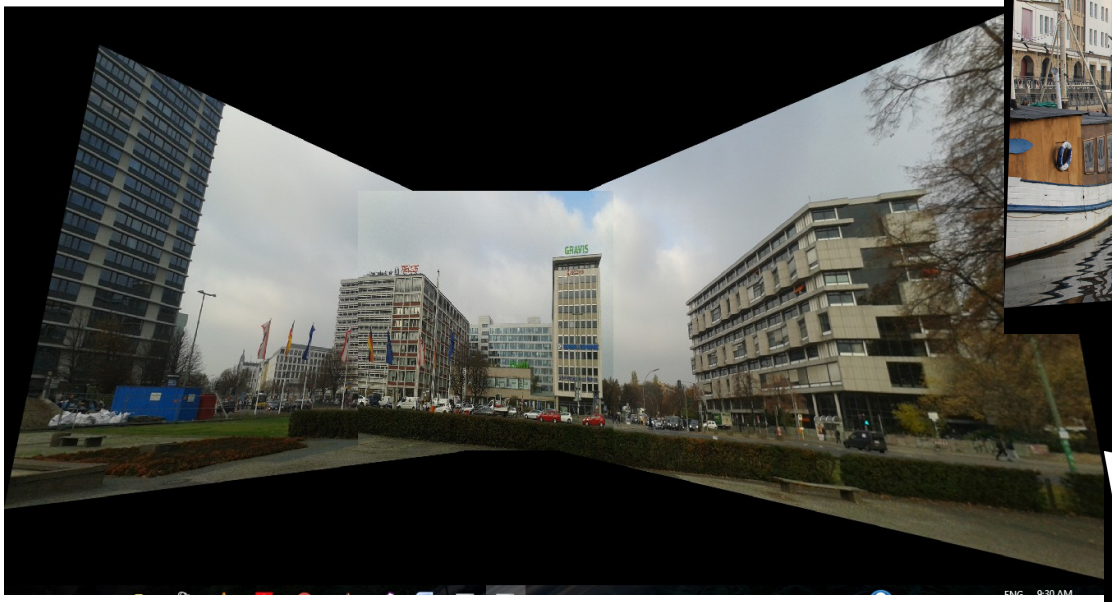
Photogrammetric Computer Vision

Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

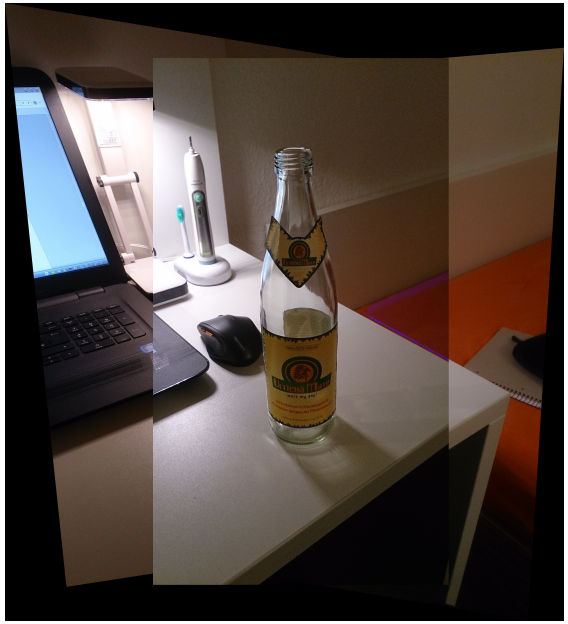
Exams

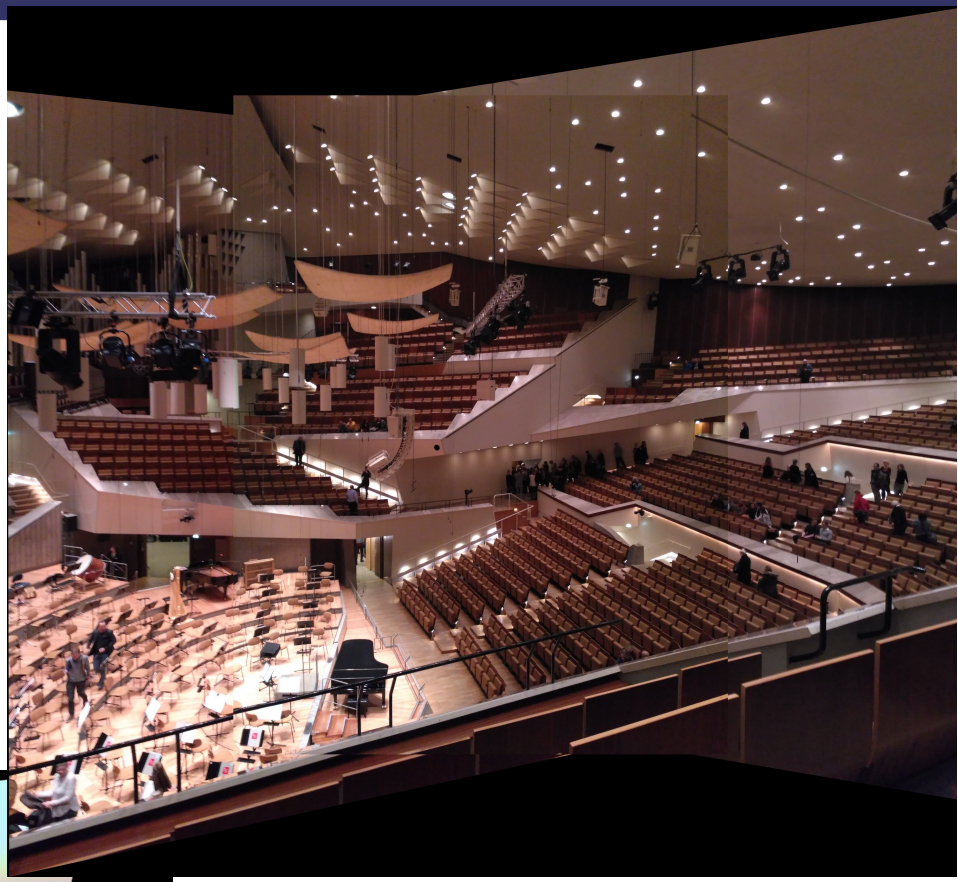
- Mid-term
  - Tuesday, **12.01.2020**
  - Online on ISIS
  - No grade, but pass is necessary to take part at the final exam

Last Exercise

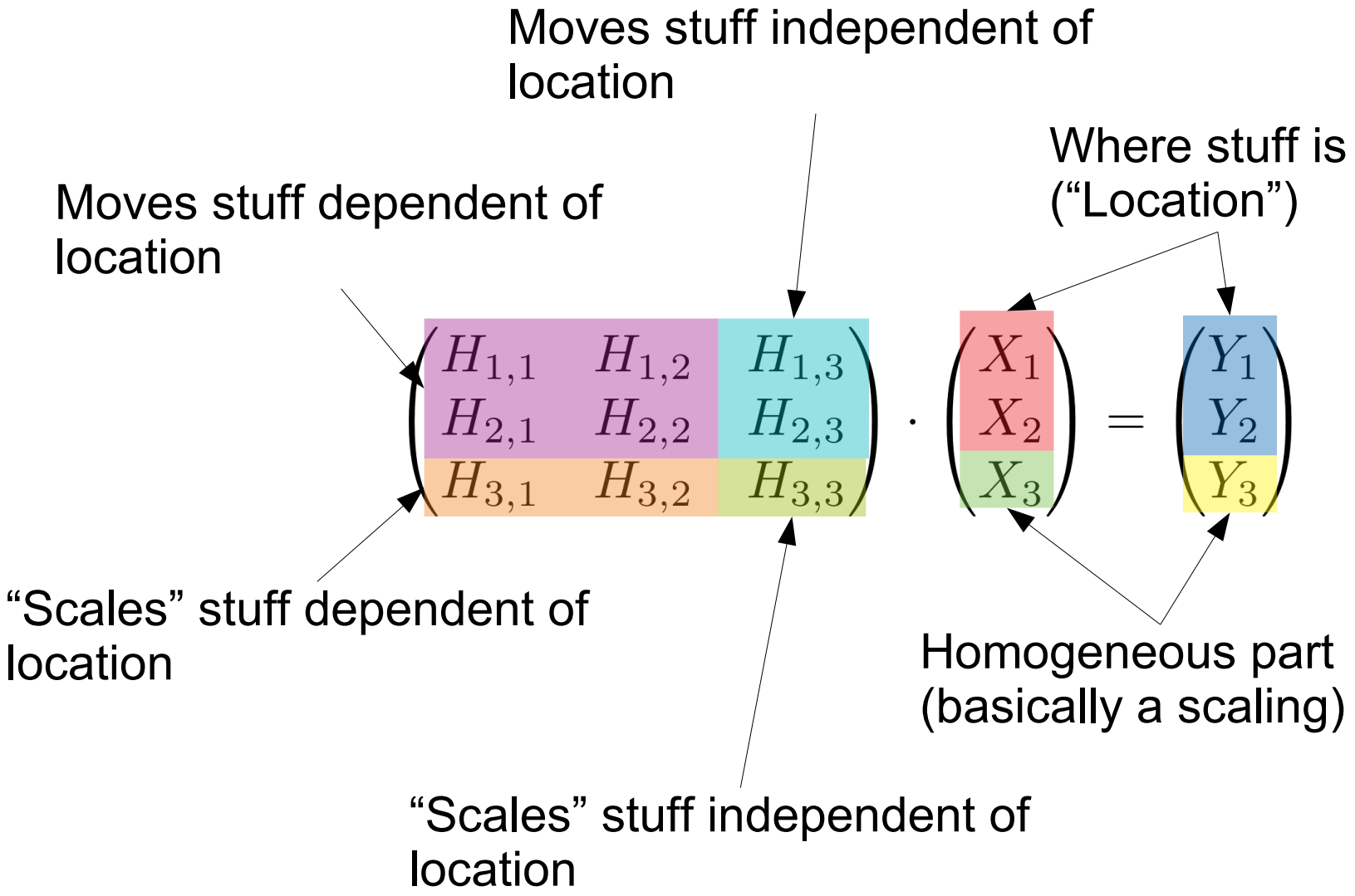


Last Exercise

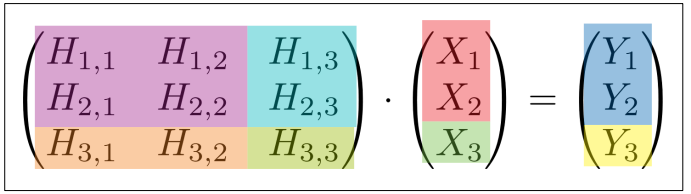




Homographies



Identity Matrix



The identity matrix:

- Does nothing

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

DoF (in 2D): 0

Invariants:

- Everything

Scaling Matrix

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

The scaling matrix:

- Scales independent of location:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \lambda \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

- Or moves dependent of location (but each direction independently):

$$\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

DoF (in 2D):

- Isotropic: 1
- Anisotropic: 2

Invariants:

- Ratio of distances (isotropic only)
- Angle (isotropic only)
- Center of mass
- Parallelism
- Incidence
- Cross ratio

Rotation Matrix

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

The rotation matrix:

- Moves dependent of location:

$$\begin{pmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

- Mnemonic device:
  - It's something with sin and cos
  - For zero angle → identity matrix
  - Point on +x axis moves towards +y
  - Point on +y axis moves towards -x
- Important properties of rotation part:
  - Orthogonal, determinant = 1

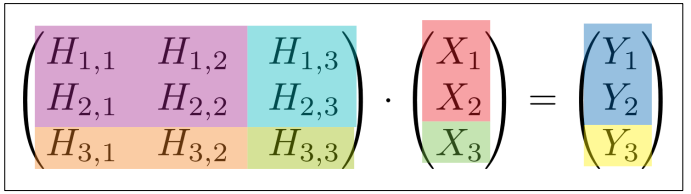
    (assuming 1 on last row)

DoF (in 2D): 1

Invariants:

- Distance
- Volume
- Ratio of distances
- Angle
- Center of mass
- Parallelism
- Incidence
- Cross ratio

Translation Matrix



The translation matrix:

- Moves independent of location:

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

DoF (in 2D): 2

Invariants:

- Distance
- Volume
- Ratio of distances
- Angle
- Center of mass
- Parallelism
- Incidence
- Cross ratio

Motion / Euclidean Transformation

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

Motion:

- Everything you can construct from translation and rotation:
  - Moves independent of location
  - Moves dependent of location
  - $\mathbf{R_a} \cdot \mathbf{R_b} = \mathbf{R_c}$

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

  - Important properties of rotation part:
    - Orthogonal, determinant 1 (assuming 1 on last row)

DoF (in 2D): 1+2=3

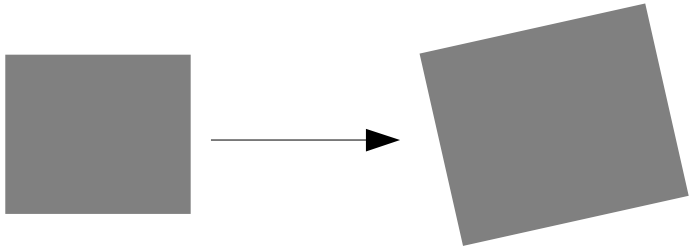
Invariants: Whatever both translation and rotation have

Similarity / Metric Transformation

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

Similarity:

- Everything you can construct from translation, rotation, and isotropic scaling:



$$\begin{pmatrix} \lambda \cdot \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

DoF (in 2D): 4

Invariants: Whatever translation, rotation, and isotropic scaling have in common

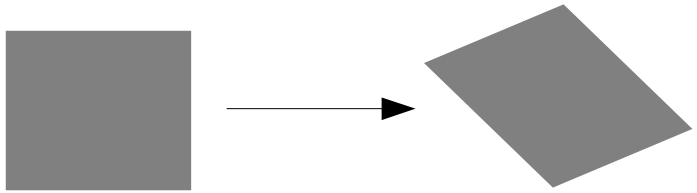
➔ We loose distance and volume

Affinity / Affine Transformation

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

Affinity:

- Everything you can construct from translation, rotation, and anisotropic scaling:
  - ➔ Rotation + Scaling + Rotation = Shearing



  - ➔ No c

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

DoF (in 2D): 6

Invariants: Whatever translation, rotation, and anisotropic scaling have in common

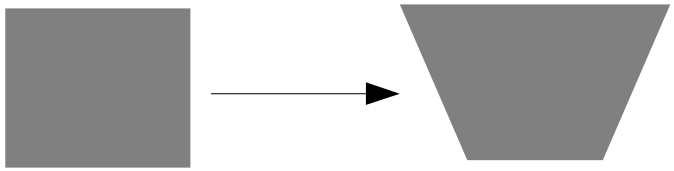
- ➔ We loose distance, volume, ratio of distances, angles

Projectivity / Projective TraFo

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

Perspective projection:

- Everything from before plus...
  - … location dependent scaling



$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

  - No constraints

DoF (in 2D): 8

Invariants:

- We loose distance, volume, ratio of distances, angles, parallelism, center of mass

Example: Projection Matrix

$$\mathbf{P} = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,1} & K_{3,2} & K_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} & H_{1,4} \\ H_{2,1} & H_{2,2} & H_{2,3} & H_{2,4} \\ H_{3,1} & H_{3,2} & H_{3,3} & H_{3,4} \\ H_{4,1} & H_{4,2} & H_{4,3} & H_{4,4} \end{pmatrix}$$

Internal

External

External Orientation

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

External Orientation translates and rotates the scene:

Translation:	Yes
Rotation:	Yes
Scaling:	No
Shearing:	No
Projection:	No

DoF (in 3D): 3+3=6

Invariants: Whatever both translation and rotation have

So it's a motion (translation + rotation)

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

Important properties of rotation part:

- Orthogonal, determinant 1

External Orientation

Which one is a reasonable external orientation?

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 7 \\ 1 & 2 & 3 & 1 \end{pmatrix}$$

- Perspective projection
  - ➔ not composable from rotation and translation

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 1 & 5 \\ 0 & 1 & 0 & 6 \\ -1 & 0 & 0 & 7 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Bottom row is correct
- Rotation part is orthogonal with det 1
  - ➔ composable from rotation and translation

Example: Projection Matrix

$$\mathbf{P} = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,1} & K_{3,2} & K_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

Internal

External

Internal Calibration

$$\begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}$$

Internal calibration:

- Scales isotropically (focal length = “zoom”)
- Possibly small shear and anisotropic scale
- Translation to aprox. image center

DoF: 5

Translation:	Yes
Rotation:	No
Scaling:	Mostly isotropic
Shearing:	Very small, asymmetric
Projection:	No

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}$$

Internal Calibration

Which one is a reasonable internal calibration for a mobile phone camera with 4000x3000 pixels ?

$$\mathbf{K} = \begin{pmatrix} 2980 & 6 & 2103 \\ -6 & 2993 & 1492 \\ 1 & 2 & 1 \end{pmatrix}$$

Not a composition of translation, scaling, and shearing

$$\mathbf{K} = \begin{pmatrix} 0.0003 & 0 & 2103 \\ 0 & 0.00029 & 1492 \\ 0 & 0 & 1 \end{pmatrix}$$

Unreasonable values for focal length

$$\mathbf{K} = \begin{pmatrix} 2980 & 6 & 2103 \\ 0 & 2993 & 1492 \\ 0 & 0 & 1 \end{pmatrix}$$

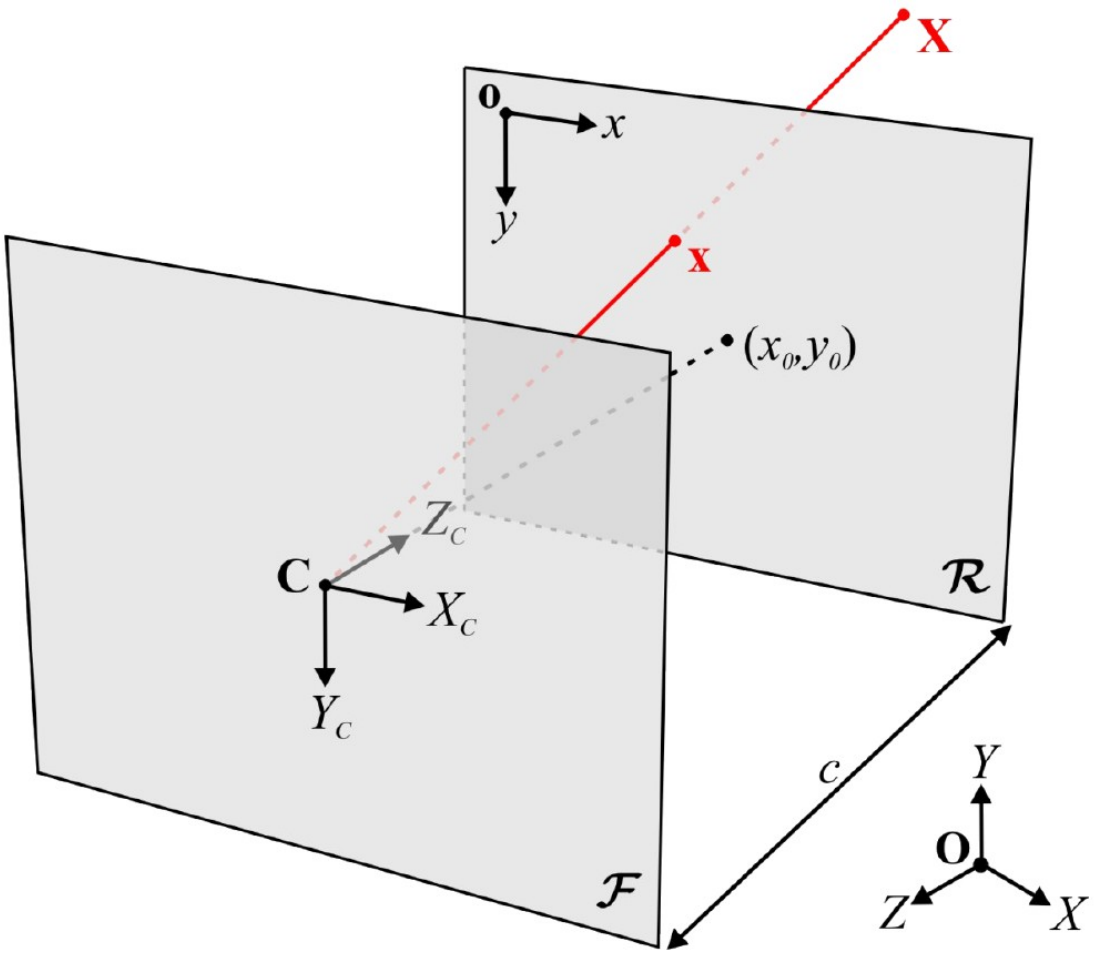
Seems legit

Example: Projection Matrix

$$\mathbf{P} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

3. Exercise

Camera calibration using a direct linear transformation

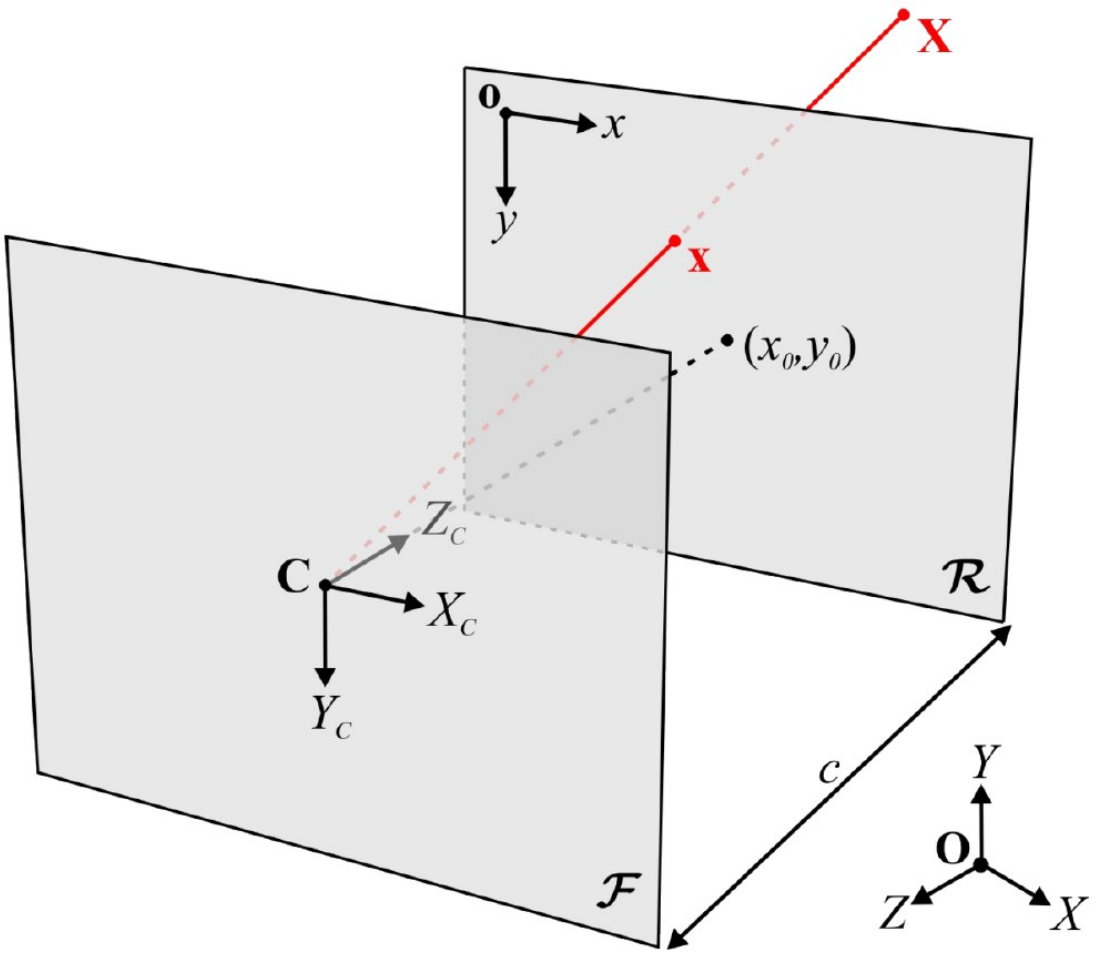


The three-dimensional reconstruction of objects from images requires that the **interior** and **exterior** orientation of the cameras are known.

$$\boldsymbol{x} = \boldsymbol{P} \cdot \boldsymbol{X}$$

3. Exercise

Camera calibration using a direct linear transformation



The three-dimensional reconstruction of objects from images requires that the **interior** and **exterior** orientation of the cameras are known.

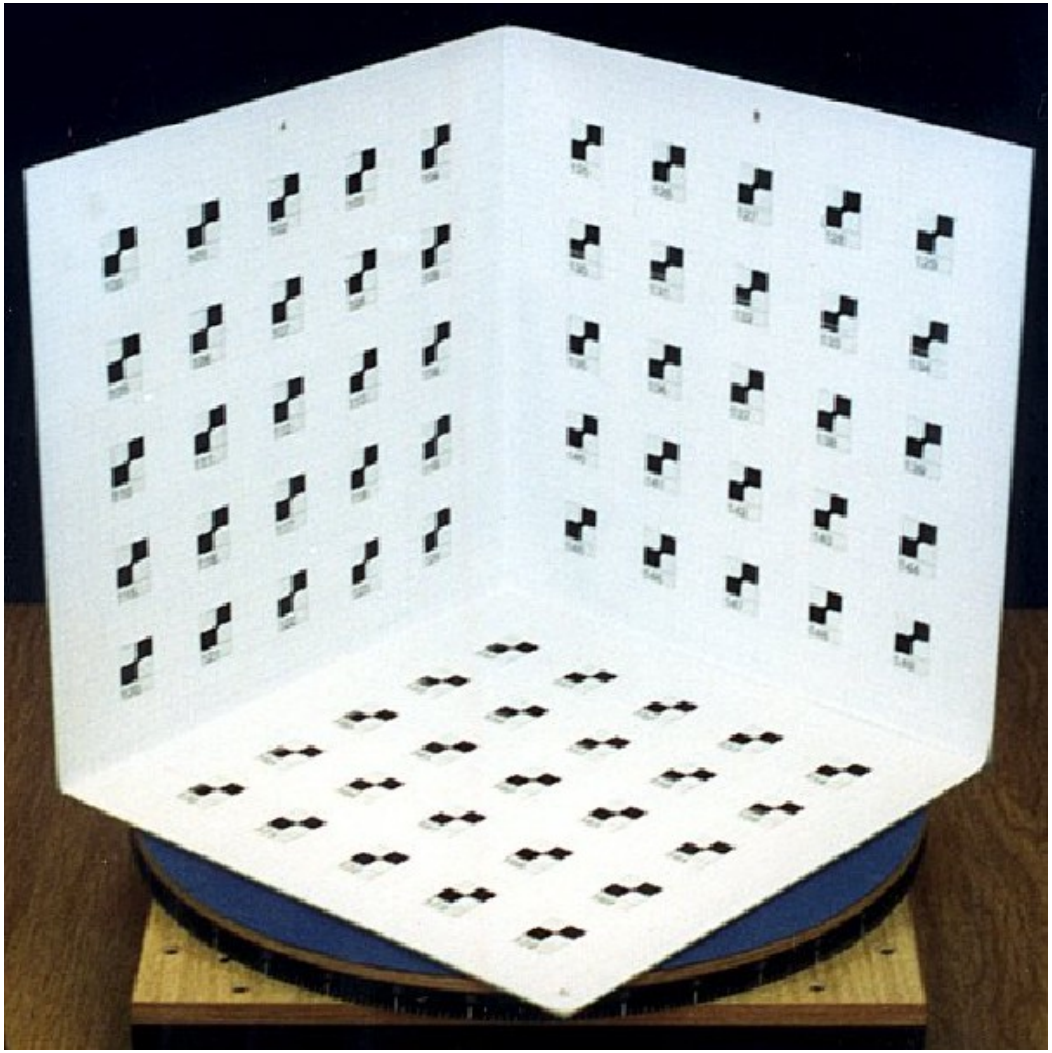
$$\boldsymbol{P} = \boldsymbol{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \boldsymbol{K}\,\boldsymbol{R}\,[\,\boldsymbol{I}\,|-\boldsymbol{C}\,]$$

- Principal distance c
- Principle point (x_0, y_0)
- Aspect ratio γ
- Shearing s

- Projection center $(X_0,Y_0,Z_0)^T$
- Rotation angles ω, ϕ, κ

3. Exercise

Camera calibration using a direct linear transformation



Acquire one image from an object

of your choice and

determine the projection matrix

using a DLT to

reconstruct the geometry

of image formation.

3. Exercise

1. Image Acquisition:

Take a picture of an appropriate calibration object
and transfer this image into the computer.

a) Describe the acquired calibration object in brief.
b) Specify important technical information of the used camera
(i.e. type, resolution, etc.)

2. Control point measurements:

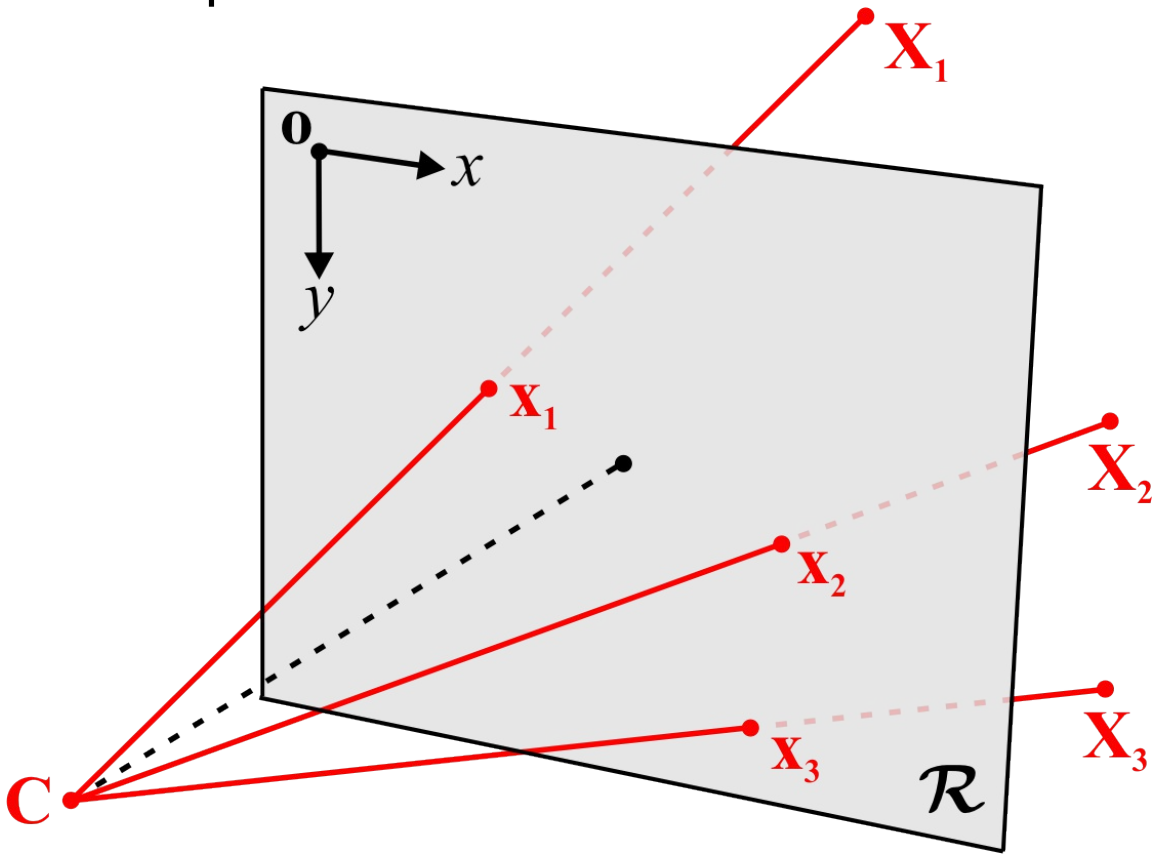
Determine the three-dimensional object coordinates of at least 6 known control points and their two-dimensional image coordinates.

a) How did you define the axes of the object coordinate system?
b) How precise where the object coordinates measured?

3. Exercise

3. Computation of the projection matrix:

Implement a function in C++ for spatial resection using the direct linear estimation method of the projection matrix with help of the singular value decomposition.



$$\boldsymbol{x} = \boldsymbol{P} \cdot \boldsymbol{X}$$

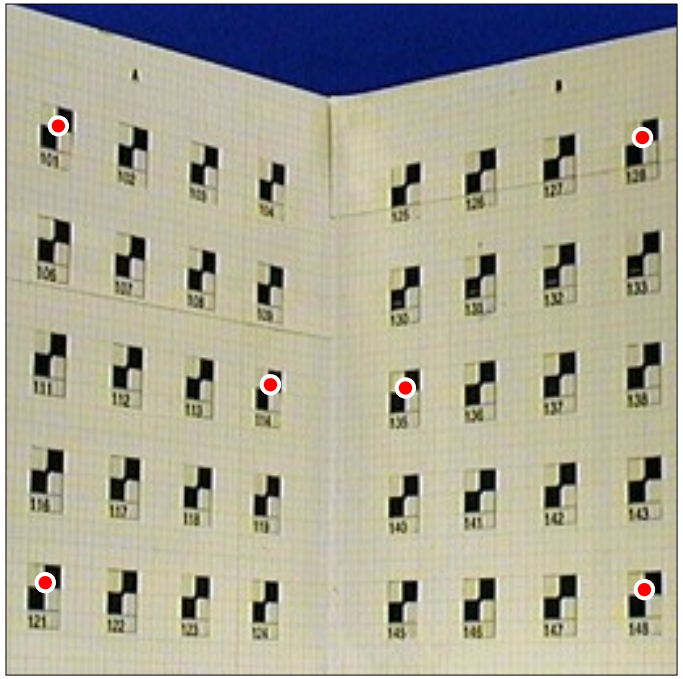
- Estimation of the **exterior orientation** of a camera
- Determination of the **interior orientation** of a camera

3. Exercise

3. Computation of the projection matrix:

Implement a function in C++ for spatial resection using the direct linear estimation method of the projection matrix with help of the singular value decomposition.

Calibration image (256 × 256 Pixel):



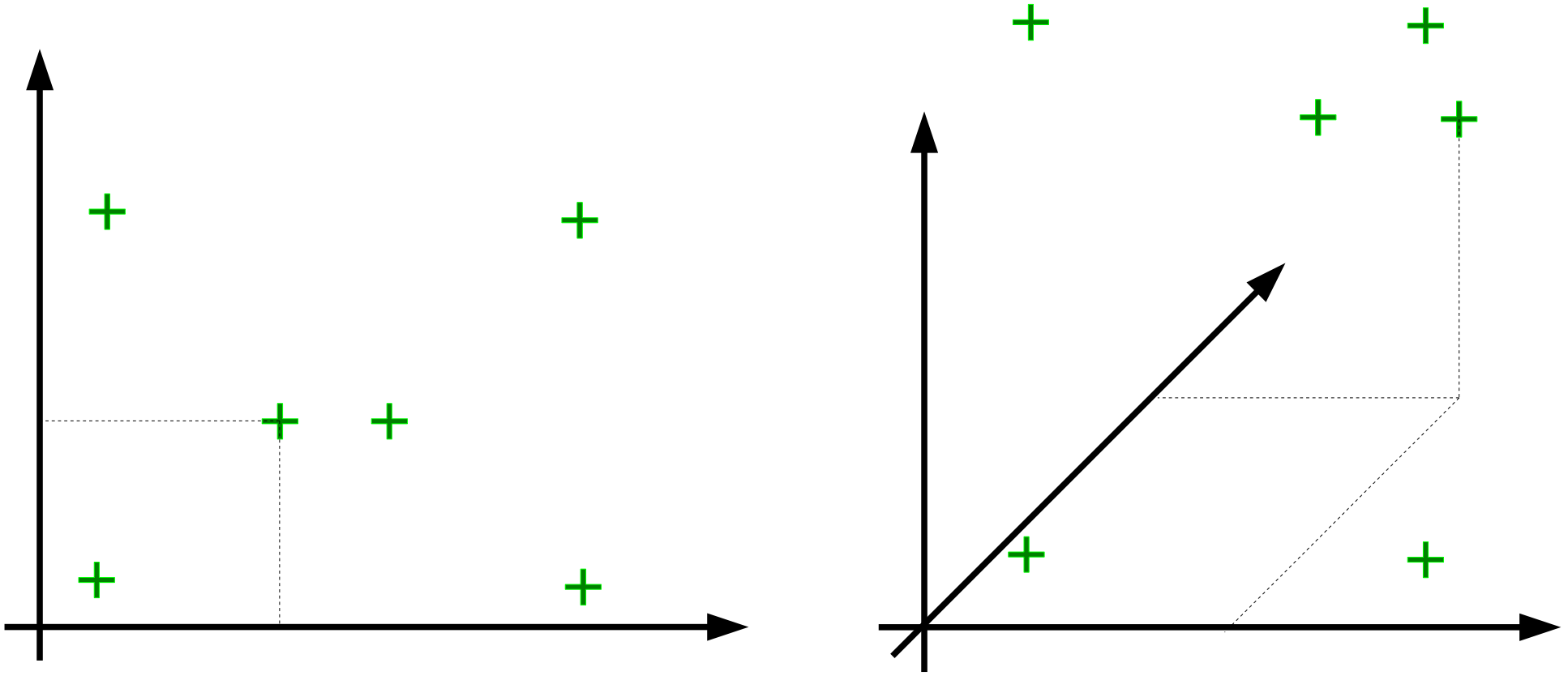
Point Correspondences:

#	Image points [pixel]		Object points [mm]		
	x	y	X	Y	Z
1	18.5	46.8	44.7	-142.4	258.7
2	99.1	146.5	-103.6	-146.6	154.4
3	13.8	221.8	47.4	-150.1	59.8
4	242.1	52.5	-152.2	59.4	245.2
5	151.1	147.1	-153.3	-96.9	151.3
6	243.1	224.5	-149.4	52.7	46.9

3. Exercise

3. Computation of the projection matrix:

Implement a function in C++ for spatial resection using the direct linear estimation method of the projection matrix with help of the singular value decomposition.

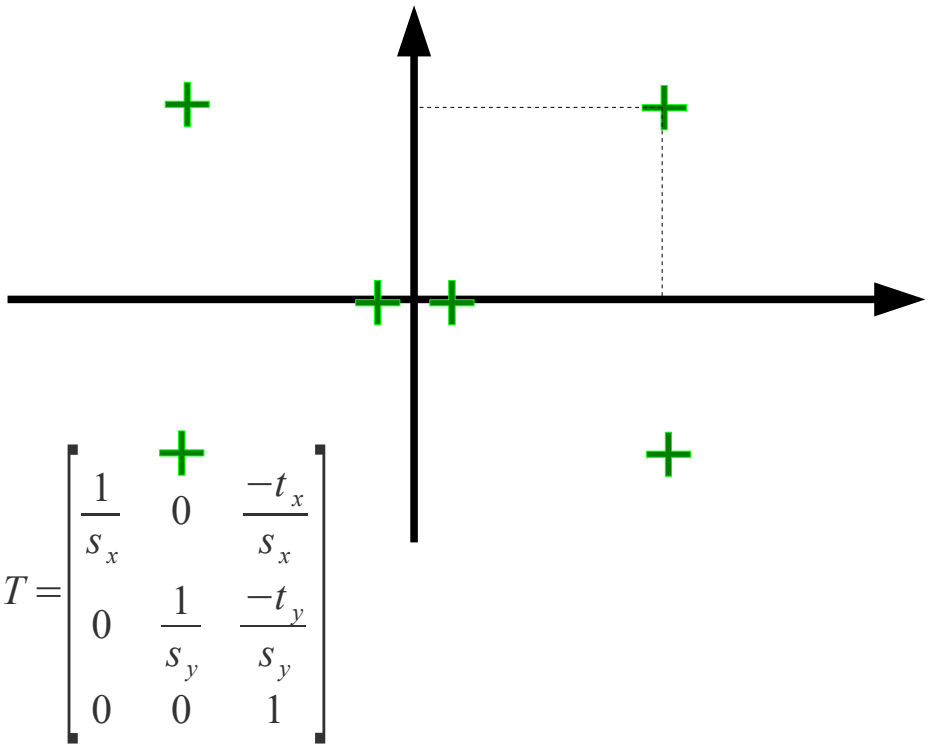


3. Exercise

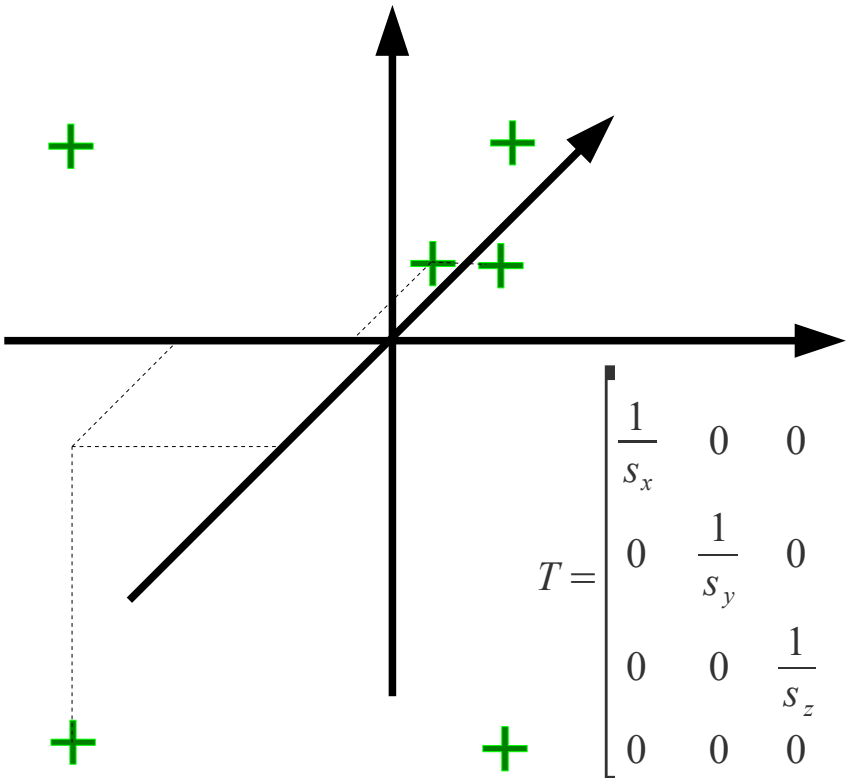
3. Computation of the projection matrix:

Implement a function in C++ for spatial resection using the direct linear estimation method of the projection matrix with help of the singular value decomposition.

1. Conditioning



$$T = \begin{bmatrix} \frac{1}{s_x} & 0 & \frac{-t_x}{s_x} \\ 0 & \frac{1}{s_y} & \frac{-t_y}{s_y} \\ 0 & 0 & 1 \end{bmatrix}$$



$$T = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & \frac{-t_x}{s_x} \\ 0 & \frac{1}{s_y} & 0 & \frac{-t_y}{s_y} \\ 0 & 0 & \frac{1}{s_z} & \frac{-t_z}{s_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Exercise

3. Computation of the projection matrix:

Implement a function in C++ for spatial resection using the direct linear estimation method of the projection matrix with help of the singular value decomposition

2. Create design matrix

$$A_i = \begin{bmatrix} -\tilde{w}' \tilde{\boldsymbol{X}}_i^T & \mathbf{0} & \tilde{u}_i' \tilde{\boldsymbol{X}}_i^T \\ \mathbf{0} & -\tilde{w}' \tilde{\boldsymbol{X}}_i^T & \tilde{v}_i' \tilde{\boldsymbol{X}}_i^T \end{bmatrix}$$

3. Solve equation system with SVD

$$\boldsymbol{A}\boldsymbol{p} = 0$$

4. Reshape and deconditioning

$$\tilde{\boldsymbol{P}} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \qquad \boldsymbol{P} = \boldsymbol{T}'^{-1} \tilde{\boldsymbol{P}} \boldsymbol{T}$$

3. Exercise

4. Interpretation of the projection matrix:

Factorize the projection matrix using a RQ-decomposition

$$\boldsymbol{P} = \boldsymbol{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \boldsymbol{K}\,\boldsymbol{R}\,[\boldsymbol{I}\,|-\boldsymbol{C}] = [\boldsymbol{K}\,\boldsymbol{R}\,|-\boldsymbol{K}\,\boldsymbol{R}\,\boldsymbol{C}] =: [\boldsymbol{M}\,|-\boldsymbol{M}\,\boldsymbol{C}]$$

3. Exercise

4. Interpretation of the projection matrix:

Factorize the projection matrix using a RQ-decomposition

$$\boldsymbol{P} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

$$\lambda = \begin{cases} +\dfrac{1}{\|\boldsymbol{m}^3\|} & \text{if } det(\boldsymbol{M}) > 0 \\ -\dfrac{1}{\|\boldsymbol{m}^3\|} & otherwise \end{cases}$$

→ 3rd row of M

$$M = RQ \Leftrightarrow \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

Additional constraint: Diagonal elements of the triangular matrix are positive!

$$r_{ii} < 0 \Rightarrow \forall\ j \in \{1,2,3\} : r_{ji} = -1 \cdot r_{ji} \quad , \quad q_{ij} = -1 \cdot q_{ij}$$

3. Exercise

4. Interpretation of the projection matrix:

Factorize the projection matrix using a RQ-decomposition

$$\boldsymbol{P} = \boldsymbol{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \boldsymbol{K}\,\boldsymbol{R}\,[\boldsymbol{I}\,|-\boldsymbol{C}] = [\boldsymbol{K}\,\boldsymbol{R}\,|-\boldsymbol{K}\,\boldsymbol{R}\,\boldsymbol{C}] =: [\boldsymbol{M}\,|-\boldsymbol{M}\,\boldsymbol{C}]$$

Camera position

Possibility 1: $\boldsymbol{P}\,\boldsymbol{C} = 0 \quad \rightarrow$ SVD

Possibility 2: $\boldsymbol{P} = [\boldsymbol{M}\,|-\boldsymbol{M}\,\boldsymbol{C}] \quad \rightarrow \quad \boldsymbol{C} = -\boldsymbol{M}^{-1}\,\boldsymbol{P}^4$

Take apart K and R according to lecture "pcv9 projection matrix" slide 9

3. Exercise

Extract Euler angles from rotation matrix

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_z \mathbf{R}_\varphi \mathbf{R}_x = \begin{bmatrix} \cos(z) & -\sin(z) & 0 \\ \sin(z) & \cos(z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(x) & -\sin(x) \\ 0 & \sin(x) & \cos(x) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\varphi)\cos(z) & \sin(x)\sin(\varphi)\cos(z) - \cos(x)\sin(z) & \cos(x)\sin(\varphi)\cos(z) + \sin(x)\sin(z) \\ \cos(\varphi)\sin(z) & \sin(x)\sin(\varphi)\sin(z) + \cos(x)\cos(z) & \cos(x)\sin(\varphi)\sin(z) - \sin(x)\cos(z) \\ -\sin(\varphi) & \sin(x)\cos(\varphi) & \cos(x)\cos(\varphi) \end{bmatrix} \end{aligned}$$

3. Exercise

4. Interpretation of the projection matrix:

Factorize the projection matrix using a RQ-decomposition
and derive all eleven parameters of the interior and exterior orientation.

a.) **Explain** the geometric meaning of the extracted parameters in brief.

b.) Evaluate the whole calibration process. How precise is the camera orientation determined and where does the quality depend on?

3. Exercise

Camera Calibration

Given:

- Main function
  - Variable declaration
  - Call of necessary functions
- Header of individual functions

Todo:

- Individual functions
  - Fill in the necessary function body parts

Given

```
int getPoints(struct winInfo& calib, string fname,
vector<Vec3f>& points2D, vector<Vec4f>& points3D)
```

calib: Image and window title of calibration image
fname: path to file that contains 3D real world points
points2D: Output, points within image (homogeneous coordinates)
points3D: Output, points at object (homogeneous coordinates)

- Displays the image.
- Catches left mouse clicks, stores position in corresponding array and marks points in images by green circles.
- Reads object points from file
  → One point per line, components divided by blanks
  → Image points have to be defined in the order of the object points

To Do

```
Matx33f getCondition2D(vector<Vec3f>& p)
Matx44f getCondition3D(vector<Vec4f>& p)
```

p: N Points as array of 3D or 4D vectors
return: The 3x3 (4x4) matrix for point conditioning

- Computes the transformation matrix to move centroid to origin and scale mean absolute distance to origin to one

$$T = \begin{bmatrix} \frac{1}{s_x} & 0 & \frac{-t_x}{s_x} \\ 0 & \frac{1}{s_y} & \frac{-t_y}{s_y} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & \frac{-t_x}{s_x} \\ 0 & \frac{1}{s_y} & 0 & \frac{-t_y}{s_y} \\ 0 & 0 & \frac{1}{s_z} & \frac{-t_z}{s_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To Do

```
vector<Vec3f> applyH_2D(vector<Vec3f>& geomObj,
                        Matx33f& H, GeometryType type)
```

`geomObj` : the geometric objects, that have to be transformed
`H` : the homography of the transformation
`type` : determines the type of the geometric object (point/line)
`return` : the transformed objects

- Reuse your implementation from exercise 1

```
vector<Vec4f> applyH_3D_points(vector<Vec4f>& geomObj,
                               Matx44f& H)
```

`geomObj` : the **points**, that have to be transformed
`H` : the homography of the transformation
`return` : the transformed points

- Extend to 3D, but only for points

To Do

```
Mat_<float> getDesignMatrix_camera(vector<Vec3f>&
        points2D, vector<Vec4f>& points3D)
```

points2D: set of 2D points within the image
points3D: set of 3D points at the object
return: Output, Designmatrix to compute projection matrix

- Takes corresponding point pairs (already conditioned)
- Generates design matrix to compute projection matrix

$$A_i = \begin{bmatrix} -\tilde{w}' \tilde{\mathbf{X}}_i^T & \mathbf{0} & \tilde{u}_i' \tilde{\mathbf{X}}_i^T \\ \mathbf{0} & -\tilde{w}' \tilde{\mathbf{X}}_i^T & \tilde{v}_i' \tilde{\mathbf{X}}_i^T \end{bmatrix}$$

To Do

```
Mat_<float> getDesignMatrix_camera(vector<Vec3f>&
        points2D, vector<Vec4f>& points3D)
```

points2D: set of 2D points within the image
points3D: set of 3D points at the object
return: Output, Designmatrix to compute projection matrix

```
Matx34f solve_dlt_camera(Mat_<float>& A)
```

A: Design matrix representing homogeneous eqn. system Ap = 0
return: Projection matrix

- Uses SVD to solve homogeneous equation system
- Note: `cv::SVD(..., SVD::FULL_UV)` returns V^T
  → last column of V is last row of V^T

To Do

```
Matx34f decondition_camera(
                    Matx33f& T_2D, Matx44f& T_3D, Matx34f& P)
```

`T_2D`: Conditioning matrix of set of 2D image points
`T_3D`: Conditioning matrix of set of 3D object points
`P`: Conditioned projection matrix
`returns`: Unconditioned projection matrix

$$\tilde{\boldsymbol{P}} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \qquad \boldsymbol{P} = \boldsymbol{T}'^{-1}\,\tilde{\boldsymbol{P}}\,\boldsymbol{T}$$

To Do

```
Matx34f calibrate(vector<Vec3f>& points2D,
                  vector<Vec4f>& points3D)
```

points2D: N points as 3xN matrix from image
points3D: N points as 4xN matrix from object
return: Output, 3x4 projection matrix

- Estimates projection matrix from given set of point pairs
- Calls:
  - `getCondition2D(...), getCondition3D(...)`
  - `applyH_2D(…), applyH_3D_points(...)`
  - `getDesignMatrix_camera(...)`
  - `solve_dlt_camera(...)`
  - `decondition_camera(...)`

To Do

```
void interprete(Matx34f& P, Matx33f &K, Matx33f R,
                ProjectionMatrixInterpretation& info)
```

P: 3x4 projection matrix
K: Return internal calibration here
R: Return 3x3 rotation here
info: Return interpretation here

- Calculates and prints all 11 parameters of the interior and exterior orientation
  - `RQDecomp3x3(...), atan2(...)`
  - Do not use `cv::decomposeProjectionMatrix(...)` !

$$M = RQ \Leftrightarrow \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

To Do

```
void interprete(Matx34f& P, Matx33f &K, Matx33f R,
                ProjectionMatrixInterpretation& info)
```

```
/// Interpretation of the internal and external parts of a projection matrix.
struct ProjectionMatrixInterpretation
{
    /// Principal distance or focal length
    float principalDistance;
    /// Skew as an angle and in degrees
    float skew;
    /// Aspect ratio of the pixels
    float aspectRatio;
    /// Location of principal point in image (pixel) coordinates
    Vec2f principalPoint;
    /// Camera rotation angle 1/3
    float omega;
    /// Camera rotation angle 2/3
    float phi;
    /// Camera rotation angle 3/3
    float kappa;
    /// 3D camera location in world coordinates
    Vec3f cameraLocation;
};
```

See lecture "pcv9 projection matrix"

Next Meeting

Deadline:

30.12.2020

Next meeting:

05.01.2020