# Projective Transforms & RANSAC

Christopher Moloney

CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler, Yannis Gkioulekas

TCIG
Toronto Computational Imaging Group

# Logistics

- A3 due on Friday!

# A3 Q1 Clarification #126

David Lindell STAFF
3 days ago in General

UNPIN STAR WATCHING 117 VIEWS

Hi all,

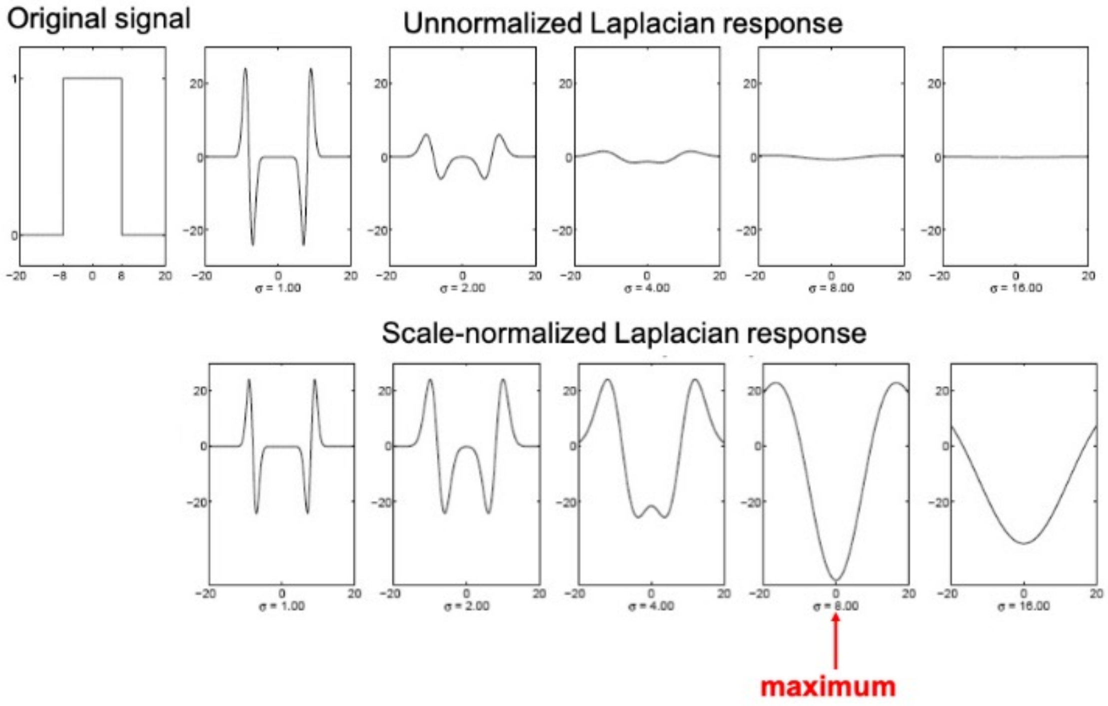
For the Laplacian of Gaussian filter in A3 Question 1, you should be sure to use the *normalized* version of this filter so that the response at different scales is consistent. See the below image, which illustrates the difference. More information can be found on these slides (http://cubs.buffalo.edu/~inwogu/teaching/Coursepage573_fa14/lectures/12LocalFeatures_contd.pdf)

More specifically, you should use this equation, but add a scale factor of $\sigma^2$ to compute the normalized Laplacian of Gaussian filter (this results in a factor of $1/\sigma^2$ in the denominator at the front of the expression).

• Laplacian of Gaussian: We mentioned it for edge detection

$$\nabla_g^2(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) \exp -\frac{x^2 + y^2}{2\sigma^2}$$

## Effect of scale normalization



# Overview

- Motivation: image matching, panoramas
- What is a homography?
- computing homographies
- Random sample consensus (RANSAC)

# Recap from last time

- Now we know how to extract scale and rotation invariant features
- We even know how to match features across images
- Can we use this to find Waldo in an even more sneaky scenario?


Someone takes a (weird) picture of him!

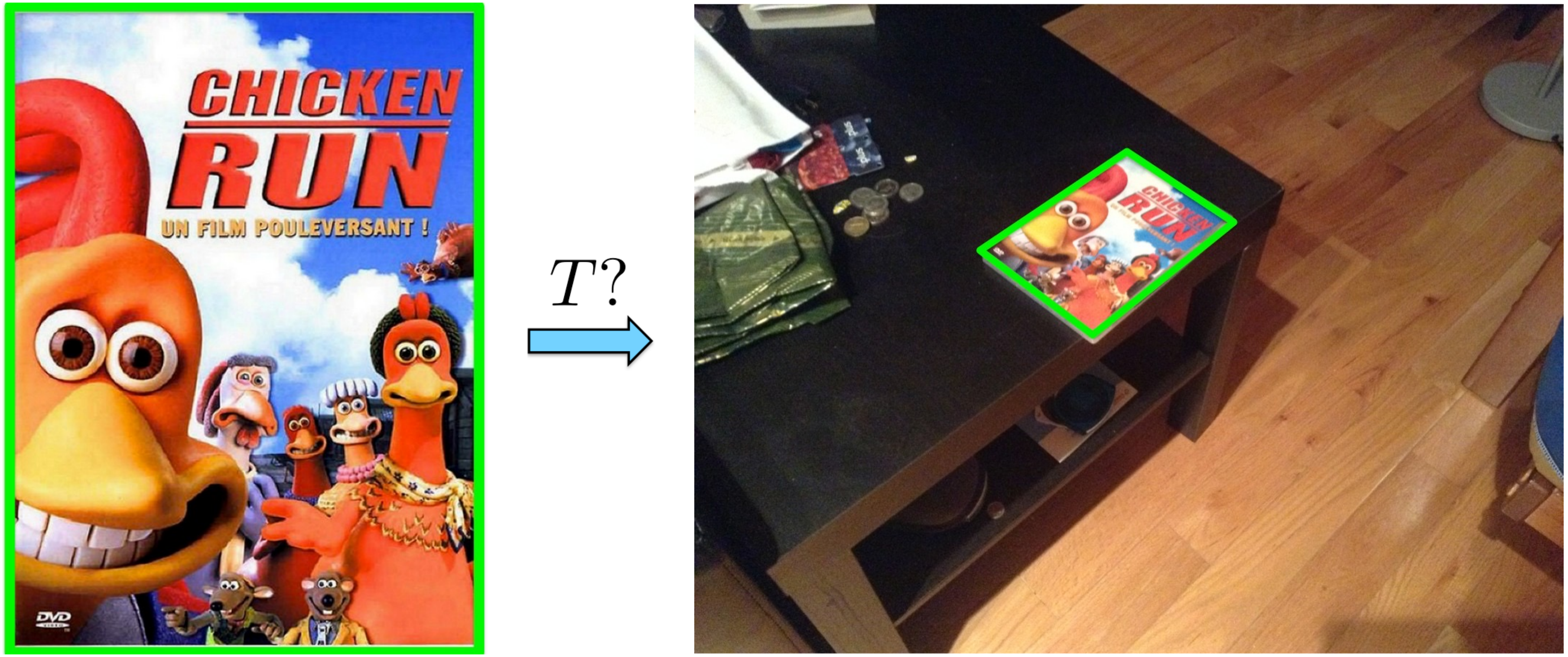
template

# Find My DVD!

- More interesting: If we have DVD covers (e.g., from Amazon), can we  match them to DVDs in real scenes?


# What Transformation Happened To My DVD?

- Rectangle goes to a parallelogram



# Affine Transformations

Affine transformations are combinations of
- Linear transformations, and translations

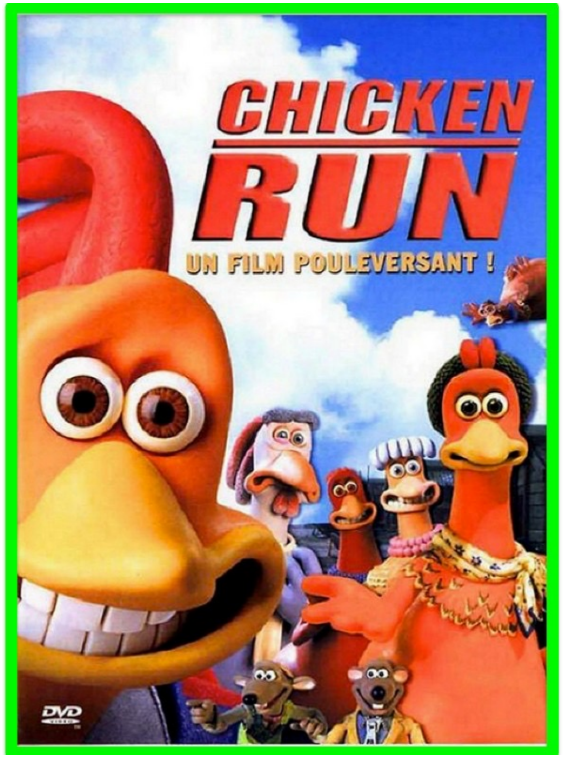
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition
- Rectangles go to parallelograms

[Source: N. Snavely, slide credit: R. Urtasun]

# What Transformation Happened To My DVD?

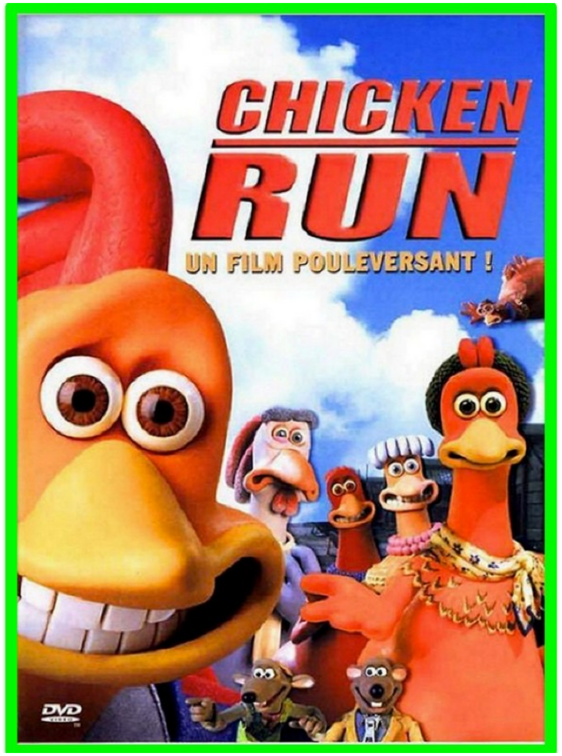
- Is this an affine transformation?




$T?$

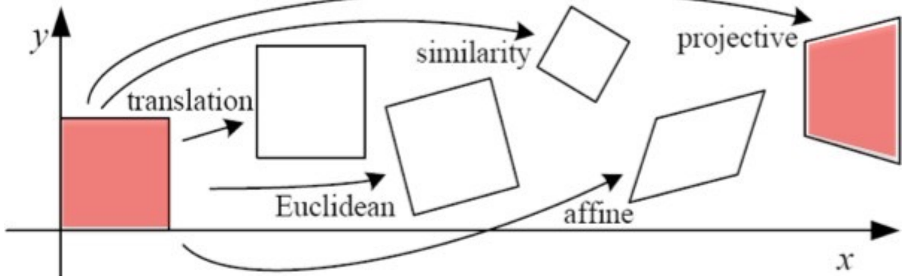
# What Transformation Happened To My DVD?

- Actually, a rectangle, maps to quadrilateral




$T?$

# 2D Image Transformations




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

- These transformations are a nested set of groups
- Closed under composition and inverse is a member

[source: R. Szeliski]

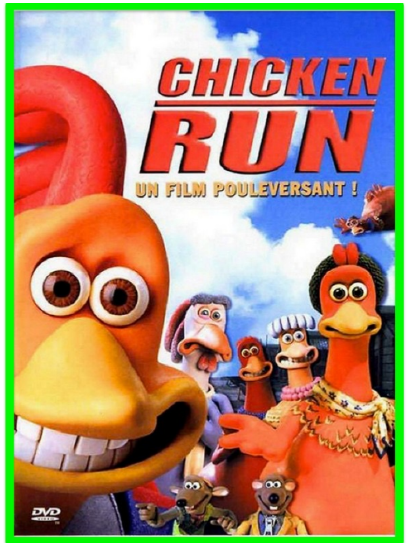
# Projective Transformations

- Homography $$a\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

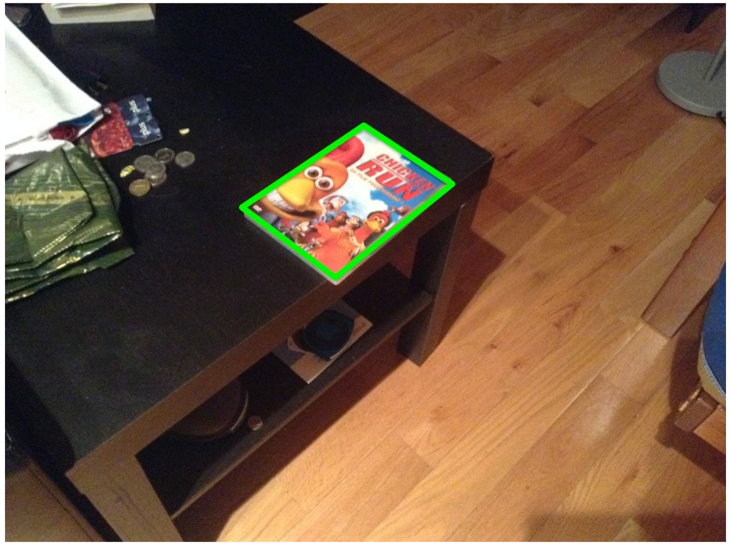
- Properties:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Rectangle goes to quadrilateral
  - Affine transformation is a special case, where $g = h = 0$ and $i = 1$

[Source: N. Snavely, slide credit: R. Urtasun]

# What Transformation Happened To My DVD?




$T?$




For planar objects:

- Viewpoint change for planar objects is a homography
- Affine transformation approximates viewpoint change for planar objects that are far away from camera

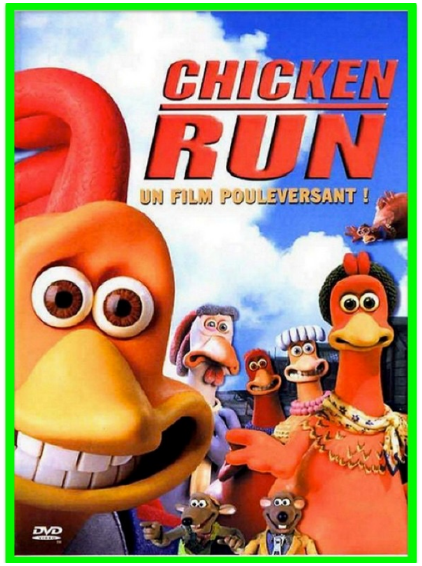
# Homography

- Why should I care about homography?
- Now that I care, how should I estimate it?
- I want to understand the geometry behind homography. That is, why aren't parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?

# Homography

- Why should I care about homography? Let’s answer this first
- Now that I care, how should I estimate it?
- I want to understand the geometry behind homography. That is, why aren’t parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?

# Homography




$T?$



- Why do we need homography? Can't we just assume that the transformation  is affine? The approximation on the right looks pretty decent to me...

# Homography


- Why do we need homography? Can't we just assume that the transformation is affine? The approximation on the right looks pretty decent to me...
- That's right. If I want to detect (match) an object in a new viewpoint, an affine transformation is a relatively decent approximation

# Homography


- Why do we need homography? Can't we just assume that the transformation is affine? The approximation on the right looks pretty decent to me…
- That's right. If I want to detect (match) an object in a new viewpoint, an affine transformation is a relatively decent approximation
- But for some applications I want to be more accurate. Which?

# Application 1: a Little Bit of CSI

- Tom Cruise is taking an exam on Monday

# Application 1: a Little Bit of CSI

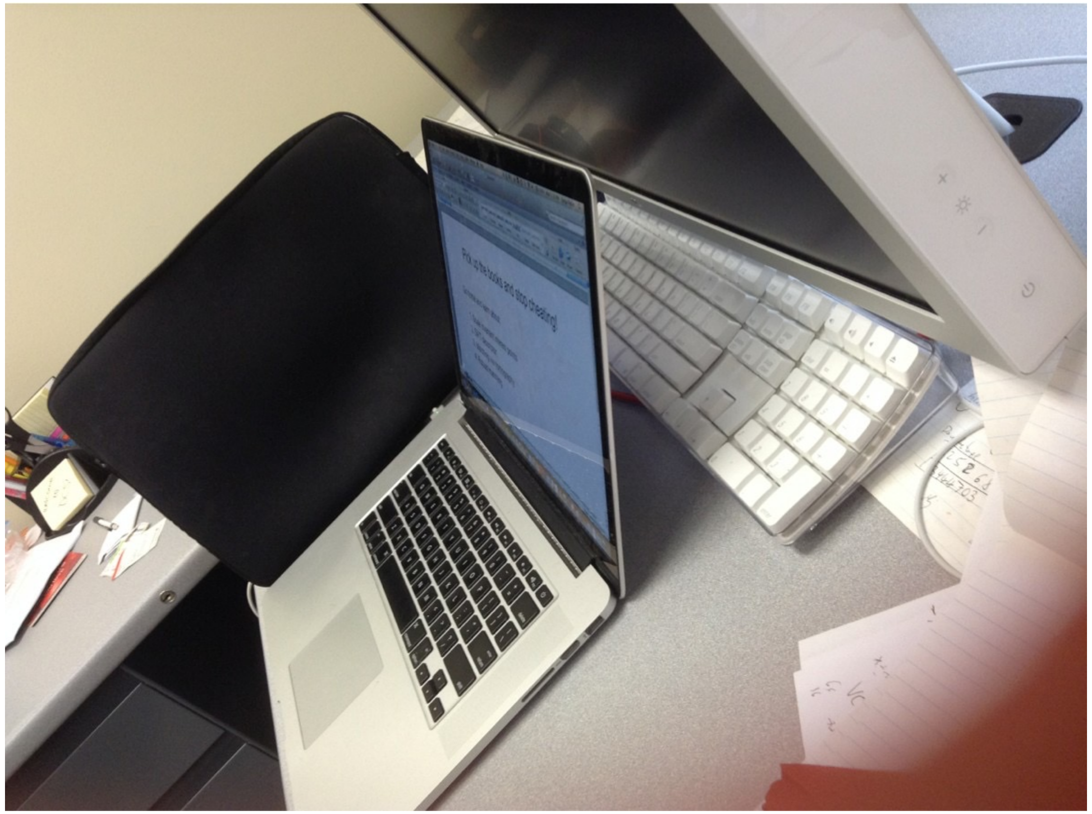
exam is here

- The professor keeps the exams in this office

# Application 1: a Little Bit of CSI

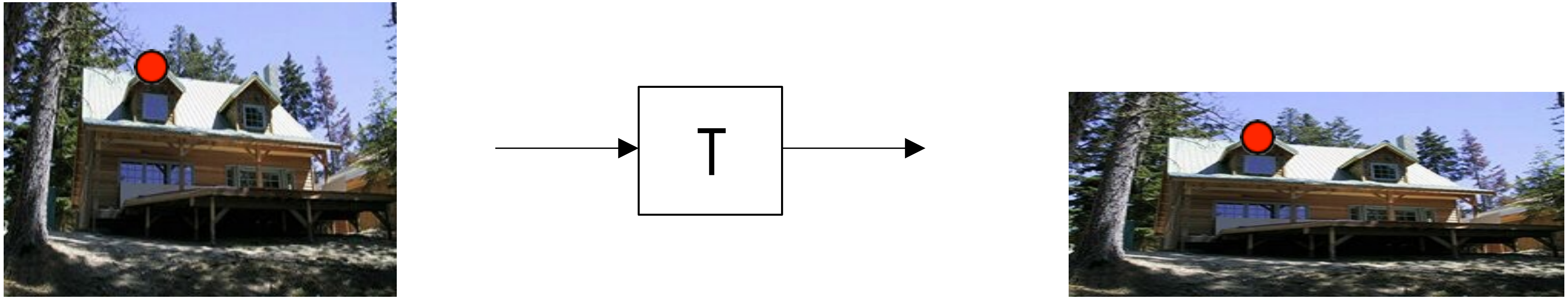

- He enters (without permission) and takes a picture of the laptop screen

# Application 1: a Little Bit of CSI



- His picture turns out to not be from a viewpoint he was shooting for (it's  difficult to take pictures while hanging)
- Can he still read the exam?

# Warping an Image with a Global Transformation

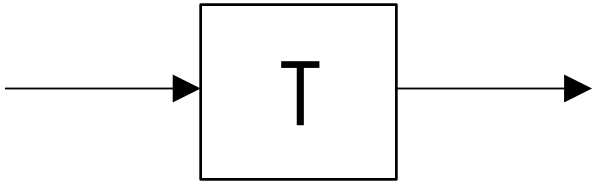



p= (x,y) P'= (x',y')

- Transformation *T* is a coordinate-changing machine:
  - $[x', y'] = T(x, y)$

[Source: N. Snavely, slide credit: R. Urtasun]

# Warping an Image with a Global Transformation




$p = (x, y)$ $P' = (x', y')$

- Transformation $T$ is a coordinate-changing machine:
    - $[x', y'] = T(x, y)$
- What does it mean that T is global?
    - Is the same for any point p
    - Can be described by just a few numbers (parameters)

[Source: N. Snavely, slide credit: R. Urtasun]

# Warping an Image with a Global Transformation

- Example of warping for different transformations:

what happens
to a pixel (x, y)?

translation

# Warping an Image with a Global Transformation

- Example of warping for different transformations:

what happens
to a pixel (x, y)?

translation

rotation

# Warping an Image with a Global Transformation

- Example of warping for different transformations:

what happens
to a pixel (x, y)?

translation

rotation

aspect

# Warping an Image with a Global Transformation

- Example of warping for different transformations:

what happens
to a pixel (x, y)?

translation

rotation

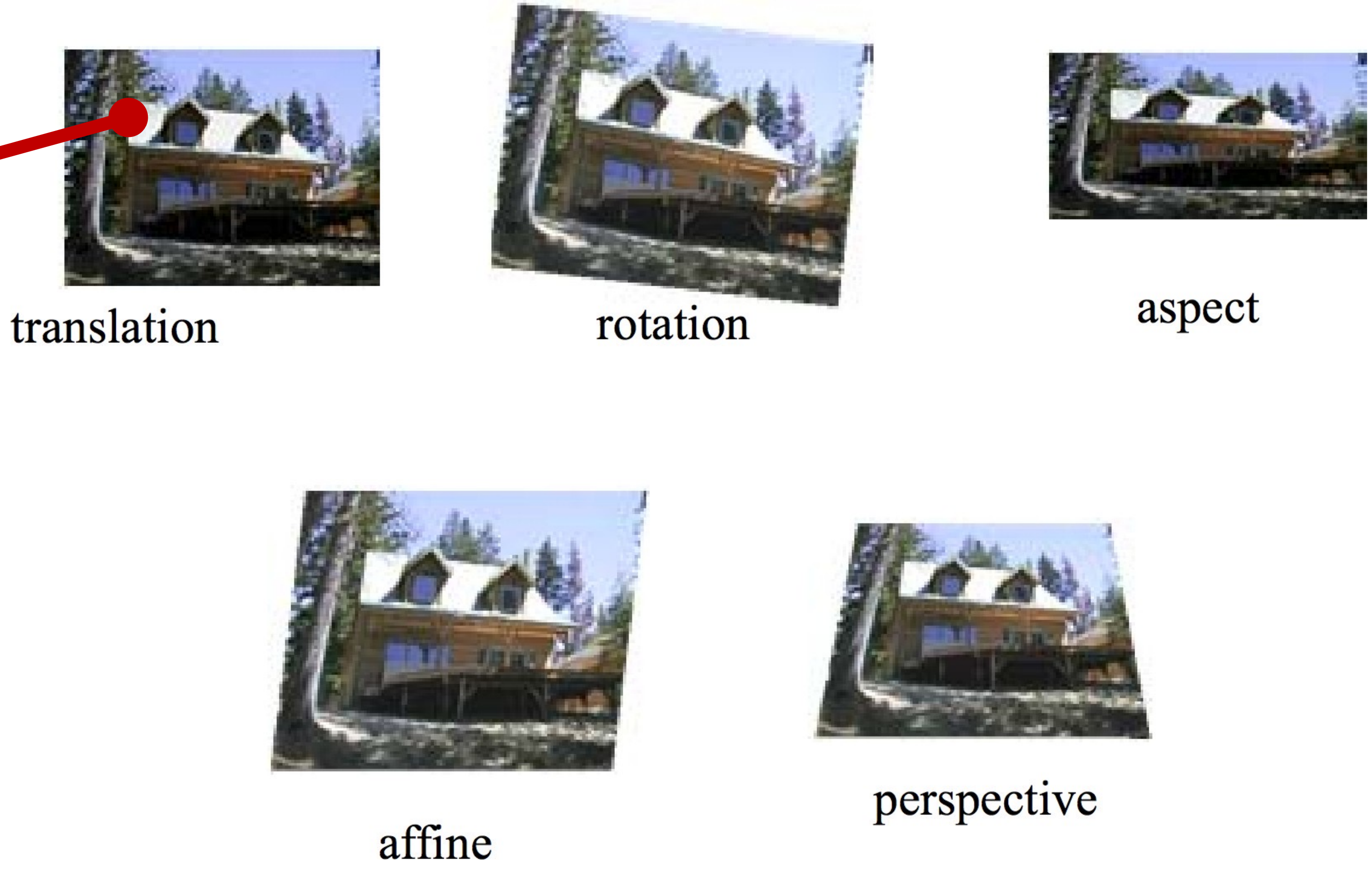
aspect

affine

# Warping an Image with a Global Transformation

- Example of warping for different transformations:

what happens
to a pixel (x, y)?



# Forward and Inverse Warping

- Forward Warping: Send each pixel $f(x)$ to its corresponding location $(x', y') = T(x, y)$ in $g(x', y')$

**procedure** *forwardWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

# Forward and Inverse Warping

- Forward Warping: Send each pixel $f(x)$ to its corresponding location $(x', y') = T(x, y)$ in $g(x', y')$

**procedure** *forwardWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

- May leave some holes in the target image.

# Forward and Inverse Warping

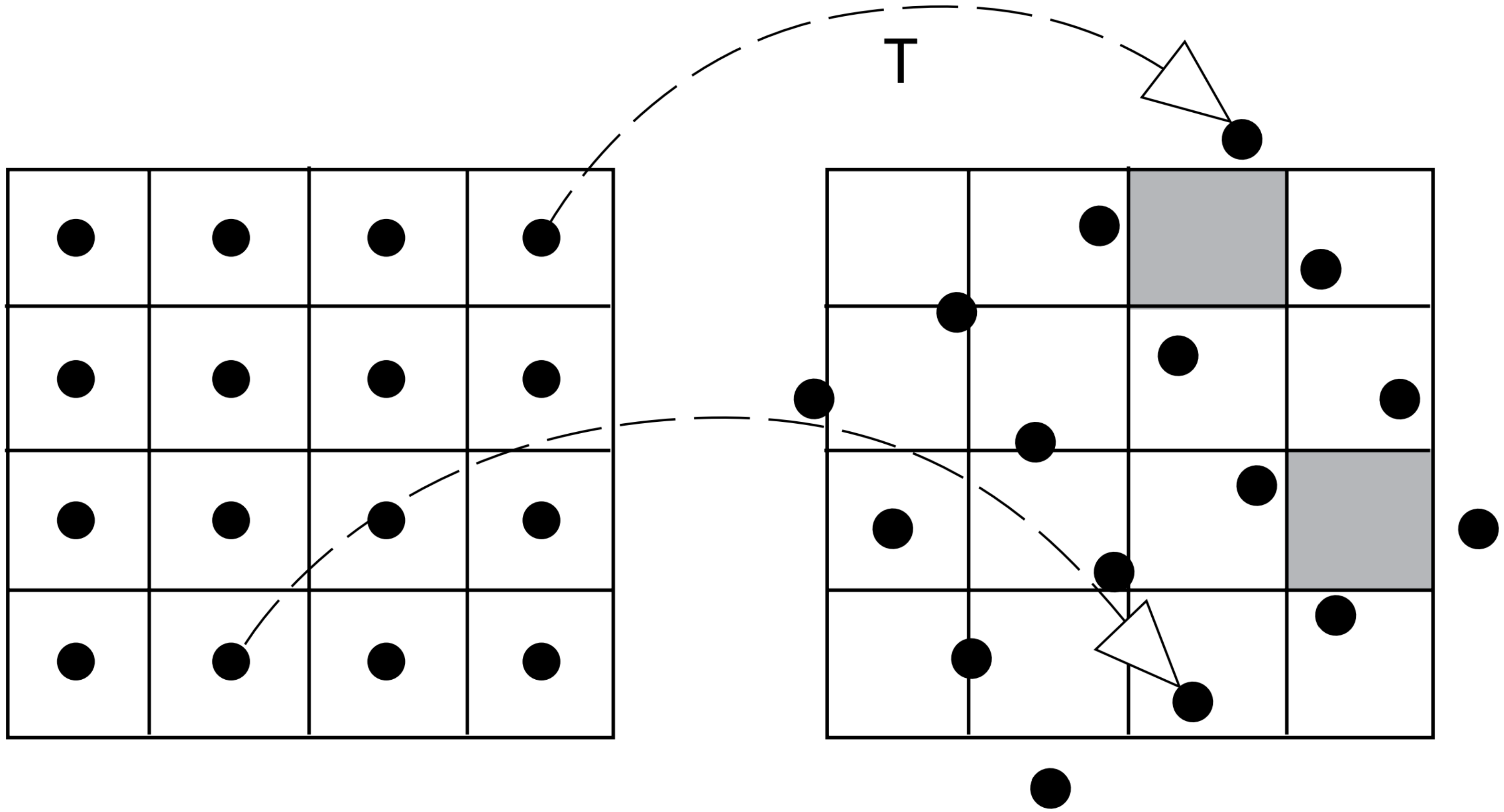



Figure: Loren Arthur Schwarz

# Forward and Inverse Warping

- Forward Warping: Send each pixel $f(x)$ to its corresponding location $(x', y') = T(x, y)$ in $g(x', y')$

**procedure** *forwardWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

- May leave some holes in the target image.
- Inverse Warping: Each pixel at destination is sampled from original image

**procedure** *inverseWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:

For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

# Forward and Inverse Warping

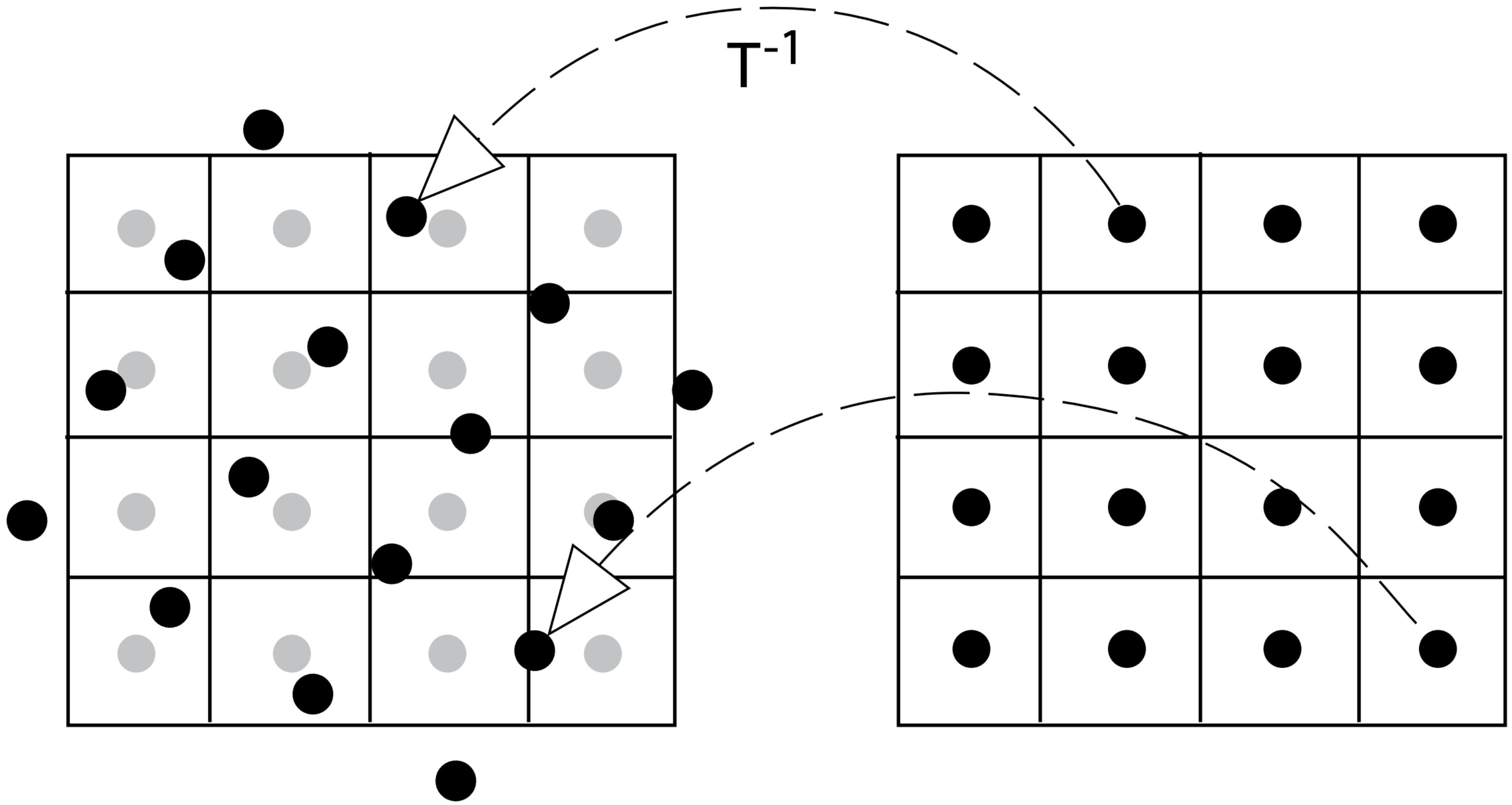
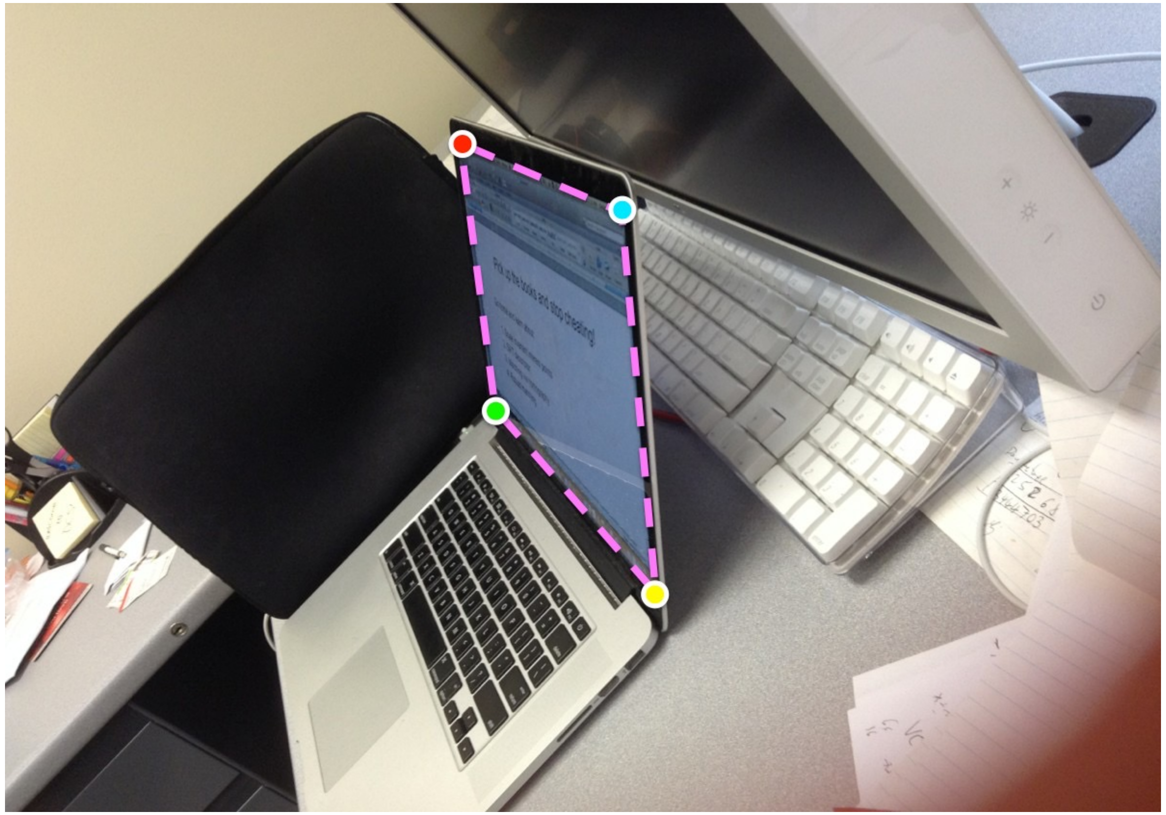


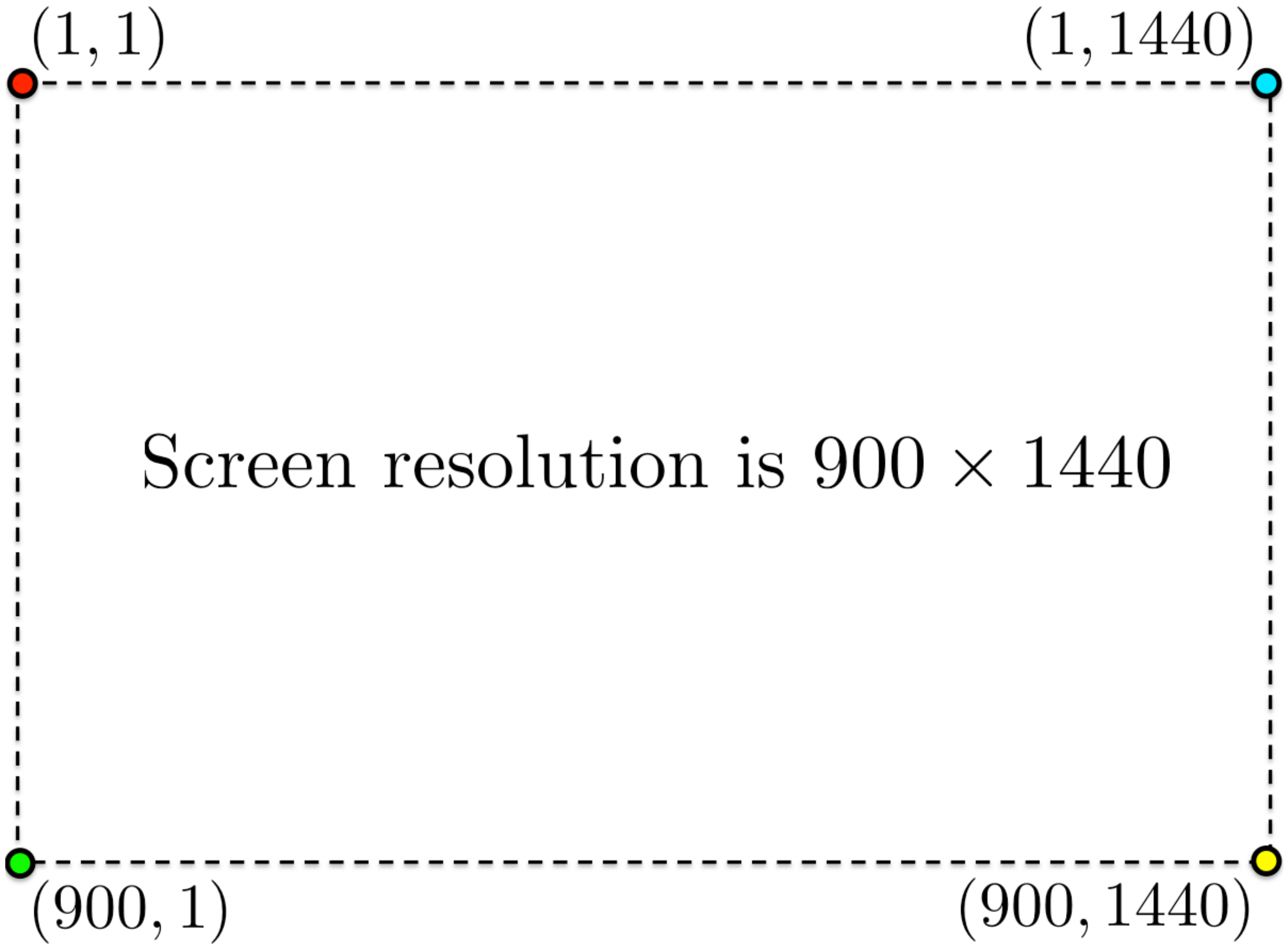

Figure: Loren Arthur Schwarz

# Application 1: a Little Bit of CSI



- We want to transform the picture (plane) inside these 4 points into a rectangle (laptop screen)

# Application 1: a Little Bit of CSI

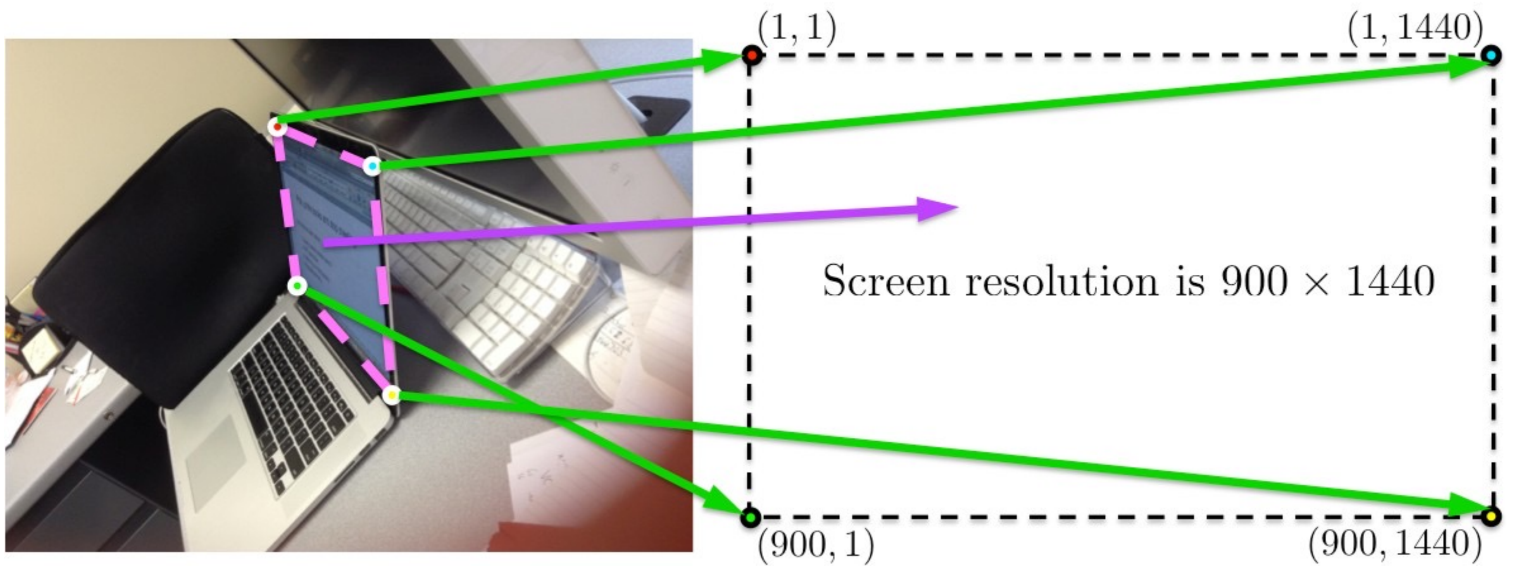



- We want it to look like this. How can we do this?

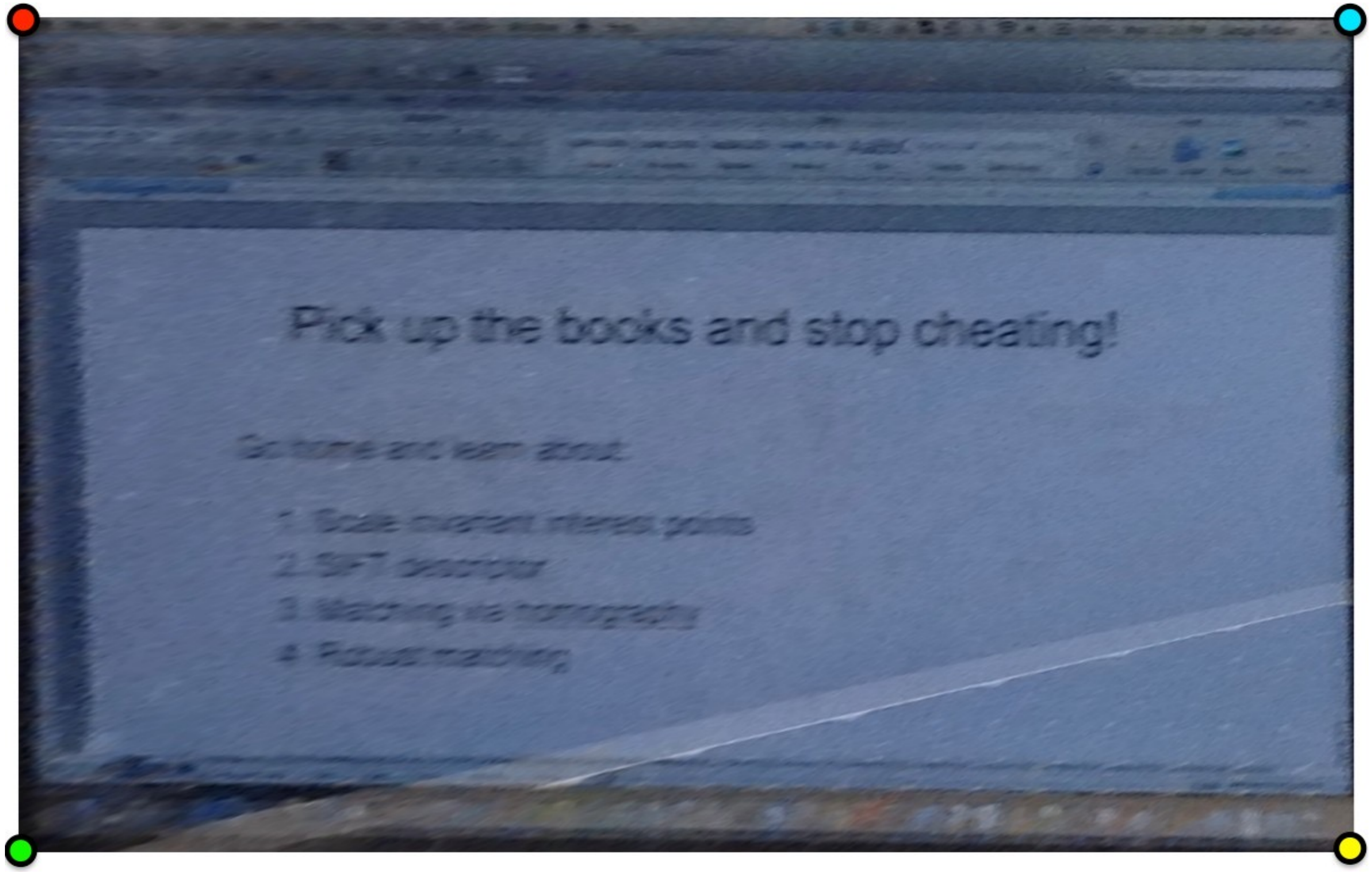
# Application 1: a Little Bit of CSI

- A transformation that maps a projective plane (a quadrilateral) to another projective plane (another quadrilateral, in this case a rectangle) is a homography

homography $H$

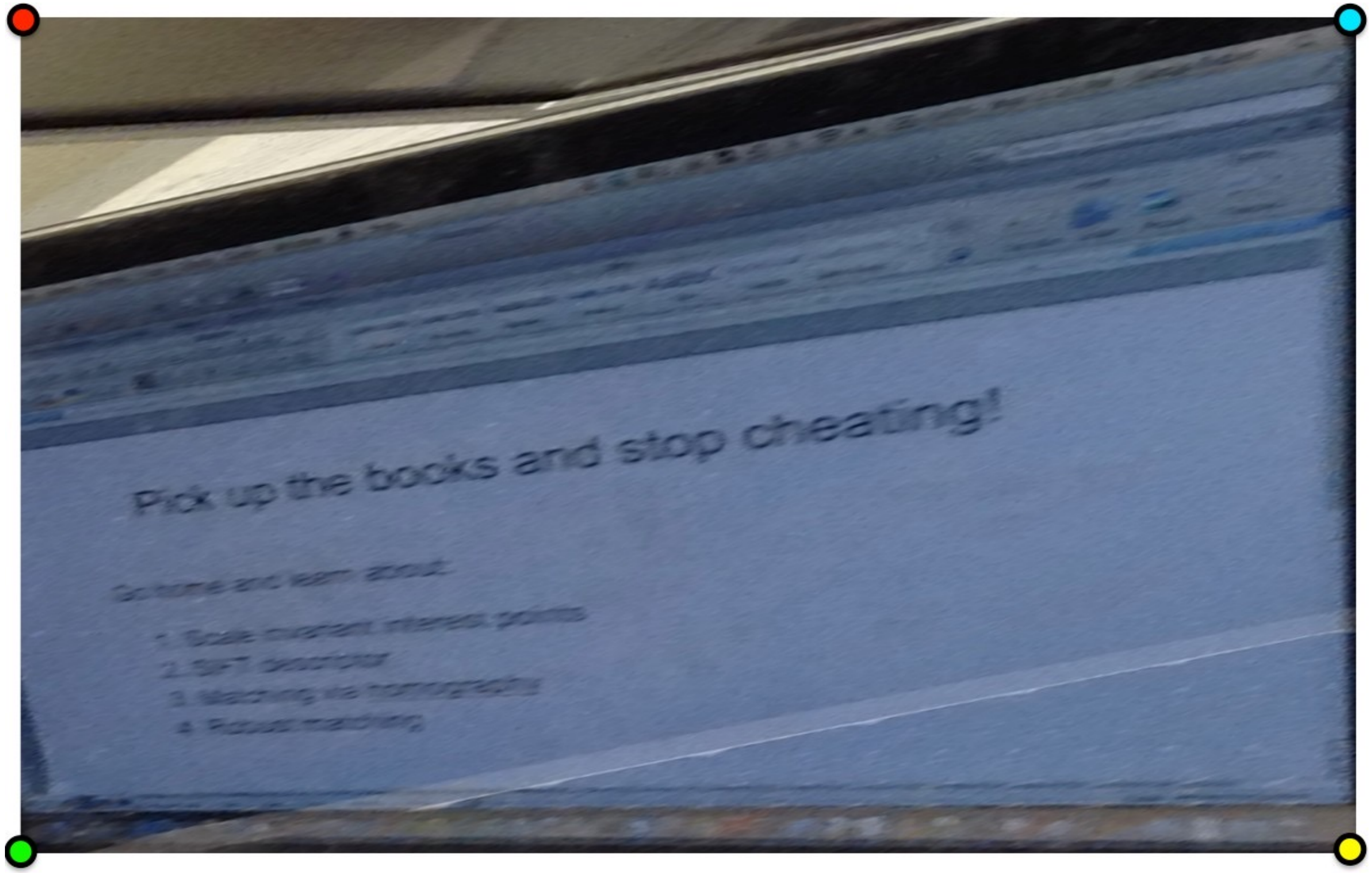


# Application 1: a Little Bit of CSI



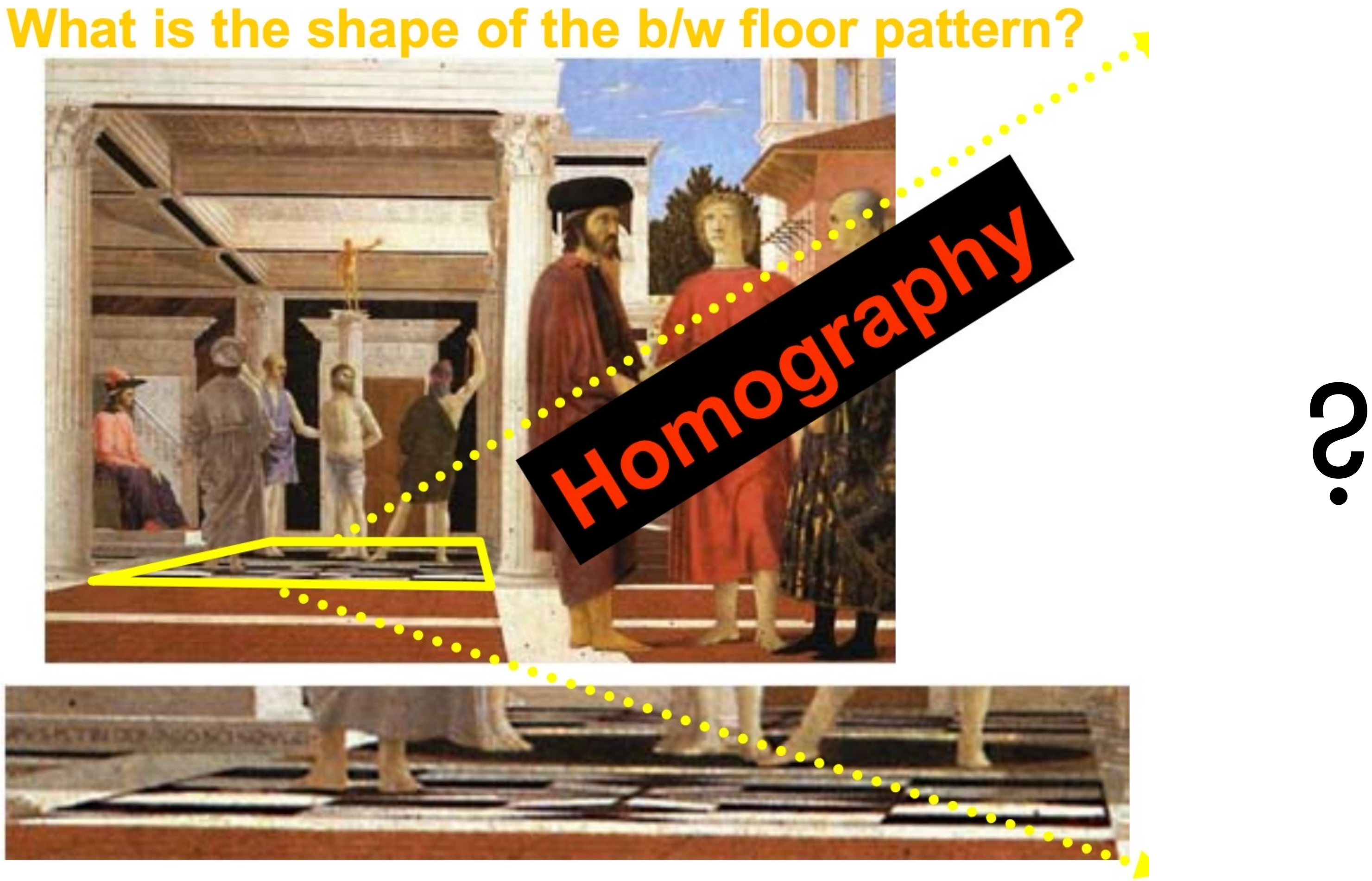

- If we compute the homography and warp the image according to it, we get this

# Application 1: a Little Bit of CSI




- If we used affine transformation instead, we'd get this. Would be even worse if our picture was taken closer to the laptop

# Application 1: a Little More of CSI

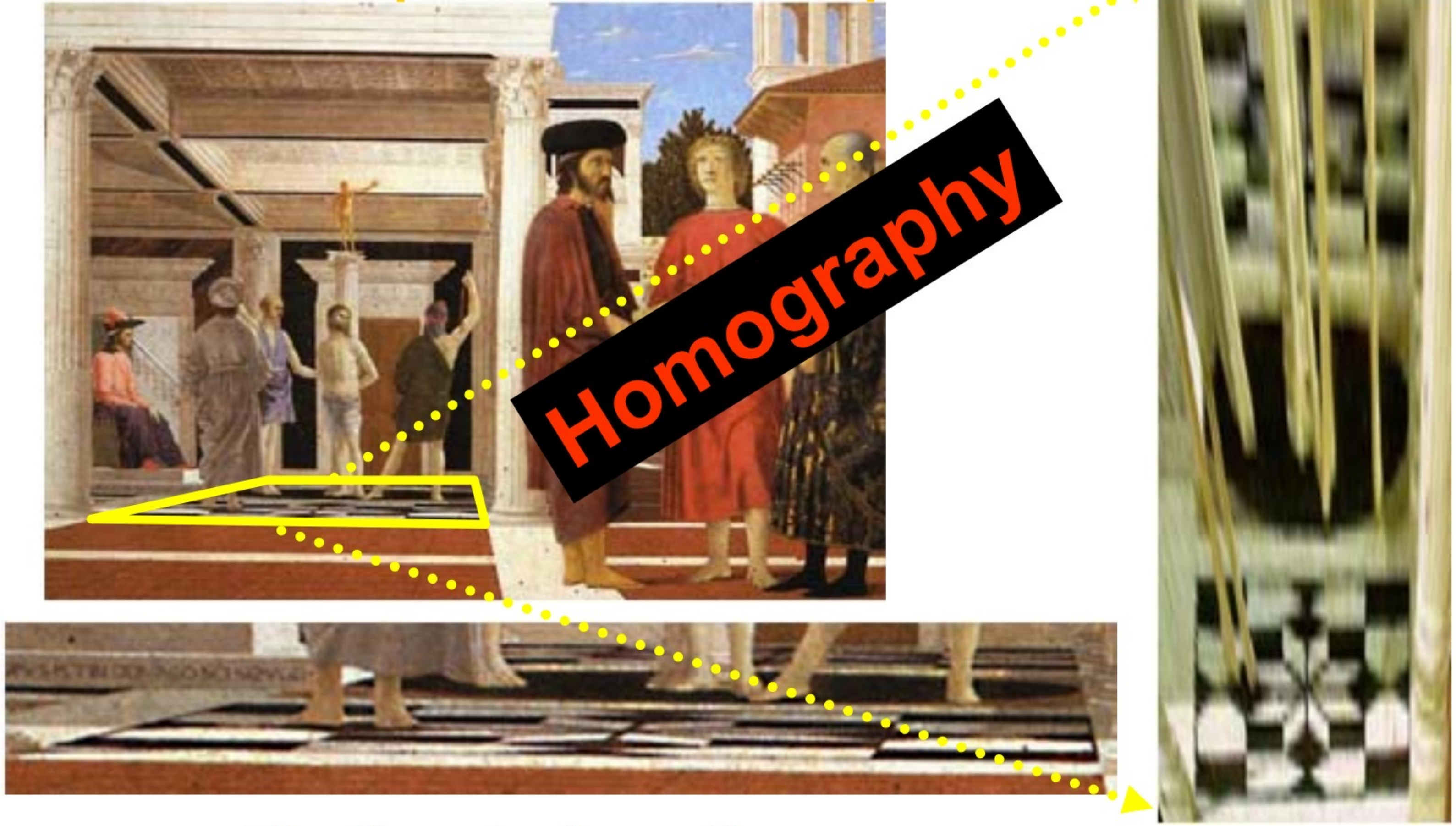



The floor (enlarged)

Slide from Antonio Criminisi

# Application 1: a Little More of CSI

What is the shape of the b/w floor pattern?



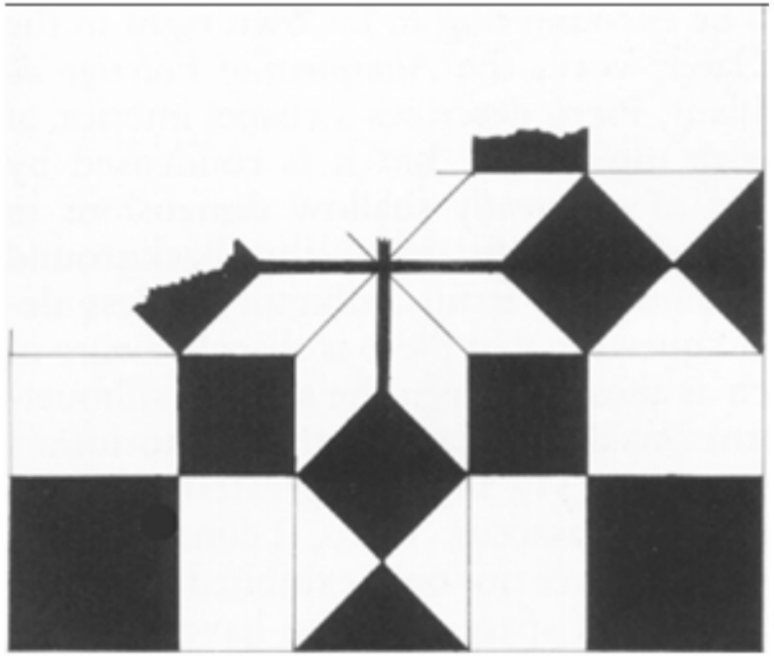

The floor (enlarged)

Automatically rectified floor

Slide from Antonio Criminisi

# Application 1: a Little More of CSI

Automatic rectification



**From Martin Kemp *The Science of Art***
***(manual reconstruction)***

Slide from Antonio Criminisi

# Application 1: a Little More of CSI

***St. Lucy Altarpiece,* D. Veneziano**

Slide from Criminisi

?

# Application 1: a Little More of CSI

***St. Lucy Altarpiece*, D. Veneziano**

Slide from Criminisi

**What is the (complicated) shape of the floor pattern?**

**Automatically rectified floor**

# Application 1: a Little More of CSI

**Automatic rectification**

**From Martin Kemp, *The Science of Art* *(manual reconstruction)***

Slide from Criminisi

# A weird drawing

Holbein, “The Ambassadors”

# A weird drawing

Holbein, “The Ambassadors”

What’s this???

# A weird drawing

Holbein, “The Ambassadors”

skull under anamorphic perspective

rectified view

# A weird drawing

Holbein, “The Ambassadors”

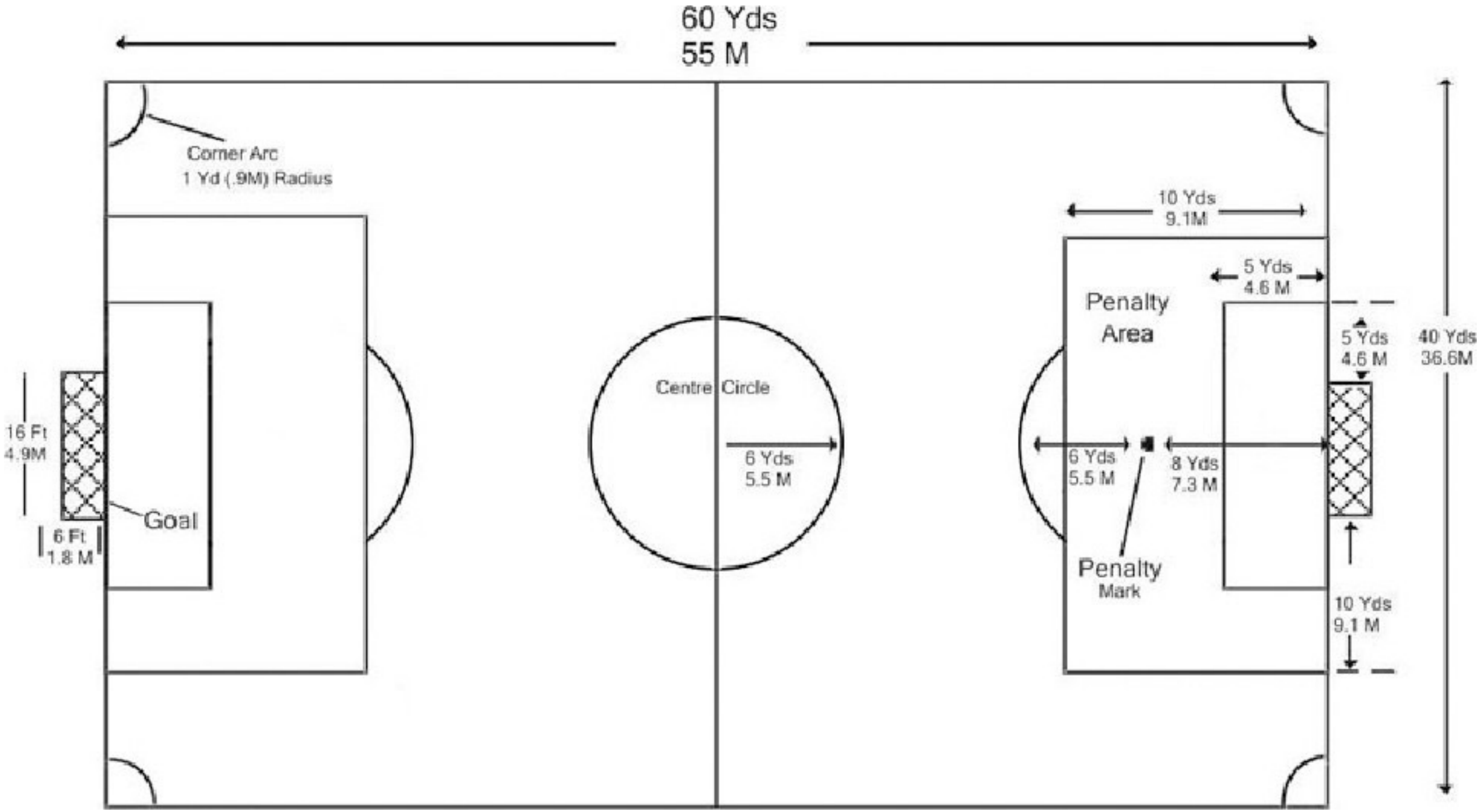
DIY: use a polished spoon to see the skull

# Application 2: How Much do Soccer Players Run?

# Application 2: How Much do Soccer Players Run?

- How many meters did this player run?

# Application 2: How Much do Soccer Players Run?

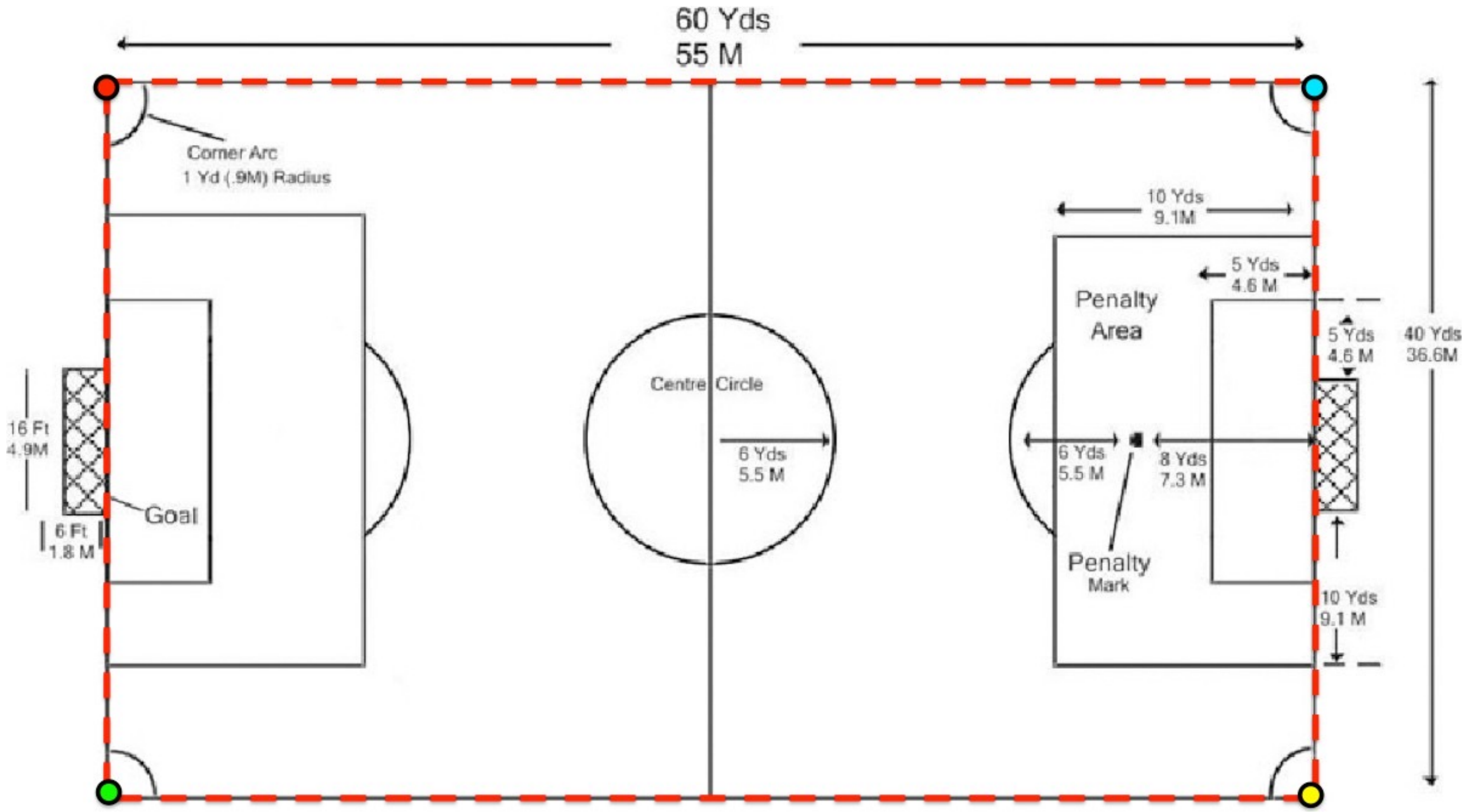



- Field is planar. We know its dimensions (look on Wikipedia).

# Application 2: How Much do Soccer Players Run?

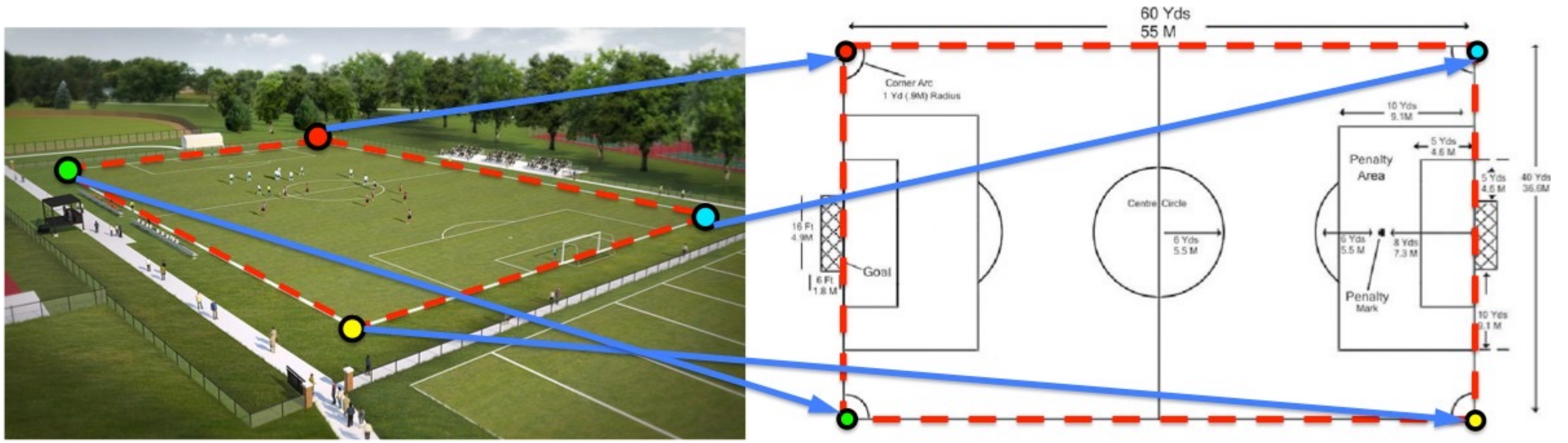
- Let’s take the 4 corner points of the field

# Application 2: How Much do Soccer Players Run?



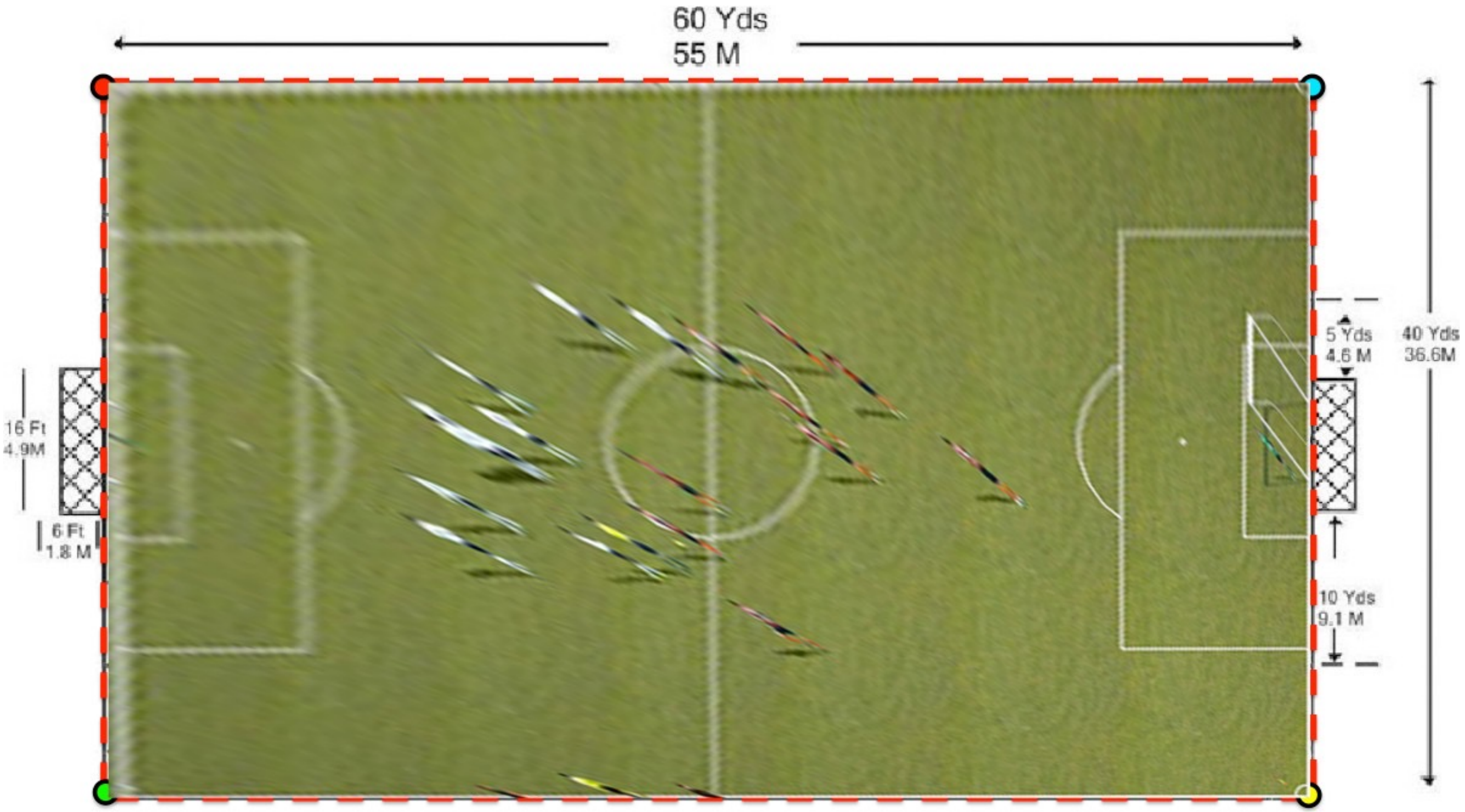

- We need to compute a homography that maps them to these 4 corners

# Application 2: How Much do Soccer Players Run?



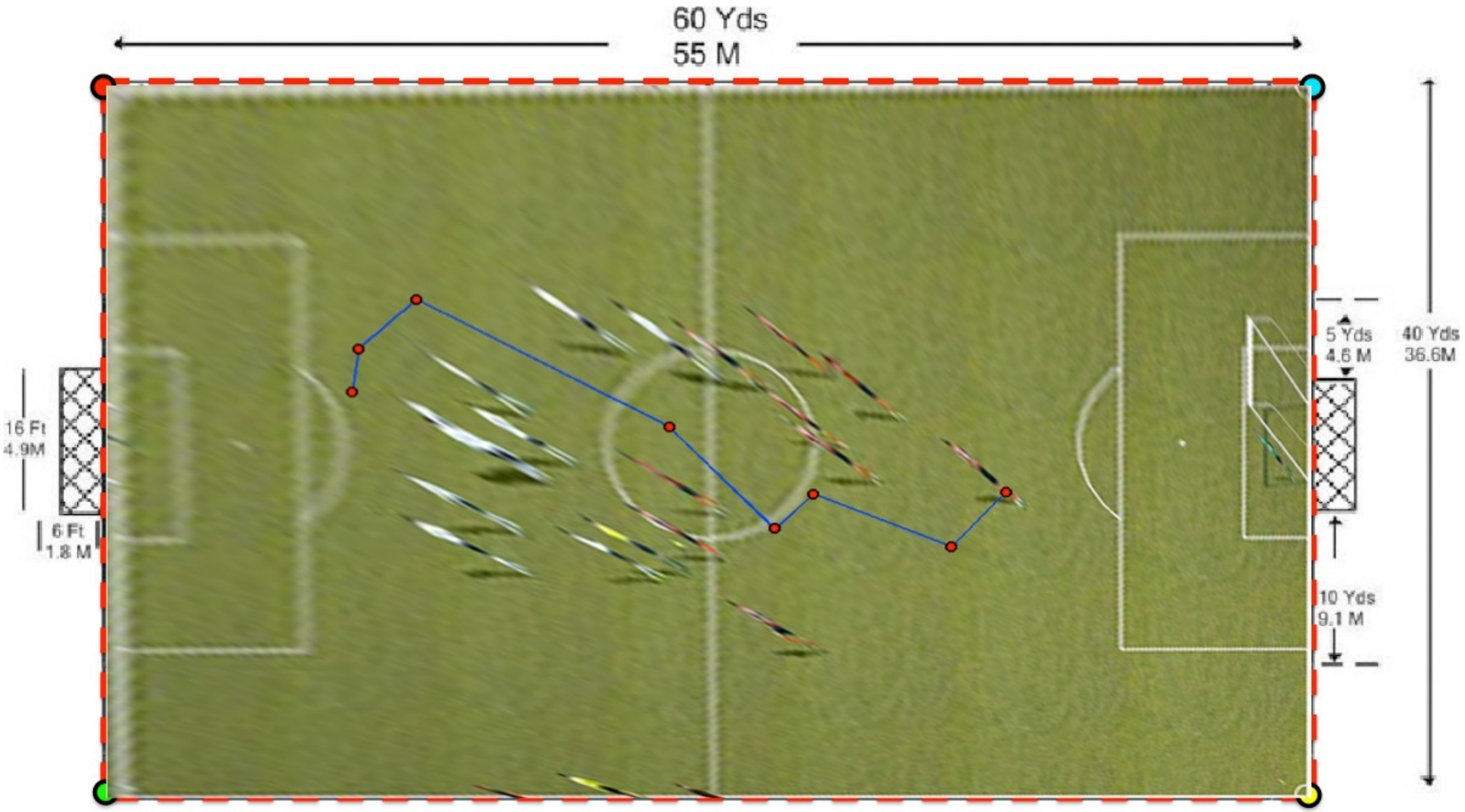

- We need to compute a homography that maps the 4 corners. Any other point from this plane (the field) also maps to the right with the same homography

# Application 2: How Much do Soccer Players Run?




- Nice. What happened to the players?

# Application 2: How Much do Soccer Players Run?

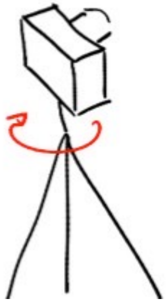



- We can now also transform the player's trajectory → and we have it in  meters!

# Application 2: How Much do Soccer Players Run?

- If we used affine transformation... Our estimations of running would not be accurate!

# Application 4: Panorama Stitching



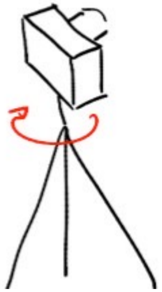
Take a tripod, rotate camera and take pictures

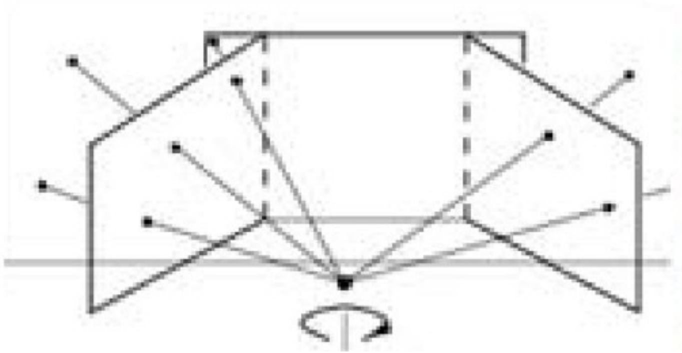
[Source: Fernando Flores-Mangas]

# Application 4: Panorama Stitching

[Source: Fernando Flores-Mangas]

# Application 4: Panorama Stitching





- Each pair of images is related by homography! If we also moved the camera, this wouldn't be true (next class)

[Source: Fernando Flores-Mangas]

# We can use homographies when...

- The scene is planar

- The scene is relatively far off or has small (relative) depth variation (i.e., scene is approximately planar

# We can use homographies when...

- The scene is captured under camera rotation only (no translation or pose change)

# Homography

- Why should I care about homography?
- Now that I care, how should I estimate it? Let’s do this now
- I want to understand the geometry behind homography. That is, why aren’t parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?

# Solving for Homographies

- Projective mapping between any two projection planes with the same centre of projection
- Let $(x_i, y_i)$ be a point on the reference (model) image, and $(x'_i, y'_i)$ its match in the test image
- A homography $H$ maps $(x_i, y_i)$ to $(x'_i, y'_i)$:

$$\begin{bmatrix} a x_i' \\ a y_i' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

# Solving for Homographies

- Projective mapping between any two projection planes with the same centre of projection
- Let $(x_i, y_i)$ be a point on the reference (model) image, and $(x'_i, y'_i)$ its match in the test image
- A homography $H$ maps $(x_i, y_i)$ to $(x'_i, y'_i)$:

$$\begin{bmatrix} a x_i' \\ a y_i' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- Expand matrix multiplication

$$\begin{aligned} a x_i' &= h_{00} x_i + h_{01} y_i + h_{02} \\ a y_i' &= h_{10} x_i + h_{11} y_i + h_{12} \\ a &= h_{20} x_i + h_{21} y_i + h_{22} \end{aligned}$$

# Solving for Homographies

- Projective mapping between any two projection planes with the same centre of projection
- Let $(x_i, y_i)$ be a point on the reference (model) image, and $(x'_i, y'_i)$ its match in the test image
- A homography $H$ maps $(x_i, y_i)$ to $(x'_i, y'_i)$:

$$\begin{bmatrix} a x_i' \\ a y_i' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- Expand matrix multiplication

$$\begin{aligned} a x_i' &= h_{00} x_i + h_{01} y_i + h_{02} \\ a y_i' &= h_{10} x_i + h_{11} y_i + h_{12} \\ a &= h_{20} x_i + h_{21} y_i + h_{22} \end{aligned}$$

- Divide out scale factor

$$x_i' = \frac{h_{00} x_i + h_{01} y_i + h_{02}}{h_{20} x_i + h_{21} y_i + h_{22}} \qquad y_i' = \frac{h_{10} x_i + h_{11} y_i + h_{12}}{h_{20} x_i + h_{21} y_i + h_{22}}$$

# Solving for Homographies

- Projective mapping between any two projection planes with the same centre of projection
- Let $(x_i, y_i)$ be a point on the reference (model) image, and $(x'_i, y'_i)$ its match in the test image
- A homography $H$ maps $(x_i, y_i)$ to $(x'_i, y'_i)$:

$$\begin{bmatrix} a{x_i}' \\ a{y_i}' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- Expand matrix multiplication

$$\begin{aligned} a{x_i}' &= h_{00}x_i + h_{01}y_i + h_{02} \\ a{y_i}' &= h_{10}x_i + h_{11}y_i + h_{12} \\ a &= h_{20}x_i + h_{21}y_i + h_{22} \end{aligned}$$

Can I rewrite this into a linear system in h?

- Divide out scale factor

$${x_i}' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \qquad {y_i}' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

# Solving for Homographies

- From: $$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \qquad y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

- We can easily get this:

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

# Solving for Homographies

- From: $$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \qquad y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

- We can easily get this:

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

- Rewriting it a little:

$$h_{00}x_i + h_{01}y_i + h_{02} - x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = 0$$
$$h_{10}x_i + h_{11}y_i + h_{12} - y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = 0$$

# Solving for Homographies

- We can re-write these equations:

$$h_{00}x_i + h_{01}y_i + h_{02} \; - x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = 0$$

$$h_{10}x_i + h_{11}y_i + h_{12} - \; y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = 0$$

- As a linear system:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

# Exact case

- If we have exactly 4 matches, this gives 8 equations (how many columns?)

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ 2n\text{x}9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ 2n}}$$

- Are the columns linearly dependent or independent?

# Exact case

- If we have exactly 4 matches, this gives 8 equations (how many columns?)

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ 2n\text{x}9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ 2n}}$$

- Are the columns linearly dependent or independent?
- Is there a null space?

# Exact case

- If we have exactly 4 matches, this gives 8 equations (how many columns?)

$$
\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ \text{2nx9}}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ \text{2n}}}
$$

- Are the columns linearly dependent or independent?
- Is there a null space?
- How does this relate to the homography?

# Solving for Homographies

- Taking all matches into account:

$$
\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ \text{2nx9}}}
\underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}}
= \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ \text{2n}}}
$$

# Solving for Homographies

- Taking all matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ \text{2nx9}}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ \text{2n}}}$$

- Can be written as a least squares problem $\min_h ||Ah||_2^2$

# Solving for Homographies

- Taking all matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ \text{2nx9}}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ \text{2n}}}$$

- Can be written as a least squares problem $\min_h ||Ah||_2^2$

- Can we use the Moore-Penrose pseudo inverse here?

# Solving for Homographies

- Taking all matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\boldsymbol{A} \\ \text{2nx9}}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\boldsymbol{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\substack{\boldsymbol{0} \\ \text{2n}}}$$

- Can be written as a least squares problem $\min_h ||Ah||_2^2$

- But this is underdetermined, (8 degrees of freedom) how do we account for the unknown scale factor?

# Solving for Homographies pt 2

- we solve the *homogeneous* least squares problem

$$\min_h E = \|Ah\|_2^2$$
$$\text{s.t. } \|h\|_2 = 1$$

- Solve with eigenvalue decomposition of $A^T A$ or SVD

$\boldsymbol{A}$
2nx9

$\boldsymbol{h}$
9

$\boldsymbol{0}$
2n

# Reminder: Least Squares

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $\quad (\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\quad \boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b} \longleftarrow$

In Matlab:

```
x = A \ b
```

Note: You almost <u>never</u> want to compute the inverse of a matrix.

# Singular Value Decomposition

ortho-normal diagonal ortho-normal

unit norm constraint

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

n x m n x n n x m m x m

$$= \sum_{i=1}^{9} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

n x 1 1 x m

General form of total least squares

(Warning: change of notation. x is a vector of parameters!)

$$E_{\mathrm{TLS}} = \sum_i (\boldsymbol{a}_i \boldsymbol{x})^2$$

$$= \|\mathbf{A}\boldsymbol{x}\|^2$$ (matrix form)

$$\|\boldsymbol{x}\|^2 = 1$$ constraint

minimize $\|\mathbf{A}\boldsymbol{x}\|^2$

subject to $\|\boldsymbol{x}\|^2 = 1$

$\Rightarrow$

(Rayleigh quotient)

minimize $\dfrac{\|\mathbf{A}\boldsymbol{x}\|^2}{\|\boldsymbol{x}\|^2}$

Solution is the eigenvector corresponding to smallest eigenvalue of

$$\mathbf{A}^\top \mathbf{A}$$

(equivalent)

Solution is the column of V corresponding to smallest singular value

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

# Solving for H using DLT

(Direct linear transformation: algorithm for solving homographies)

Given $\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$ solve for H such that $\boldsymbol{x}' = \mathbf{H}\boldsymbol{x}$

1. For each correspondence, create 2x9 matrix $\mathbf{A}_i$

2. Concatenate into single 2n x 9 matrix $\mathbf{A}$

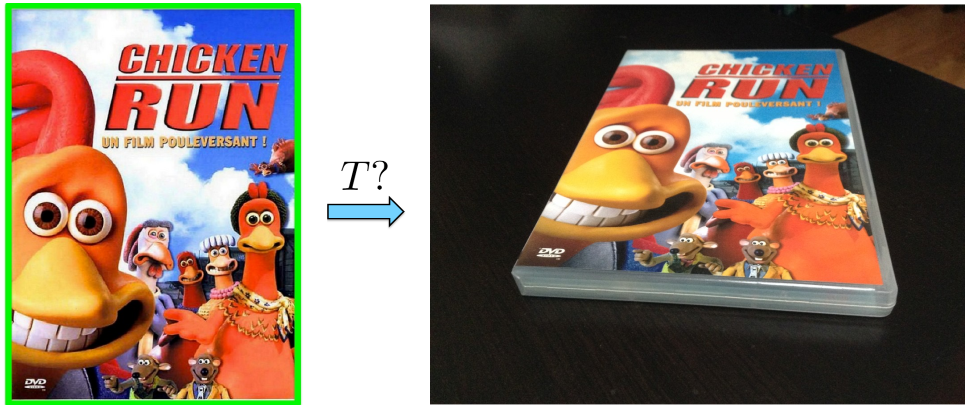
3. Compute SVD of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

4. Store singular vector of the smallest singular value $\boldsymbol{h} = \boldsymbol{v}_{\hat{i}}$

5. Reshape to get $\mathbf{H}$

# Image Alignment Algorithm: Homography

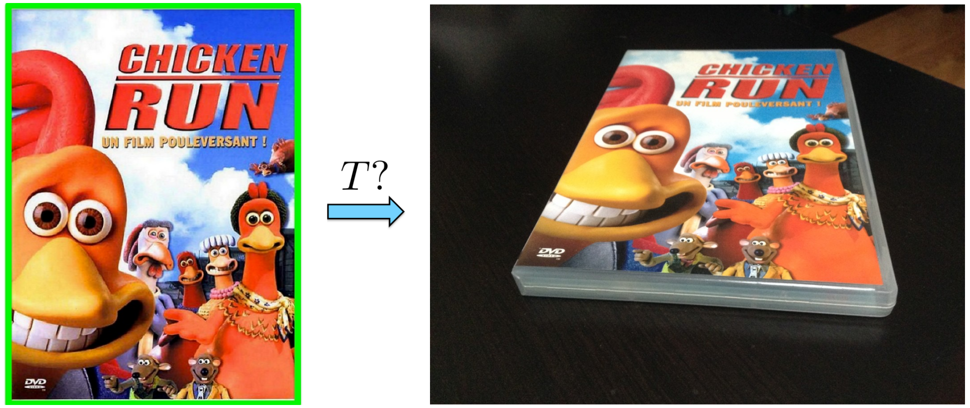
- Given images I and J
  - How can we find the alignment between images?



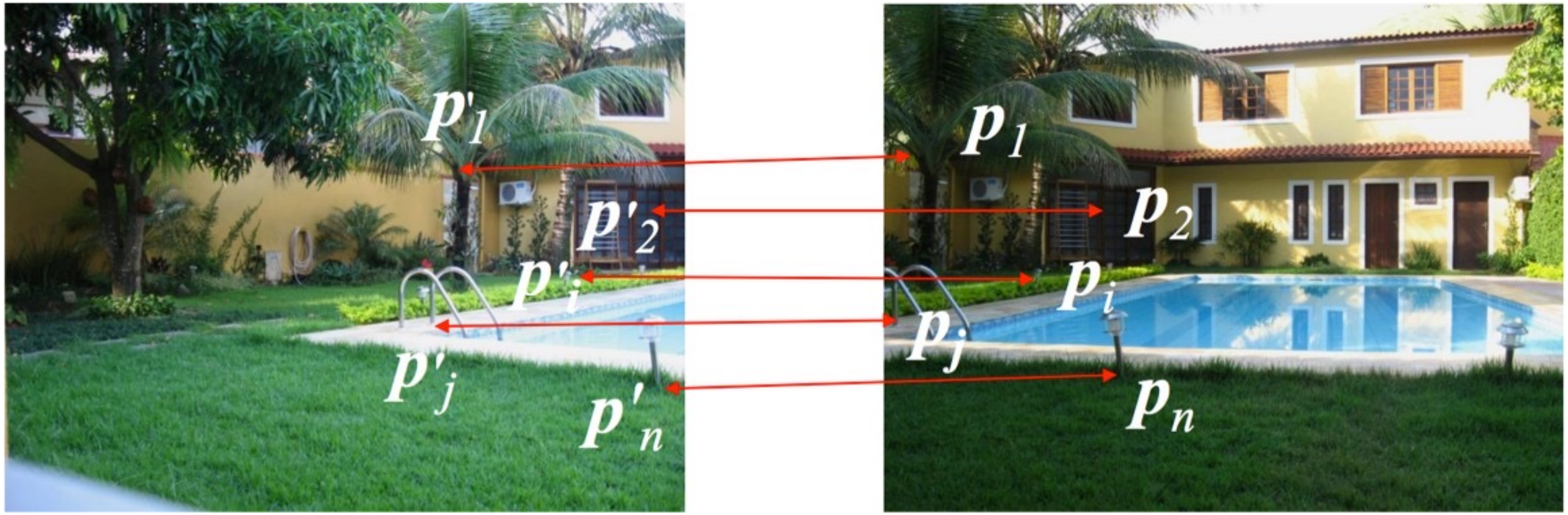

1. Compute image features for *I* and *J*
2. Match features between *I* and *J*
3. Compute homography transformation A between I and J

# Image Alignment Algorithm: Homography

- Given images I and J
    - How can we find the alignment between images?



# Panorama Stitching: Example 1




- Compute the matches

[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 1

- Estimate the homography and warp

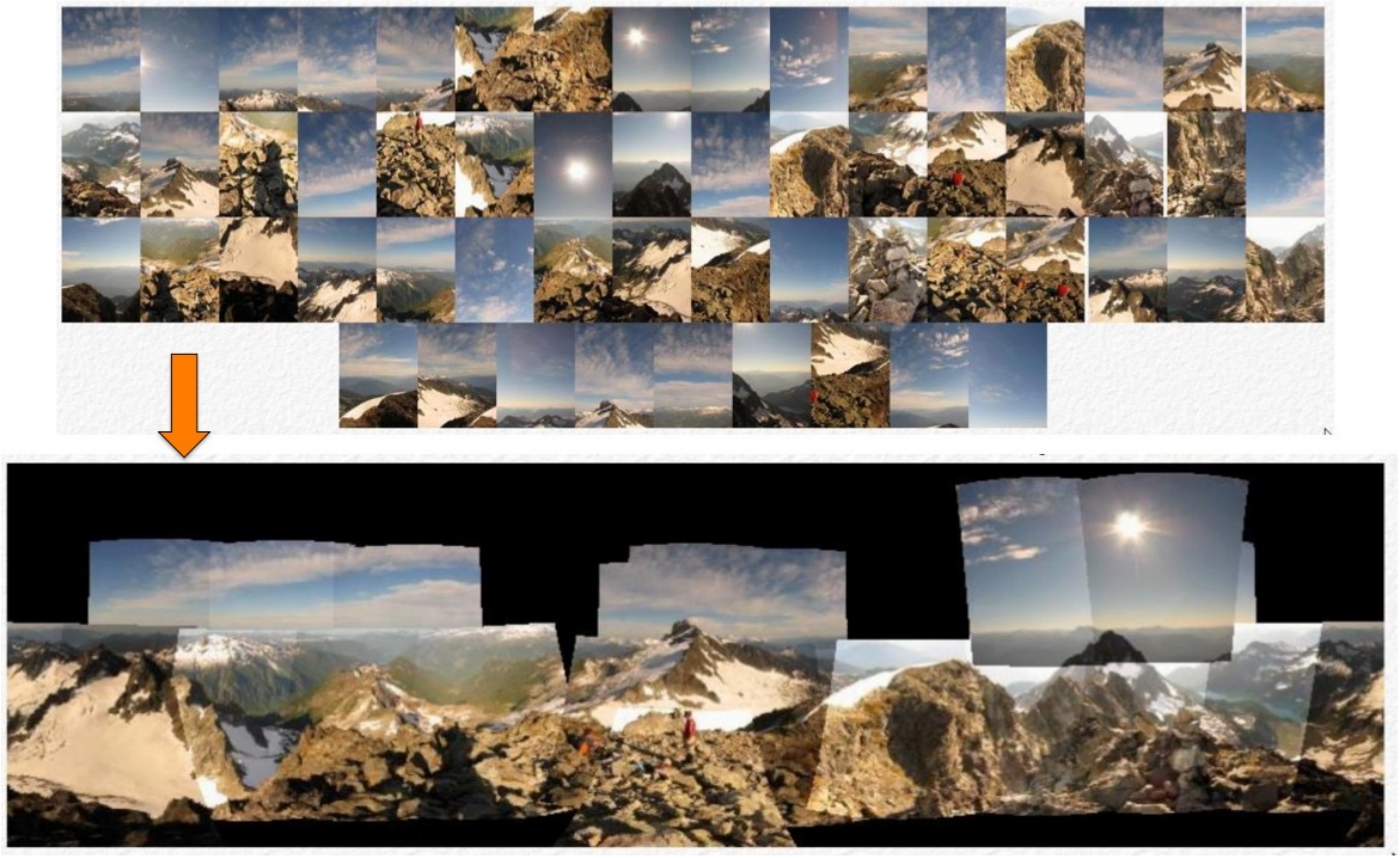
[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 1

- Stitch

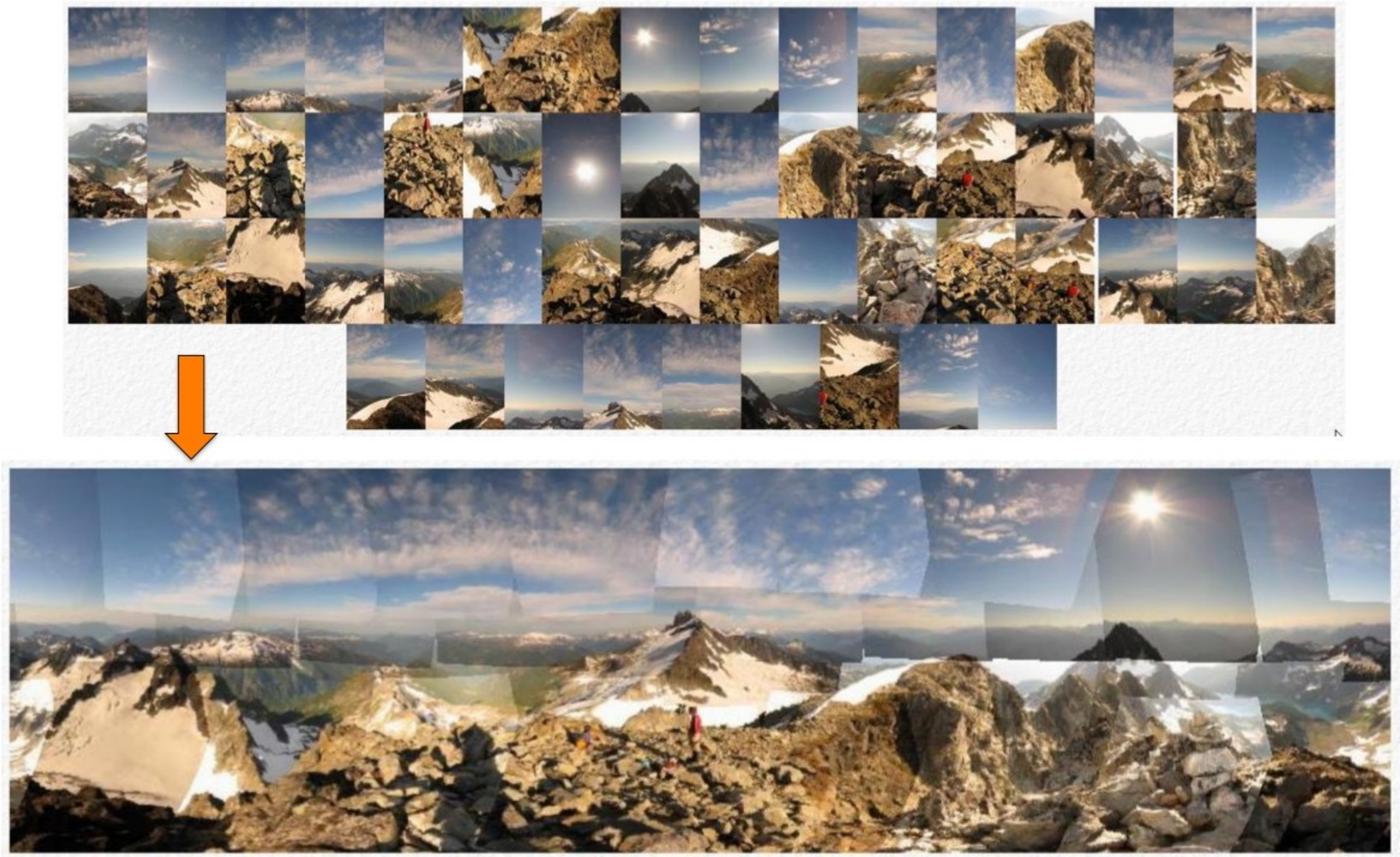
[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 2



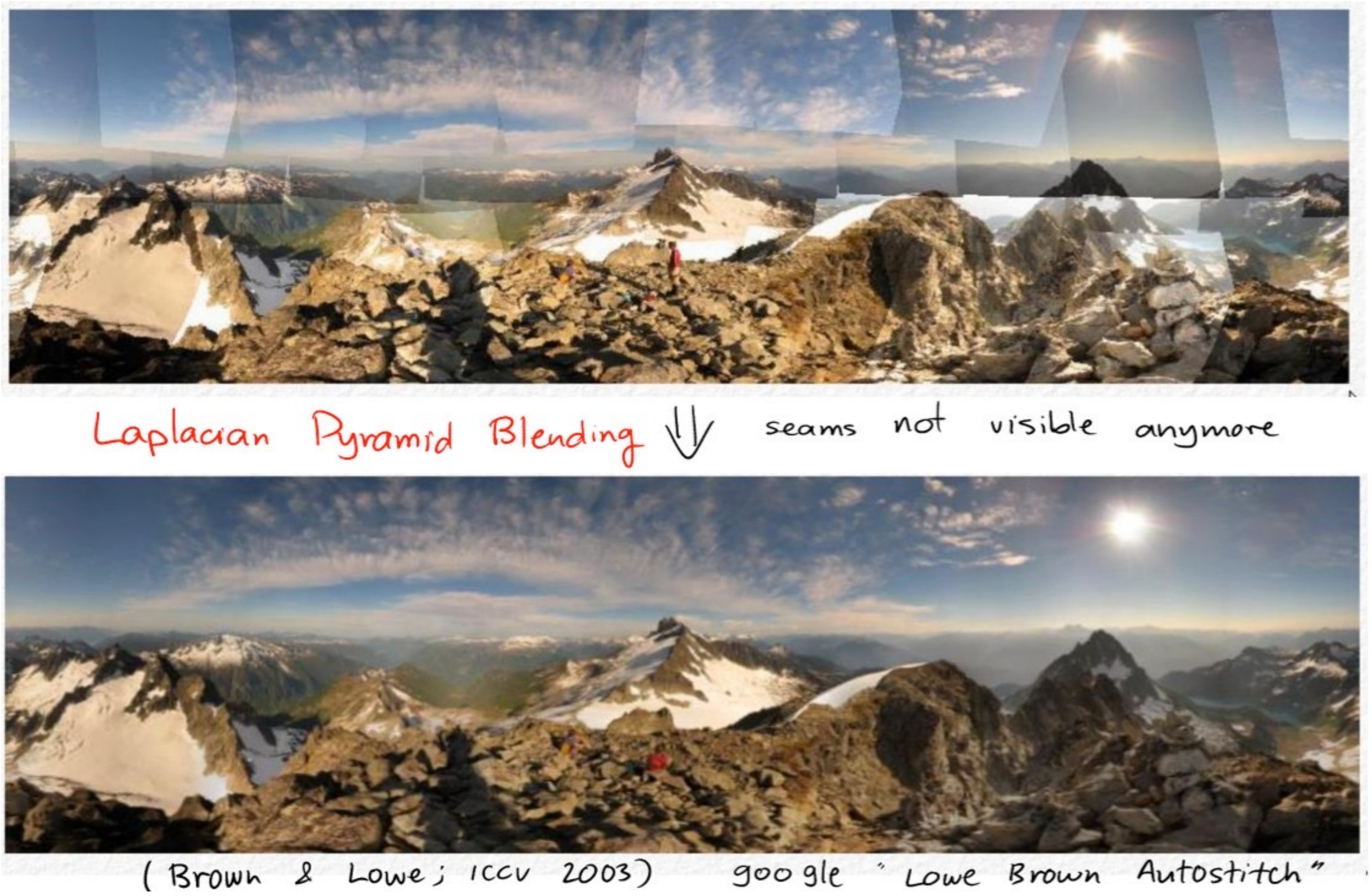
[Source: Fernando Flores-Mangas]

# Panorama Stitching: Example 2



[Source: Fernando Flores-Mangas]

# Panorama Stitching: Example 2



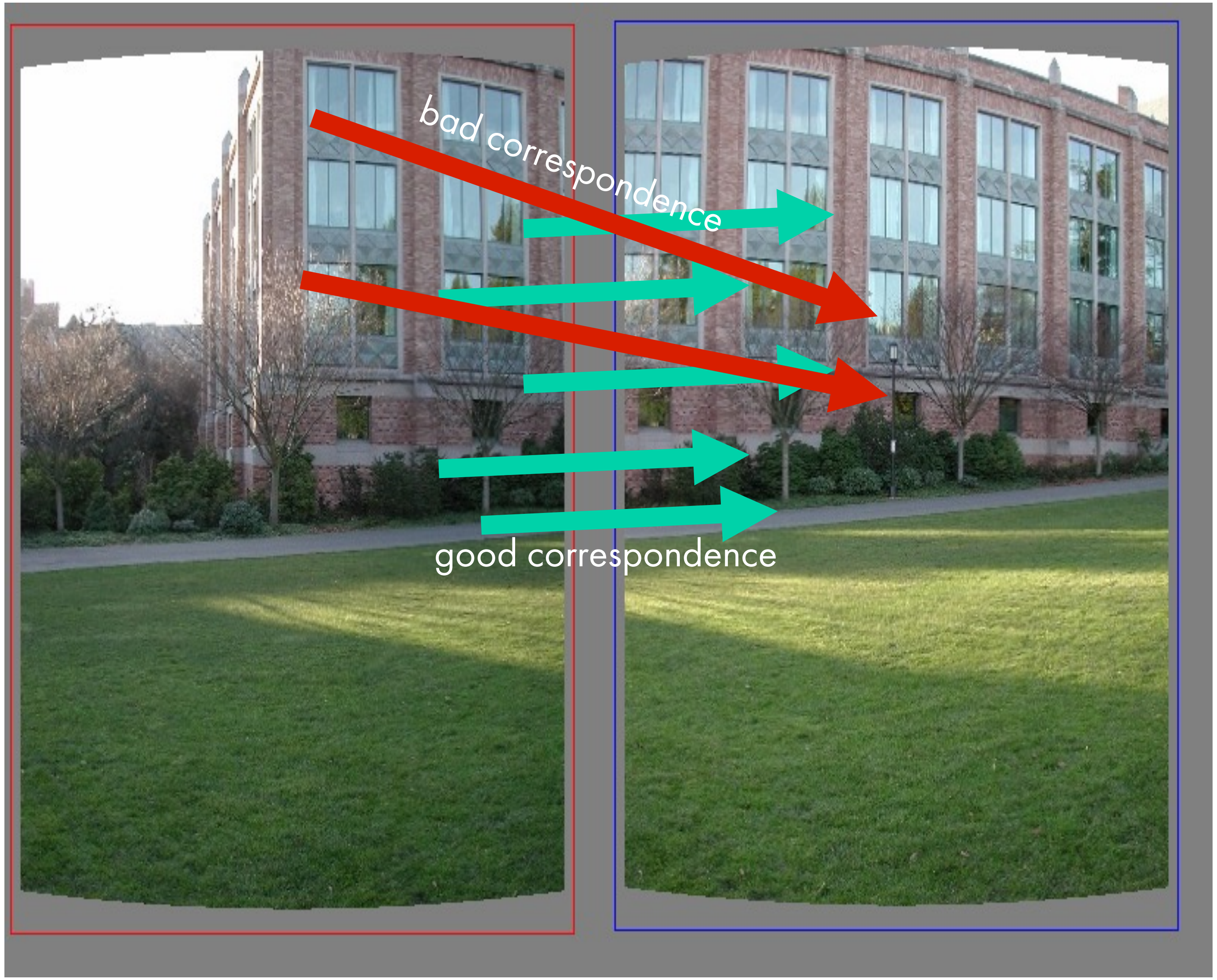

[Source: Fernando Flores-Mangas]

# The image correspondence pipeline

1. Feature point detection
   - Detect blobs using, e.g., multi-scale LoG

2. Feature point description
   - Describe features using the SIFT descriptor.

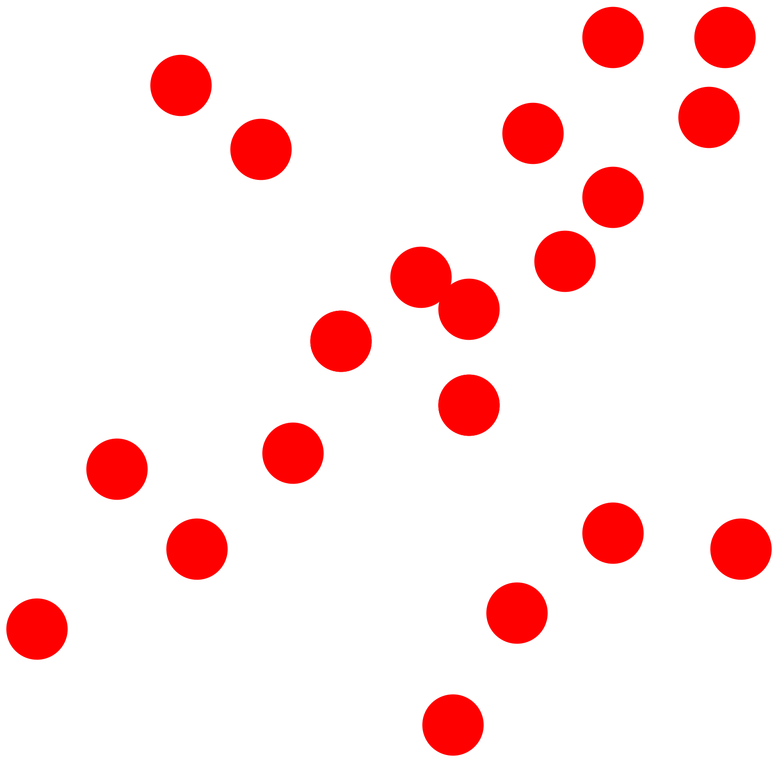
3. Feature matching




How do we estimate homographies when we have both good and bad correspondences?

# Random Sample Consensus (RANSAC)

Fitting lines
(with outliers)

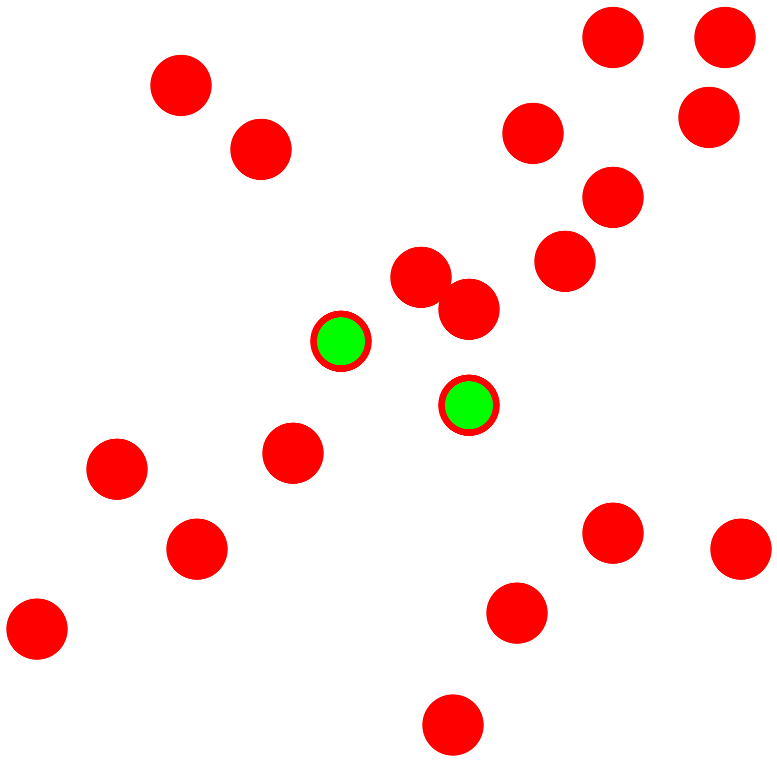


**Algorithm:**

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

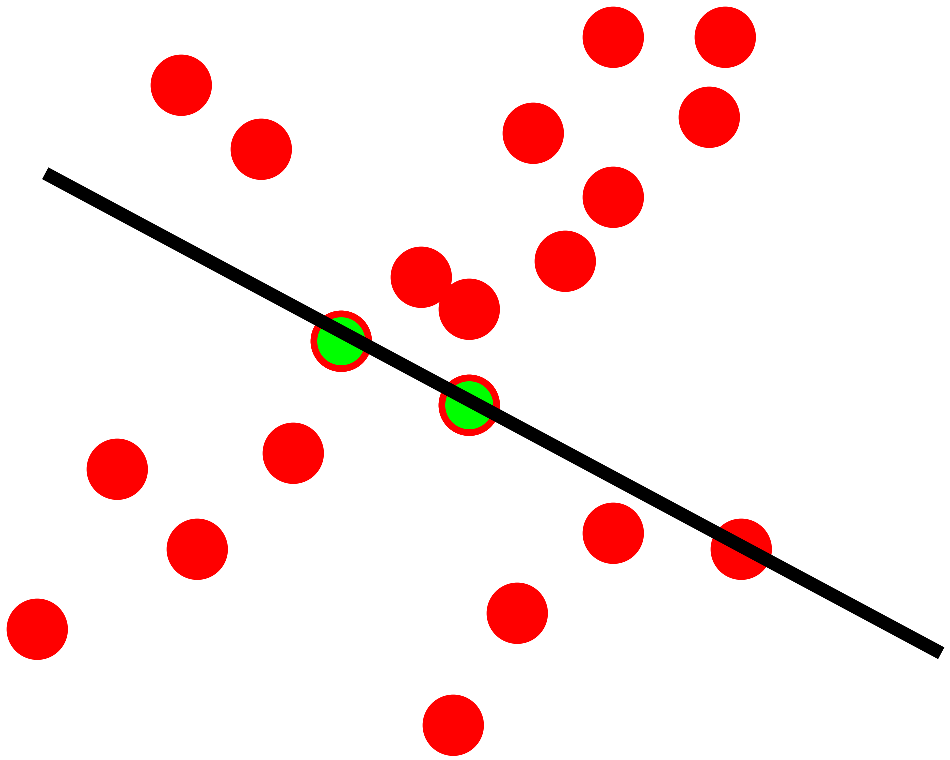


## Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

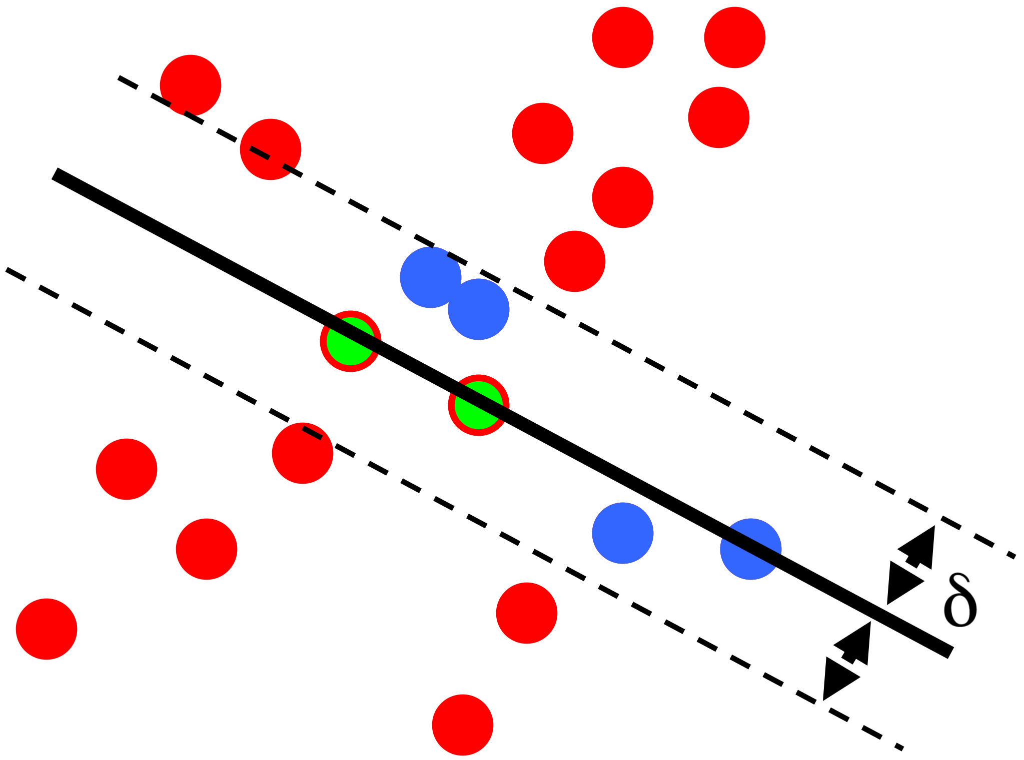


## Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



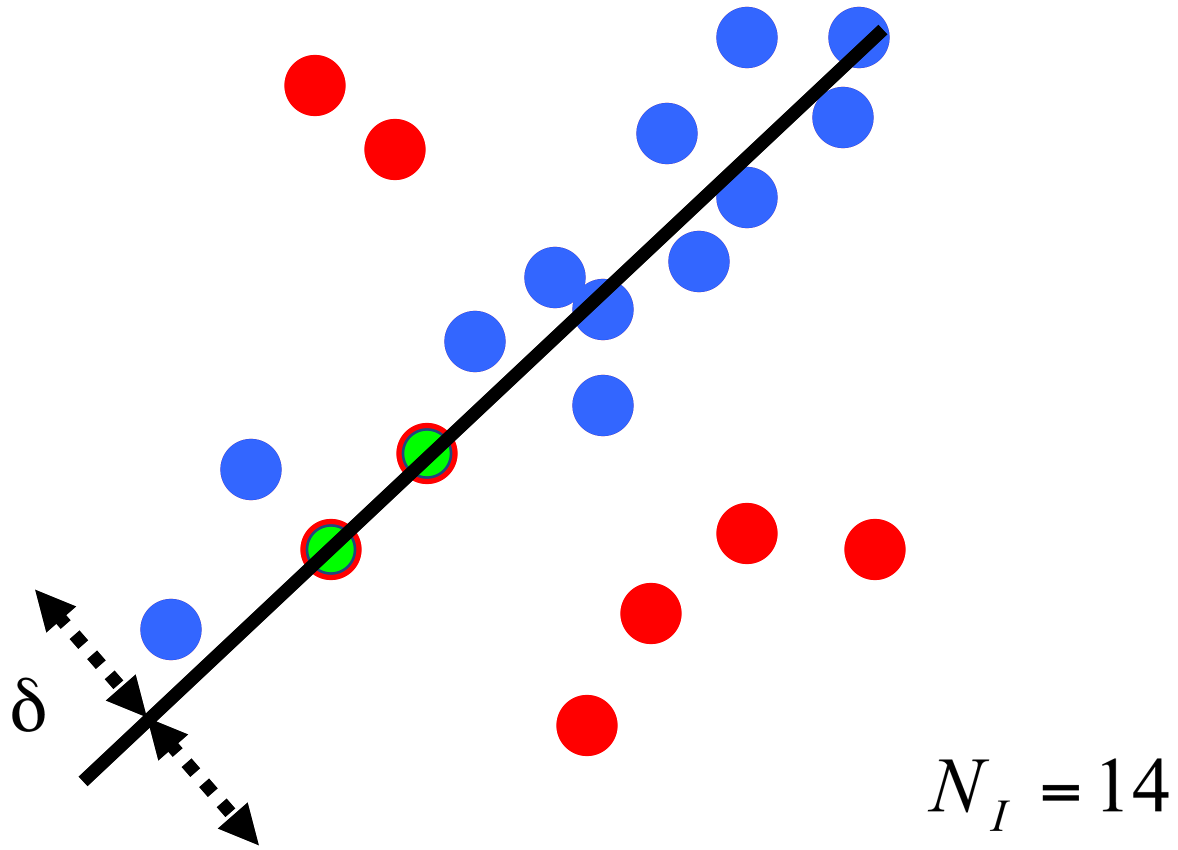

$N_I = 6$

## Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)




**Algorithm:**

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

**Repeat 1-3 until the best model is found with high confidence**

# How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e )

# How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e )
- Number of sampled points s
  - Minimum number needed to fit the model

# How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e )
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

# How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e )
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

p = desired probability that we get a good sample
s = number of points in a sample
N = number of samples (we want to compute this)
e = probability that a point is an outlier

# How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e )
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

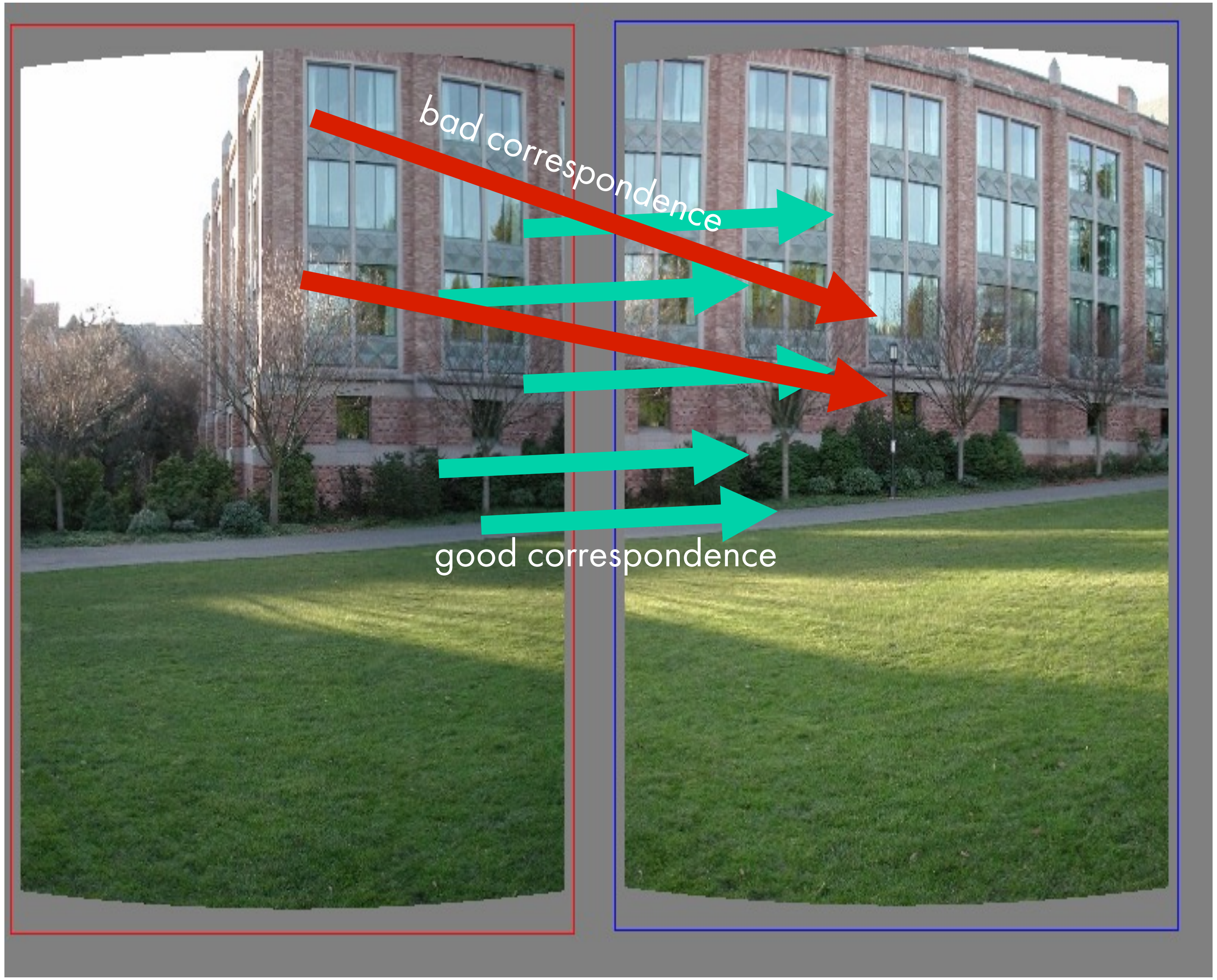
p = desired probability that we get a good sample
s = number of points in a sample
N = number of samples (we want to compute this)
e = probability that a point is an outlier

can you derive this? Hint: start with writing the probability that you choose one inlier point, then that you choose ‘s’ inlier points…

Given two images...

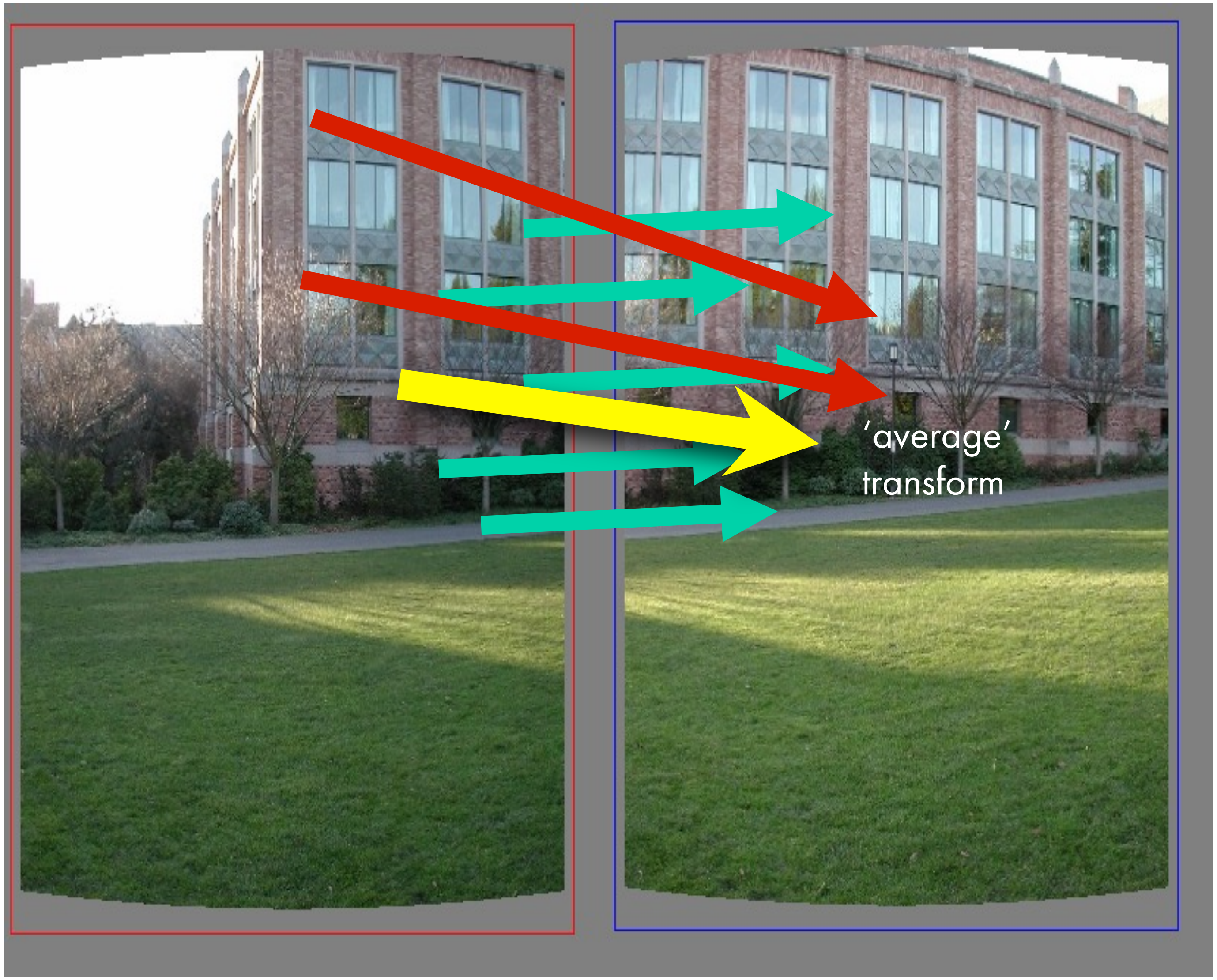
find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences



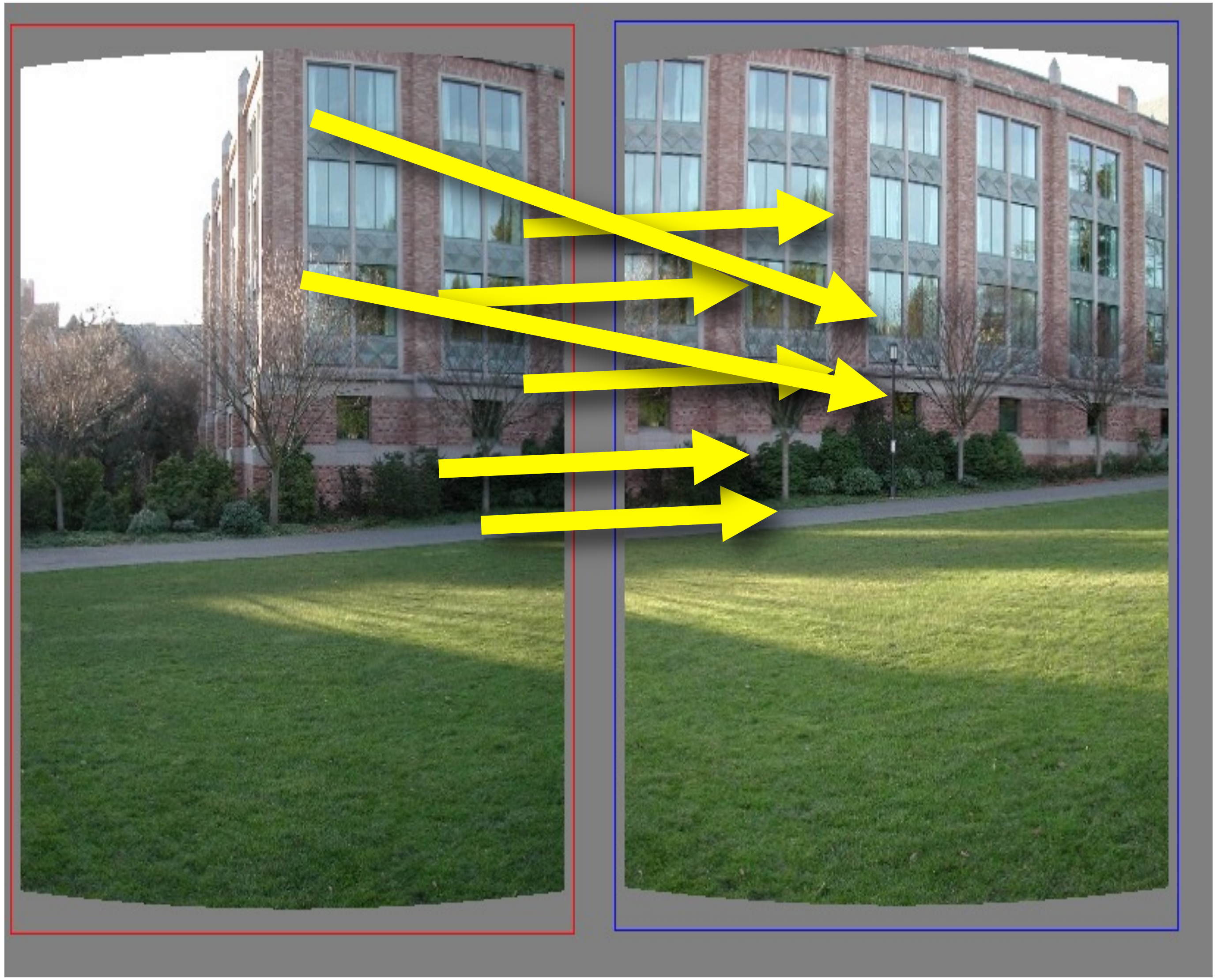

how should we estimate the transform?

LLS will find the 'average' transform

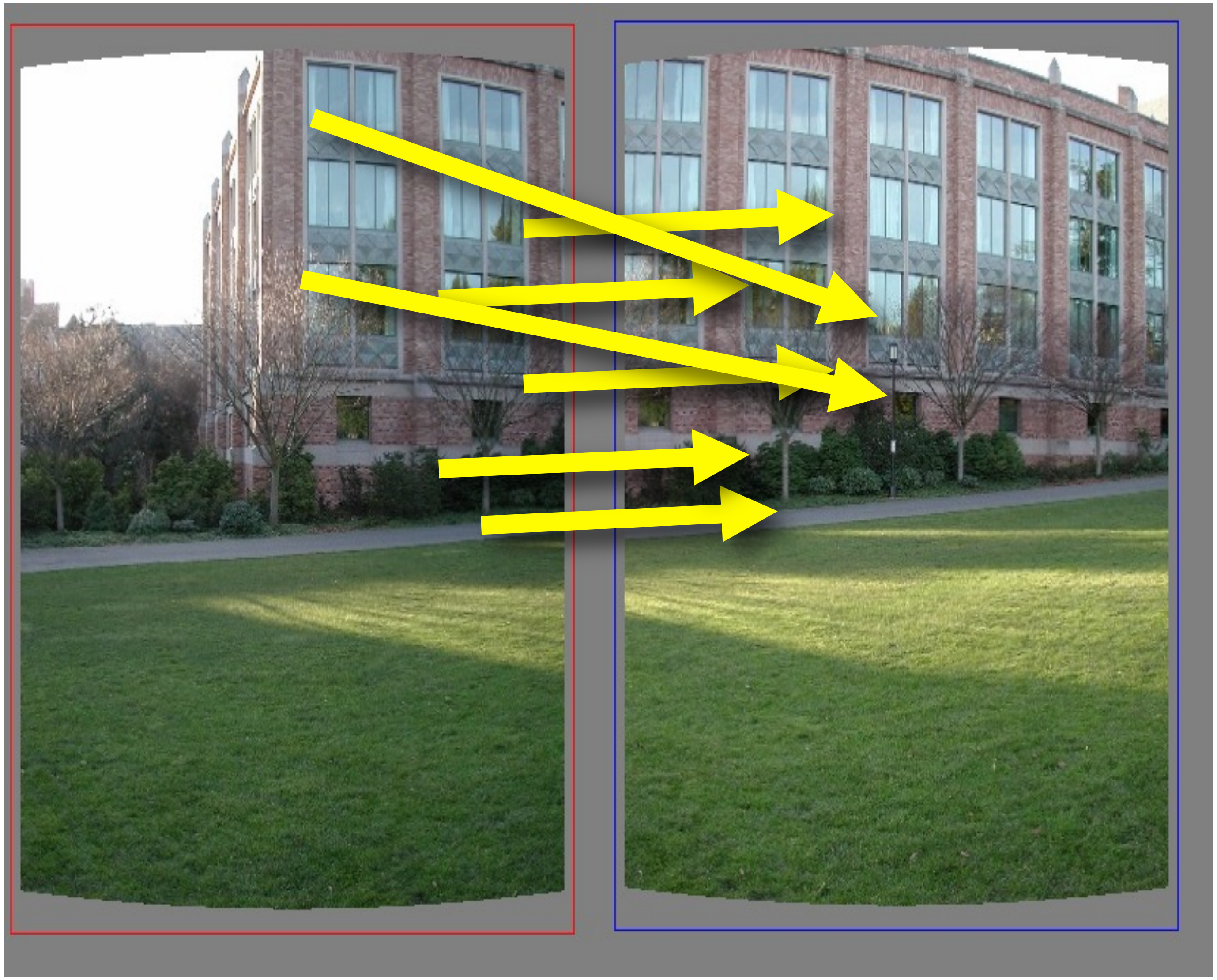



solution is corrupted by bad correspondences

# Use RANSAC

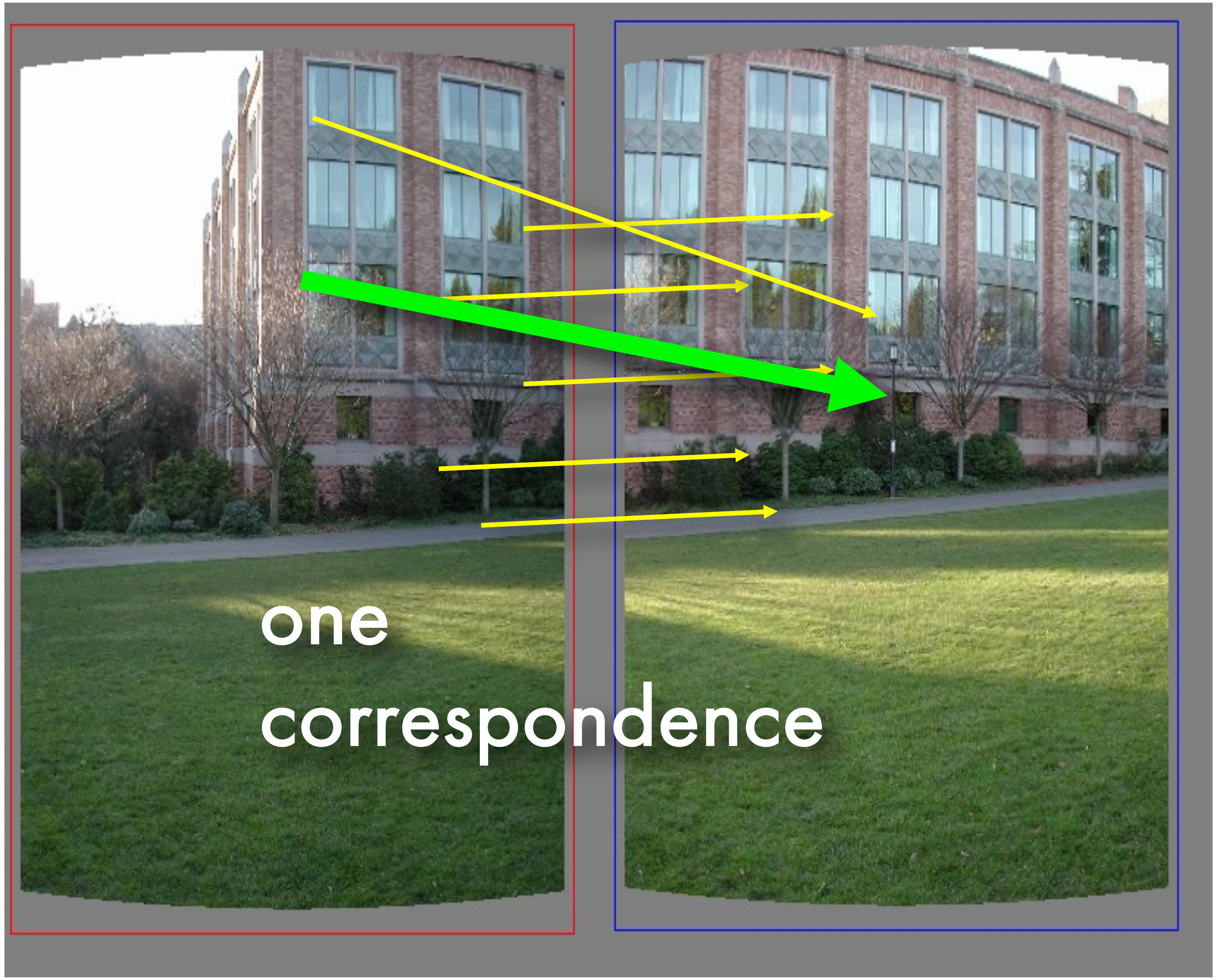


How many correspondences to compute translation transform?

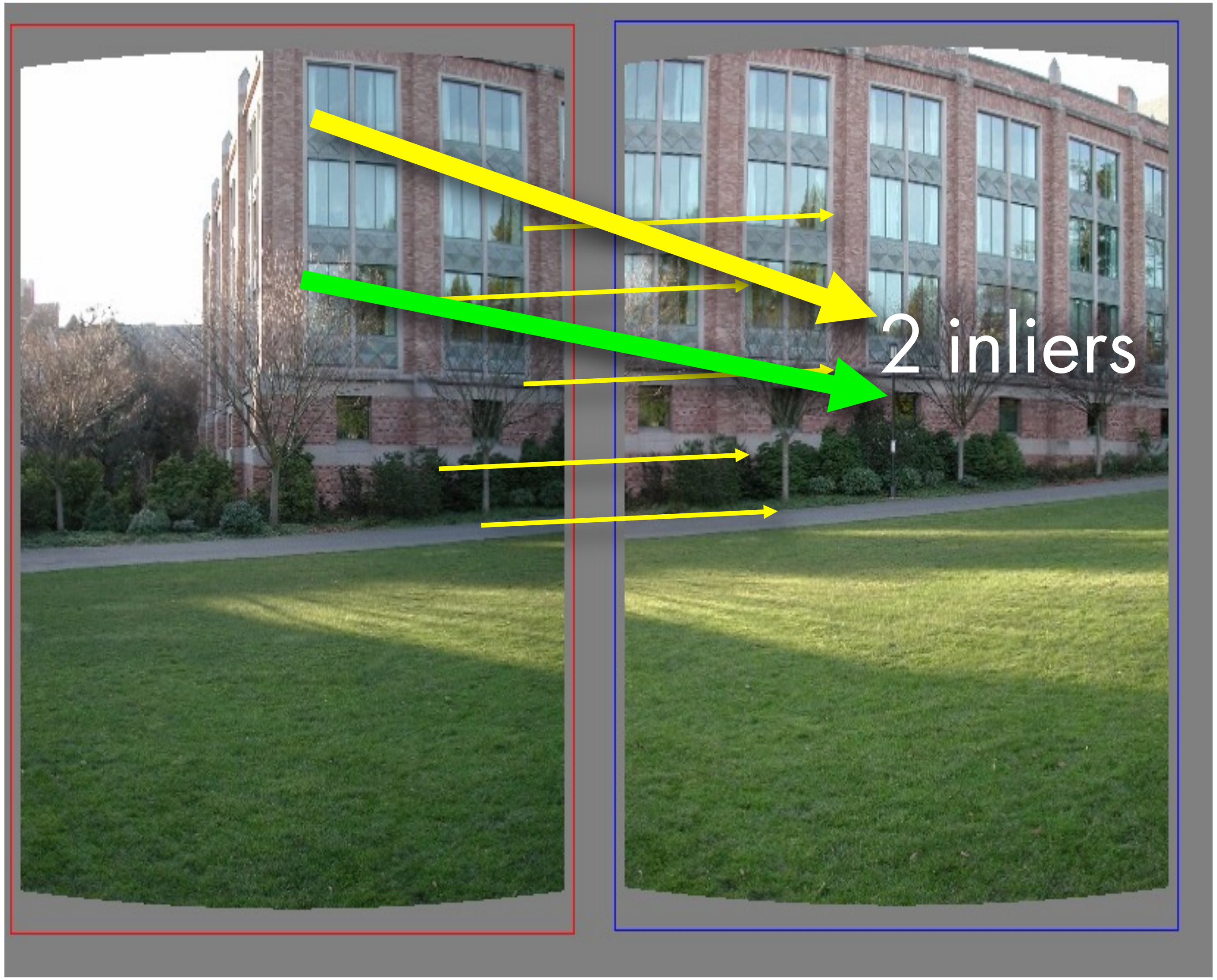


Need only one correspondence, to find translation model

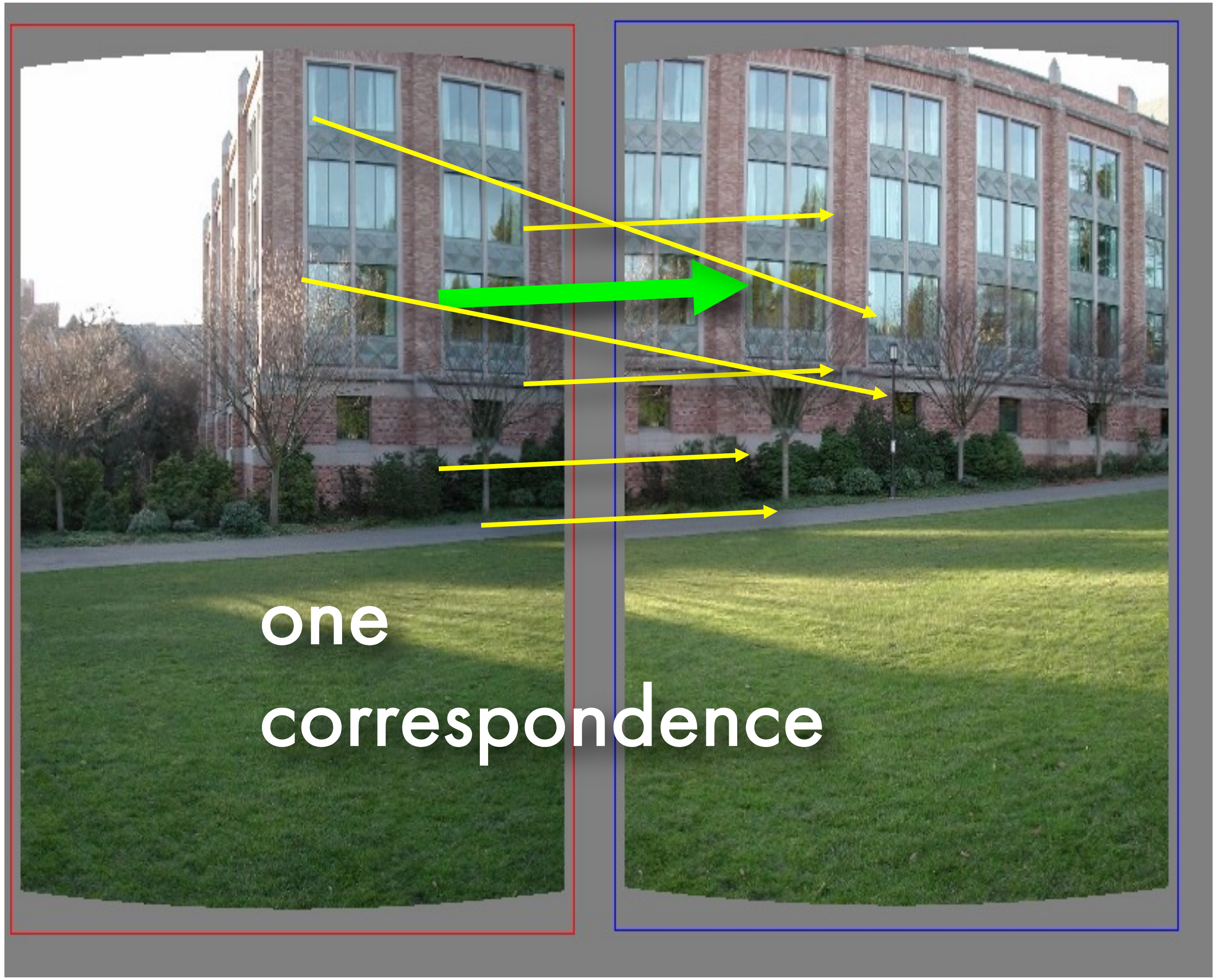
Pick one correspondence, count inliers



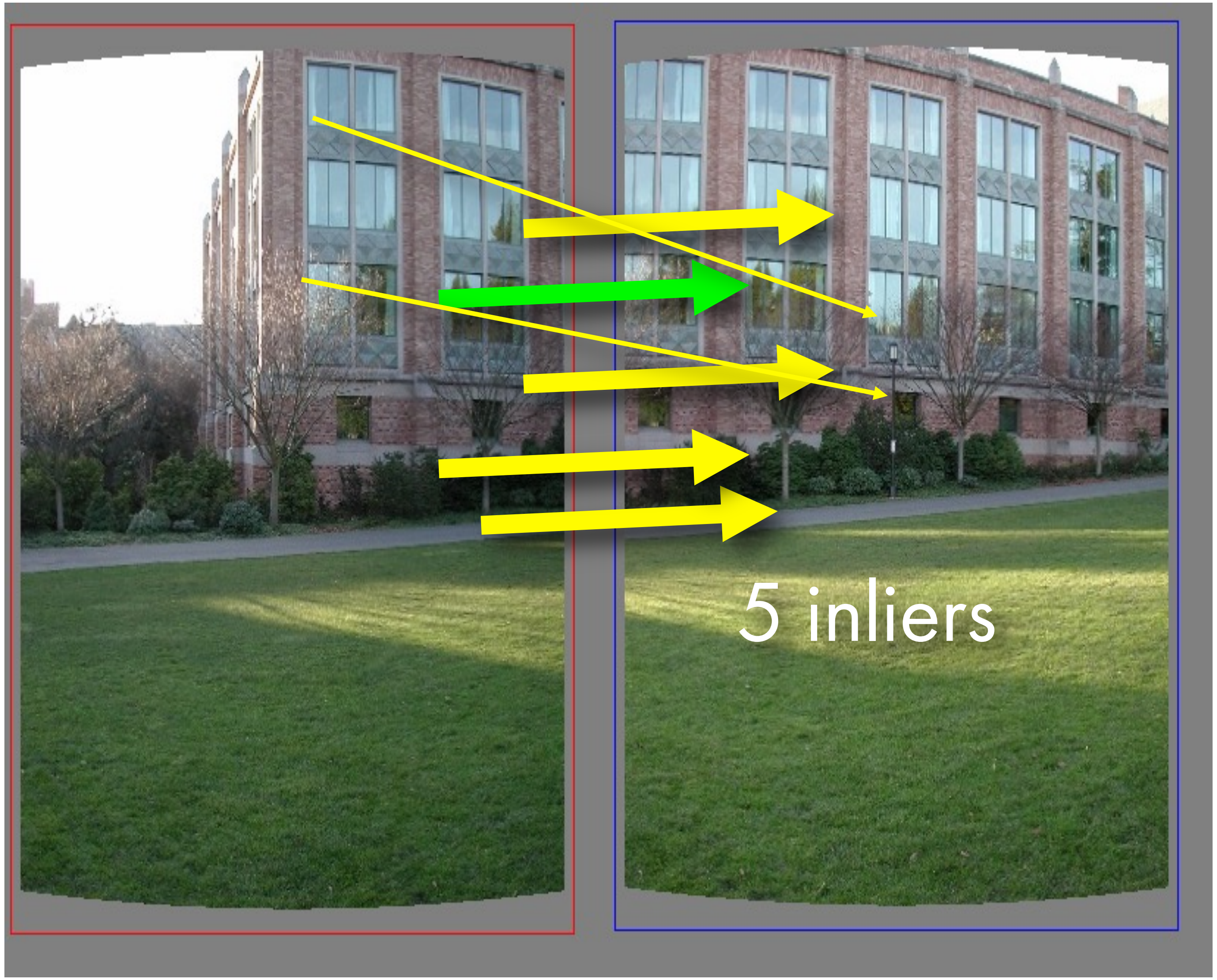
# Pick one correspondence, count inliers



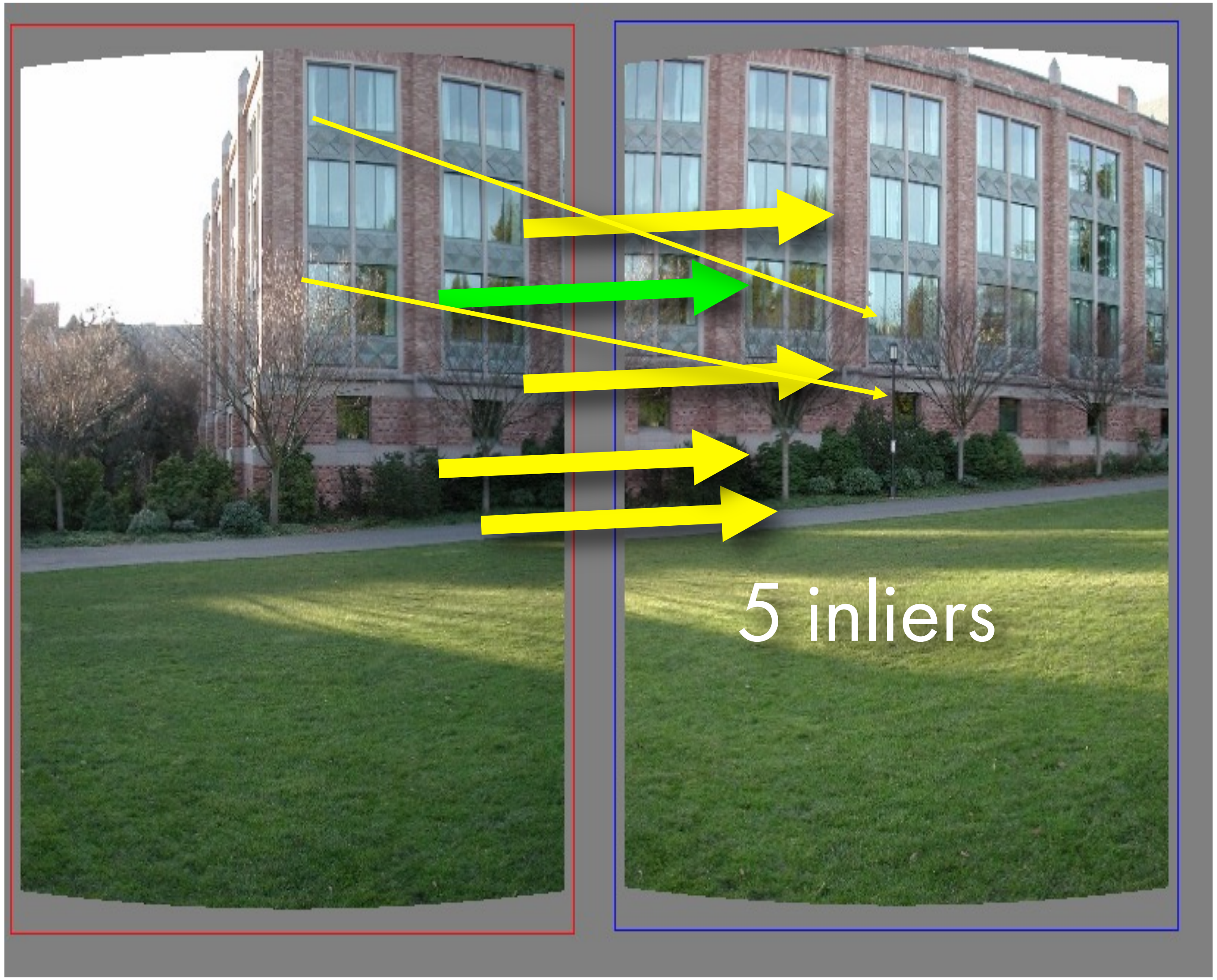
# Pick one correspondence, count inliers



# Pick one correspondence, count inliers



Pick one correspondence, count inliers




Pick the model with the highest number of inliers!

# Estimating homography using RANSAC

- RANSAC loop
    1. Get point correspondences (randomly)

# Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using

# Estimating homography using RANSAC

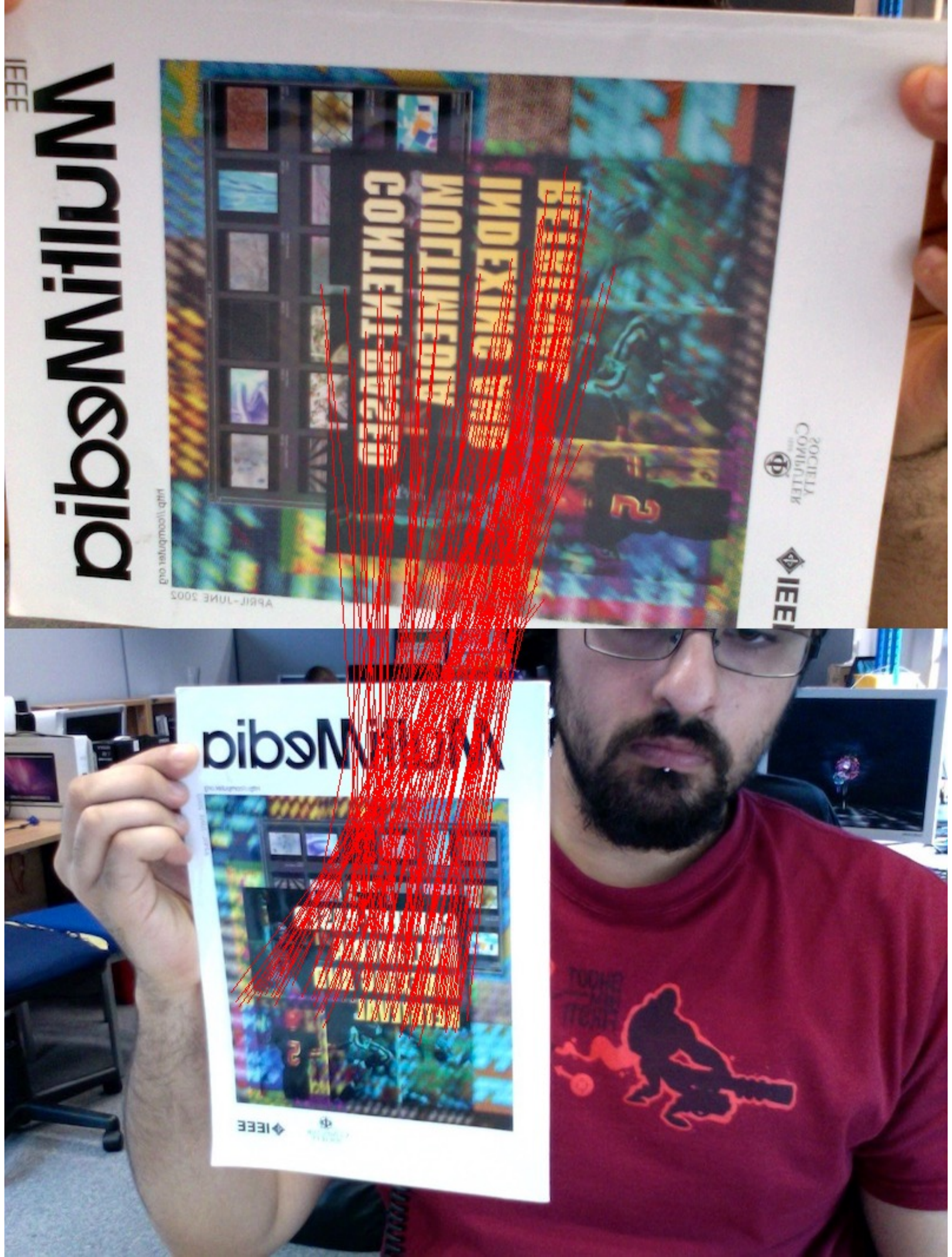
- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count

# Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if

# Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if largest number of inliers
- Recompute H using all inliers



MultiMedia
IEEE

# The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching and homography estimation
   - Do both simultaneously using RANSAC.

# Summary – Stuff You Need To Know

- A homography is a mapping between projective planes
- You need at least 4 correspondences (matches) to compute it
- How do we assemble and solve the constrained least squares system?
- What is the RANSAC algorithm and how can I apply it to a model fitting problem?

- OpenCV (Python):
  - Affine transformation/warp:
    - getAffineTransform(pts_src, pts_dst)
    - warpAffine
  - Perspective transformation/warp:
    - either: getPerspectiveTransform (pts_src, pts_dst) (without RANSAC)
    - or: h, status = cv2.findHomography(pts_src, pts_dst) (with RANSAC)
    - im_dst = warpPerspective(im_src, h, size)

# Birdseye View on What We Learned So Far

| Problem | Detection | Description | Matching |
|---|---|---|---|
| Find Planar Distinctive Objects | Scale Invariant Interest Points | Local feature: SIFT | All features to all features + Affine / Homography |

# More than one image

- We're in 2023…

- Think not (only) what you can do with one image, but what lots and lots of images can do for you

# More than one image

- We're in 2023…

- Think not (only) what you can do with one image, but what lots and lots of images can do for you

- Would our current matching method work with lots of data?

# Looking forward (object detection)

- So far we matched a known object in a new viewpoint
- What if we have to match an object to LOTS of images? Or LOTS of objects to one image?
- We’ll discuss this in a few weeks (object recognition)

# Next Time: Camera Models