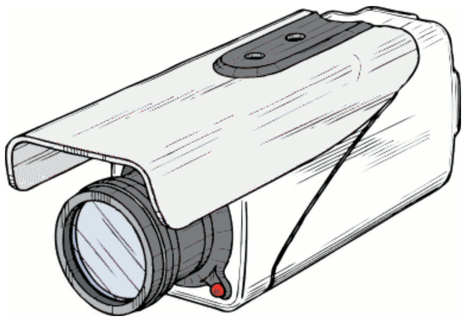
Univerza v Ljubljani



Machine perception
Camera geometry

Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Extracting 3D information from a 2D image?

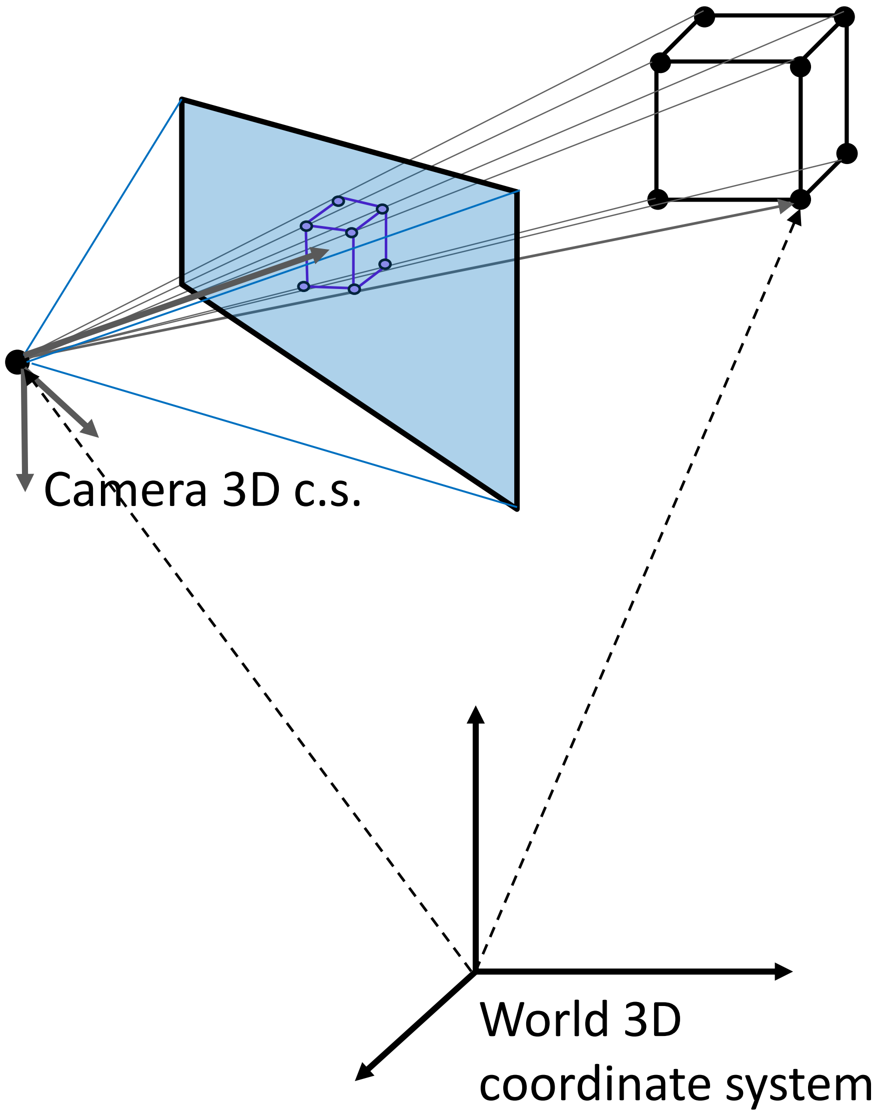
- Shading, Texture, Focus, Perspective, …
- Humans learn how 3D structure *looks* in a 2D image
- In computer vision, we require a model of 3D-to-2D transform to understand the 3D content

Single-view geometry

- Points in a world 3D coordinate system (c.s.)
- Project to image plane into 2D pixels

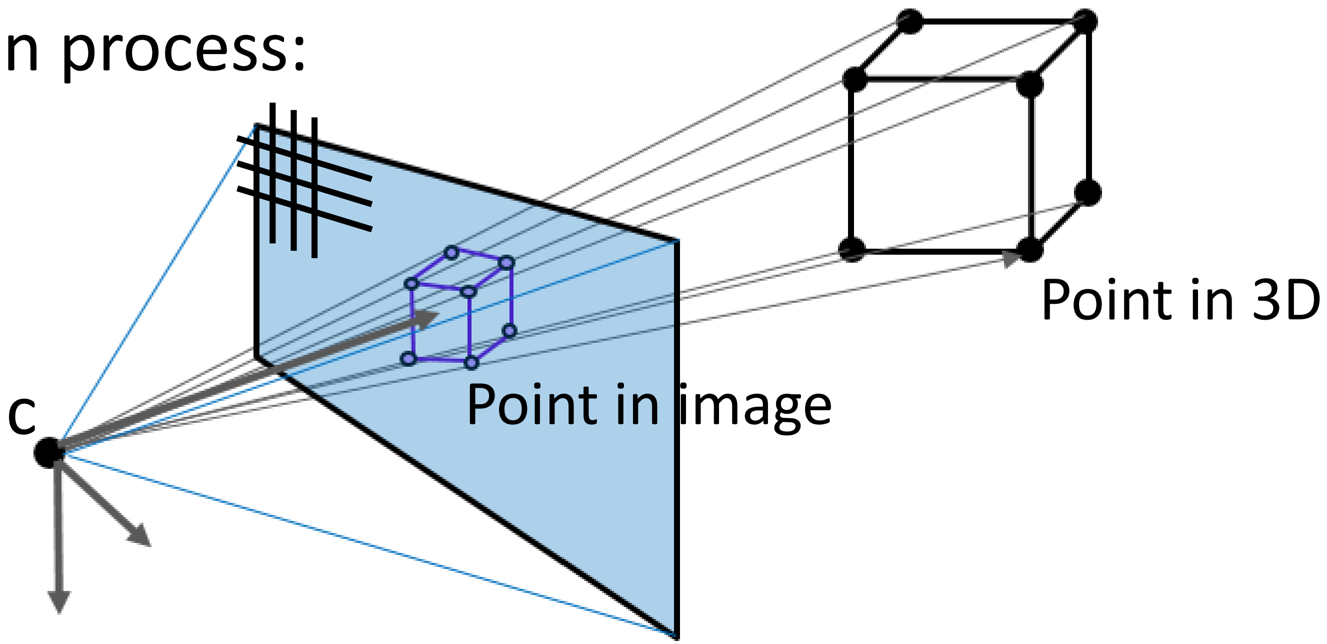
Two kinds of projection:

1. "Extrinsic" projection
   3D World → 3D Camera
2. "Intrinsic" projection
   3D Camera → 2D Image



Consider "Intrinsic" projection first

Recall the image formation process:



- A point written in camera 3D coordinate system (meters)
- Projected to camera image plane (meters)
- Projected to discretized image (pixels)
- Let's derive transformations for a pinhole camera!

Homogeneous coordinates

- Euclidean geometry uses Cartesian coordinate system
- But for a projective geometry, homogenous coordinates are much more appropriate
- E.g., can easily encode a point in infinity (try that in Euclidean...)

Cartesian form $\Rightarrow$ Homogeneous form

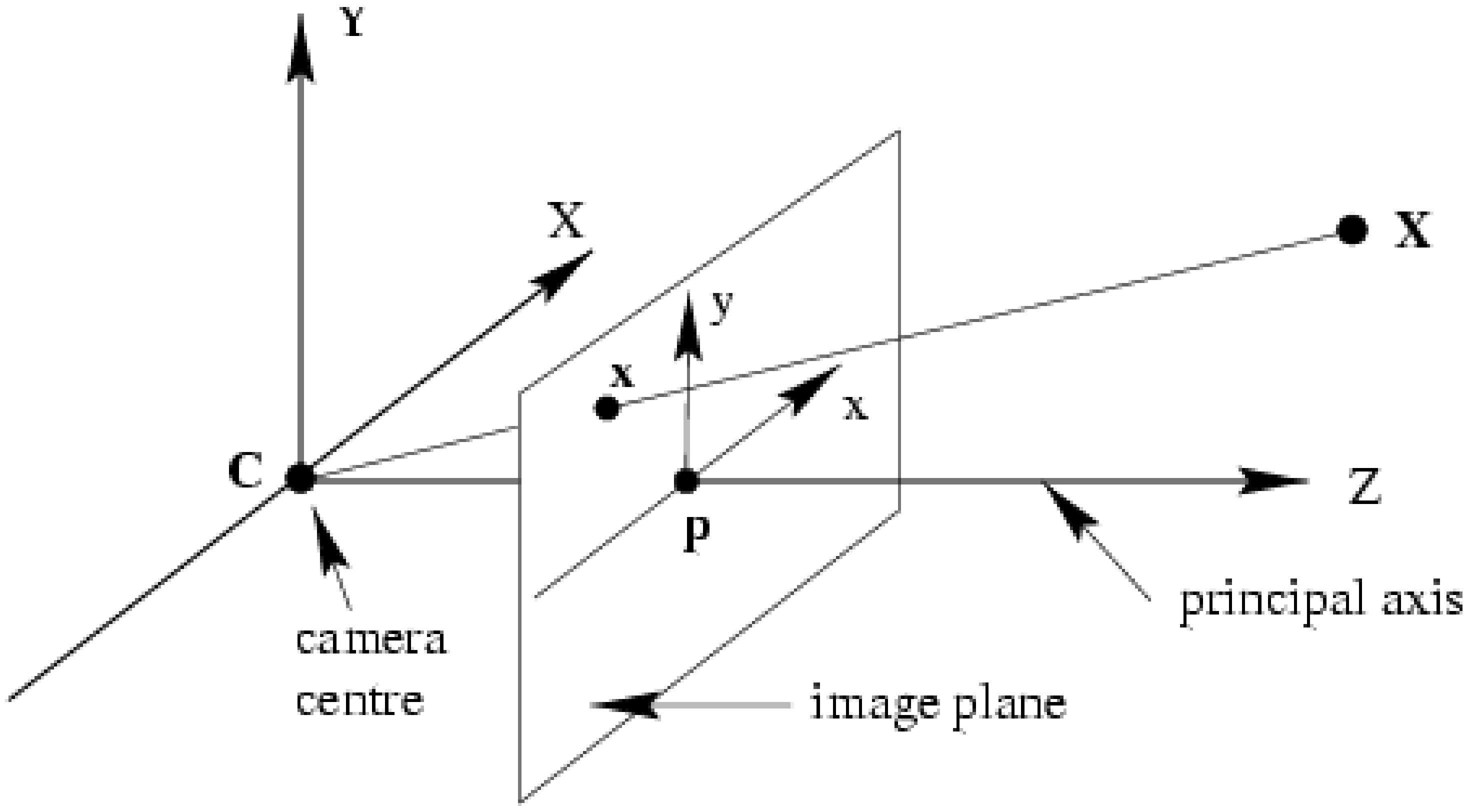
$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Multiplying by a scalar ($\neq 0$) value does not change a point!

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

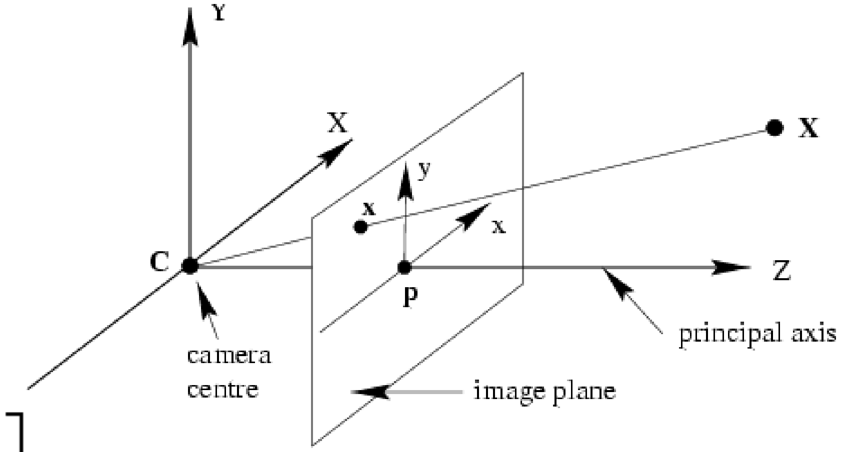
- From homogeneous system to Euclidian:
  Simply divide by the last coordinate to make it 1.

Camera coordinate system (meters)

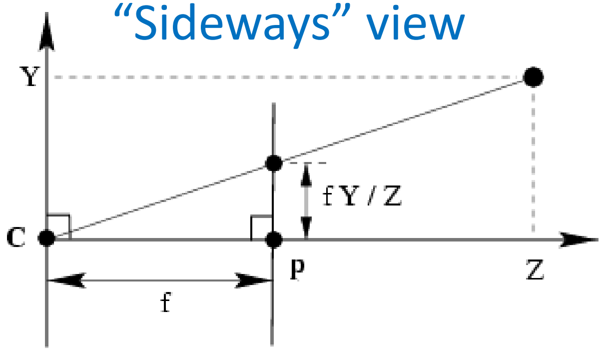


- *Principal axis*:
  A line from camera center perpendicular to image plane.
- *Principal point* ($\boldsymbol{p}$): A point where the principal axis punctures the image plane.
- *Normalized (camera) coordinate system*: 2D system with origin at the *principal point*.

A pinhole camera revisited



"Sideways" view



3D point in world c.s. — 2D projection to image plane

$$[X, Y, Z]^T \mapsto [f\,X / Z,\, f\,Y / Z]^T$$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \leftarrow \text{Rewrite in homogeneous coordinates} \rightarrow \begin{bmatrix} fX/Z \\ fY/Z \\ 1 \end{bmatrix}$$

- Projection as vector-matrix multiplication:

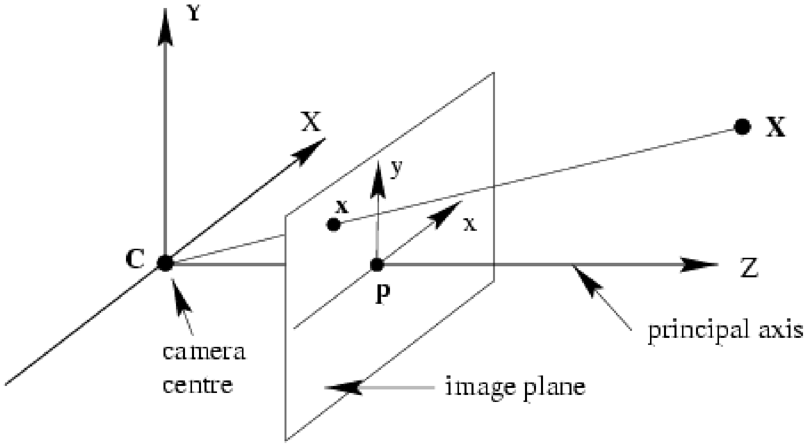
$$\mathrm{x} = \mathrm{P}_0 \mathrm{X}$$

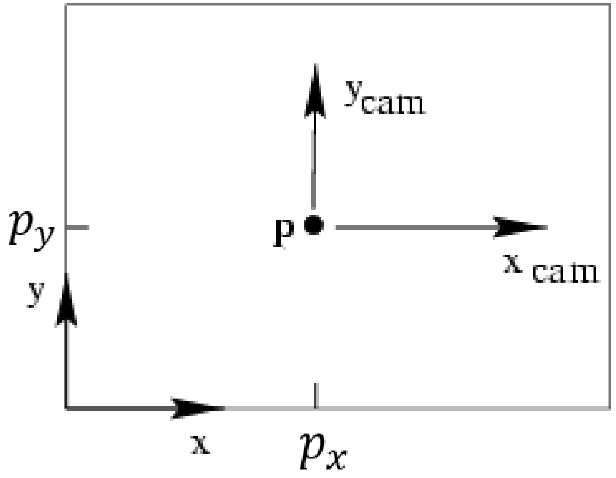
In 3D camera c.s. (X); In 2D image plane c.s. (x)

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix}}_{\boldsymbol{P}_0} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

From image plane to image pixels

- Change of coordinate system to image corner





- Normalized camera coordinate system:

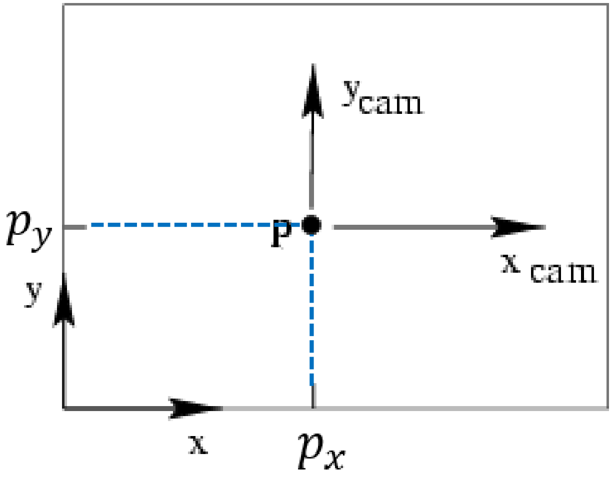
  Origin in principal point $\boldsymbol{p} = [\boldsymbol{p}_x, \boldsymbol{p}_y]^T$.

- Image coordinate system:

  Origin in the corner of the image sensor.

From image plane to image pixels (1/3)

- Change the c.s. origin by the principal point $\boldsymbol{p}$:



- Write the transformation:

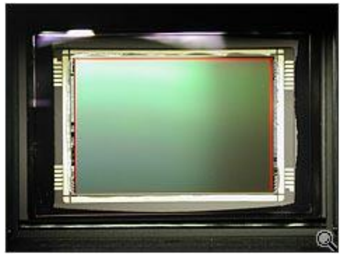
$$(X, Y, Z) \mapsto (f\,X / Z + p_x, f\,Y / Z + p_y)$$

- Rewrite in vector-matrix multiplication:

$$\begin{pmatrix} f\,X + Z\,p_x \\ f\,Y + Z\,p_y \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{P}_0} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \mathrm{x} = \mathrm{P}_0 \mathrm{X}$$

From image plane to image pixels (2/3)

- Projection to a sensor of size $W_S \times H_S$ (in meters).
- Pixels are arranged into a *rectangular* $M_x \times M_y$ pixels matrix.



- Let $m_x = M_x / W_S$ and $m_y = M_y / H_S$.
- Construct projection to pixels:

Just multiply by another matrix:

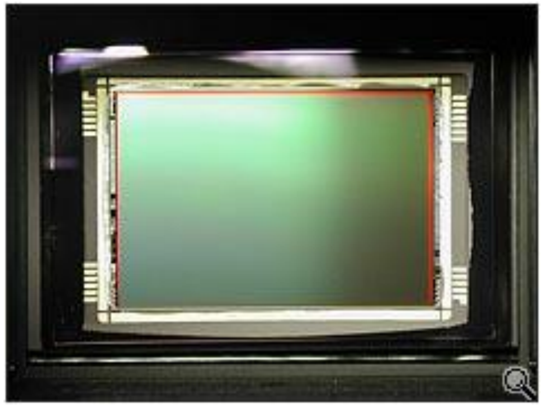
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{pixel/m}} \underbrace{\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{m}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

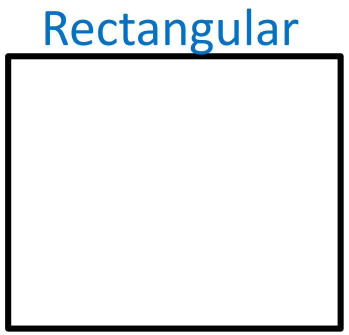
Abbreviated form:

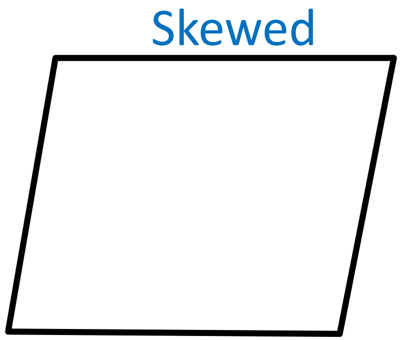
$$\Rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

From image plane to image pixels (3/3)

- In general difficult to guarantee a rectangular sensor.







Rectangular

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Projection matrix P_0

Skewed

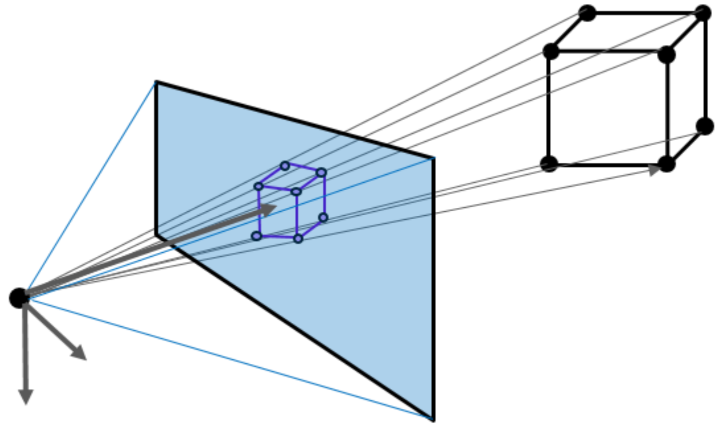
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Projection matrix P_0

Calibration matrix

- Expand the projection matrix $\boldsymbol{P}_0$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$



$$\mathrm{P}_0 = \mathrm{K}\left[\mathrm{I} \mid 0\right]$$

- Calibration matrix $\boldsymbol{K}$:

"Prescribes projection of 3D point in camera c.s. into pixles!"

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

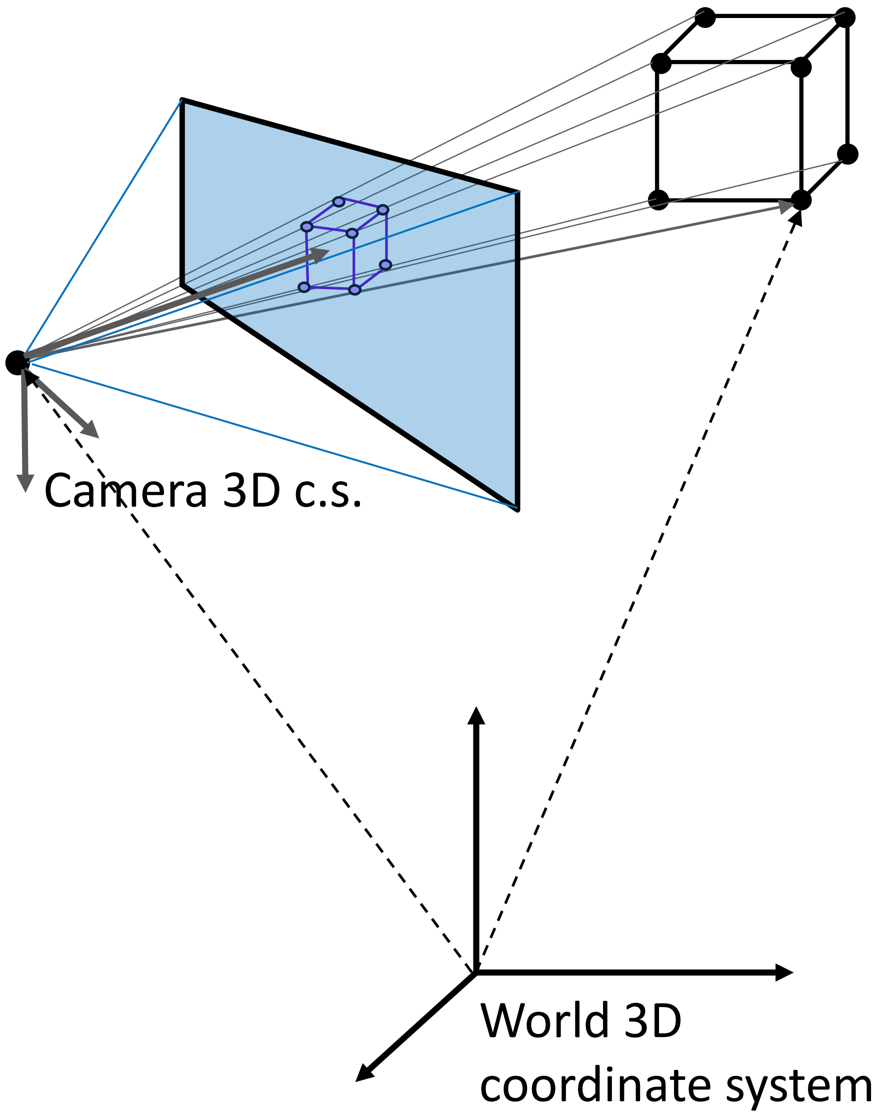
Q: What is the meaning of each element of the calibration matrix?

Single-view geometry

- Points in a world 3D coordinate system (c.s.)
- Project to image plane into 2D pixels

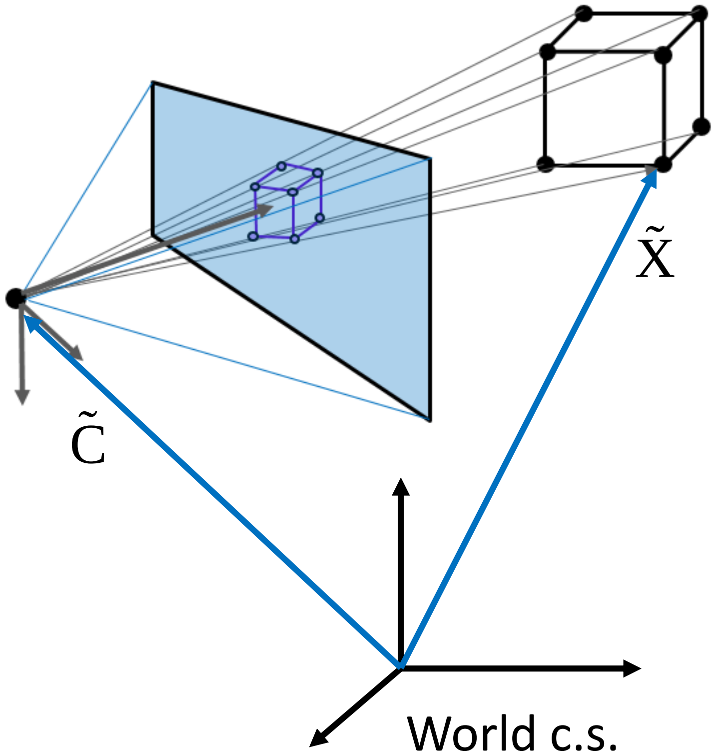
Two kinds of projection:

1. “Extrinsic” projection
   3D World → 3D Camera ?
2. “Intrinsic” projection
   3D Camera → 2D Image ✓



From world c.s. to 3D camera c.s.

- The 3D camera coordinate system (c.s.) is related to 3D world c.s. by a rotation matrix $\boldsymbol{R}$ and translation $\tilde{\boldsymbol{t}} = \tilde{\boldsymbol{C}}$.



$\boldsymbol{R}$ ... How to rotate the world c.s. about its own origin to align it with the camera c.s.

$\tilde{\boldsymbol{C}}$ ... Camera origin in world c.s.

$\tilde{\boldsymbol{X}}$ ... Point in 3D world c.s.

$\tilde{\boldsymbol{X}}_{cam}$ ... Same point $\tilde{\boldsymbol{X}}$, but written in 3D camera c.s.

World-to-camera c.s. transformation:

$$\tilde{X}_{cam} = R\left(\tilde{X} - \tilde{C}\right)$$

(Euclidean)

From world c.s. to 3D camera c.s.

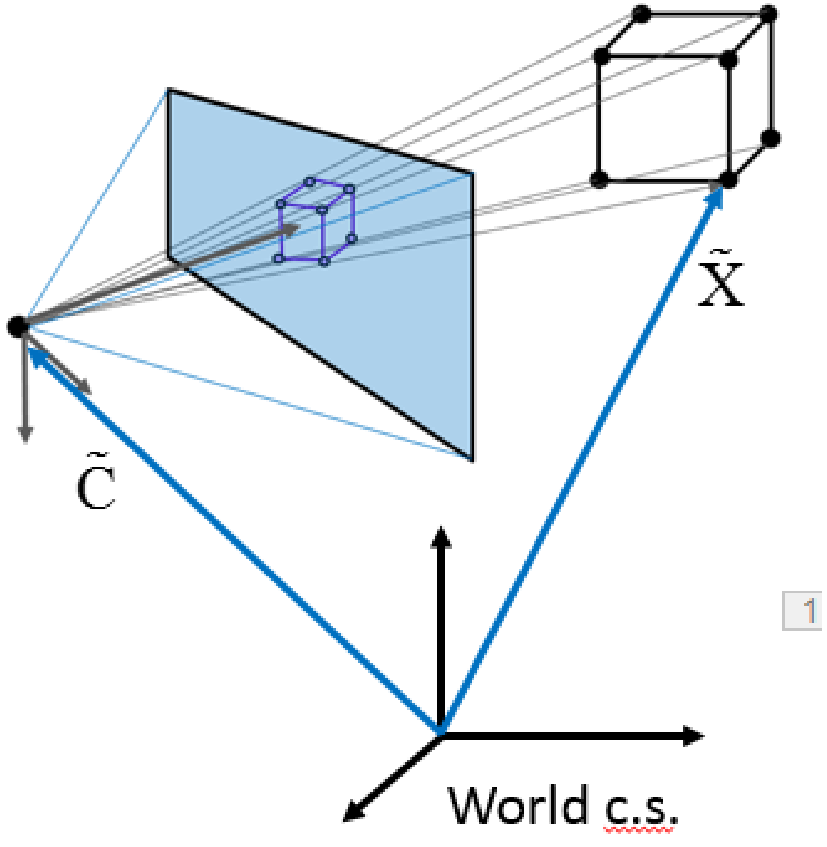
- Euclidean form:

$$\tilde{X}_{cam} = R\left(\tilde{X} - \tilde{C}\right)$$

- Rewrite by using homogeneous coordinates:

$$X_{cam} = \begin{bmatrix} \tilde{X}_{cam} \\ 1 \end{bmatrix} \qquad X = \begin{bmatrix} \tilde{X} \\ 1 \end{bmatrix}$$

$$X_{cam} = \begin{pmatrix} \tilde{X}_{cam} \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \tilde{X} \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$



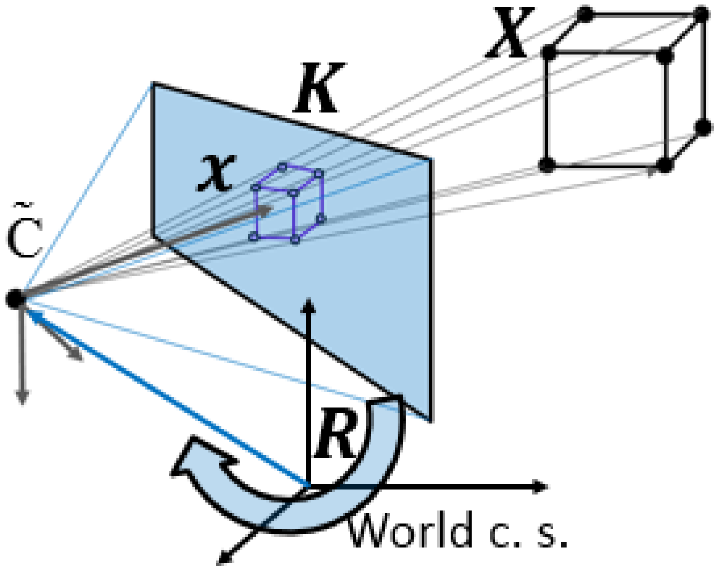
Putting it all together

- Camera is specified by a calibration matrix $\boldsymbol{K}$, the projection center in world c.s. $\tilde{\boldsymbol{C}}$ and rotation matrix $\boldsymbol{R}$.
- A 3D point in world coordinates (homogeneous) $\boldsymbol{X}$, is projected into pixels $\boldsymbol{x}$ by the following relation:

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$

$$x = K[I|0]X_{cam} = K\left[R \,|\, -R\tilde{C}\right]X = PX$$

$$P = K[R|t], \qquad t = -R\tilde{C}$$



Note the structure of the projection matrix!

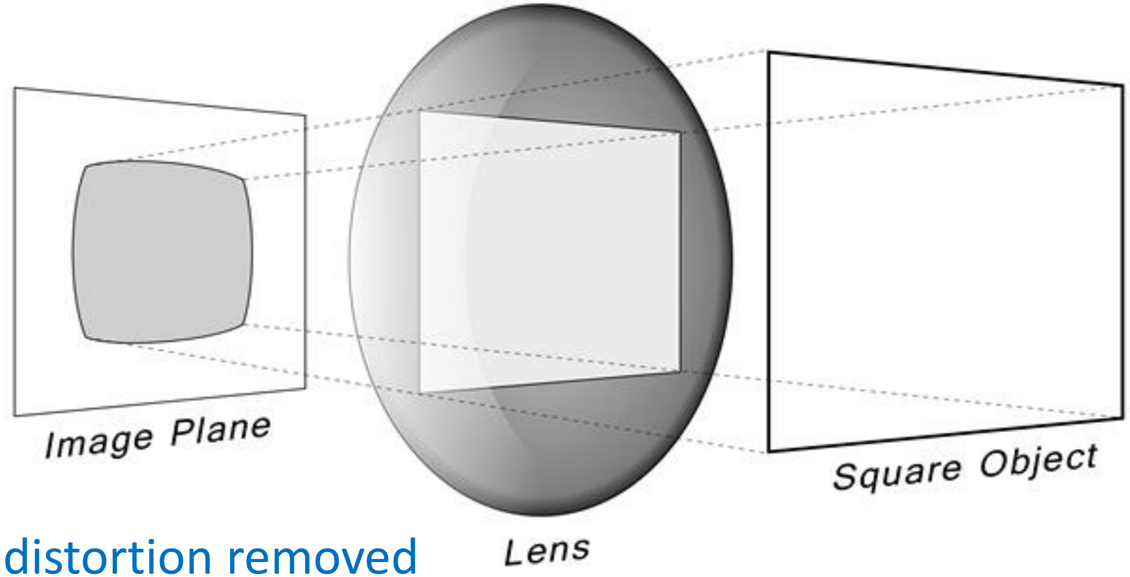
Q: What needs to be known to construct the projection matrix?

Lens adds a nasty nonlinearity

Straight lines are no longer straight!

Nonlinearity *should be removed*

to apply a pinhole model!



radially distorted image

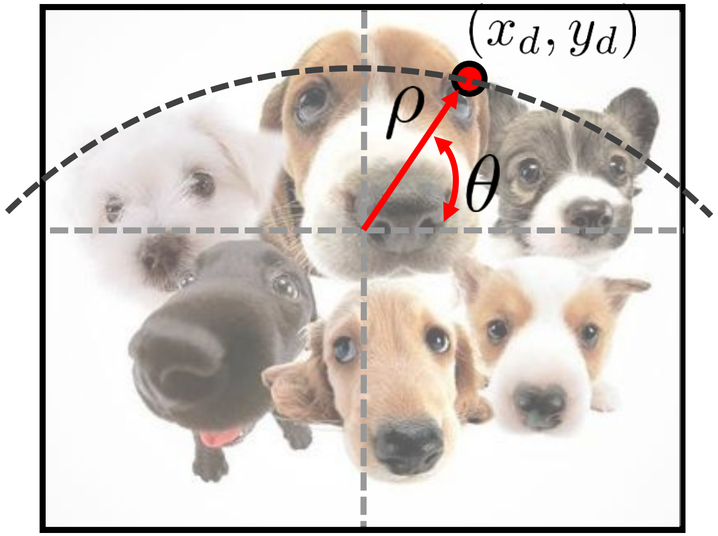
radial distortion removed

http://vipbase.net/doc/img_trans_radial.htm

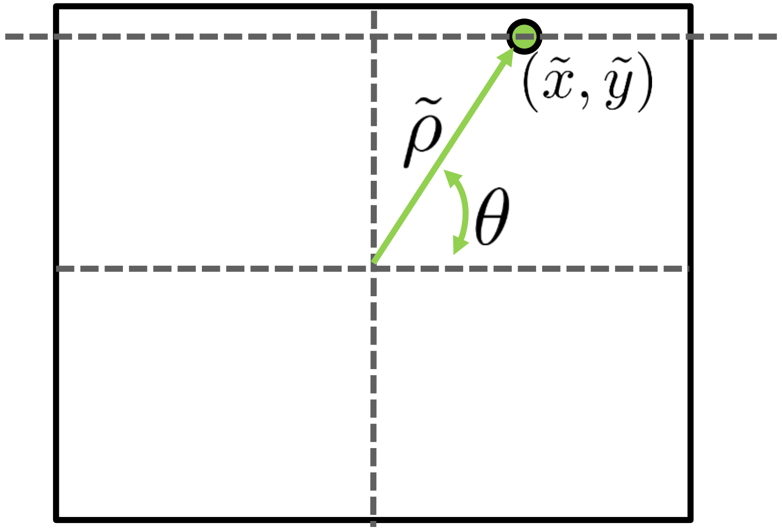
Lens adds a nasty nonlinearity

- Lens distortion assumed radially symmetric
- Radially expand an image to un-distort

distorted



undistorted

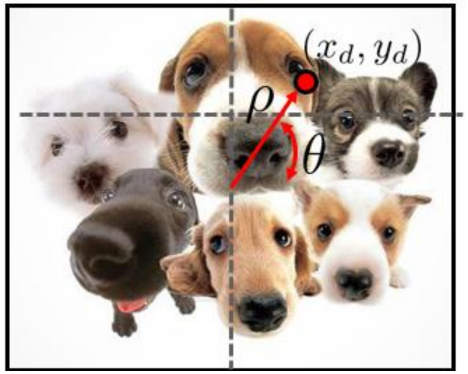


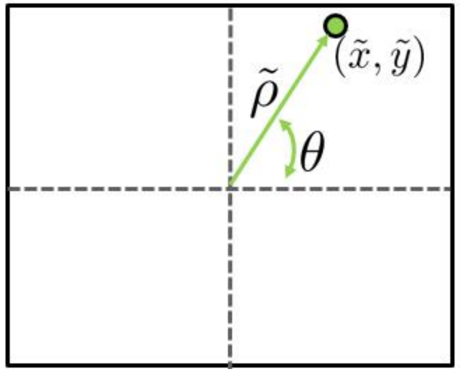
$$(\rho, \theta) \rightarrow (\tilde{\rho}, \theta)$$

- In this transformation, only the radius of transformed point changes, but the angle remains unchanged.

Lens adds a nasty nonlinearity

- What kind of analytic function to use for transforming ρ?





- Typically, a polynomial is used (3rd degree good enough):

$$\tilde{x} = x_d + (x_d - c_x)(K_1\rho^2 + K_2\rho^4 + \dots)$$
$$\tilde{y} = y_d + (y_d - c_y)(K_1\rho^2 + K_2\rho^4 + \dots)$$

- Parameters estimated by adjusting them until straight lines become straight.

  (in Matlab use fminsearch for optimization method)

Summary: camera parameters

Degrees of freedom (DoF)

- Intrinsic parameters: **DoF**
  - Principal point coordinates — 2
  - Focal length — 1
  - Pixel scaling factor (rectangular pixels) — 1
  - *Shear (non-rectangular pixels)* — 1
  - *Radial distortion*

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

- Extrinsic parameters
  - Rotation R — 3
  - Translation t — 3
- Camera projection matrix $\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathrm{t}]$

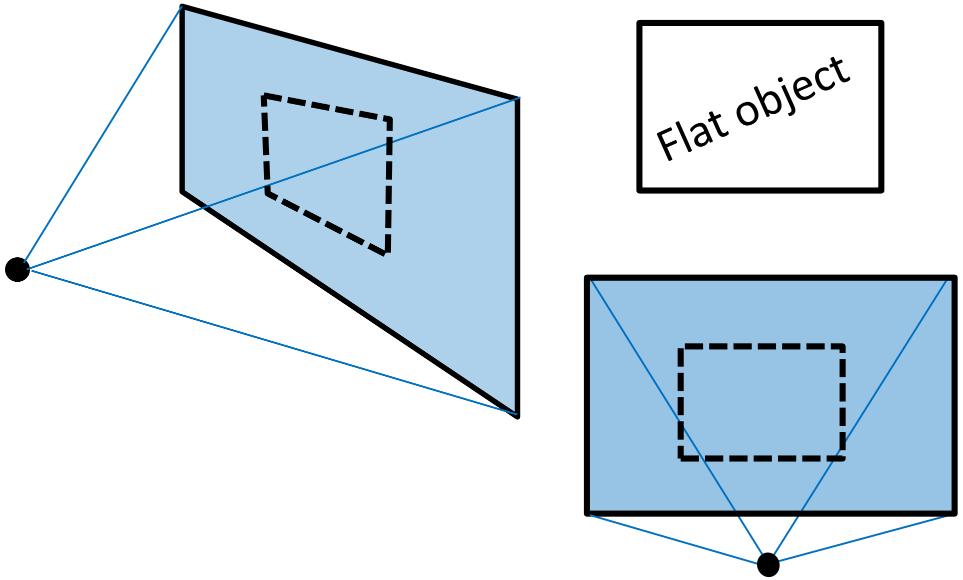
$\Rightarrow$ A pinhole camera: 9 DoF

$\Rightarrow$ Camera with rectangular pixels: 10 DoF

$\Rightarrow$ General camera (skewed pixes): 11 DoF

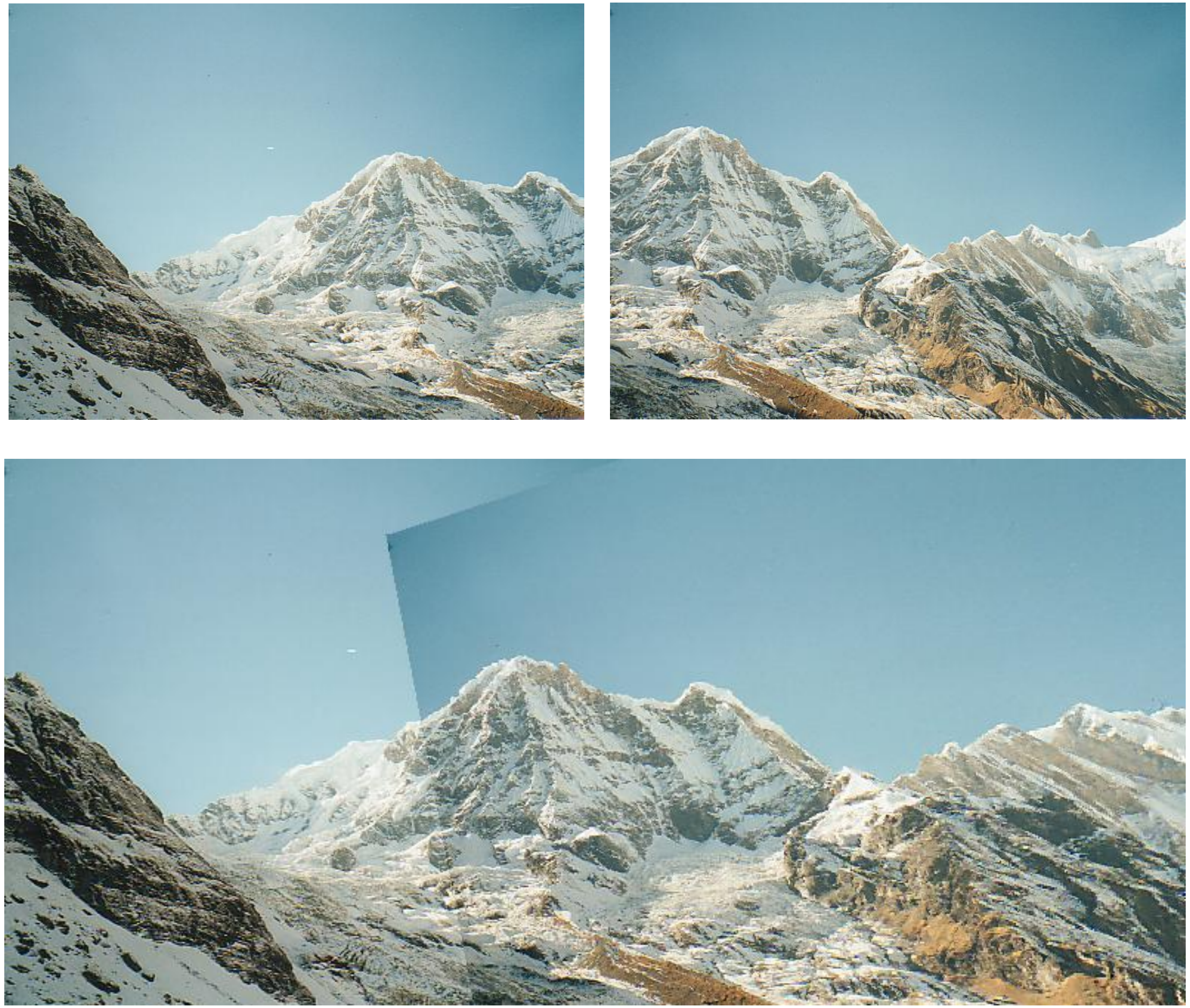
Looking at flat objects

- A camera looking at the some planar object
- How would it look if the camera changed position?



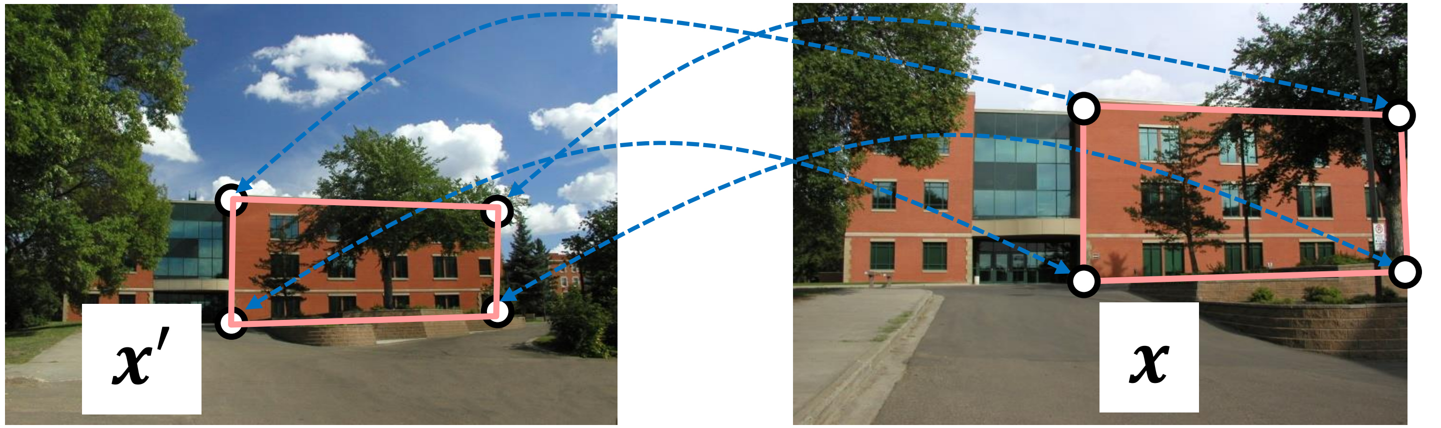
- A plane-to-plane projection is called a *Homography*

Apps: Panoramas, Augmented reality, etc.



Homography estimation from correspondences

- Example of four corresponding points



$$w\mathbf{x}' = \mathbf{H}\mathbf{x}$$

$$w\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- The elements of $\mathbf{H}$ can be estimated by applying a direct linear transform (DLT)!

Matrix form of a vector product

- Before we continue…

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

$$\mathbf{c}^T \mathbf{a} = 0$$

$$\mathbf{c}^T \mathbf{b} = 0$$

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

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}_\times] \mathbf{b}$$

Homography estimation by DLT

$$w\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$$

$$w\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \\ \boldsymbol{h}_3^T \end{bmatrix}\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{h}_1^T \boldsymbol{x}_i \\ \boldsymbol{h}_2^T \boldsymbol{x}_i \\ \boldsymbol{h}_3^T \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_i^T \boldsymbol{h}_1 \\ \boldsymbol{x}_i^T \boldsymbol{h}_2 \\ \boldsymbol{x}_i^T \boldsymbol{h}_3 \end{bmatrix}$$

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = 0$$

Change the vector product into vector-matrix:

$$\mathbf{x}'_i \times \begin{bmatrix} \mathbf{x}_i^T \mathbf{h}_1 \\ \mathbf{x}_i^T \mathbf{h}_2 \\ \mathbf{x}_i^T \mathbf{h}_3 \end{bmatrix} = [\mathbf{x}'_{i\times}]\begin{bmatrix} \mathbf{x}_i^T \mathbf{h}_1 \\ \mathbf{x}_i^T \mathbf{h}_2 \\ \mathbf{x}_i^T \mathbf{h}_3 \end{bmatrix} = \begin{bmatrix} 0 & -1 & y'_i \\ 1 & 0 & -x'_i \\ -y'_i & x'_i & 0 \end{bmatrix}\begin{bmatrix} \mathbf{x}_i^T \mathbf{h}_1 \\ \mathbf{x}_i^T \mathbf{h}_2 \\ \mathbf{x}_i^T \mathbf{h}_3 \end{bmatrix}$$

Homography estimation by DLT

$$\mathrm{x}'_i \times \mathrm{H}\mathrm{x}_i = 0$$

Multiply in the matrix terms…

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{bmatrix} 0 & -1 & y'_i \\ 1 & 0 & -x'_i \\ -y'_i & x'_i & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_i^T \mathbf{h}_1 \\ \mathbf{x}_i^T \mathbf{h}_2 \\ \mathbf{x}_i^T \mathbf{h}_3 \end{bmatrix} = \begin{bmatrix} -\mathbf{x}_i^T \mathbf{h}_2 + y'_i \mathbf{x}_i^T \mathbf{h}_3 \\ \mathbf{x}_i^T \mathbf{h}_1 - x'_i \mathbf{x}_i^T \mathbf{h}_3 \\ -y'_i \mathbf{x}_i^T \mathbf{h}_1 + x'_i \mathbf{x}_i^T \mathbf{h}_2 \end{bmatrix}$$

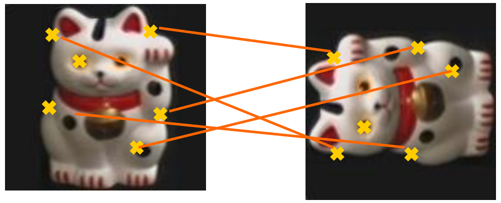
Expose the homography terms $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ into a single vector:

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{bmatrix} 0^T & -\mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = 0$$

A single point contains three coordinates, but gives only two linearly independent equations

Homography estimation by DLT

n Correspondences...



$$\mathbf{x}'_1 \leftrightarrow \mathbf{x}_1$$
$$\mathbf{x}'_2 \leftrightarrow \mathbf{x}_2$$
$$\vdots$$

The *n* points yields a system of equations:

$$\begin{bmatrix} 0^T & -\mathbf{x}'^T_1 & y'_1\,\mathbf{x}^T_1 \\ \mathbf{x}^T_1 & 0^T & -x'_1\,\mathbf{x}^T_1 \\ \cdots & \cdots & \cdots \\ 0^T & -\mathbf{x}^T_n & y'_n\,\mathbf{x}^T_n \\ \mathbf{x}^T_n & 0^T & -x'_n\,\mathbf{x}^T_n \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

Homogeneous system!

$$\mathbf{Ah} = \mathbf{0}$$

SVD

$$\mathbf{A} = \mathbf{UDV}^T = \mathbf{U} \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T$$

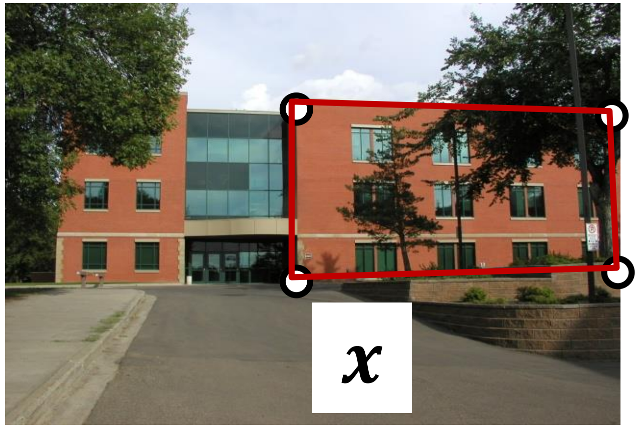
$$\mathbf{h} = \frac{[v_{19}, \cdots, v_{99}]}{v_{99}}$$

Minimizes the mean squared error.

Reshape **h** into **H**.

Preconditioning

- DLT works well if the corresponding points are normalized separately in each view!
- Transformation T_{pre}:
  - Subtract the average
  - Scale to average distance 1.



$$\tilde{x} = T_{pre} x$$

$$T_{pre} = \begin{bmatrix} a & 0 & c \\ 0 & b & d \\ 0 & 0 & 1 \end{bmatrix}$$

  - Set [a,b,c,d] such that the mean of the points $\tilde{x}_i$ is zero and their variance is 1.

Homography estimation

1. Apply preconditioning (i.e., multiply by the transform matrices) to points in each image separately:

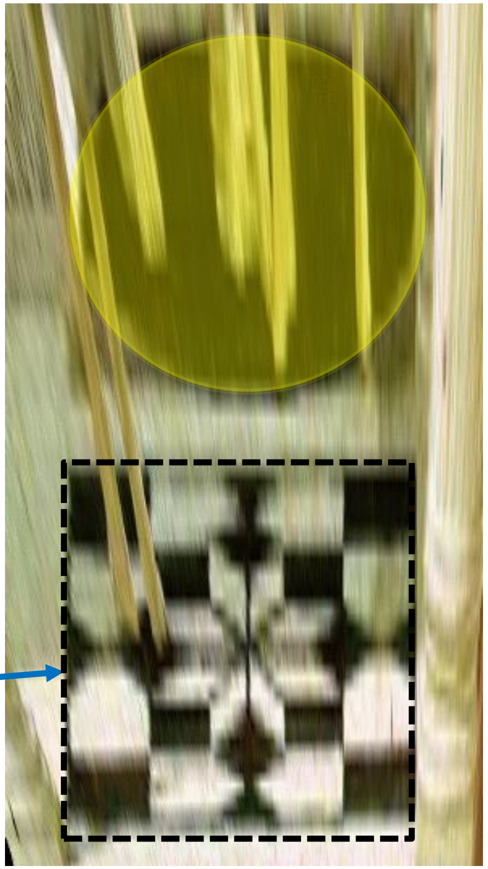
$$\tilde{\mathbf{x}}' = \mathbf{T}'_{pre}\mathbf{x}' \qquad \tilde{\mathbf{x}} = \mathbf{T}_{pre}\mathbf{x}$$

2. Apply DLT to estimate the homography $\tilde{H}$: $\tilde{\mathbf{x}}' = \tilde{H}\tilde{\mathbf{x}}$
3. Transform back the solution to remove preconditioning: $\mathbf{H} = \mathbf{T}'^{-1}_{pre}\tilde{\mathbf{H}}\mathbf{T}_{pre}$

Secret knowledge

Flagellation of Christ (Piero della Francesca, ~1460)

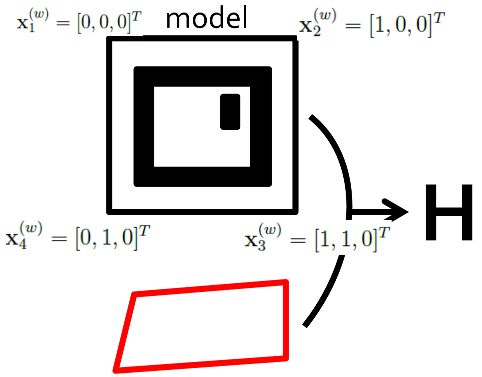
Compute a homography to this rectangle

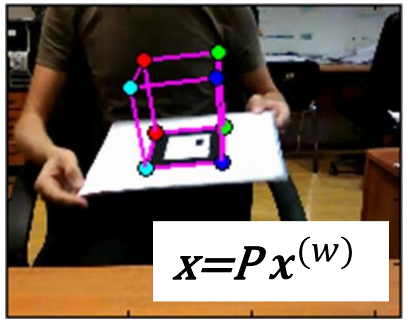


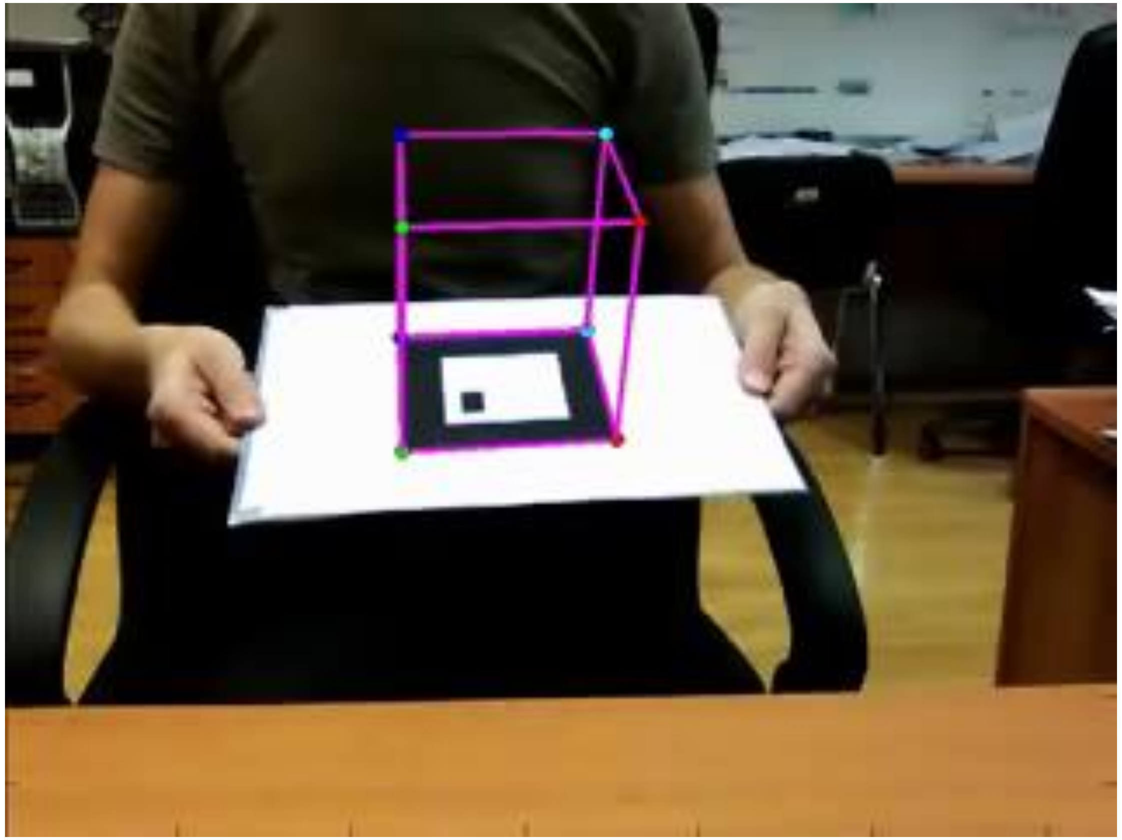
Zoom-in of the floor

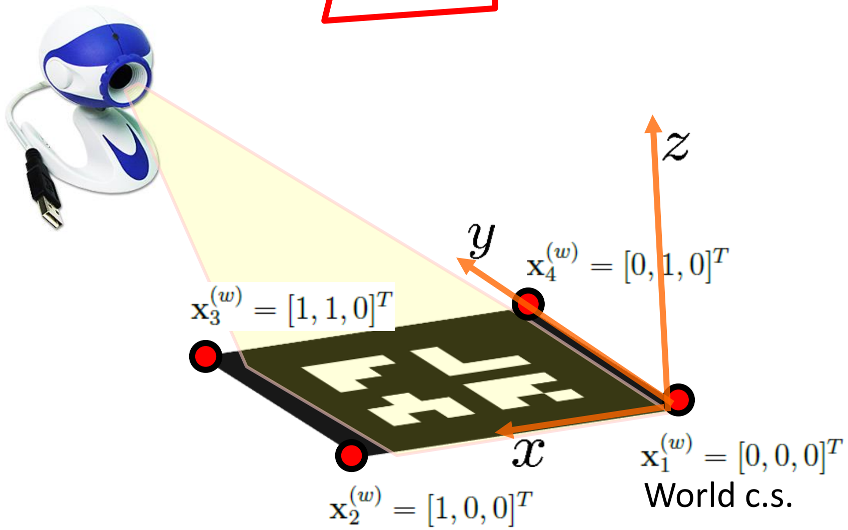
Check out : [Secret Knowledge] by David Hockney, 2002

Marker-based Augmented Reality









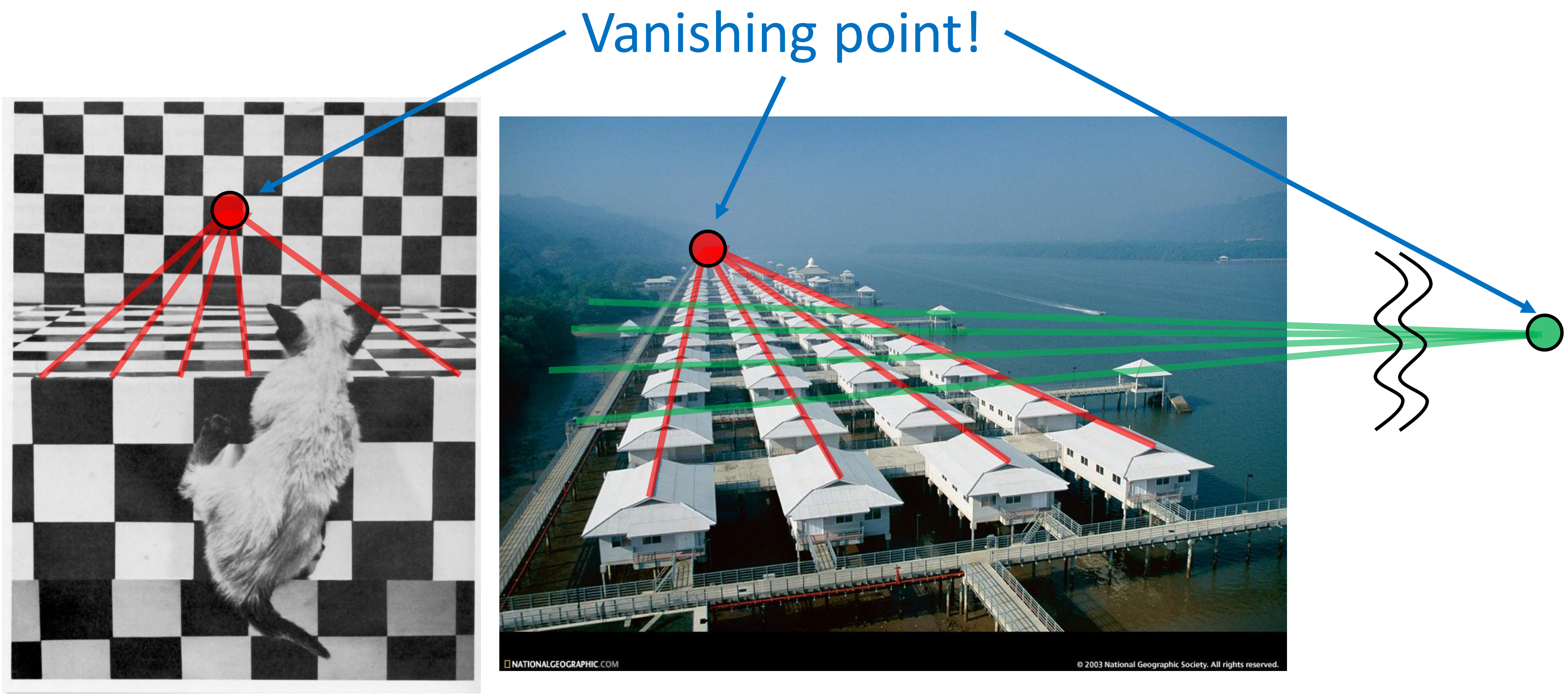
$$\boldsymbol{x} = \boldsymbol{H}\boldsymbol{x}^{(w)}$$

$$\boldsymbol{x} = \boldsymbol{K}[\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t}]\boldsymbol{x}^{(w)}$$

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{t}]$$

Vanishing points

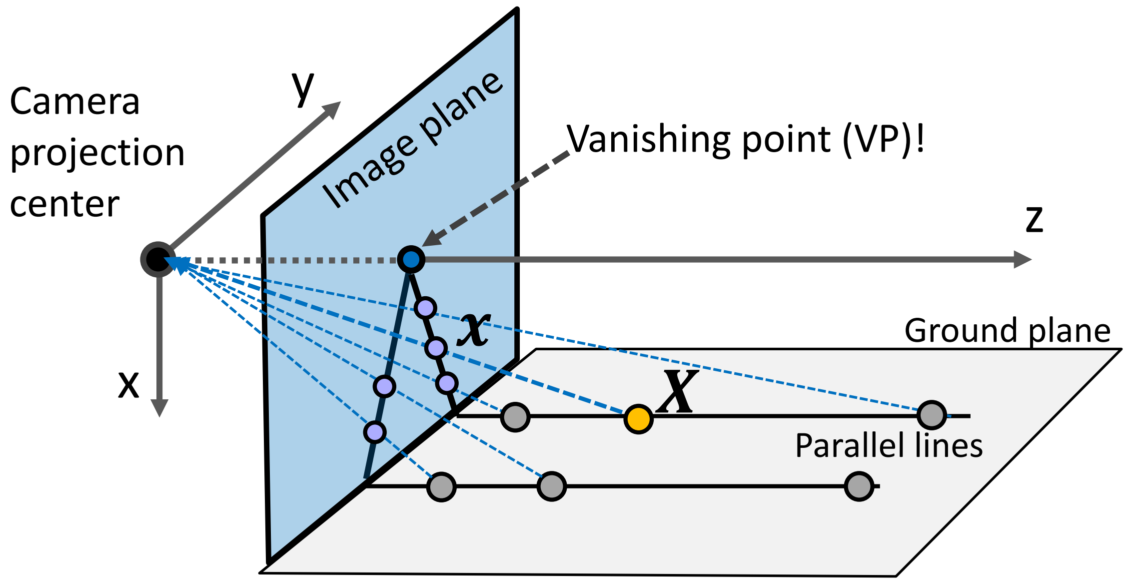
- What happens with projection of parallel lines?



- Sets of 3D parallel lines intersect at a vanishing point!

Vanishing points

- Where in image do sets of 3D parallel lines *projections* intersect?



A 3D point:

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

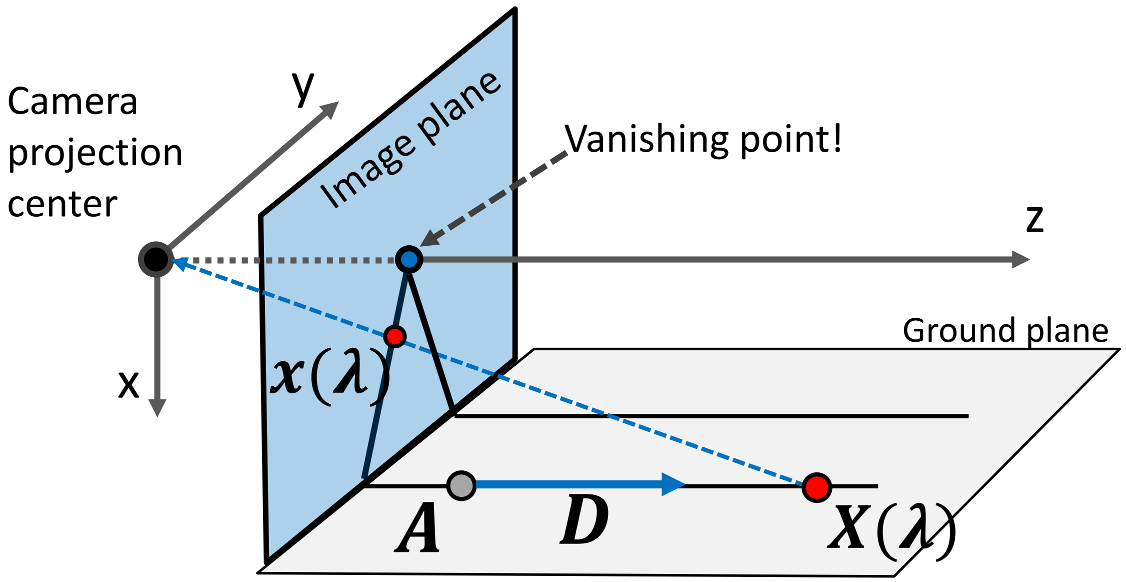
Perspective projection:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \end{bmatrix}$$

- Note that this image shows a special case with lines parallel with principal axis.
- But our derivation of VP will be general.

Vanishing point: calculation (1/2)

- Consider a point on one of parallel lines



A 3D point **A** and vector **D**:

$$\mathbf{A} = \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} \quad D = \begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix}$$

A point on a line:

$$\mathbf{X}(\lambda) = \mathbf{A} + \lambda \mathbf{D}$$

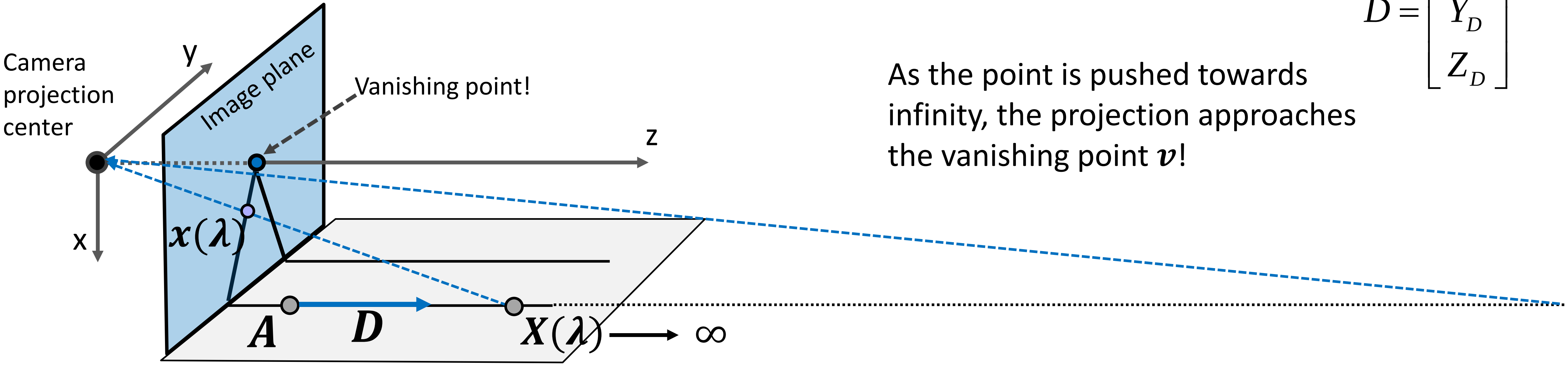
Perspective projection:

$$\mathbf{x}(\lambda) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \end{bmatrix} = \begin{bmatrix} \frac{f(X_A+\lambda X_D)}{(Z_A+\lambda Z_D)} \\ \frac{f(Y_A+\lambda Y_D)}{(Z_A+\lambda Z_D)} \end{bmatrix}$$

Vanishing point: calculation (2/2)

- Now push the point far away from the camera…

$$D = \begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix}$$



As the point is pushed towards infinity, the projection approaches the vanishing point $\boldsymbol{v}$!

Projection of a point at infinity, i.e., $\boldsymbol{X}(\infty)$:

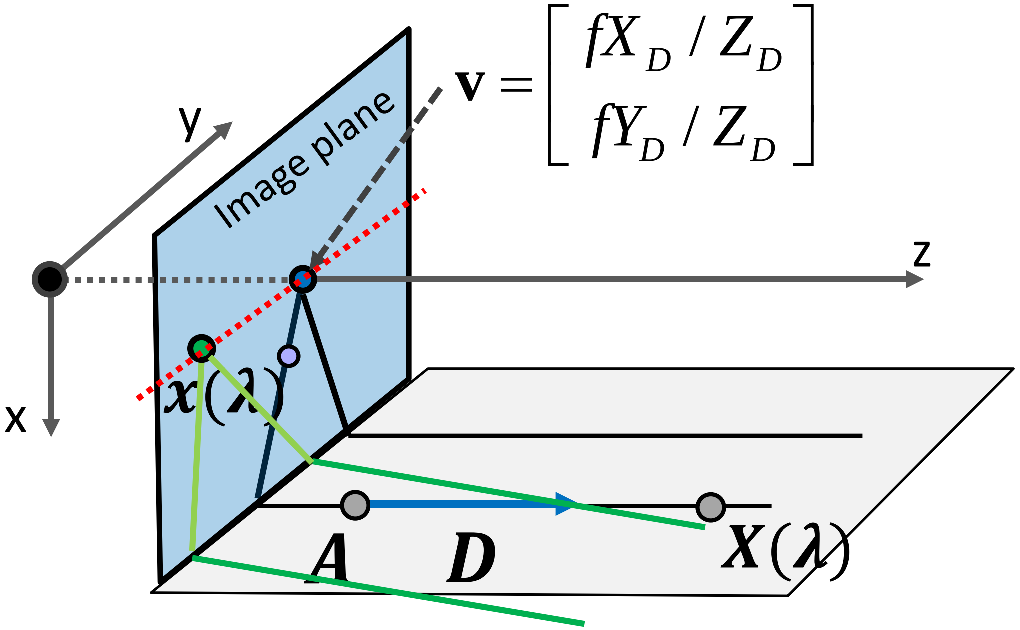
$$\mathbf{v} = \lim_{\lambda\to\infty} x(\lambda) = \lim_{\lambda\to\infty} \begin{bmatrix} f\dfrac{X_A + \lambda X_D}{Z_A + \lambda Z_D} \\ f\dfrac{Y_A + \lambda Y_D}{Z_A + \lambda Z_D} \end{bmatrix} \Rightarrow$$

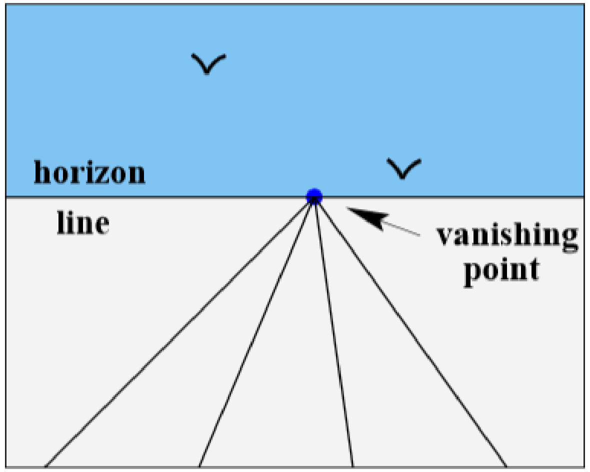
Vanishing point!

$$\mathbf{v} = \begin{bmatrix} fX_D / Z_D \\ fY_D / Z_D \end{bmatrix}$$

Vanishing points

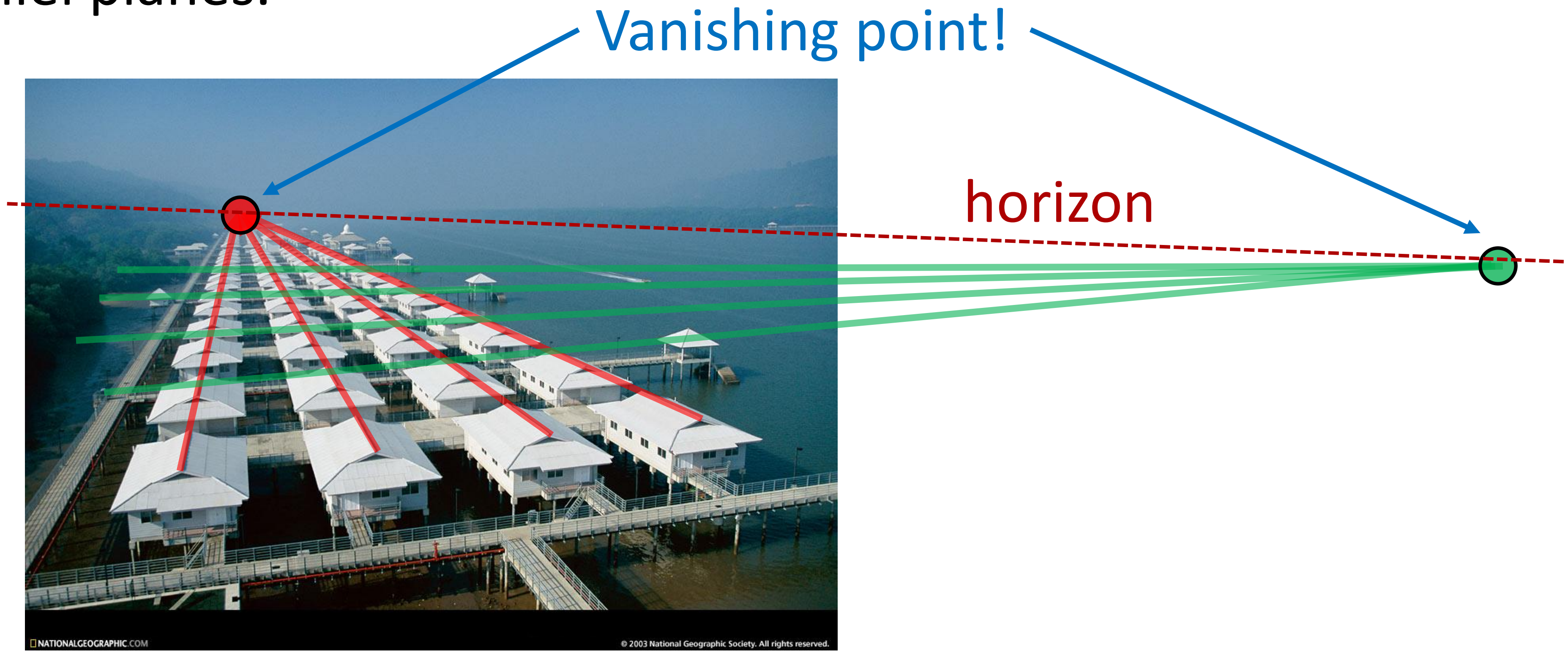
- VP depends on direction $\boldsymbol{D}$, not on point $\boldsymbol{A}$.
- A different set of parallel lines correspond to a different VP!
- Horizon is formed by connecting the vanishing points of a plane





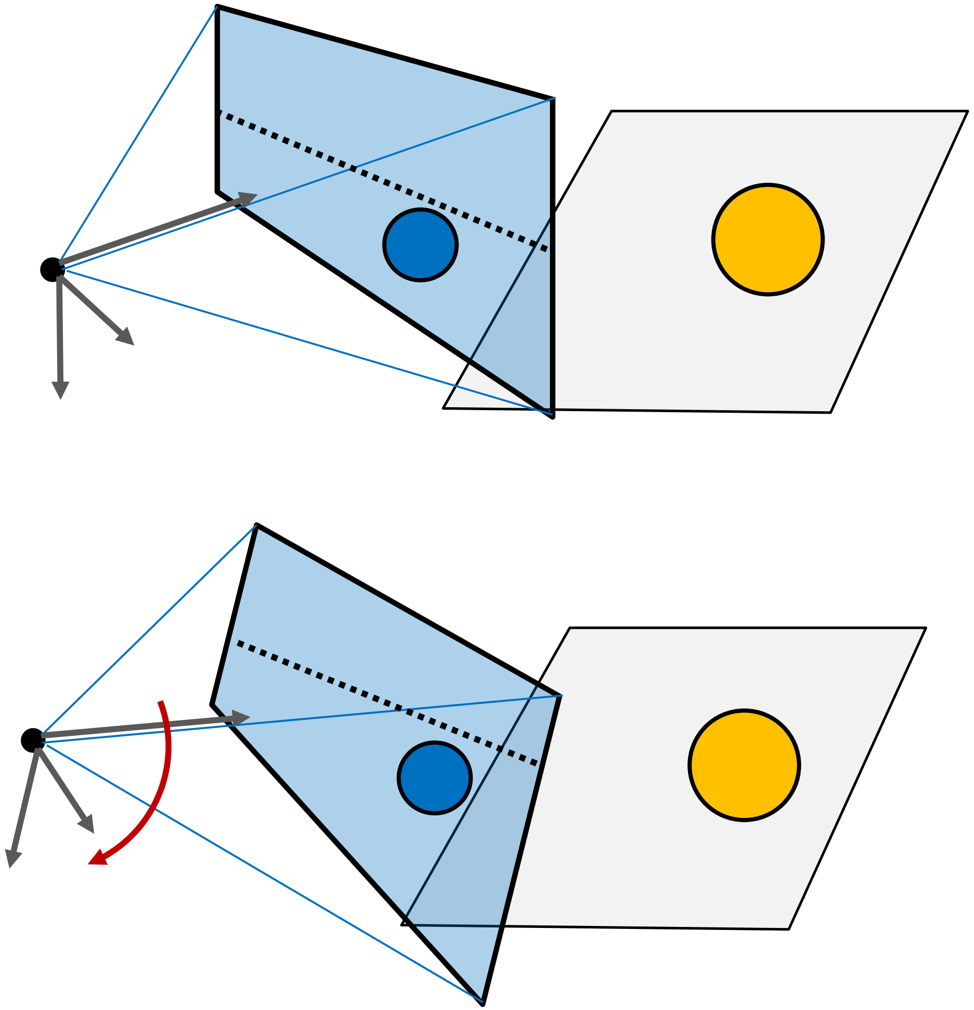
Vanishing points

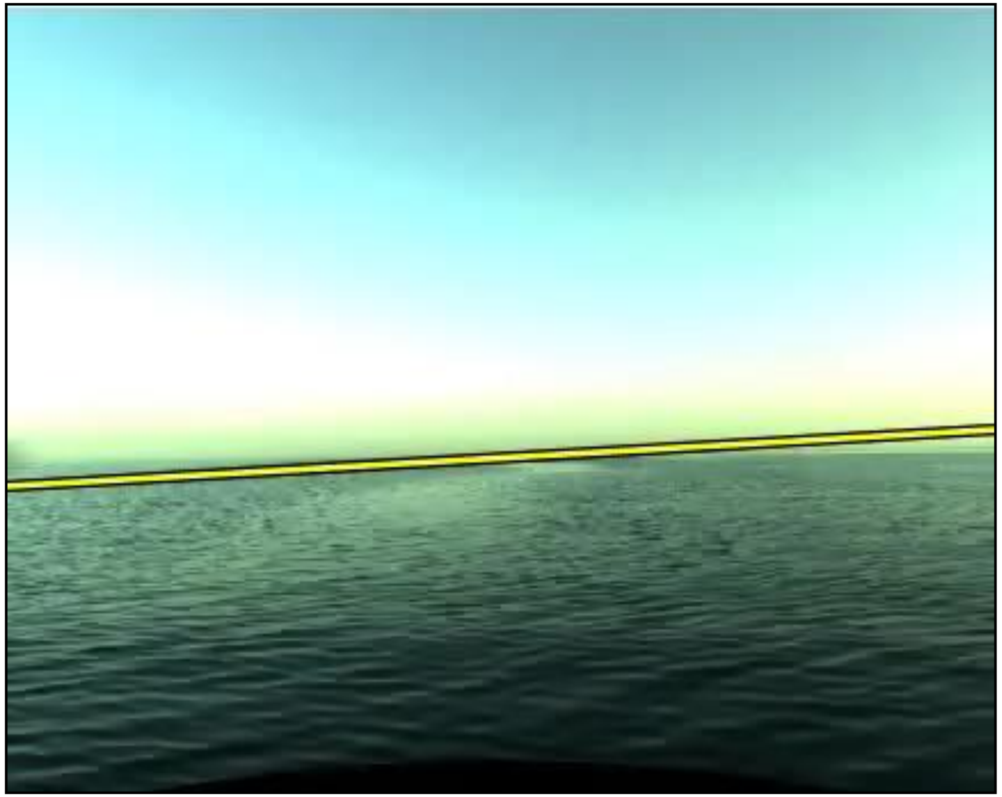
- Horizon is a collection of all the vanishing points corresponding to a set of parallel planes.



- Sets of 3D parallel lines intersect at a vanishing point!

Example: Use IMU to estimate horizon projection



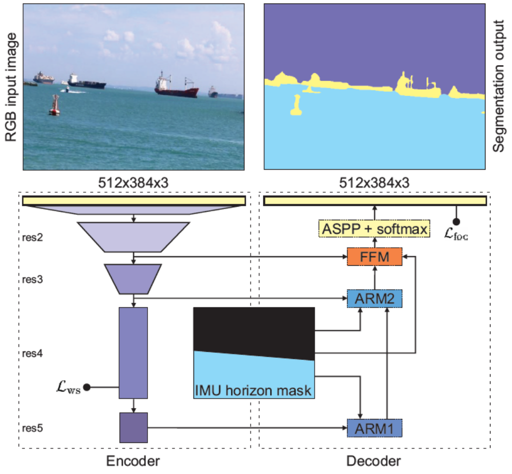


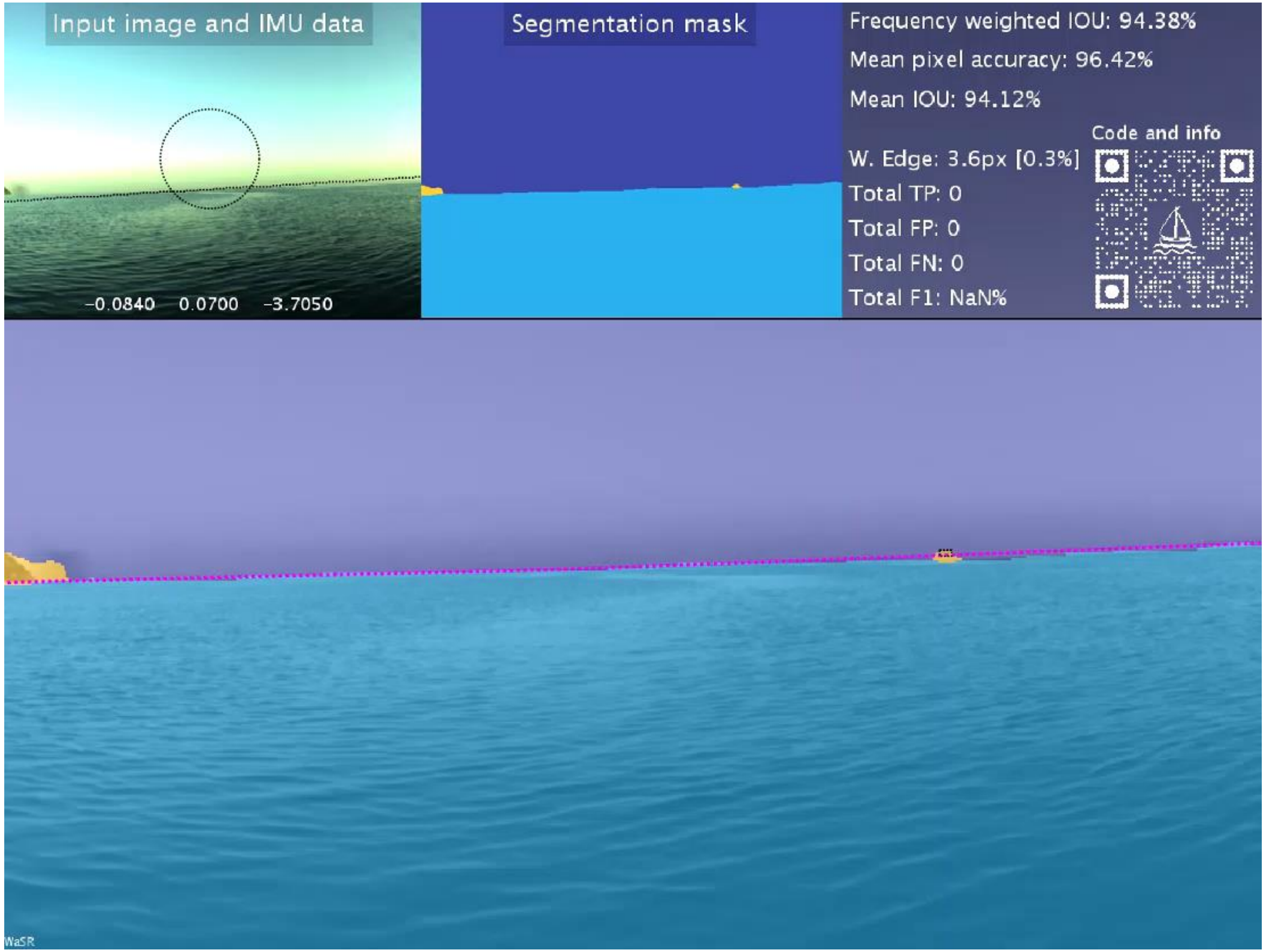
Camera tilt estimated from IMU, horizon projected into image

Bovcon, Perš, Mandeljc, Kristan, Stereo Obstacle Detection for Unmanned Surface Vehicles by IMU-assisted Semantic Segmentation, RAS 2018

Example: Use IMU for obstacle detection

WaSR: Water Separation and Refinement Network





Bovcon, Kristan, A water-obstacle separation and refinement network for unmanned surface vehicles, ICRA 2020

Camera calibration

- Assume a fixed camera in 3D that you want to use for measuring

$$\lambda \mathrm{x} = PX$$

What is this distance in mm?

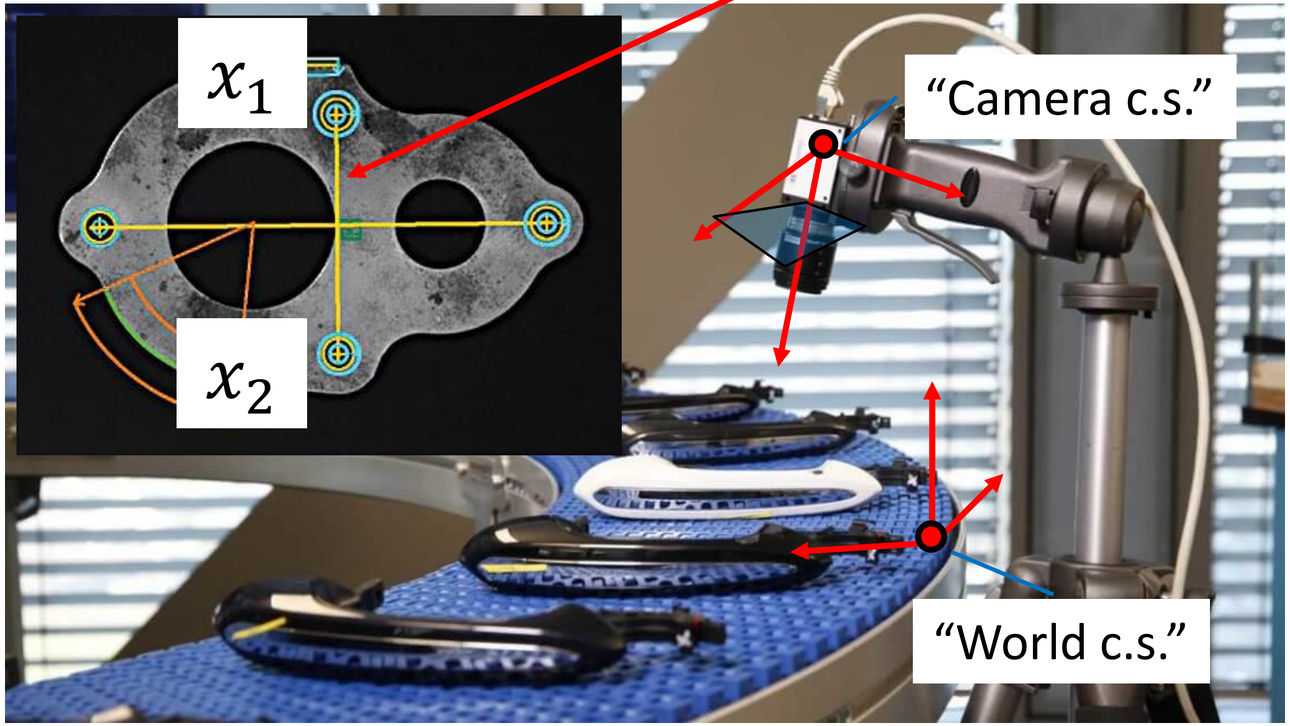
In principle (not really that easy…):

$$X_1 = P^{-1}\mathrm{x}_1, X_2 = P^{-1}\mathrm{x}_2$$

$$d = \| X_1 - X_2 \|$$

What is required to form P?

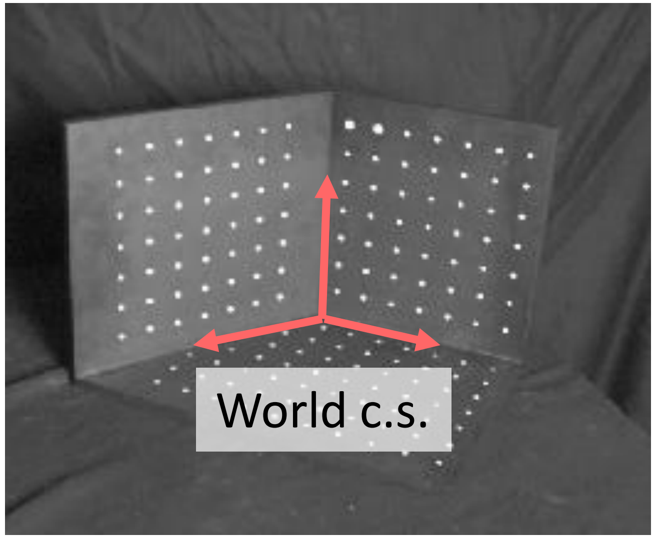
$$\mathrm{x} = \mathrm{K}\left[\mathrm{R} \mid -\mathrm{R}\tilde{\mathrm{C}}\right]\mathrm{X}$$

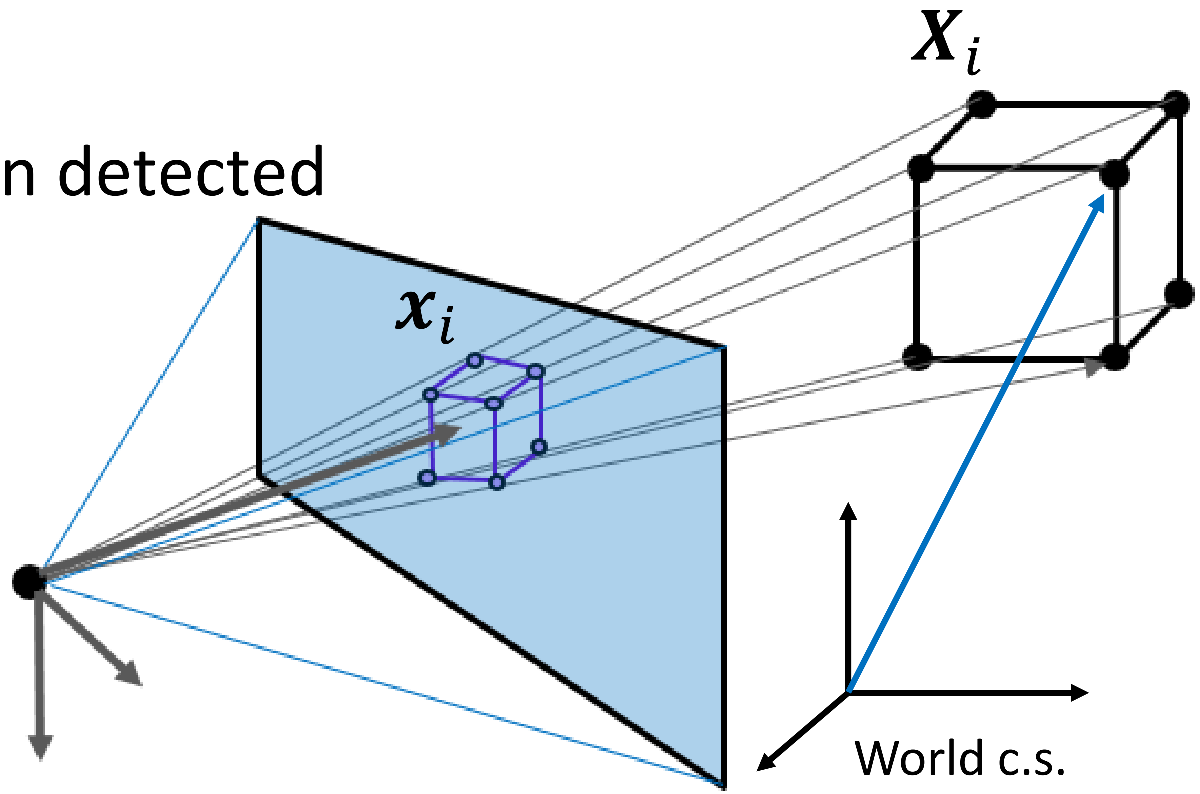


https://vizworld.com/2017/04/watsons-cognitive-visual-inspection-in-lean-manufacturing-processes/

Camera calibration

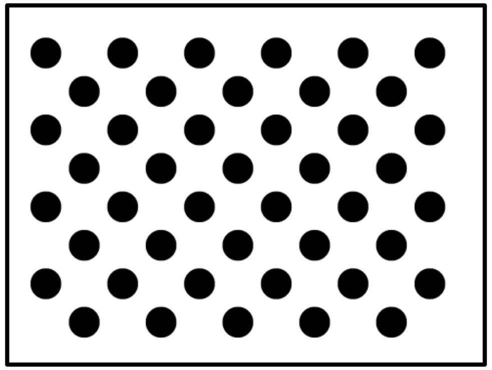
- Camera calibration: *estimate projection matrix* $\boldsymbol{P}$ *from a known calibration object.*
- Corner structures on calibration object for easy and accurate detection
- Coordinates (meters) in 3D known
- Coordinates (pixels) in 2D projection detected



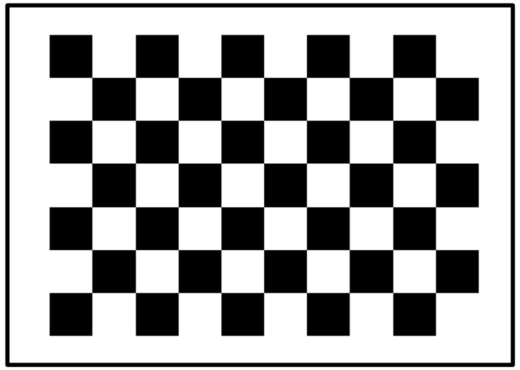


Camera calibration: point detection

- Proper calibration requires measuring the points at sub-pixel accuracy.
- Highly depends on the calibration pattern.



Gives better results



- How many point correspondences are required?
- A rule of thumb:
  - Number of constraints exceeds the number of unknowns by a factor 5.
  - $\Rightarrow$ For 11 parameters in P, use at least 28 points (2 eqs. per point pair).

Camera calibration by DLT

- Standard approach for parameter estimation (DLT)

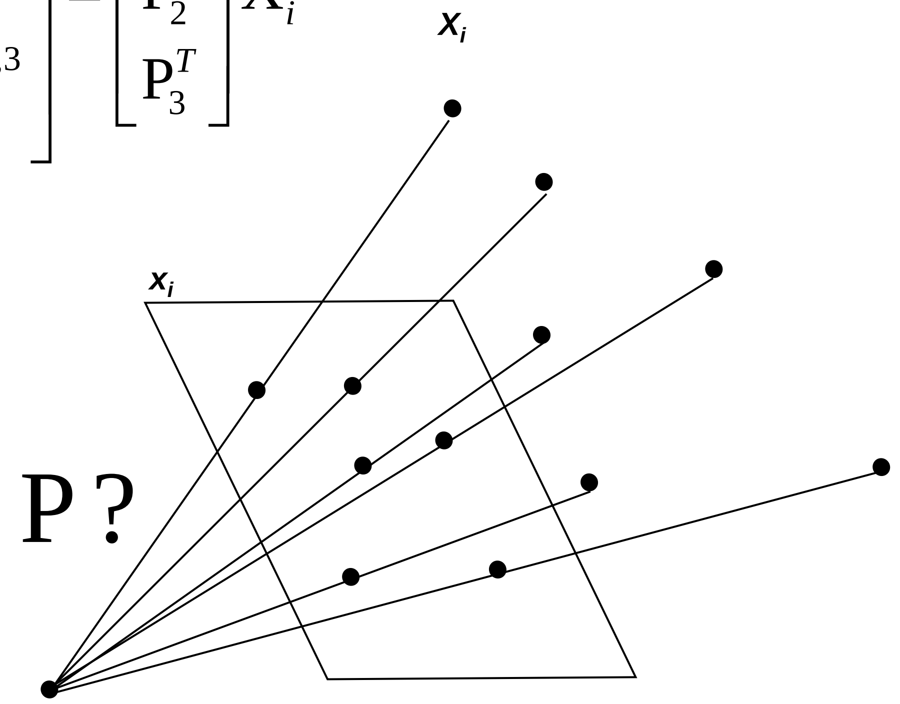
$$\lambda \mathrm{x}_i = \mathrm{P}\mathrm{X}_i$$

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} \mathrm{X}_{i,1} \\ \mathrm{X}_{i,2} \\ \mathrm{X}_{i,3} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{P}_1^T \\ \mathrm{P}_2^T \\ \mathrm{P}_3^T \end{bmatrix} \mathrm{X}_i$$

$$\mathrm{x}_i \times \mathrm{P}\mathrm{X}_i = 0$$

Same approach as with Homography:

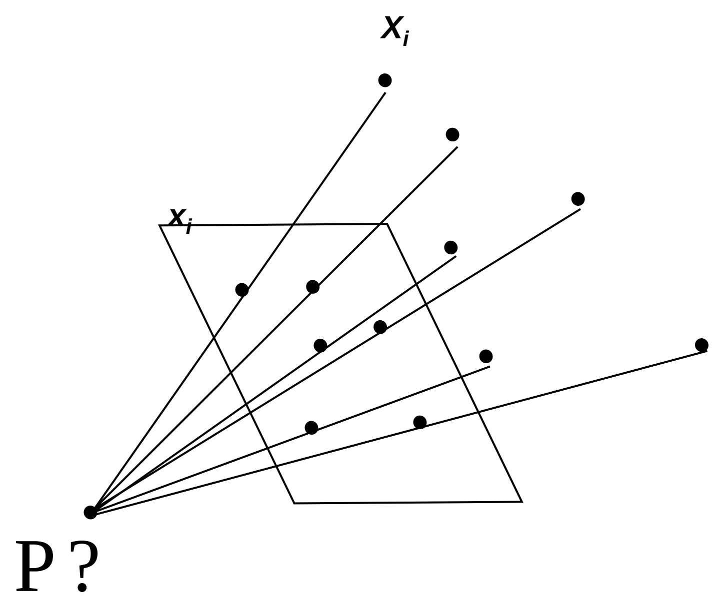
$$\begin{bmatrix} 0^T & -\mathrm{X}_i^T & y_i \mathrm{X}_i^T \\ \mathrm{X}_i^T & 0^T & -x_i \mathrm{X}_i^T \\ -y_i \mathrm{X}_i^T & x_i \mathrm{X}_i^T & 0^T \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$



Camera calibration by DLT

$$\begin{bmatrix} 0^T & X_1^T & -y_1 X_1^T \\ X_1^T & 0^T & -x_1 X_1^T \\ \cdots & \cdots & \cdots \\ 0^T & X_n^T & -y_n X_n^T \\ X_n^T & 0^T & -x_n X_n^T \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0$$

$$AP = 0$$



- P has 11 DoF (12 parameters, but the scale is arbitrary).
- A single 2D-3D correspondence gives two linearly independent equations.
- Homogeneous system is solved by SVD of ***A***.
- Solution requires at least 5 ½ correspondences.
- Caution: coplanar points yield degenerate solutions.
- Apply preconditioning as with Homography estimation.

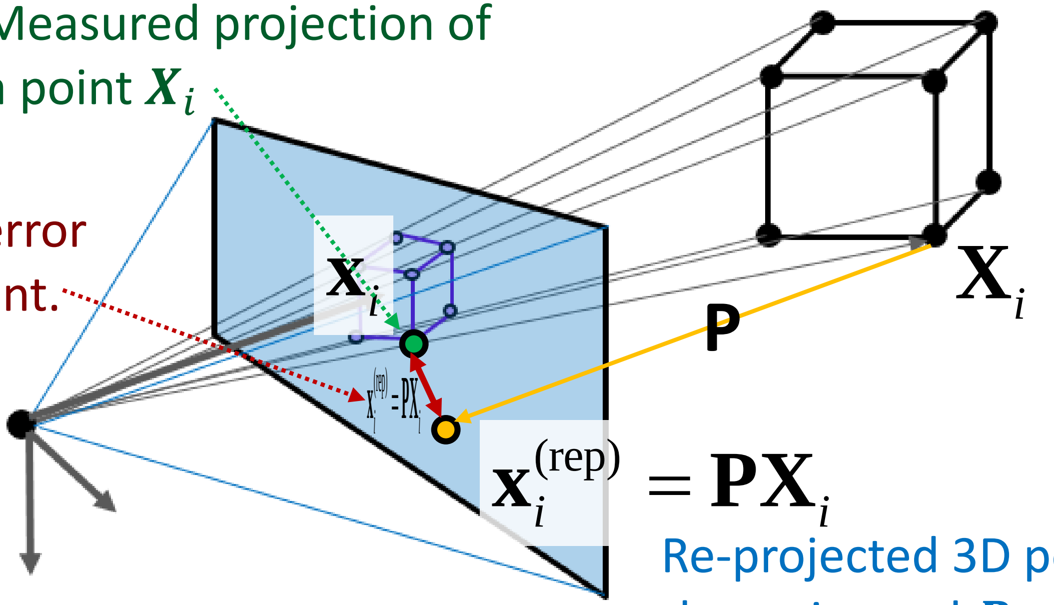
Camera calibration

- Once the projection matrix P is known, we need to figure out its external and internal parameters, i.e., $P=P_{int}P_{ext}=K[R|t]$.
- This is a matrix decomposition problem.
- Intrinsic and extrinsic matrix have a particular form, that makes such a decomposition possible.
- Solution can be found in Forsyth&Ponce, Chapter 3.2, 3.3. for those who are interested to learn more about camera calibration.

Camera calibration: practical advices

- The DLT implementation is pretty simple, but it is an algebraic solution.
- In reality we would like to minimize a *re-projection error*:

$$\boldsymbol{\varepsilon}_i = \begin{bmatrix} \varepsilon_{xi} \\ \varepsilon_{yi} \end{bmatrix} = (\mathbf{x}_i - \mathbf{P}\mathbf{X}_i)$$



- The re-projection error:

$$\mathrm{E}(\mathbf{p}) = \sum_{i=1}^{N} \boldsymbol{\varepsilon}_i^T \boldsymbol{\varepsilon}_i$$

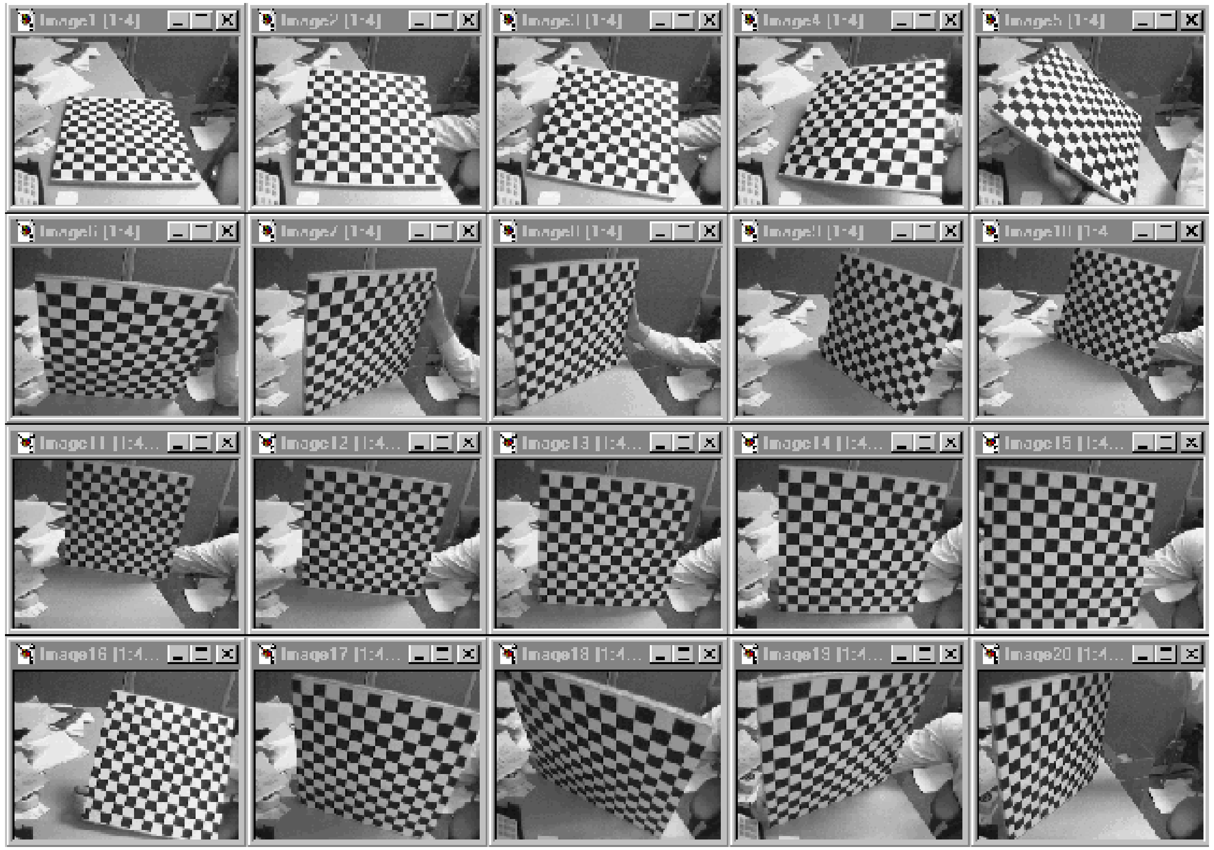
Camera calibration: practical advices

- Nonlinear optimization required (Hartley&Zisserman, Chapter 7.2)
- In practice, initialize by (preconditioned) DLT.
- For practical applications you will need to first remove the radial distortion (H&Z sec. 7.4, or F&P sec. 3.3.).

- Fast and accurate approaches for P matrix estimation still an active research topic

Multiplane camera calibration

- Widely-used approach
- Requires only many images of a single plane
- Does not require knowing positions/orientations



- Good code available online!
  - OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/

Images courtesy Jean-Yves Bouguet, Intel Corp.

Thanks.