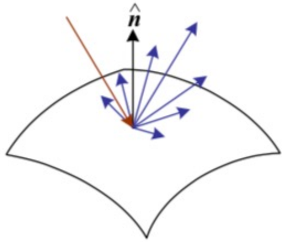


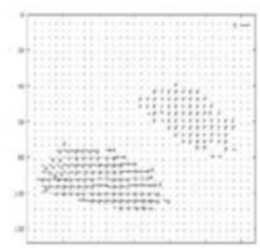
2. Image Formation

3. Image Processing

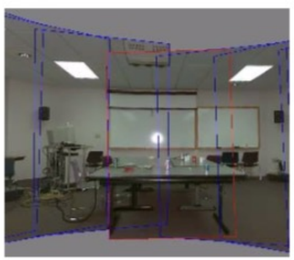
4. Features

5. Segmentation

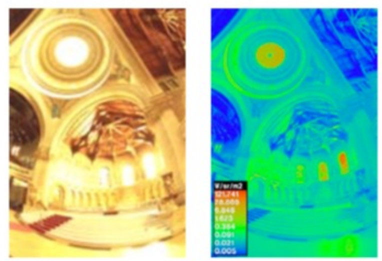
6-7. Structure from Motion



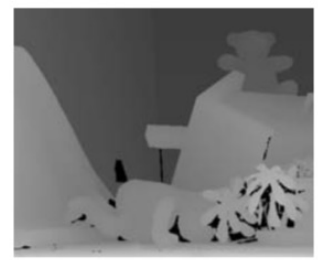
8. Motion



9. Stitching

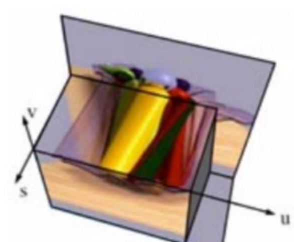


10. Computational Photography



11. Stereo

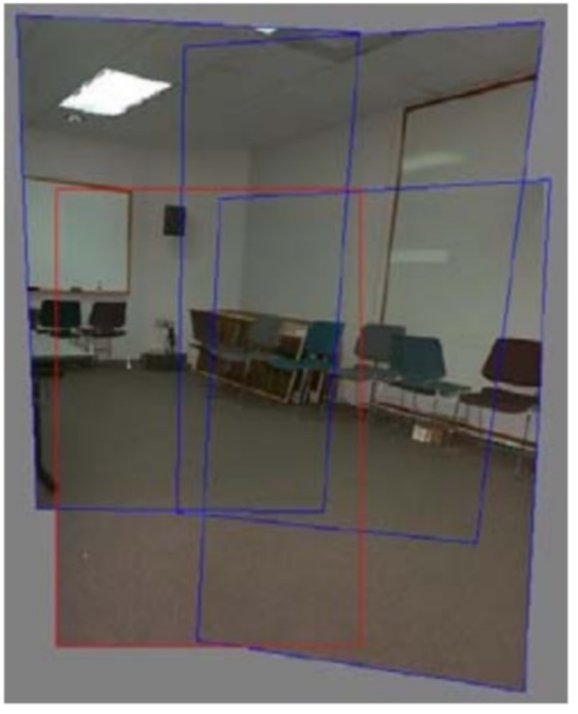
12. 3D Shape

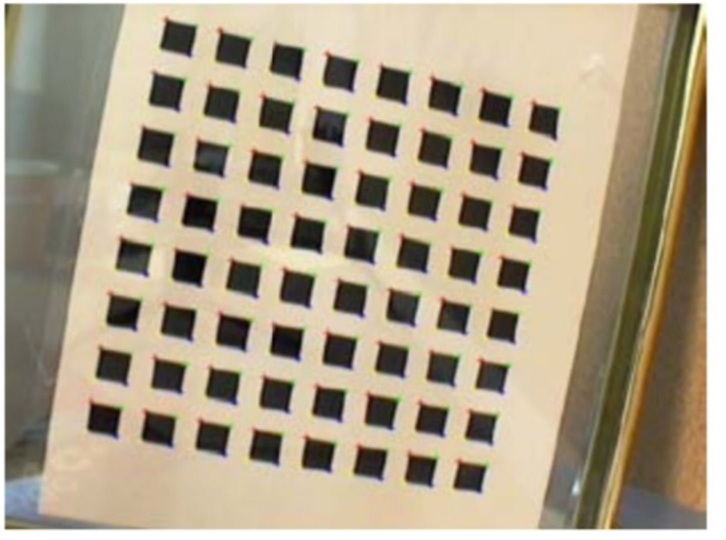


13. Image-based Rendering

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

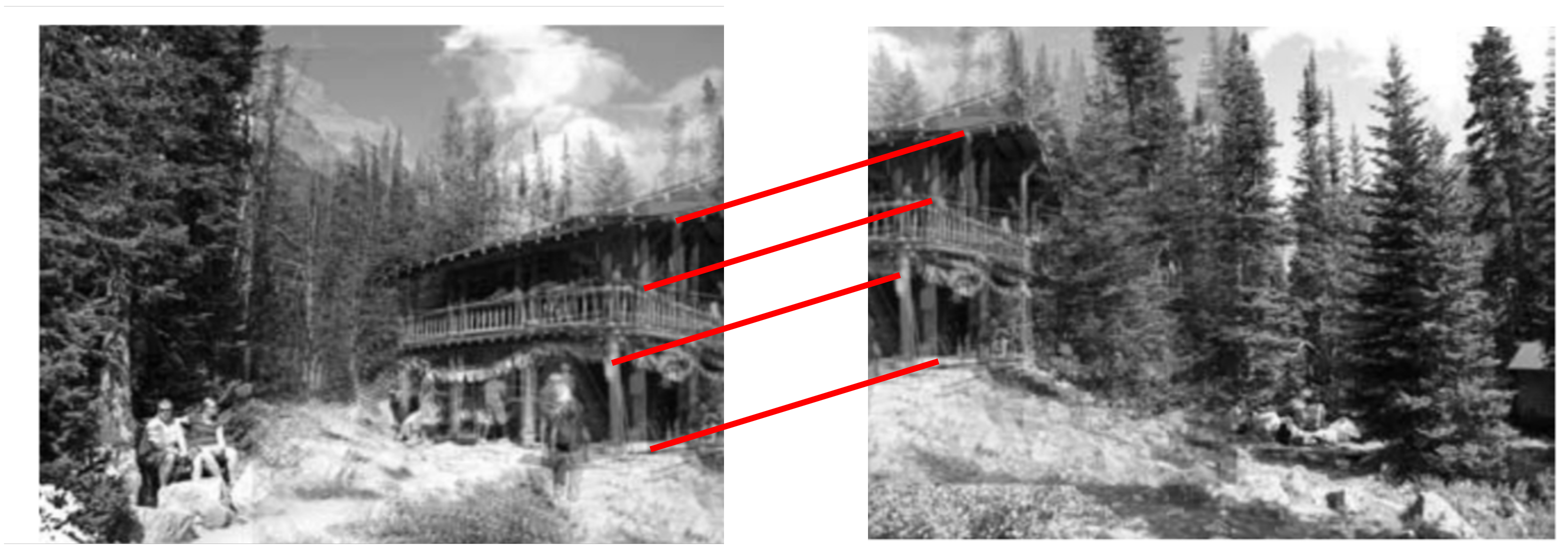
- Input:
  - Set of matches $\{(x_1, x_1'), (x_2, x_2'), (x_3, x_3'), (x_4, x_4')\}$
  - Parametric model: $f(x; t) = x + t$
  - Parameters p == t, location of origin of A in B
- Output:
  - Best model p*

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* ??

Image credit Szeliski book

2D translation estimation



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

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:

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

How to solve?

Jacobian

In many cases, parametric model is linear:

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

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

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

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

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

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

Linearize around a current guess p:

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

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

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

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

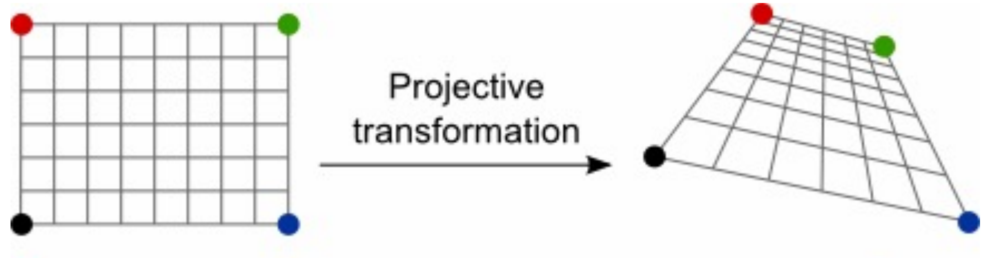


Image credit Graphics Mill
(educational Use)

- 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

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

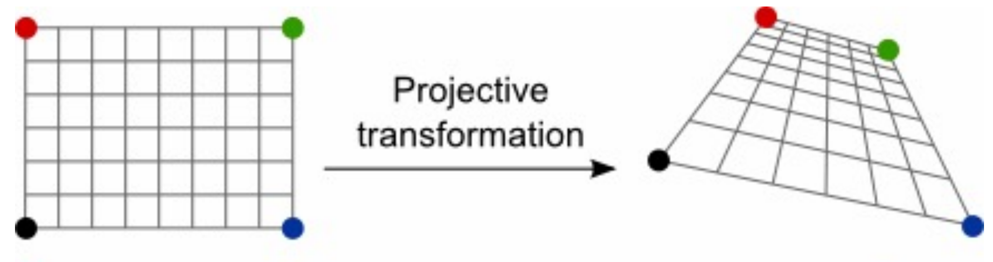


Image credit Graphics Mill

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

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

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

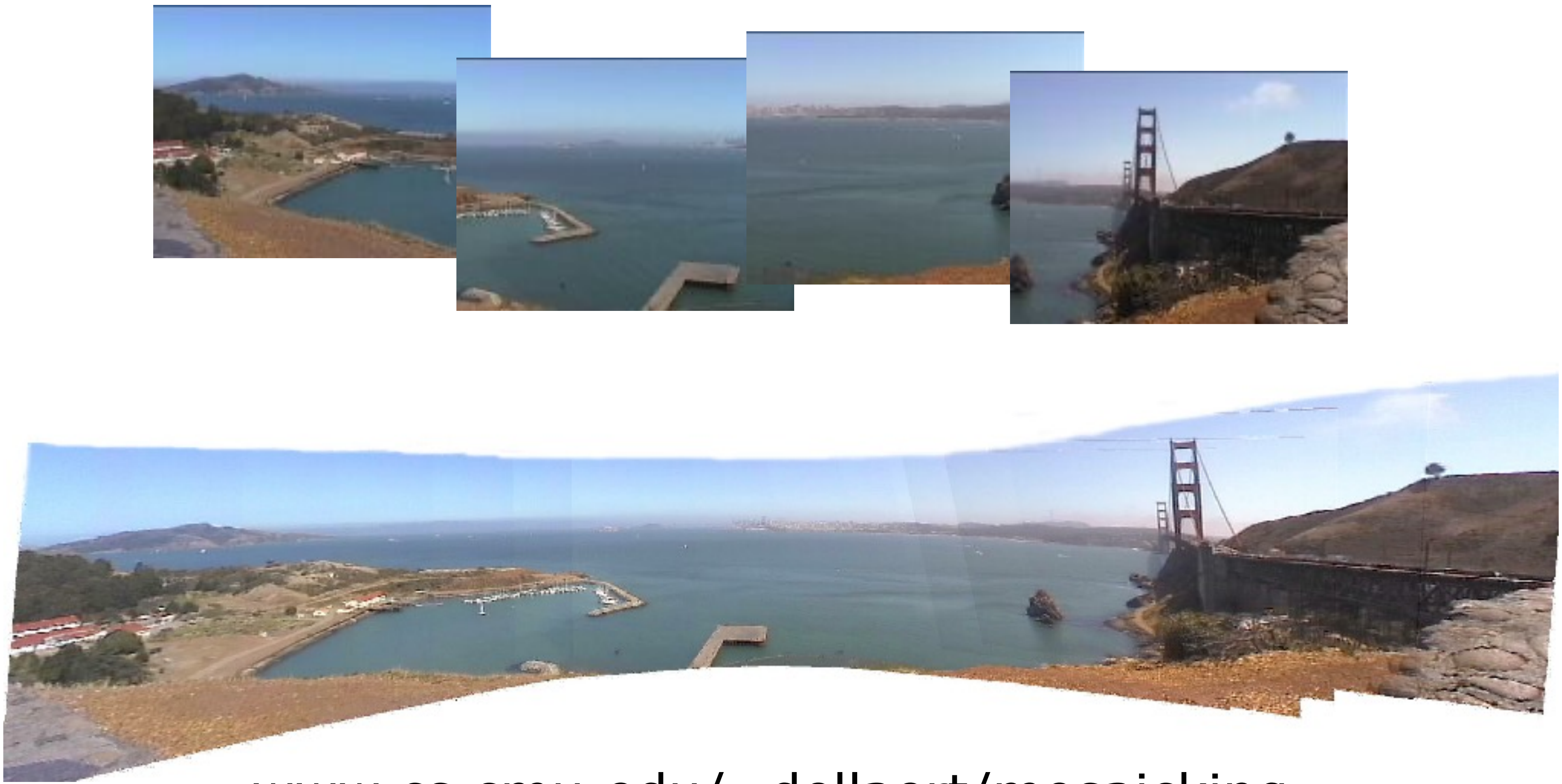
- 4 matches => system of 8 linear equations

RANSAC

Motivation

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

Mosaicking: Homography



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

Color photography avant-la-lettre

Prokudin-Gorskii Images

Color photographs from the Russian Empire taken a century ago (1909-1915).

1-500

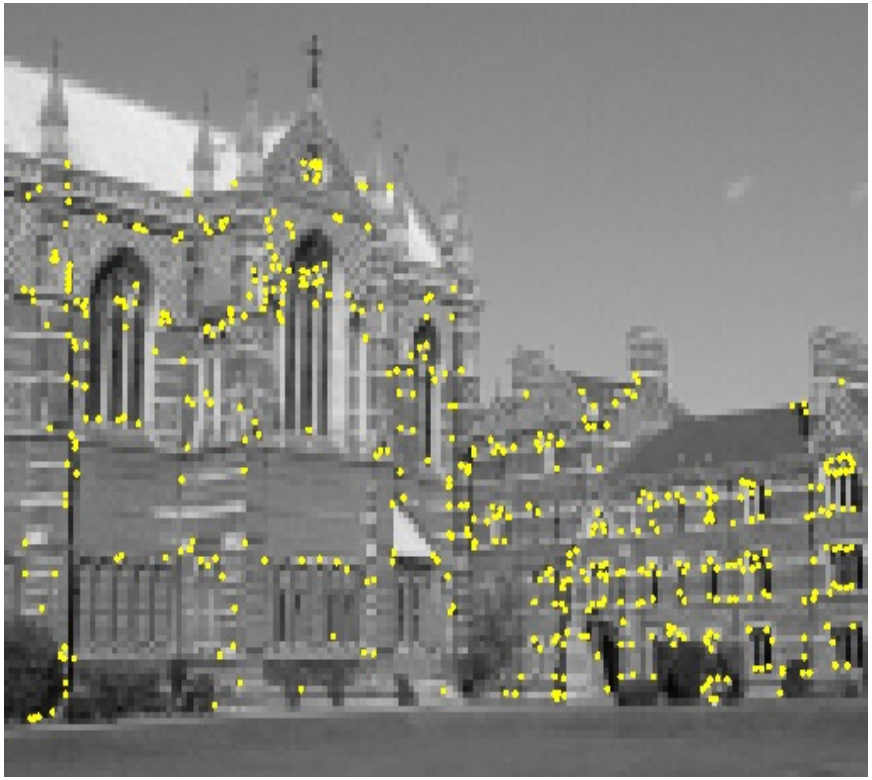
Sergei Mikhailovich Prokudin-Gorskii was a color photographer before his time, who undertook a photographic survey of the Russian Empire for Tsar Nicholas II. He was able to capture color by taking three pictures of each scene, each with a different red, green or blue color filter. Walter Frankhauser, a photographer contracted by the Library of Congress, manually registered and cleaned up some 120 of the original high-resolution scans, with breathtakingly beautiful results. The results of his effort can be seen at the online-exhibit The Empire That Was Russia.

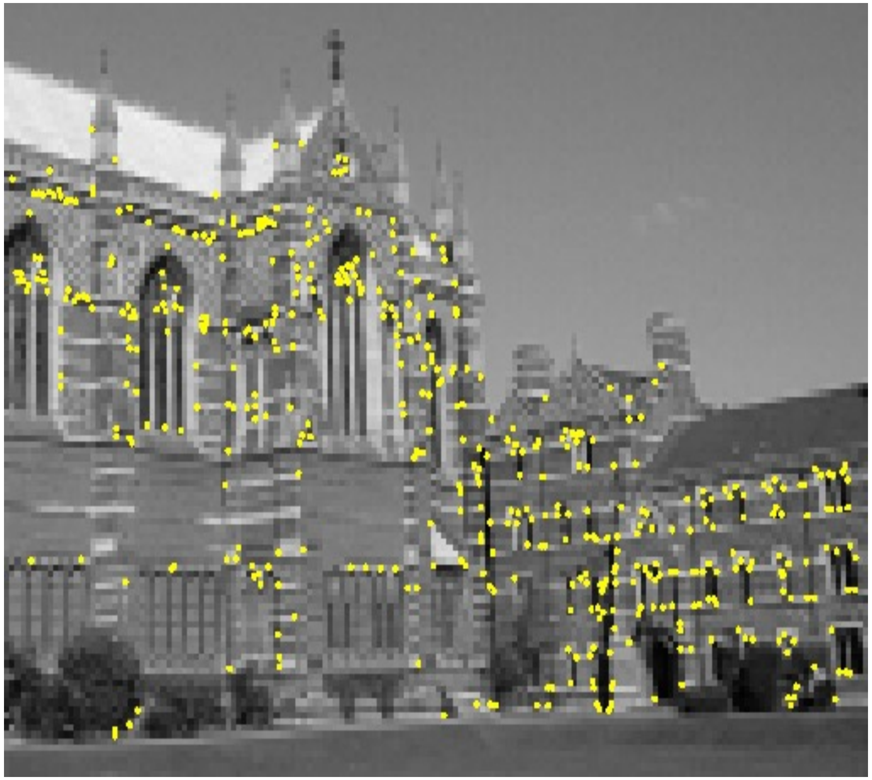
501-1000

Using computer-vision technology to automate the registration process, one can now for the first time view almost the entire collection of the Prokudin-Gorskii photographs in color. The color images on these pages were obtained by automatically registering these three pictures to obtain a color image of each scene. The computer program to do this was written in MATLAB by Frank Dellaert using computer-vision technology commonly used in 'mosaicking'. By clicking the links on this page you can view thumbnails of the almost 2000 images purchased by the Library of Congress, and each thumbnail is linked to a larger version of the corresponding color image.

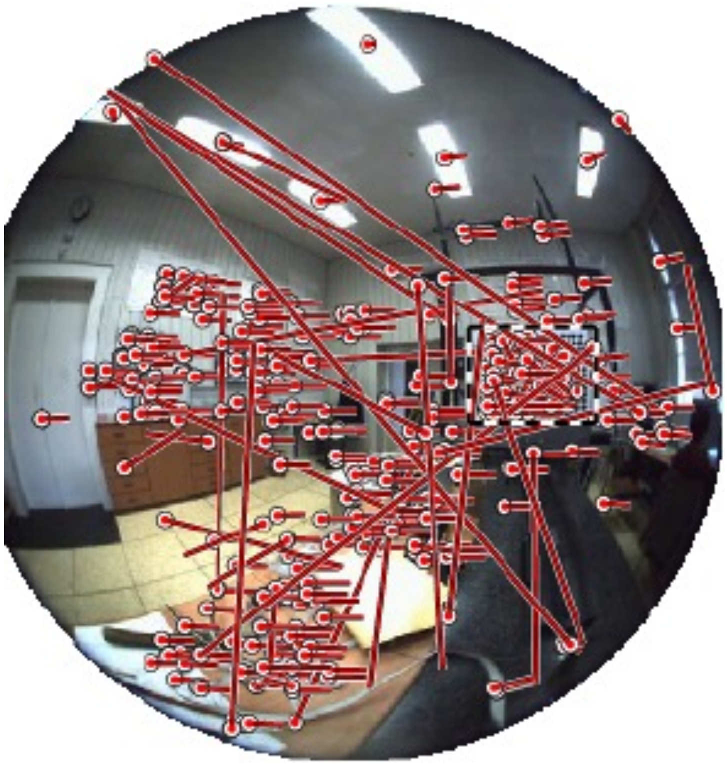
https://www.cs.cmu.edu/~dellaert/aligned/

Two-view geometry (next lecture)





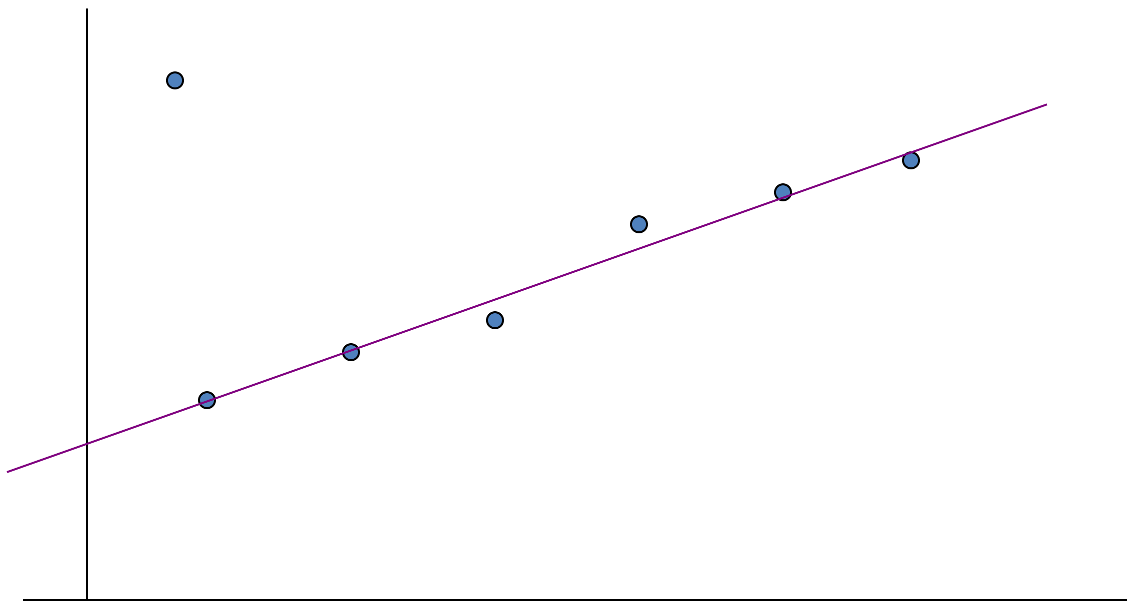
Omnidirectional example



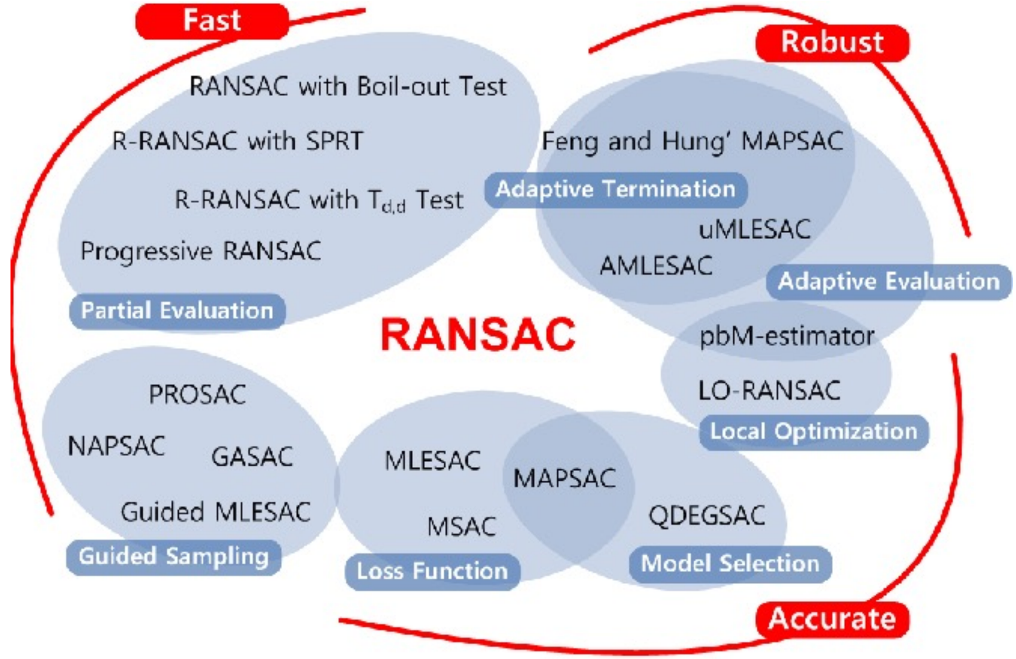
Images by Branislav Micusik, Tomas Pajdla,
cmp.felk.cvut.cz/ demos/Fishepip/

Simpler Example

- Fitting a straight line

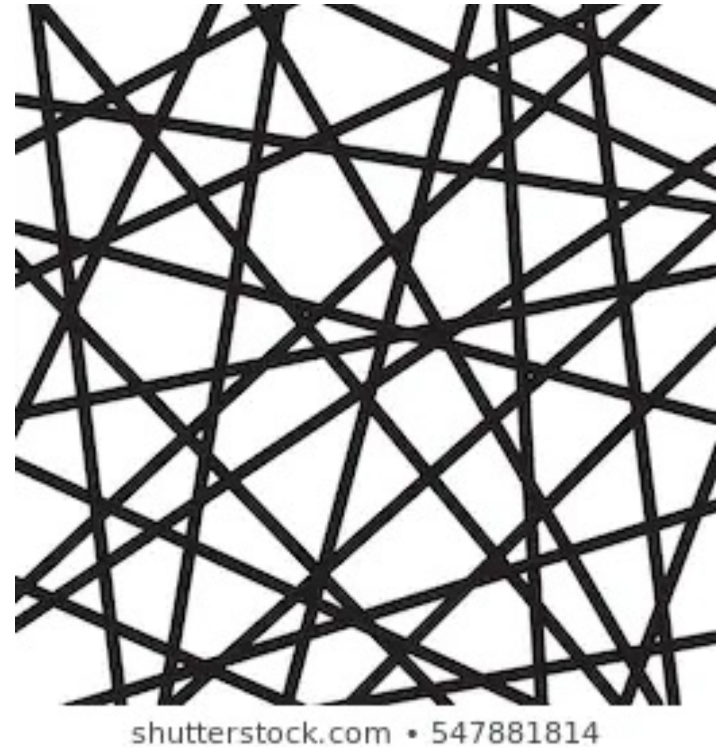


Discard Outliers



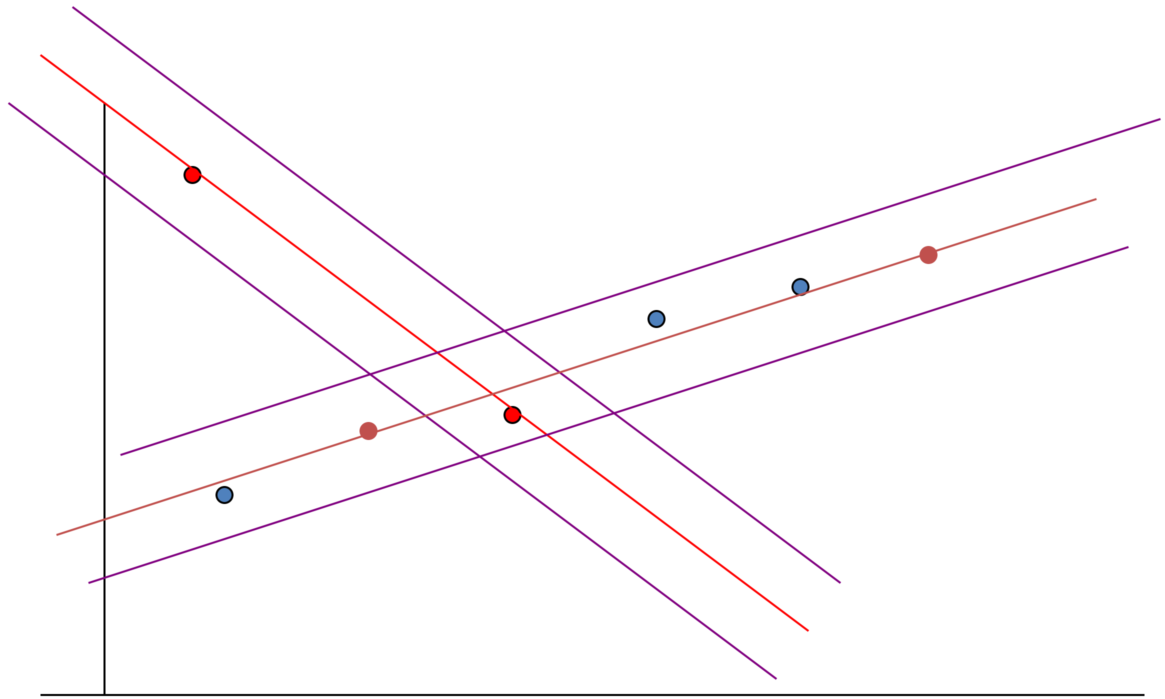
- No point with d>t
- RANSAC:
  - RANdom SAmple Consensus
  - Fischler & Bolles 1981
  - Copes with a large proportion of outliers

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

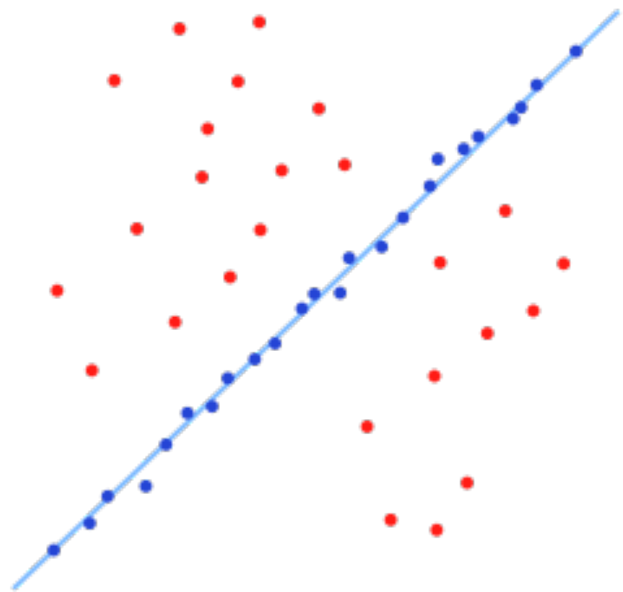
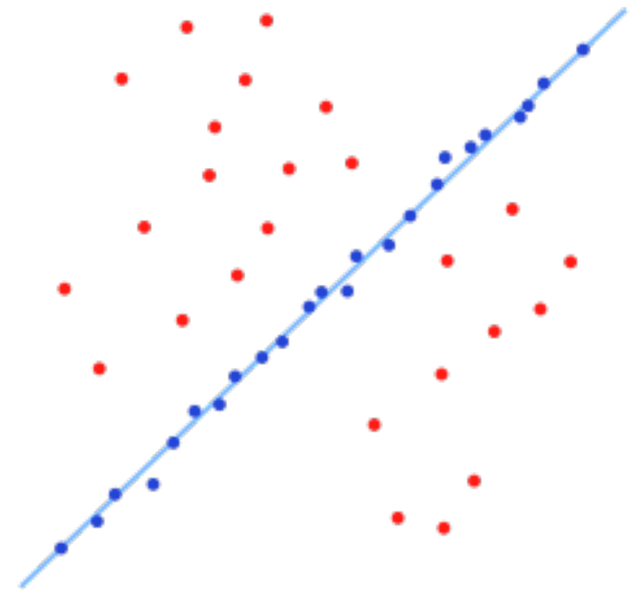


Image credit Wikipedia

RANSAC



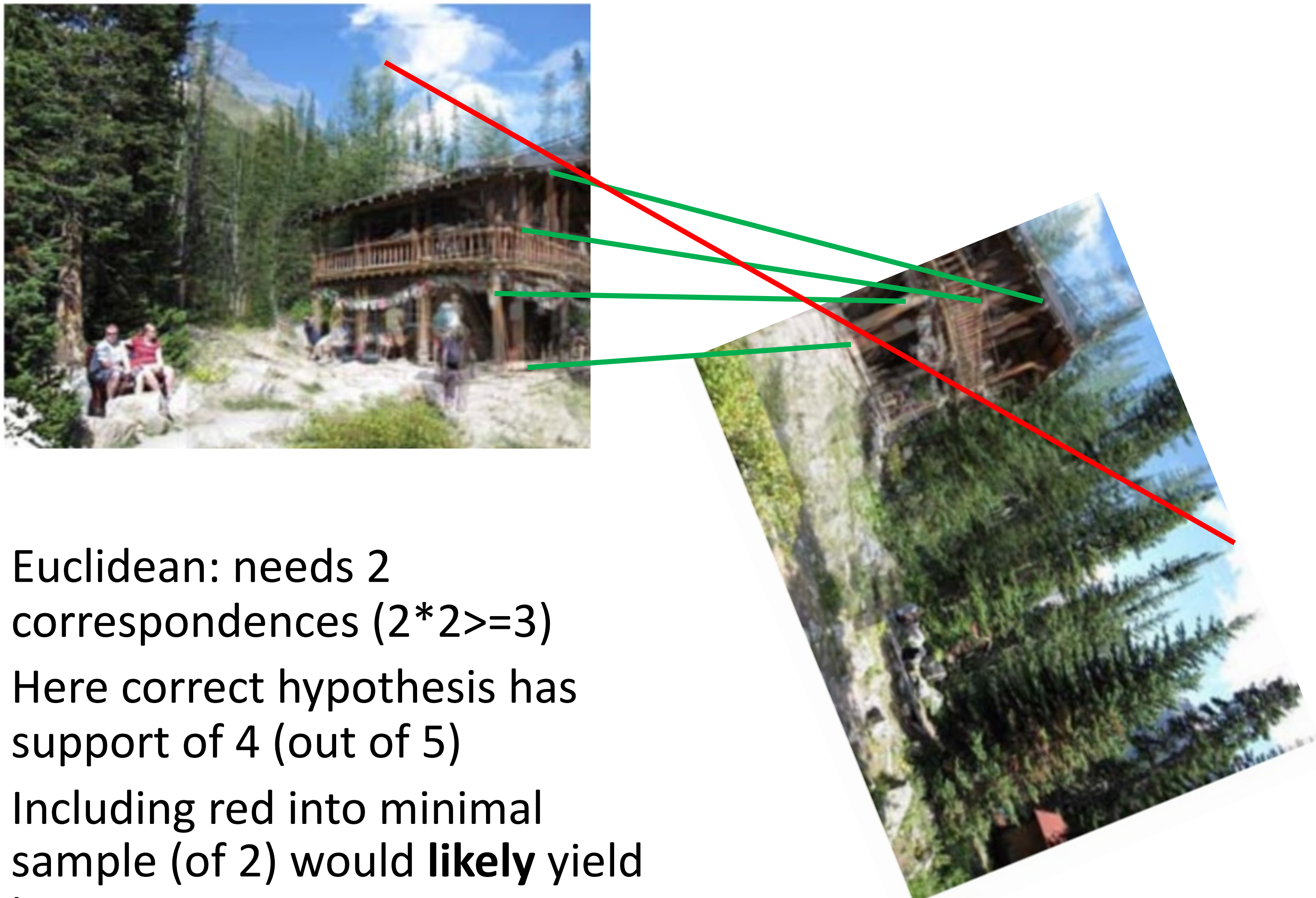
- Objective:
  - Robust fit of a model to data D
- Algorithm
  - Randomly select s points
  - Instantiate a model
  - Get consensus set D_i
  - If $|D_i|>T$, terminate and return model
  - Repeat for N trials, return model with max $|D_i|$

Image credit Wikipedia

In General

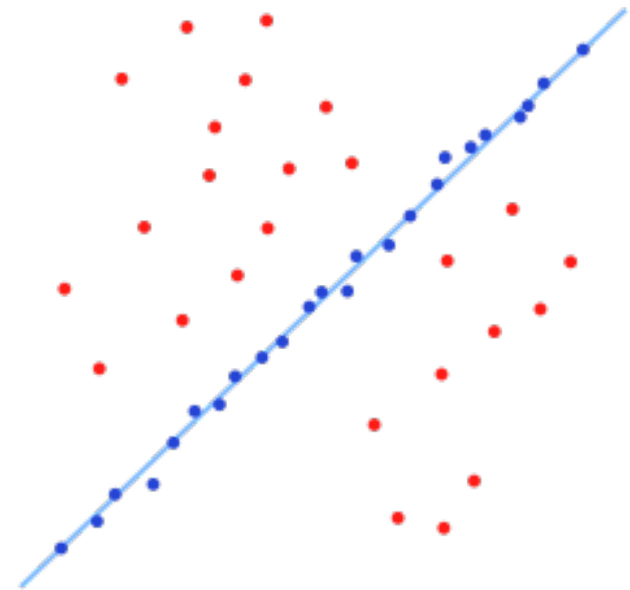
- Fit a more general model
- Sample = minimal subset
  - Translation ?
  - Homography ?
  - Euclidean transorm ?

Example



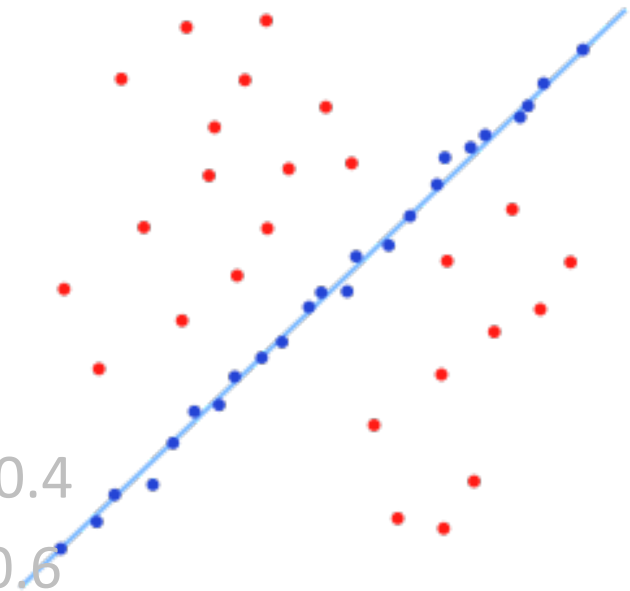
- Euclidean: needs 2 correspondences (2*2>=3)
- Here correct hypothesis has support of 4 (out of 5)
- Including red into minimal sample (of 2) would **likely** yield low support

How many samples ?



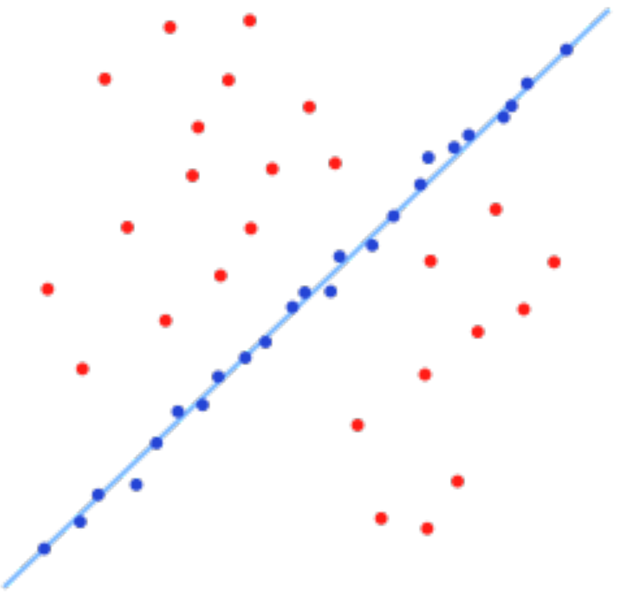
- We want: at least one sample with **all inliers**
- Can't guarantee: probability P
- E.g. P = 0.99

Calculate N



- If ε = outlier probability — $\varepsilon = 0.4$
- proportion of inliers $p = 1 - \varepsilon$ — $p=0.6$
- P(sample with all inliers) = p^s — $s=2 \rightarrow p^s=0.36$
- P(sample with an outlier) = $1-p^s$ — 0.64
- P(N samples an outlier) = $(1-p^s)^N$ — $N=3 \rightarrow 0.26$
- We want P(N samples an outlier) < 1-P (e.g. 0.01)
- $(1-p^s)^N < 1-P$ — $0.64^N < 0.01$
- $N > \log(1-P)/\log(1-p^s)$ — $N > 10.3$

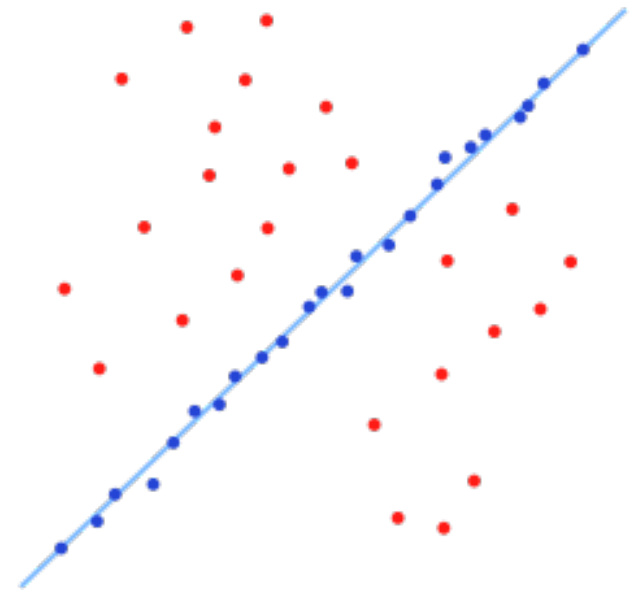
Example



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

Image credit Wikipedia

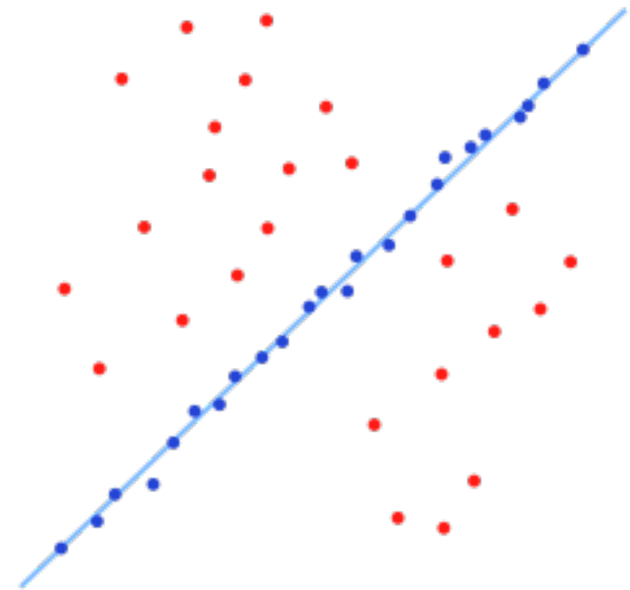
Remarks



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

Image credit Wikipedia

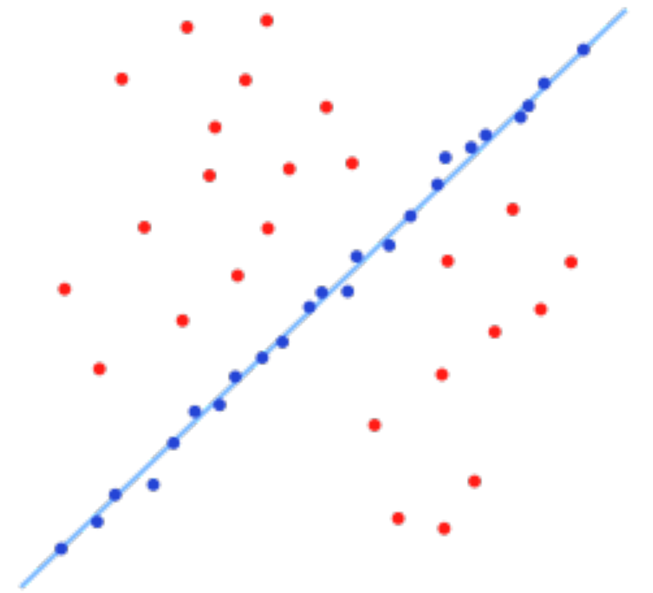
Distance Threshold



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

Image credit Wikipedia

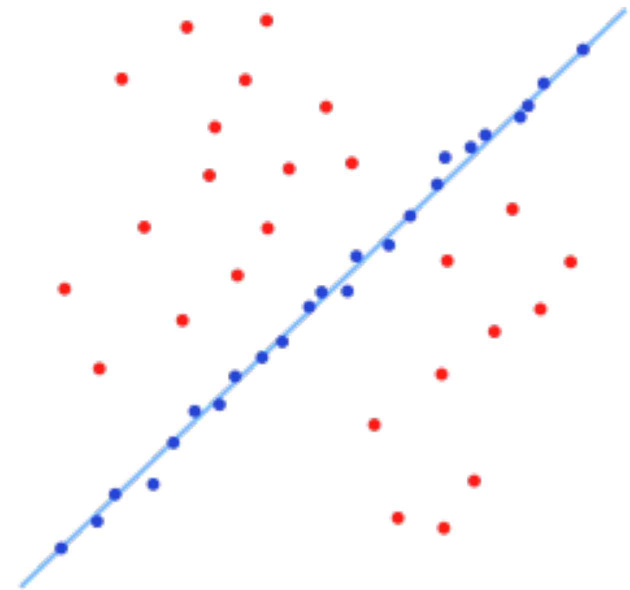
Threshold T



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

Image credit Wikipedia

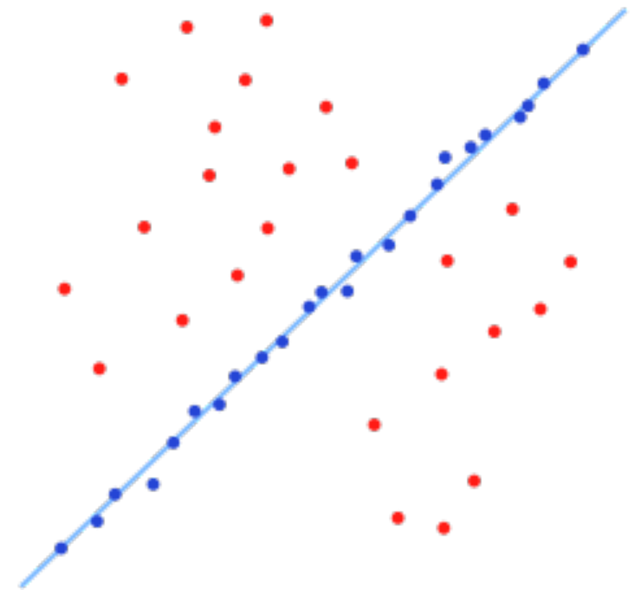
Adaptive N



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

Image credit Wikipedia

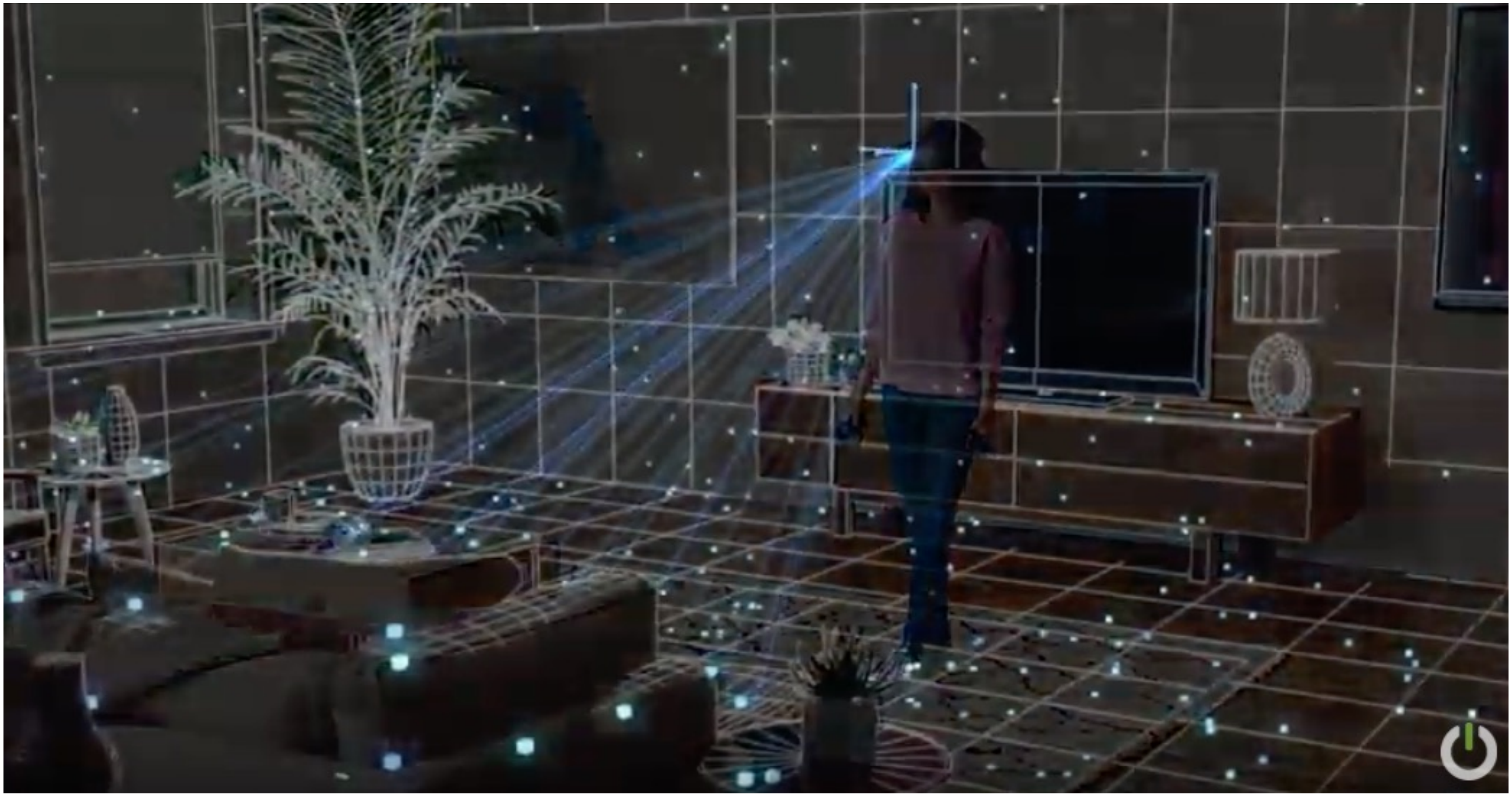
Summary: RANSAC



- 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

Pose Estimation in VR



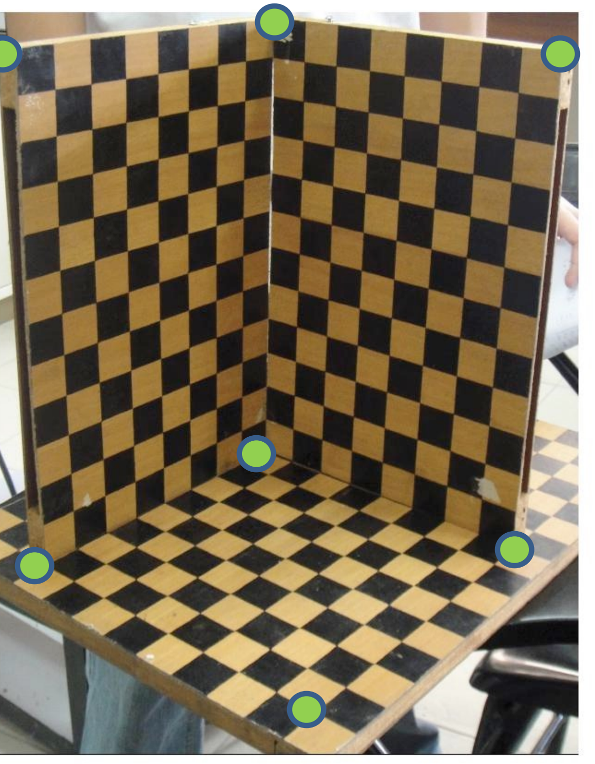
https://youtu.be/nrj3JE-NHMw

Review: 2D Alignment

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

Image credit Szeliski book

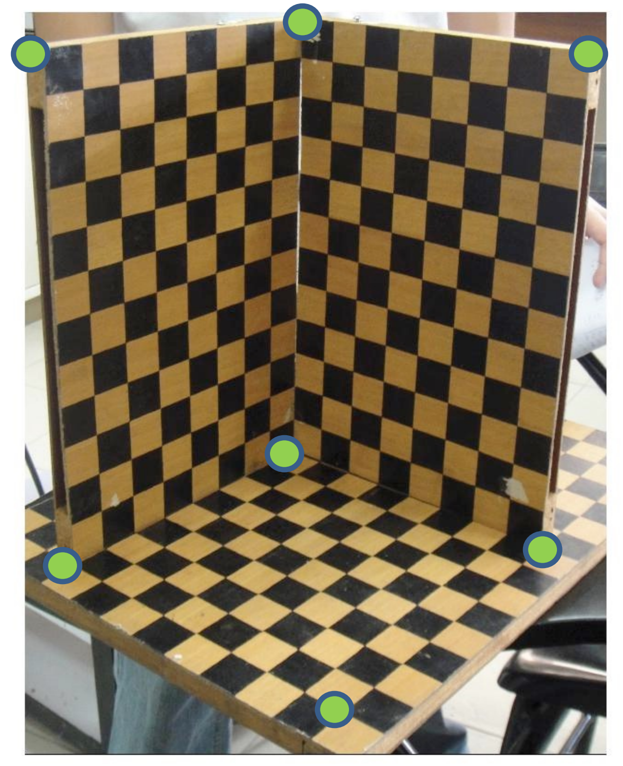
Now: 3D-2D Alignment



- Input:
  - A set of 3D->2D matches $\{(X_i, x_i)\}$
  - A parametric model $f(X; p)$
- Output:
  - Best model p^*
- How?

Image credit Szeliski book

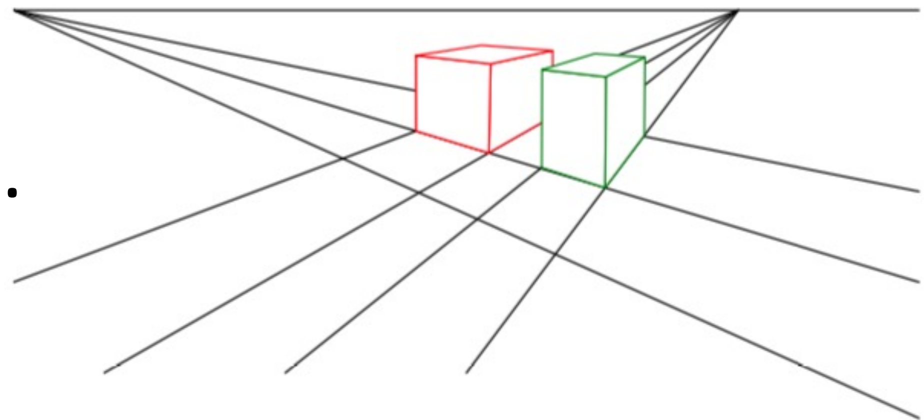
Pose Estimation



- Input:
  - A set of 2D measurements x_i of known 3D points X_i
  - Parametric model is camera matrix P, i.e., $x = f(X; P)$
- Output:
  - Best camera matrix P
- How?

Review: Projective Camera Matrix

- Chapter 2 in book
- Homogeneous coord.
- 3D TO 2D projection:



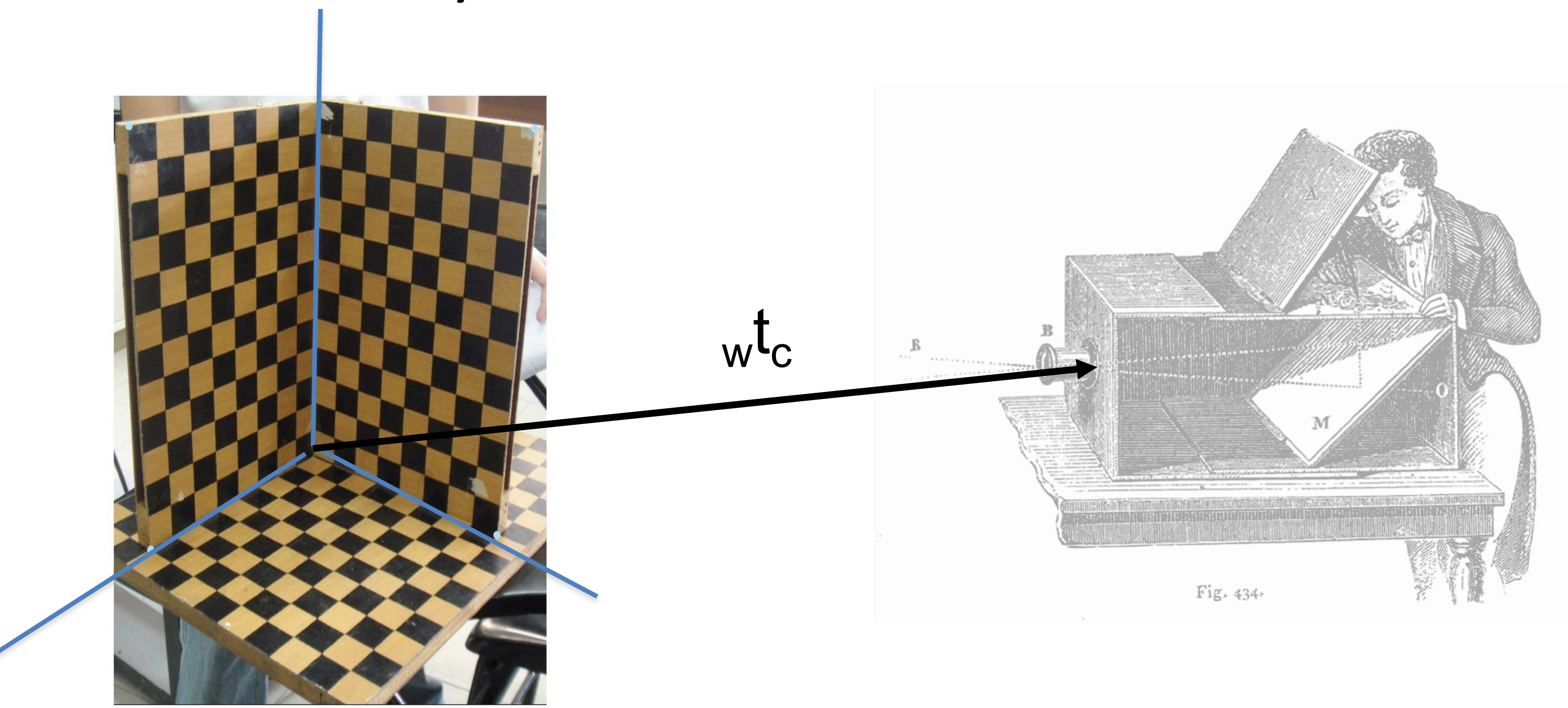
$$x = K[R|t]X = PX$$

where $P = 3\text{x}4$ camera matrix
and K the 3x3 calibration

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

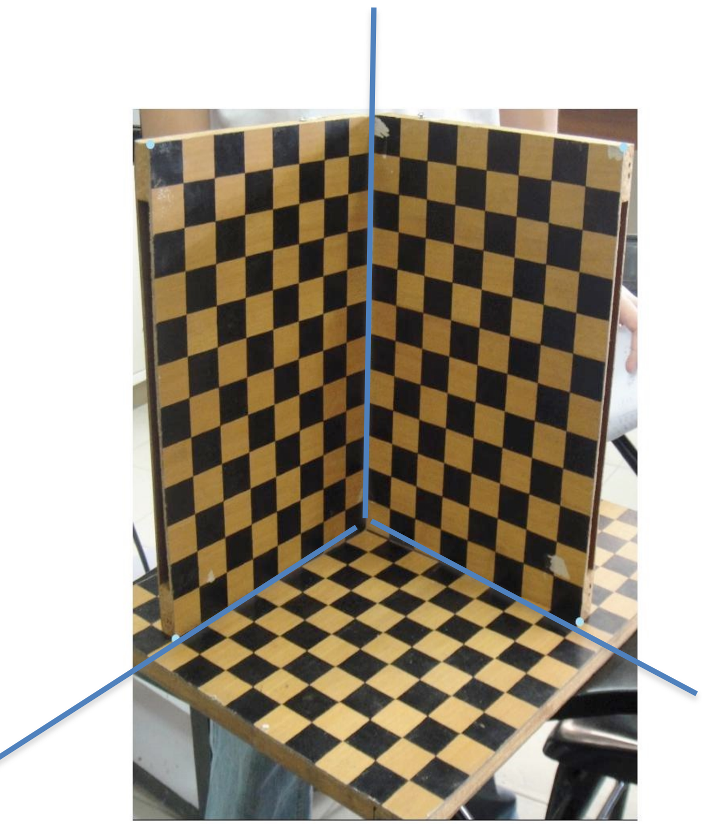
Camera Extrinsics: a Pose in 3D

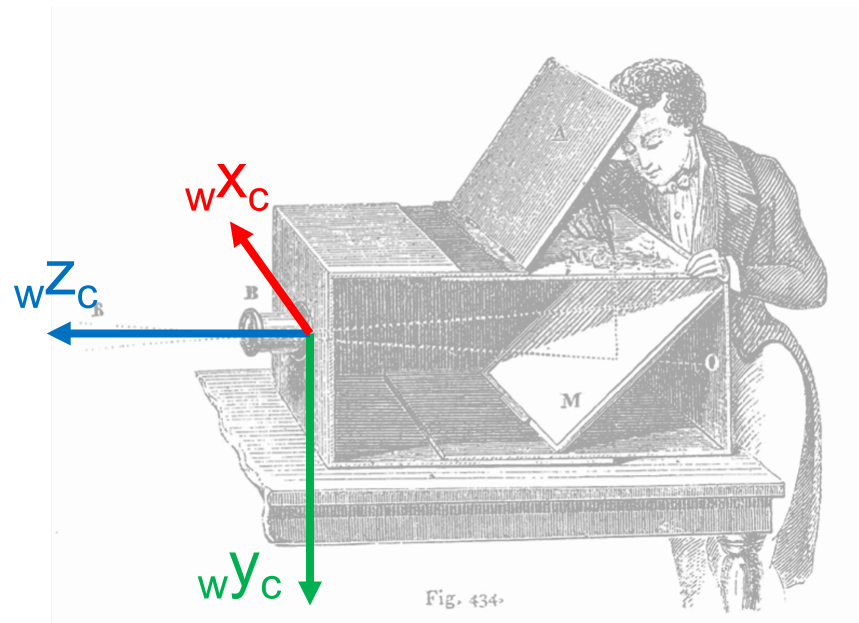
- What is the geometric meaning of R and t ??
- Intuitive: camera is at a position ${}_w t_c$
  Indices say: camera *in* world coordinate frame



Camera Extrinsics: a Pose in 3D

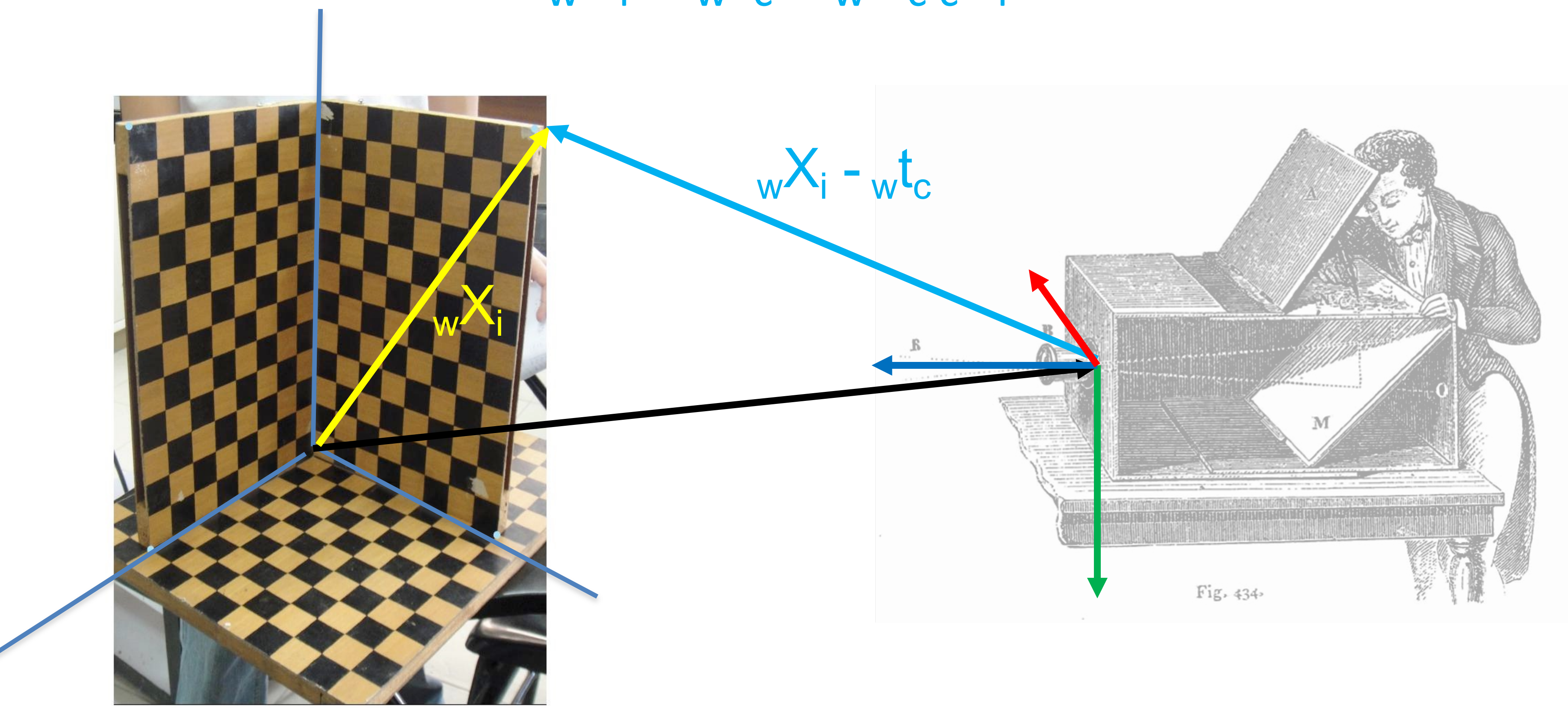
- What is the geometric meaning of R and t ??
- Rotation is given by 3x3 matrix ${}_wR_c$ whose *columns* are the camera axes ${}_wx_c$, ${}_wy_c$, ${}_wz_c$





Camera Extrinsics: a Pose in 3D

- What is the geometric meaning of R and t ??
- Transforming point X_i from world to camera coordinates: ${}_wX_i - {}_wt_c = {}_wR_c\,{}_cX_i$



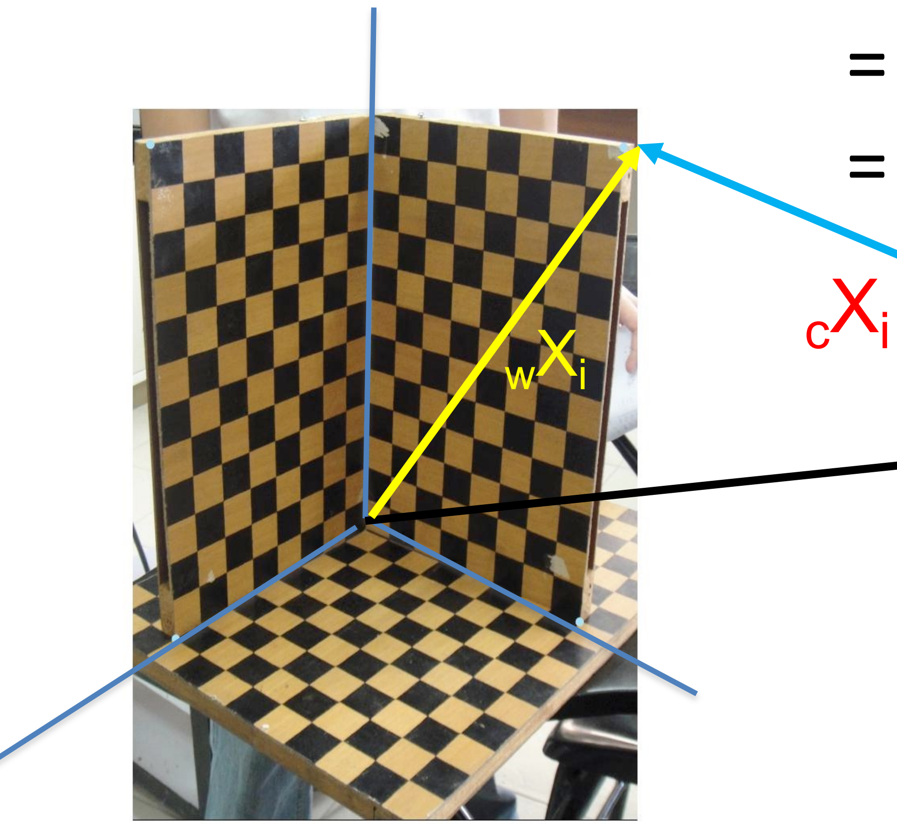
Camera Extrinsics: a Pose in 3D

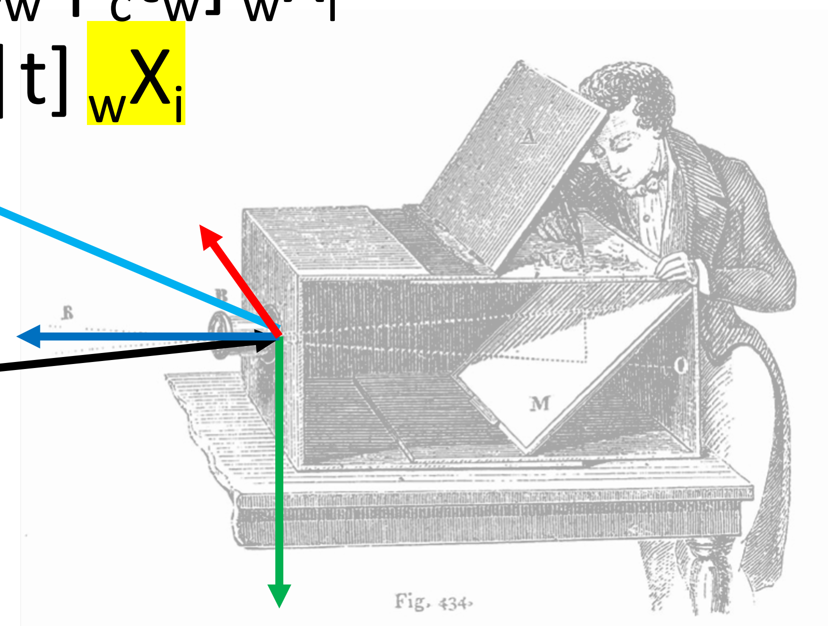
- Expressed in homogeneous coordinates:
- ${}_cX_i = {}_wR_c^T ({}_wX_i - {}_wt_c) = {}_wR_c^T [I| - {}_wt_c]\, {}_wX_i$

$$= [{}_wR_c^T \mid - {}_wR_c^T\, {}_wt_c]\, {}_wX_i$$

$$= [{}_cR_w \mid {}_ct_w]\, {}_wX_i$$

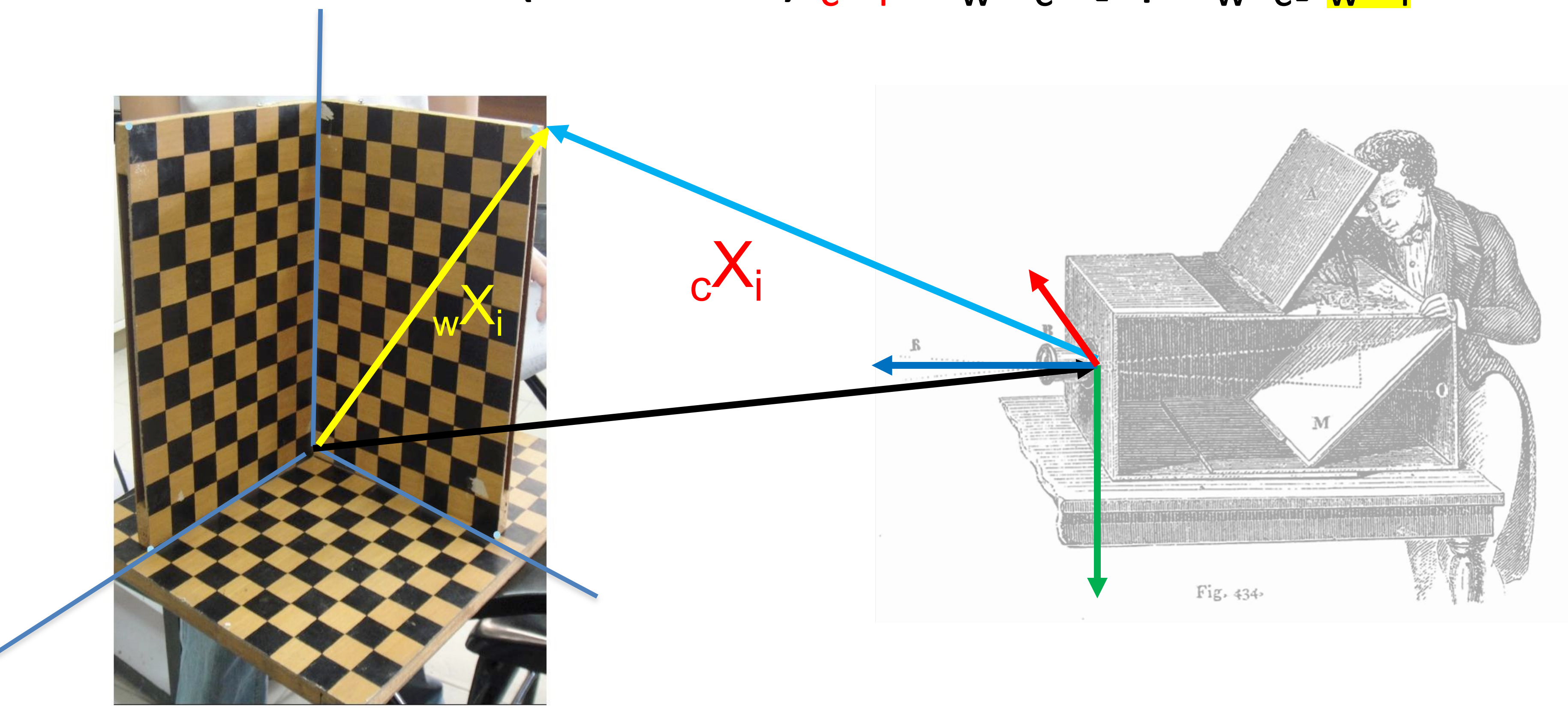
$$= [R|t]\, {}_wX_i$$





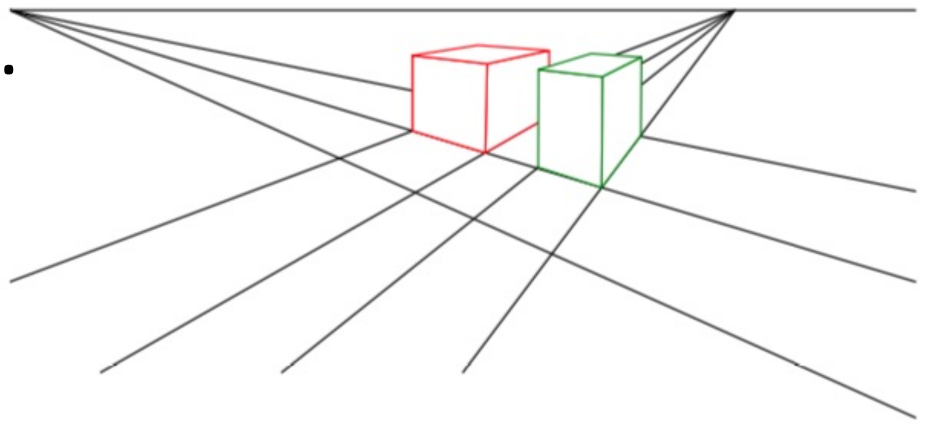
Camera Extrinsics: a Pose in 3D

- Conclusion: when people write ${}_{c}X_i = [R|t]\ {}_{w}X_i$ they are talking about (unintuitive) $[{}_{c}R_{w} \mid {}_{c}t_{w}]$
- We like use (intuitive) ${}_{c}X_i = {}_{w}R_{c}^{T}\,[I|-{}_{w}t_{c}]\ {}_{w}X_i$



Revision: Projective Camera Matrix

- Homogeneous coord.
- 3D TO 2D projection:



Camera-centric: $x = K[{}_cR_w \mid {}_ct_w]\, X = PX$

World-centric: $x = K\, {}_wR_c^T\, [I \mid -\, {}_wt_c]\, X = PX$

P = **same** 3x4 camera matrix
and K the 3x3 calibration $\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

Looking at the (opaque) camera matrix

Can you interpret the columns of P with entities in the scene?

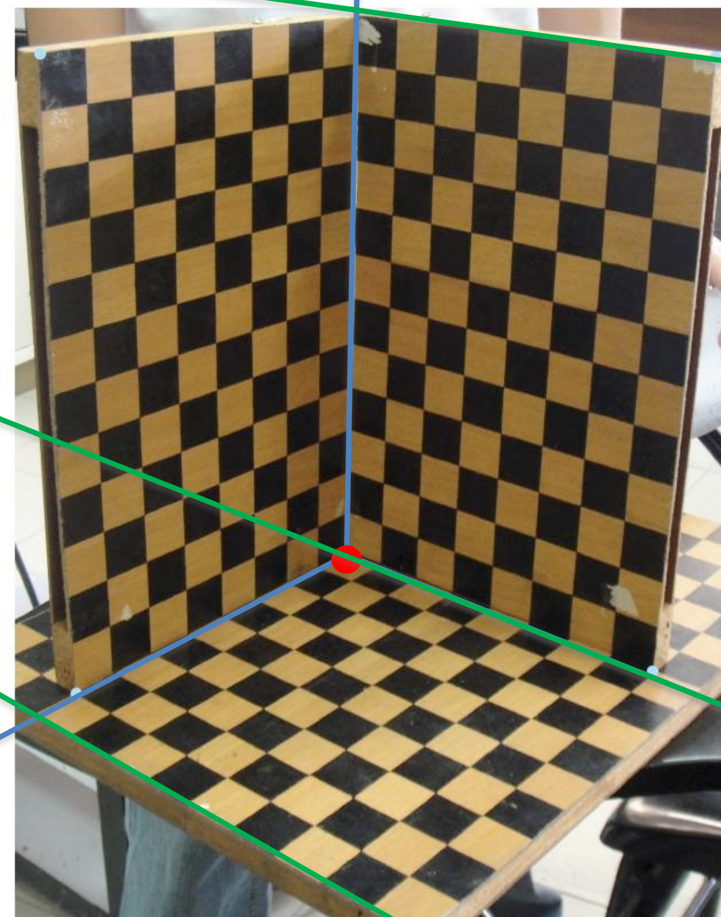
$$P = \begin{bmatrix} P^1 & P^2 & P^3 & P^4 \end{bmatrix}$$

Answer:
P^1 == the image of [1 0 0 0]
P^2 == the image of [0 1 0 0]
P^3 == the image of [0 0 1 0]
P^4 == the image of [0 0 0 1]

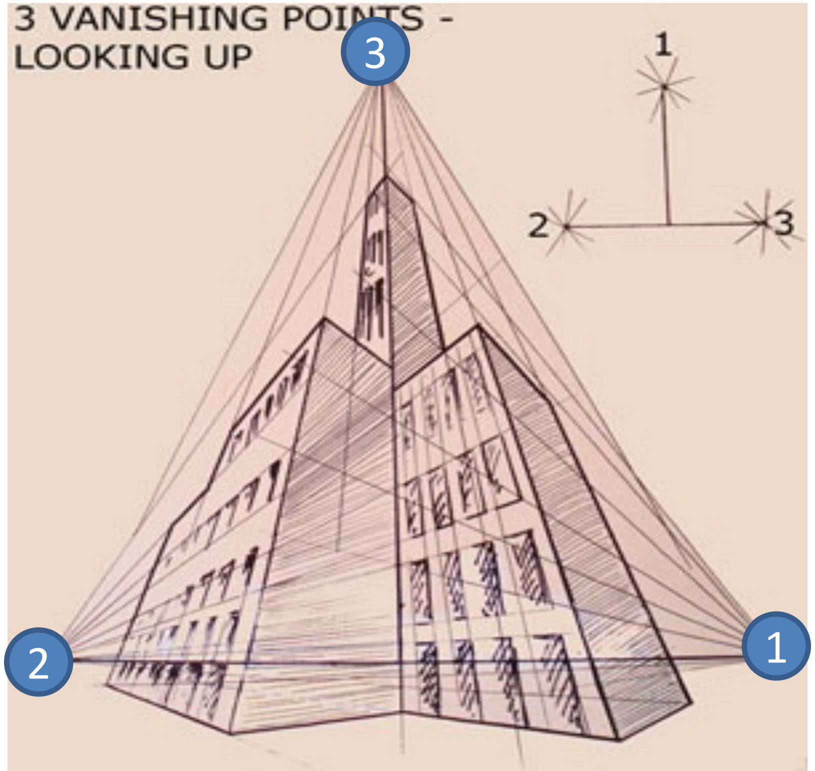
What are those ?
[0 0 0 1] is easy…

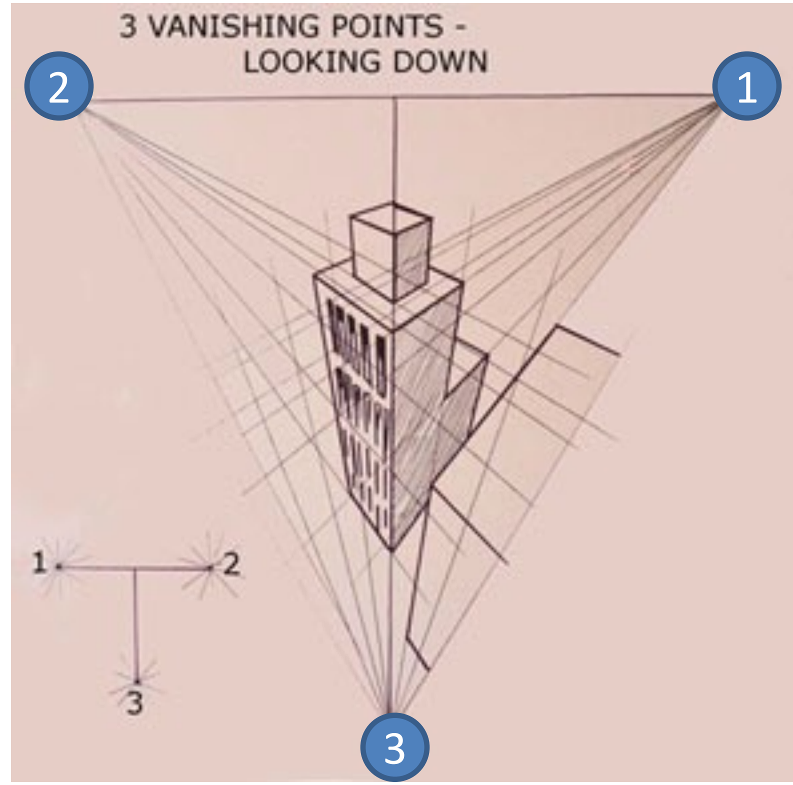
Answer:
[0 0 0 1] is the origin, so P4 is the image of the origin.

[0 1 0 0] is a point at infinity in the X-direction, so it is the vanishing point of all lines parallel with the X direction!



Vanishing points, revisited





Columns of P !

$$P = \begin{bmatrix} P^1 & P^2 & P^3 & P^4 \end{bmatrix}$$

P^4 is arbitrary: wherever you defined the world origin.

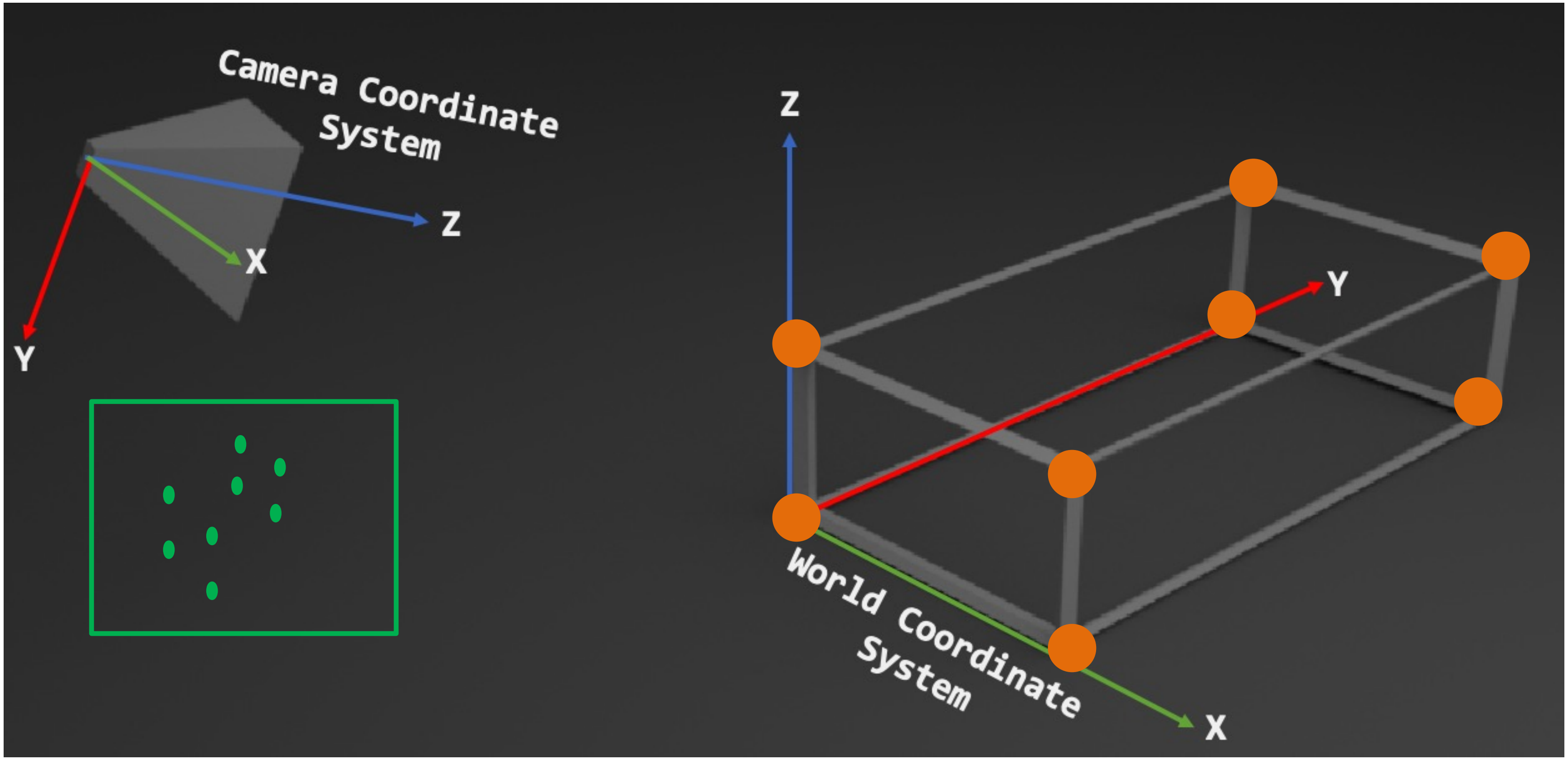
https://www.artinstructionblog.com/perspective-drawing-tutorial-for-artists-part-2

Back to Pose Estimation!

- Simple algorithm: just measure the coordinates of the origin and the three vanishing points?

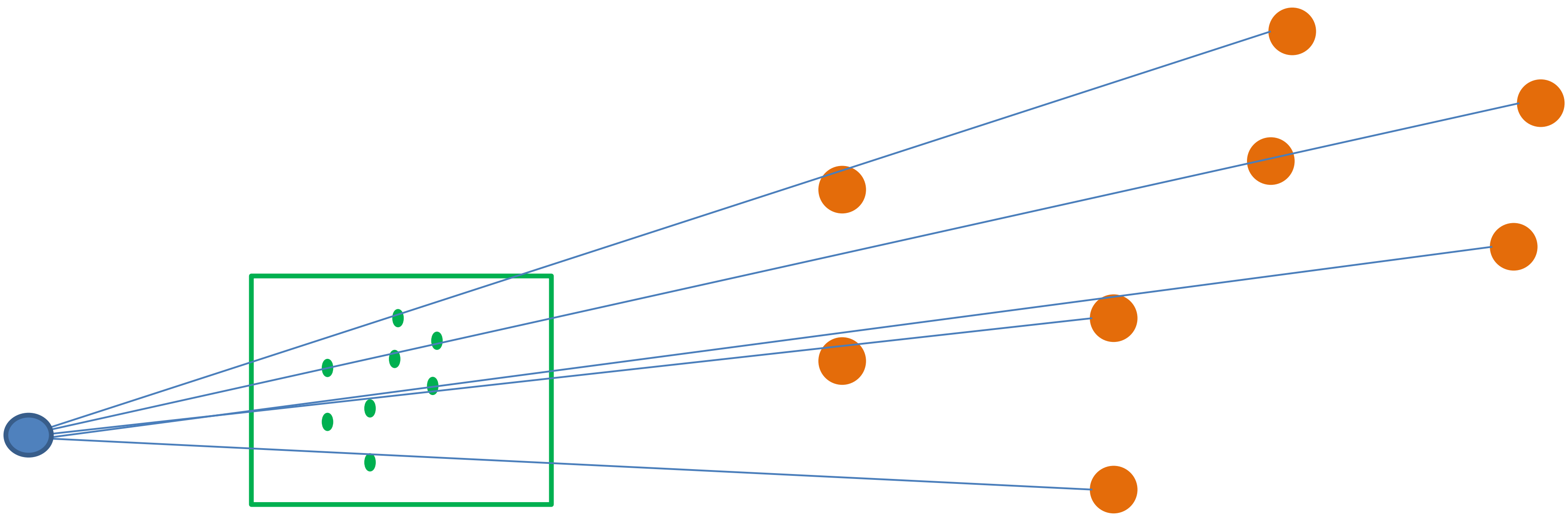
- Does not work ☹:
  - Columns are only measured up to a scale.
  - 4 points * 2DOF = only 8 DOF! Missing 11-8=3
  - 3 missing numbers are exactly those scales.

Least Squares Pose Estimation...



- Input:
  - A set of 2D measurements x_i of known 3D points X_i
  - Parametric model is camera matrix P, i.e., $x = f(X; P)$
- Output:
  - Best camera matrix P

Pose estimation = "Resectioning"



$$\mathbf{x} = f(\mathbf{X}_w; \mathbf{P}) = \mathbf{P}\mathbf{X}_w = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cong \begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\arg\min_{\hat{\mathbf{P}}} \sum_{i=1}^{N} ||\hat{\mathbf{P}}\mathbf{X}_w^i - \mathbf{x}^i||_2.$$

- Opposite of triangulation.

Pose estimation

$$\arg\min_{\hat{\mathbf{P}}} \sum_{i=1}^{N} ||\hat{\mathbf{P}}\mathbf{X}_w^i - \mathbf{x}^i||_2.$$

- In project 4, you will use `scipy.optimize.least_squares` to do exactly that. Working knowledge of 3D poses will be required.
- Note before we compute the 2D reprojection error we need to convert back *PX* to non-homogeneous coordinates:

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html