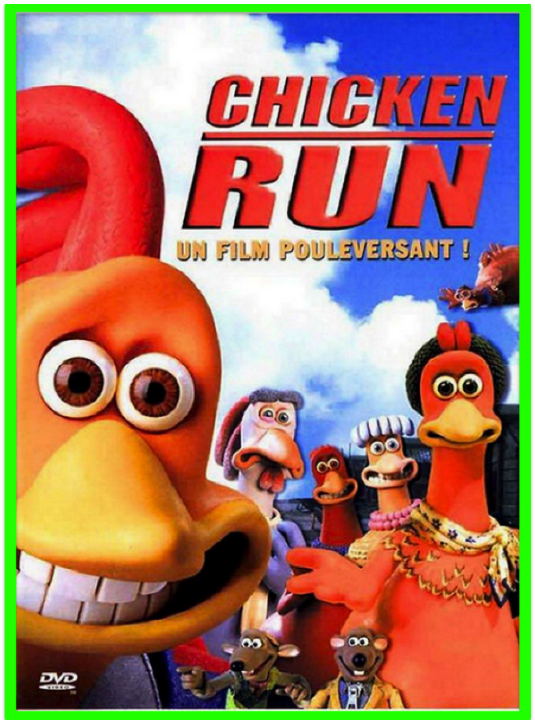
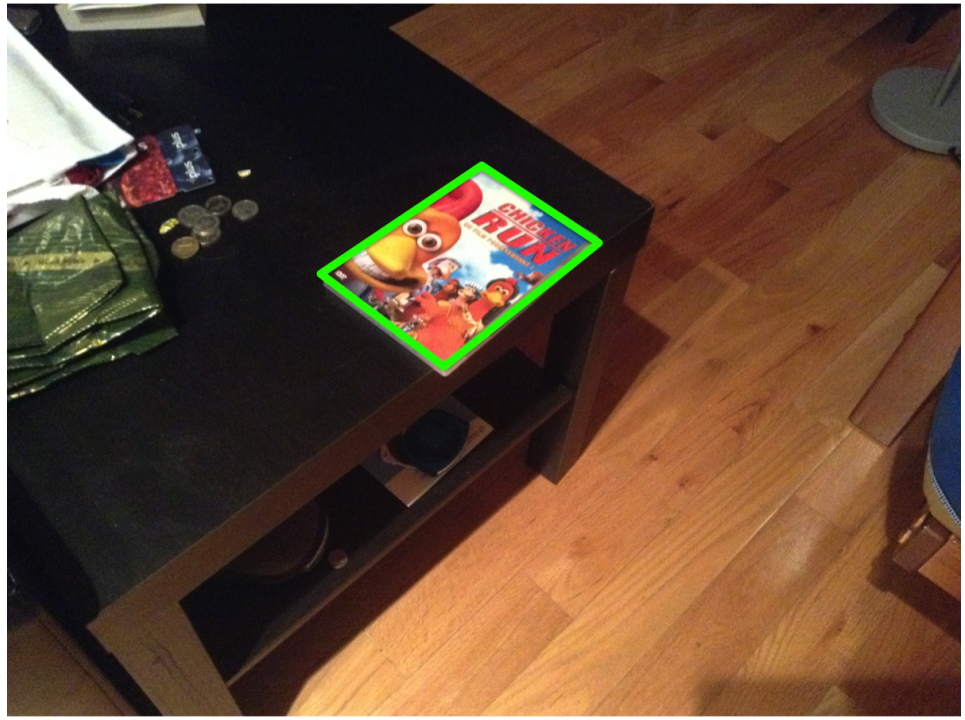
# Matching Planar Objects In New Viewpoints ... And Much More – via Homography

# What Transformation Happened To My DVD?

- Rectangle goes to a parallelogram




$T$?



# Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

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

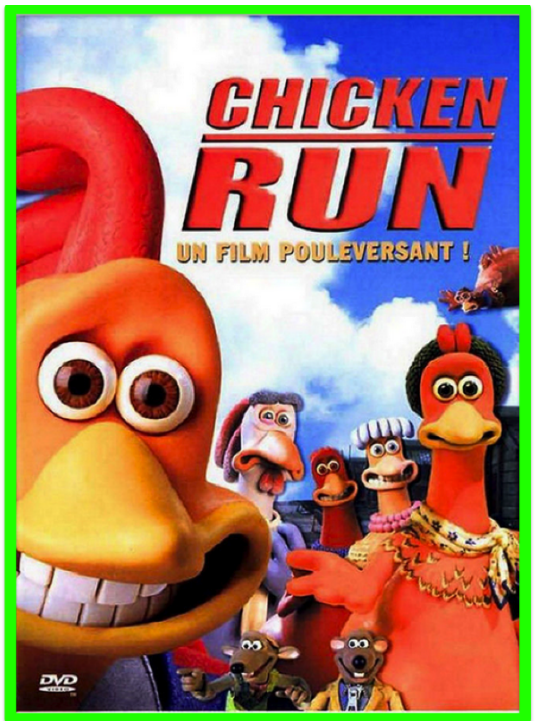
Properties of affine transformations:

- 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 Really Happened To My DVD?

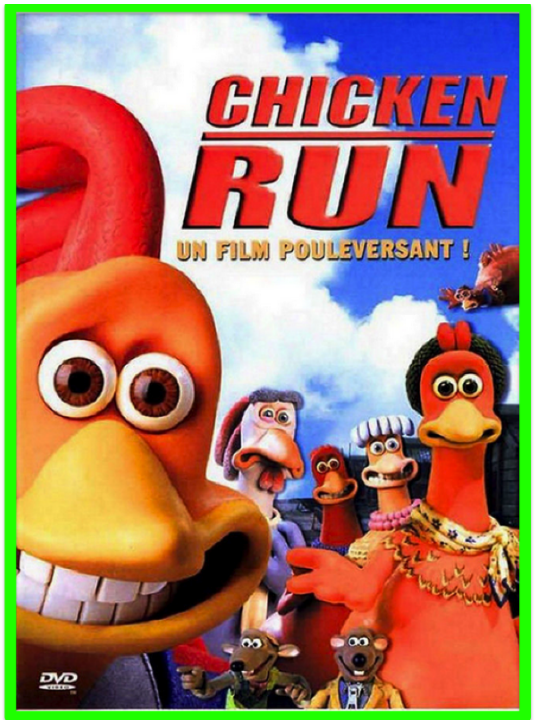
- What about now?




$T?$

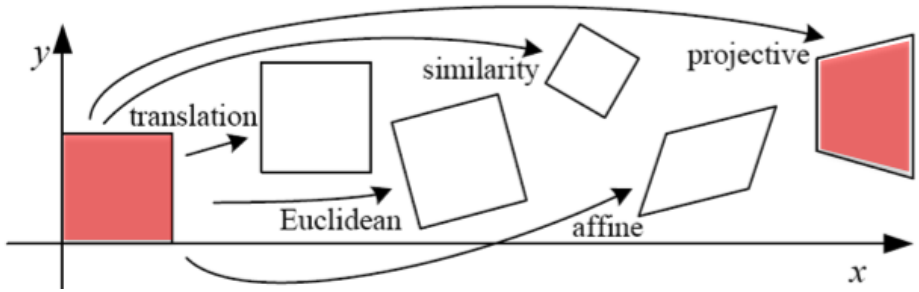
# What Transformation Really Happened To My DVD?

- Actually a rectangle goes to **quadrilateral**




$T?$

# 2D Image Transformations




| Transformation | Matrix | # DoF | Preserves | Icon |
|---|---|---|---|---|
| translation | $\left[ \boldsymbol{I} \mid \boldsymbol{t} \right]_{2\times 3}$ | 2 | orientation | |
| rigid (Euclidean) | $\left[ \boldsymbol{R} \mid \boldsymbol{t} \right]_{2\times 3}$ | 3 | lengths | |
| similarity | $\left[ s\boldsymbol{R} \mid \boldsymbol{t} \right]_{2\times 3}$ | 4 | angles | |
| affine | $\left[ \boldsymbol{A} \right]_{2\times 3}$ | 6 | parallelism | |
| projective | $\left[ \tilde{\boldsymbol{H}} \right]_{3\times 3}$ | 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:

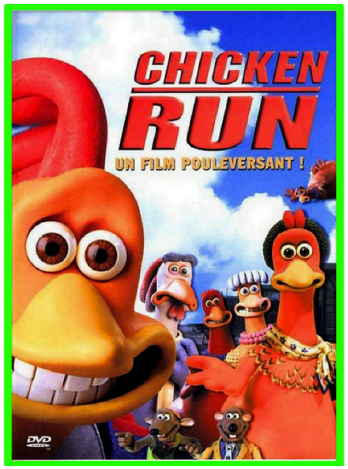
$$w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Properties:

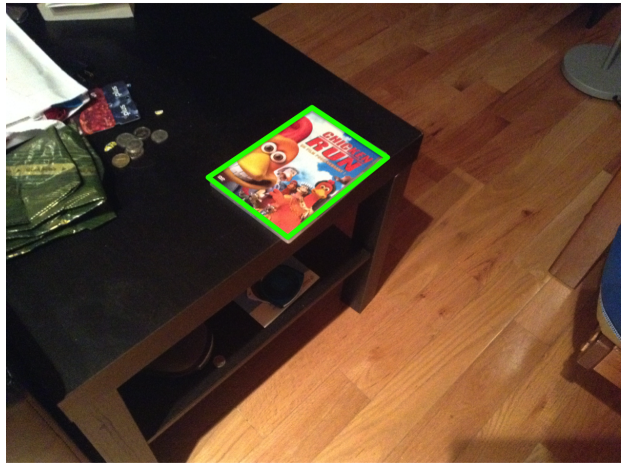
- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are **not** preserved
- Closed under composition
- Rectangle goes to quadrilateral
- Affine transformation is a special case, where $g = h = 0$ and $i = 1$

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

# What Transformation Really 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

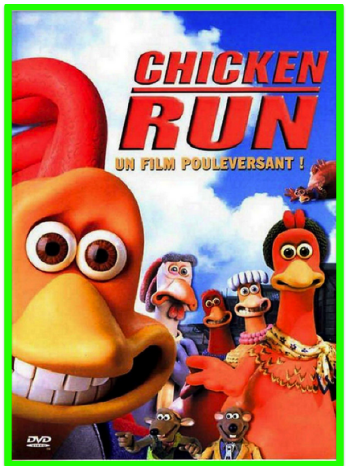
# What Transformation Happened to My DVD?

- 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...
- 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?

# 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

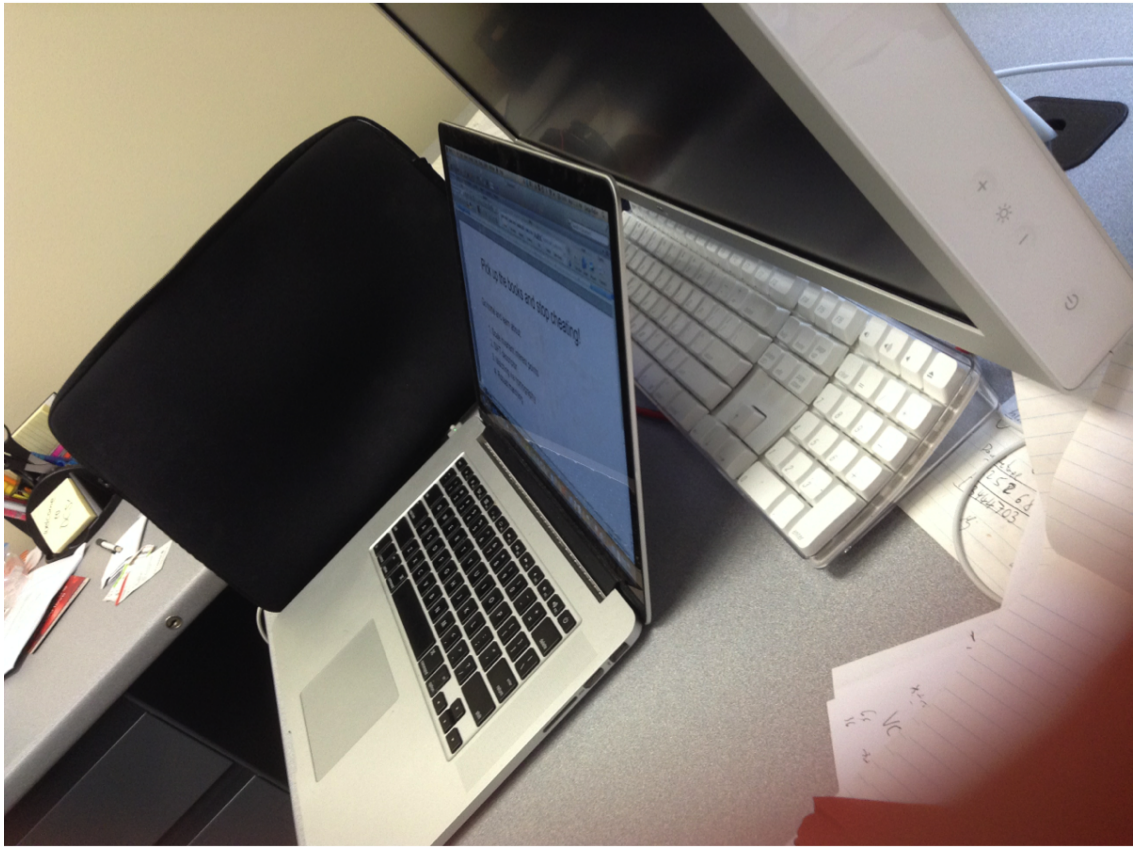
# Application 1: a Little Bit of CSI


- The professor keeps the exams in this office

# Application 1: a Little Bit of CSI

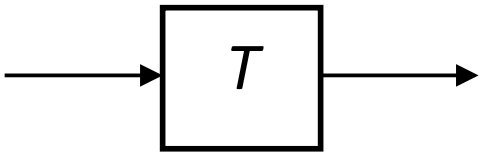

- He enters (without permission) and takes a picture of the laptop screen

# 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:

$$[x', y'] = T(x, y)$$

- 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:

translation

rotation

aspect

affine

perspective

# Forward and Inverse Warping

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

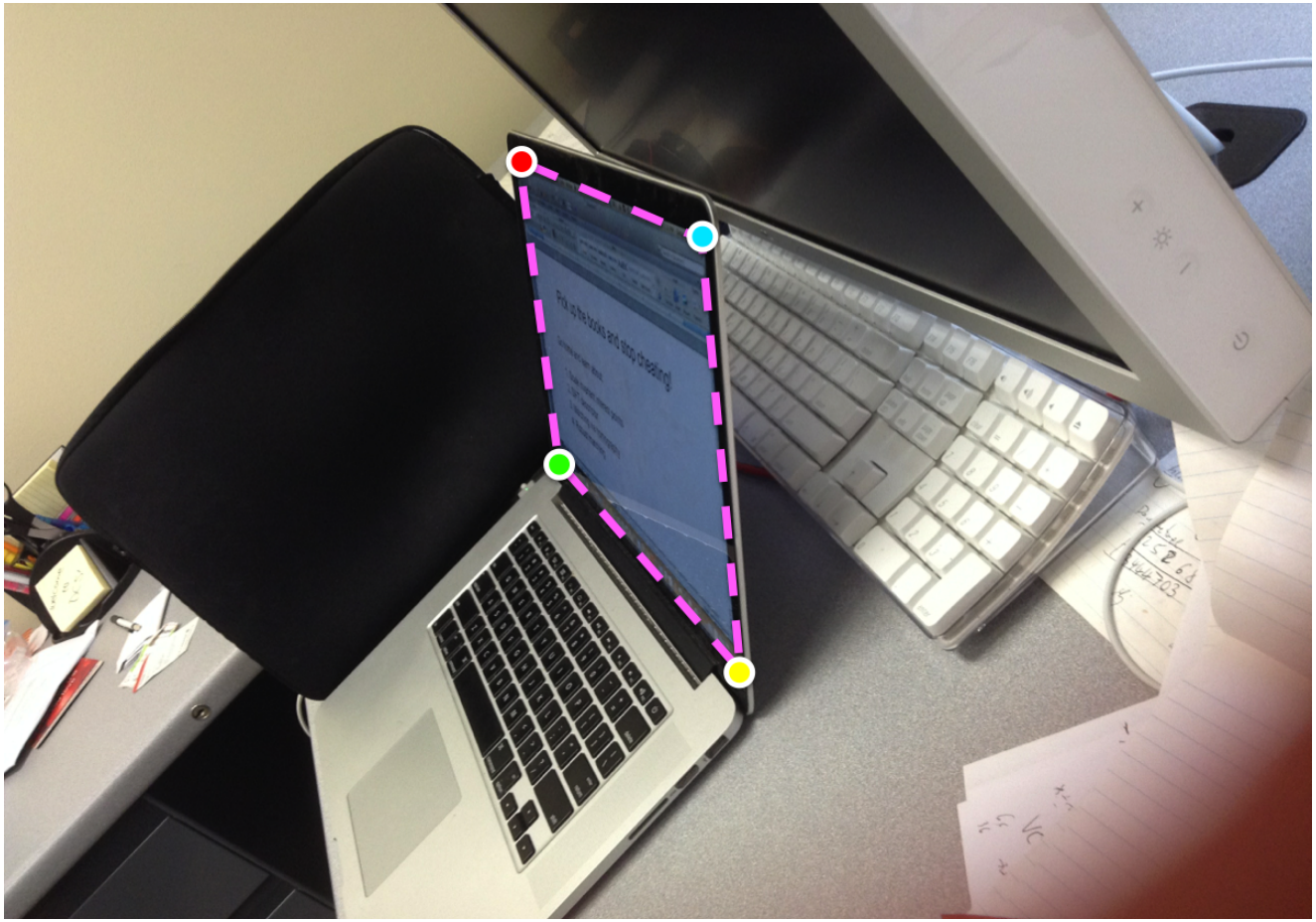
> **procedure** *forwardWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:
>
> For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$
>
> 1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
> 2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

May leave some holes in the target image.

- **Inverse Warping:** Each pixel at destination is sampled from original image

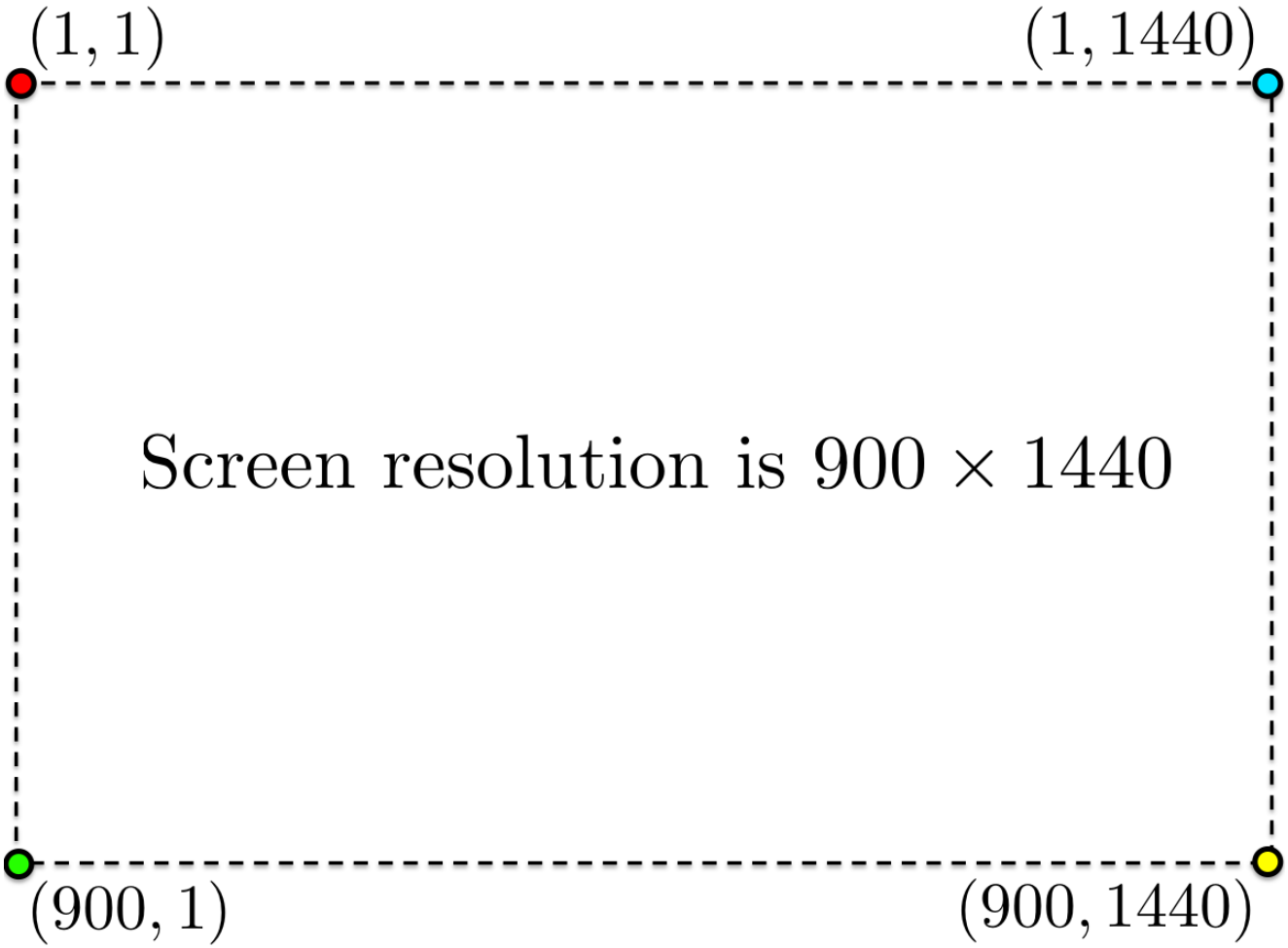
> **procedure** *inverseWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:
>
> For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$
>
> 1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
> 2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

# Application 1: a Little Bit of CSI



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

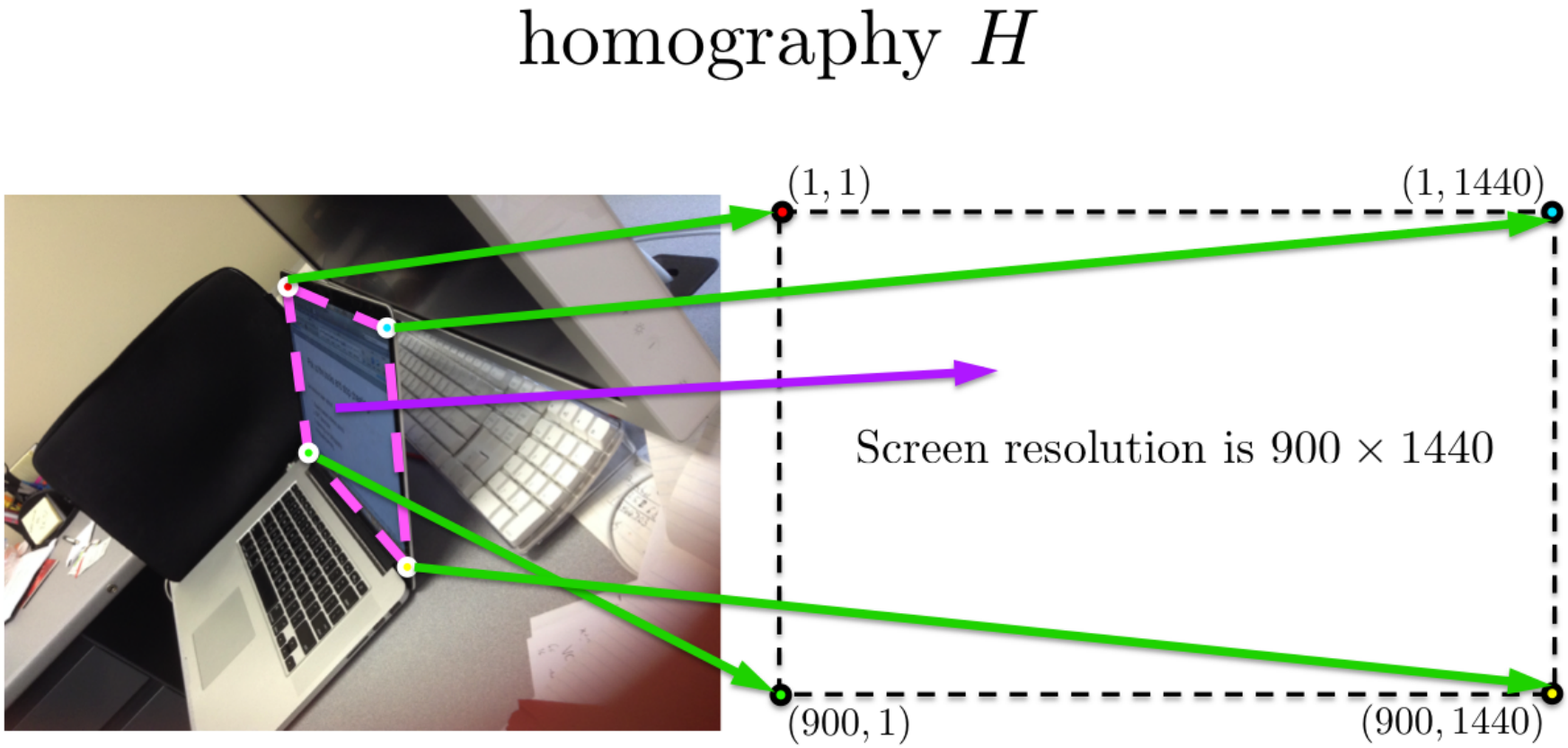
# Application 1: a Little Bit of CSI



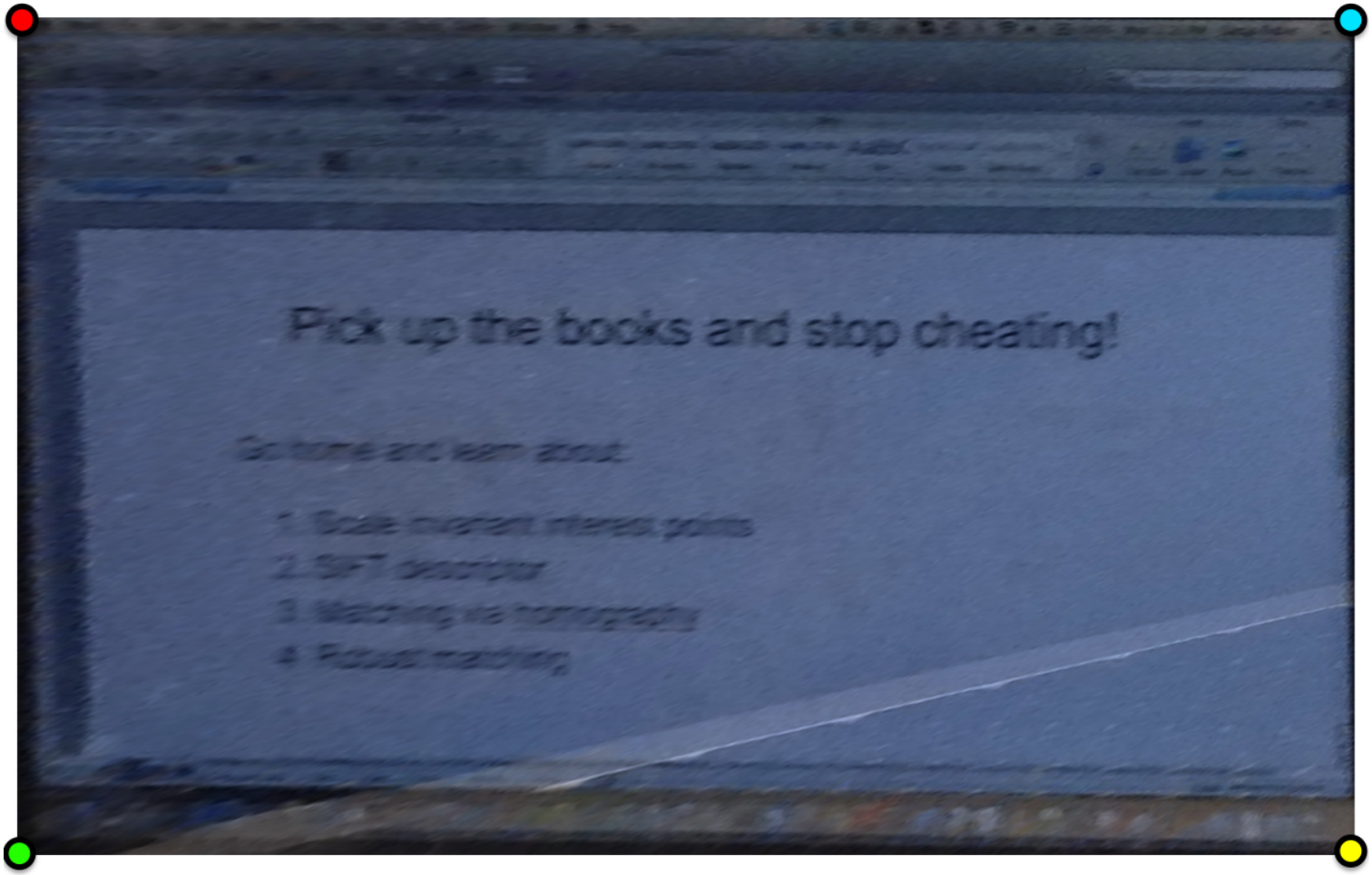

- We want it to look like this. How can we do this?

# 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

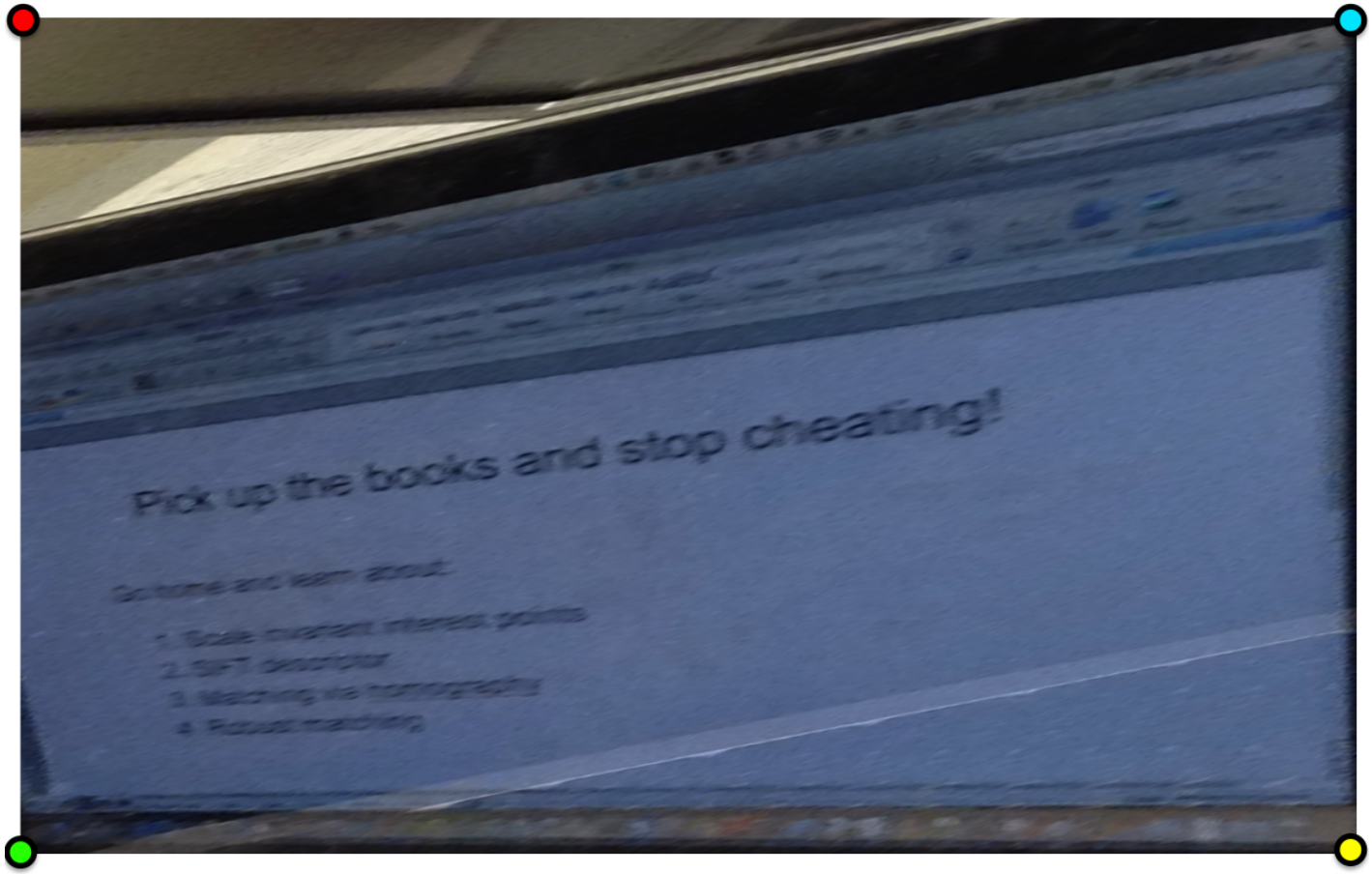


# Application 1: a Little Bit of CSI



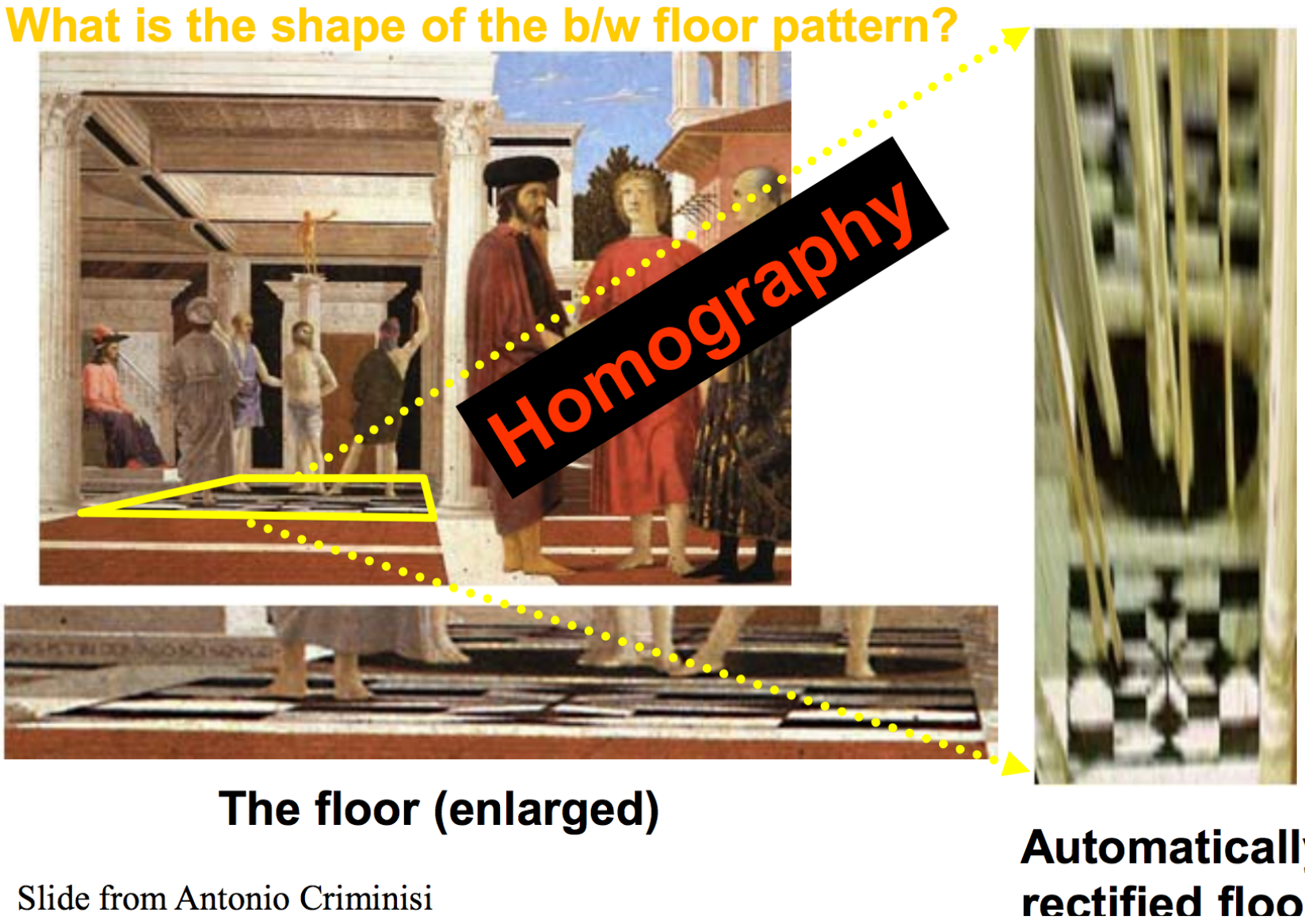

- If we compute the homography and warp the image according to it, we get this

# Application 1: a Little Bit of CSI



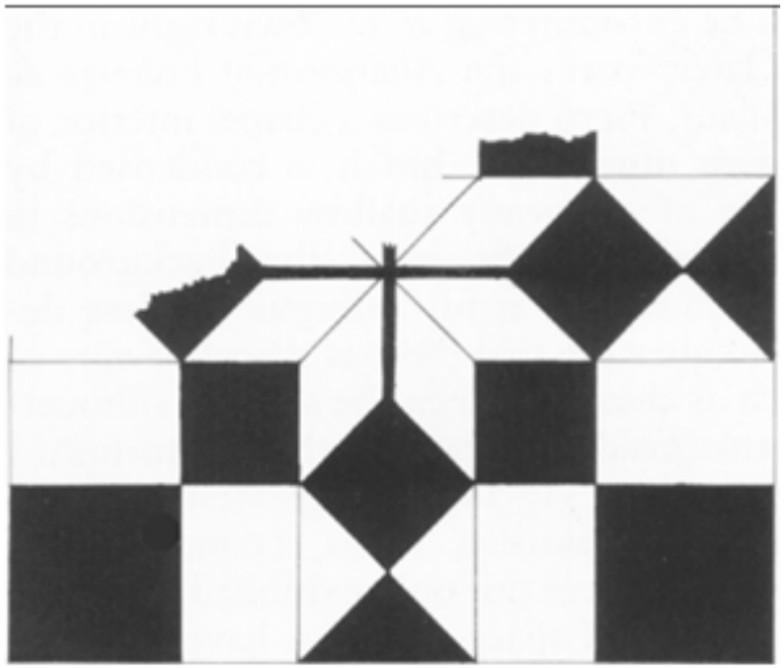

- If we used affine transformation instead, we'd get this. Would be even worse if our picture was taken closer to the laptop

# Application 1: a Little More of CSI



# Application 1: a Little More of CSI


Slide from Antonio Criminisi



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

# 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)***

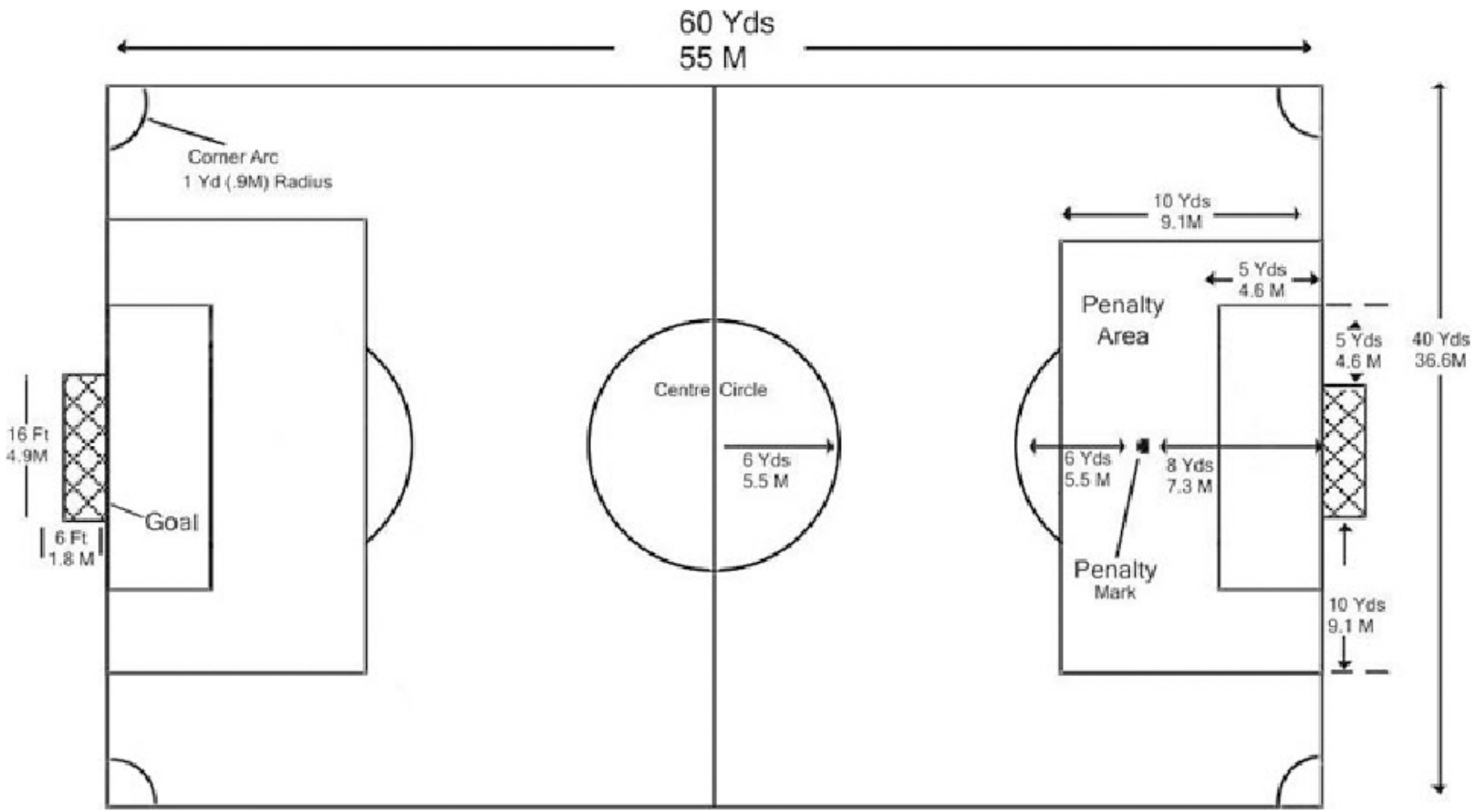
Slide from Criminisi

# 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?

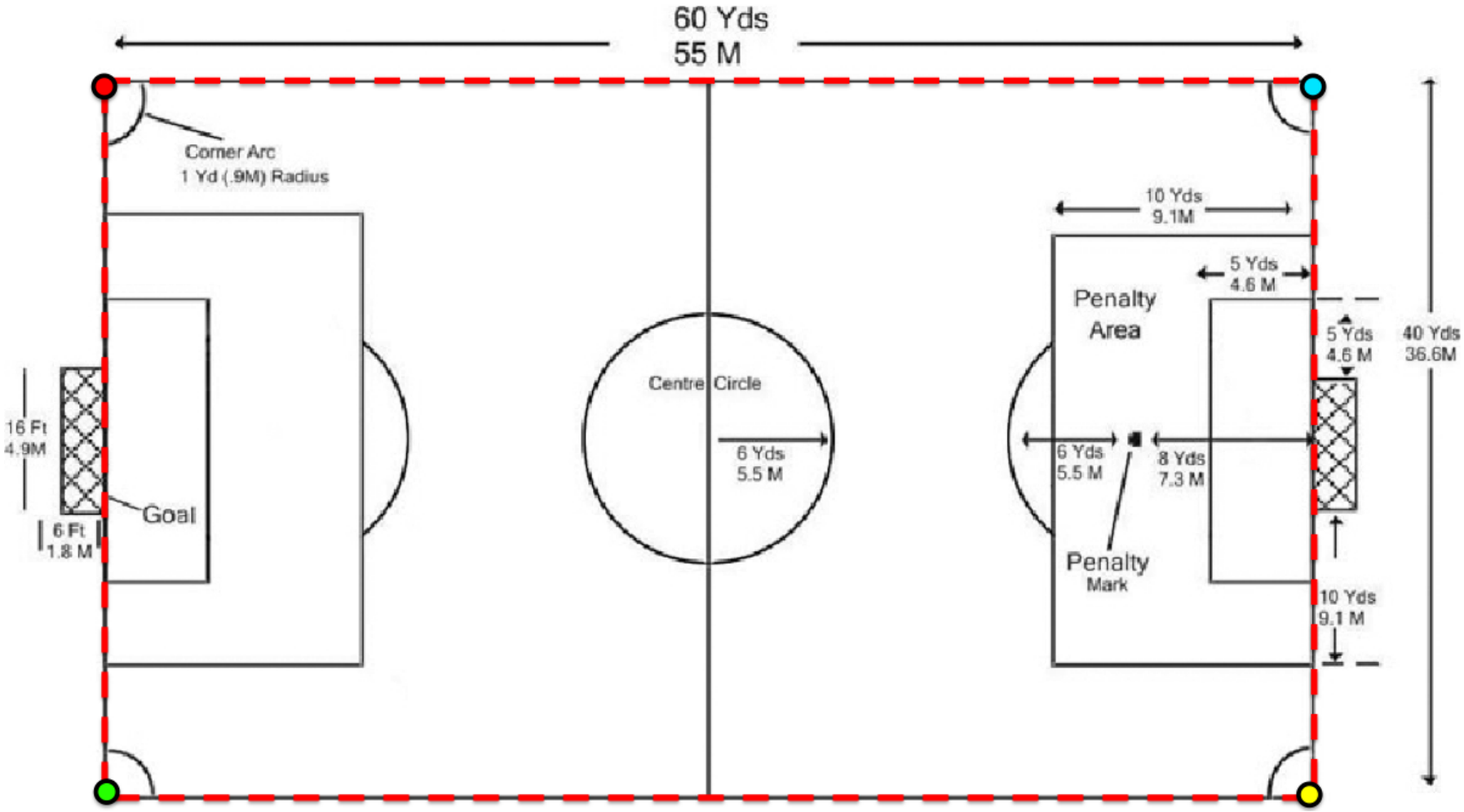



- Field is planar. We know its dimensions (look on Wikipedia).

# Application 2: How Much do Soccer Players Run?

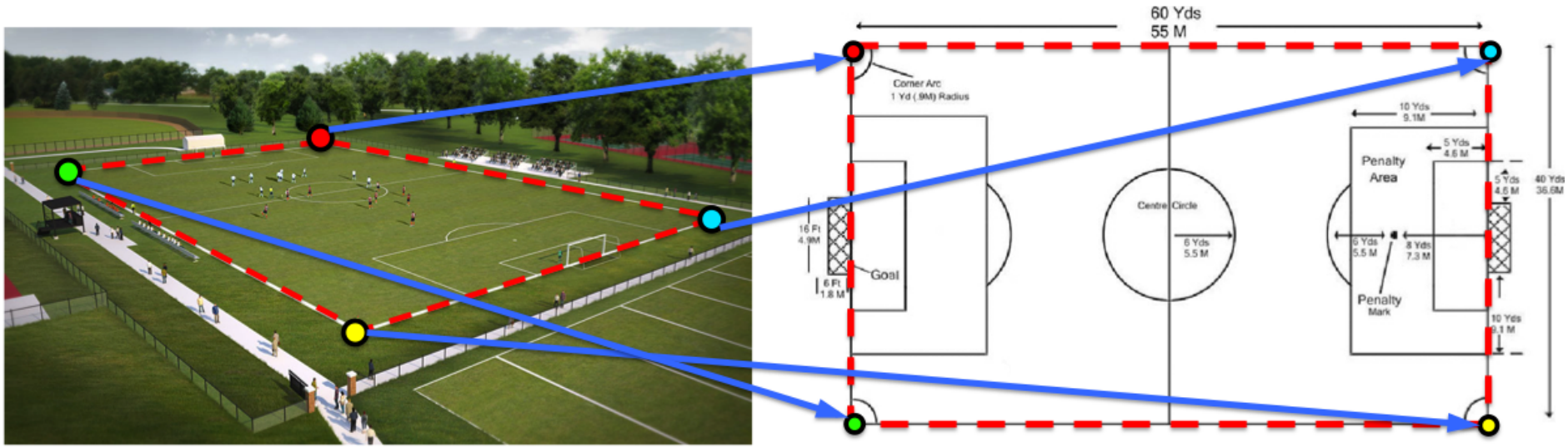
- Let's take the 4 corner points of the field

# Application 2: How Much do Soccer Players Run?



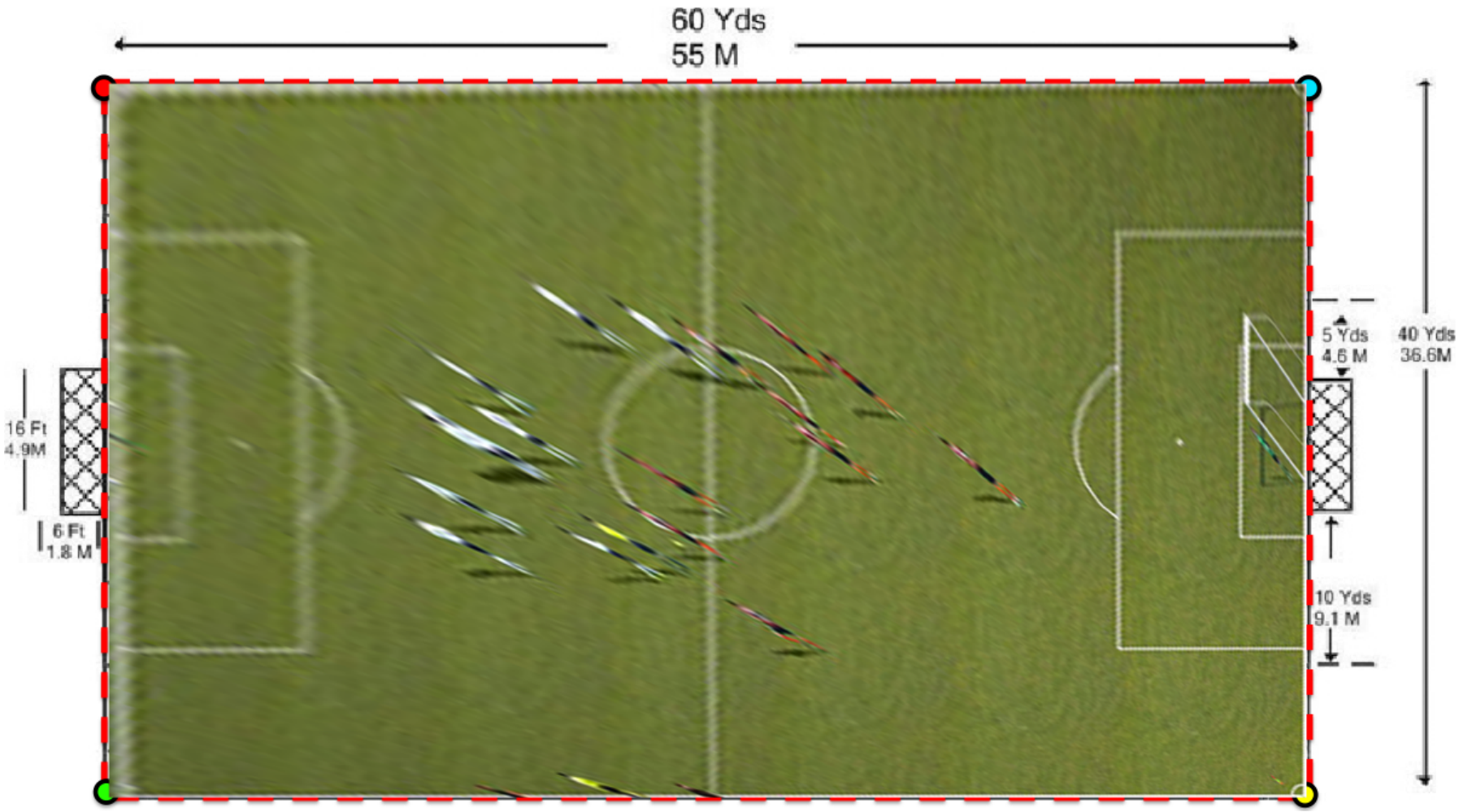

- We need to compute a homography that maps them to these 4 corners

# Application 2: How Much do Soccer Players Run?



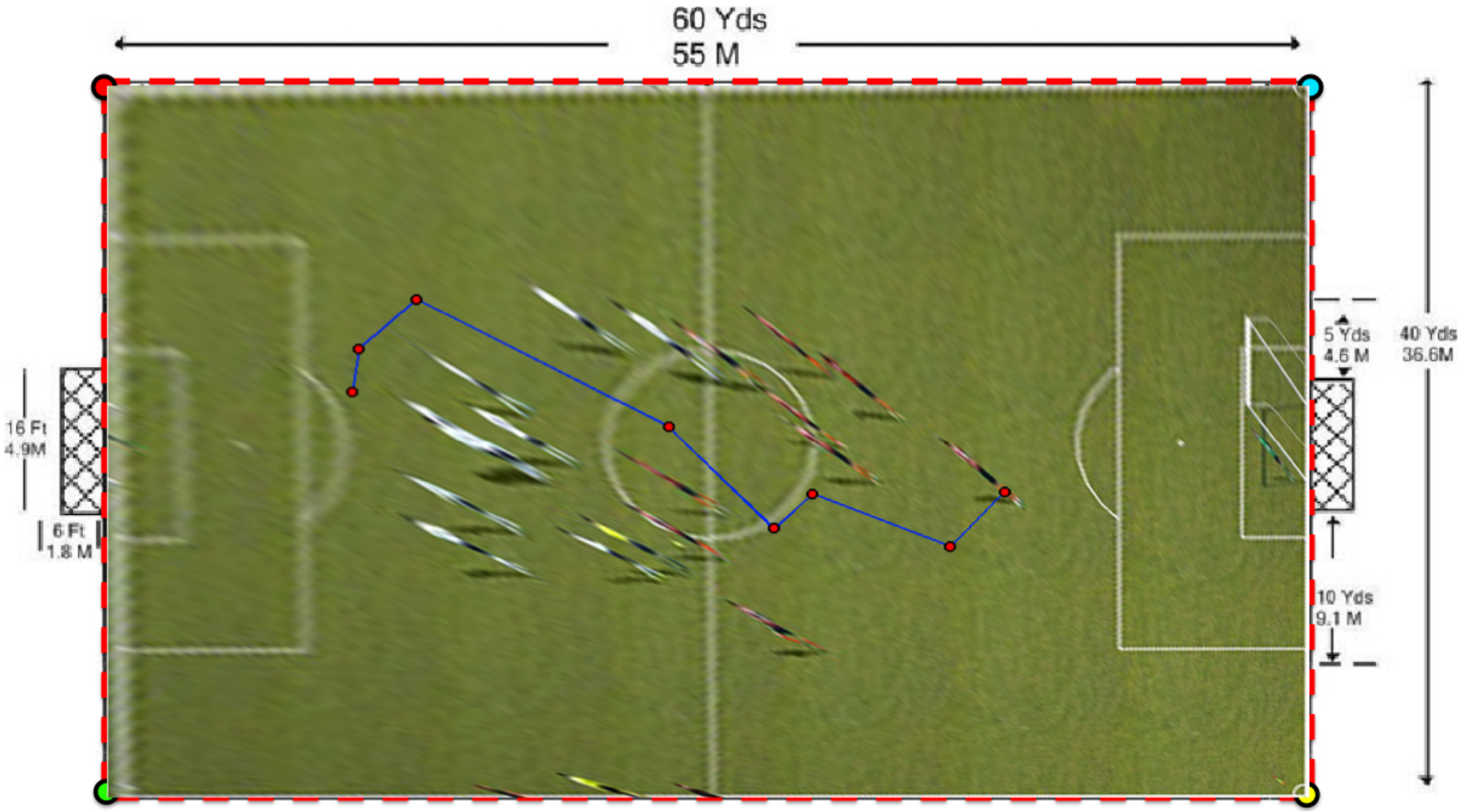

- We need to compute a homography that maps the 4 corners. Any other point from this plane (the field) also maps to the right with the same homography

# Application 2: How Much do Soccer Players Run?




- Nice. What happened to the players?

# Application 2: How Much do Soccer Players Run?

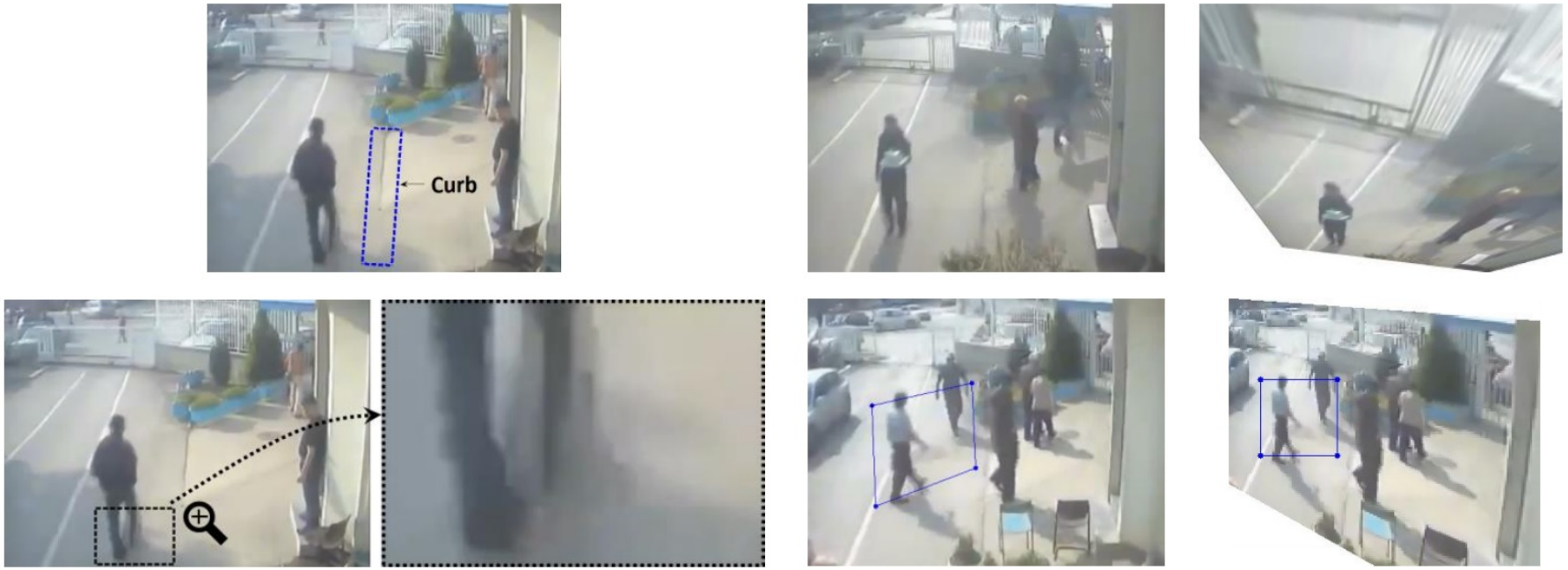



- We can now also transform the player's trajectory $\rightarrow$ and we have it in meters!

# Application 2: How Much do Soccer Players Run?

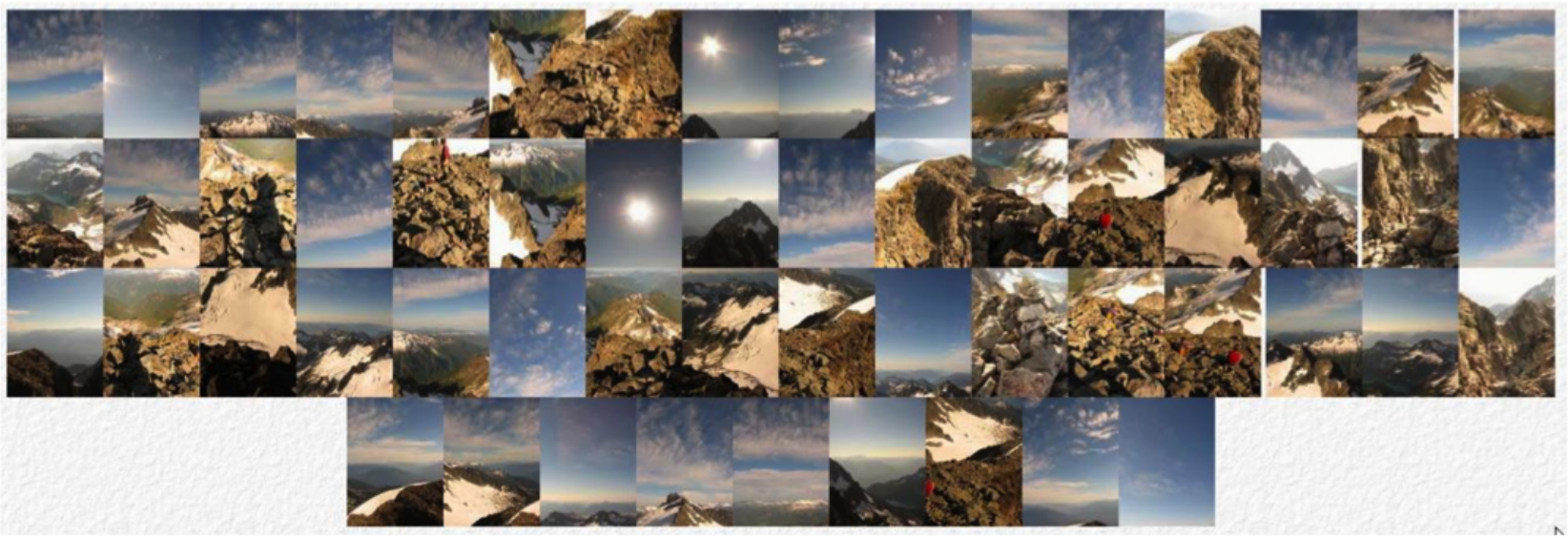
- If we used affine transformation... Our estimations of running would not be accurate!

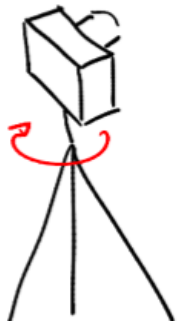
# Application 3: Video Analysis of "YouTube Funnies"




- Video Analysis of YouTube Funnies to Aid the Study of Human Gait and Falls

# Application 4: Panorama Stitching





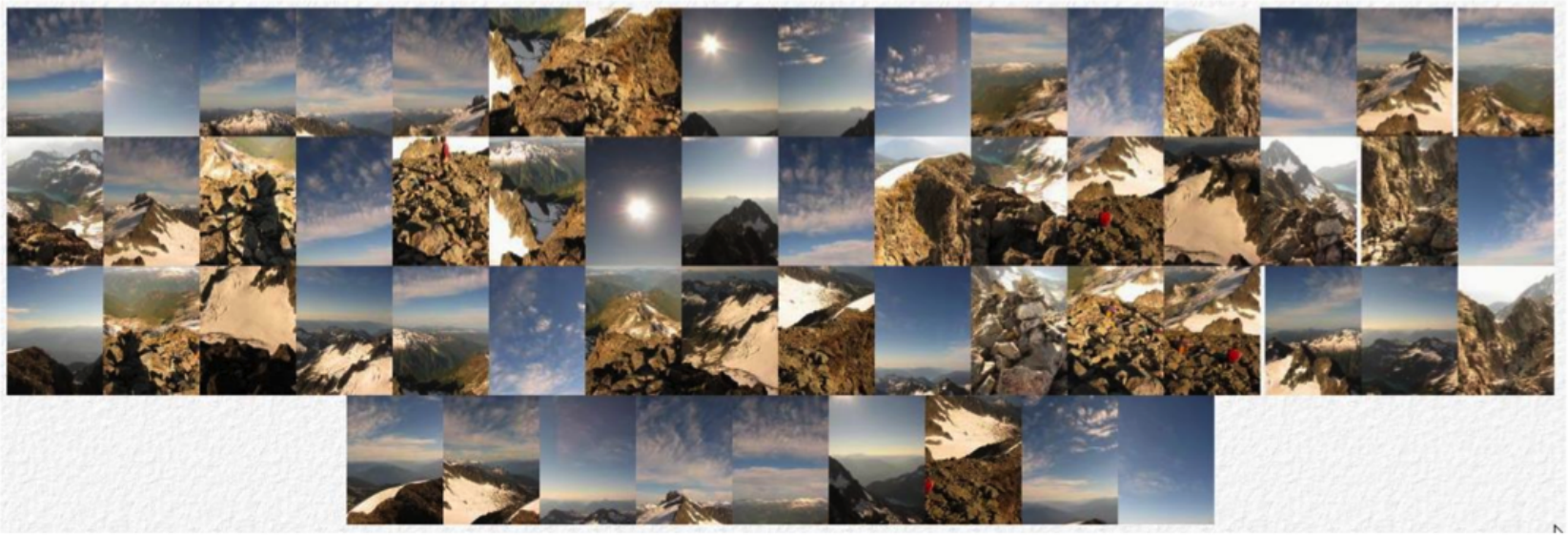
Take a tripod, rotate camera and take pictures

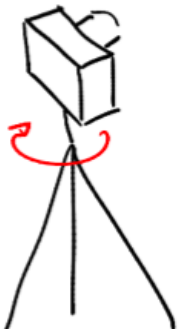
[Source: Fernando Flores-Mangas]

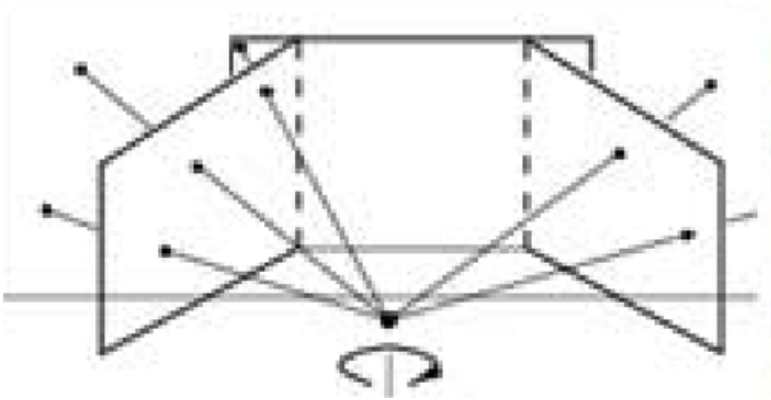
# Application 4: Panorama Stitching

[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]

# Application 3: Panorama Stitching

- To do panorama stitching, we need to:
  - Match points between pairs of images I and J
  - Compute a transformation between the between matches in I and J : a homography
  - Do it robustly (RANSAC)
  - Warp the first image to the second using the estimated homography
- Apart from the last point, this is exactly the same procedure as for the problem of matching planar objects across viewpoints
- So this should motivate the **why do I care part** of the homographies

# 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 $(x_i', y_i')$:

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

# 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 $(x_i', y_i')$:

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

- We can get rid of that $a$ on the left (we need a 2D image):

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

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

# 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 $(x_i', y_i')$:

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

- We can get rid of that $a$ on the left (we need a 2D image):

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

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

- Hmmmm... Can I still rewrite this into a linear system in $h$?

# Solving for homographies

- From:

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

- We can easily get this:

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

- Rewriting it a little:

$$\begin{aligned} h_{00}x_i + h_{01}y_i + h_{02} - x_i' \left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= 0 \\ h_{10}x_i + h_{11}y_i + h_{12} - y_i' \left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= 0 \end{aligned}$$

# Solving for homographies

- We can re-write these equations:

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

- as a linear system!

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

# Solving for homographies

- Taking all our matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_{9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

# Solving for homographies

- Taking all our matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2^2$$

# Solving for homographies

- Taking all our matches into account:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2^2$$

- Can we use Moore-Penrose like last time?

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- This is an example of "Constrained" Optimization

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n\times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since **h** is only defined up to scale, solve for unit vector
- This is an example of "Constrained" Optimization
- Method of Lagrange Multipliers

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n\times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- This is an example of "Constrained" Optimization
- Method of Lagrange Multipliers
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n\times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- This is an example of "Constrained" Optimization
- Method of Lagrange Multipliers
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- How many matches do I need to estimate $H$?

# Solving for homographies Pt.2

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n\times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- This defines a constrained least squares problem:

$$\min_{\mathbf{h}} E = ||\mathbf{Ah}||_2^2 \quad (1)$$

$$s.t. \quad ||\mathbf{h}||_2 = 1 \quad (2)$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- This is an example of "Constrained" Optimization
- Method of Lagrange Multipliers
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- How many matches do I need to estimate $H$?
- Works with 4 or more matches, only 8 unknowns!

[Source: R. Urtasun]

# Image Alignment Algorithm: Homography

Given images $I$ and $J$

1. Compute image features for $I$ and $J$
2. Match features between $I$ and $J$

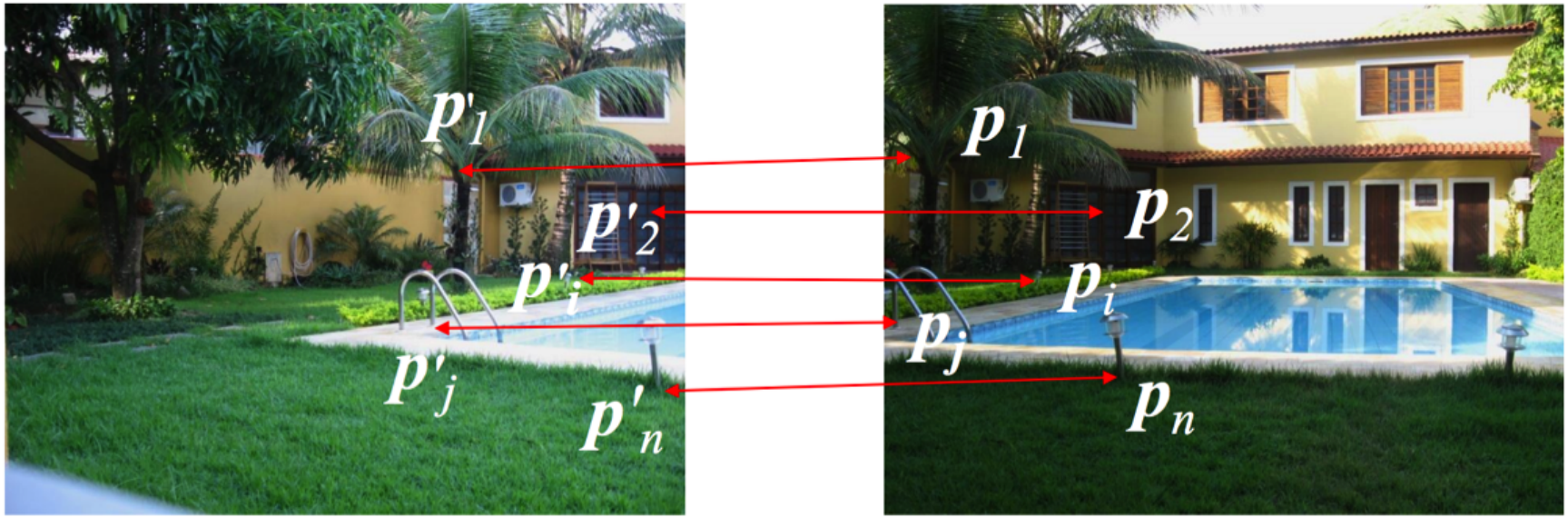
# Image Alignment Algorithm: Homography

Given images $I$ and $J$

1. Compute image features for $I$ and $J$
2. Match features between $I$ and $J$
3. Compute **homography** transformation $A$ between $I$ and $J$ (with RANSAC)

[Source: N. Snavely]

# Panorama Stitching: Example 1




- Compute the matches

[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 1

- Estimate the homography and warp

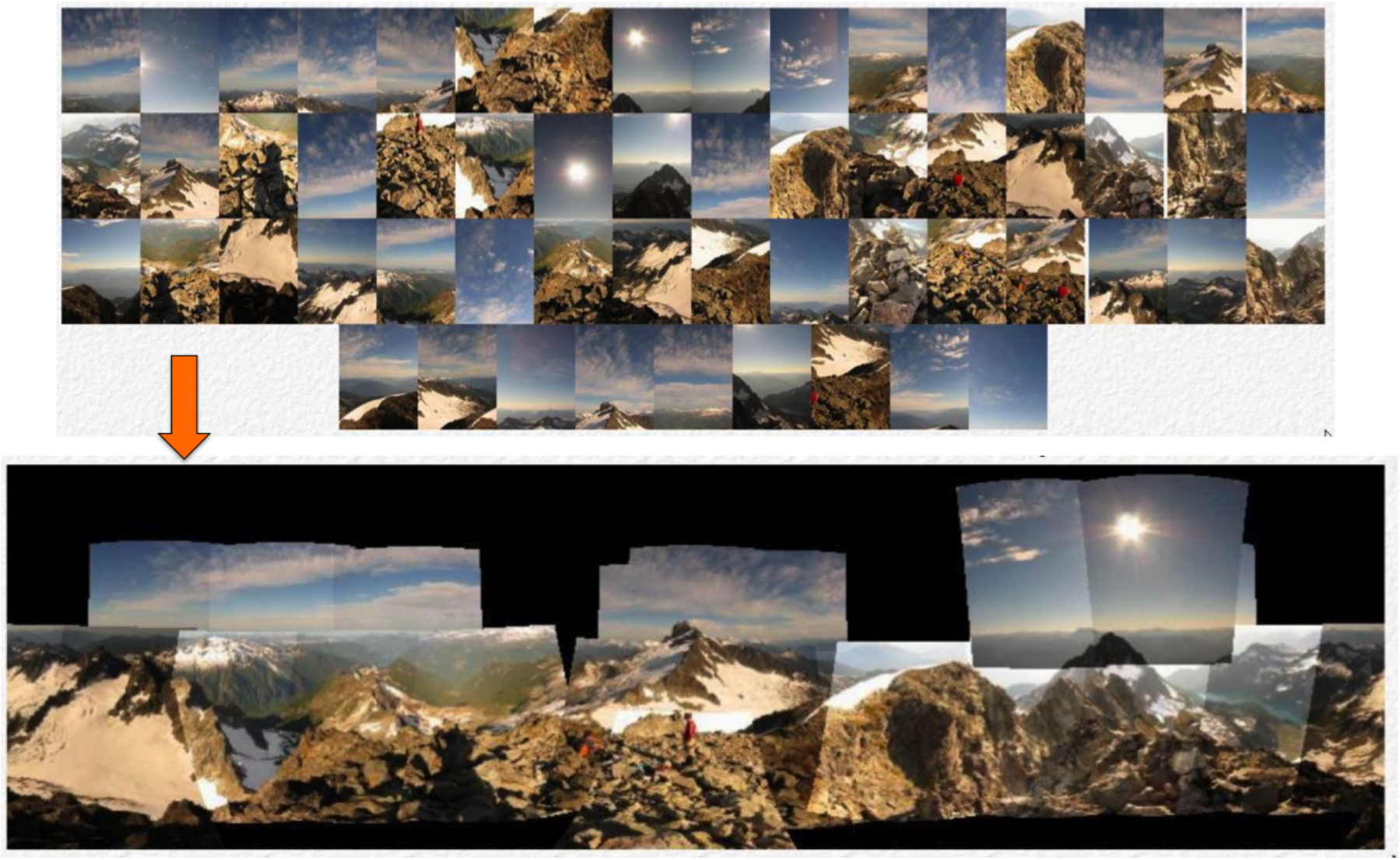
[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 1

- Stitch

[Source: R. Queiroz Feitosa]

# Panorama Stitching: Example 2

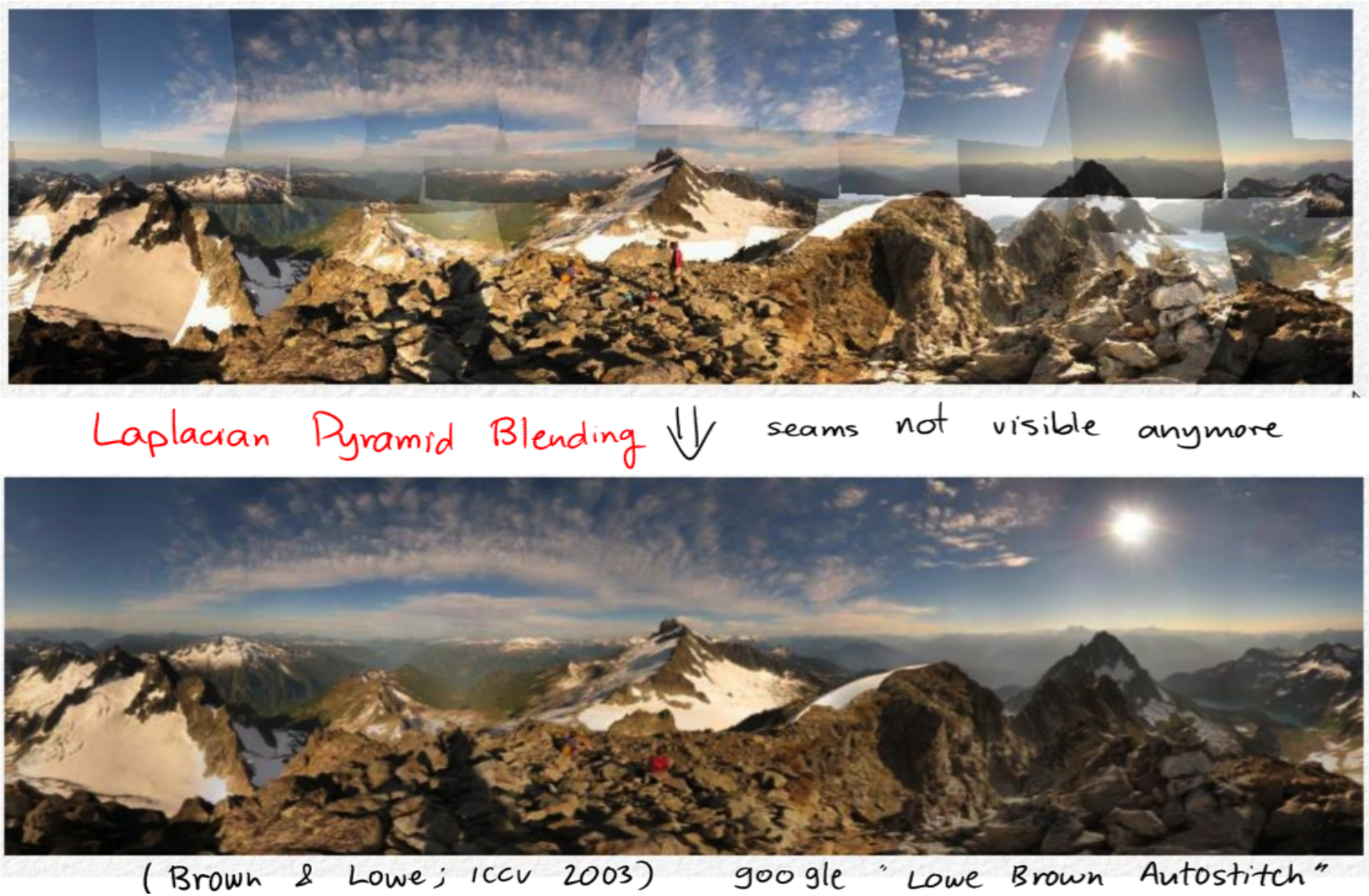


[Source: Fernando Flores-Mangas]

# Panorama Stitching: Example 2

[Source: Fernando Flores-Mangas]

# Panorama Stitching: Example 2




[Source: Fernando Flores-Mangas]

# Summary – Stuff You Need To Know

- A homography is a mapping between projective planes
- You need at least 4 correspondences (matches) to compute it

**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 |
| Panorama Stitching | Scale Invariant Interest Points | Local feature: SIFT | All features to all features + Homography |

# More than one image

- We're in 2019...

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 2019...

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?

# Big Data

- 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