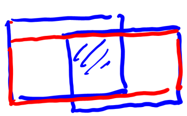
8 Image Matching

8.1 Image Matching

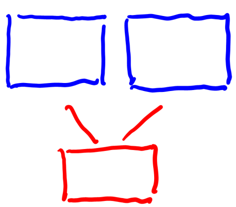
Align similar images to compare them or process them together

Example applications

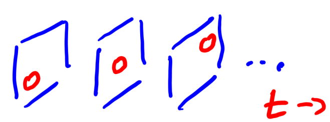
- Mosaicing (alignment of overlapping image parts)
- Multiview image processing
- Medical imaging
- Motion compensation
- Object detection, recognition, tracking



panorama imaging



2D + depth



motion compensation

Homogeneous Coordinates

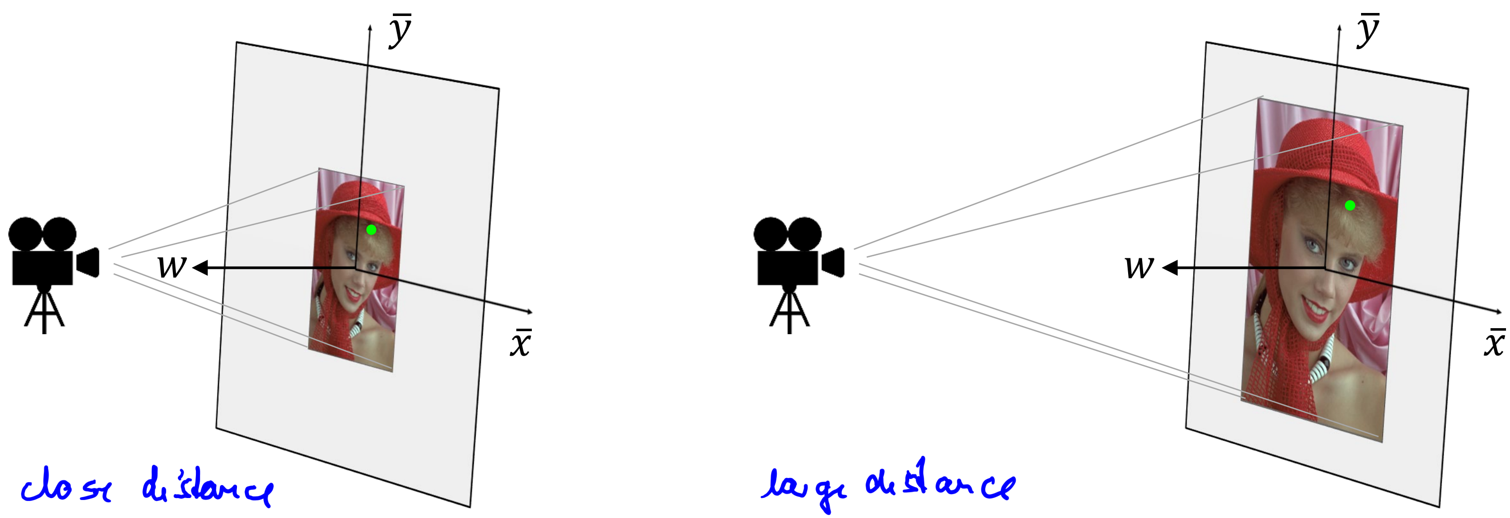
Homogeneous coordinates: $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y}, w)$; w ~ distance from projector to screen

- Projective space

Cartesian coordinates: $\boldsymbol{x} = (\bar{x}/w\,, \bar{y}/w)$

- Euclidean space

Illustration: different projections of the same image



Homogeneous Coordinates

Homogeneity: $(\bar{x}, \bar{y}, w) = (\alpha\bar{x}, \alpha\bar{y}, \alpha w) \quad \forall\alpha \neq 0$

- Converting from homogeneous to Cartesian is unique but **not** vice versa

a) cart. -> hom. $(x, y) \rightarrow (\bar{x}, \bar{y}, 1)$

b) $(\bar{x}, \bar{y}, w) \rightarrow (\bar{x}/w, \bar{y}/w, 1)$

„correct" homogeneous coordinates as used e.g. in computer graphics

Points at **infinity**: $w = 0$

$(\bar{x}, \bar{y}, 0) \rightarrow (\infty, \infty, 0)$

With homogeneous coordinates all affine and projective transforms are matrix multiplications

- Example: translation -> advantage of homogeneous coordinates

$$\underbrace{\begin{bmatrix} x' \\ y' \end{bmatrix}}_{\text{second image}} = \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\text{first image}} + \underbrace{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}_{\text{rel. shift}} \quad <=> \quad [\bar{x}' \quad \bar{y}' \quad 1] = [\bar{x} \quad \bar{y} \quad 1] \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}}_{\text{matrix}}$$

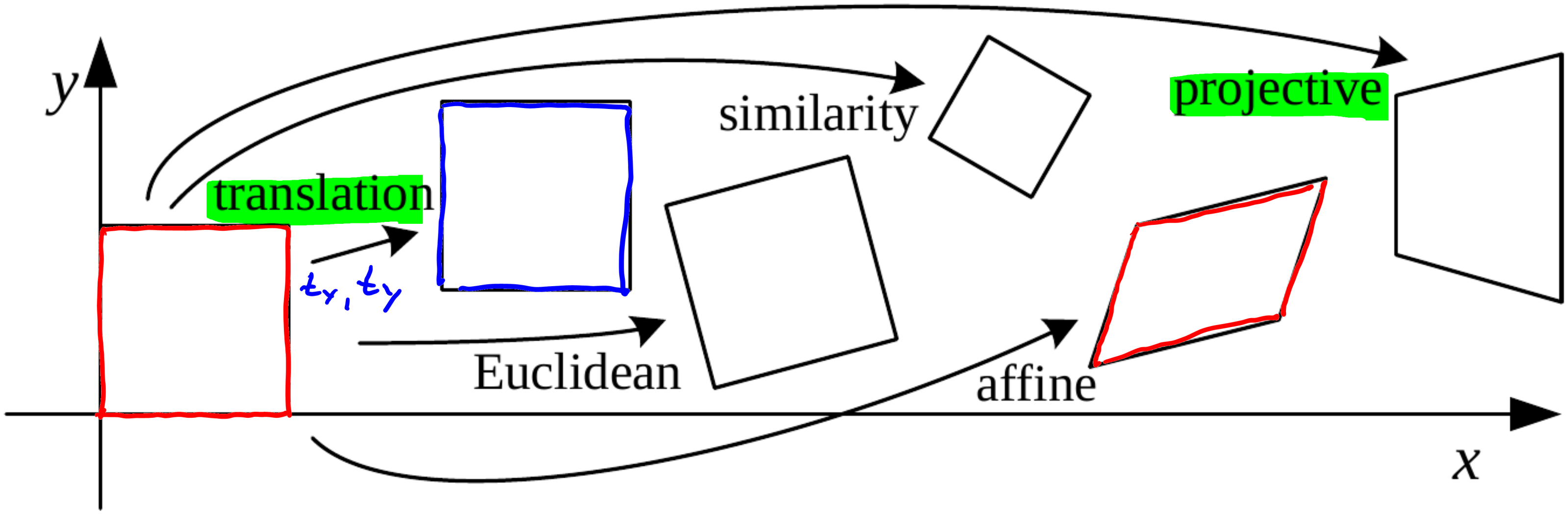
Cartesian coordinates

Homogeneous coordinates

8.2 Taxonomy of 2D Correspondence Maps

Correspondence map also known as motion model

- Reference image coordinates: $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y}, 1)$
- Input image coordinates: $\bar{\boldsymbol{x}}' = (\bar{x}', \bar{y}', 1)$



$$\boxed{\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}}\boldsymbol{T}}$$

$\boldsymbol{T}$ – transformation matrix

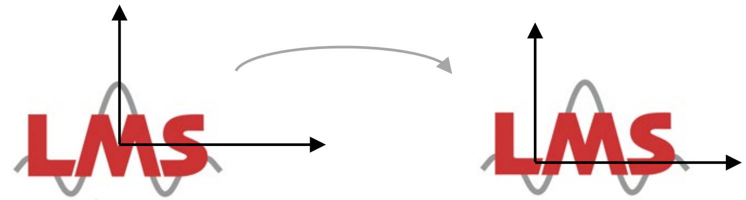
↳ defines number of parameters

R. Szeliski, „Computer Vision: Algorithms and Applications“, 2010

Correspondence Maps

Translation

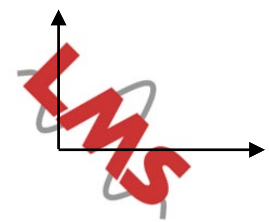
$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$



2 parameters

Euclidean transformation (rotation and/or translation)

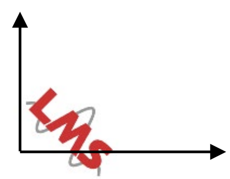
$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ t_x & t_y & 1 \end{bmatrix}$$



3 parameters

Similarity (scaled rotation)

$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \begin{bmatrix} s\cos\theta & s\sin\theta & 0 \\ -s\sin\theta & s\cos\theta & 0 \\ t_x & t_y & 1 \end{bmatrix}$$



4 parameters

Affine Transformation

General form:

$$\boxed{\overline{\boldsymbol{x}}' = \overline{\boldsymbol{x}} \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}}$$

6 parameters

Preserves parallel lines

Combination of scaling, rotation, translation and shear

- It can be obtained by **concatenation** (sequential matrix multiplication)

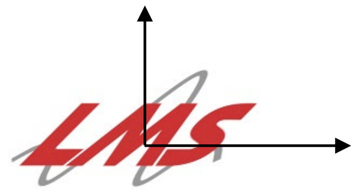
$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Rotation & Translation Rotation Translation

Affine Transformation

Vertical shear

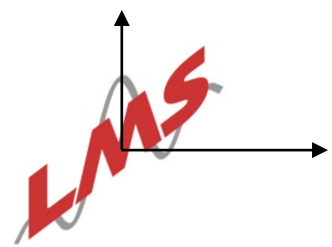
$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{T_{VS}}$$



⇨ x-offset depends on y-position

Horizontal shear

$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \overbrace{\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{T_{HS}}$$



⇨ y-offset depends on x-position

Homography

Perspective (or projective) **transformation**

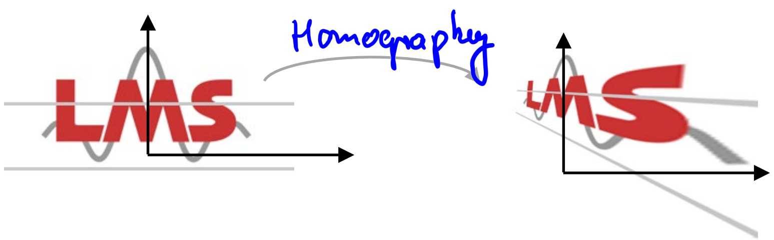
$$\bar{\boldsymbol{x}}' = \bar{\boldsymbol{x}} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Unlike affine transformation homography does *not* preserve the homogeneous coordinate w equal to 1

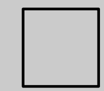
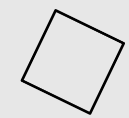
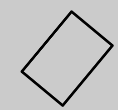
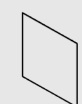
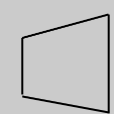
- Normalization is necessary to get Cartesian coordinates

Straight lines are preserved, but not parallel lines as for affine case

Example:



2D Correspondence Maps - Summary

Name	Matrix	DOF	Preserves:	Icon
Translation	$\begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \boldsymbol{t} & 1 \end{bmatrix}_{3\times3}$	2	Orientation + …	
Rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \boldsymbol{t} & 1 \end{bmatrix}_{3\times3}$	3	Lengths + …	
Similarity	$\begin{bmatrix} s\boldsymbol{R} & \mathbf{0} \\ \boldsymbol{t} & 1 \end{bmatrix}_{3\times3}$	4	Angles + …	
Affine	$[\boldsymbol{A}]_{3\times3}$	6	Parallelism + …	
Projective	$[\boldsymbol{H}]_{3\times3}$	8	Straight lines	

#parameters ; w is arbitrary

Degrees of Freedom (DOF) – number of free parameters in transformation matrix

How to Find the Correspondence Map?

Direct methods

- Based on pixel values
- Search (direct matching) or gradient-based methods
- Block matching, optical flow…

Feature-based methods

- Based on (robust) feature detection
- Motion parameters estimated using feature correspondences

Simplest matching procedure

- Suitable only for translational motion (t_x, t_y)
- Sub-pixel accuracy possible by pixel interpolation

Match with displacement (k, l) found by **error minimization**

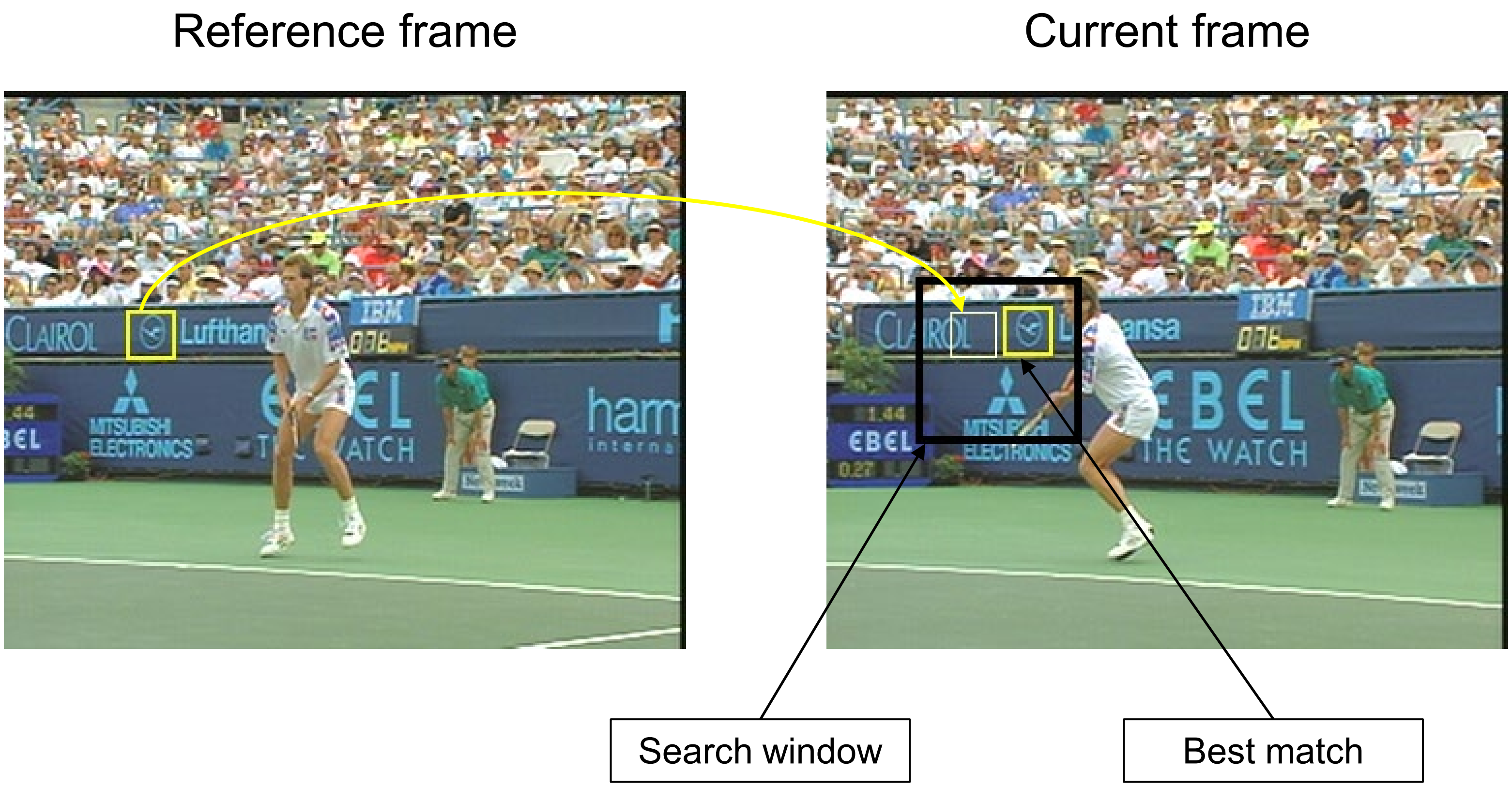
- Sum of Squared Differences (SSD) between current image $s[m,n]$ and reference image $g[m,n]$

$$E_{SSD}[k,l] = \sum_{\substack{(m,n)\in B \\ \text{Block}}} (\underbrace{s[m,n]}_{\text{current image}} - \underbrace{g[m+\overset{t_x}{k}, n+\overset{t_y}{l}]}_{\text{reference image}})^2$$

typ. 8x8; 16x16 for B

- Calculated per image block B in a search range for $\boldsymbol{k}$
- Other error metrics: Sum of Absolute Differences (SAD), $(\cdot)^2 \rightarrow |\cdot|$ cross-correlation, mutual information, etc.
- Sensible to changes in brightness and contrast

Block Matching Example



Block Matching: Search Strategies

Full search

- All possible displacements within search area are checked $\forall t_x, t_y$
- Computationally expensive (brute force approach)
- Always finds global minimum

Conjugate direction search

- Alternate search in x and y direction (reduce 2D problem to 1D)
- Can be trapped in local minimum

Coarse to fine

- Full search for large motion, then refine with fine small motion

Disadvantage of block matching

- Only accounts for uniform translational motion for whole blocks

8.4 Optical Flow

Consider space-time continuous brightness function $s(x(t), y(t), t)$

Constant brightness assumption: $\frac{\mathrm{d}s}{\mathrm{d}t} = 0$ (*)

- Apparent brightness of objects remains constant between frames

Expand via chain rule:

$$(*) \qquad \frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\partial s}{\partial x}\underbrace{\frac{\mathrm{d}x}{\mathrm{d}t}}_{u} + \frac{\partial s}{\partial y}\underbrace{\frac{\mathrm{d}y}{\mathrm{d}t}}_{v} + \frac{\partial s}{\partial t} = 0$$

$u = \frac{\mathrm{d}x}{\mathrm{d}t}$ horizontal shift

$v = \frac{\mathrm{d}y}{\mathrm{d}t}$ vertical shift

Therefore:

$$\boxed{\frac{\partial s}{\partial x}u + \frac{\partial s}{\partial y}v = -\frac{\partial s}{\partial t}}$$

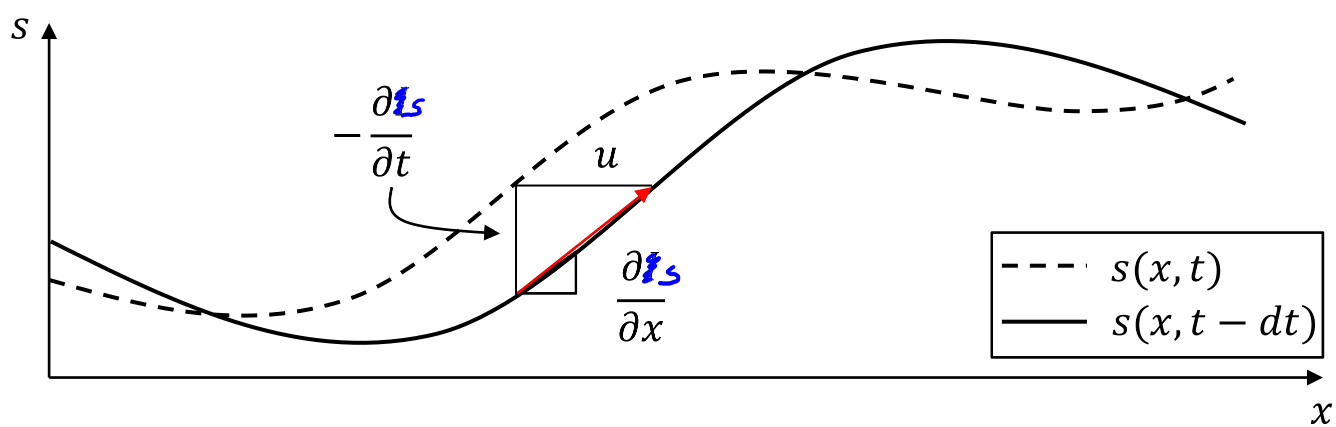
Spatial gradients

Temporal gradient (frame difference)

Geometric Interpretation

Gradient-based image matching

- 1D illustration (simplification)



$$\frac{\partial s}{\partial x}u + \frac{\partial s}{\partial y}v = -\frac{\partial s}{\partial t}$$

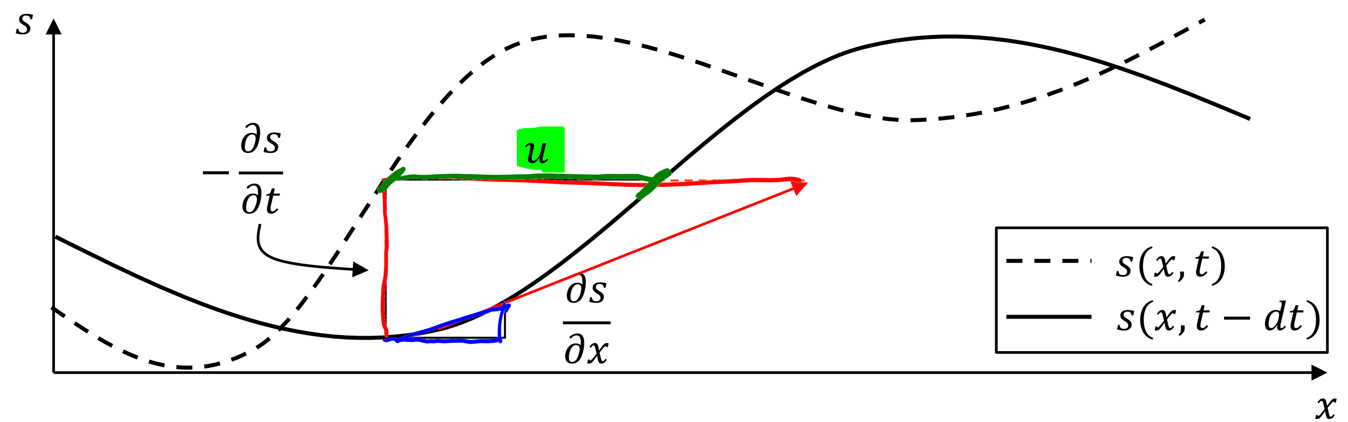
Works well for locally linear motion

- Well modeled by Taylor first order approximation

Geometric Interpretation

Optical flow is **not** a good approximation for too large motions

- They tend to be locally non linear



$$\frac{\partial s}{\partial x}u + \frac{\partial I}{\partial y}v \neq -\frac{\partial s}{\partial t}$$

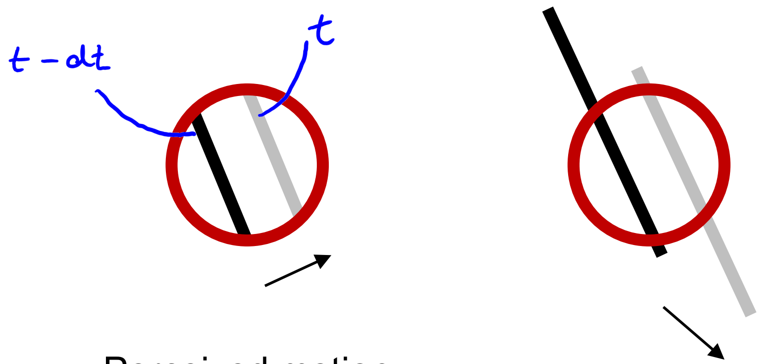
Aperture Problem

Optical flow equation

- One equation, $\underbrace{\text{two unknowns}}_{\text{shifts } u,v}$ → no unique solution

Only detects motion in direction of spatial gradient (**normal flow**)

- Motion orthogonal to spatial gradient cannot be estimated in local region (**aperture problem**)



Perceived motion through local aperture

True motion

Barber pole illusion
for small aperture (o) perceive correct motion (↙)

Solution

- Combine at least two observations with gradients of different directions

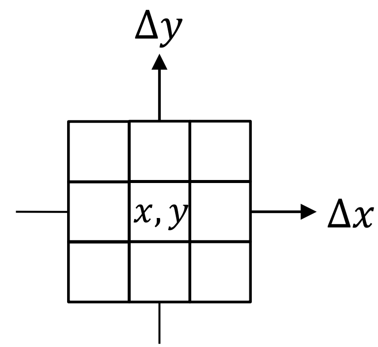
8.5 Lucas-Kanade Algorithm

Method to estimate optical flow

Consider local neighborhood of N pixels

- This leads to an overdetermined system of N equation with 2 unknowns

$$\boldsymbol{S}\underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\text{motion to be estimated}} = \boldsymbol{t} \qquad \text{where} \qquad \begin{aligned} \boldsymbol{S} &= \Big[\overbrace{s_x[x+\Delta x, y+\Delta y]}^{\partial/\partial x\, s} \quad \overbrace{s_y[x+\Delta x, y+\Delta y]}^{\partial/\partial y\, s}\Big]_{N\times 2} \\ \boldsymbol{t} &= \Big[\underbrace{-s_t[x+\Delta x, y+\Delta y]}_{\partial/\partial t\, s}\Big]_{N\times 1} \end{aligned}$$



Minimize mean squared error: $\left(\boldsymbol{S}\begin{bmatrix} u \\ v \end{bmatrix} - \boldsymbol{t}\right)^2 \rightarrow \min$

Least squares solution

- Taking derivatives with respect to (u, v) and setting them to 0

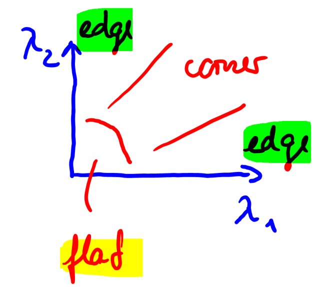
$$\boxed{\begin{bmatrix} u \\ v \end{bmatrix} = \underbrace{\left(\boldsymbol{S}^T\boldsymbol{S}\right)^{-1}\boldsymbol{S}^T}_{\text{pseudo inverse of } \underline{S}}\boldsymbol{t}}$$

Lucas-Kanade Algorithm

Invertibility of **structure matrix** $(\boldsymbol{S}^T\boldsymbol{S})$

$$\boldsymbol{S}^T\boldsymbol{S} = \begin{bmatrix} \sum\limits_N s_x^2 & \sum\limits_N s_x\, s_y \\ \sum\limits_N s_x\, s_y & \sum\limits_N s_y^2 \end{bmatrix}$$

← Remember Harris detector? (6-45)



Not invertible in regions with no structure (flat areas)

- Both eigenvalues are zero

Even if invertible, it can be ill-conditioned

- Ratio of eigenvalues too large (presence of edges) → aperture problem

Not singular for corners and textured areas

Lucas-Kanade Algorithm

Typically, pixels further away from the center at (x, y) are less relevant for motion estimation

- Distance-based weighting (e.g. Gaussian)

$$\boxed{\begin{bmatrix} u \\ v \end{bmatrix} = \left(\boldsymbol{S}^T \boldsymbol{W} \boldsymbol{S}\right)^{-1} \boldsymbol{S}^T \boldsymbol{W} \boldsymbol{t}}$$

$\underline{W}$ like in Harris's corner detection

- $\boldsymbol{W}$ is $N \times N$ **diagonal matrix** containing the weights

Optical flow cannot handle too large motions

- Optical flow estimation on down-sampled images (lower scales) to "reduce" motion

afterwards: upsampling plus motion refinement

Optical Flow Example

Previous frame

$s(x,y,t-\Delta t)$

Current frame

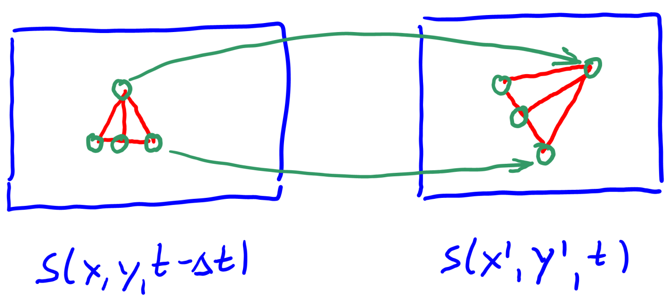
$s(x,y,t)$

Optical flow

motion vector field
(subsampled,
thresholded)

Feature-based Image Matching

1. Detect robust keypoints (Harris, SIFT, SURF, etc.) 1
2. Establish correspondences based on feature descriptors
   - By finding the nearest neighbor in descriptor space
3. Obtain model parameters from correspondences 3
   - Homography estimation



$\Rightarrow$ 3 estimate homography $\hat{=}$ rotation & zoom

8.6 Homography Estimation

Given feature correspondence $(x', y', 1) \leftrightarrow (x, y, 1)$

here: column vectors

$(\bar{x}' = \bar{x} \cdot T)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}}^{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Note: 9 parameters but only 8 degrees of freedom (scale is arbitrary)

This yields a set of equations:

(1) $h_1 x + h_2 y + h_3 = x'$
(2) $h_4 x + h_5 y + h_6 = y'$
(3) $h_7 x + h_8 y + h_9 = 1$

$\longrightarrow$

(1)$-x'$(3)
(2)$-y'$(3)

$\longrightarrow$

$$\boxed{\begin{aligned} -h_1 x - h_2 y - h_3 + (h_7 x + h_8 y + h_9) x' &= 0 \quad (*) \\ -h_4 x - h_5 y - h_6 + (h_7 x + h_8 y + h_9) y' &= 0 \quad (**) \end{aligned}}$$

One correspondence yields two equations

- We need at least 4 points (correspondences) to solve the problem
- Usually we need **many** more (noise, keypoint inaccuracies, outliers…)

Homography Estimation

Given N feature correspondences $(x'_i, y'_i, 1) \leftrightarrow (x_i, y_i, 1)$

- They can be stacked together

In matricial notation:

$$\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_N \end{bmatrix} \boldsymbol{h} = 0 \longrightarrow \boxed{\boldsymbol{A}\boldsymbol{h} = 0}$$

where

$$\boldsymbol{A}_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x'_i x_i & x'_i y_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \end{bmatrix} \begin{matrix} \leftarrow (*) \\ \leftarrow (**) \end{matrix}$$

$$\boldsymbol{h} = [h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6 \quad h_7 \quad h_8 \quad h_9]^T$$

Homography Estimation

The relation $\boldsymbol{Ah} = 0$ never holds in practice

- Noise, keypoint inaccuracies, outliers, different object motions, etc.
- Trivial solution ($\boldsymbol{h}$ - zero vector) is obviously not considered

Solved by minimizing a suitable cost ξ

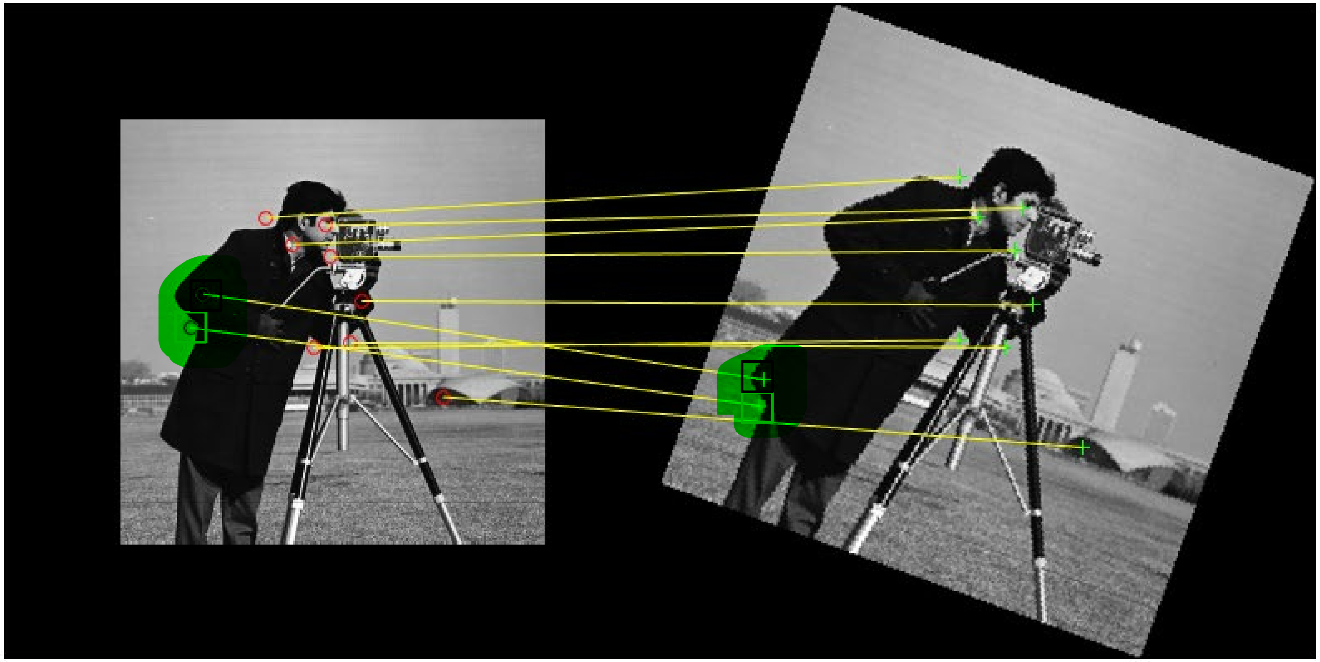
- Algebraic distance $\xi = \|\boldsymbol{Ah}\|$ should be close to zero
- Geometric distance $\xi = \sum_i d(\boldsymbol{x}'_i, \boldsymbol{H}\boldsymbol{x}_i)^2$ map x_i to next frame x'_i
- Symmetric transfer error $\xi = \sum_i [\underbrace{d(\boldsymbol{x}'_i, \boldsymbol{H}\boldsymbol{x}_i)^2}_{\text{forward mapping}} + \underbrace{d(\boldsymbol{x}_i, \boldsymbol{H}^{-1}\boldsymbol{x}'_i)^2}_{\text{backward mapping}}]$
- Reprojection error, Sampson error

$d(\cdot,\cdot)$ – Euclidean distance

Outliers

Outliers are bad matches that have huge effect on cost function

- Inaccurate homography estimation



Problem: how to reject outliers?

8.7 RANSAC

RANdom **SA**mple **C**onsensus

- Rejects outliers by voting (consensus)
- Iterative approach

Concept: hypothesize-and-test framework

1) **Hypothesis:** selects minimal sample set to fit the model
   - In case of homography: 4 feature correspondences
   - Solve for model parameters

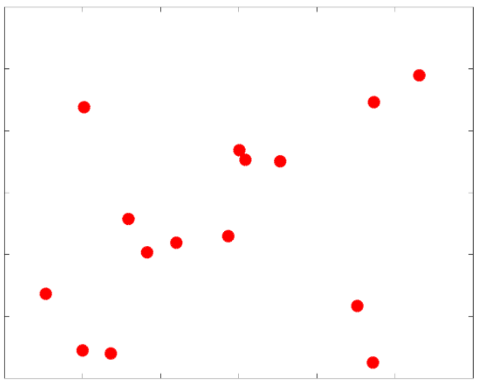
2) **Test:** check all features for compliance with hypothesis
   - Score hypothesis by counting inliers

RANSAC **assumption**: more inliers imply better fit

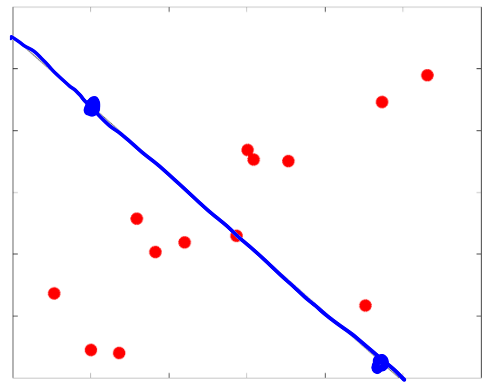
RANSAC

Illustration with 1D case

- Two points required to fit a line

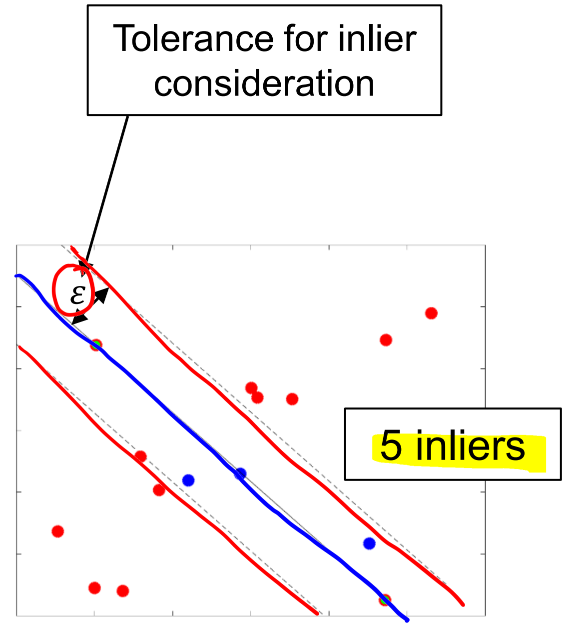


Available sample set



Hypothesis

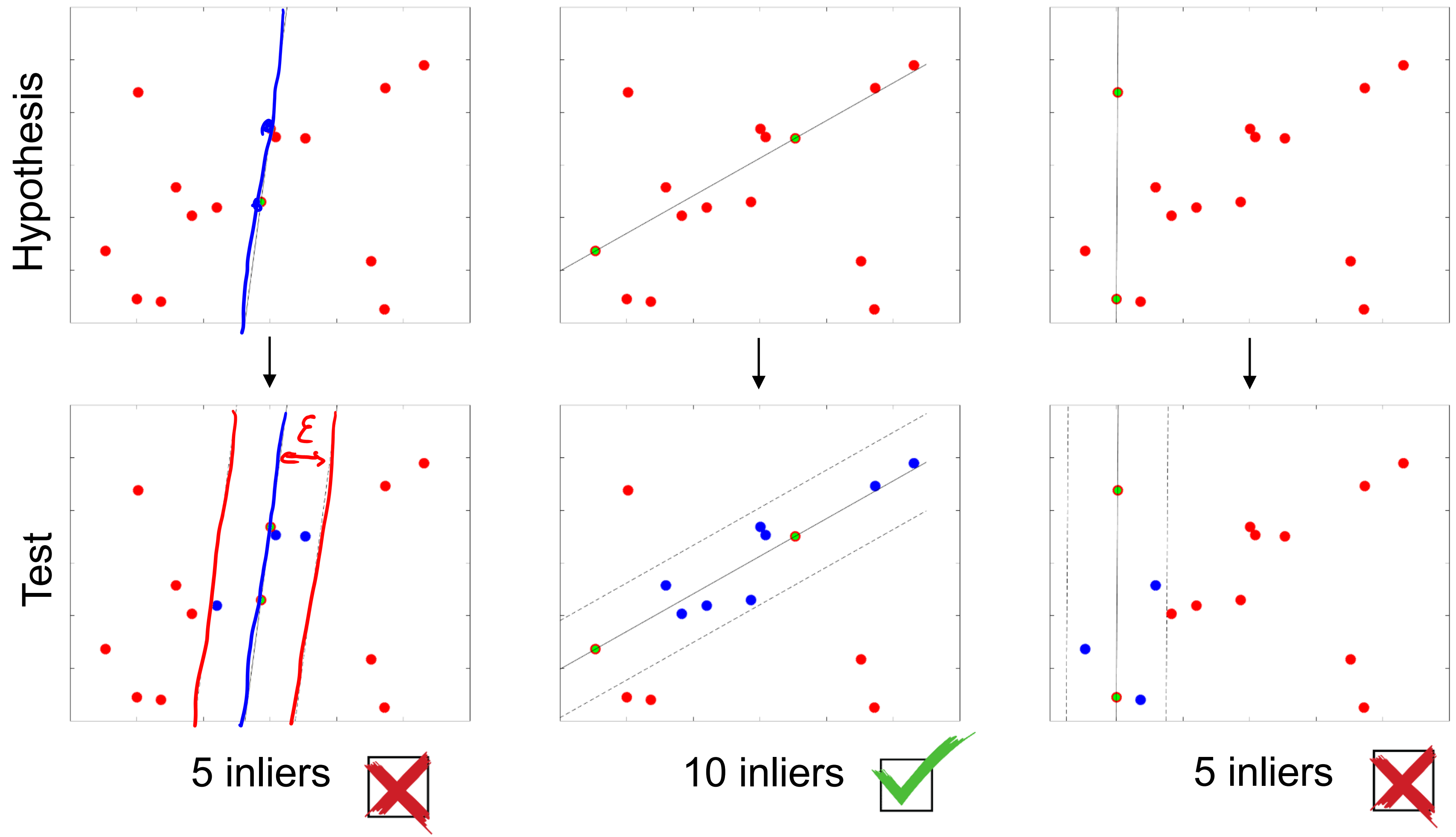
- Get random 2 samples
- Fit model (compute slope & offset)



Test

- Count inliers

RANSAC Illustration



RANSAC for Homography

Iterative procedure:

- 1 Select four features correspondences at random
- 2 Compute homography $\boldsymbol{H}$ (exact solution)

Hypothesis

- 3 Determine inliers
  - Apply $\boldsymbol{H}$ to all features and check $d(\boldsymbol{x}'_i, \boldsymbol{H}\boldsymbol{x}_i)^2 < \varepsilon$

Test

- 4 Keep $\boldsymbol{H}$ with largest set of inliers

5 Re-compute $\boldsymbol{H}$ using all inliers from the largest set

RANSAC Robustness

Let p_{in} be the probability (proportion) of inliers within feature set

Hypothesis steps need 4 pairs to compute homography

- Probability of picking (randomly) 4 inliers?

$$p = p_{\text{in}}^4$$

- Probability that after N iterations we have **not** picked a set of 4 inliers?

$$p_N = \left(1 - p_{\text{in}}^4\right)^N$$

i.e. no good solution has been obtained

Conclusion: it is a bad idea to pick more pairs per iteration

- It will be more likely that an outlier is picked

because $p_{in}^4 \rightarrow \underbrace{p_{in}^m}_{\text{smaller}}$ $(m > 4)$

smaller $\rightarrow$ p_N bigger

RANSAC Robustness

Example: $p_{\text{in}} = 0.5$

- Probability of picking 4 inliers? $p = p_{\text{in}}^4 = 0.5^4 \cong 6\%$
- Probability of not picking 4 inliers after N iterations? (i.e. no good solution found)

$$p_N = \left(1 - p_{\text{in}}^4\right)^N \longrightarrow p_{100} \cong 0.2\%$$

Example: $p_{\text{in}} = 0.1$ only 10%, low!

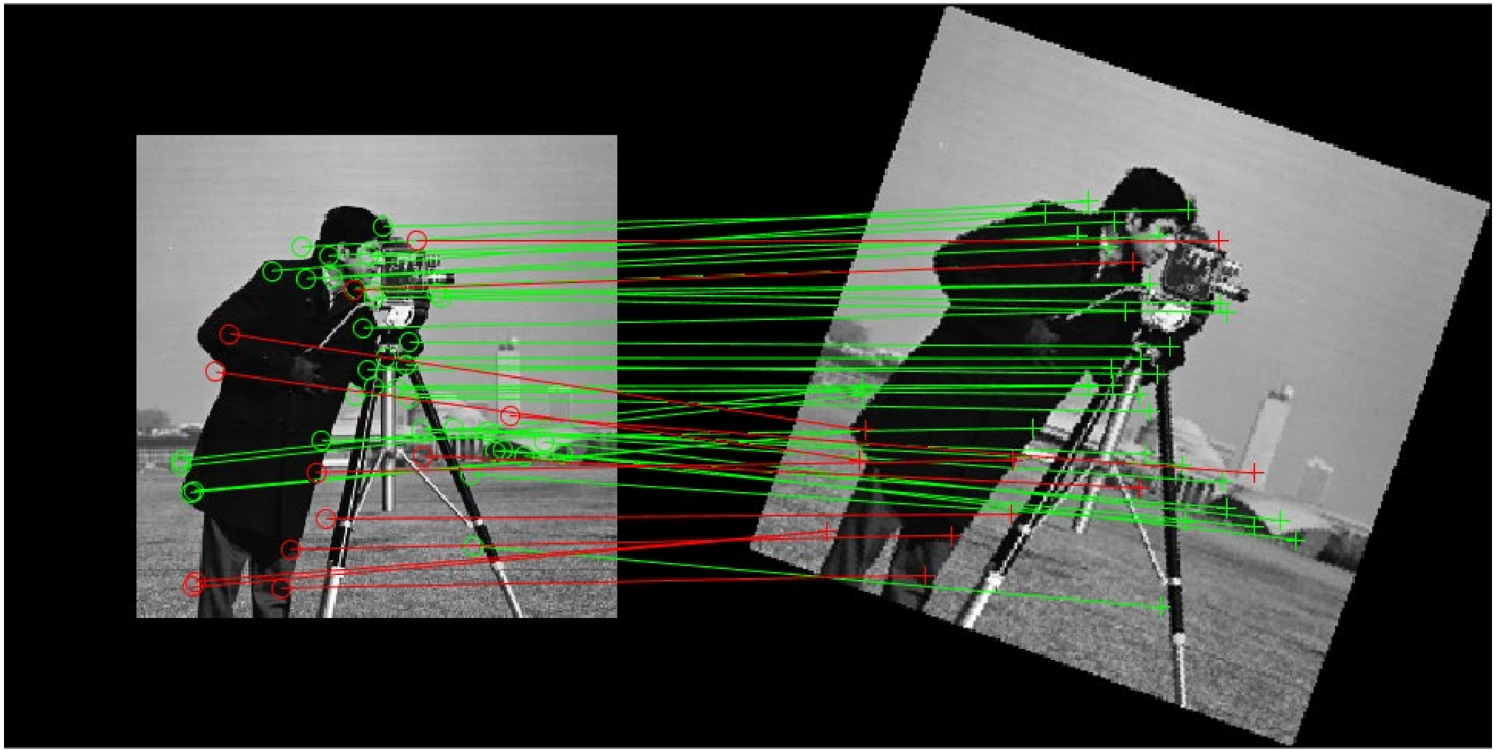
- Probability of picking 4 inliers? $p = p_{\text{in}}^4 = 0.1^4 = 0.01\%$
- Probability of not picking 4 inliers after N iterations?

$$p_{100} \cong 99\% \qquad p_{1000} \cong 90\% \qquad p_{10000} \cong 37\% \qquad p_{50000} \cong 0.7\%$$

Conclusion: having more iterations is more important than higher p_{in}

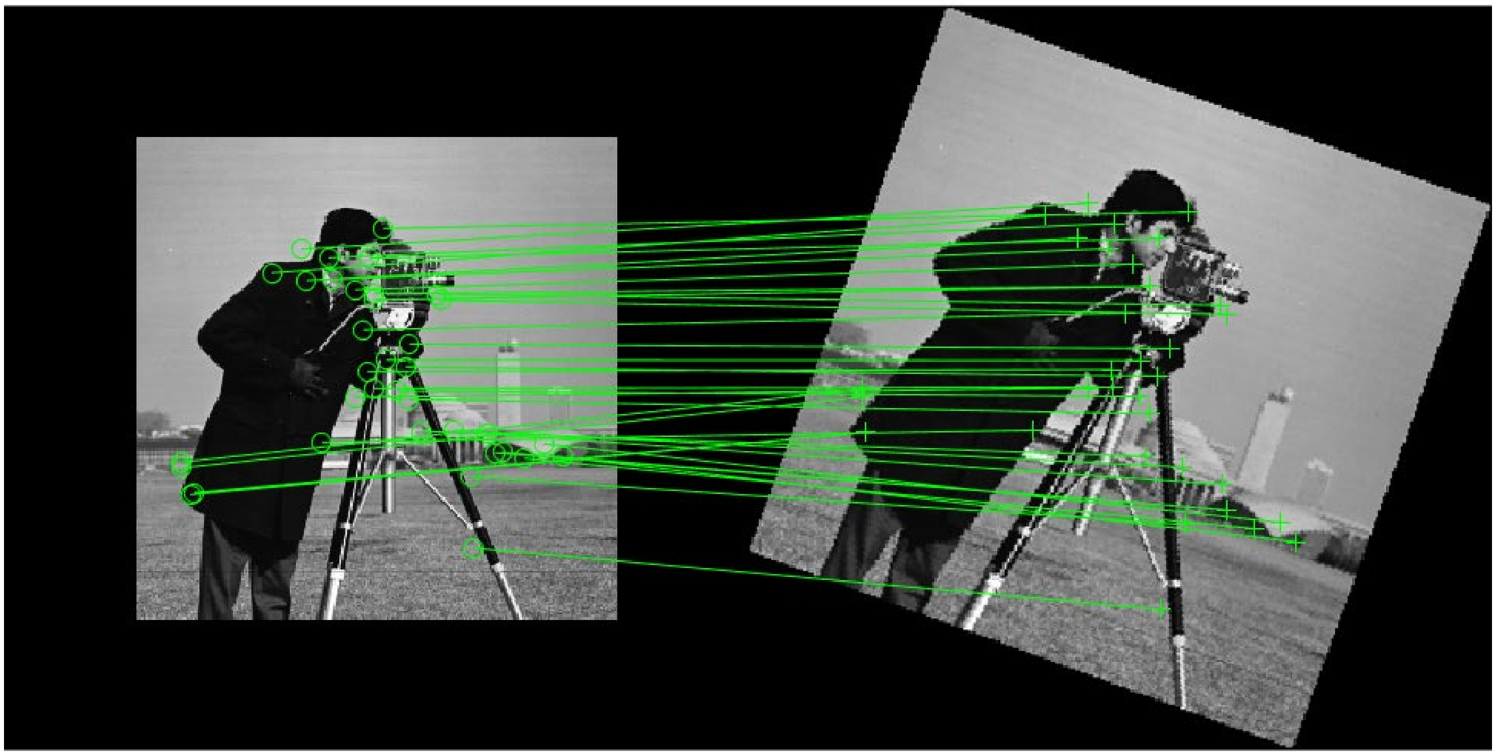
RANSAC Example

SURF matching (48 matches)



RANSAC Example

SURF matching after RANSAC (36 matches)



Note: Number (and quality) of matches after RANSAC depends on how strict test step is (ε!)