電腦視覺與應用
Computer Vision and Applications

Lecture07-Camera calibration

Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology

CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

Camera calibration

校正

- projection matrix
  - perspective projection $\Rightarrow [[P]]X_{3D} = x_{2D}$ (12 unknown)
  - orthographic projection $\Rightarrow [\begin{matrix} P \\ 0001 \end{matrix}] x_{3D} = x_{2D}$ (8)

telephoto

[ ] 表 pick up points

Intrinsic parameter

① system calibration
- projection
- Cam, 3D, 3D (mm), 2D (pixel)

② Camera Calibration
- k, distance

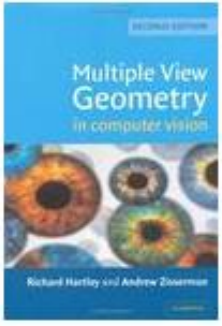
After Calib
- Intrinic parameter
- $k \Rightarrow$ [lens distortion]

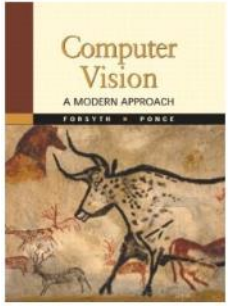
a. Camera => itself
- Intrinic, lens

b. system focus
- Extrinic
- Intrinic

Camera calibration

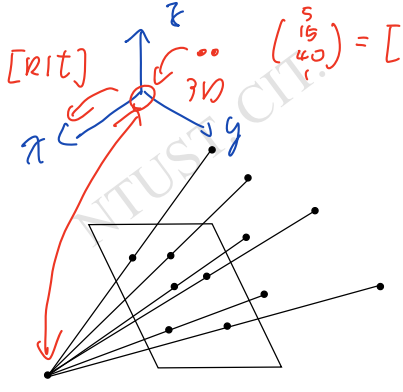
- Lecture Reference at:
  - Multiple View Geometry in Computer Vision, (Chapter7)
  - Computer Vision A Modern Approach, Chapter 3 (Geometric Camera Calibration).



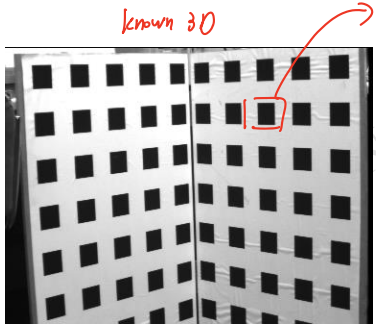


Camera calibration

- Problem description:
  - Camera calibration is to use precise geometric structure, then to adjust the camera parameter under measurements and constraints.
  - In short, given $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$
  - then determine the mapping transformation



$$\begin{pmatrix} 5 \\ 15 \\ 40 \\ 1 \end{pmatrix} = [R|t] \begin{bmatrix} 10 \\ 20 \\ 50 \\ 1 \end{bmatrix} \Rightarrow 3D$$



Camera calibration: A linear approach

- The camera mathematical model:

$$\mathbf{x}_{\text{img}} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}_{\text{world}}$$

can know

[4] opencv [5] Matlab

- To determine the mapping matrix (3x4), rewrite as

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

3 X 4 Matrix

6 - DoF => 6 unknow

3 X 4 matrix (11 unknown)

Camera model

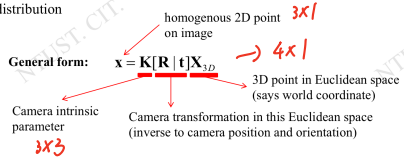
Requirement for forming a image (from geometrical viewpoint)

For Graphics model (or ideal pinhole)

- Extrinsic parameter
- Intrinsic parameter → perspective projection (in most cases, otherwise orthographical projection)

For real Cameras

- Extrinsic parameter
- Intrinsic parameter → perspective (with camera calibration matrix)
- Lens distortion distribution



Camera calibration: A linear approach—cont.

- To determine P in $\mathbf{x} = \mathbf{P} \cdot \mathbf{X}$

3x4 mapping matrix

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{x}_i = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix}_{3\times 4} \cdot \mathbf{X}_i$$

1x4 vector, 1x4 vector, 1x4 vector

$\mathbf{X}_i = [X_{i1} \quad X_{i2} \quad X_{i3} \quad X_{i4}]^T$: i-th 3D point in space

$\mathbf{x}_i = [u_i \quad v_i \quad 1]^T$: i-th 2D point on image

inner product (dot) of two vectors

$$\begin{cases} u_i = \dfrac{\mathbf{p}_1 \cdot \mathbf{X}_i}{\mathbf{p}_3 \cdot \mathbf{X}_i} \\ v_i = \dfrac{\mathbf{p}_2 \cdot \mathbf{X}_i}{\mathbf{p}_3 \cdot \mathbf{X}_i} \end{cases} \rightarrow \begin{cases} u_i \mathbf{p}_3 \cdot \mathbf{X}_i = \mathbf{p}_1 \cdot \mathbf{X}_i \\ v_i \mathbf{p}_3 \cdot \mathbf{X}_i = \mathbf{p}_2 \cdot \mathbf{X}_i \end{cases} \rightarrow \begin{cases} (\mathbf{p}_1 - u_i \mathbf{p}_3) \cdot \mathbf{X}_i = 0 \\ (\mathbf{p}_2 - v_i \mathbf{p}_3) \cdot \mathbf{X}_i = 0 \end{cases}$$

4x1 vector

2 eqs

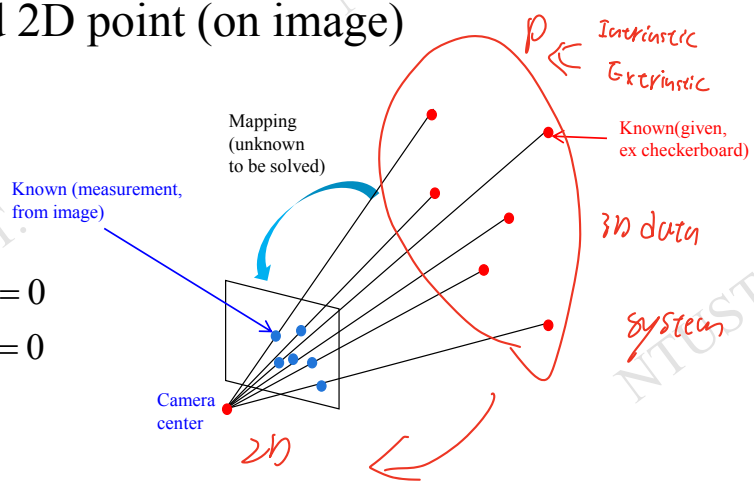
$\Rightarrow x = Px$

Camera calibration: A linear approach—cont.

- Again, one feature has two constraints.
- Note : one feature means that you have already known a 3D point (in 3D space) and a projected 2D point (on image)

$$\begin{cases} (\mathbf{p}_1 - u_i\mathbf{p}_3)\cdot\mathbf{X}_i = 0 \\ (\mathbf{p}_2 - v_i\mathbf{p}_3)\cdot\mathbf{X}_i = 0 \end{cases}$$

$$\rightarrow \begin{cases} [\mathbf{p}_1 + 0\mathbf{p}_2 + (-u_i)\mathbf{p}_3]\cdot\mathbf{X}_i = 0 \\ [0\mathbf{p}_1 + \mathbf{p}_2 + (-v_i)\mathbf{p}_3]\cdot\mathbf{X}_i = 0 \end{cases}$$



Camera calibration: A linear approach—cont.

$$\begin{cases} [\mathbf{p}_1 + 0\mathbf{p}_2 + (-u_i)\mathbf{p}_3] \cdot \mathbf{X}_i = 0 \\ [0\mathbf{p}_1 + \mathbf{p}_2 + (-v_i)\mathbf{p}_3] \cdot \mathbf{X}_i = 0 \end{cases}$$

known, solve, unknown, known

- So, u_i, v_i and $\mathbf{X}_i$ are known, and $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ (12 elements) are unknown and to be solved.
- Re-arrange the equations:

$$\begin{pmatrix} \mathbf{X}_i^{\mathrm{T}} & 0^{\mathrm{T}} & -u_i\mathbf{X}_i^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_i^{\mathrm{T}} & -v_i\mathbf{X}_i^{\mathrm{T}} \end{pmatrix}_{2\times 12} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}_{2\times 1}$$

(1×4) (1×4) (1×4) 1×12 1×12 (1×4) 非正式寫法 unknown P known x = Px

Camera calibration: A linear approach—cont.

- In practice, given n features:

$$\begin{bmatrix} \begin{pmatrix} \mathbf{X}_1^{\mathrm{T}} & 0^{\mathrm{T}} & -u_1\mathbf{X}_1^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_1^{\mathrm{T}} & -v_1\mathbf{X}_1^{\mathrm{T}} \end{pmatrix} \\ \begin{pmatrix} \mathbf{X}_2^{\mathrm{T}} & 0^{\mathrm{T}} & -u_2\mathbf{X}_2^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_2^{\mathrm{T}} & -v_2\mathbf{X}_2^{\mathrm{T}} \end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix} \mathbf{X}_n^{\mathrm{T}} & 0^{\mathrm{T}} & -u_n\mathbf{X}_n^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_n^{\mathrm{T}} & -v_n\mathbf{X}_n^{\mathrm{T}} \end{pmatrix} \end{bmatrix}_{2n\times 12} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{pmatrix}_{2n\times 1}$$

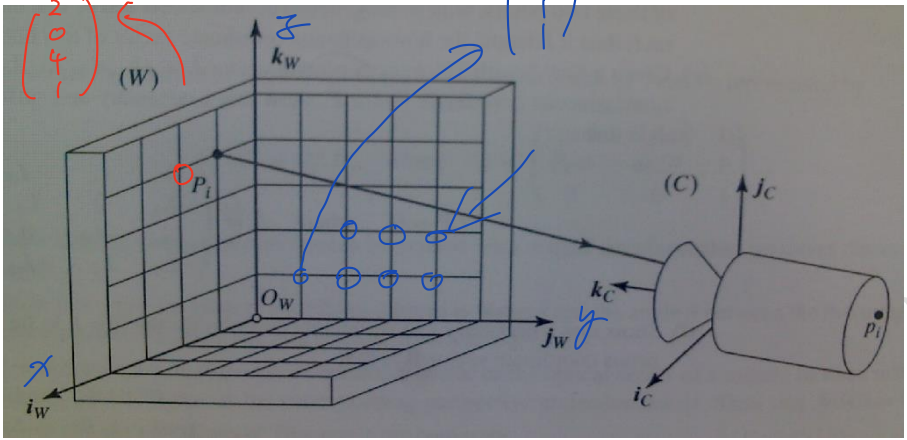
⇐ SVD

① 6,
②
③ SVD

- Solution? At least 6 features are needed for solving 12-1 unknowns.
- Least square or SVD…

Camera calibration: A linear approach—cont.

- A simple framework:
  - 3D Checkerboard → known 3D points
  - 2D feature → projected on image

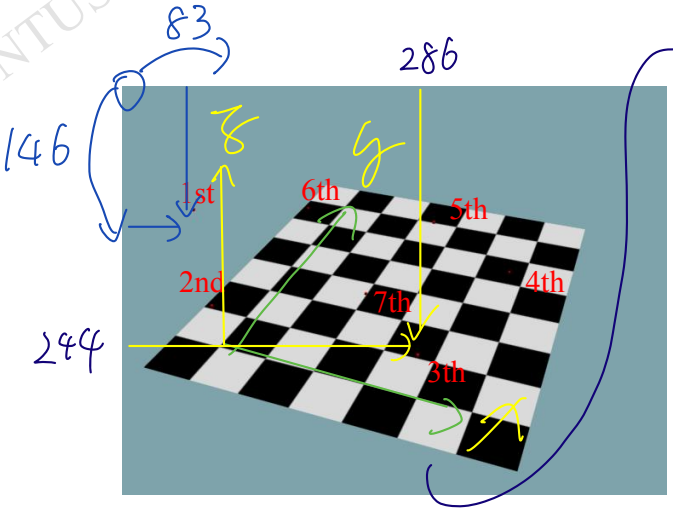


Picture from [Foysyth 2003]

Camera calibration: A linear approach

HW3

- Example



7 point

3D points	2D points (on image)
$X1=[0,0,75,1]^T$	$uv1=[83,146,1]^T$
$X2=[0,0,25,1]^T$	$uv2=[103,259,1]^T$
$X3=[100,0,25,1]^T$	$uv3=[346,315,1]^T$
$X4=[120,90,15,1]^T$	$uv4=[454,218,1]^T$
$X5=[90,50,60,1]^T$	$uv5=[365,161,1]^T$
$X6=[0,100,25,1]^T$	$uv6=[218,144,1]^T$
$X7=[60,40,20,1]^T$	$uv7=[286,244,1]^T$

3D 2D

To determine a transformation **P**

Camera calibration: A linear approach

- Example—cont.
- Calculation using Matlab

```
Puvmatrix=
[ X1'   zero' -uv1(1).*X1';
  zero' X1'   -uv1(2).*X1';
  X2'   zero' -uv2(1).*X2';
  zero' X2'   -uv2(2).*X2';
  X3'   zero' -uv3(1).*X3';
  zero' X3'   -uv3(2).*X3';
  X4'   zero' -uv4(1).*X4';
  zero' X4'   -uv4(2).*X4';
  X5'   zero' -uv5(1).*X5';
  zero' X5'   -uv5(2).*X5';
  X6'   zero' -uv6(1).*X6';
  zero' X6'   -uv6(2).*X6';
  X7'   zero' -uv7(1).*X7';
  zero' X7'   -uv7(2).*X7']
```

Puvmatrix =

$$\begin{pmatrix}
0 & 0 & 75 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -6225 & -83 \\
0 & 0 & 0 & 0 & 0 & 0 & 75 & 1 & 0 & 0 & -10950 & -146 \\
0 & 0 & 25 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -2575 & -103 \\
0 & 0 & 0 & 0 & 0 & 0 & 25 & 1 & 0 & 0 & -6475 & -259 \\
100 & 0 & 25 & 1 & 0 & 0 & 0 & 0 & -34600 & 0 & -8650 & -346 \\
0 & 0 & 0 & 0 & 100 & 0 & 25 & 1 & -31500 & 0 & -7875 & -315 \\
120 & 90 & 15 & 1 & 0 & 0 & 0 & 0 & -54480 & -40860 & -6810 & -454 \\
0 & 0 & 0 & 0 & 120 & 90 & 15 & 1 & -26160 & -19620 & -3270 & -218 \\
90 & 50 & 60 & 1 & 0 & 0 & 0 & 0 & -32850 & -18250 & -21900 & -365 \\
0 & 0 & 0 & 0 & 90 & 50 & 60 & 1 & -14490 & -8050 & -9660 & -161 \\
0 & 100 & 25 & 1 & 0 & 0 & 0 & 0 & 0 & -21800 & -5450 & -218 \\
0 & 0 & 0 & 0 & 0 & 100 & 25 & 1 & 0 & -14400 & -3600 & -144 \\
60 & 40 & 20 & 1 & 0 & 0 & 0 & 0 & -17160 & -11440 & -5720 & -286 \\
0 & 0 & 0 & 0 & 60 & 40 & 20 & 1 & -14640 & -9760 & -4880 & -244
\end{pmatrix}
*
\begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}_{12\times 1}
=
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{pmatrix}_{14\times 1}$$

Camera calibration: A linear approach

- Example—cont.

Solve by SVD:

V =

-0.0016	0.0007	0.0000	0.0718	-0.3876	0.3934	-0.0600	0.1281	0.6003	-0.5561	0.0143	0.0063
-0.0010	-0.0023	0.0006	-0.0004	-0.2591	-0.4549	-0.0224	0.2755	0.5839	0.5553	0.0115	0.0049
-0.0005	-0.0000	0.0025	0.0349	-0.1557	-0.0202	0.7303	-0.6355	0.1789	0.0672	0.0193	-0.0016
-0.0000	-0.0000	0.0000	-0.0010	-0.0054	-0.0008	0.0112	-0.0100	0.0185	-0.0008	-0.9400	0.3404
-0.0009	0.0009	-0.0001	-0.1100	0.6689	-0.4338	0.2098	0.1291	0.3196	-0.4352	0.0047	0.0009
-0.0005	-0.0014	0.0005	-0.0898	0.4693	0.6693	0.2133	0.2152	0.2383	0.4185	0.0006	-0.0023
-0.0003	0.0001	0.0026	-0.0785	0.2475	0.0391	-0.6100	-0.6638	0.3259	0.1096	0.0021	-0.0073
-0.0000	0.0000	0.0000	-0.0040	0.0075	0.0024	-0.0067	-0.0045	-0.0108	0.0034	0.3402	0.9402
0.8391	-0.4776	0.2602	-0.0046	-0.0007	0.0001	0.0003	0.0006	0.0007	-0.0020	0.0000	-0.0000
0.4881	0.8724	0.0272	-0.0032	-0.0009	0.0001	0.0004	0.0009	0.0016	0.0019	0.0000	0.0000
0.2399	-0.1042	-0.9651	-0.0133	-0.0020	0.0003	0.0007	-0.0029	0.0017	0.0001	0.0001	-0.0000
0.0087	-0.0009	-0.0116	0.9834	0.1711	-0.0124	-0.0273	-0.0056	0.0336	0.0368	-0.0006	0.0030

Get **P**:

P =[V(1:4,12)';V(5:8,12)';V(9:12,12)']

P =

0.0063	0.0049	-0.0016	0.3404
0.0009	-0.0023	-0.0073	0.9402
-0.0000	0.0000	-0.0000	0.0030

→ (normalized)

P =

2.0791	1.6213	-0.5162	111.7871
0.3027	-0.7658	-2.4001	308.7853
-0.0007	0.0024	-0.0016	1.0000

Camera calibration: A linear approach

- Example—cont.
  - Verify solution (in Matlab):

3D points
$X1=[0,0,75,1]^T$
$X2=[0,0,25,1]^T$
$X3=[100,0,25,1]^T$
$X4=[120,90,15,1]^T$
$X5=[90,50,60,1]^T$
$X6=[0,100,25,1]^T$
$X7=[60,40,20,1]^T$

2D points (on image)
$uv1=[83,146,1]^T$
$uv2=[103,259,1]^T$
$uv3=[346,315,1]^T$
$uv4=[454,218,1]^T$
$uv5=[365,161,1]^T$
$uv6=[218,144,1]^T$
$uv7=[286,244,1]^T$

Estimated — measured ⇒ (Estimated project 2D)

>> pp1= P*X1	>> pp2= P*X2	>> pp3= P*X3	>> pp4= P*X4	>> pp5= P*X5	>> pp6= P*X6	>> pp7= P*X7
pp1 =	pp2 =	pp3 =	pp4 =	pp5 =	pp6 =	pp7 =
73.0713 128.7766 0.8807	98.8818 248.7824 0.9602	306.7967 279.0484 0.8862	499.4553 240.1835 1.1002	349.0009 153.7293 0.9562	261.0078 172.2054 1.1968	291.0622 248.3118 1.0184
>> pp1=pp1./pp1(3)	>> pp2=pp2./pp2(3)	>> pp3=pp3./pp3(3)	>> pp4=pp4./pp4(3)	>> pp5=pp5./pp5(3)	>> pp6=pp6./pp6(3)	>> pp7=pp7./pp7(3)
pp1 =	pp2 =	pp3 =	pp4 =	pp5 =	pp6 =	pp7 =
82.9740 146.2286 1.0000	102.9785 259.0896 1.0000	346.1898 314.8785 1.0000	453.9658 218.3080 1.0000	364.9935 160.7738 1.0000	218.0963 143.8936 1.0000	285.8079 243.8292 1.0000
Estimated						→ Estimated
uv1=[83,146,1]^T	uv2=[103,259,1]^T	uv3=[346,315,1]^T	uv4=[454,218,1]^T	uv5=[365,161,1]^T	uv6=[218,144,1]^T	uv7=[286,244,1]^T

measured

Camera calibration: A linear approach

- Example—cont.
  - Compared with "openCV calibration result"

P =

2.0791	1.6213	-0.5162	111.7871
0.3027	-0.7658	-2.4001	308.7853
-0.0007	0.0024	-0.0016	1.0000

pick up 6 points

default 3P point

Compare data

Intrinsic parameter 3x3 Matrix (for all views)
797.467667 0.000000 318.980339
0.000000 797.569342 243.459839
0.000000 0.000000 1.000000

Distortion factor K
-0.002244 0.030546 -0.000019 -0.000223 0.0

view1.bmp: Extrinsic parameter 3x4 matrix
0.937760 0.347283 -0.000317 -83.818298
0.201549 -0.544979 -0.813865 25.437572
-0.282814 0.763146 -0.581054 325.901184

```
>> P=K*rt

P =

 1.0e+004 *

   0.0658   0.0520  -0.0186   3.7114
   0.0092  -0.0249  -0.0791   9.9632
  -0.0000   0.0001  -0.0001   0.0326

>> P=P./P(3,4)

P =

   2.0179   1.5967  -0.5695  113.8802
   0.2820  -0.7636  -2.4258  305.7125
  -0.0009   0.0023  -0.0018   1.0000
```

Camera calibration: A linear approach

- Decomposition for **K**[**R**|**t**]
- In case of: **P**=**K**[**R**|**t**]

known known

- then, [**R**|**t**]=$\mathbf{K}^{-1}\mathbf{P}$ →up to scale
- Note! Constraints for R are orthogonal and unit column/row vectors. In simple, you need to scale it.
- [**R**|**t**]= $s\mathbf{K}^{-1}\mathbf{P}$

Camera calibration: A linear approach

- Example—cont.

P =

2.0791	1.6213	-0.5162	111.7871
0.3027	-0.7658	-2.4001	308.7853
-0.0007	0.0024	-0.0016	1.0000

```
>> RT=inv(K)*P
```

RT =

0.0029	0.0011	-0.0000	-0.2598
0.0006	-0.0017	-0.0025	0.0819
-0.0007	0.0024	-0.0016	1.0000

Length=0.0030566112

Not a unit vector from inv(K)*P. Need a scale.

```
>> RT=inv(K)*P./0.0030566112
```

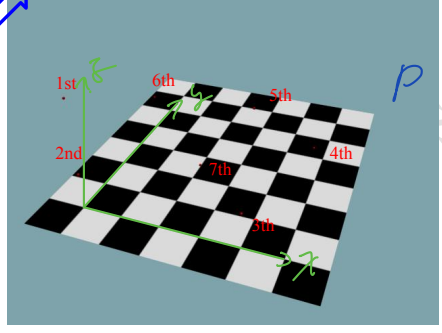
RT =

0.9498	0.3556	-0.0035	-85.0007
0.1981	-0.5503	-0.8256	26.7962
-0.2421	0.7739	-0.5206	327.1597

Intrinsic parameter 3x3 Matrix (for all views)
797.467667 0.000000 318.980339
0.000000 797.569342 243.459839
0.000000 0.000000 1.000000

Distortion factor K
-0.002244 0.030546 -0.000019 -0.000223 0.0

view1.bmp: Extrinsic parameter 3x4 matrix
0.937760 0.347283 -0.000317 -83.818298
0.201549 -0.544979 -0.813865 25.437572
-0.282814 0.763146 -0.581054 325.901184



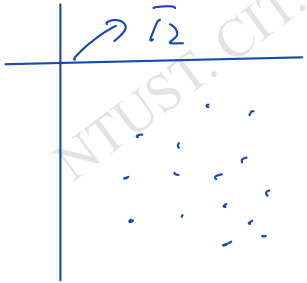
Gold Standard algorithm

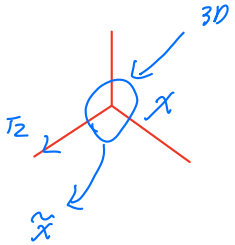
Objective

- Given $n \geq 6$ 3D to 2D point correspondences $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i'\}$, determine the Maximum Likehood Estimation of mapping transformation P
- Algorithm
  - Linear solution:
    - Normalization: $\widetilde{\mathbf{P}}$
    - DLT (using SVD or least-square …) → get
  - Minimization of geometric error: using the linear estimate as a starting point minimize the geometric error. (RANSIC)
  - Denormalization. $\mathbf{P} = \left(\left[[\mathbf{s}][\mathbf{t}]\right]_{3\times 3}\right)^{-1} \widetilde{\mathbf{P}} \cdot \left(\left[[\mathbf{S}][\mathbf{T}]\right]_{4\times 4}\right)$

Algorithm 7.1 [Hartley04]

Gold Standard algorithm—cont.





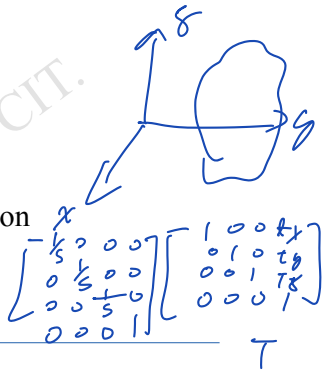
$$\mathbf{x} = \mathbf{P} \cdot \mathbf{X}$$

3D (Big)

normalization

small

$$\tilde{\mathbf{x}} = \tilde{\mathbf{P}} \cdot \tilde{\mathbf{X}}$$



Step-1a $\tilde{\mathbf{X}} = [[\mathbf{S}][\mathbf{T}]]_{4\times4}\mathbf{X}$ →normalization (3D points)

Translate center to the origin

Scale all points to the unit cube (or average the vector distance to be 1.414 or $\sqrt{2}$ →textbook suggestion value)

$\tilde{\mathbf{x}} = [[\mathbf{s}][\mathbf{t}]]_{3\times3}\mathbf{x}$ →normalization (2D points)

Gold Standard algorithm—cont.

Step-1b

$$\tilde{\mathbf{x}} = \tilde{\mathbf{P}} \cdot \tilde{\mathbf{X}}$$

20 30

→solve $\tilde{\mathbf{P}}$ by SVD or least-square method

Hint:

$$\begin{bmatrix} \begin{pmatrix} \mathbf{X}_1^{\mathrm{T}} & 0^{\mathrm{T}} & -u_1\mathbf{X}_1^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_1^{\mathrm{T}} & -v_1\mathbf{X}_1^{\mathrm{T}} \end{pmatrix} \\ \begin{pmatrix} \mathbf{X}_2^{\mathrm{T}} & 0^{\mathrm{T}} & -u_2\mathbf{X}_2^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_2^{\mathrm{T}} & -v_2\mathbf{X}_2^{\mathrm{T}} \end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix} \mathbf{X}_n^{\mathrm{T}} & 0^{\mathrm{T}} & -u_n\mathbf{X}_n^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_n^{\mathrm{T}} & -v_n\mathbf{X}_n^{\mathrm{T}} \end{pmatrix} \end{bmatrix}_{2n\times 12} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{pmatrix}_{2n\times 1}$$

Step-2 →minimize the re-projection error, if necessary.

Normalizing transformations—cont.

- What the **T** means?:
  - A composition of "Translation" and "Scale".
  - Indeed,

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}_{\text{Normalized}} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\bar{x} \\ 0 & 1 & -\bar{y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}_{\text{Original}}$$

$$\mathbf{T} = \begin{bmatrix} s_x & 0 & -s_x\bar{x} \\ 0 & s_y & -s_y\bar{y} \\ 0 & 0 & 1 \end{bmatrix}$$

(i) The points are translated so that their centroid is at the origin.
(ii) The points are then scaled so that the average distance from the origin is equal to $\sqrt{2}$.
(iii) This transformation is applied to each of the two images independently.

$(\bar{x}, \bar{y})$ ⟶ centroid of all points

(s_x, s_y) ⟶ scale (could be $s_x = s_y$), and be suggested to be $\frac{\sqrt{2}}{l}$ l is the average distance to centroid.

Gold Standard algorithm—cont.

HW3

Step-3 De-normalization (determine **P**) $\mathbf{x} = \mathbf{P} \cdot \mathbf{X}$

$$\mathbf{P} = \left(\left[\mathbf{s}\right]\left[\mathbf{t}\right]_{3\times3}\right)^{-1} \tilde{\mathbf{P}} \cdot \left(\left[\mathbf{S}\right]\left[\mathbf{T}\right]_{4\times4}\right)$$

$$\tilde{\mathbf{x}} = \tilde{\mathbf{P}} \cdot \tilde{\mathbf{X}}$$

$$\left[\mathbf{s}\right]\left[\mathbf{t}\right]_{3\times3} \mathbf{x} = \tilde{\mathbf{P}}\left(\left[\mathbf{S}\right]\left[\mathbf{T}\right]_{4\times4} \mathbf{X}\right)$$

$$\mathbf{x} = \left(\left[\mathbf{s}\right]\left[\mathbf{t}\right]_{3\times3}\right)^{-1} \tilde{\mathbf{P}}\left(\left[\mathbf{S}\right]\left[\mathbf{T}\right]_{4\times4}\right)\mathbf{X}$$

$$\therefore \mathbf{P} = \left(\left[\mathbf{s}\right]\left[\mathbf{t}\right]_{3\times3}\right)^{-1} \tilde{\mathbf{P}} \cdot \left(\left[\mathbf{S}\right]\left[\mathbf{T}\right]_{4\times4}\right)$$

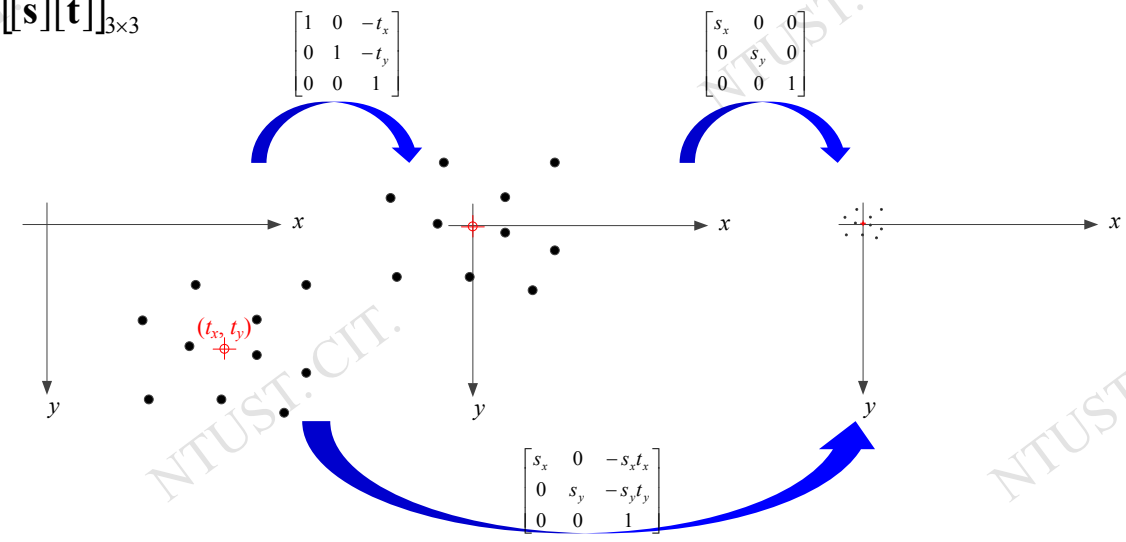
big → small

3D ⇒ 2D

Big - small

Gold Standard algorithm—remark

$[[\mathbf{s}][\mathbf{t}]]_{3\times3}$



Camera calibration: A linear approach

- Special case:
  - Affine camera model for mapping transform

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \mathbf{p}_3^{\mathrm{T}} \end{bmatrix}_{3\times 4} \cdot \mathbf{X}_i$$

$$\mathbf{x}_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \cdot \mathbf{X}_i$$

Projective camera

P: full Rank

$$\mathbf{x}_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{X}_i$$

→ 4

→ telephoto

Affine camera (orthography)

Camera calibration: A linear approach

- Special case:
  - Affine camera model for mapping transform

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

3x4 mapping matrix

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \mathbf{p}_3^{\mathrm{T}} \end{bmatrix}_{3\times 4} \cdot \mathbf{X}_i$$

1x4 vector
1x4 vector
1x4 vector

$\mathbf{X}_i = [X_{i1} \quad X_{i2} \quad X_{i3} \quad X_{i4}]^{\mathrm{T}}$: *i*-th 3D point in space

$\mathbf{x}_i = [u_i \quad v_i \quad 1]^{\mathrm{T}}$: *i*-th 2D point on image

If in special case (affine camera): $\mathbf{p}_3^{\mathrm{T}} = [0 \quad 0 \quad 0 \quad 1]$

$$\begin{bmatrix} \begin{pmatrix} \mathbf{X}_1^{\mathrm{T}} & 0^{\mathrm{T}} & -u_1\mathbf{X}_1^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_1^{\mathrm{T}} & -v_1\mathbf{X}_1^{\mathrm{T}} \end{pmatrix} \\ \begin{pmatrix} \mathbf{X}_2^{\mathrm{T}} & 0^{\mathrm{T}} & -u_2\mathbf{X}_2^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_2^{\mathrm{T}} & -v_2\mathbf{X}_2^{\mathrm{T}} \end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix} \mathbf{X}_n^{\mathrm{T}} & 0^{\mathrm{T}} & -u_n\mathbf{X}_n^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_n^{\mathrm{T}} & -v_n\mathbf{X}_n^{\mathrm{T}} \end{pmatrix} \end{bmatrix}_{2n\times 12} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{pmatrix}_{2n\times 1}$$

→ **Reduce to 8 unknowns**

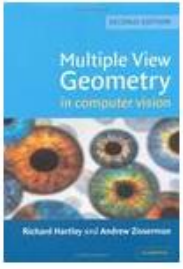
$$\begin{bmatrix} \begin{pmatrix} \mathbf{X}_1^{\mathrm{T}} & 0^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_1^{\mathrm{T}} \end{pmatrix} \\ \begin{pmatrix} \mathbf{X}_2^{\mathrm{T}} & 0^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_2^{\mathrm{T}} \end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix} \mathbf{X}_n^{\mathrm{T}} & 0^{\mathrm{T}} \\ 0^{\mathrm{T}} & \mathbf{X}_n^{\mathrm{T}} \end{pmatrix} \end{bmatrix}_{2n\times 8} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{pmatrix}_{8\times 1} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \cdots \\ u_n \\ v_n \end{pmatrix}_{2n\times 1}$$

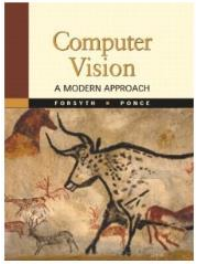
Gold Standard algorithm

- Objective
- Given n≥4 3D to 2D point correspondences {Xi↔xi'}, determine the Maximum Likelyhood Estimation of **P**, here $\mathbf{p}_3^T=[0,0,0,1]$.
- Algorithm
  - Normalization:
  - Correspondence (governing equation)
  - Determine P by SVD or DLT.
  - Denormalization:

Camera calibration

- Intrinsic parameter & extrinsic parameter
- Lecture Reference at:
  - Multiple View Geometry in Computer Vision, (Chapter8.4, 8.5)
  - Computer Vision A Modern Approach, NA
  - Select paper: Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. IEEE International Conference on Computer Vision. 1, (1999), 666-673.





Camera calibration

- Intrinsic parameter calibration
  - From absolute conic
  - From homography

Camera calibration

- From absolute conic
- Points on the plane at infinity and absolute conic

$$\mathbf{x} = \mathbf{P} \cdot \mathbf{X}_\infty$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \cdot \mathbf{X}_\infty$$

$\mathbf{X}_\infty$ is one 3D point at infinity. Let $\mathbf{X}_\infty = \begin{bmatrix} \mathbf{d}_{3\times1} \\ 0 \end{bmatrix}$

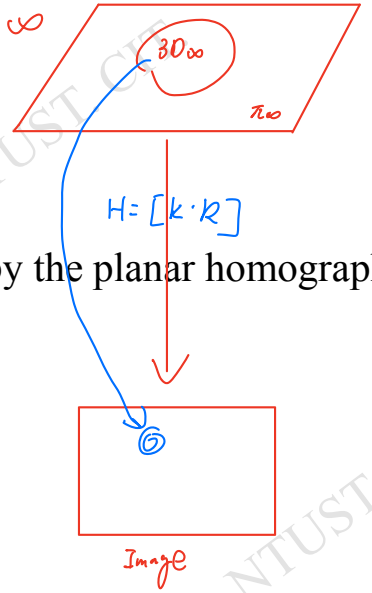
$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \cdot \mathbf{X}_\infty = \mathbf{K}[\mathbf{R} \mid \cancel{\mathbf{t}}] \cdot \begin{bmatrix} \mathbf{d}_{3\times1} \\ 0 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{KRd} \longrightarrow \text{2D point to 2D point mapping} \rightarrow \text{homography}$$

$$\mathbf{x} = \mathbf{Hd}$$

Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic



- The mapping between $\boldsymbol{\pi}_\infty$ and an image is given by the planar homography $\mathbf{x} = \mathbf{Hd}$ with

$$\mathbf{H} = \mathbf{KR}$$

Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic

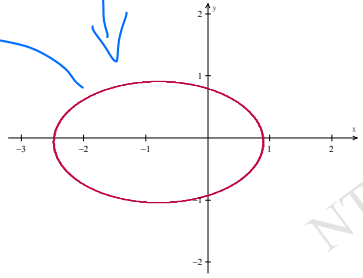
- Recall the property of the conic: if one point is on the conic C, it forms

$$\mathbf{x}^{\mathsf{T}}\mathbf{C}\mathbf{x} = 0$$

- For example, one ellipse:

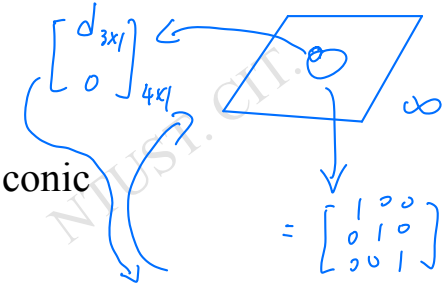
$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 & 4 \\ 0 & 15 & 1 \\ 4 & 1 & -11 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

對稱 3x3



Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic



- Absolute conic $\boldsymbol{\Omega}_\infty$: the a (point) conic on $\boldsymbol{\pi}_\infty = (0,0,0,1)^T$

$$\mathbf{X}_\infty = (X_1, X_2, X_3, X_4)$$

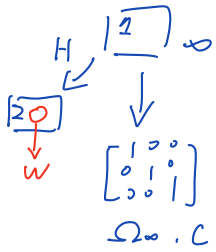
$$\begin{cases} X_1^2 + X_2^2 + X_3^2 = 0 \\ X_4 = 0 \end{cases}$$

$$(X_1, X_2, X_3)\mathbf{I}(X_1, X_2, X_3)^T = 0$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic



- The image of the absolute conic (called IAC) is the conic :

$$\boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^{\mathrm{T}})^{-1}$$

- Proof: $\mathbf{x}' = \mathbf{H}\mathbf{x}$ → points $\mathbf{x}$ are mapped to $\mathbf{x}'$ by homography $\mathbf{H}$
  $\mathbf{C}' = \mathbf{H}^{-\mathrm{T}}\mathbf{C}\mathbf{H}^{-1}$ → conic $\mathbf{C}$ will be mapped to $\mathbf{C}'$ (according this eqs)

- Since we want to determine the mapping result (says w as a new notation) of the absolute conic ($\mathbf{C}=\mathbf{I}$), the $\mathbf{H}=\mathbf{K}\mathbf{R}$

$$\boldsymbol{\omega} = \mathbf{H}^{-\mathrm{T}}\mathbf{I}\mathbf{H}^{-1} = (\mathbf{K}\mathbf{R})^{-\mathrm{T}}\mathbf{I}(\mathbf{K}\mathbf{R})^{-1} = \mathbf{K}^{-\mathrm{T}}\mathbf{R}^{-\mathrm{T}}\mathbf{I}\mathbf{R}^{-1}\mathbf{K}^{-1}$$
$$= \mathbf{K}^{-\mathrm{T}}\mathbf{R}\mathbf{I}\mathbf{R}^{-1}\mathbf{K}^{-1} = \mathbf{K}^{-\mathrm{T}}\mathbf{K}^{-1} = (\mathbf{K}\mathbf{K}^{\mathrm{T}})^{-1}$$

to determine $\mathbf{K}$ from a known $\boldsymbol{\omega}$, Cholesky factorization is used.

Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic

- Absolute conic: points $[1 \quad \pm i \quad 0]$ are on absolute conic (imaginary point).

$$[1 \quad \pm i \quad 0]\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1\\ \pm i\\0\end{bmatrix}=0$$

Points at infinity plane
(and on absolute conic)

$\mathbf{h}_1 \pm i\mathbf{h}_2$

- Suppose these points can be mapped by a known H, the mapped points will be

$$\mathbf{H}\begin{bmatrix}1\\ \pm i\\0\end{bmatrix}=\begin{bmatrix}\mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3\end{bmatrix}\begin{bmatrix}1\\ \pm i\\0\end{bmatrix}=\mathbf{h}_1 \pm i\mathbf{h}_2$$

Points mapping back (via homography) to image plane

3D Points on the plane at infinity and absolute conic

$\boldsymbol{\Omega}_\infty = (X_1, X_2, X_3, X_4)$

∞

$\mathbf{X}_\infty = (X_1, X_2, X_3, X_4) \quad \boldsymbol{\pi}_\infty = (0,0,0,1)^T$

$\mathbf{H} = \mathbf{KR}$

$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Image plane (taken pic)

$\boldsymbol{\omega}$

$(\mathbf{h}_1 \pm i\mathbf{h}_2)$

$\mathbf{x}' = \mathbf{Hx}$

$\mathbf{C}' = \mathbf{H}^{-T}\mathbf{C}\mathbf{H}^{-1}$

$\boldsymbol{\omega} = \mathbf{H}^{-T}\boldsymbol{\Omega}_\infty\mathbf{H}^{-1}$

$\boldsymbol{\omega} = (\mathbf{KR})^{-T}\boldsymbol{\Omega}_\infty(\mathbf{KR})^{-1}$

Camera calibration

- From absolute conic—cont.
- Points on the plane at infinity and absolute conic

Conics

- Curve described by 2nd-degree equation in the plane
$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$
- or homogenized $x \mapsto x_1/x_3, y \mapsto x_2/x_3$
$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$$
- or in matrix form
$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0 \longrightarrow \mathbf{x}^T\mathbf{C}\mathbf{x} = 0$$
5DOF: $\{a:b:c:d:e:f\}$ with $\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$

5 DoF

- According the "points on conic" equation $\mathbf{x}^T\mathbf{C}\mathbf{x} = 0$

Image

$$(\mathbf{h}_1 \pm i\mathbf{h}_2)^T \boldsymbol{\omega} (\mathbf{h}_1 \pm i\mathbf{h}_2) = 0$$

The mapping back points will be on the conic

- Deal with real part & imaginary part individually:

$(h_1^T\omega \pm i h_2^T\omega)(h_1 \pm i h_2)$

$$\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_1 + i^2\mathbf{h}_2^T\boldsymbol{\omega}\mathbf{h}_2 + 2i(\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_2) = 0$$

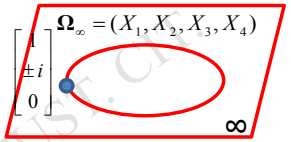
$\Rightarrow h_1^T\omega h_1 + i^2 h_2^T\omega h_2 + 2i(h_1^T\omega h_2) = 0$

$$\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_2 = 0$$

$$\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_1 = \mathbf{h}_2^T\boldsymbol{\omega}\mathbf{h}_2$$

$\Rightarrow$ 2

3D Points on the plane at infinity and absolute conic

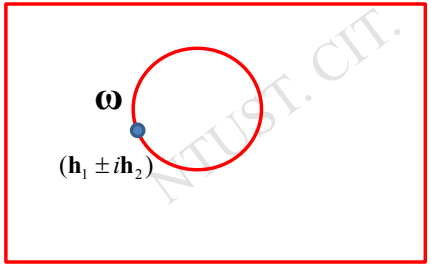


$$\mathbf{X}_\infty = (X_1, X_2, X_3, X_4) \quad \boldsymbol{\pi}_\infty = (0,0,0,1)^{\mathrm{T}}$$

$$(\mathbf{h}_1 \pm i\mathbf{h}_2) = \mathbf{H}\begin{bmatrix} 1 \\ \pm i \\ 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Image plane (taken pic)



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

$$\mathbf{C}' = \mathbf{H}^{-\mathrm{T}}\mathbf{C}\mathbf{H}^{-1}$$

$$\boldsymbol{\omega} = \mathbf{H}^{-\mathrm{T}}\boldsymbol{\Omega}_\infty\mathbf{H}^{-1}$$

$$\boldsymbol{\omega} = (\mathbf{K}\mathbf{R})^{-\mathrm{T}}\boldsymbol{\Omega}_\infty(\mathbf{K}\mathbf{R})^{-1}$$

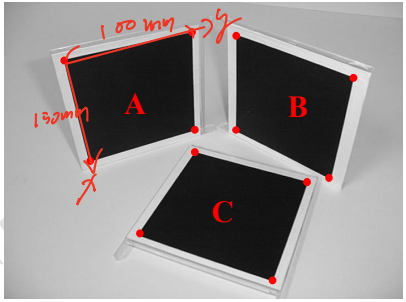
Camera calibration

- Summary: A simple calibration from the absolute conic
  - Step-1: compute H for each square →for example 3 squares induce 3 **H** (unit corners (0,0),(1,0),(0,1),(1,1))
  - Step-2: compute the imaged circular points $\mathbf{H}(1,\pm i,0)^T$
  - Step-3: fit a conic to 6 circular points ω → SVD
  - Step-4: compute K from ω through Cholesky factorization

[algorithm from Hartley textbook example8.18]

Camera calibration

- Example: from absolute conic



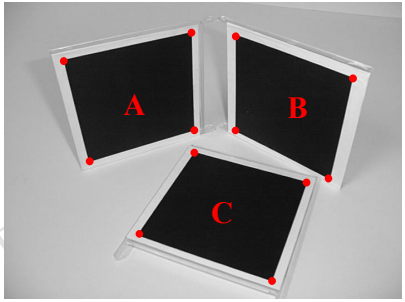
unit square
0,0
0,1
1,1
1,0

Homograph Homograph Homograph

A-square	B-square	C-square
152,149	596,84	490,387
218,413	596,334	343,602
490,332	838,458	689,722
482,77	898,195	780,465

Camera calibration

- Example: from absolute conic—cont.



Homography (unit square to **A-square**)
379.199677 111.830818 152.000000
-64.140297 350.826263 149.000000
0.102074 0.210233 1.000000

Homography (unit square to **B-square**)
168.271439 125.763161 596.000000
81.960945 320.478027 84.000000
-0.148918 0.211012 1.000000

Homography (unit square to **C-square**)
228.969971 -209.572891 490.000000
41.616714 105.178200 387.000000
-0.078244 -0.182428 1.000000

Camera calibration

- Example: from absolute conic—cont.
- To determine w, remember this is a "conic", a symmetric 3x3 matrix form

$$\mathbf{h}_1^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_2 = 0$$

$$\mathbf{h}_1^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_1 = \mathbf{h}_2^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_2$$

$$\mathbf{h}_1 = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \end{bmatrix}, \mathbf{h}_2 = \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix}$$

$$\mathbf{h}_1^{\mathrm{T}} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \mathbf{h}_2 = 0 \quad (1)$$

$$\mathbf{h}_1^{\mathrm{T}} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \mathbf{h}_1 = \mathbf{h}_2^{\mathrm{T}} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \mathbf{h}_2 \quad (2)$$

$$h_{11}h_{21}\omega_{11} + h_{11}h_{22}\omega_{12} + h_{11}h_{23}\omega_{13} + h_{12}h_{21}\omega_{21} + h_{12}h_{22}\omega_{22} + h_{12}h_{23}\omega_{23} + h_{13}h_{21}\omega_{31} + h_{13}h_{22}\omega_{32} + h_{13}h_{23}\omega_{33} = 0$$

$$h_{11}h_{11}\omega_{11} + h_{11}h_{12}\omega_{12} + h_{11}h_{13}\omega_{13} + h_{12}h_{11}\omega_{21} + h_{12}h_{12}\omega_{22} + h_{12}h_{13}\omega_{23} + h_{13}h_{11}\omega_{31} + h_{13}h_{12}\omega_{32} + h_{13}h_{13}\omega_{33} =$$

$$h_{21}h_{21}\omega_{11} + h_{21}h_{22}\omega_{12} + h_{21}h_{23}\omega_{13} + h_{22}h_{21}\omega_{21} + h_{22}h_{22}\omega_{22} + h_{22}h_{23}\omega_{23} + h_{23}h_{21}\omega_{31} + h_{23}h_{22}\omega_{32} + h_{23}h_{23}\omega_{33}$$

Note! $\boldsymbol{\omega}$ is symmetric

Camera calibration

- Example: from absolute conic—cont.

$$h_{11}h_{21}\omega_{11} + (h_{11}h_{22} + h_{12}h_{21})\omega_{12} + (h_{11}h_{23} + h_{13}h_{21})\omega_{13} + h_{12}h_{22}\omega_{22} + (h_{12}h_{23} + h_{13}h_{22})\omega_{23} + h_{13}h_{23}\omega_{33} = 0$$

$$(h_{11}h_{11} - h_{21}h_{21})\omega_{11} + (h_{11}h_{12} + h_{12}h_{11} - h_{21}h_{22} - h_{22}h_{21})\omega_{12} + (h_{11}h_{13} + h_{13}h_{11} - h_{21}h_{23} - h_{23}h_{21})\omega_{13} + (h_{12}h_{12} - h_{22}h_{22})\omega_{22}$$
$$+ (h_{12}h_{13} + h_{13}h_{12} - h_{22}h_{23} - h_{23}h_{22})\omega_{23} + (h_{13}h_{13} - h_{23}h_{23})\omega_{33} = 0$$

$$\begin{bmatrix} h_{11}h_{21} & (h_{11}h_{22} + h_{12}h_{21}) & (h_{11}h_{23} + h_{13}h_{21}) & h_{12}h_{22} & (h_{12}h_{23} + h_{13}h_{22}) & h_{13}h_{23} \\ (h_{11}^2 - h_{21}^2) & 2(h_{11}h_{12} - h_{21}h_{22}) & 2(h_{11}h_{13} - h_{21}h_{23}) & (h_{12}^2 - h_{22}^2) & 2(h_{12}h_{13} - h_{22}h_{23}) & (h_{13}^2 - h_{23}^2) \end{bmatrix} \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{bmatrix} = 0$$

2Eqs

known

Two constraints from one homography

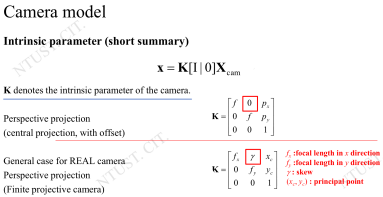
6 unknowns (actually 5 unknowns upto scale), so it needs at least **3** homography

```
A=[
h(1,1)*h(2,1) h(1,1)*h(2,2)+h(1,2)*h(2,1) h(1,1)*h(2,3)+h(1,3)*h(2,1) h(1,2)*h(2,2) h(1,2)*h(2,3)+h(1,3)*h(2,2) h(1,3)*h(2,3);
h(1,1)^2-h(2,1)^2 2*(h(1,1)*h(1,2)-h(2,1)*h(2,2)) 2*(h(1,1)*h(1,3)-h(2,1)*h(2,3)) h(1,2)^2-h(2,2)^2
2*(h(1,2)*h(1,3)-h(2,2)*h(2,3)) h(1,3)^2-h(2,3)^2 ]
```

BE CAREFUL! This notation is different from Matlab.

Camera calibration

- Example: from absolute conic—cont.
- To determine K in $\omega = (KK^T)^{-1}$



Assume $\mathbf{K} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \boldsymbol{\omega}^{-1} = \begin{bmatrix} \varpi_{11} & \varpi_{12} & \varpi_{13} \\ \varpi_{21} & \varpi_{22} & \varpi_{23} \\ \varpi_{31} & \varpi_{32} & \varpi_{33} \end{bmatrix} = s^2 \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix}^T$

$$= s^2 \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ b & d & 0 \\ c & e & 1 \end{bmatrix} = s^2 \begin{bmatrix} a^2+b^2+c^2 & bd+ce & c \\ bd+ce & d^2+e^2 & e \\ c & e & 1 \end{bmatrix}$$

$$\boldsymbol{\omega}^{-1} = \begin{bmatrix} \varpi_{11} & \varpi_{12} & \varpi_{13} \\ \varpi_{21} & \varpi_{22} & \varpi_{23} \\ \varpi_{31} & \varpi_{32} & \varpi_{33} \end{bmatrix} = s^2 \begin{bmatrix} a^2+b^2+c^2 & bd+ce & c \\ bd+ce & d^2+e^2 & e \\ c & e & 1 \end{bmatrix} \qquad s = \pm\sqrt{\varpi_{33}}$$

Camera calibration

- Example: from absolute conic—cont.

$$\begin{bmatrix} \varpi_{11} & \varpi_{12} & \varpi_{13} \\ \varpi_{21} & \varpi_{22} & \varpi_{23} \\ \varpi_{31} & \varpi_{32} & \varpi_{33} \end{bmatrix} = s^2 \begin{bmatrix} a^2+b^2+c^2 & bd+ce & c \\ bd+ce & d^2+e^2 & e \\ c & e & 1 \end{bmatrix}$$

let $\varpi_{33} = 1$

then, $c = \varpi_{13}$

$e = \varpi_{23}$

Solve d: (by 2nd row, 2nd column) $d = \pm\sqrt{\varpi_{22} - e^2}$ (use positive value)

Solve e:(by 1st row, 2nd column) $b = (\varpi_{12} - ce)/d$

Solve a:(by 1st row, 1st column) $a = \pm\sqrt{\varpi_{11} - b^2 - c^2}$ (use positive value)

Finally, $\mathbf{K} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix}$

Another solution for $\boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^{\mathrm{T}})^{-1}$ is Cholesky factorization (matlab2011)

Camera calibration

- Example: from absolute conic—cont.

opencv

Of course you can assume $\mathbf{K}=\begin{bmatrix} a & 0 & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix}$, since most digital CCDs have NO (or tiny) skew effect.

$$\boldsymbol{\omega}^{-1}=\begin{bmatrix} \varpi_{11} & \varpi_{12} & \varpi_{13} \\ \varpi_{21} & \varpi_{22} & \varpi_{23} \\ \varpi_{31} & \varpi_{32} & \varpi_{33} \end{bmatrix}=s^2\begin{bmatrix} a^2+c^2 & ce & c \\ ce & d^2+e^2 & e \\ c & e & 1 \end{bmatrix}$$

let $\varpi_{33}=1$

then, $c=\varpi_{13}$

$e=\varpi_{23}$

Solve d: (by 2nd row, 2nd column) $d=\pm\sqrt{\varpi_{22}-e^2}$ (use positive value)

Solve a:(by 1st row, 1st column) $a=\pm\sqrt{\varpi_{11}-c^2}$ (use positive value)

Check $|\varpi_{12}-ce|$ value, if it is not neglected, this assumption is poor.

Solution in Matlab (for example)

$$\begin{bmatrix} h_{11}h_{21} & (h_{11}h_{22}+h_{12}h_{21}) & (h_{11}h_{23}+h_{13}h_{21}) & h_{12}h_{22} & (h_{12}h_{23}+h_{13}h_{22}) & h_{13}h_{23} \\ (h_{11}^2-h_{21}^2) & 2(h_{11}h_{12}-h_{21}h_{22}) & 2(h_{11}h_{13}-h_{21}h_{23}) & (h_{12}^2-h_{22}^2) & 2(h_{12}h_{13}-h_{22}h_{23}) & (h_{13}^2-h_{23}^2) \end{bmatrix} \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{bmatrix} = 0$$

Step-1 determine every homography
(in this example 3 square→ 3 homography)

Step-2 determine every **h**1 & **h**2 from 3 homography. One homography gives two equations (constraints)
(NOTE! The transpose operation is ONLY for notation purpose in Matlab)

```
>> hap=ha'
hap =
```

h1	379.1997	-64.1403	0.1021
h2	111.8308	350.8263	0.2102
	152.0000	149.0000	1.0000

```
>> hbp=hb'
hbp =
```

168.2714	81.9609	-0.1489
125.7632	320.4780	0.2110
596.0000	84.0000	1.0000

```
>> hcp=hc'
hcp =
```

228.9700	41.6167	-0.0782
-209.5729	105.1782	-0.1824
490.0000	387.0000	1.0000

Step-3 construct the matrix form, then solve it by SVD for determine **ω**.

```
A=[hap(1,1)*hap(2,1) hap(1,1)*hap(2,2)+hap(1,2)*hap(2,1) hap(1,1)*hap(2,3)+hap(1,3)*hap(2,1) hap(1,2)*hap(2,2) hap(1,2)*hap(2,3)+hap(1,3)*hap(2,2) hap(1,3)*hap(2,3);
hap(1,1)^2-hap(2,1)^2 2*(hap(1,1)*hap(1,2)-hap(2,1)*hap(2,2)) 2*(hap(1,1)*hap(1,3)-hap(2,1)*hap(2,3)) hap(1,2)^2-hap(2,2)^2 2*(hap(1,2)*hap(1,3)-hap(2,2)*hap(2,3)) hap(1,3)^2-hap(2,3)^2;
hbp(1,1)*hbp(2,1) hbp(1,1)*hbp(2,2)+hbp(1,2)*hbp(2,1) hbp(1,1)*hbp(2,3)+hbp(1,3)*hbp(2,1) hbp(1,2)*hbp(2,2) hbp(1,2)*hbp(2,3)+hbp(1,3)*hbp(2,2) hbp(1,3)*hbp(2,3);
hbp(1,1)^2-hbp(2,1)^2 2*(hbp(1,1)*hbp(1,2)-hbp(2,1)*hbp(2,2)) 2*(hbp(1,1)*hbp(1,3)-hbp(2,1)*hbp(2,3)) hbp(1,2)^2-hbp(2,2)^2 2*(hbp(1,2)*hbp(1,3)-hbp(2,2)*hbp(2,3)) hbp(1,3)^2-hbp(2,3)^2;
hcp(1,1)*hcp(2,1) hcp(1,1)*hcp(2,2)+hcp(1,2)*hcp(2,1) hcp(1,1)*hcp(2,3)+hcp(1,3)*hcp(2,1) hcp(1,2)*hcp(2,2) hcp(1,2)*hcp(2,3)+hcp(1,3)*hcp(2,2) hcp(1,3)*hcp(2,3);
hcp(1,1)^2-hcp(2,1)^2 2*(hcp(1,1)*hcp(1,2)-hcp(2,1)*hcp(2,2)) 2*(hcp(1,1)*hcp(1,3)-hcp(2,1)*hcp(2,3)) hcp(1,2)^2-hcp(2,2)^2 2*(hcp(1,2)*hcp(1,3)-hcp(2,2)*hcp(2,3)) hcp(1,3)^2-hcp(2,3)^2]
```

```
[U,S,V]=svd(A)
```

→

```
V =
```

-0.4214	0.6339	-0.6485	-0.0012	-0.0000	0.0000
0.7277	0.6631	0.1753	-0.0000	-0.0002	0.0000
0.0001	0.0003	-0.0015	0.9972	-0.0750	-0.0003
0.5412	-0.3981	-0.7407	-0.0012	-0.0011	0.0000
0.0008	-0.0003	-0.0009	0.0750	0.9972	-0.0003
0.0000	-0.0000	0.0000	0.0003	0.0002	1.0000

→

```
>> w=[V(1:3,6)';V(2,6) V(4:5,6)';V(3,6) V(5,6) V(6,6)]
w =
```

0.0000	0.0000	-0.0003
0.0000	0.0000	-0.0003
-0.0003	-0.0003	1.0000

Solution in Matlab (for example)—cont.

Step-4 solve **K**, take inverse of **ω**, then use close-form solution

```
>> inw=inv(w)

inw =

  1.0e+006 *

    2.1064    0.2599    0.0007
    0.2599    1.8919    0.0006
    0.0007    0.0006    0.0000
```

Normalized
Let (3,3)=1

```
>> inw=inw./inw(3,3)

inw =

  1.0e+006 *

    1.5347    0.1894    0.0005
    0.1894    1.3784    0.0004
    0.0005    0.0004    0.0000
```

$$\boldsymbol{\omega}^{-1} = \begin{bmatrix} \varpi_{11} & \varpi_{12} & \varpi_{13} \\ \varpi_{21} & \varpi_{22} & \varpi_{23} \\ \varpi_{31} & \varpi_{32} & \varpi_{33} \end{bmatrix}$$

```
>> c=inw(1,3)

c =

   531.5349
```

```
>> e=inw(2,3)

e =

   409.5670
```

```
>> d=sqrt(inw(2,2)-e^2)

d =

  1.1003e+003
```

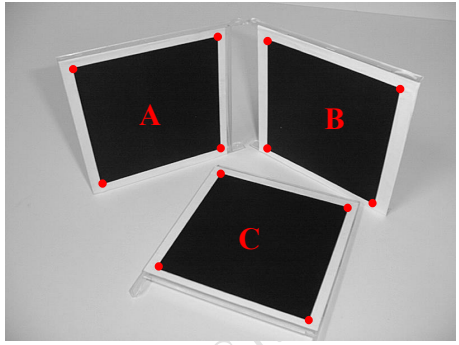
```
>> b=(inw(1,2)-c*e)/d

b =

   -25.7338
```

```
>> a=sqrt(inw(1,1)-b^2-c^2)

a =

  1.1187e+003
```

$$\mathbf{K} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1118.7 & -25.7 & 531.5 \\ 0 & 1100.3 & 409.6 \\ 0 & 0 & 1 \end{bmatrix}$$

Camera calibration

- Finally,



$$\mathbf{K} = \begin{bmatrix} 1108.3 & -9.8 & 525.8 \\ 0 & 1097.8 & 395.9 \\ 0 & 0 & 1 \end{bmatrix}$$

(from textbook)

What's NEXT?

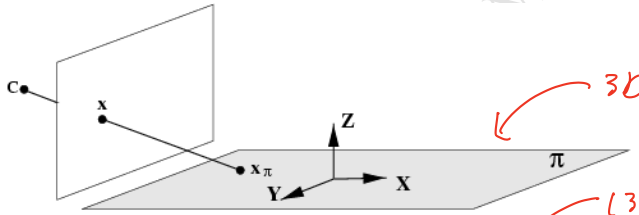
Determine distortion & extrinsic parameter

Refer to the following paper:

Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. *IEEE International Conference on Computer Vision*. 1, 666-673.

Camera calibration

- from homography—Zhang's method



$$\mathbf{x} = \mathbf{PX} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_4] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

2D points on image plane

Homography

3D points on a plane
(considered as 2D points)

Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. *IEEE International Conference on Computer Vision*. 1, 666-673.

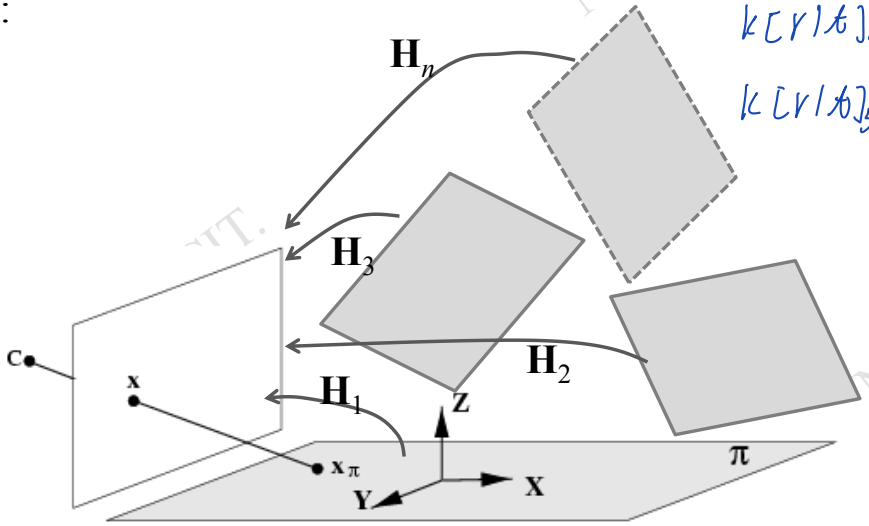
Camera calibration

- from homography—Zhang's method—cont.
  - The idea:

$k[r_1 t]_1 \leftarrow H_1$

$k[r_1 t]_2 \leftarrow H_2$

$k[r_1 t]_3 \leftarrow H_3$



Camera calibration

- from homography—Zhang's method—cont.

Points on a 3D plane:

$$\mathbf{x} = \mathbf{PX} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_4]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Recall the pin-hole projection formula:

$$\mathbf{x} = \mathbf{PX} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}$$

K: intrinsic parameter.
[**R**|**t**]: extrinsic parameter

can be reduced to:

$$\mathbf{x} = s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

The mapping (indeed homography) can be defined as

$$\mathbf{H} = \mathbf{K}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

Remark: **H** consists of intrinsic and part of extrinsic parameters

1x3 column vector

Note the notation in paper.

Camera calibration

- from homography—Zhang's method—cont.

Since homography is up to scale, we define 3 column vectors for **H** and rewrite the equation:

$$\mathbf{H} = \mathbf{K}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}] \rightarrow [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3] = \lambda \mathbf{K}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

here, some information is important:

$$\begin{cases} \mathbf{h}_1 = \lambda \mathbf{K}\mathbf{r}_1 \\ \mathbf{h}_2 = \lambda \mathbf{K}\mathbf{r}_2 \end{cases} \rightarrow \begin{cases} \mathbf{r}_1 = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h}_2 \end{cases}$$

$r_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ $\sqrt{a^2+b^2+c^2}$

$r_1^T \cdot r_1 = (a^2+b^2+c^2)$ Recall the behavior of "**Rotation Matrix**": orthogonal & unit length

Orthogonal: inner product=0 → $\mathbf{r}_1^T\mathbf{r}_2 = 0$

unit length: the same length → $\mathbf{r}_1^T\mathbf{r}_1 = \mathbf{r}_2^T\mathbf{r}_2$ $\Rightarrow r_1^T \cdot r_1 = \text{length}^2$

Camera calibration

- from homography—Zhang's method—cont.

$$\begin{cases} \mathbf{r}_1^{\mathrm{T}}\mathbf{r}_2 = 0 \\ \mathbf{r}_1^{\mathrm{T}}\mathbf{r}_1 = \mathbf{r}_2^{\mathrm{T}}\mathbf{r}_2 \end{cases} \qquad \begin{cases} (\mathbf{K}^{-1}\mathbf{h}_1)^{\mathrm{T}}(\mathbf{K}^{-1}\mathbf{h}_2) = 0 \\ (\mathbf{K}^{-1}\mathbf{h}_1)^{\mathrm{T}}(\mathbf{K}^{-1}\mathbf{h}_1) = (\mathbf{K}^{-1}\mathbf{h}_2)^{\mathrm{T}}(\mathbf{K}^{-1}\mathbf{h}_2) \end{cases}$$

$$\rightarrow \quad \begin{cases} \mathbf{h}_1^{\mathrm{T}}(\mathbf{K}^{-\mathrm{T}}\mathbf{K}^{-1})\mathbf{h}_2 = 0 \\ \mathbf{h}_1^{\mathrm{T}}(\mathbf{K}^{-\mathrm{T}}\mathbf{K}^{-1})\mathbf{h}_1 = \mathbf{h}_2^{\mathrm{T}}(\mathbf{K}^{-\mathrm{T}}\mathbf{K}^{-1})\mathbf{h}_2 \end{cases}$$

→ The same result with absolute conic method

$$\mathbf{h}_1^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_2 = 0 \qquad \mathbf{h}_1^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_1 = \mathbf{h}_2^{\mathrm{T}}\boldsymbol{\omega}\mathbf{h}_2 \qquad \boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^{\mathrm{T}})^{-1}$$

Summary:

$$\mathbf{v}_{ij} = [h_{i1}h_{j1} \quad h_{i1}h_{j2} + h_{i2}h_{j1} \quad h_{i2}h_{j2} \quad h_{i3}h_{j1} + h_{i1}h_{j3} \quad h_{i3}h_{j2} + h_{i2}h_{j3} \quad h_{i3}h_{j3}]^{\mathrm{T}}$$

$$\begin{bmatrix} \mathbf{v}_{12}^{\mathrm{T}} \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^{\mathrm{T}} \end{bmatrix}\boldsymbol{\omega} = 0 \quad \text{and} \quad \boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^{\mathrm{T}})^{-1} \qquad \mathbf{v}_{ij}^{\mathrm{T}}\mathbf{b} = 0$$

Note, the difference between **ω** and **b**

$$\mathbf{b} = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}]^{\mathrm{T}}$$

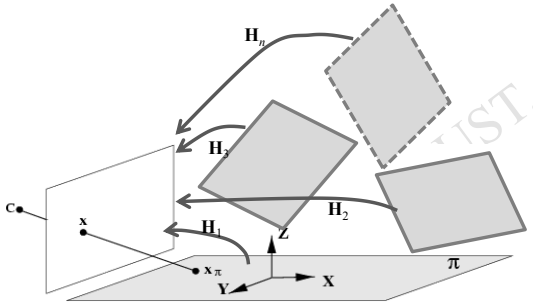
Camera calibration

- from homography—Zhang's method—cont.

$$\mathbf{r}_1 = \lambda \mathbf{K}^{-1}\mathbf{h}_1$$
$$\mathbf{r}_2 = \lambda \mathbf{K}^{-1}\mathbf{h}_2$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$
$$\mathbf{t} = \lambda \mathbf{K}^{-1}\mathbf{h}_3$$
$$\mathbf{R} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]$$
$$\mathbf{x} = \mathbf{P} \cdot \mathbf{X} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}$$

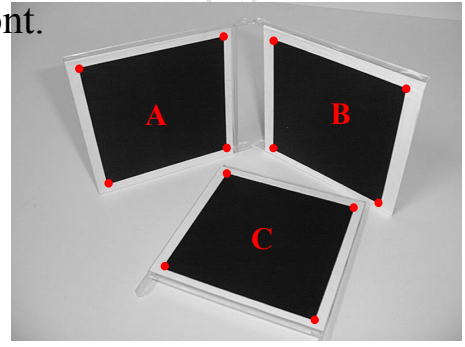
$$\lambda = \frac{1}{|\mathbf{K}^{-1}\mathbf{h}_1|} = \frac{1}{|\mathbf{K}^{-1}\mathbf{h}_2|}$$

One square has one **H** (of course, one **R**|**t**)



Camera calibration

- Example: from homography—Zhang's—cont.
  - follow the previous example



Homography (unit square to **A-square**)
379.199677 111.830818 152.000000
-64.140297 350.826263 149.000000
0.102074 0.210233 1.000000

Homography (unit square to **B-square**)
168.271439 125.763161 596.000000
81.960945 320.478027 84.000000
-0.148918 0.211012 1.000000

Homography (unit square to **C-square**)
228.969971 -209.572891 490.000000
41.616714 105.178200 387.000000
-0.078244 -0.182428 1.000000

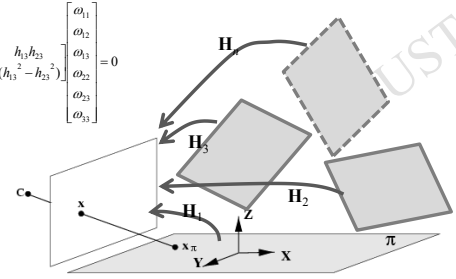
Solve equation:

$$\begin{bmatrix} h_{11}h_{21} & (h_{11}h_{22}+h_{12}h_{21}) & (h_{11}h_{23}+h_{13}h_{21}) & h_{12}h_{22} & (h_{12}h_{23}+h_{13}h_{22}) & h_{13}h_{23} \\ (h_{11}^2-h_{21}^2) & 2(h_{11}h_{12}-h_{21}h_{22}) & 2(h_{11}h_{13}-h_{21}h_{23}) & (h_{12}^2-h_{22}^2) & 2(h_{12}h_{13}-h_{22}h_{23}) & (h_{13}^2-h_{23}^2) \end{bmatrix} \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{bmatrix} = 0$$

Finally get:

$$\mathbf{K} = \begin{bmatrix} 1118.7 & -25.7 & 531.5 \\ 0 & 1100.3 & 409.6 \\ 0 & 0 & 1 \end{bmatrix}$$

What are **R**|**t** of square A, B and C?



Camera calibration

(for square A)

ha =

379.1997	111.8308	152.0000
-64.1403	350.8263	149.0000
0.1021	0.2102	1.0000

$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$

(in Matlab)

ra=inv(K)*ha

0.2883	0.0056	-0.3447
-0.0963	0.2406	-0.2368
0.1021	0.2102	1.0000

(length=0.3206) (length=0.3195) (needs adjustment)

1) scale -> orthogonal

2) $r_2 =$

3) $r_3 = r_1 \times r_2$

4) $r_2 = r_3 \times r_1$

r1= (1) $\mathbf{r}_1$

0.8991
-0.3004
0.3184

r2= (2)

0.0175
0.7504
0.6558

(not unit length)

r3=cross(r1,r2) (3) $\mathbf{r}_2$

-0.4359
-0.5840
0.6800

(not unit length)

r3= (4)

-0.4373
-0.5860
0.6822

(normalized)

r2=cross(r3,r1) (5)

0.0183
0.7526
0.6582

(unit length)

t =

-1.0751
-0.7388
3.1192

(inv(K)*h3./0.3206)

[R | t] =

0.8991	0.0183	-0.4373	-1.0751
-0.3004	0.7526	-0.5860	-0.7388
0.3184	0.6582	0.6822	3.1192

$K[RIT]_A$

Using "Gram-Schmidt" ortho-normalization

(for square B)

hb =

168.2714	125.7632	596.0000
81.9609	320.4780	84.0000
-0.1489	0.2110	1.0000

(in Matlab)

rb=inv(K)*hb

0.2242	0.0171	0.0509
0.1299	0.2127	-0.2959
-0.1489	0.2110	1.0000

[R | t] =

0.7501	0.0565	0.6589	0.1702
0.4348	0.7086	-0.5557	-0.9902
-0.4983	0.7033	0.5070	3.3463

$K[RIT]_B$ $\frac{1}{0.3206}$ t

(for square C)

hc =

228.9700	-209.5729	490.0000
41.6167	105.1782	387.0000
-0.0782	-0.1824	1.0000

(in Matlab)

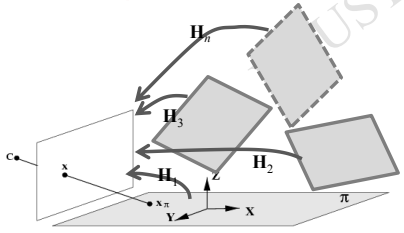
rc=inv(K)*hc

0.2434	-0.0969	-0.0376
0.0670	0.1635	-0.0205
-0.0782	-0.1824	1.0000

[R | t] =

0.9210	-0.3896	0.0083	-0.1422
0.2533	0.6148	0.7468	-0.0777
-0.2961	-0.6857	0.6649	3.7839

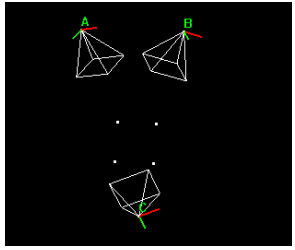
$K[RIT]_C$

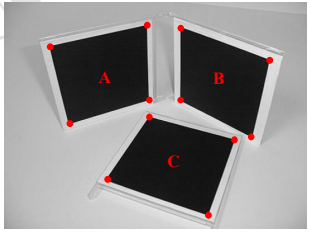


Camera calibration

- Draw camera positions relative to “Single Square”

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} u_x' & v_x' & w_x' & t_x \\ u_y' & v_y' & w_y' & t_y \\ u_z' & v_z' & w_z' & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \mathbf{X}_{\text{world}}$$

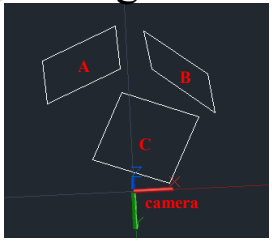


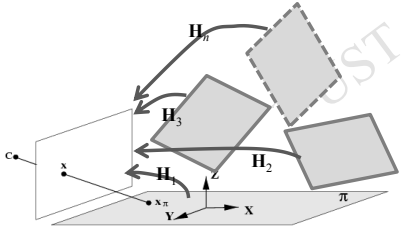


- Draw squares respective to Single camera

draw

$$[\mathbf{R}\,|\,\mathbf{t}]\begin{bmatrix}0\\0\\0\\1\end{bmatrix},[\mathbf{R}\,|\,\mathbf{t}]\begin{bmatrix}1\\0\\0\\1\end{bmatrix},[\mathbf{R}\,|\,\mathbf{t}]\begin{bmatrix}1\\1\\0\\1\end{bmatrix},[\mathbf{R}\,|\,\mathbf{t}]\begin{bmatrix}0\\1\\0\\1\end{bmatrix}$$

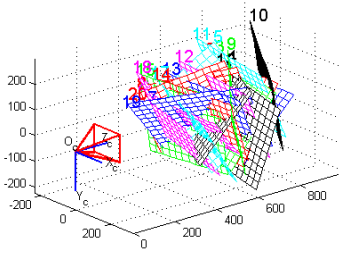


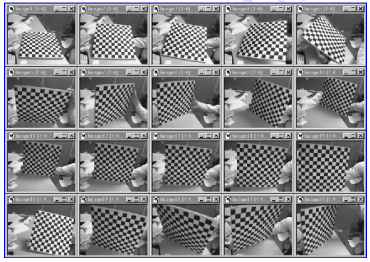


Note! the result is up to scale, unless the actual size is known

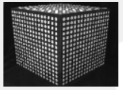
Camera calibration

- Other resources:





Camera calibration toolbox for Matlab



Modern CCD cameras are usually capable of a spatial accuracy greater than 1/50 of the pixel size. However, such accuracy is not easily attained due to various error sources that can affect the image formation process. Current calibration methods typically assume that the observations are unbiased, the only error is the zero-mean independent and identically distributed random noise in the observed image coordinates, and the camera model completely explains the mapping between the 3D coordinates and the image coordinates. In general, these conditions are not met, causing the calibration results to be less accurate than expected.

http://www.vision.caltech.edu/bouguetj/calib_doc/
http://www.ee.oulu.fi/~jth/calibr/

Camera calibration

- Deal with the lens distortion:

Camera model

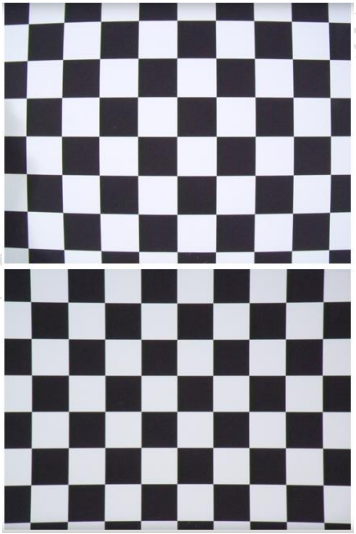
- How the image un-distorted—cont.?

$$\mathbf{x}_{id} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + k_0 r^2 + k_1 r^4 + k_4 r^6) \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} 2k_2 x_s y_s + k_3 (r^2 + 2x_s^2) \\ k_2 (r^2 + 2y_s^2) + 2k_3 x_s y_s \end{bmatrix}$$

After un-distortion, we get new 2D coordinate → which is normalized and close to "linear" perspective model

$$\mathbf{x}_p = \mathbf{K}\mathbf{x}_{id} = \begin{bmatrix} f_x & \gamma & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$$

pixel ⇐ $K \cdot \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$, K^{-1}, MCA, 2D Normlized, pixel, $(K)^{-1} \begin{pmatrix} 2D \\ 3D \\ 1 \end{pmatrix}_{2D}$



Traditional Method: Refer to this book

C. C. Slama, editor. Manual of Photogrammetry. American Society of Photogrammetry, 4th ed., 1980.

Camera calibration

- Lens distortion from Zhang's method

$[u \quad v \quad 1]^T$ →ideal points on image (distortion-free)

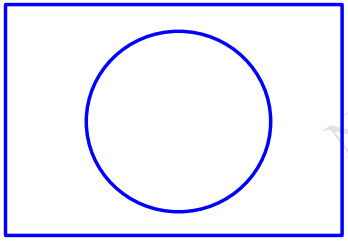
$[\breve{u} \quad \breve{v} \quad 1]^T$ →measurement points on image (distortion)

$[x \quad y \quad 1]^T$ →ideal points on real 3D space (normalized, distortion-free)

$[\breve{x} \quad \breve{y} \quad 1]^T$ →measurement on real 3D space (normalized, distortion)

Radial distortion model:

$$\begin{cases} \breve{x} = x + x[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \\ \breve{y} = y + y[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \end{cases}$$



Camera calibration

- Lens distortion from Zhang's method—cont.

$$\mathbf{x} = \mathbf{P} \cdot \mathbf{X} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}$$

$$\mathbf{K} = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$[x \quad y \quad 1]^T$

$$\begin{cases} \breve{x} = x + x[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \\ \breve{y} = y + y[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \end{cases}$$

$$\begin{cases} \breve{u} = u_0 + \alpha\breve{x} + c\breve{y} \\ \breve{v} = v_0 + \beta\breve{y} \end{cases}$$

Initial guess

$$\begin{cases} \breve{u} = u + (u-u_0)[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \\ \breve{v} = v + (v-v_0)[k_1(x^2+y^2) + k_2(x^2+y^2)^2] \end{cases}$$

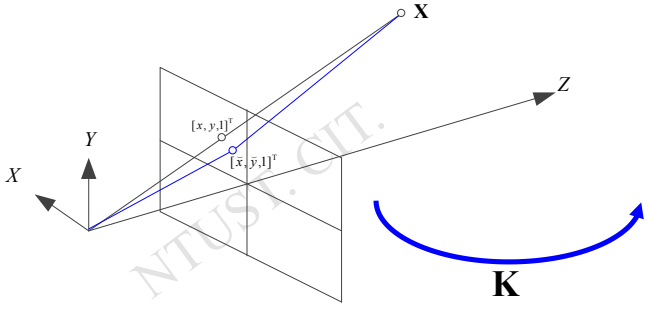
Alternatively solve k_1, k_2.

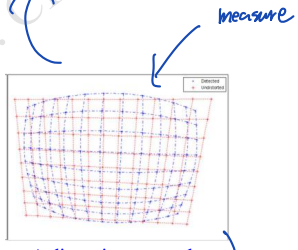
$$\begin{bmatrix} (u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u}-u \\ \breve{v}-v \end{bmatrix}$$

→solve by SVD, then iteratively minimize the error

Camera calibration

- Lens distortion from Zhang's method—cont.





CIT 色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on: facebook