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.

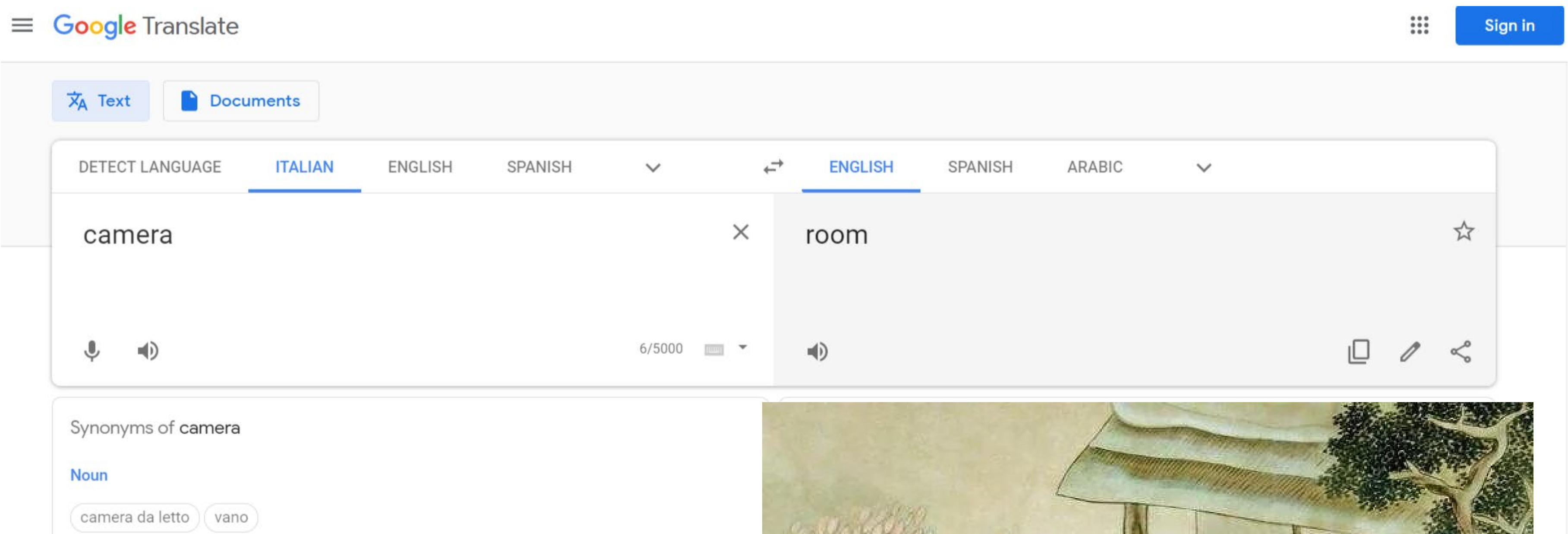
Key skills

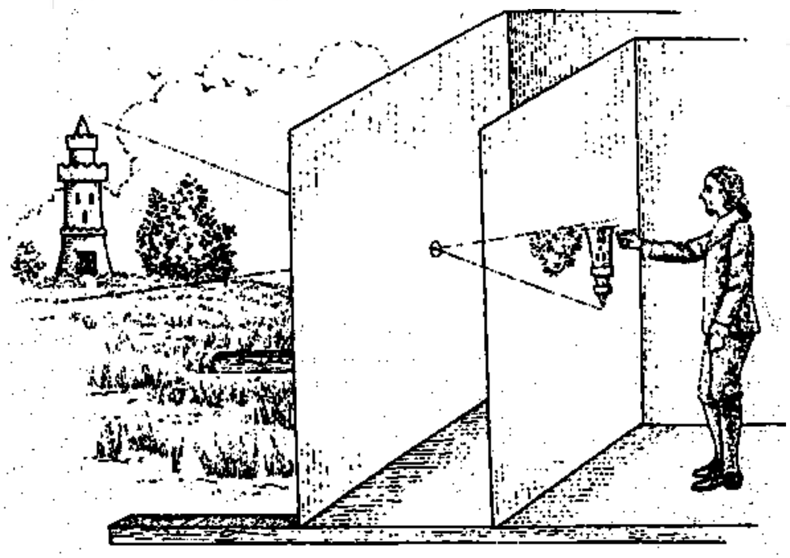
- Constructing 3D scene map based on image/video captured by the camera

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/

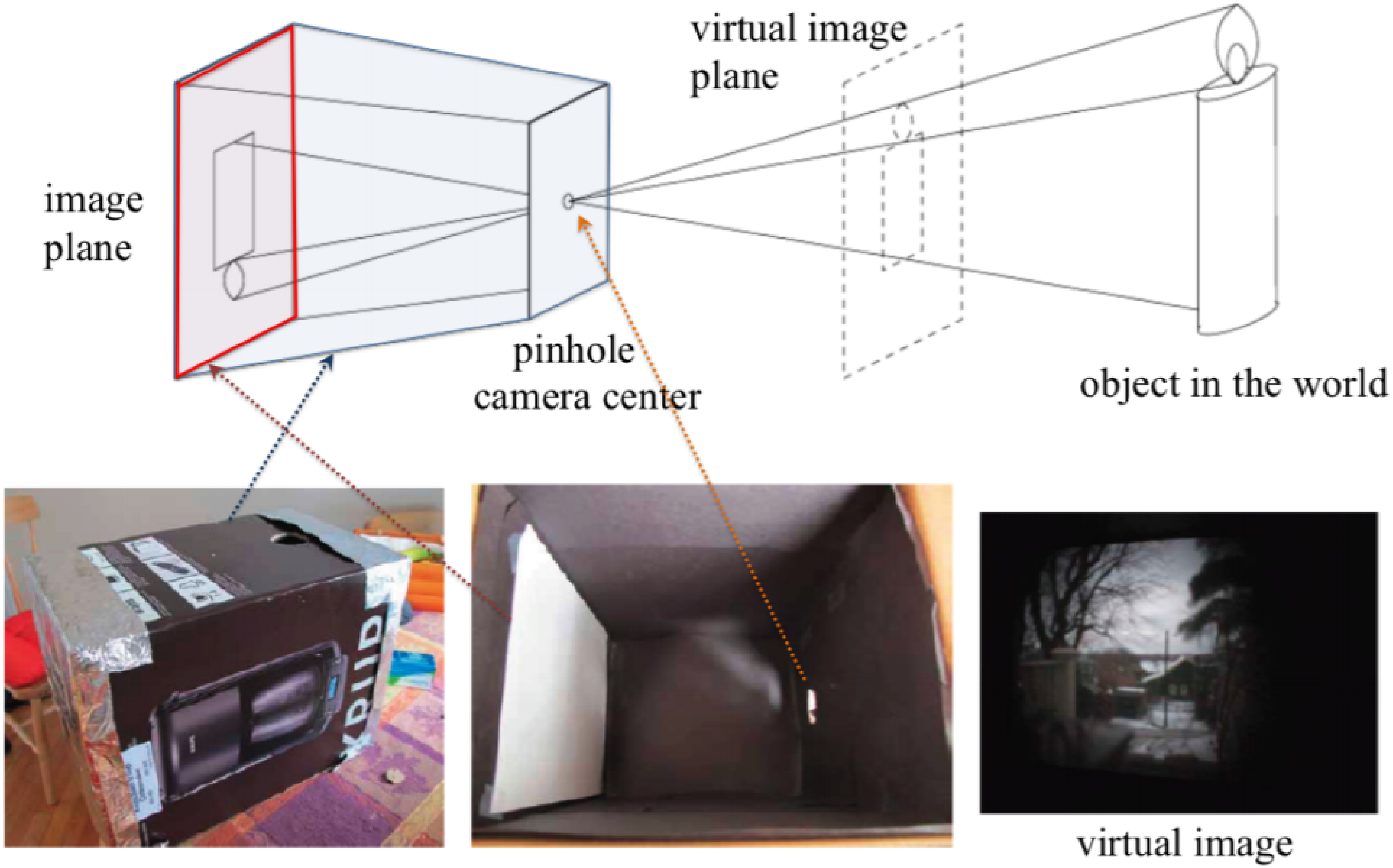
What is camera?





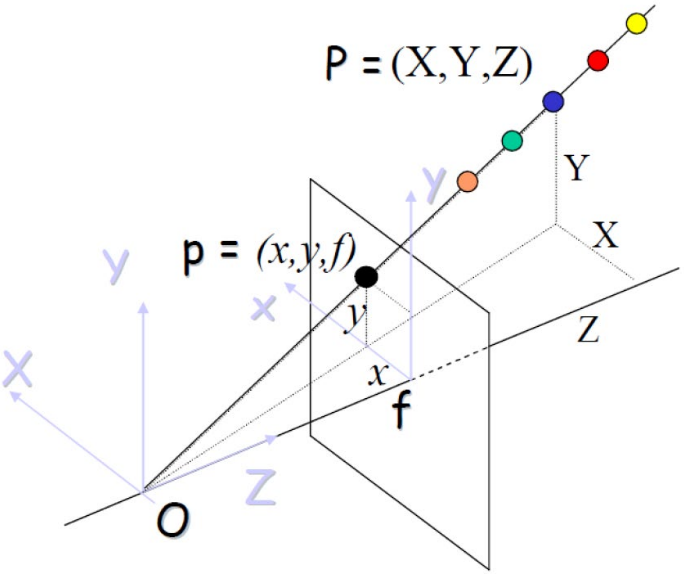
Known during classical period in China and Greece (470BC to 390BC), https://en.wikipedia.org/wiki/Camera_obscura

Camera model



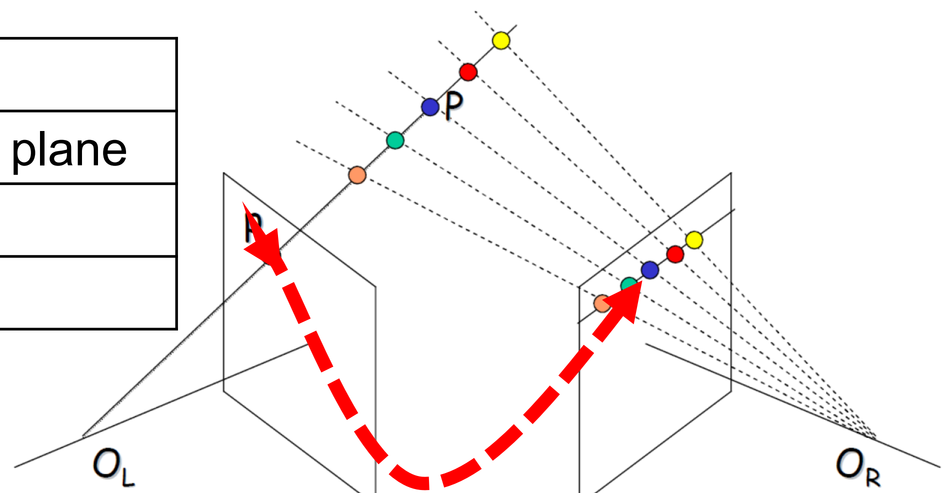
It is easier to think in a non-upsidedown world, we will work with the virtual image plane.

Q1: Distance estimation (between physical point to camera)



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

X, Y, Z	Physical position of the point P
x, y, f	Projected point P on the image plane
f	Focal length of the camera
O	Camera center

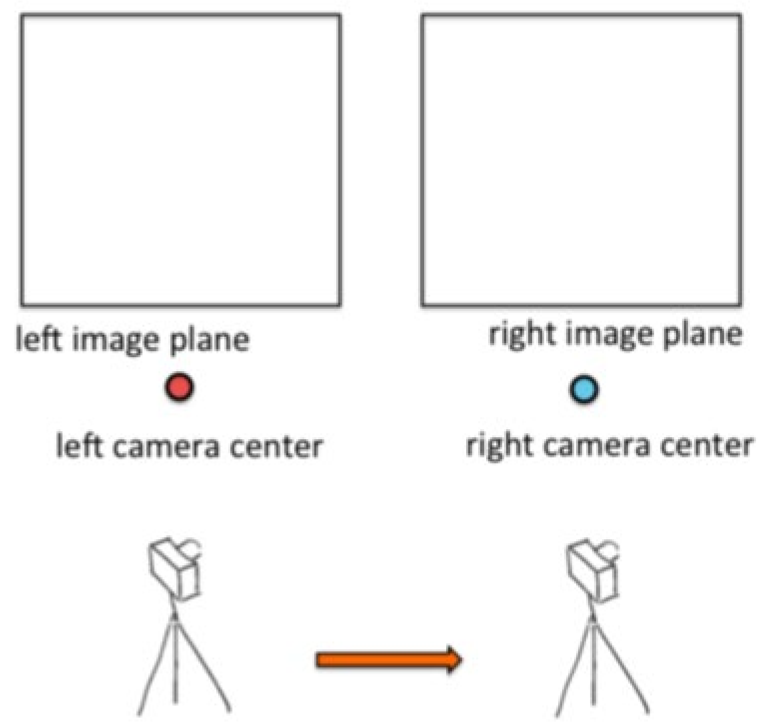


Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

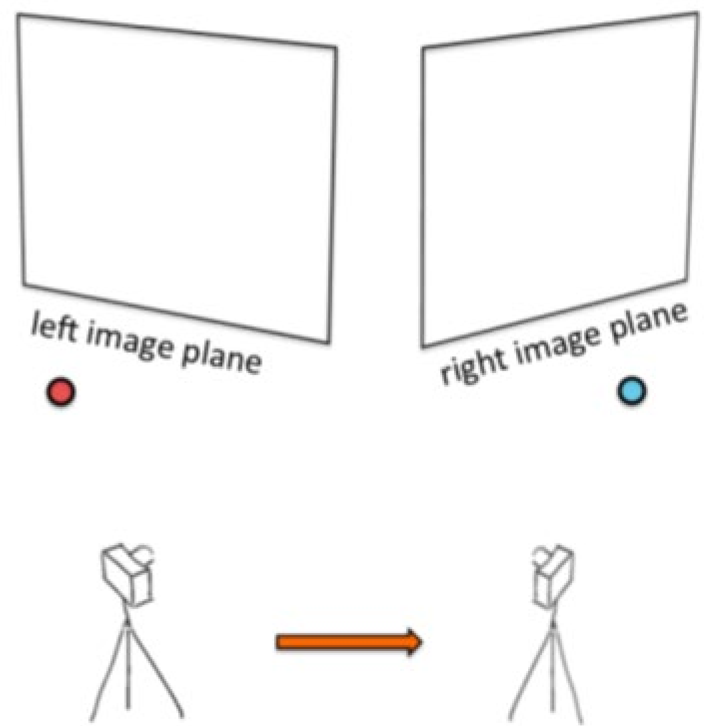
Stereo vision: Camera system

Two popular stereo camera systems

Parallel stereo camera system



General stereo camera system



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

Stereo vision: Camera system

Given two calibrated parallel cameras, 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

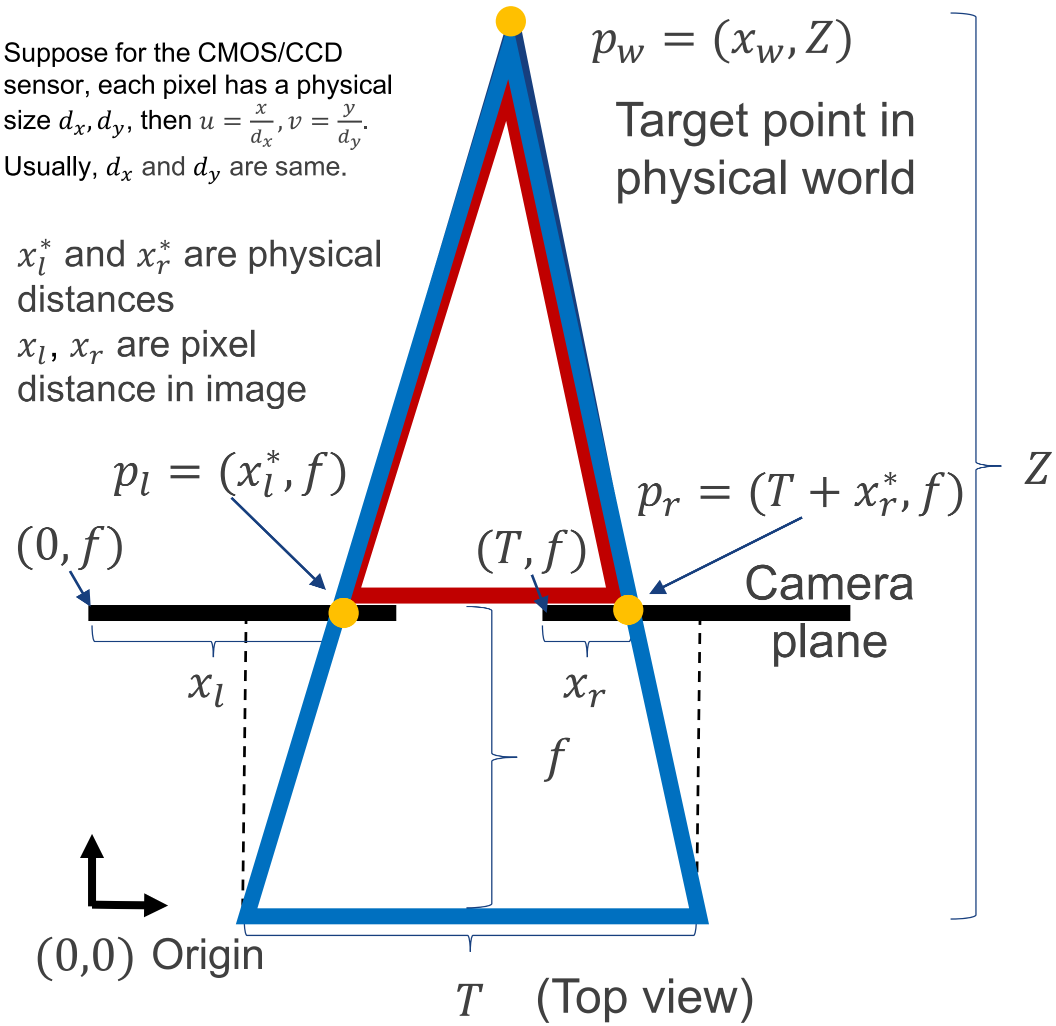
$$\frac{T}{Z} = \frac{T + x_r^* - x_l^*}{Z - f}$$

$$x_l = \frac{x_l^*}{d_x},\ x_r = \frac{x_r^*}{d_x}$$

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

Suppose for the CMOS/CCD sensor, each pixel has a physical size d_x, d_y, then $u = \frac{x}{d_x}, v = \frac{y}{d_y}$. Usually, d_x and d_y are same.

x_l^* and x_r^* are physical distances
x_l, x_r are pixel distance in image



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

	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, d_y	Physical size of a pixel in camera sensor CMOS/CCD	Physical distance per pixel

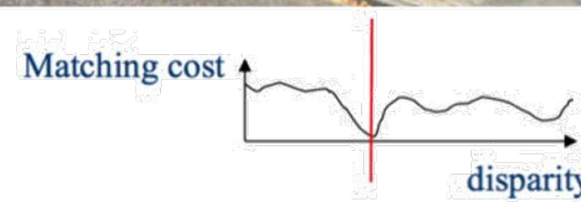
Stereo vision: Camera system

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 centered at (x_r, y_r) should be similar to the image patch centered at (x_l, y_l). We scan along the horizontal 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), e.g.,

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

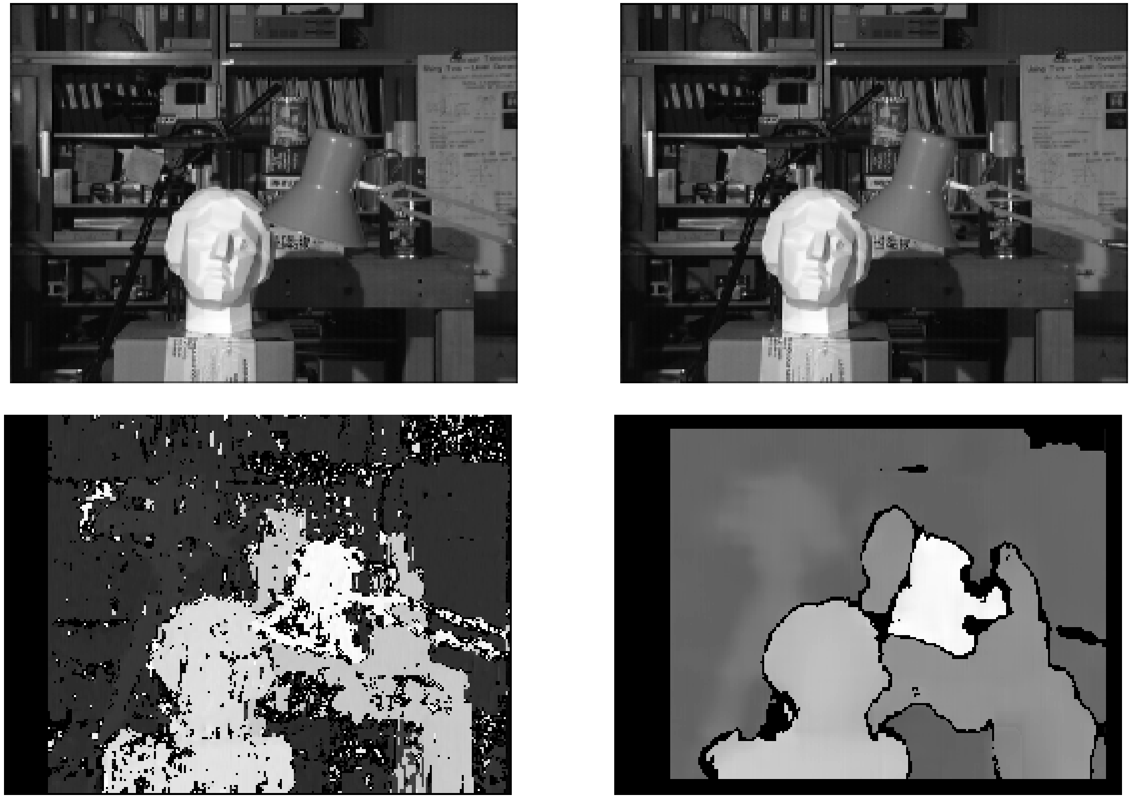
left image



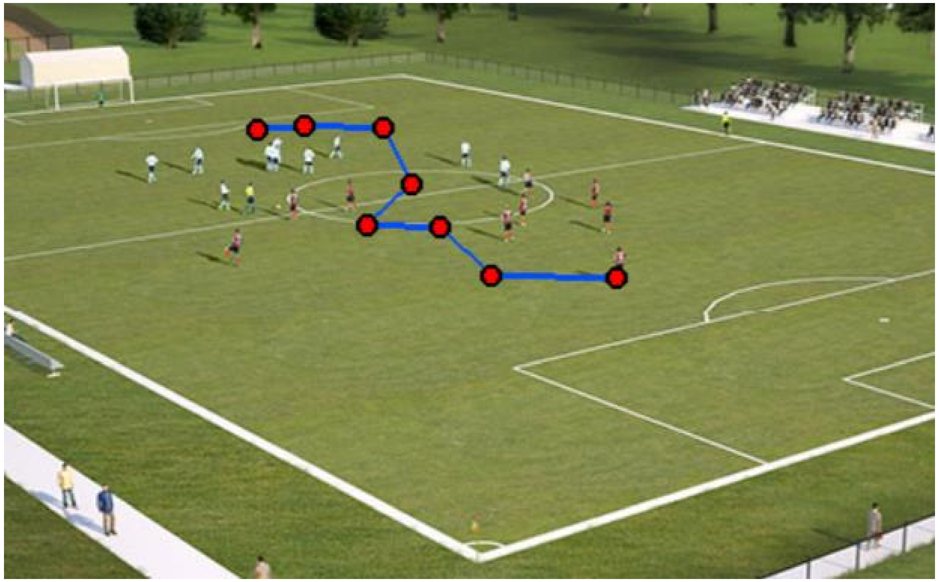
Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

Example: Depth estimation from stereo images

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



Q2: Distance estimation (between two pixel positions in the image)

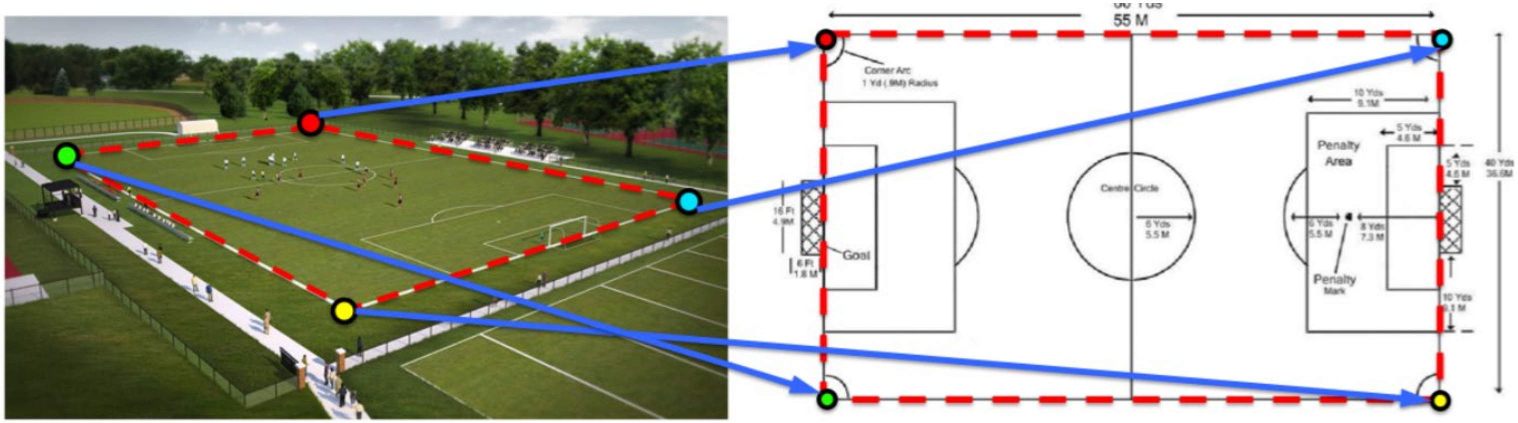


What is social distance (meters) between two detected persons?

How long (meters) has the football player run?

Reference:
- http://www.cs.toronto.edu/~fidler/slides/2021Winter/CSC420/lecture9.pdf
- https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers-monitor-social-distancing-in-the-workplace/

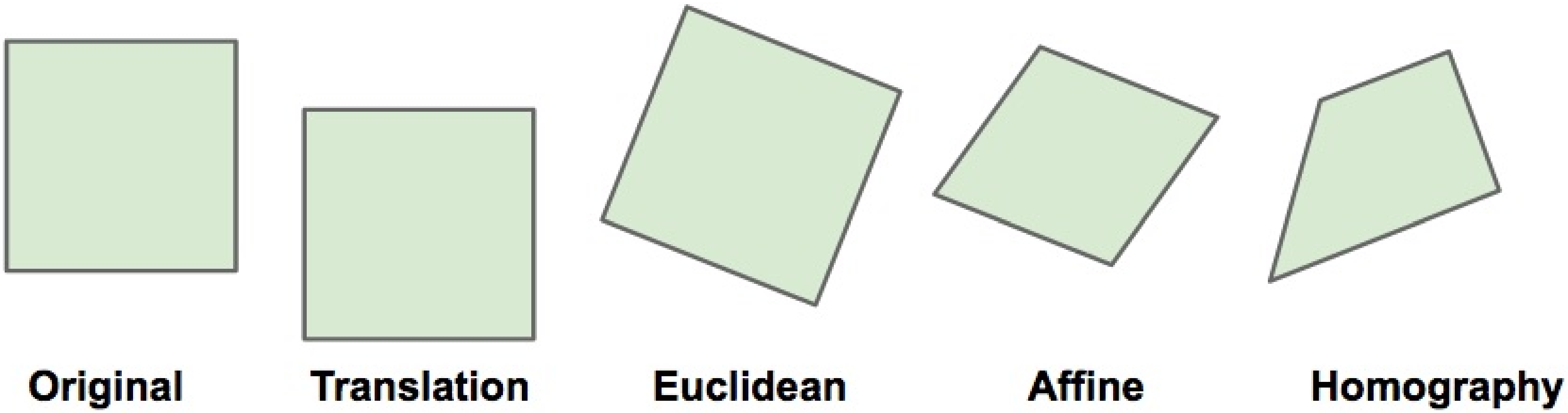
Intuition: Transform the CCTV view to a bird-view (top-view)



Reference:

- http://www.cs.toronto.edu/~fidler/slides/2021Winter/CSC420/lecture9.pdf
- https://github.com/Mentos05/SAS_DeepLearning/tree/master/Social%20Distancing%20Demo

Various transformations



Transformation	Preserves
Translation	Orientation
Euclidean (rigid)	Lengths
Affine	Parallelism
Homography (projective)	Straight lines

Reference: https://scikit-image.org/docs/stable/auto_examples/transform/plot_transform_types.html

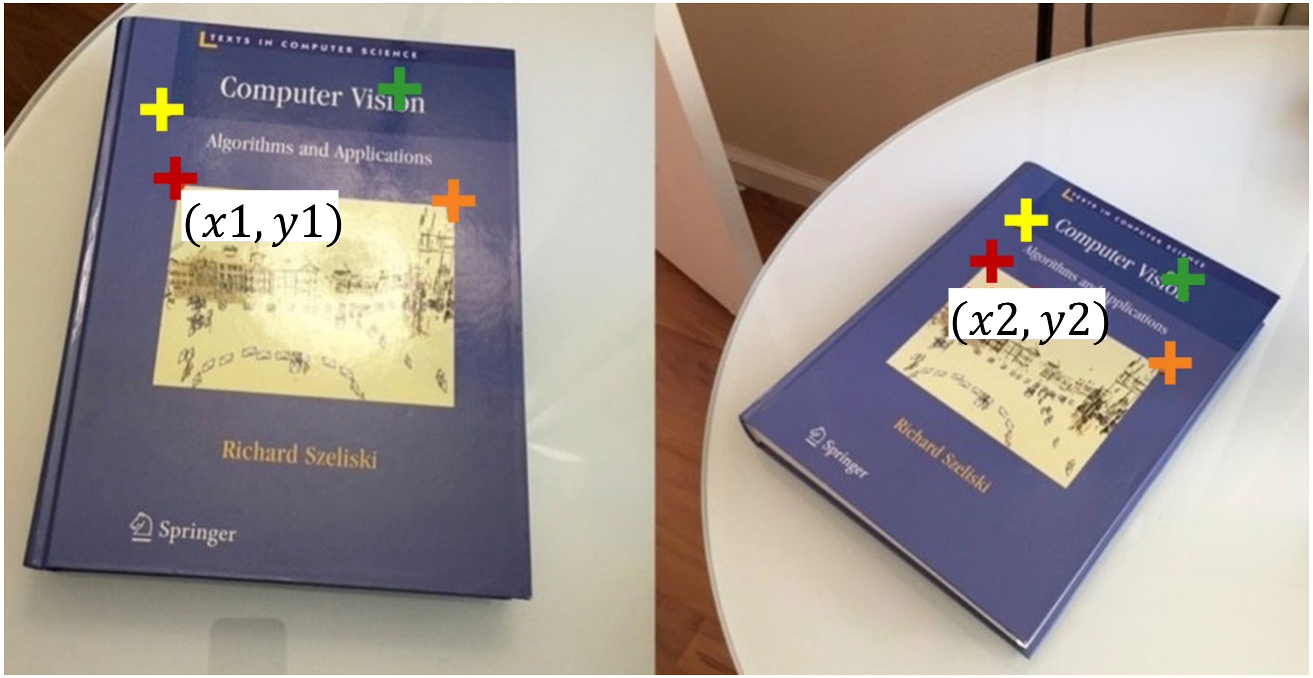
Homography matrix

- A Homography matrix is a transformation matrix (3×3) maps the point located at $(x1, y1)$ in one image to the corresponding point located at $(x2, y2)$ in the other image. It is true for ALL sets of corresponding points as long as they lie on the same plane in the real world.
- We need (at least) four pairs to solve eight unknown parameters in the Homography matrix.

cv2.findHomography()

$$\begin{pmatrix} x1 \\ y1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x2 \\ y2 \\ 1 \end{pmatrix}$$

homogeneous coordinates (see next few slides)



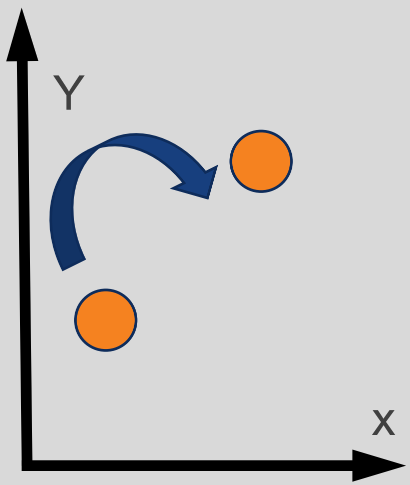
Reference: https://learnopencv.com/homography-examples-using-opencv-python-c/

Appendix: 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, $\mathbf{R}\mathbf{R}^T = \mathbf{I}$

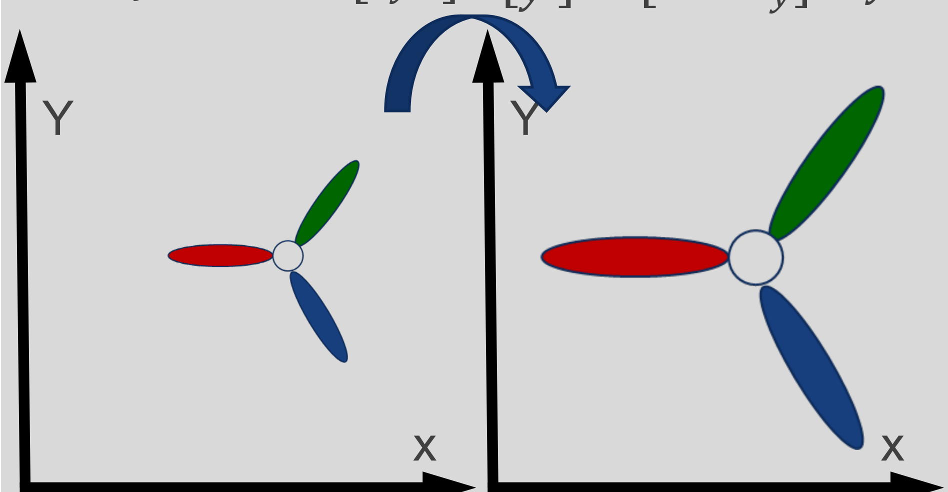
$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} \cos\theta\, x - \sin\theta\, y \\ \cos\theta\, y + \sin\theta\, x \end{bmatrix}$$

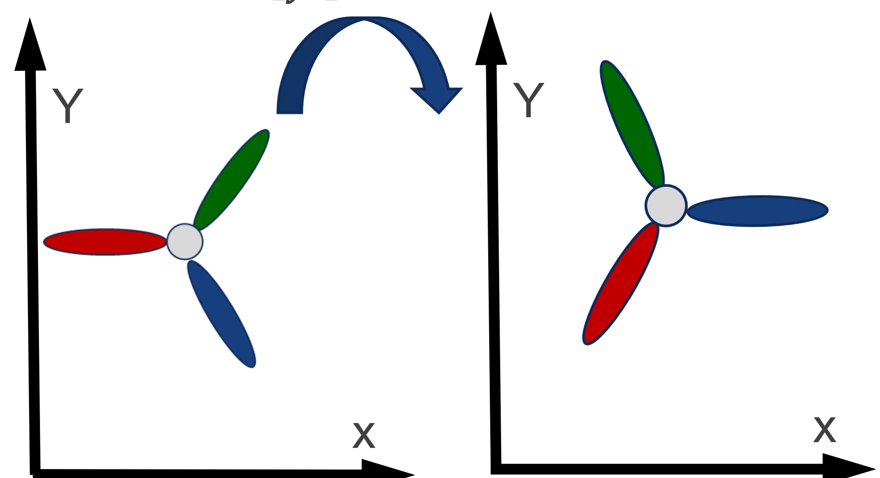
$$\mathbf{p}' = \mathbf{R} \cdot \mathbf{p}$$

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

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

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





Appendix: 2D transformations

- 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

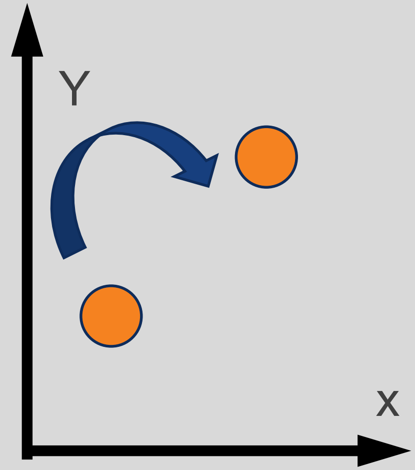
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Appendix: 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

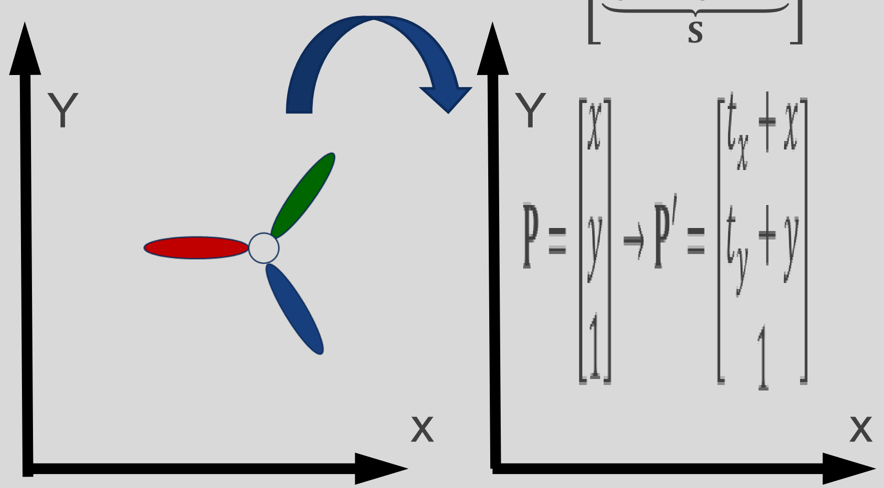
$$\mathbf{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} \cos\theta\, x - \sin\theta\, y \\ \cos\theta\, y + \sin\theta\, x \\ 1 \end{bmatrix}$$

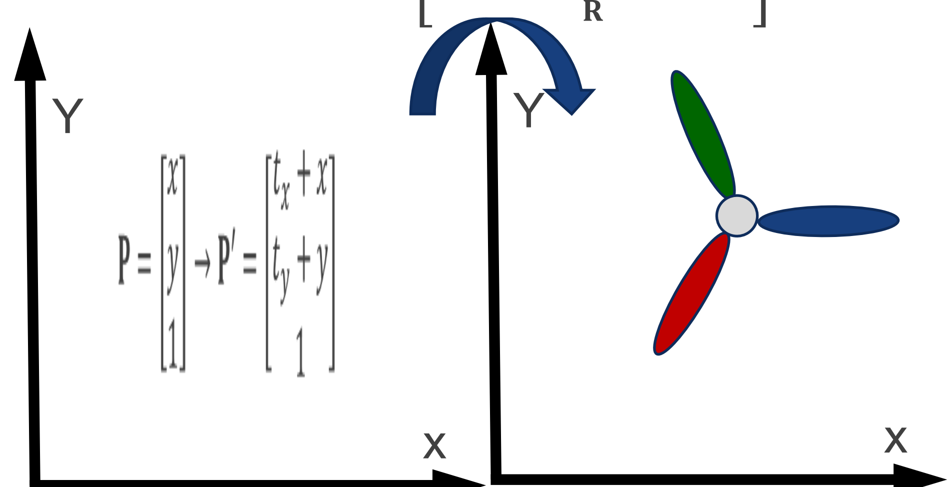
$$\mathbf{p}' = \mathbf{R} \cdot \mathbf{p}$$

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

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

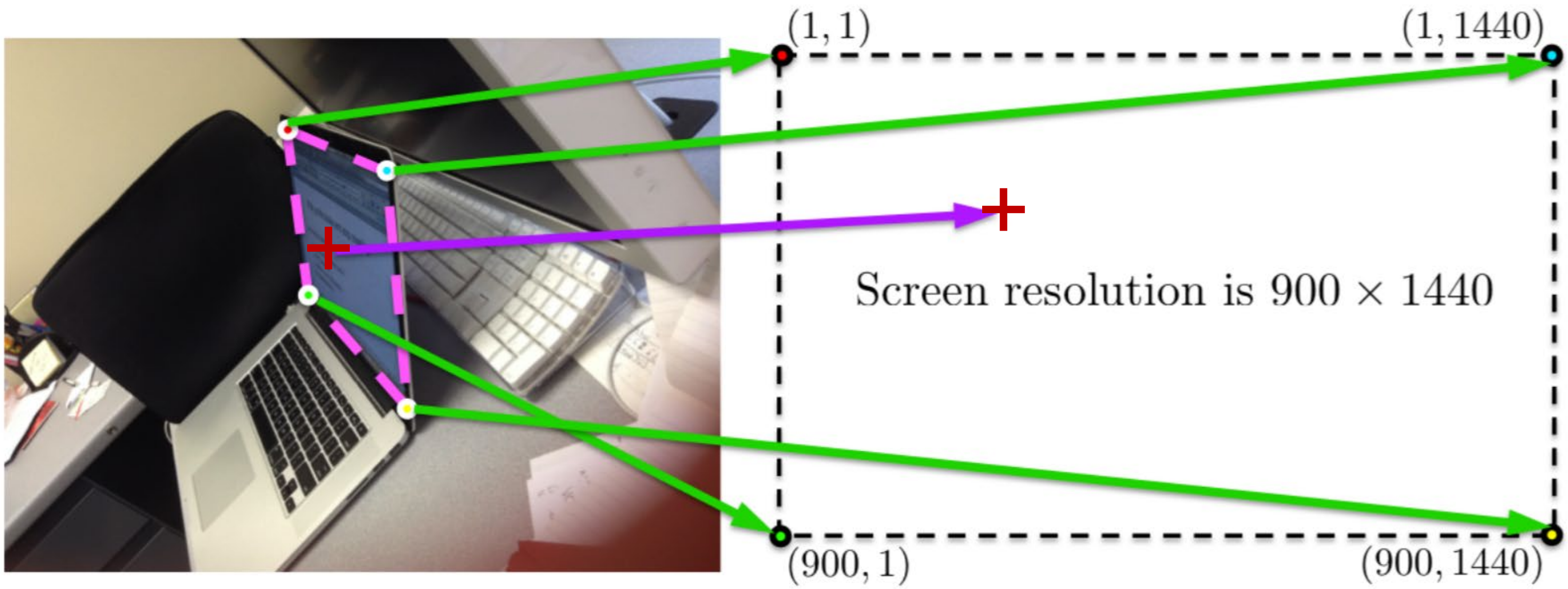
$$\mathbf{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \mathbf{P}' = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} \quad \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{matrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{matrix}}_{\mathbf{S}}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





Homography: Perspective correction

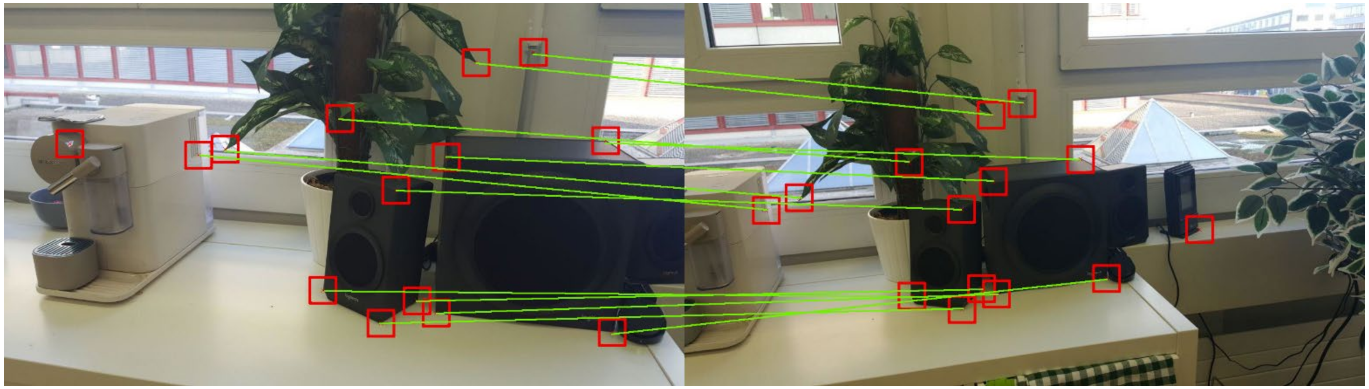
- Collect four corners of the object plane (e.g., laptop screen), called *pts_src*.
- Set the target plane (resolution and aspect ratio is decided by users based on domain knowledge), called *pts_dst*.
- Obtain the homography matrix using *pts_src* and *pts_dst* via OpenCV function *cv2.findHomography()* .
- Map the position (red cross '+') from the source image to the target plane to obtain its location.



Reference: http://www.cs.toronto.edu/~fidler/slides/2021Winter/CSC420/lecture9.pdf

Fundamental task: Keypoint matching in multiple-view images

Input data (multiple-view images)



Methods focused today (next module)

With matched pairs of keypoints in multiple-view images, we are able to

- Given a pair of matched keypoints (generated from the same physical point) in multiple-view images, we can estimate their disparity, then further estimate the distance between the physical point to the camera.
- Estimate the transformation (e.g., Homography matrix) between multiple-view images, which can help to estimate distance between two pixel positions in the image.

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

Thank you!

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