MAIA - MedicAl Imaging and Applications

Advanced Image Analysis

Lectures on Advanced Color Image Processing

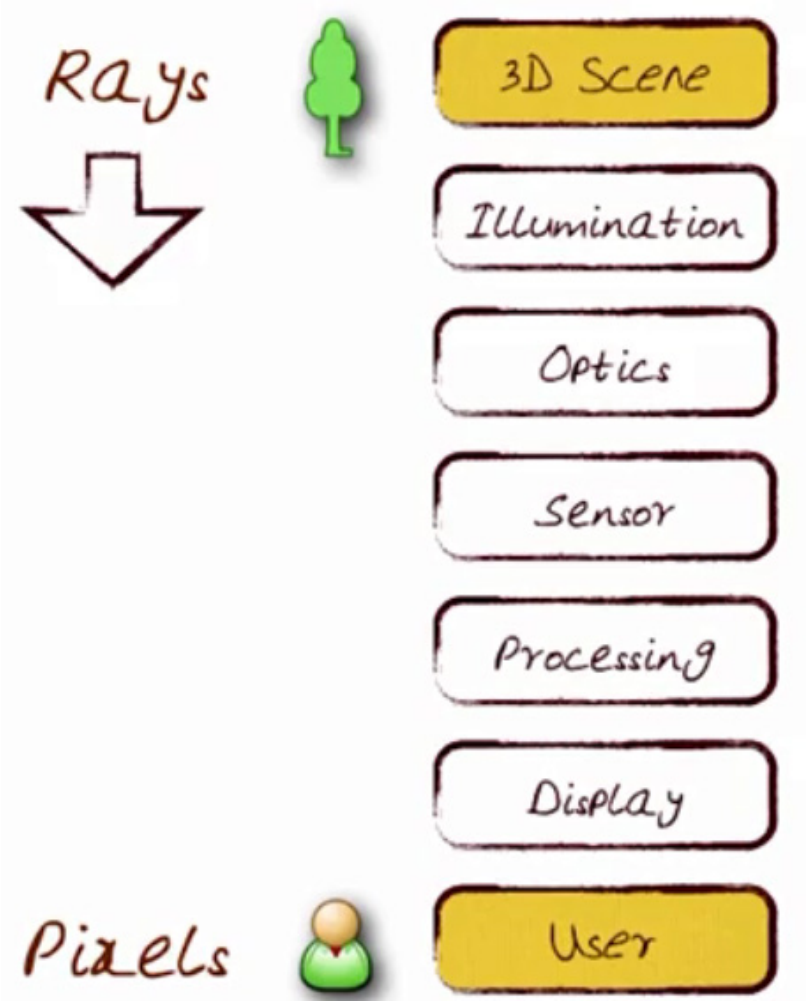
B5 - Digital Image Processing: The Camera

March/2018

Adrian Galdran - Post-Doctoral Researcher
Biomedical Imaging Lab – INESC TEC Porto (Portugal)
http://bioimglab.inesctec.pt/

1 - The Image Acquisition Pipeline

How do we go from rays to pixels?

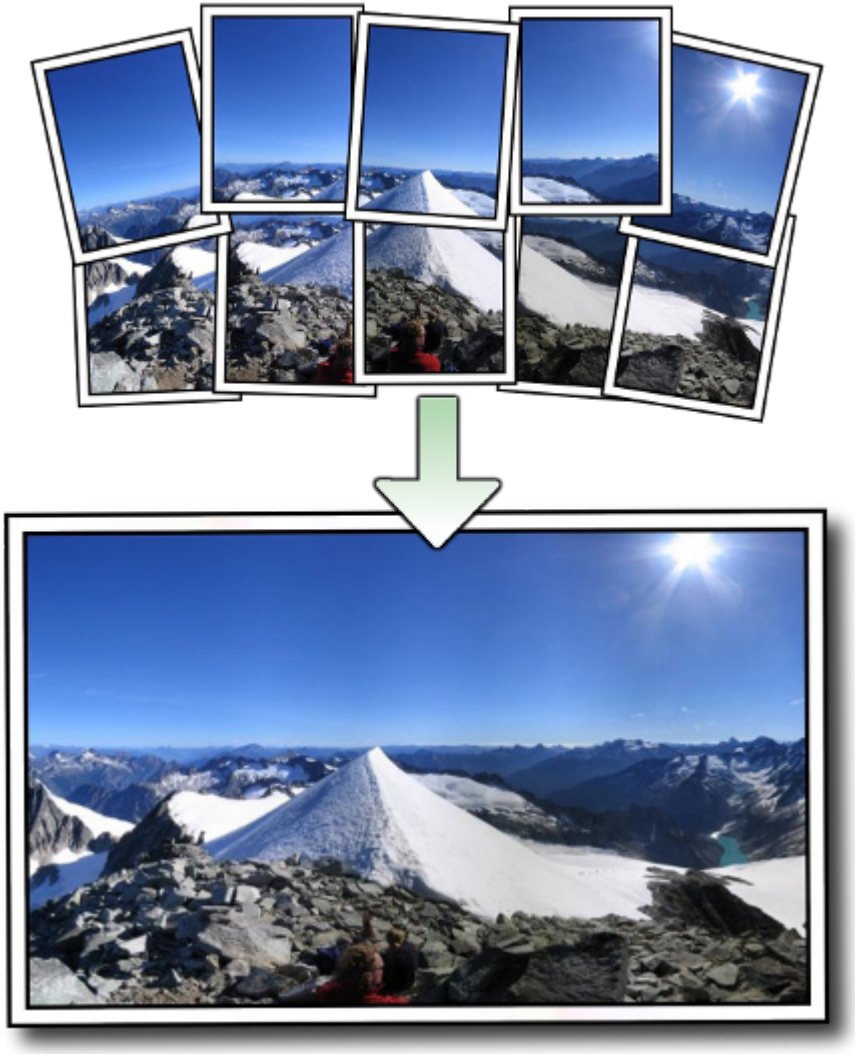


Outline of these lectures - Week 1

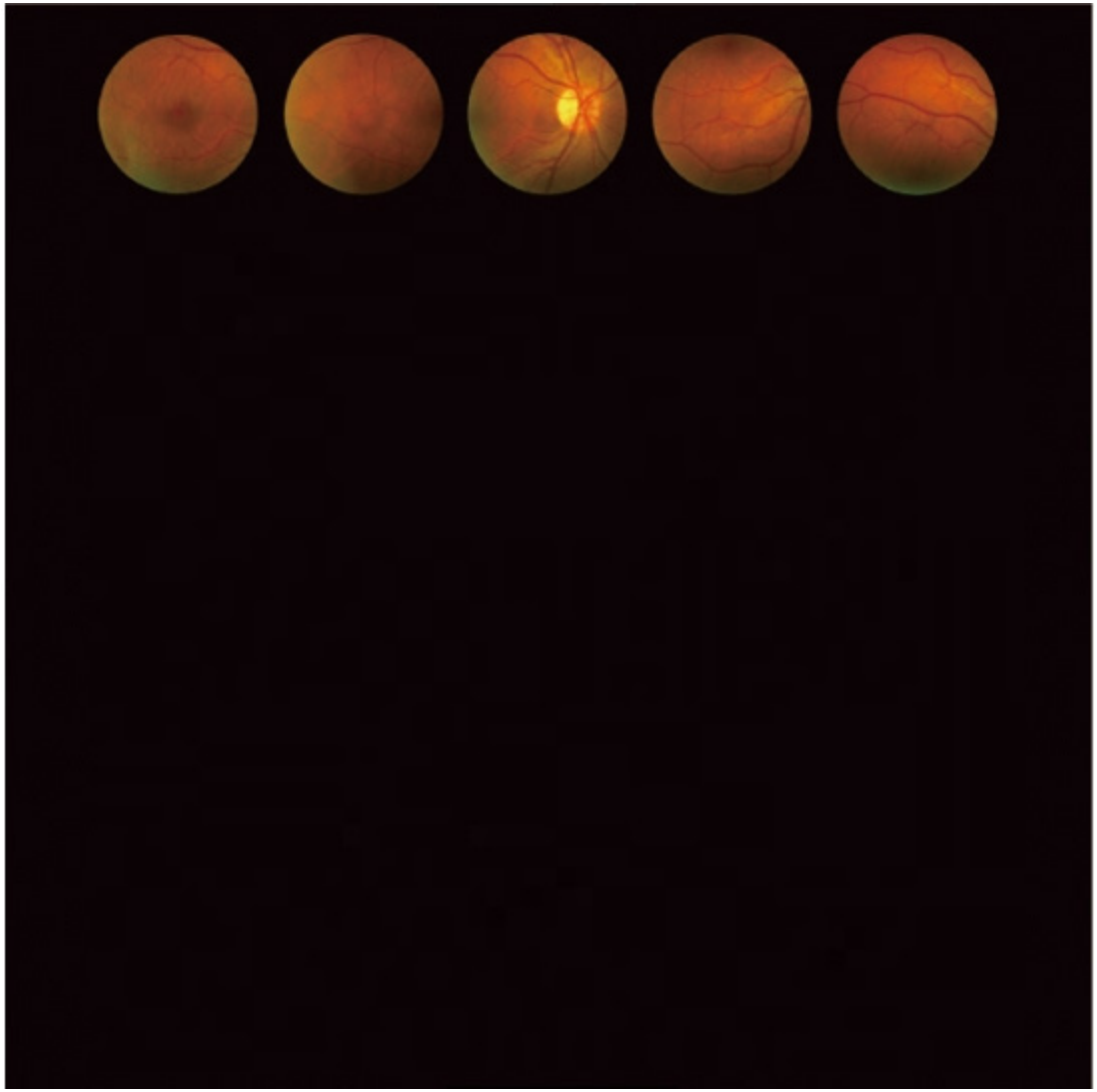
- **B1 - Human Vision System: The Retina and the Brain**
  1. **Basics of Human Perception.**
  2. **Perception and Color. Measuring Color.**
  3. **Color Spaces. Perceptually Uniform Color Spaces.**
  4. **Application: Color Image Segmentation. Superpixels.**
  5. **Introduction to the Retinex model for Human Perception.**
  6. **Introduction to Computational Image Processing.**
  7. **Application: A first example - Image Composition.**

- **B2 - Digital Image Processing: The Camera**
  1. **The Image Acquisition Pipeline.**
  2. **Cameras: Lenses, Exposure, Focus, White Balance, Sensors, Output.**
  3. **Unconventional Image Acquisition: Panorama.**

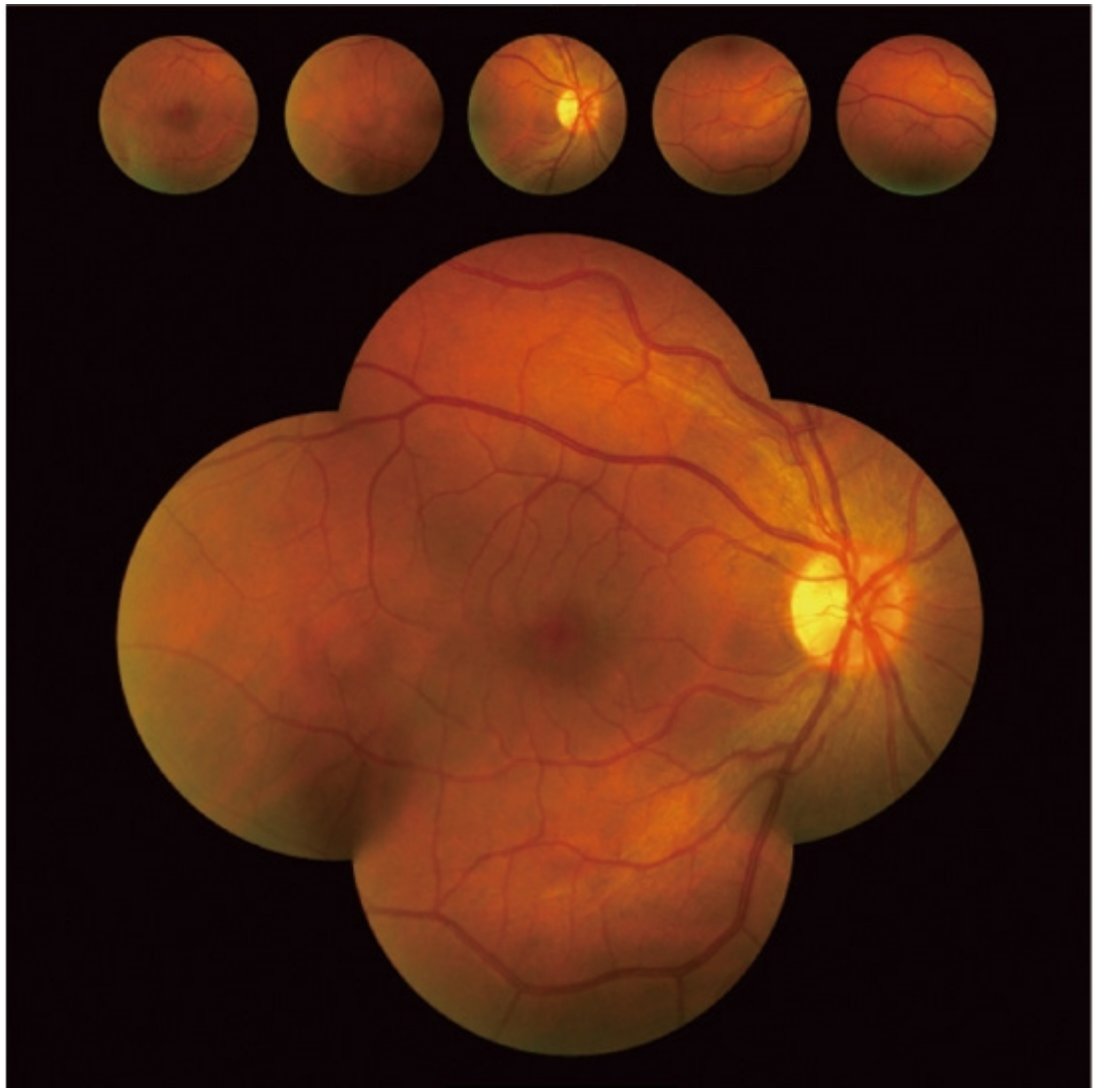
Extending the Field of View



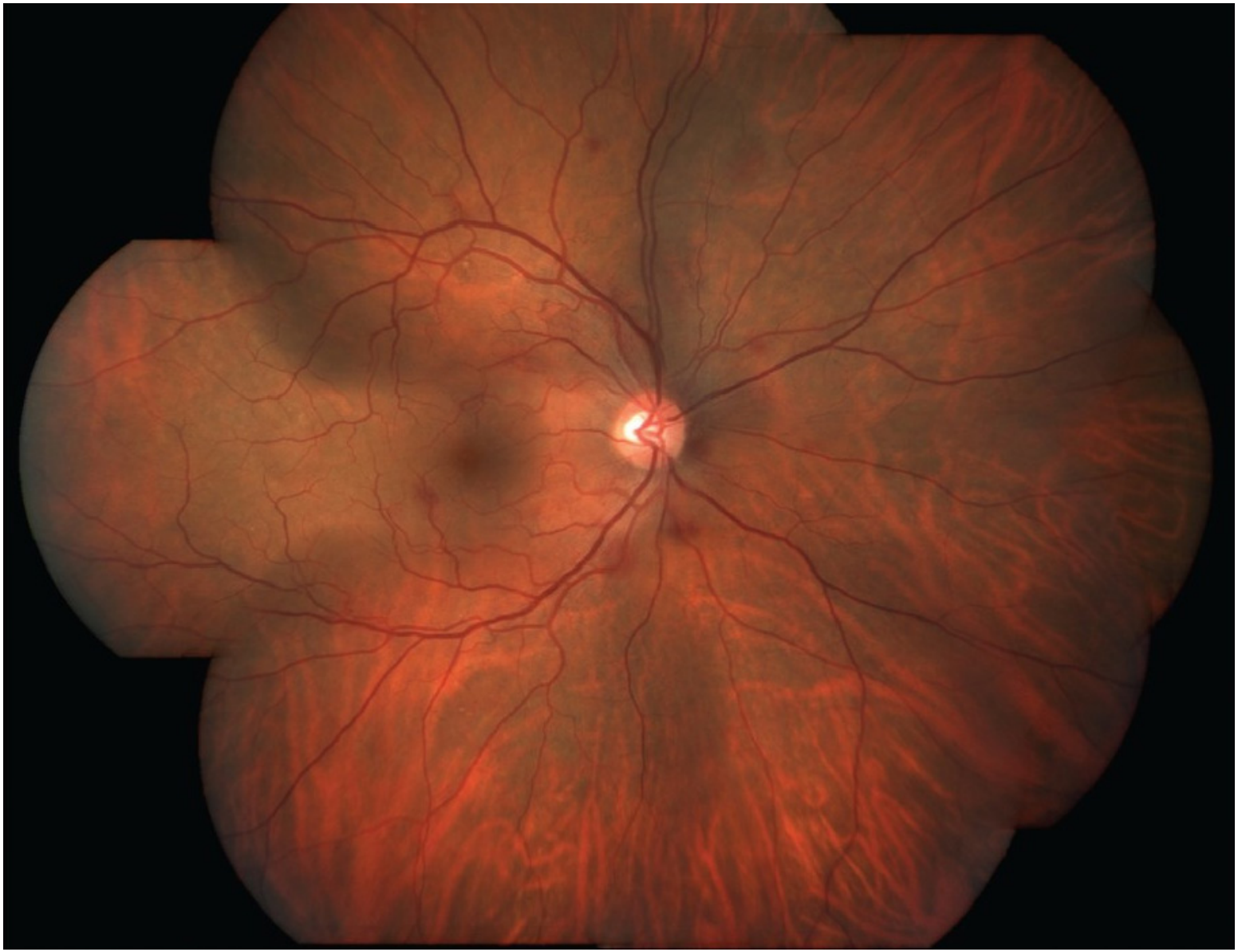
3 - Unconventional Image Acquisition: Panorama



3 - Unconventional Image Acquisition: Panorama



3 - Unconventional Image Acquisition: Panorama



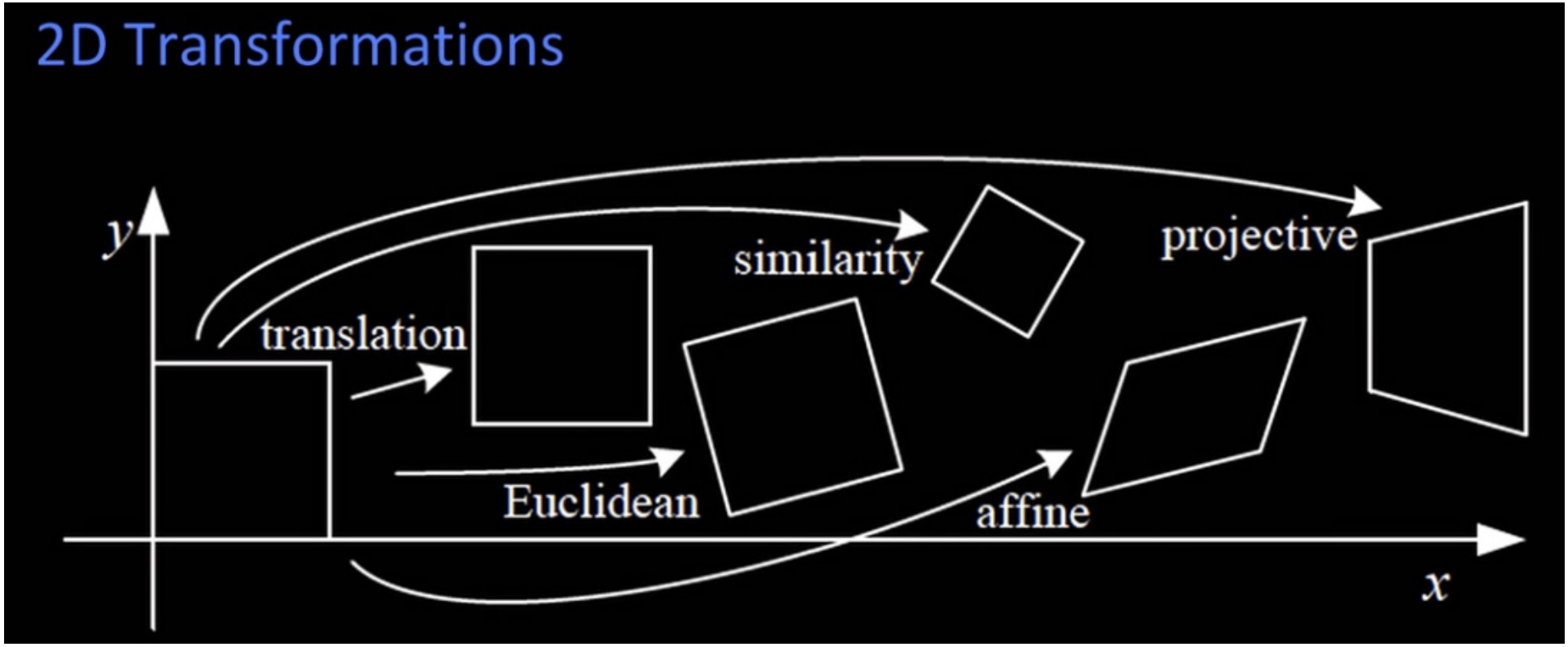
3 - Unconventional Image Acquisition: Panorama

1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. **Corner Detection**
4. **Scale-Invariant Detection**
5. **Scale-Invariant Description**
6. **RANSAC**

3 - Unconventional Image Acquisition: Panorama

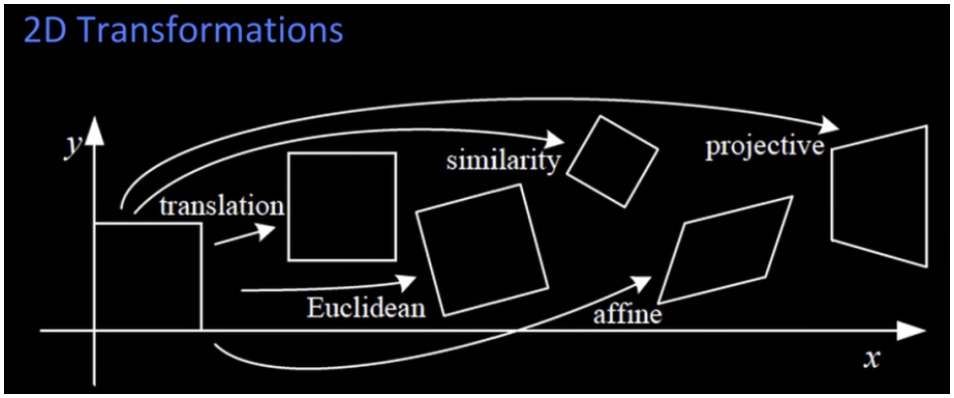
1. **Homogeneous Coordinates and Image Transforms**
2. Visual Features For Matching
3. Corner Detection
4. Scale-Invariant Detection
5. Scale-Invariant Description
6. RANSAC

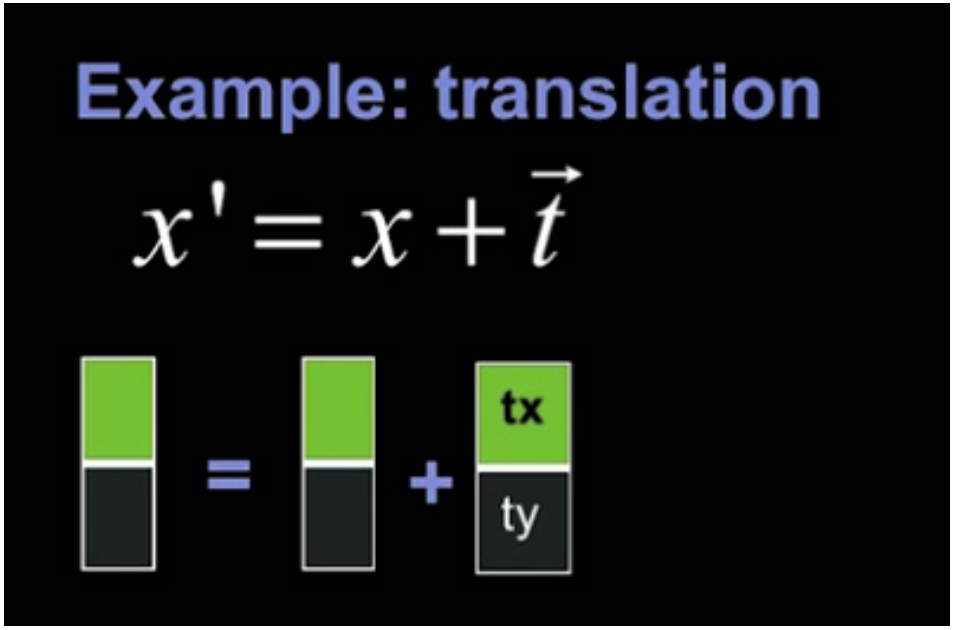
3.1 – Homogeneous Coordinates and Image Transforms



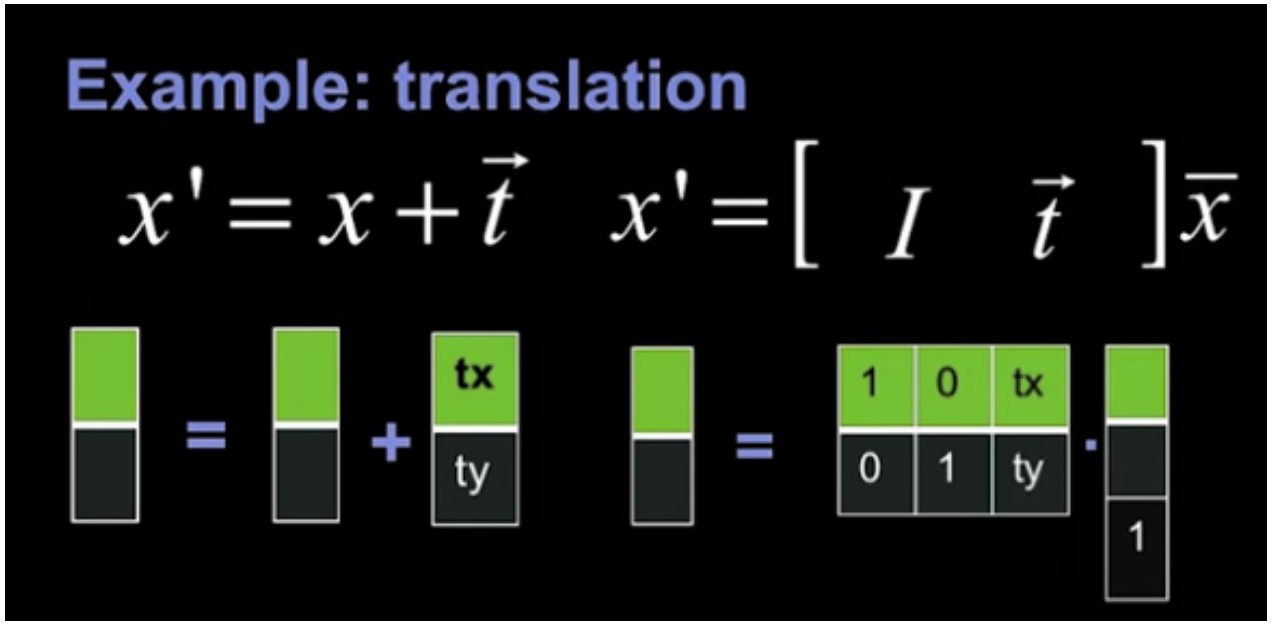
EUCLIDEAN = Rotation + Translation

3.1 – Homogeneous Coordinates and Image Transforms

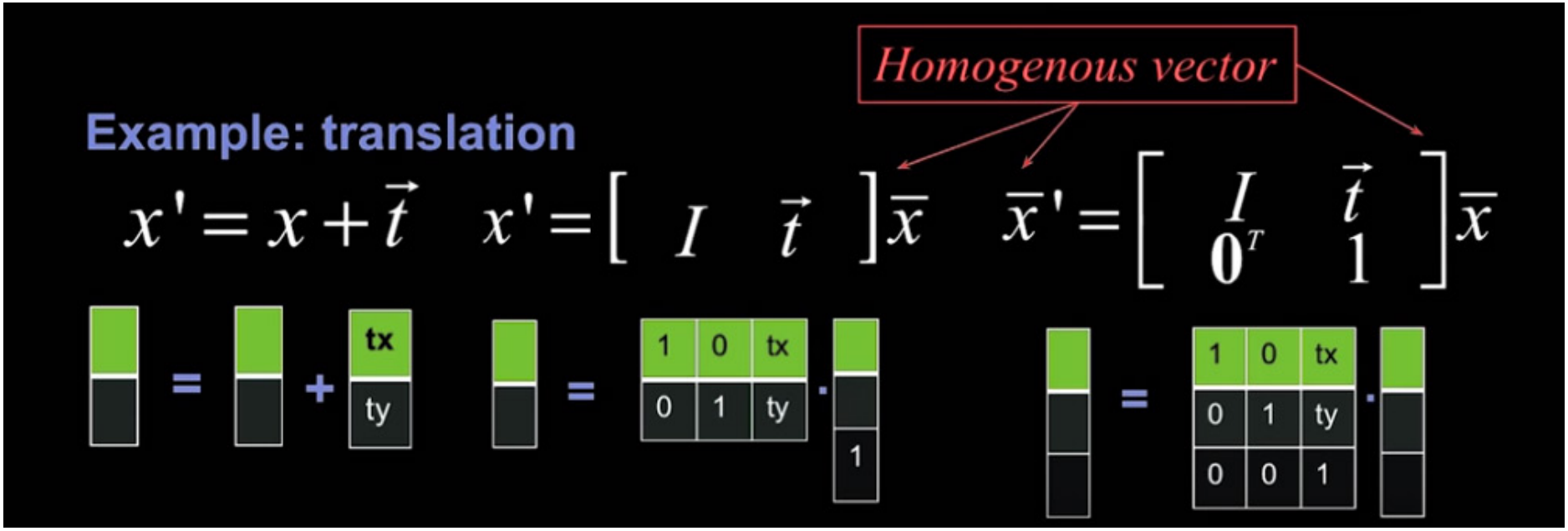




3.1 – Homogeneous Coordinates and Image Transforms



3.1 – Homogeneous Coordinates and Image Transforms



Generalization: Projective Transforms

Projective Transformations

- *Projective* transformations: for 2D images it's a 3x3 matrix applied to homogenous coordinates

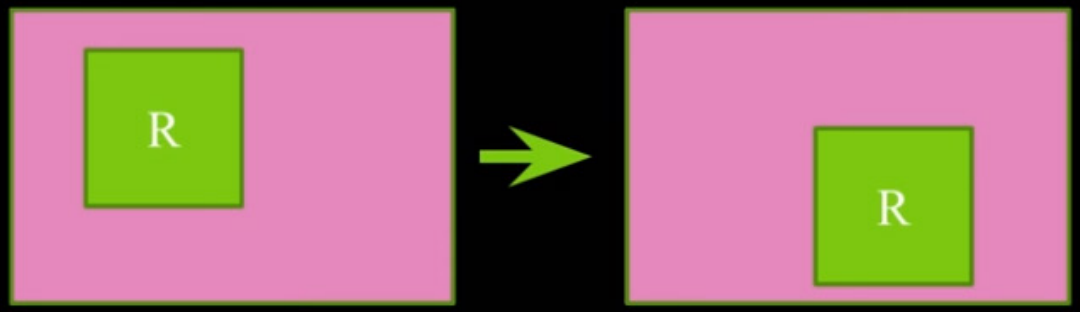
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

3.1 – Homogeneous Coordinates and Image Transforms

- Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Preserves:
  - Lengths/Areas
  - Angles
  - Orientation
  - Lines

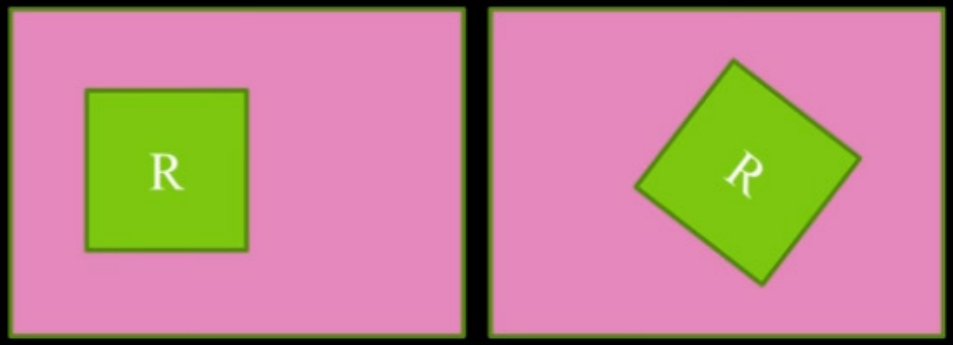


3.1 – Homogeneous Coordinates and Image Transforms

- Euclidean (Rigid body)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Preserves:
  - Lengths/Areas
  - Angles
  - Lines

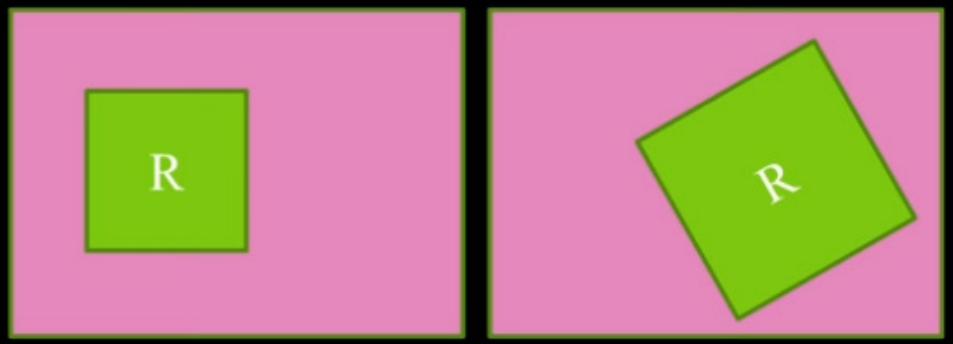


3.1 – Homogeneous Coordinates and Image Transforms

- Similarity (trans, rot, scale) transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a\cos(\theta) & -a\sin(\theta) & t_x \\ a\sin(\theta) & a\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

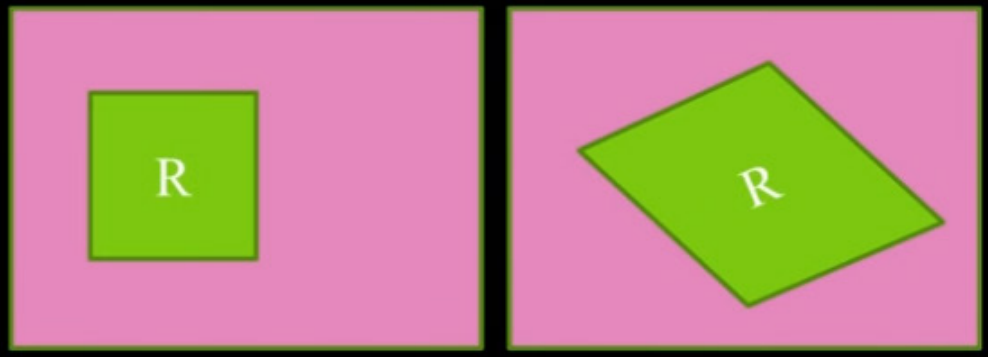
- Preserves:
  - Lengths/Areas
  - Angles
  - Lines



- Affine transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Preserves:
  - Parallel Lines
  - Ratio of Areas
  - Lines



- Remember, these are homogeneous coordinates

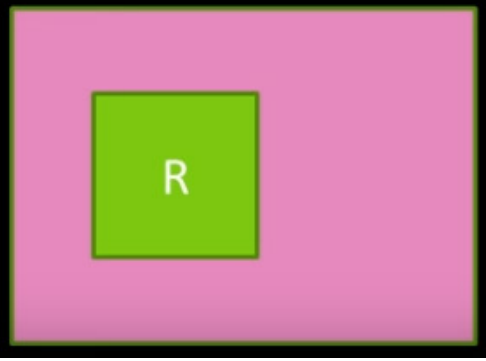
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong \begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

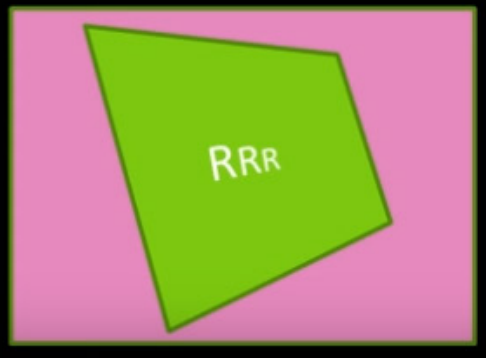
3.1 – Homogeneous Coordinates and Image Transforms

- General projective transform (or Homography)

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

- Preserves:
  - Lines
  - Also cross ratios





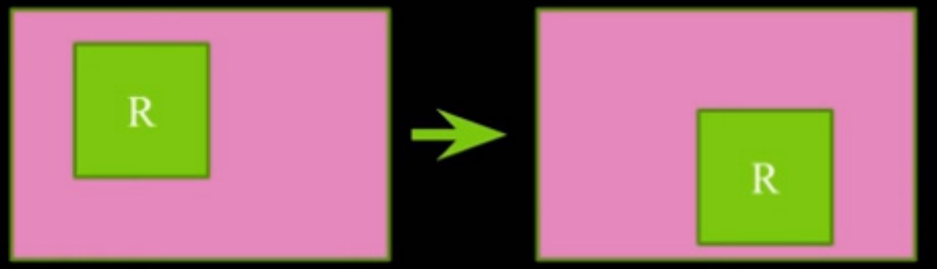
Quiz 1

Suppose I told you the transform from image A to image B is a ***translation***. How many pairs of corresponding points would you need to know to compute the transformation?

Quiz 1 – answer

- Translation: a 1 point transformation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



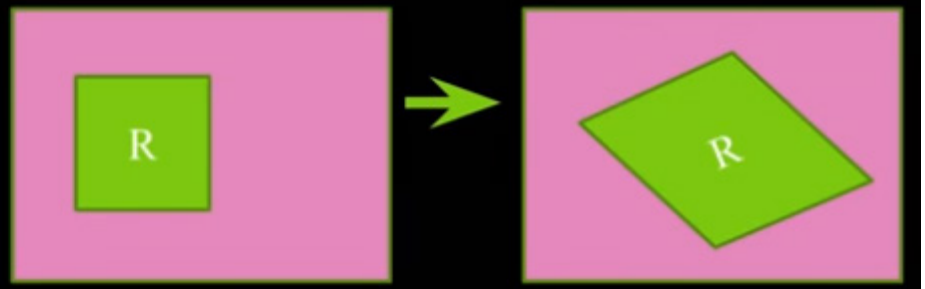
Quiz 2

Suppose I told you the transform from image A to image B is ***affine***. How many pairs of corresponding points would you need to know to compute the transformation?

Quiz 2 - answer

- Affine transform: a 3 point transformation
  - 6 unknowns – each point pair gives two equations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

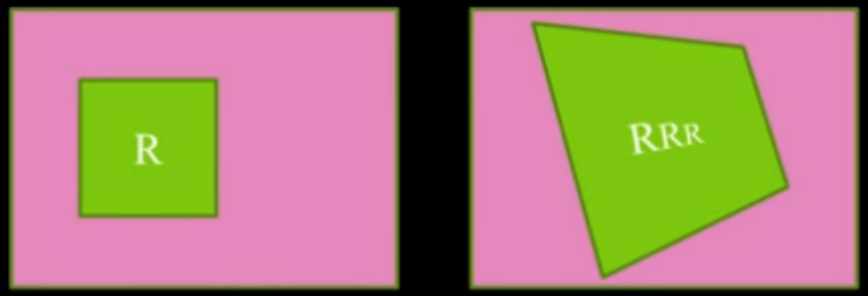


Quiz 3

Suppose I told you the transform from image A to image B is a ***homography***. How many pairs of corresponding points would you need to know to compute the transformation?

Quiz 3 -answer

- Homography



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

Homogeneous coordinates

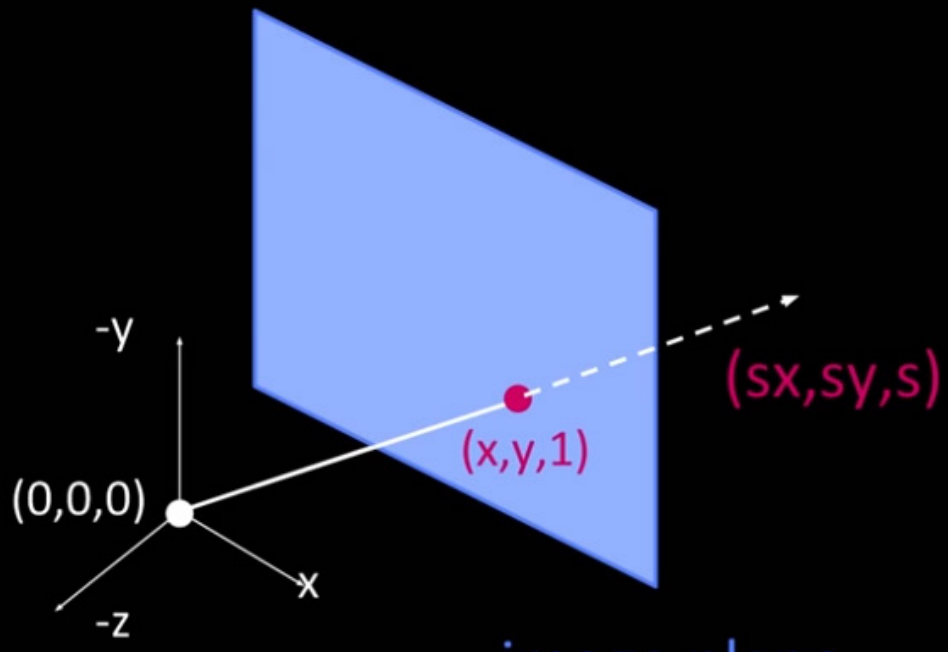
2D Points:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \longrightarrow p' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad p' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \longrightarrow p = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$$

The projective plane

What is the geometric intuition of using homogeneous coordinates ?

- A point in the image is a ray in projective space



The projective plane

Each *point* (x,y) on the plane (at z=1) is represented by a *ray* (sx, sy,s)

All points on the ray are equivalent:

$(x, y, 1) \cong (sx, sy, s)$

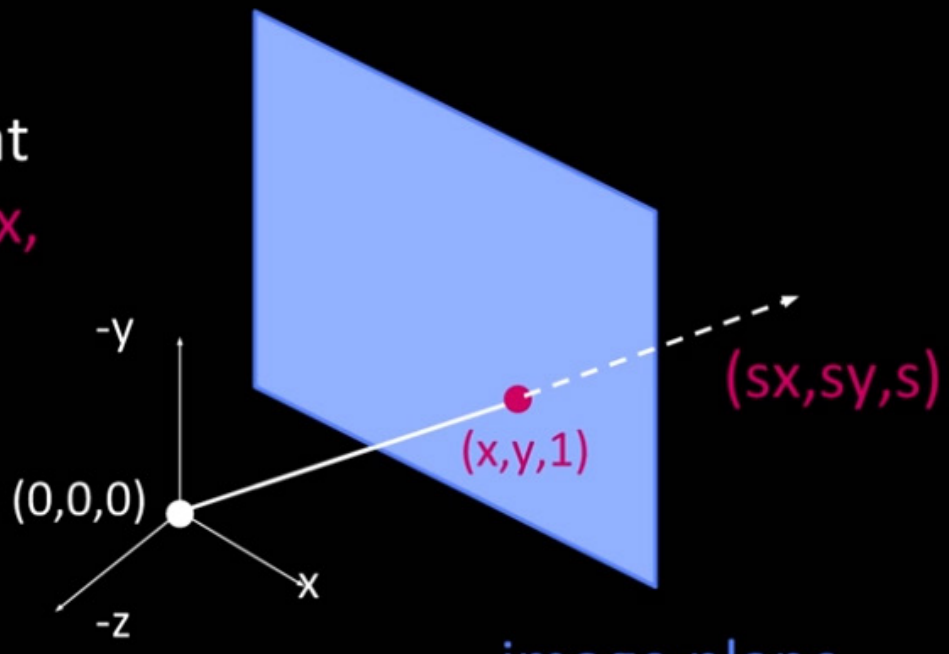


Image reprojection

Basic question:

- How to relate two images from the same camera center?
- How to map a pixel from projective plane PP1 to PP2?

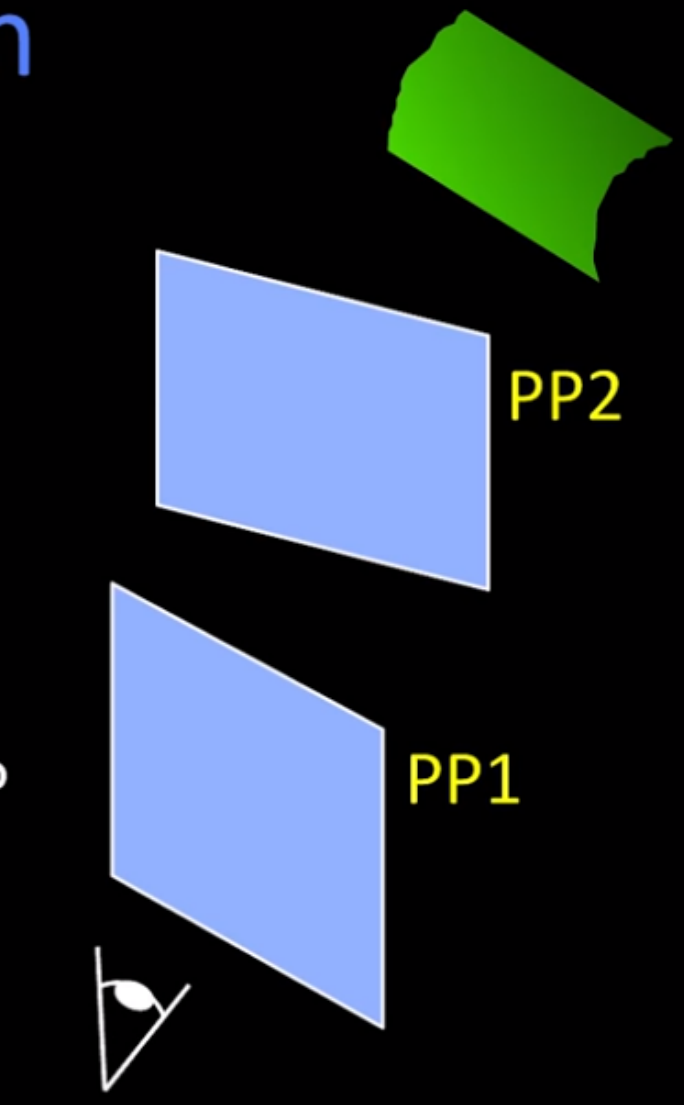


Image reprojection

Answer

- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2

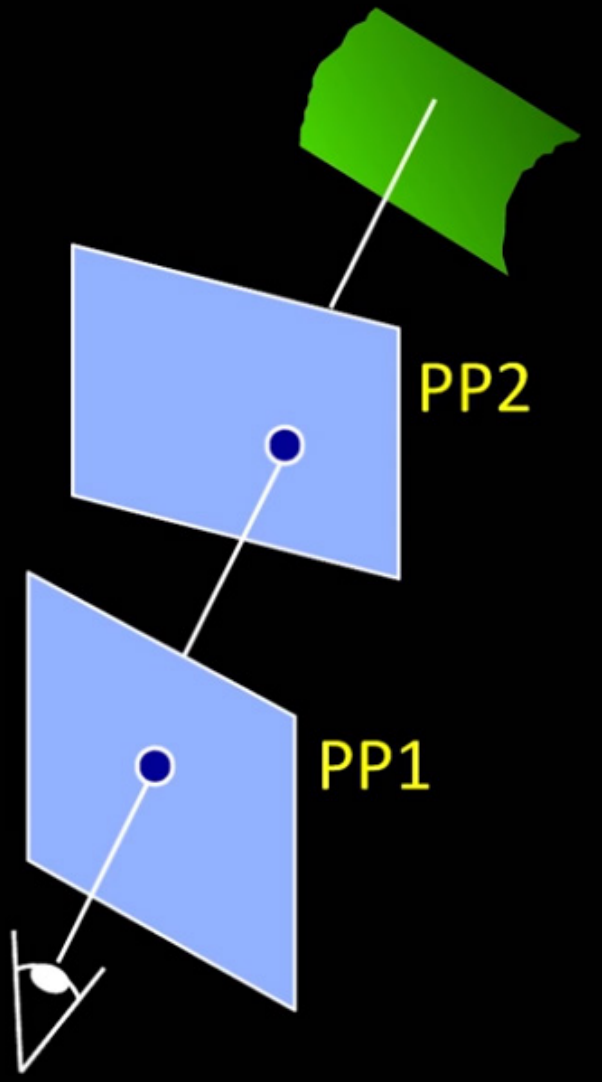
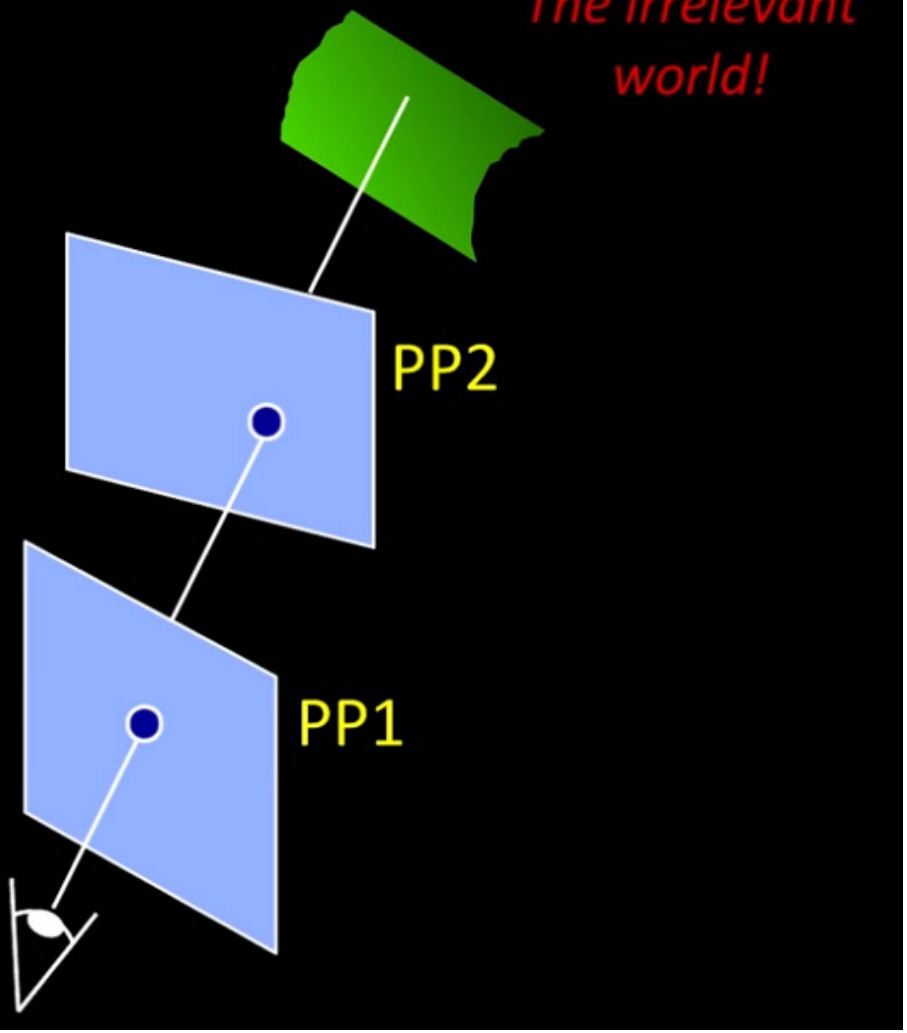
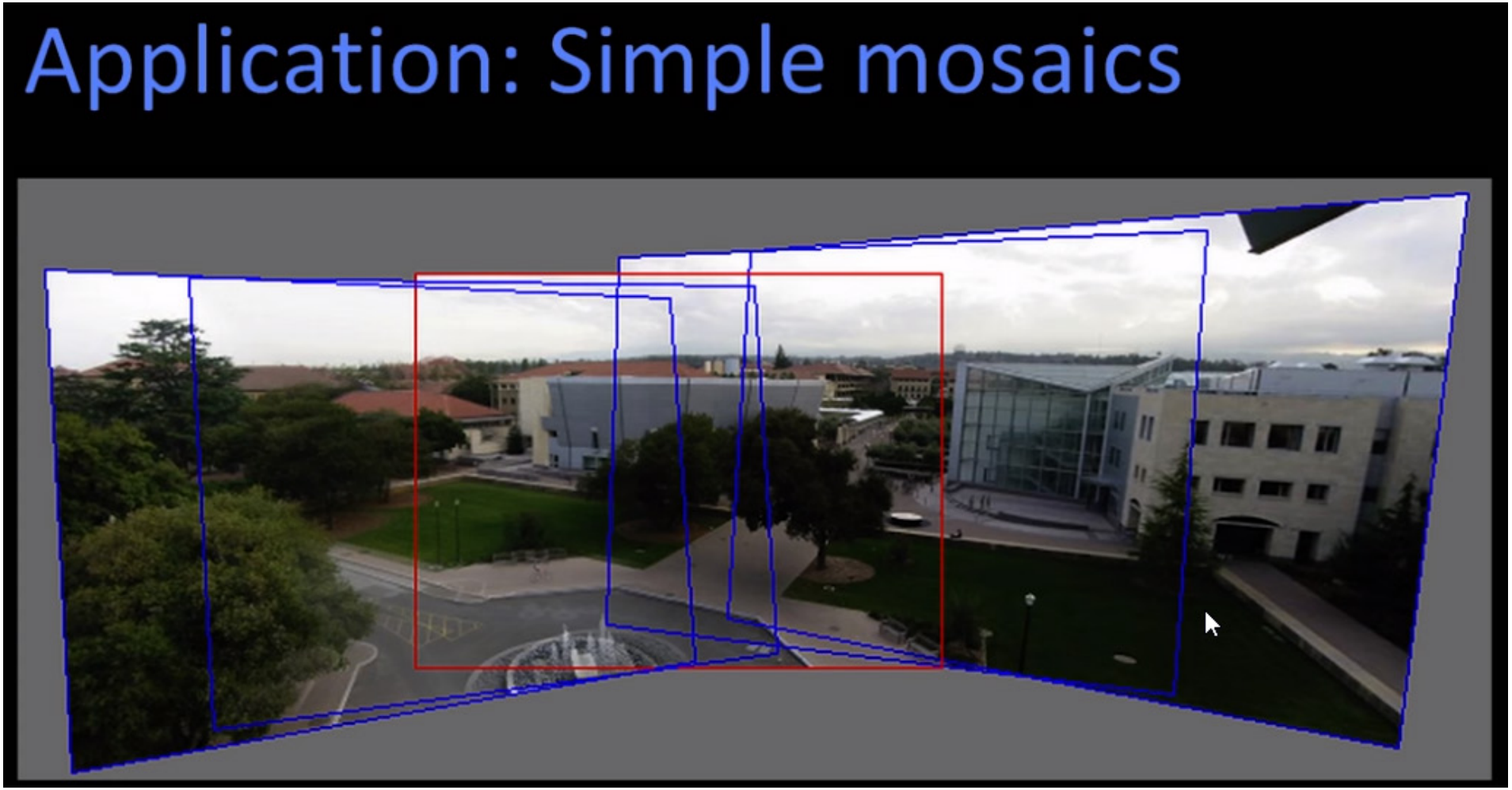


Image reprojection

Observation:

- Rather than thinking of this as a 3D reprojection, think of it as a 2D **image warp** from one image (plane) to another (plane).



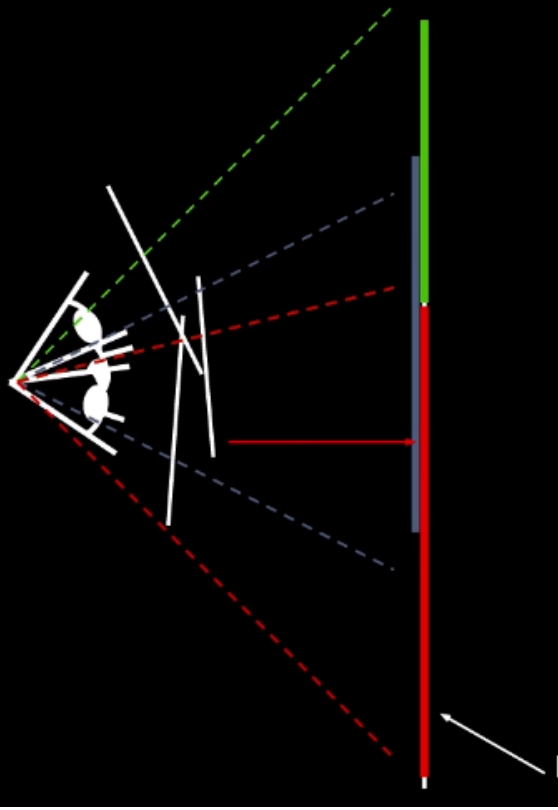

Application: Simple mosaics

How to stitch together a panorama

Basic Procedure

- Take a sequence of images from the same position
  > Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- (If there are more images, repeat)

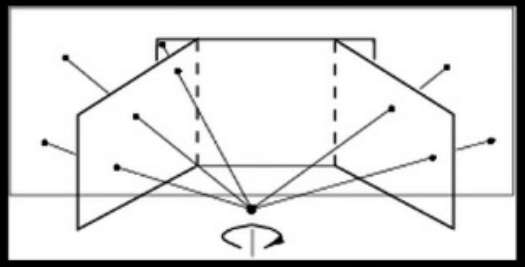
Image reprojection

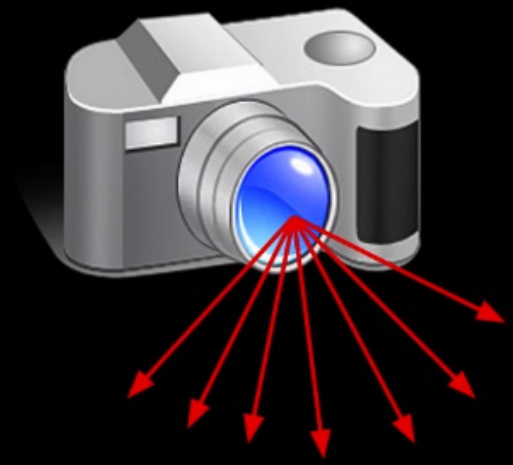


The mosaic has a natural interpretation in 3D:

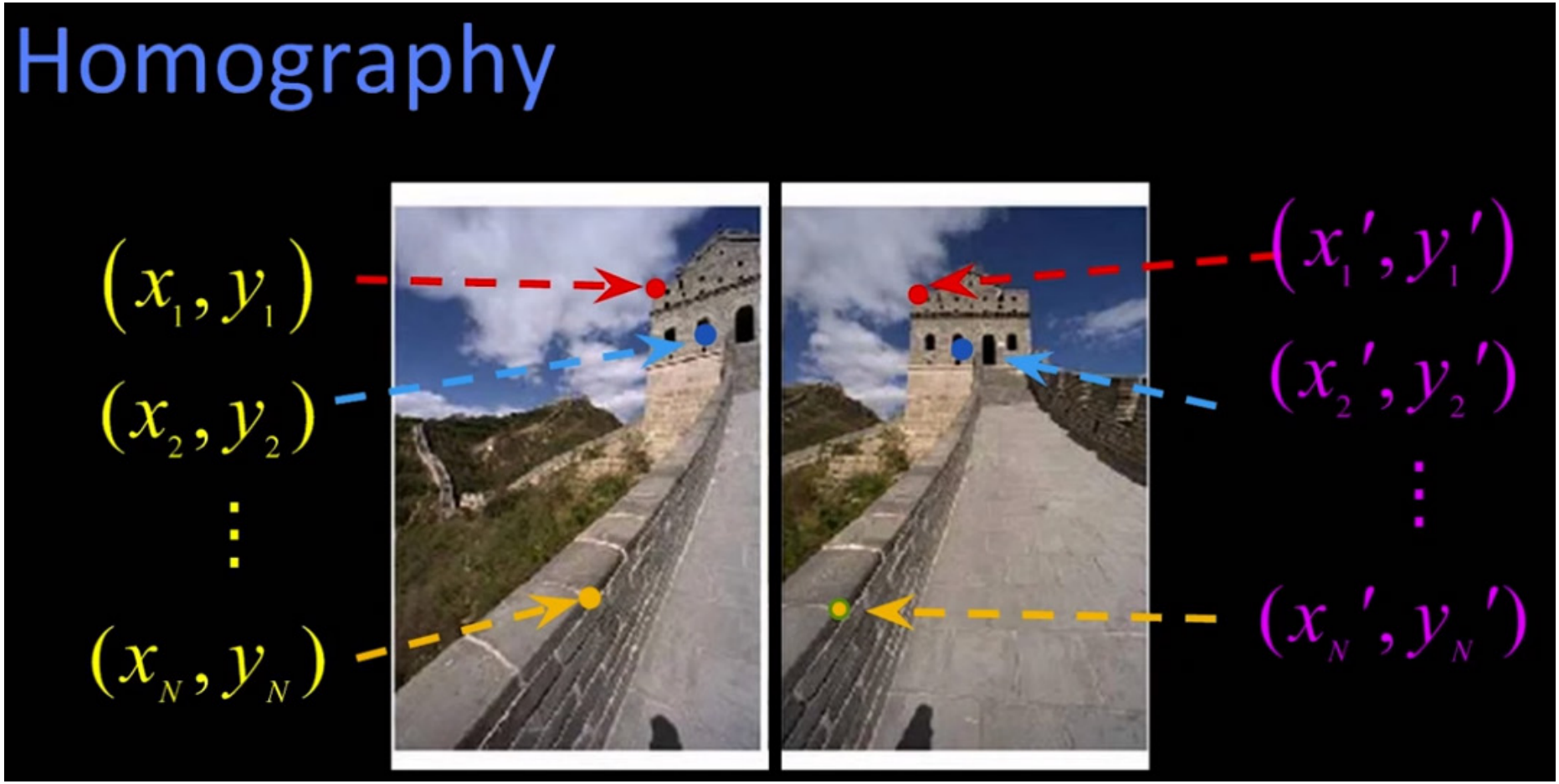
- The images are *reprojected* onto a common plane
- The mosaic is formed on this plane.

Mosaics





Obtain a wider angle view by combining multiple images ***all of which are taken from the same camera center***.


Homography
(x_1, y_1)
(x_2, y_2)
(x_N, y_N)
(x_1', y_1')
(x_2', y_2')
(x_N', y_N')

Solving for homographies

$$\mathbf{p'} = \mathbf{Hp} \qquad \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Solving for homographies

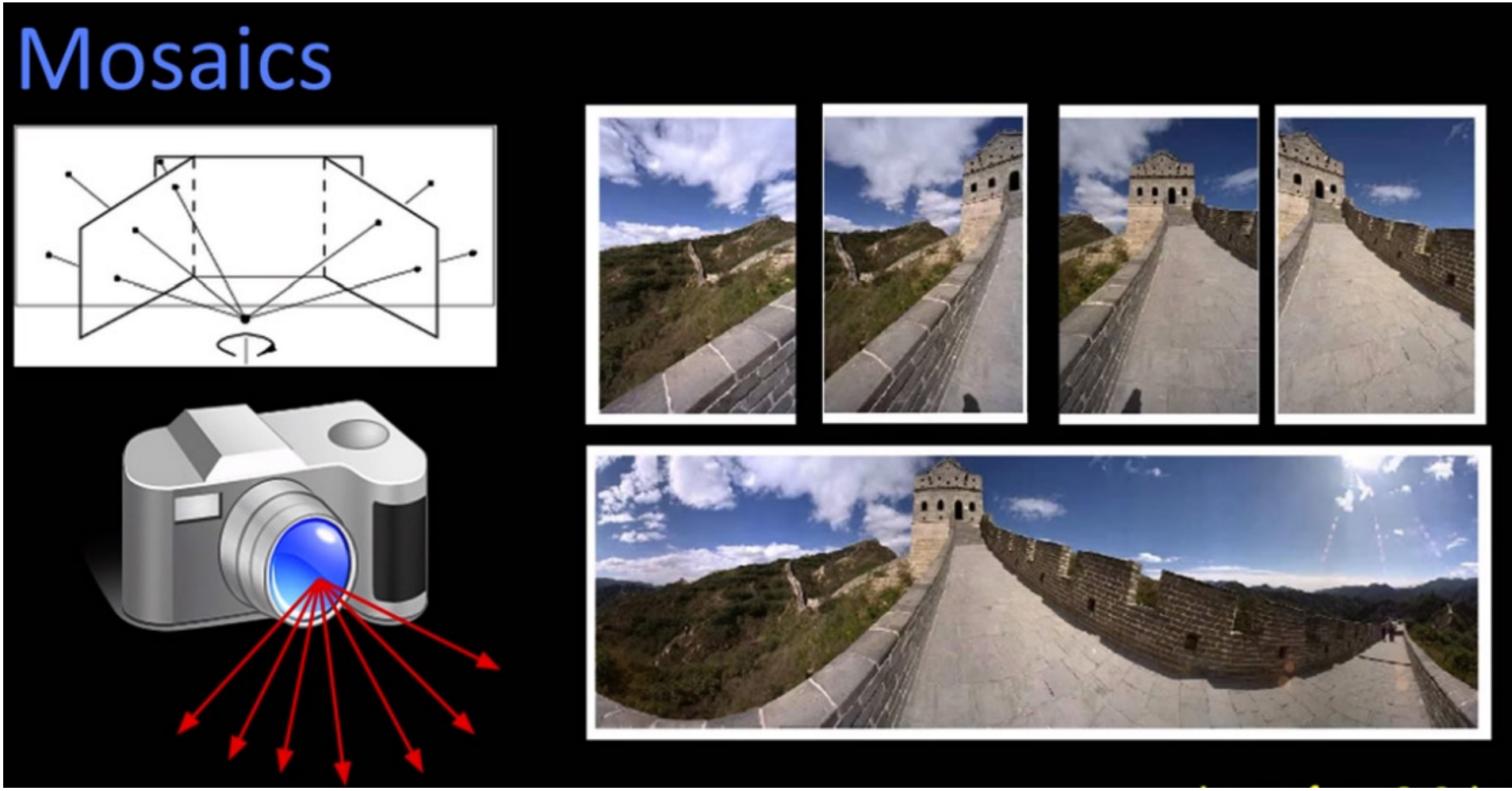
Since 8 unknowns, can set scale factor i=1.

Set up a system of linear equations $\mathbf{Ah} = \mathbf{b}$ where vector of unknowns

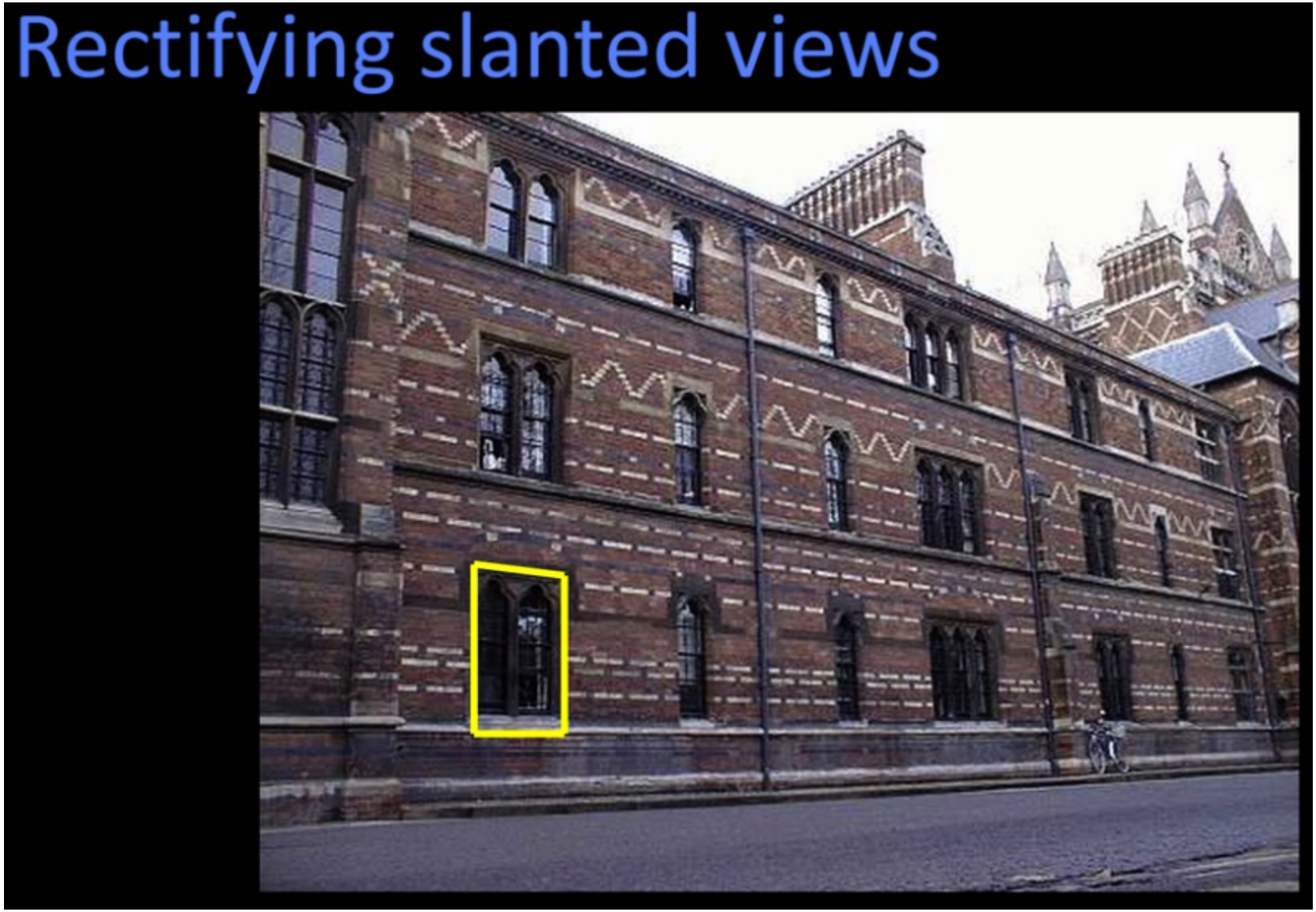
$$h = [a, b, c, d, e, f, g, h]^T$$

Need at least 4 points for 8 eqs, but the more the better…

Solve for h by $\min \|Ah - b\|^2$ using least-squares


Mosaics

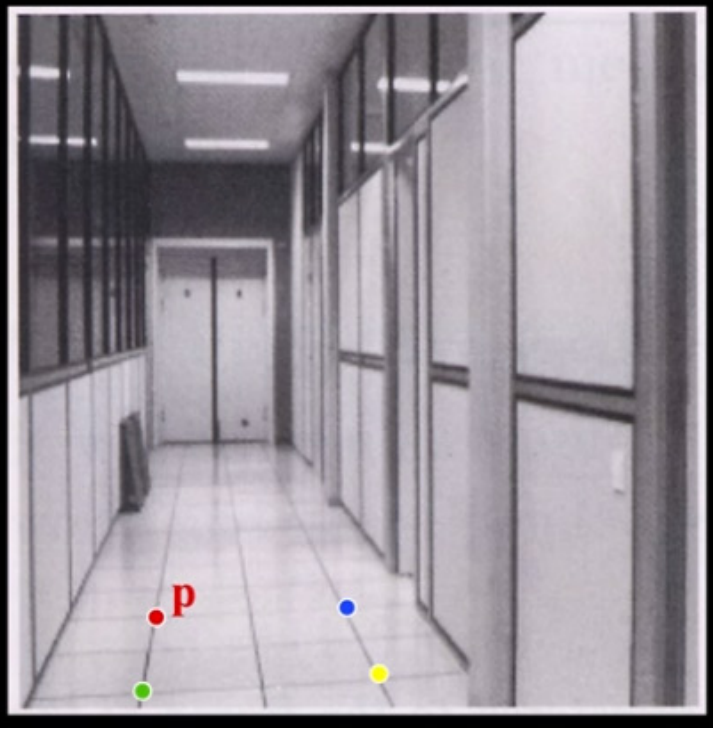

Rectifying slanted views


Rectifying slanted views


Rectifying slanted views
Corrected image (front-to-parallel)

Image rectification

If there is a planar rectangular grid in the scene you can map it into a rectangular grid in the image…



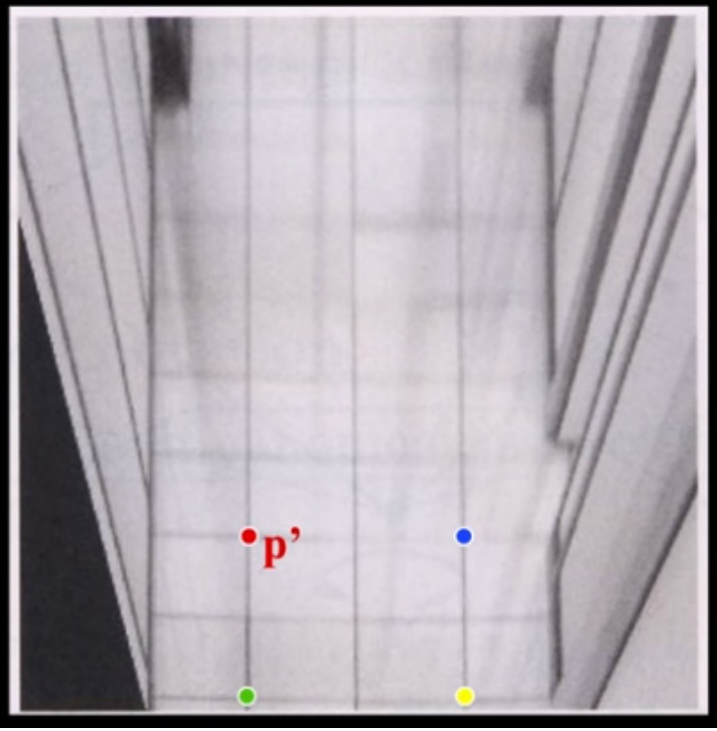
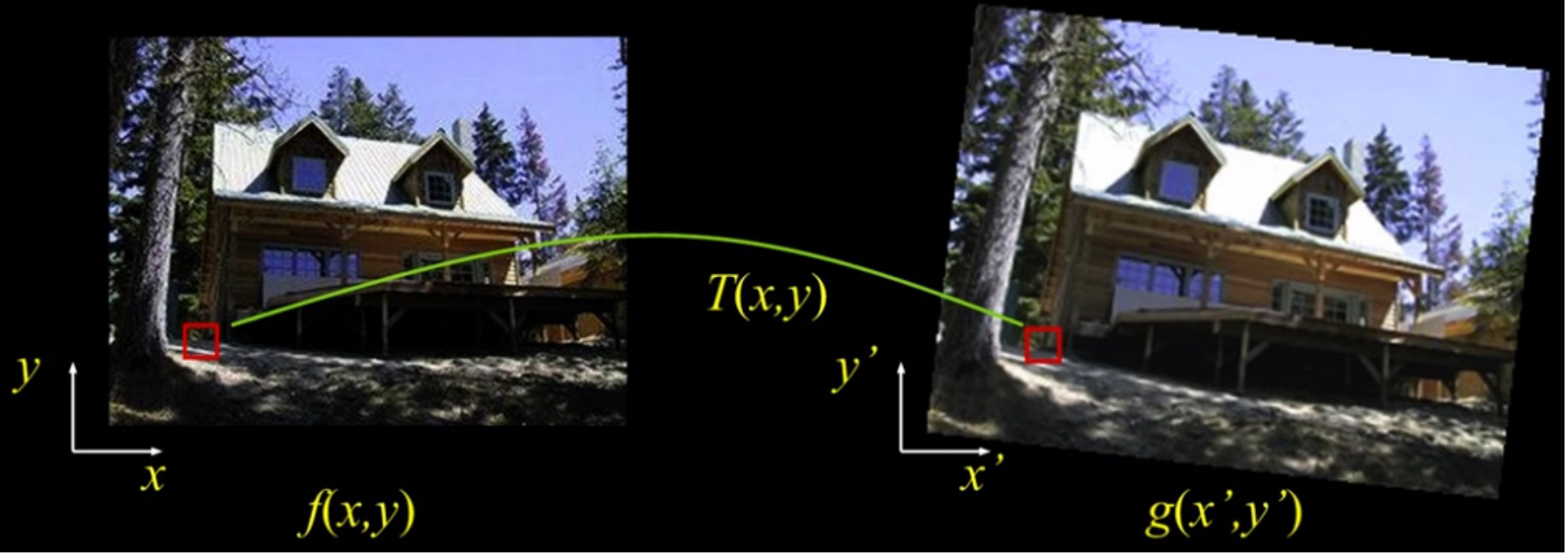


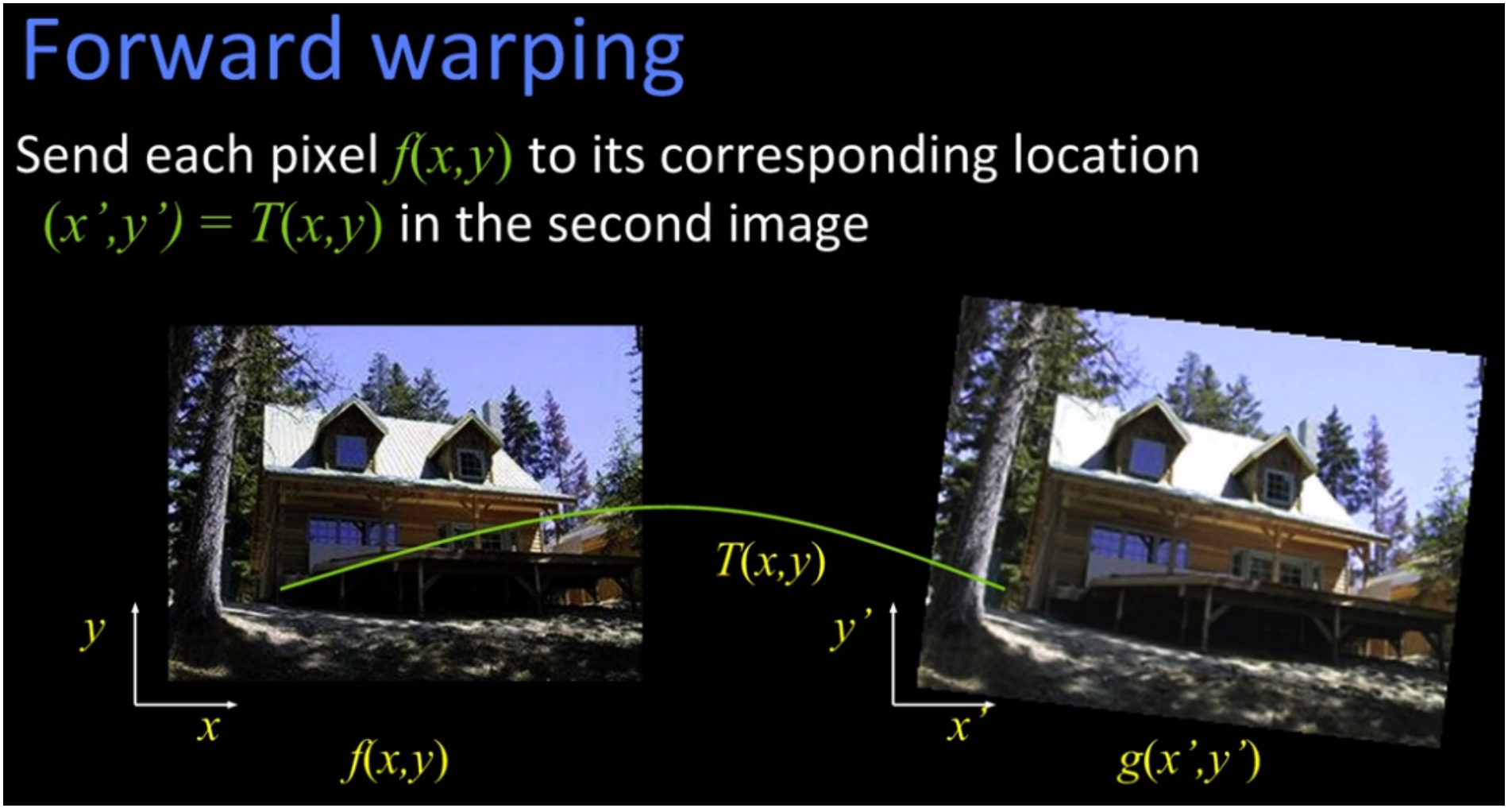
Image warping

Given a coordinate transform and a source image $f(x,y)$, how do we compute a transformed image $g(x',y')$?



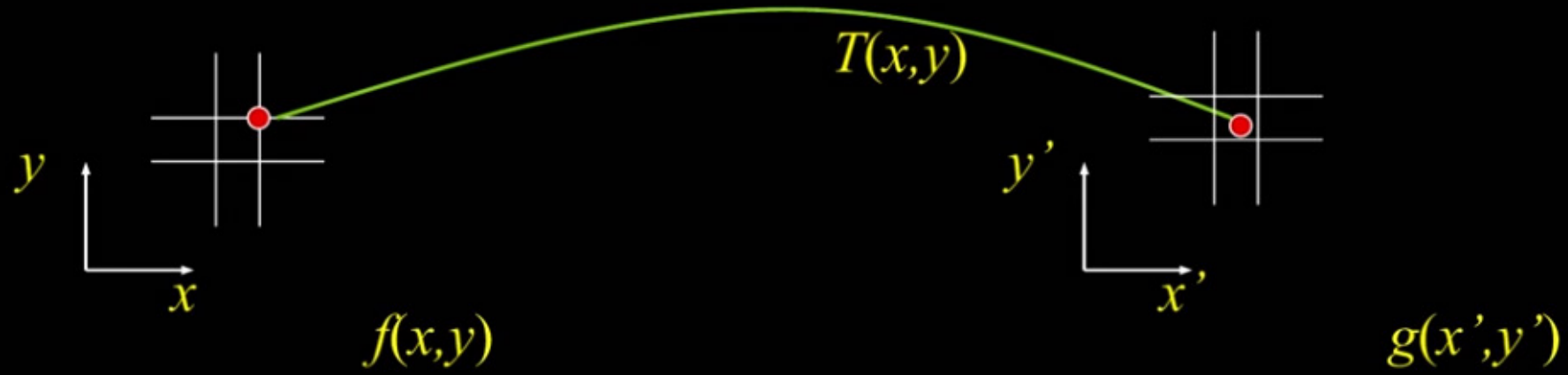
Forward warping

Send each pixel $f(x,y)$ to its corresponding location
$(x',y') = T(x,y)$ in the second image



Forward warping

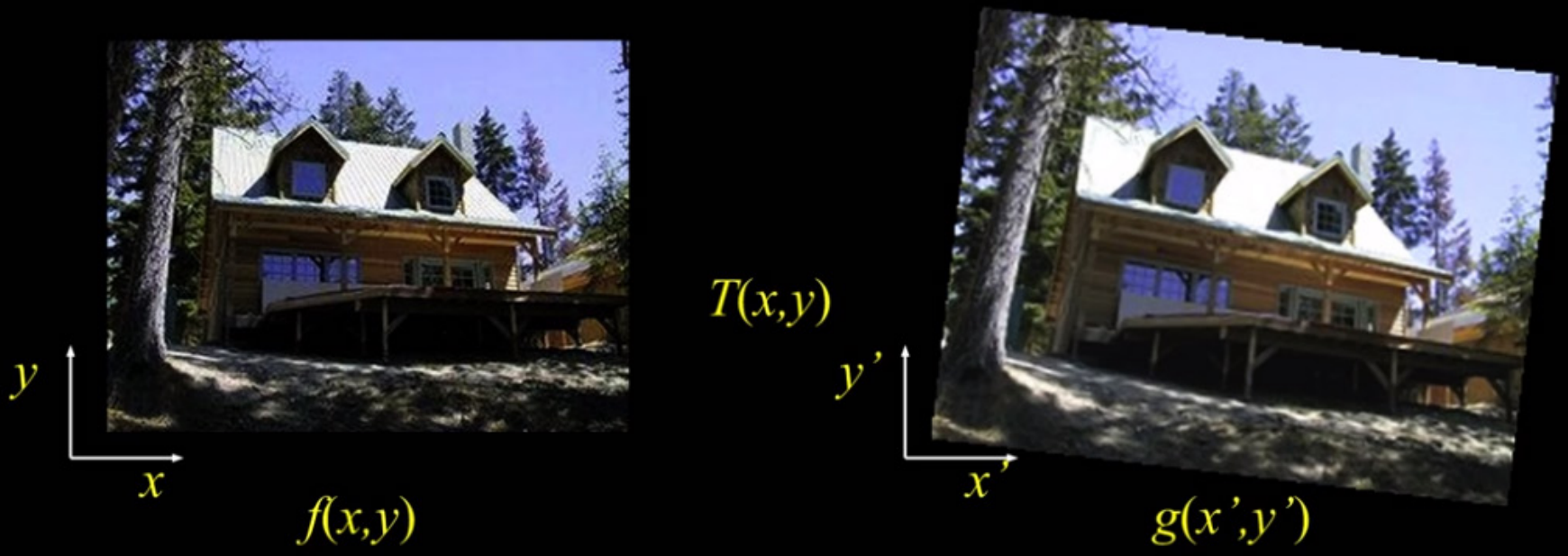
Send each pixel $f(x,y)$ to its corresponding location
$(x',y') = T(x,y)$ in the second image



Inverse warping

Get each pixel $g(x',y')$ from its corresponding location

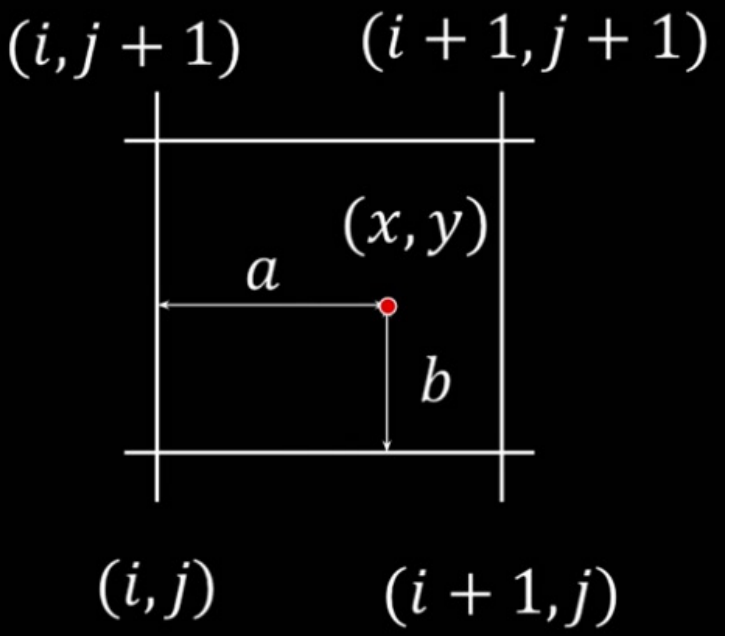
$(x,y) = T^{-1}(x',y')$ in the first image



Given some pixel in x prime, y prime, in the new place, transform back.

Bilinear interpolation

$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad f[i+1,j] \\
& +ab \quad f[i+1,j+1] \\
& +(1-a)b \quad f[i,j+1]
\end{aligned}
$$



3 - Unconventional Image Acquisition: Panorama

1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. Corner Detection
4. Scale-Invariant Detection
5. Scale-Invariant Description
6. RANSAC

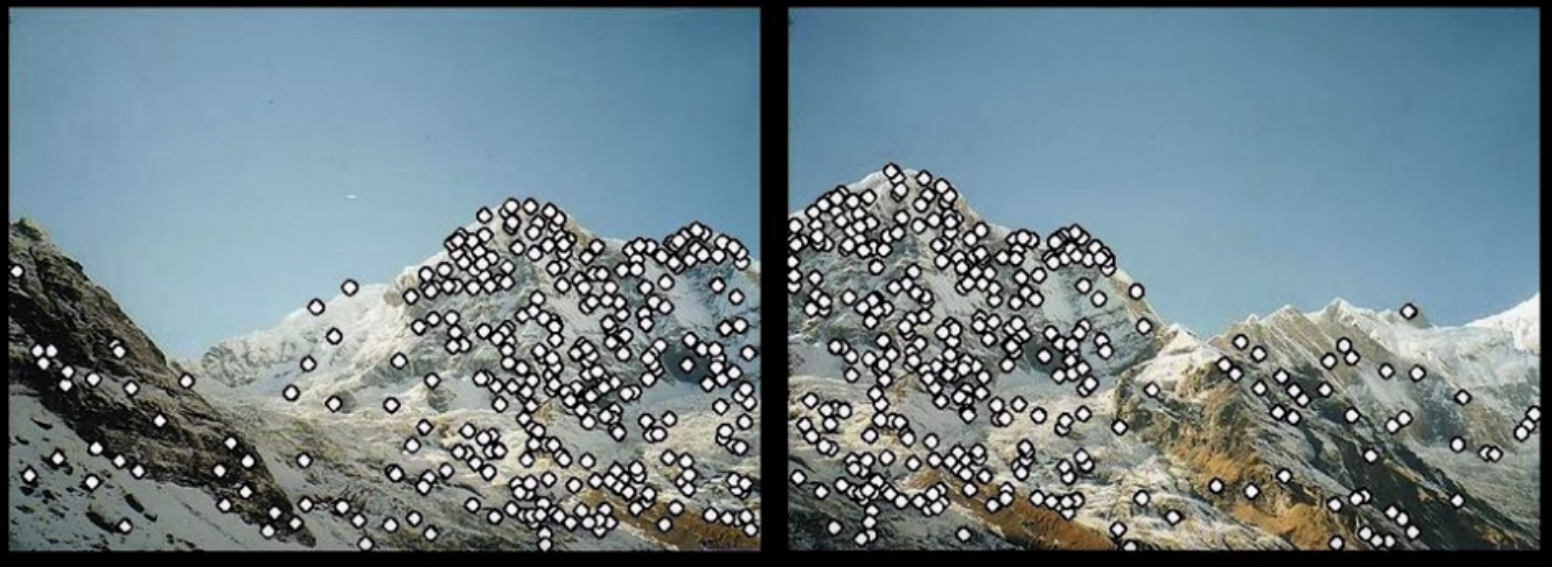
How do we build panorama?

- We need to match (align) images

- Goal: Find points in an image that can be:
  - Found in other images
  - Found precisely – well localized
  - Found reliably – well matched

Matching with Features

- Detect features (feature points) in both images

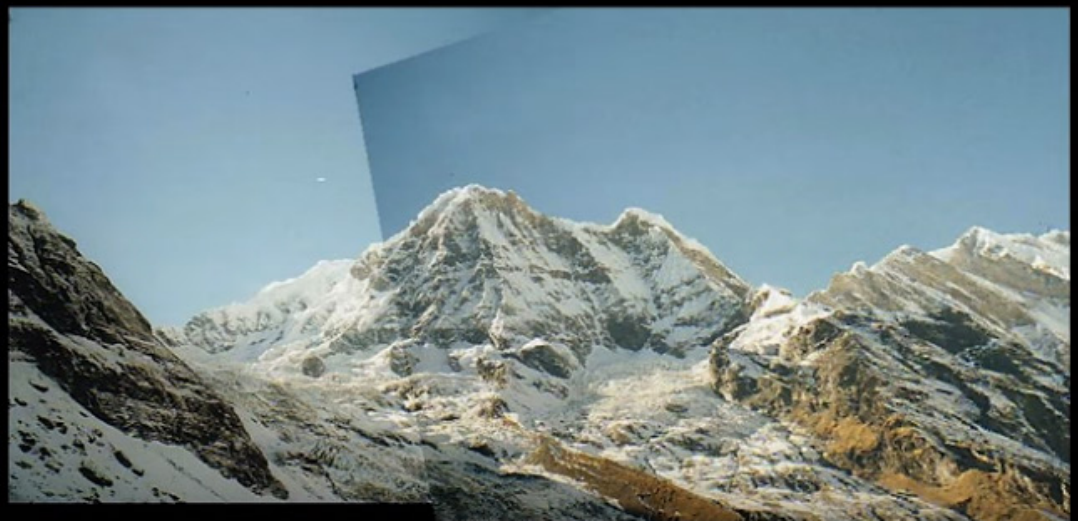


Matching with Features

- Detect features (feature points) in both images
- Match features - find corresponding pairs

Matching with Features

- Detect features (feature points) in both images
- Match features - find corresponding pairs
- Use these pairs to align images



Matching with Features

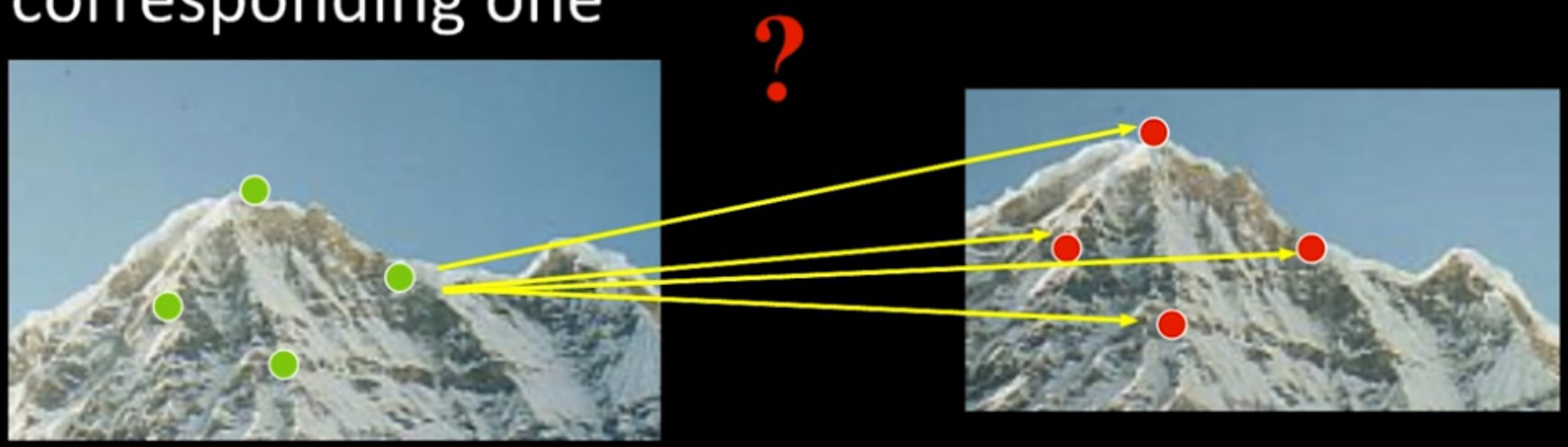
- Problem 1:
  - Detect the same point independently in both

no chance to match!

We need a repeatable detector

Matching with Features

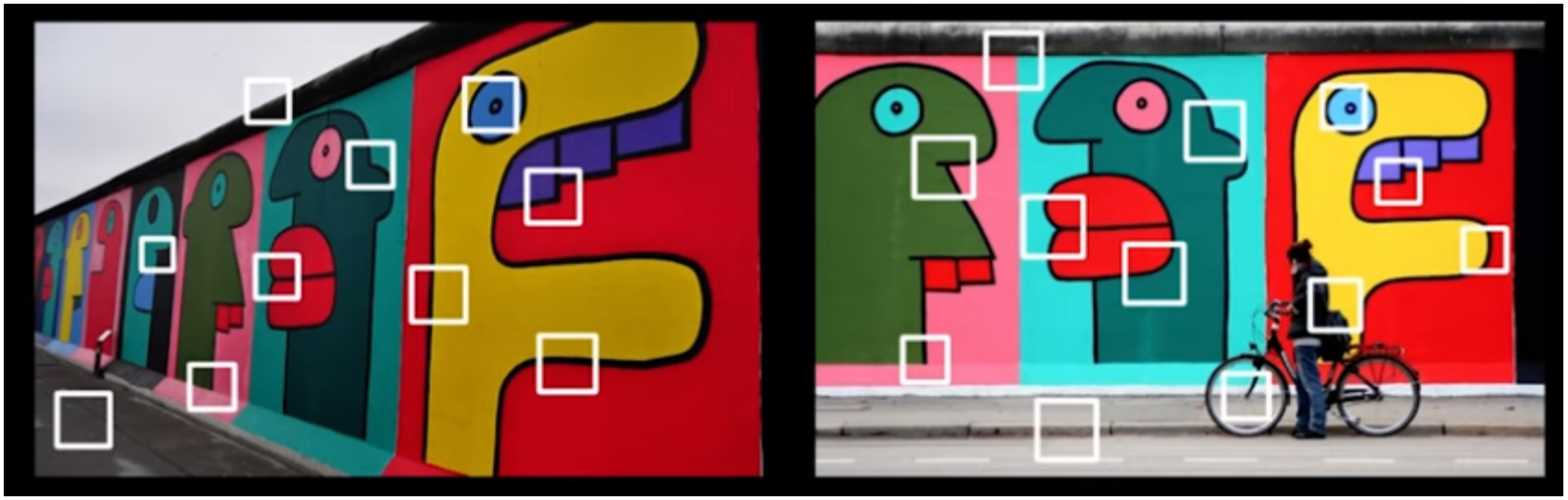
- Problem 2:
  - For each point correctly recognize the corresponding one



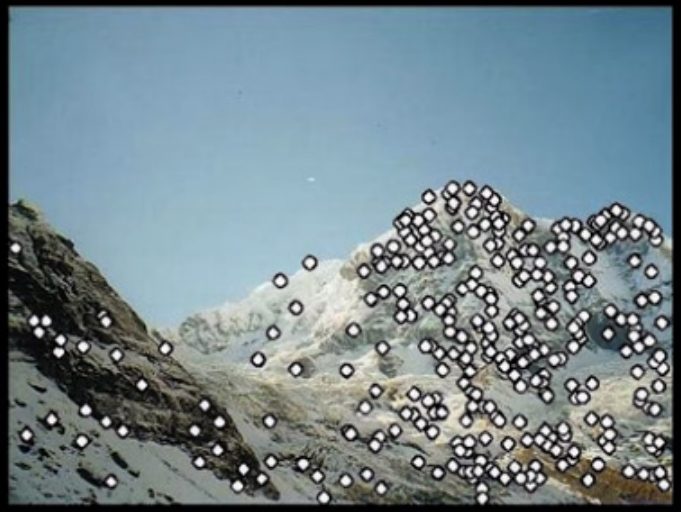
We need a reliable and distinctive ***descriptor***

3.2 - Visual Features For Matching

3.2 - Visual Features For Matching



Characteristics of good features



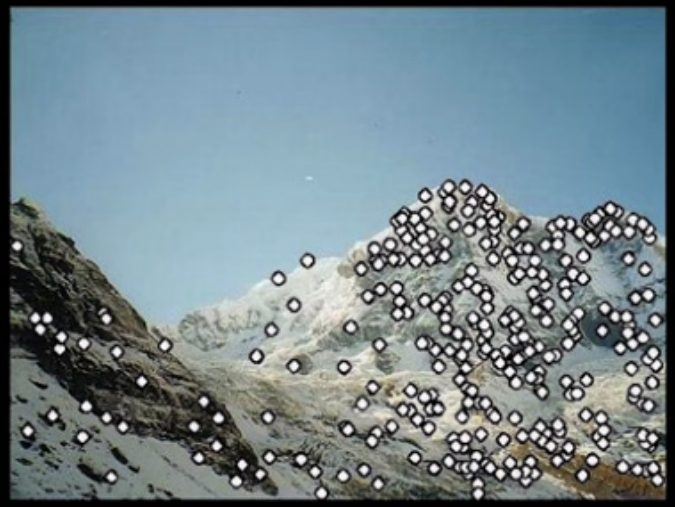
Saliency/Matchability

- Each feature has a distinctive description

Compactness and efficiency

- Many fewer features than image pixels

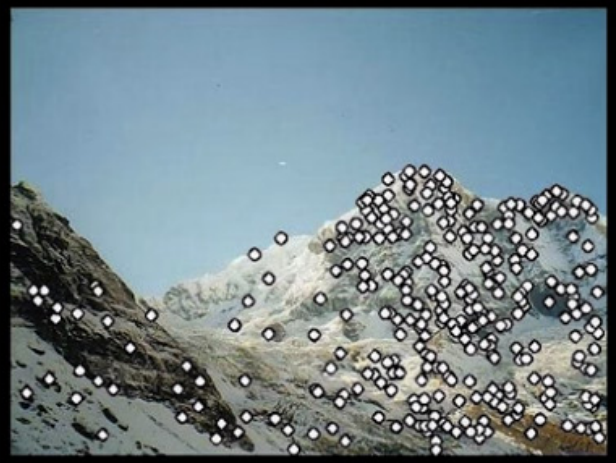
Characteristics of good features



Saliency/Matchability

- Each feature has a distinctive description

Characteristics of good features

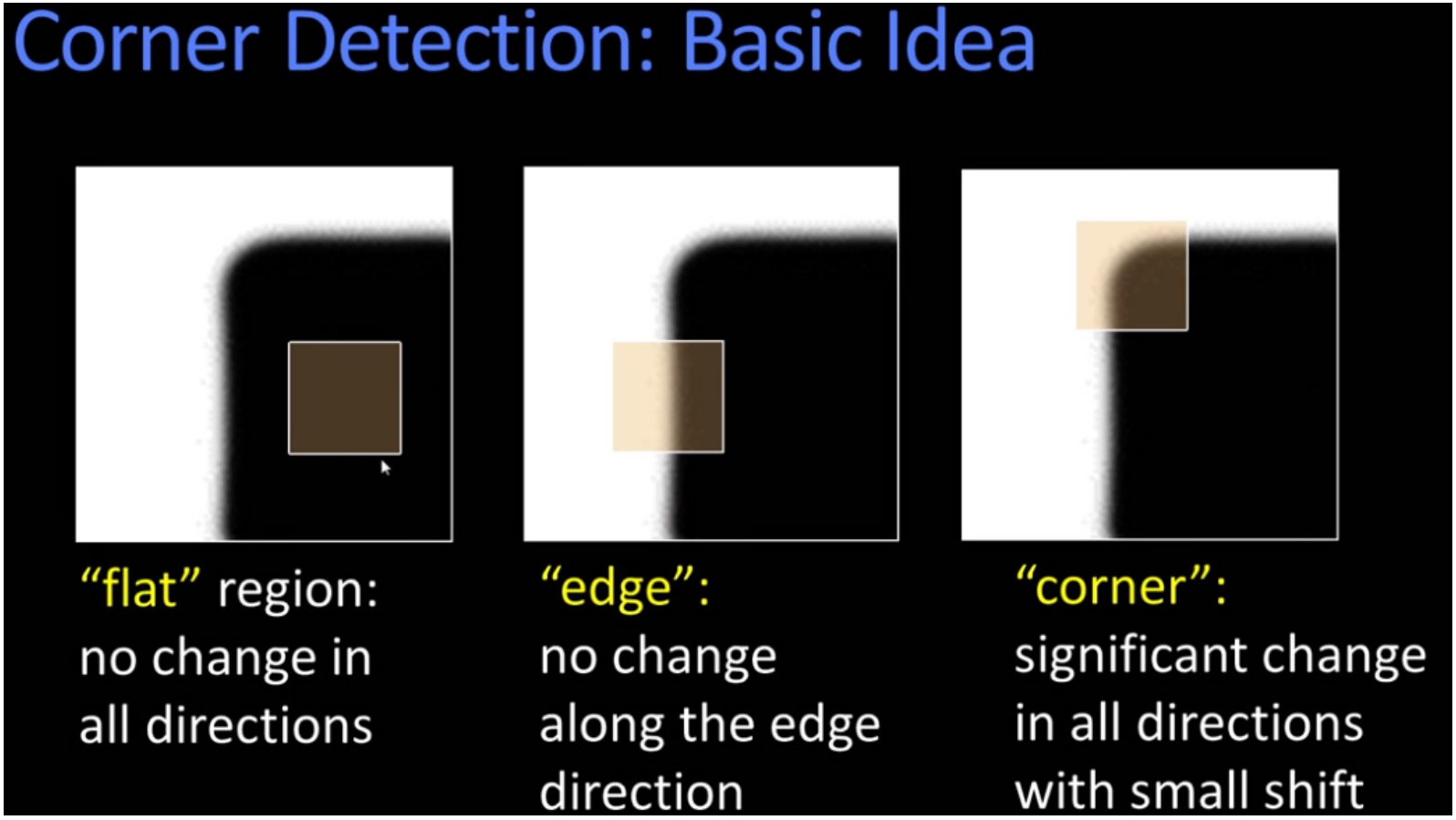


Locality

- A feature occupies a relatively small area of the image; robust to clutter and occlusion

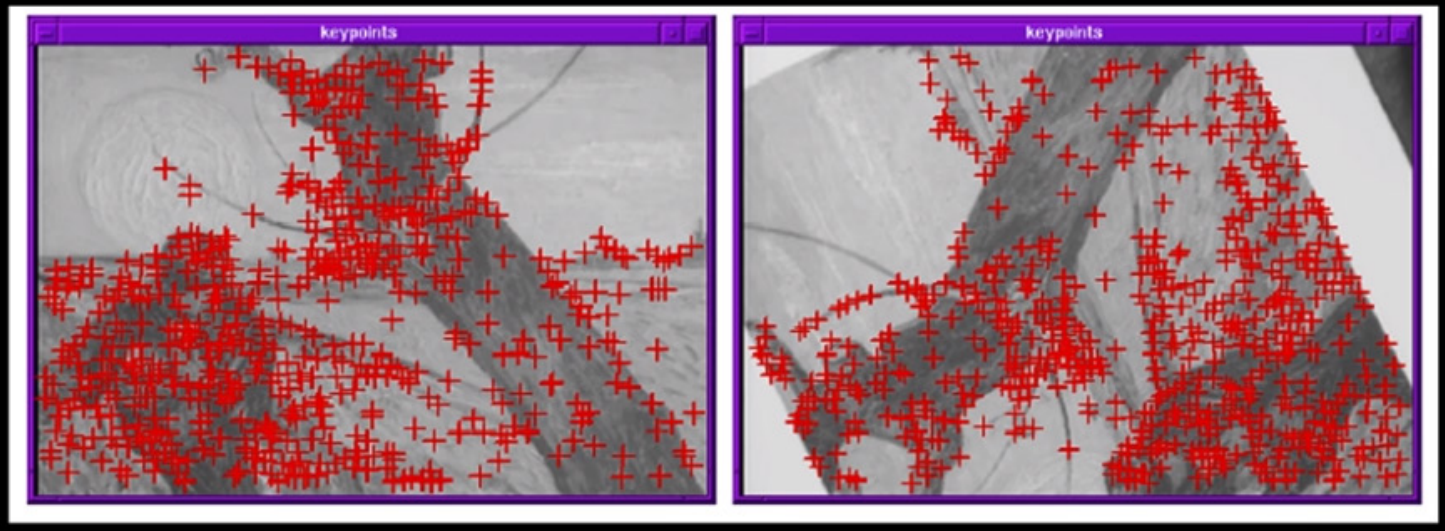
3 - Unconventional Image Acquisition: Panorama

1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. **Corner Detection**
4. **Scale-Invariant Detection**
5. **Scale-Invariant Description**
6. **RANSAC**


Corner Detection: Basic Idea
"flat" region:
no change in
all directions
"edge":
no change
along the edge
direction
"corner":
significant change
in all directions
with small shift

Finding Corners

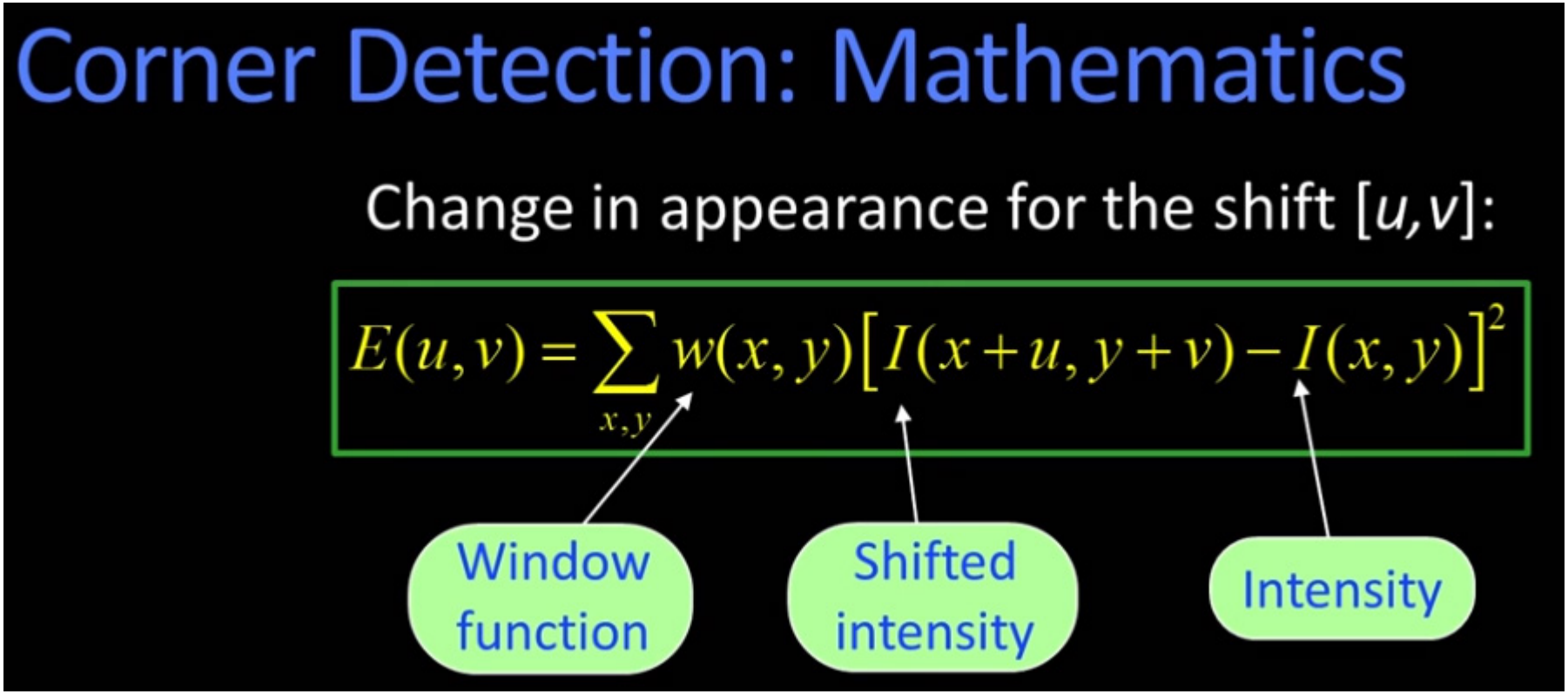
C. Harris and M. Stephens. *"A Combined Corner and Edge Detector,"* *Proceedings of the 4th Alvey Vision Conference*: **1988**



Corner Detection: Mathematics

Change in appearance for the shift [*u*,*v*]:

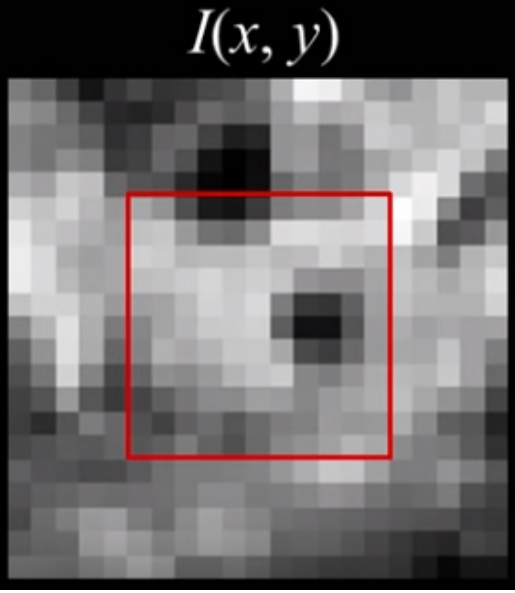
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$


Corner Detection: Mathematics
Change in appearance for the shift [u,v]:
$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$
Window function
Shifted intensity
Intensity

Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

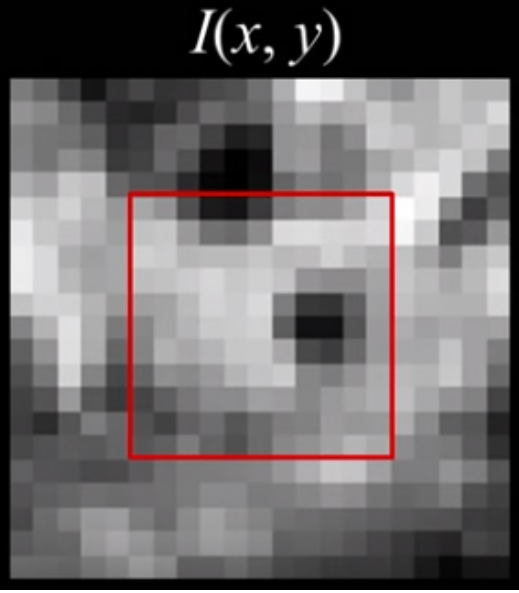


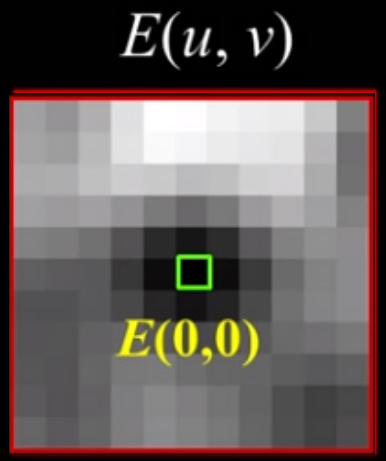
Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$\mathbf{E(0,0) = 0}$





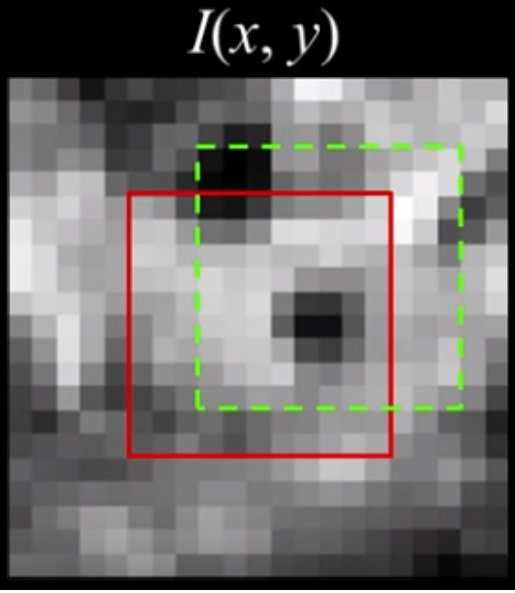
Corner Detection: Mathematics

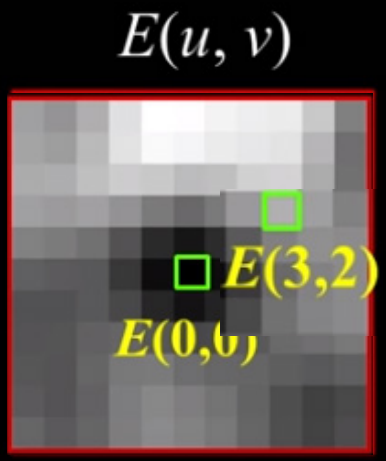
Change in appearance for the shift [*u*,*v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$\mathbf{E(0,0) = 0}$

$\mathbf{E(3,2) > 0}$





Corner Detection: Mathematics

Change in appearance for the shift [*u*,*v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

We want to find out how this function behaves for ***small*** shifts (u,v near 0,0)

Corner Detection: Mathematics

Change in appearance for the shift [*u*,*v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0) (local quadratic approximation for small u,v):

Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

$$F(\delta x) \approx F(0) + \delta x \cdot \frac{dF(0)}{dx} + \frac{1}{2}\delta x^2 \cdot \frac{d^2F(0)}{dx^2}$$

Corner Detection: Mathematics

Change in appearance for the shift [*u*,*v*]:

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

$$F(\delta x)\approx F(0)+\delta x\cdot\frac{dF(0)}{dx}+\frac{1}{2}\delta x^2\cdot\frac{d^2F(0)}{dx^2}$$

$$E(u,v)\approx E(0,0)+[u\ \ v]\begin{bmatrix}E_u(0,0)\\E_v(0,0)\end{bmatrix}+\frac{1}{2}[u\ \ v]\begin{bmatrix}E_{uu}(0,0) & E_{uv}(0,0)\\E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}$$

3.3 – Corner Detection

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

3.3 – Corner Detection

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Let's digest this by looking at the derivatives one-by-one:

$$E_u(u,v) = \sum_{x,y} 2w(x,y)\left[I(x+u, y+v) - I(x,y)\right] I_x(x+u, y+v)$$

3.3 – Corner Detection

Now, the second derivative (the derivative of the derivative):

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y) I_x(x+u,y+v) I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xx}(x+u,y+v)$$

The derivatives wrt v are computed in exactly the same way, and now we need the cross-derivatives.

3.3 – Corner Detection

What is the intuition behind homogeneous coordinates?

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y) I_y(x+u,y+v) I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xy}(x+u,y+v)$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(u,v) = \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_x(x+u,y+v)$$

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y) I_x(x+u,y+v) I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y) I_y(x+u,y+v) I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xy}(x+u,y+v)$$

Evaluate E and its derivatives at (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(0,0) = \sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right] I_x(x,y)$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right] I_{xx}(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_x(x,y)$$

$$+ \sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right] I_{xy}(x,y)$$

Evaluate E and its derivatives at (0,0):

$$E(u,v) \approx \underbrace{E(0,0)}_{=0} + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(0,0) = \sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right]I_x(x,y)$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y)I_x(x,y)I_x(x,y)$$

$$+\sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right]I_{xx}(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y)I_y(x,y)I_x(x,y)$$

$$+\sum_{x,y} 2w(x,y)\left[I(x,y) - I(x,y)\right]I_{xy}(x,y)$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0 \qquad E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_u(0,0) = 0 \qquad E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_v(0,0) = 0 \qquad E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx [u \;\; v] \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0 \qquad E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_u(0,0) = 0 \qquad E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_v(0,0) = 0 \qquad E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a *second moment matrix* computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

The second moment matrix M:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Each product is a rank 1 2x2

Can be written (without the weight):

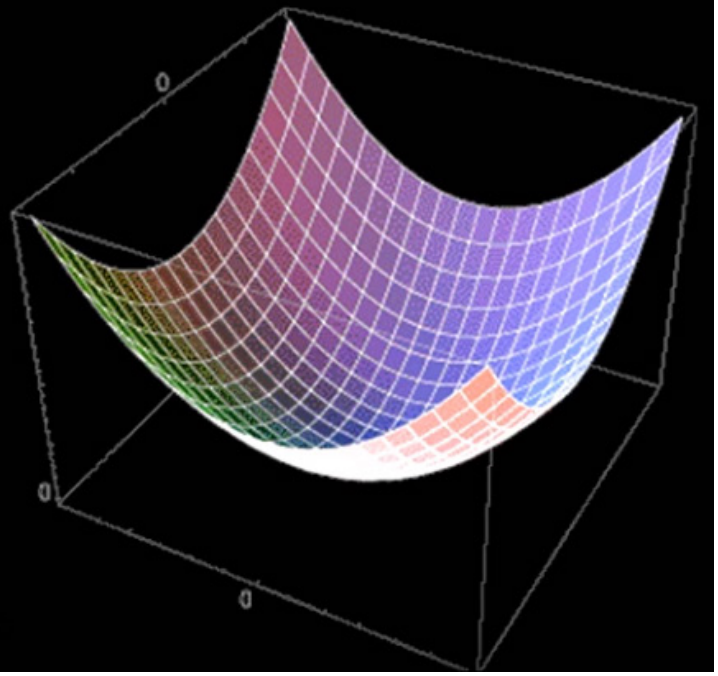
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \left(\begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \right) = \sum \nabla I (\nabla I)^T$$

Interpreting the second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

Consider a constant "slice" of $E(u, v)$:

$$\Sigma I_x^2 u^2 + 2\Sigma I_x I_y uv + \Sigma I_y^2 v^2 = k$$

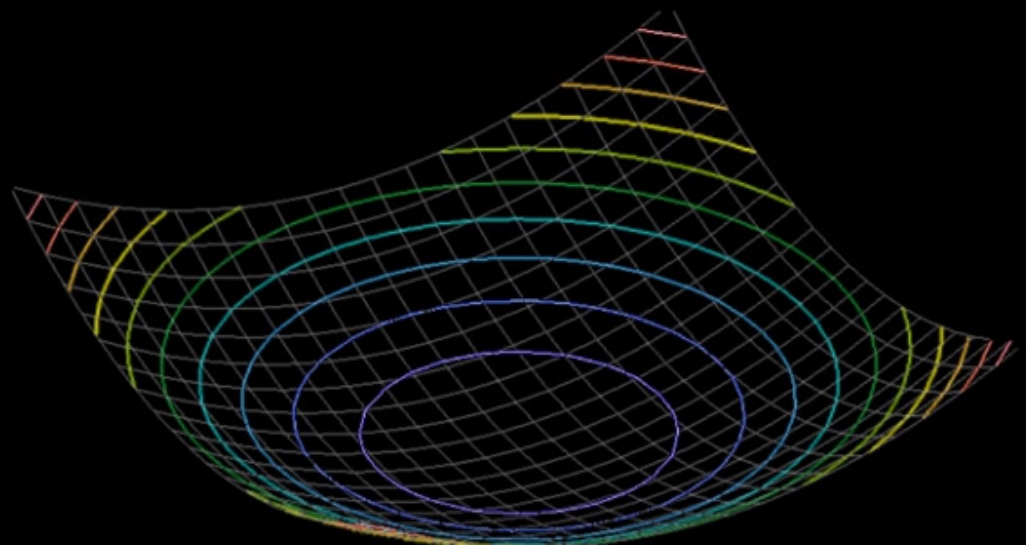
$$[u \quad v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

Interpreting the second moment matrix

Consider a constant "slice" of $E(u, v)$:

$$\Sigma I_x^2 u^2 + 2\Sigma I_x I_y uv + \Sigma I_y^2 v^2 = k$$

This is the equation of an ellipse.



Interpreting the second moment matrix

First, consider the axis-aligned case where gradients are either horizontal or vertical

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

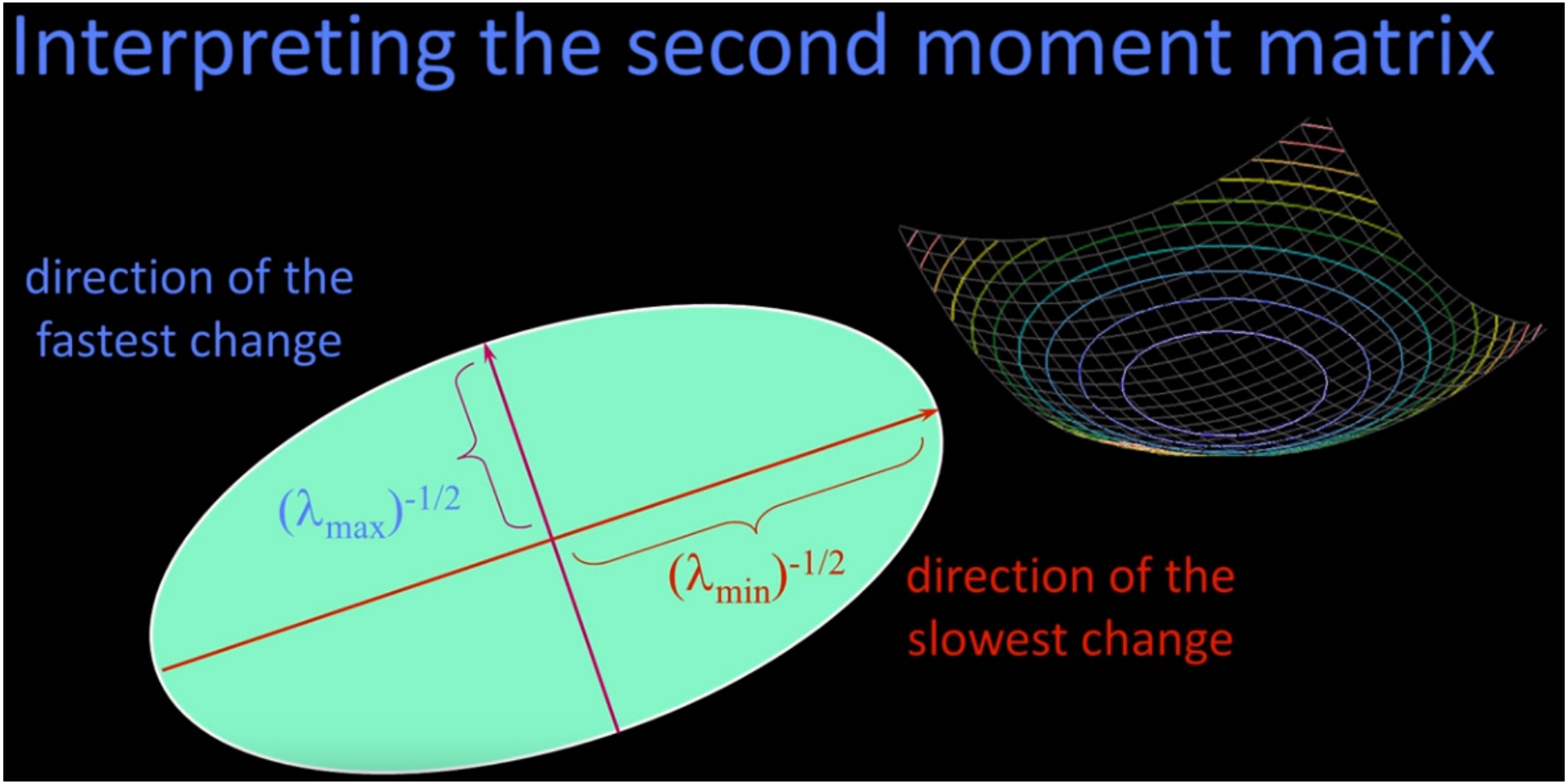
Image Transformations

Interpreting the second moment matrix

Diagonalization of M: $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

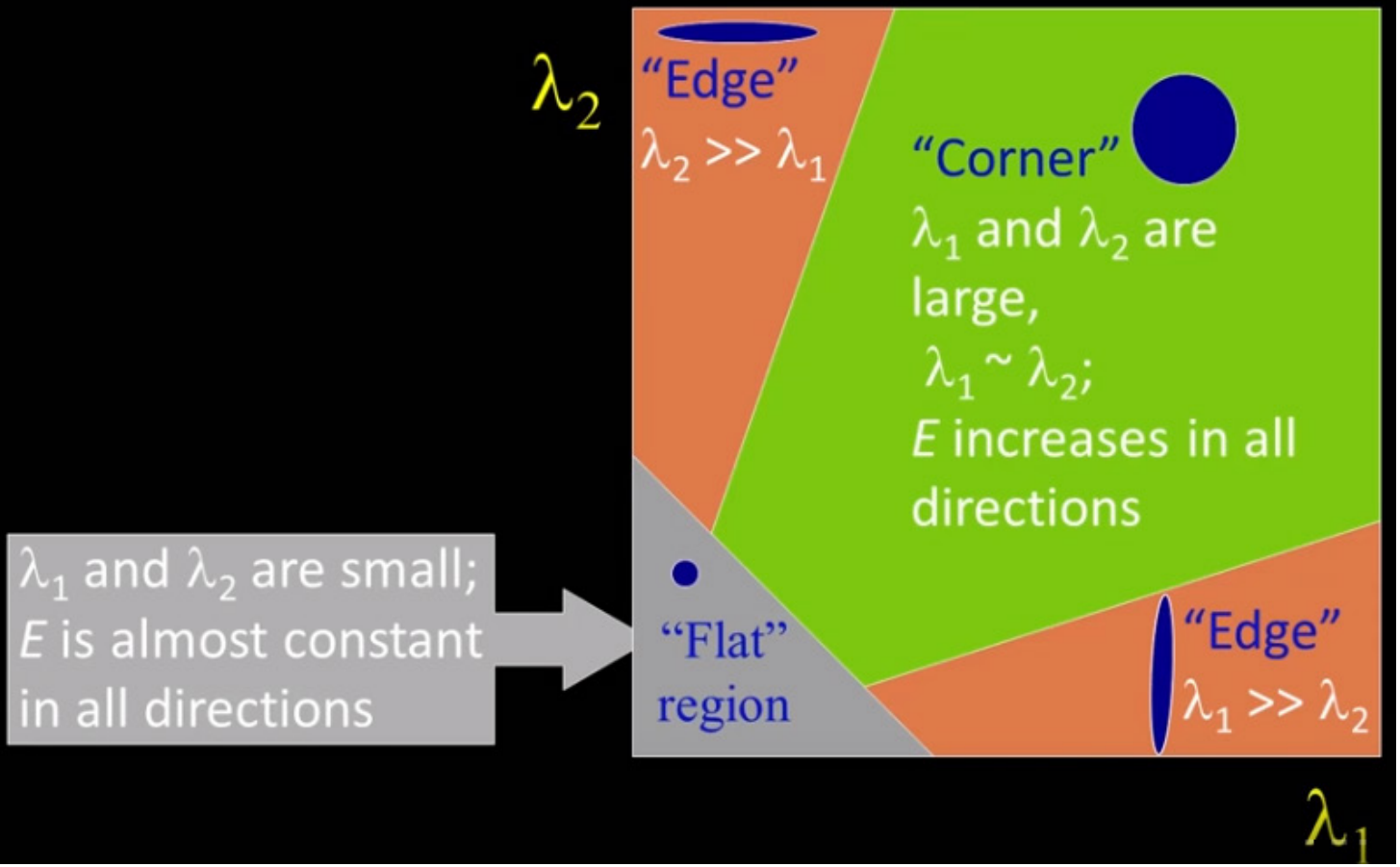
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by *R*

Image Transformations



Interpreting the eigenvalues

Classification of image points using eigenvalues of *M*:



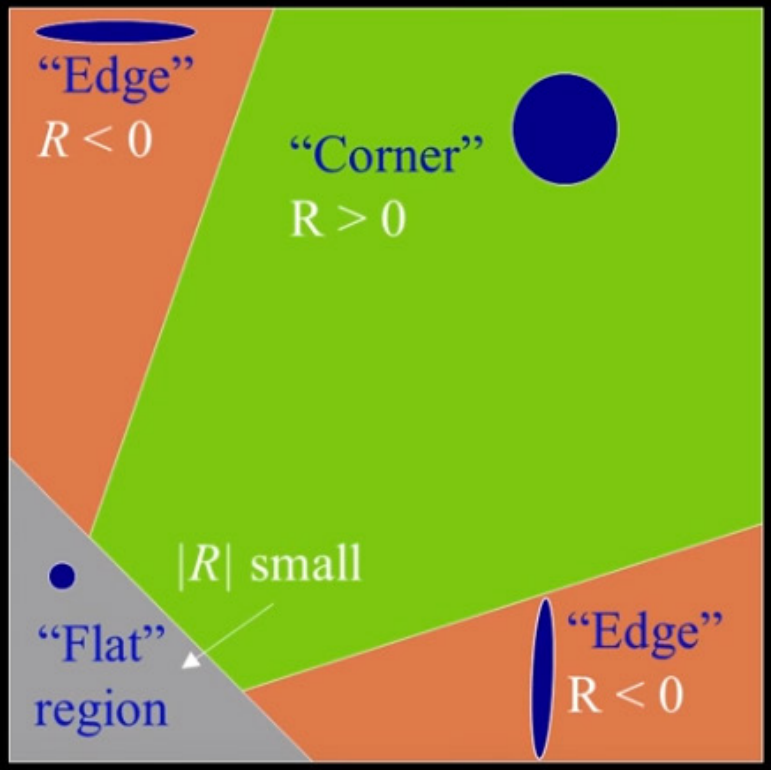
It turns out we do not even need to compute eigenvalues!

Harris corner response function

$$R = \det(M) - \alpha \,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)

R depends only on eigenvalues of M, but don't compute them (no sqrt, so really fast even in the '80s).



It turns out we do not even need to compute eigenvalues!

Harris corner response function

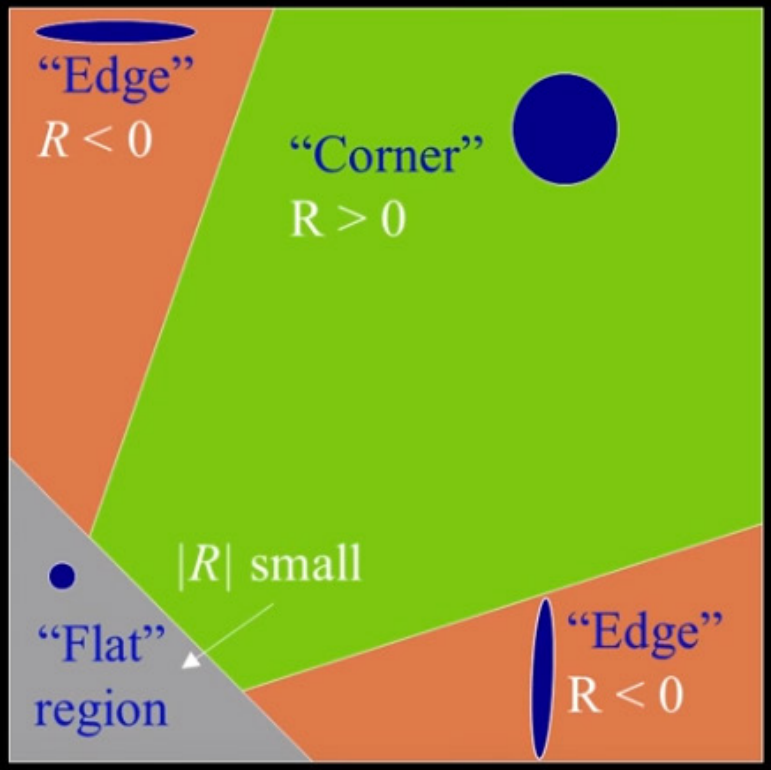
$$R = \det(M) - \alpha \,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)

R is large for a corner

R is negative with large magnitude for an edge

|*R*| is small for a flat region

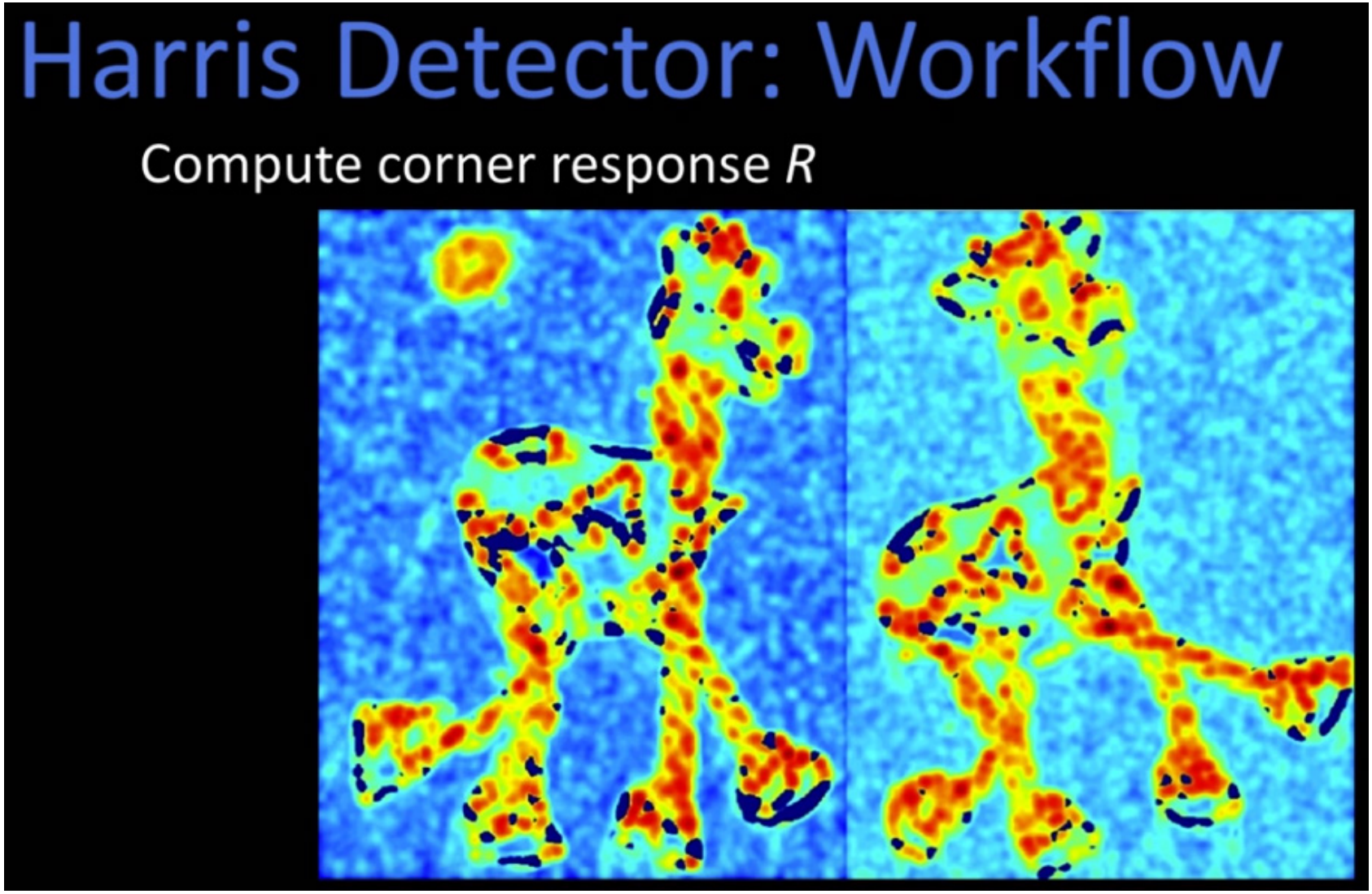


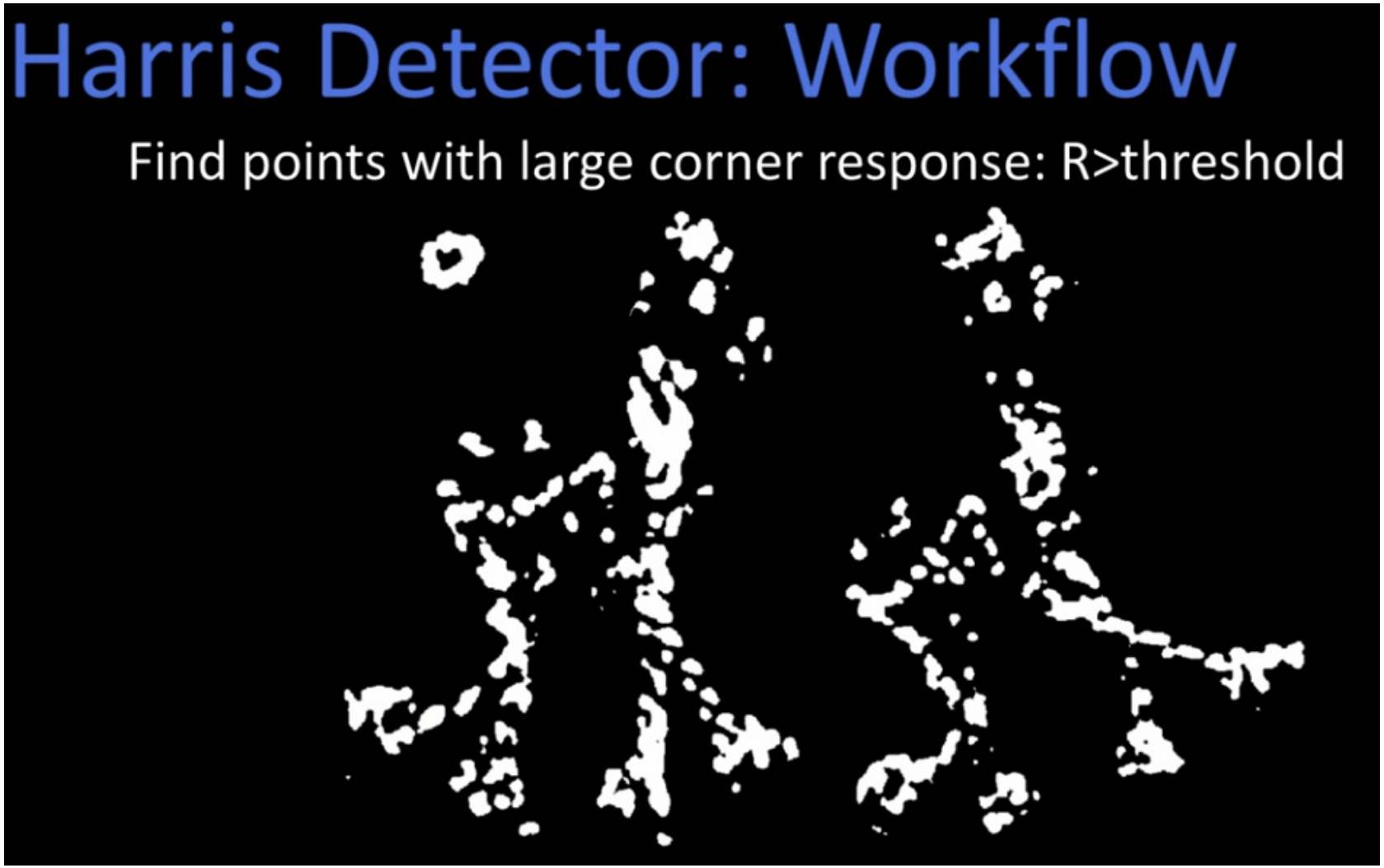
Harris detector: Algorithm

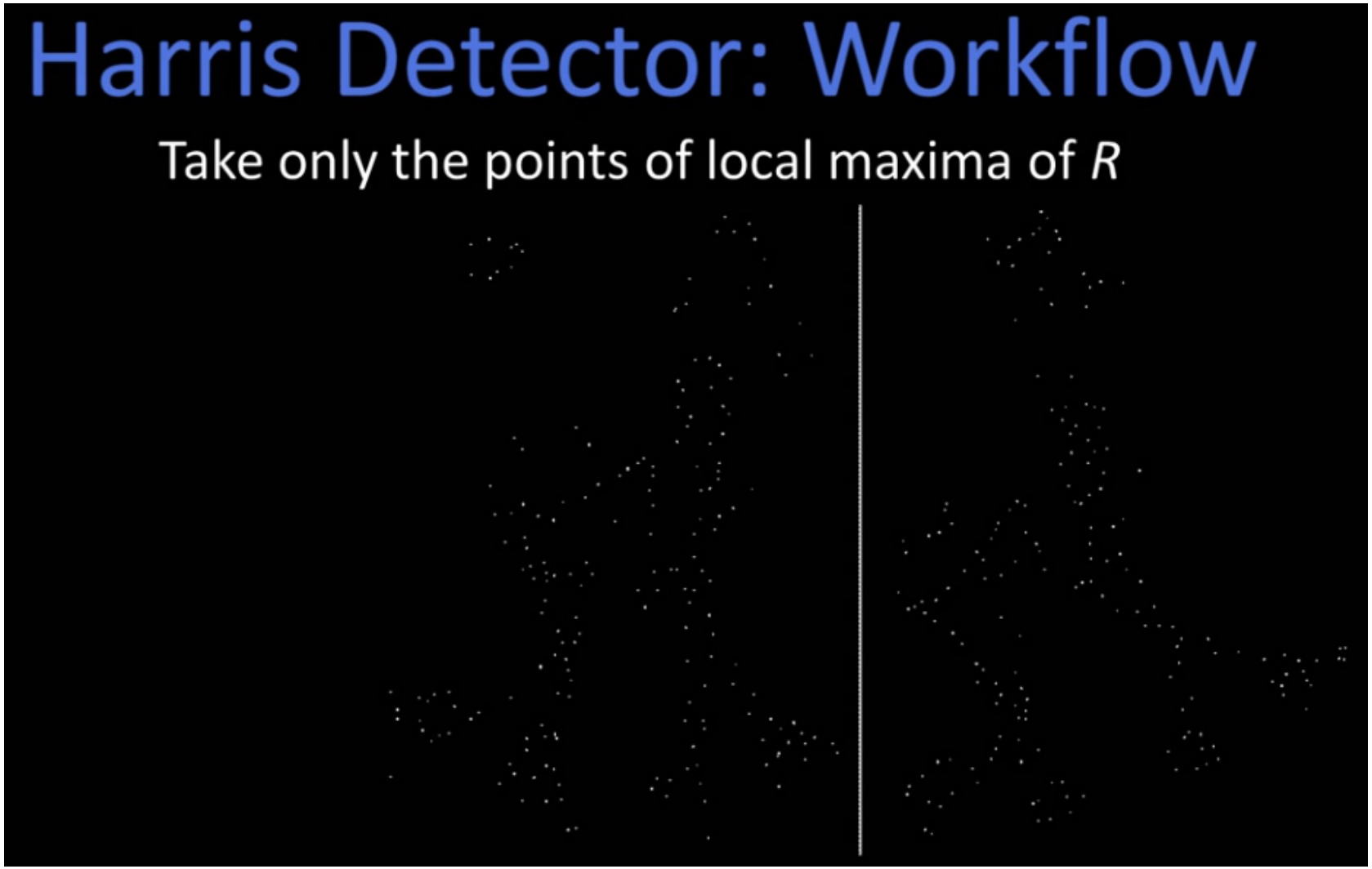
1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix M in a Gaussian window around each pixel
3. Compute corner response function R
4. Threshold R
5. Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.


Harris Detector: Workflow


Harris Detector: Workflow
Compute corner response R

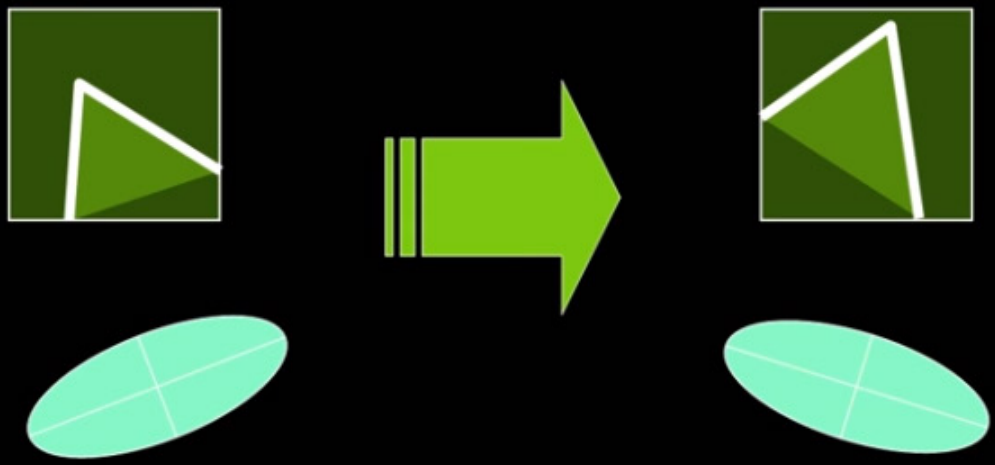

Harris Detector: Workflow
Find points with large corner response: R>threshold


Harris Detector: Workflow
Take only the points of local maxima of R


Harris Detector: Workflow

Harris Detector: Some Properties

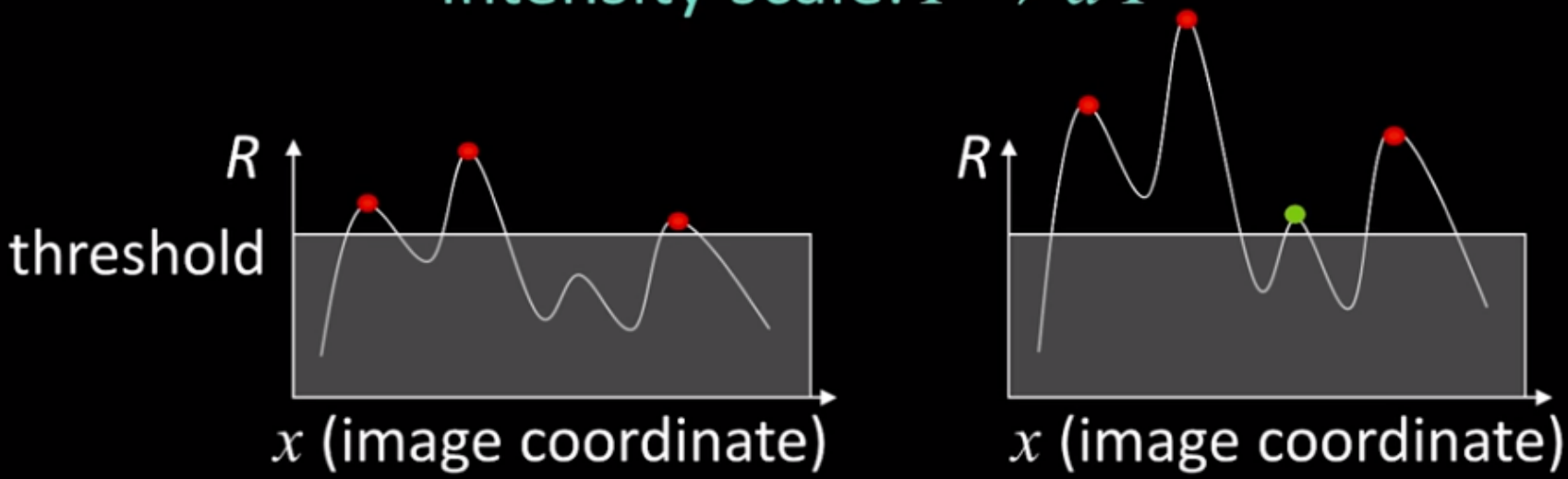
Rotation invariance?



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

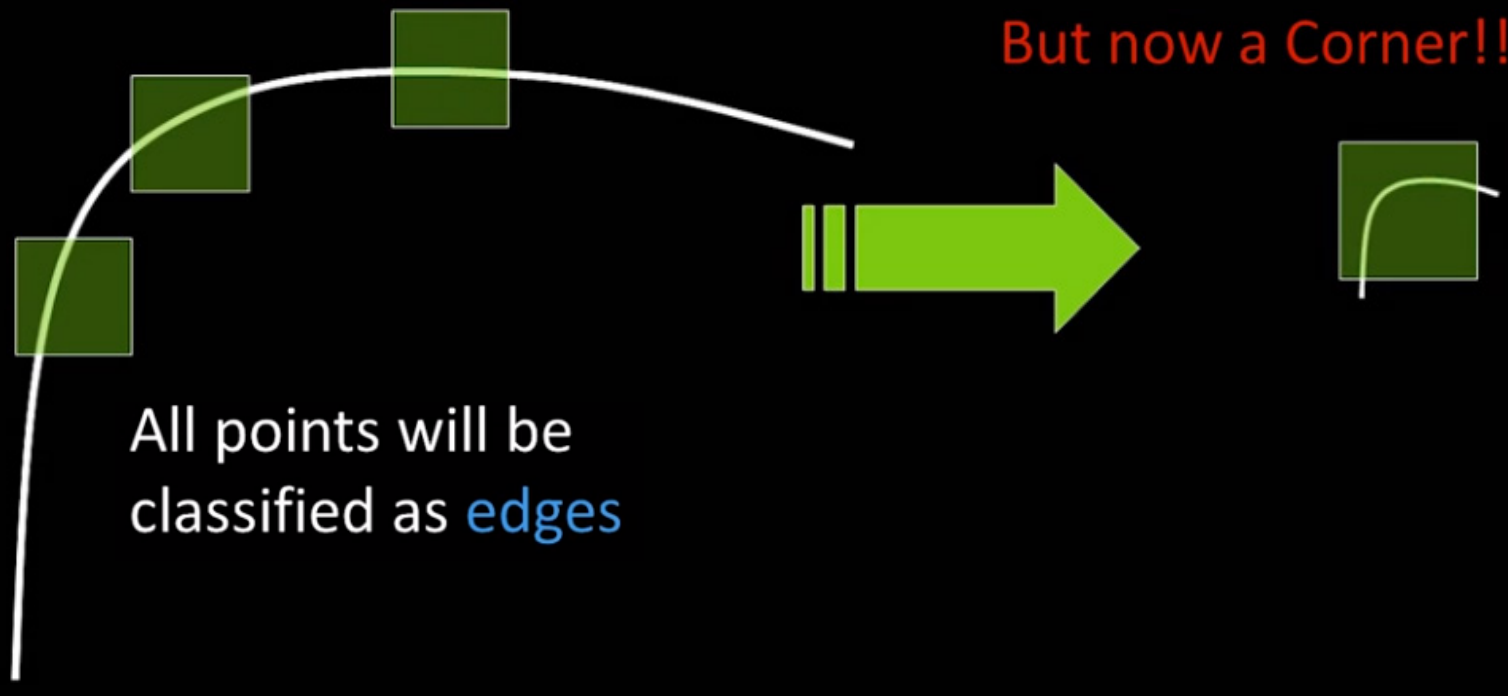
Harris Detector: Some Properties

- Mostly invariant to additive and multiplicative intensity changes (threshold issue for multiplicative)
  - ✓ Intensity scale: $I \rightarrow a\,I$



Harris Detector: Some Properties

- Not invariant to *image scale*!



3 - Unconventional Image Acquisition: Panorama

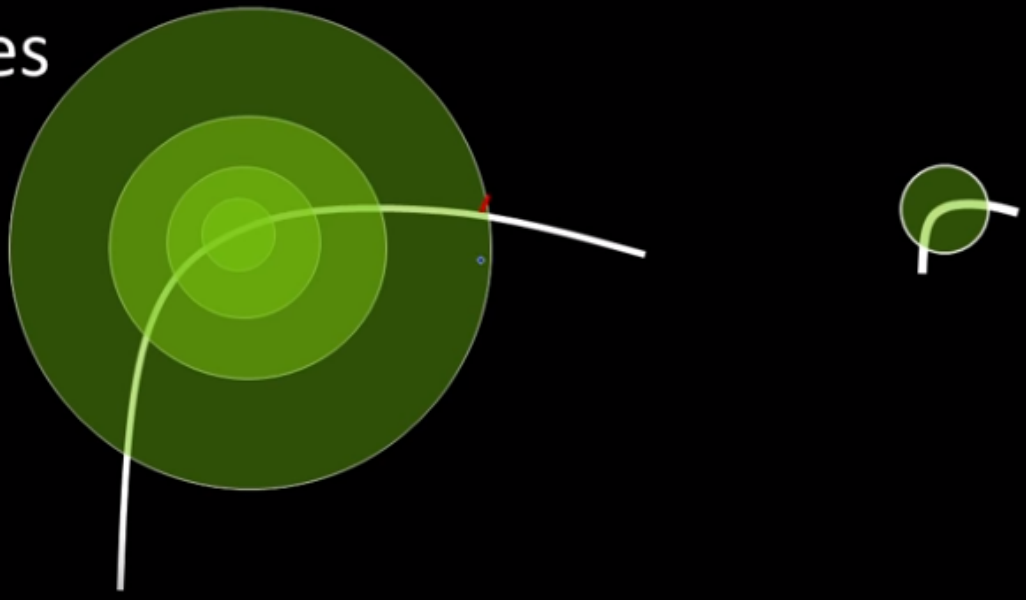
1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. **Corner Detection**
4. **Scale-Invariant Detection**
5. **Scale-Invariant Description**
6. **RANSAC**

Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

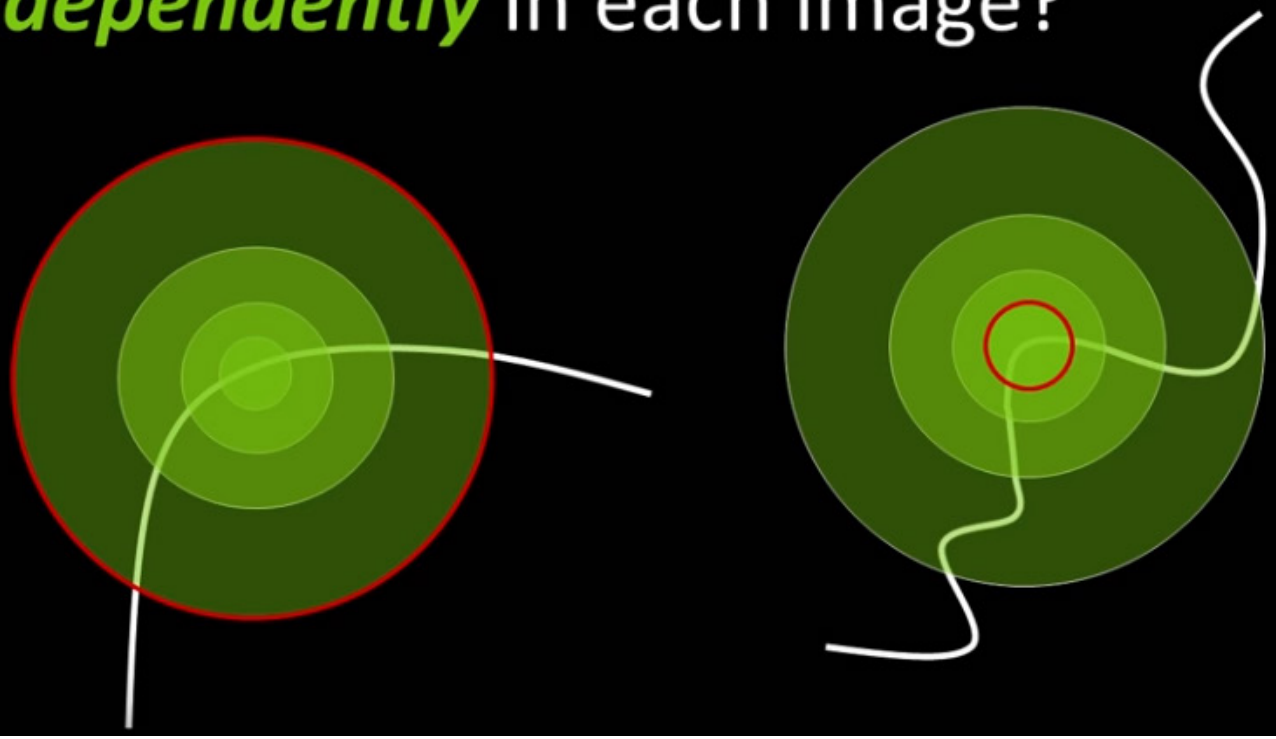
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

- The problem: how do we choose corresponding circles ***independently*** in each image?



Scale Invariant Detection

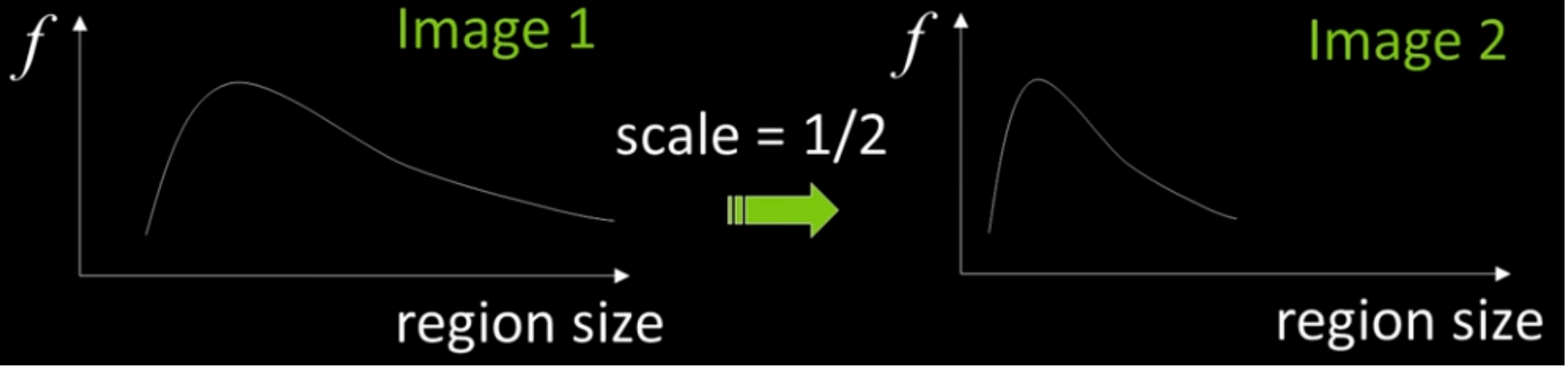
- Solution:
  - Design a function on the region (circle), which is "scale invariant" - not affected by the size but will be the same for "corresponding regions, " even if they are at different sizes/scales.

Example: Average intensity. For corresponding regions (even of different sizes) it will be the same.

Scale Invariant Detection

One method:

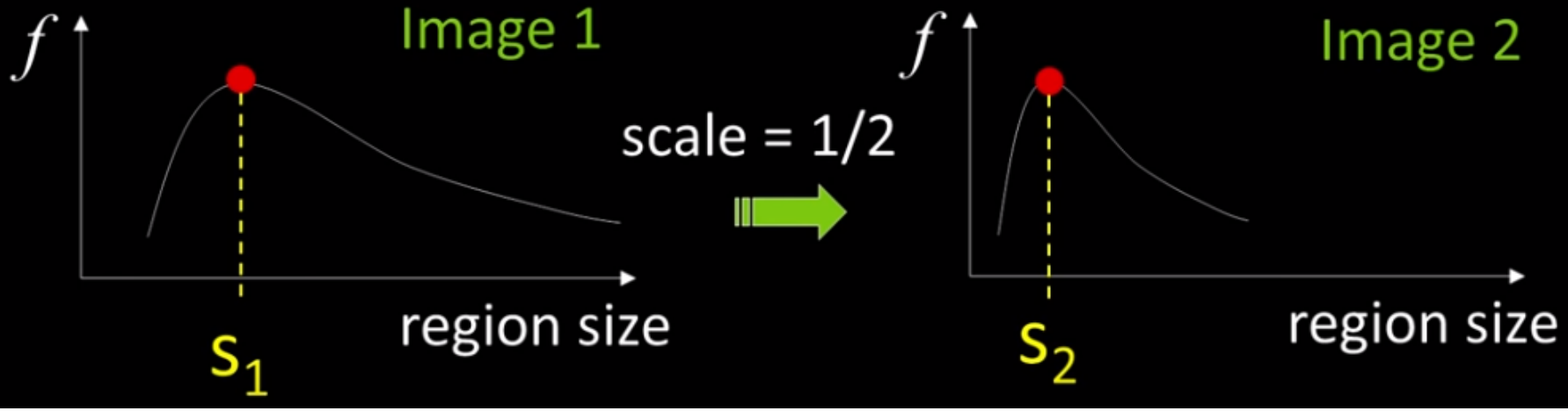
- At a point, compute the scale invariant function over different size neighborhoods (different scales).



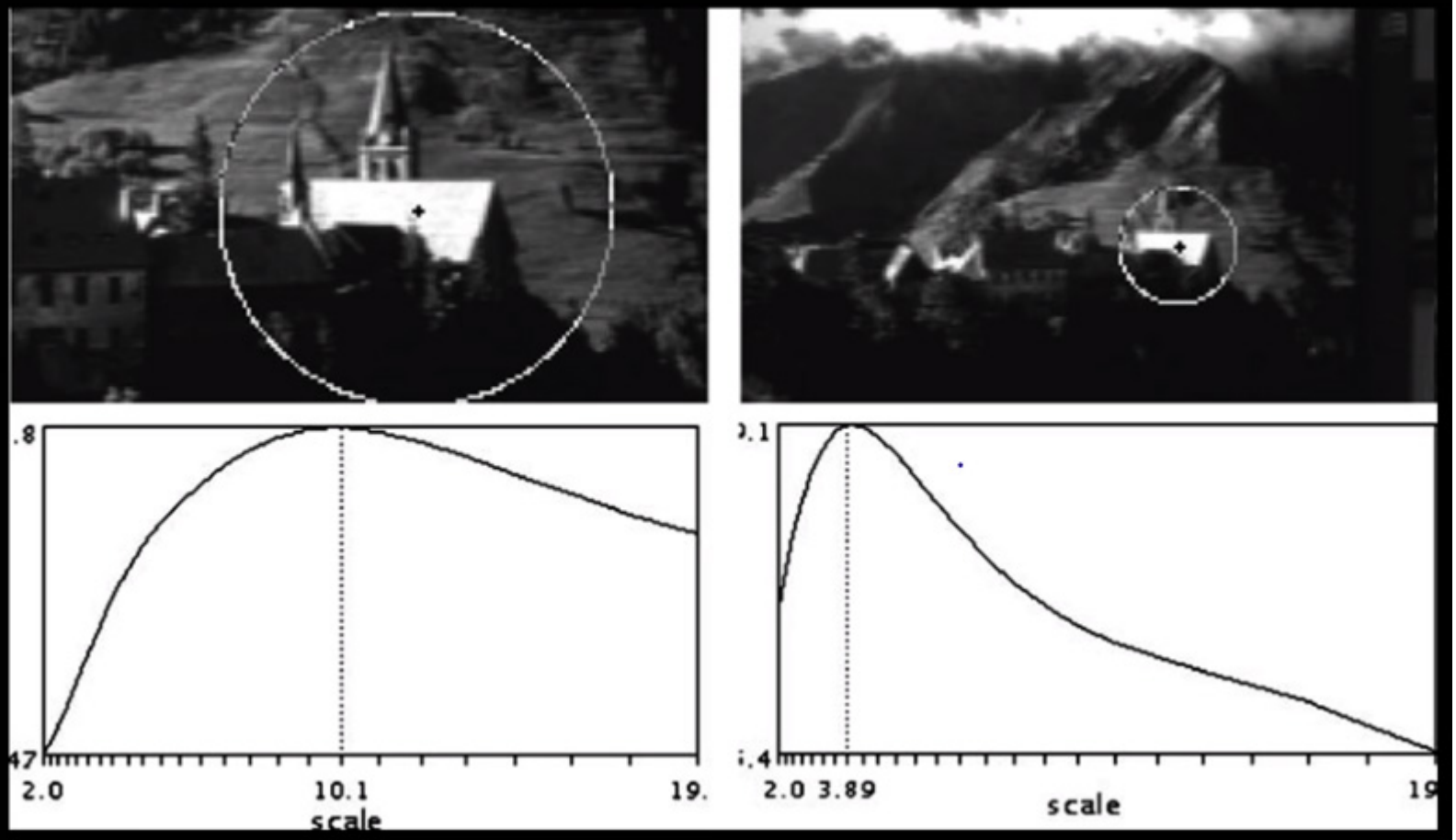
Scale Invariant Detection

One method:

- At a point, compute the scale invariant function over different size neighborhoods (different scales).
- Choose the scale for each image at which the function is a maximum

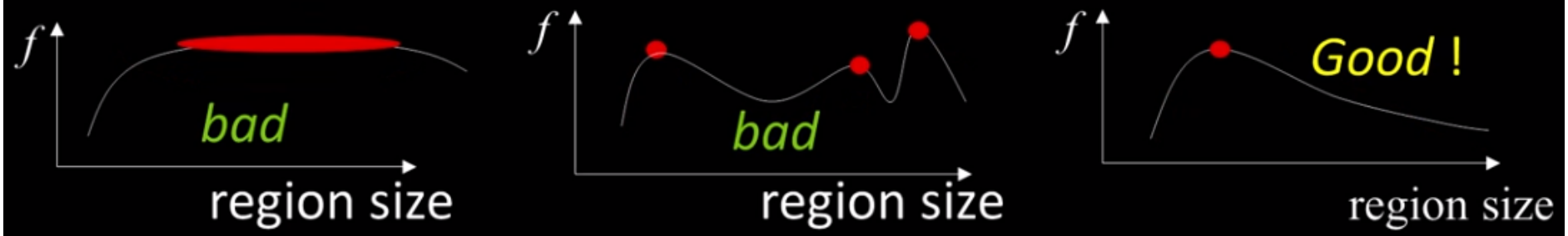


Scale sensitive response



Scale Invariant Detection

- A "good" function for scale detection: has one stable sharp peak



For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

Scale Invariant Detection

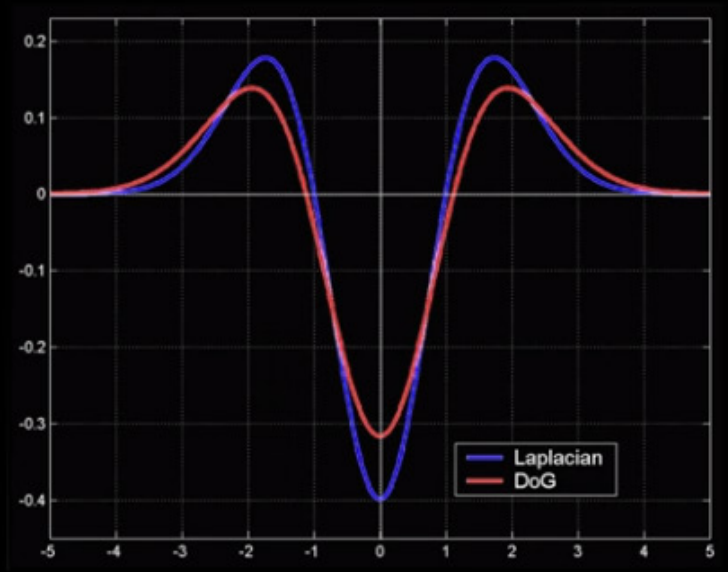
Function is just application of a kernel: $f = \text{Kernel} * \text{Image}$

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian of Gaussian - LoG)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

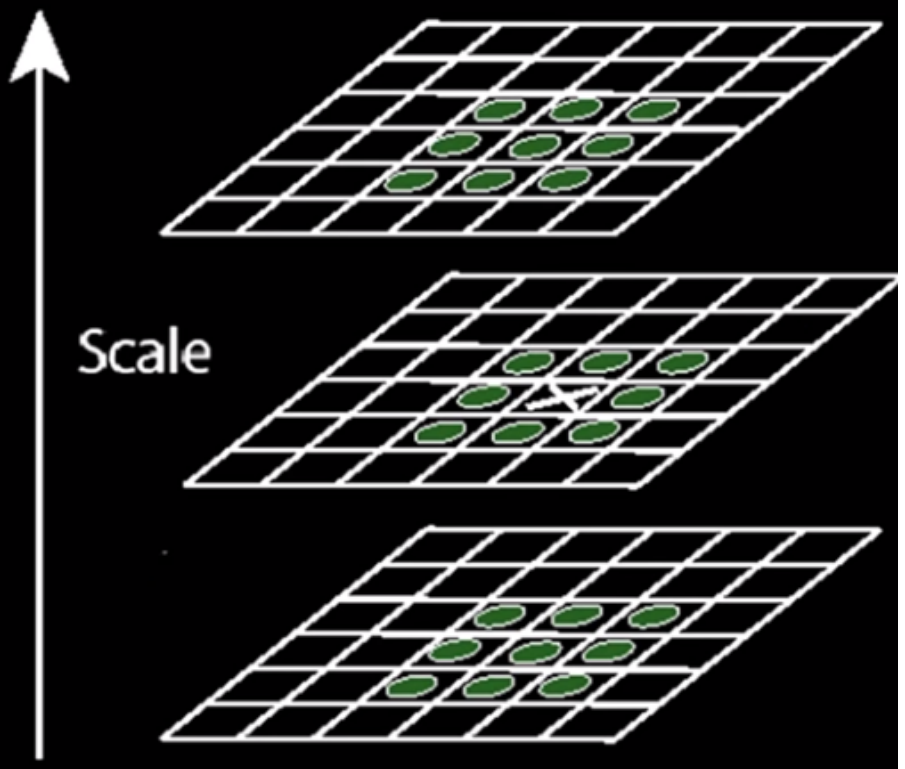
(Difference of Gaussians)



Note: both kernels are invariant to *scale* and *rotation*

Key point localization

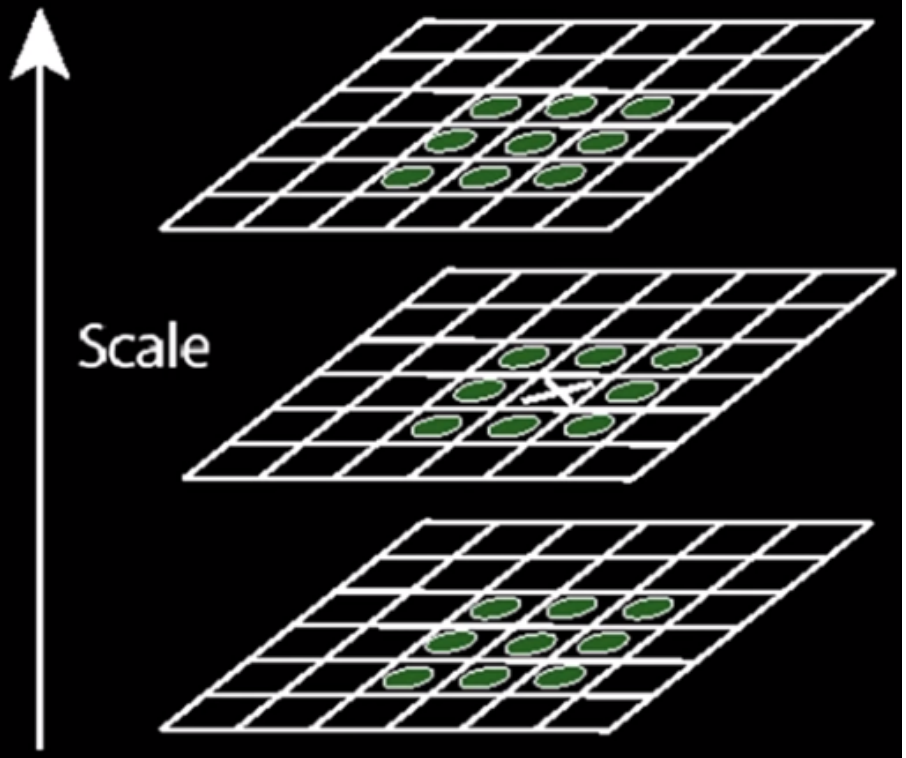
- General idea: find robust extremum (maximum or minimum) both in space and in scale.



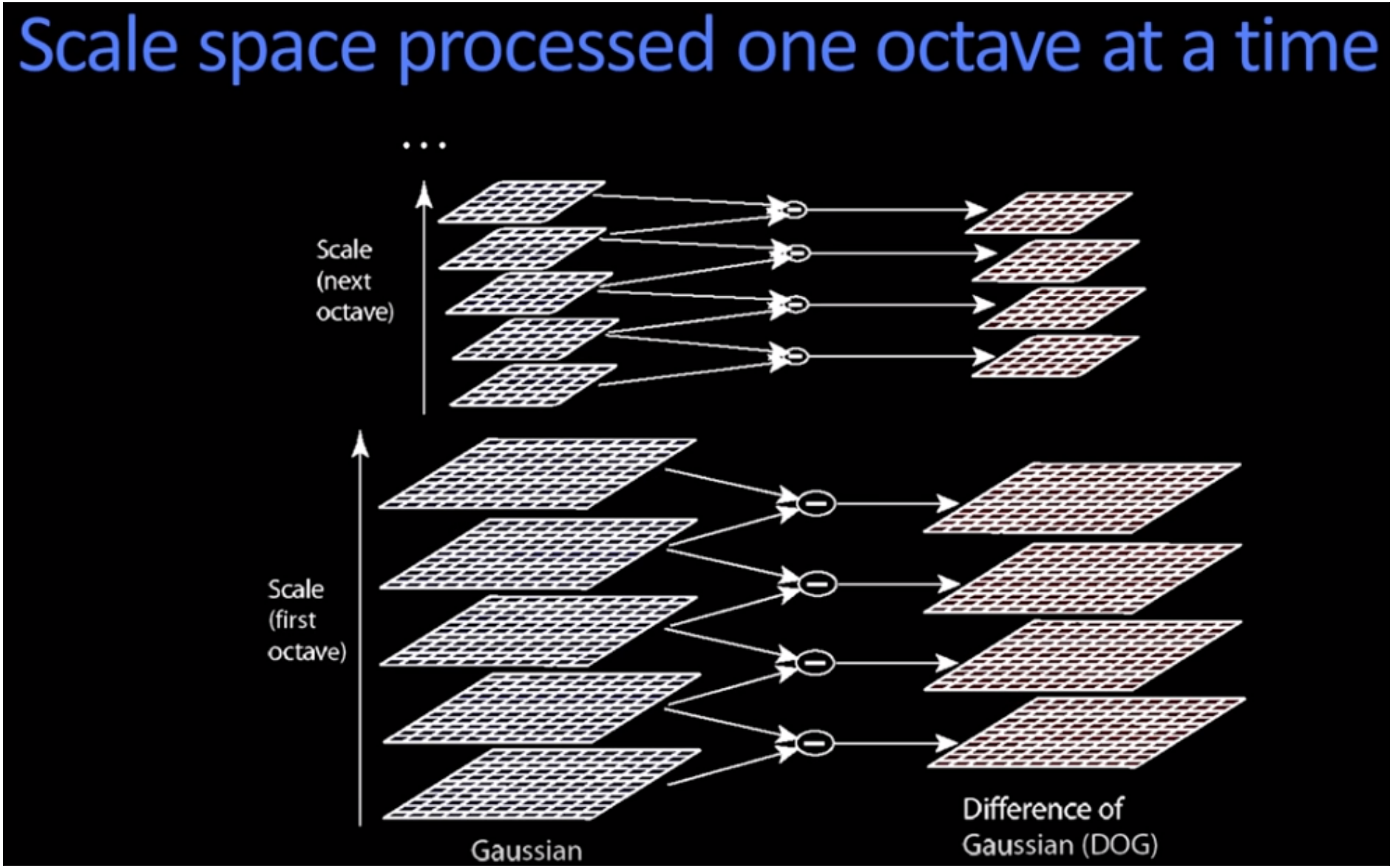
Key point localization

- SIFT: Scale Invariant Feature Transform

- Specific suggestion: use DoG pyramid to find maximum values (remember edge detection?) – then eliminate "edges" and pick only corners.

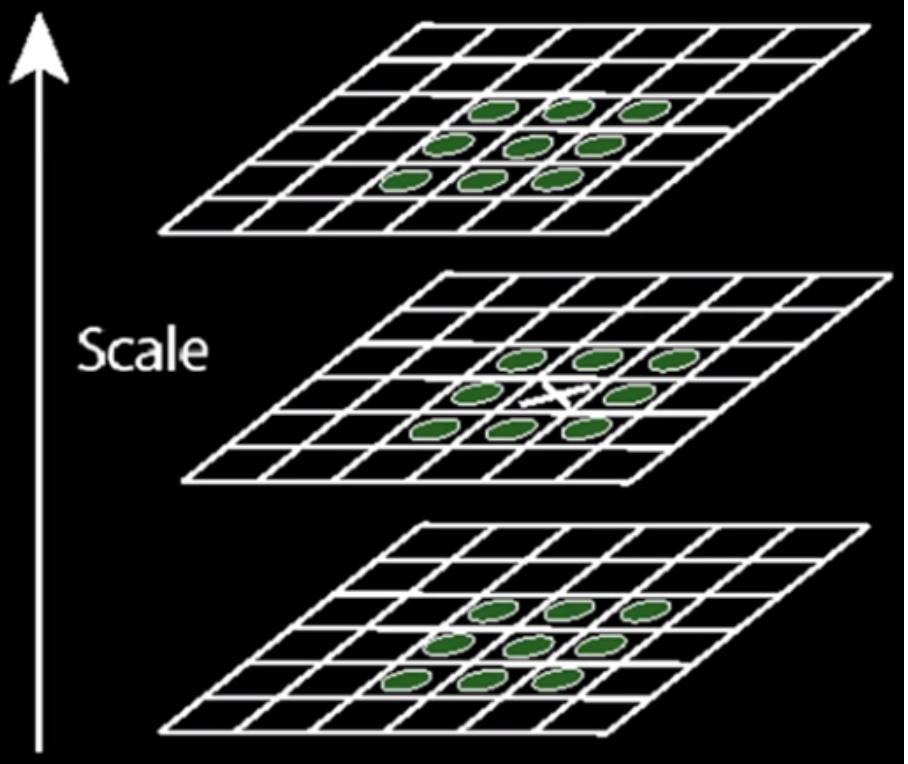


3.4 – Scale-Invariant Detection

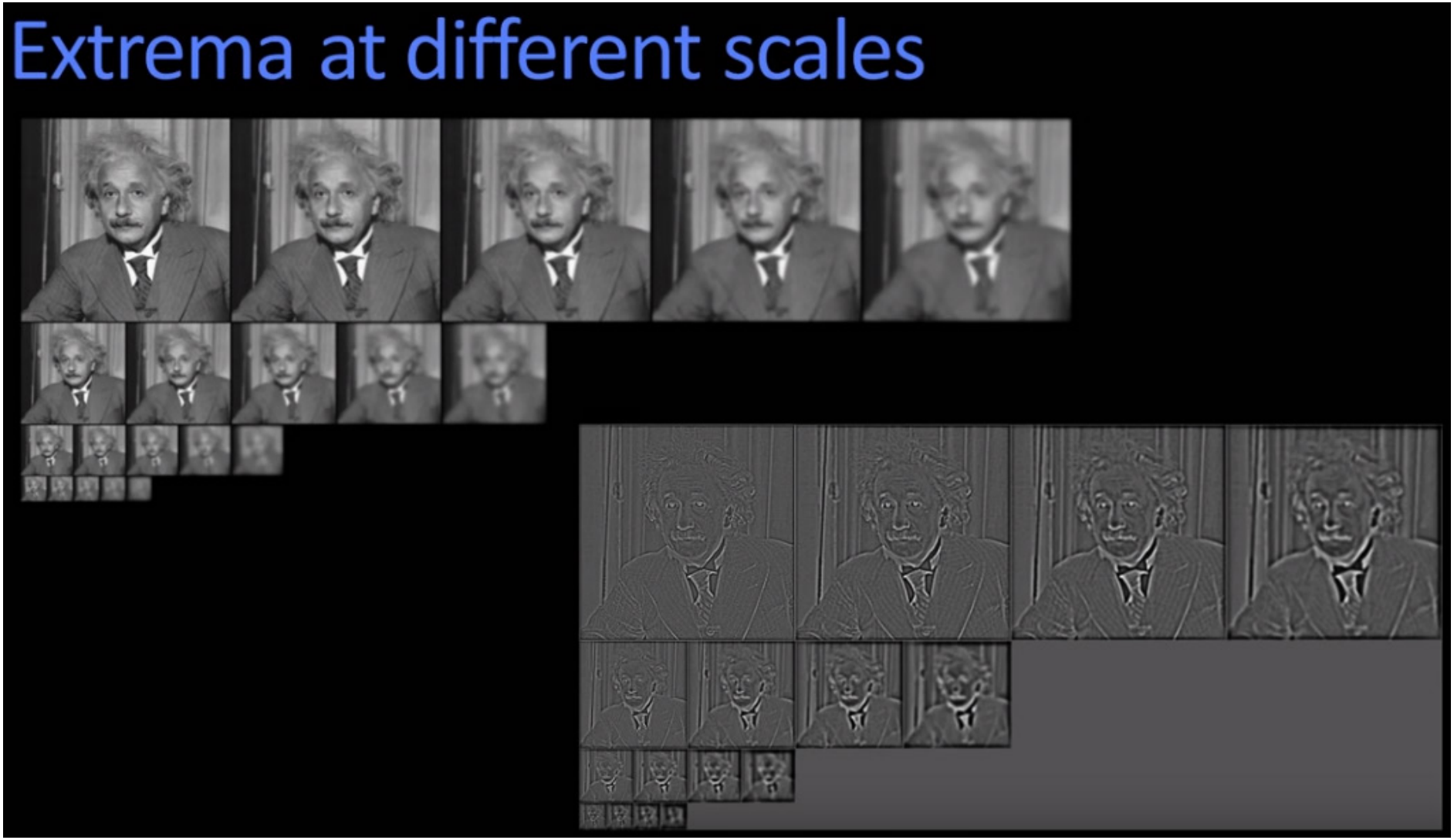


Key point localization

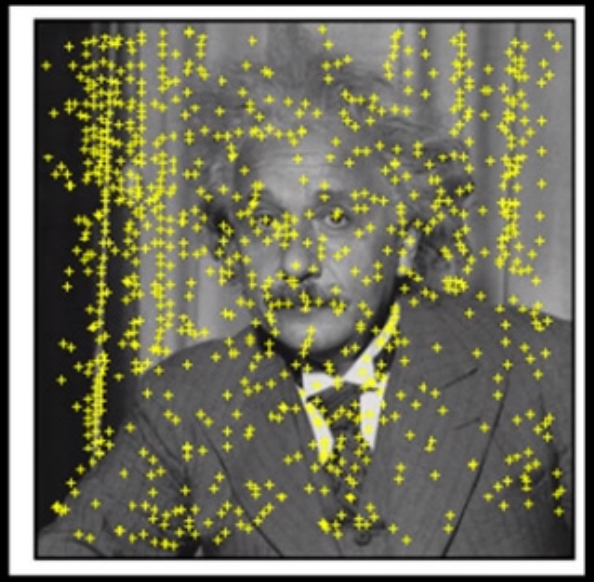
(Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.)



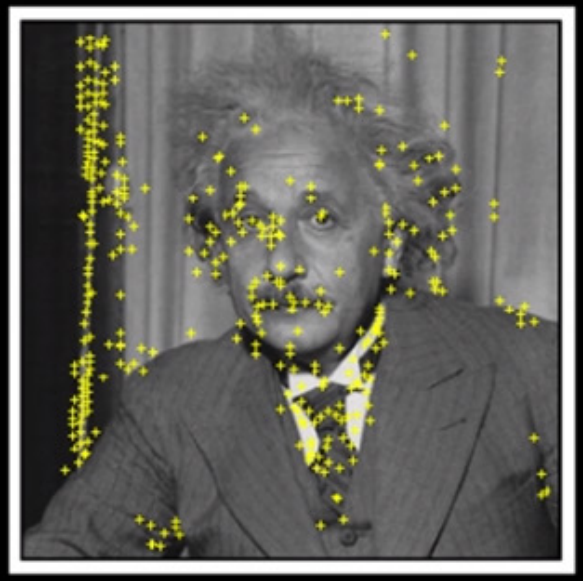
3.4 – Scale-Invariant Detection



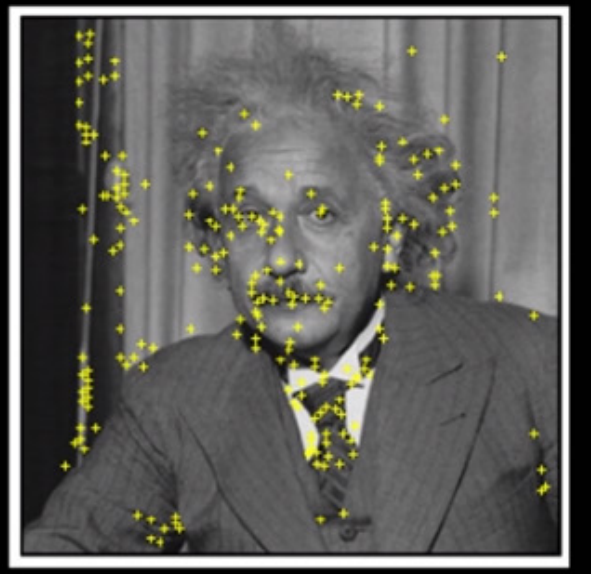
Remove low contrast, edge bound



Extrema points



Contrast > C



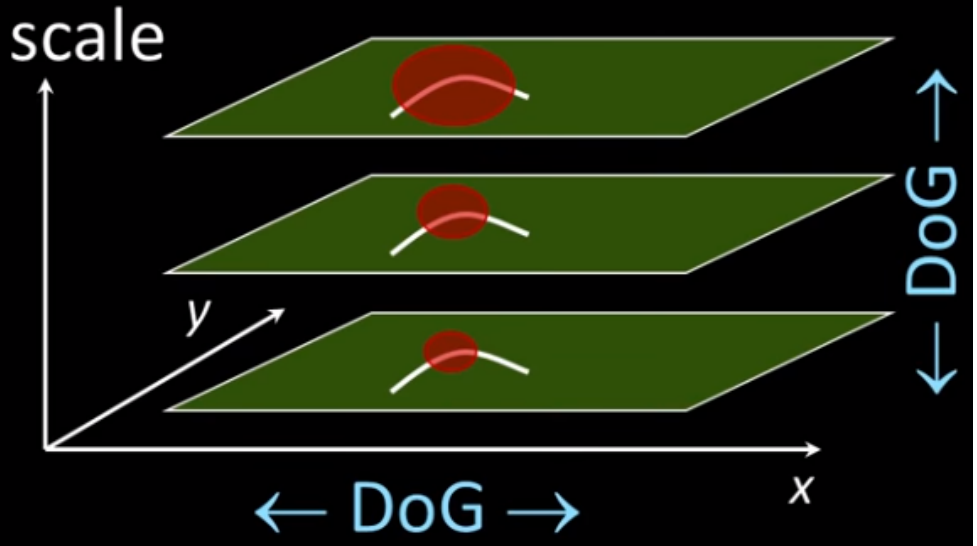
Not on edge

Scale Invariant Detectors

SIFT (Lowe)

Find local maximum of:

– Difference of Gaussians in space and scale



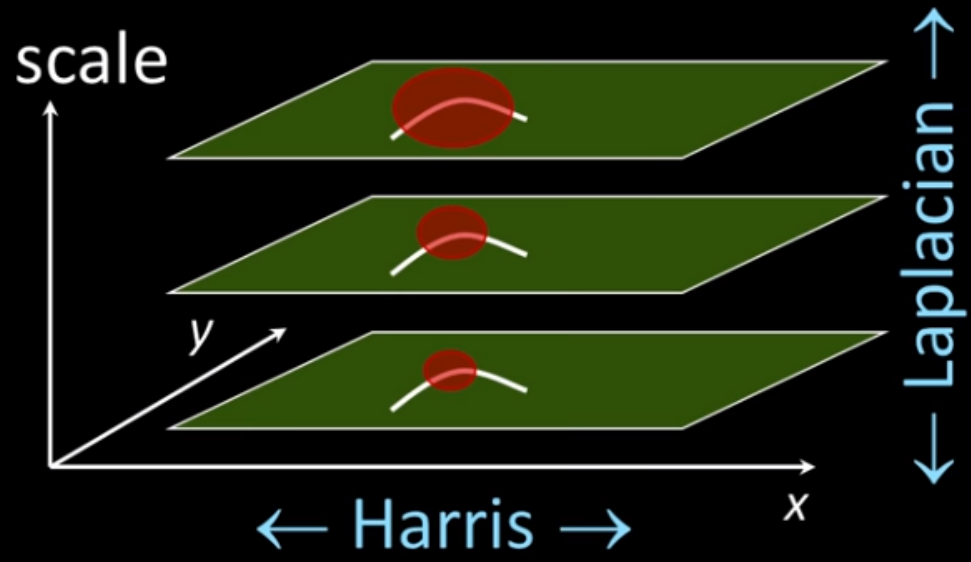
D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

Scale Invariant Detectors

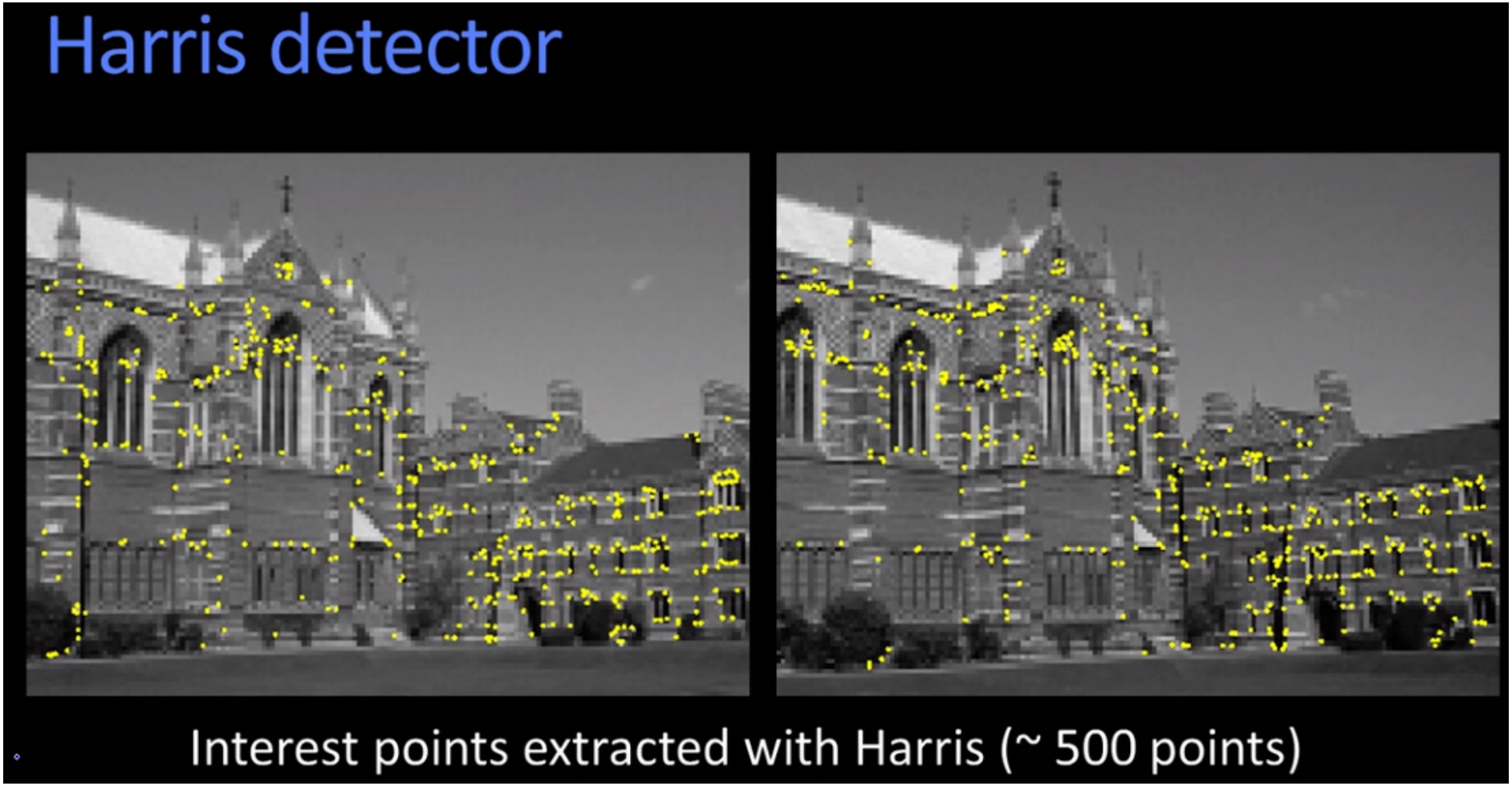
Harris-Laplacian

Find local maximum of:

- Harris corner detector in space (image coordinates)
- Laplacian in scale



K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001

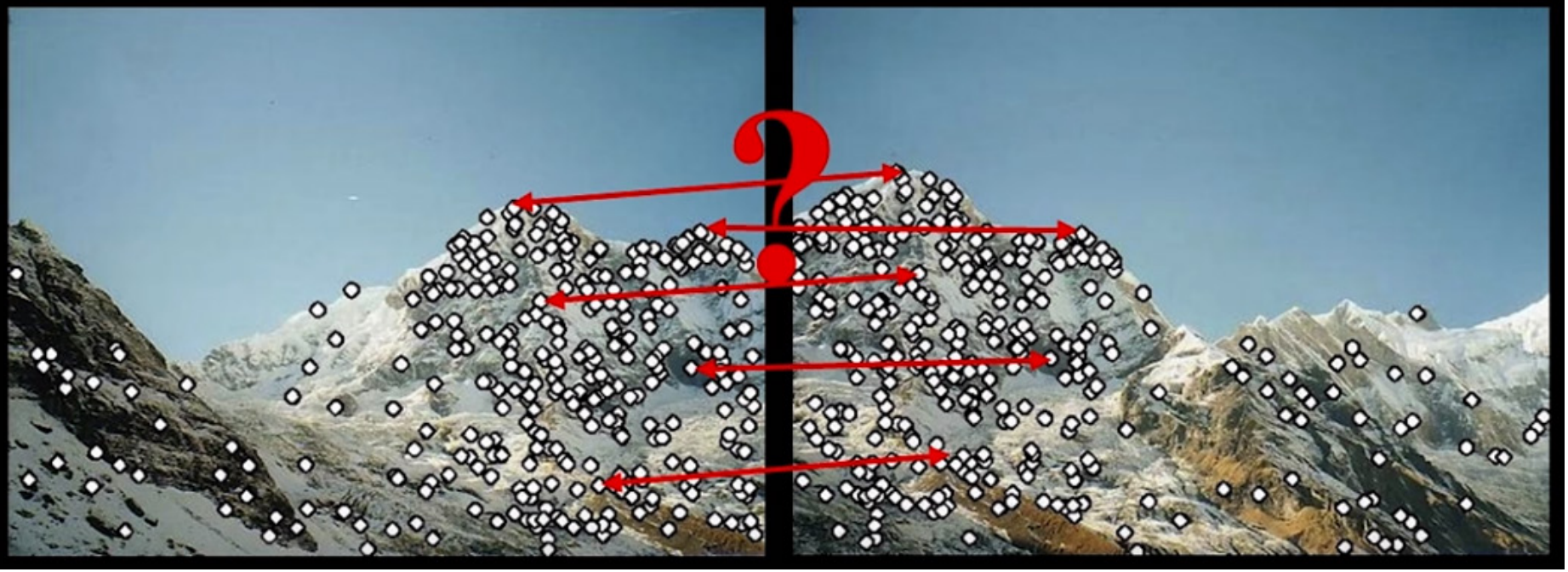

Harris detector
Interest points extracted with Harris (~ 500 points)

3 - Unconventional Image Acquisition: Panorama

1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. **Corner Detection**
4. **Scale-Invariant Detection**
5. **Scale-Invariant Description**
6. **RANSAC**

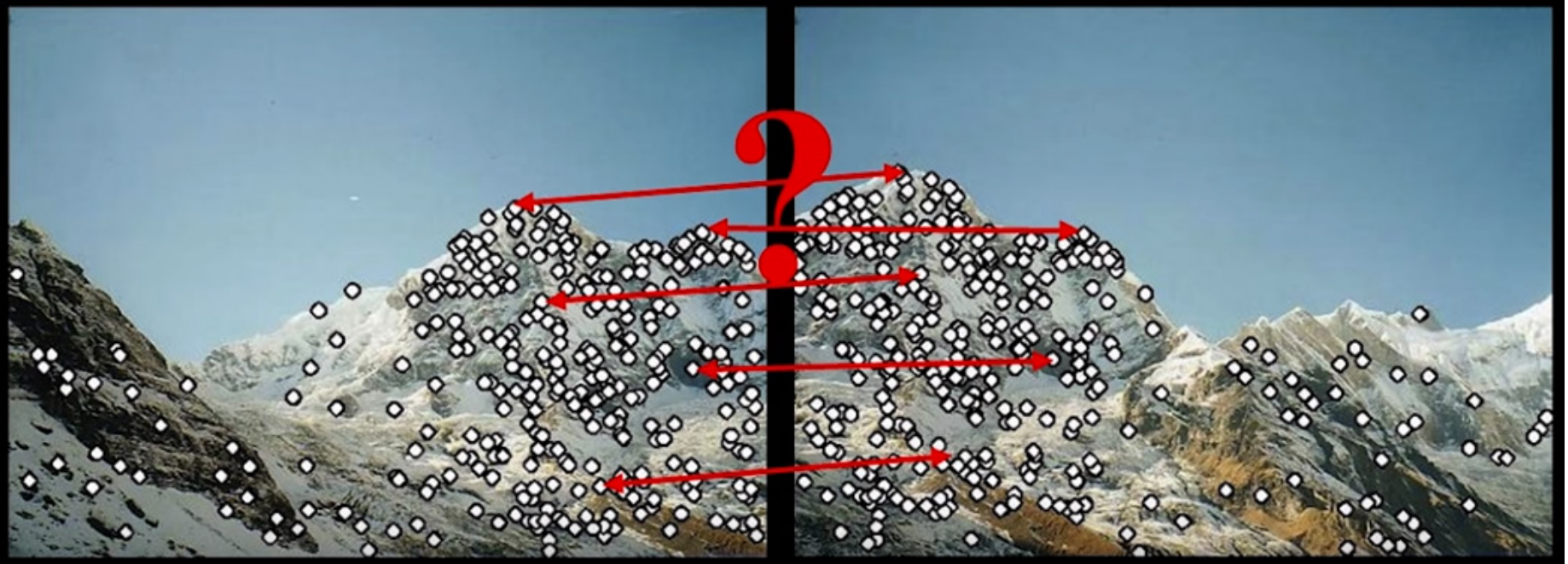
Point Descriptors

- Now: How to match them?



Criteria for Point Descriptors

- We want the descriptors to be the (almost) same in both image – *invariant.*



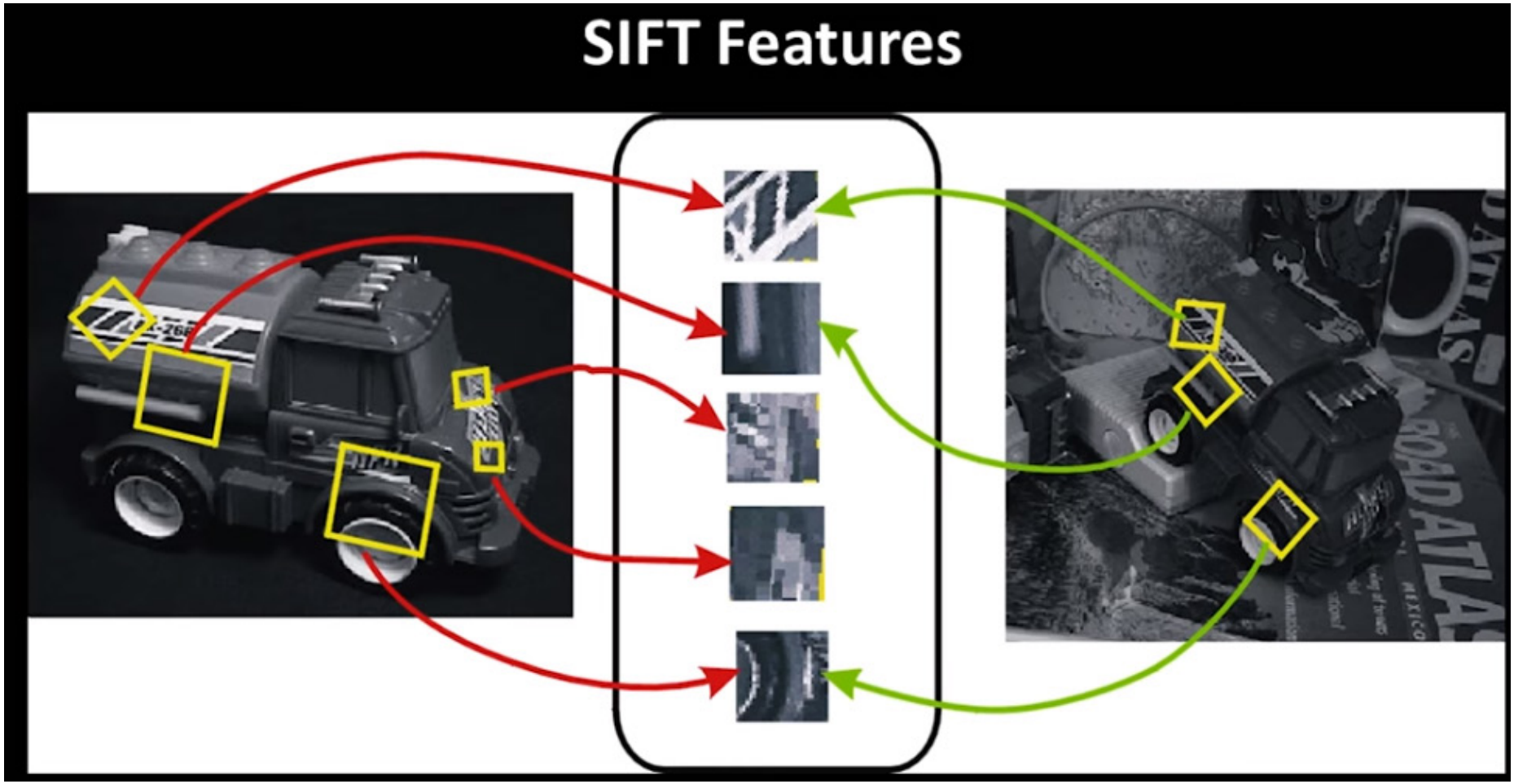
- We also need the descriptors to be *distinctive.*

SIFT: Scale Invariant Feature Transform

- Motivation: The Harris operator was not invariant to scale and correlation was not invariant to rotation.

- For better image matching, Lowe's goals were:
  - To develop an interest operator – a *detector* – that is invariant to scale and rotation.
  - Also: create a descriptor that was robust to the variations corresponding to typical viewing conditions. *The descriptor is the most-used part of SIFT.*

3.5 – Scale-Invariant Description



Overall SIFT Procedure

- Scale-space extrema detection
- Keypoint localization

} *Or use Harris-Laplace or other method*

- Orientation assignment
- Keypoint description

Overall SIFT Procedure

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment

   Compute best orientation(s) for each keypoint region.
4. Keypoint description

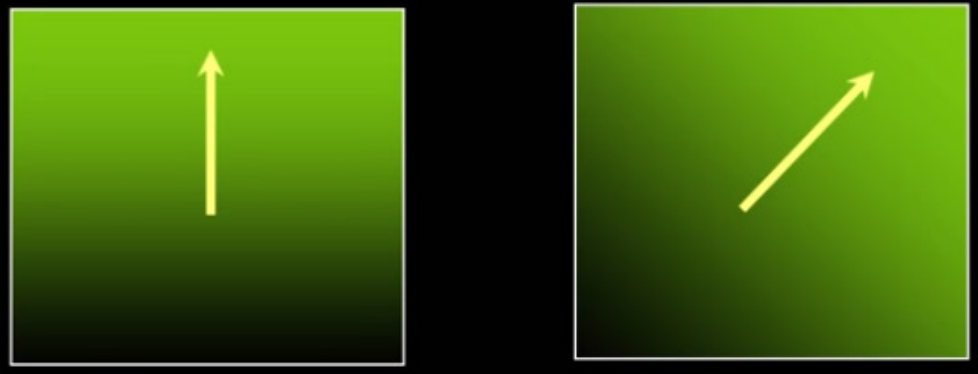
   Use local image gradients at selected scale and rotation to describe each keypoint region.

But first... normalization...

- Rotate the window to standard orientation
- Scale the window size based on the scale at which the point was found.

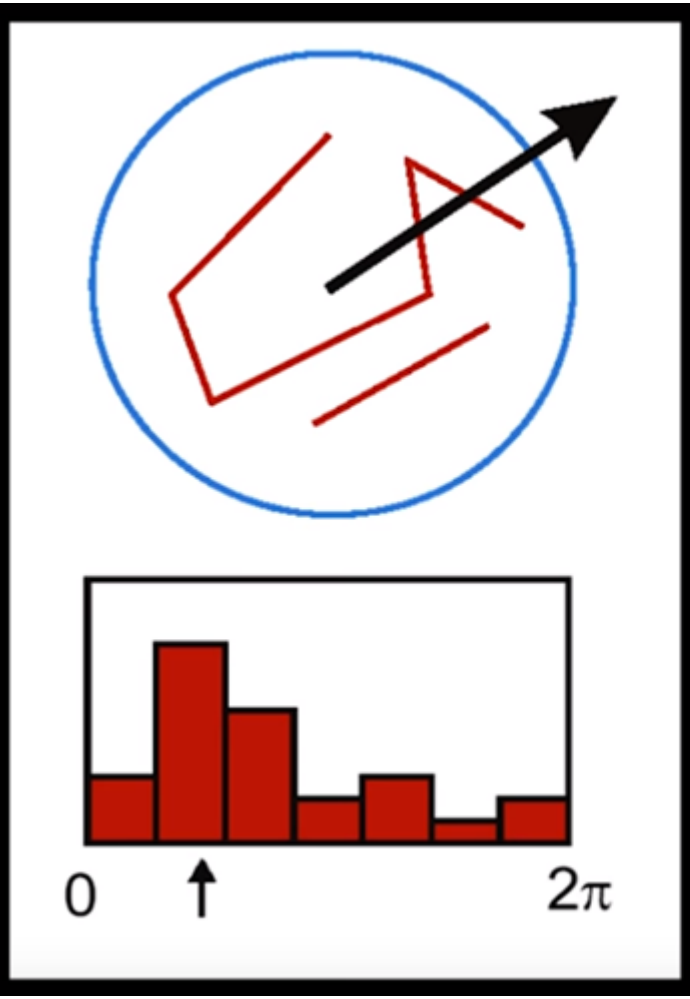
Descriptors Invariant to Rotation

- Find the dominant direction of gradient – that is the base orientation.



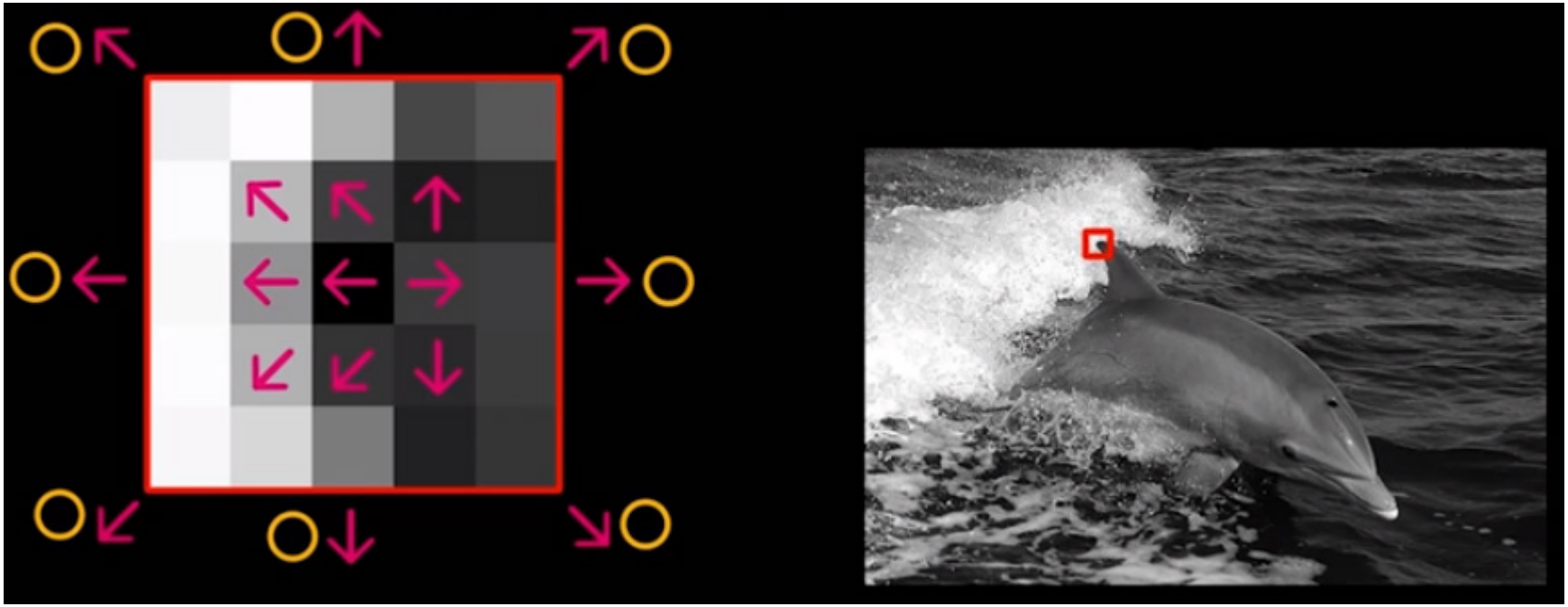
Compute image derivatives relative to this orientation

3.5 – Scale-Invariant Description



- Create histogram of local gradient directions at *selected* scale – 36 bins
- Assign canonical orientation at peak of smoothed histogram

3.5 – Scale-Invariant Description

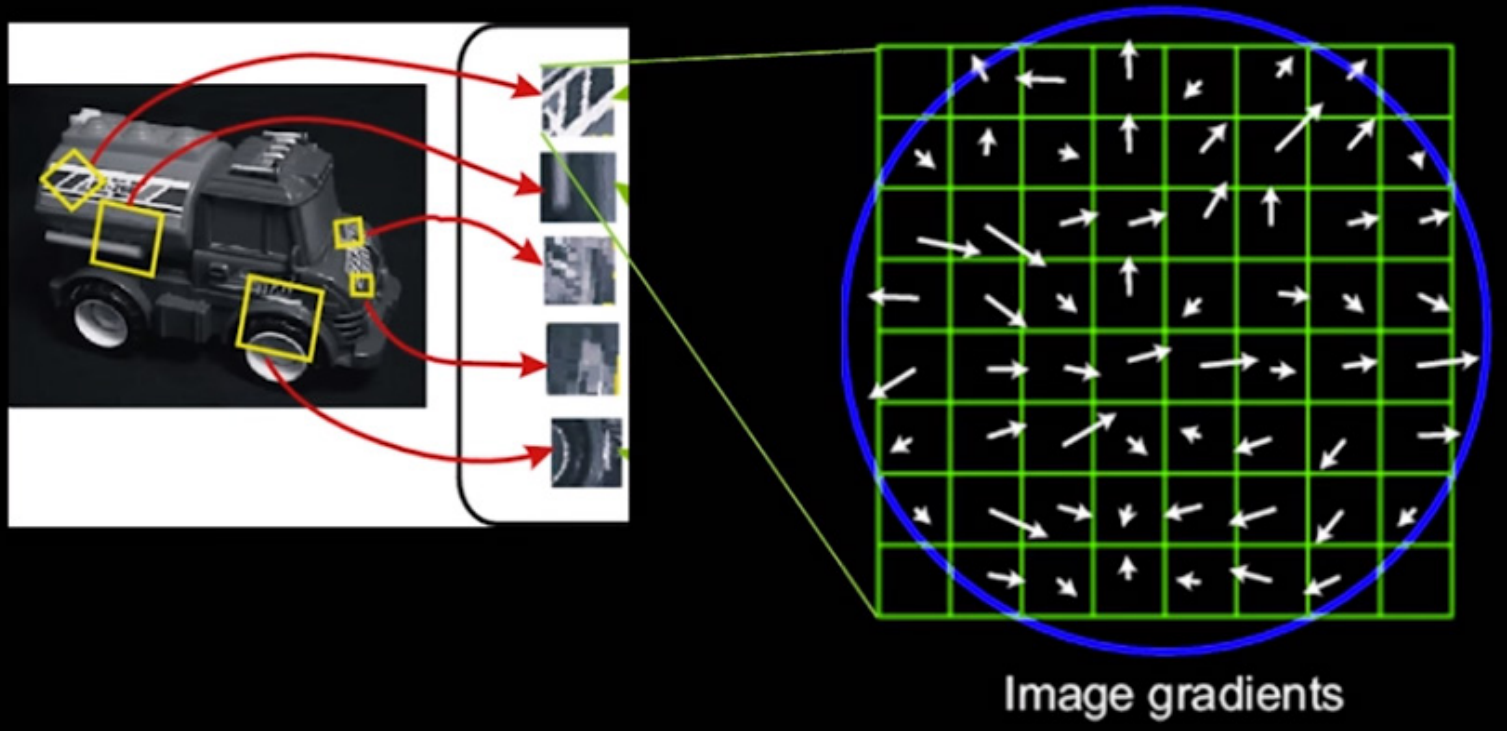


4. Keypoint Descriptors

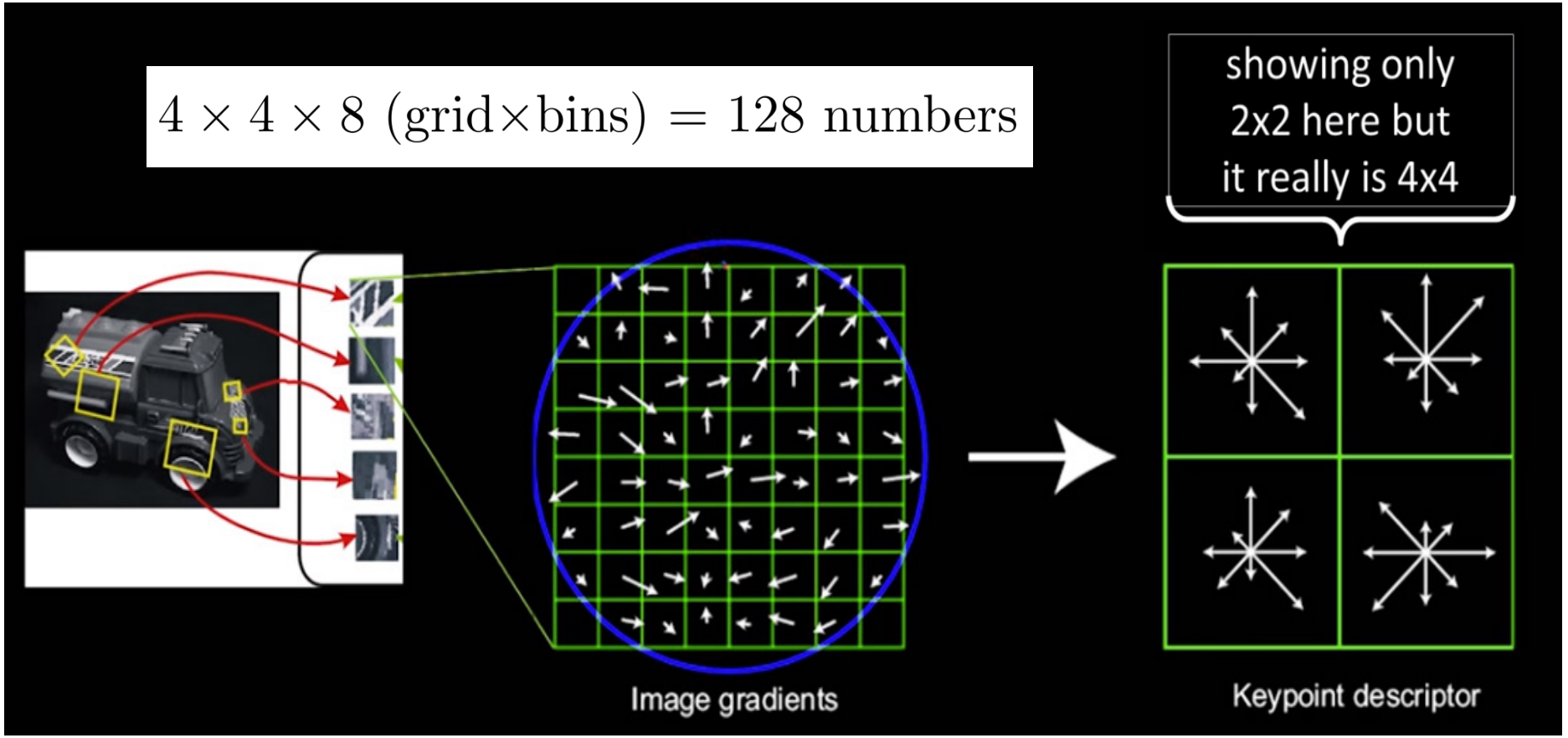
- Next is to compute a descriptor for the local image region about each keypoint that is:
  - Highly ***distinctive***
  - As ***invariant*** as possible to variations such as changes in viewpoint and illumination

Compute a feature vector based upon:

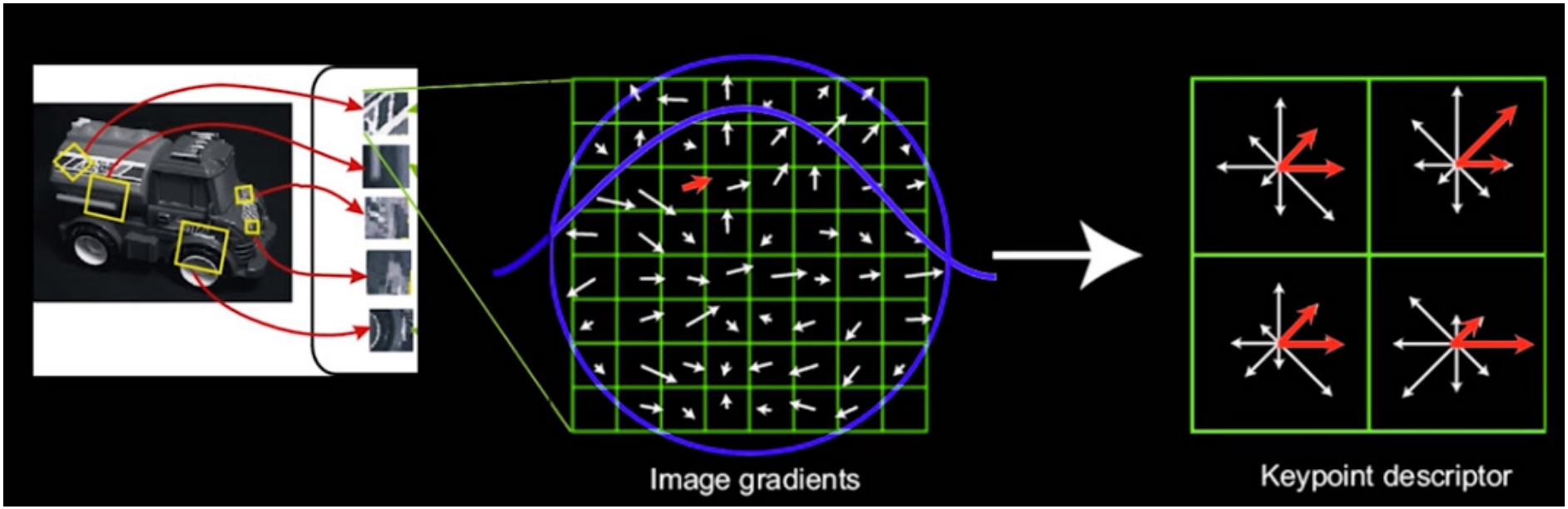
- histograms of gradients
- weighted by a centered Gaussian



3.5 – Scale-Invariant Description

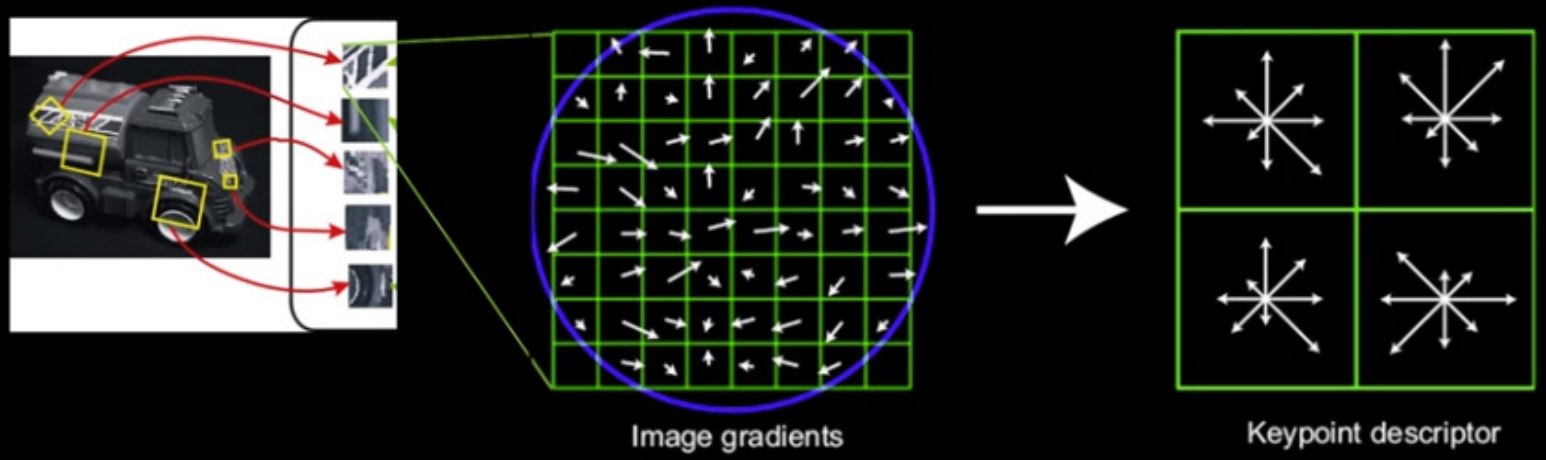


3.5 – Scale-Invariant Description



Reduce effect of illumination

- Clip gradient magnitudes to avoid excessive influence of high gradients
- 128-dim vector normalized to magnitude 1.0



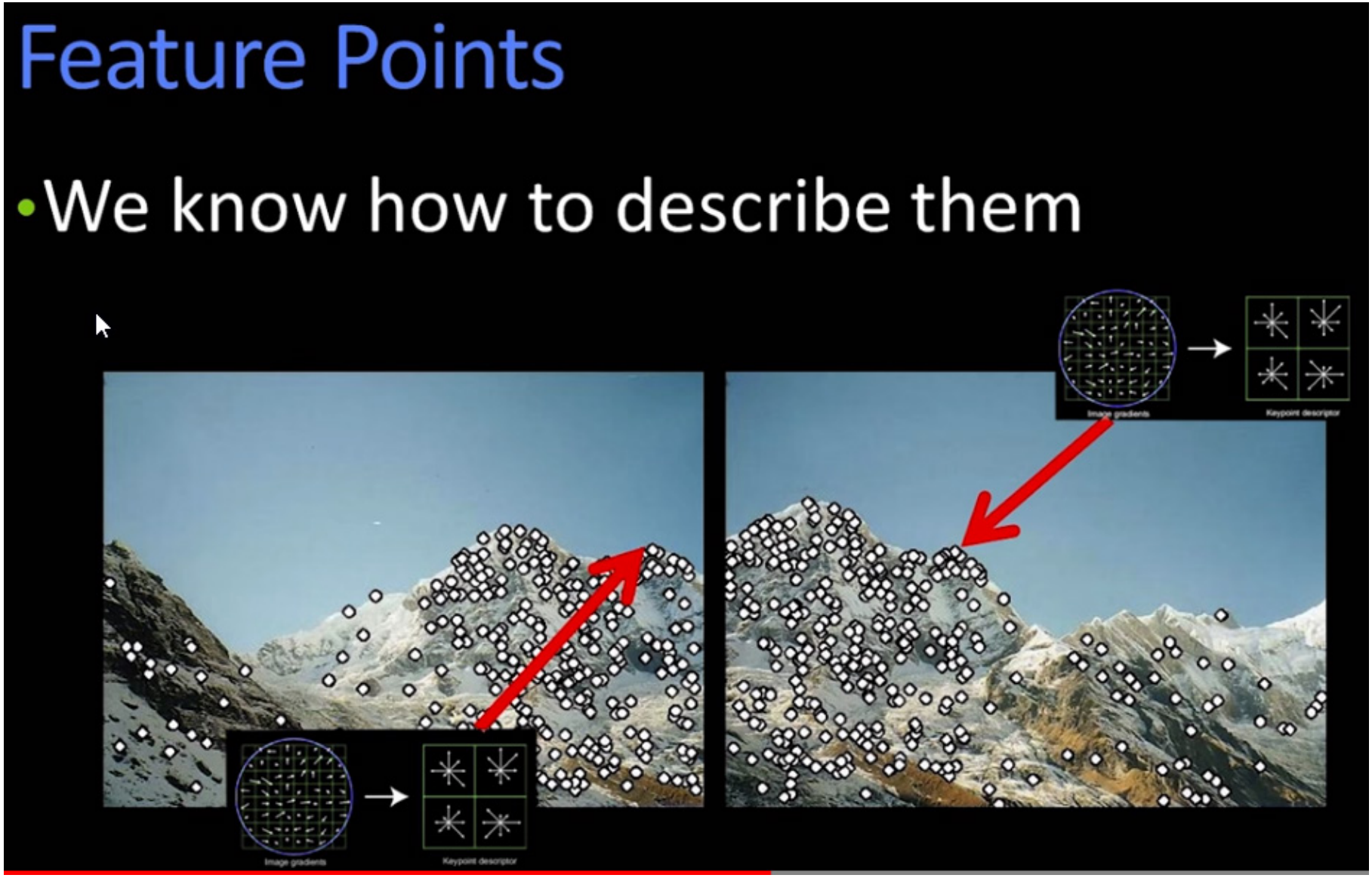
3.5 – Scale-Invariant Description

Image transformation	Location and scale match	Orientation match
Decrease constrast by 1.2	89.0 %	86.6 %
Decrease intensity by 0.2	88.5 %	85.9 %
Rotate by 20°	85.4 %	81.0 %
Scale by 0.7	85.1 %	80.3 %
Stretch by 1.2	83.5 %	76.1 %
Stretch by 1.5	77.7 %	65.0 %
Add 10% pixel noise	90.3 %	88.4 %
All previous	**78.6 %**	**71.8 %**

20 different images, around 15,000 keys

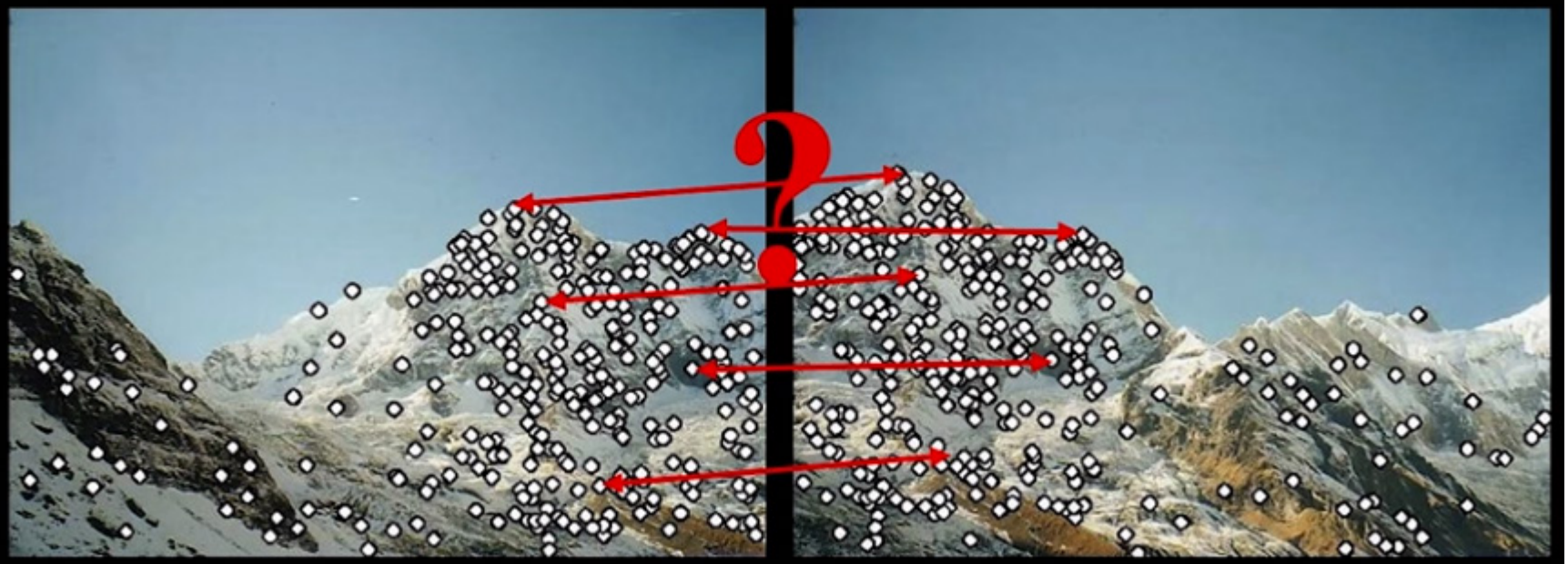
3 - Unconventional Image Acquisition: Panorama

1. **Homogeneous Coordinates and Image Transforms**
2. **Visual Features For Matching**
3. **Corner Detection**
4. **Scale-Invariant Detection**
5. **Scale-Invariant Description**
6. **RANSAC**


Feature Points
We know how to describe them
Image gradients
Keypoint descriptor
Image gradients
Keypoint descriptor

Feature Points

- Next question: How to match them?



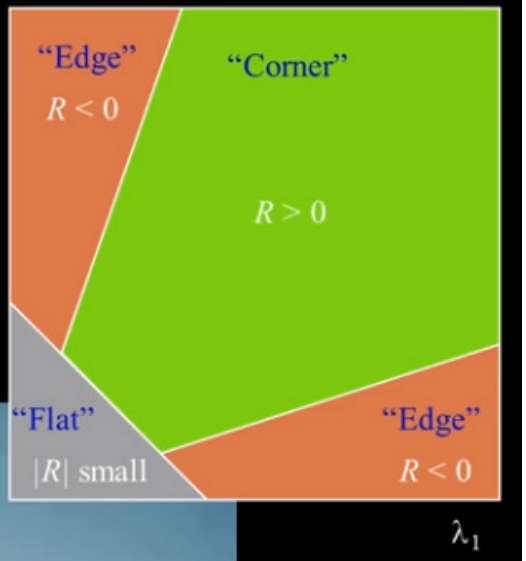
Nearest-neighbor matching to feature database

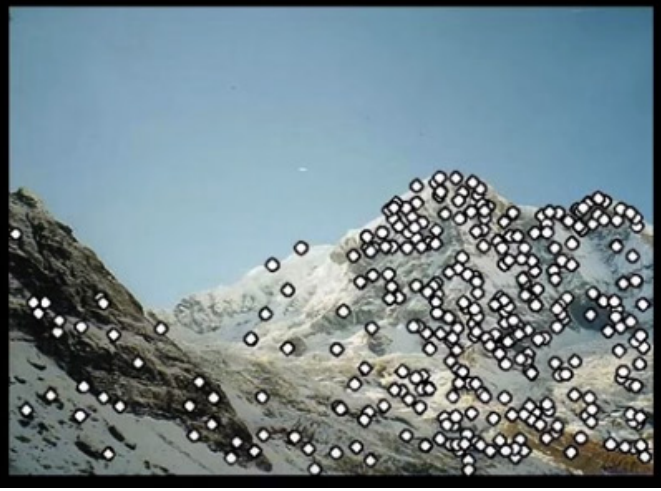
- Better: hypotheses are generated by *approximate nearest neighbor* matching of each feature to vectors in the database
  - SIFT uses best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

Feature-based alignment to find transforms

Overall strategy:

1. Compute features – detect and describe

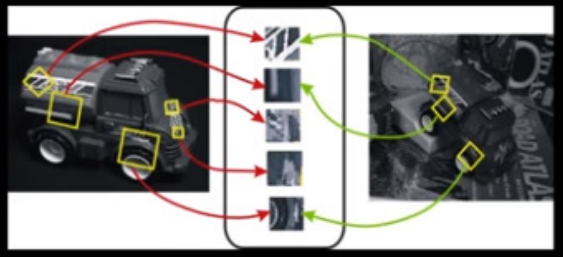


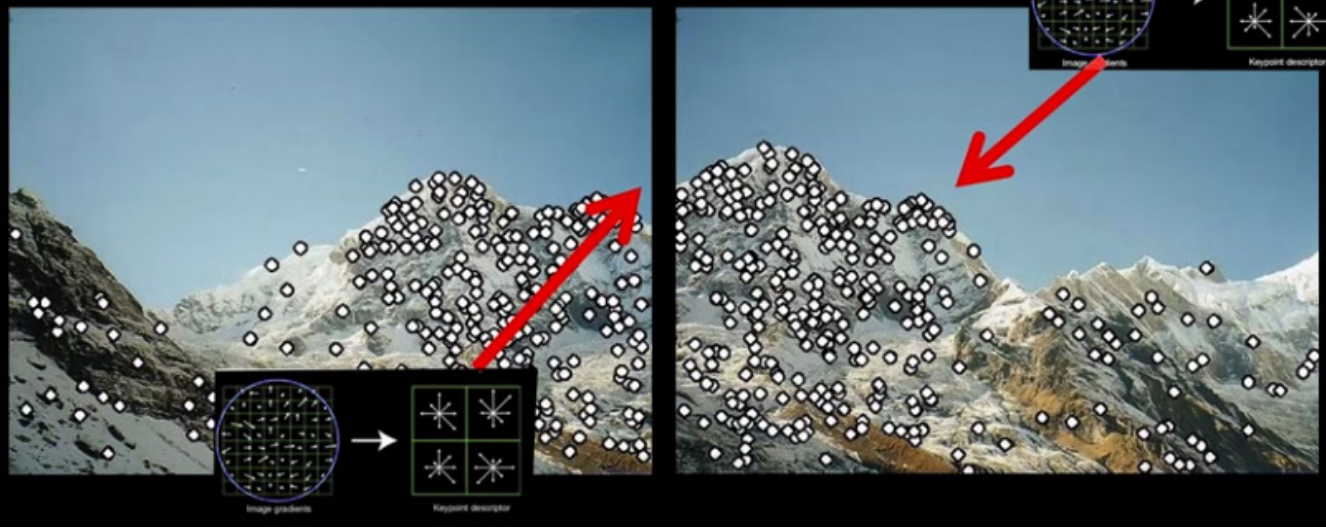


Feature-based alignment to find transforms

Overall strategy:

1. Compute features – detect and describe



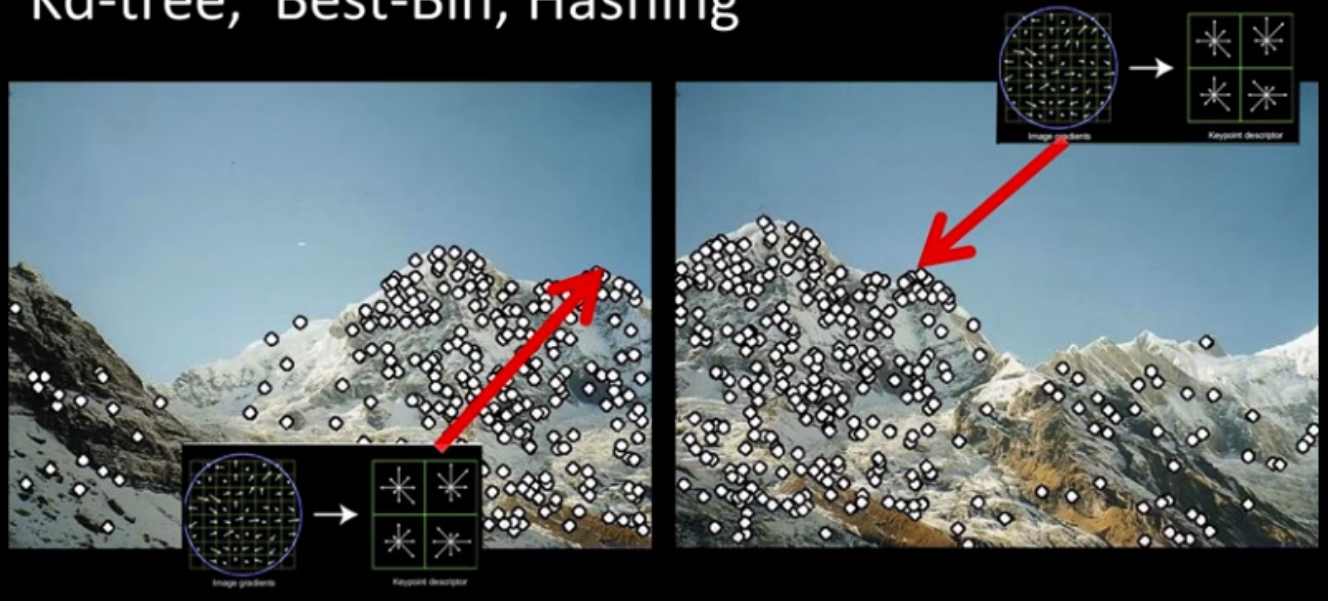


Feature-based alignment to find transforms

Overall strategy:

2. Find some useful matches:

   Kd-tree, Best-Bin, Hashing

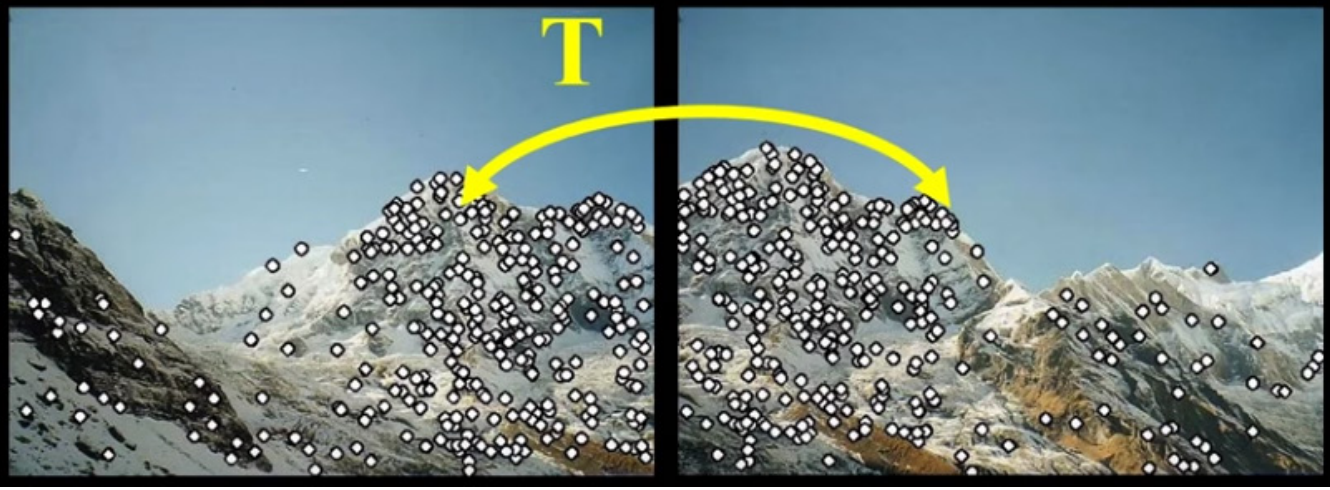


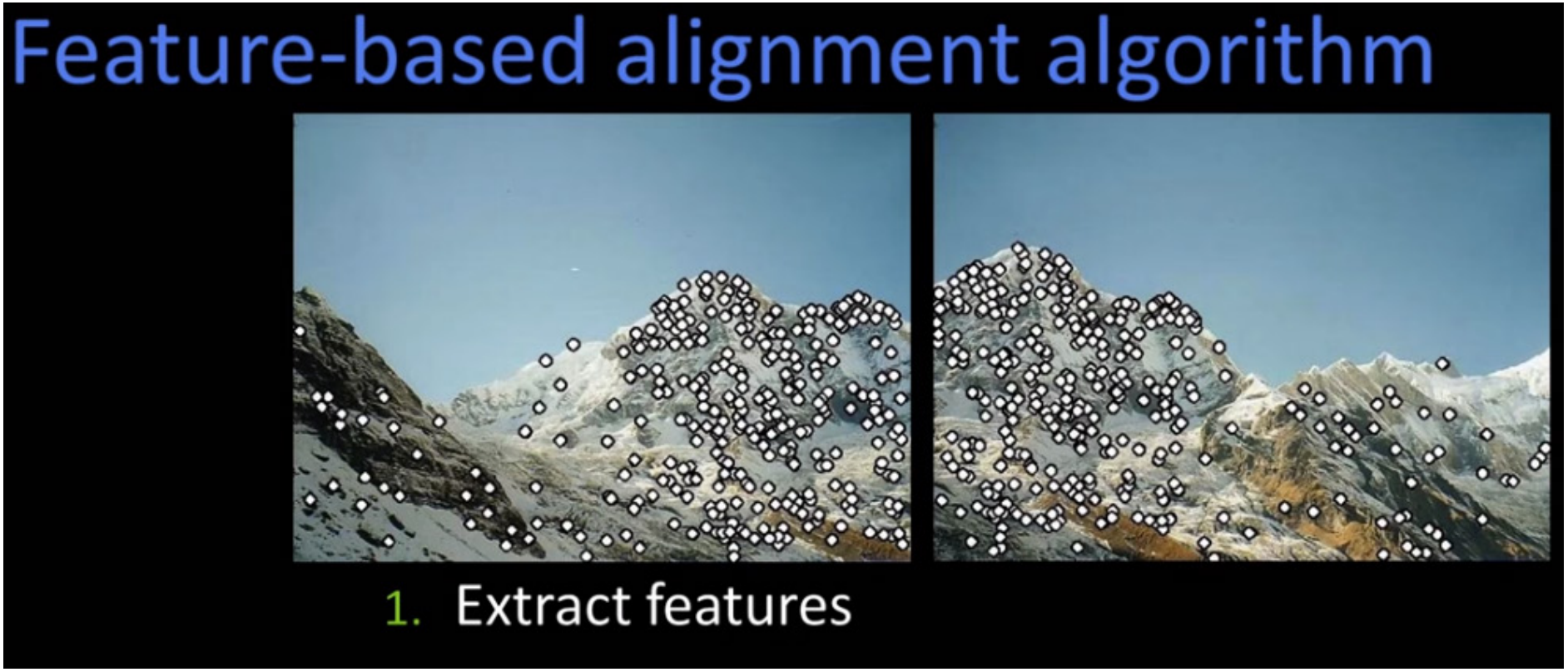
Feature-based alignment to find transforms

Overall strategy:

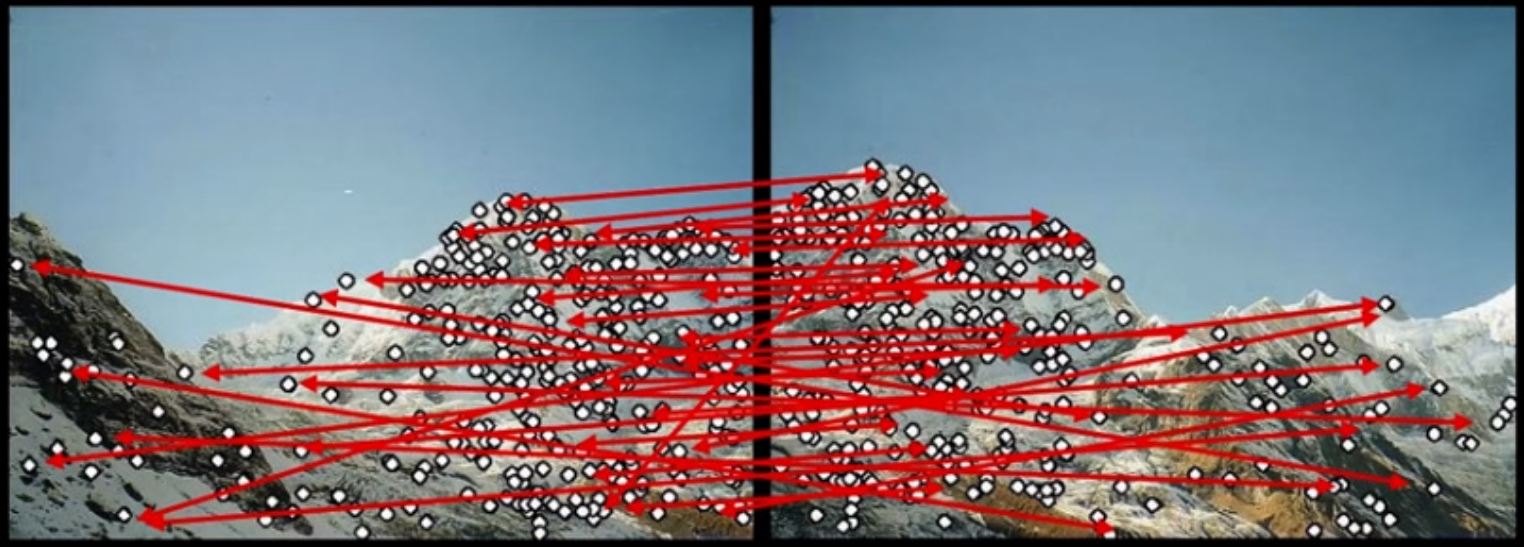
3. Compute and apply the best transformation:

   e.g. affine, translation, or homography




Feature-based alignment algorithm
1. Extract features

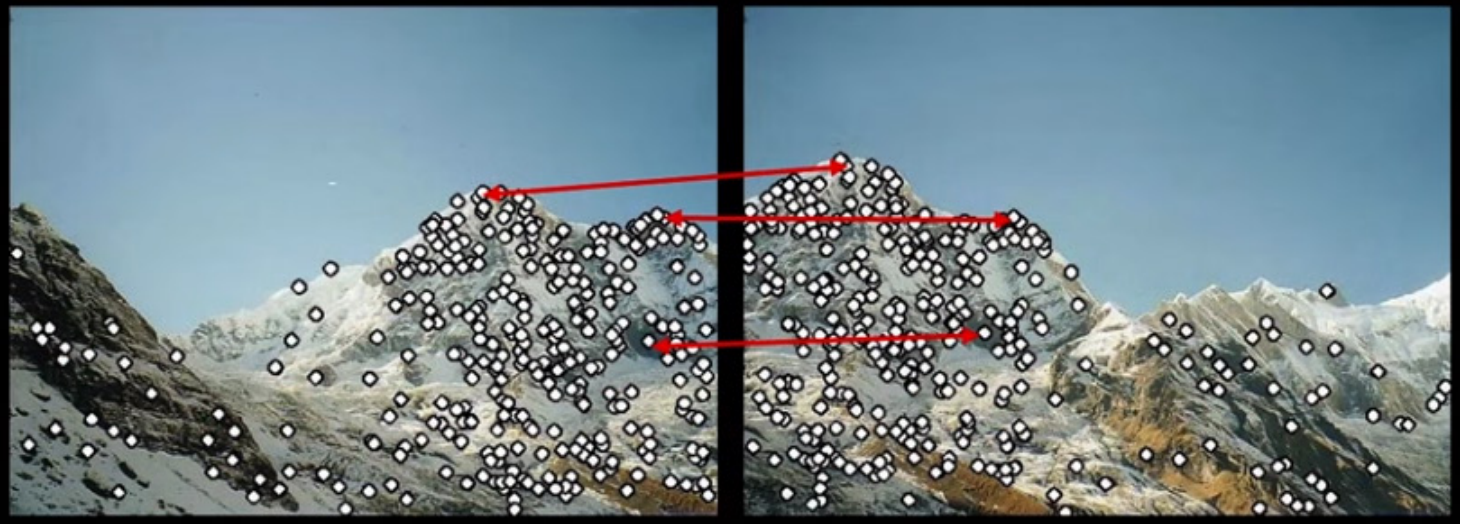
Feature-based alignment algorithm



2. Compute ***putative*** *matches* – e.g. "closest descriptor"

Kd-tree, best bin, etc...

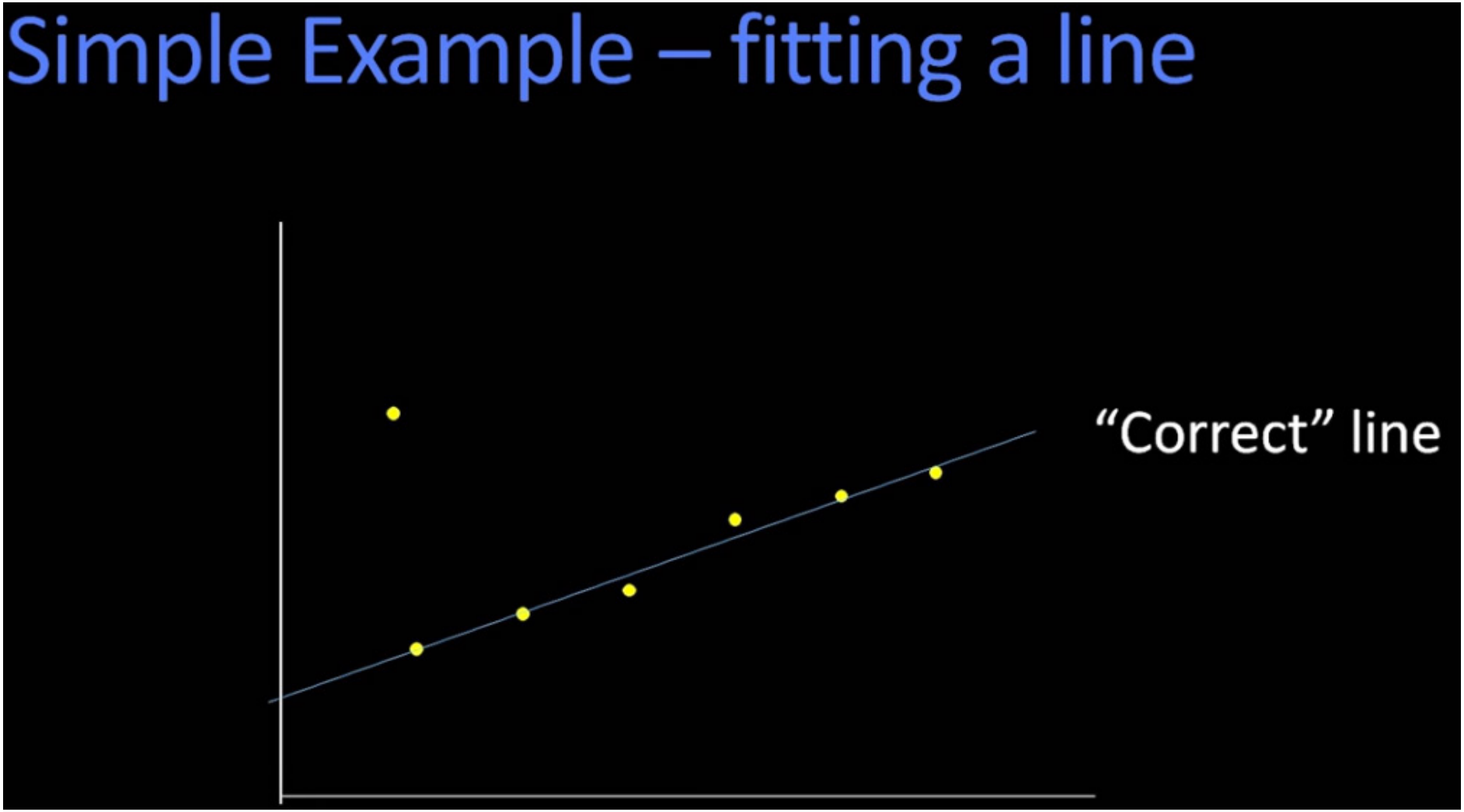
Feature-based alignment algorithm

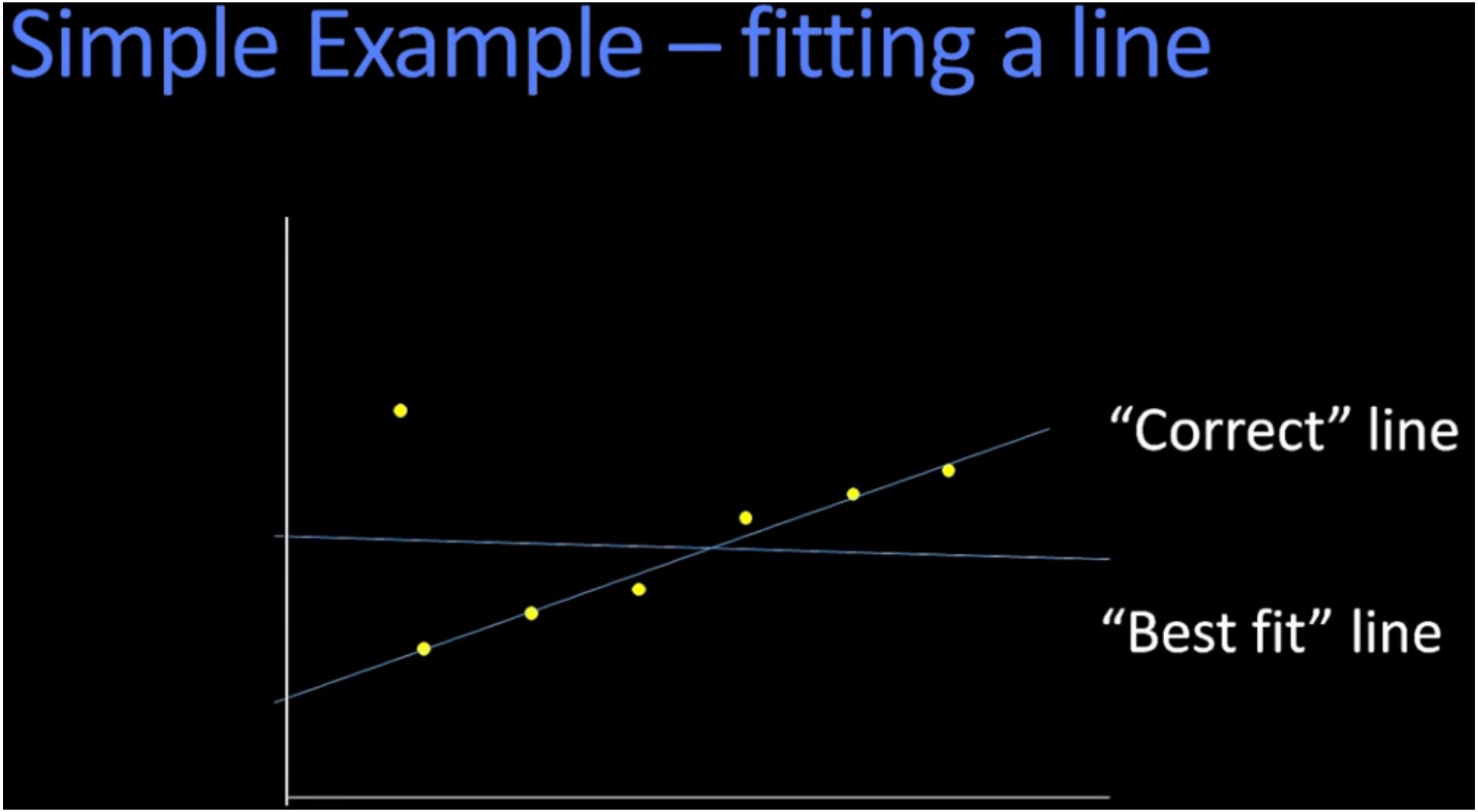


3. Loop until happy:
   - *Hypothesize* transformation *T* from some matches
   - *Verify* transformation (search for other matches consistent with *T*) – mark best

"Find consistent matches"?

- Some "best" matches are correct
- Some are not. And the "not" are not part of any other consistent match…
- Need to find the right ones so can compute the pose/transform/fundamental… *the model.*
- Today: Random Sample Consensus (RANSAC)


Simple Example – fitting a line
"Correct" line


Simple Example – fitting a line
"Correct" line
"Best fit" line

RANSAC: main idea

- Fitting a line (model) is easy if we know which points belong and which do not.
- If we had a proposed line (model), we could probably guess which points belong to that line (model): *inliers*.
- RANdom SAmple Consensus: randomly pick some points to define your line (model). Repeat enough times until you find a good line (model) – on with many inliers.

RANSAC for general model

A given model type has a *minimal set* – the smallest number of samples from which the model can be computed.

- Line: 2 points

RANSAC for general model

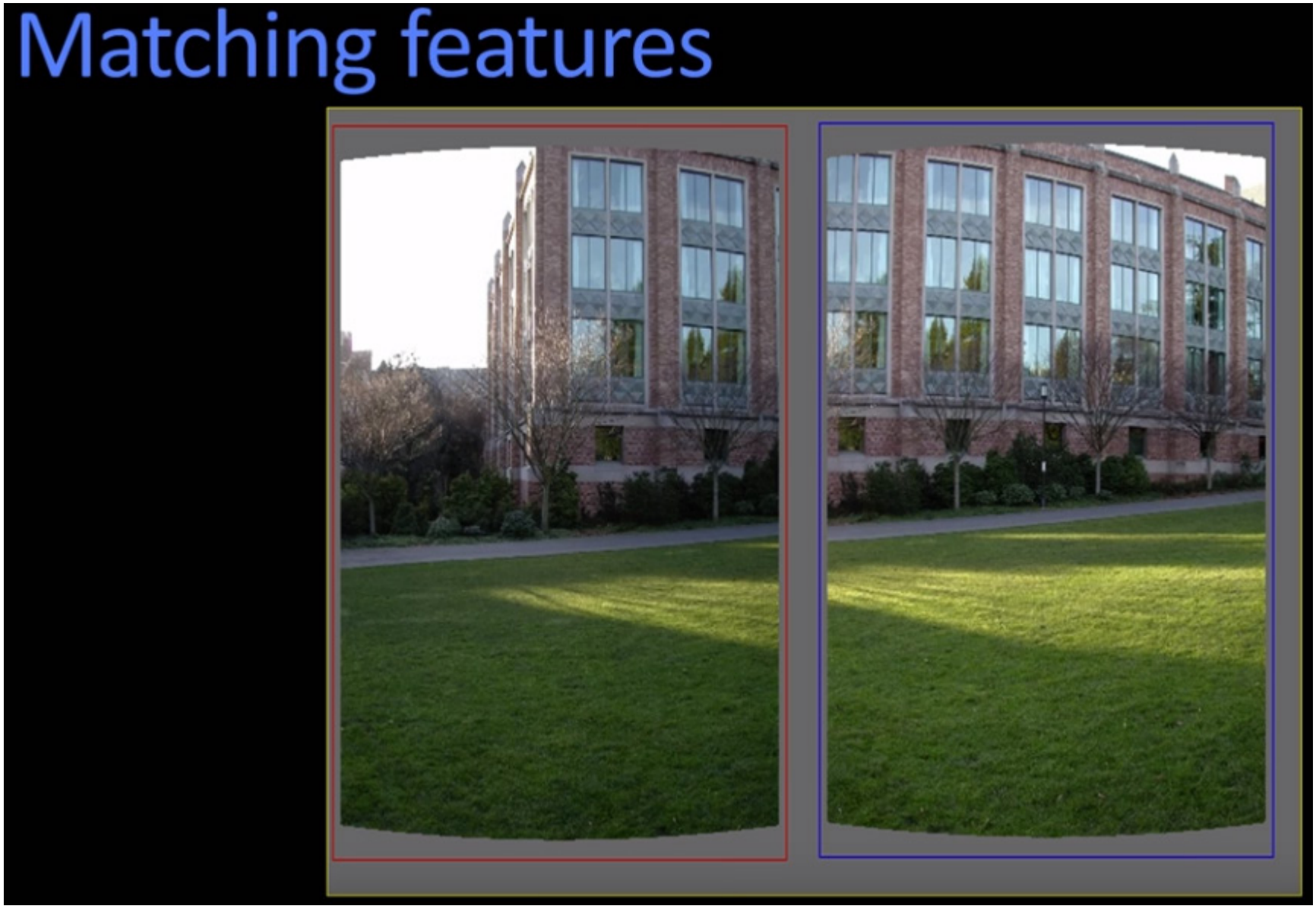
Image transformations are models. Minimal set of s of point pairs/matches:

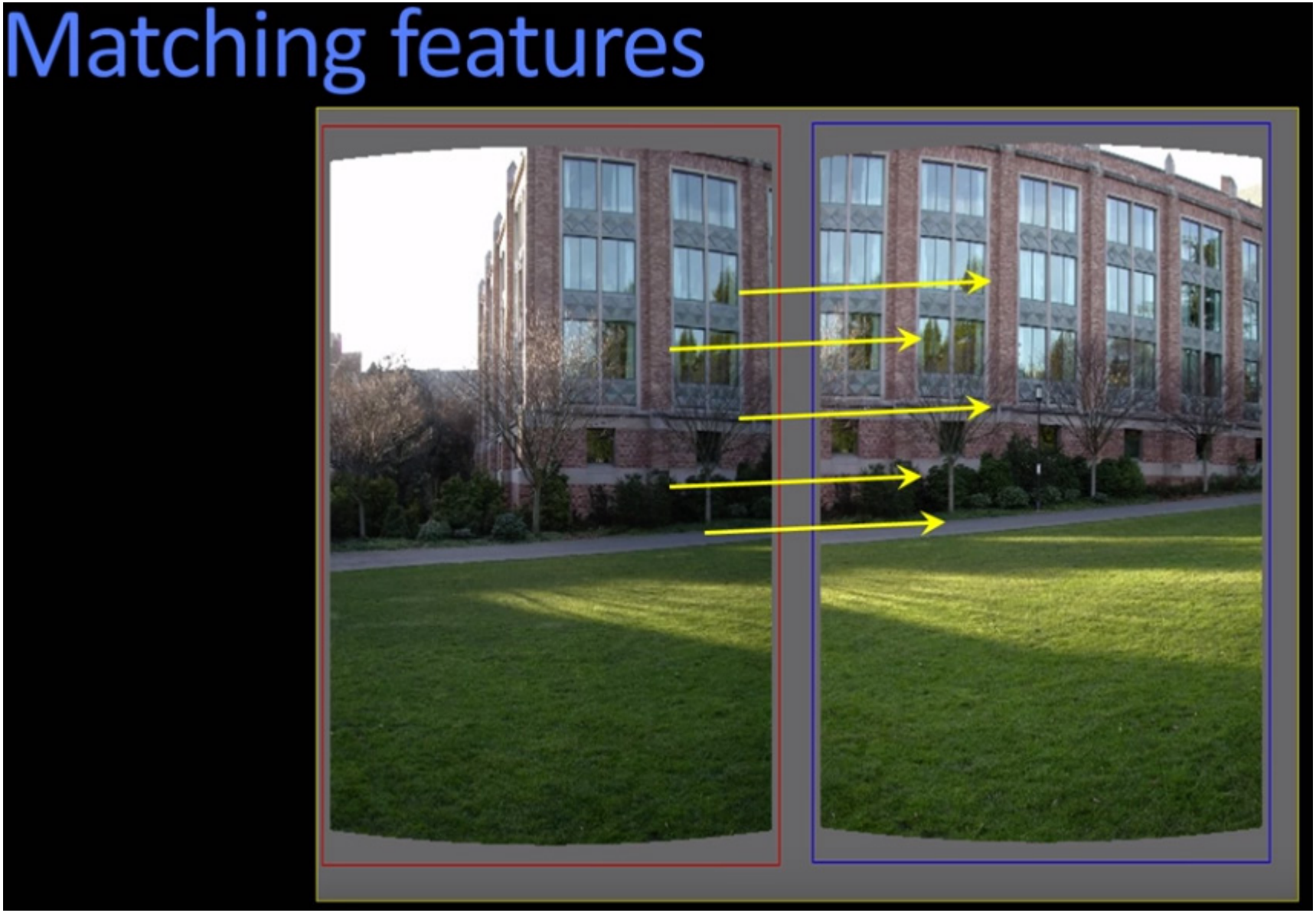
- Translation: pick one pair of matched points
- Homography (for plane) – pick 4 point pairs

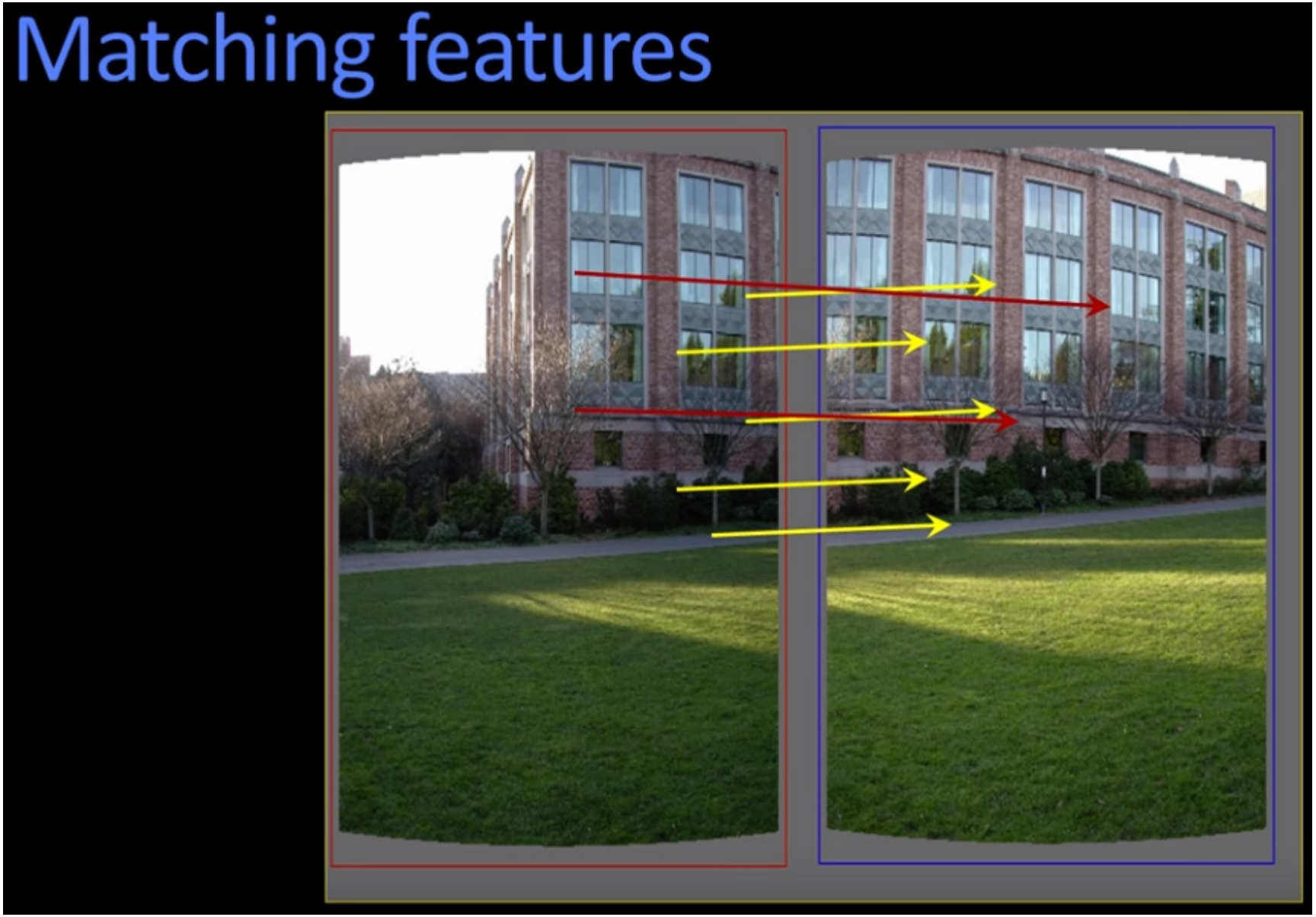
RANSAC for general model

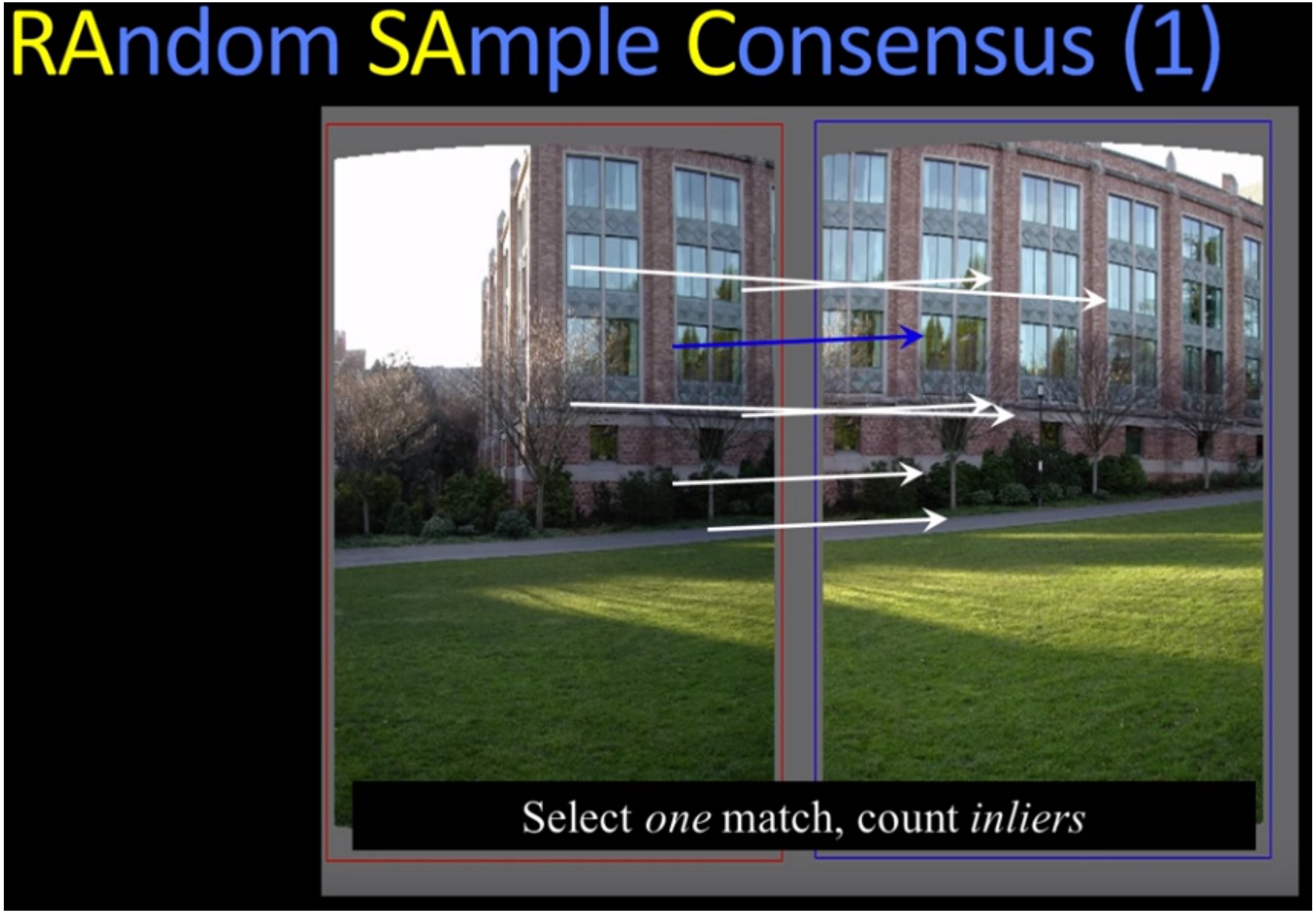
General RANSAC algorithm

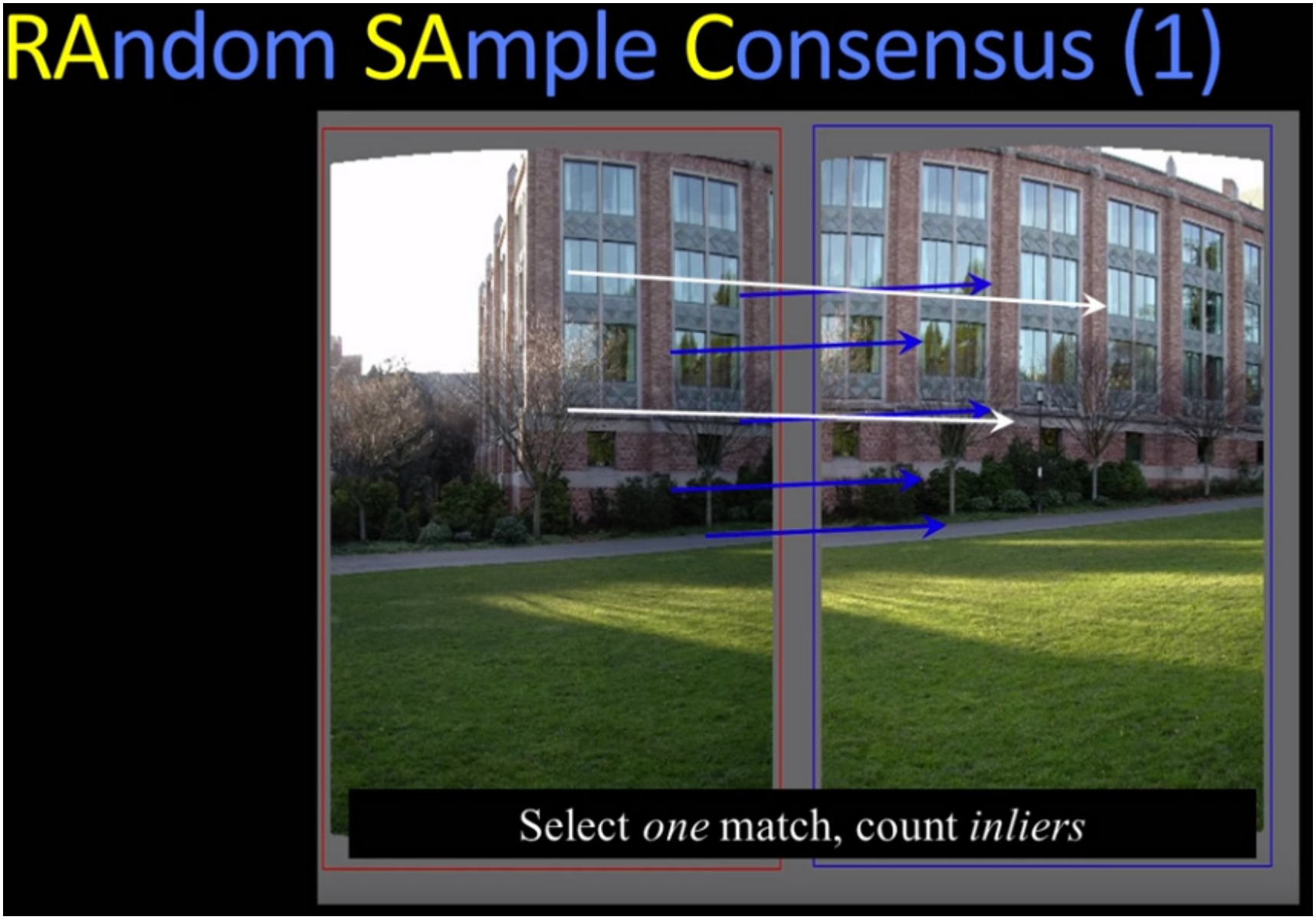
- Randomly select s points (or point pairs) to form a *sample*
- Instantiate the model
- Get consensus set C_i- the points within error bounds (distance threshold) of the model
- If $|C_i| > T$, terminate and return model
- Repeat for N trials, return model with max $|C_i|$

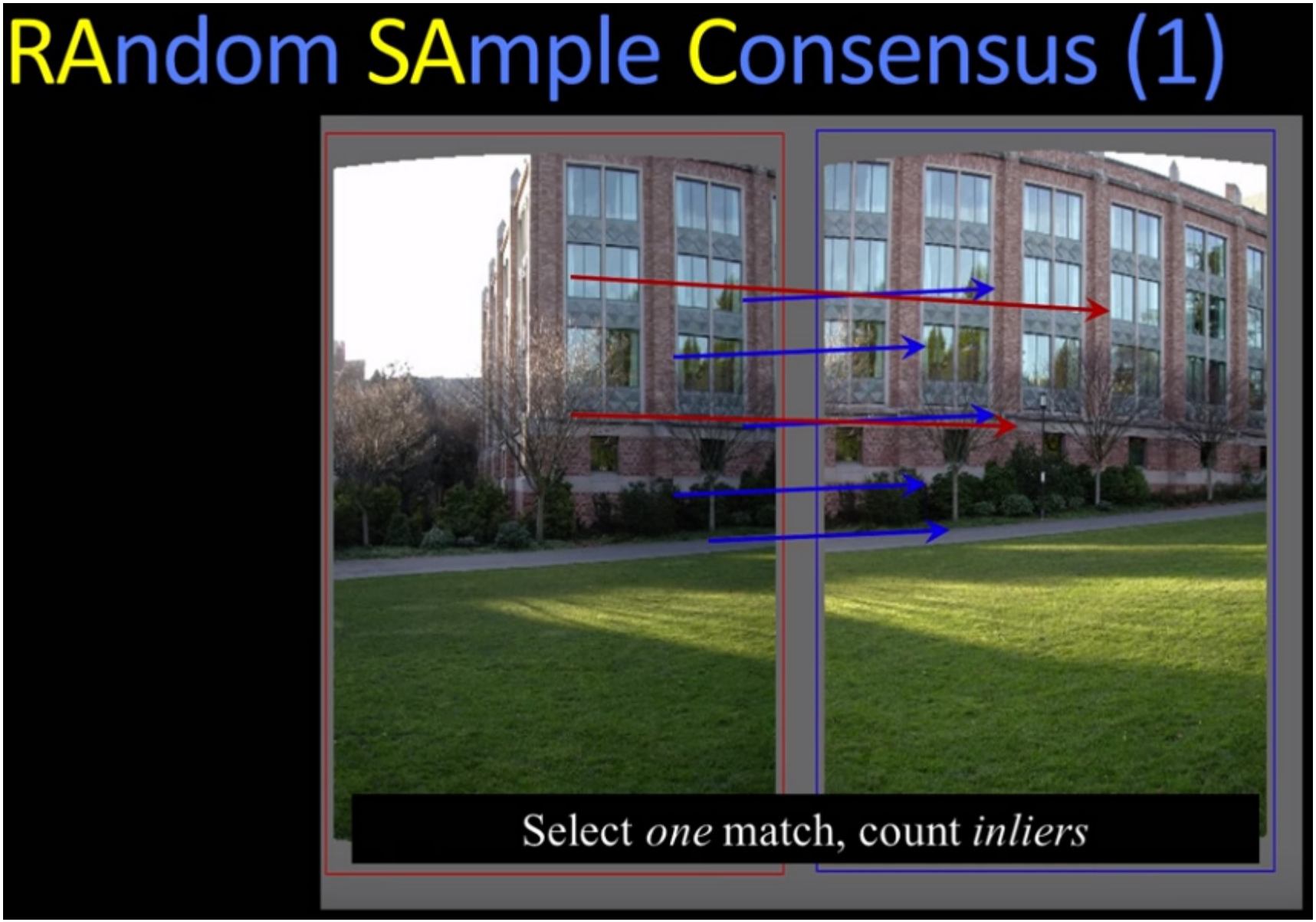

Matching features

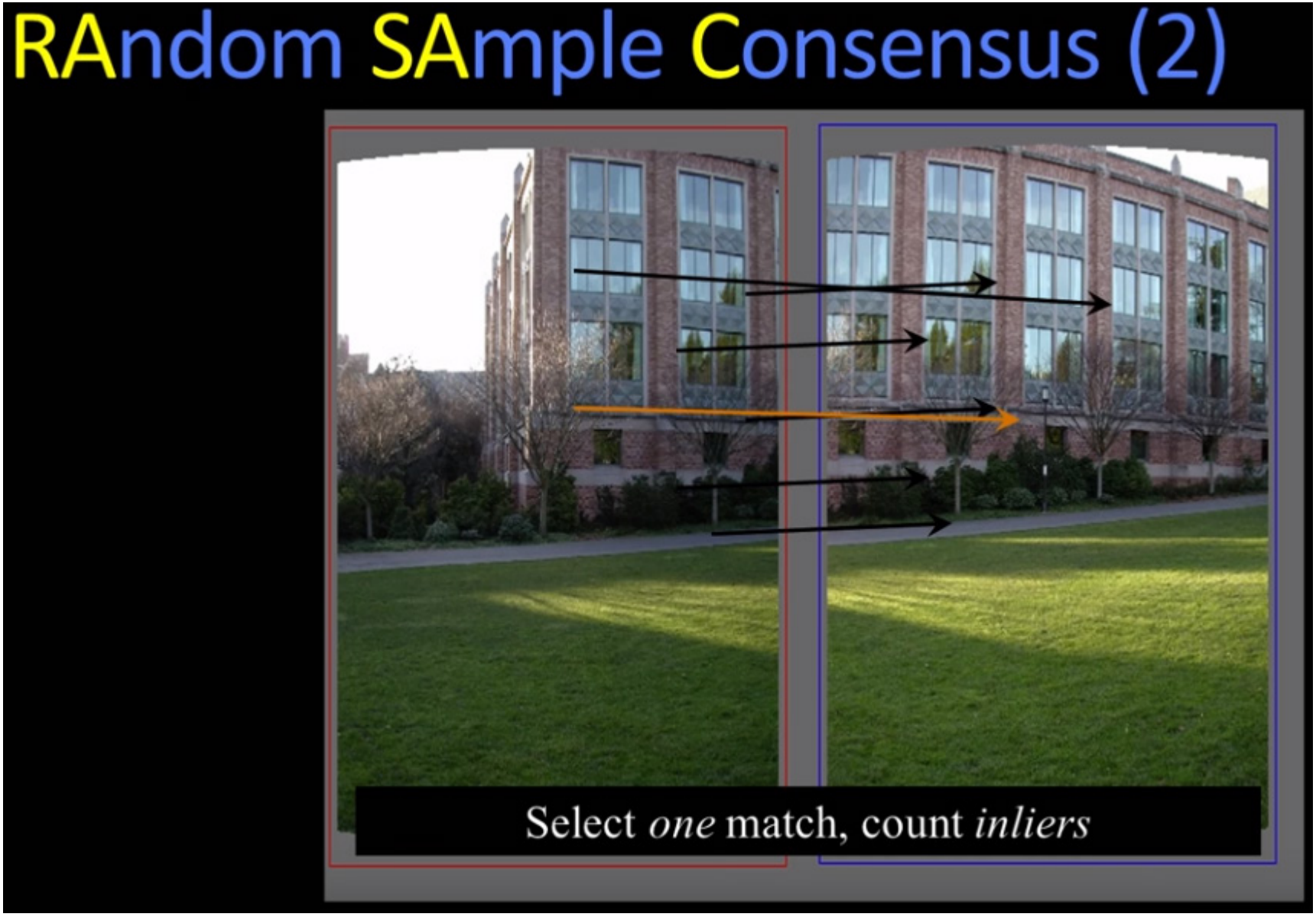

Matching features

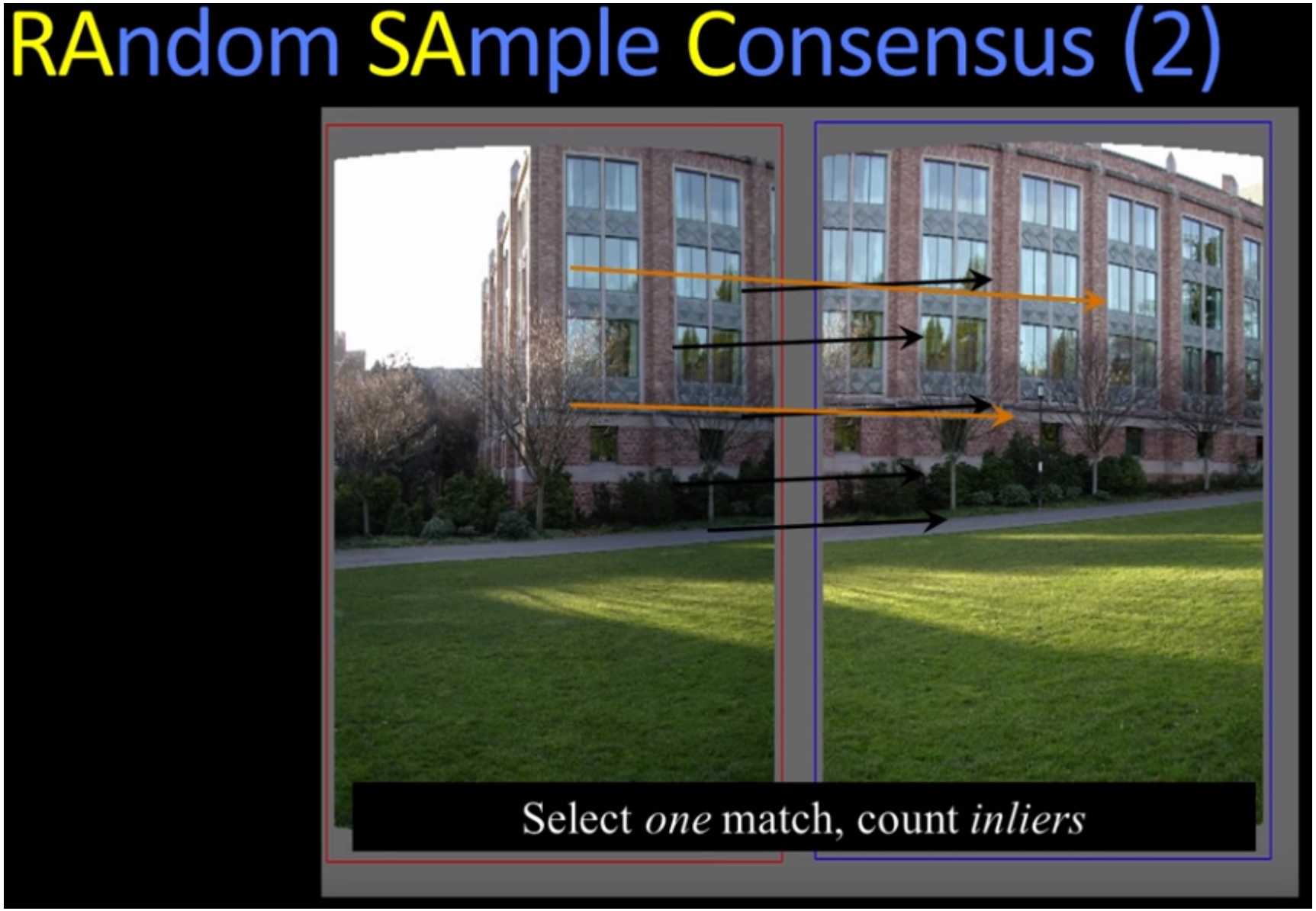

Matching features

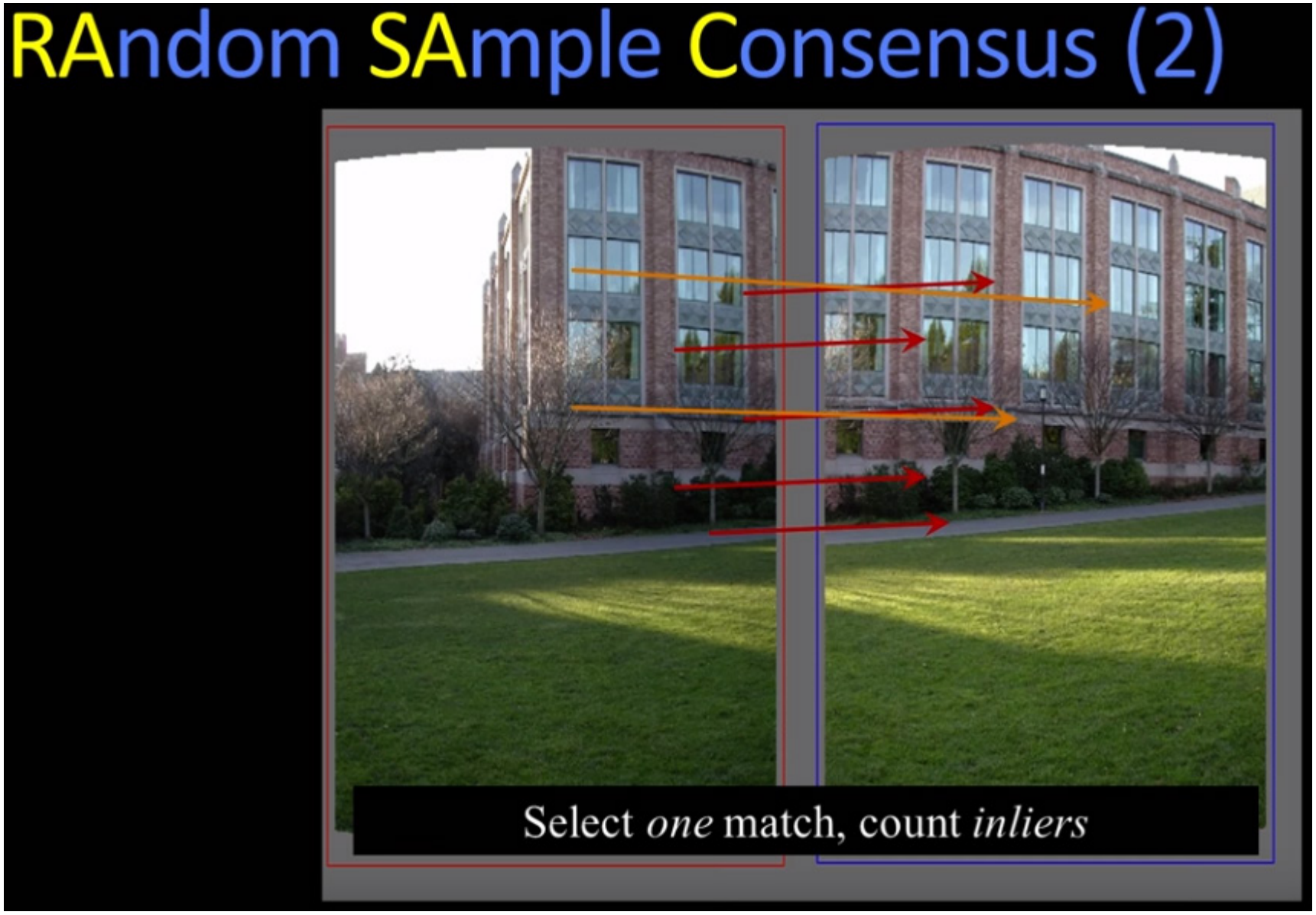

RAndom SAmple Consensus (1)
Select *one* match, count *inliers*


RAndom SAmple Consensus (1)
Select *one* match, count *inliers*

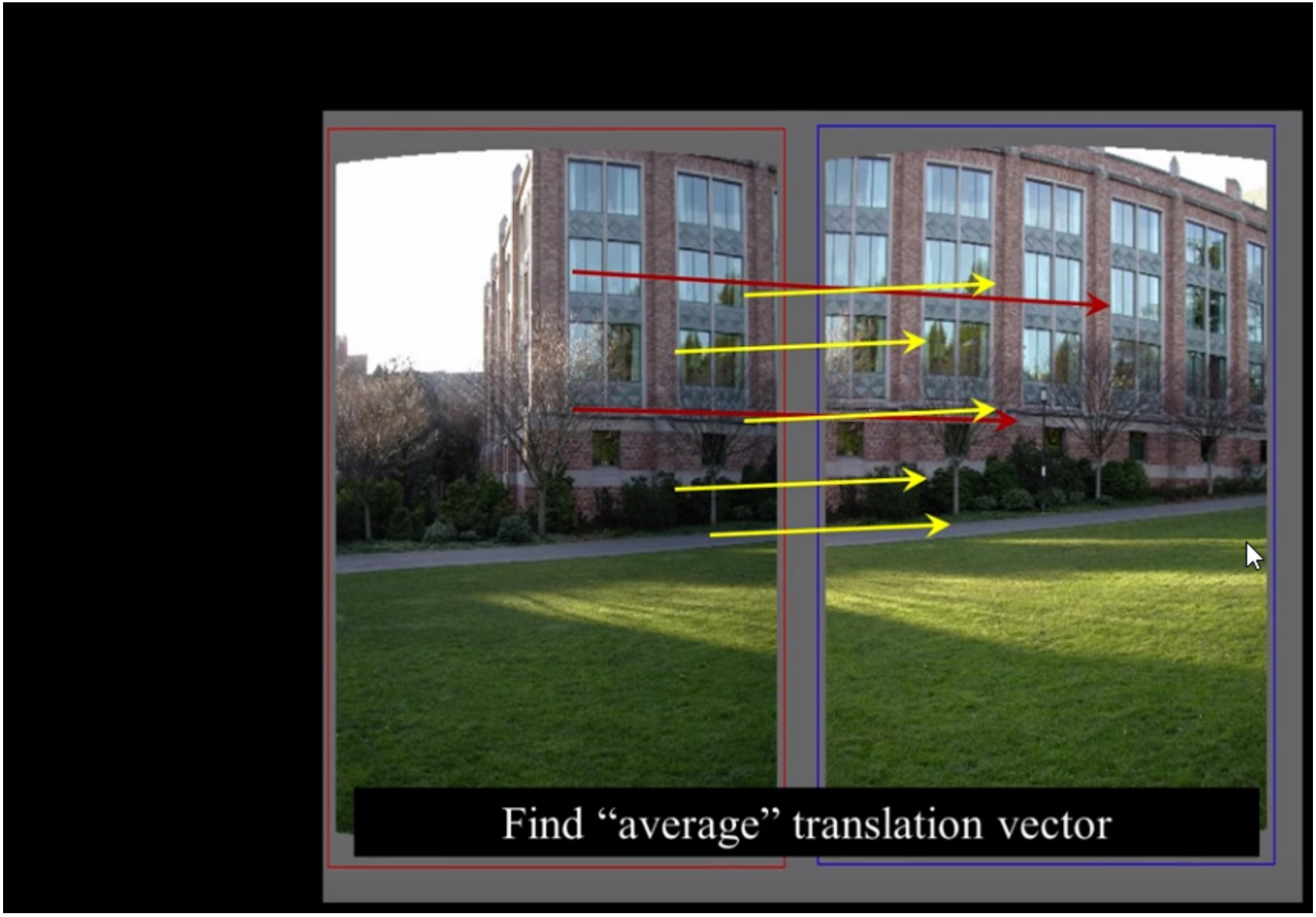

RAndom SAmple Consensus (1)
Select *one* match, count *inliers*


RAndom SAmple Consensus (2)
Select *one* match, count *inliers*

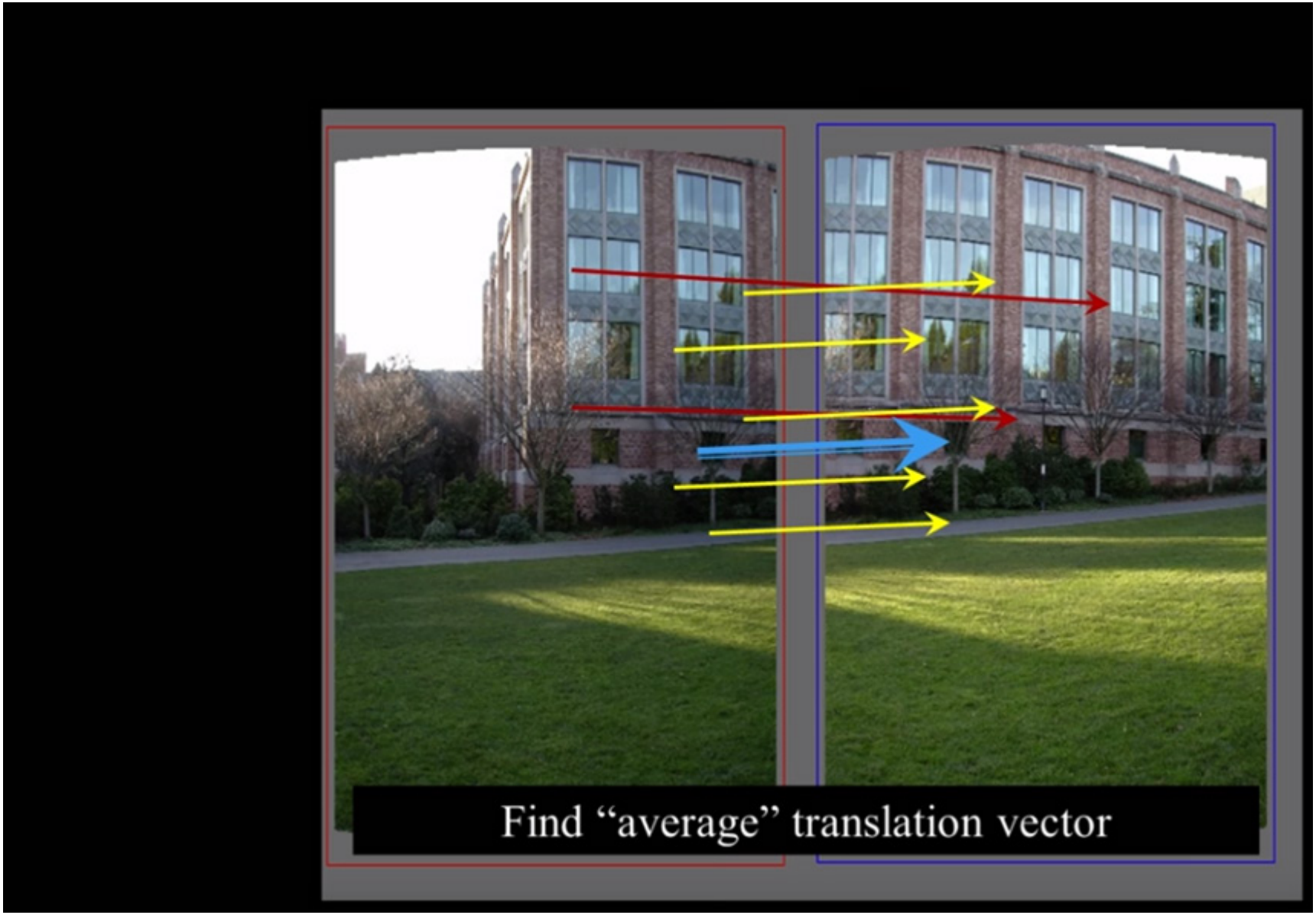

RAndom SAmple Consensus (2)
Select *one* match, count *inliers*


RAndom SAmple Consensus (2)
Select *one* match, count *inliers*

3.6 – RANSAC



3.6 – RANSAC



RANSAC for estimating homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography $\boldsymbol{H}$ (exact)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H}\, pi) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares $\boldsymbol{H}$ estimate on all of the inliers

Panorama in Practice

Open panorama.ipynb