# 電腦視覺與應用
# Computer Vision and Applications

Lecture-04

Estimation for 2D projective transformations


**Tzung-Han Lin**

National Taiwan University of Science and Technology

Graduate Institute of Color and Illumination Technology

e-mail: thl@mail.ntust.edu.tw

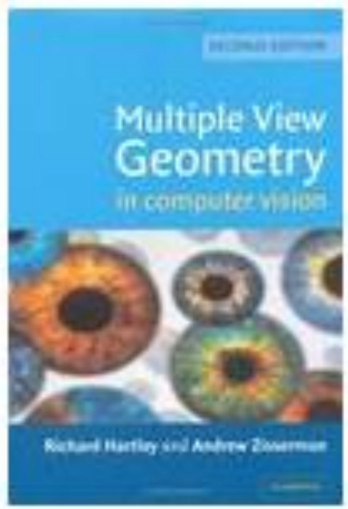


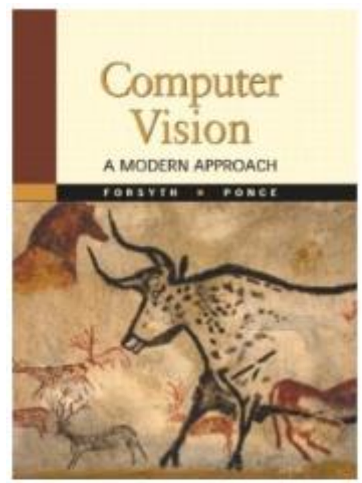
TAIWAN TECH National Taiwan University of Science and Technology

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

# Estimation for 2D projective transformations

- Lecture Reference at:
  - Multiple View Geometry in Computer Vision, Chapter 4. (major)
  - Computer Vision A Modern Approach, (NA).



# Parameter estimation

- 2D homography

  Given a set of $(\mathbf{x}_i, \mathbf{x}_i')$, compute $\mathbf{H}$ $(\mathbf{x}_i'=\mathbf{H}\mathbf{x}_i)$
- 3D to 2D camera projection

  Given a set of $(\mathbf{X}_i, \mathbf{x}_i)$, compute $\mathbf{P}$ $(\mathbf{x}_i=\mathbf{P}\mathbf{X}_i)$
- Fundamental matrix

  Given a set of $(\mathbf{x}_i, \mathbf{x}_i')$, compute $\mathbf{F}$ $(\mathbf{x}_i'^{\mathrm{T}}\mathbf{F}\mathbf{x}_i=0)$
- Trifocal tensor

  Given a set of $(\mathbf{x}_i, \mathbf{x}_i', \mathbf{x}_i'')$, compute $\mathbf{T}$

# Number of measurements required

- At least as many independent equations as degrees of freedom required
- Example:

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

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

2 independent equations / point
8 degrees of freedom

$4 \times 2 \geq 8$

# Approximate solutions

- Minimal solution
  - 4 points yield an exact solution for **H**
- More points
  - No exact solution, because measurements are inexact ("noise")
  - Search for "best" according to some cost function
  - Algebraic or geometric/statistical cost

# Gold Standard algorithm

- Cost function that is optimal for some assumptions
- Computational algorithm that minimizes it is called “Gold Standard” algorithm
- Other algorithms can then be compared to it

# Direct Linear Transformation (DLT)

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

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$ → indeed, $[0\ 0\ 0]^\mathsf{T}$

$$\mathbf{x}_i' = \left(x_i', y_i', w_i'\right)^\mathsf{T}$$

$$\mathbf{H}\mathbf{x}_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix} = \begin{pmatrix} h_{11}x_i + h_{12}y_i + h_{13}w_i \\ h_{21}x_i + h_{22}y_i + h_{23}w_i \\ h_{31}x_i + h_{32}y_i + h_{33}w_i \end{pmatrix} = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

$$\mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

here

$$\mathbf{h}^{1\mathsf{T}} = [h_{11} \quad h_{12} \quad h_{13}]$$
$$\mathbf{h}^{2\mathsf{T}} = [h_{21} \quad h_{22} \quad h_{23}]$$
$$\mathbf{h}^{3\mathsf{T}} = [h_{31} \quad h_{32} \quad h_{33}]$$

# Direct Linear Transformation (DLT)—cont.



$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = 0 \rightarrow \begin{pmatrix} x'_i \\ y'_i \\ w'_i \end{pmatrix} \times \begin{pmatrix} \mathbf{h}^{1^\top}\mathbf{x}_i \\ \mathbf{h}^{2^\top}\mathbf{x}_i \\ \mathbf{h}^{3^\top}\mathbf{x}_i \end{pmatrix} = \begin{pmatrix} y'_i\mathbf{h}^{3^\top}\mathbf{x}_i - w'_i\mathbf{h}^{2^\top}\mathbf{x}_i \\ w'_i\mathbf{h}^{1^\top}\mathbf{x}_i - x'_i\mathbf{h}^{3^\top}\mathbf{x}_i \\ x'_i\mathbf{h}^{2^\top}\mathbf{x}_i - y'_i\mathbf{h}^{1^\top}\mathbf{x}_i \end{pmatrix} = 0$$

$$\rightarrow \begin{pmatrix} y'_i\mathbf{x}_i^\top\mathbf{h}^3 - w'_i\mathbf{x}_i^\top\mathbf{h}^2 \\ w'_i\mathbf{x}_i^\top\mathbf{h}^1 - x'_i\mathbf{x}_i^\top\mathbf{h}^3 \\ x'_i\mathbf{x}_i^\top\mathbf{h}^2 - y'_i\mathbf{x}_i^\top\mathbf{h}^1 \end{pmatrix} = \begin{pmatrix} 0^\top\mathbf{h}^1 - w'_i\mathbf{x}_i^\top\mathbf{h}^2 + y'_i\mathbf{x}_i^\top\mathbf{h}^3 \\ w'_i\mathbf{x}_i^\top\mathbf{h}^1 + 0^\top\mathbf{h}^2 - x'_i\mathbf{x}_i^\top\mathbf{h}^3 \\ -y'_i\mathbf{x}_i^\top\mathbf{h}^1 + x'_i\mathbf{x}_i^\top\mathbf{h}^2 + 0^\top\mathbf{h}^3 \end{pmatrix} = 0$$

# Direct Linear Transformation (DLT)—cont.

$$\begin{bmatrix} \mathbf{0}^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \\ -y_i'\mathbf{x}_i^\mathsf{T} & x_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

- Indeed, the above equation is an abbreviation of the following format:

$$\begin{bmatrix} [0 \quad 0 \quad 0] & -w_i'[x_i \quad y_i \quad w_i] & y_i'[x_i \quad y_i \quad w_i] \\ w_i'[x_i \quad y_i \quad w_i] & [0 \quad 0 \quad 0] & -x_i'[x_i \quad y_i \quad w_i] \\ -y_i'[x_i \quad y_i \quad w_i] & x_i'[x_i \quad y_i \quad w_i] & [0 \quad 0 \quad 0] \end{bmatrix} \begin{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \end{pmatrix} \\ \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix} \\ \begin{pmatrix} h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} \end{pmatrix} = 0 \quad \Rightarrow \quad \mathbf{A}_i\mathbf{h} = 0$$

# Direct Linear Transformation (DLT)—cont.

- Equations are linear in **h**

$$\mathbf{A}_i\mathbf{h} = 0$$

- Only 2 out of 3 are linearly independent (WHY?) (indeed, 2 eq/pt)

$$\begin{bmatrix} 0^{\mathsf T} & -w'_i\mathbf{x}_i^{\mathsf T} & y'_i\mathbf{x}_i^{\mathsf T} \\ w'_i\mathbf{x}_i^{\mathsf T} & 0^{\mathsf T} & -x'_i\mathbf{x}_i^{\mathsf T} \\ -y'_i\mathbf{x}_i^{\mathsf T} & x'_i\mathbf{x}_i^{\mathsf T} & 0^{\mathsf T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

- Since equation 3 (3th row) could be a linear combination of equation 1 and equation 2. Example:

$$\begin{aligned} &\text{Eq1: } -w'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^2 + y'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^3 = 0 \\ &\text{Eq2: } w'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^1 - x'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^3 = 0 \\ &\text{Eq3: } -y'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^1 + x'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^2 = 0 \end{aligned} \quad \rightarrow$$

eliminated

$$\begin{aligned} &x'_i(\text{Eq1}) + y'_i(\text{Eq2}) = 0 \\ &x'_i(-w'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^2 + \boxed{y'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^3}) + y'_i(w'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^1 \boxed{-x'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^3}) = 0 \\ &(-w'_i)(-y'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^1 + x'_i\mathbf{x}_i^{\mathsf T}\mathbf{h}^2) = 0 \\ &(-w'_i)(\text{Eq3}) = 0 \end{aligned}$$

# Direct Linear Transformation (DLT)—cont.



- Equations are linear in **h**

$$\mathbf{A}_i \mathbf{h} = 0$$

$$\begin{bmatrix} 0^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & 0^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

- Holds for any homogeneous representation, e.g. ($x_i'$, $y_i'$, 1)

# Direct Linear Transformation (DLT)—cont.

- Solving for **H**

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} \mathbf{h} = 0 \qquad \mathbf{A}\mathbf{h} = 0$$

size **A** is 8x9 or 12x9, but rank 8

Trivial solution is $\mathbf{h}=0_9^T$ is not interesting

1-D null-space yields solution of interest

pick for example the one with $\|\mathbf{h}\| = 1$

# Direct Linear Transformation (DLT)—cont.

- Over-determined solution

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \mathbf{h} = 0 \qquad\qquad \mathbf{Ah} = 0$$

No exact solution because of inexact measurement

i.e. “noise”

Find approximate solution

- Additional constraint needed to avoid 0, e.g. $\|\mathbf{h}\| = 1$
- $\mathrm{Ah} = 0$ not possible, so minimize $\|\mathbf{Ah}\|$

# DLT algorithm

- Summary:

<u>Objective</u>

Given n≥4 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$, determine the 2D homography matrix **H** such that $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$

<u>Algorithm</u>

1) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ compute $\mathbf{A}_i$. Usually only two first rows needed.
2) Assemble $n$ (2 x 9) matrices $\mathbf{A}_i$ into a single ($2n$ x 9) matrix **A**
3) Obtain SVD of **A**. Solution for **h** is last column of V
4) Determine **H** from **h**

# DLT algorithm—in practice:

- For **ONE** correspondence, you will have **TWO** equations:

$$\begin{bmatrix} 0^\top & -w_i'\mathbf{x}_i^\top & y_i'\mathbf{x}_i^\top \\ w_i'\mathbf{x}_i^\top & 0^\top & -x_i'\mathbf{x}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 & 0 & -w_i'x_i & -w_i'y_i & -w_i'w_i & y'x_i & y'y_i & y'w_i \\ w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

# DLT algorithm—in practice:

- For *n* correspondence you have two equations:

$$\mathbf{Ah} = 0$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -x_1 w_1' & -y_1 w_1' & -w_1 w_1' & x_1 y_1' & y_1 y_1' & w_1 y_1' \\ x_1 w_1' & y_1 w_1' & w_1 w_1' & 0 & 0 & 0 & -x_1 x_1' & -y_1 x_1' & -w_1 x_1' \\ 0 & 0 & 0 & -x_2 w_2' & -y_2 w_2' & -w_2 w_2' & x_2 y_2' & y_2 y_2' & w_2 y_2' \\ x_2 w_2' & y_2 w_2' & w_2 w_2' & 0 & 0 & 0 & -x_2 x_2' & -y_2 x_2' & -w_2 x_2' \\ & & & & \ldots\ldots & & & & \\ 0 & 0 & 0 & -x_n w_n' & -y_n w_n' & -w_n w_n' & x_n y_n' & y_n y_n' & w_n y_n' \\ x_n w_n' & y_n w_n' & w_n w_n' & 0 & 0 & 0 & -x_n x_n' & -y_n x_n' & -w_n x_n' \end{bmatrix} \begin{array}{l} \text{From 1st correspondence} \\ \\ \text{From 2nd correspondence} \\ \\ \cdots \\ \text{From } n\text{-th correspondence} \\ \\ \end{array}$$

- Using singular value decomposition (SVD) :

$$\mathbf{A} = \mathbf{USV}^{\mathrm{T}}$$

$$\begin{pmatrix} \mathbf{A} \end{pmatrix} = \begin{pmatrix} \mathbf{U} \end{pmatrix} \cdot \begin{pmatrix} w_0 & & & \\ & w_1 & & \\ & & \cdots & \\ & & \cdots & \\ & & & w_{N-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{V}^T \end{pmatrix} \qquad (2.6.2)$$

Reference book: Numerical recipes

# DLT algorithm—example:

- For example: a planar card on 3D environment. Six set correspondences are detected in image-AB, image-BC and image-CA

# DLT algorithm—example:—cont.

- Find all correspondences:


$pA1=[651,386,1]^T$ $pB1=[459,392,1]^T$ $pC1=[522,406,1]^T$

$pA2=[576,696,1]^T$ $pB2=[282,667,1]^T$ $pC2=[446,688,1]^T$

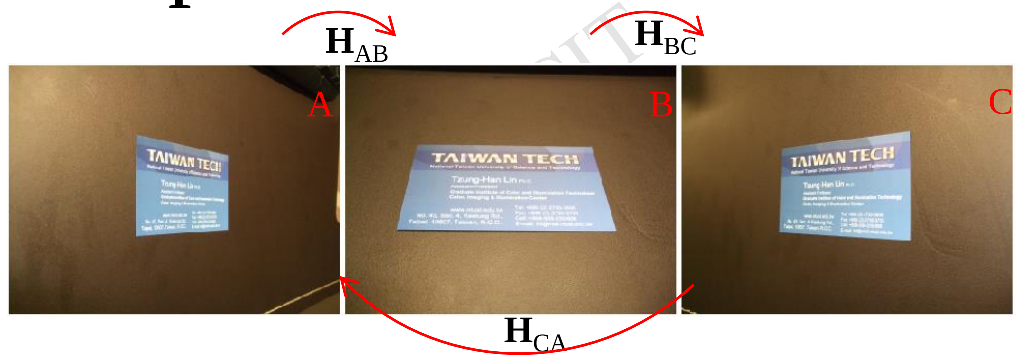
$pA3=[730,651,1]^T$ $pB3=[592,629,1]^T$ $pC3=[605,657,1]^T$

$pA4=[859,686,1]^T$ $pB4=[913,677,1]^T$ $pC4=[801,708,1]^T$

$pA5=[784,509,1]^T$ $pB5=[711,484,1]^T$ $pC5=[682,499,1]^T$

$pA6=[916,460,1]^T$ $pB6=[1009,424,1]^T$ $pC6=[918,402,1]^T$

# DLT algorithm—example:—cont.




- Equation:

$$\mathbf{Ah} = 0$$

Aab=[0 0 0 -pB1(3)*pA1' pB1(2)*pA1'; pB1(3)*pA1' 0 0 0 -pB1(1)*pA1';
0 0 0 -pB2(3)*pA2' pB2(2)*pA2'; pB2(3)*pA2' 0 0 0 -pB2(1)*pA2';
0 0 0 -pB3(3)*pA3' pB3(2)*pA3'; pB3(3)*pA3' 0 0 0 -pB3(1)*pA3';
0 0 0 -pB4(3)*pA4' pB4(2)*pA4'; pB4(3)*pA4' 0 0 0 -pB4(1)*pA4';
0 0 0 -pB5(3)*pA5' pB5(2)*pA5'; pB5(3)*pA5' 0 0 0 -pB5(1)*pA5';
0 0 0 -pB6(3)*pA6' pB6(2)*pA6'; pB6(3)*pA6' 0 0 0 -pB6(1)*pA6';]
Abc=[0 0 0 -pC1(3)*pB1' pC1(2)*pB1'; pC1(3)*pB1' 0 0 0 -pC1(1)*pB1';
0 0 0 -pC2(3)*pB2' pC2(2)*pB2'; pC2(3)*pB2' 0 0 0 -pC2(1)*pB2';
0 0 0 -pC3(3)*pB3' pC3(2)*pB3'; pC3(3)*pB3' 0 0 0 -pC3(1)*pB3';
0 0 0 -pC4(3)*pB4' pC4(2)*pB4'; pC4(3)*pB4' 0 0 0 -pC4(1)*pB4';
0 0 0 -pC5(3)*pB5' pC5(2)*pB5'; pC5(3)*pB5' 0 0 0 -pC5(1)*pB5';
0 0 0 -pC6(3)*pB6' pC6(2)*pB6'; pC6(3)*pB6' 0 0 0 -pC6(1)*pB6';]
Aca=[0 0 0 -pA1(3)*pC1' pA1(2)*pC1'; pA1(3)*pC1' 0 0 0 -pA1(1)*pC1';
0 0 0 -pA2(3)*pC2' pA2(2)*pC2'; pA2(3)*pC2' 0 0 0 -pA2(1)*pC2';
0 0 0 -pA3(3)*pC3' pA3(2)*pC3'; pA3(3)*pC3' 0 0 0 -pA3(1)*pC3';
0 0 0 -pA4(3)*pC4' pA4(2)*pC4'; pA4(3)*pC4' 0 0 0 -pA4(1)*pC4';
0 0 0 -pA5(3)*pC5' pA5(2)*pC5'; pA5(3)*pC5' 0 0 0 -pA5(1)*pC5';
0 0 0 -pA6(3)*pC6' pA6(2)*pC6'; pA6(3)*pC6' 0 0 0 -pA6(1)*pC6';]

Aab =

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -651 | -386 | -1 | 255192 | 151312 | 392 |
| 651 | 386 | 1 | 0 | 0 | 0 | -298809 | -177174 | -459 |
| 0 | 0 | 0 | -576 | -696 | -1 | 384192 | 464232 | 667 |
| 576 | 696 | 1 | 0 | 0 | 0 | -162432 | -196272 | -282 |
| 0 | 0 | 0 | -730 | -651 | -1 | 459170 | 409479 | 629 |
| 730 | 651 | 1 | 0 | 0 | 0 | -432160 | -385392 | -592 |
| 0 | 0 | 0 | -859 | -686 | -1 | 581543 | 464422 | 677 |
| 859 | 686 | 1 | 0 | 0 | 0 | -784267 | -626318 | -913 |
| 0 | 0 | 0 | -784 | -509 | -1 | 379456 | 246356 | 484 |
| 784 | 509 | 1 | 0 | 0 | 0 | -557424 | -361899 | -711 |
| 0 | 0 | 0 | -916 | -460 | -1 | 388384 | 195040 | 424 |
| 916 | 460 | 1 | 0 | 0 | 0 | -924244 | -464140 | -1009 |

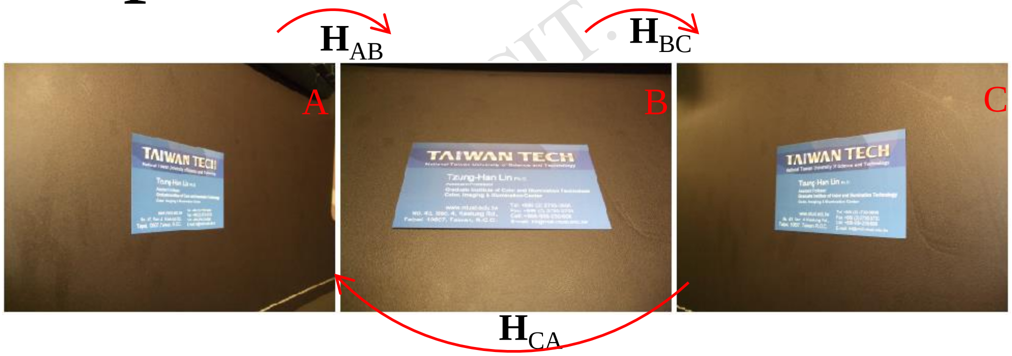
Abc =

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -459 | -392 | -1 | 185436 | 158368 | 404 |
| 459 | 392 | 1 | 0 | 0 | 0 | -221238 | -188944 | -482 |
| 0 | 0 | 0 | -282 | -667 | -1 | 194016 | 458896 | 688 |
| 282 | 667 | 1 | 0 | 0 | 0 | -125772 | -297482 | -446 |
| 0 | 0 | 0 | -592 | -629 | -1 | 388944 | 413253 | 657 |
| 592 | 629 | 1 | 0 | 0 | 0 | -358160 | -380545 | -605 |
| 0 | 0 | 0 | -913 | -677 | -1 | 646404 | 479316 | 708 |
| 913 | 677 | 1 | 0 | 0 | 0 | -731313 | -542277 | -801 |
| 0 | 0 | 0 | -711 | -484 | -1 | 354789 | 241516 | 499 |
| 711 | 484 | 1 | 0 | 0 | 0 | -484902 | -330088 | -682 |
| 0 | 0 | 0 | -1009 | -424 | -1 | 405618 | 170448 | 402 |
| 1009 | 424 | 1 | 0 | 0 | 0 | -926262 | -389232 | -918 |

Aca =

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -482 | -404 | -1 | 186052 | 155944 | 386 |
| 482 | 404 | 1 | 0 | 0 | 0 | -313782 | -263004 | -651 |
| 0 | 0 | 0 | -446 | -688 | -1 | 310416 | 478848 | 696 |
| 446 | 688 | 1 | 0 | 0 | 0 | -256896 | -396288 | -576 |
| 0 | 0 | 0 | -605 | -657 | -1 | 393855 | 427707 | 651 |
| 605 | 657 | 1 | 0 | 0 | 0 | -441650 | -479610 | -730 |
| 0 | 0 | 0 | -801 | -708 | -1 | 549486 | 485688 | 686 |
| 801 | 708 | 1 | 0 | 0 | 0 | -688059 | -608172 | -859 |
| 0 | 0 | 0 | -682 | -499 | -1 | 347138 | 253991 | 509 |
| 682 | 499 | 1 | 0 | 0 | 0 | -534688 | -391216 | -784 |
| 0 | 0 | 0 | -918 | -402 | -1 | 422280 | 184920 | 460 |
| 918 | 402 | 1 | 0 | 0 | 0 | -840888 | -368232 | -916 |

# DLT algorithm—example:—cont.




- Solve by SVD

$$\mathbf{Ah} = 0$$

Solve by SVD method.

Example in Matlab:

```
[U,S,V]=svd(Aab)
Hab=[V(1:3,9)';V(4:6,9)';V(7:9,9)']
```

→last column

```
Hab =
  -0.0027   0.0007   0.6249
   0.0007  -0.0010  -0.7807
   0.0000   0.0000  -0.0031
Hbc =
   0.0023   0.0020   0.9096
  -0.0011   0.0077  -0.4155
  -0.0000   0.0000   0.0047
Hca =
  -0.0032  -0.0001   0.5620
  -0.0014  -0.0020   0.8271
  -0.0000  -0.0000  -0.0006
```

$$\mathbf{H}_{CA} \approx (\mathbf{H}_{BC}\mathbf{H}_{AB})^{-1}$$

Up to scale

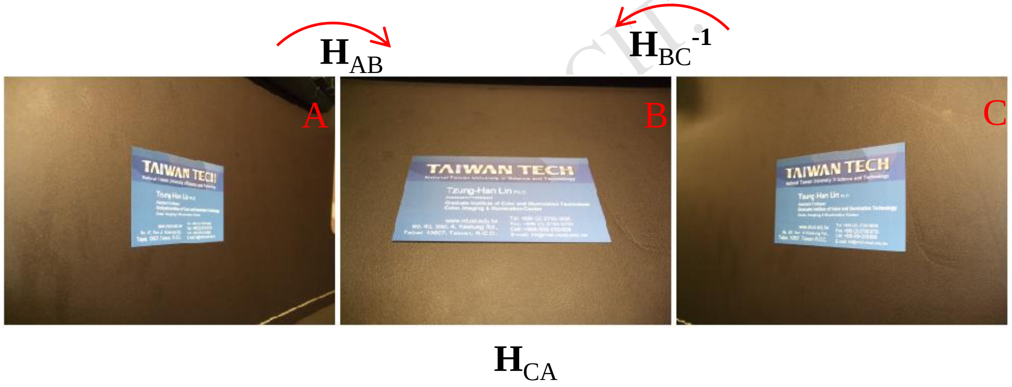
estimated points measured points

$$\bar{p}_B = \mathbf{H}_{AB}(p_A)$$
$$\bar{p}_C = \mathbf{H}_{BC}(p_B)$$
$$\bar{p}_A = \mathbf{H}_{CA}(p_C)$$

$$\bar{\bar{p}}_C = \mathbf{H}_{BC}\mathbf{H}_{AB}(p_A)$$

# DLT algorithm—example:—cont.

- Verify points




**Measurement**

$pB1=[459,392,1]^T$
$pB2=[282,667,1]^T$
$pB3=[592,629,1]^T$
$pB4=[913,677,1]^T$
$pB5=[711,484,1]^T$
$pB6=[1009,424,1]^T$

**Estimation from image A to B**

| | | |
|---|---|---|
| $\mathbf{H}_{AB}$*pA1= | 459.3547 | 391.963 |
| $\mathbf{H}_{AB}$*pA2= | 281.7844 | 667.7953 |
| $\mathbf{H}_{AB}$*pA3= | 593.3631 | 628.0585 |
| $\mathbf{H}_{AB}$*pA4= | 912.6841 | 677.0122 |
| $\mathbf{H}_{AB}$*pA5= | 708.9633 | 483.3161 |
| $\mathbf{H}_{AB}$*pA6= | 1009.858 | 424.8571 |

**Note: 3rd element is 1**

**Estimation from image C to B**

| | | |
|---|---|---|
| $\mathbf{H}_{BC}^{-1*}$pC1= | 458.6791 | 392.246 |
| $\mathbf{H}_{BC}^{-1*}$pC2 = | 282.0898 | 667.0208 |
| $\mathbf{H}_{BC}^{-1*}$pC3 = | 592.38 | 629.2953 |
| $\mathbf{H}_{BC}^{-1*}$pC4 = | 912.6669 | 677.023 |
| $\mathbf{H}_{BC}^{-1*}$pC5 = | 710.8009 | 483.0512 |
| $\mathbf{H}_{BC}^{-1*}$pC6 = | 1009.367 | 424.3639 |

**Note: 3rd element is 1**

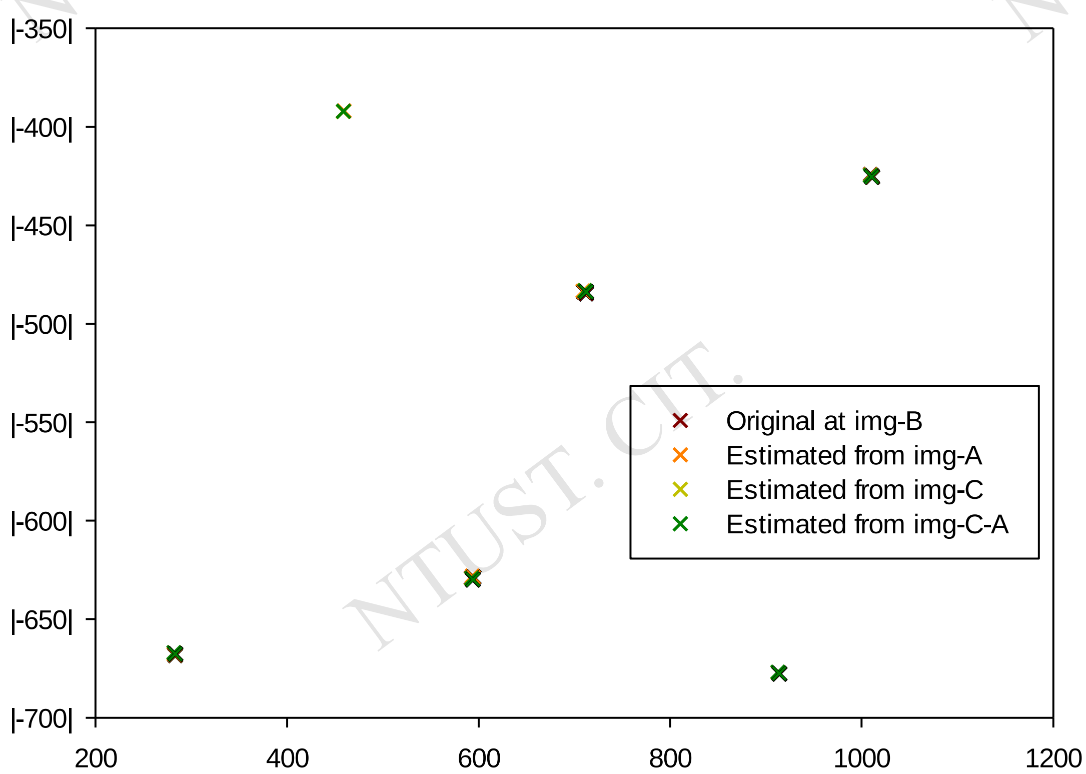
**Estimation from images C to A to B**

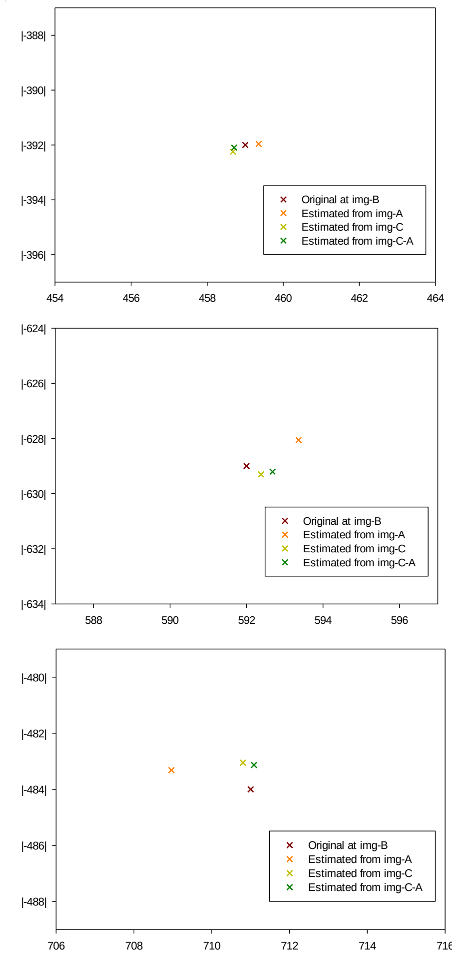
| | | |
|---|---|---|
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC1= | 458.7103 | 392.0953 |
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC2 = | 281.851 | 666.9389 |
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC3 = | 592.6809 | 629.201 |
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC4 = | 912.5832 | 676.8083 |
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC5 = | 711.0876 | 483.1301 |
| $\mathbf{H}_{AB}^{*}\mathbf{H}_{CA}^{*}$pC6 = | 1009.163 | 424.7749 |

**Note: 3rd element is 1**

# DLT algorithm—example:—cont.

- Reprojection error:



# DLT algorithm—example:—cont.

- With comparison to openCV result.



Matlab (using SVD)

Hab =

| | | |
|---|---|---|
| 0.8816 | -0.2139 | -204.4555 |
| -0.2171 | 0.3386 | 255.4139 |
| -0.0004 | -0.0003 | 1.0000 |

Hbc =

| | | |
|---|---|---|
| 0.4706 | 0.4011 | 199.6538 |
| -0.2304 | 1.5985 | -75.7620 |
| -0.0004 | 0.0007 | 1.0000 |

Hca =

1.0e+003 *

| | | |
|---|---|---|
| 0.0058 | 0.0002 | -1.0419 |
| 0.0025 | 0.0036 | -1.5110 |
| 0.0000 | 0.0000 | 0.0010 |

OpenCV (stored as float)

Homography Matrix (A to B):
0.879630 -0.214684 -203.041306
-0.217263 0.337555 255.723053
-0.000377 -0.000339 1.000000

Homography Matrix (B to C):
0.471623 0.402092 199.173584
-0.230184 1.600397 -76.327538
-0.000360 0.000672 1.000000

Homography Matrix (C to A):
5.713303 0.233439 -1023.777222
2.430560 3.548679 -1491.053589
0.003936 0.000247 1.000000

OpenCV (stored as double)

Homography Matrix (A to B):
0.879630 -0.214684 -203.041299
-0.217263 0.337555 255.723051
-0.000377 -0.000339 1.000000

Homography Matrix (B to C):
0.471623 0.402092 199.173589
-0.230184 1.600397 -76.327540
-0.000360 0.000672 1.000000

Homography Matrix (C to A):
5.713303 0.233439 -1023.777224
2.430560 3.548679 -1491.053634
0.003936 0.000247 1.000000

# Inhomogeneous solution

- Since **h** can only be computed up to scale, pick $h_{33}$=1, and solve for 8-vector

$$\begin{bmatrix} 0 & 0 & 0 & -x_i w_i' & -y_i w_i' & -w_i w_i' & x_i y_i' & y_i y_i' \\ x_i w_i' & y_i w_i' & w_i w_i' & 0 & 0 & 0 & x_i x_i' & y_i x_i' \end{bmatrix} \tilde{\mathbf{h}} = \begin{pmatrix} -w_i y_i' \\ w_i x_i' \end{pmatrix}$$

- Solve using Gaussian elimination (4 points) or using linear least-squares (more than 4 points)
- However, if $h_{33}$=0 this approach fails also poor results if $h_9$ close to zero. Therefore, not recommended.
- Note $h_{33}$=0 if origin is mapped to infinity $\mathbf{l}_\infty^{\mathsf{T}} \mathbf{H} \mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{H} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$

# Inhomogeneous solution—cont.

- Least square method: (n>4)

$$
\begin{bmatrix}
\begin{bmatrix} 0 & 0 & 0 & -x_1w_1' & -y_1w_1' & -w_1w_1' & x_1y_1' & y_1y_1' \\ x_1w_1' & y_1w_1' & w_1w_1' & 0 & 0 & 0 & x_1x_1' & y_1x_1' \end{bmatrix} \\
\begin{bmatrix} 0 & 0 & 0 & -x_2w_2' & -y_2w_2' & -w_2w_2' & x_2y_2' & y_2y_2' \\ x_2w_2' & y_2w_2' & w_2w_2' & 0 & 0 & 0 & x_2x_2' & y_2x_2' \end{bmatrix} \\
\cdots \\
\begin{bmatrix} 0 & 0 & 0 & -x_nw_n' & -y_nw_n' & -w_nw_n' & x_ny_n' & y_ny_n' \\ x_nw_n' & y_nw_n' & w_nw_n' & 0 & 0 & 0 & x_nx_n' & y_nx_n' \end{bmatrix}
\end{bmatrix}
\tilde{\mathbf{h}} =
\begin{pmatrix} -w_1y_1' \\ -w_1x_1' \\ -w_2y_2' \\ -w_2x_2' \\ \cdots \\ -w_ny_n' \\ -w_nx_n' \end{pmatrix}
\begin{array}{l} \text{From 1st correspondence} \\ \text{From 2nd correspondence} \\ \cdots \\ \text{From } n\text{-th correspondence} \end{array}
$$

- Rewrite as a matrix form:

$$\mathbf{N}_{2n\times 8}\tilde{\mathbf{h}}_{8\times 1} = \mathbf{d}_{2n\times 1}$$

# Inhomogeneous solution—cont.

- Solution for the matrix

known (given, or calculated by feature detection algorithms)

$$\mathbf{N}_{2n\times 8}\tilde{\mathbf{h}}_{8\times 1} = \mathbf{d}_{2n\times 1}$$

unknown (to be determined)

- Apply a transpose of N to the equation :

$$[\mathbf{N}^T]_{8\times 2n}[\mathbf{N}]_{2n\times 8}\tilde{\mathbf{h}}_{8\times 1} = [\mathbf{N}^T]_{8\times 2n}\mathbf{d}_{2n\times 1}$$

- Let: $\mathbf{M}_{8\times 8} = [\mathbf{N}^T]_{8\times 2n}[\mathbf{N}]_{2n\times 8}$

$$\tilde{\mathbf{d}}_{8\times 1} = [\mathbf{N}^T]_{8\times 2n}\mathbf{d}_{2n\times 1}$$

- Then, $\mathbf{M}_{8\times 8}\tilde{\mathbf{h}}_{8\times 1} = \tilde{\mathbf{d}}_{8\times 1}$

$$\tilde{\mathbf{h}}_{8\times 1} = [\mathbf{M}^{-1}]_{8\times 8}\tilde{\mathbf{d}}_{8\times 1} = \left([\mathbf{N}^T]_{8\times 2n}[\mathbf{N}]_{2n\times 8}\right)^{-1}[\mathbf{N}^T]_{8\times 2n}\mathbf{d}_{2n\times 1}$$

# Inhomogeneous solution—example

- For example (the same with previous example):

$$\mathbf{N}_{2n\times 8}\tilde{\mathbf{h}}_{8\times 1}=\mathbf{d}_{2n\times 1}$$

Nab =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -651 | -386 | -1 | 255192 | 151312 |
| 651 | 386 | 1 | 0 | 0 | 0 | -298809 | -177174 |
| 0 | 0 | 0 | -576 | -696 | -1 | 384192 | 464232 |
| 576 | 696 | 1 | 0 | 0 | 0 | -162432 | -196272 |
| 0 | 0 | 0 | -730 | -651 | -1 | 459170 | 409479 |
| 730 | 651 | 1 | 0 | 0 | 0 | -432160 | -385392 |
| 0 | 0 | 0 | -859 | -686 | -1 | 581543 | 464422 |
| 859 | 686 | 1 | 0 | 0 | 0 | -784267 | -626318 |
| 0 | 0 | 0 | -784 | -509 | -1 | 379456 | 246356 |
| 784 | 509 | 1 | 0 | 0 | 0 | -557424 | -361899 |
| 0 | 0 | 0 | -916 | -460 | -1 | 388384 | 195040 |
| 916 | 460 | 1 | 0 | 0 | 0 | -924244 | -464140 |

dab =

| |
|---|
| -392 |
| 459 |
| -667 |
| 282 |
| -629 |
| 592 |
| -677 |
| 913 |
| -484 |
| 711 |
| -424 |
| 1009 |

$$\tilde{\mathbf{h}}_{8\times 1}=\left([\mathbf{N}^{\mathrm{T}}]_{8\times 2n}[\mathbf{N}]_{2n\times 8}\right)^{-1}[\mathbf{N}^{\mathrm{T}}]_{8\times 2n}\mathbf{d}_{2n\times 1}$$

Implement in Matlab:

```
hab=inv(Nab'*Nab)*Nab'*dab
```

hab =

| |
|---|
| 0.8803 |
| -0.2141 |
| -203.6746 |
| -0.2172 |
| 0.3381 |
| 255.5480 |
| -0.0004 |
| -0.0003 |

Hab =

| | | |
|---|---|---|
| 0.8803 | -0.2141 | -203.6746 |
| -0.2172 | 0.3381 | 255.5480 |
| -0.0004 | -0.0003 | 1.0000 |

# Inhomogeneous solution—example

- Compare result with SVD method:

Matlab (using inhomogenous sol.)

Hab =

```
  0.8803   -0.2141 -203.6746
 -0.2172    0.3381  255.5480
 -0.0004   -0.0003    1.0000
```

Hbc =

```
  0.4710    0.4017  199.4039
 -0.2304    1.5995  -76.0091
 -0.0004    0.0007    1.0000
```

Hca =

```
1.0e+003 *

  0.0056    0.0002   -1.0019
  0.0024    0.0035   -1.4654
  0.0000    0.0000    0.0010
```

Matlab (using SVD)

Hab =

```
  0.8816   -0.2139 -204.4555
 -0.2171    0.3386  255.4139
 -0.0004   -0.0003    1.0000
```

Hbc =

```
  0.4706    0.4011  199.6538
 -0.2304    1.5985  -75.7620
 -0.0004    0.0007    1.0000
```

Hca =

```
1.0e+003 *

  0.0058    0.0002   -1.0419
  0.0025    0.0036   -1.5110
  0.0000    0.0000    0.0010
```

OpenCV (stored as float)

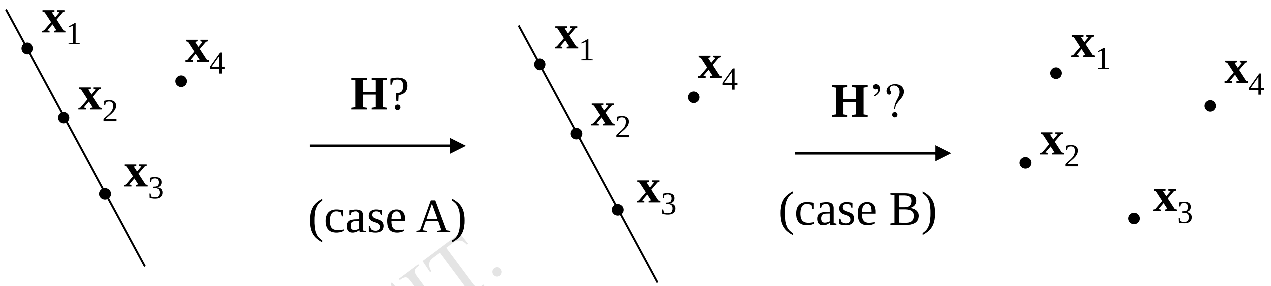
Homography Matrix (A to B):
0.879630 -0.214684 -203.041306
-0.217263 0.337555 255.723053
-0.000377 -0.000339 1.000000

Homography Matrix (B to C):
0.471623 0.402092 199.173584
-0.230184 1.600397 -76.327538
-0.000360 0.000672 1.000000

Homography Matrix (C to A):
5.713303 0.233439 -1023.777222
2.430560 3.548679 -1491.053589
0.003936 0.000247 1.000000

# Degenerate configurations

- Sometimes, degeneration happens…



# Cost functions (for evaluation)

- Algebraic distance
- Geometric distance
- Re-projection error

# Cost functions (for evaluation)

- Algebraic distance

DLT minimizes $\|\mathbf{Ah}\|$

$\boldsymbol{\varepsilon} = \mathbf{Ah}$ residual vector

*algebraic error vector*

$\boldsymbol{\varepsilon}_i$ partial vector for each ($\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$)

*algebraic distance*

$$\sum_i d_{\mathrm{alg}}(\mathbf{x}_i', \mathbf{Hx}_i)^2 = \sum_i \|\boldsymbol{\varepsilon}_i\|^2 = \|\mathbf{Ah}\|^2 = \|\boldsymbol{\varepsilon}\|^2$$

Not geometrically/statistically meaningful, but given good normalization it works fine and is very fast (use for initialization)

# Cost functions (for evaluation)

- Algebraic distance—cont.

## DLT –original format

(one correspondence)

$$\begin{bmatrix} 0 & 0 & 0 & -w_i'x_i & -w_i'y_i & -w_i'w_i & y'x_i & y'y_i & y'w_i \\ w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

governing equation, or so called constraint

unknown (to be determined)

known (given, or by feature detection)

## Algebraic distance

(one correspondence)

known (has been determined)

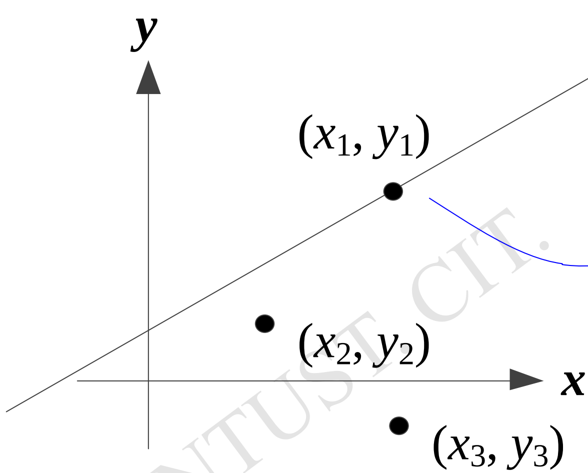
evaluated cost values

$$\begin{bmatrix} 0 & 0 & 0 & -w_i'x_i & -w_i'y_i & -w_i'w_i & y'x_i & y'y_i & y'w_i \\ w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} \varepsilon_{xi} \\ \varepsilon_{yi} \end{bmatrix} = \boldsymbol{\varepsilon}_i$$

Cost function

# Cost functions (for evaluation)

- Algebraic distance—cont.
  - A very simple example by the line equation




line equation:

$ax + by + c = 0$

$ax_1 + by_1 + c = 0$

$ax_2 + by_2 + c = \varepsilon_2 \neq 0$

$ax_3 + by_3 + c = \varepsilon_3 \neq 0$

# Cost functions (for evaluation)

- Geometric distance
  - Error in one image

$$\sum_i d\left(\mathbf{x}'_i, \mathbf{H}\tilde{\mathbf{x}}_i\right)^2$$

$\mathbf{x}$ measured coordinates
$\hat{\mathbf{x}}$ estimated coordinates
$\tilde{\mathbf{x}}$ true coordinates
$d(.,.)$ Euclidean distance (in image)

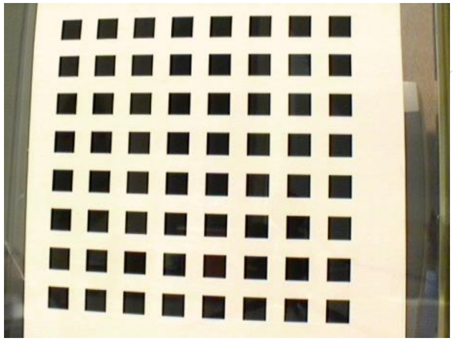
  - Symmetric transfer error

$$\sum_i [d(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2]$$

  - Reprojection error

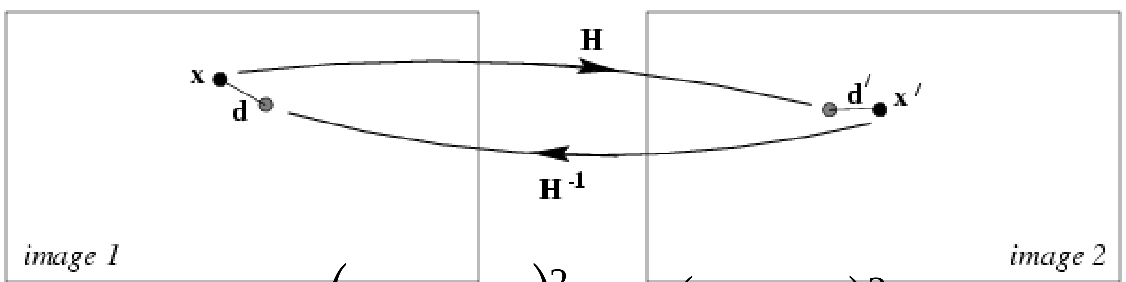
$$\sum_i [d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2]$$

e.g. calibration pattern

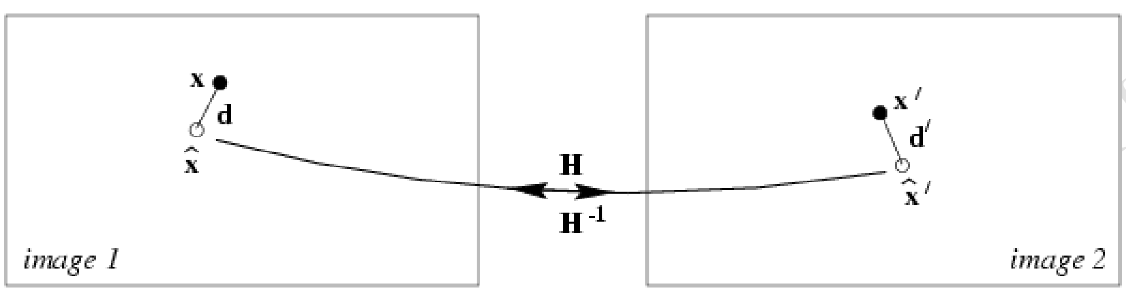


# Cost functions (for evaluation)

- Geometric distance
  - Reprojection error



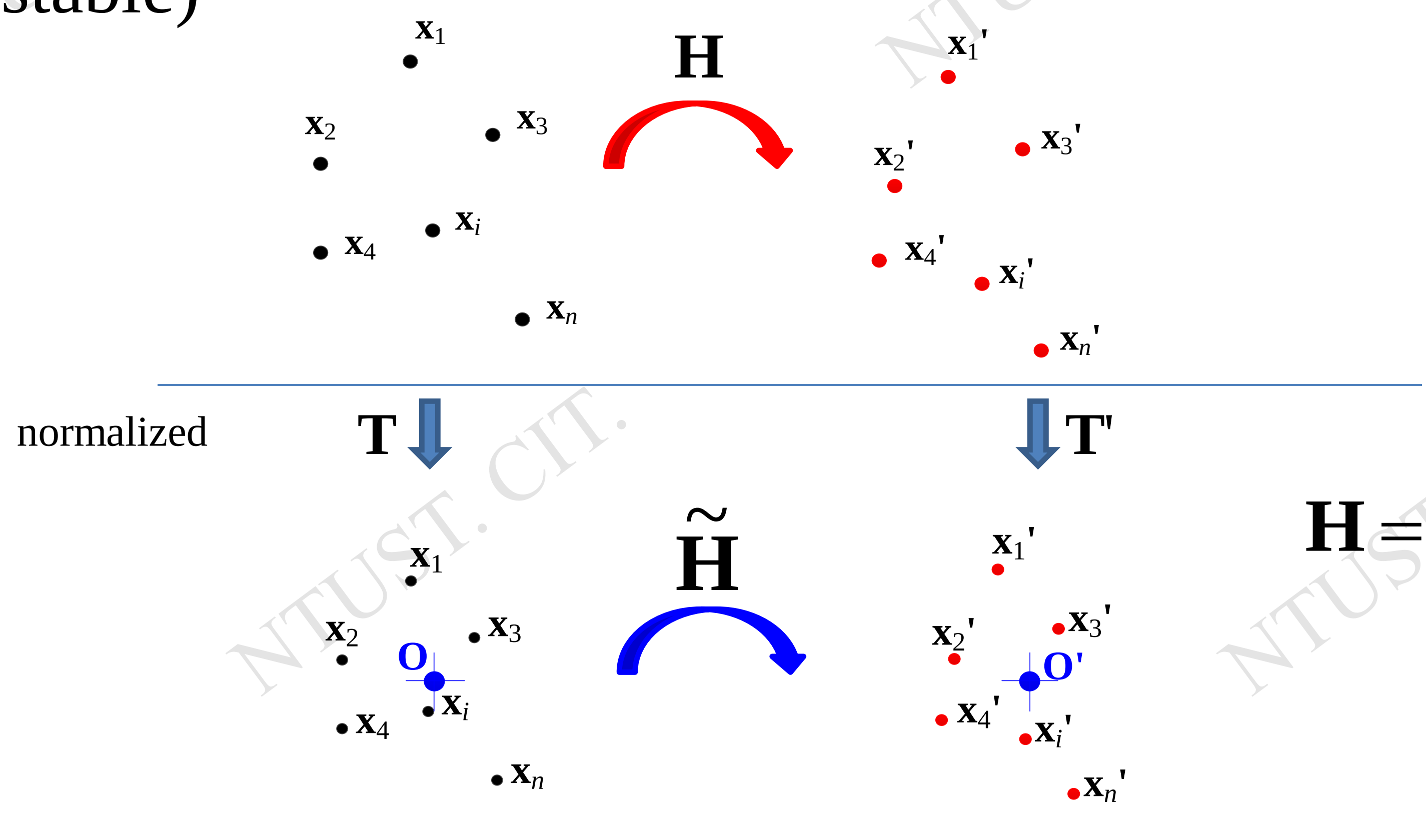

$$d\left(\mathbf{x}, \mathbf{H}^{-1}\mathbf{x}'\right)^2 + d\left(\mathbf{x}', \mathbf{H}\mathbf{x}\right)^2$$




$$d\left(\mathbf{x}, \hat{\mathbf{x}}\right)^2 + d\left(\mathbf{x}', \hat{\mathbf{x}}'\right)^2$$

# Normalizing transformations

- To have a better solution for DLT algorithm (much stable)




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

# Normalizing transformations—cont.

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

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

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

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

$(s_x, s_y) \longrightarrow$ scale (could be $s_x$=$s_y$), and be suggested to be $\sqrt{2}/l$

$l$ is the average distance to centroid.

# Normalized DLT algorithm



Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$, determine the 2D homography matrix $\mathbf{H}$ such that $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$

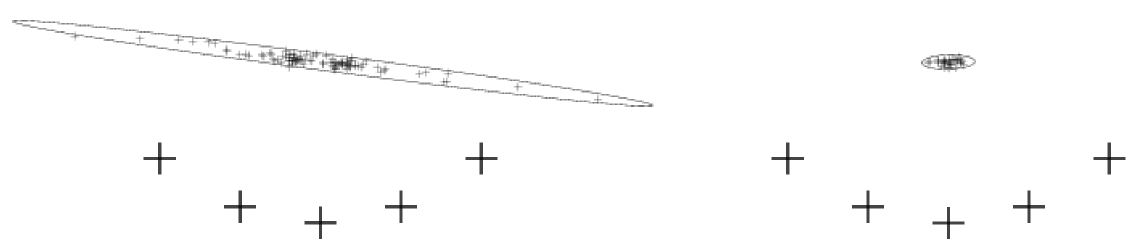
Algorithm

(i) Normalize points, here we have and

(ii) Apply DLT algorithm to determine

(iii) Denormalize solution, then get

Algorithm 4.2 [Hartley04]



# RANSAC (RANdom SAmple Consensus)

- A robust estimation

<u>Objective</u>

Robust fit of model to data set S which contains outliers

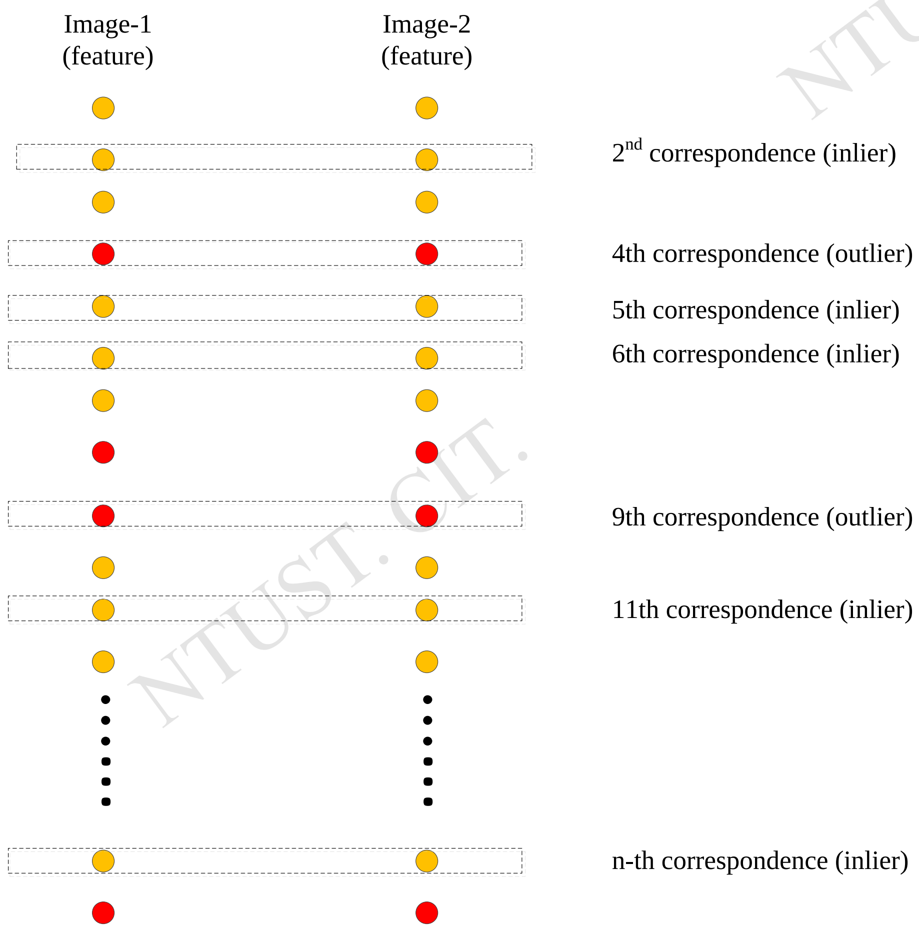
<u>Algorithm</u>

1) Randomly select a sample of *s* data points from S and instantiate the model from this subset.
2) Determine the set of data points $S_i$ which are within a distance threshold $t$ of the model. The set $S_i$ is the consensus set of samples and defines the inliers of $S$.
3) If the subset of $S_i$ (the number of inliers) is greater than some threshold $T$, re-estimate the model using all the points in $S_i$ and terminate
4) If the size of $S_i$ is less than $T$, select a new subset and repeat the above.
5) After $N$ trials the largest consensus set $S_i$ is selected, and the model is re-estimated using all the points in the subset $S_i$

Algorithm 4.4 [Hartley04]

Fischler, M.A. and Bolles, R.C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*. 24, 6 (1981).

# RANSAC—cont. (RANdom SAmple Consensus)




For example, randomly select 7 correspondences, then calculate homography (H).

# RANSAC—cont. (RANdom SAmple Consensus)

- Summary:
  1. Two thresholds are assigned
     - Distance of error (誤差距離)
     - Number of inliers (正確的對應點數量)
  2. Randomly select correspondences
  3. Unique solution ?

# Automatic estimation of a homography (RANSAC)

Objective

Compute the 2D homography between two images.

Algorithm

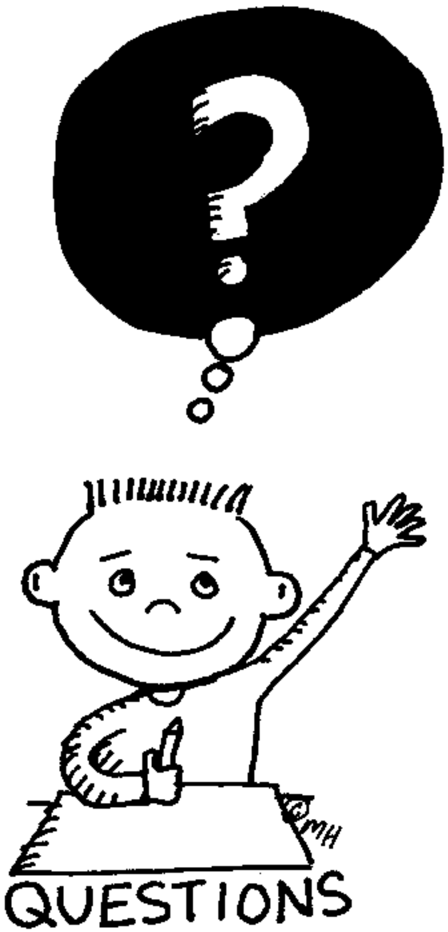
(i) **Interest points:** Compute interest points in each image.
(ii) **Putative correspondences:** Compute a set of interest point matches based on proximity and similarity of their intensity neighbourhood.
(iii) **RANSAC robust estimation:** Repeat for $N$ samples, where $N$ is determined adaptively as in algorithm 4.5:
  (a) Select a random sample of 4 correspondences and compute the homography $\mathtt{H}$.
  (b) Calculate the distance $d_\perp$ for each putative correspondence.
  (c) Compute the number of inliers consistent with $\mathtt{H}$ by the number of correspondences for which $d_\perp < t = \sqrt{5.99}\,\sigma$ pixels.

  Choose the $\mathtt{H}$ with the largest number of inliers. In the case of ties choose the solution that has the lowest standard deviation of inliers.
(iv) **Optimal estimation:** re-estimate $\mathtt{H}$ from all correspondences classified as inliers, by minimizing the ML cost function (4.8–$p$95) using the Levenberg–Marquardt algorithm of section A6.2($p$600).
(v) **Guided matching:** Further interest point correspondences are now determined using the estimated $\mathtt{H}$ to define a search region about the transferred point position.

The last two steps can be iterated until the number of correspondences is stable.

Algorithm 4.6. *Automatic estimation of a homography between two images using RANSAC.*

Algorithm 4.6 [Hartley04]




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

Find us on: facebook



Figure from : http://www.gabankruptcylawyersnetwork.com/files/2014/08/question.gif