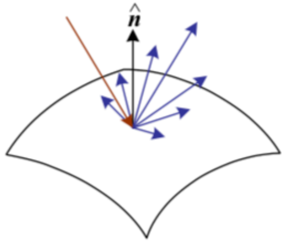
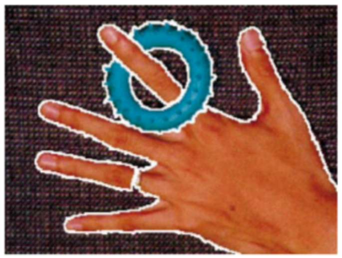


2. Image Formation

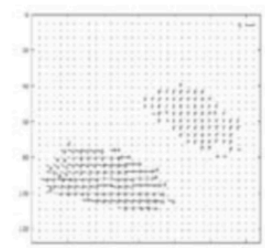
3. Image Processing

4. Features

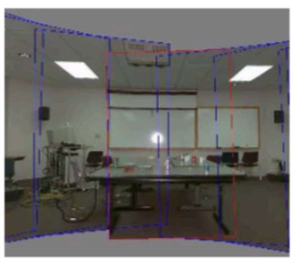


5. Segmentation

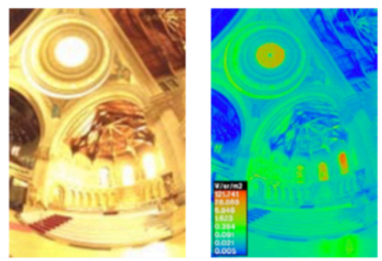
6-7. Structure from Motion



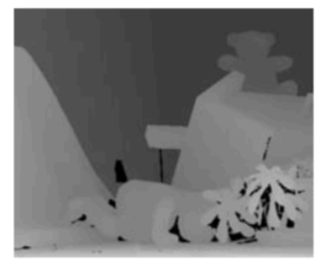
8. Motion



9. Stitching

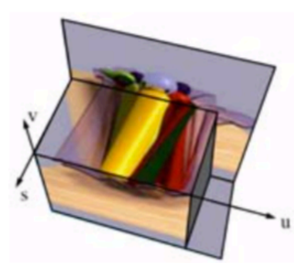


10. Computational Photography



11. Stereo

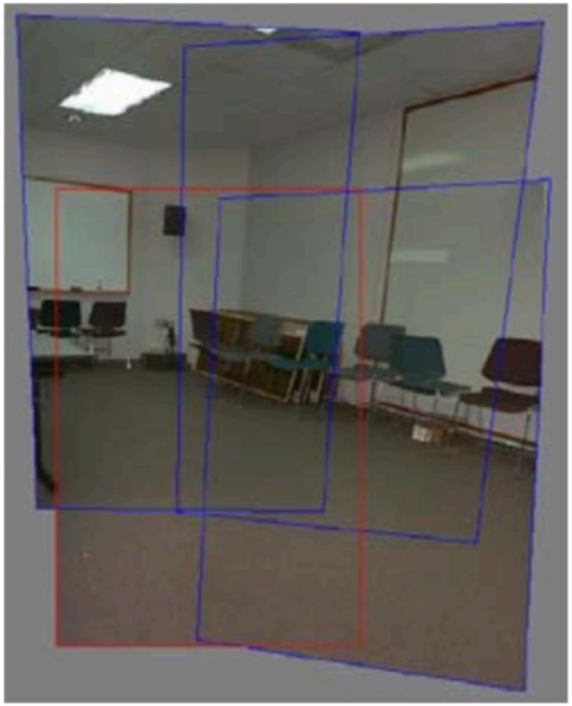
12. 3D Shape

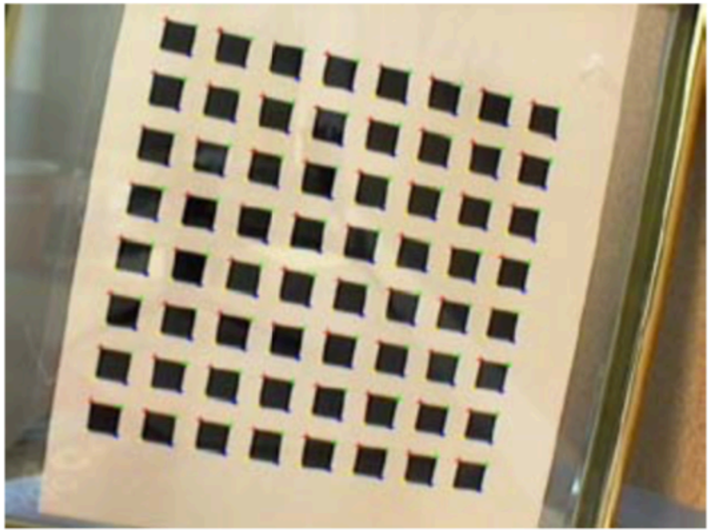


13. Image-based Rendering

14. Recognition

Feature-based Image Alignment





- Geometric image registration
  - 2D or 3D transforms between them
  - Special cases: pose estimation, calibration

2D Alignment

- 3 photos
- Translational model

2D Alignment

- Input:
  - A set of matches $\{(x_i, x_i')\}$
  - A parametric model $f(x; p)$
- Output:
  - Best model p^*
- How?

2D translation estimation

- Input:
  - Set of matches $\{(x_1, x_1'), (x_2, x_2'), (x_3, x_3'), (x_4, x_4')\}$
  - Parametric model: $f(x; t) = x + t$
  - Parameters p == t, location of origin of A in B
- Output:
  - Best model p*

Image credit Szeliski book

2D translation estimation

- Input:
  - Set of matches $\{(x_1, x_1'), (x_2, x_2'), (x_3, x_3'), (x_4, x_4')\}$
  - Parametric model: $f(x; t) = x + t$
  - Parameters p == t, location of origin of A in B
- Question for class:
  - What is your best guess for model p* ??

2D translation estimation

- How?
  - One correspondence x1 = [600, 150], x1' = [50, 50]
  - Parametric model: x' = f(x; t) = x + t
    => t = x'- x

    => t = [50-600, 40-150] = [-550, -100]

2D translation via least-squares

- How?
  - A set of matches $\{(x_i, x_i')\}$
  - Parametric model: $f(x; t) = x + t$
  - Minimize sum of squared residuals:

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2,$$

Image credit Szeliski book

How to solve?

In many cases, parametric model is linear:

Jacobian

$$f(x;p) = x + J(x)p$$

$$\Delta x = x' - x = J(x)p$$

$$E_{LS} = \sum_i \|J(x)p + x - x'_i\|^2 = \sum_i \|J(x_i)p - \Delta x_i\|^2$$

Differentiate and set to 0:

$$2\sum_i J^T(x_i)\left(J(x_i)p - \Delta x_i\right) = 0$$

Normal equations —

$$\left[\sum_i J^T(x_i)J(x_i)\right] p = \sum_i J^T(x_i)\Delta x_i$$

$$Ap = b$$

$$p* = A^{-1}b$$

Hessian

Linear models menagerie

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

- All the simple 2D models are linear!
- Exception: perspective transform

Figure credit Szeliski book

2D translation via least-squares

For translation: $J = I$ and normal equations are particularly simple:

$$\left[\sum_i I^T I\right] p = \sum_i \Delta x_i$$

$$p* = \frac{1}{n} \sum_i \Delta x_i$$

In other words: just **average** the "flow vectors" $\Delta x = x' - x$

Oops I lied !!! Euclidean is not linear!

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

- ~~All the simple 2D models are linear!~~
- Euclidean Jacobians are a function of θ!

Nonlinear Least Squares

$$E_{NLS} = \sum_i \left\| f(x_i;p) - x'_i \right\|^2$$

Linearize around a current guess p:

$$f(x;p+\Delta p) = f(x;p) + J(x;p)\Delta p$$

$$r = x' - f(x;p) = J(x;p)\Delta p$$

$$E_{NLS} = \sum_i \left\| f(x;p) + J(x;p)\Delta p - x'_i \right\|^2 = \sum_i \left\| J(x;p)\Delta p - r_i \right\|^2$$

Differentiate and set to 0:

$$2\sum_i J^T(x_i;p)\left(J(x_i;p)\Delta p - r_i\right) = 0$$

$$\left[\sum_i J^T(x_i;p)J(x_i;p)\right]\Delta p = \sum_i J^T(x_i;p)r_i$$

$$A\Delta p = b$$

$$\Delta p* = A^{-1}b$$

Projective/H

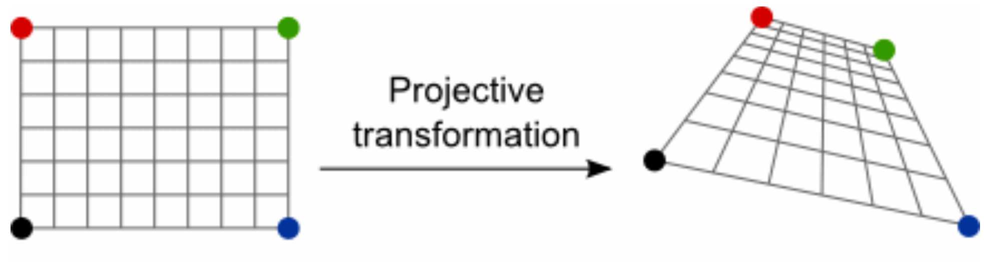


Image credit Graphics Mill (educational Use)

- Jacobians a bit harder
- Parameterization:

$$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \qquad (h_{00}, h_{01}, \ldots, h_{21})$$

- x'= f(x,p):

$$x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \text{ and } y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}.$$

- And Jacobian:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \frac{1}{D} \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix}$$

$$D = h_{20}x + h_{21}y + 1$$

Closed Form H

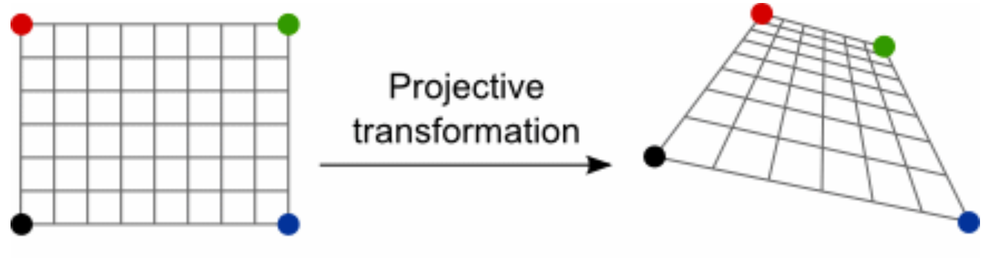


Image credit Graphics Mill

- Taking x'=f(x,p):

$$x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \text{ and } y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}.$$

- Divide both sides by $D = h_{20}x + h_{21}y + 1$:

$$\begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix}$$

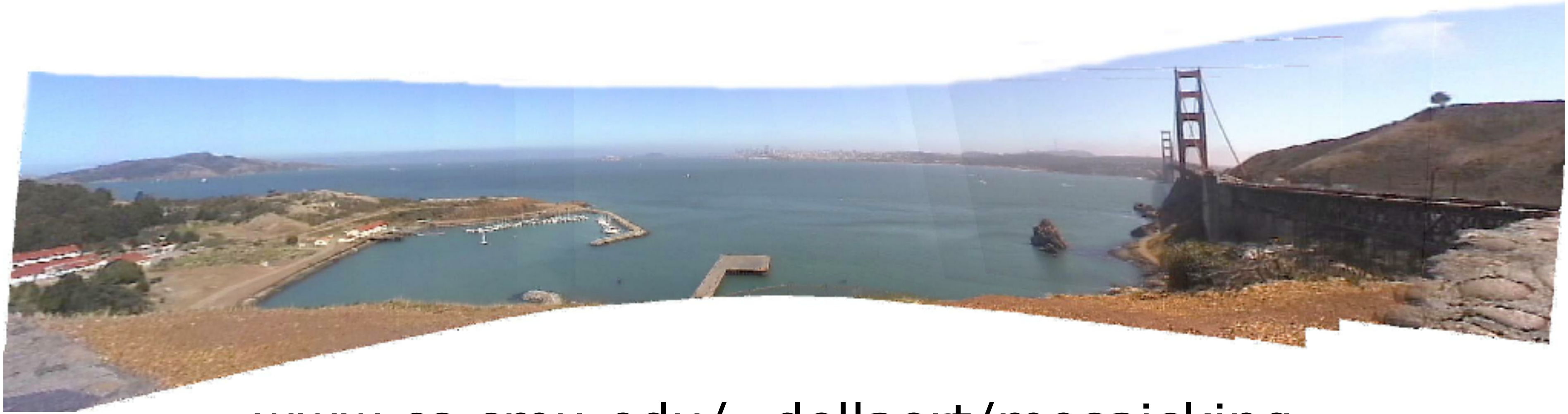
- 4 matches => system of 8 linear equations

RANSAC

Motivation

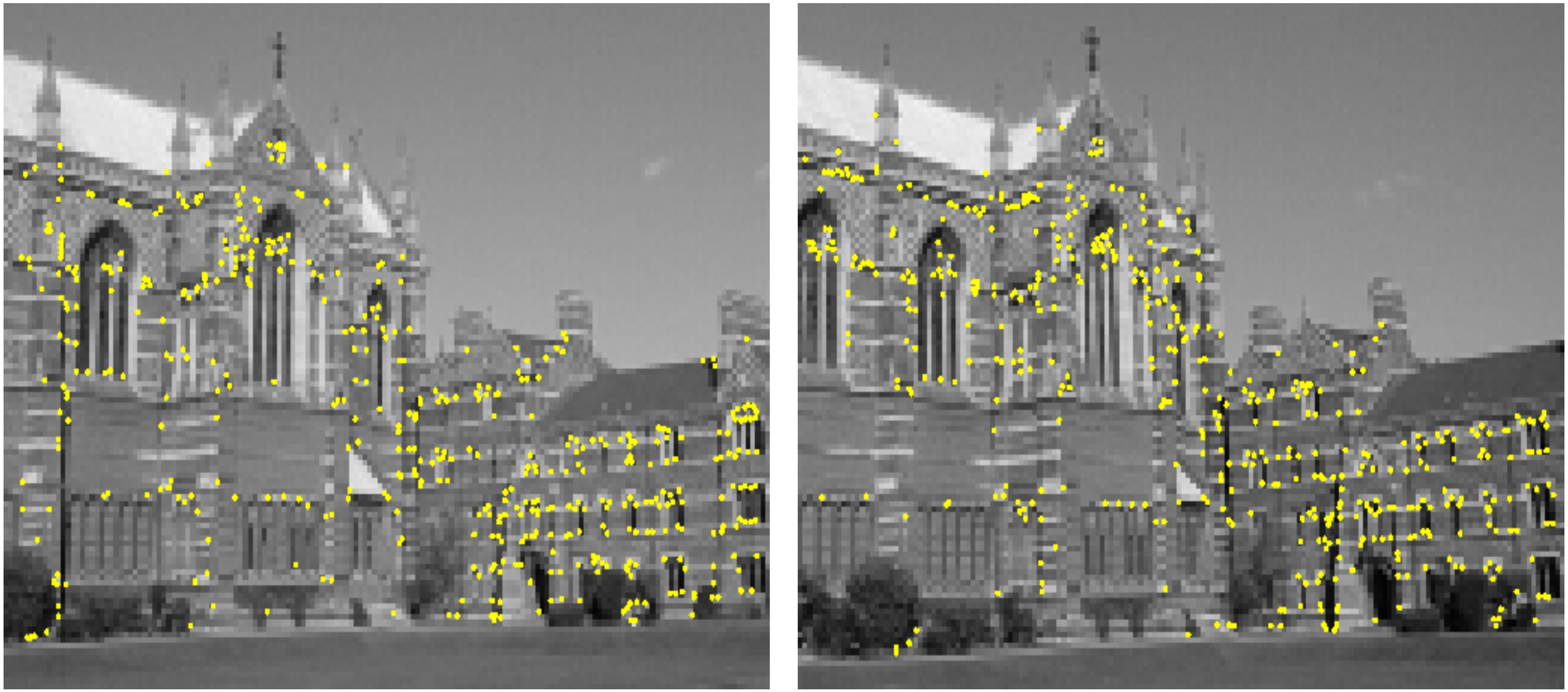
- Estimating motion models
- Typically: points in two images
- Candidates:
  - Translation
  - Homography
  - Fundamental matrix

Mosaicking: Homography

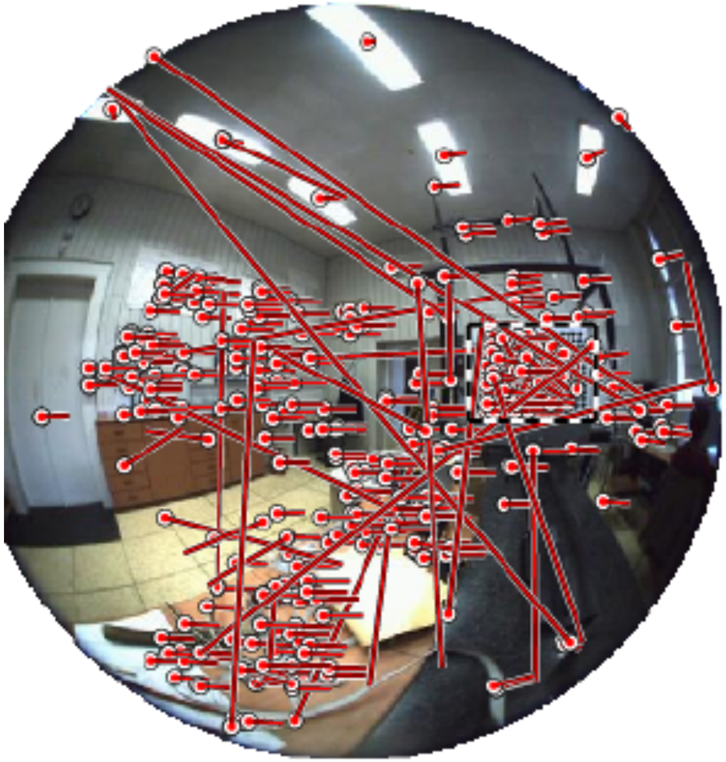


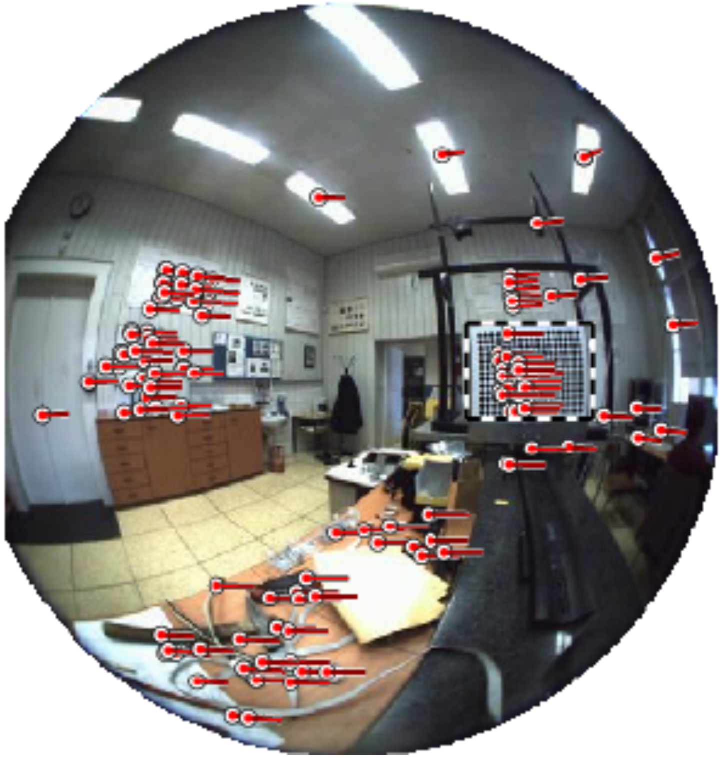
www.cs.cmu.edu/~dellaert/mosaicking

Two-view geometry (next lecture)



Omnidirectional example

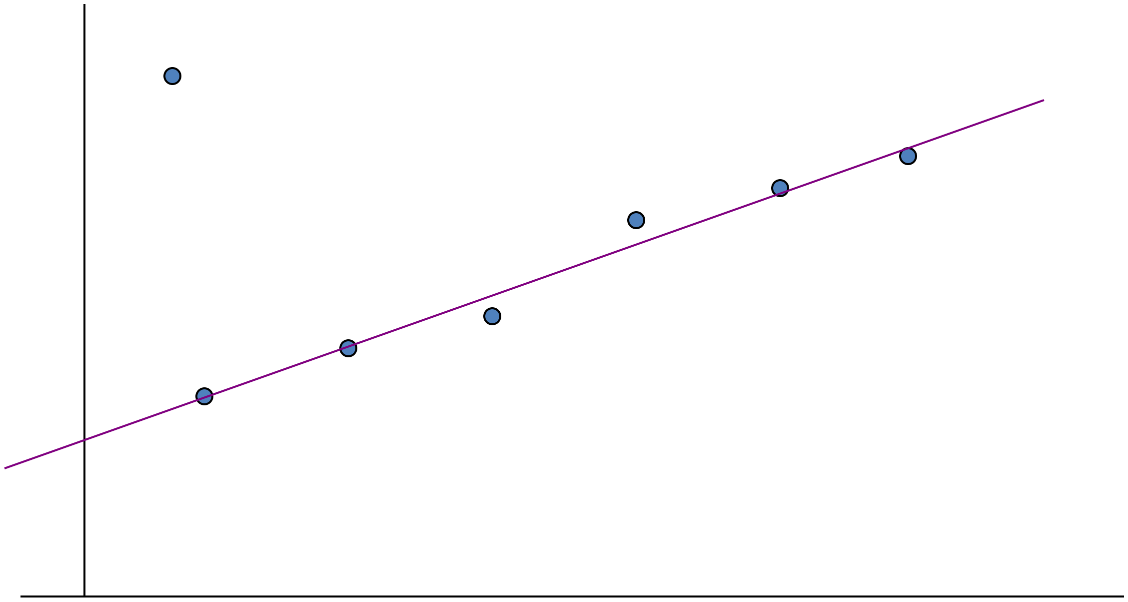




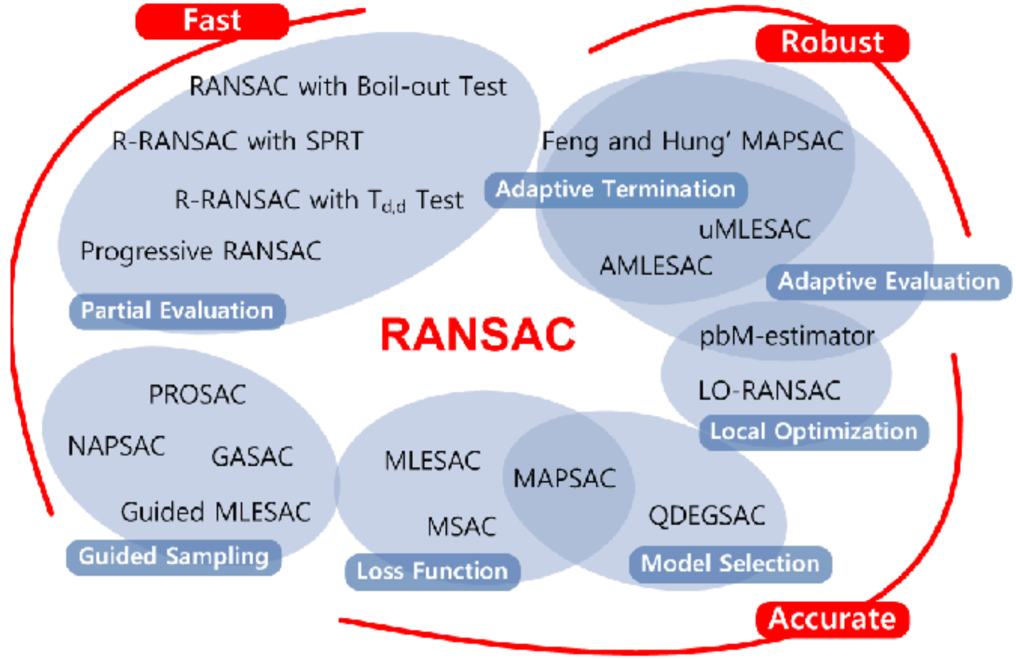
Images by Branislav Micusik, Tomas Pajdla,
cmp.felk.cvut.cz/ demos/Fishepip/

Simpler Example

- Fitting a straight line



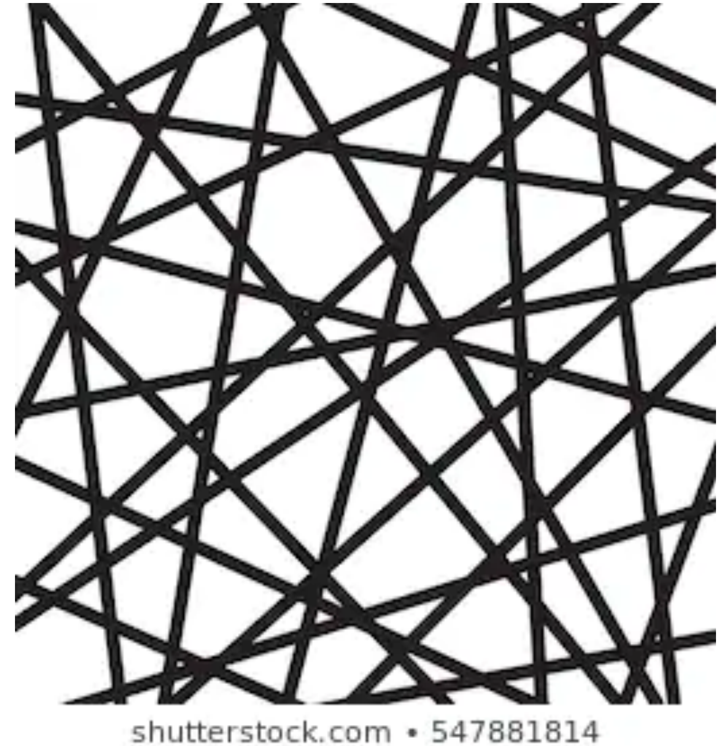
Discard Outliers



- No point with d>t
- RANSAC:
  - RANdom SAmple Consensus
  - Fischler & Bolles 1981
  - Copes with a large proportion of outliers

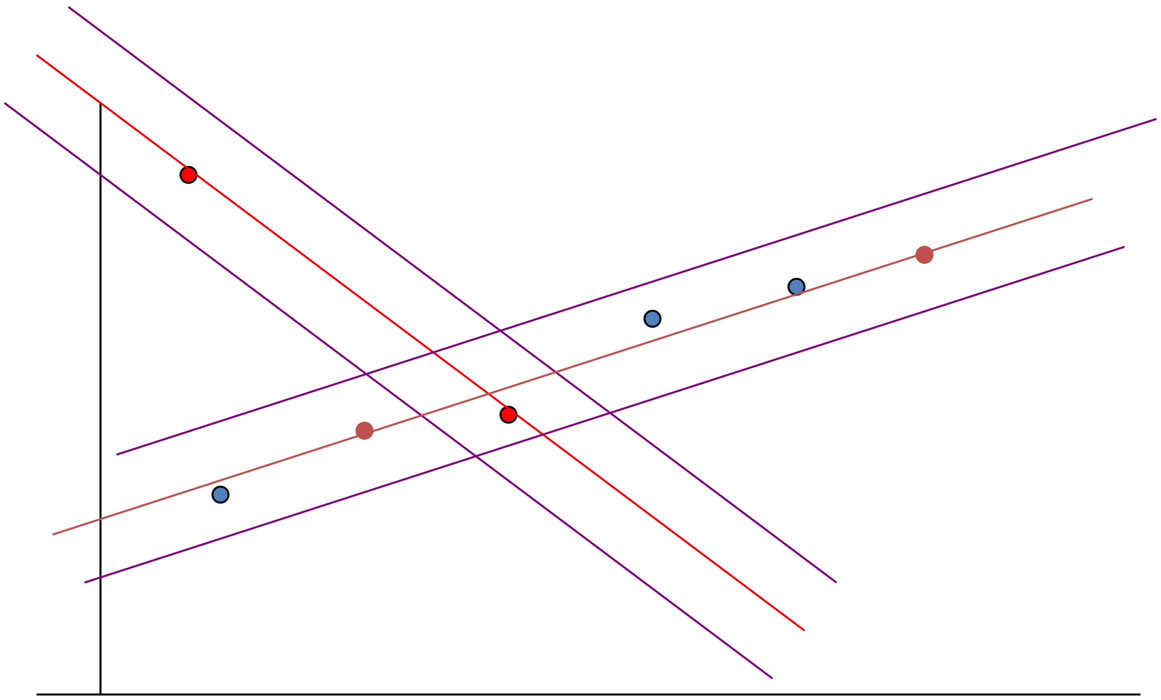
Image credit Choi et al BMVC 2009

Main Idea



- Select 2 points at random
- Fit a line
- “Support” = number of inliers
- Line with most inliers wins

Why will this work ?



Best Line has most support

- More support -> better fit

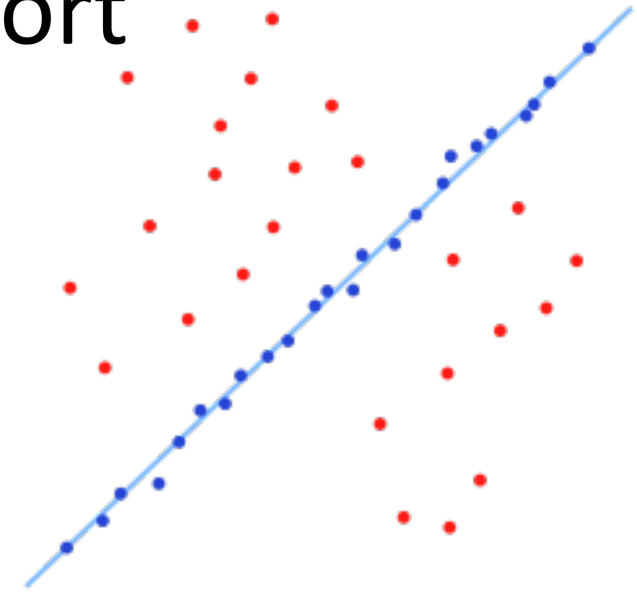
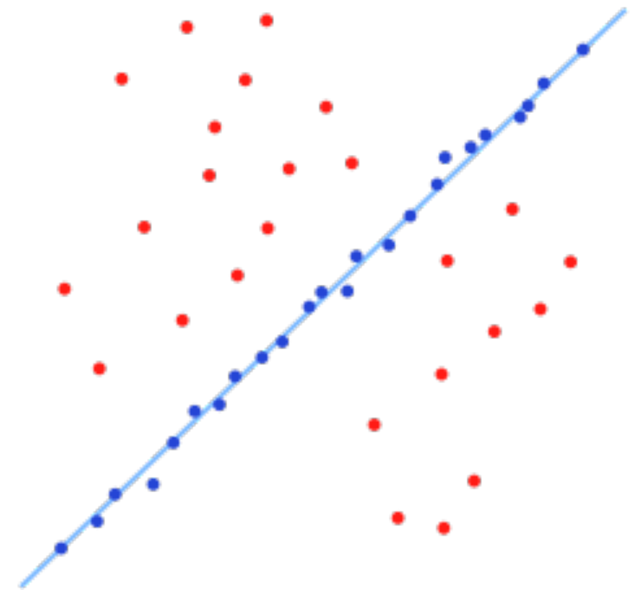


Image credit Wikipedia

In General

- Fit a more general model
- Sample = minimal subset
  - Translation ?
  - Homography ?
  - Euclidean transorm ?

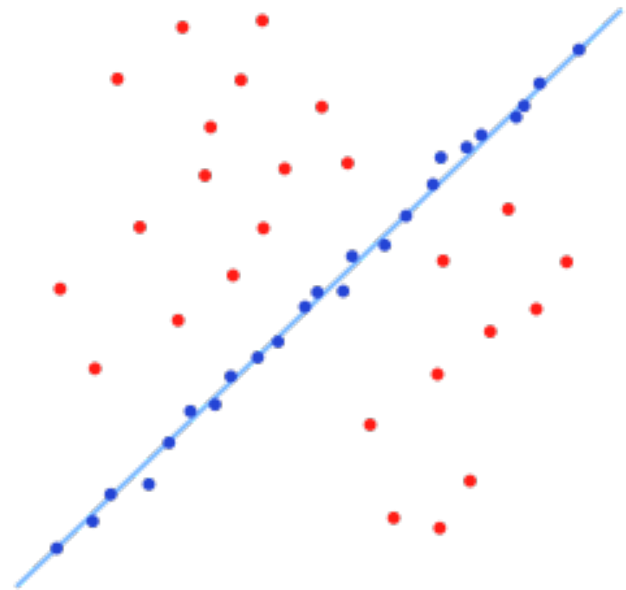
RANSAC



- Objective:
  - Robust fit of a model to data D
- Algorithm
  - Randomly select s points
  - Instantiate a model
  - Get consensus set D_i
  - If $|D_i|>T$, terminate and return model
  - Repeat for N trials, return model with max $|D_i|$

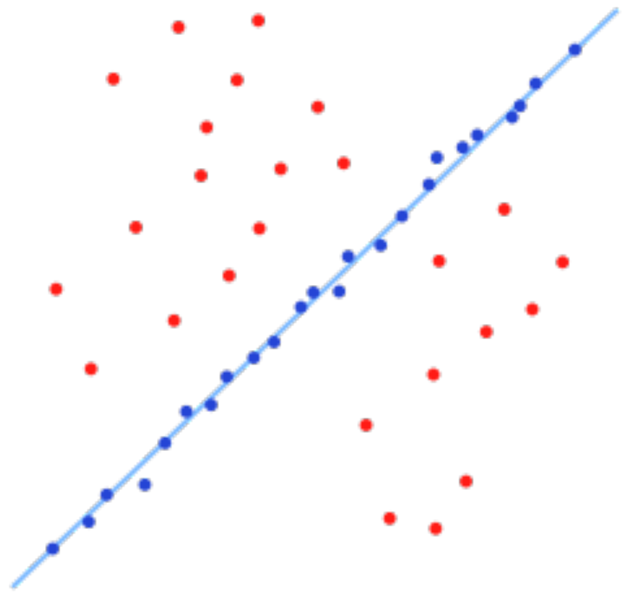
Image credit Wikipedia

Distance Threshold



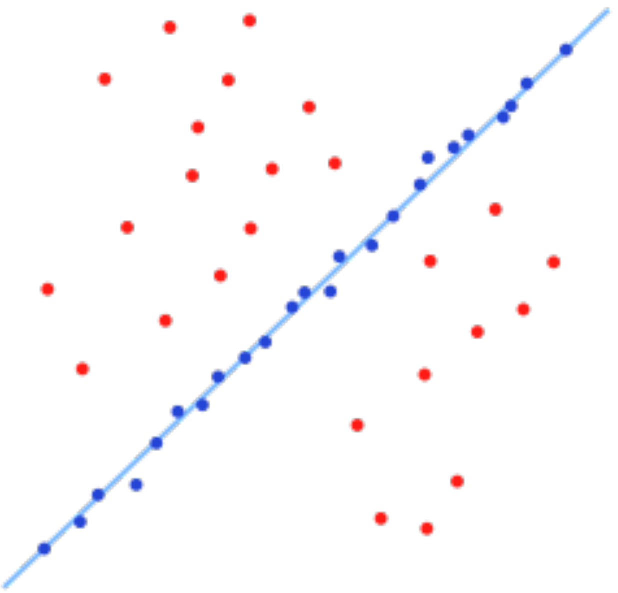
- Requires noise distribution
- Gaussian noise with σ
- Chi-squared distribution with DOF m
  - 95% cumulative:
  - Line, F: m=1, t=3.84 σ^2
  - Translation, homography: m=2, t=5.99\ σ^2
- I.e. -> 95% prob that d<t is inlier

How many samples ?



- We want: at least one sample with all inliers
- Can't guarantee: probability P
- E.g. P = 0.99

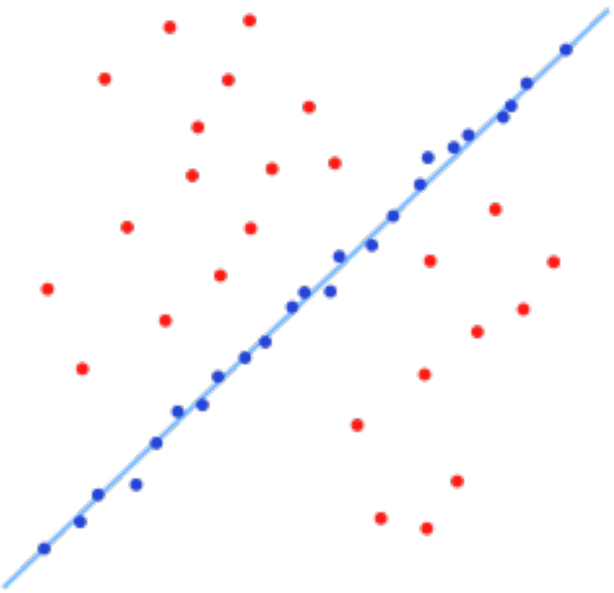
Calculate N



- If etha = outlier probability
- proportion of inliers p = 1-etha
- P(sample with all inliers) = p^s
- P(sample with an outlier) = $1-p^s$
- P(N samples an outlier) = $(1-p^s)$^N
- We want P(N samples an outlier) < 1-P e.g. 0.01
- $(1-p^s)$^N < 1-P
- N > $\log(1-P)/\log(1-p^s)$

Image credit Wikipedia

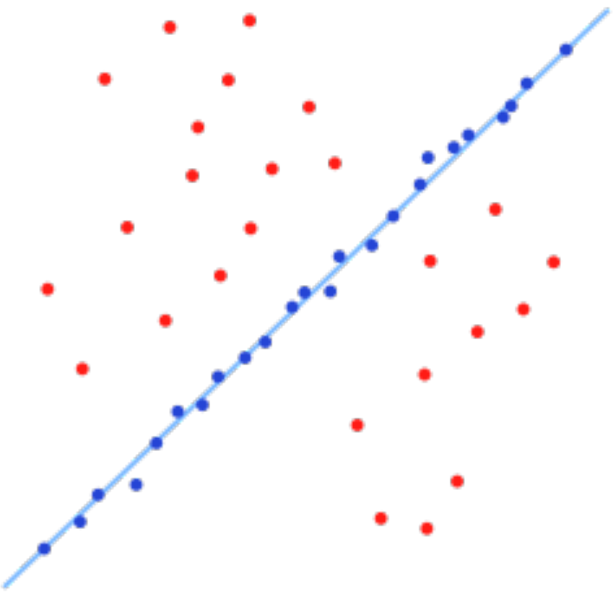
Example



- P=0.99
- s=2, etha=5% => N=2
- s=2, etha=50% => N=17
- s=4, etha=5% => N=3
- s=4, etha=50% => N=72
- s=8, etha=5% => N=5
- s=8, etha=50% => N=1177

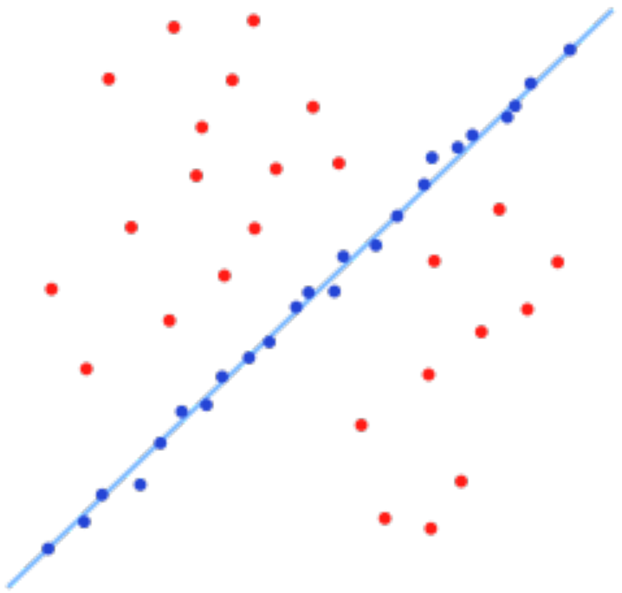
Image credit Wikipedia

Remarks



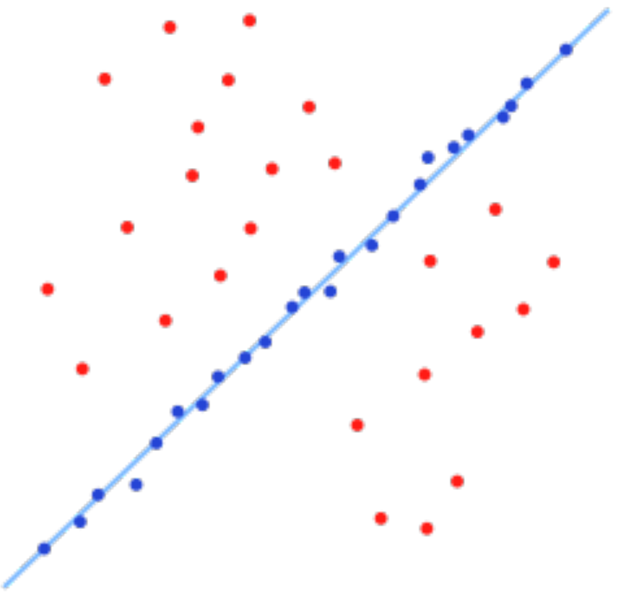
- N = f(etha), not the number of points
- N increases steeply with s

Threshold T



- Terminate if $|D_i|>T$
- Rule of thumb: T ≈ #inliers
- So, T = (1-etha)n = pn

Adaptive N



- When etha is unknown ?
- Start with etha = 50%, N=inf
- Repeat:
  - Sample s, fit model
  - -> update etha as |outliers|/n
  - -> set N=f(etha, s, p)
- Terminate when N samples seen