SPATIAL SENSING

3D SENSOR DATA REPRESENTATION AND MODELLING

Dr TIAN Jing
tianjing@nus.edu.sg

Module objective

Knowledge and understanding

- Understand the fundamentals of spatial sensing: 3D sensor data representation and modelling, such as camera model, feature extraction and matching from multi-view images.

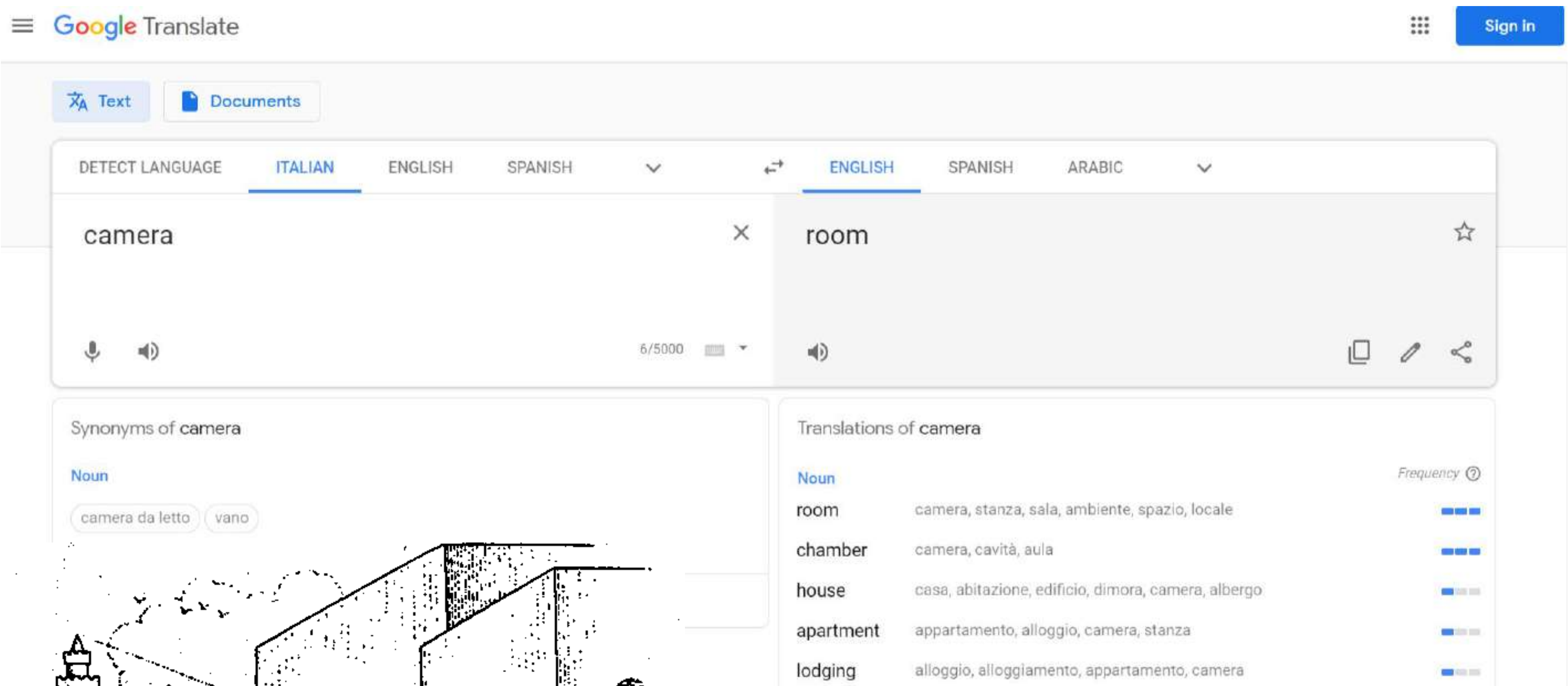
Key skills

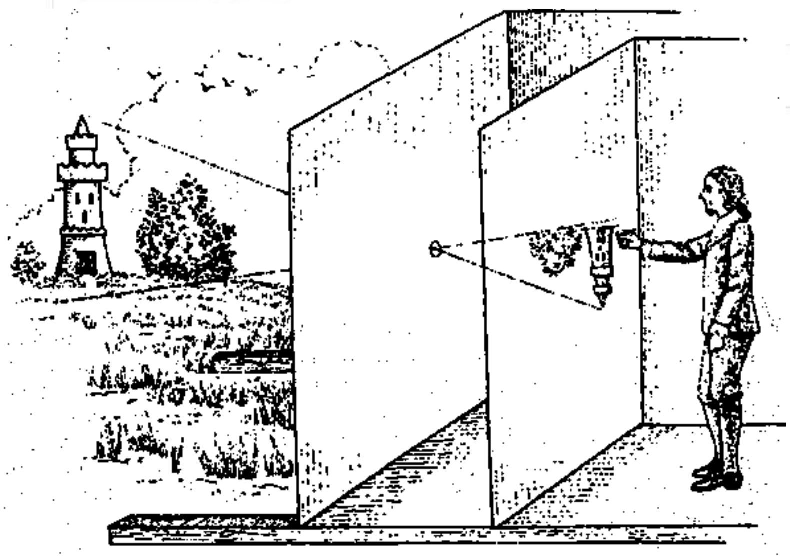
- Workshop on 3D sensor data representation and modelling, such as constructing 3D scene map based on image/video captured by the camera

Major reference

- [Most relevant] Vision Algorithms for Mobile Robotics, http://rpg.ifi.uzh.ch/teaching.html
- [Advanced] EE290T, Advanced Topics in Signal Processing: 3D Image Processing and Computer Vision, http://inst.eecs.berkeley.edu/~ee290t/fa19/
- [Advanced] CS231A: Computer Vision, From 3D Reconstruction to Recognition, http://web.stanford.edu/class/cs231a/index.html
- [Comprehensive] R. Szeliski, Computer Vision: Algorithms and Applications, http://szeliski.org/Book/

What is camera?

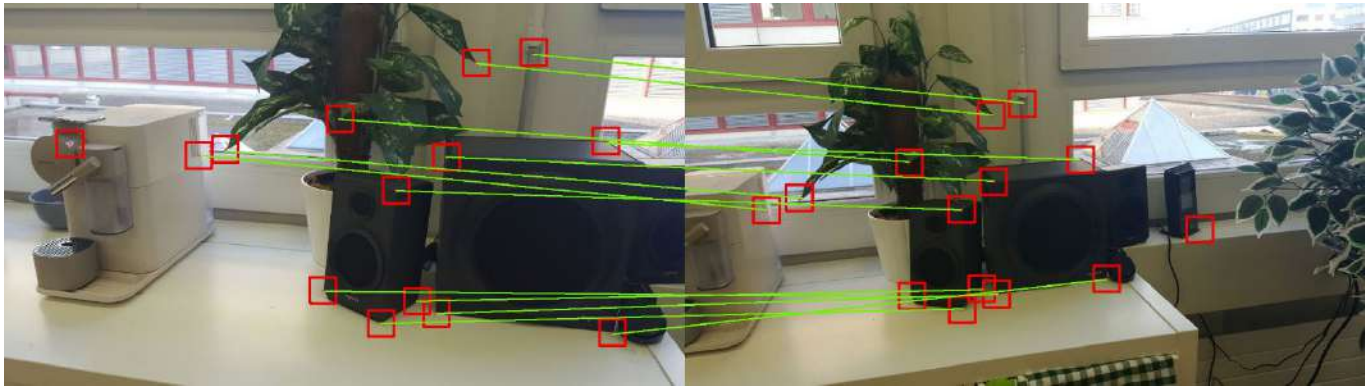




Known during classical period in China and Greece (470BC to 390BC), https://en.wikipedia.org/wiki/Camera_obscura

Roadmap of following slides

Input data
(multiple images)



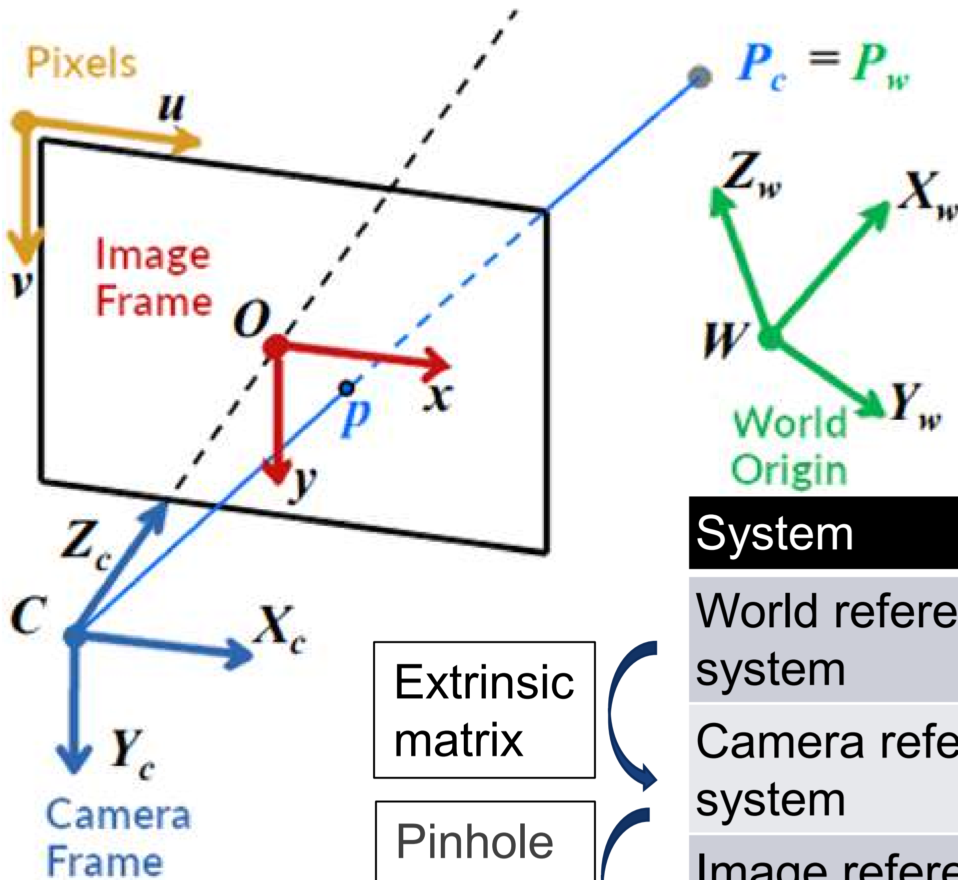
Methods focused
today (next module)

Deliverables

1. Transformation between input images
2. 3D position in the physical world

Image: http://rpg.ifi.uzh.ch/docs/teaching/2019/01_introduction.pdf

Overview: Four reference systems in camera model



Extrinsic matrix

Pinhole camera model

Intrinsic matrix

System	Origin	Unit
World reference system	Physical world	Meter
Camera reference system	Camera center point	Meter
Image reference system	Center point of image plane (*charge-coupled device* (CCD))	Millimeter
Pixel reference system	Top left point of the image	Pixel

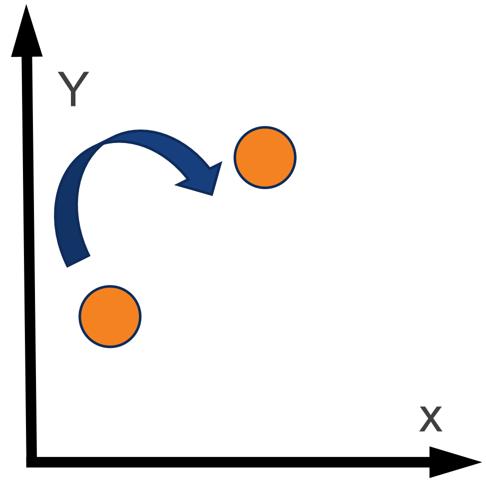
Reference: Module 2, Vision Algorithms for Mobile Robotics, http://rpg.ifi.uzh.ch/teaching.html

Preliminary: 2D transformations

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} t_x + x \\ t_y + y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{T} + \mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$



Translation	Legend
Scaling	Rotation

$\mathbf{R}$ is an orthogonal matrix, $\mathbf{R}\mathbf{R}^T = \mathbf{I}$

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} \cos\theta\, x - \sin\theta\, y \\ \cos\theta\, y + \sin\theta\, x \end{bmatrix}$$

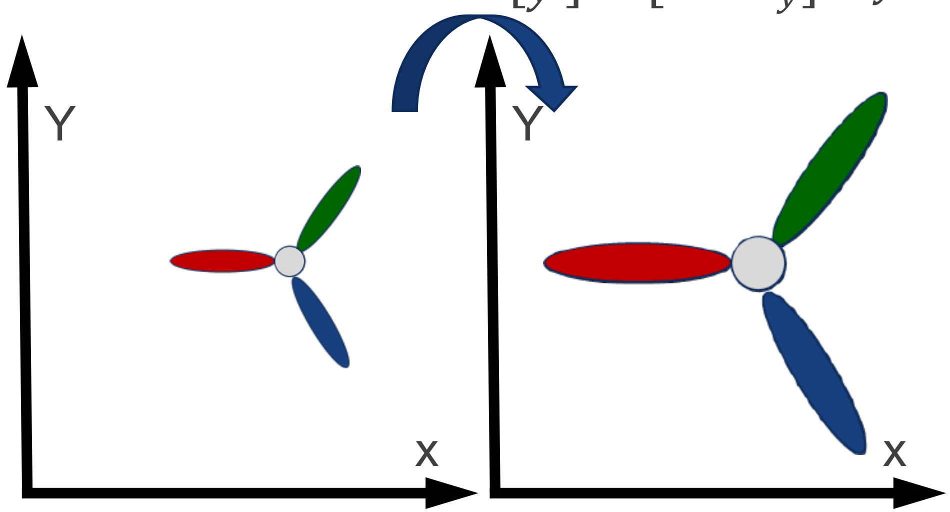
$$\mathbf{p}' = \mathbf{R} \cdot \mathbf{p}$$

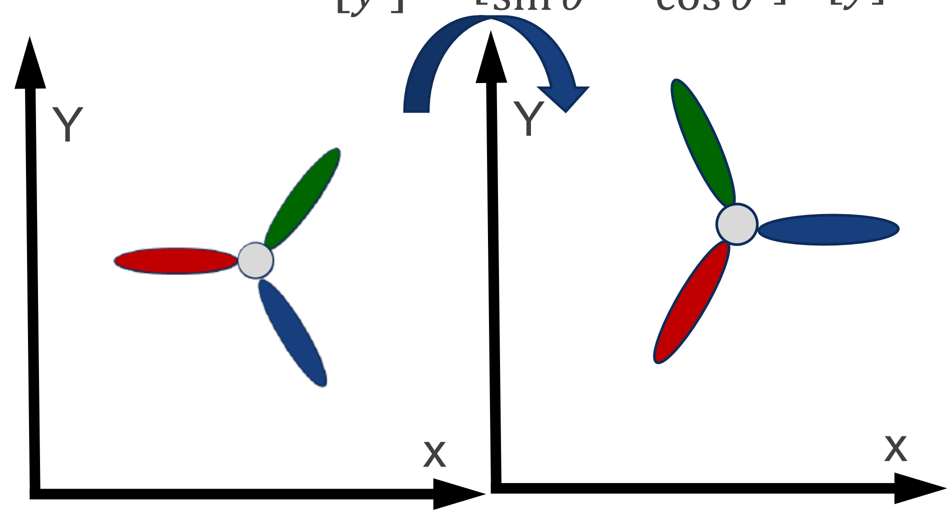
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{S} \cdot \mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$





Preliminary: Transformation matrices

- In general, a matrix multiplication lets us linearly combine components of vector

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

- This is sufficient for scaling and rotate transformations.

- How about translation?
  - Solution: Stick '1' at end of every vector, called homogeneous coordinates

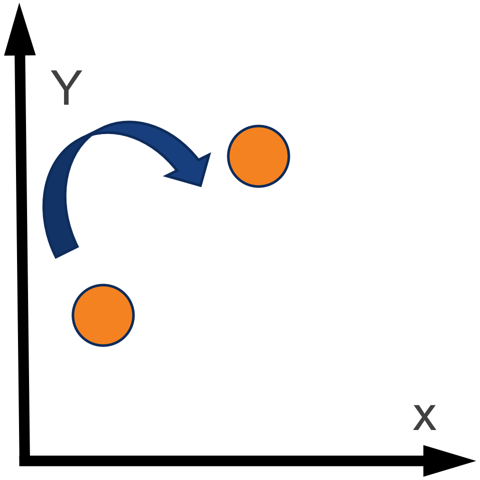
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Preliminary: 2D transformations in homogeneous coordinates

$$\mathbf{P}=\begin{bmatrix}x\\y\\1\end{bmatrix}\rightarrow\mathbf{P}'=\begin{bmatrix}t_x+x\\t_y+y\\1\end{bmatrix}$$

$$\mathbf{p}'=\mathbf{T}\cdot\mathbf{p}$$

$$\begin{bmatrix}t_x+x\\t_y+y\\1\end{bmatrix}=\begin{bmatrix}\underbrace{\begin{matrix}1&0&t_x\\0&1&t_y\\0&0&1\end{matrix}}_{\mathbf{T}}\end{bmatrix}\cdot\begin{bmatrix}x\\y\\1\end{bmatrix}$$



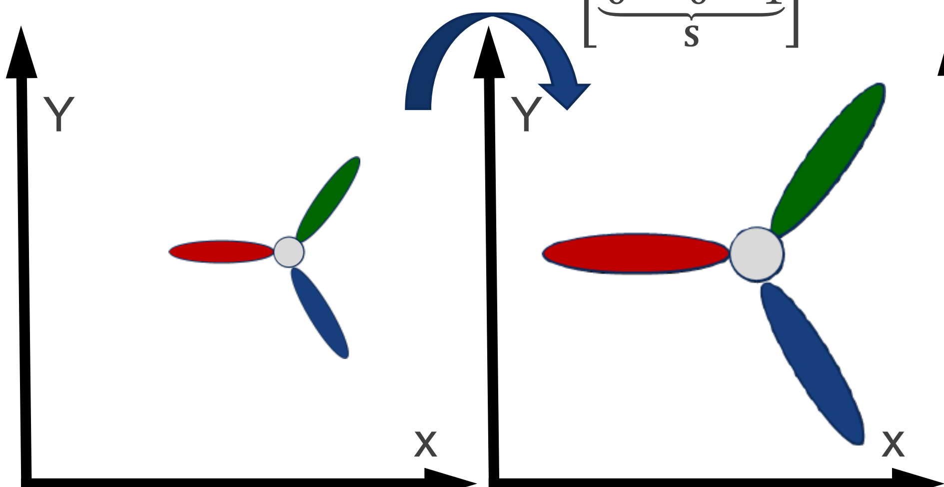
Translation	Legend
Scaling	Rotation

$$\mathbf{P}=\begin{bmatrix}x\\y\\1\end{bmatrix}\rightarrow\mathbf{P}'=\begin{bmatrix}\cos\theta\, x-\sin\theta\, y\\\cos\theta\, y+\sin\theta\, x\\1\end{bmatrix}$$

$$\mathbf{P}=\begin{bmatrix}x\\y\\1\end{bmatrix}\rightarrow\mathbf{P}'=\begin{bmatrix}s_x x\\s_y y\\1\end{bmatrix}$$

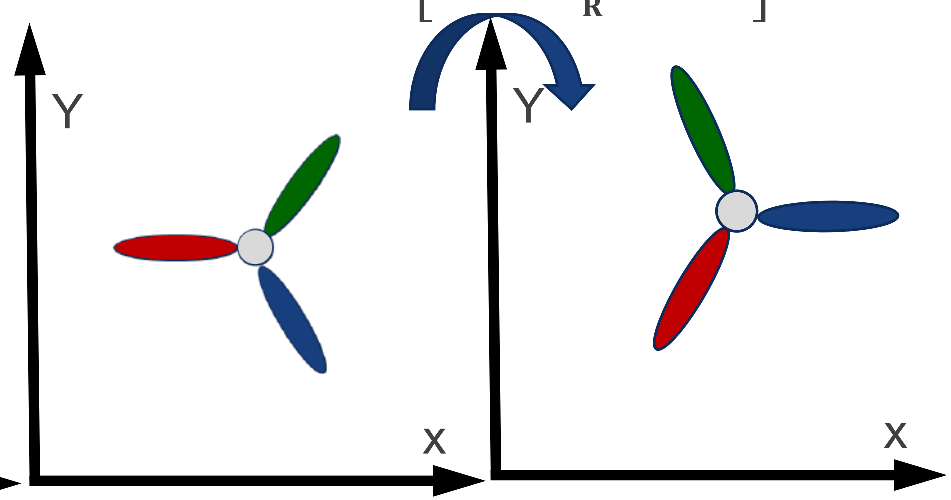
$$\mathbf{p}'=\mathbf{S}\cdot\mathbf{p}$$

$$\begin{bmatrix}s_x x\\s_y y\\1\end{bmatrix}=\begin{bmatrix}\underbrace{\begin{matrix}s_x&0&0\\0&s_y&0\\0&0&1\end{matrix}}_{\mathbf{S}}\end{bmatrix}\cdot\begin{bmatrix}x\\y\\1\end{bmatrix}$$



$$\mathbf{p}'=\mathbf{R}\cdot\mathbf{p}$$

$$\begin{bmatrix}\cos\theta\, x-\sin\theta\, y\\\cos\theta\, y+\sin\theta\, x\\1\end{bmatrix}=\begin{bmatrix}\underbrace{\begin{matrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{matrix}}_{\mathbf{R}}\end{bmatrix}\cdot\begin{bmatrix}x\\y\\1\end{bmatrix}$$

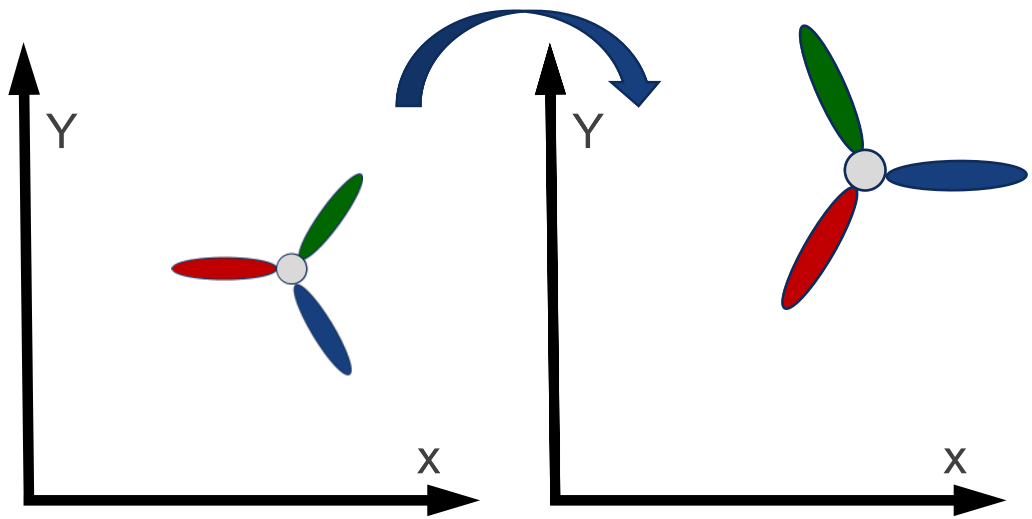


Preliminary: 2D transformations in homogeneous coordinates

Sequential transformations, e.g., Scaling + Rotation + Translation

$$\mathbf{p}' = \mathbf{T} \cdot \mathbf{R} \cdot \mathbf{S} \cdot \mathbf{p}$$

$$\mathbf{p}' = \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \cdot \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}} \cdot \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Camera model: Extrinsic matrix

3D transformations in homogeneous coordinates

Translation: $\mathbf{p}' = \mathbf{T} \cdot \mathbf{p}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{matrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{matrix}}_{\mathbf{T}} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaling: $\mathbf{p}' = \mathbf{S} \cdot \mathbf{p}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{matrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}_{\mathbf{S}} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation: around z axis $\mathbf{p}' = \mathbf{R}_z \cdot \mathbf{p}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{matrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}_{\mathbf{R}_z} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Convert from world reference system $\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$ To be camera reference system $\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$

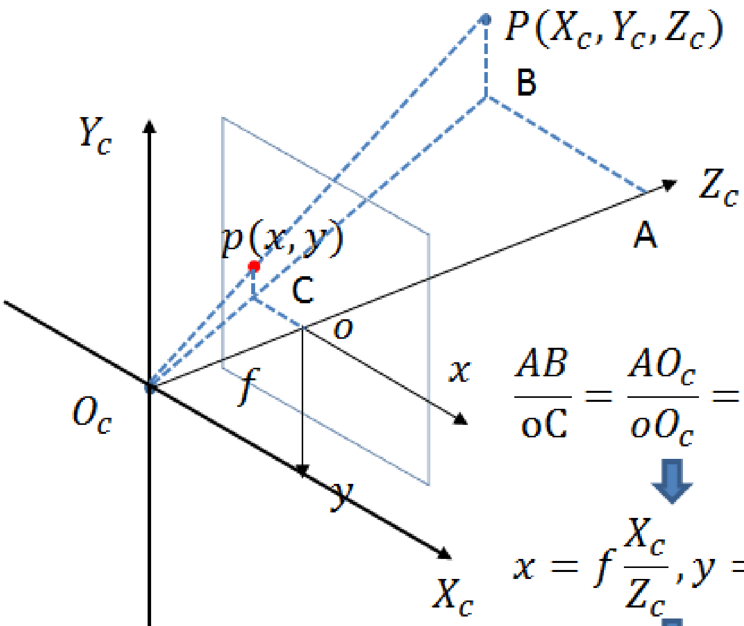
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \text{Extrinsic matrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Camera model: Pinhole camera

Pinhole camera model: Convert from camera reference system $P(X_c, Y_c, Z_c)$ to the image reference system $P(x, y)$. It depends on camera model focal length f.

Convert from camera reference system $\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$ To be image reference system $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f/z_c & 0 & 0 & 0 \\ 0 & f/z_c & 0 & 0 \\ 0 & 0 & 1/z_c & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$



Triangle similarity theorem

$$\frac{AB}{oC} = \frac{AO_c}{oO_c} = \frac{PB}{pC} = \frac{X_c}{x} = \frac{Z_c}{f} = \frac{Y_c}{y}$$

Re-arrange as matrix format

$$x = f\frac{X_c}{Z_c}, y = f\frac{Y_c}{Z_c}$$

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

$\Delta ABO_c \sim \Delta oCO_c$

$\Delta PBO_c \sim \Delta pCO_c$

Reference: Module 2, Vision Algorithms for Mobile Robotics, http://rpg.ifi.uzh.ch/teaching.html

Camera model: Intrinsic matrix

Suppose for the CMOS/CCD sensor, each pixel has a physical size d_x, d_y, the image plane origin is located at the position $(u_0, v_0, 1)$, then $u = \frac{x}{d_x} + u_0, v = \frac{y}{d_y} + v_0$

Convert from image reference system $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ To be pixel reference system $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/d_x & 0 & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/d_x & 0 & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f/z_c & 0 & 0 & 0 \\ 0 & f/z_c & 0 & 0 \\ 0 & 0 & 1/z_c & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \Rightarrow z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f/d_x & 0 & u_0 & 0 \\ 0 & f/d_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- α_x, α_y focal length in pixels
- γ skew between x and y axes (often zero)
- u_0, v_0 principal point (typically center of image)

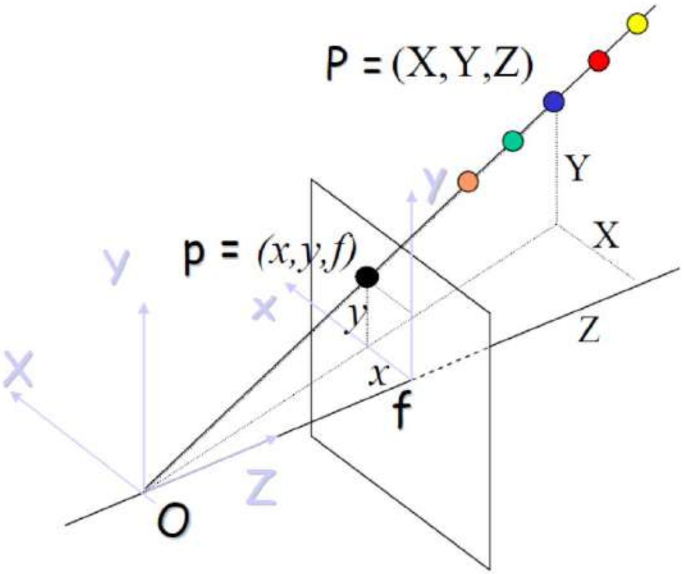
Intrinsic matrix: Convert from camera reference system to pixel reference system.

Example: Given an image resolution of 640×480 pixels and a focal length of 210 pixels, the intrinsic matrix could be

$$K = \begin{bmatrix} 210 & 0 & 320 & 0 \\ 0 & 210 & 240 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

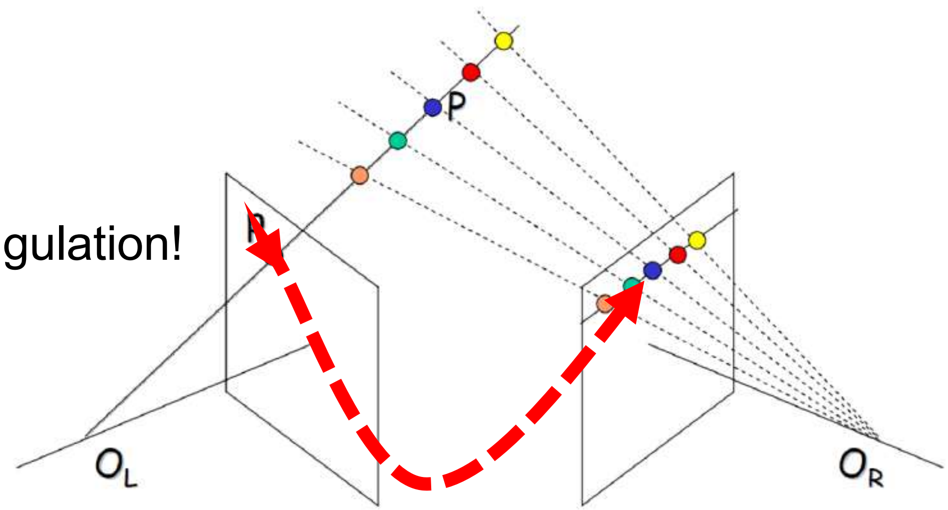
Reference
- https://www.mathworks.com/help/vision/ug/camera-calibration.html
- https://en.wikipedia.org/wiki/Camera_resectioning

Camera model: Estimation



- Ambiguity: Any point on the ray (the line from the point O to the point P) can be projected on the image point P.
- Solution: A second camera can resolve the ambiguity, enabling measurement of depth via triangulation.

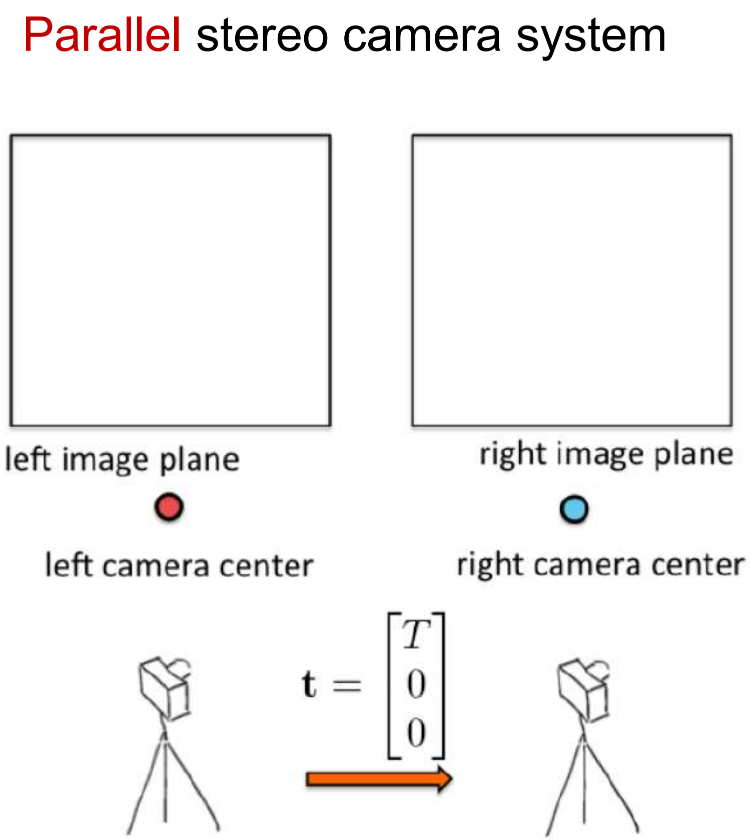
We can get a point in 3D by triangulation!

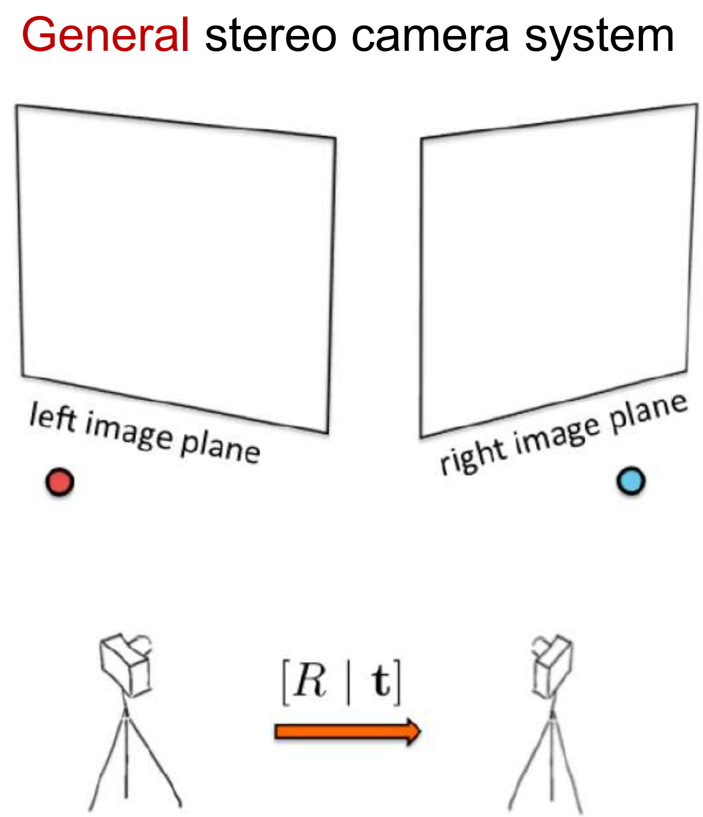


Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

Stereo vision: Camera system

Two popular stereo camera systems





Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

Stereo vision: Camera system

Given two calibrated parallel cameras (we know both intrinsic and extrinsic matrices), i.e. the right camera is some distance to the right of the left camera. According to triangle similarity theorem between blue triangle and red triangle, we have

$$\frac{T}{Z} = \frac{T + x_r^* - x_l^*}{Z - f}$$

(See slide 13)

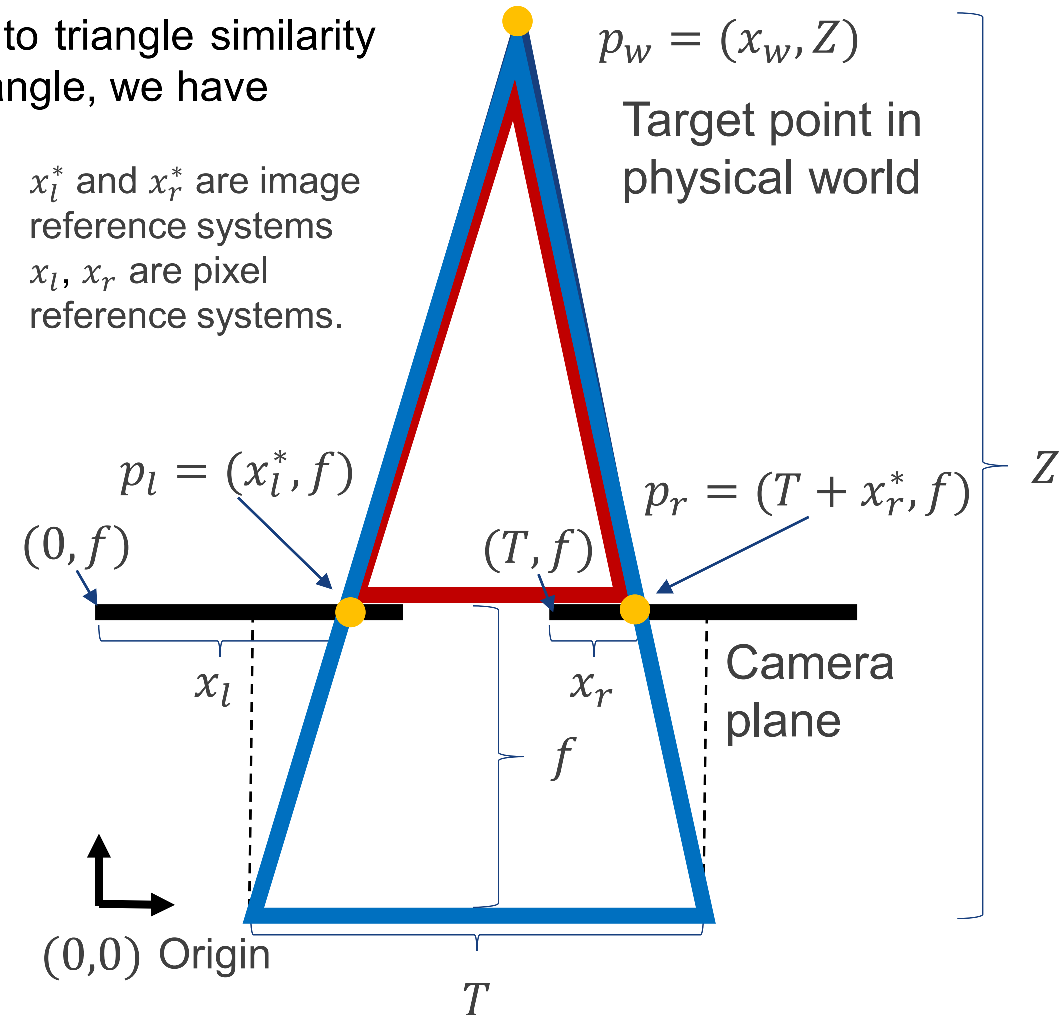
$$x_l = \frac{x_l^*}{d_x} + u_0 \quad x_r = \frac{x_r^*}{d_x} + u_0$$

$$z = \frac{fT}{d_x(x_l - x_r)}$$

x_l^* and x_r^* are image reference systems

x_l, x_r are pixel reference systems.

	Descriptions	Unit
Z	Distance between point p to camera	Physical distance, meter
T	Baseline distance between two cameras	Physical distance, meter
f	Focal length of the camera	Physical distance, meter
x_l, x_r	Locations of point p_l, p_r in images	Pixels
d_x	Physical size of a pixel in camera sensor CMOS/CCD	Physical distance per pixel



Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

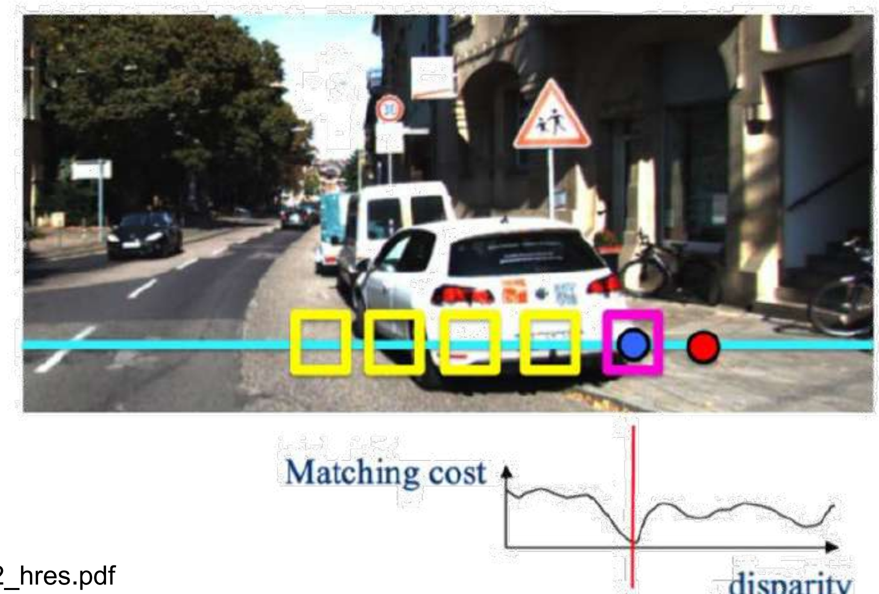
Stereo vision: Camera system

For each point $\mathbf{p}_l = (x_l, y_l)$, how to get $\mathbf{p}_r = (x_r, y_r)$ by matching?

- Idea: Image patch around (x_r, y_r) should be similar to the image patch around (x_l, y_l). We scan the line and compare patches to the one in the left image and we are looking for a patch on scanline most similar to patch on the left.
- The matching cost can be defined as *SSD* (sum of squared differences), such as

$$SSD(\text{patch}_l, \text{patch}_r) = \sum_x \sum_y \left(I_{\text{patch}_l}(x, y) - I_{\text{patch}_r}(x, y) \right)^2$$

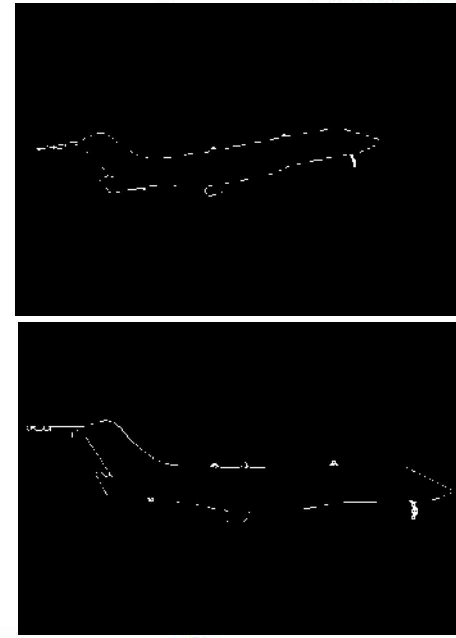
left image



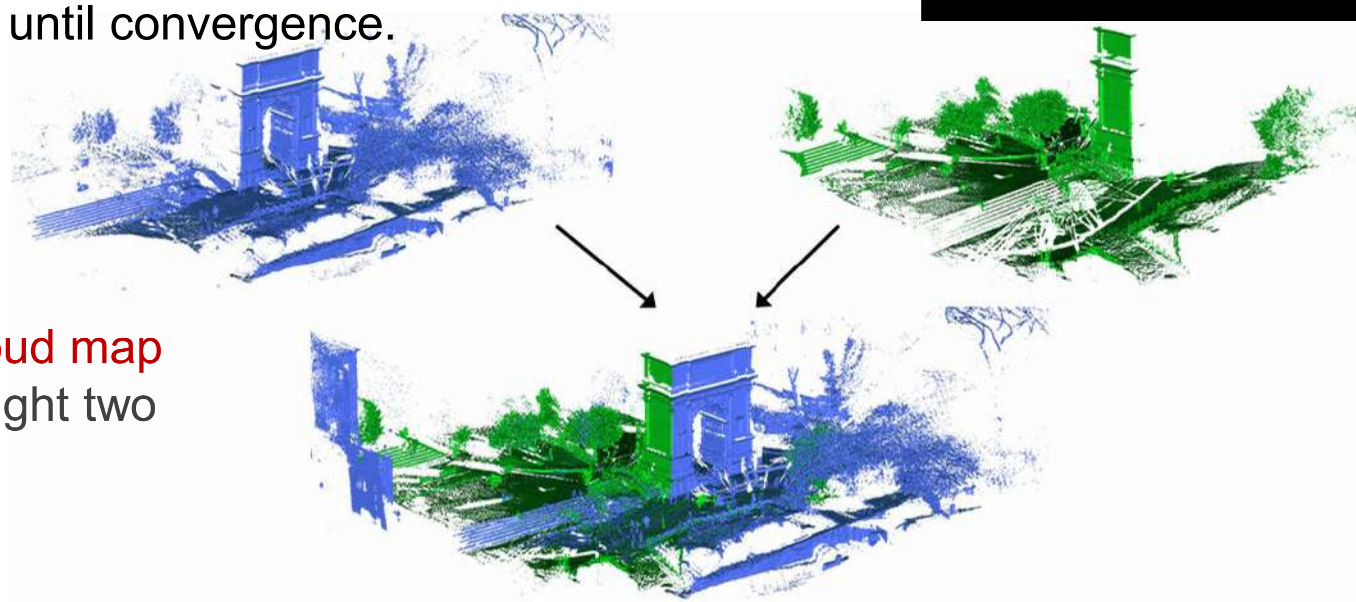
Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

Stereo vision: Iterative closest points (ICP) method

Application: Boundary alignment (see the right two airplanes)



1. Extract edge pixels $p_1, \cdots, p_n$ and $q_1, \cdots, q_m$
2. Compute initial transformation (e.g., translation and scaling)
3. For each point p_i find corresponding $\mathrm{match}_i = \mathrm{argmin}_j\, dist(p_i, q_j)$
4. Compute transformation T based on matches, warp points according to the transformation T
5. Repeat steps 3-4 until convergence.

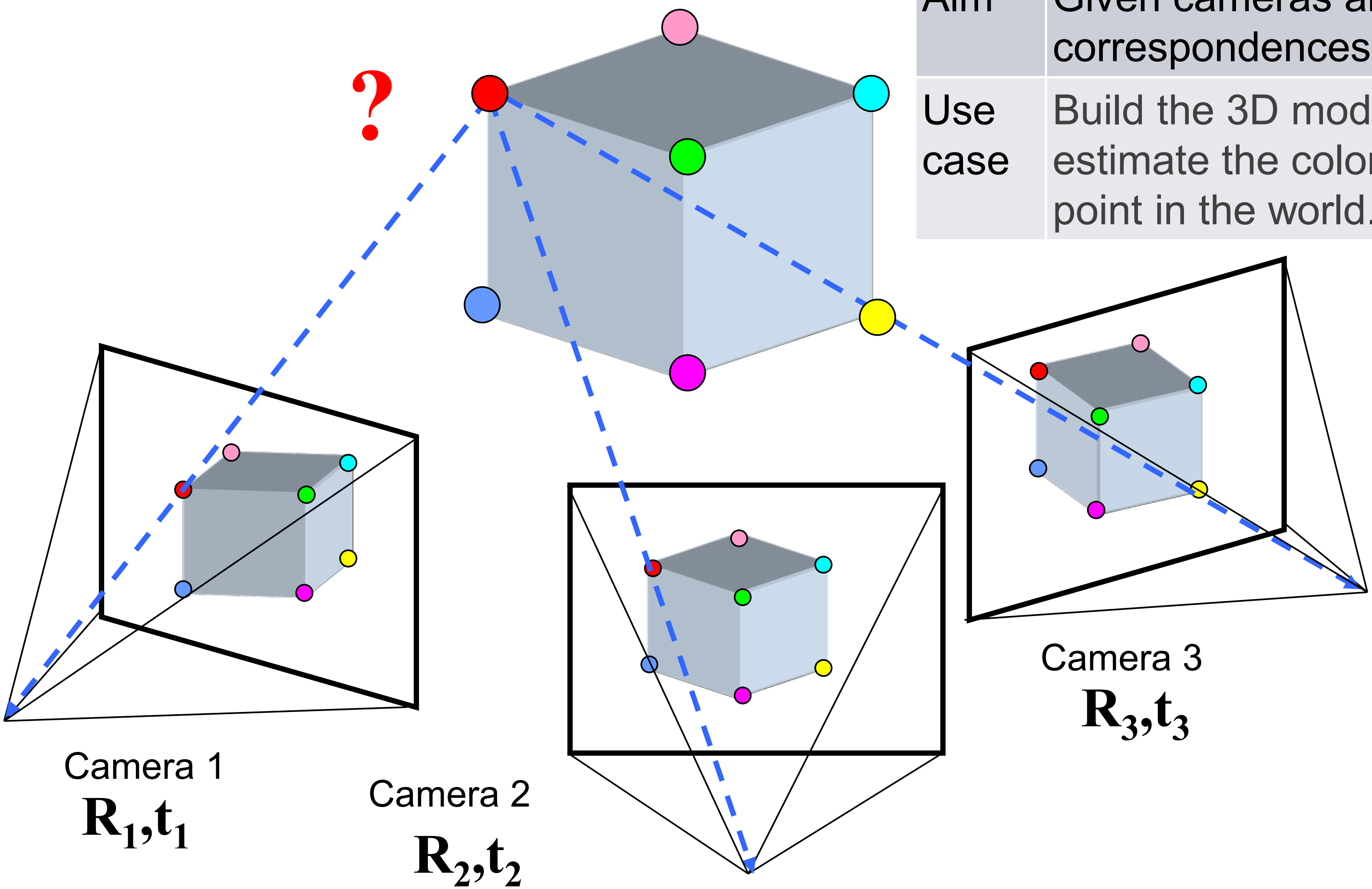


Application: Point cloud map calibration (see the right two buildings).

Photo: https://prs.igp.ethz.ch/research/completed_projects/automatic_registration_of_point_clouds.html

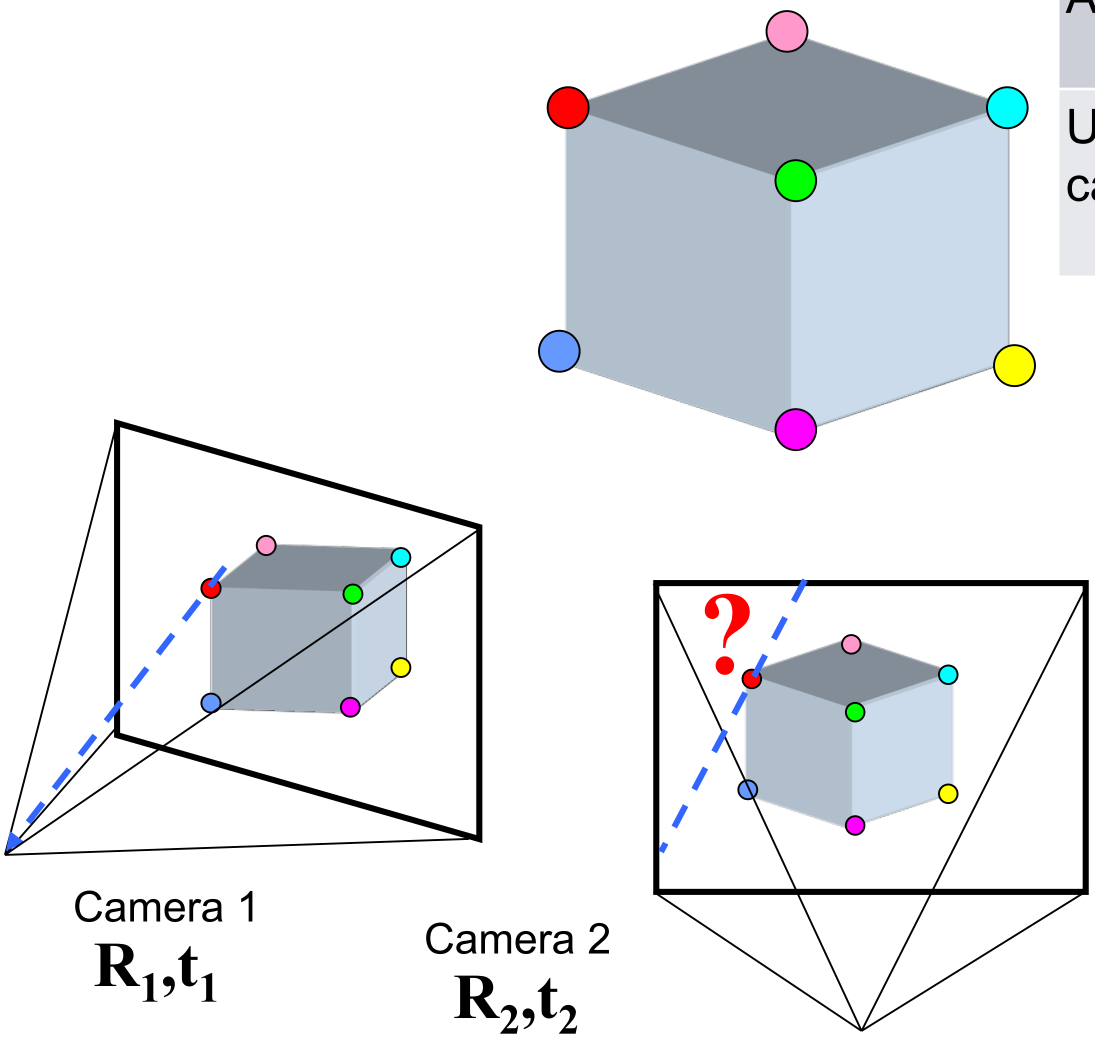
Multi-view geometry tasks

Task: Recovering structure	
Aim	Given cameras and correspondences, find 3D.
Use case	Build the 3D model to estimate the color of the point in the world.



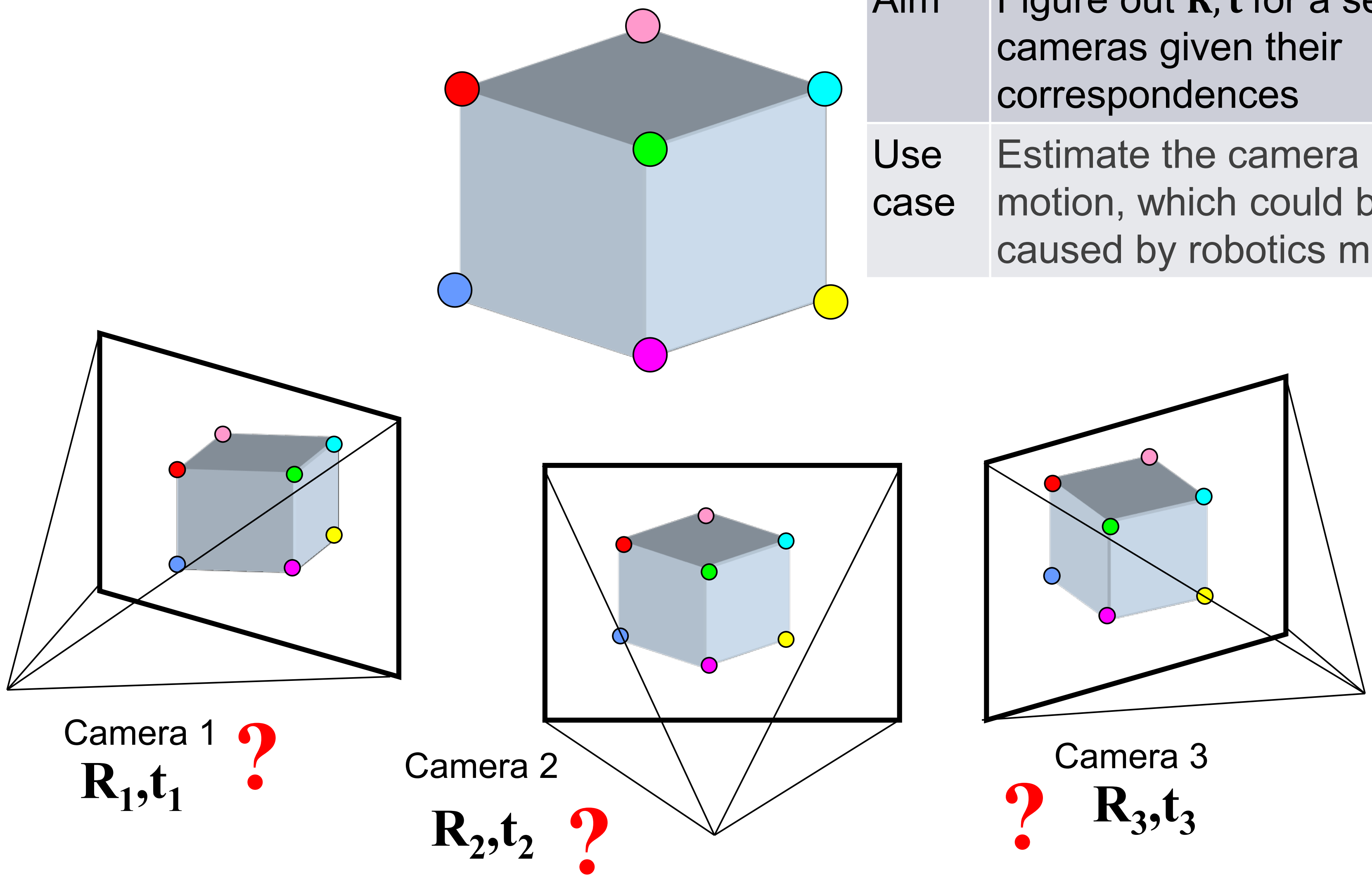
Multi-view geometry tasks

Task: Stereo geometry	
Aim	Given two cameras and find where a point could be
Use case	Find the matching points in two images so that we can do other estimation tasks.



Multi-view geometry tasks

Task: Structure from motion	
Aim	Figure out $\mathbf{R}, \mathbf{t}$ for a set of cameras given their correspondences
Use case	Estimate the camera motion, which could be caused by robotics motion

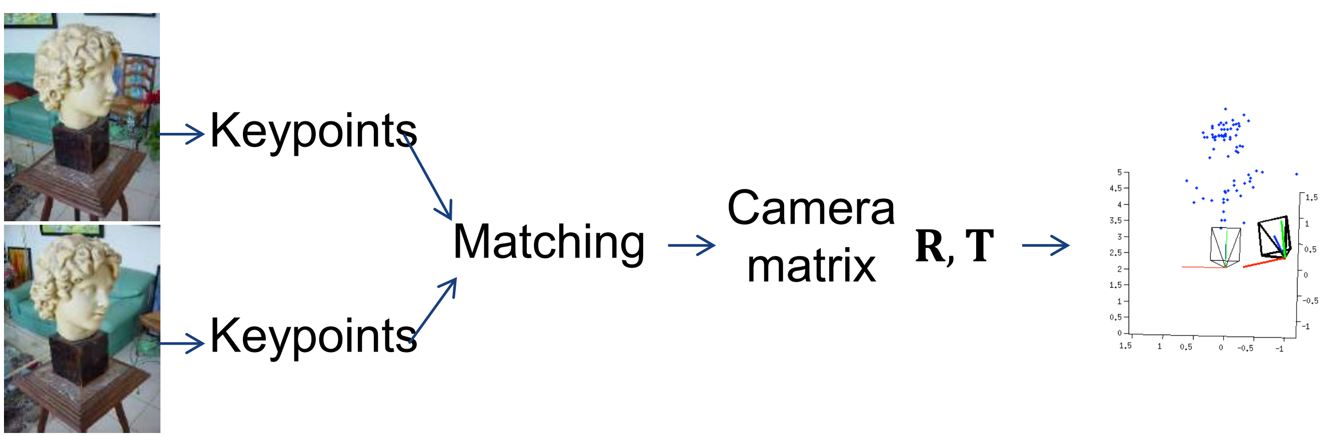


3D scene reconstruction pipeline

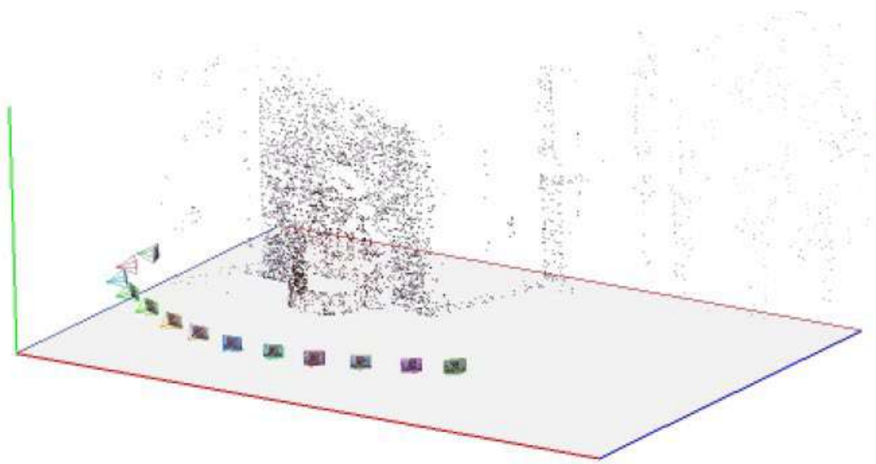
- Feature point detection
- Feature point matching between the images

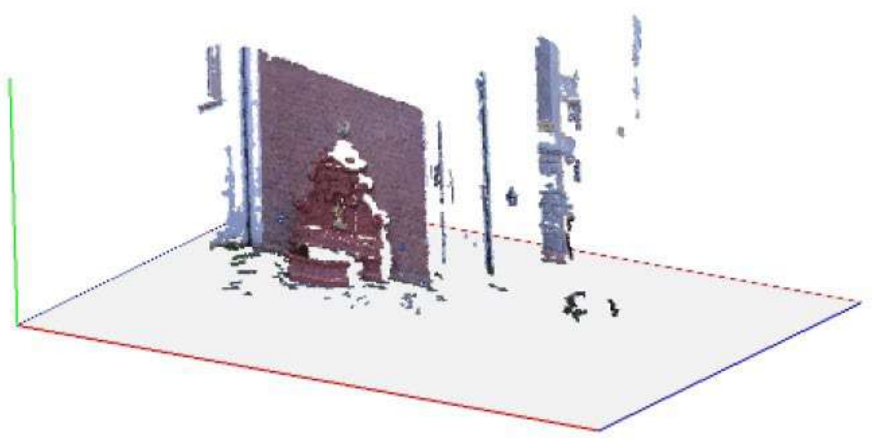
Our focus in next part

- Computation of relative camera matrix (rotation and translation) of the second camera from the first
- Use matched pairs of points to produce 3D points (infer the world coordinate from the pixel coordinates)
- Adjustment of multiple views images and transformation matrices
- Creation and plotting of 3D point cloud for visualization



3D scene reconstruction pipeline

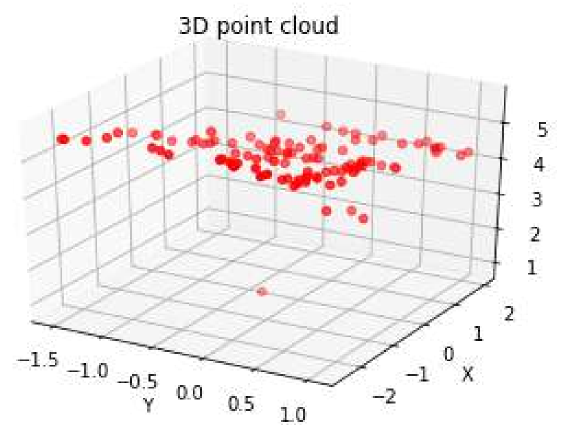




Reference:
- Fountain dataset, Dense Multi View Stereo datasets (EPFL Computer Vision Lab), http://cvlabwww.epfl.ch/data/multiview/
- S. Mccann, 3D reconstruction from multiple images, http://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/CS231a-FinalReport-sgmccann.pdf

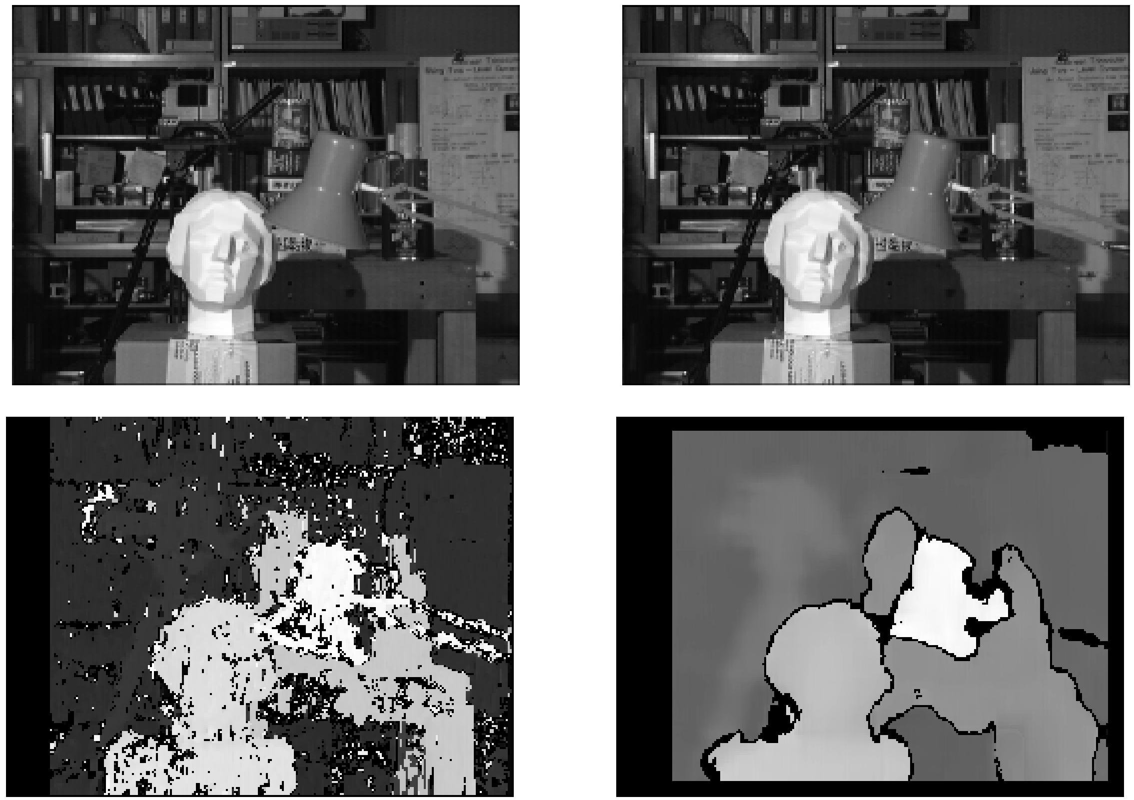
Demo: 3D scene reconstruction (point cloud) from static images

- Demo: 3D scene reconstruction (point cloud) from static images
- Dataset: SfM Camera trajectory quality evaluation, https://github.com/openMVG/SfM_quality_evaluation



Demo: Depth estimation from stereo images

- Task: Depth estimation from stereo images.
- Dataset: Middlebury Stereo Vision Dataset, http://vision.middlebury.edu/stereo/data/scenes2001/



Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg

