Week 9 – Robotic Vision 2

Advanced Robotic Systems – MANU2453

Dr Ehsan Asadi, School of Engineering
RMIT University, Victoria, Australia
Email: ehsan.asadi@rmit.edu.au

RMIT UNIVERSITY

Lectures

Wk	Date	Lecture (NOTE: video recording)	Maths Difficulty	Hands-on Activity	Related Assessment
1	24/7	• Introduction to the Course • Spatial Descriptions & Transformations			
2	31/7	• Spatial Descriptions & Transformations • Robot Cell Design			Robot Cell Design Assignment
3	7/8	• Forward Kinematics • Inverse Kinematics			
4	14/8	• ABB Robot Programming via Teaching Pendant • ABB RobotStudio Offline Programming		ABB RobotStudio Offline Programming	Offline Programming Assignment
5	21/8	• Jacobians: Velocities and Static Forces			
6	28/8	• Manipulator Dynamics			
7	11/9	• Manipulator Dynamics		MATLAB Simulink Simulation	
8	18/9	• Robotic Vision		MATLAB Simulation	Robotic Vision Assignment
9	25/9	• Robotic Vision II		MATLAB Simulation	
10	2/10	• Trajectory Generation			
11	9/10	• Linear & Nonlinear Control		MATLAB Simulink Simulation	
12	16/10	• Introduction to I4.0 • Revision			Final Exam

Content

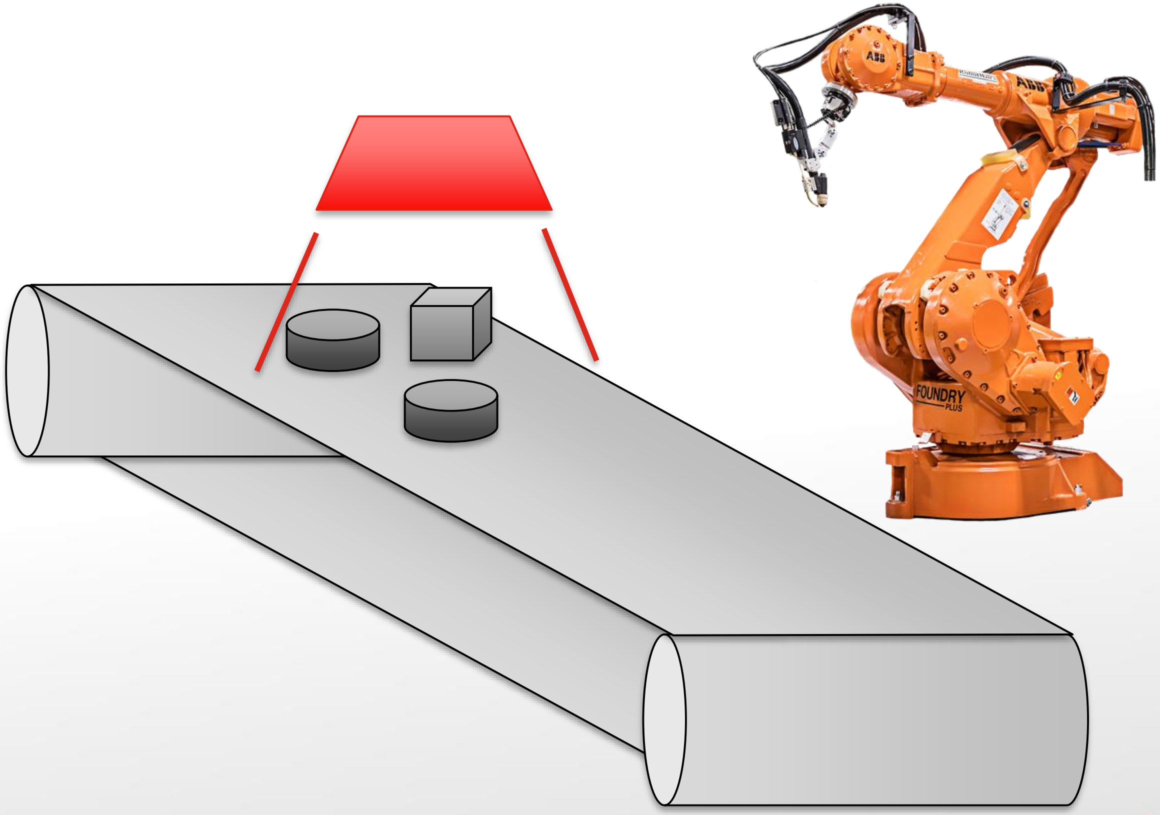
- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Multiple Blobs

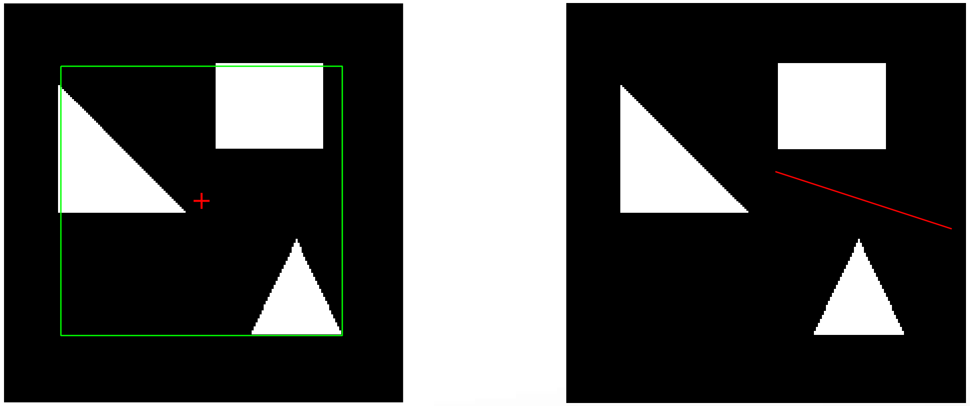
- Last week, we have learnt how to identify the position of single object.
- What happens if a few objects are within the field of view of the camera?



https://www.robots.com/articles/the-speed-of-abb-arc-welding-robots

Multiple Blobs

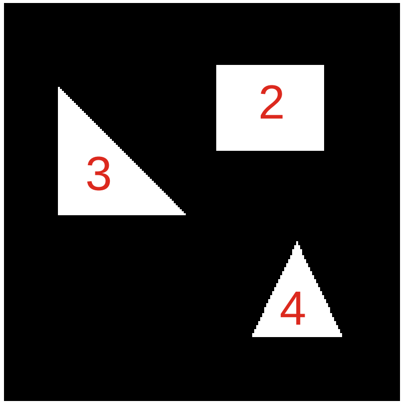
- If we were to use the previously-mentioned methods to find the bounding box, centroids etc., we will end up having the following results (example):



- As can be seen, the algorithms see all the objects as 1 big blob.
- This creates wrong results.

Multiple Blobs

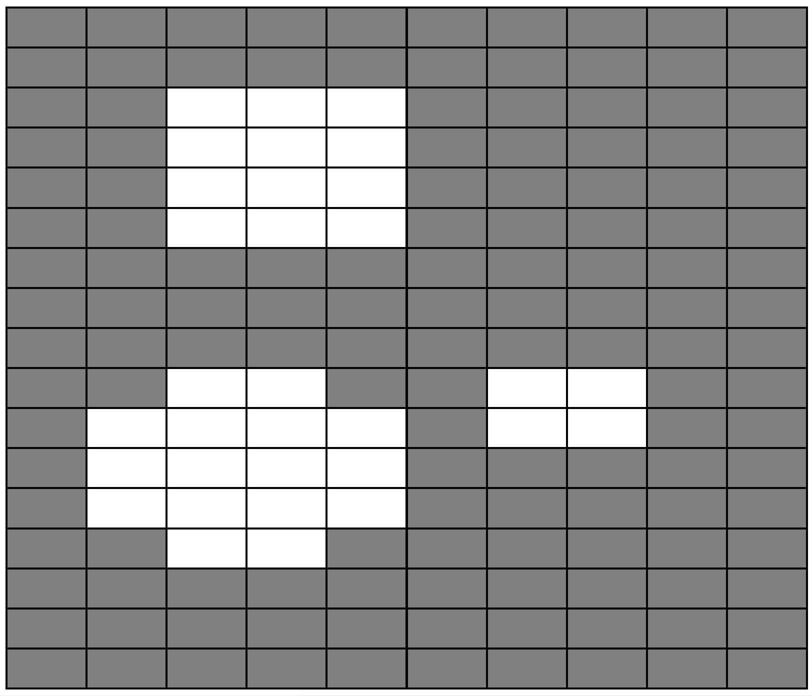
- To solve this problem, we need a way to label the blobs individually:



- After this, we can then call all previously-learned algorithms to work specifically for blob number 2, 3, or 4.
- Question: How do we create the labels?

Connected Components

- The basic idea is straightforward.
- Given the following image (not the same as previous slides):
- Firstly, label all the background as 0 and the foreground as 1.



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- Find the first pixel which is a foreground, and change the label from 1 to 2.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- Then, for each of the foreground pixels from left to right and from top to bottom, check if any of the adjacent top and left pixels has been labelled 2.
  - If yes, change the label to 2 as well.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- Continue on the same process, and we will get:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- After some time, we will obtain:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- Continue checking, for each of the foreground pixels from left to right and from top to bottom, if any of the adjacent top and left pixels has been labelled 2.
  - If no, change the label to 3.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	3	1	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Connected Components

- Continuing on, we see that the highlighted foreground pixel has no adjacent pixels with labels 2 or 3.
  - Thus we label it 4.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	3	3	0	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	3	3	0	0	4	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

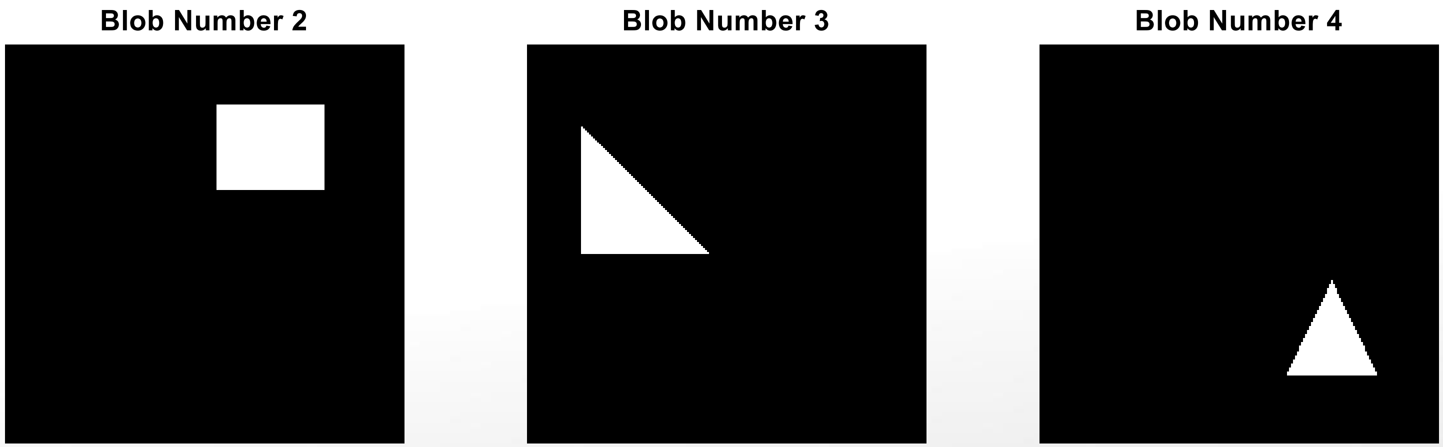
Connected Components

- Finally, we will get the result which we wanted:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	3	3	0	0	4	4	0	0
0	3	3	3	3	0	4	4	0	0
0	3	3	3	3	0	0	0	0	0
0	3	3	3	3	0	0	0	0	0
0	0	3	3	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

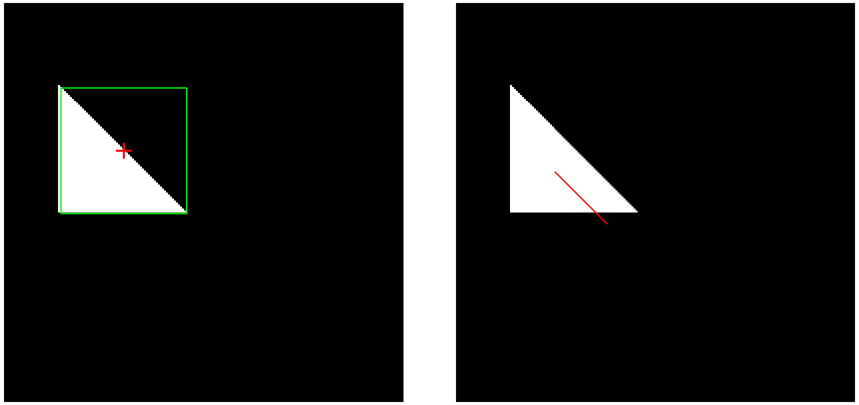
Labeled Blobs

- By using the algorithm, the blobs are separated individually, using the command:
  - Blob2 = (Label == 2);
  - Blob3 = (Label == 3); etc.

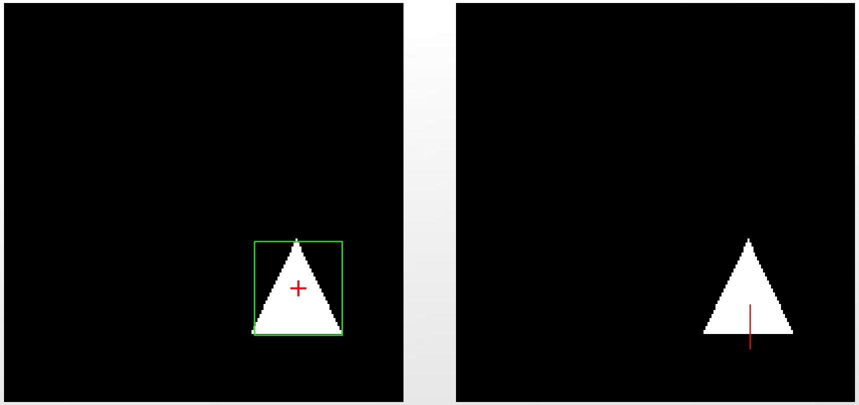


Individual Blob Analysis

- With this, we can now perform analysis on the individual blob.
- E.g. for blob = 3.



- And for blob = 4:



Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

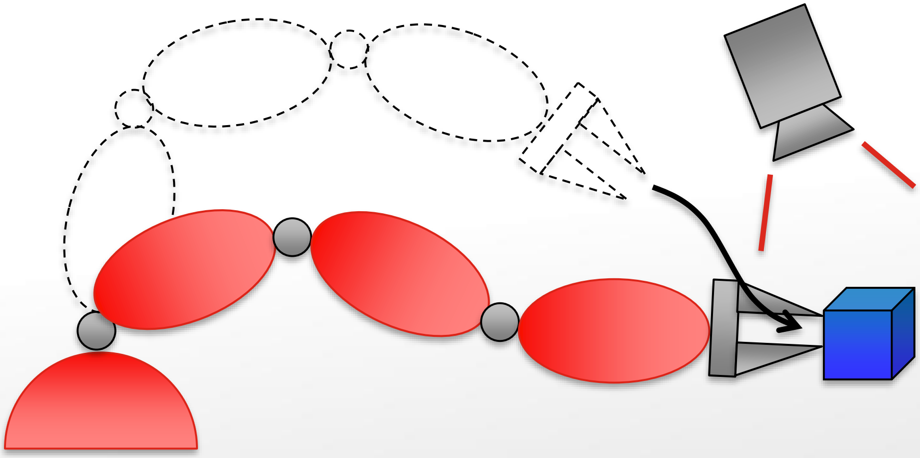
Introduction

- Last week, we have learnt a few techniques in robot vision or image processing to perform:
  - Feature extraction – e.g. detect edges, corners
  - Part identification – e.g. selecting conical shaped parts out of many different parts.
- Today, we will learn about:
  - Pose estimation – obtaining the 3D pose (translation and orientation) of parts, to allow robotic handling.

Robot identifying parts and esimating the 3D pose
https://i.ytimg.com/vi/mQpVCSM8Vgc/maxresdefault.jpg

Introduction

- The idea behind 3D pose estimation is to estimate the position and orientation of the object, with respect to a camera (location known to robot).
- Once these are known, we can command the robot to manipulate the object.



Introduction

- Estimation of the position/orientation of camera can be captured under the topic "Camera Calibration".
- The goal of camera calibration is to find out:
  - The intrinsic parameters of the camera:
    - Focal length
    - Scaling factor
    - Distortion
    - Etc.
  - The extrinsic parameters of the camera:
    - Translation to world coordinate frame
    - Rotation to world coordinate frame

This is what we were looking for

- We will obtain both the intrinsic and extrinsic parameters through the process of calibration, the latter representing the 3D pose of the camera.

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Image Formation

- Pinhole Projection Model:
  - Light ray comes through the pinhole (camera center), and is projected onto the film or CCD, which is at focal length, f, distance away from pinhole.

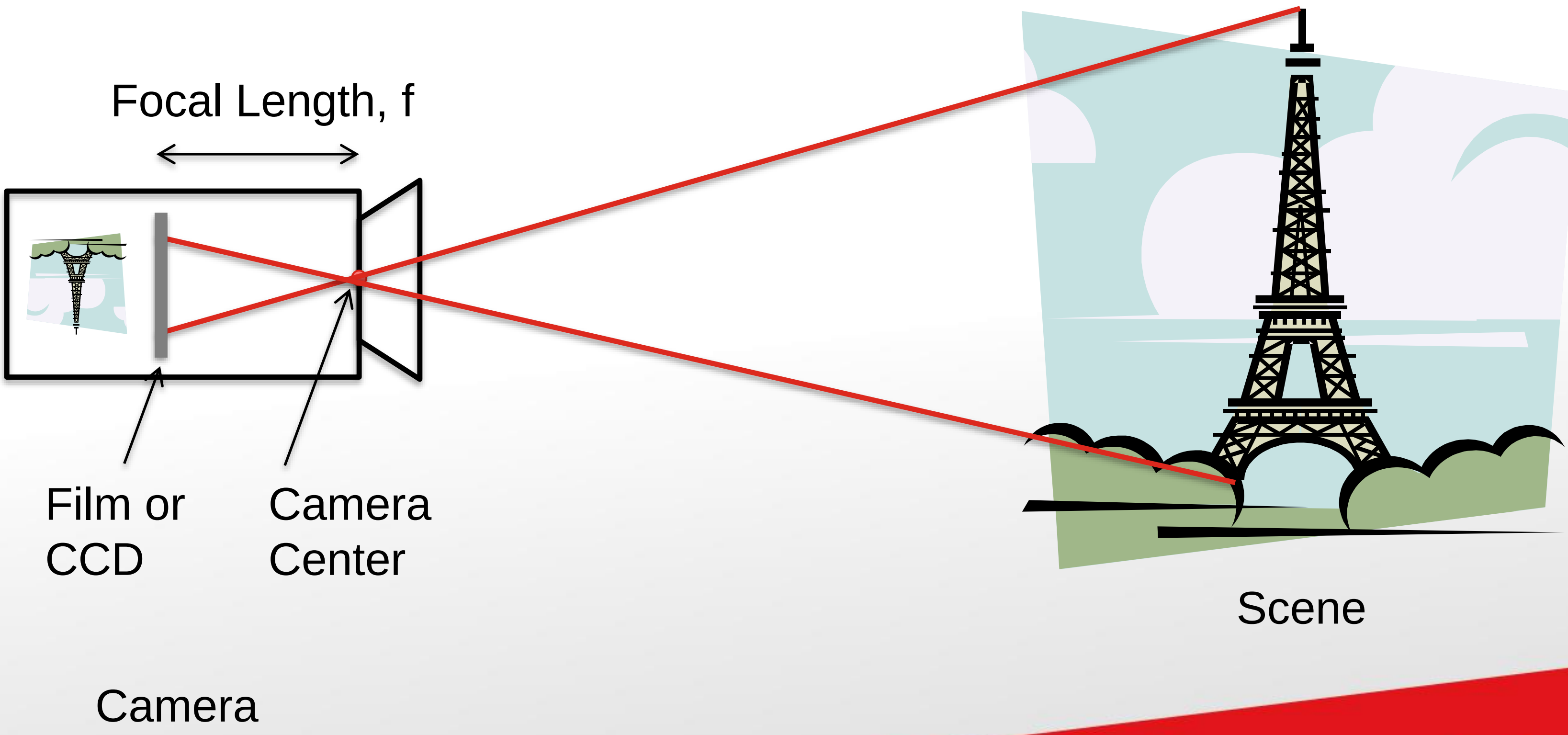


Image Formation

- It is obvious that the image will become upside down.
- To simplify calculation, it is proposed to have a "virtual" image plane at distance f in front of the camera instead, so that the image is not rotated.

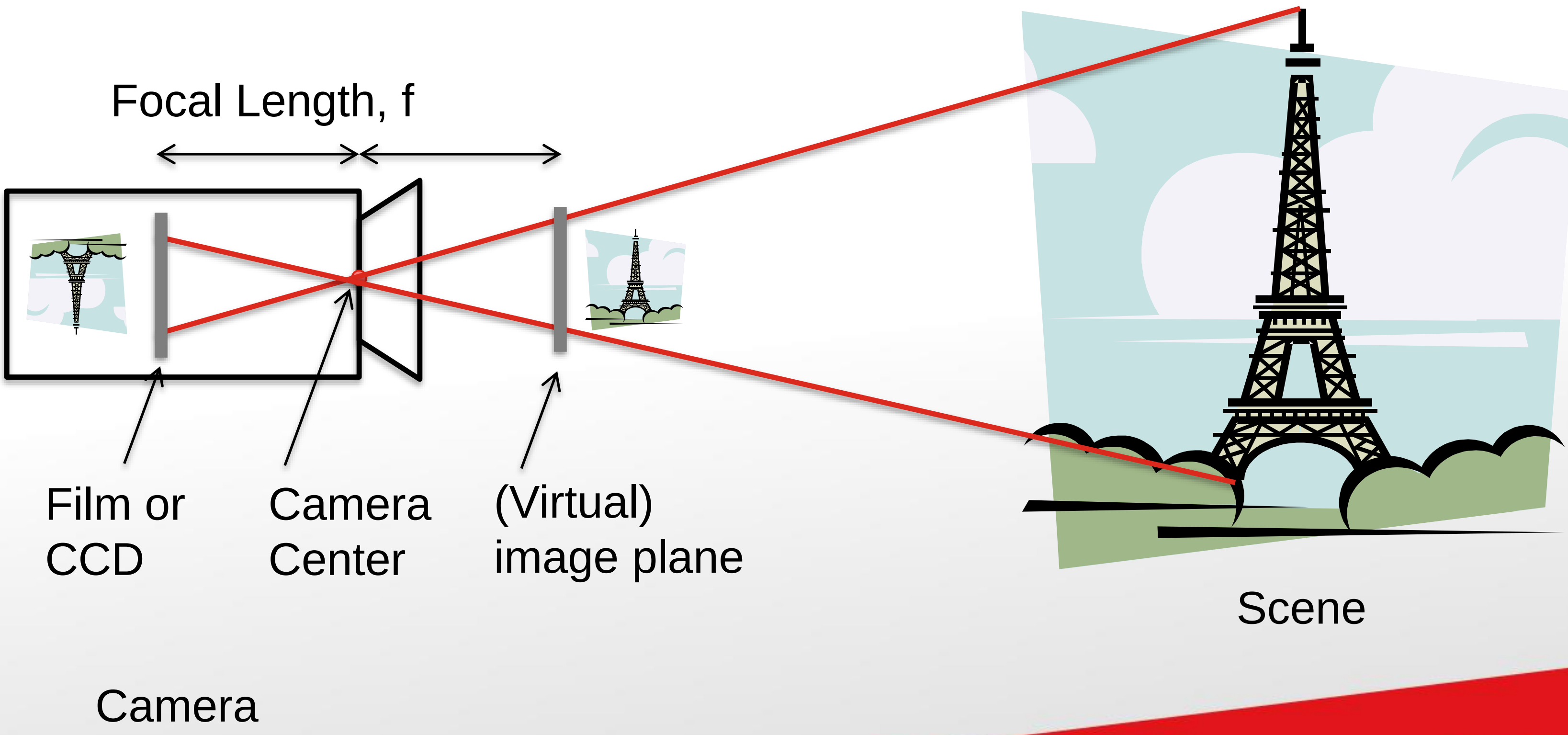
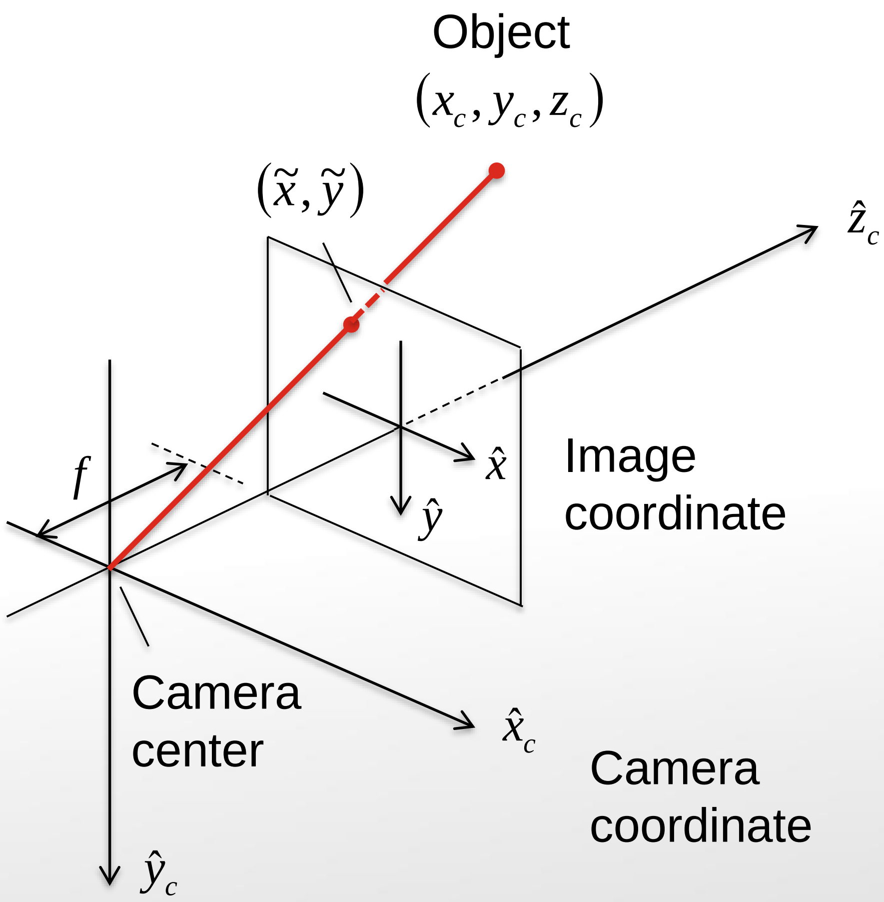
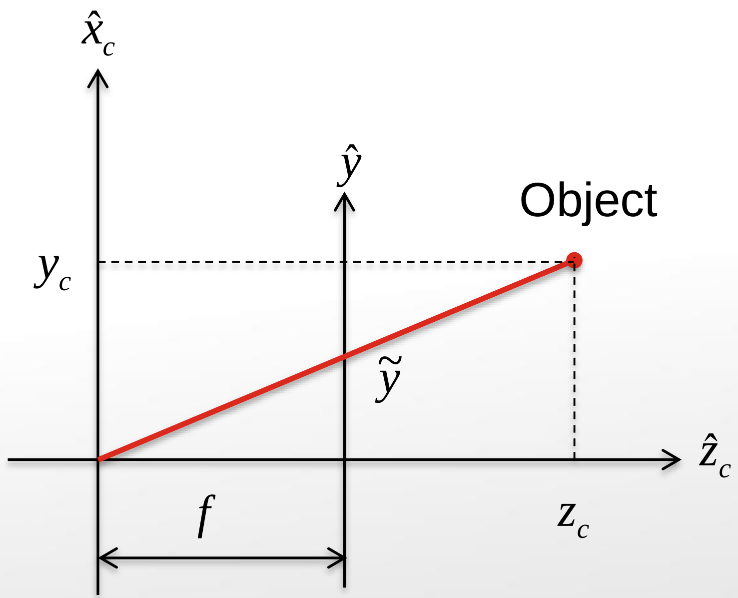


Image Formation

- The scenario is thus as follows:





Pinhole Projection Equation

- From the 2-dimensional sketch, it is easy to see that (due to similar triangles):

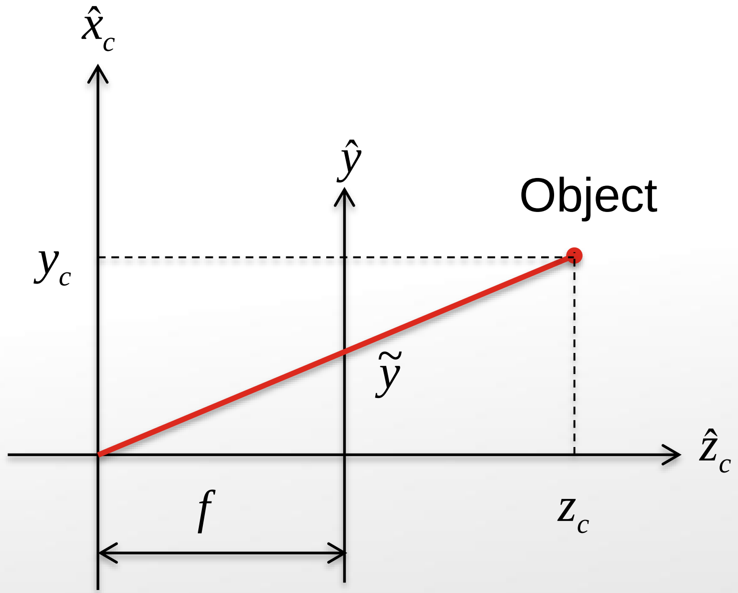
$$\frac{\tilde{y}}{f} = \frac{y_c}{z_c}$$

- This gives:

$$\tilde{y} = f\,\frac{y_c}{z_c}$$

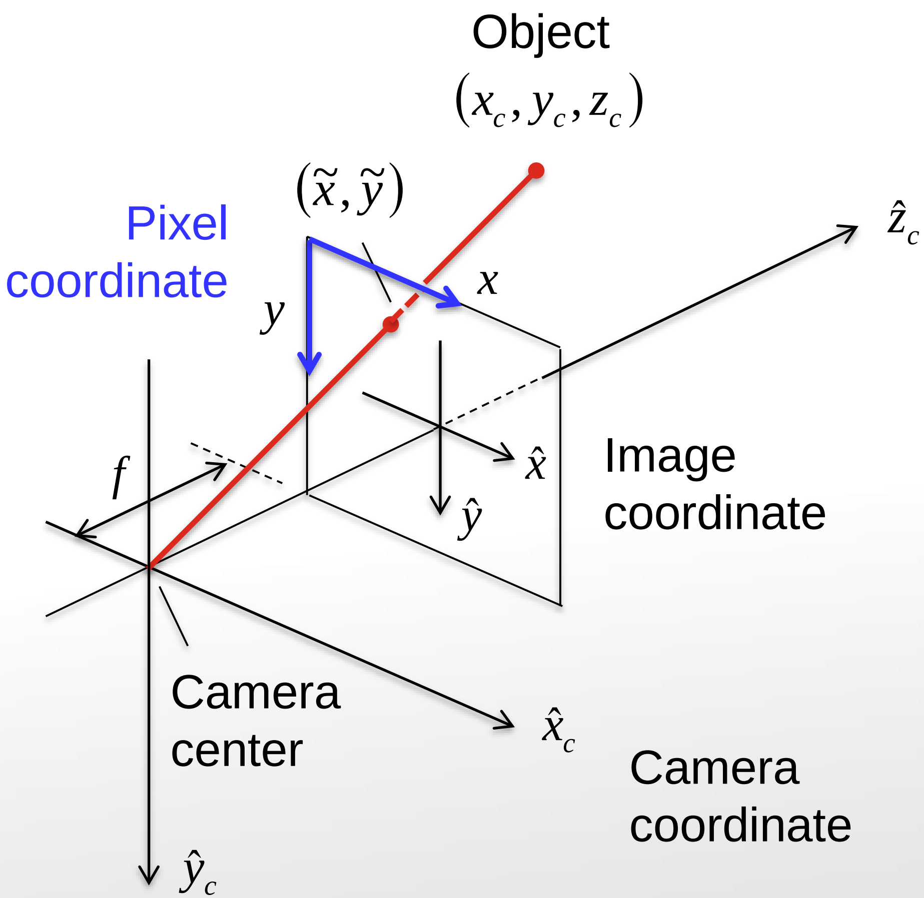
- Similarly, we will have:

$$\tilde{x} = f\,\frac{x_c}{z_c}$$



Pixel Value

- The point location in the image coordinate will then need to be given in terms of the pixels.



- With reference to the pixel coordinate system, the point $(\tilde{x}, \tilde{y})$ has the value:

Location in image plane

Shift the center (0,0) of image to a corner

$$x = \frac{\tilde{x}}{dx} + x_0 \qquad y = \frac{\tilde{y}}{dy} + y_0$$

Location in terms of pixels

Scale by physical dimension of pixel

Pixel Value

- The point location in the image coordinate will then need to be given in terms of the pixels.
- With reference to the pixel coordinate system, the point $(\tilde{x}, \tilde{y})$ has the value:
- For example:
  - If the x-location of a point in image plane is $\tilde{x} = 3\mu m$,
  - And if the dimension of a pixel is $dx = 1.5\mu m$,
  - Then the pixel value (ignoring the translation) is 2.

Location in image plane

Shift the center (0,0) of image to a corner

$$x = \frac{\tilde{x}}{dx} + x_0 \qquad y = \frac{\tilde{y}}{dy} + y_0$$

Location in terms of pixels

Scale by physical dimension of pixel

Camera Calibration Matrix

- Combining all equations we have so far, i.e.
  - From camera coordinate system to image coordinate system:

$$\tilde{x} = f\frac{x_c}{z_c} \qquad \tilde{y} = f\frac{y_c}{z_c}$$

  - From image coordinate system to pixel coordinate system:

$$x = \frac{\tilde{x}}{dx} + x_0 \qquad y = \frac{\tilde{y}}{dy} + y_0$$

  - We can write:

$$\boxed{x = \frac{f}{dx}\frac{x_c}{z_c} + x_0 \qquad y = \frac{f}{dy}\frac{y_c}{z_c} + y_0}$$

Camera Calibration Matrix

- The final equations,

$$x = \frac{f}{dx}\frac{x_c}{z_c} + x_0 \qquad y = \frac{f}{dy}\frac{y_c}{z_c} + y_0$$

- Can be expressed in a matrix form (homogeneous form, i.e. adds a component to a 2D vector to make it a 3D vector) :

$$\boxed{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}}$$

or

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Note: This is proportional sign, NOT equal sign.

- Where:

$$\boxed{K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}} \qquad \alpha_x = \frac{f}{dx} \qquad \alpha_y = \frac{f}{dy}$$

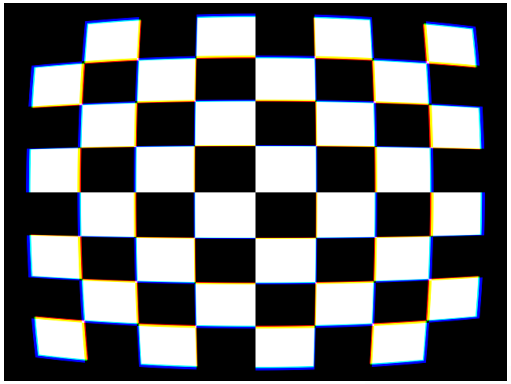
is called the Camera Calibration Matrix.

Camera Calibration Matrix

- How does the equation work? $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$
- The proportional sign means "Equal up to Scale".
- The equation gives: $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} \alpha_x x_c + x_0 z_c \\ \alpha_y y_c + y_0 z_c \\ z_c \end{bmatrix}$$
- It is clear that the row should be 1 = 1. Therefore, we divide the right hand by z_c and get: $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x \dfrac{x_c}{z_c} + x_0 \\ \alpha_y \dfrac{y_c}{z_c} + y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{f}{d_x} \dfrac{x_c}{z_c} + x_0 \\ \dfrac{f}{d_y} \dfrac{y_c}{z_c} + y_0 \\ 1 \end{bmatrix}$$

Distortion

- The pinhole camera model is not necessarily valid for all camera.
- Most images suffer from lens distortion:
- Barrel Distortion:



  - A type of “radial distortion”.
  - The amount of “bulging out” depends on how far a point is from the center.

Distortion

- The relationship between undistorted and distorted point (in image coordinate system) is:

$$\begin{bmatrix} \tilde{x}_{dist} \\ \tilde{y}_{dist} \end{bmatrix} = \left(1 + K_1 r^2 + K_2 r^4\right) \begin{bmatrix} \tilde{x}_{un} \\ \tilde{y}_{un} \end{bmatrix}$$

$$= \left(1 + K_1\left(\tilde{x}_{un}^2 + \tilde{y}_{un}^2\right) + K_2\left(\tilde{x}_{un}^2 + \tilde{y}_{un}^2\right)^2\right) \begin{bmatrix} \tilde{x}_{un} \\ \tilde{y}_{un} \end{bmatrix}$$

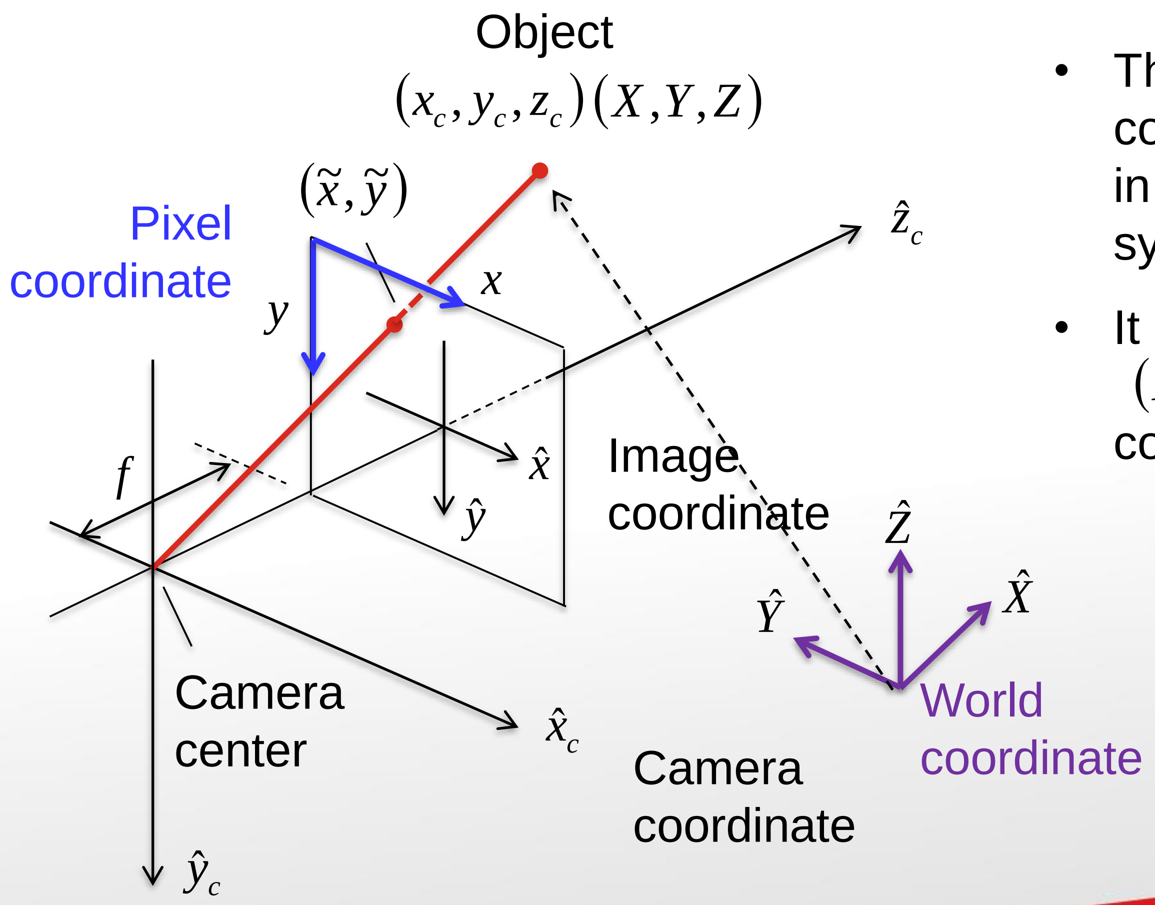
  - We can stop at r^2 if the distortion not serious, or we can go up to higher degree if distortion is serious.
- We can estimate K1 and K2 using checkerboard, for e.g. using Least Squares Algorithm.
- Then, to undo the distortion, we can use the inverse relationship between distorted and undistorted point.
- For the remainder of this lecture, we will not consider this distortion effect.

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Extrinsic Parameters

- The extrinsic parameters give the relationship between the World Coordinate System and the Camera Coordinate System.



- The object point has coordinates (x_c, y_c, z_c) in Camera coordinate system.
- It also has coordinates (X, Y, Z) in World coordinate system.

Extrinsic Parameters

- We can convert the point from World Coordinate System to Camera Coordinate System by a rotation and translation:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

  - R = Orientation of World Coordinate System wrt. Camera Coordinate System.
  - T = Position of the origin of World Coordinate System expressed in Camera Coordinate System.
- The values of the rotation matrix and translation vector are what we call the Extrinsic Parameters of a camera.

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Camera Matrix

- Summary:
- The extrinsic parameters give relationship between World Coordinate System (X, Y, Z) and Camera Coordinate System (x_c, y_c, z_c):

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

- The intrinsic parameters give relationship between Camera Coordinate System (x_c, y_c, z_c) and Pixel Coordinate System (x, y, z):

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Camera Matrix

- We can combine the both to get:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = K\left(R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) = K\begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

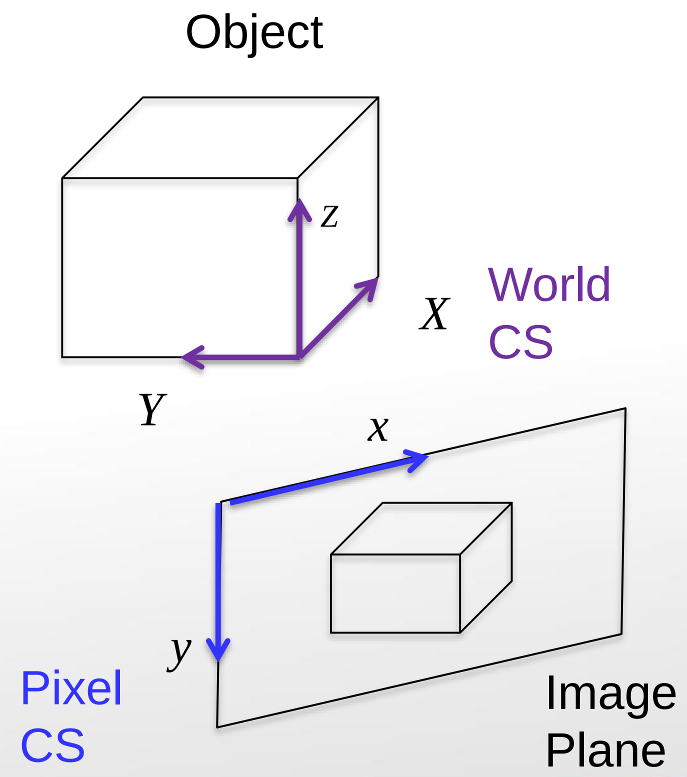
- i.e.:

$$\boxed{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}$$

- Where $P = K\begin{bmatrix} R & T \end{bmatrix}$ is called the Camera Matrix. (Not to be confused with Camera Calibration Matrix K).

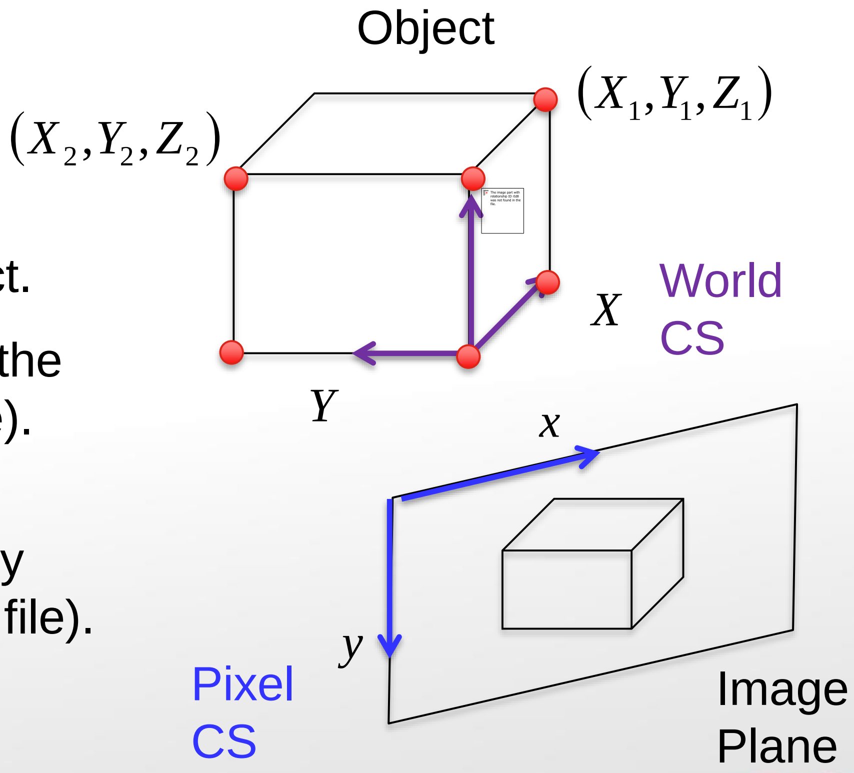
Camera Calibration

- But how do we get P?
- This is the goal of camera calibration (also called resectioning) → To estimate P from known x and X.
- Imagine the following scenario:
- Now, do the following:
  - Attach the World CS onto the object.



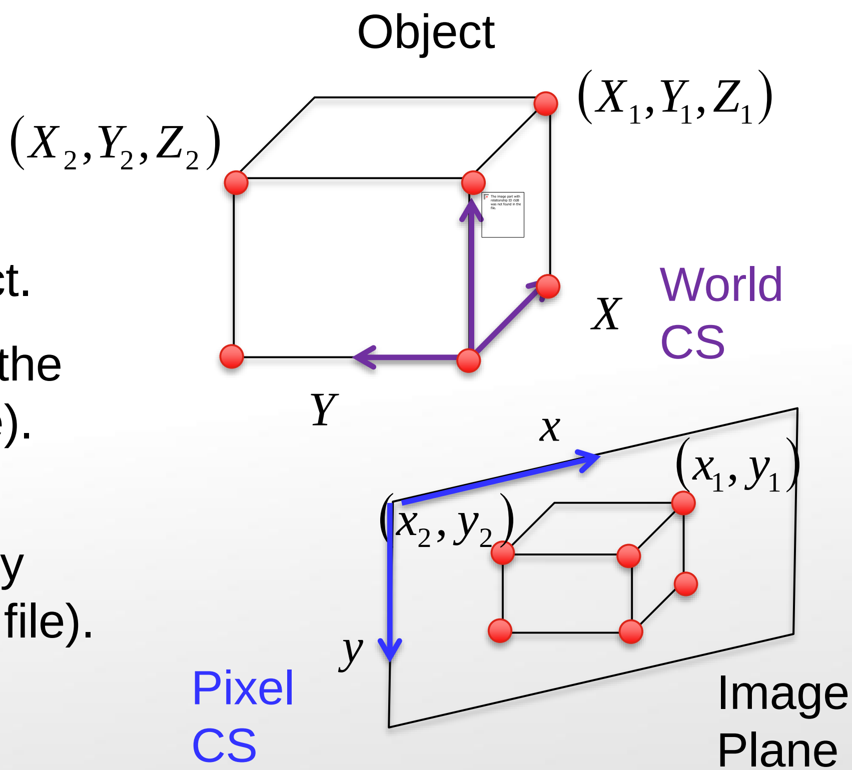
Camera Calibration

- But how do we get P?
- This is the goal of camera calibration (also called resectioning) → To estimate P from known x and X.
- Imagine the following scenario:
- Now, do the following:
  - Attach the World CS onto the object.
  - Then choose at least six points on the object (Not all on the same Z-plane).
  - The location of these points with reference to World CS can be easily determined (measurement or CAD file).



Camera Calibration

- But how do we get P?
- This is the goal of camera calibration (also called resectioning) → To estimate P from known x and X.
- Imagine the following scenario:
- Now, do the following:
  - Attach the World CS onto the object.
  - Then choose at least six points on the object (Not all on the same Z-plane).
  - The location of these points with reference to World CS can be easily determined (measurement or CAD file).
  - Determine the pixel value of the corresponding points on the image plane.



Camera Calibration

- For each point, we have:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

- Remember, the relationship is only "proportional", not equal. How can we solve it?
- The proportionality means that $\begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$ is a scalar multiple of $P\begin{bmatrix} X_i & Y_i & Z_i & 1 \end{bmatrix}^T$
- Therefore, their cross product is zero.

Camera Calibration

- In other words:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \times \begin{bmatrix} p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14} \\ p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24} \\ p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{vmatrix} i & j & k \\ x_i & y_i & 1 \\ \begin{pmatrix} p_{11}X_i + p_{12}Y_i \\ + p_{13}Z_i + p_{14} \end{pmatrix} & \begin{pmatrix} p_{21}X_i + p_{22}Y_i \\ + p_{23}Z_i + p_{24} \end{pmatrix} & \begin{pmatrix} p_{31}X_i + p_{32}Y_i \\ + p_{33}Z_i + p_{34} \end{pmatrix} \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$y_i\left(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}\right) - \left(p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}\right) = 0$$
$$x_i\left(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}\right) - \left(p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}\right) = 0$$

- (Only two independent equations).

Camera Calibration

- From the last equation, we can write:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -y_i X_i & -y_i Y_i & -y_i Z_i & -y_i \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -x_i X_i & -x_i Y_i & -x_i Z_i & -x_i \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- There are 12 parameters but only 2 equations, for one point.
- Not solvable.

RMIT UNIVERSITY

Camera Calibration

- If we now use 6 or more points, we can obtain:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ & & & & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Camera Calibration

- The equation is of the form:

$$Ap = 0$$

- Because it is a homogeneous equation (right hand side equals zero), the solution is not unique.
- There are a few ways to solve for p, for e.g.
  - If exactly six points measured: Find null-space of A. Then pick the one with $\|p\| = 1$.
    - If more than six points are measured, it is not possible to get null space of A due to measurement noise.
  - Minimize $\|Ap\|$ subject to $\|p\| = 1$.
  - Using Singular Value Decomposition of A $A = U\Sigma V^T$.
    - Then set p = last column of V.
  - One more method on the next slide…

Camera Calibration

- We know that $$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$
  - i.e. the equation is correct up to a scale.
- We can arbitrarily fix one element, e.g. $P_{34} = 1$, and then solve for the remaining ones.
- (Continue next slide)

Camera Calibration

- This means:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ & & & & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

- (Continue next slide)

Camera Calibration

- Or:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 \\ X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 \\ & & & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} = \begin{bmatrix} y_1 \\ x_1 \\ y_2 \\ x_2 \\ \vdots \\ y_n \\ x_n \end{bmatrix}$$

$$\tilde{A}\tilde{P} = \theta$$

- With this, the vector p can be calculated using least squares method, i.e.

$$\tilde{P} = \left(\tilde{A}^T\tilde{A}\right)^{-1}\tilde{A}^T\theta$$

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Recovering the Parameters

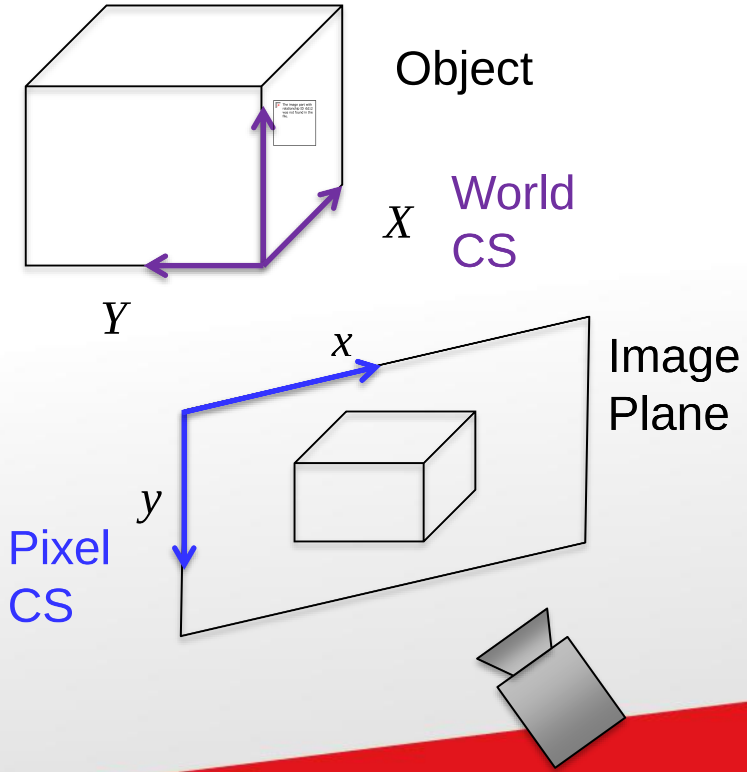
- In the last section, we have obtained the matrix P.
- We now need to recover all the individual parameters (intrinsic and extrinsic) from the matrix P.
- We split the (3 x 4) matrix P into: $P = [P_1 \quad P_2]$
- Also, recall that: $P = K[R \quad T]$
  - Therefore:

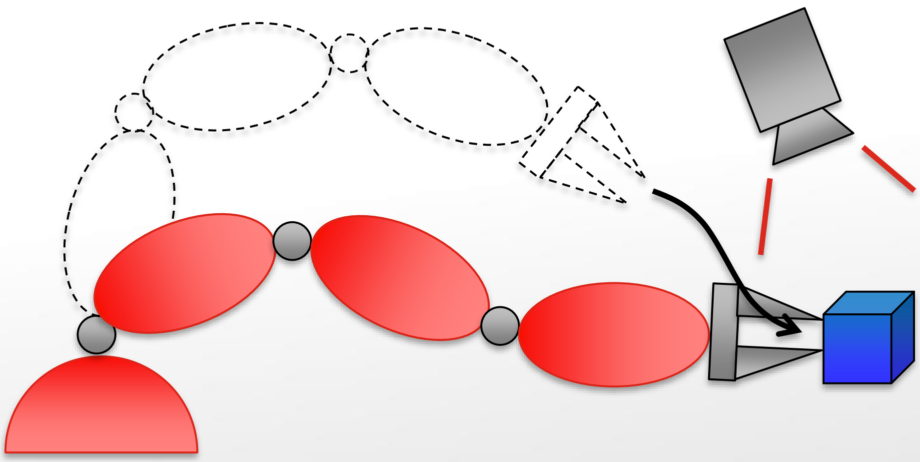
$$\underbrace{P_1}_{3\times 3} = K \cdot R \qquad \underbrace{P_2}_{3\times 1} = K \cdot T$$

- For P_1, K is an upper triangular matrix, and R is orthogonal (rotation matrix).
  - There is a standard algorithm, called RQ decomposition to solve it.
  - Thus, assume we have K and R now.
- With known K, we can then calculate T from: $T = K^{-1} \cdot P_2$

3D Pose Estimation

- Up to this stage, we have already calculated the R and T matrices.
- Thus, we have already estimated the 3D pose of the camera w.r.t. the world frame (also object, since we attach the world frame onto the object).
- Finally, we can command the robot manipulator to move towards the object and grasp it.





Some Details

- Note, in MATLAB we only have QR decomposition. (Q orthogonal and R upper triangular)
- However, what we need is RQ decomposition.
- Trick: use inverse, i.e.:
  - We know $\underbrace{P_1}_{3\times 3} = \underbrace{K}_{\substack{upper\\triangle}} \cdot \underbrace{R}_{orthogonal}$
  - Then $\underbrace{P_1^{-1}}_{3\times 3} = \left(\underbrace{K}_{\substack{upper\\triangle}} \cdot \underbrace{R}_{orthogonal} \right)^{-1} = \underbrace{R^{-1}}_{orthogonal} \cdot \underbrace{K^{-1}}_{\substack{upper\\triangle}}$
  - This is suitable for QR decomposition. → Matlab $[Rinv, Kinv] = qr(P1inv)$
  - After decomposition, we then invert $Rinv$ and $Kinv$ to get R and K

Some Details

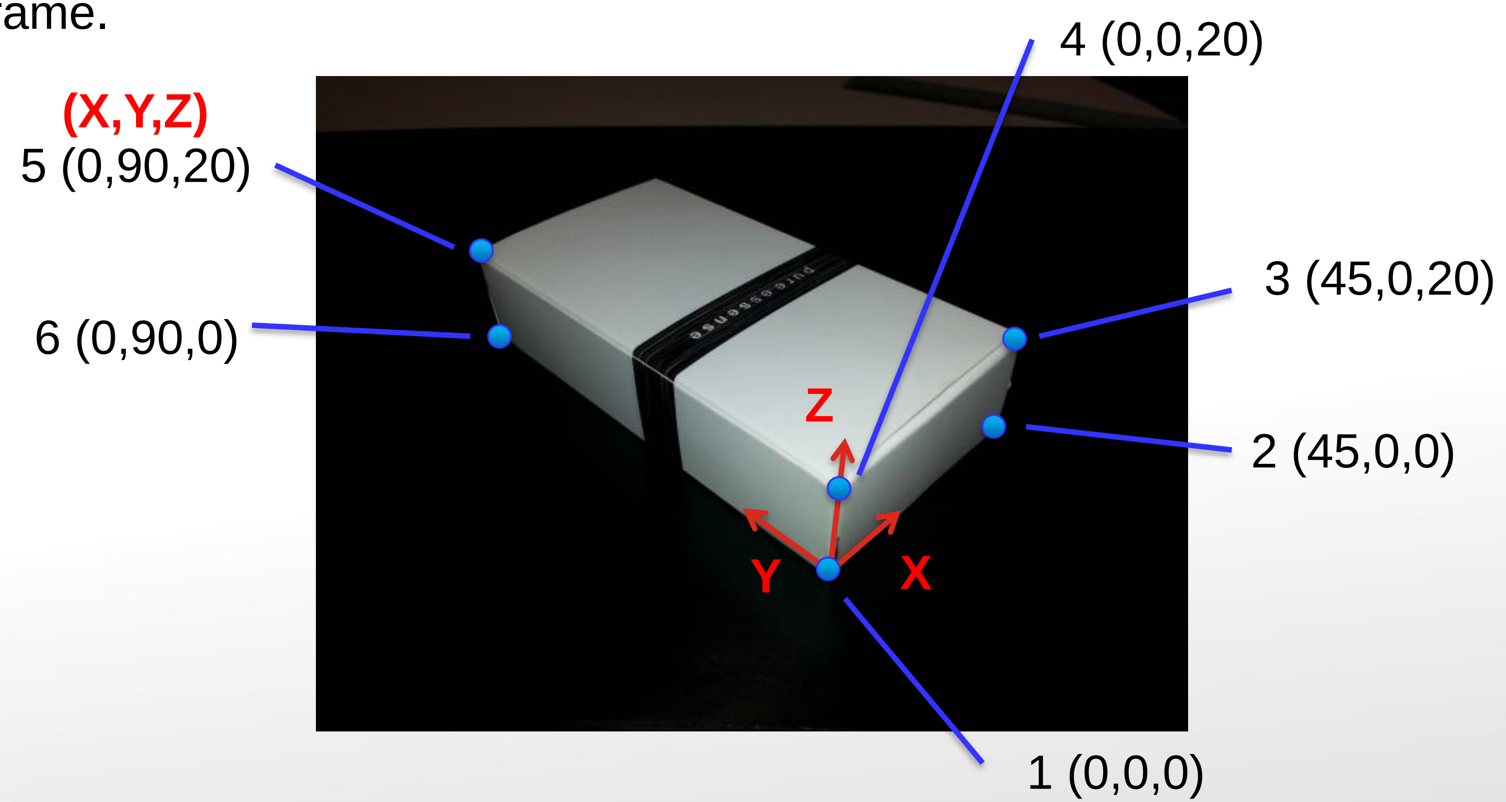
- Another issue with the RQ decomposition is that the answer is not unique!
  - Sometimes we might get negative diagonal elements of K, which is weird because if the camera looks in positive direction, f must be positive.

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \alpha_x = \frac{f}{dx} \qquad \alpha_y = \frac{f}{dy}$$

- Solution:
  - Notice that if any column of K is negated, and the corresponding row of R is also negated, then P_1 = KR is still the same.
  - Therefore, we can force the diagonal terms of K to be positive.

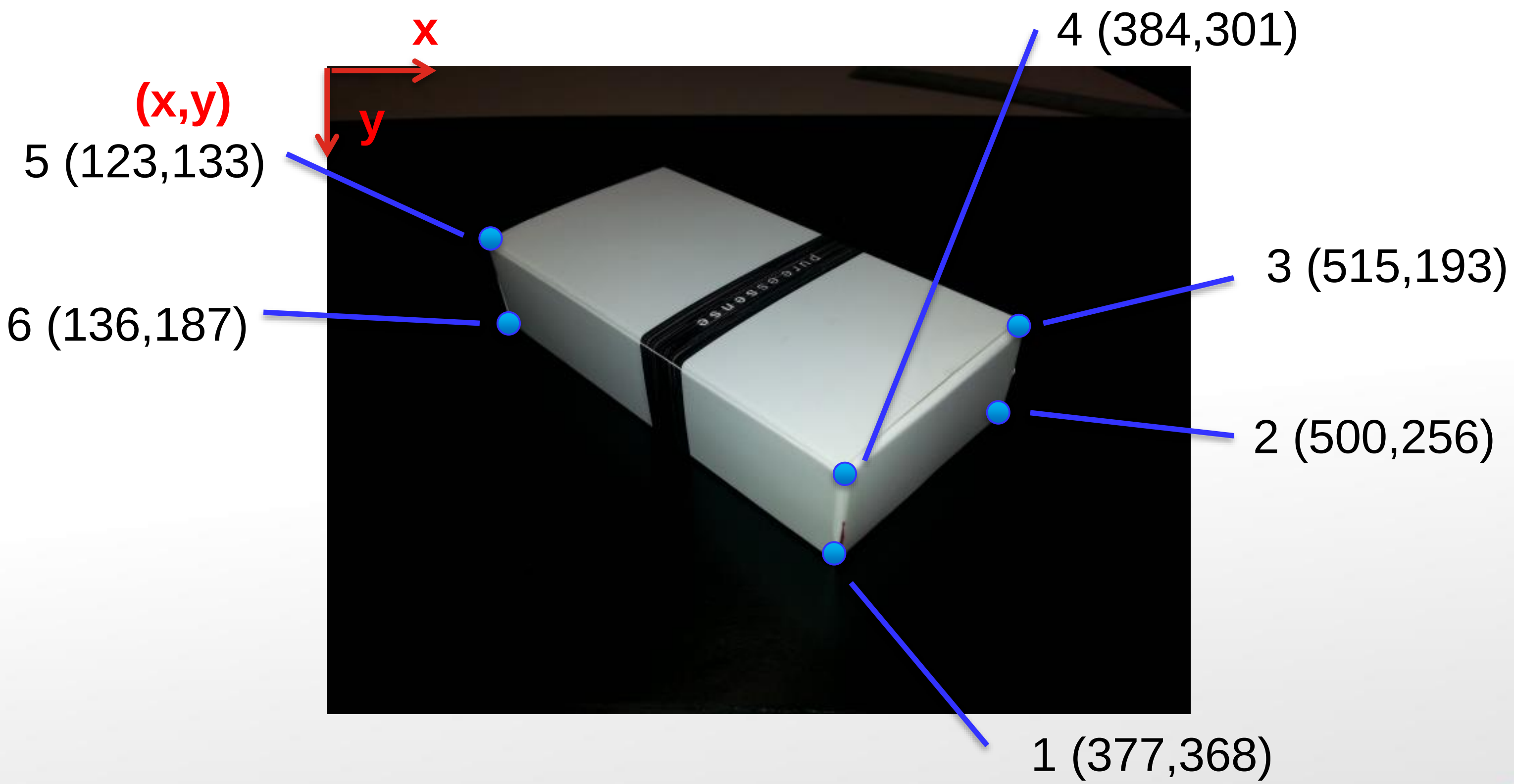
Complete Example

- Following is a box with known dimension.
- A frame is fixed at one of the vertices and the other points are given wrt. the frame.



Complete Example

- The pixel coordinates of the points are as follows:



Complete Example

- Thus in summary, we have:

$X_1 = 0$	$Y_1 = 0$	$Z_1 = 0$
$X_2 = 45$	$Y_2 = 0$	$Z_2 = 0$
$X_3 = 45$	$Y_3 = 0$	$Z_3 = 20$
$X_4 = 0$	$Y_4 = 0$	$Z_4 = 20$
$X_5 = 0$	$Y_5 = 90$	$Z_5 = 20$
$X_6 = 0$	$Y_6 = 90$	$Z_6 = 0$

$x_1 = 377$	$y_1 = 368$
$x_2 = 500$	$y_2 = 256$
$x_3 = 515$	$y_3 = 193$
$x_4 = 384$	$y_4 = 301$
$x_5 = 123$	$y_5 = 133$
$x_6 = 136$	$y_6 = 187$

Complete Example

- We can then set the matrix equation below using the numerical values from the previous page:

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 \\
X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 \\
0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 \\
X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 \\
 & & & & & & \vdots & & & & \\
0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n \\
X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n
\end{bmatrix}
\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}
=
\begin{bmatrix} y_1 \\ x_1 \\ y_2 \\ x_2 \\ \vdots \\ y_n \\ x_n \end{bmatrix}
$$

Complete Example

- The MATLAB Code is as follows:

```
LHS = [0 0 0 0 X1 Y1 Z1 1 -y1*X1 -y1*Y1 -y1*Z1;
    X1 Y1 Z1 1 0 0 0 0 -x1*X1 -x1*Y1 -x1*Z1;
    0 0 0 0 X2 Y2 Z2 1 -y2*X2 -y2*Y2 -y2*Z2;
    X2 Y2 Z2 1 0 0 0 0 -x2*X2 -x2*Y2 -x2*Z2;
    0 0 0 0 X3 Y3 Z3 1 -y3*X3 -y3*Y3 -y3*Z3;
    X3 Y3 Z3 1 0 0 0 0 -x3*X3 -x3*Y3 -x3*Z3;
    0 0 0 0 X4 Y4 Z4 1 -y4*X4 -y4*Y4 -y4*Z4;
    X4 Y4 Z4 1 0 0 0 0 -x4*X4 -x4*Y4 -x4*Z4;
    0 0 0 0 X5 Y5 Z5 1 -y5*X5 -y5*Y5 -y5*Z5;
    X5 Y5 Z5 1 0 0 0 0 -x5*X5 -x5*Y5 -x5*Z5;
    0 0 0 0 X6 Y6 Z6 1 -y6*X6 -y6*Y6 -y6*Z6;
    X6 Y6 Z6 1 0 0 0 0 -x6*X6 -x6*Y6 -x6*Z6];

RHS = [y1 x1 y2 x2 y3 x3 y4 x4 y5 x5 y6 x6]';

P = LHS\RHS;
```

Complete Example

- The MATLAB Code continued…

```
%%%%%%%%%%%%%%%%%%%%%%%
% Getting K, R from P %
%%%%%%%%%%%%%%%%%%%%%%%

P1 = [P(1) P(2) P(3);
    P(5) P(6) P(7);
    P(9) P(10) P(11)];

P1inv = inv(P1);
[Rinv, Kinv] = qr(P1inv);
K = inv(Kinv);
R = inv(Rinv);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% make diagonal of K positive %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SIGNS = diag(sign(diag(K)));

K = K * SIGNS
R = SIGNS * R % Orientation of world CS wrt. camera-centered CS
```

Complete Example

- The MATLAB Code continued…

```
%%%%%%%%%%%%%%%%%%%%
% Getting T from P %
%%%%%%%%%%%%%%%%%%%%

P2 = [P(4) P(8) 1]'; % Recall that P34 = P(12) = 1

T = inv(K)*P2 % Origin of the world CS expressed in camera-centered CS
```

Complete Example

- And the answer given by MATLAB is:

```
K =

    5.2722   -0.0534    2.6288
         0    4.8751    1.3524
         0         0    0.0095


R =

    0.7348   -0.6763   -0.0517
   -0.3881   -0.3567   -0.8498
    0.5563    0.6445   -0.5245


T =

   19.5326
   46.2685
  105.3472
```

Complete Example

- Let's interpret the results. We normalize K such that K(3,3) = 1:

```
>> K/K(3,3)

ans =

  555.4112   -5.6276  276.9332
         0  513.5750  142.4748
         0         0    1.0000
```

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\alpha_x = \frac{f}{dx}$$

$$\alpha_y = \frac{f}{dy}$$

- From camera data sheet, the sensor size is 4.54mm x 3.42mm.
- The image has 640 pixel x 480 pixel.
- Thus each pixel size is 0.07mm x 0.07mm. → dx = 0.07, dy = 0.07
- Focal length of camera is 3.7mm. → f = 3.7
- Therefore $\alpha_x = f/dx = 530$ $\alpha_y = f/dy = 530$
- Answer (555 and 513) quite close to actual values (530 and 530).
- Also, x0 = 277 pixel and y0 = 142 pixel from the pixel CS origin (somewhat off-centered).

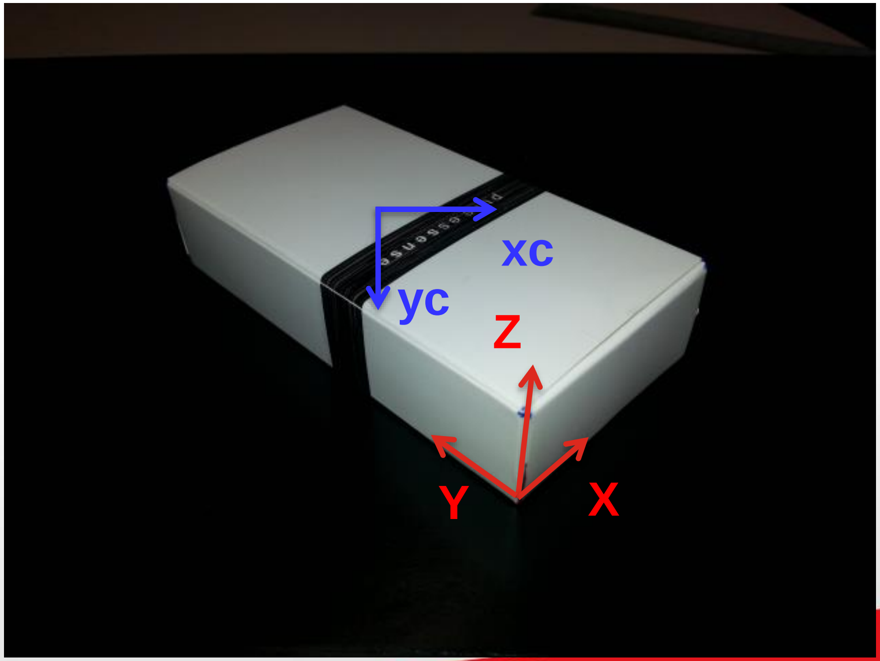
RMIT UNIVERSITY

Complete Example

- The translation vector was:

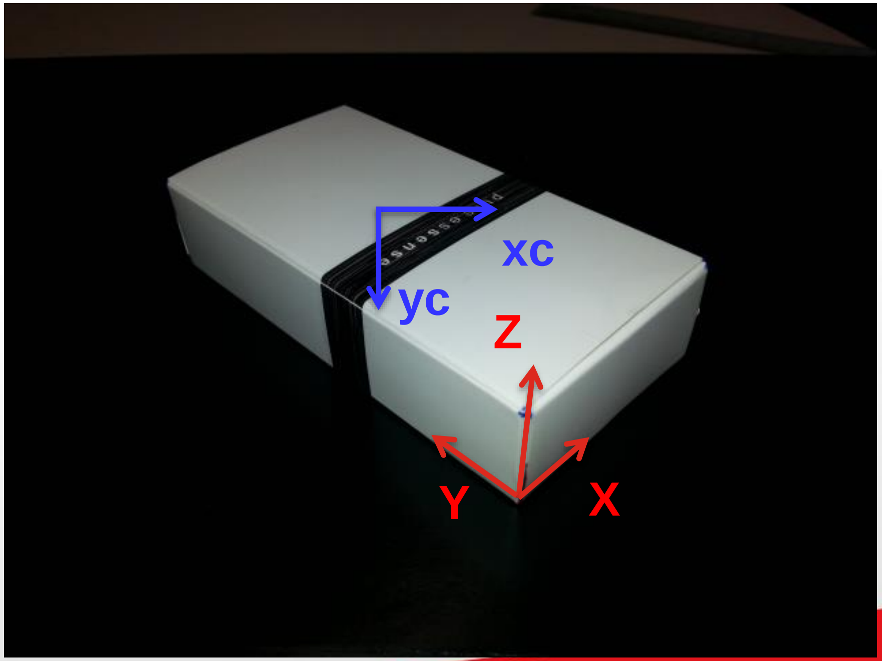
```
T =

   19.5326
   46.2685
  105.3472
```

- The answer of T looks correct from the figure below. (Remember that camera CS is somewhat off-centered).



Complete Example

- The rotation matrix interpreted as Z-Y-X-Euler angles are:
  - Z: -27.8 degrees
  - Y: -33.8 degrees
  - X: 129.1 degrees
  - Which seems correct.

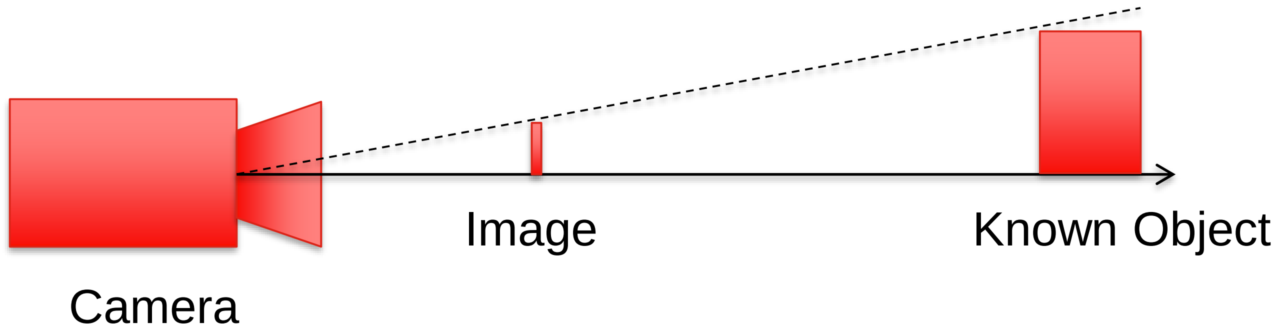


Content

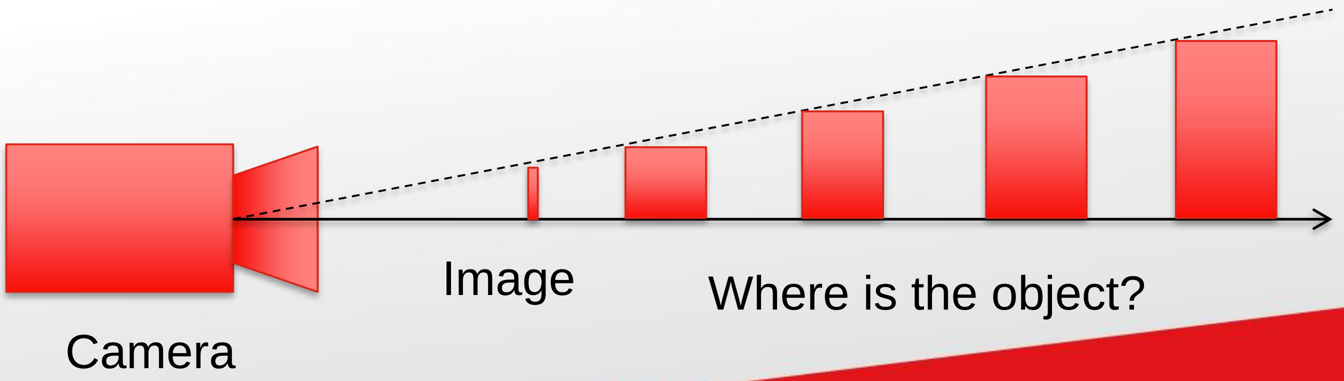
- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Introduction to Depth Perception

- If we have a calibrated camera AND a known object / model, then we know the depth (i.e. z-distance) of the object by doing pose estimation.

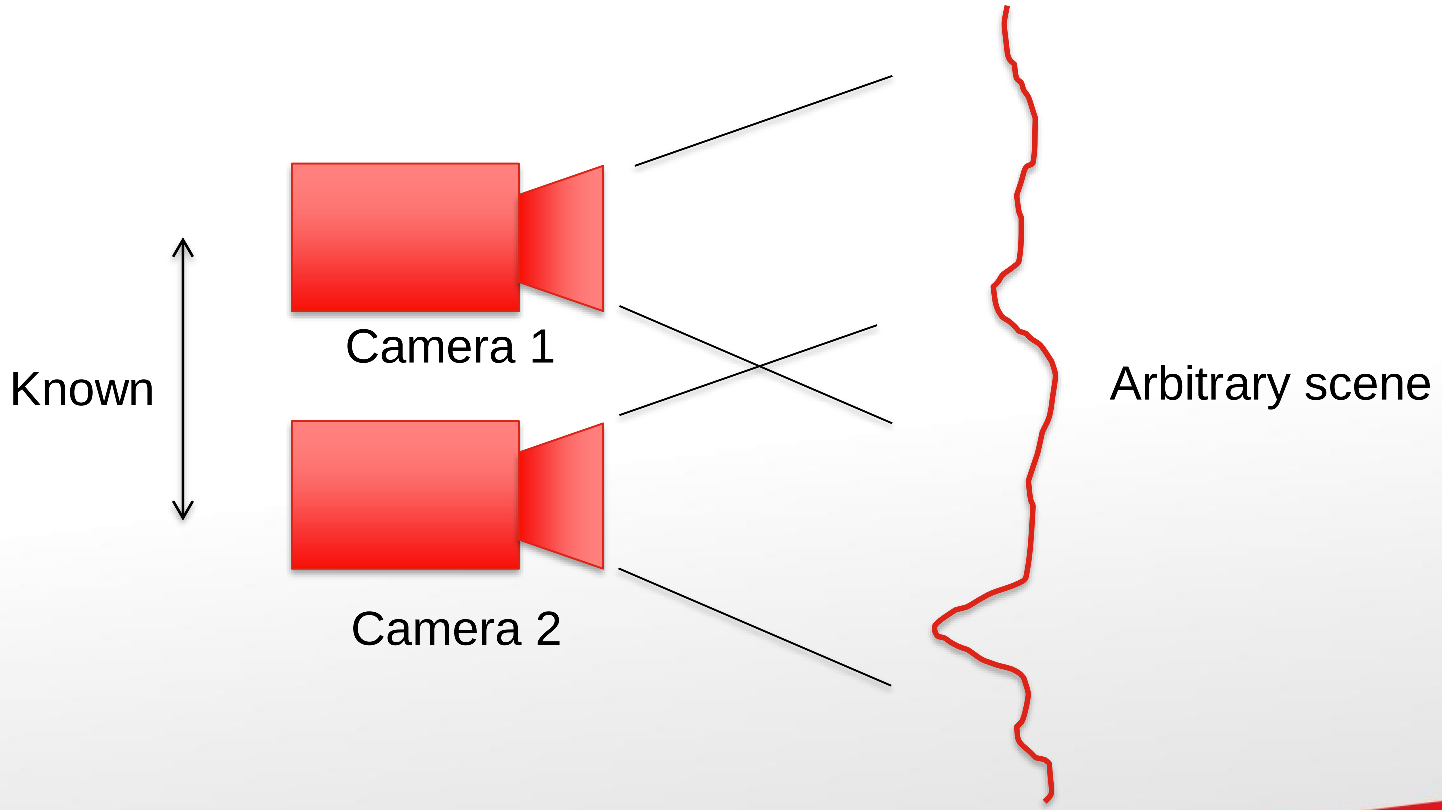


- However, if we do not know the object / model, then the depth is unknown!



Introduction to Depth Perception

- To be able to find out the distance for unknown / arbitrary objects, we need two calibrated cameras, with known relative pose.



Introduction to Depth Perception

- Stereo: Getting 3D information from 2 or more images.
- This method is used by human and animals to estimate distance:

https://www.zeiss.com

http://thestoneset.com/tigers-eye/

- And now, it's also used by computers / robots.

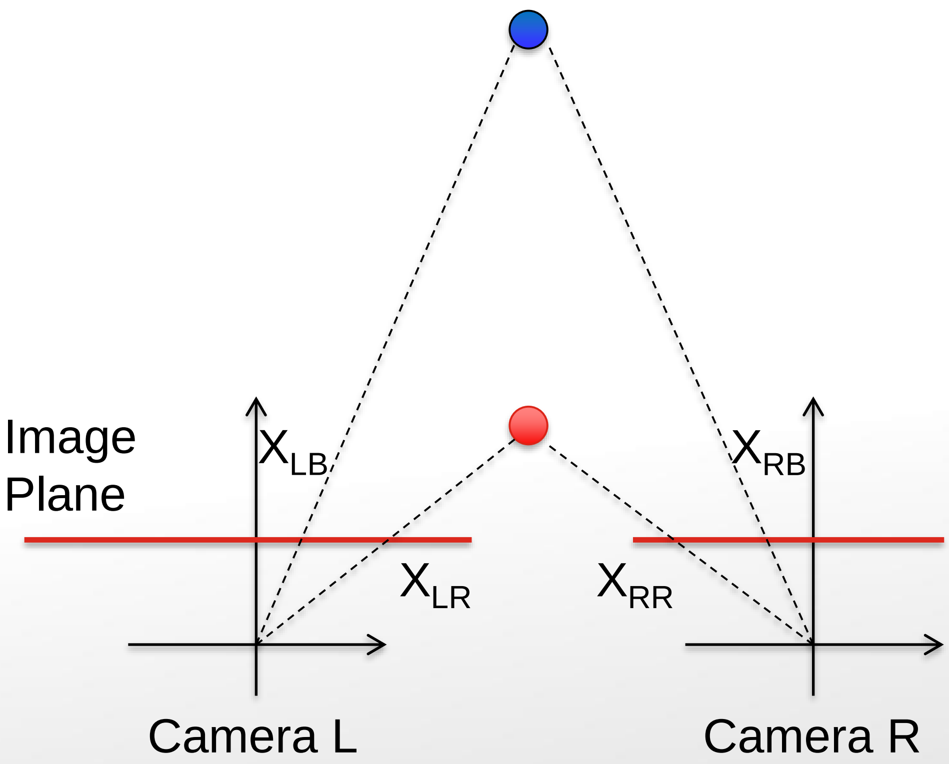
http://pfrommer.us/stereo-vision

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Stereo Disparity

- But how does having two eyes or two cameras solve the depth issue?
- Let's look at the following situation:



- The rays emanating from the cameras will pass through the points X_{LB}, X_{LR}, X_{RR} and X_{RB} on the image planes, before hitting the red and blue points.
- Assume the following numbers:
  - $X_{LB} = 2$
  - $X_{LR} = 5$
  - $X_{RR} = -5$
  - $X_{RB} = -2$

Stereo Disparity

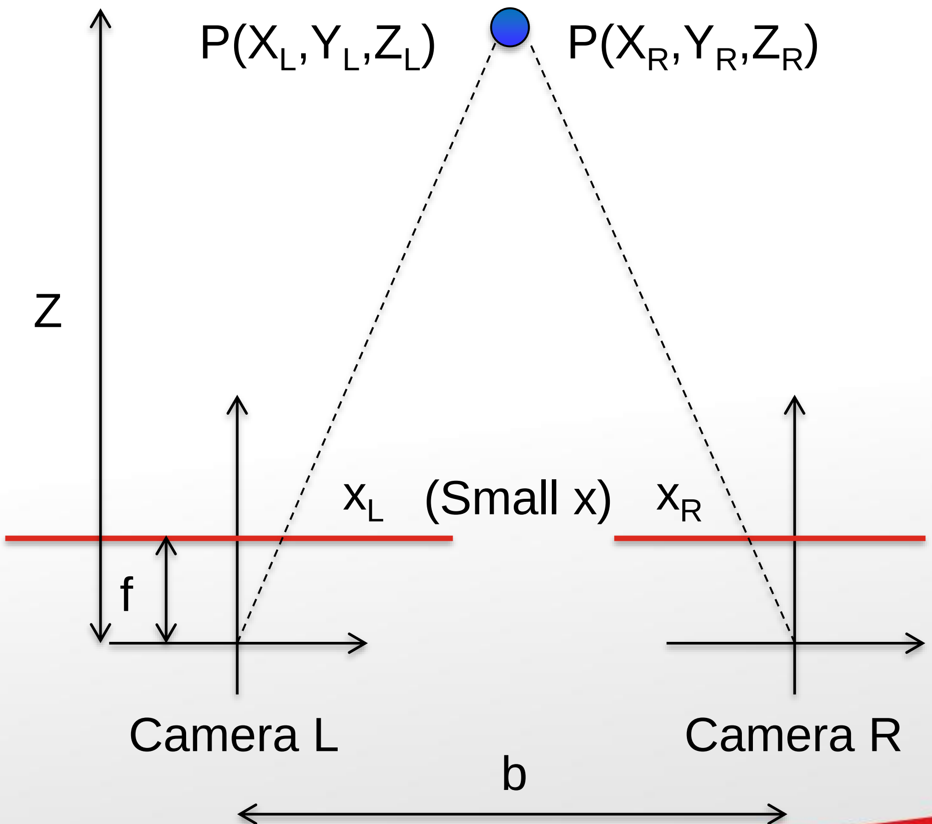
- We now compute the "disparity", i.e. the coordinate difference of a particular point in the left and right cameras.
  - For red point: $X_{RR} – X_{LR} = (-5) – (5) = -10$ → absolute disparity 10
  - For blue point: $X_{RB} – X_{LB} = (-2) – (2)= -4$ → absolute disparity 4
- As can be seen, a nearby point (red) gives a large disparity, and a faraway point (blue) gives as smaller disparity.

→ Disparity provides information about depth!

- Experiment: Close your right eye and look at two objects at different distance using your left eye. Switch you eye (left & right) continuously and you will notice that the nearer objects moves further between your eyes.

Stereo Disparity

- What is the equation between disparity and depth?
- Assume both cameras are coplanar, but right camera is located at a known distance "b" (called "baseline") from the left camera in the x-direction.



$$\frac{x_L}{f} = \frac{X_L}{Z_L} \Rightarrow x_L = f\frac{X_L}{Z_L}$$

$$\frac{x_R}{f} = \frac{X_R}{Z_R} \Rightarrow x_R = f\frac{X_R}{Z_R}$$

However, $Z_L = Z_R = Z$

And $X_R = X_L - b$

Therefore: $x_R = f\dfrac{(X_L - b)}{Z}$

Stereo Disparity

- The disparity "d" is defined as:

$$d = x_L - x_R = f\frac{X_L}{Z} - f\frac{(X_L - b)}{Z} = f\frac{b}{Z}$$

- Thus, the relationship between disparity (d) and depth (Z) is:

$$Z = f\frac{b}{d}$$

  - Inverse relationship: Smaller Z gives larger d, and vice versa.

- E.g. if d = 10 pixels, f = 400 pixels, b = 20cm

$$Z = f\frac{b}{d} = 400\text{pixels} \cdot \frac{20\text{cm}}{10\text{pixels}} = 800\text{cm}$$

Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

Correspondence Problem

- In Summary:
- If you know:
  - The intrinsic parameters of the cameras (in this case f)
  - The relative pose between the cameras (in this case b)
- If you measure
  - An image point in the left camera
  - A corresponding point in the right camera
- You can intersect the rays (triangulate) to find the absolute point position.

This is called the "Correspondence Problem", and is in fact the most difficult problem in stereo vision!

- How to write an algorithm to find the exact matching points?

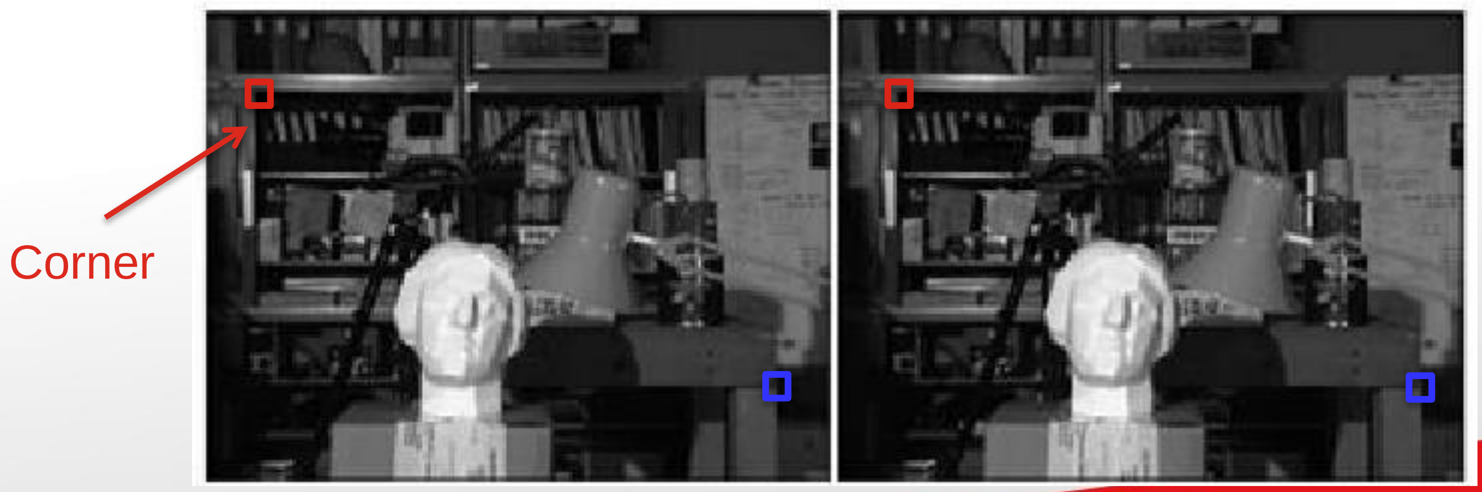
Correspondence Problem

- For example, how to write an algorithm that recognises that the region marked by the blue boxes / red boxes on both pictures as being the "same" regions?

https://www.researchgate.net/publication/229592067_Stereo_Matching_From_the_Basis_to_Neuromorphic_Engineering/figures?lo=1

Feature-Based Matching

- Two major approaches:
  - Feature-based:
    - Pick a feature type (e.g. edges / corners) using detection methods.
    - Define a matching criteria (e.g. orientation and contrast sign)
    - Then look for matches within disparity range.
    - Other points in between features can be linearly interpolated.



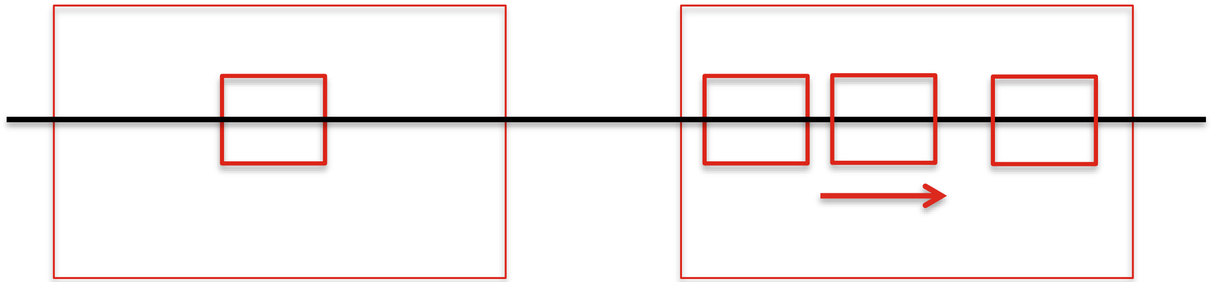
Region-Based Matching

- Region-based:
  - Forget about features.
  - Pick a region in the image, and find the matching region in the 2^{nd} image by
    - – minimizing some measure, e.g. sum of squared difference (SSD), sum of absolute difference (SAD) etc; or
    - – maximizing some measure, e.g. (normalized) cross correlation

200	200	200	200	200
200	200	200	200	200
200	200	0	0	0
200	200	0	0	0
200	200	0	0	0

200	200	200	200	200
200	200	200	200	200
200	200	0	0	0
200	200	0	0	0
200	200	0	0	0

Region-Based Matching



Epipolar line for coplanar cameras with no y-shift

- Sum of squared difference (SSD):

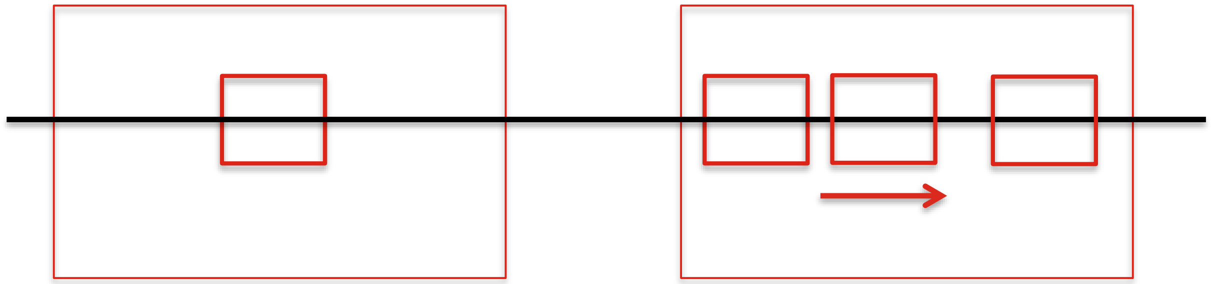
$$SSD = \sum_{(i,j)\in \text{window}} \left(I_L(i,j) - I_R(i,j)\right)^2$$

Surrounding Pixels   Element-wise

- Sum of absolute difference (SAD):

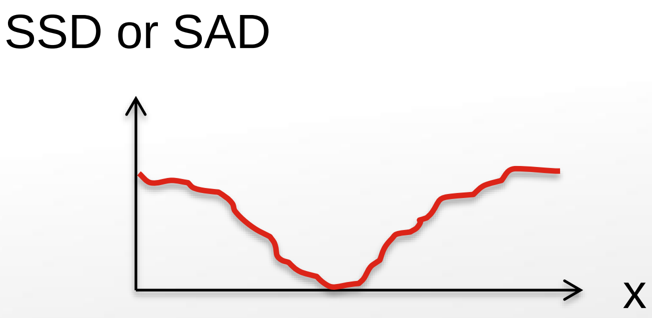
$$SAD = \sum_{(i,j)\in \text{window}} \left|I_L(i,j) - I_R(i,j)\right|$$

Region-Based Matching



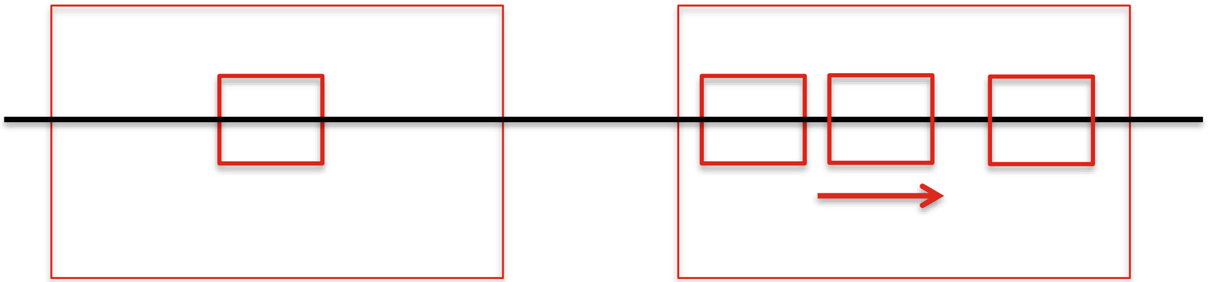
Epipolar line for coplanar cameras with no y-shift

- As we move along the epipolar line, the SSD or SAD would look something like this:



- The minimum error would thus correspond to the matching point.

Region-Based Matching



Epipolar line for coplanar cameras with no y-shift

- Cross Correlation (CC)

$$CC = \sum_{(i,j)\in \text{window}} \left(I_L(i,j) \times I_R(i,j) \right)$$

Surrounding Pixels

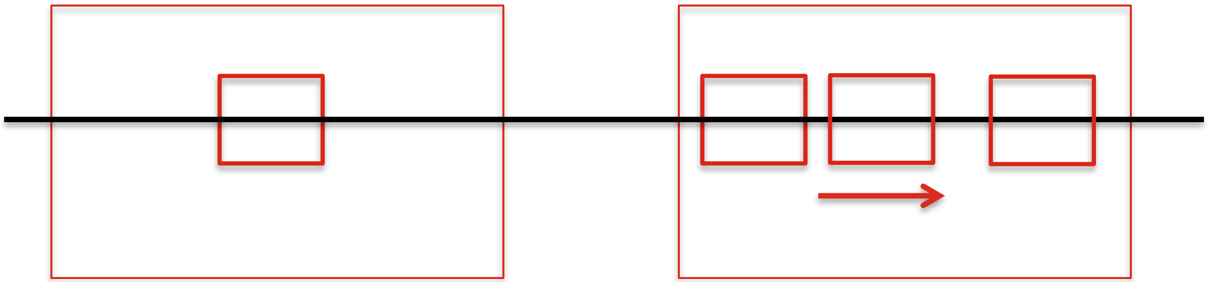
Element-wise

- Normalized Cross Correlation (NCC) to remove the effect of different illumination:

Mean

$$NCC = \sum_{(i,j)\in \text{window}} \left(\frac{I_L(i,j) - \bar{I}_L}{\sqrt{\sum \left(I_L(i,j) - \bar{I}_L \right)^2}} \times \frac{I_R(i,j) - \bar{I}_R}{\sqrt{\sum \left(I_R(i,j) - \bar{I}_R \right)^2}} \right)$$

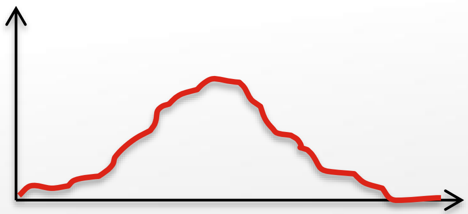
Region-Based Matching



Epipolar line for coplanar cameras with no y-shift

- As we move along the epipolar line, the CC or NCC would look something like this:

CC or NCC



x

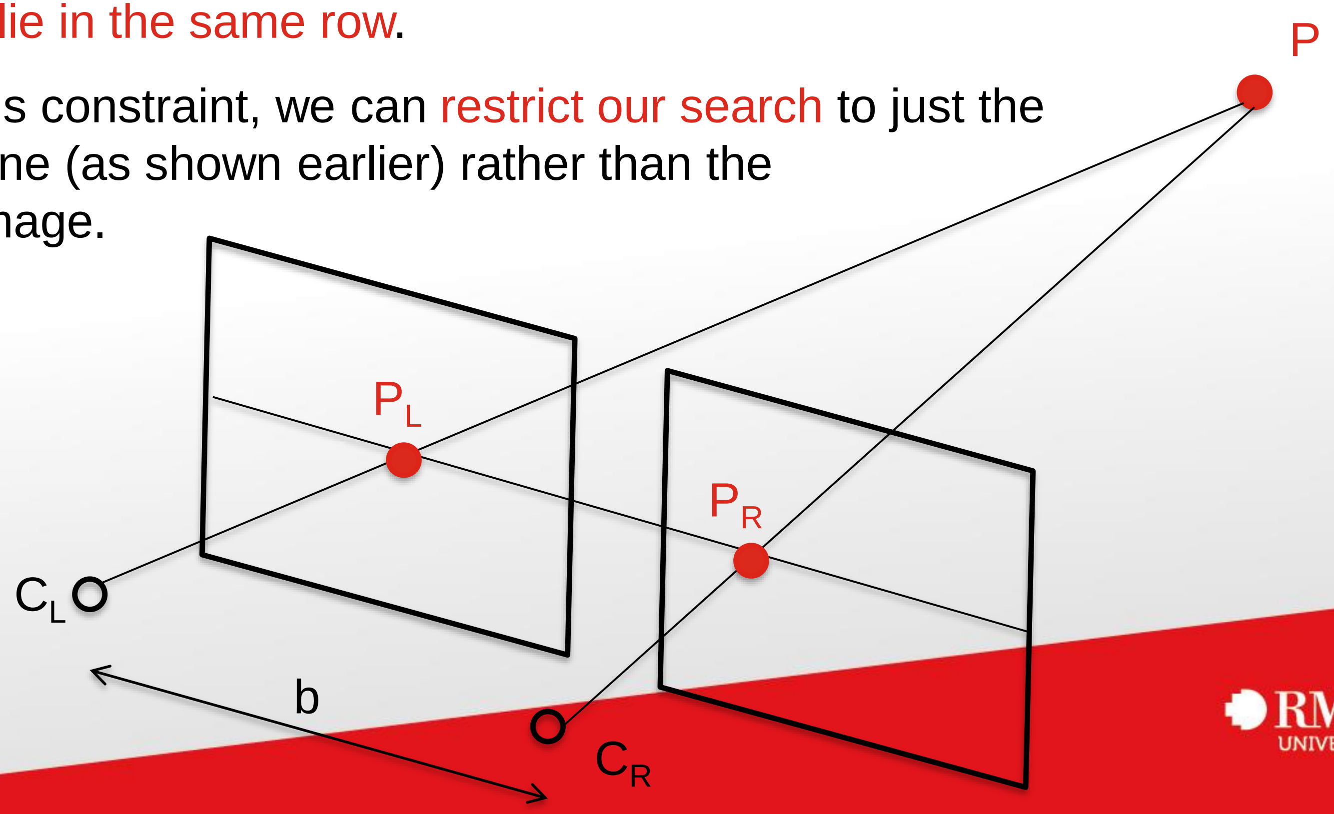
- The maximum correlation would thus correspond to the matching point.

Region-Based Matching

- Choice of Window Size:
- Smaller window:
  - Good precision, more details
  - Sensitive to noise
- Larger window:
  - Robust to noise
  - Reduced precision, less details

Epipolar Constraint

- We have used the term "Epipolar" just now. What does that mean?
- As shown in figure below, C_L, C_R and P form a plane.
- The image points would definitely lie on this plane.
  - This is called the Epipolar constraint.
- For coplanar cameras with no y-shift, the image points on both cameras would thus lie in the same row.
- By using this constraint, we can restrict our search to just the horizontal line (as shown earlier) rather than the complete image.

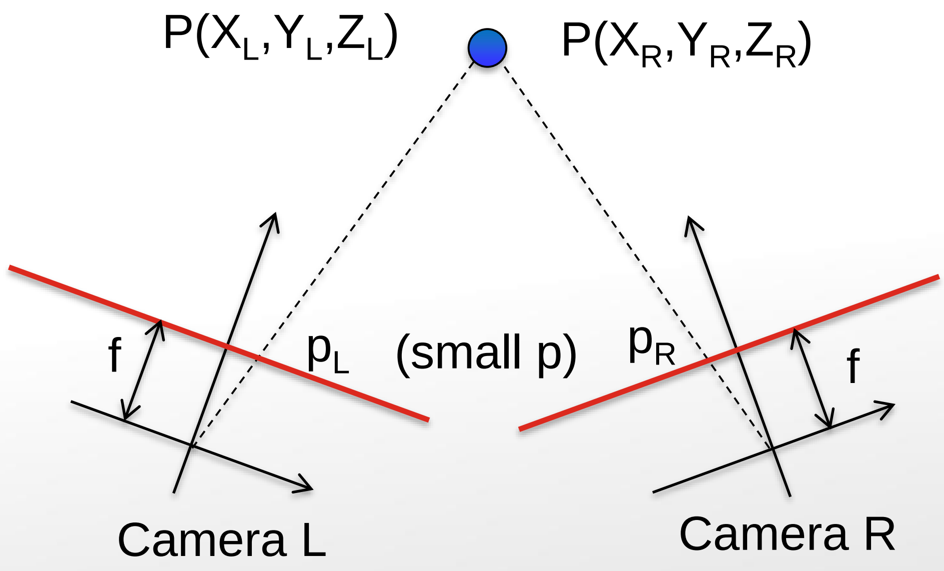


Content

- Segmenting Multiple Blobs
- 3D Pose Estimation for Known Objects
  - Introduction
  - Camera Intrinsic Parameters
  - Camera Extrinsic Parameters
  - Camera Calibration
  - 3D Pose Estimation
- Depth Perception for Arbitrary Objects
  - Introduction
  - Stereo Disparity
  - Correspondence Problem
  - Non-coplanar Cameras

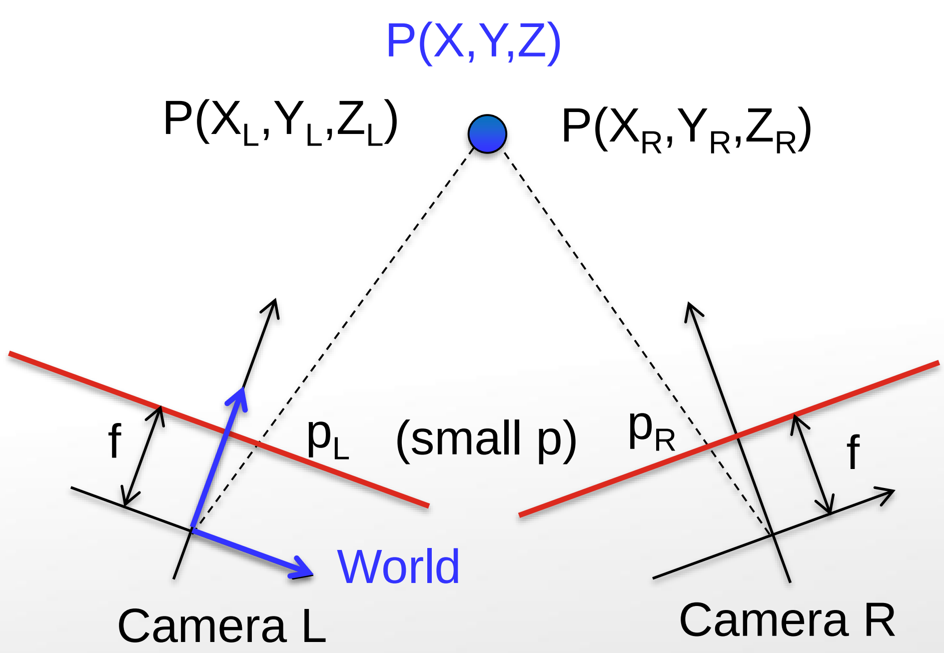
Non-Coplanar Cameras

- We have so far looked at the case where the cameras are co-planar.
- What if the cameras are not co-planar?
- Assumption: we know the relative pose of the cameras, and they are also calibrated.



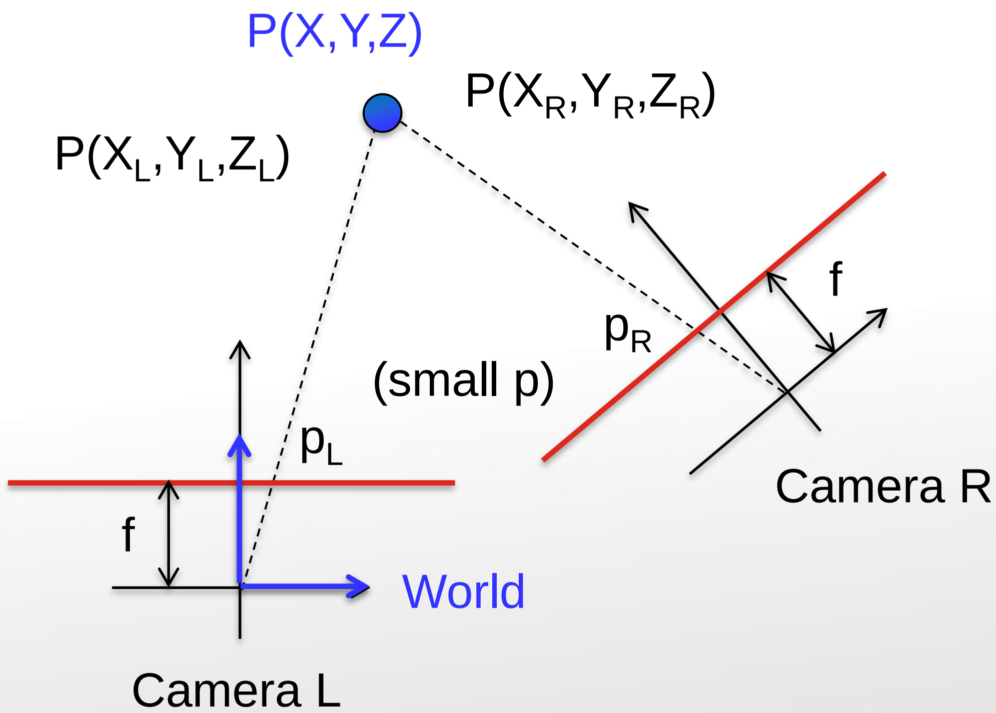
Non-Coplanar Cameras

- Let's fix the world coordinate frame at the left camera frame:



Non-Coplanar Cameras

- The view below is rotated for easier visualisation of theory later:



Reminder on Camera Matrix

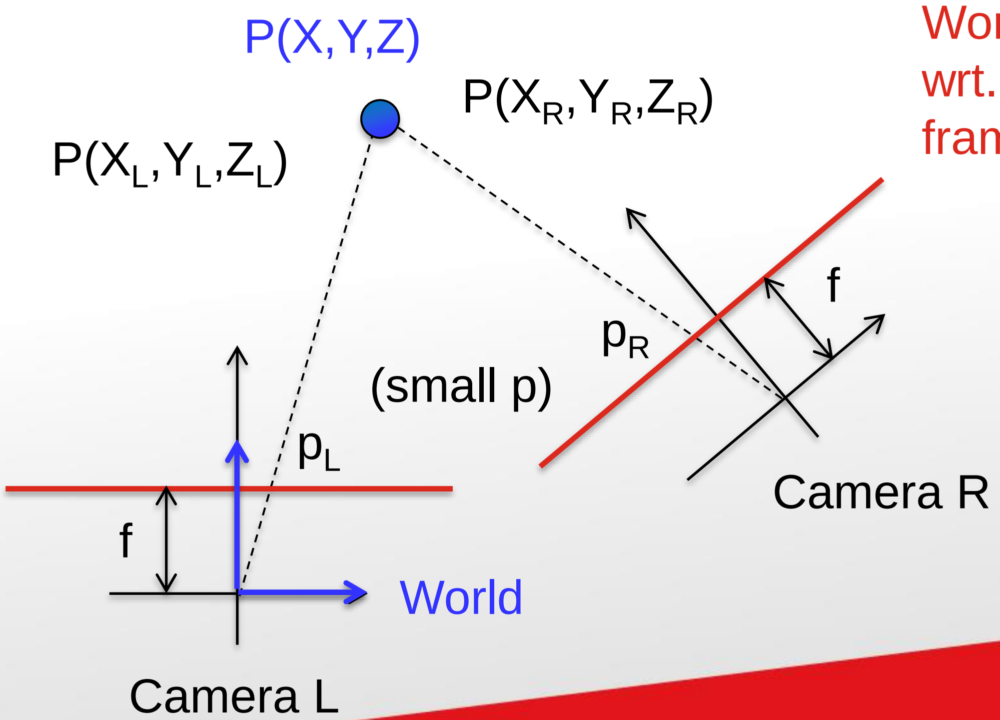
- A quick reminder of what we learnt earlier:

Pixel coordinate

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = K\left(R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) = K\begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

World coordinate

World frame wrt. Camera frame



$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\alpha_x = \frac{f}{dx} \qquad \alpha_y = \frac{f}{dy}$$

Left Camera Matrix

- For the left camera:

$$[R \quad T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Therefore:

$$\begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} \sim K[R \quad T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_{lx} & 0 & x_{l0} \\ 0 & \alpha_{ly} & y_{l0} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_{lx} & 0 & x_{l0} & 0 \\ 0 & \alpha_{ly} & y_{l0} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} p_{l11} & p_{l12} & p_{l13} & p_{l14} \\ p_{l21} & p_{l22} & p_{l23} & p_{l24} \\ p_{l31} & p_{l32} & p_{l33} & p_{l34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Right Camera Matrix

- For the right camera:

$$[R \quad T] = [{}^{R}_{L}R \quad {}^{R}P_{LORG}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

- Therefore:

$$\begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} \sim K[R \quad T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_{rx} & 0 & x_{r0} \\ 0 & \alpha_{ry} & y_{r0} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_{rx} r_{11} + x_{r0} r_{31} & \alpha_{rx} r_{12} + x_{r0} r_{32} & \alpha_{rx} r_{13} + x_{r0} r_{33} & \alpha_{rx} t_x + x_{r0} t_z \\ \alpha_{ry} r_{21} + y_{r0} r_{31} & \alpha_{ry} r_{22} + y_{r0} r_{32} & \alpha_{ry} r_{23} + y_{r0} r_{33} & \alpha_{ry} t_y + y_{r0} t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & p_{r14} \\ p_{r21} & p_{r22} & p_{r23} & p_{r24} \\ p_{r31} & p_{r32} & p_{r33} & p_{r34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Solving for X Y Z

- Remember that "proportional" means "equal up to scale", and thus the cross products of the left and right items are zero.
- Thus, for the left camera:

$$\begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} \times \begin{bmatrix} p_{l11} & p_{l12} & p_{l13} & p_{l14} \\ p_{l21} & p_{l22} & p_{l23} & p_{l24} \\ p_{l31} & p_{l32} & p_{l33} & p_{l34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- And for the right camera:

$$\begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} \times \begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & p_{r14} \\ p_{r21} & p_{r22} & p_{r23} & p_{r24} \\ p_{r31} & p_{r32} & p_{r33} & p_{r34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving for X Y Z

- If we carry out the cross product, the first two rows of the left camera vector equations are:

$$\begin{bmatrix} (y_l p_{l31} - p_{l21})X + (y_l p_{l32} - p_{l22})Y + (y_l p_{l33} + p_{l23})Z + (y_l p_{l34} - p_{l24}) \\ (p_{l11} - x_l p_{l31})X + (p_{l12} - x_l p_{l32})Y + (p_{l13} - x_l p_{l33})Z + (p_{l14} - x_l p_{l34}) \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ * \end{bmatrix}$$

- Similarly, for the right camera, the first two rows are:

$$\begin{bmatrix} (y_r p_{r31} - p_{r21})X + (y_r p_{r32} - p_{r22})Y + (y_r p_{r33} + p_{r23})Z + (y_r p_{l34} - p_{r24}) \\ (p_{r11} - x_r p_{l31})X + (p_{r12} - x_r p_{l32})Y + (p_{r13} - x_r p_{l33})Z + (p_{r14} - x_r p_{l34}) \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ * \end{bmatrix}$$

Solving for X Y Z

- The four equations can then be brought into the "$A\theta = b$" form:

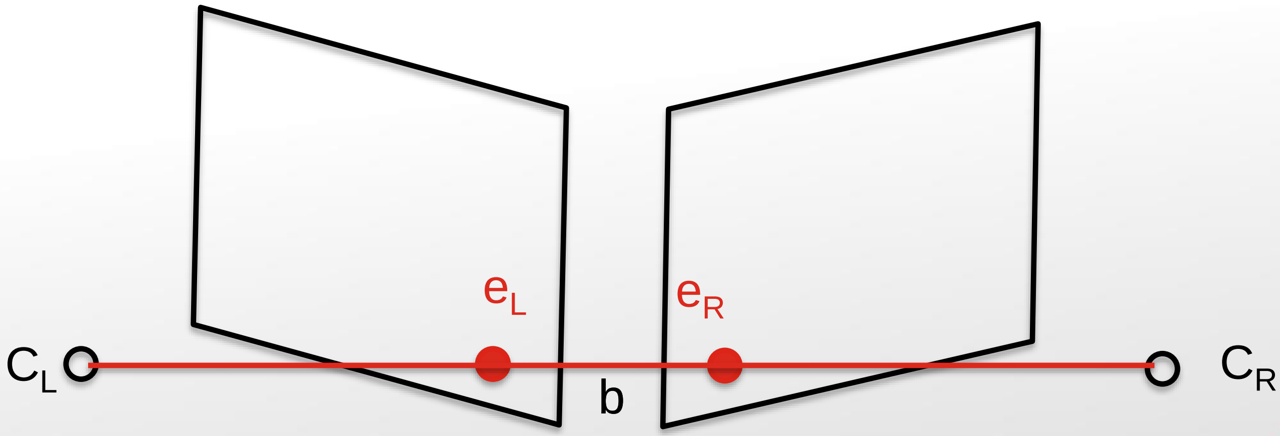
$$\begin{bmatrix} (y_l p_{l31} - p_{l21}) & (y_l p_{l32} - p_{l22}) & (y_l p_{l33} + p_{l23}) \\ (p_{l11} - x_l p_{l31}) & (p_{l12} - x_l p_{l32}) & (p_{l13} - x_l p_{l33}) \\ (y_r p_{r31} - p_{r21}) & (y_r p_{r32} - p_{r22}) & (y_r p_{r33} + p_{r23}) \\ (p_{r11} - x_r p_{l31}) & (p_{r12} - x_r p_{l32}) & (p_{r13} - x_r p_{l33}) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (p_{l24} - y_l p_{l34}) \\ (x_l p_{l34} - p_{l14}) \\ (p_{r24} - y_r p_{l34}) \\ (x_r p_{l34} - p_{r14}) \end{bmatrix}$$

- We can finally solve for X, Y, Z using least squares method.

$$\theta = (A^T A)^{-1} A^T b$$

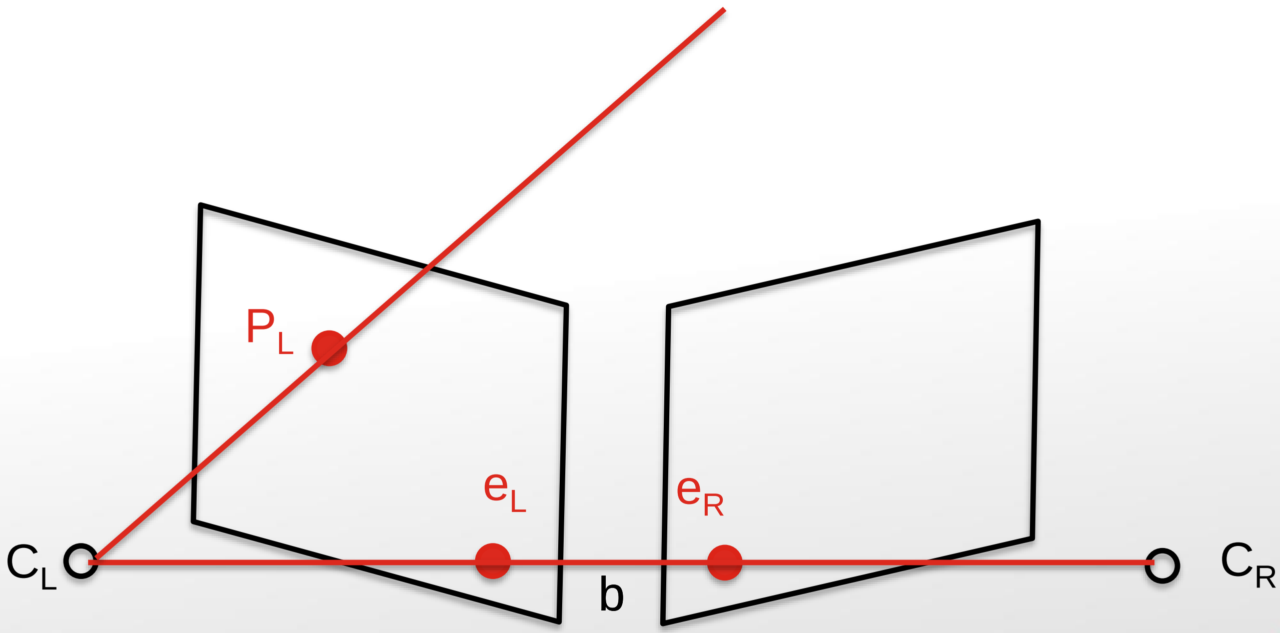
Correspondence Problem

- The concept of correspondence matching is still the same (as coplanar case) – We need to find matching portions of the left and right images.
  - Use SSD, SAD, CC or NCC as discussed earlier.
- However, the epipolar line is not necessarily horizontal anymore!
- Firstly, introduce the terms "epipoles (e_L and e_R)", i.e. the points where the camera baseline (C_L-C_R line) hits the left and right image planes.



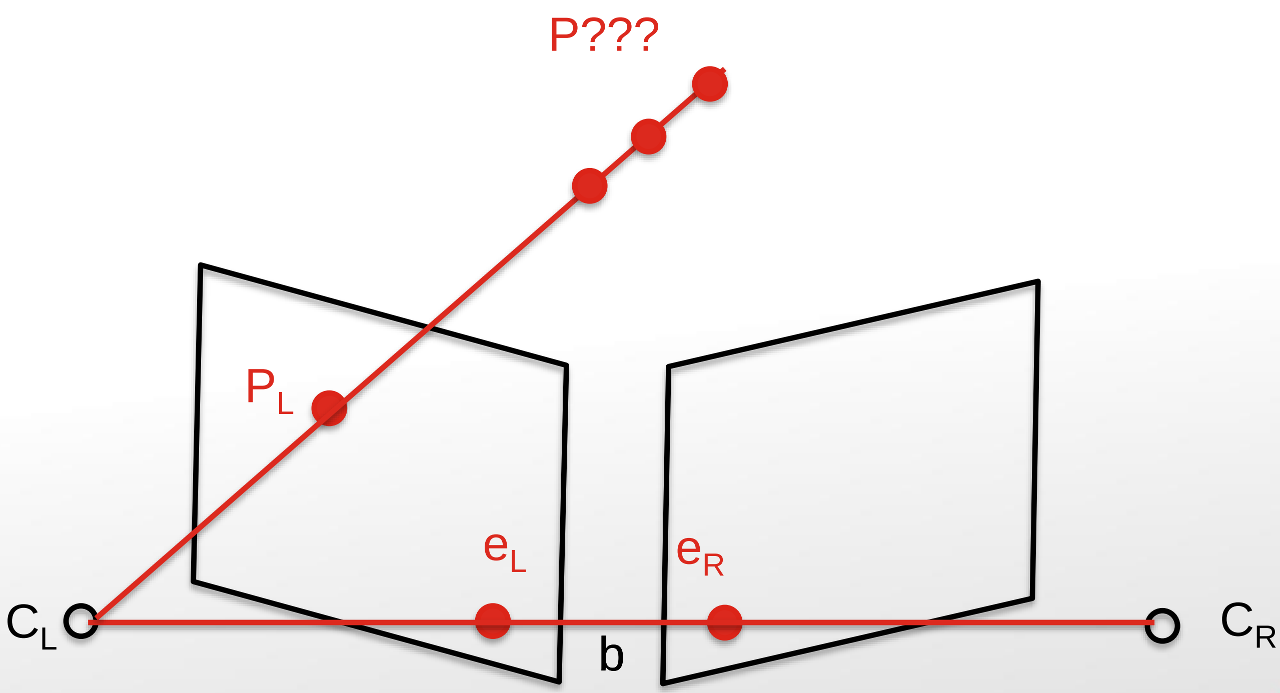
Epipolar Constraint

- Next, assume we have an image point on the left image.
- So the ray from C_L through P_L is known.



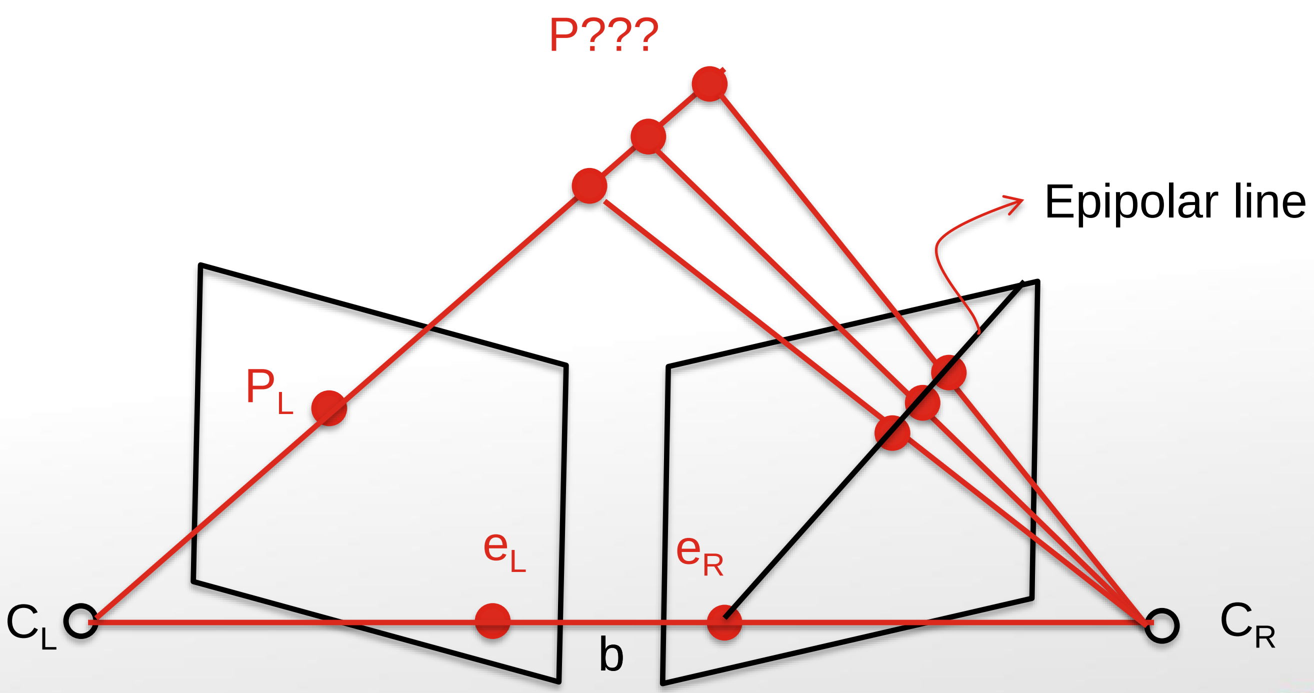
Epipolar Constraint

- However, where is the actual point P?
- It must lie along the C_L-P_L ray!



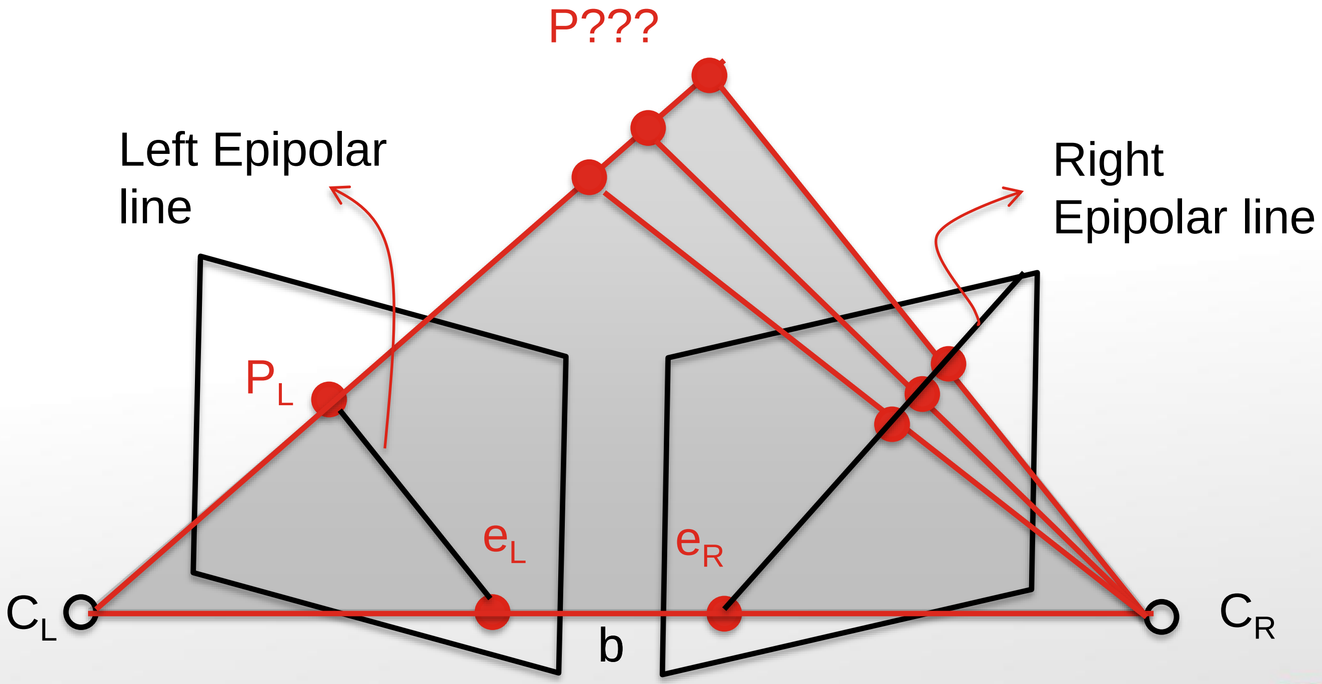
Epipolar Constraint

- If we project all these possible P's to the right camera,
- They would all lie along the epipolar line as shown in the figure.
  - We only need to search along this line for correspondence matching!



Epipolar Constraint

- Note that the left and right epipolar lines lie on the epipolar plane, which contains C_L, P_L and e_L.
- Thus once we know these three points, we can find out the epipolar lines.



Tutorial Assignments

- There is no tutorial assignment for this week.

Thank you!

Have a good evening.