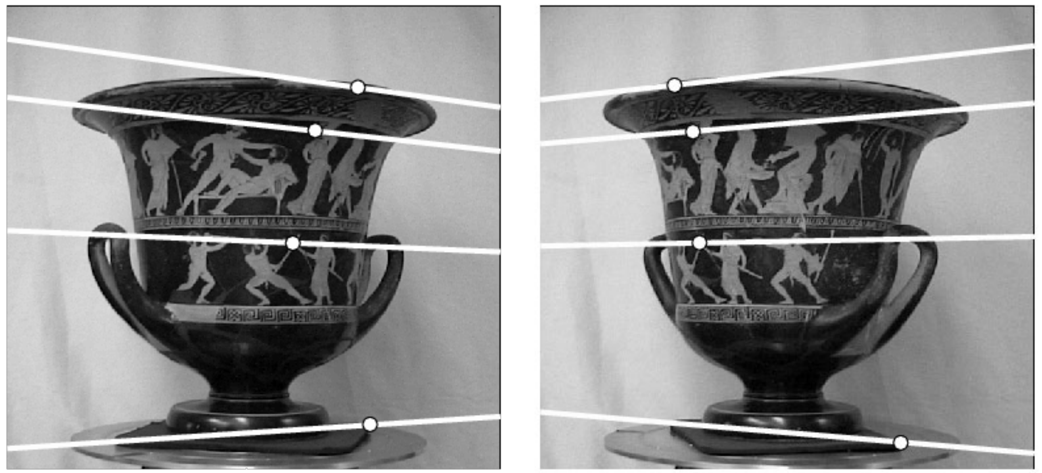


Right image

Example: Converging Cameras

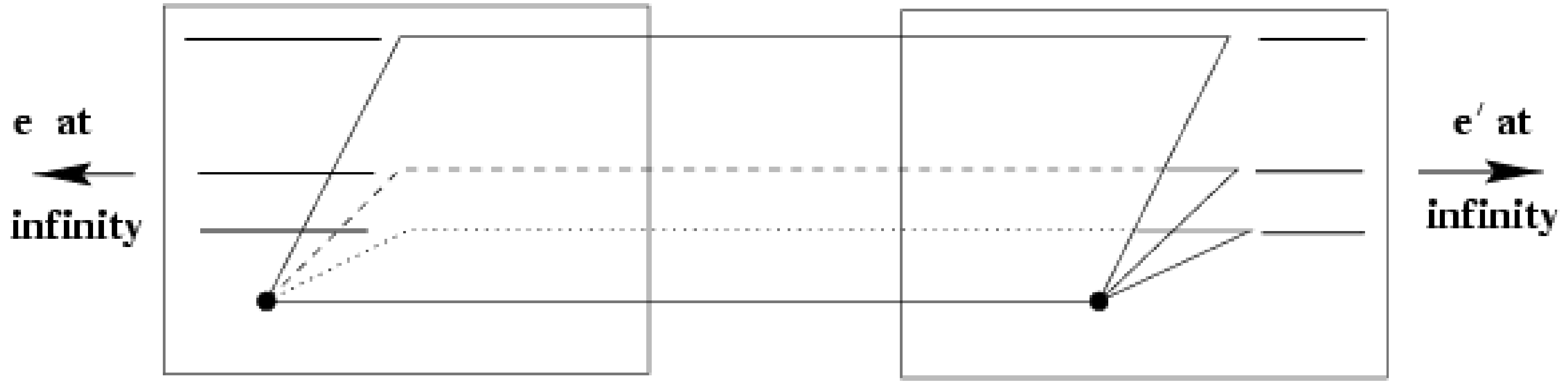
- **Remember**: all the epipolar lines intersect at the epipole
- As the position of the 3D point *P* changes, the epipolar lines *rotate* about the baseline

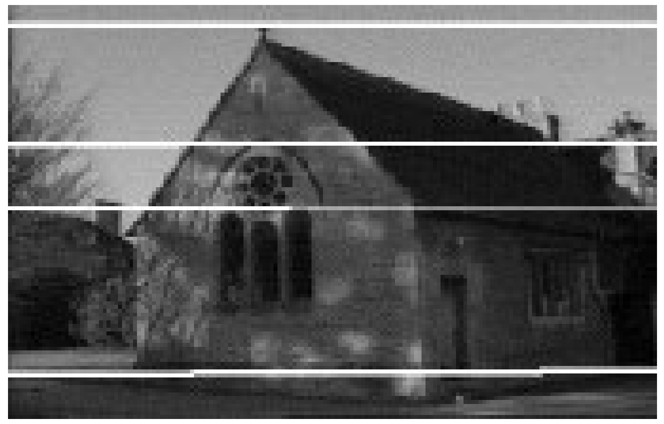




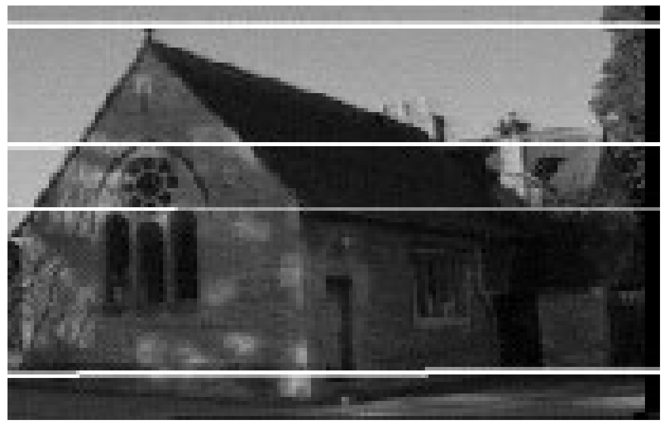
Left image Right image

Example: Identical and Horizontally-Aligned Cameras





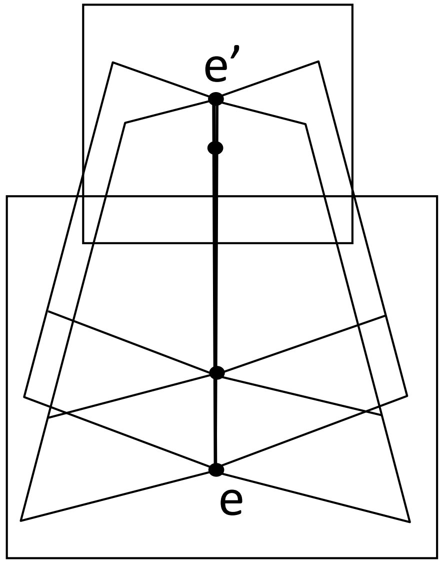
Left image

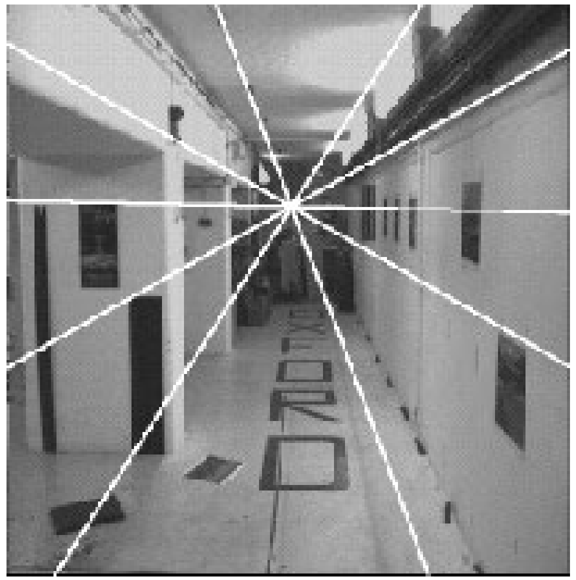


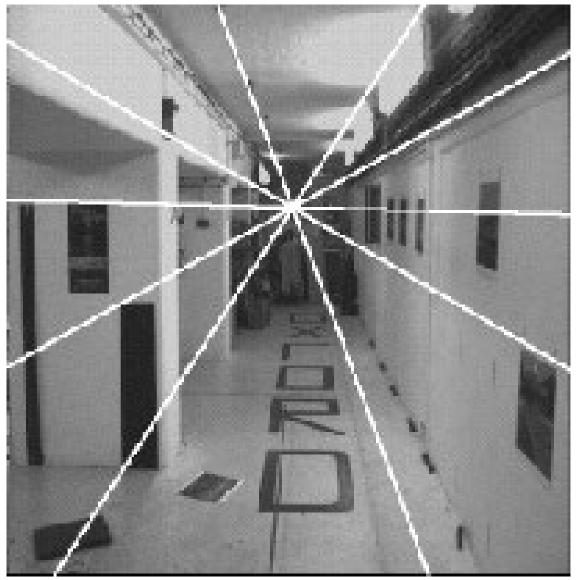
Right image

Example: Forward Motion (parallel to the optical axis)

- Epipole has the **same coordinates** in both images
- Points move along lines **radiating from the epipole**: "**Focus of expansion**"




Left image

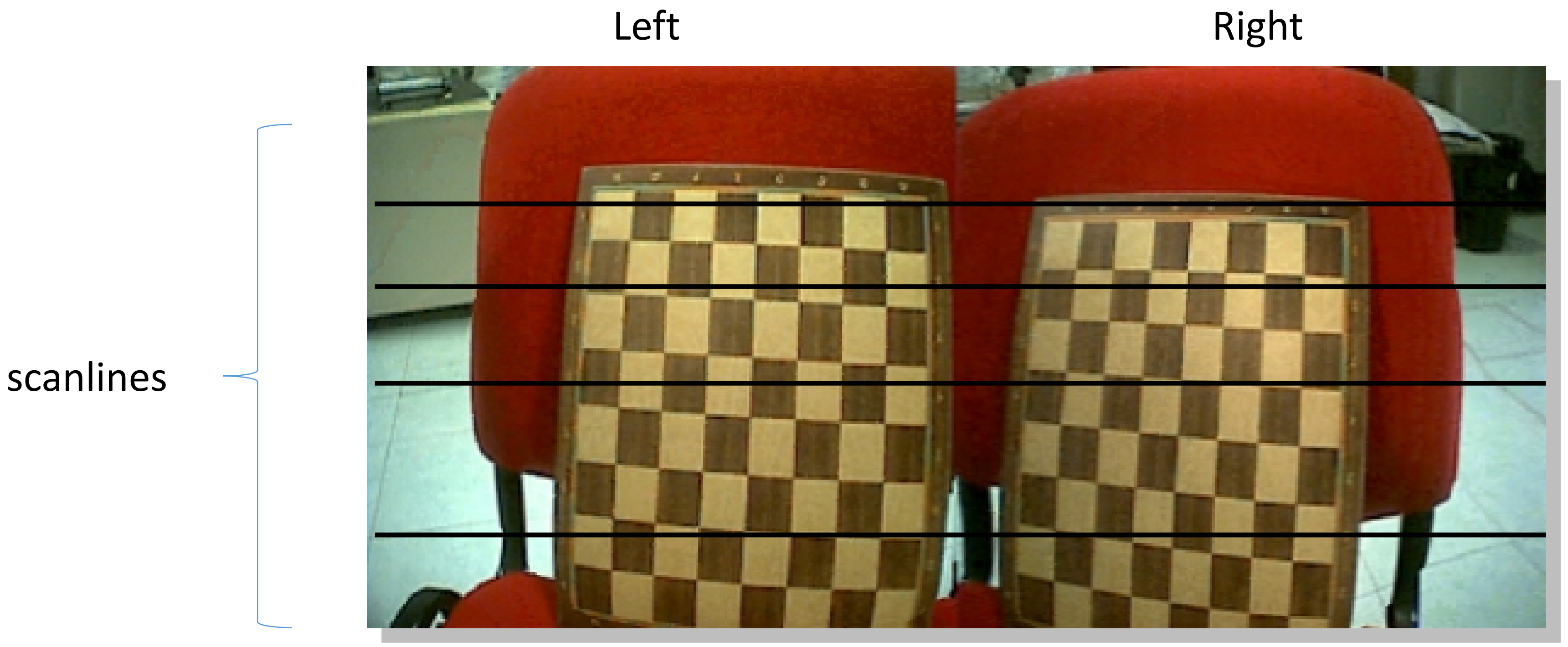

Right image

Stereo Vision

- Triangulation
  - Simplified case
  - General case
- Correspondence problem
- Stereo rectification

Stereo Rectification

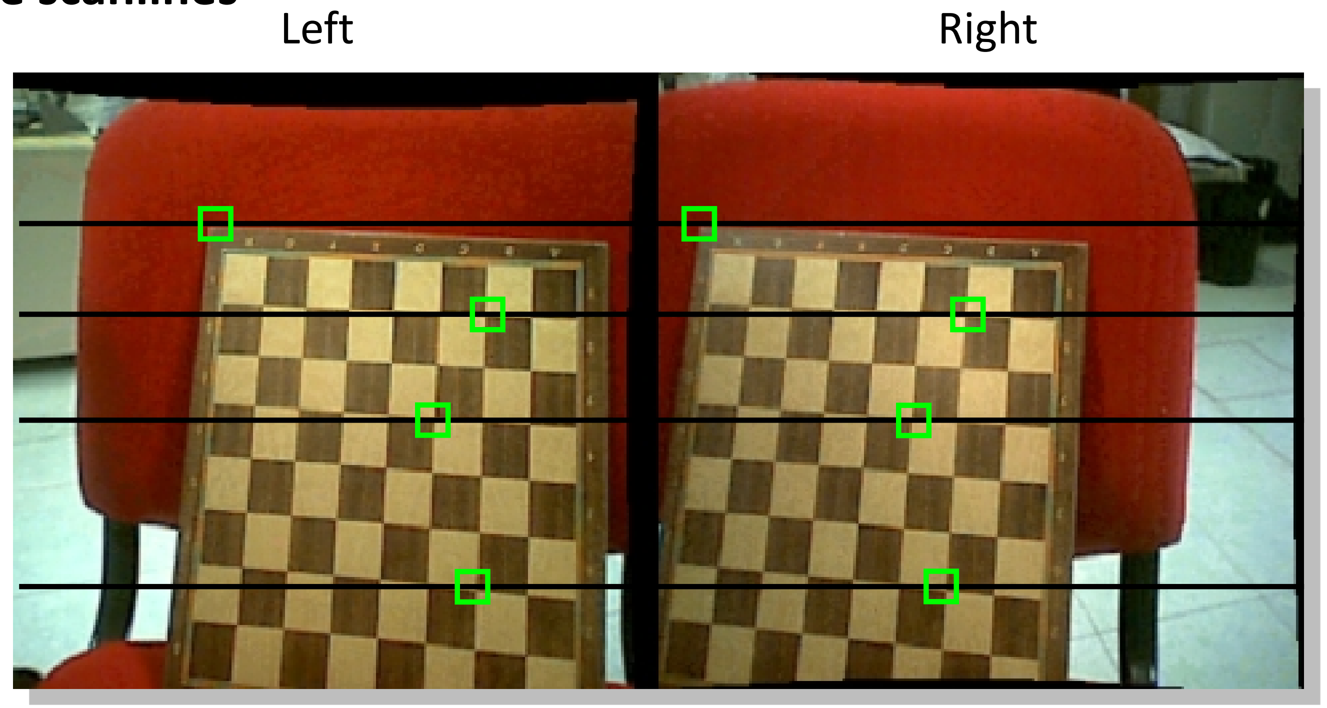
- Even in **commercial stereo cameras** the left and right images are **never perfectly aligned**
- In practice, it is **convenient** if image **scanlines are the epipolar lines** because then the correspondence search can be made very efficient (only search the point along the same scanlines)



Raw stereo pair (**unrectified**): scan lines do not coincide with epipolar lines

Stereo Rectification

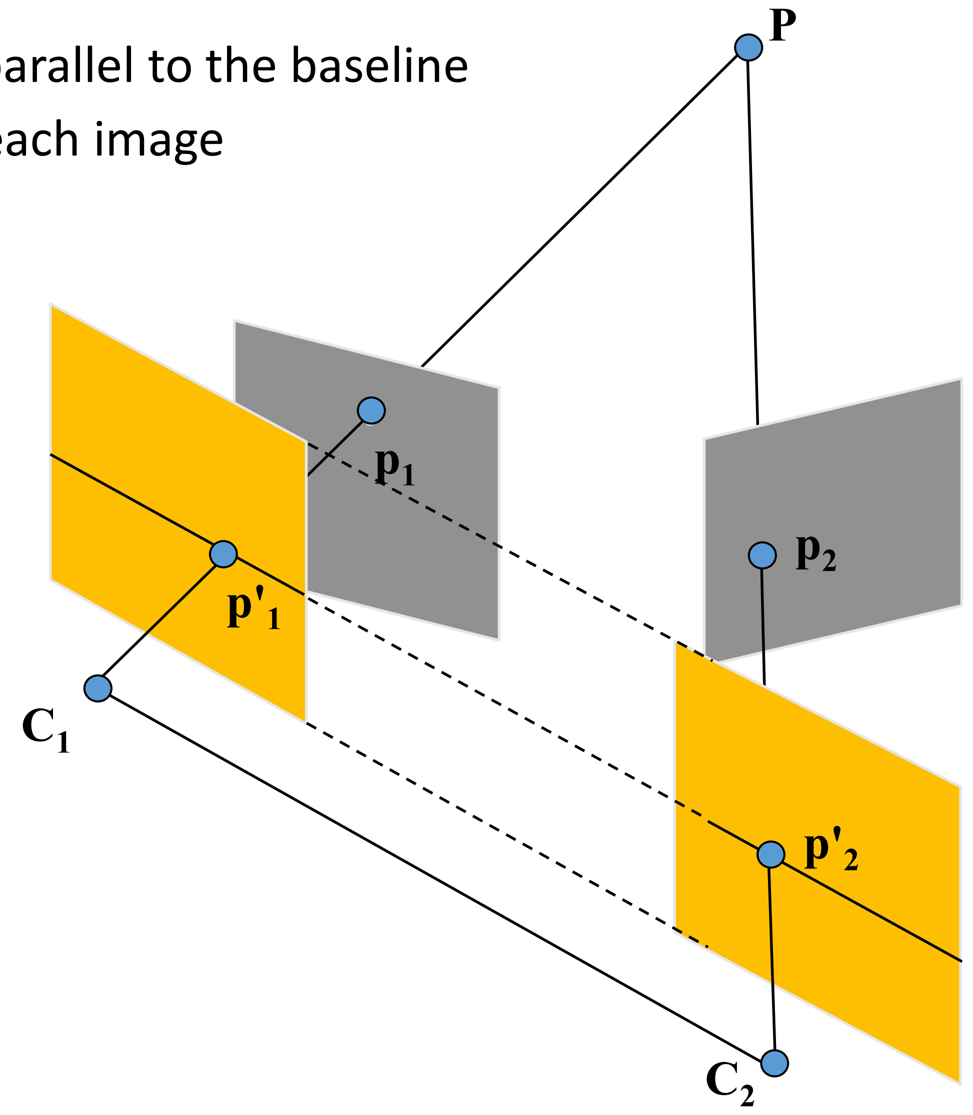
- Even in **commercial stereo cameras** the left and right images are **never perfectly aligned**
- In practice, it is **convenient** if image **scanlines are the epipolar lines** because then the correspondence search can be made very efficient (only search the point along the same scanlines)
- **Stereo rectification warps the left and right images** into new "rectified" images such that the **epipolar lines coincide with the scanlines**



Rectified stereo pair: **scanlines coincide with epipolar lines**

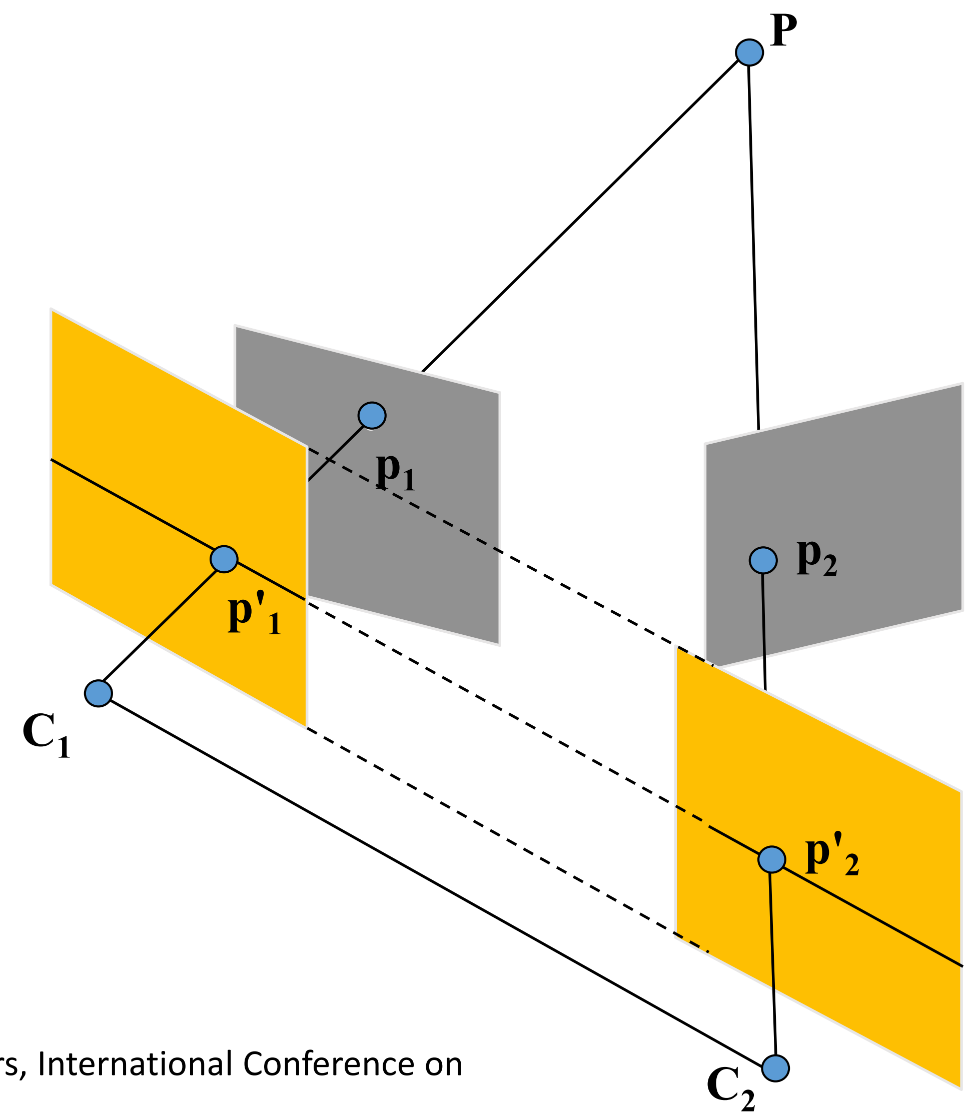
Stereo Rectification

- Warps original image planes onto coplanar planes parallel to the baseline
- It works by computing two homographies, one for each image
- As a result, the new **epipolar lines** coincide
  the **scanlines** of the left and right image
  are aligned



Stereo Rectification

- The idea behind rectification is to define two new Perspective Projection Matrices (PPMs) obtained by rotating the old ones around their optical centers until the image planes become parallel to each other.
- This ensures that epipoles are at infinity, hence epipolar lines are parallel.
- To have **horizontal epipolar lines**, the **baseline** must be **parallel to the new X axis** of both cameras.
- In addition, to have a proper rectification, corresponding points must have the **same vertical coordinate**. This is obtained by requiring that the new cameras have the **same intrinsic parameters**.
- Note that, being the focal length the same, the new image planes are coplanar too

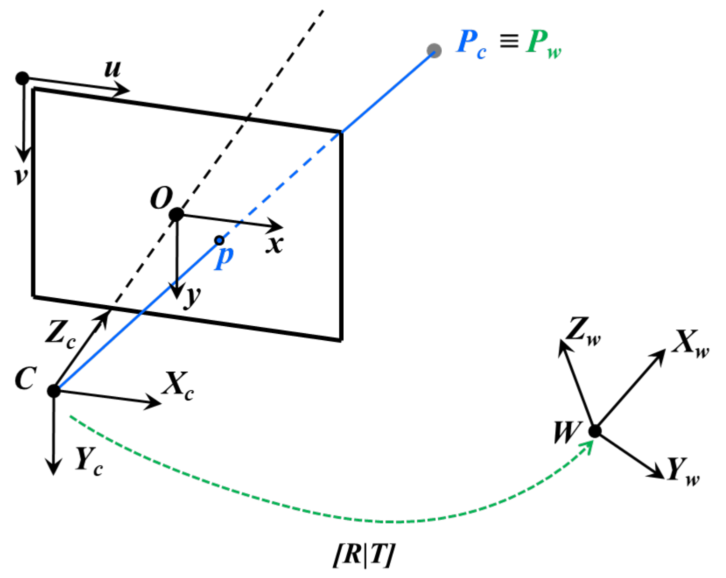


Fusiello, Trucco, Verri, "A Compact Algorithm for Rectification of Stereo Pairs, International Conference on Computer Vision (ICCV), 1999. PDF.

Stereo Rectification (1/5)

In Lecture 02, we have seen that the Perspective Equation for a point P_w in the world frame is defined by this equation, where $R = R_{cw}$ and $T = T_{cw}$ transform points **from the World frame to the Camera frame**.

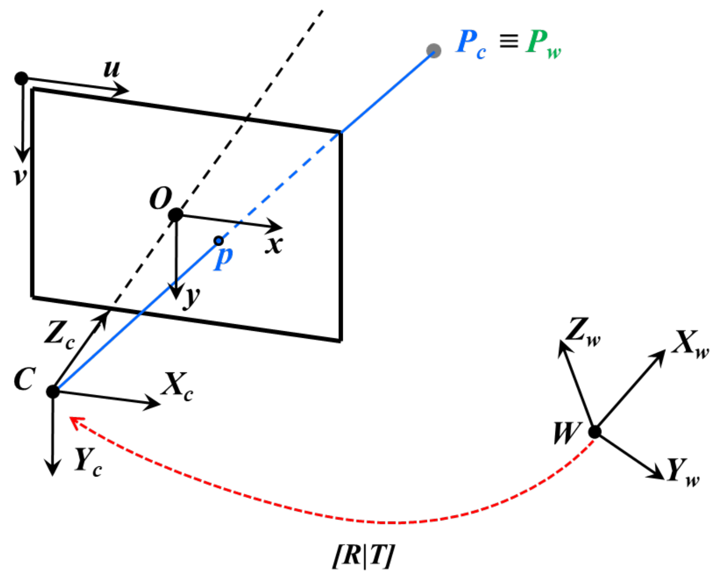
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \left(R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \right)$$



Stereo Rectification (1/5)

- For Stereo Vision, however, it is more common to use $\boldsymbol{R} \equiv \boldsymbol{R_{wc}}$ and $\boldsymbol{T} \equiv \boldsymbol{T_{wc}}$, where now $\boldsymbol{R}$, and $\boldsymbol{T}$ transform points **from the Camera frame to the World frame**. This is more convenient because $\boldsymbol{T} \equiv \boldsymbol{C}$ directly represents the world coordinates of the camera center. The projection equation can be re-written as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KR^{-1}\left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - T\right) \quad \rightarrow \quad \boxed{\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KR^{-1}\left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C\right)}$$



Stereo Rectification (2/5)

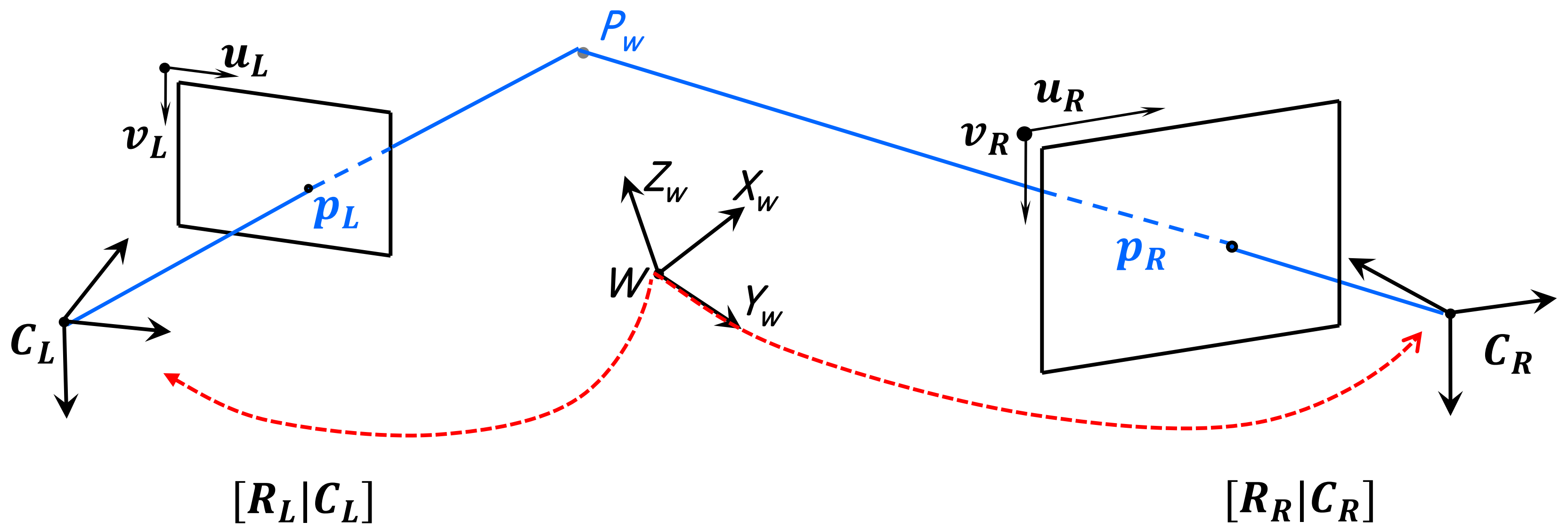
We can now write the Perspective Equation for the Left and Right cameras. For generality, we assume that Left and Right cameras have different intrinsic parameter matrices, K_L, K_R:

Left camera

$$\lambda_L \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} = K_L R_L{}^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_L \right)$$

Right camera

$$\lambda_R \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = K_R R_R{}^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_R \right)$$



Stereo Rectification (3/5)

The goal of stereo rectification is to **warp** the **left** and **right camera** images such that their **image planes** are **coplanar (i.e., same $\hat{\boldsymbol{R}}$)** and their **intrinsic** parameters are **identical (i.e., same $\hat{\boldsymbol{K}}$)**

$$\lambda_L \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} = K_L R_L^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_L \right)$$

Old Left camera

$$\rightarrow \hat{\lambda}_L \begin{bmatrix} \hat{u}_L \\ \hat{v}_L \\ 1 \end{bmatrix} = \hat{\boldsymbol{K}} \hat{\boldsymbol{R}}^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_L \right)$$

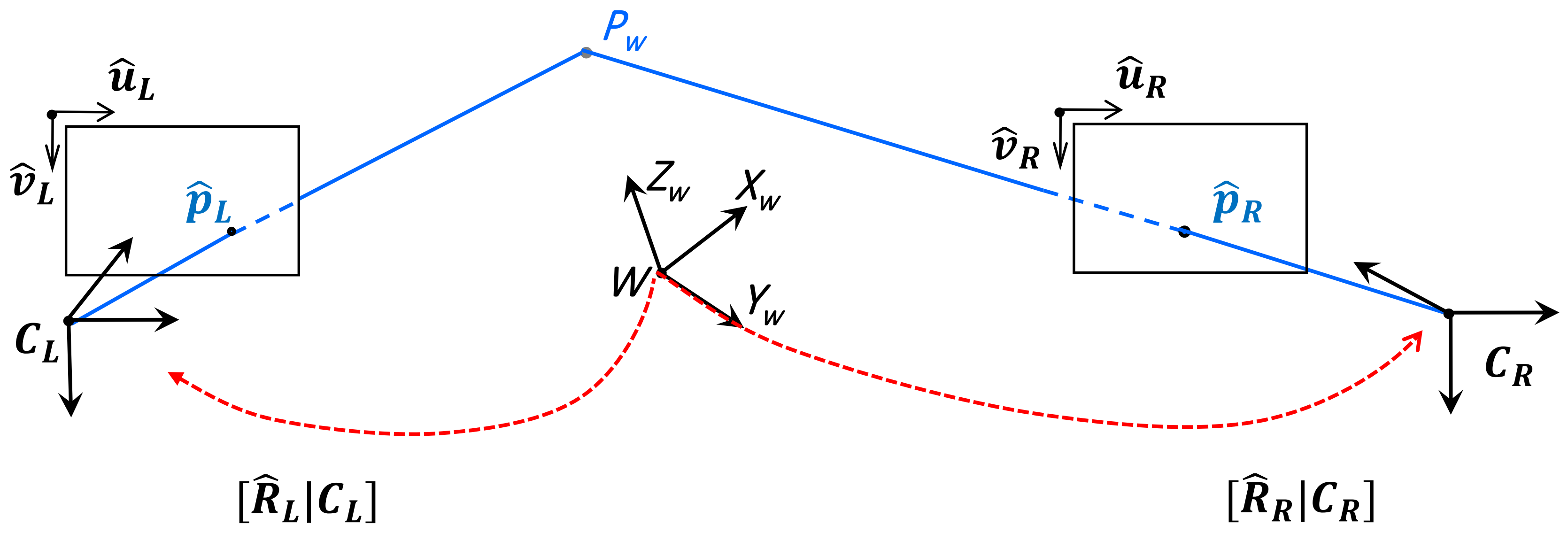
New Left camera

$$\lambda_R \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = K_R R_R^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_R \right)$$

Old Right camera

$$\rightarrow \hat{\lambda}_R \begin{bmatrix} \hat{u}_R \\ \hat{v}_R \\ 1 \end{bmatrix} = \hat{\boldsymbol{K}} \hat{\boldsymbol{R}}^{-1} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - C_R \right)$$

New Right camera

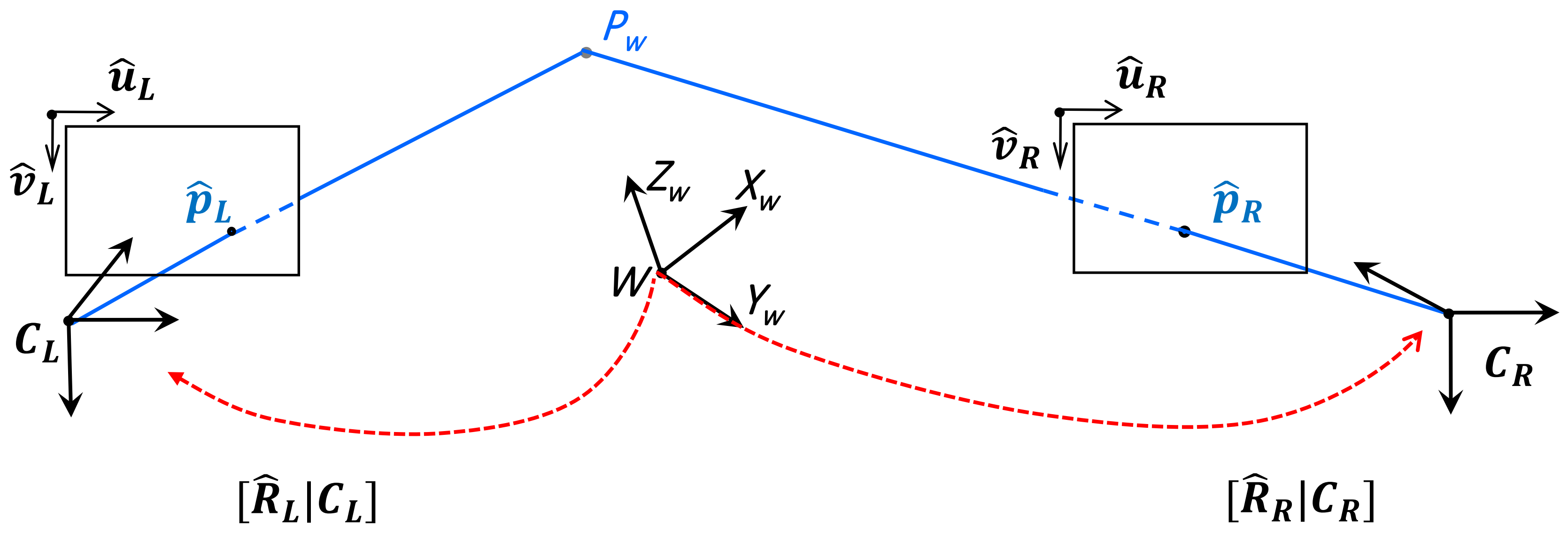


Stereo Rectification (4/5)

By solving with respect to (X_w, Y_w, Z_w) for each camera, we can compute the Homography that needs to be applied to rectify each camera image:

$$\hat{\lambda}_L \begin{bmatrix} \hat{u}_L \\ \hat{v}_L \\ 1 \end{bmatrix} = \underbrace{\lambda_L \widehat{\boldsymbol{K}} \widehat{\boldsymbol{R}}^{-1} R_L {K_L}^{-1}}_{\text{Homography of Left Camera}} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix}$$

$$\hat{\lambda}_R \begin{bmatrix} \hat{u}_R \\ \hat{v}_R \\ 1 \end{bmatrix} = \underbrace{\lambda_R \widehat{\boldsymbol{K}} \widehat{\boldsymbol{R}}^{-1} R_R {K_R}^{-1}}_{\text{Homography of Right Camera}} \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix}$$



Stereo Rectification (5/5)

How do we chose the new $\widehat{\boldsymbol{K}}$ and $\widehat{\boldsymbol{R}}$? A good choice is to impose that:

$$\widehat{\boldsymbol{K}} = \frac{K_L + K_R}{2}$$

$$\widehat{\boldsymbol{R}} = [\widehat{r_1}, \widehat{r_2}, \widehat{r_3}]$$

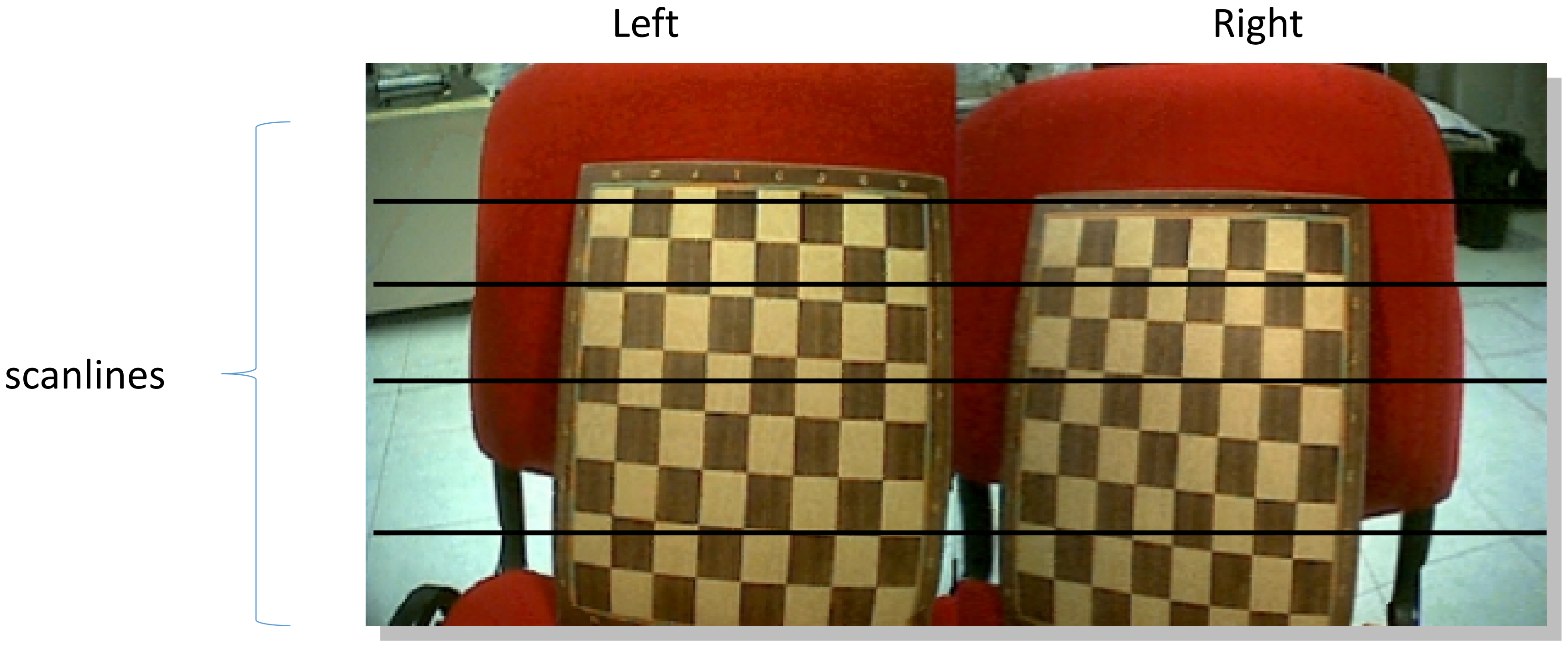
with $\widehat{r_1}, \widehat{r_2}, \widehat{r_3}$ being the column vectors of $\widehat{R}$, where:

$$\widehat{r_1} = \frac{C_R - C_L}{\|C_R - C_L\|}$$ This makes the new image planes are parallel to the baseline

$$\widehat{r_2} = r_3 \times \widehat{r_1}$$ where r_3 is the 3rd column of the rotation matrix of the left camera, i.e., R_L

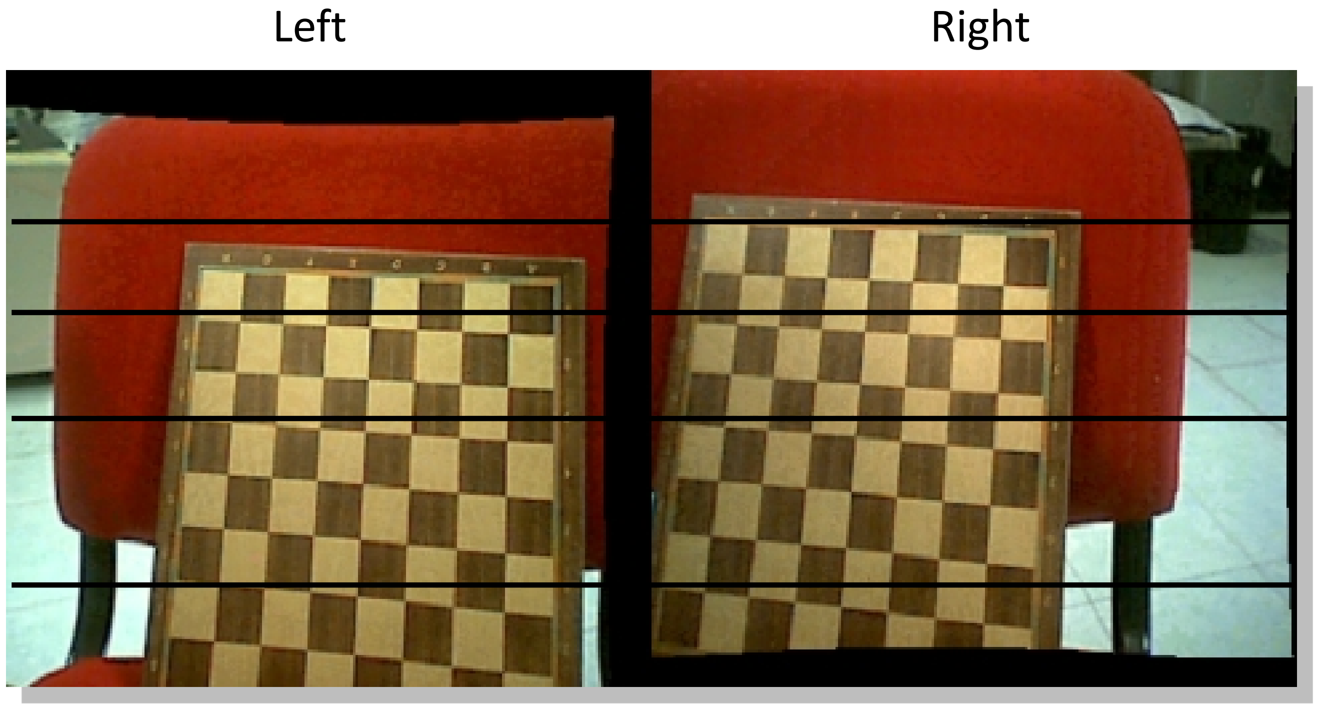
$$\widehat{r_3} = \widehat{r_1} \times \widehat{r_2}$$

Stereo Rectification: Example



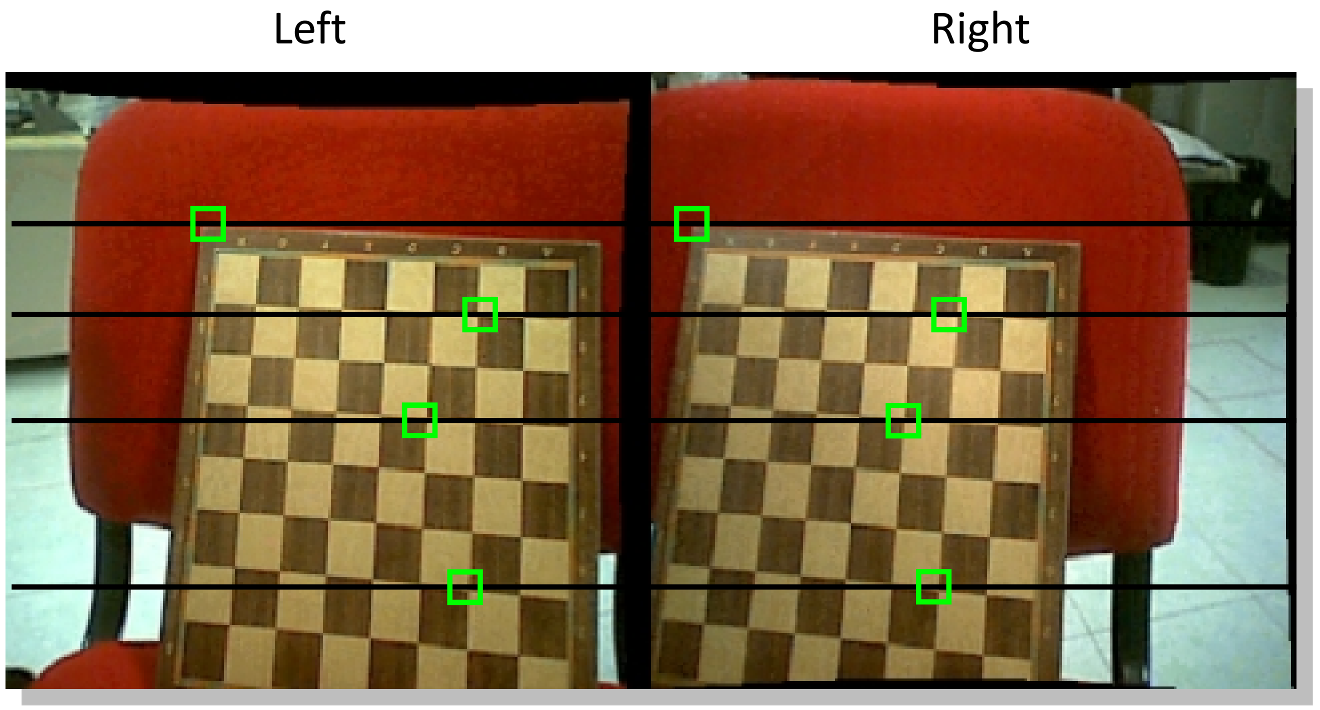
Stereo Rectification: Example

- First, **compute lens distortion**

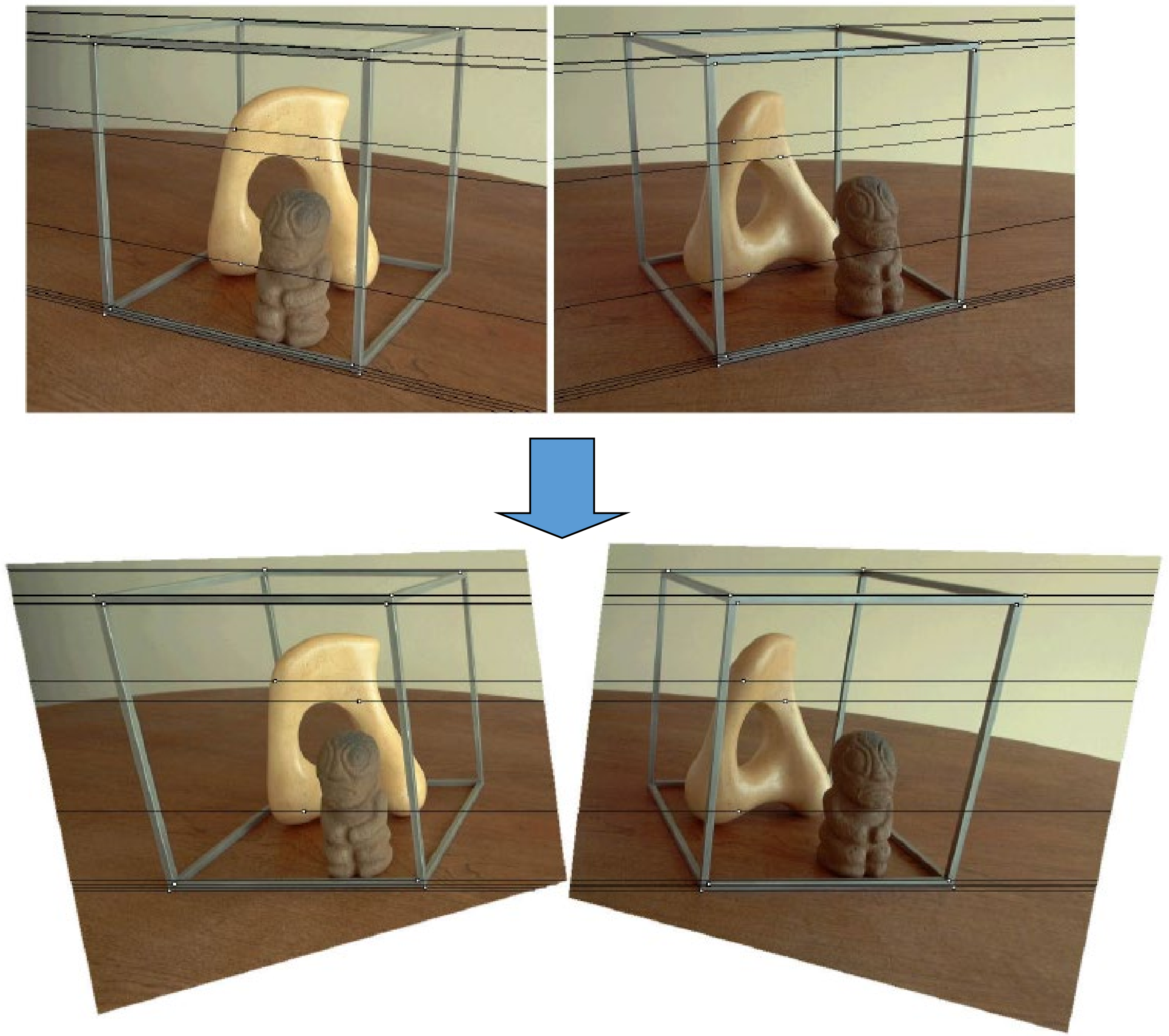


Stereo Rectification: Example

- First, **compute lens distortion**
- Then, **compute homographies and rectify**
- Use **bilinear interpolation for warping** (see lect. 06)

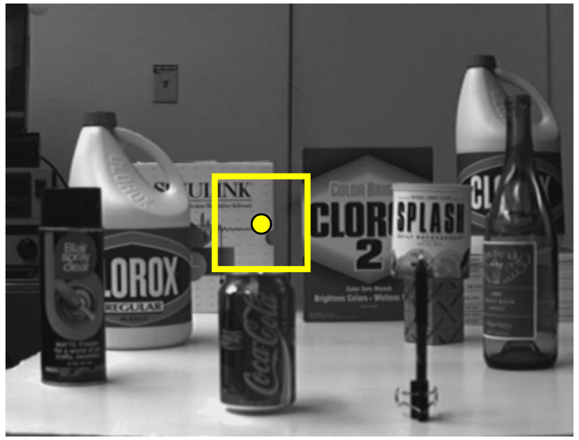


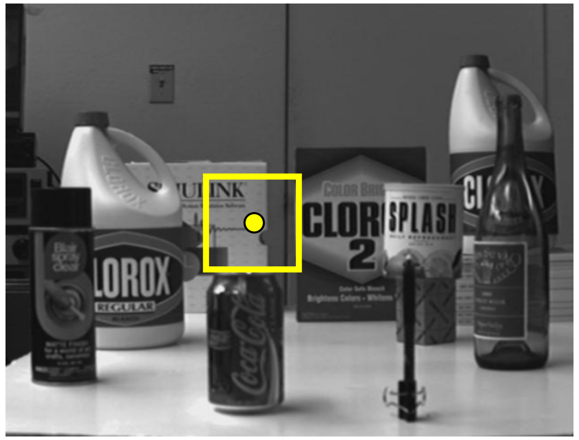
Stereo Rectification: Example

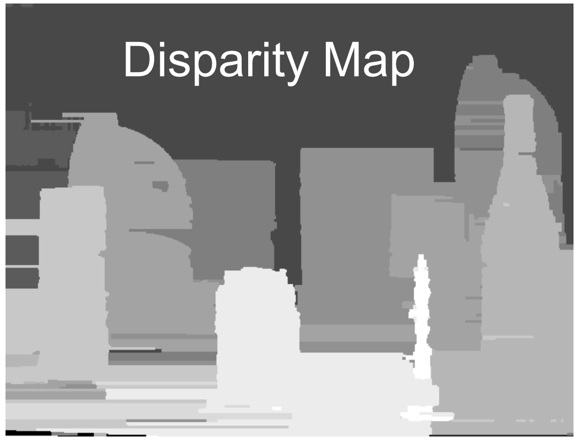


Dense Stereo Correspondence: Disparity Map

1. **Rectify stereo pair** (if not already rectified)
2. For **every pixel in the left image**, **find its corresponding** point in the right image
3. Compute the **disparity** for each found pair of correspondences
4. Visualize it as a grayscale or color-coded image: **Disparity map**


Left image


Right image



Close objects experience bigger disparity
→ appear brighter in disparity map

From Disparity Map to Point Cloud

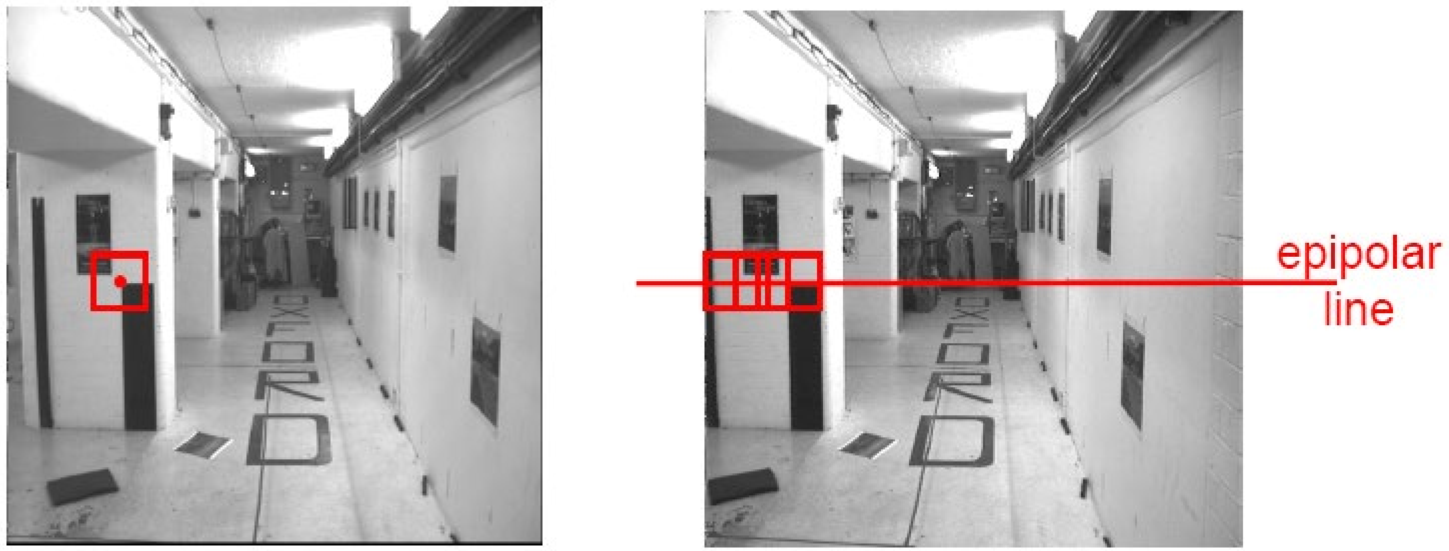
Once the stereo pair is rectified, the depth of each point can be computed recalling that: $Z_P = \frac{bf}{u_l - u_r}$

Stereo Vision

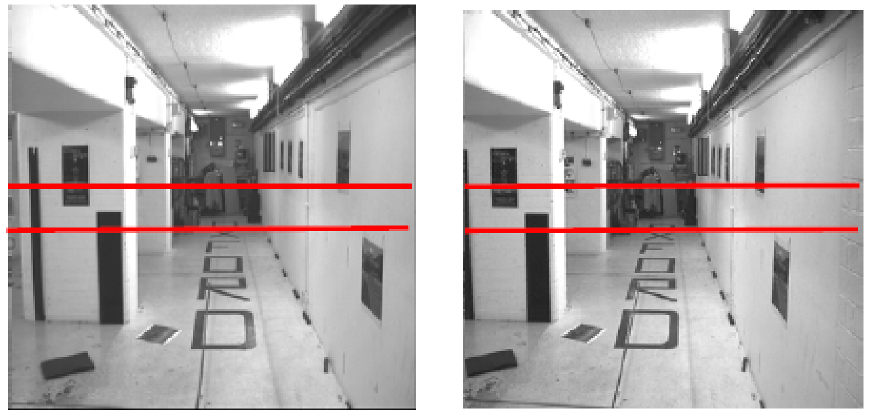
- Triangulation
  - Simplified case
  - General case
- Correspondence problem: continued
- Stereo rectification

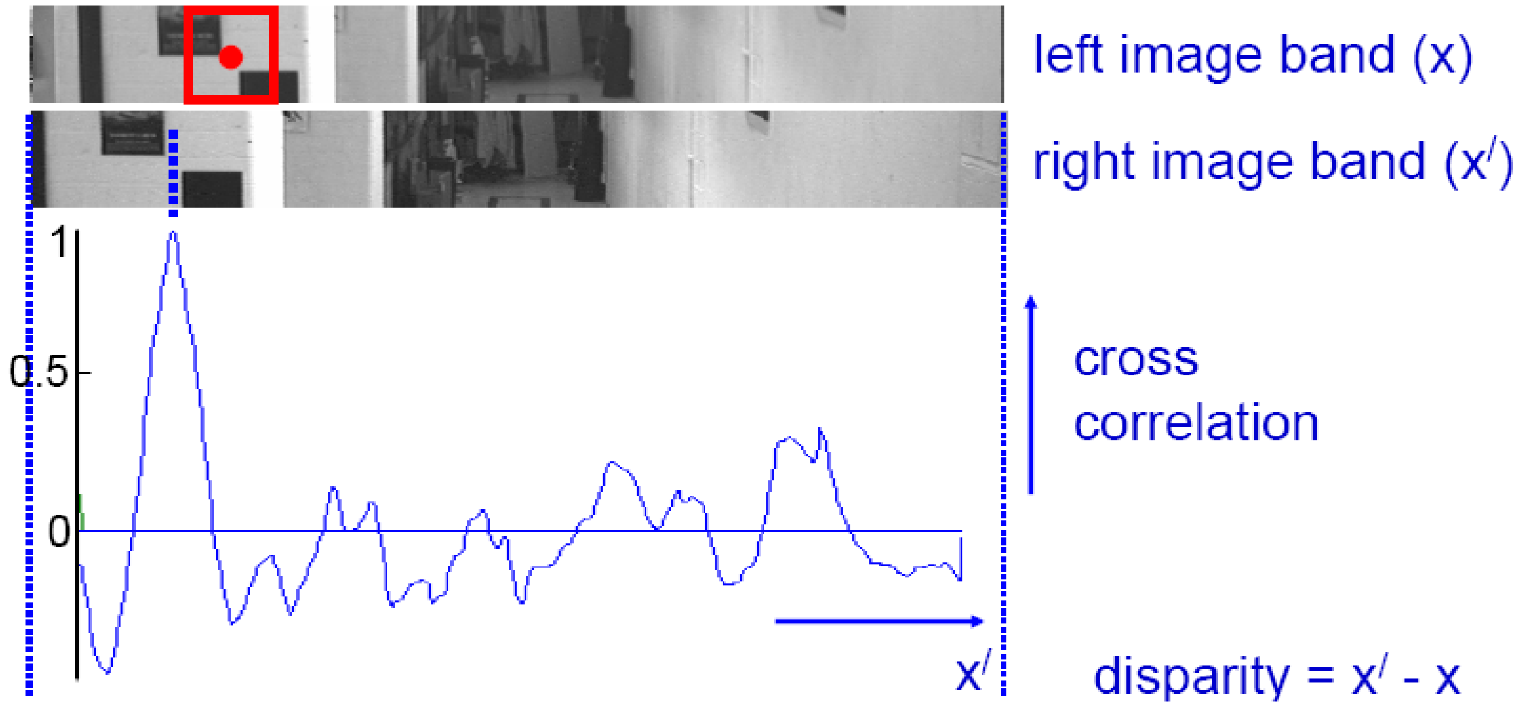
Correspondence Problem

- Once left and right images are rectified, correspondence search can be done along the same scanlines
- To **average noise effects**, use a window around the point of interest (**assumption: neighboring pixels have similar intensity**)
- Find correspondence by maximizing or minimizing: **(Z)NCC, (Z)SSD, (Z)SAD, Census Transform** plus Hamming distance

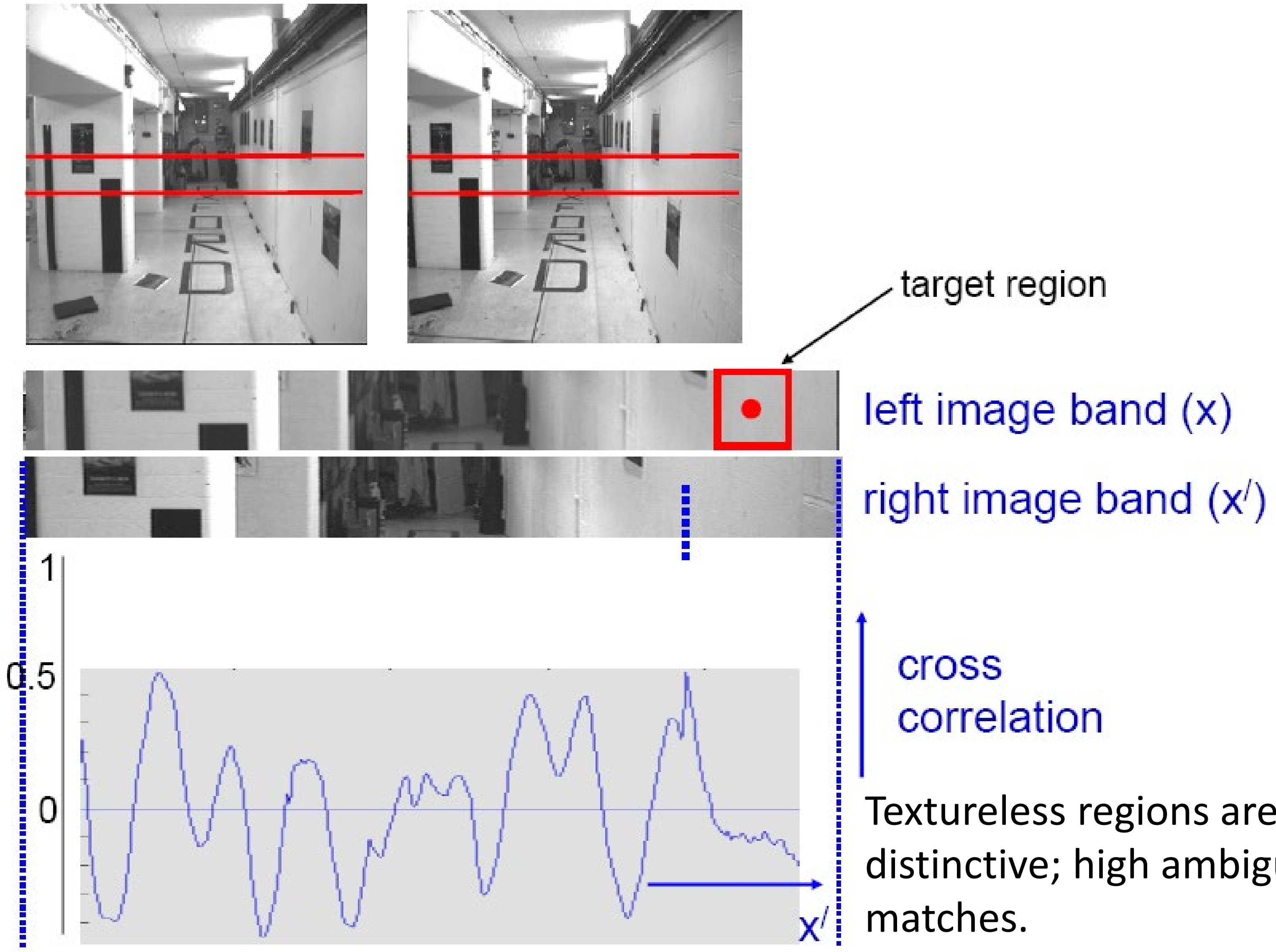


Example: (Z)NCC





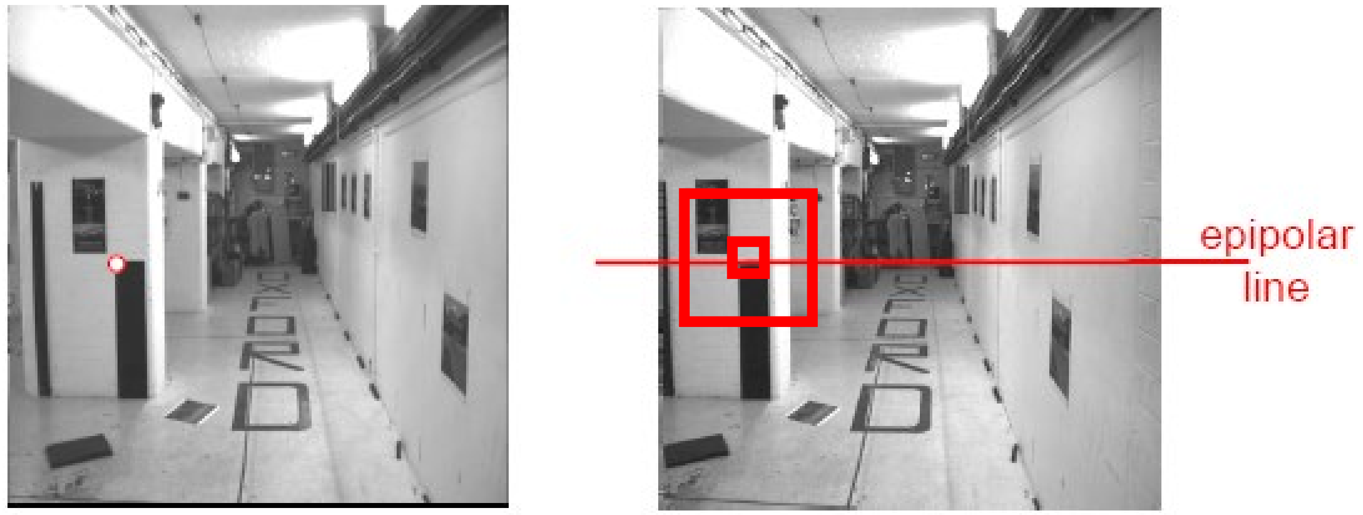
Textureless regions: the aperture problem



Textureless regions are not distinctive; high ambiguity for matches.

Textureless regions: the aperture problem

Solution: increase window size



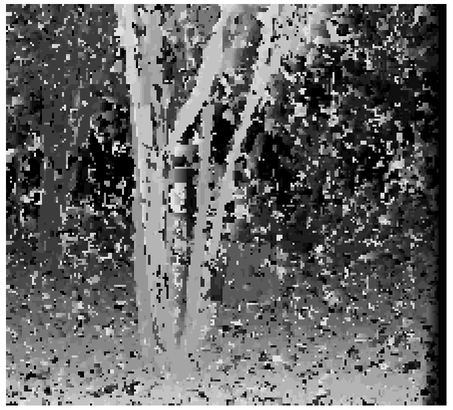
Effects of window size on the disparity map

Smaller window

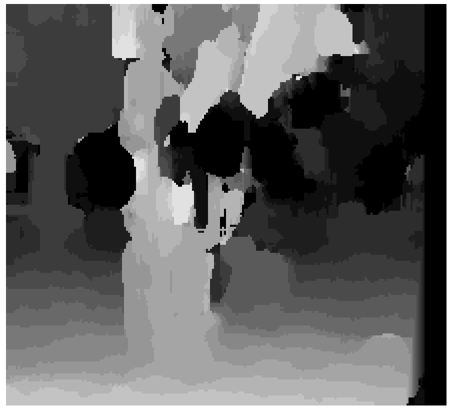
- more detail 👍
- but more noise 👎

Larger window

- smoother disparity maps 👍
- but less detail 👎



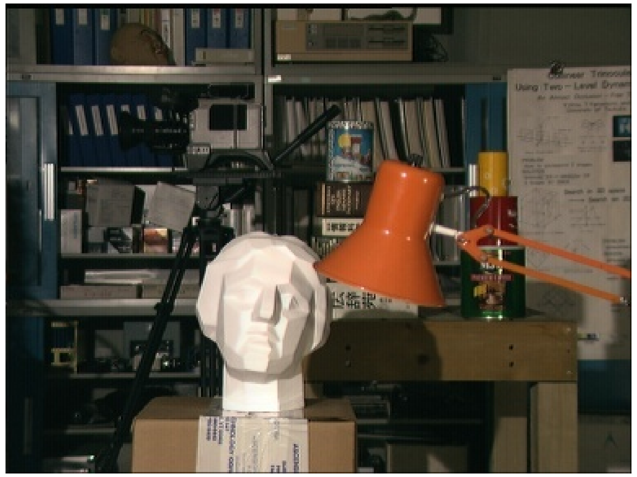
W = 3



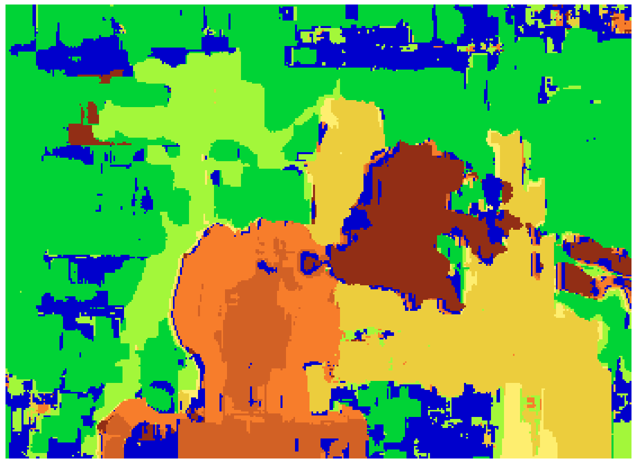
W = 20

Accuracy

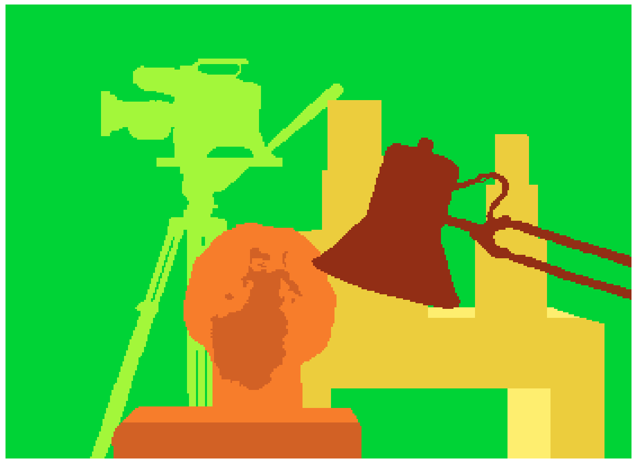
Data



Block matching



Ground truth



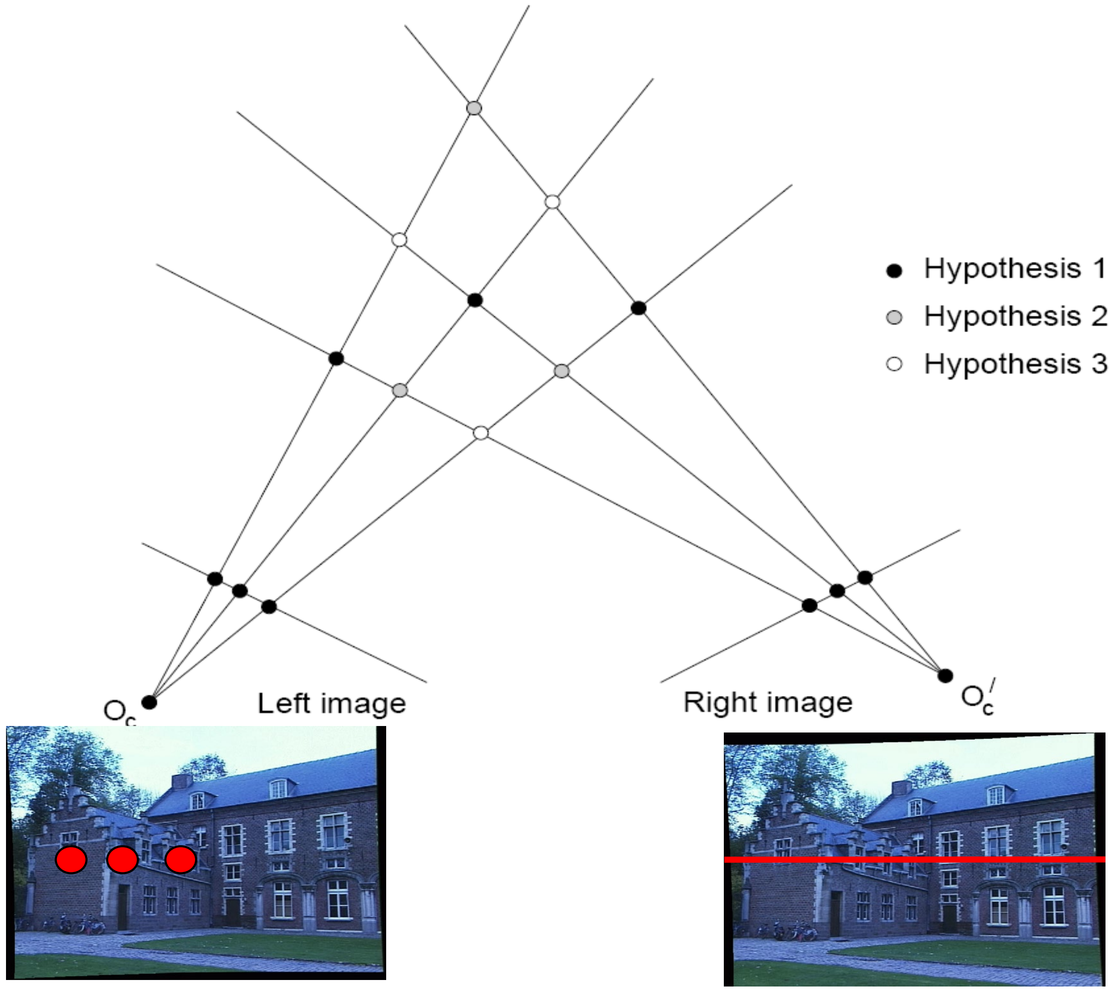
Challenges

Occlusions and repetitive patterns

Non-Lambertian surfaces (e.g., specularities), textureless surfaces

Correspondence Problems: Multiple matches

- Multiple match hypotheses satisfy epipolar constraint, but which one is correct?



How can we improve window-based matching?

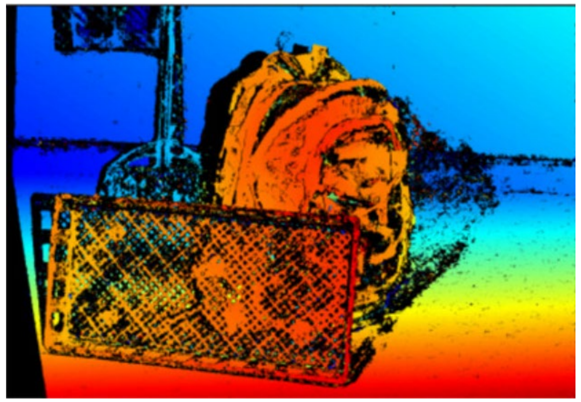
- Beyond the epipolar constraint, there are “soft” constraints to help identify corresponding points
  - Uniqueness
    - Only one match in right image for every point in left image
  - Ordering
    - Points on **same surface** will be in same order in both views
  - Disparity gradient
    - Disparity changes smoothly between points on the same surface

Example: Semi-Global Matching (SGM)

- SGM is a popular **open-source algorithm** that estimates a dense disparity map from a rectified stereo image pair

Left Image

Right Image



Estimated Disparity

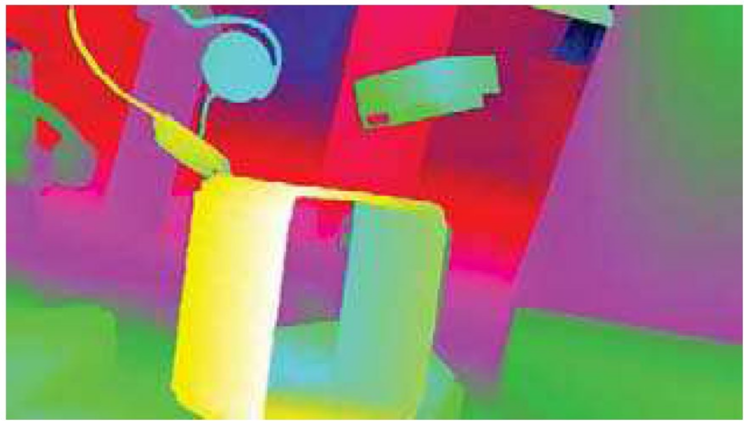
- **Main idea:** Perform **coarse-to-fine block matching followed by regularization** (e.g. **smoothing**): the estimated disparity map is a piece-wide smooth **surface** passing through the initial disparity map (see Lecture 12a)

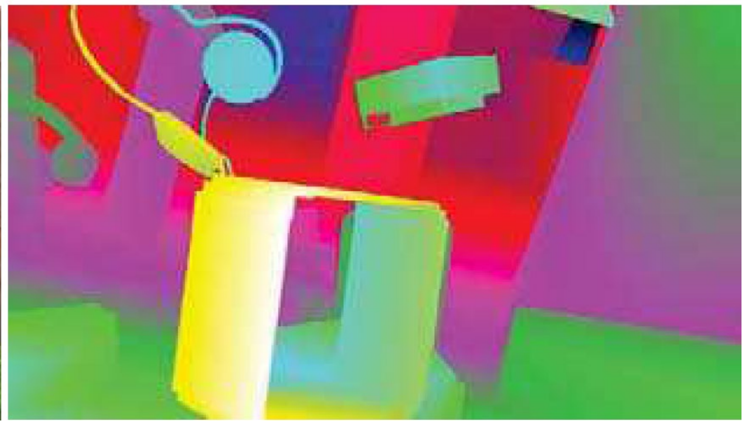
Hirschmuller, *Stereo processing by semiglobal matching and mutual information*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2007. PDF. Code.

Better methods exist

For the **latest and greatest**:

- **Middlebury dataset** and leader board: http://vision.middlebury.edu/stereo/
- **KITTI dataset** and leader board: http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo


Using Deep Learning


Ground truth

Jia-Ren Chang Yong-Sheng Chen, "Pyramid Stereo Matching Network", International Conference on Pattern Recognition, CVPR'18. PDF.

Things to Remember

- Disparity
- Triangulation: simplified and general case, linear and non linear approach
- Choosing the baseline
- Correspondence problem: epipoles, epipolar lines, epipolar plane
- Stereo rectification

Reading

- Szeliski book 2nd edition: Chapter 12
- Autonomous Mobile Robot book (link): Chapter 4.2.5
- Peter Corke book: Chapter 14.3

Understanding Check

Are you able to answer the following questions?

- Can you relate Structure from Motion to 3D reconstruction? What's their difference?
- Can you define disparity in both the simplified and the general case?
- Can you provide a mathematical expression of depth as a function of the baseline, the disparity and the focal length?
- Can you apply error propagation to derive an expression for depth uncertainty? How can we improve the uncertainty?
- Can you analyze the effects of a large/small baseline?
- What is the closest depth that a stereo camera can measure?
- Are you able to show mathematically how to compute the intersection of two lines (linearly and non-linearly)?
- What is the geometric interpretation of the linear and non-linear approaches and what error do they minimize?
- Are you able to provide a definition of epipole, epipolar line and epipolar plane?
- Are you able to draw the epipolar lines for two converging cameras, for a forward motion situation, and for a side-moving camera?
- Are you able to define stereo rectification and to derive mathematically the rectifying homographies?
- How is the disparity map computed?
- How can one establish stereo correspondences with subpixel accuracy?
- Describe one or more simple ways to reject outliers in stereo correspondences.
- Is stereo vision the only way of estimating depth information? If not, are you able to list alternative options? (make link to other lectures)