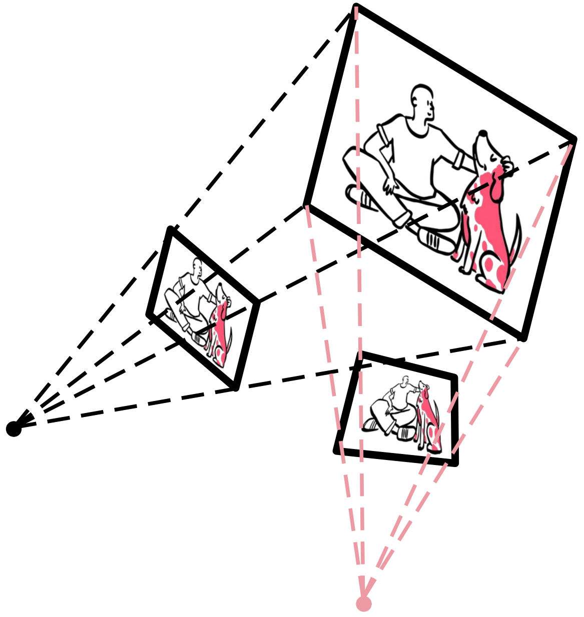
Computer Vision

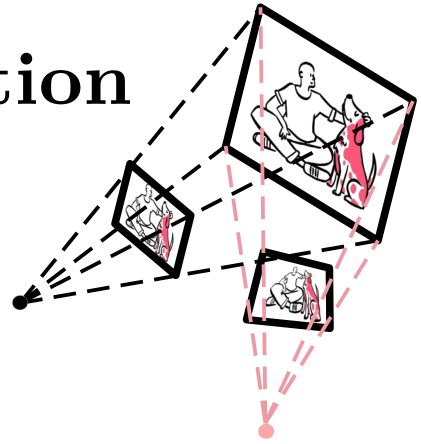


Class 05

Raquel Frizera Vassallo

2D Homography

Parameter Estimation



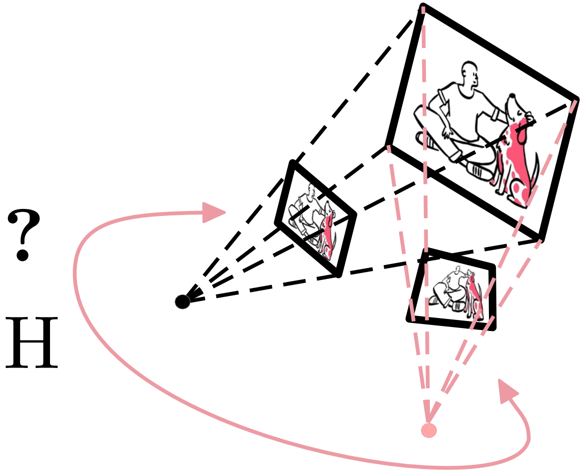
Summary

- Homography
- DLT
- Normalized DLT
- Gold Standard Algorithm
- RANSAC
- Automatic computation of H using RANSAC

DLT

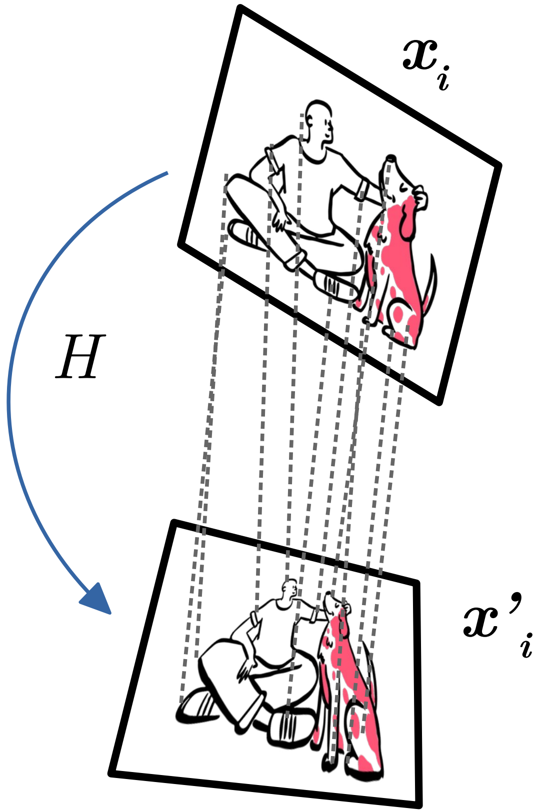
Direct Linear

Transformation



Parameter Estimation

- 2D homography
- Given a set of ($\boldsymbol{x}_i$,$\boldsymbol{x}_i$'), compute H ($\boldsymbol{x}_i' = H\boldsymbol{x}_i$)
- Estimation will be based on the Direct Linear Transformation Algorithm (DLT)



Number of measurements required

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

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

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

Number of measurements required

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

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

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

- 2 independent equations per point
- 8 degrees of freedom

Number of measurements required

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

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

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

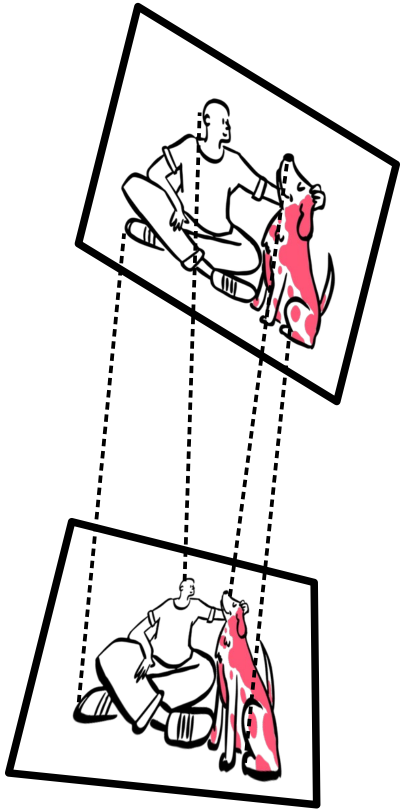
- 2 independent equations per point
- 8 degrees of freedom

We need 4 or more matchings

$N \times 2 \geq 8 \rightarrow N \geq 4$

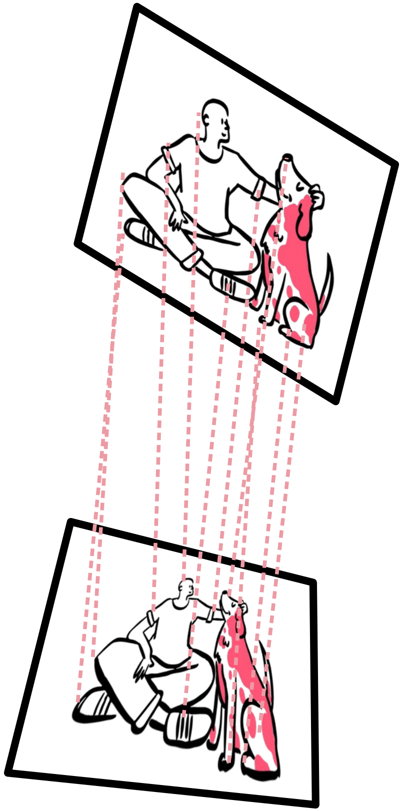
Approximate solutions

- Minimal solution:
  - 4 points yield an exact solution for H.



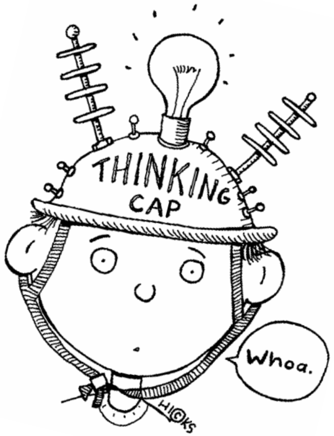
Approximate solutions

- 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.



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)

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

$$\boldsymbol{x}'_i = (x'_i, y'_i, w'_i)^T$$

$$H\boldsymbol{x}_i = \begin{bmatrix} {h^1}^T \boldsymbol{x}_i \\ {h^2}^T \boldsymbol{x}_i \\ {h^3}^T \boldsymbol{x}_i \end{bmatrix}$$

Direct Linear Transformation (DLT)

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

$$\boldsymbol{x'}_i = (x'_i, y'_i, w'_i)^T$$

$$H\boldsymbol{x}_i = \begin{bmatrix} h^{1^T}\boldsymbol{x}_i \\ h^{2^T}\boldsymbol{x}_i \\ h^{3^T}\boldsymbol{x}_i \end{bmatrix}$$

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

Direct Linear Transformation (DLT)

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

$$\boldsymbol{x'}_i = (x'_i, y'_i, w'_i)^T$$

$$H\boldsymbol{x}_i = \begin{bmatrix} h^{1T}\boldsymbol{x}_i \\ h^{2T}\boldsymbol{x}_i \\ h^{3T}\boldsymbol{x}_i \end{bmatrix}$$

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

$$\boldsymbol{x'}_i \times H\boldsymbol{x}_i = \begin{bmatrix} y'_i h^{3T}\boldsymbol{x}_i - w'_i h^{2T}\boldsymbol{x}_i \\ w'_i h^{1T}\boldsymbol{x}_i - x'_i h^{3T}\boldsymbol{x}_i \\ x'_i h^{2T}\boldsymbol{x}_i - y'_i h^{1T}\boldsymbol{x}_i \end{bmatrix}$$

Direct Linear Transformation (DLT)

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

$$\boldsymbol{x}'_i = (x'_i, y'_i, w'_i)^T$$

$$H\boldsymbol{x}_i = \begin{bmatrix} {h^1}^T \boldsymbol{x}_i \\ {h^2}^T \boldsymbol{x}_i \\ {h^3}^T \boldsymbol{x}_i \end{bmatrix}$$

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

$$\boldsymbol{x}'_i \times H\boldsymbol{x}_i = \begin{bmatrix} y'_i {h^3}^T \boldsymbol{x}_i - w'_i {h^2}^T \boldsymbol{x}_i \\ w'_i {h^1}^T \boldsymbol{x}_i - x'_i {h^3}^T \boldsymbol{x}_i \\ x'_i {h^2}^T \boldsymbol{x}_i - y'_i {h^1}^T \boldsymbol{x}_i \end{bmatrix}$$

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

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

Direct Linear Transformation (DLT)

- Equations are linear in $h \rightarrow A_i h = 0$
- Only 2 out of 3 are linearly independent (indeed, 2 equations per point)

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

Direct Linear Transformation (DLT)

- Equations are linear in $h \rightarrow A_i h = 0$
- Only 2 out of 3 are linearly independent (indeed, 2 equations per point)

$$\begin{bmatrix} 0^T & -w_i' \boldsymbol{x}_i{}^T & y_i' \boldsymbol{x}_i{}^T \\ w_i' \boldsymbol{x}_i{}^T & 0^T & -x_i' \boldsymbol{x}_i{}^T \\ \cancel{-y_i' \boldsymbol{x}_i{}^T} & \cancel{x_i' \boldsymbol{x}_i{}^T} & \cancel{0^T} \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ \cancel{h^3} \end{bmatrix} = 0 \qquad \begin{bmatrix} 0^T & -w_i' \boldsymbol{x}_i{}^T & y_i' \boldsymbol{x}_i{}^T \\ w_i' \boldsymbol{x}_i{}^T & 0^T & -x_i' \boldsymbol{x}_i{}^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

(only drop third row if $w_i{}' \neq 0$)

Direct Linear Transformation (DLT)

- Equations are linear in $h \rightarrow A_i h = 0$
- Only 2 out of 3 are linearly independent (indeed, 2 equations per point)

$$\begin{bmatrix} 0^T & -w_i' \boldsymbol{x}_i{}^T & y_i' \boldsymbol{x}_i{}^T \\ w_i' \boldsymbol{x}_i{}^T & 0^T & -x_i' \boldsymbol{x}_i{}^T \\ \cancel{-y_i' \boldsymbol{x}_i{}^T} & \cancel{x_i' \boldsymbol{x}_i{}^T} & \cancel{0^T} \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ \cancel{h^3} \end{bmatrix} = 0 \qquad \begin{bmatrix} 0^T & -w_i' \boldsymbol{x}_i{}^T & y_i' \boldsymbol{x}_i{}^T \\ w_i' \boldsymbol{x}_i{}^T & 0^T & -x_i' \boldsymbol{x}_i{}^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

(only drop third row if $w_i' \neq 0$)

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

Direct Linear Transformation (DLT)

- Solving for H

$$Ah = 0 \qquad \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} h = 0$$

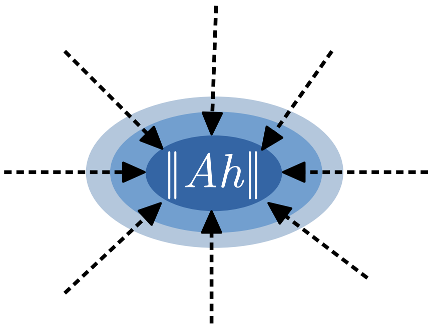
size A is 8x9 or 12x9, but rank 8

- Trivial solution is $h{=}0_9{}^T$ is not interesting
- 1-D null-space yields solution of interest pick for example the one with: $\|h\|{=}1$

Direct Linear Transformation (DLT)

- Over-determined solution
- No exact solution because of inexact measurement, i.e. “noise”.
- Find approximate solution:
  - Additional constraint needed to avoid 0, e.g. $\|h\|=1$
  - Since $Ah=0$ is not possible, so minimize $\|Ah\|$

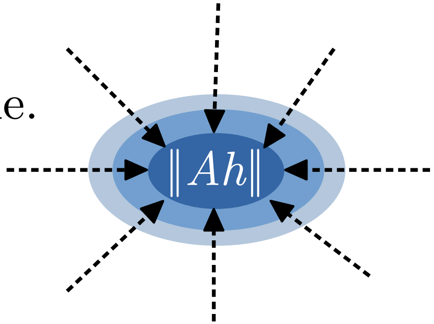
$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} h = 0$$



Direct Linear Transformation (DLT)

- Over-determined solution
- No exact solution because of inexact measurement, i.e. “noise”.
- Find approximate solution:
  - Additional constraint needed to avoid 0, e.g. $\|h\|=1$
  - Since $Ah=0$ is not possible, so minimize $\|Ah\|$
- The solution is the eigenvector of $A^T A$ with the least eigenvalue.
  - That is equivalent to the singular vector corresponding to the smallest singular value of A.

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} h = 0$$



DLT algorithm

<u>Objective</u>

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

DLT algorithm

Objective

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

Algorithm

(i) For each correspondence $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$ compute A_i. Usually only two first rows needed.

DLT algorithm

Objective

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

Algorithm

(i) For each correspondence $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2x9 matrices A_i into a single $2n$x9 matrix A

DLT algorithm

Objective

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

Algorithm

(i) For each correspondence $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2x9 matrices A_i into a single $2n$x9 matrix A

(iii) Obtain SVD of A. Solution for h is last column of V

DLT algorithm

Objective

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

Algorithm

(i) For each correspondence $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2x9 matrices A_i into a single $2n$x9 matrix A

(iii) Obtain SVD of A. Solution for h is last column of V

(iv) Determine H from h

Some Details in the DLT

Inhomogeneous solution

- Since h can only be computed up to scale, pick h_j=1, e.g. h_9=1, and solve for 8-vector

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

- Solve using Gaussian elimination (4 points) or using linear least-squares (more than 4 points).

Inhomogeneous solution

- Since h can only be computed up to scale, pick h_j=1, e.g. h_9=1, and solve for 8-vector

$$\begin{bmatrix} 0 & 0 & 0 & -x_i w'_i & -y_i w'_i & -w_i w'_i & x_i y'_i & y_i y'_i \\ x_i w'_i & y_i w'_i & w_i w'_i & 0 & 0 & 0 & x_i x'_i & y_i x'_i \end{bmatrix} \tilde{h} = \begin{bmatrix} -w_i y'_i \\ w_i x'_i \end{bmatrix}$$

- Solve using Gaussian elimination (4 points) or using linear least-squares (more than 4 points).
- However, if h_9=0 this approach fails. Also gets poor results if h_9 close to zero.
  Therefore, not recommended.
- Note h_9=H_{33}=0 if origin is mapped to infinity.

Inhomogeneous solution

- Since h can only be computed up to scale, pick h_j=1, e.g. h_9=1, and solve for 8-vector

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

- Solve using Gaussian elimination (4 points) or using linear least-squares (more than 4 points).
- However, if h_9=0 this approach fails. Also gets poor results if h_9 close to zero.

Therefore, not recommended.

- Note h_9=H_{33}=0 if origin is mapped to infinity.

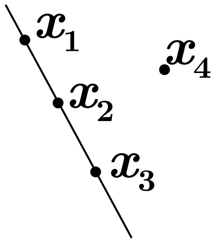
Example:
When the horizon passes by the image center (0,0),
that means the image center lies on the infinity!

$$\boldsymbol{l}_\infty^T H \boldsymbol{x}_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} H \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

Degenerate Configurations

Three collinear points out of 4.

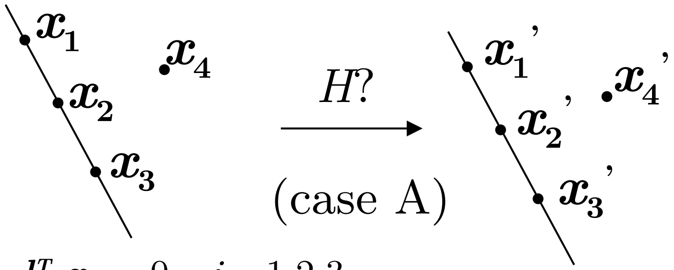
PROBLEM!!!



$l^T x_i = 0 \quad i = 1,2,3$

Degenerate Configurations

Three collinear points out of 4.
PROBLEM!!!



$\boldsymbol{l}^T \boldsymbol{x}_i = 0 \quad i = 1,2,3$

Constraints: $\boldsymbol{x}_i' \times H\boldsymbol{x}_i = 0 \qquad i = 1,2,3,4$

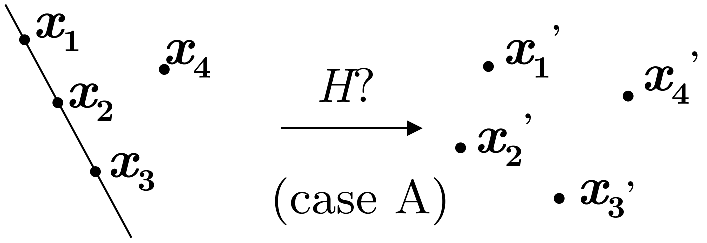
Define: $H^* = \boldsymbol{x}_4' \, \boldsymbol{l}^T$

Then, $H^* \boldsymbol{x}_i = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_i) = 0, \; i = 1,2,3$

$H^* \boldsymbol{x}_4 = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_4) = \mathrm{k}\boldsymbol{x}_4'$

Degenerate Configurations

Three collinear points out of 4.
PROBLEM!!!



$\boldsymbol{l}^T \boldsymbol{x}_i = 0 \quad i = 1,2,3$

Case A: If H^* is unique solution, then no homography mapping $\boldsymbol{x}_i \rightarrow \boldsymbol{x}_i'$ exist $\rightarrow$ every homography must preserve collinearity.

Constraints: $\boldsymbol{x}_i' \times H\boldsymbol{x}_i = 0 \qquad i = 1,2,3,4$

Define: $H^* = \boldsymbol{x}_4' \, \boldsymbol{l}^T$

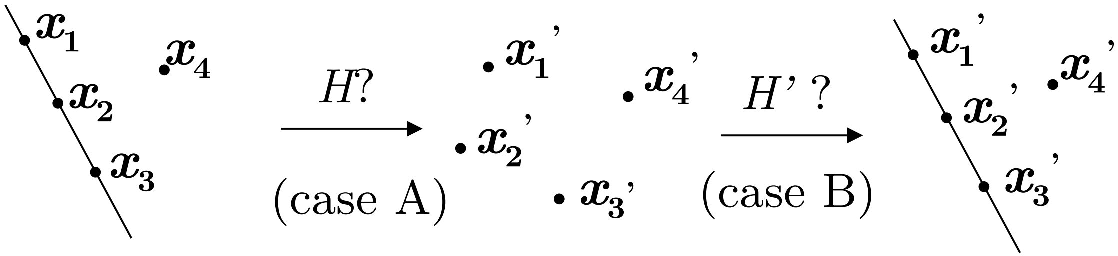
Then, $H^* \boldsymbol{x}_i = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_i) = 0, \; i = 1,2,3$

$H^* \boldsymbol{x}_4 = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_4) = \mathrm{k}\boldsymbol{x}_4'$

H^* is rank-1 matrix and thus not a homography (rank-3)!

Degenerate Configurations

Three collinear points out of 4.
PROBLEM!!!



$\boldsymbol{l}^T \boldsymbol{x}_i = 0 \quad i = 1,2,3$

Constraints: $\boldsymbol{x}_i' \times H\boldsymbol{x}_i = 0 \qquad i = 1,2,3,4$

Define: $H^* = \boldsymbol{x}_4' \, \boldsymbol{l}^T$

Then, $H^* \boldsymbol{x}_i = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_i) = 0,\ i = 1,2,3$

$H^* \boldsymbol{x}_4 = \boldsymbol{x}_4' \, (\boldsymbol{l}^T \boldsymbol{x}_4) = \mathrm{k}\boldsymbol{x}_4'$

Case A: If H^* is unique solution, then no homography mapping $\boldsymbol{x}_i \rightarrow \boldsymbol{x}_i'$ exist $\rightarrow$ every homography must preserve collinearity.

Case B: If there is a solution H, then also $\alpha H^* + \beta H \rightarrow$ **many solutions**

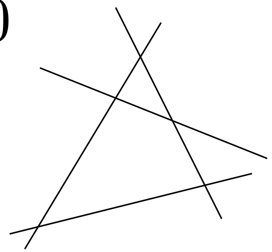
H^* is rank-1 matrix and thus not a homography (rank-3)!

Solutions from lines and points

2D homographies from 2D lines

$$\boldsymbol{l'}_i = H^T \boldsymbol{l}_i \qquad Ah = 0$$

Minimum of 4 lines

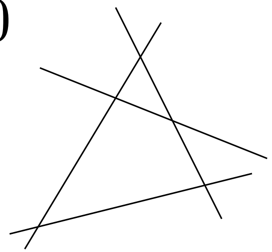


Solutions from lines and points

2D homographies from 2D lines

$$\boldsymbol{l'}_i = H^T \boldsymbol{l}_i \qquad Ah = 0$$

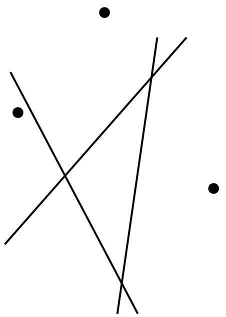
Minimum of 4 lines



But can also be determined by:

- 3 general points and 1 line or
- 3 general lines and 1 point.

Equivalent to 4 general points

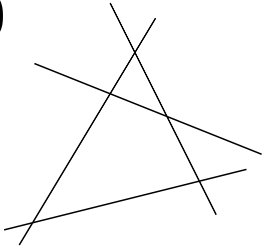


Solutions from lines and points

2D homographies from 2D lines

$$\boldsymbol{l}'_i = H^T \boldsymbol{l}_i \qquad Ah = 0$$

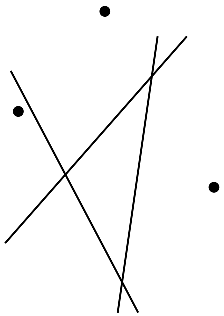
Minimum of 4 lines



But can also be determined by:

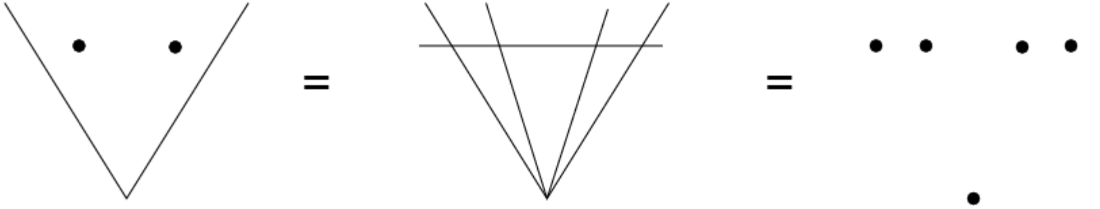
- 3 general points and 1 line or
- 3 general lines and 1 point.

Equivalent to 4 general points



Mixed configurations that do not work. For example:

- two points and two lines = four concurrent lines = four collinear points.

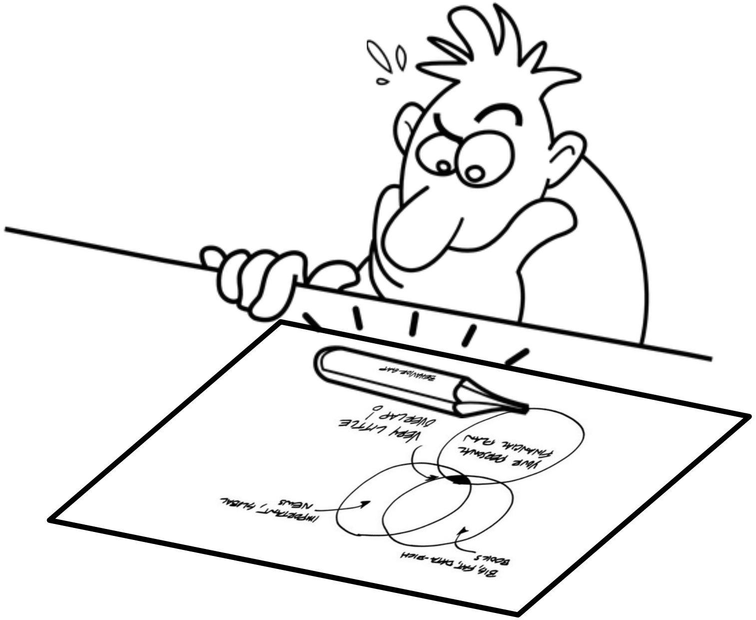


Improving DLT

Normalized DLT

Cost functions

- Algebraic distance
- Geometric distance
- Reprojection error



Algebraic distance

DLT minimizes $\|Ah\| \longrightarrow e = Ah \longrightarrow$ residual vector

$e_i \longrightarrow$ partial error vector for each ($\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$)

algebraic error vector

Algebraic distance

DLT minimizes $\|Ah\| \longrightarrow e = Ah \longrightarrow$ residual vector

$e_i \longrightarrow$ partial error vector for each $(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i')$

algebraic error vector

$$d_{alg}(\boldsymbol{x}'_i, H\boldsymbol{x}_i)^2 = \|e_i\|^2 = \left\| \begin{bmatrix} 0^T & -w'_i\boldsymbol{x}_i^T & -y'_i\boldsymbol{x}_i^T \\ -w'_i\boldsymbol{x}_i^T & 0^T & -x'_i\boldsymbol{x}_i^T \end{bmatrix} h \right\|^2$$ *algebraic distance*

$$d_{alg}(\boldsymbol{x}_1, \boldsymbol{x}_2)^2 = a_1{}^2 + a_2{}^2 \text{ where } \boldsymbol{a} = (a_1, a_2, a_3)^T = \boldsymbol{x}_1 \times \boldsymbol{x}_2$$

Algebraic distance

DLT minimizes $\|Ah\| \longrightarrow e = Ah \longrightarrow$ residual vector

$e_i \longrightarrow$ partial error vector for each $(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i')$

algebraic error vector

$$d_{alg}(\boldsymbol{x}'_i, H\boldsymbol{x}_i)^2 = \|e_i\|^2 = \left\| \begin{bmatrix} 0^T & -w'_i \boldsymbol{x}_i^T & -y'_i \boldsymbol{x}_i^T \\ -w'_i \boldsymbol{x}_i^T & 0^T & -x'_i \boldsymbol{x}_i^T \end{bmatrix} h \right\|^2 \quad \textit{algebraic distance}$$

$$d_{alg}(\boldsymbol{x}_1, \boldsymbol{x}_2)^2 = a_1{}^2 + a_2{}^2 \text{ where } \boldsymbol{a} = (a_1, a_2, a_3)^T = \boldsymbol{x}_1 \times \boldsymbol{x}_2$$

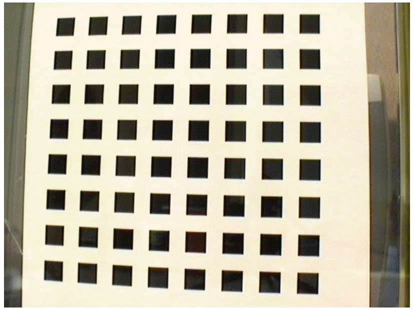
$$\boxed{\sum_i d_{alg}(\boldsymbol{x}'_i, H\boldsymbol{x}_i)^2 = \sum_i \|e_i\|^2 = \|Ah\|^2 = \|e\|^2}$$

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

Geometric distance

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

Ex: Homography between calibration pattern and its image.



2 options:

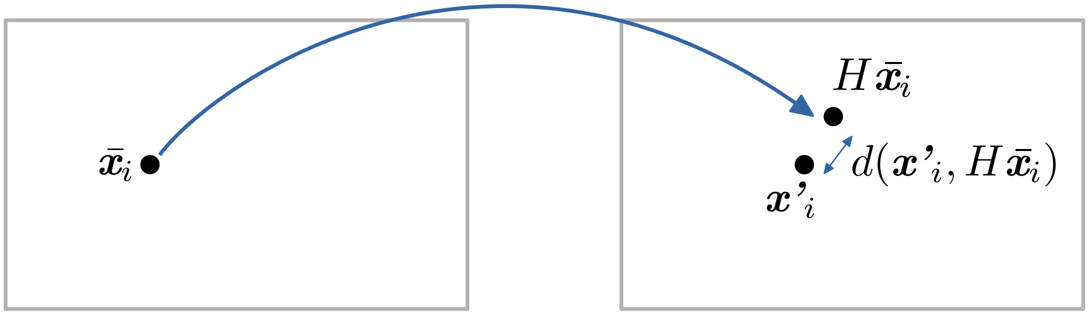
Consider 1 image

Consider 2 images

Geometric distance

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

Error in just one image



$$\hat{H} = \underset{H}{argmin} \sum_i d(\boldsymbol{x}'_i, H\bar{\boldsymbol{x}}_i)^2$$

Point in the second image

True point mapped with H from the first image

Geometric distance

x measured coordinates

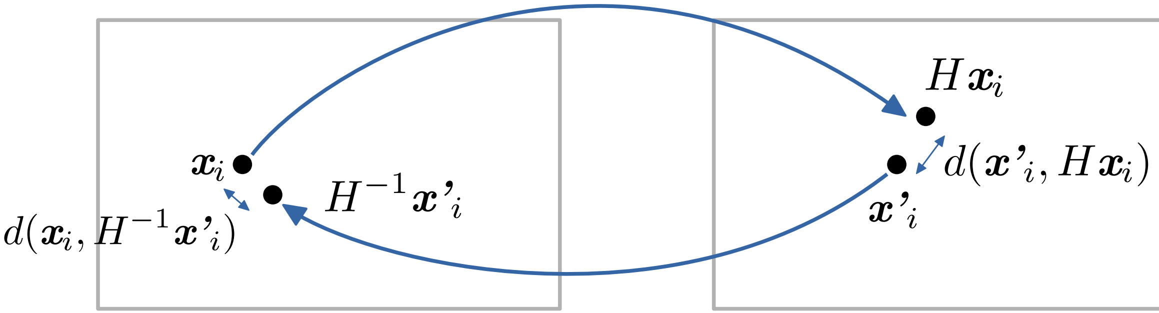
$\hat{x}$ estimated coordinates

$\bar{x}$ true coordinates

$d(.,.)$ Euclidean distance (in image)

Symmetric transfer error

Error in both images



$$\hat{H} = \underset{H}{argmin} \sum_i d(\boldsymbol{x}_i, H^{-1}\boldsymbol{x}'_i)^2 + d(\boldsymbol{x}'_i, H\boldsymbol{x}_i)^2$$

Transfer error in the first image

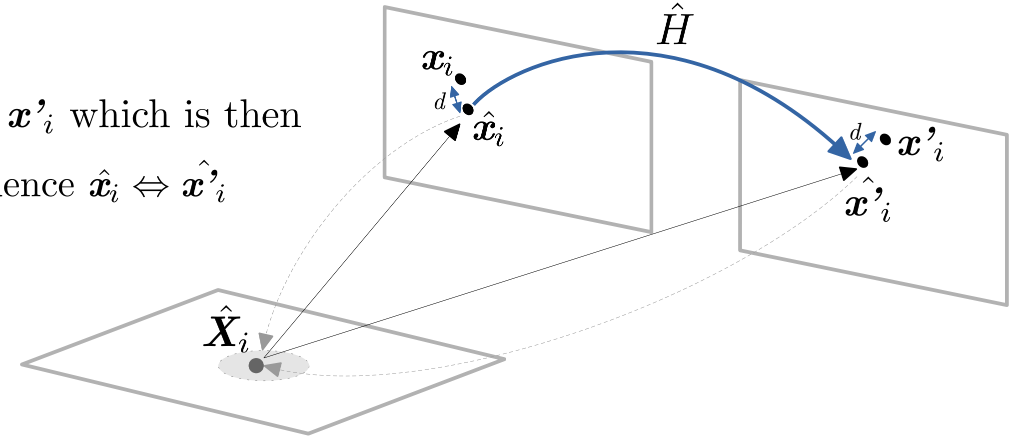
Transfer error in the second image

Reprojection error

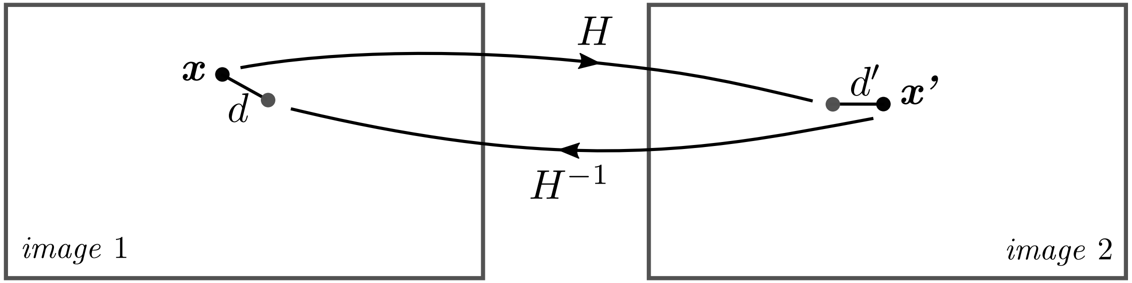
Minimizing the cost function involves determining the homography and a set of correspondences $\rightarrow \hat{H}, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}'_i$

$$(\hat{H}, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}'_i) = \underset{\hat{H}, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}'_i}{argmin} \sum_i d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)^2 + d(\boldsymbol{x}'_i, \hat{\boldsymbol{x}}'_i)^2 \text{ subject to } \hat{\boldsymbol{x}}'_i = \hat{H}\hat{\boldsymbol{x}}_i$$

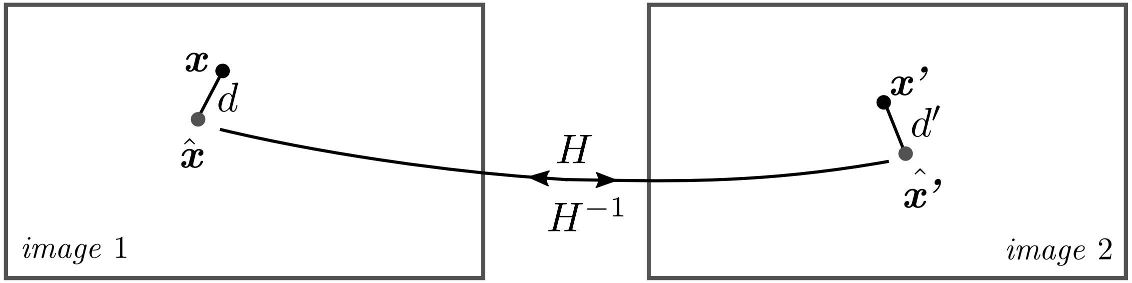
e.g. Estimate a point on the world plane $\hat{\boldsymbol{X}}_i$ from $\boldsymbol{x}_i \Leftrightarrow \boldsymbol{x}'_i$ which is then reprojected to the estimated perfectly matched correspondence $\hat{\boldsymbol{x}}_i \Leftrightarrow \hat{\boldsymbol{x}}'_i$



Symmetric transfer error X Reprojection error



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



$$d(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + d(\boldsymbol{x'}, \hat{\boldsymbol{x}}')^2$$

Statistical cost function and Maximum Likelihood Estimation

- Optimal estimate of $H \rightarrow$ need a noise model.
- Assume zero-mean isotropic Gaussian noise (assume outliers removed).
- Defining the pdf of the noise and the MLE of the homography results on minimizing the geometric error.
- Error in one image.

Maximum Likelihood Estimate

$$\boxed{\sum_i d(\boldsymbol{x}'_i, H\bar{\boldsymbol{x}}_i)^2}$$

Statistical cost function and Maximum Likelihood Estimation

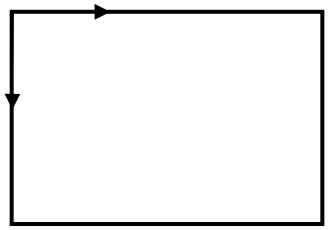
- Optimal estimate of H $\rightarrow$ need a noise model.
- Assume zero-mean isotropic Gaussian noise (assume outliers removed).
- Defining the pdf of the noise and the MLE of the homography results on minimizing the reprojection error.
- Error in both images.

Maximum Likelihood Estimate

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

Invariance to transformations?

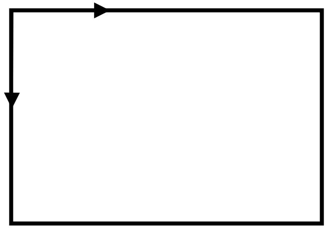
- Are the properties and performance of the DLT algorithm invariant to transformations?
- Will result change?
- For which algorithms? For which transformations?



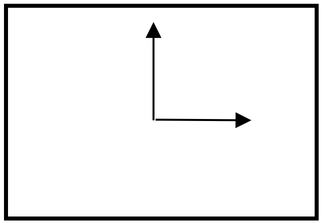
$$\boldsymbol{x}' = H\boldsymbol{x}$$

Invariance to transformations?

- Are the properties and performance of the DLT algorithm invariant to transformations?
- Will result change?
- For which algorithms? For which transformations?



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

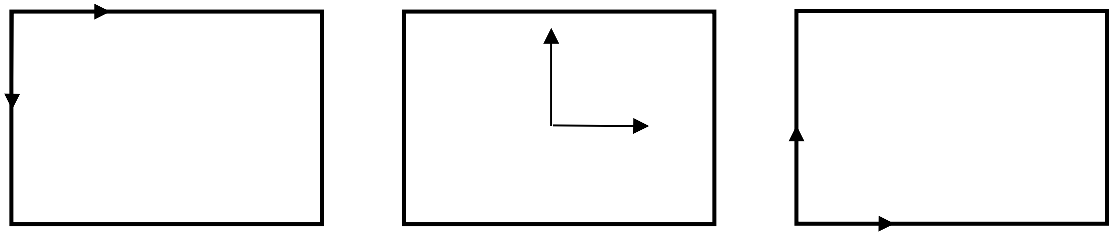


$$\tilde{\boldsymbol{x}} = T\boldsymbol{x}$$

$$\tilde{\boldsymbol{x}}' = T'\boldsymbol{x}'$$

Invariance to transformations?

- Are the properties and performance of the DLT algorithm invariant to transformations?
- Will result change?
- For which algorithms? For which transformations?



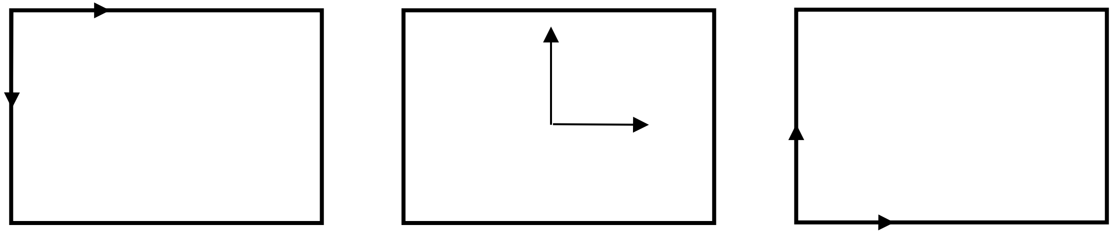
$$\boldsymbol{x}' = H\boldsymbol{x}$$

$$\tilde{\boldsymbol{x}} = T\boldsymbol{x}$$
$$\tilde{\boldsymbol{x}}' = T'\boldsymbol{x}'$$

$$\tilde{\boldsymbol{x}}' = \tilde{H}\tilde{\boldsymbol{x}}$$
$$T'\boldsymbol{x}' = \tilde{H}T\boldsymbol{x}$$
$$\boldsymbol{x}' = T'^{-1}\tilde{H}T\boldsymbol{x}$$

Invariance to transformations?

- Are the properties and performance of the DLT algorithm invariant to transformations?
- Will result change?
- For which algorithms? For which transformations?



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

$$\tilde{\mathbf{x}} = T\mathbf{x}$$
$$\tilde{\mathbf{x}}' = T'\mathbf{x}'$$

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

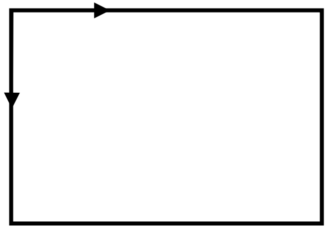
$$\boxed{H \stackrel{?}{=} T'^{-1}\tilde{H}T}$$

Invariance to transformations?

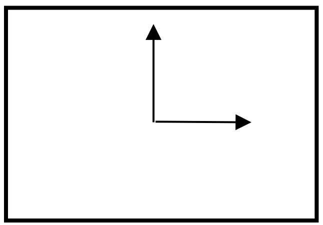
- Are the properties and performance of the DLT algorithm invariant to transformations?
- Will result change?
- For which algorithms? For which transformations?

NO

It is not invariant

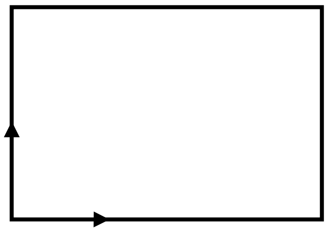


$$\boldsymbol{x}' = H\boldsymbol{x}$$



$$\tilde{\boldsymbol{x}} = T\boldsymbol{x}$$

$$\tilde{\boldsymbol{x}}' = T'\boldsymbol{x}'$$



$$\tilde{\boldsymbol{x}}' = \tilde{H}\tilde{\boldsymbol{x}}$$

$$T'\boldsymbol{x}' = \tilde{H}T\boldsymbol{x}$$

$$\boldsymbol{x}' = T'^{-1}\tilde{H}T\boldsymbol{x}$$

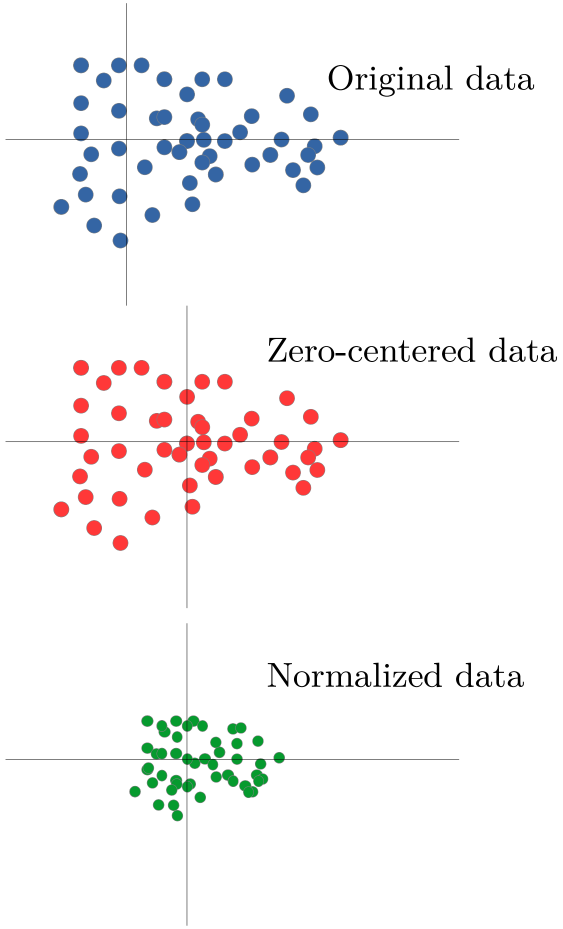
$$\boxed{H \overset{?}{=} T'^{-1}\tilde{H}T}$$

Non-invariance of DLT

- DLT is not invariant to similarity transformations on the image.
- Results depend on the coordinate frame in which points are expressed.
- Some coordinate systems are in some way better than others for computing a 2D homography.
- Data normalization is an essential step in the DLT algorithm.
  It **MUST NOT** be considered optional.

Normalizing transformations

- Algorithms with initial normalization step will be invariant to arbitrary choices of scale and coordinate origin
- What is a good choice of coordinates?
  - Translate centroid to origin.
  - Scale to a $\sqrt{2}$ average distance to the origin. This means that the average point is equal to $(1,1,1)^T$.
  - This transformation is applied independently on both images.



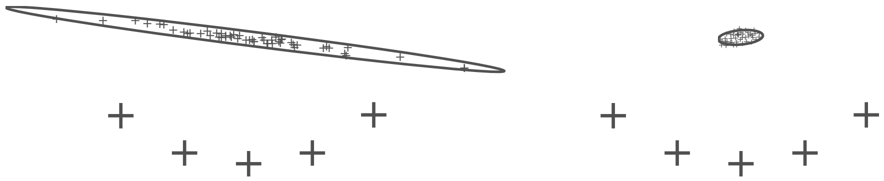
Importance of normalization

$$\begin{bmatrix} 0 & 0 & 0 & -\boldsymbol{x'}_i & -\boldsymbol{y'}_i & -1 & \boldsymbol{y'}_i\boldsymbol{x}_i & \boldsymbol{y'}_i\boldsymbol{y}_i & \boldsymbol{y'}_i \\ \boldsymbol{x}_i & \boldsymbol{y}_i & 1 & 0 & 0 & 0 & -\boldsymbol{x'}_i\boldsymbol{x}_i & -\boldsymbol{x'}_i\boldsymbol{y}_i & -\boldsymbol{x'}_i \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

$\sim 10^2$ $\sim 10^2$ 1 $\sim 10^2$ $\sim 10^2$ 1 $\sim 10^4$ $\sim 10^4$ $\sim 10^2$

orders of magnitude difference!

The effect of normalization (simulation): 5 point (crosses) were used to compute a 2D Homography. The homography H is the identity mapping. 100 trials were made adding 0.1 pixel Gaussian noise to the points. The result H was applied to a further point. (a) result without normalization and (b) with normalization.



Normalized DLT algorithm

Objective

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

Normalized DLT algorithm

Objective

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

Algorithm

- Normalize points $\tilde{\boldsymbol{x}}_i = T\,\boldsymbol{x}_i$, $\tilde{\boldsymbol{x}}_i' = T'\boldsymbol{x}_i'$

Normalized DLT algorithm

Objective

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

Algorithm

- Normalize points $\tilde{\boldsymbol{x}}_i = T\boldsymbol{x}_i$, $\tilde{\boldsymbol{x}}_i' = T'\boldsymbol{x}_i'$
- Apply DLT algorithm to $\tilde{\boldsymbol{x}}_i \leftrightarrow \tilde{\boldsymbol{x}}_i'$

Normalized DLT algorithm

<u>Objective</u>

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

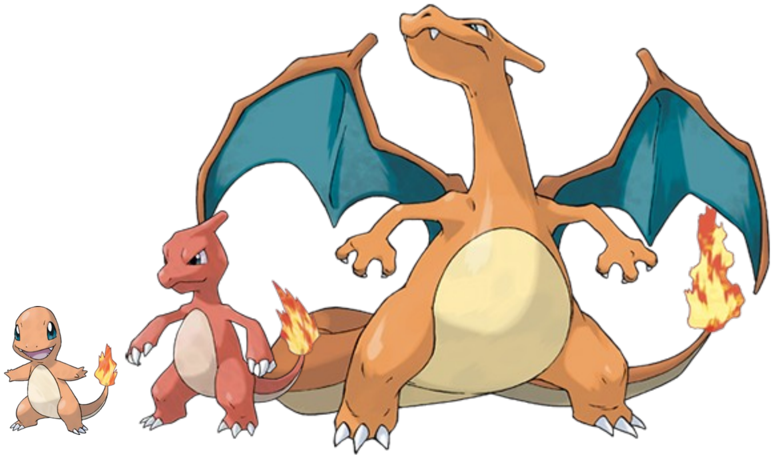
<u>Algorithm</u>

- Normalize points $\tilde{\boldsymbol{x}}_i = T\boldsymbol{x}_i$, $\tilde{\boldsymbol{x}}_i' = T'\boldsymbol{x}_i'$
- Apply DLT algorithm to $\tilde{\boldsymbol{x}}_i \leftrightarrow \tilde{\boldsymbol{x}}_i'$
- Denormalize solution $H = T'^{-1}\tilde{H}T$

Normalized DLT

↓

Gold Standard Algorithm



Iterative minimization methods

Required to minimize geometric error:

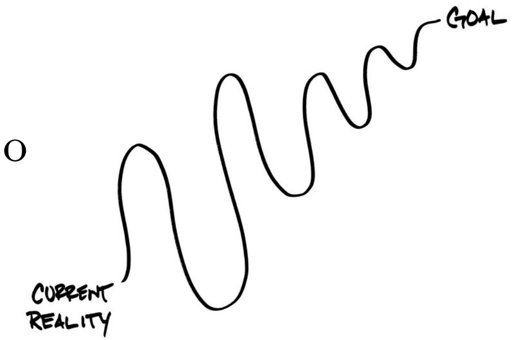
(i) Often slower than DLT
(ii) Require initialization
(iii) No guaranteed convergence, local minima
(iv) Stopping criterion required

Therefore, careful implementation required:

(i) Cost function
(ii) Parameterization (minimal or not)
(iii) Function specification(cost $\leftrightarrow$ parameters)
(iv) Initialization
(v) Iterations

Parametrization

- Parameters should cover complete space and allow efficient estimation of cost.
- Minimal or over-parameterized? e.g. 8 or 9
  - Minimal often more complex, also cost surface.
  - Sometimes stuck in local minimum.
  - Good algorithms can deal with over-parameterization.
- Parametrization can also be used to restrict transformation to particular class.



Function specifications

- Measurement vector $X \in \mathbb{R}^N$ with covariance Σ

(i) Set of parameters represented by vector $P \in \mathbb{R}^{M}$

(ii) Mapping $f : \mathbb{R}^{M} \rightarrow \mathbb{R}^{N}$

(iii) Cost function to be minimized

The goal is to find a set of parameters P such that $f(P) = X$ or failing that, to bring $f(P)$ as close to X as possible

Cost functions

Error in one image

- Points in the first image without error.
- Measurements $\rightarrow$ point in the second image.
- Find H that minimizes the cost function.

$$\sum_i d(\boldsymbol{x'}_i, H\bar{\boldsymbol{x}}_i)^2$$

$$f : h \rightarrow (H\overline{\boldsymbol{x}}_1, H\overline{\boldsymbol{x}}_2, \ldots, H\overline{\boldsymbol{x}}_n)$$

$$\|\boldsymbol{X} - f(h)\|$$

Cost functions

Error in one image

- Points in the first image without error.
- Measurements $\rightarrow$ point in the second image.
- Find H that minimizes the cost function.

$$\sum_i d(\boldsymbol{x}'_i, H\bar{\boldsymbol{x}}_i)^2$$

$$f : h \rightarrow (H\boldsymbol{x}_1, H\boldsymbol{x}_2, \ldots, H\boldsymbol{x}_n)$$

$$\|\boldsymbol{X} - f(h)\|$$

Symmetric transfer error

- Measurements $\rightarrow$ points in both images.
- Find H that minimizes the cost function.

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

$$f : h \rightarrow (H^{-1}\boldsymbol{x}'_1, \ldots, H^{-1}\boldsymbol{x}'_n, H\boldsymbol{x}_1, \ldots, H\boldsymbol{x}_n)$$

$$\|\boldsymbol{X} - f(h)\|$$

Cost functions

Error in one image

- Points in the first image without error.
- Measurements $\rightarrow$ point in the second image.
- Find H that minimizes the cost function.

$$\sum_i d(\boldsymbol{x}'_i, H\bar{\boldsymbol{x}}_i)^2$$

$$f : h \rightarrow (H\boldsymbol{x}_1, H\boldsymbol{x}_2, \ldots, H\boldsymbol{x}_n)$$

$$\|\boldsymbol{X} - f(h)\|$$

Symmetric transfer error

- Measurements $\rightarrow$ points in both images.
- Find H that minimizes the cost function.

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

$$f : h \rightarrow (H^{-1}\boldsymbol{x}'_1, \ldots, H^{-1}\boldsymbol{x}'_n, H\boldsymbol{x}_1, \ldots, H\boldsymbol{x}_n)$$

$$\|\boldsymbol{X} - f(h)\|$$

Reprojection error

- Find H and points that minimizes the cost function.

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

$$f : (h, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n) \rightarrow (\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}'_1, \ldots, \hat{\boldsymbol{x}}_n, \hat{\boldsymbol{x}}'_n)$$

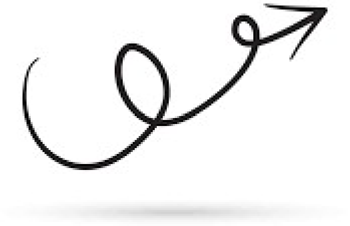
$$\|\boldsymbol{X} - f(h)\|$$

Initialization

- Typically, use linear solution
- If outliers, use robust algorithm

Iteration methods

- Most popular:
  - Newton's method
  - Levenberg-Marquardt



Gold Standard algorithm

<u>Objective</u>

Given $n \geq 4$ 2D to 2D point correspondences $\{\boldsymbol{x}_i \longleftrightarrow \boldsymbol{x}_i'\}$, determine the Maximum Likelyhood Estimation of H (this also implies computing optimal $\hat{\boldsymbol{x}}_i' = \hat{H}\hat{\boldsymbol{x}}_i$)

Gold Standard algorithm

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\boldsymbol{x}_i \longleftrightarrow \boldsymbol{x}_i'\}$, determine the Maximum Likelyhood Estimation of H (this also implies computing optimal $\hat{\boldsymbol{x}}_i' = \hat{H}\hat{\boldsymbol{x}}_i$)

Algorithm

(i) **Initialization:** compute an initial estimate using normalized DLT or RANSAC

Gold Standard algorithm

<u>Objective</u>

Given $n \geq 4$ 2D to 2D point correspondences $\{\boldsymbol{x}_i \longleftrightarrow \boldsymbol{x}_i'\}$, determine the Maximum Likelyhood Estimation of H (this also implies computing optimal $\hat{\boldsymbol{x}}_i' = \hat{H}\hat{\boldsymbol{x}}_i$)

<u>Algorithm</u>

(i) **Initialization:** compute an initial estimate using normalized DLT or RANSAC

(ii) **Geometric minimization of Gold Standard error:**

- compute initial estimate for optimal $\{\hat{\boldsymbol{x}}_i\}$ from $\{\boldsymbol{x}_i\}$
- minimize cost $\Sigma\, \mathrm{d}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)^2 + \mathrm{d}(\boldsymbol{x}_i', \hat{\boldsymbol{x}}_i')^2$ over $\{\hat{H}, \hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \hat{\boldsymbol{x}}_3, \ldots, \hat{\boldsymbol{x}}_n\}$
- if many points, use sparse method

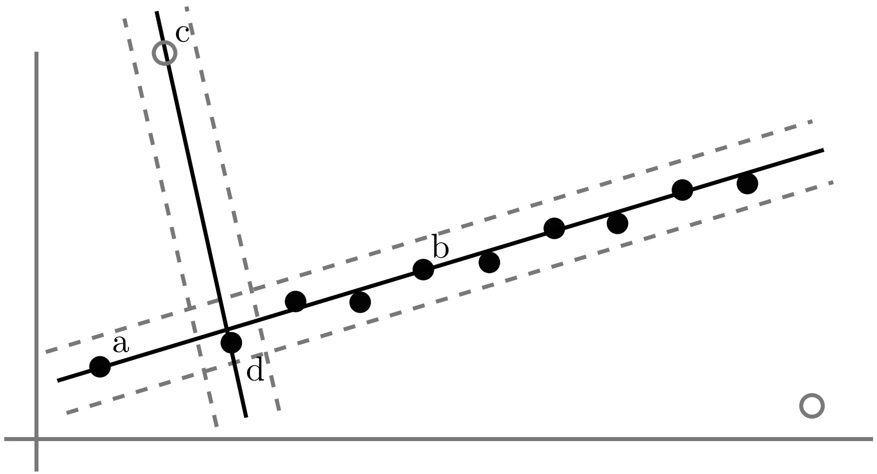
RANSAC

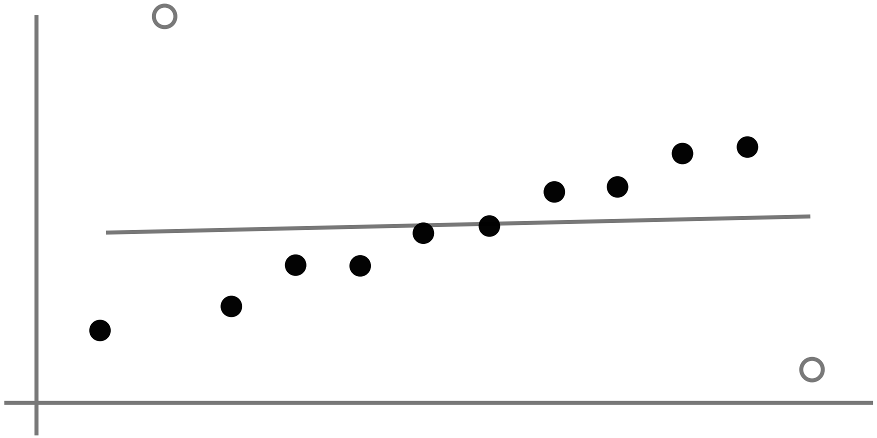
RANdom Sample

Consensus

Robust estimation

- What if set of matches contains gross outliers?





RANSAC

Objective

Robust fit of model to data set S which contains outliers

RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

(ii) 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.

RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

(ii) 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.

(iii) If the subset of S_i is greater than some threshold T, re-estimate the model using all the points in S_i and terminate

RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

(ii) 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.

(iii) If the subset of S_i is greater than some threshold T, re-estimate the model using all the points in S_i and terminate

(iv) If the size of S_i is less than T, select a new subset and repeat the above.

RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

(ii) 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.

(iii) If the subset of S_i is greater than some threshold T, re-estimate the model using all the points in S_i and terminate

(iv) If the size of S_i is less than T, select a new subset and repeat the above.

(v) 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

Distance threshold

Choose t so probability for inlier is α (e.g. 0.95)

- Often empirically.
- Zero-mean Gaussian noise σ then the square of the point distance $d_{\perp}^2$ follows a X_m^2 distribution with m = codimension of model.

(dimension+codimension=dimension space)

Codimension	Model	t^2
1	l,F	$3.84\ \sigma^2$
2	H,P	$5.99\ \sigma^2$
3	T	$7.81\ \sigma^2$

How many samples?

Choose N so that, with probability p, at least one random sample is free from outliers. e.g. p=0.99. Suppose e is the propability that any selected point is an outlier, then at least N selections (each of s points) are require so that $\left(1-(1-e)^s\right)^N = 1-p$.

Sample size	Proportion of Outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

$$N = \frac{log(1-p)}{log(1-(1-e)^s)}$$

Acceptable consensus set?

- Typically, terminate when the consensus set is similar to the expected number of inliers

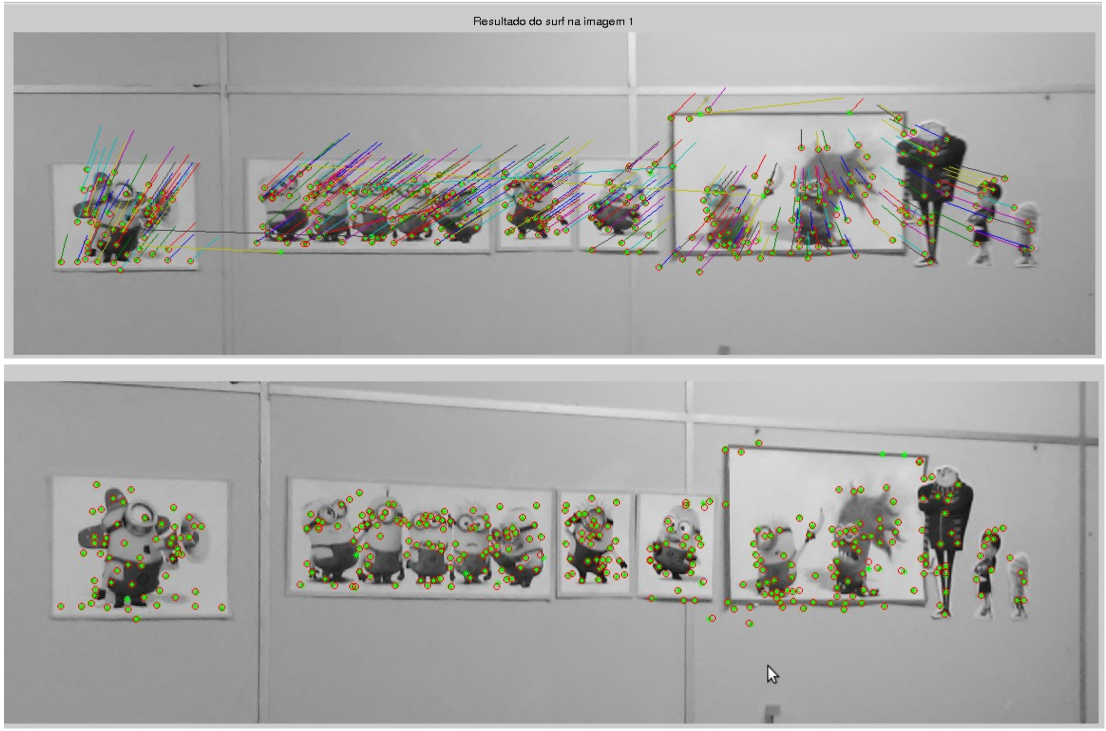
$$\boxed{T = (1 - e)n}$$

- Adaptively determining the number of samples:
  - e is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield e=0.2
  - N=∞, *sample_count* =0
  - While N >*sample_count* repeat
    - Choose a sample and count the number of inliers
    - Set e=1-(number of inliers)/(total number of points)
    - Recompute N from e, usually with p = 0.99
    - Increment the *sample_count* by 1
  - Terminate

$$\boxed{N = log(1 - p)/log\left(1 - (1 - e)^s\right)}$$

Robust Algorithm

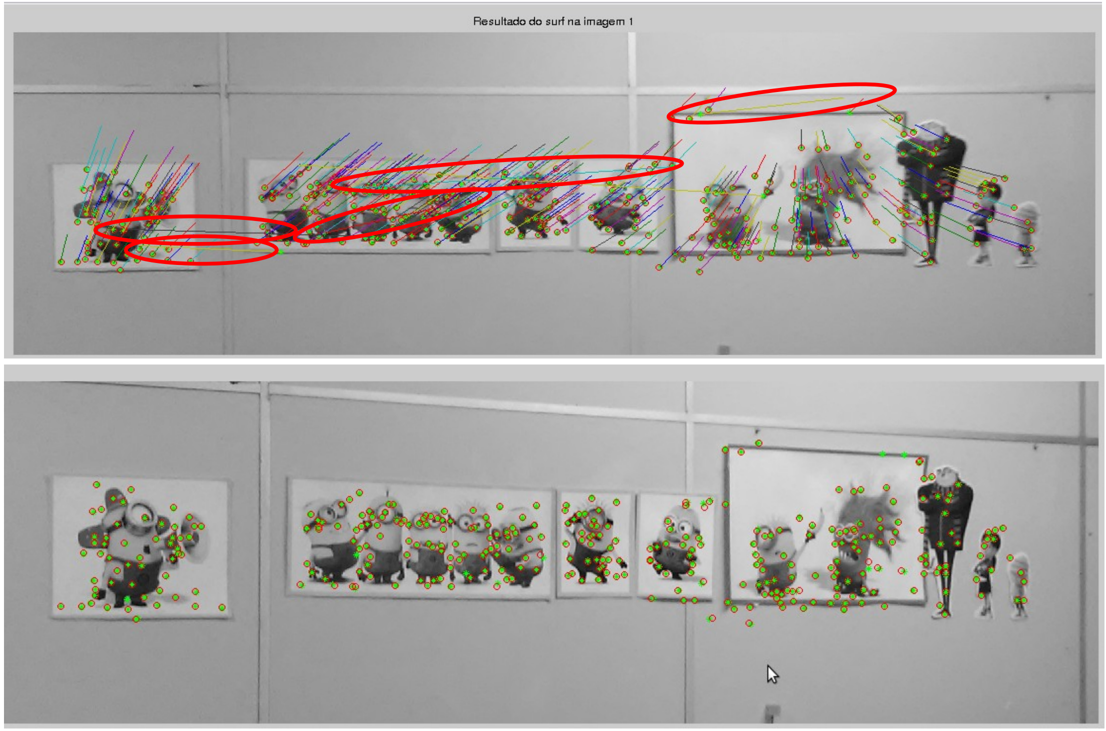
- Use RANSAC to maximize the number of inliers



Robust Algorithm

- Use RANSAC to maximize the number of inliers

Bad matches → outliers



Automatic computation of H using RANSAC

<u>Objective</u>

Compute homography between two images

Automatic computation of H using RANSAC

Objective

Compute homography between two images

Algorithm

(i) **Interest points:** Compute interest points in each image

Automatic computation of H using RANSAC

Objective

Compute 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 some similarity measure

Automatic computation of H using RANSAC

Objective

Compute 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 some similarity measure

(iii) **RANSAC robust estimation:** Repeat for N samples

(a) Select 4 correspondences and compute H

(b) Calculate the distance d_i for each putative match

(c) Compute the number of inliers consistent with H ($d_i < t$)

Choose H with most inliers

Automatic computation of H using RANSAC

Objective

Compute 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 some similarity measure

(iii) **RANSAC robust estimation:** Repeat for N samples

(a) Select 4 correspondences and compute H

(b) Calculate the distance d_i for each putative match

(c) Compute the number of inliers consistent with H ($d_i < t$)

Choose H with most inliers

(vii) **Optimal estimation:** re-estimate H from all inliers by minimizing ML cost function with Levenberg-Marquardt

Automatic computation of H using RANSAC

<u>Objective</u>

Compute homography between two images

<u>Algorithm</u>

(i) **Interest points:** Compute interest points in each image

(ii) **Putative correspondences:** Compute a set of interest point matches based on some similarity measure

(iii) **RANSAC robust estimation:** Repeat for N samples

(a) Select 4 correspondences and compute H

(b) Calculate the distance d_i for each putative match

(c) Compute the number of inliers consistent with H ($d_i < t$)

Choose H with most inliers

(vii) **Optimal estimation:** re-estimate H from all inliers by minimizing ML cost function with Levenberg-Marquardt

(viii) **Guided matching:** Determine more matches using prediction by computed H

Optionally iterate last two steps until convergence

Example

Image 1

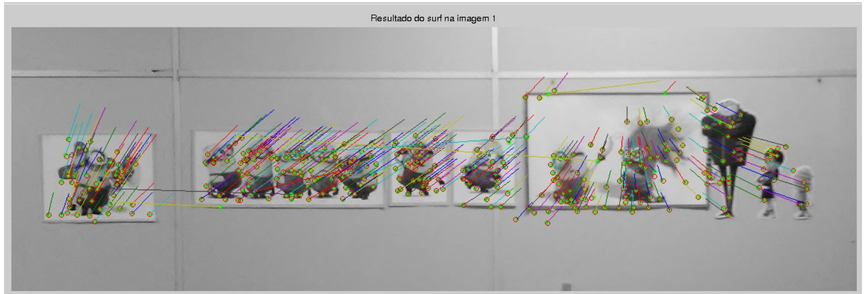


Image 2

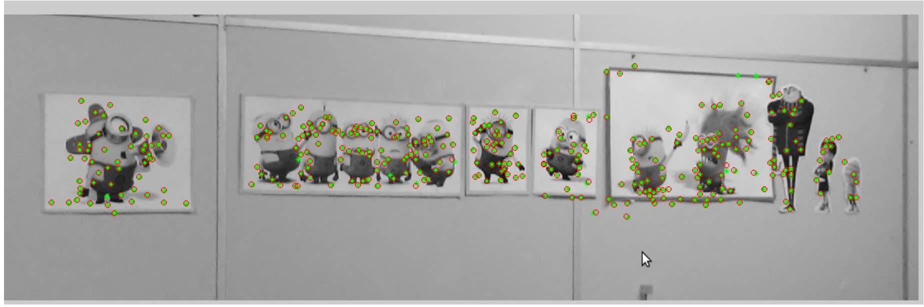


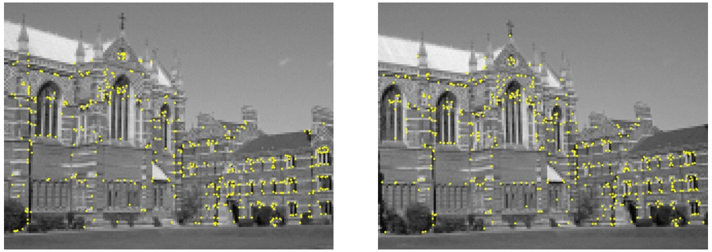
Image 2 $\rightarrow$ Image 1

Image 1 $\rightarrow$ Image 2

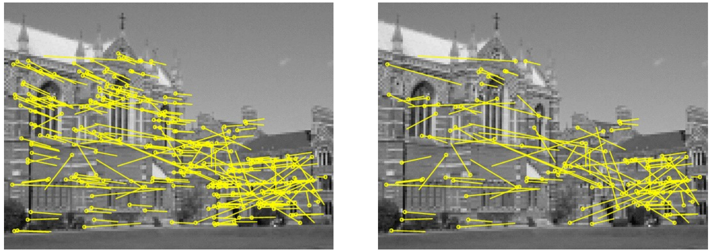
Determine putative correspondences

- Compare interest points using similarity measure:
  - SAD, SSD, ZNCC on small neighborhood
- If motion is limited, only consider interest points with similar coordinates
- More advanced approaches exist, based on invariance…
  - SIFT
  - SURF
  - others

Example: robust computation

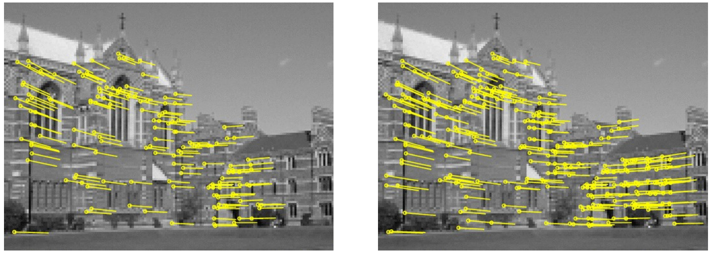


Interest points (500/image)



Putative correspondences (268)

Outliers (117)



Inliers (151)

Final inliers (262)

Credits

- Richard Hartley and Andrew Zisserman. **Multiple View Geometry in Computer Vision.** Cambridge, ISBN 0521623049
- Based on slides from Marc Pollefeys https://www.cs.unc.edu/~marc/mvg/slides.html