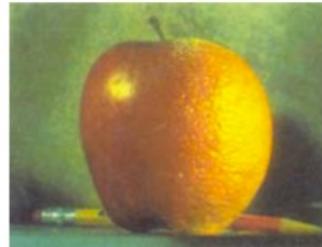


2. Image Formation



3. Image Processing



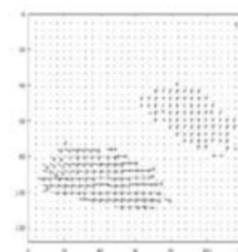
4. Features



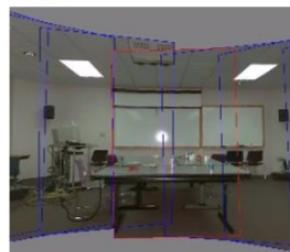
5. Segmentation



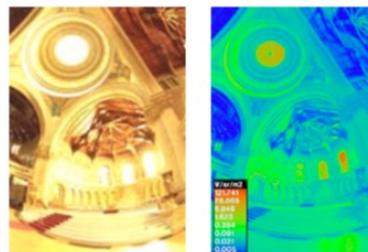
6-7. Structure from Motion



8. Motion



9. Stitching



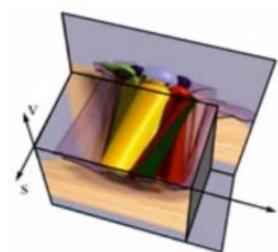
10. Computational Photography



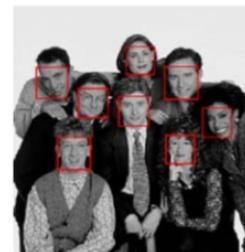
11. Stereo



12. 3D Shape

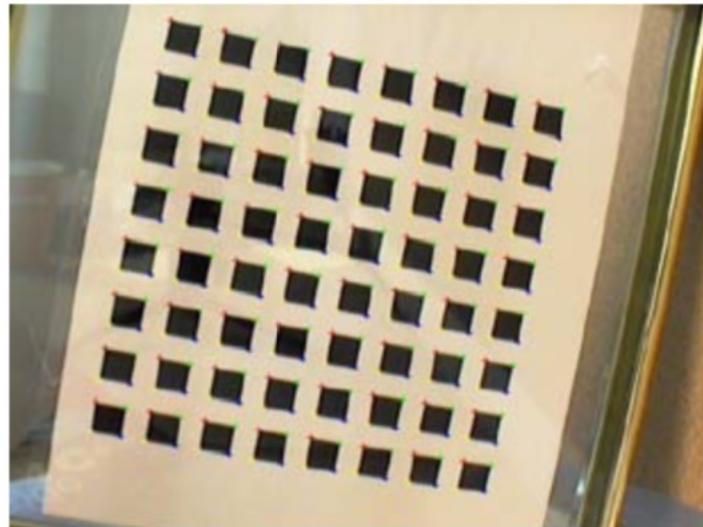
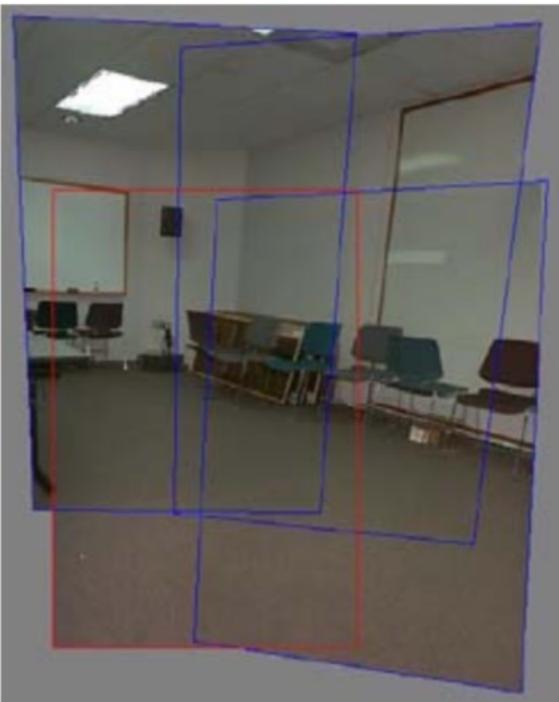


13. Image-based Rendering



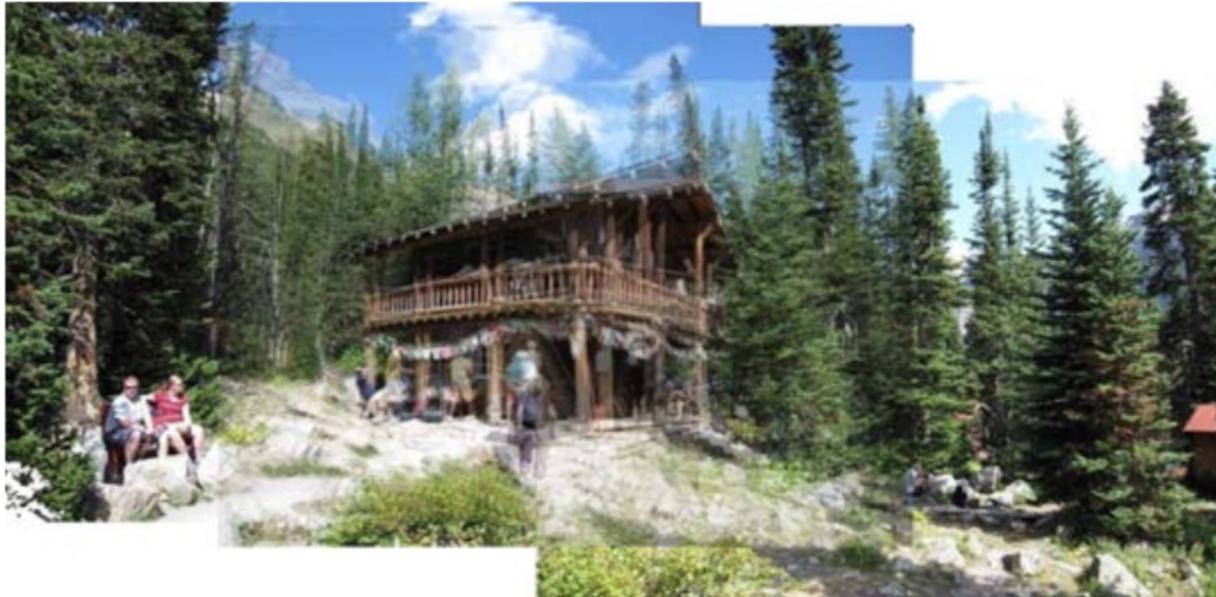
14. Recognition

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



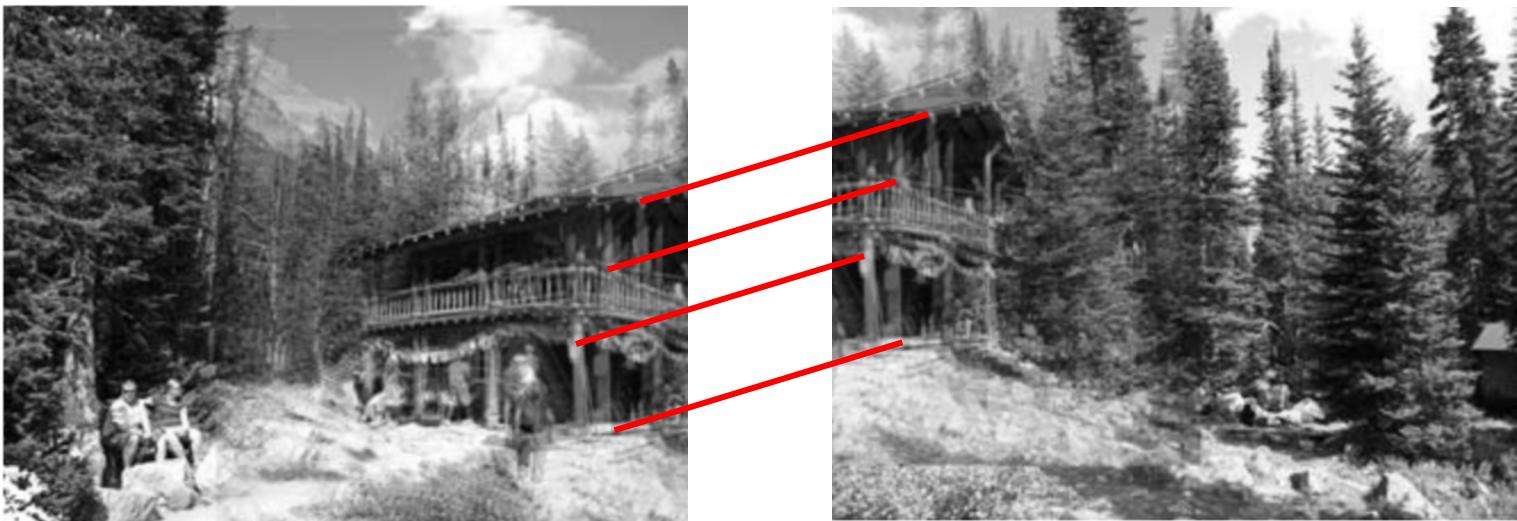
- Input:
 - Set of matches $\{(x_1, x'_1), (x_2, x'_2), (x_3, x'_3), (x_4, x'_4)\}$
 - Parametric model: $f(x; t) = x + t$
 - Parameters $p == t$, location of origin of A in B
- Output:
 - Best model p^*

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^* ??

2D translation estimation



- How?
 - One correspondence $x_1 = [600, 150]$, $x_1' = [50, 50]$
 - Parametric model: $x' = f(x; t) = x + t$

2D translation estimation

$[-550, -100]$



- How?
 - One correspondence $x_1 = [600, 150]$, $x_1' = [50, 50]$
 - Parametric model: $x' = f(x; t) = x + t$
 $\Rightarrow t = x' - x$
 - $\Rightarrow t = [50-600, 40-150] = [-550, -100]$

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:**

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

How to solve?

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:

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

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

Hessian $\xrightarrow{\hspace{1cm}}$ $Ap = b$
 $p^* = A^{-1}b$

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:

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

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

$$A\Delta p = b$$

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

Projective/H

- Jacobians a bit harder
- Parameterization:

$$\begin{bmatrix} 1 + h_{00} & h_{01} & h_{02} \\ h_{10} & 1 + h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$$

Image credit Graphics Mill
(educational Use)

$$(h_{00}, h_{01}, \dots, h_{21})$$

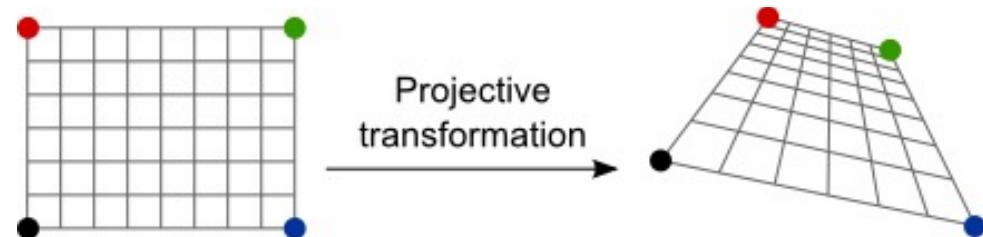
- $\mathbf{x}' = f(\mathbf{x}, \mathbf{p})$:

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

- And Jacobian:

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

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



Closed Form H

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

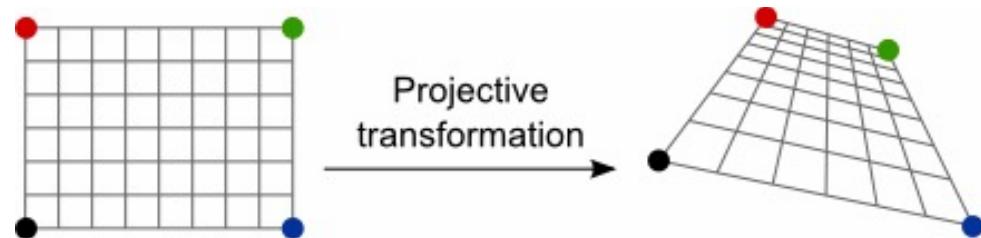


Image credit Graphics Mill

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

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

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

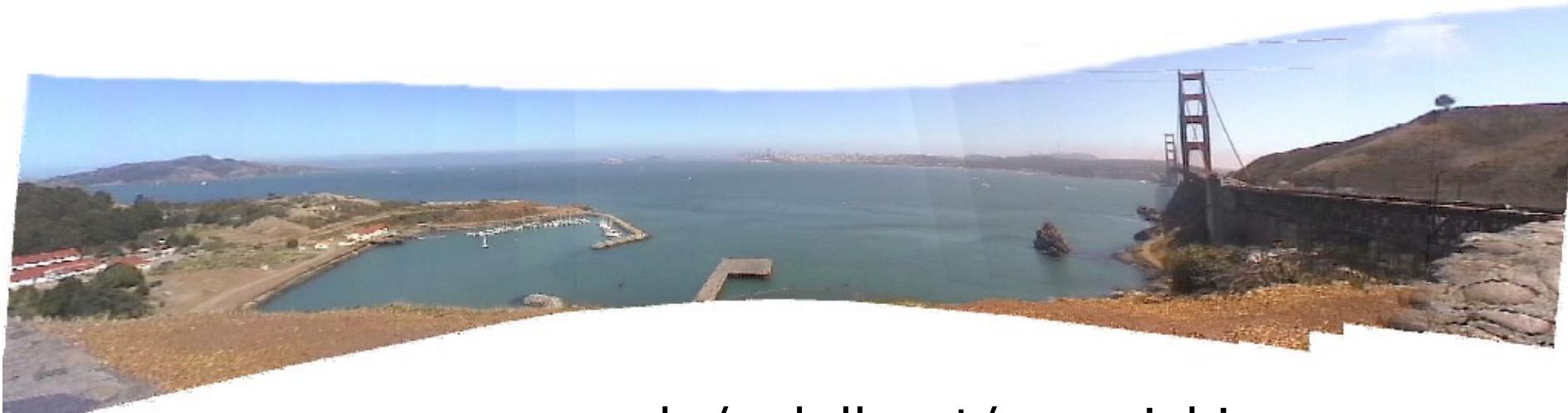
- 4 matches \Rightarrow system of 8 linear equations

RANSAC

Motivation

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

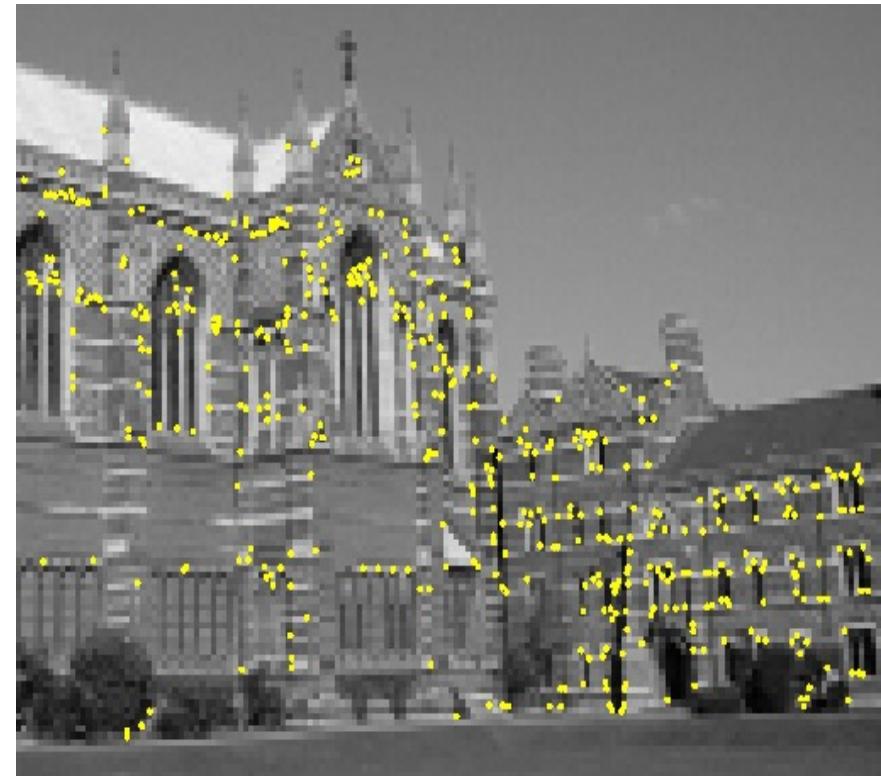
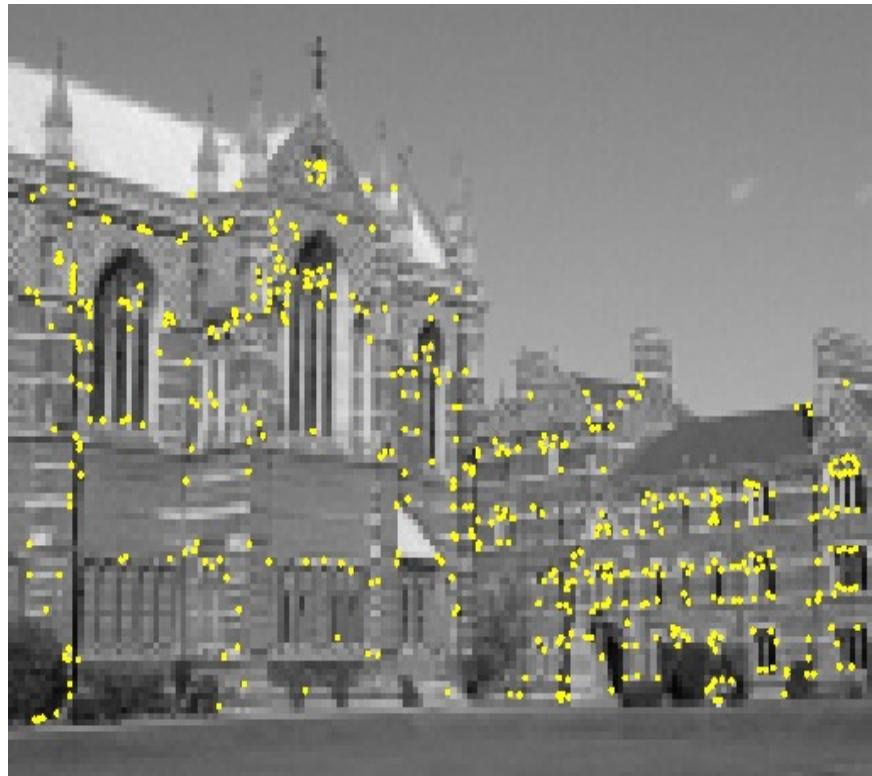
Mosaicking: Homography



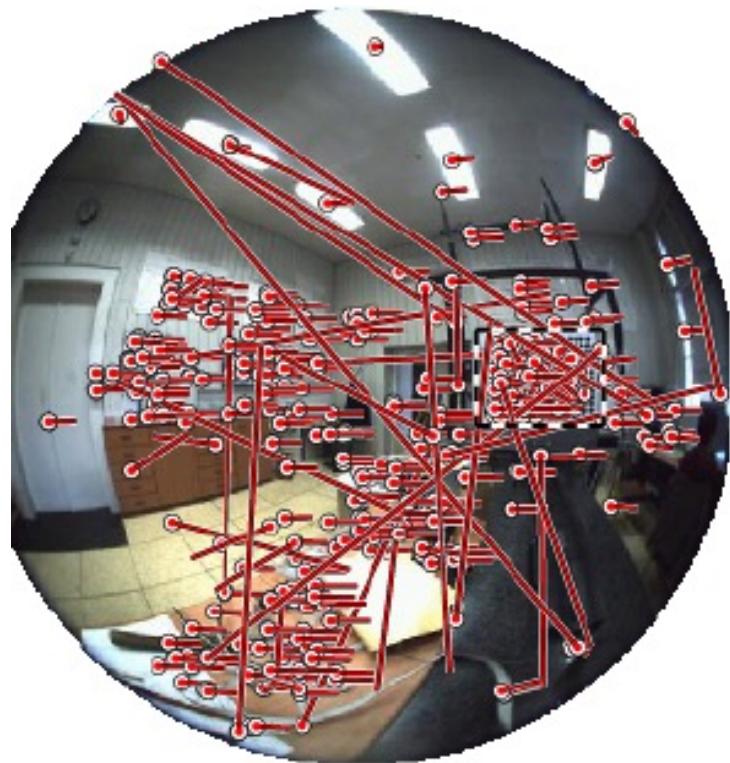
www.cs.cmu.edu/~dellaert/mosaicking

Frank Dellaert Fall 2020

Two-view geometry (next lecture)



Omnidirectional example

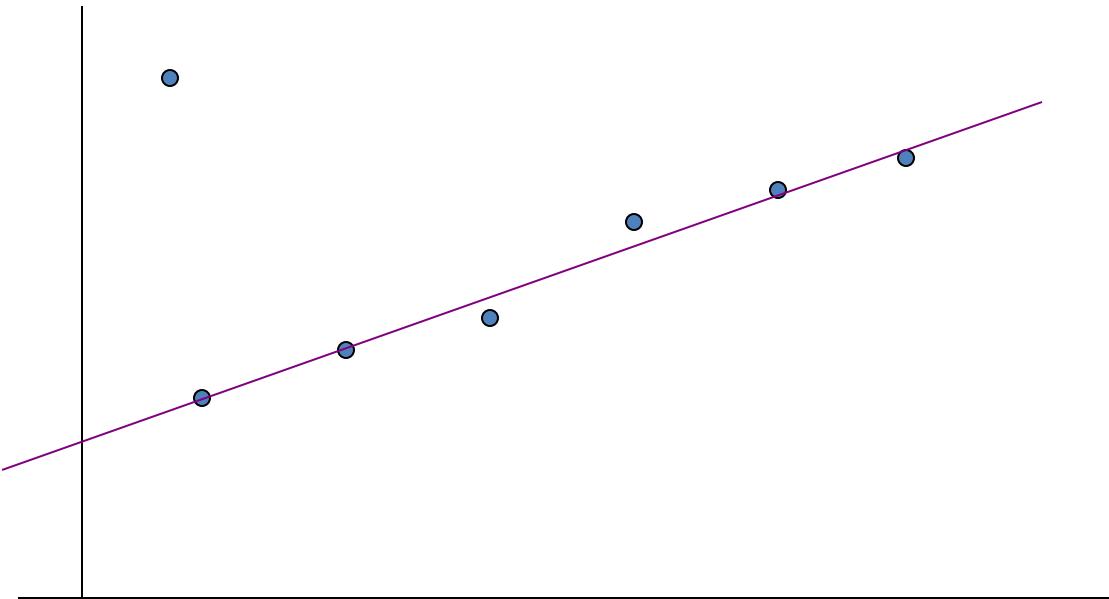


Images by Branislav Micusik, Tomas Pajdla,
[cmp.felk.cvut.cz/ demos/Fishepip/](http://cmp.felk.cvut.cz/demos/Fishepip/)

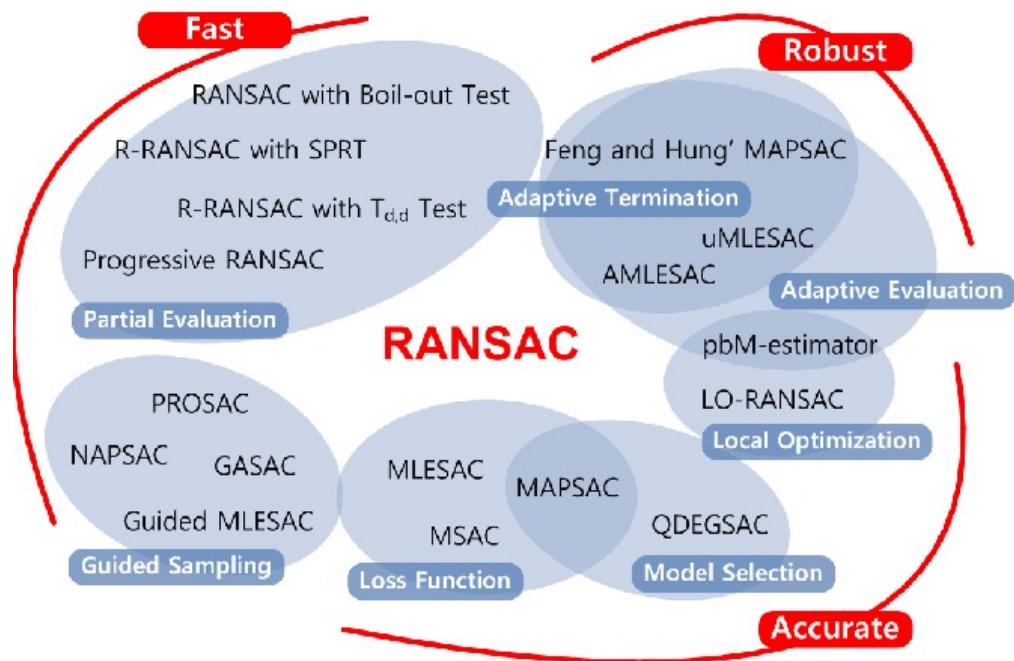
Frank Dellaert Fall 2020

Simpler Example

- Fitting a straight line

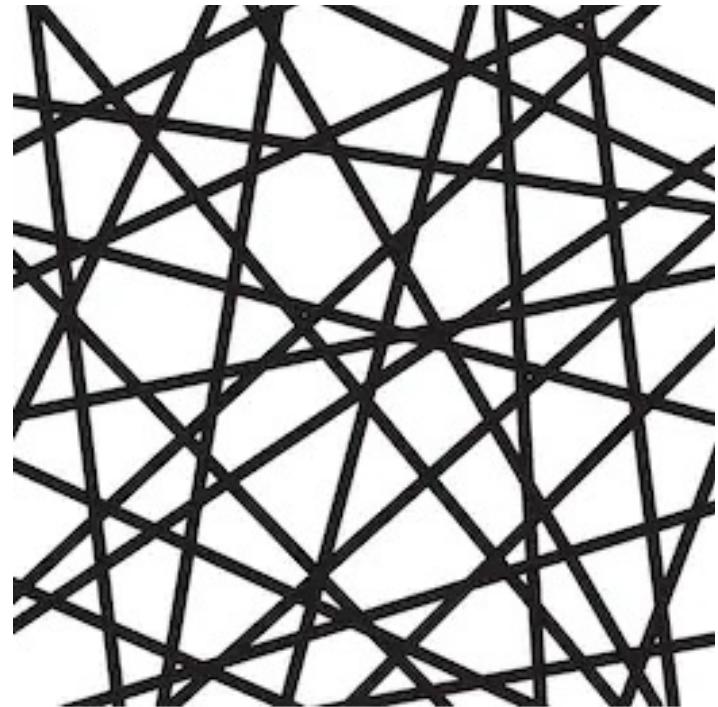


Discard Outliers



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

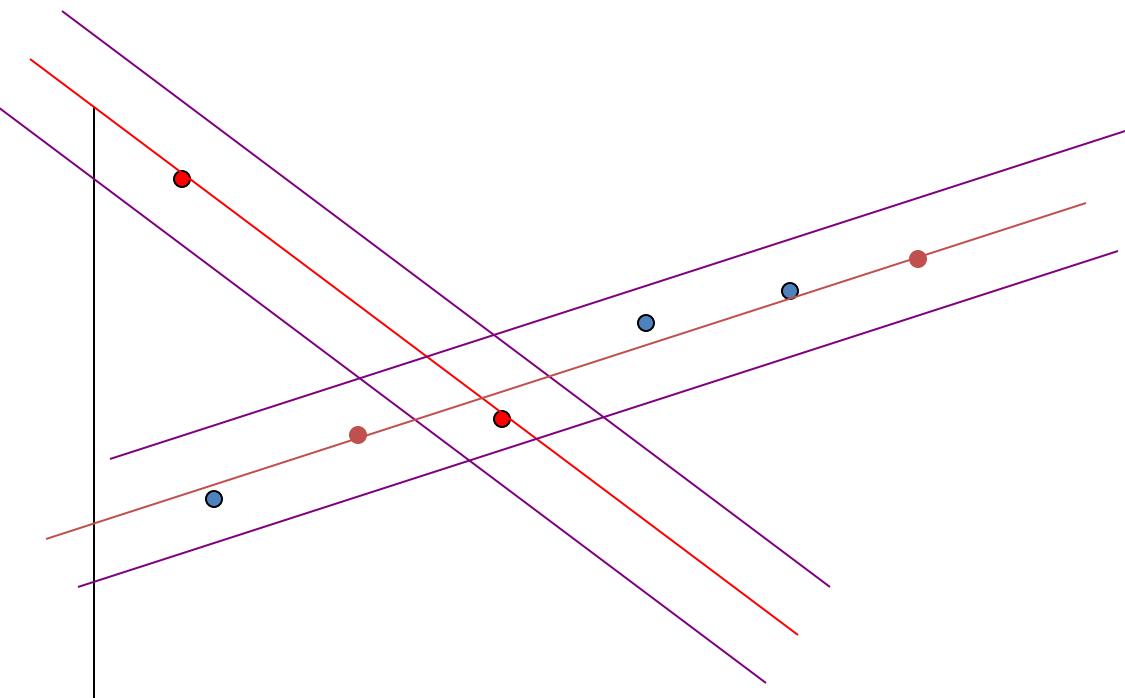
Main Idea



shutterstock.com • 547881814

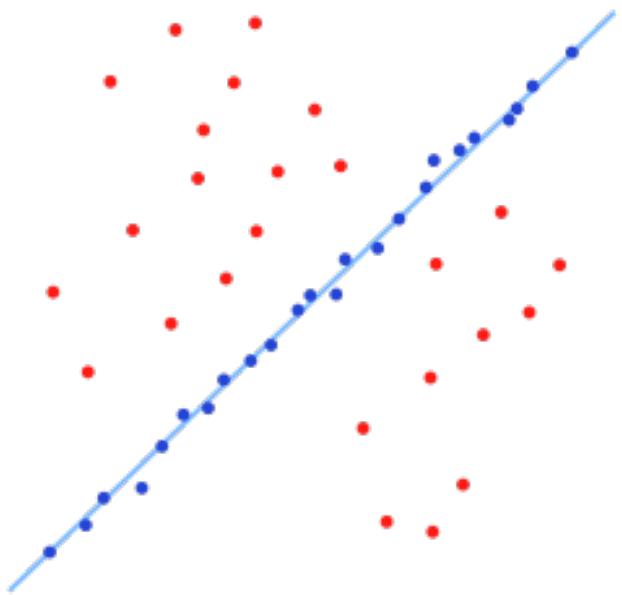
- Select 2 points at random
- Fit a line
- “Support” = number of inliers
- Line with most inliers wins

Why will this work ?



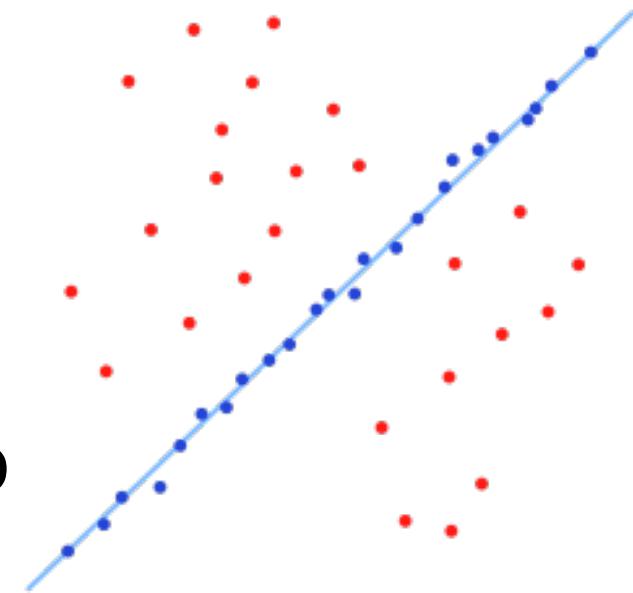
Best Line has most support

- More support -> better fit



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 $|D_i|$



In General



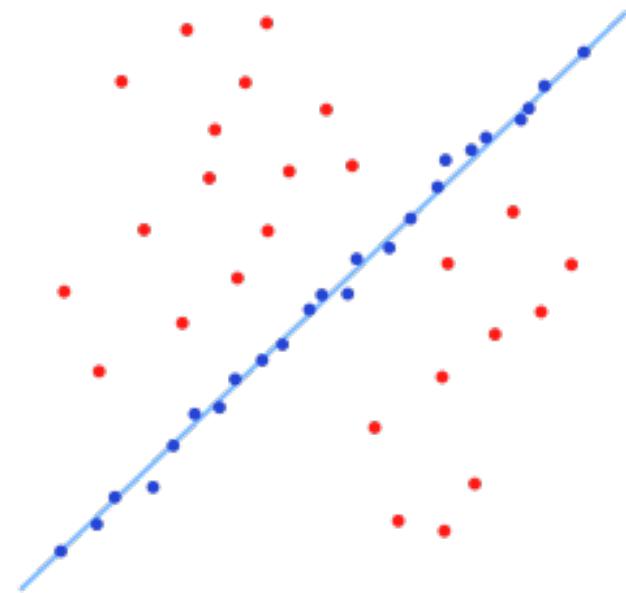
- Fit a more general model
- Sample = minimal subset
 - Translation ?
 - Homography ?
 - Euclidean transform ?

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

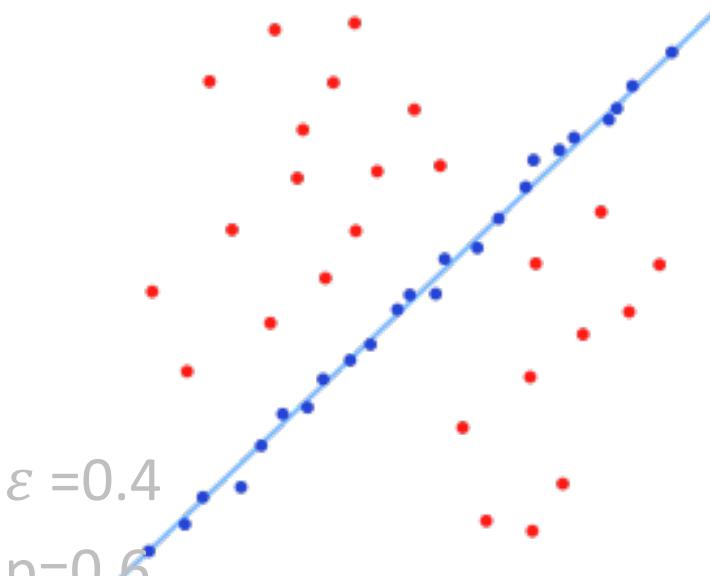
How many samples ?



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

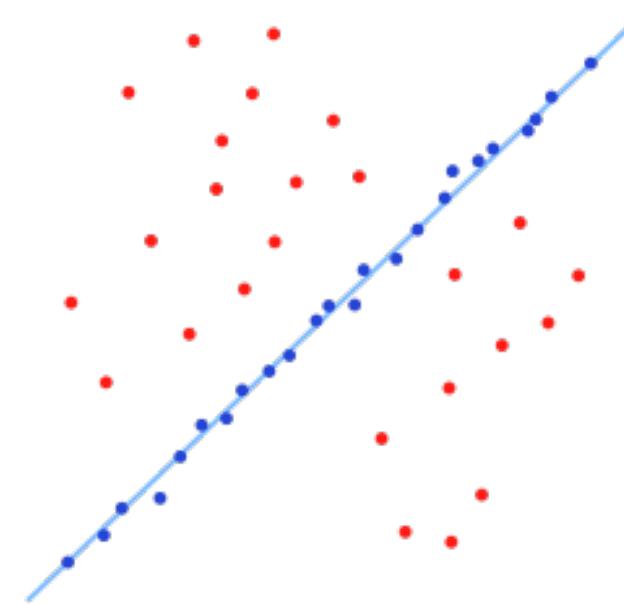
Calculate N

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

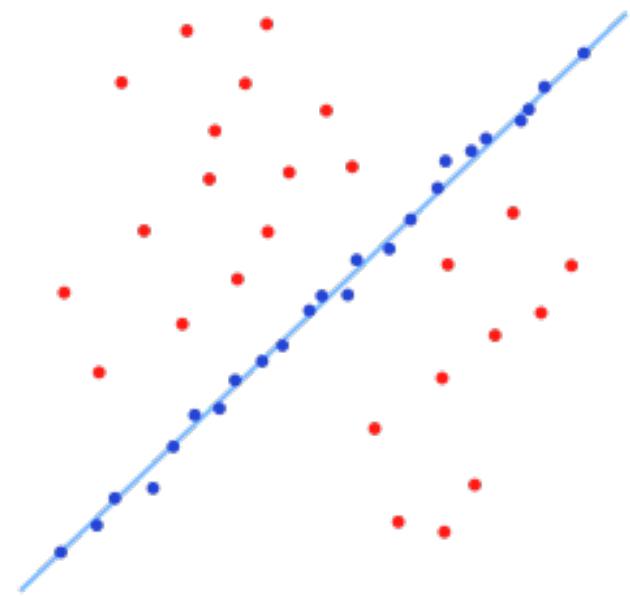


Example

- $P=0.99$
- $s=2$
 - $\varepsilon = 5\%$ $\Rightarrow N=2$
 - $\varepsilon = 50\%$ $\Rightarrow N=17$
- $s=4$
 - $\varepsilon = 5\%$ $\Rightarrow N=3$
 - $\varepsilon = 50\%$ $\Rightarrow N=72$
- $s=8$
 - $\varepsilon = 5\%$ $\Rightarrow N=5$
 - $\varepsilon = 50\%$ $\Rightarrow N=1177$



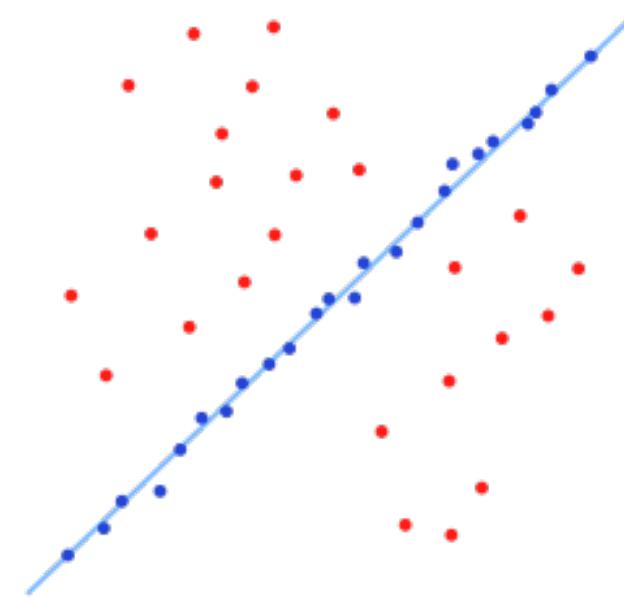
Remarks



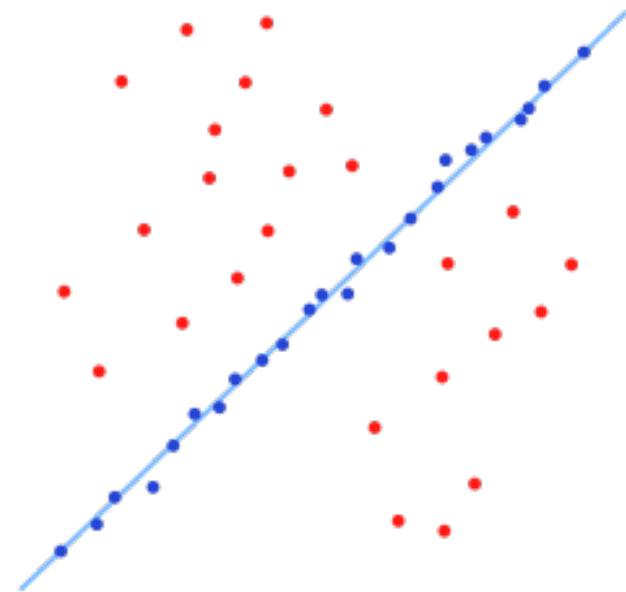
- $N = f(\varepsilon)$, **not** the number of points
- N increases steeply with s

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. \rightarrow 95% prob that $d < t$ is inlier



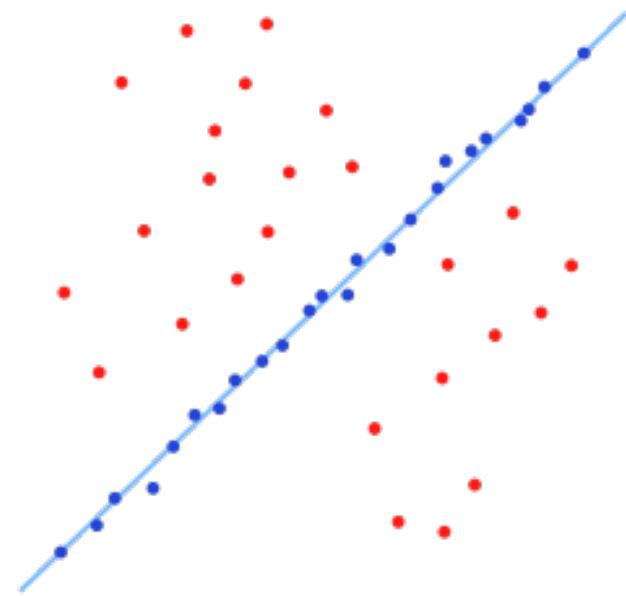
Threshold T



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

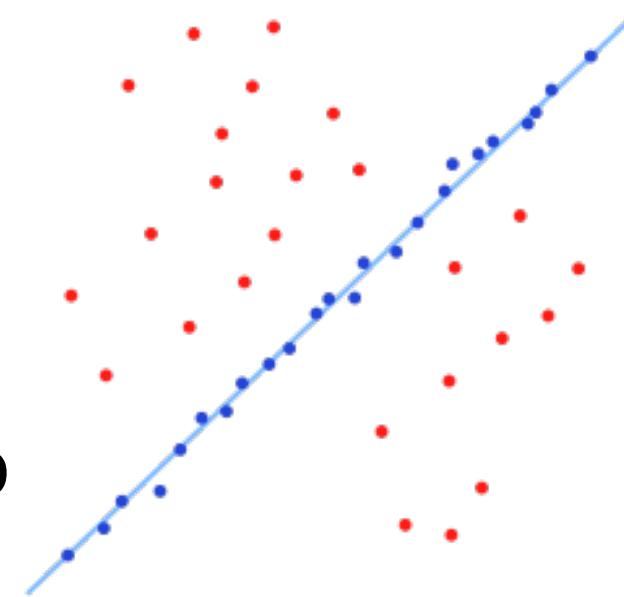
Adaptive N

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

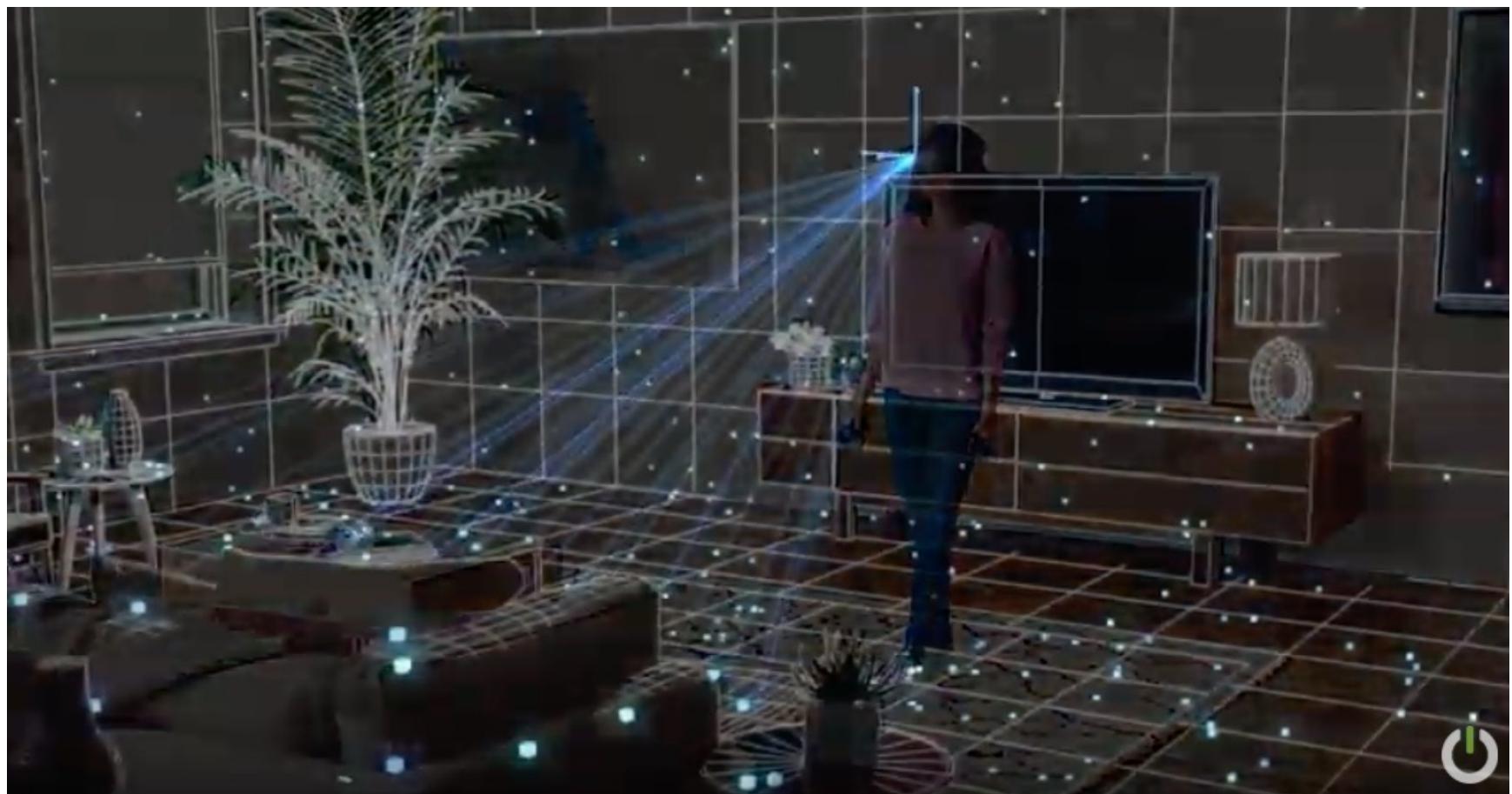


Summary: RANSAC

- 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 $|D_i|$



Pose Estimation in VR



<https://youtu.be/nrj3JE-NHMw>

Frank Dellaert Fall 2020

Review: 2D Alignment



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

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?

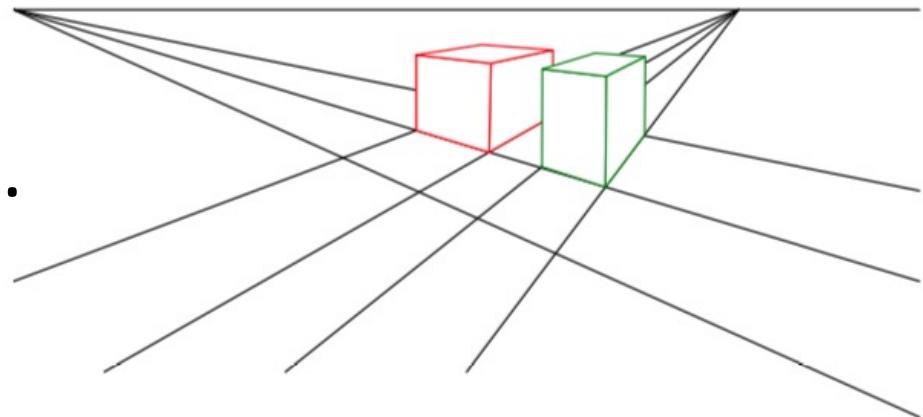
Pose Estimation



- 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:



$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} = \mathbf{P}\mathbf{X}$$

where $P = 3 \times 4$ camera matrix

and K the 3×3 calibration

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

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



$${}_w t_c$$

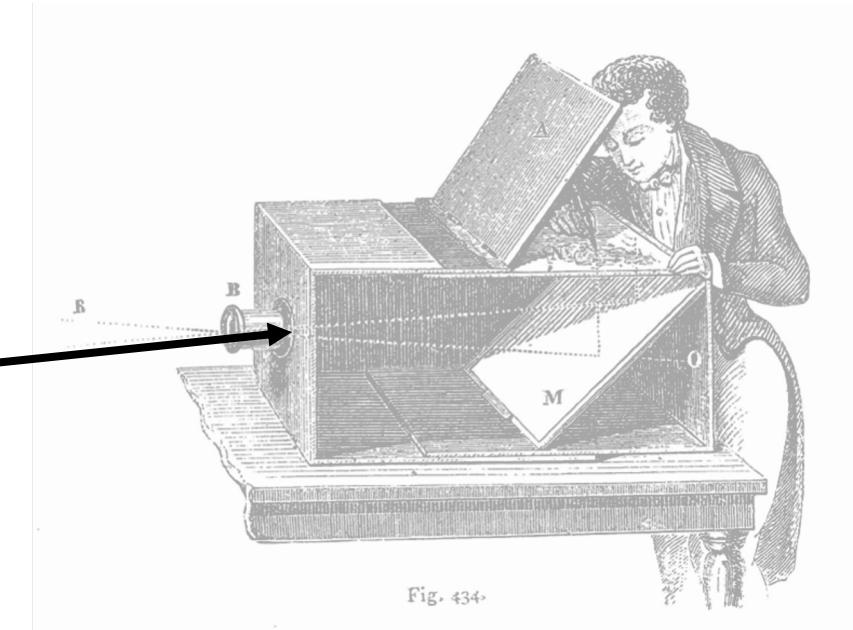


Fig. 434.

Camera Extrinsic: a Pose in 3D

- What is the geometric meaning of R and t ??
- Rotation is given by 3x3 matrix ${}^w R_c$ whose *columns* are the camera axes ${}^w X_c$, ${}^w Y_c$, ${}^w Z_c$

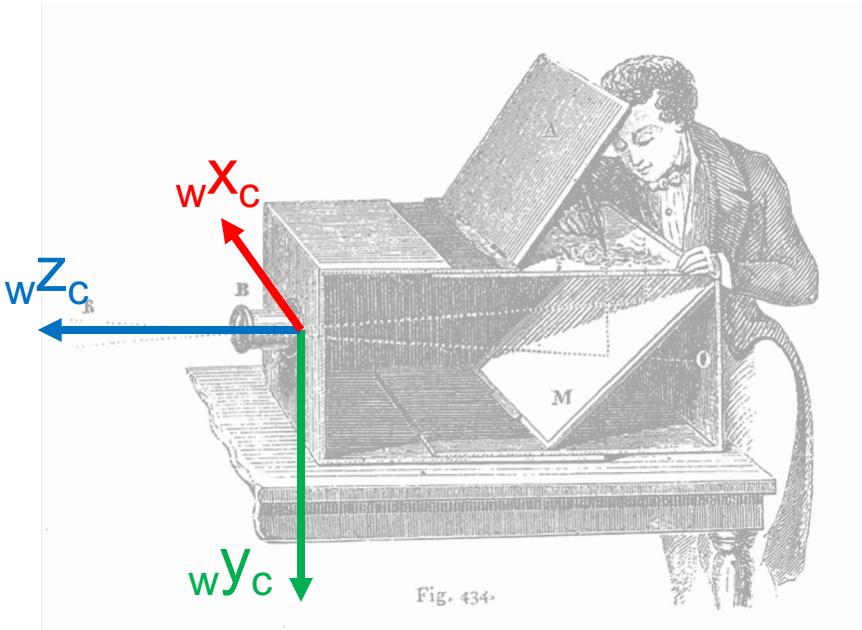


Fig. 434.

Camera Extrinsic: a Pose in 3D

- What is the geometric meaning of R and t ??
- Transforming point X_i from world to camera coordinates: ${}_{\text{w}}X_i - {}_{\text{w}}t_c = {}_{\text{w}}R_{\text{c}\text{c}}{}_{\text{c}}X_i$

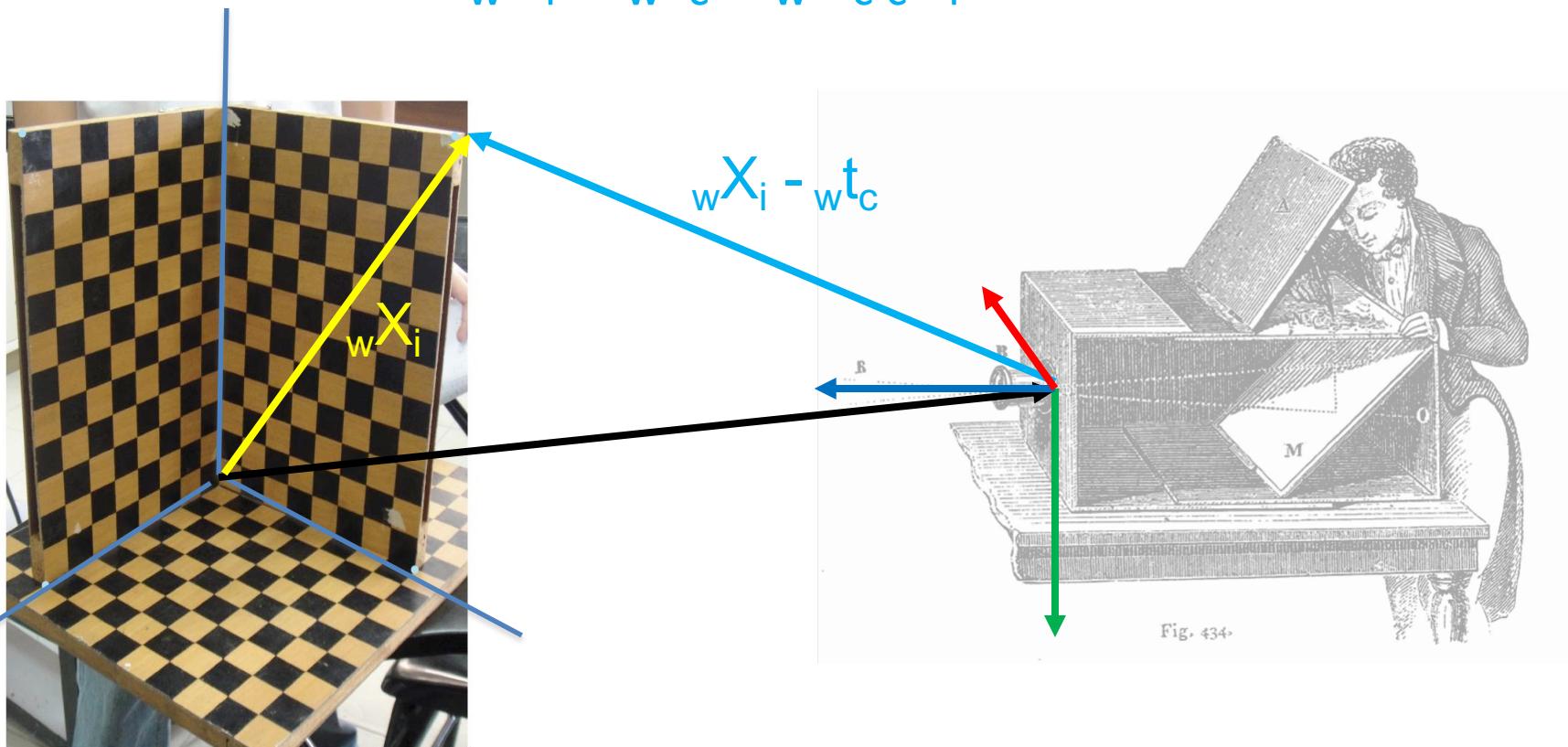
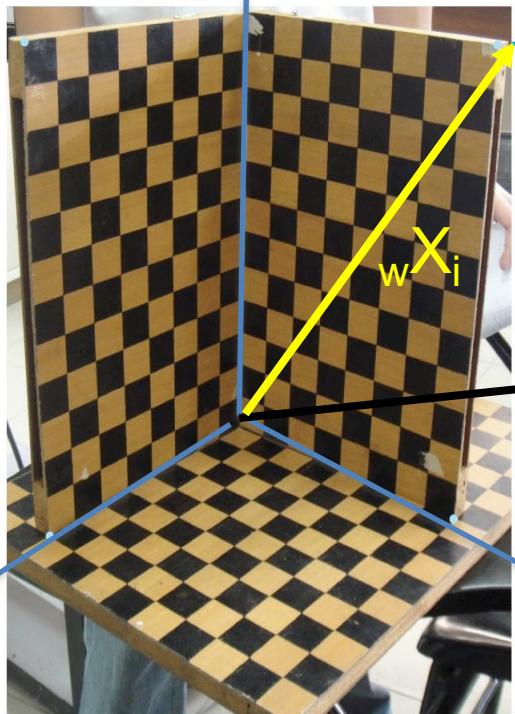


Fig. 434.

Camera Extrinsic: a Pose in 3D

- Expressed in homogeneous coordinates:

$$\begin{aligned} {}_c X_i &= {}_w R_c^T ({}_w X_i - {}_w t_c) = {}_w R_c^T [I | - {}_w t_c] {}_w X_i \\ &= [{}_w R_c^T I | - {}_w R_c^T {}_w t_c] {}_w X_i \\ &= [{}_c R_w | {}_c t_w] {}_w X_i \\ &= [R | t] {}_w X_i \end{aligned}$$



cX_i

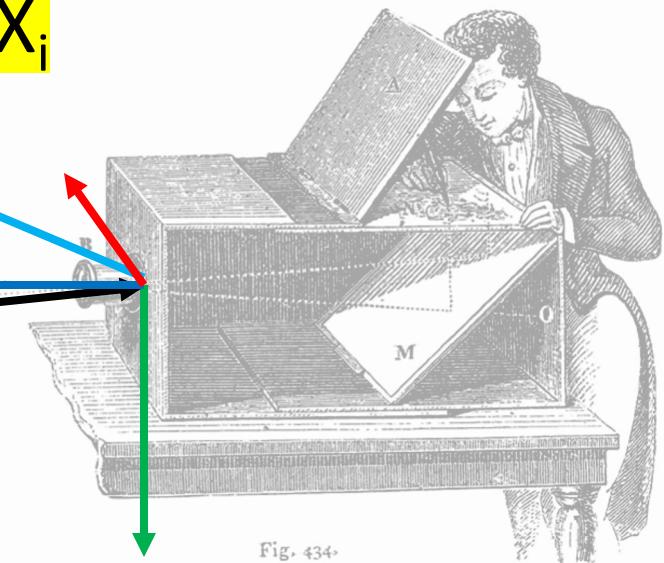
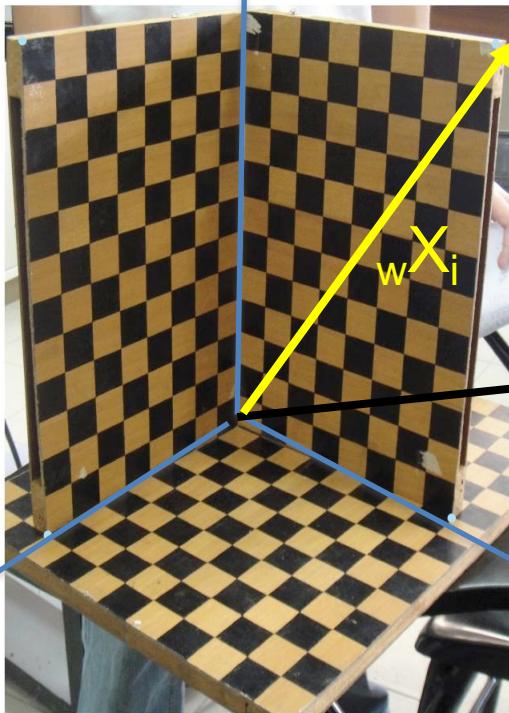


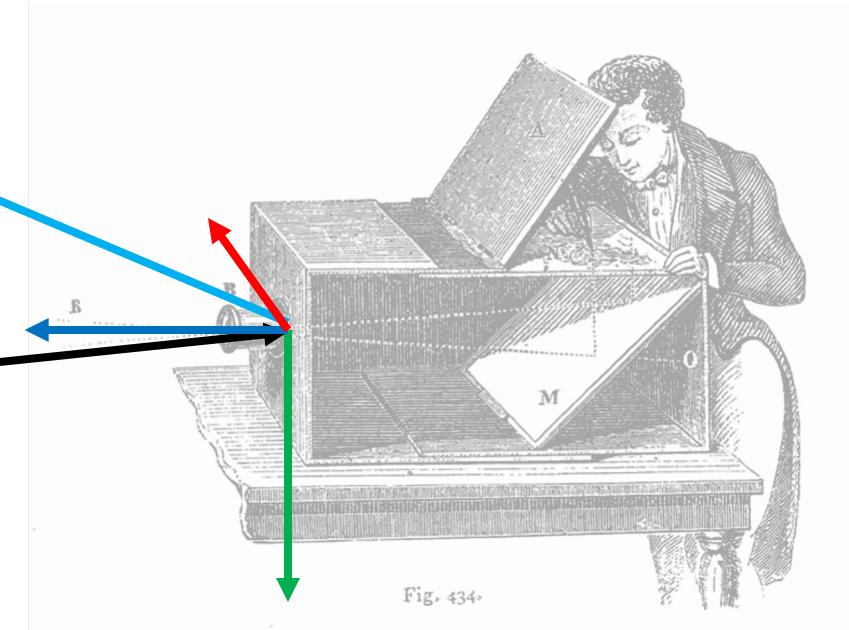
Fig. 434.

Camera Extrinsic: a Pose in 3D

- Conclusion: when people write $cX_i = [R|t] wX_i$ they are talking about (unintuitive) $[cR_w | c t_w]$
- We like use (intuitive) $cX_i = {}_wR_c^T [I| - {}_w t_c] wX_i$

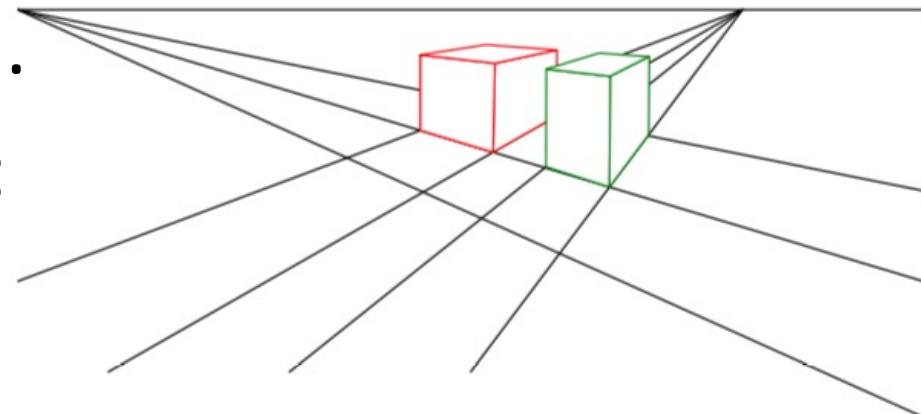


cX_i



Revision: Projective Camera Matrix

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



Camera-centric: $\mathbf{x} = \mathbf{K}_c [\mathbf{R}_w \mid {}_c\mathbf{t}_w] \mathbf{X} = \mathbf{P}\mathbf{X}$

World-centric: $\mathbf{x} = \mathbf{K}_w \mathbf{R}_c^T [\mathbf{I} \mid -{}_w\mathbf{t}_c] \mathbf{X} = \mathbf{P}\mathbf{X}$

P = same 3x4 camera matrix

and K the 3x3 calibration

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

Looking at the (opaque) camera matrix

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

$$P = [P^1 \ P^2 \ P^3 \ P^4]$$

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 $[0 \ 0 \ 0 \ 1]$

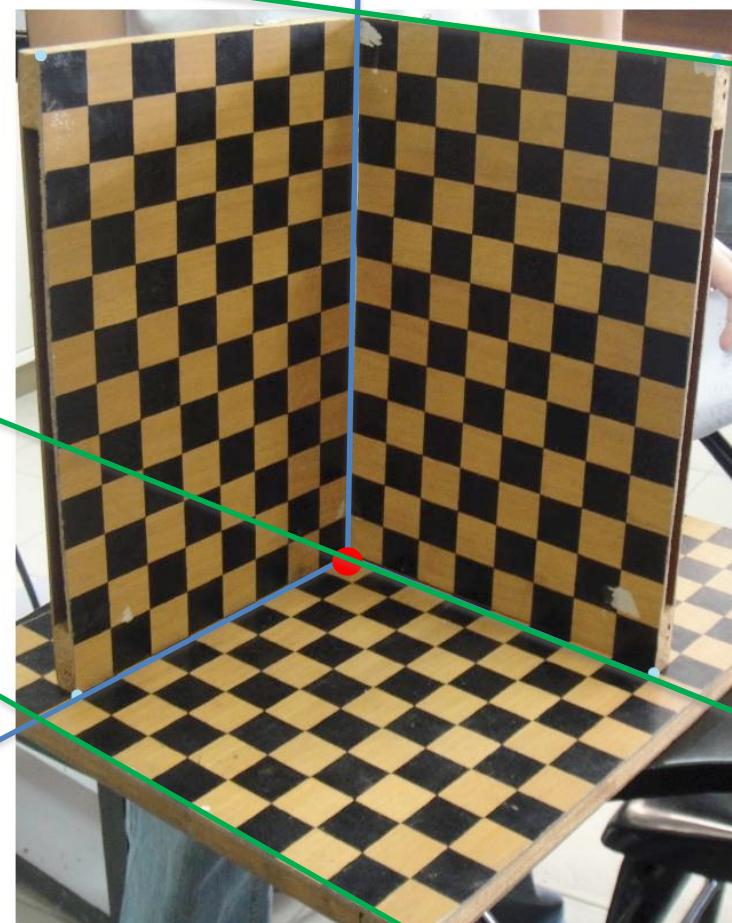
What are those ?

$[0 \ 0 \ 0 \ 1]$ is easy...

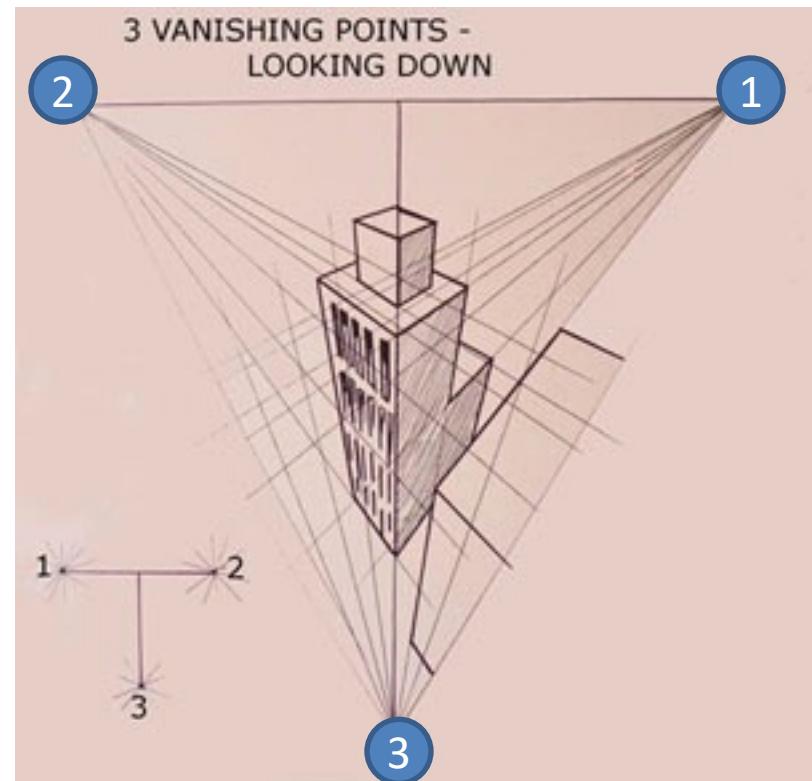
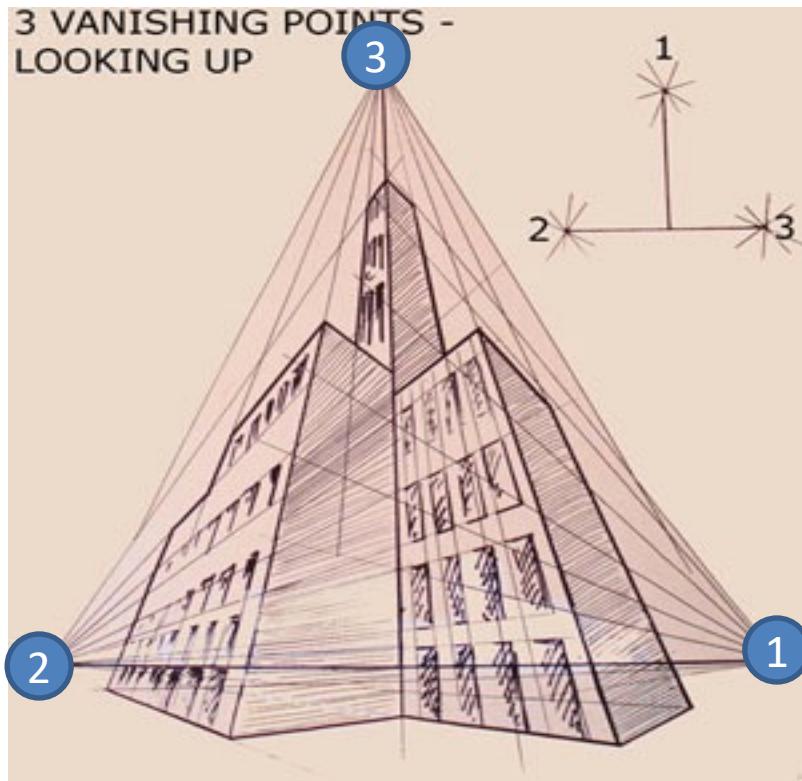
Answer:

$[0 \ 0 \ 0 \ 1]$ is the origin, so P^4 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 !

$$P = [P^1 \quad P^2 \quad P^3 \quad P^4]$$

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

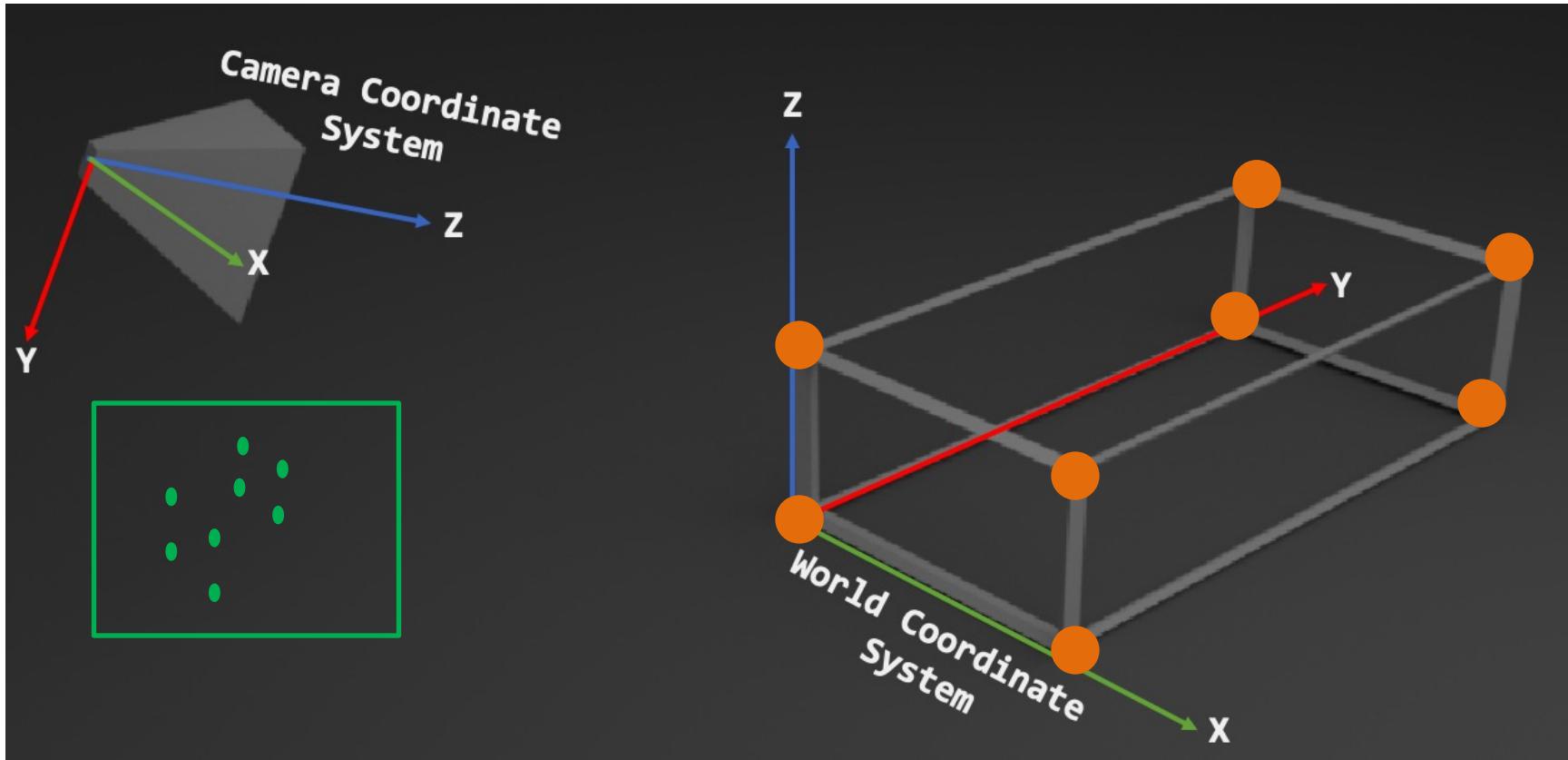
<https://www.artinstructionblog.com/perspective-drawing-tutorial-for-artists-part-2>

Frank Dellaert Fall 2020

Back to Pose Estimation!

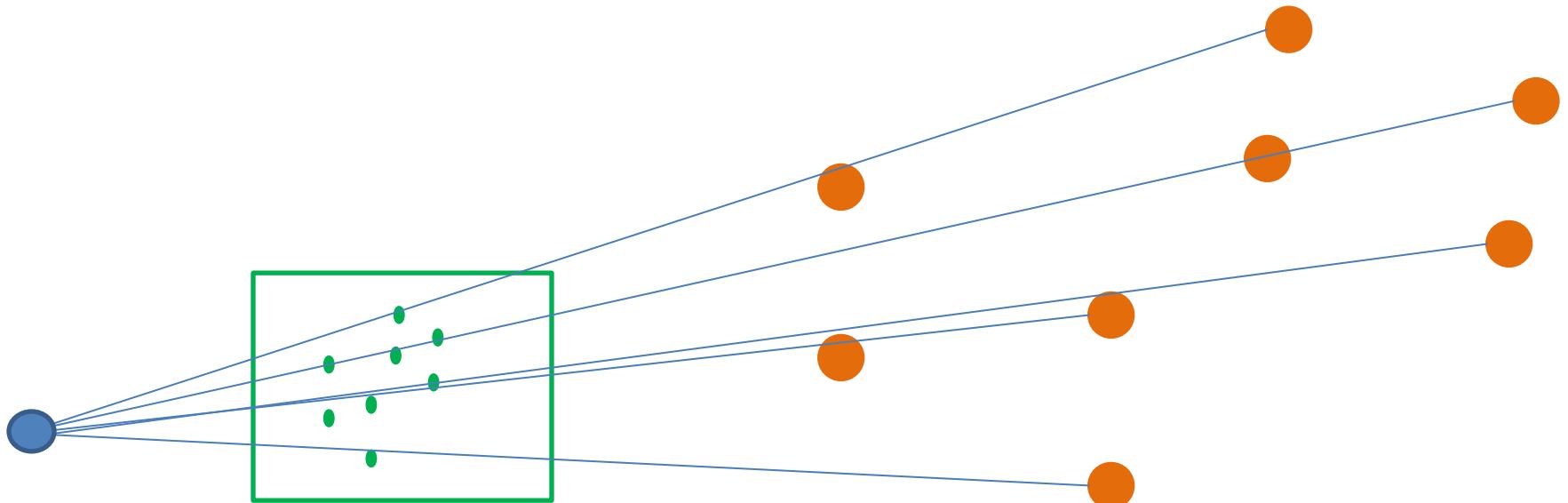
- Simple algorithm: just measure the coordinates of the origin and the three vanishing points?
- Does not work ☹:
 - Columns are only measured up to a scale.
 - $4 \text{ points} * 2\text{DOF} = \text{only } 8 \text{ DOF!}$ Missing $11-8=3$
 - 3 missing numbers are exactly those scales.

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