Computer Vision: Lecture 3

2023-11-06

Planned contents

Week 1	Intro, camera model	Projective geometry
Week 2	Camera calibration, DLT I	DLT II, feature matching
Week 3	Two-view geometry I	Two-view geometry II
Week 4	Robust estimation	Minimal solvers
Week 5	MLE & Non-Linear opt., Factorization	Non-Seq. SfM, project pres.
Week 6	Bundle adjustment I, Uncertainty	Bundle adjustment II
Week 7	Non-rigid SfM (guest)	Dense reconstruction

Last Time

Projective Geometry

- Homogeneous coordinates, projective space
- Lines, planes and conics
- Projective transformations and homographies

Projective Transformations

Projective transformation

A projective transformation $\mathtt{H}$ is an invertible linear mapping from $\mathbb{P}^k \rightarrow \mathbb{P}^k$,

$$\mathbf{y} = \mathtt{H}\mathbf{x}.$$

It can be represented by an invertible matrix $\mathtt{H} \in \mathbb{R}^{(k+1)\times(k+1)}$.

- Projective transformations are defined (non-zero) up-to-scale

$$\mathtt{H}\mathbf{x} \sim \lambda\mathtt{H}\mathbf{x}$$

- *Homographies* are projective transformations in $\mathbb{P}^2$, $\mathtt{H} \in \mathbb{R}^{3\times3}$
  - 8 degrees of freedom (why?)
  - 4 correspondences $\mathbf{x}_i \leftrightarrow \mathbf{y}_i$ to estimate $\mathtt{H}$

Homographies

Plane transfer between images

If $\mathbf{X}$ is on a 3D plane Π, then there exists a homography $\mathtt{H}$ such that

$$\mathbf{y} = \mathtt{P}_2\mathbf{X} = \mathtt{H}\mathtt{P}_1\mathbf{X} = \mathtt{H}\mathbf{x}$$

- Assume $\Pi = (0, 0, 1, 0)^\top$ ($z = 0$ plane). Then $\mathbf{X} = (X_1, X_2, 0, \mu)^\top \in \mathbb{P}^3$.
- Write $\mathtt{P}_i = (\mathtt{A}^i \mid \mathbf{b}^i)$ for $i = 1, 2$

$$\mathbf{x} = \mathtt{A}^1 \begin{pmatrix} X_1 \\ X_2 \\ 0 \end{pmatrix} + \mu\mathbf{b}^1 = \underbrace{(\mathtt{A}^1_{1:3,1:2} \mid \mathbf{b}^1)}_{=:\mathtt{C}} \begin{pmatrix} X_1 \\ X_2 \\ \mu \end{pmatrix}$$

- Solve for $(X_1, X_2, \mu)^\top = \mathtt{C}^{-1}\mathbf{x}$
- Insert into 2nd camera

$$\mathbf{y} \sim \mathtt{P}_2 \begin{pmatrix} X_1 \\ X_2 \\ 0 \\ \mu \end{pmatrix} = \underbrace{(\mathtt{A}^2_{1:3,1:2} \mid \mathbf{b}^2)\mathtt{C}^{-1}}_{=\mathtt{H}} \mathbf{x}$$

Homographies

Plane transfer between images

If $\mathbf{X}$ is on a 3D plane Π, then there exists a homography $\mathtt{H}$ such that

$$\mathbf{y} = \mathtt{P}_2\mathbf{X} = \mathtt{H}\mathtt{P}_1\mathbf{X} = \mathtt{H}\mathbf{x}$$

- Assume $\Pi = (0, 0, 1, 0)^\top$ ($z = 0$ plane). Then $\mathbf{X} = (X_1, X_2, 0, \mu)^\top \in \mathbb{P}^3$.
- Write $\mathtt{P}_i = (\mathtt{A}^i \mid \mathbf{b}^i)$ for $i = 1, 2$

$$\mathbf{x} = \mathtt{A}^1 \begin{pmatrix} X_1 \\ X_2 \\ 0 \end{pmatrix} + \mu\mathbf{b}^1 = \underbrace{(\mathtt{A}^1_{1:3,1:2} \mid \mathbf{b}^1)}_{=:\mathtt{C}} \begin{pmatrix} X_1 \\ X_2 \\ \mu \end{pmatrix}$$

- Solve for $(X_1, X_2, \mu)^\top = \mathtt{C}^{-1}\mathbf{x}$
- Insert into 2nd camera

$$\mathbf{y} \sim \mathtt{P}_2 \begin{pmatrix} X_1 \\ X_2 \\ 0 \\ \mu \end{pmatrix} = \underbrace{(\mathtt{A}^2_{1:3,1:2} \mid \mathbf{b}^2)\mathtt{C}^{-1}}_{=\mathtt{H}} \mathbf{x}$$

Homographies

Plane transfer between images

If $\mathbf{X}$ is on a 3D plane Π, then there exists a homography $\mathtt{H}$ such that

$$\mathbf{y} = \mathtt{P}_2\mathbf{X} = \mathtt{H}\mathtt{P}_1\mathbf{X} = \mathtt{H}\mathbf{x}$$

- Assume $\Pi = (0, 0, 1, 0)^\top$ ($z = 0$ plane). Then $\mathbf{X} = (X_1, X_2, 0, \mu)^\top \in \mathbb{P}^3$.
- Write $\mathtt{P}_i = (\mathtt{A}^i \mid \mathbf{b}^i)$ for $i = 1, 2$

$$\mathbf{x} = \mathtt{A}^1 \begin{pmatrix} X_1 \\ X_2 \\ 0 \end{pmatrix} + \mu\mathbf{b}^1 = \underbrace{(\mathtt{A}^1_{1:3,1:2} \mid \mathbf{b}^1)}_{=:\mathtt{C}} \begin{pmatrix} X_1 \\ X_2 \\ \mu \end{pmatrix}$$

- Solve for $(X_1, X_2, \mu)^\top = \mathtt{C}^{-1}\mathbf{x}$
- Insert into 2nd camera

$$\mathbf{y} \sim \mathtt{P}_2 \begin{pmatrix} X_1 \\ X_2 \\ 0 \\ \mu \end{pmatrix} = \underbrace{(\mathtt{A}^2_{1:3,1:2} \mid \mathbf{b}^2)\mathtt{C}^{-1}}_{=\mathtt{H}} \mathbf{x}$$

Homographies

Plane transfer between images

If $\mathbf{X}$ is on a 3D plane Π, then there exists a homography $\mathtt{H}$ such that

$$\mathbf{y} = \mathtt{P}_2\mathbf{X} = \mathtt{H}\mathtt{P}_1\mathbf{X} = \mathtt{H}\mathbf{x}$$

- Assume $\Pi = (0, 0, 1, 0)^\top$ ($z = 0$ plane). Then $\mathbf{X} = (X_1, X_2, 0, \mu)^\top \in \mathbb{P}^3$.
- Write $\mathtt{P}_i = (\mathtt{A}^i \mid \mathbf{b}^i)$ for $i = 1, 2$

$$\mathbf{x} = \mathtt{A}^1 \begin{pmatrix} X_1 \\ X_2 \\ 0 \end{pmatrix} + \mu\mathbf{b}^1 = \underbrace{(\mathtt{A}^1_{1:3,1:2} \mid \mathbf{b}^1)}_{=:\mathtt{C}} \begin{pmatrix} X_1 \\ X_2 \\ \mu \end{pmatrix}$$

- Solve for $(X_1, X_2, \mu)^\top = \mathtt{C}^{-1}\mathbf{x}$
- Insert into 2nd camera

$$\mathbf{y} \sim \mathtt{P}_2 \begin{pmatrix} X_1 \\ X_2 \\ 0 \\ \mu \end{pmatrix} = \underbrace{(\mathtt{A}^2_{1:3,1:2} \mid \mathbf{b}^2)\mathtt{C}^{-1}}_{=\mathtt{H}} \mathbf{x}$$

Homographies

Plane transfer between images

If $\mathbf{X}$ is on a 3D plane Π, then there exists a homography $\mathtt{H}$ such that

$$\mathbf{y} = \mathtt{P}_2\mathbf{X} = \mathtt{H}\mathtt{P}_1\mathbf{X} = \mathtt{H}\mathbf{x}$$

- Assume $\Pi = (0, 0, 1, 0)^\top$ ($z = 0$ plane). Then $\mathbf{X} = (X_1, X_2, 0, \mu)^\top \in \mathbb{P}^3$.
- Write $\mathtt{P}_i = (\mathtt{A}^i \mid \mathbf{b}^i)$ for $i = 1, 2$

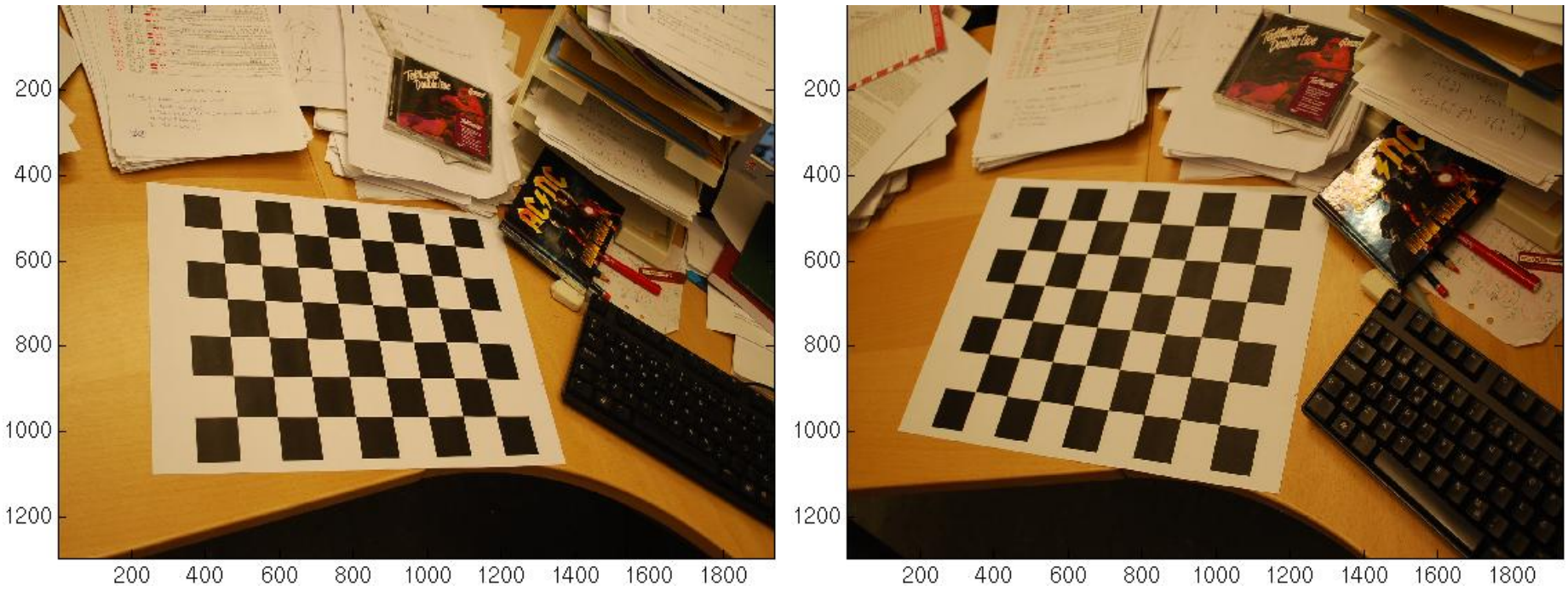
$$\mathbf{x} = \mathtt{A}^1 \begin{pmatrix} X_1 \\ X_2 \\ 0 \end{pmatrix} + \mu\mathbf{b}^1 = \underbrace{(\mathtt{A}^1_{1:3,1:2} \mid \mathbf{b}^1)}_{=:\mathtt{C}} \begin{pmatrix} X_1 \\ X_2 \\ \mu \end{pmatrix}$$

- Solve for $(X_1, X_2, \mu)^\top = \mathtt{C}^{-1}\mathbf{x}$
- Insert into 2nd camera

$$\mathbf{y} \sim \mathtt{P}_2 \begin{pmatrix} X_1 \\ X_2 \\ 0 \\ \mu \end{pmatrix} = \underbrace{(\mathtt{A}^2_{1:3,1:2} \mid \mathbf{b}^2)\mathtt{C}^{-1}}_{=\mathtt{H}} \mathbf{x}$$

Projective Transformations

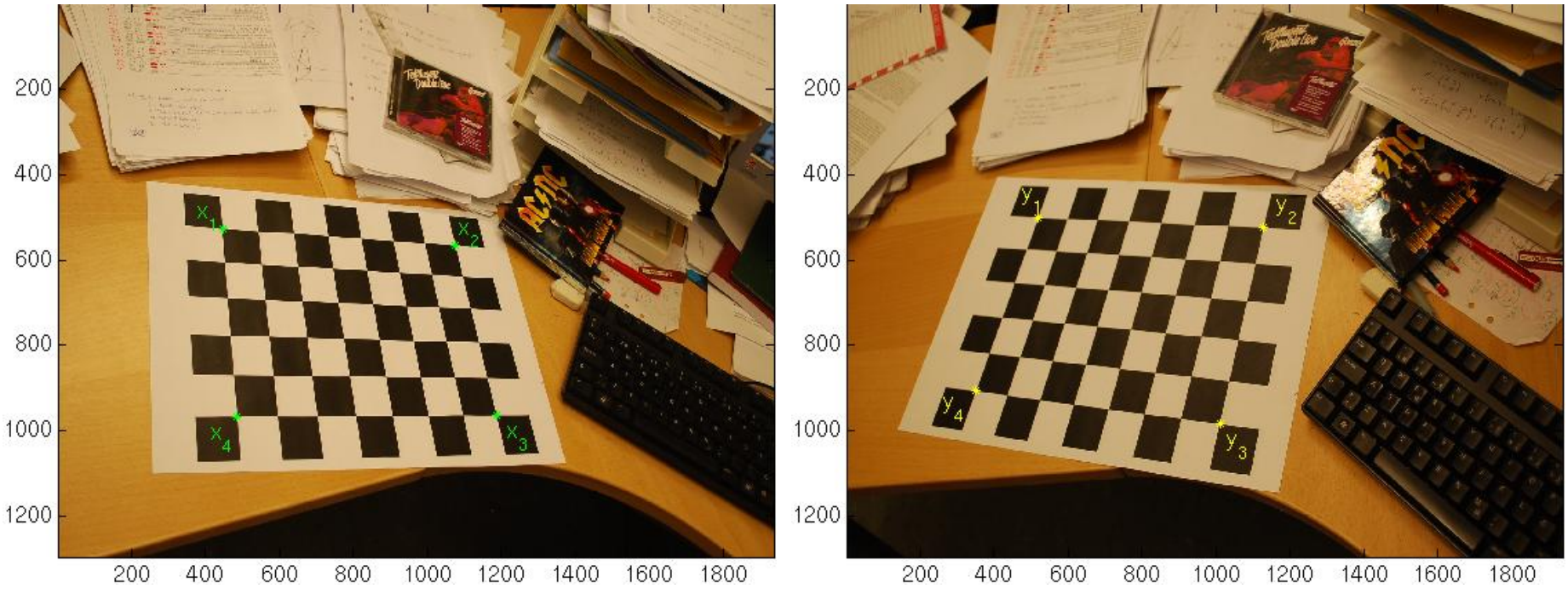
Example: Point Transfer via a Plane.



If a set of points $\mathbf{X}_i$ lying on the same plane is projected into two cameras $\mathbf{x}_i \sim \mathrm{P}_1\mathbf{X}_i$, $\mathbf{y}_i \sim \mathrm{P}_2\mathbf{X}_i$, then there is a homography such that $\mathbf{x}_i \sim \mathrm{H}\mathbf{y}_i$.

Projective Transformations

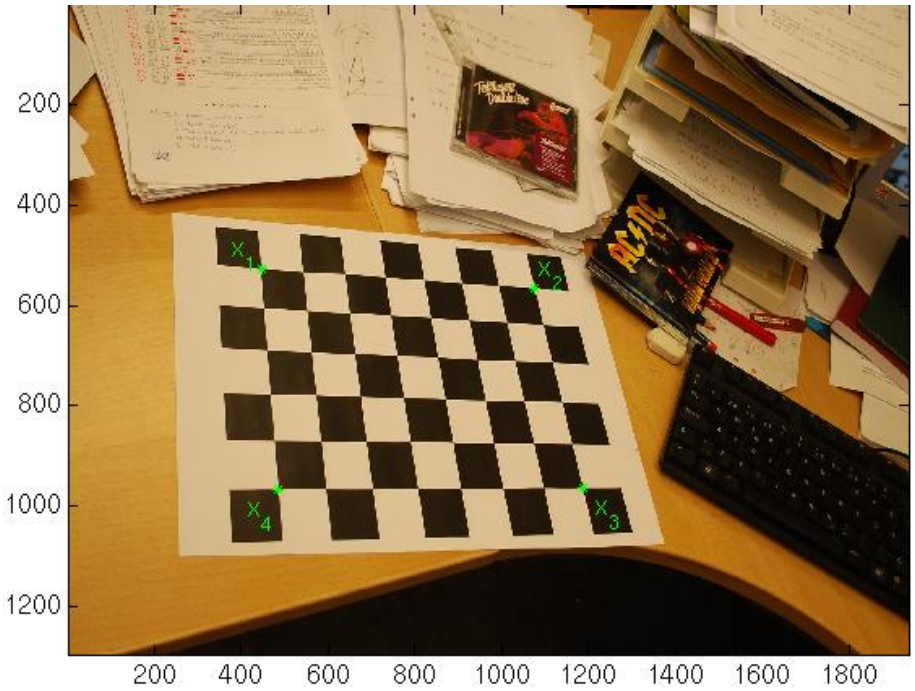
Example: Point Transfer via a Plane.

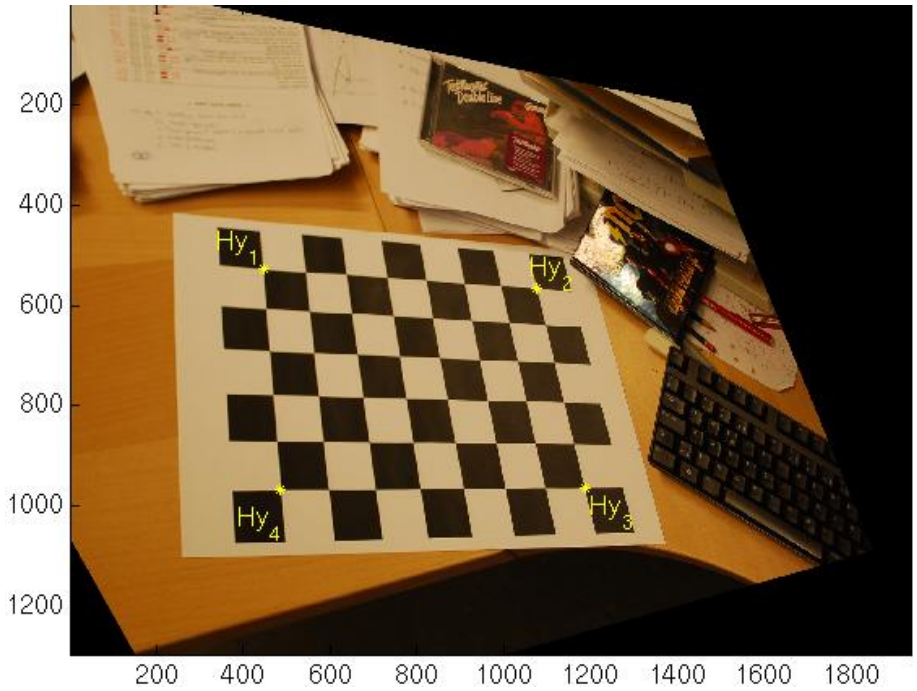


Compute the homography by selecting (at least) 4 points, and solving $\lambda_i \mathbf{x}_i = \mathtt{H}\mathbf{y}_i$.

Projective Transformations

Example: Point Transfer via a Plane.

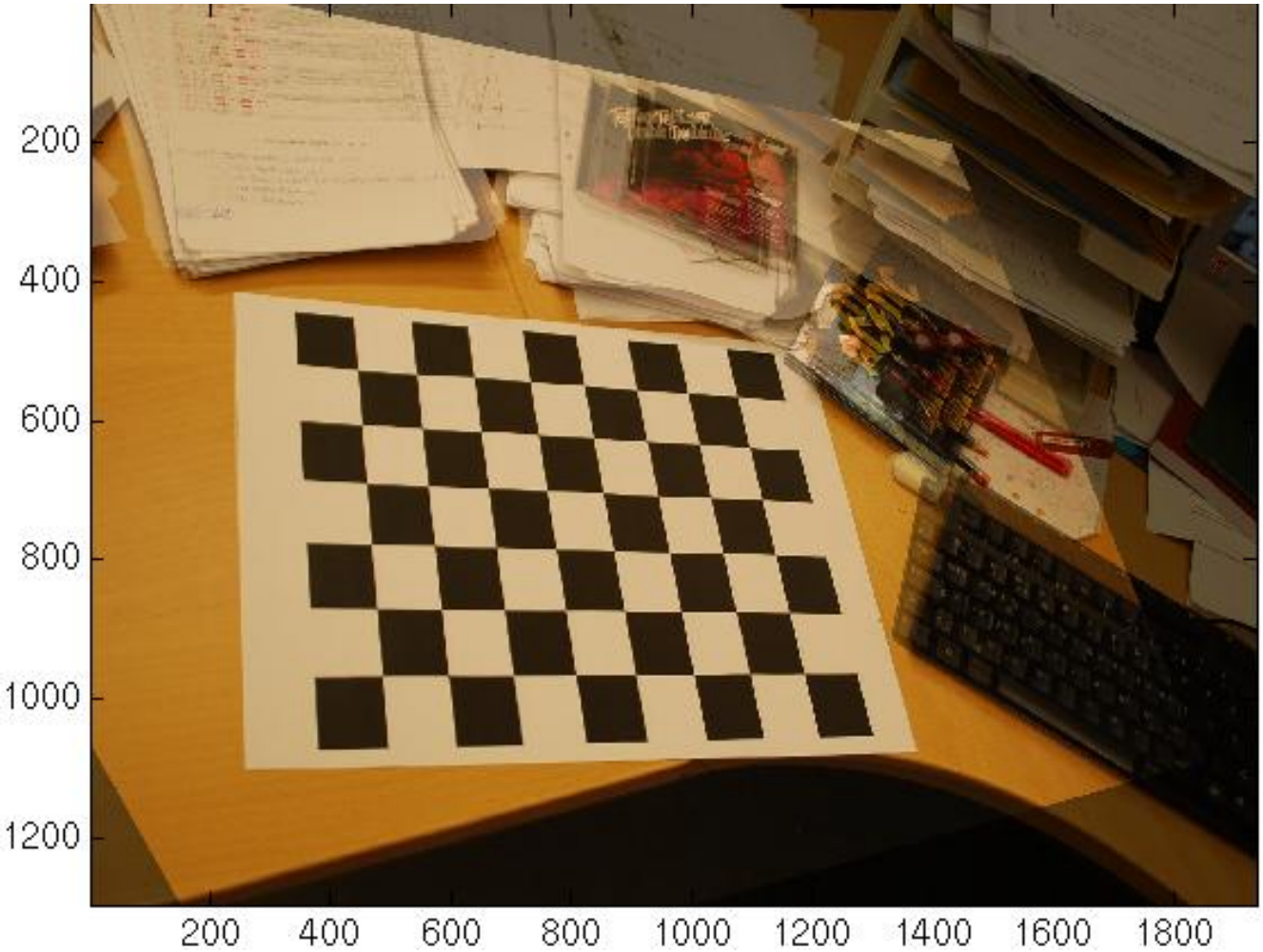




Apply transformation to right image.
(Uses matlabs imtransform.)

Projective Transformations

Example: Point Transfer via a Plane.



Mean value of the two images. Points on the plane seem to agree.

Today's Lecture

Camera Calibration

- The inner parameters - `K`
- Projective vs. Euclidean Reconstruction
- Finding the camera matrix
- Lens distortion

The Inner Parameters - $\mathtt{K}$

The matrix $\mathtt{K}$ is the upper triangular matrix:

$$\mathtt{K} = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

- f - focal length
- γ - aspect ratio
- s - skew
- (x_0, y_0) - principal point

The Inner Parameters - K

The focal length f

$$\begin{pmatrix} fx \\ fy \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Re-scales the images/image coordinates (e.g. mm $\rightarrow$ pixels).

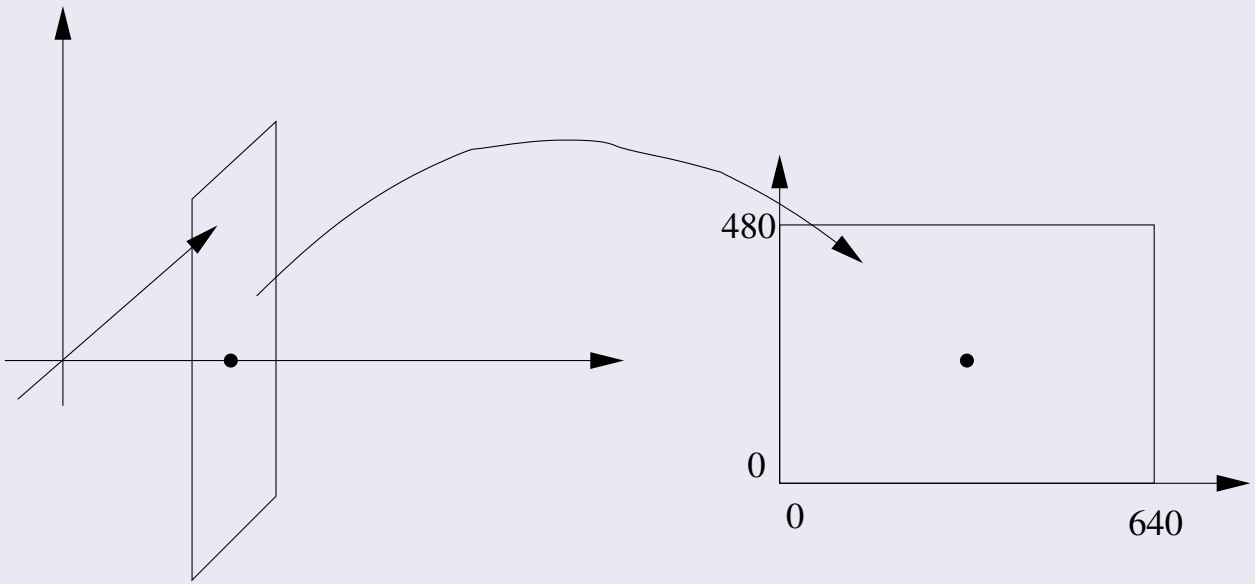
- Image point: 2D location on e.g. 36×24mm sensor
- First step of converting to sensor coordinates (in pixel)

The Inner Parameters - K

The principal point (x_0, y_0)

$$\begin{pmatrix} fx + x_0 \\ fy + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Re-centers the image. Typically transforms the point $(0, 0, 1)$ to the middle of the image.



The Inner Parameters - K

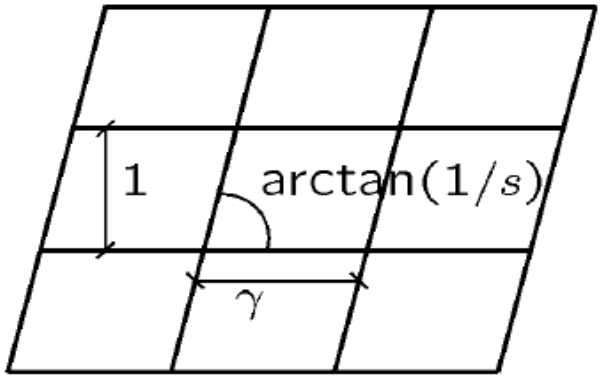
Aspect ratio

$$\begin{pmatrix} \gamma f x + x_0 \\ f y + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} \gamma f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Pixels are not always squares but can be rectangular (rare nowadays). In such cases the scaling in the x-direction should be different from the y-direction.

Skew

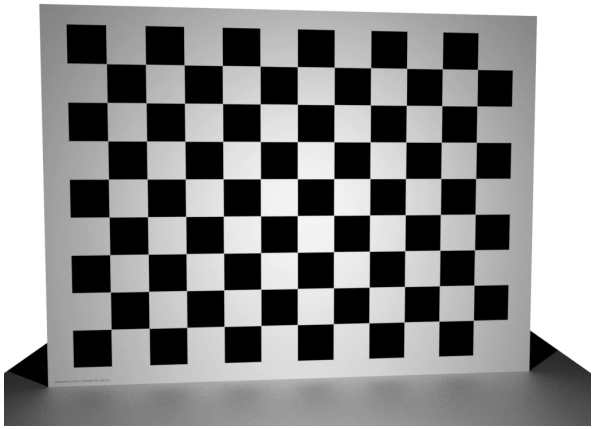
$$\begin{pmatrix} \gamma f & s f & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

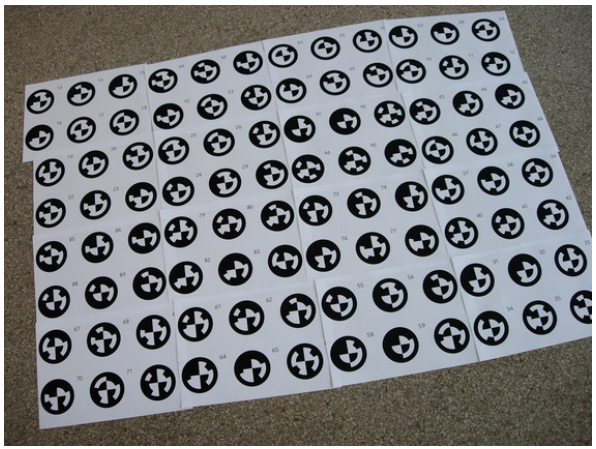


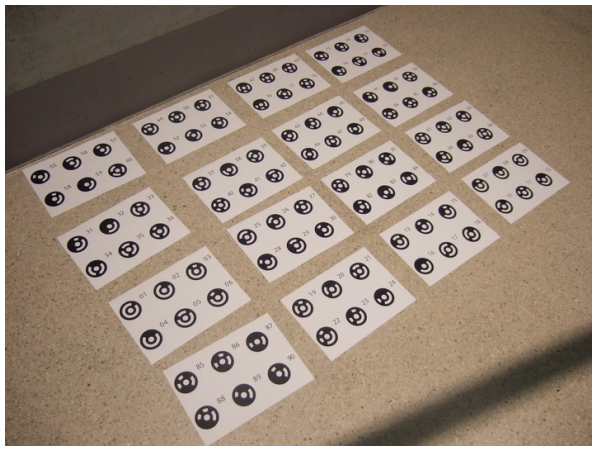
Corrects for tilted pixels. Typically zero.

How to obtain $\mathtt{K}$?

- Dedicated (off-line) camera calibration
  - Required for highest precision, e.g. photogrammetry







- Self-calibration
  - Find $\mathtt{H}$ such that $\mathtt{PH} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$
  - $\mathtt{K}$ is upper triangular and $\mathtt{R}$ is a rotation matrix
  - Numerically very sensitive
- Approximate $\mathtt{K}$ from EXIF meta-data
  - The approach you take for your project

How to obtain K?

- E.g. in a Linux terminal

```
$ exiftran -d kronan1.JPG | grep Focal
   0x920a  Focal Length                    30.0 mm
   0xa405  Focal Length in 35mm Film       45
$ exiftran -d kronan1.JPG | grep Dim
   0xa002  Pixel X Dimension               1936
   0xa003  Pixel Y Dimension               1296
```

- 30mm is the sensor-dependent focal length
- 45mm is the 35mm equivalent
- Focal length in pixels: $f = \text{X-Dimension} \times \frac{f_{35mm}}{35} \approx 2489$
- Calibration matrix

$$\mathrm{K} = \begin{pmatrix} f & 0 & \text{X-Dimension}/2 \\ 0 & f & \text{Y-Dimension}/2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2489 & 0 & 968 \\ 0 & 2489 & 648 \\ 0 & 0 & 1 \end{pmatrix}$$

- Attention: image viewing software may display auto-rotated picture

How to obtain K?

- E.g. in a Linux terminal

```
$ exiftran -d kronan1.JPG | grep Focal
   0x920a  Focal Length                    30.0 mm
   0xa405  Focal Length in 35mm Film       45
$ exiftran -d kronan1.JPG | grep Dim
   0xa002  Pixel X Dimension               1936
   0xa003  Pixel Y Dimension               1296
```

- 30mm is the sensor-dependent focal length
- 45mm is the 35mm equivalent
- Focal length in pixels: $f = \text{X-Dimension} \times \frac{f_{35mm}}{35} \approx 2489$
- Calibration matrix

$$\mathrm{K} = \begin{pmatrix} f & 0 & \text{X-Dimension}/2 \\ 0 & f & \text{Y-Dimension}/2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2489 & 0 & 968 \\ 0 & 2489 & 648 \\ 0 & 0 & 1 \end{pmatrix}$$

- Attention: image viewing software may display auto-rotated picture

How to obtain K?

- E.g. in a Linux terminal

```
$ exiftran -d kronan1.JPG | grep Focal
   0x920a  Focal Length                    30.0 mm
   0xa405  Focal Length in 35mm Film       45
$ exiftran -d kronan1.JPG | grep Dim
   0xa002  Pixel X Dimension               1936
   0xa003  Pixel Y Dimension               1296
```

- 30mm is the sensor-dependent focal length
- 45mm is the 35mm equivalent
- Focal length in pixels: $f = \text{X-Dimension} \times \frac{f_{35mm}}{35} \approx 2489$
- Calibration matrix

$$\mathtt{K} = \begin{pmatrix} f & 0 & \text{X-Dimension}/2 \\ 0 & f & \text{Y-Dimension}/2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2489 & 0 & 968 \\ 0 & 2489 & 648 \\ 0 & 0 & 1 \end{pmatrix}$$

- Attention: image viewing software may display auto-rotated picture

Lens Distortion

Not modeled by the K-matrix.
Cannot be removed by a projective mapping since lines are not mapped onto lines (see Szeliski, Section 2.1.5 on Lens distortions).

Lens Distortion

- Additional non-linearity after perspective division
- Forward distortion model (radially symmetric scaling)

$$\mathbf{x}_u = \pi(\mathtt{R}\mathbf{X} + \mathbf{T}) \qquad \text{ideal undistorted image point}$$

$$\mathbf{x}_d = \left(1 + \kappa_1 \left\|\mathbf{x}_u\right\|^2 + \kappa_2 \left\|\mathbf{x}_u\right\|^4\right) \mathbf{x}_u \qquad \text{distorted image point}$$

- Distortion coefficients κ_1 and κ_2
  - Estimated via calibration toolbox or refined in bundle adjustment
- Sensor coordinates (pixels): $\mathtt{K}\,\mathbf{x}_d$
- Forward model good for bundle adjustment
  - Explaining features in images as good as possible
- Forward model needs inversion to "undistort" images
  - Usually no closed form solution
  - Some distortion models have closed form expression in both directions

Lens Distortion

- Additional non-linearity after perspective division
- Forward distortion model (radially symmetric scaling)

$$\mathbf{x}_u = \pi(\mathtt{R}\mathbf{X} + \mathbf{T}) \qquad \text{ideal undistorted image point}$$

$$\mathbf{x}_d = \left(1 + \kappa_1 \left\|\mathbf{x}_u\right\|^2 + \kappa_2 \left\|\mathbf{x}_u\right\|^4\right) \mathbf{x}_u \qquad \text{distorted image point}$$

- Distortion coefficients κ_1 and κ_2
  - Estimated via calibration toolbox or refined in bundle adjustment
- Sensor coordinates (pixels): $\mathtt{K}\,\mathbf{x}_d$
- Forward model good for bundle adjustment
  - Explaining features in images as good as possible
- Forward model needs inversion to "undistort" images
  - Usually no closed form solution
  - Some distortion models have closed form expression in both directions

Factorizing $\mathtt{P}$

- Given $\mathtt{P} \in \mathbb{R}^{3\times 4}$, how can we explicitly write it as

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})?$$

  - $\mathtt{K}$ is upper triangular
  - $\mathtt{R}$ is a rotation matrix

Simpler case

How can we compute $\mathtt{K}$ and $\mathtt{R}$ if we assume $\mathtt{K}$ is a diagonal matrix?

Factorizing P

- Given $\mathtt{P} \in \mathbb{R}^{3\times 4}$, how can we explicitly write it as

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})?$$

  - $\mathtt{K}$ is upper triangular
  - $\mathtt{R}$ is a rotation matrix

Simpler case

How can we compute $\mathtt{K}$ and $\mathtt{R}$ if we assume $\mathtt{K}$ is a diagonal matrix?

Factorizing P

- Task: decompose $\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$ with $\mathtt{K} = \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{pmatrix}$
- $\mathtt{P}_{1:3,1:3} = \mathtt{KR} = \begin{pmatrix} f_1 \mathtt{R}_{1,:} \\ f_2 \mathtt{R}_{2,:} \\ f_3 \mathtt{R}_{3,:} \end{pmatrix}$, where $\mathtt{R}_{k,:}$ is the k-th row of $\mathtt{R}$
- Each $\mathtt{R}_{k,:}$ has unit length $\implies f_k = \|\mathtt{P}_{k,1:3}\|$ and $\mathtt{R}_{k,:} = \mathtt{P}_{k,1:3}/f_k$
- $\mathtt{K}_{3,3}$ should be 1, so set

$$\mathtt{K} = \begin{pmatrix} f_1/f_3 & 0 & 0 \\ 0 & f_2/f_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- How to get $\mathbf{T}$?

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathtt{P}_{:,4} = \mathtt{K}\mathbf{T} \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_{:,4}$$

Factorizing P

- Task: decompose $\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$ with $\mathtt{K} = \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{pmatrix}$
- $\mathtt{P}_{1:3,1:3} = \mathtt{KR} = \begin{pmatrix} f_1 \mathtt{R}_{1,:} \\ f_2 \mathtt{R}_{2,:} \\ f_3 \mathtt{R}_{3,:} \end{pmatrix}$, where $\mathtt{R}_{k,:}$ is the k-th row of $\mathtt{R}$
- Each $\mathtt{R}_{k,:}$ has unit length $\implies f_k = \|\mathtt{P}_{k,1:3}\|$ and $\mathtt{R}_{k,:} = \mathtt{P}_{k,1:3}/f_k$
- $\mathtt{K}_{3,3}$ should be 1, so set

$$\mathtt{K} = \begin{pmatrix} f_1/f_3 & 0 & 0 \\ 0 & f_2/f_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- How to get $\mathbf{T}$?

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathtt{P}_{:,4} = \mathtt{K}\mathbf{T} \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_{:,4}$$

Factorizing P

- Task: decompose $\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$ with $\mathtt{K} = \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{pmatrix}$
- $\mathtt{P}_{1:3,1:3} = \mathtt{KR} = \begin{pmatrix} f_1 \mathtt{R}_{1,:} \\ f_2 \mathtt{R}_{2,:} \\ f_3 \mathtt{R}_{3,:} \end{pmatrix}$, where $\mathtt{R}_{k,:}$ is the k-th row of $\mathtt{R}$
- Each $\mathtt{R}_{k,:}$ has unit length $\implies$ $f_k = \|\mathtt{P}_{k,1:3}\|$ and $\mathtt{R}_{k,:} = \mathtt{P}_{k,1:3}/f_k$
- $\mathtt{K}_{3,3}$ should be 1, so set

$$\mathtt{K} = \begin{pmatrix} f_1/f_3 & 0 & 0 \\ 0 & f_2/f_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- How to get $\mathbf{T}$?

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathtt{P}_{:,4} = \mathtt{K}\mathbf{T} \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_{:,4}$$

Factorizing P

- Task: decompose $\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$ with $\mathtt{K} = \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{pmatrix}$
- $\mathtt{P}_{1:3,1:3} = \mathtt{KR} = \begin{pmatrix} f_1 \mathtt{R}_{1,:} \\ f_2 \mathtt{R}_{2,:} \\ f_3 \mathtt{R}_{3,:} \end{pmatrix}$, where $\mathtt{R}_{k,:}$ is the k-th row of $\mathtt{R}$
- Each $\mathtt{R}_{k,:}$ has unit length $\implies$ $f_k = \|\mathtt{P}_{k,1:3}\|$ and $\mathtt{R}_{k,:} = \mathtt{P}_{k,1:3}/f_k$
- $\mathtt{K}_{3,3}$ should be 1, so set

$$\mathtt{K} = \begin{pmatrix} f_1/f_3 & 0 & 0 \\ 0 & f_2/f_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- How to get $\mathbf{T}$?

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathtt{P}_{:,4} = \mathtt{K}\mathbf{T} \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_{:,4}$$

Factorizing P

- Task: decompose $\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T})$ with $\mathtt{K} = \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{pmatrix}$
- $\mathtt{P}_{1:3,1:3} = \mathtt{KR} = \begin{pmatrix} f_1 \mathtt{R}_{1,:} \\ f_2 \mathtt{R}_{2,:} \\ f_3 \mathtt{R}_{3,:} \end{pmatrix}$, where $\mathtt{R}_{k,:}$ is the k-th row of $\mathtt{R}$
- Each $\mathtt{R}_{k,:}$ has unit length $\implies$ $f_k = \|\mathtt{P}_{k,1:3}\|$ and $\mathtt{R}_{k,:} = \mathtt{P}_{k,1:3}/f_k$
- $\mathtt{K}_{3,3}$ should be 1, so set

$$\mathtt{K} = \begin{pmatrix} f_1/f_3 & 0 & 0 \\ 0 & f_2/f_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- How to get $\mathbf{T}$?

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathtt{P}_{:,4} = \mathtt{K}\mathbf{T} \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_{:,4}$$

General $\mathtt{K}$: RQ-factorization

Theorem

If $\mathtt{A}$ is an $n \times n$ matrix then there is an orthogonal matrix $\mathtt{Q}$ and an upper triangular matrix $\mathtt{R}$ such that $\mathtt{A} = \mathtt{RQ}$.

(If $\mathtt{A}$ is invertible and the diagonal elements are chosen the be positive, then the factorization is unique.)

- Related to better known QR-factorization
- We use $\mathtt{K}$ for the upper triangular matrix and $\mathtt{R}$ for the rotation
- Given projection matrix $\mathtt{P}$, how to compute RQ-decomposition $\mathtt{P}_{33} = \mathtt{KR}$?
  - Gram-Schmidt orthogonalization, see e.g. lecture notes
  - $\mathtt{P} = (\mathtt{P}_{33} \mid \mathtt{P}_4) = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_4$

General $\mathtt{K}$: RQ-factorization

Theorem

If $\mathtt{A}$ is an $n \times n$ matrix then there is an orthogonal matrix $\mathtt{Q}$ and an upper triangular matrix $\mathtt{R}$ such that $\mathtt{A} = \mathtt{RQ}$.

(If $\mathtt{A}$ is invertible and the diagonal elements are chosen the be positive, then the factorization is unique.)

- Related to better known QR-factorization
- We use $\mathtt{K}$ for the upper triangular matrix and $\mathtt{R}$ for the rotation
- Given projection matrix $\mathtt{P}$, how to compute RQ-decomposition $\mathtt{P}_{33} = \mathtt{KR}$?
  - Gram-Schmidt orthogonalization, see e.g. lecture notes
  - $\mathtt{P} = (\mathtt{P}_{33} \mid \mathtt{P}_4) = \mathtt{K}(\mathtt{R} \mid \mathbf{T}) = (\mathtt{KR} \mid \mathtt{K}\mathbf{T}) \implies \mathbf{T} = \mathtt{K}^{-1}\mathtt{P}_4$

Projective vs. Euclidean Reconstruction

Calibrated Cameras

A camera

$$\mathtt{P} = \mathtt{K}(\mathtt{R} \mid \mathbf{T}),$$

where the calibration matrix/inner parameters $\mathtt{K}$ are known is called calibrated. If we change coordinates in the image using

$$\tilde{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x},$$

we get a so called normalized (calibrated) camera

$$\tilde{\mathbf{x}} = \mathtt{K}^{-1}\mathtt{K}(\mathtt{R} \mid \mathbf{T})\mathbf{X} = (\mathtt{R} \mid \mathbf{T})\mathbf{X}.$$

Projective vs. Euclidean Reconstruction

Projective

The reconstruction is determined up to a projective transformation.
If $\lambda \mathbf{x} = \mathtt{P}\mathbf{X}$, then for any projective transformation

$$\tilde{\mathbf{X}} = \mathtt{H}^{-1}\mathbf{X}$$

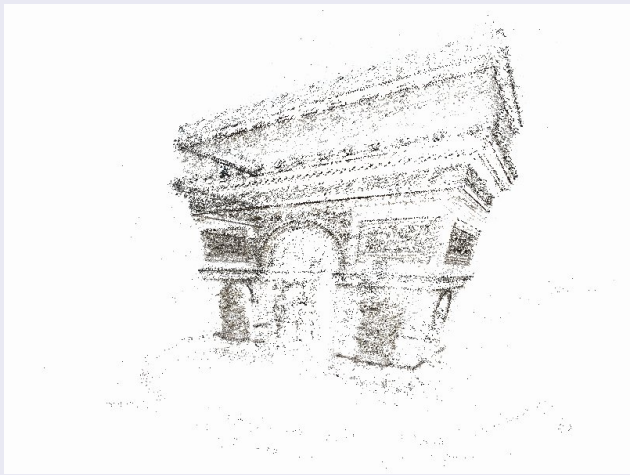
we have

$$\lambda \mathbf{x} = \mathtt{P}\mathtt{H}\mathtt{H}^{-1}\mathbf{X} = (\mathtt{P}\mathtt{H})(\mathtt{H}^{-1}\mathbf{X}) = \tilde{\mathtt{P}}\tilde{\mathbf{X}},$$

where $\tilde{\mathtt{P}} = \mathtt{P}\mathtt{H}$ is also a valid camera.

Projective vs. Euclidean Reconstruction

Projective



Euclidean

Arc de triomphe, Paris

The reconstructions have exactly the same reprojections in the original images. But the projective coordinate system makes things look strange.

(Projective reconstructions can look really weird. The plane at infinity can be in the scene and separate the 3D model.)

Discussion

- What properties are lost in a projective 3D reconstruction due to projective distortions?
- How can we upgrade a projective 3D reconstruction to a Euclidean one? (Hint: This can be done in many different ways.)

Projective vs. Euclidean Reconstruction

Euclidean

The reconstruction is determined up to a similarity transformation.
If $\lambda \mathbf{x} = (\mathtt{R} \mid \mathbf{T})\mathbf{X}$, then for any similarity transformation

$$\tilde{\mathbf{X}} = \begin{pmatrix} s\mathtt{Q} & \mathbf{v} \\ \mathbf{0}^\top & 1 \end{pmatrix}^{-1} \mathbf{X}$$

we have

$$\frac{\lambda}{s}\mathbf{x} = (\mathtt{R} \mid \mathbf{T}) \begin{pmatrix} \mathtt{Q} & \mathbf{v}/s \\ \mathbf{0}^\top & 1/s \end{pmatrix} \tilde{\mathbf{X}} = \left(\mathtt{RQ} \mid \tfrac{1}{s}\left(\mathtt{R}\mathbf{v} + \mathbf{T}\right)\right) \tilde{\mathbf{X}}.$$

Since $\mathtt{RQ}$ is a rotation this is a normalized camera.

- You have to scale the depth to preserve orthogonality of $\mathtt{RQ}$.

Discussion

- For calibrated cameras, we do not have the projective ambiguity/distortions. Why?
- How can we upgrade a Euclidean 3D reconstruction with unknown scale to a metric one (that is, where the unit is metres)? (Hint: This can be done in many different ways.)

Direct Linear Transformation – (Simplified) DLT

Finding the camera matrix (camera resectioning)

Use images of a known object to eliminate the projective ambiguity. If $\mathbf{X}_i$ are 3d-points of a known object, and $\mathbf{x}_i$ corresponding projections we have

$$\begin{aligned} \lambda_1 \mathbf{x}_1 &= \mathtt{P}\mathbf{X}_1 \\ \lambda_2 \mathbf{x}_2 &= \mathtt{P}\mathbf{X}_2 \\ &\vdots \\ \lambda_N \mathbf{x}_N &= \mathtt{P}\mathbf{X}_N. \end{aligned}$$

There are $3N$ equations and $11 + N$ unknowns. We need $3N \geq 11 + N \Rightarrow N \geq 6$ points to solve the problem.

Direct Linear Transformation – (Simplified) DLT

Matrix Formulation

$$\mathtt{P} = \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix}$$

where $\mathbf{p}_i$ are the rows of $\mathtt{P}$ The first equation is

$$\begin{aligned} \mathbf{X}_1^\top \mathbf{p}_1 - \lambda_1 x_1 &= 0 \\ \mathbf{X}_1^\top \mathbf{p}_2 - \lambda_1 y_1 &= 0 \\ \mathbf{X}_1^\top \mathbf{p}_3 - \lambda_1 &= 0, \end{aligned}$$

where $\mathbf{x}_1 = (x_1, y_1, 1)$. In matrix form this is

$$\begin{bmatrix} \mathbf{X}_1^\top & 0 & 0 & -x_1 \\ 0 & \mathbf{X}_1^\top & 0 & -y_1 \\ 0 & 0 & \mathbf{X}_1^\top & -1 \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Direct Linear Transformation – (Simplified) DLT

Matrix Formulation

More equations:

$$\underbrace{\begin{pmatrix} \mathbf{X}_1^\top & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \mathbf{X}_1^\top & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & \mathbf{X}_1^\top & -1 & 0 & 0 & \dots \\ \mathbf{X}_2^\top & 0 & 0 & 0 & -x_2 & 0 & \dots \\ 0 & \mathbf{X}_2^\top & 0 & 0 & -y_2 & 0 & \dots \\ 0 & 0 & \mathbf{X}_2^\top & 0 & -1 & 0 & \dots \\ \mathbf{X}_3^\top & 0 & 0 & 0 & 0 & -x_3 & \dots \\ 0 & \mathbf{X}_3^\top & 0 & 0 & 0 & -y_3 & \dots \\ 0 & 0 & \mathbf{X}_3^\top & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}}_{=\mathtt{M}} \underbrace{\begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix}}_{=\mathbf{v}} = \mathbf{0}.$$

Direct Linear Transformation – (Simplified) DLT

- We want to solve a (potentially overcomplete) linear system $\mathbf{M}\mathbf{v} = \mathbf{0}$
- What if there is no solution?
  - Image points $\mathbf{x}_i$ and 3D points $\mathbf{X}_i$ will not be 100% accurate
  - Noise in 3D measurements
  - Limited resolution in extracting image points
  - Limited precision in real number representation
- We can ask for $\mathbf{v}$ such that $\|\mathbf{M}\mathbf{v}\|^2$ is minimal
  - But we want to avoid degenerate solution $\mathbf{v} = \mathbf{0}$
  - Introduce constraint $\|\mathbf{v}\| \geq 1$ to keep $\mathbf{v}$ away from $\mathbf{0}$

Direct Linear Transformation – (Simplified) DLT

- We want to solve a (potentially overcomplete) linear system $\mathtt{M}\mathbf{v} = \mathbf{0}$
- What if there is no solution?
  - Image points $\mathbf{x}_i$ and 3D points $\mathbf{X}_i$ will not be 100% accurate
  - Noise in 3D measurements
  - Limited resolution in extracting image points
  - Limited precision in real number representation
- We can ask for $\mathbf{v}$ such that $\|\mathtt{M}\mathbf{v}\|^2$ is minimal
  - But we want to avoid degenerate solution $\mathbf{v} = \mathbf{0}$
  - Introduce constraint $\|\mathbf{v}\| \geq 1$ to keep $\mathbf{v}$ away from $\mathbf{0}$

Direct Linear Transformation – (Simplified) DLT

Homogeneous Least Squares

Find solution of

$$\min_{\mathbf{v}:\|\mathbf{v}\|\geq 1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|=1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|^2=1} \|\mathtt{M}\mathbf{v}\|^2 .$$

- First order optimality conditions

$$\mathtt{M}^\top \mathtt{M}\mathbf{v}^* = \mu\mathbf{v}^*$$

  for a Lagrange multiplier $\mu \in \mathbb{R}$
- Therefore $\mathbf{v}^*$ is an eigenvector of $\mathtt{M}^\top\mathtt{M}$ with eigenvalue μ
- Pick eigenvector corresponding to the smallest eigenvalue of $\mathtt{M}^\top\mathtt{M}$ to minimize $\|\mathtt{M}\mathbf{v}\|^2$
- Eigenvalue decomposition of $\mathtt{M}^\top\mathtt{M}$ may be numerically unstable
  $\Longrightarrow$ SVD is preferable

Direct Linear Transformation – (Simplified) DLT

Homogeneous Least Squares

Find solution of

$$\min_{\mathbf{v}:\|\mathbf{v}\|\geq 1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|=1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|^2=1} \|\mathtt{M}\mathbf{v}\|^2 .$$

- First order optimality conditions

$$\mathtt{M}^\top \mathtt{M}\mathbf{v}^* = \mu\mathbf{v}^*$$

  for a Lagrange multiplier $\mu \in \mathbb{R}$

- Therefore $\mathbf{v}^*$ is an eigenvector of $\mathtt{M}^\top\mathtt{M}$ with eigenvalue μ
- Pick eigenvector corresponding to the smallest eigenvalue of $\mathtt{M}^\top\mathtt{M}$ to minimize $\|\mathtt{M}\mathbf{v}\|^2$
- Eigenvalue decomposition of $\mathtt{M}^\top\mathtt{M}$ may be numerically unstable
  $\Longrightarrow$ SVD is preferable

Direct Linear Transformation – (Simplified) DLT

Homogeneous Least Squares

Find solution of

$$\min_{\mathbf{v}:\|\mathbf{v}\|\geq 1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|=1} \|\mathtt{M}\mathbf{v}\|^2 = \min_{\mathbf{v}:\|\mathbf{v}\|^2=1} \|\mathtt{M}\mathbf{v}\|^2 .$$

- First order optimality conditions

$$\mathtt{M}^\top \mathtt{M}\mathbf{v}^* = \mu\mathbf{v}^*$$

  for a Lagrange multiplier $\mu \in \mathbb{R}$
- Therefore $\mathbf{v}^*$ is an eigenvector of $\mathtt{M}^\top\mathtt{M}$ with eigenvalue μ
- Pick eigenvector corresponding to the smallest eigenvalue of $\mathtt{M}^\top\mathtt{M}$ to minimize $\|\mathtt{M}\mathbf{v}\|^2$
- Eigenvalue decomposition of $\mathtt{M}^\top\mathtt{M}$ may be numerically unstable
  $\implies$ SVD is preferable

Singular value decomposition

Theorem

Each $m \times n$ matrix $\mathtt{M}$ (with real coefficients) can be factorized into

$$\mathtt{M} = \mathtt{U}\mathtt{S}\mathtt{V}^\top,$$

where $\mathtt{U}$ and $\mathtt{V}$ are orthogonal ($m \times m$ and $n \times n$ respectively),

$$\mathtt{S} = \begin{pmatrix} \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_r) & 0 \\ 0 & 0 \end{pmatrix}.$$

$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ and r is the rank of the matrix.

- Solving $\min_{\mathbf{v}:\|\mathbf{v}\|=1} \|\mathtt{M}\mathbf{v}\|^2$
  - Compute SVD of $\mathtt{M} = \mathtt{U}\Sigma\mathtt{V}^\top$
  - Optimal solution $\mathbf{v}^*$ is last column of $\mathtt{V}$
- See lecture notes

Direct Linear Transformation – (Simplified) DLT

Improving the Numerics (Normalization of uncalibrated cameras)

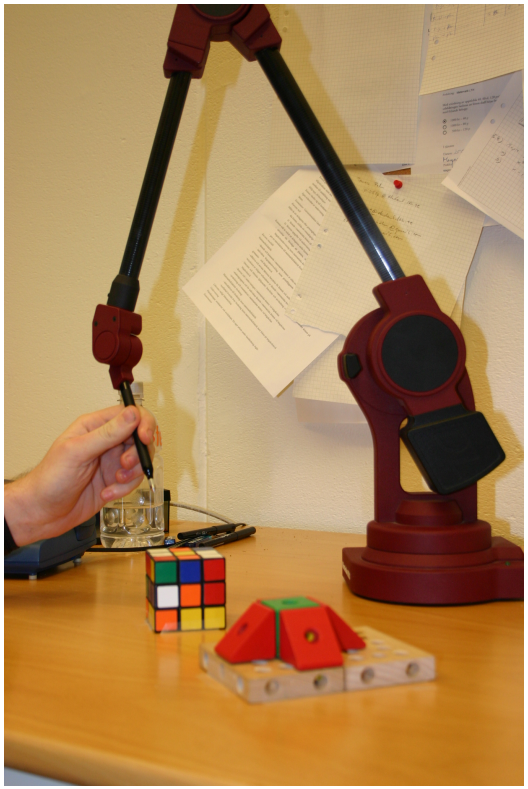
The matrix contains entries x_i, y_j and ones. Since x_i and y_i can be about a thousand, the numerics are often greatly improved by translating the coordinates such that their "center of mass" is zero and then rescaling the coordinates to be roughly 1.

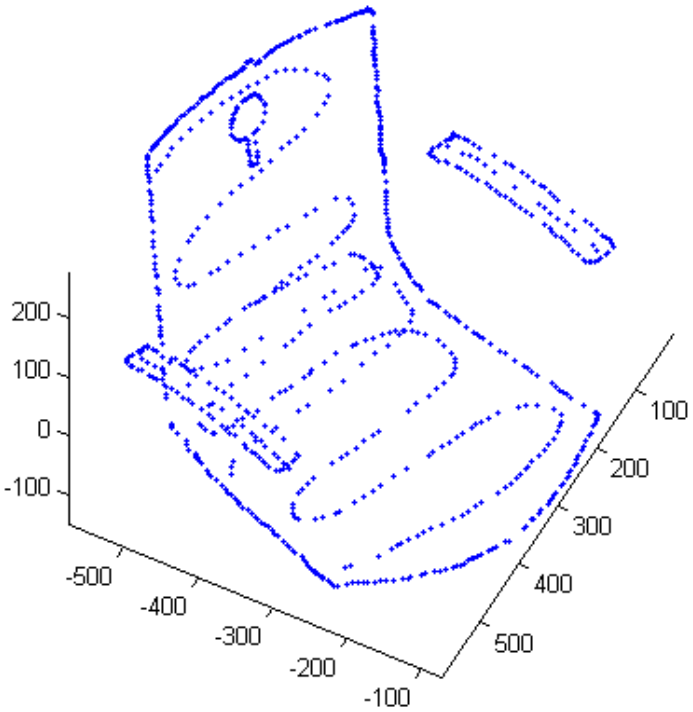
- Change coordinates according to

$$\tilde{\mathbf{x}} = \begin{bmatrix} s & 0 & -s\bar{x} \\ 0 & s & -s\bar{y} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}.$$

- Solve the homogeneous linear least squares system and transform back to the original coordinate system.
- Similar transformations for the 3D-points $\mathbf{X}_i$ may also improve the results.

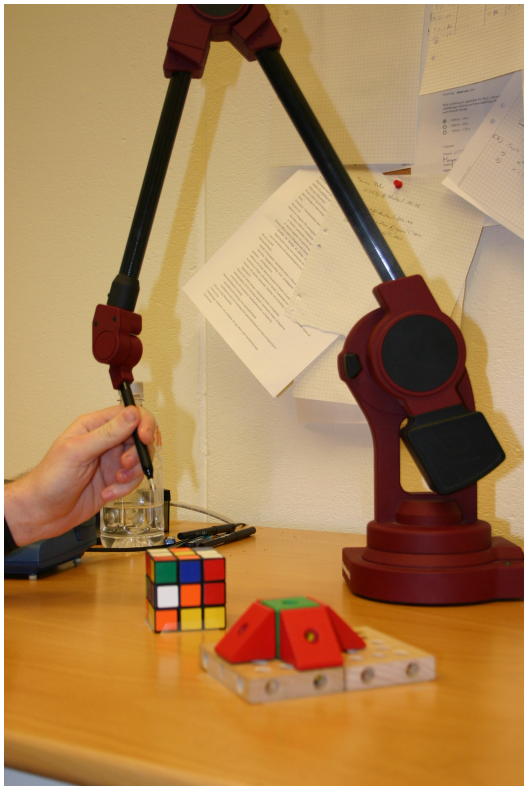
Pose estimation using DLT

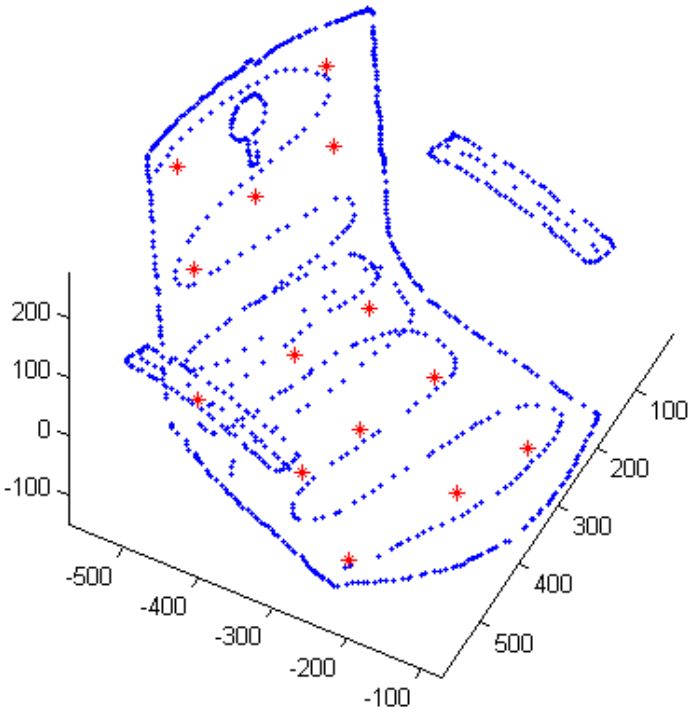


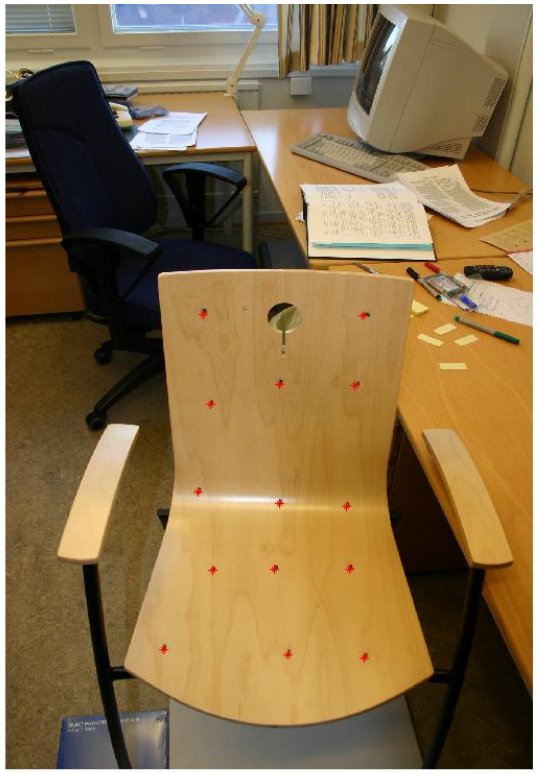


3D points measured using scanning arm.

Pose estimation using DLT

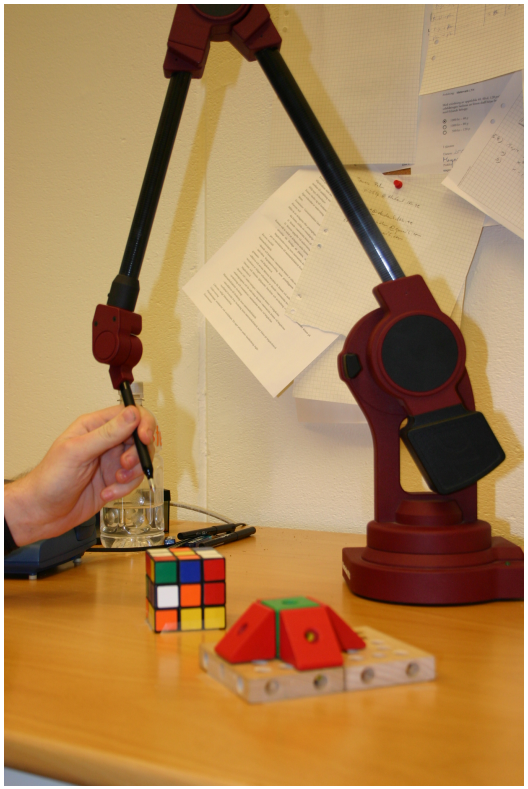


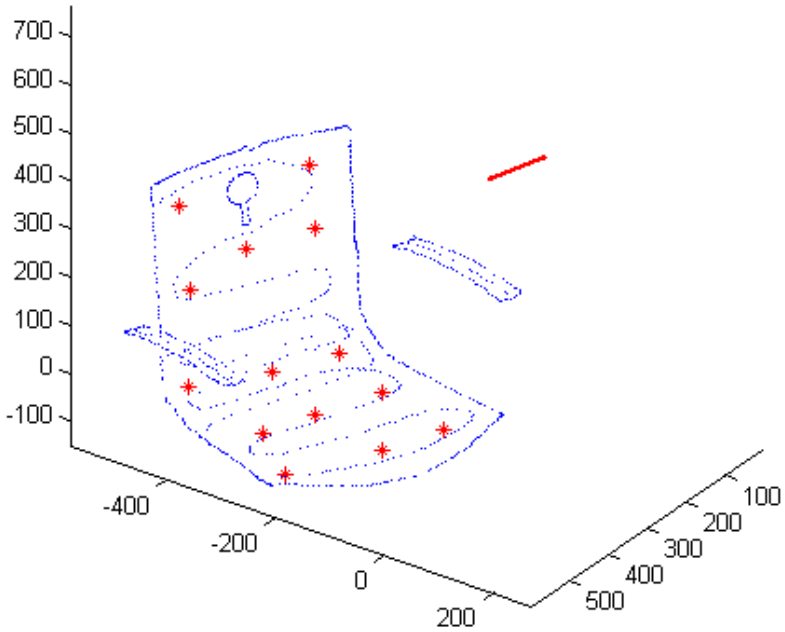


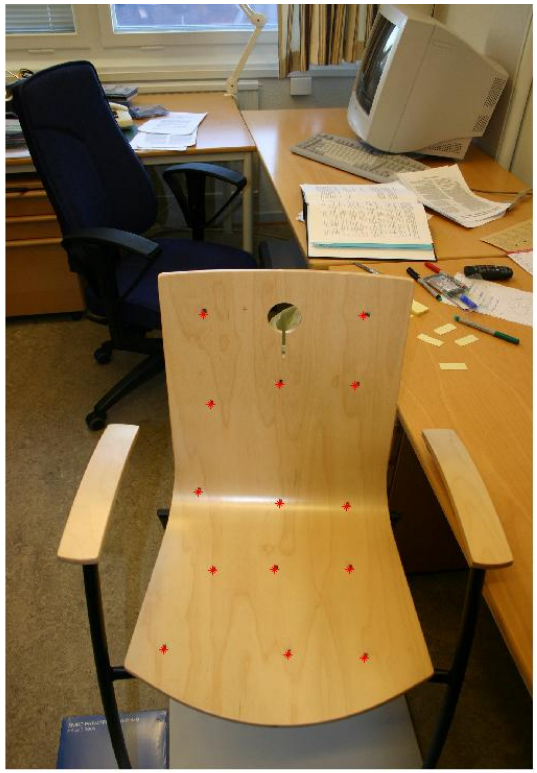


14 points used for computing the camera matrix.

Pose estimation using DLT

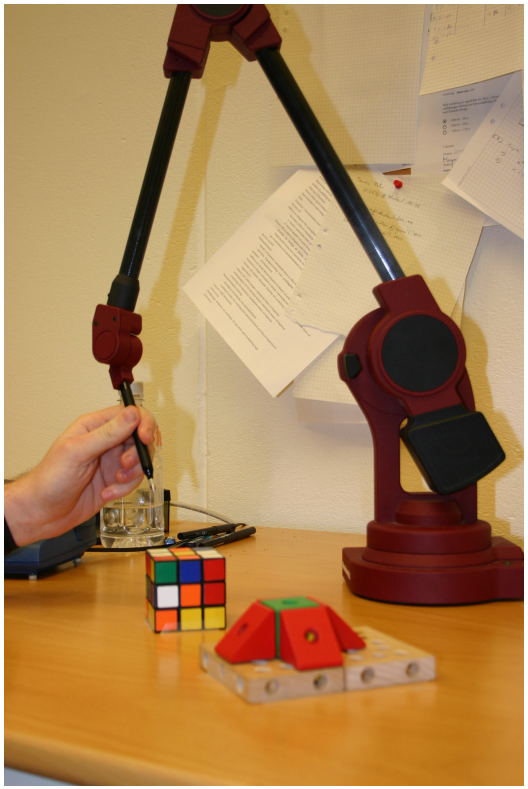


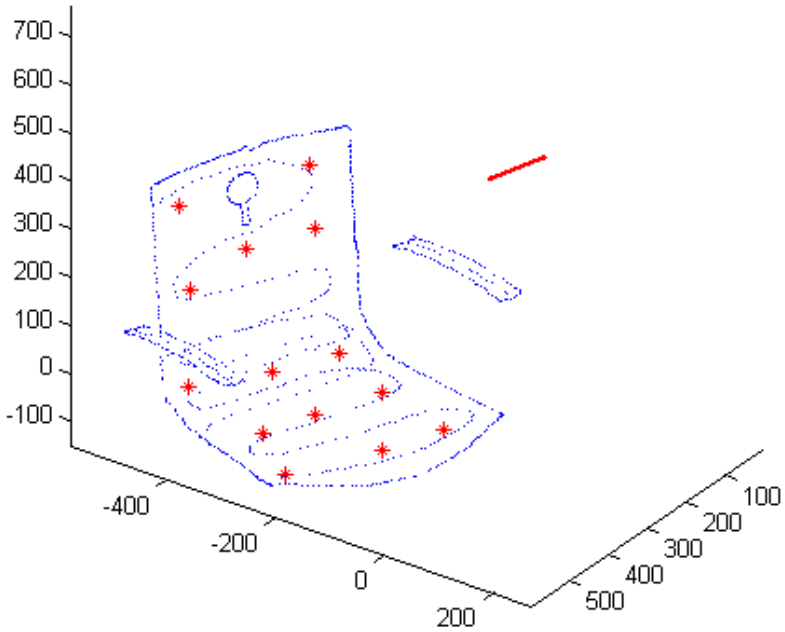


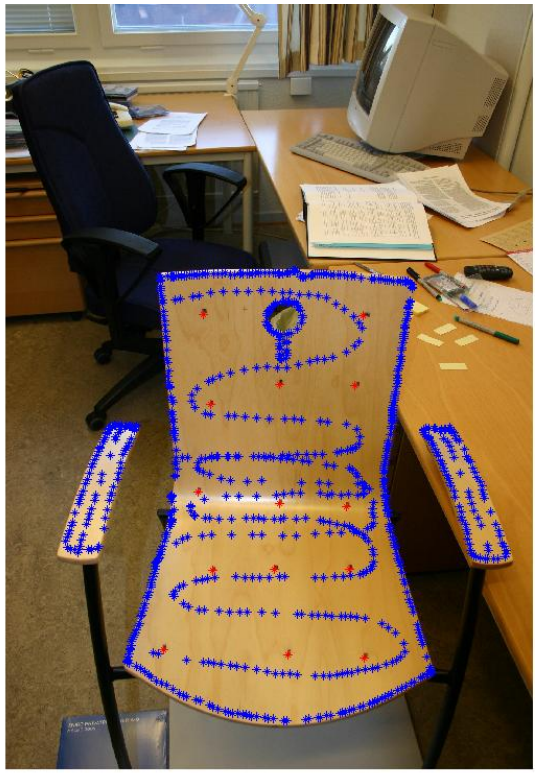


14 points used for computing the camera matrix.

Texturing the chair

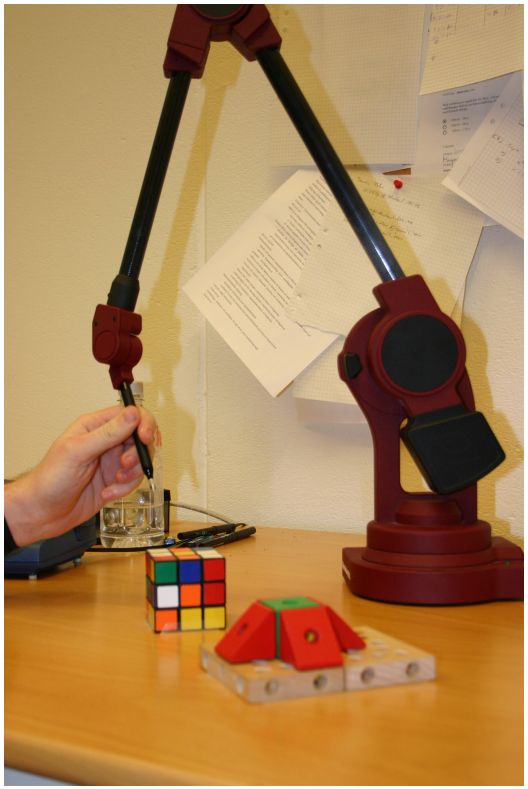


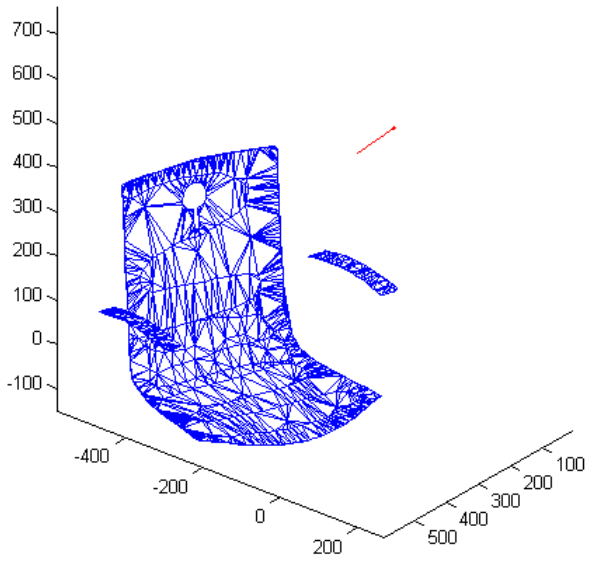


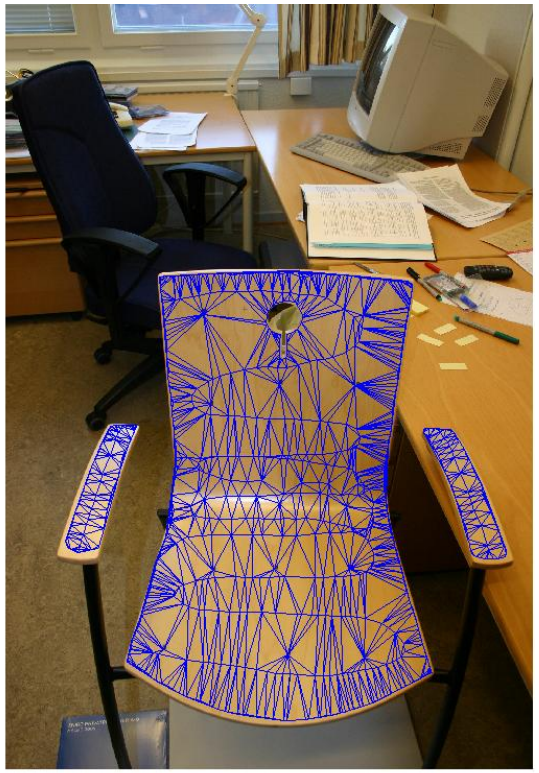


Project the rest of the points into the image.

Texturing the chair

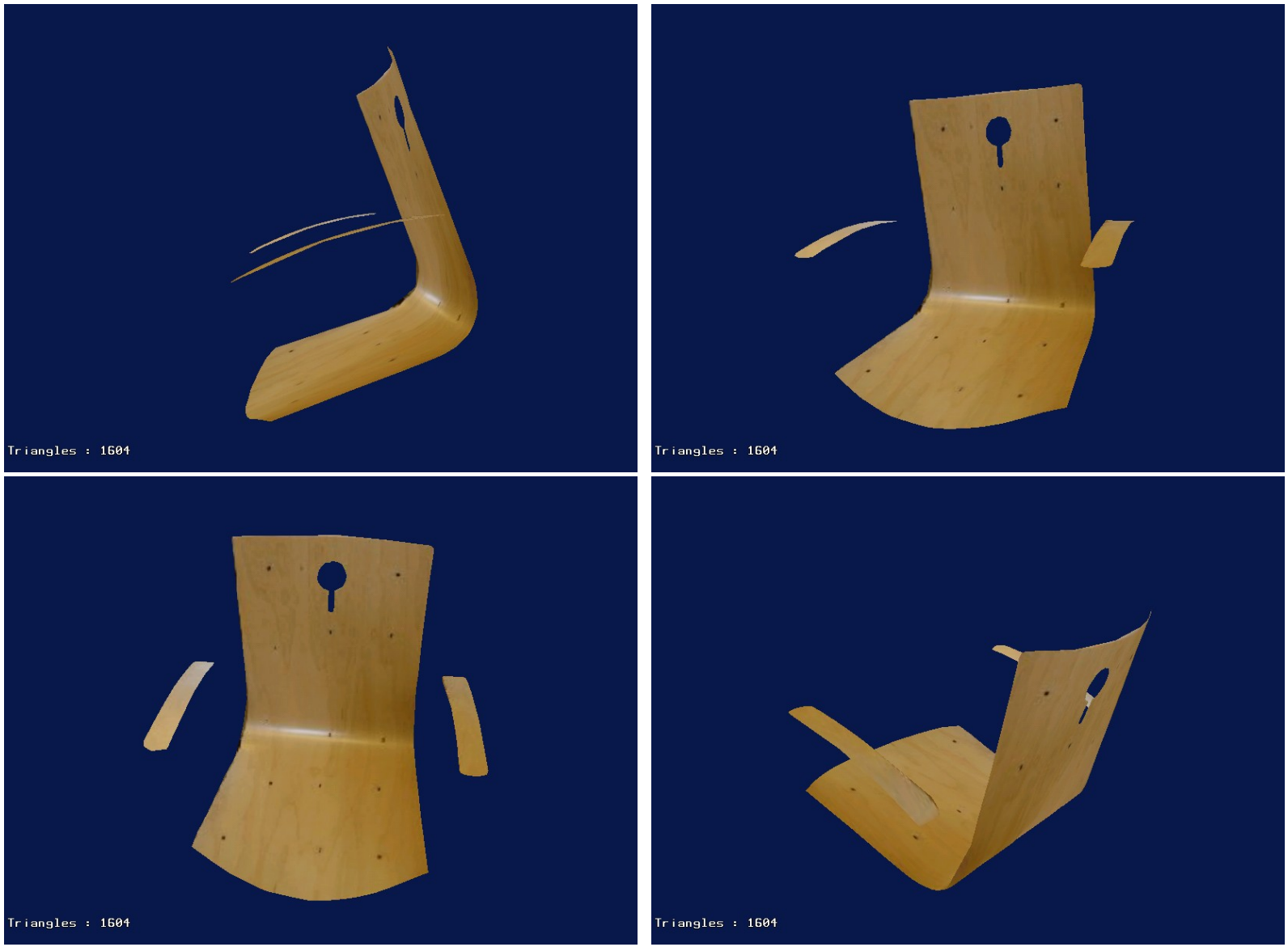






Form triangles. Use the texture from the image.

Textured chair



Direct Linear Transformation – (Simplified) DLT

Disadvantages of the simplified DLT

- Need to compute SVD of $3N \times (12 + N)$ matrix
  - Okay for $N \approx 20$, but what if $N = 10000$?
- Standard DLT discussed next time: we eliminate all λ_i
  - Now compute SVD of $2N \times 12$ matrix
  - Or eigendecomposition of 12×12 matrix
- Standard DLT also applicable to other problems

Advantages of the simplified DLT

- You obtain camera matrix $\mathtt{P}$ and all depths λ_i in one step

What about the calibrated setup?

- Known $\mathtt{K}$, use $\tilde{\mathbf{x}}_i = \mathtt{K}^{-1}\mathbf{x}_i$
- $\mathtt{P} = (\mathtt{R} \mid \mathbf{T})$ has 6 d.o.f. (why?)
- How to leverage/enforce additional constraints?
- Use constraints and use ≥ 3 correspondences (P3P)
- Use DLT on $\tilde{\mathbf{x}}_i \leftrightarrow \mathbf{X}_i$ to obtain $\hat{\mathtt{P}} = (\hat{\mathtt{R}} \mid \hat{\mathbf{T}})$
  - Find $\mathtt{P} = (\mathtt{R} \mid \mathbf{T})$ close to $\hat{\mathtt{P}}$ such that $\mathtt{R}$ is a rotation matrix
  - E.g. $\mathtt{R} = \mathtt{U}\mathtt{V}^\top$ where $\hat{\mathtt{R}} = \mathtt{U}\mathtt{\Sigma}\mathtt{V}^\top$
  - Optionally followed by local optimization (week 5)

To do

- Work on assignment 1
- Lab session after this lecture: E-D2480, ES61, ES62 & ES63

More reading:

- Szeliski, Section 2.1.5 on Lens distortions.
- Szeliski, Section 7.1 on Feature detection and matching.