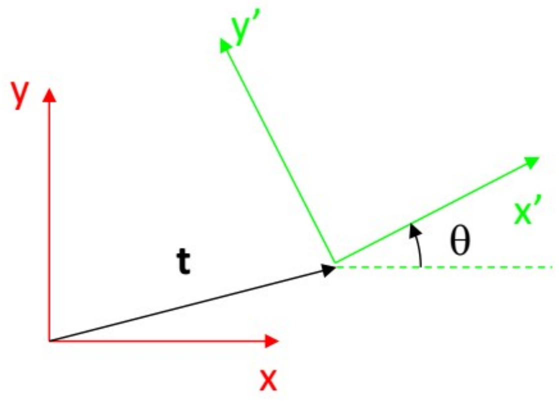
Motion models and transformations

By

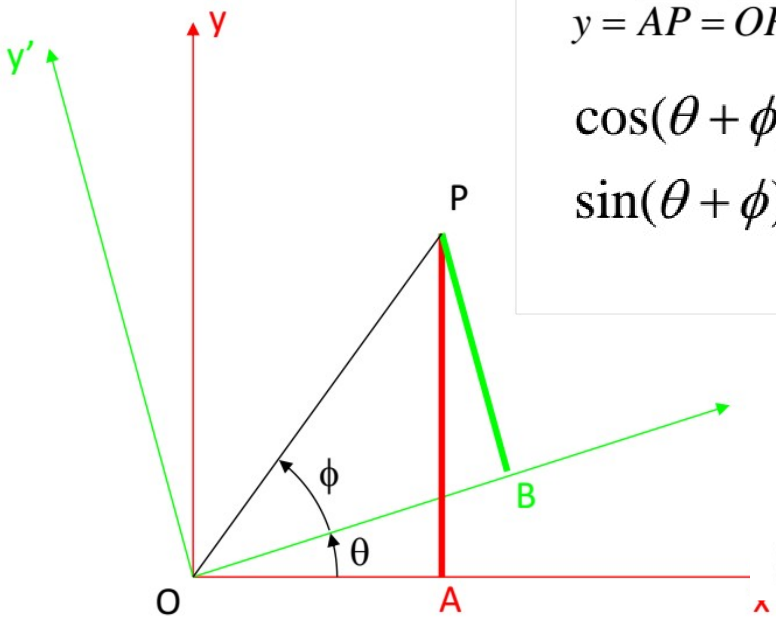
Asif Ali

Let’s start with 2D transforms

- The pose of one 2D frame with respect to another is described by
  - Translation vector **t**=$(\Delta x, \Delta y)^T$
  - Rotation angle θ
    - Rotation can also be represented as a 2x2 matrix **R**
- Object shape and size is preserved
- Number of degrees of freedom for a 2D rigid transformation?



Rotations in 2D



$$x = \overline{OA} = \overline{OP}\cos(\theta + \phi)$$

$$y = \overline{AP} = \overline{OP}\sin(\theta + \phi)$$

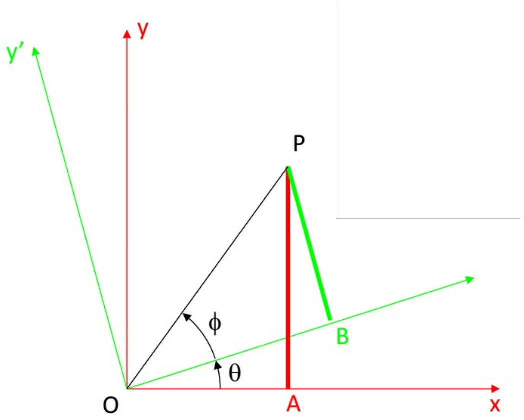
$$\cos(\theta + \phi) = \cos\theta\cos\phi - \sin\theta\sin\phi$$

$$\sin(\theta + \phi) = \cos\theta\sin\phi + \sin\theta\cos\phi$$

$$x = \underbrace{\overline{OP}\cos\phi}_{x'}\cos\theta - \underbrace{\overline{OP}\sin\phi}_{y'}\sin\theta$$

$$= x'\cos\theta - y'\sin\theta$$

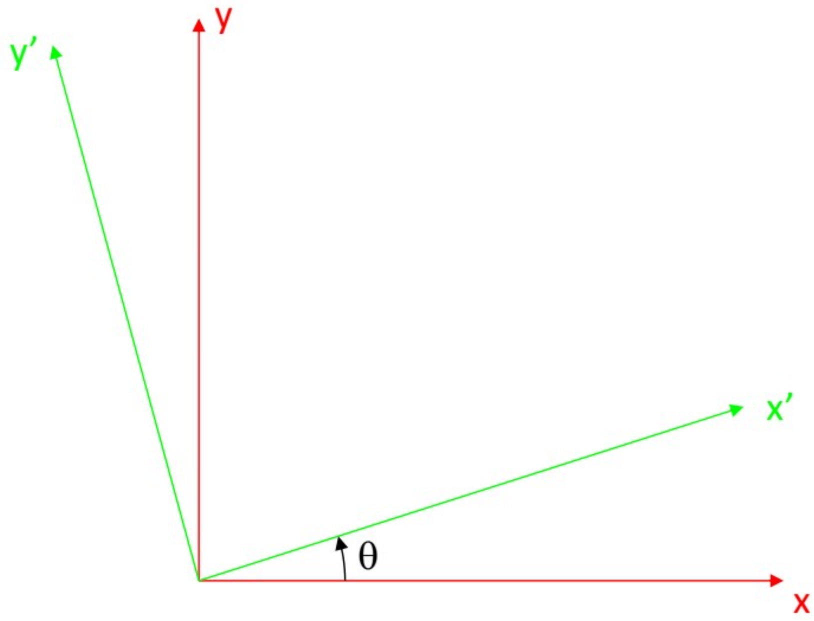
Similarly, $y = x'\sin\theta + y'\cos\theta$



$$\text{So}\quad \begin{pmatrix} x \\ y \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}}_{\mathbf{R}} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$\boxed{\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}}$$

Properties of rotation in 2D



$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{R}\begin{pmatrix} x' \\ y' \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

R describes the orientation of the “primed” frame (x’,y’) with respect to the “unprimed” frame (x,y)

- **R** is orthonormal
  - Rows, columns are orthogonal ($\mathbf{r}_1 \cdot \mathbf{r}_2 = 0$, $\mathbf{c}_1 \cdot \mathbf{c}_2 = 0$)
  - Transpose is the inverse; $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
  - Determinant is $|\mathbf{R}| = 1$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{R}^T \begin{pmatrix} x \\ y \end{pmatrix}$$

Scaling and Translation

Let's simplify things
(by making them more complex)

- Homogeneous Coordinates
- Points can be represented by homogeneous coordinates
- So rotation can be written as
  - This simplifies transform equations; instead of

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{R}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \qquad \mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$$

  - we have

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0\ \ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \widetilde{\mathbf{x}'} = \mathbf{H}\tilde{\mathbf{x}}$$

$\tilde{\mathbf{x}} \in P^2$, where $P^2 = R^3 - (0,0,0)$ is called a 2D projective space

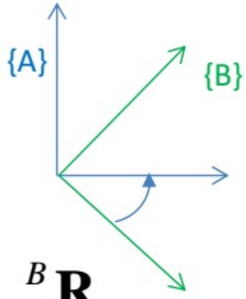
Example

- Transform the 2D point x = $(10, 20)^T$ using a rotation of 45 degrees and translation of (+40, -30).
- Solution:
  - The point in homogeneous coords is $\tilde{x} = (10, 20, 1)^T$
  - The transformation matrix is

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 \;\; 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos 45 & -\sin 45 & 40 \\ \sin 45 & \cos 45 & -30 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} .707 & -.707 & 40 \\ .707 & .707 & -30 \\ 0 & 0 & 1 \end{pmatrix}$$

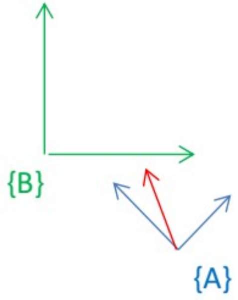
  - Transforming the point:

$$\mathbf{H}\tilde{\mathbf{x}} = \begin{pmatrix} .707 & -.707 & 40 \\ .707 & .707 & -30 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 \\ 20 \\ 1 \end{pmatrix} = \begin{pmatrix} 32.9 \\ -8.8 \\ 1 \end{pmatrix}$$



$^B_A\mathbf{R}$

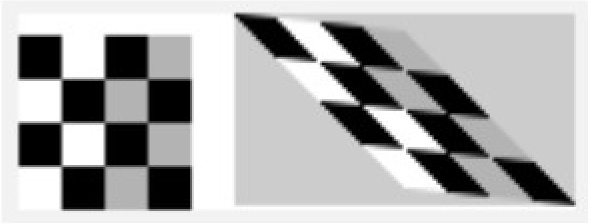
describes the orientation of A with respect to B



Other 2D-2D transforms

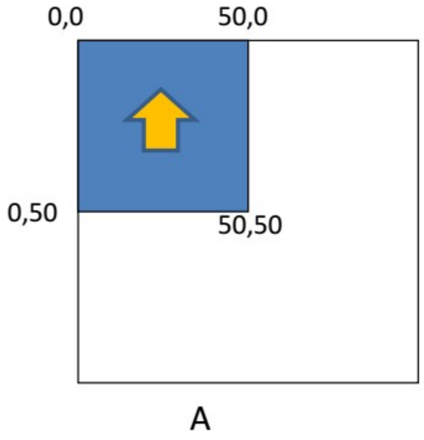
- Scaling
- Translation
- Affine
  - Models rotation, translation, scaling, shearing, and reflection

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$



- Image "A" is modified by the affine transform below. Sketch image "B"

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0.25 & 1.5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$



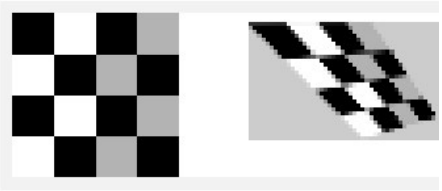
Percpective Transform (aka Homography)

- Most general type of linear 2D-2D transform
- H is an arbitrary 3x3 matrix
- We still need to divide by 3rd element, so:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

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

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 / x_3 \\ x_2 / x_3 \\ 1 \end{pmatrix}$$



As we will see later, a homography maps points from the projection of one plane to the projection of another plane

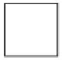
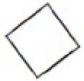
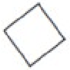
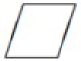
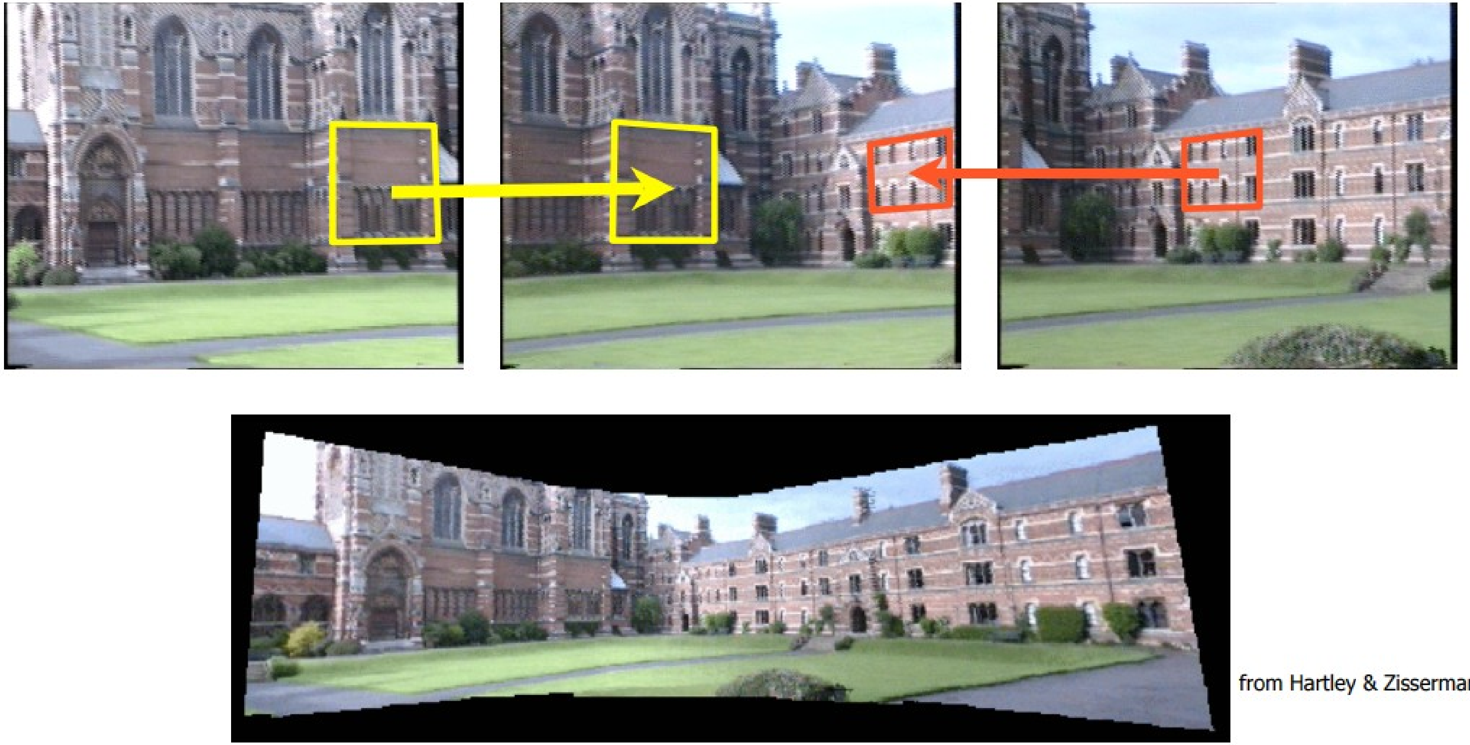
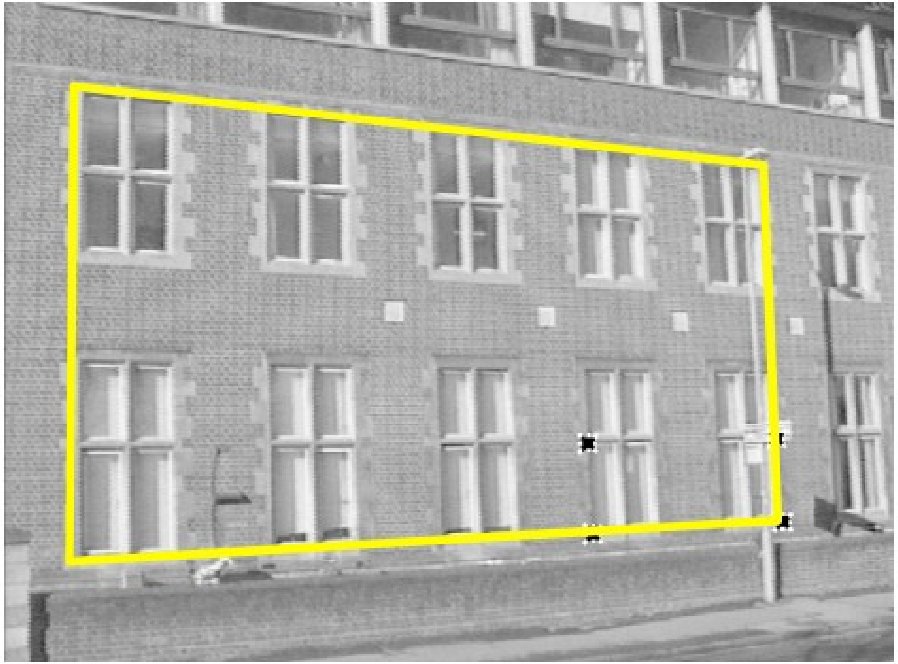
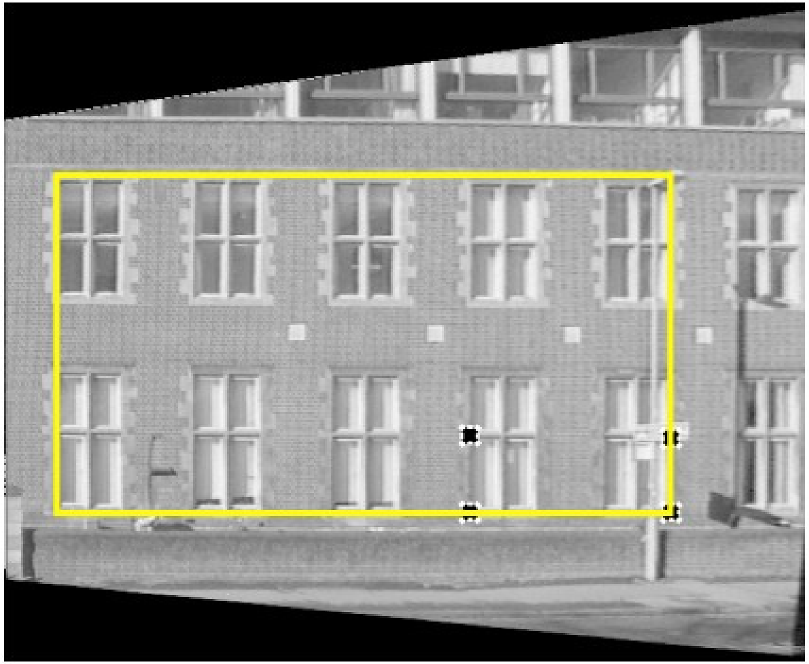
Transformation	**Matrix**	**# DoF**	**Preserves**	**Icon**
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Table 2.1 Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 2×3 matrices are extended with a third $[\mathbf{0}^T\ 1]$ row to form a full 3×3 matrix for homogeneous coordinate transformations.

How to use homography in real-life?

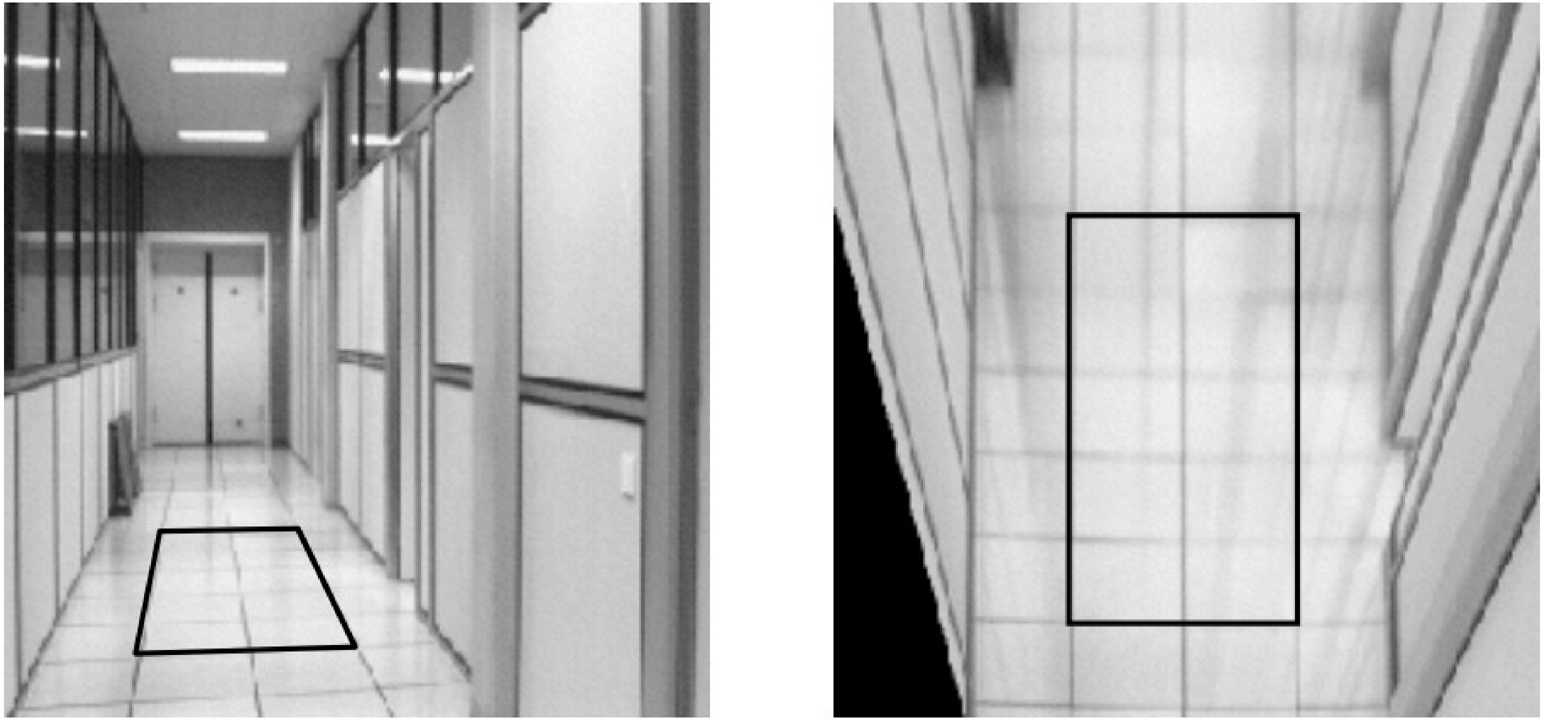


from Hartley & Zisserman

from Hartley & Zisserman

Bird’s eye view correction



from Hartley & Zisserman

Tow

Least Squares Fitting to a Line

- We have some measurement data (*xi,yi*)
- We want to fit the data to $y = f(x) = mx + b$
- We will find the parameters (*m,b*) that minimize the objective function

$$E = \sum_i \left| y_i - f(x_i) \right|^2$$

Example
(x1,y1) = (1,1)
(x2,y2) = (2,3)
(x3,y3) = (3,4)

Linear Least Squares

- In general
  - The input data can be vectors
  - The function can be a linear combination of the input data
- We write **A x = b**
  - The parameters to be fit are in the vector **x**
  - The input data is in **A,b**
- Example of a line
  - Parameter vector

$$\mathbf{x} = \begin{pmatrix} m \\ b \end{pmatrix}$$

  - Linear equations

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

- So for a line

$$\mathbf{A} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_N & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Solving Linear Least Squares

- Want to minimize

$$E = \left|\mathbf{A}\mathbf{x} - \mathbf{b}\right|^2$$

- Expanding we get

$$E = \mathbf{x}^T\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} - 2\mathbf{x}^T\left(\mathbf{A}^T\mathbf{b}\right) + \left|\mathbf{b}\right|^2$$

- To find the minimum, take derivative wrt x and set to zero, getting

$$\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

Called the "normal equations"

- To solve, can do

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

"pseudo inverse"

$$\mathbf{A}^+ = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$$

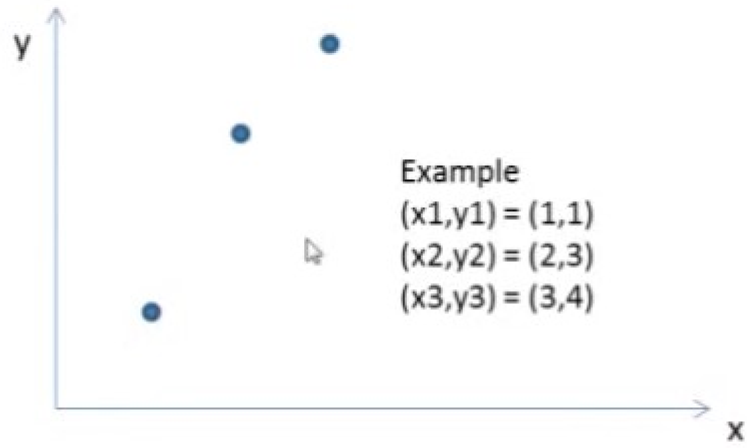
- In Matlab can do
  - x = pinv(A)*b;
  - or x = A\b;

- Note – it is preferable to solve the normal equations using Cholesky decomposition

Example

- The linear system for the line example earlier is **Ax=b**, where

$$\mathbf{A}=\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix},\quad \mathbf{b}=\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}=\begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix}$$



- Normal equations $\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x}=\mathbf{A}^T\mathbf{b}$

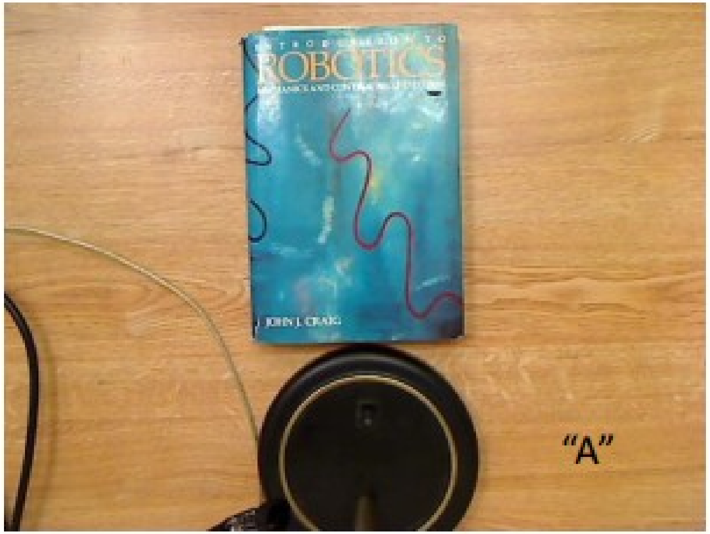
$$\mathbf{A}^T\mathbf{A}=\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}=\begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix},\quad \mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}=\begin{pmatrix} 1.5 \\ -0.333 \end{pmatrix}$$

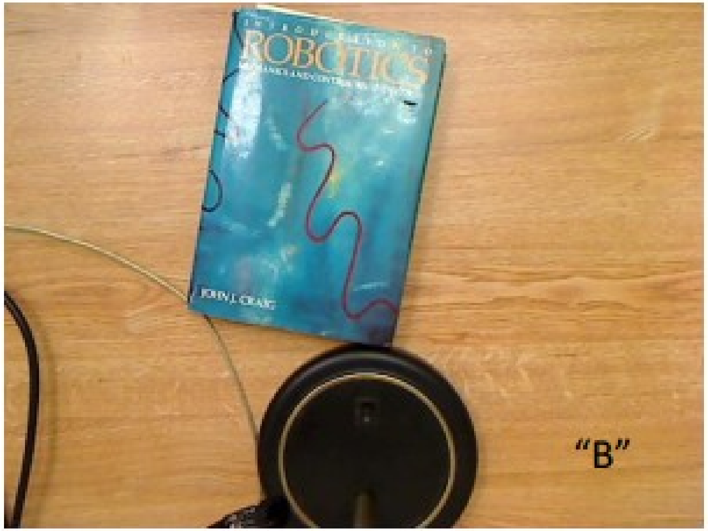
- So the best fit line is $y = 1.5x - 0.333$

Finding an image transform

- If you know a set of point correspondences, you can estimate the parameters of the transform
- Example – find the rotation and translation of the book in the images below

```
% Using imtool, we manually find
% corresponding points (x;y), which are
% the four corners of the book
pA = [
    221 413  416  228;
    31   20  304  308];
pB = [
    214 404  352  169;
    7    34  314  280];
```





Example (continued)

- A 2D rigid transform is

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} c & -s & t_x \\ s & c & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}, \quad \text{where } c = \cos\theta,\ s = \sin\theta$$

- Or

$$x_B = cx_A - sy_A + t_x$$
$$y_B = sx_A + cy_A + t_y$$

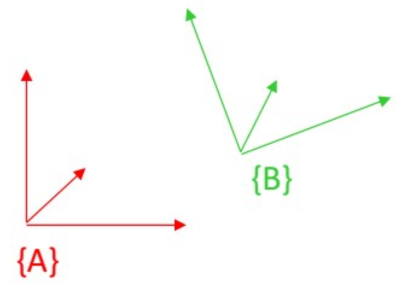
- We put into the form **Ax = b**, where

$$\mathbf{A} = \begin{pmatrix} x_A^{(1)} & -y_A^{(1)} & 1 & 0 \\ y_A^{(1)} & x_A^{(1)} & 0 & 1 \\ \vdots & & & \\ y_A^{(N)} & x_A^{(N)} & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} c \\ s \\ t_x \\ t_y \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} x_B^{(1)} \\ y_B^{(1)} \\ \vdots \\ y_B^{(N)} \end{pmatrix}$$

Note: c and s are not really independent variables; however we treat them as independent so that we get a system of linear equations

3D-3D coordinate transforms
(excellent reference is Introduction to Robotics)

- Coordinate frames
  - Denote as {A}, {B}, etc
  - Examples: camera, world, model



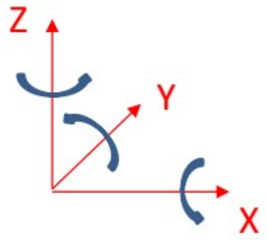
- The pose of {B} with respect to {A} is described by
  - Translation vector **t**
  - Rotation matrix **R**
- Rotation is a 3x3 matrix
  - It represents 3 angles

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

- The rotation matrix has 9 elements
- We only have 3 degrees of freedom (DOF)
  - Roll, pitch and yaw
- Unfortunately, there are many possible ways to represent a rotation with 3 numbers, depending on the convention that you use
  - We will look at a few of them
- However, the rotation matrix is always unique

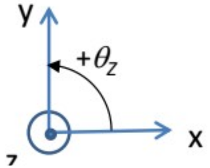
XYZ angles to represent rotation

- One way to represent a 3D rotation is by doing successive rotations about the X,Y, and Z axes
- We'll present this first, because it is easy to understand



- However, it is not the best way for several reasons:
  - The result depends on the order in which the transforms are applied
  - Sometimes one or more angles change dramatically in response to a small change in orientation
  - Some orientations have singularities; i.e., the angles are not well defined

- Rotation about the Z axis

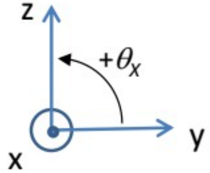


⊙ Points toward me

⊗ Points away from me

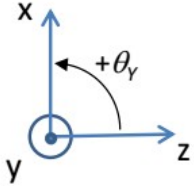
$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix}$$

- Rotation about the X axis



$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix}$$

- Rotation about the Y axis



$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix}$$

Note signs are different than in

3D Rotation matrix

- We can concatenate the 3 rotations to yield a single 3x3 rotation matrix; e.g.,

$$\mathbf{R} = \mathbf{R}_Z \mathbf{R}_Y \mathbf{R}_X$$
$$= \begin{pmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{pmatrix}$$

- Note: we use the convention that to rotate a vector, we pre-multiply it; i.e., **v' = R v**
  - This means that if $\mathbf{R} = \mathbf{R}_Z\, \mathbf{R}_Y\, \mathbf{R}_X$, we actually apply the X rotation first, then the Y rotation, then the Z rotation

Transforming to-and-fro

- We can rotate a vector in frame A to obtain its representation in frame B

$$ {}^{B}\mathbf{v} = {}^{B}_{A}\mathbf{R}\ {}^{A}\mathbf{v} $$

- Note: as in 2D, rotation matrices are orthonormal so the inverse of a rotation matrix is just its transpose

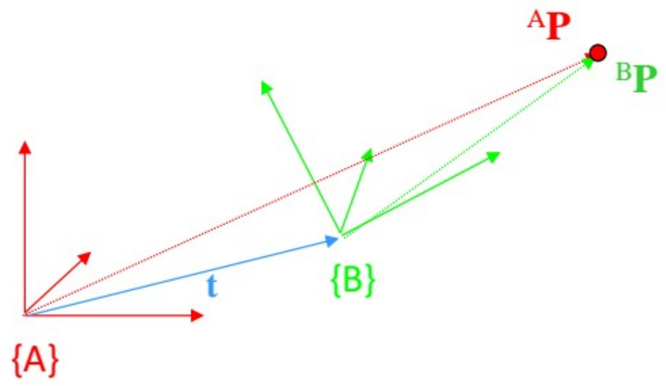
$$ \left({}^{B}_{A}\mathbf{R}\right)^{-1} = \left({}^{B}_{A}\mathbf{R}\right)^{T} = {}^{A}_{B}\mathbf{R} $$

Transforming Point

- We can use **R,t** to transform a point from coordinate frame {B} to frame {A}

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\ {}^{B}\mathbf{P} + \mathbf{t} $$

- Where
  - ${}^{A}\mathbf{P}$ is the representation of **P** in frame {A}
  - ${}^{B}\mathbf{P}$ is the representation of **P** in frame {B}



- Note

$\mathbf{t}$ is the translation of B's origin in the A frame, ${}^{A}\mathbf{t}_{Borg}$

General Rigid Transformation

- A general rigid transformation is a rotation followed by a translation

$$^{B}\mathbf{P} = {}^{B}_{A}\mathbf{R}\,{}^{A}\mathbf{P} + {}^{B}\mathbf{t}_{Aorg}$$

- Can be represented by a single 4x4 homogeneous transformation matrix

$${}^{B}_{A}\mathbf{H} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & x_0 \\ r_{21} & r_{22} & r_{23} & y_0 \\ r_{31} & r_{32} & r_{33} & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$^{B}\mathbf{P} = {}^{B}_{A}\mathbf{H}\,{}^{A}\mathbf{P} = \begin{pmatrix} r_{11}x + r_{12}y + r_{13}z + x_0 \\ r_{21}x + r_{22}y + r_{23}z + y_0 \\ r_{31}x + r_{32}y + r_{33}z + z_0 \\ 1 \end{pmatrix}$$

Inverse Transformation

- The matrix inverse is the inverse transformation

$$ {}^{A}_{B}\mathbf{H} = \left({}^{B}_{A}\mathbf{H}\right)^{-1} $$

- Note – unlike rotation matrices, the inverse of a full 4x4 homogeneous transformation matrix is not the transpose

$$ {}^{A}_{B}\mathbf{H} \neq \left({}^{B}_{A}\mathbf{H}\right)^{T} $$

- What is the transformation inverse?

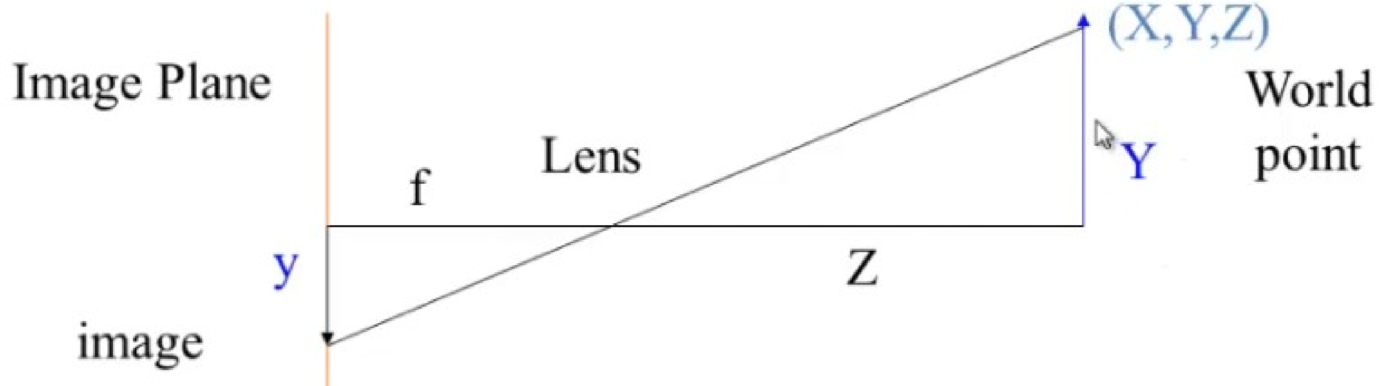
Go through that reference book

Let’s get carried away

- Can concatenate transformations together
  - Leading subscripts cancel trailing superscripts

$$ {}^{C}_{A}\mathbf{H} = {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H} \qquad {}^{D}_{A}\mathbf{H} = {}^{D}_{C}\mathbf{H}\ {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H}, \quad \text{etc} $$

Image Formation: Perspective Projection



$$\frac{-y}{Y} = \frac{f}{Z}$$

$$y = -\frac{fY}{Z} \qquad x = -\frac{fX}{Z}$$

What about generic homography?

Estimating a Homograpy

Matrix Form:

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

Equations:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

DOF

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

- 9 Numbers means 9DOF… why?
- Notice we can multiply H_ij by non-zero without changing the equations:

$$x' = \frac{kh_{11}x + kh_{12}y + kh_{13}}{kh_{31}x + kh_{32}y + kh_{33}}$$

$$y' = \frac{kh_{21}x + kh_{22}y + kh_{23}}{kh_{31}x + kh_{32}y + kh_{33}}$$

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Let’s now kill the odd guy

One approach: Set $h_{33} = 1$.

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

Some math hocus-pocus

Multiplying through by denominator

$$(h_{31}x + h_{32}y + 1)x' = h_{11}x + h_{12}y + h_{13}$$

$$(h_{31}x + h_{32}y + 1)y' = h_{21}x + h_{22}y + h_{23}$$

Rearrange

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' = x'$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' = y'$$

Linear Algebra

$$
\begin{array}{r c c c c}
 & \mathbf{2N \times 8} & \mathbf{8 \times 1} & & \mathbf{2N \times 1} \\
\begin{array}{r} \textbf{Point 1} \\ \\ \textbf{Point 2} \\ \\ \textbf{Point 3} \\ \\ \textbf{Point 4} \\ \\ \end{array} &
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\
0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\
x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\
0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\
x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\
0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4
\end{bmatrix} &
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} & = &
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} \\
\textbf{additional points} & \vdots & & & \vdots
\end{array}
$$

Linear Equations

Linear equations

$$\underset{2N\times 8}{\mathbf{A}}\;\underset{8\times 1}{\mathbf{h}} = \underset{2N\times 1}{\mathbf{b}}$$

Solve:

$$\overset{8\times 2N}{\mathbf{A}^T}\;\overset{2N\times 8}{\mathbf{A}}\;\overset{8\times 1}{\mathbf{h}} = \overset{8\times 2N}{\mathbf{A}^T}\;\overset{2N\times 1}{\mathbf{b}}$$

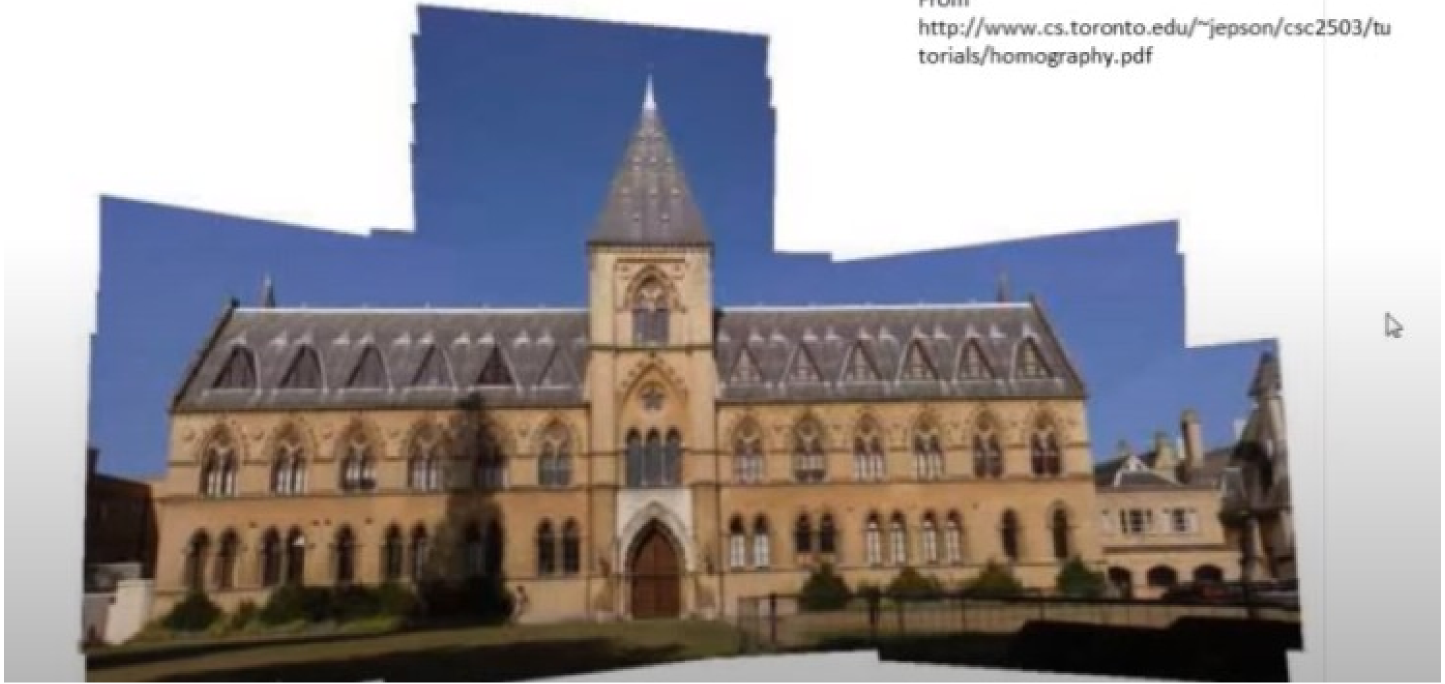
$$\overbrace{(\mathbf{A}^T\;\mathbf{A})}^{8\times 8}\;\overset{8\times 1}{\mathbf{h}} = \overbrace{(\mathbf{A}^T\;\mathbf{b})}^{8\times 1}$$

$$\mathbf{h} = (\mathbf{A}^T\;\mathbf{A})^{-1}\,(\mathbf{A}^T\;\mathbf{b})$$

Matlab: $\mathbf{h} = \mathbf{A} \setminus \mathbf{b}$

Example Application - Building Mosaics

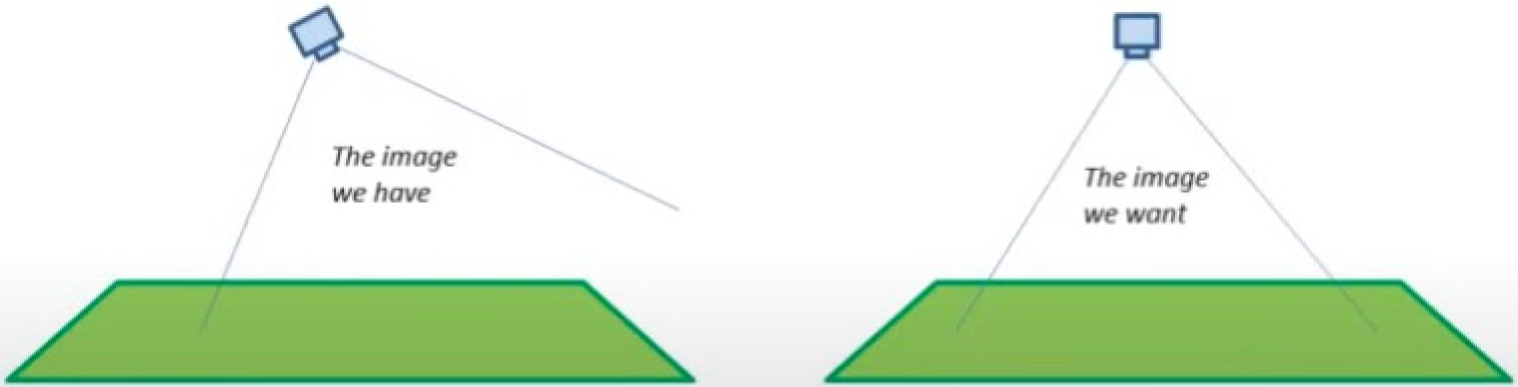
- Assume we have two images of the same scene from the same position but different camera angles
- The mapping between the two image planes is a homography
- We find a set of corresponding points between the left and the right image
  - Since the homography matrix has 8 degrees of freedom, we need at least 4 corresponding point pairs
  - We solve for the homography matrix using least squares fitting
- We then apply the homography transform to one image, to map it into the plane of the other image



From
http://www.cs.toronto.edu/~jepson/csc2503/tutorials/homography.pdf

Example Application: Generating an Orthophoto

- An "orthophoto" is an aerial photograph geometrically corrected such that the scale is uniform
  - Like a map, an orthophotograph can be used to measure true distances
- Essentially, we want to take the image taken by a camera at some off-axis angle, and transform it as if it were taken looking straight down



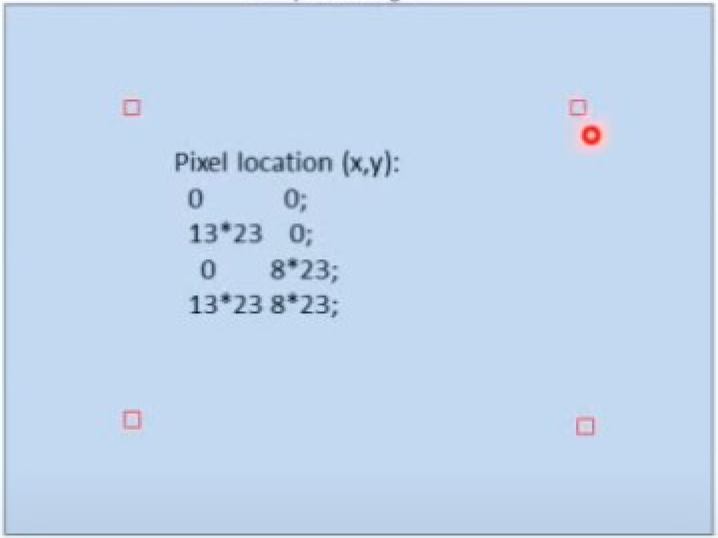
- One way to calculate the transform is to find some known "control points" in the input image, and specify where those points should appear in the output image

Example

- Transform the image as if it were taken from a camera perpendicular to the wall

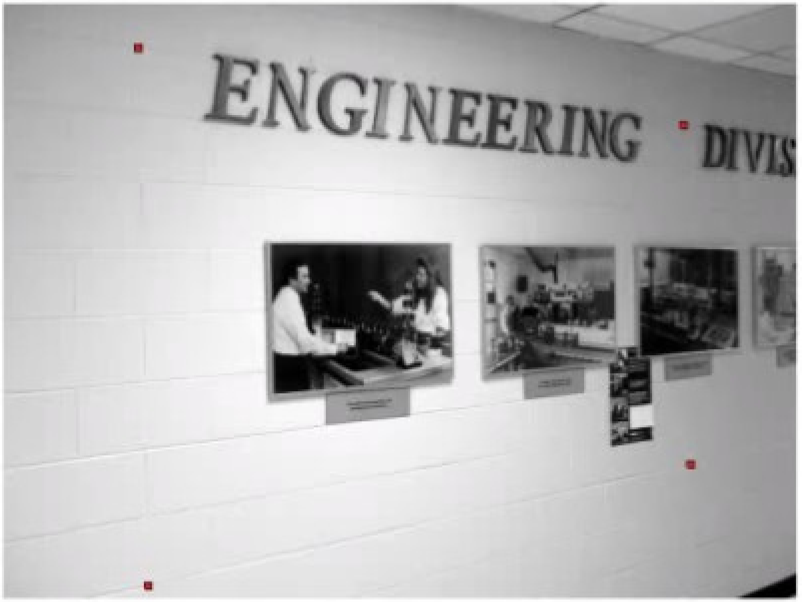
Input image

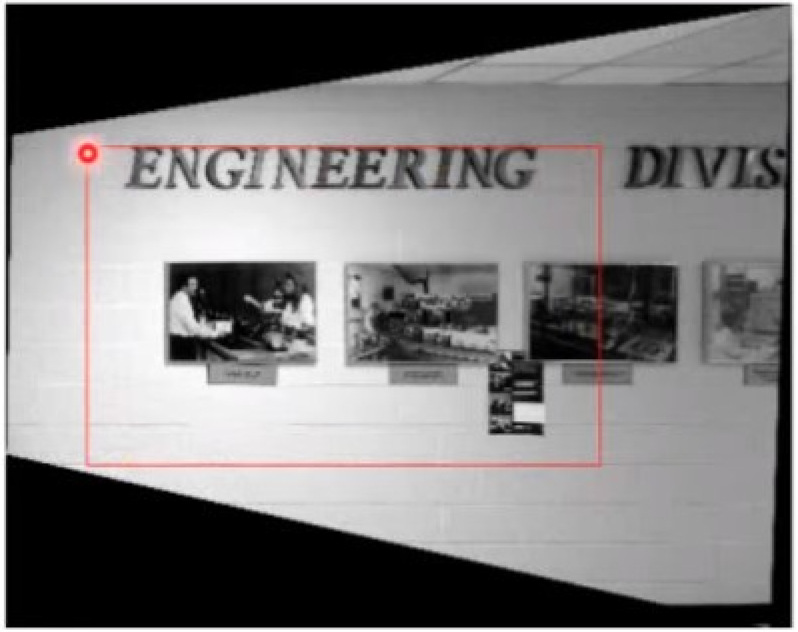
Output image



- For control points, we use four brick corners that define a rectangle of known size
  - The rectangle is 8 bricks high and 13 bricks wide
  - Each brick is about 23 cm, so rectangle is 8*23=184 cm high and 13*23=299 cm wide
- We'll specify the corresponding rectangle in the output image
  - Use scale of 1 cm = 1 pixel
  - Put upper left corner at 0,0

Results





- Note – the upper left corner is not at (0,0) in the output image
- `imtransform` automatically enlarges the output image so that it contains the entire transformed image (you can override this)
- To see the location of the output image in the output XY space, use
  - `[ITrans, xdata, ydata] = imtransform(Iin1,Tform1);`

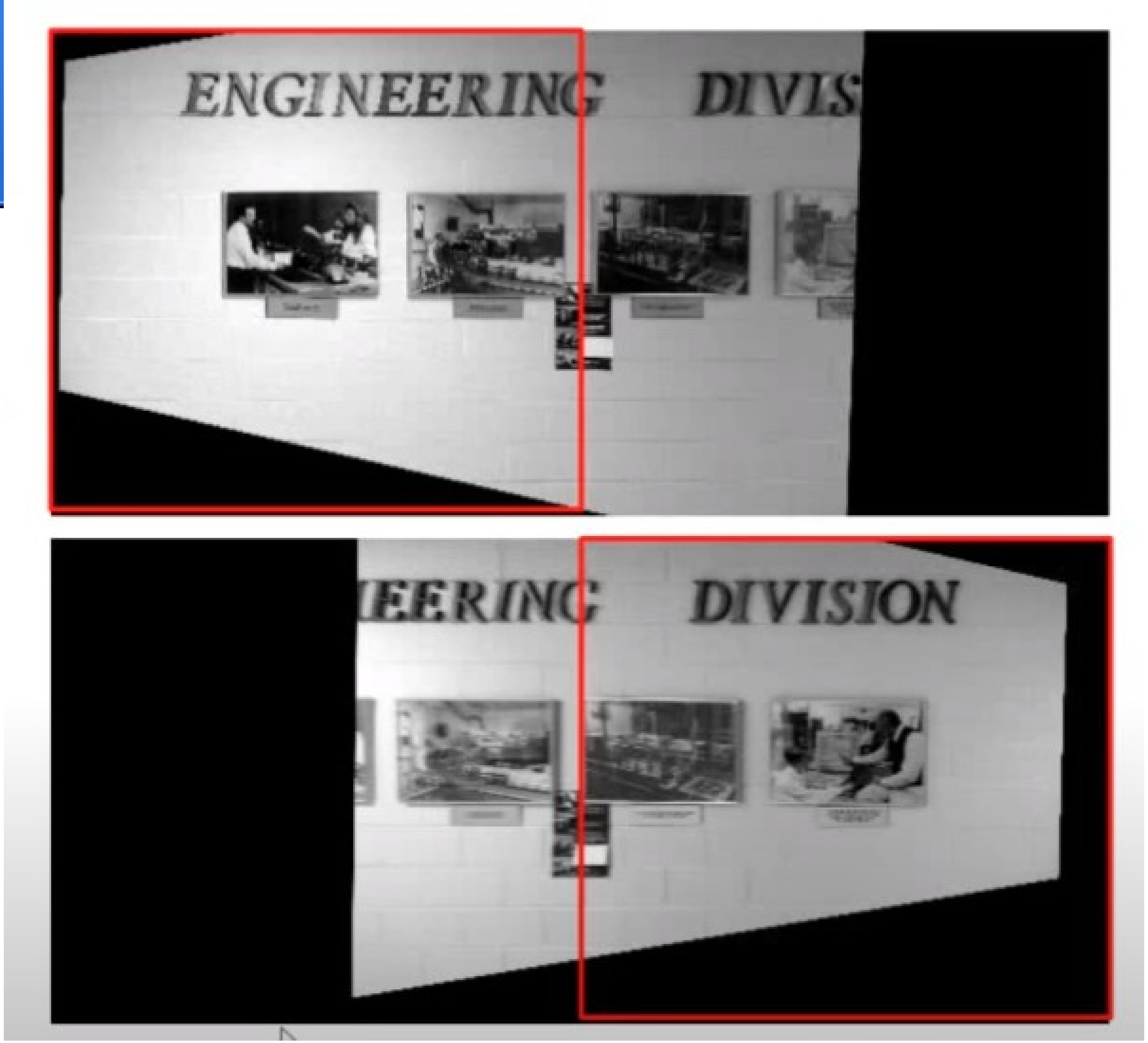
Mapping two images of the same scene

- We have a second image of the wall

Repeat process to transform image to an orthophoto

Then merge the two images

But to get them to merge properly, explicitly set output coordinates for both images


ENGINEERING DIVIS
IEERING DIVISION

Please watch the lecture by Prof. William Hoff

- Lecture