Computer Vision Group

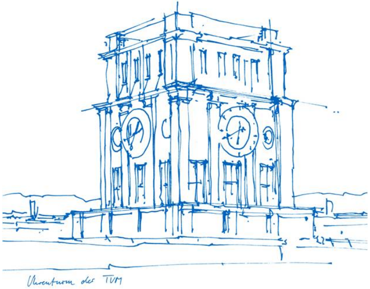
Technische Universität München

Computer Vision II: Multiple View Geometry (IN2228)

Chapter 06 2D-2D Geometry
(Part 3 3D Reconstruction)

Dr. Haoang Li

14 June 2023 12:00-13:30



Announcement Before Class

➢ Updated Lecture Schedule

For updates, slides, and additional materials:
https://cvg.cit.tum.de/teaching/ss2023/cv2

90-minute course; 45-minute course

Foundation

Wed 19.04.2023	Chapter 00: Introduction
Thu 20.04.2023	Chapter 01: Mathematical Backgrounds
Wed 26.04.2023	Chapter 02: Motion and Scene Representation (Part 1)
Thu 27.04.2023	Chapter 02: Motion and Scene Representation (Part 2)
Wed 03.05.2023	Chapter 03: Image Formation (Part 1)
Thu 04.05.2023	Chapter 03: Image Formation (Part 2)
Wed 10.05.2023	Chapter 04: Camera Calibration
Thu 11.05.2023	Chapter 05: Correspondence Estimation (Part 1)
Wed 17.05.2023	Chapter 05: Correspondence Estimation (Part 2)
Thu 18.05.2023	No lecture (Public Holiday)
Wed 24.05.2023	No lecture (Conference)
Thu 25.05.2023	No lecture (Conference)

} Videos and reading materials about the combination of deep learning and multi-view geometry

Core part

Wed 31.05.2023	Chapter 05: Correspondence Estimation (Part 3)
Thu 01.06.2023	Chapter 06: 2D-2D Geometry (Part 1)
Wed 07.06.2023	Chapter 06: 2D-2D Geometry (Part 2)
Thu 08.06.2023	No lecture (Public Holiday)
Wed 14.06.2023	Chapter 06: 2D-2D Geometry (Part 3)
Thu 15.06.2023	Chapter 06: 2D-2D Geometry (Part 4)
Wed 21.06.2023	Chapter 07: 3D-2D Geometry
Thu 22.06.2023	Chapter 08: 3D-3D Geometry
Wed 28.06.2023	Chapter 09: Single-view Geometry
Thu 29.06.2023	Chapter 10: Combination of Different Configurations

Advanced topics and high-level tasks

Wed 05.07.2023	Chapter 11: Photometric Error (Direct Method)
Thu 06.07.2023	Chapter 12: Bundle Adjustment and Optimization
Wed 12.07.2023	Chapter 13: Robust Estimation
Thu 13.07.2023	Question Explanation and Knowledge Review
Wed 19.07.2023	Chapter 14: SLAM and SFM (Part 2)
Thu 20.07.2023	Chapter 14: SLAM and SFM (Part 1)

Today’s Outline

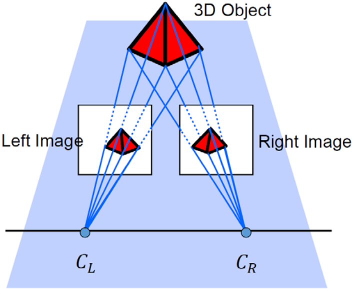
- Overview of 3D Reconstruction
- Triangulation (General Case)
- Stereo Vision (Simplified Case)

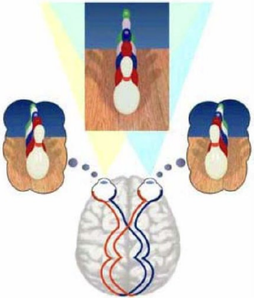
Overview

➢ Intuitive Illustration

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

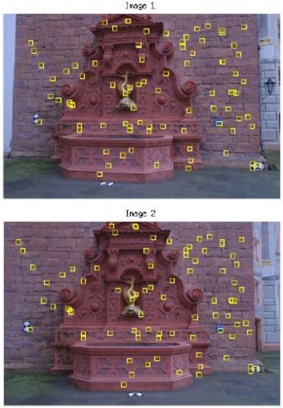
✓ Working principle of human eye: Objects projected on our retinas are up-side-down, but our brain makes us perceive them as upright objects.



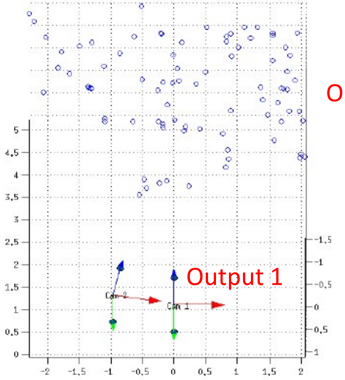


Overview

- Input and Output



Input: 2D-2D point correspondence



Output 2

Output 1

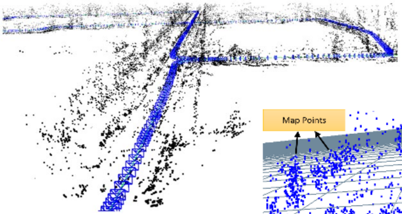
Estimated poses and 3D structure

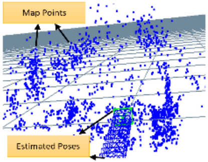
Overview

➢ Classification

✓ General case (for sparse reconstruction)

- Triangulation



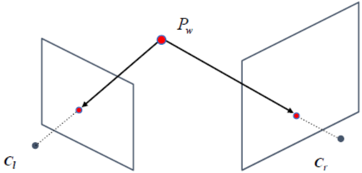


(a)

(b)

General case

(non identical cameras and not aligned)



Overview

- ➢ Classification

✓ Simplified case (for dense reconstruction)

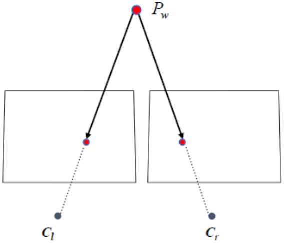
- Depth from disparity

Input Stereo Sequence

Simplified case

(identical cameras and aligned)

virtual

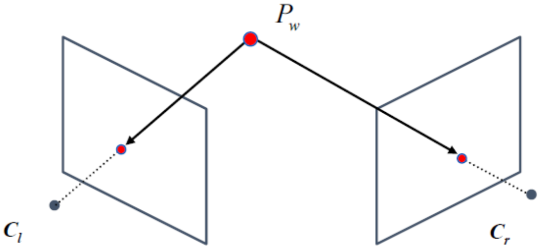


Triangulation

➢ Overview

✓ Prior information

- Extrinsic parameters (relative rotation and translation) obtained by epipolar constraint (or the other methods, e.g., PnP and ICP).
- Intrinsic parameters (focal length , principal point of each camera). We can obtain them by using a calibration method e.g., Tsai’s method or Zhang’s method.

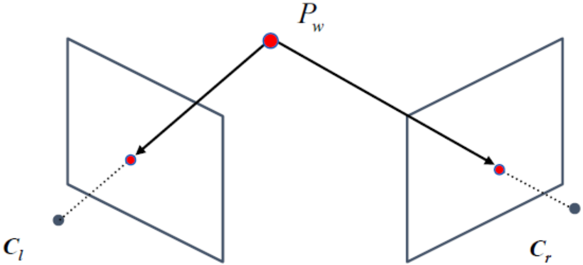


Triangulation

➢ Overview

✓ Definition

Triangulation is the problem of determining the **3D position of a point** given a set of **corresponding 2D points** and known **camera poses**.



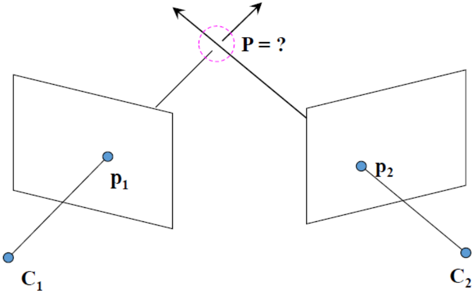
Triangulation

➢ Overview

✓ Definition

We want to **intersect** the two projection rays corresponding to p_1 and p_2.

Because of noise and numerical errors, two rays won't meet exactly, so we can only compute an approximation.



Triangulation

➢ Basic Constraints

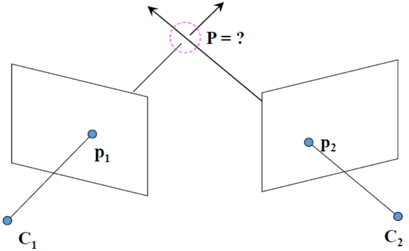
In the left camera frame, we have the perspective projection constraints:

Left camera:

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

Right camera:

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



We express 3D point in the **left camera frame**.

Triangulation

- Basic Constraints

We generate the system of equations of the left and right cameras:

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

Known

Collinearity (up-to-scale)

Cross product

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

Triangulation

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

- Least Square Approximation

We get a homogeneous system of equations

Left camera: $0 = p_1 \times M_1 \cdot P \quad \Rightarrow [p_{1\times}] \cdot M_1 \cdot \boxed{P} = 0$ Two independent linear constraints

Unknown

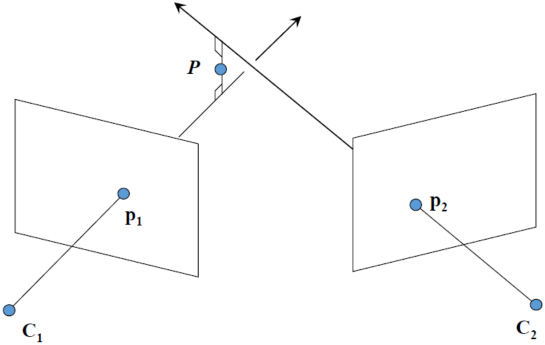
Right camera: $0 = p_2 \times M_2 \cdot P \quad \Rightarrow [p_{2\times}] \cdot M_2 \cdot \boxed{P} = 0$ Two independent linear constraints

We get a homogeneous system of equations.
Mathematically, 3D point ***P*** can be determined using SVD.

Triangulation

- Least Square Approximation

Geometrically, ***P*** is computed as the midpoint of **the shortest 3D line segment** connecting the two lines.

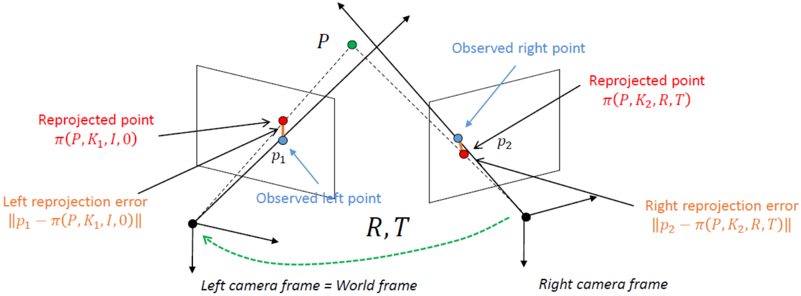


Triangulation

➢ Follow-up Non-linear Optimization (Optional)

- ✓ Initialize ***P*** using the least square approximation introduced before
- ✓ Refine ***P*** by minimizing the sum of left and right squared re-projection errors:
- ✓ We can only optimize 3D point, or jointly optimize pose and 3D point.

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



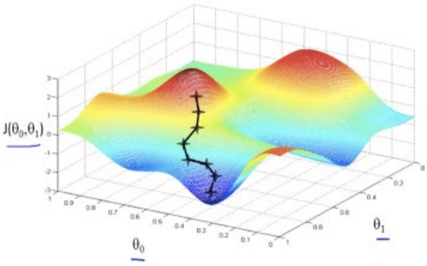
Triangulation

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

- Non-linear Optimization

The reprojection error can be minimized using Levenberg-Marquardt (more robust than Gauss-Newton method to local minima)

The gradient-descent algorithms will be introduced in the future.



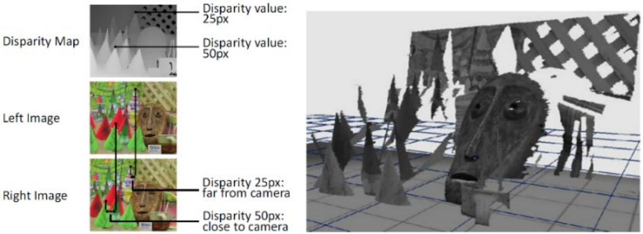
Stereo Vision

➢ Overview

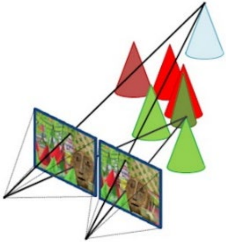
✓ Input: known extrinsic camera parameters measured/calibrated beforehand

✓ Main knowledge

- Disparity and Depth
- Stereo Rectification
- Dense Correspondence Establishment (introduced in the next class)



Disparity and depth



Stereo rectification

Stereo Vision

➢ Disparity and Depth

✓ Intuitive illustration

- Our brain allows us to perceive **disparity (displacement vector of a point)** from the left and right images.
- **Depth** is inversely proportional to the **disparity**.

Image from the left eye

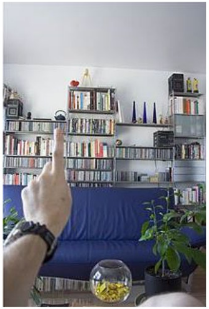
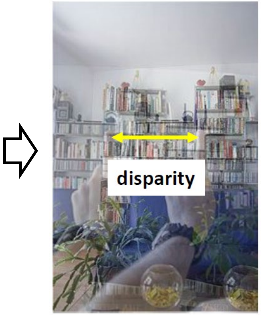
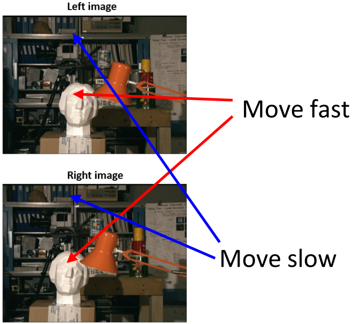


Image from the right eye



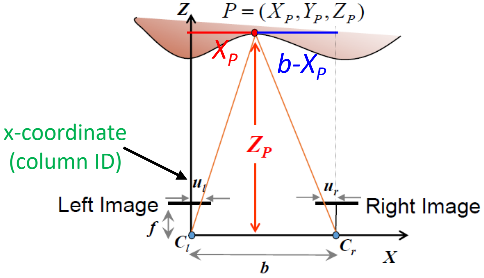


Stereo Vision

➢ Disparity and Depth

✓ Mathematical computation

Assume that both cameras are identical (i.e., have the same intrinsic parameters) 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

Stereo Vision

➢ Disparity and Depth

✓ Mathematical computation

Once the stereo pair is rectified, the **disparity and depth** of each point can be computed.

Baseline

Focal length

$$Z_P = \frac{bf}{u_l - u_r}$$

Depth

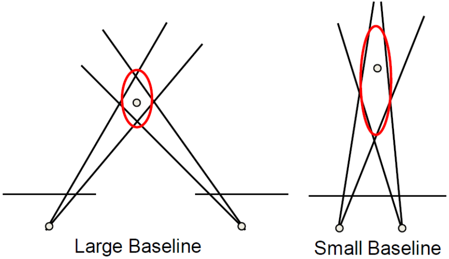
Disparity

Stereo Vision

➢ Disparity and Depth

✓ What's the optimal baseline?

$$Z_P = \frac{\boxed{b}f}{u_l - u_r}$$



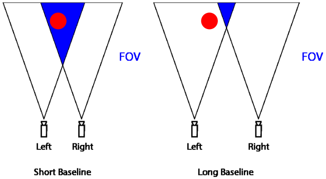
Large baseline:

- Advantage: Small depth error
- Disadvantage: Difficult search problem for close objects (projection may be outside the right image)

Small baseline:

- Advantage: Large depth error
- Disadvantage: Cons: Easier search problem for close objects

Outside the FOV of right camera

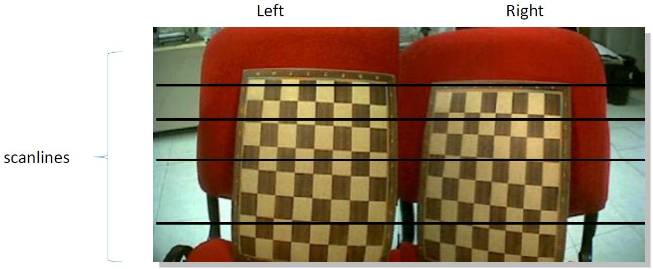


Stereo Vision

➢ Stereo Rectification

✓ Motivation

- In our previous derivations, we assume that the image pairs have been rectified.
- Even for a commercial stereo camera, the left and right images are never perfectly aligned.
- In practice, **it is convenient if the epipolar lines are aligned to the horizontal scanlines** because the correspondence search can be very efficient (only search the point along the same scanlines).



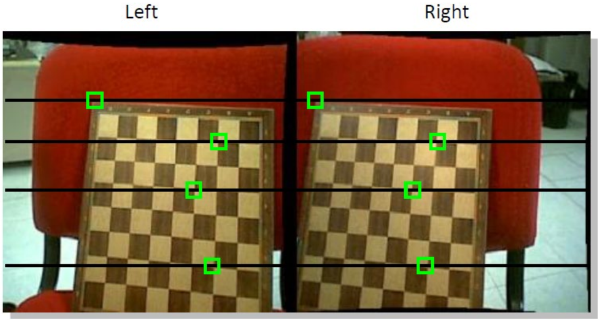
Raw stereo pair (**unrectified**): scanlines do not coincide with epipolar lines

Stereo Vision

➢ Stereo Rectification

✓ Definition

- Stereo rectification warps the left and right images into new “rectified” images such that the **epipolar lines coincide with the horizontal scanlines.**



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

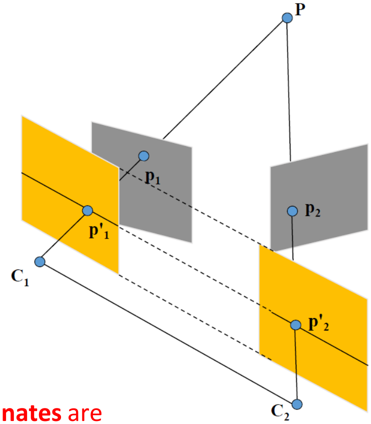
Stereo Vision

➢ Stereo Rectification

✓ Overview

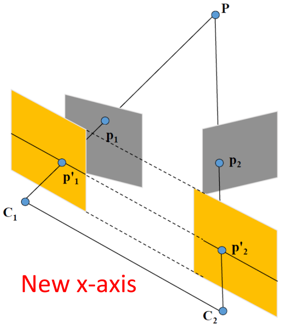
- Warps the original image planes onto a (virtual) planes **parallel to the baseline**
- It works by computing two transformations, one for each image
- As a result, the new epipolar lines are aligned to the horizontal scanlines.

Such a transformation describes **2D coordinate transformation**.
If an image plane is transformed in 3D, a **2D projection point’s coordinates** are changed accordingly.



Stereo Vision

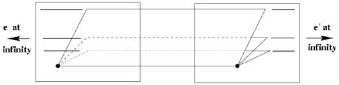
➤ Stereo Rectification



✓ Pipeline

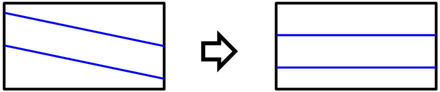
- We define two new matrices to **rotate** the old image planes **respectively** around their optical centers. The new image planes become coplanar, and are both parallel to the baseline.
- This ensures that **epipolar lines are parallel**.

- To have **horizontal** (not just parallel) epipolar lines, the baseline must be parallel to the new X axis of both new camera frames.



Left image

Right image



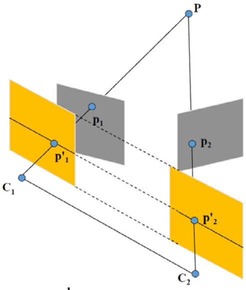
Stereo Vision

➢ Overview

✓ Pipeline

- In addition, to have a proper rectification, corresponding points must **have the same y-coordinate (row ID)**. This is obtained by requiring that the new cameras **have the same intrinsic parameters**.

- In other words, the **displacement of a 2D point** in the image is **only** caused by extrinsic parameters (relative pose of camera).



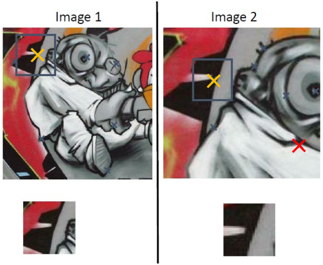


Image pairs obtained by different **focal lengths**

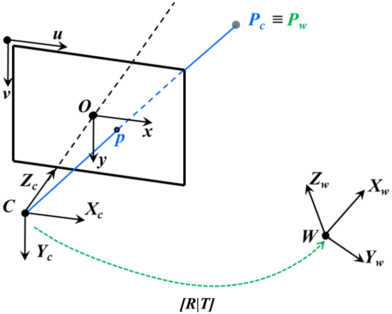
Stereo Vision

➢ Detailed Procedures (Step 1)

✓ Recap on perspective projection

The perspective equation for a point P_w in the world frame is defined by the following 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 Vision

$$\boxed{Y = RX + t \quad \Rightarrow \quad X = R^T(Y-t) = \boxed{R^T}Y \boxed{-R^Tt}}$$

Inverse transformation introduced before

- Detailed Procedures (Step 1)

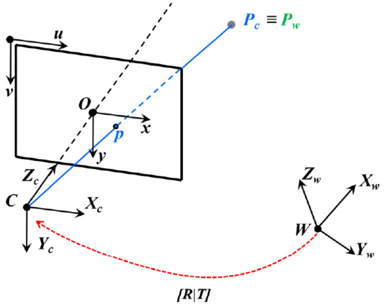
✓ For Stereo Vision, however, it is more common to use $\boldsymbol{R} \equiv \boldsymbol{R}_{wc}$ and $\boldsymbol{T} \equiv \boldsymbol{T}_{wc}$, where now $\boldsymbol{R}$, and $\boldsymbol{T}$ transform points from the **Camera frame** to the **World frame**.

✓ This is more convenient because $\boldsymbol{T}$=$\boldsymbol{C}$ directly represents the **coordinates** of the camera center **in the world frame** (see page 17/57 of Chapter02 Part1).

✓ The projection equation can be re written as:

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

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



Stereo Vision

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

- Detailed Procedures (Step 2)

We can now write the Perspective Equation for the **Left** and **Right** cameras, respectively. Here, we assume that Left and Right cameras have different intrinsic parameter matrices, K

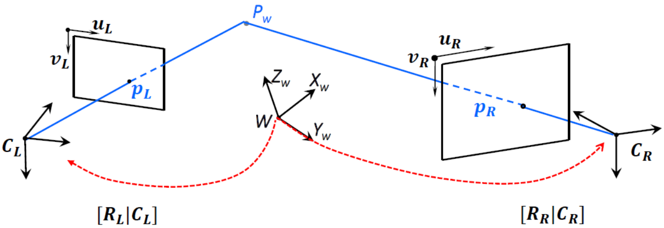
Left camera

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

Right camera

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

If matrices $\{K_L, K_R\}$ are the same, and $\{R_L, R_R\}$ are the same, image planes are aligned.



Stereo Vision

➢ Detailed Procedures (Step 3)

The goal of stereo rectification is to warp the left and right camera images such that their image planes are aligned (by introducing the same **new** rotation $\hat{\boldsymbol{R}}$ and **new** intrinsic parameters $\hat{\boldsymbol{K}}$).

If an image plane is transformed in 3D, a **2D projection's coordinates** are changed accordingly.

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

Old Left camera

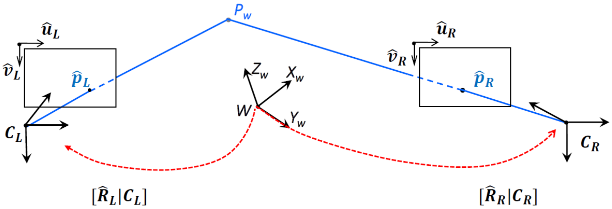
Relation expression in 2D?

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

New Left camera

Computed image coordinates

Known matrices that are **manually assigned**.



Stereo Vision

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

Old Left camera

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

New Left camera

➢ Detailed Procedures (Step 4)

✓ Coordinate change in 2D can be expressed by a 3*3 transformation.

✓ We first compute the transformation, and then use it to compute the warped coordinates:

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

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

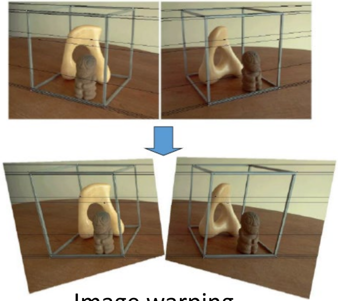
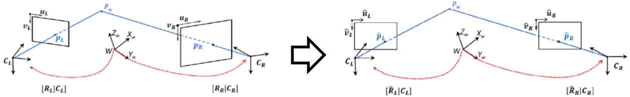


Image warping

Stereo Vision

➢ Intrinsic and Rotation Matrices



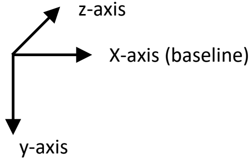
Origins of cameras remain unchanged

✓ How do we choose the new $\hat{\boldsymbol{K}}$? A common choice is to take the arithmetic average of K_L and K_R

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

✓ How do we choose the new $\hat{\boldsymbol{R}} = [\hat{r}_1, \hat{r}_2, \hat{r}_3]$ with $\hat{r}_1, \hat{r}_2, \hat{r}_3$ being the column vectors of $\hat{R}$?

A common choice is as follows:



$$\hat{r}_1 = \frac{C_R - C_L}{\|C_R - C_L\|}$$ This makes the new image planes parallel to the baseline

$$\hat{r}_2 = r_{3L} \times \hat{r}_1$$ where r_{3L} is the 3rd column of the rotation matrix of the left camera, i.e., R_L

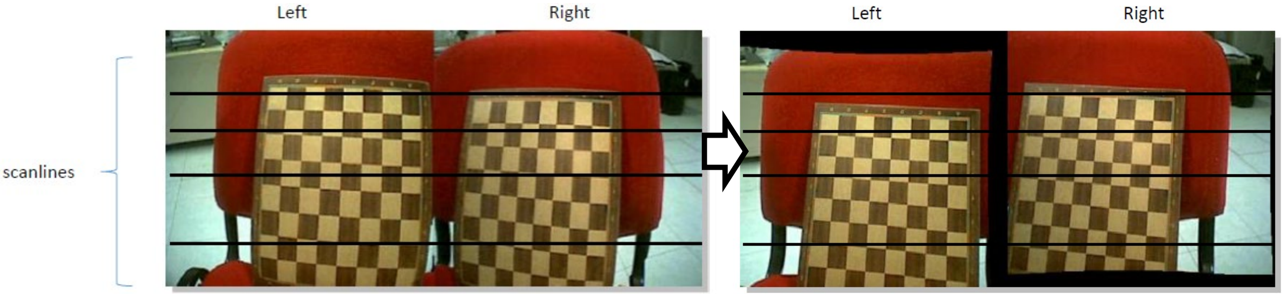
Old r_3 New r_1

$$\hat{r}_3 = \hat{r}_1 \times \hat{r}_2$$

Stereo Vision

- ➢ Example

✓ Preprocessing step: image undistortion (introduced before)



Input image pair

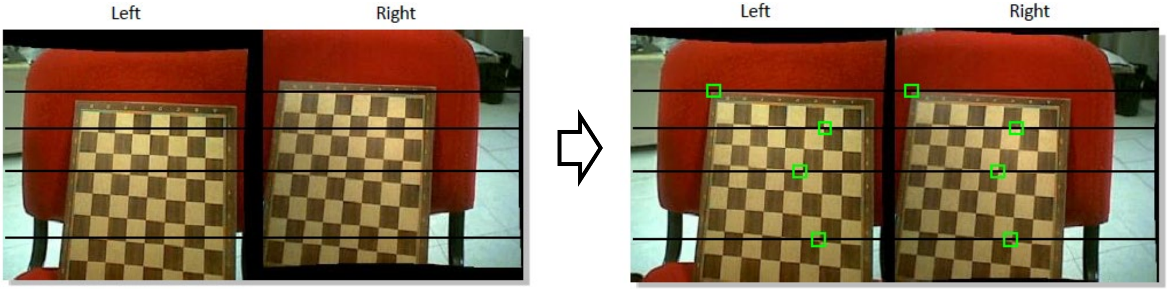
Compute lens distortion

Stereo Vision

- Example

✓ Then, compute transformation to rectify/warp images.

✓ Use **interpolation** to generate the warped image. The transformed coordinates are float numbers, but pixel coordinates are typical integer numbers. (if necessary, I will introduce how to solve this problem in the future).

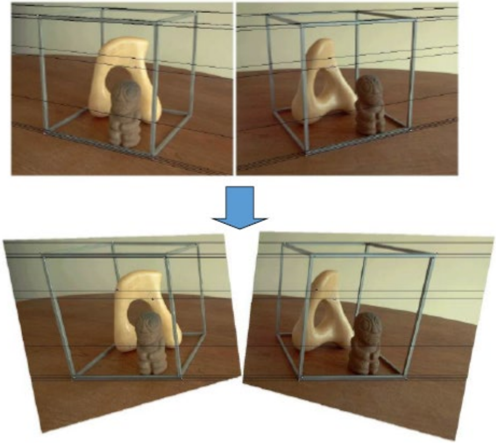


Stereo Vision

- ➢ Follow-up Task of 3D Reconstruction

- ✓ Result of stereo rectification: Corresponding epipolar lines are horizontal and collinear.

- ✓ We can conduct the 1D dense correspondence search.

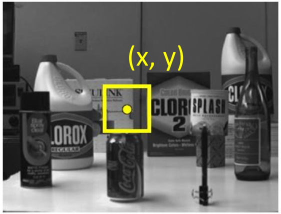


Stereo Vision

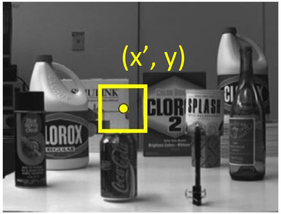
➢ Follow-up Task of 3D Reconstruction

✓ From Rectified Image Pair to Disparity Map

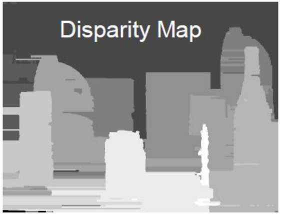
- For every pixel in the left image, find its corresponding point in the right image based on descriptor similarity (introduced in the next class).
- Compute the **disparity** for each found pair of correspondences, i.e., x’-x



Left image



Right image



Close objects experience bigger disparity
→ appear brighter in disparity map

Stereo Vision

➢ Follow-up Task of 3D Reconstruction

✓ From Disparity Map to 3D Point Cloud

- Once the disparity is obtained, the depth of each point can be computed recalling that:

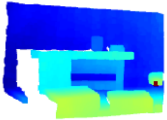
Baseline Focal length

$$Z_P = \frac{bf}{u_l - u_r}$$

Depth Disparity

- From depth to 3D (introduced before)

$$\begin{cases} z = depth(i, j) \\ x = \dfrac{(j - c_x) \times z}{f_x} \\ y = \dfrac{(i - c_y) \times z}{f_y} \end{cases}$$



Summary

- Overview of 3D Reconstruction
- Triangulation (General Case)
- Stereo Vision (Simplified Case)

Thank you for your listening!

If you have any questions, please come to me :-)