3D Reconstruction with Computer Vision
Meeting 4: Local Features 1

CS 378 Fall 2014, UT Austin, Bryan Klingner, 9 September

Project 0 Recap

```
def flip_image(image, horizontal, vertical):
    """Returns a flipped copy of 'image'.

    Arguments:
      image: the image to be flipped.
      horizontal: If True, the image is flipped horizontally.
      vertical: If True, the image is flipped vertically.
    """
    flipped = numpy.copy(image)
    if horizontal:
        flipped = cv2.flip(flipped, 1)
    if vertical:
        flipped = cv2.flip(flipped, 0)
    return flipped
```

Project 0 Recap

```
def negate_image(image):
    """Returns a copy of 'image' where each pixel value is negated."""
    return numpy.invert(image)
```

Project 0 Recap

```
def swap_blue_and_green(image):
    """Returns a copy of RGB 'image' with blue and green channels swapped."""
    return image[:, :, [1, 0, 2]]
```

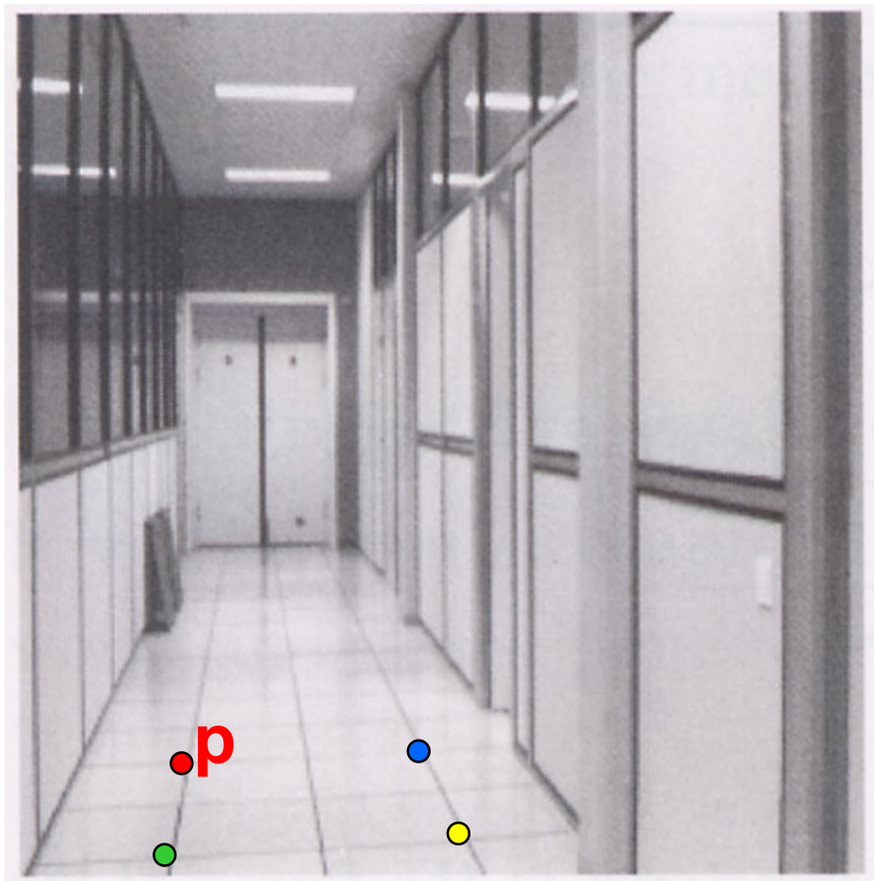
Check out numpy indexing and slicing tutorial!

Recap: How to stitch together a panorama (a.k.a. mosaic)?

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

Source: Steve Seitz

Image rectification



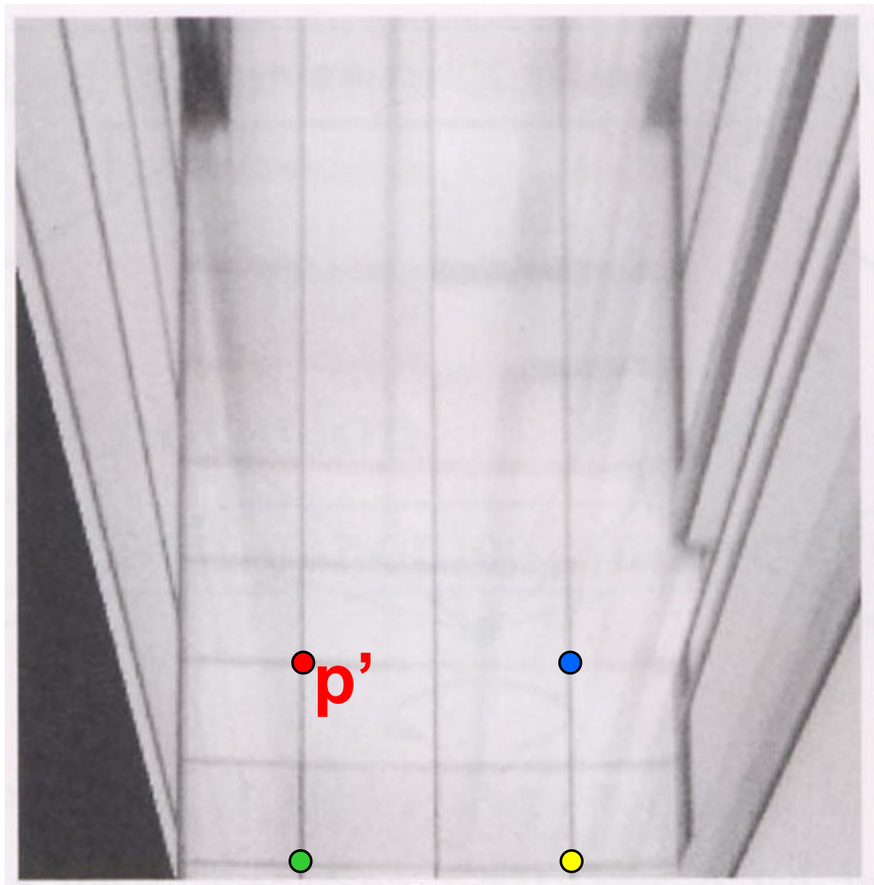
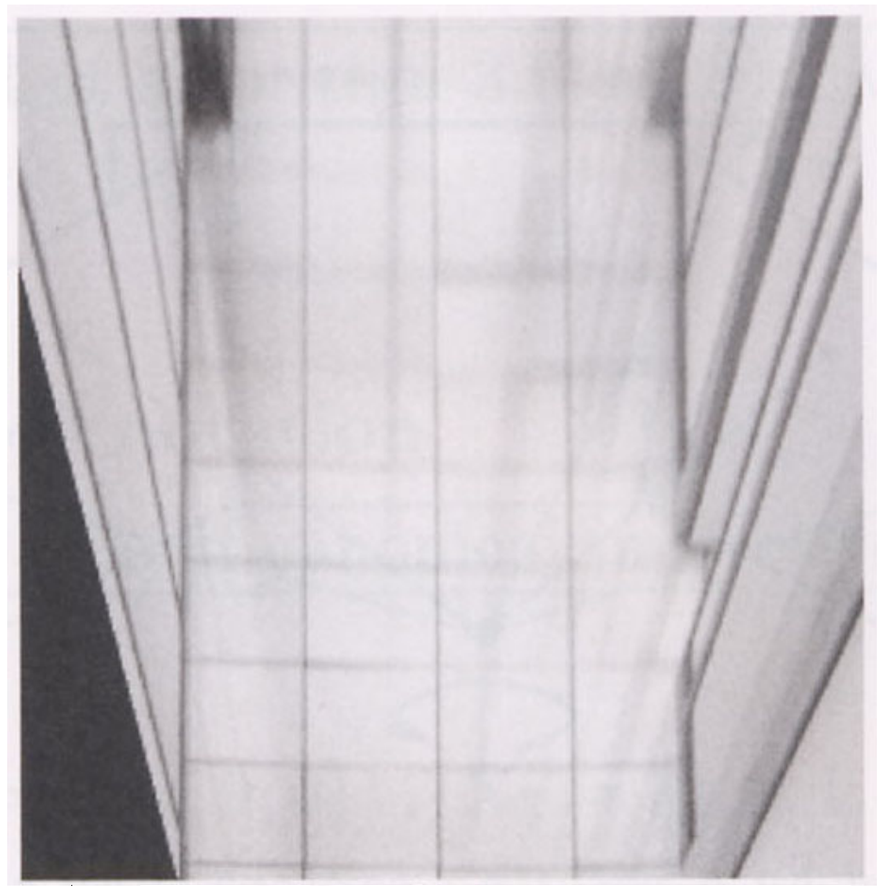
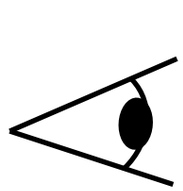
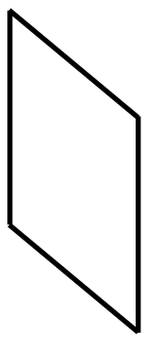


Image warping with homographies







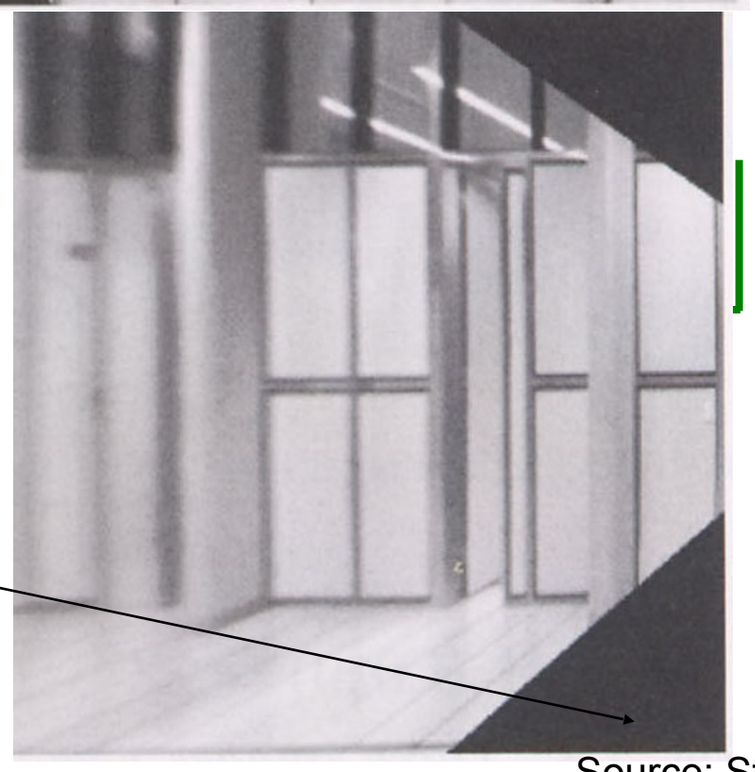


image plane in front

black area
where no pixel
maps to

Source: Steve Seitz

Analyzing patterns and shapes

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

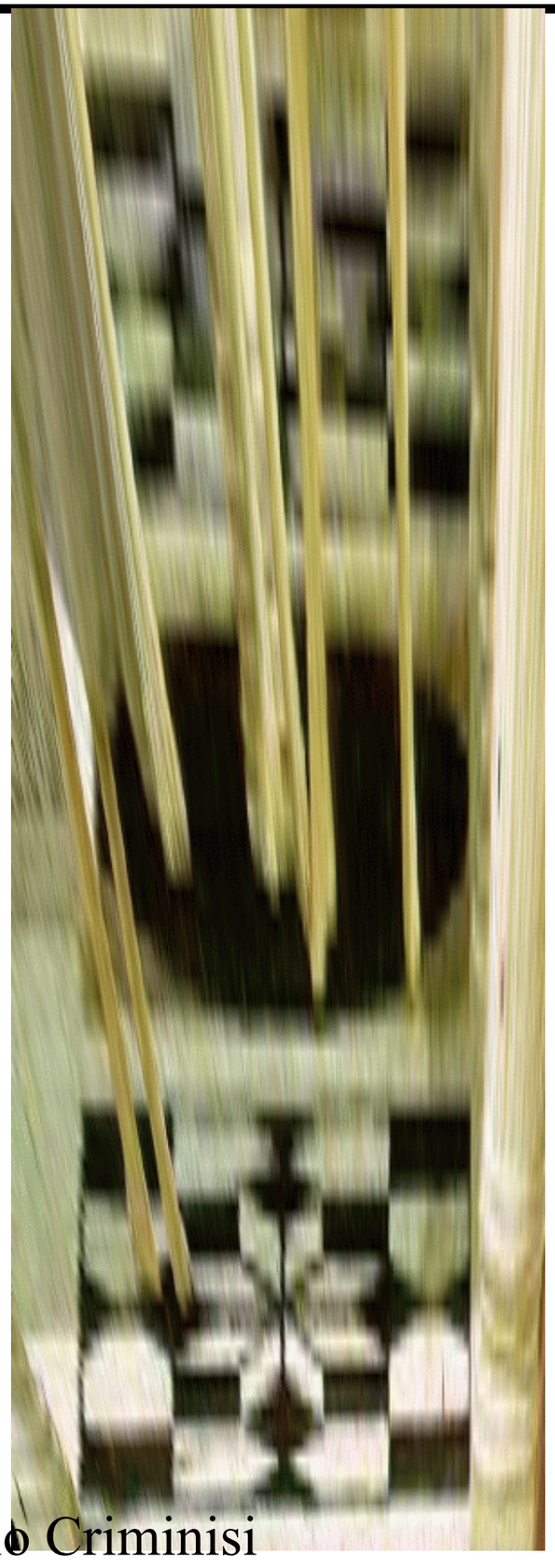
Homography

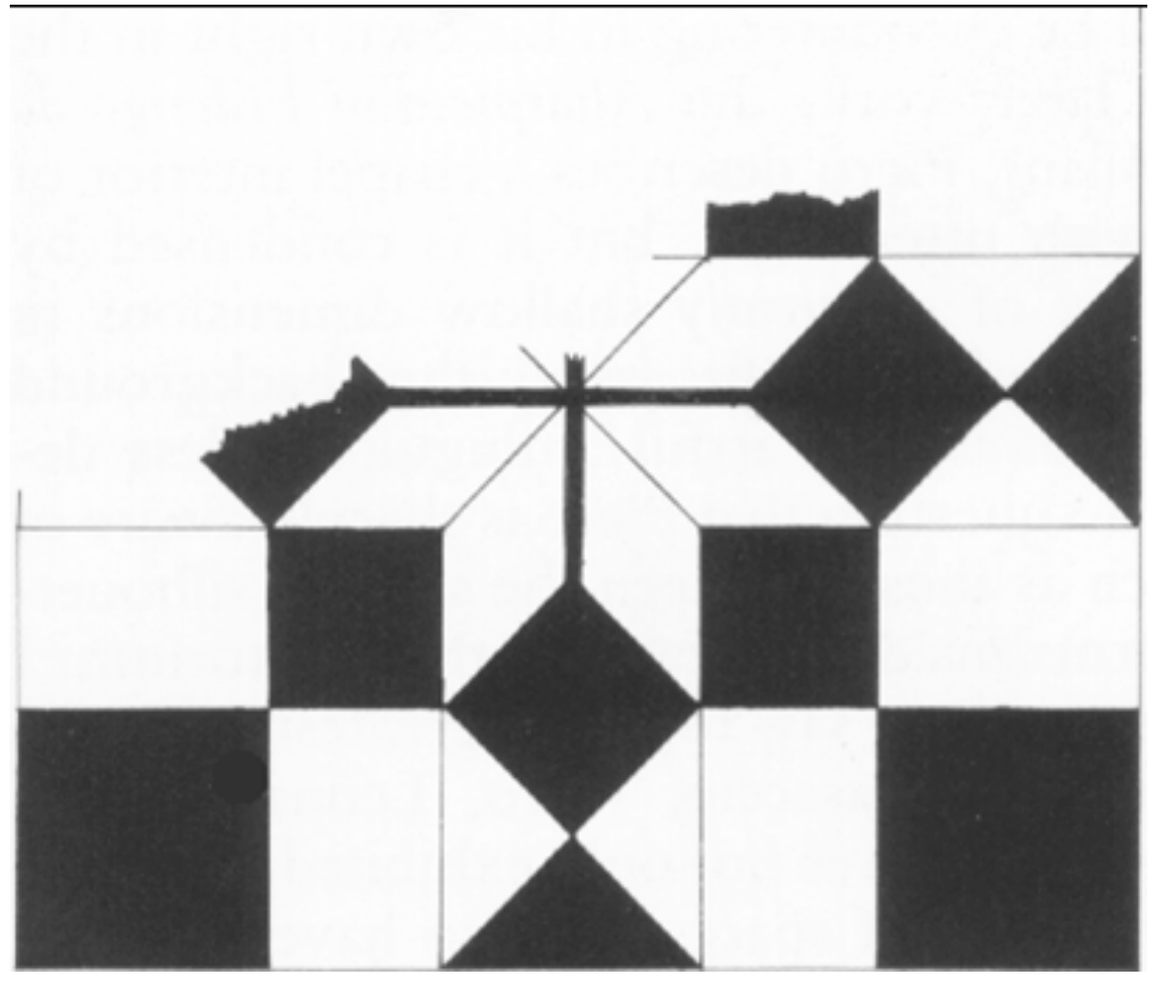
The floor (enlarged)

Automatically rectified floor

Slide from Antonio Criminisi

Analysing patterns and shapes





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

Slide from Antonio Criminisi

Analysing patterns and shapes

St. Lucy Altarpiece, D. Veneziano

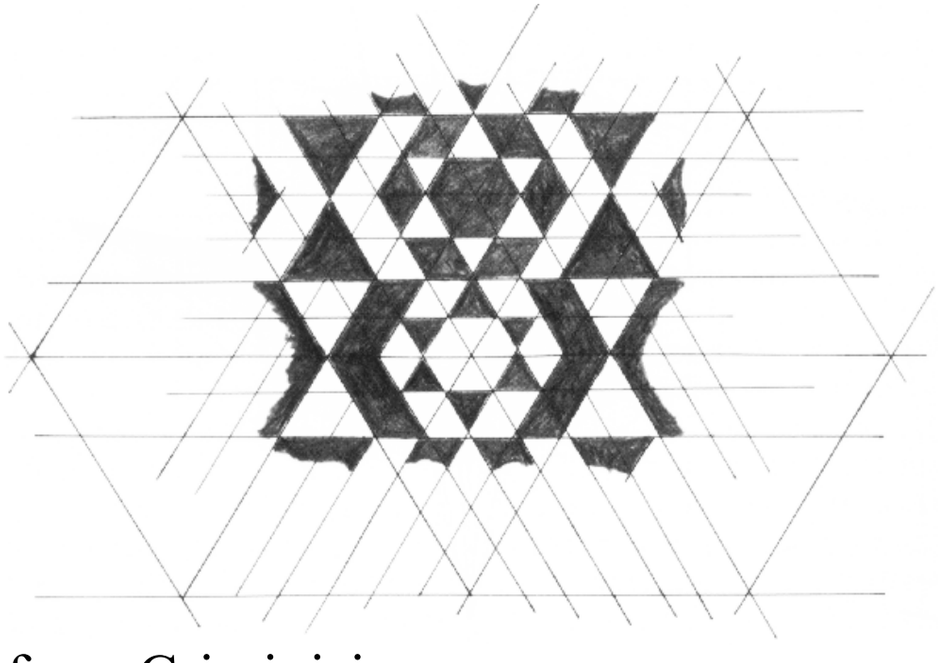
Slide from Criminisi

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

Automatically rectified floor

Analysing patterns and shapes

Automatic rectification

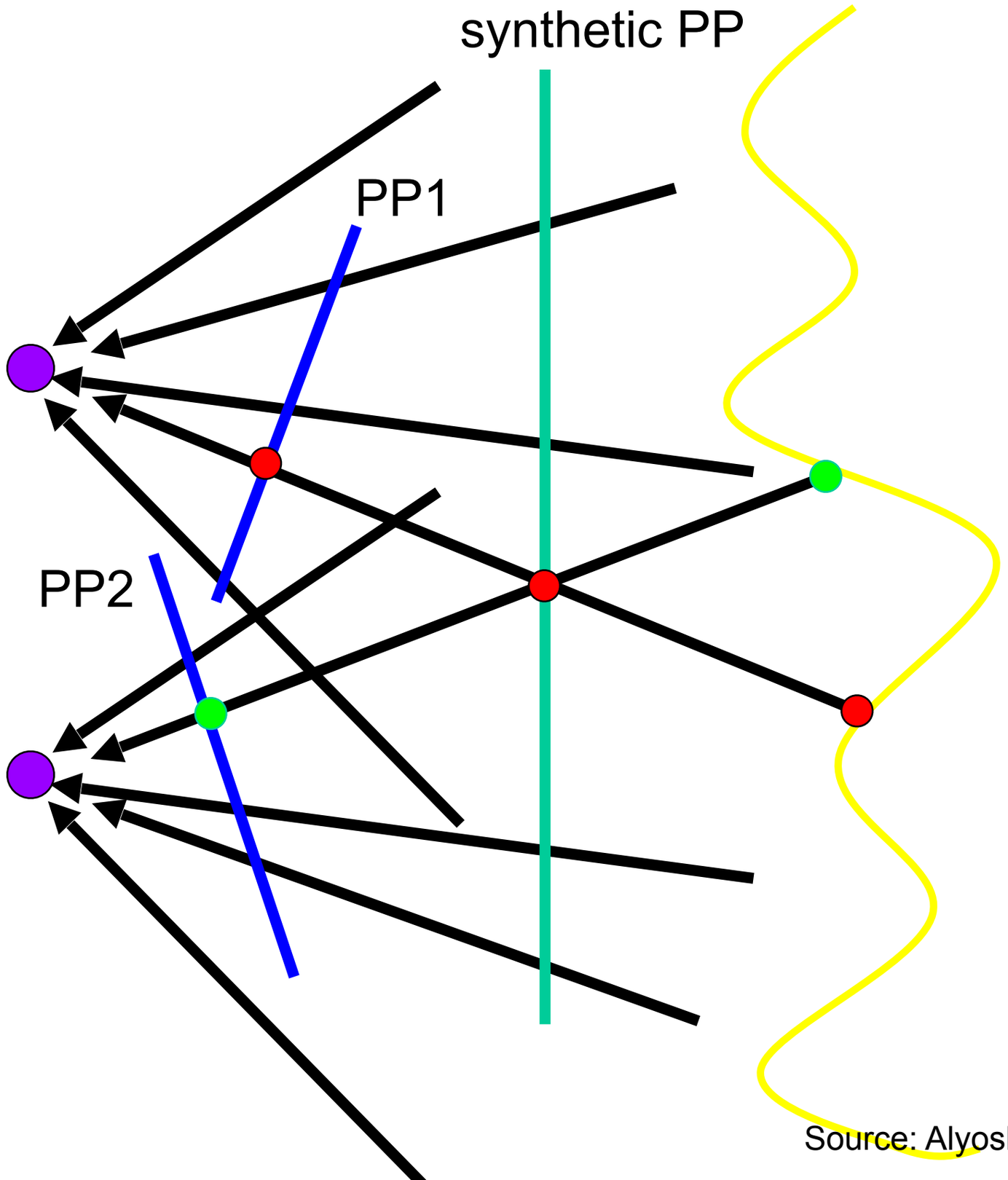


From Martin Kemp, *The Science of Art* *(manual reconstruction)*

Slide from Criminisi

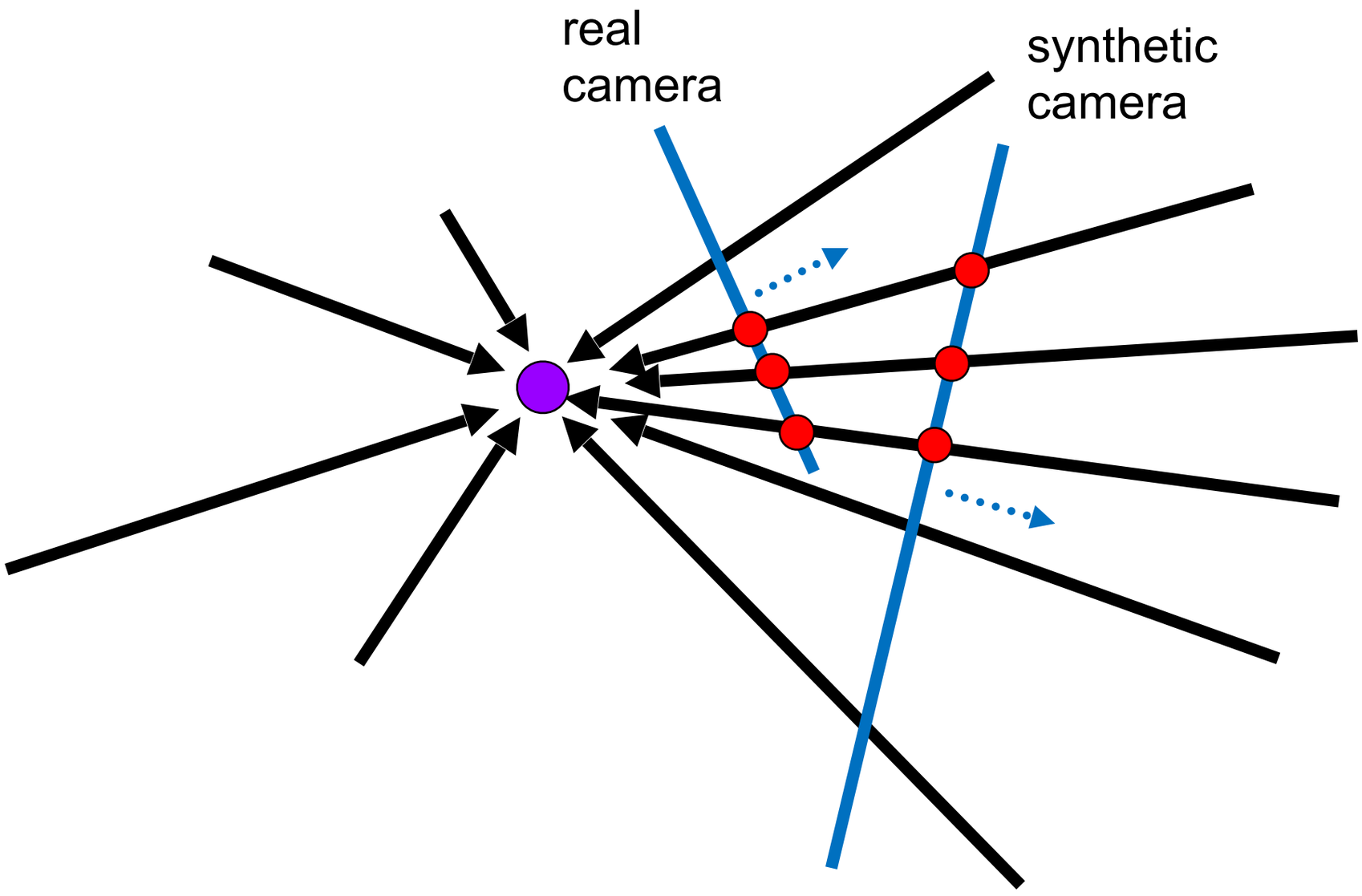
Changing camera center

Does it still work?



Source: Alyosha Efros

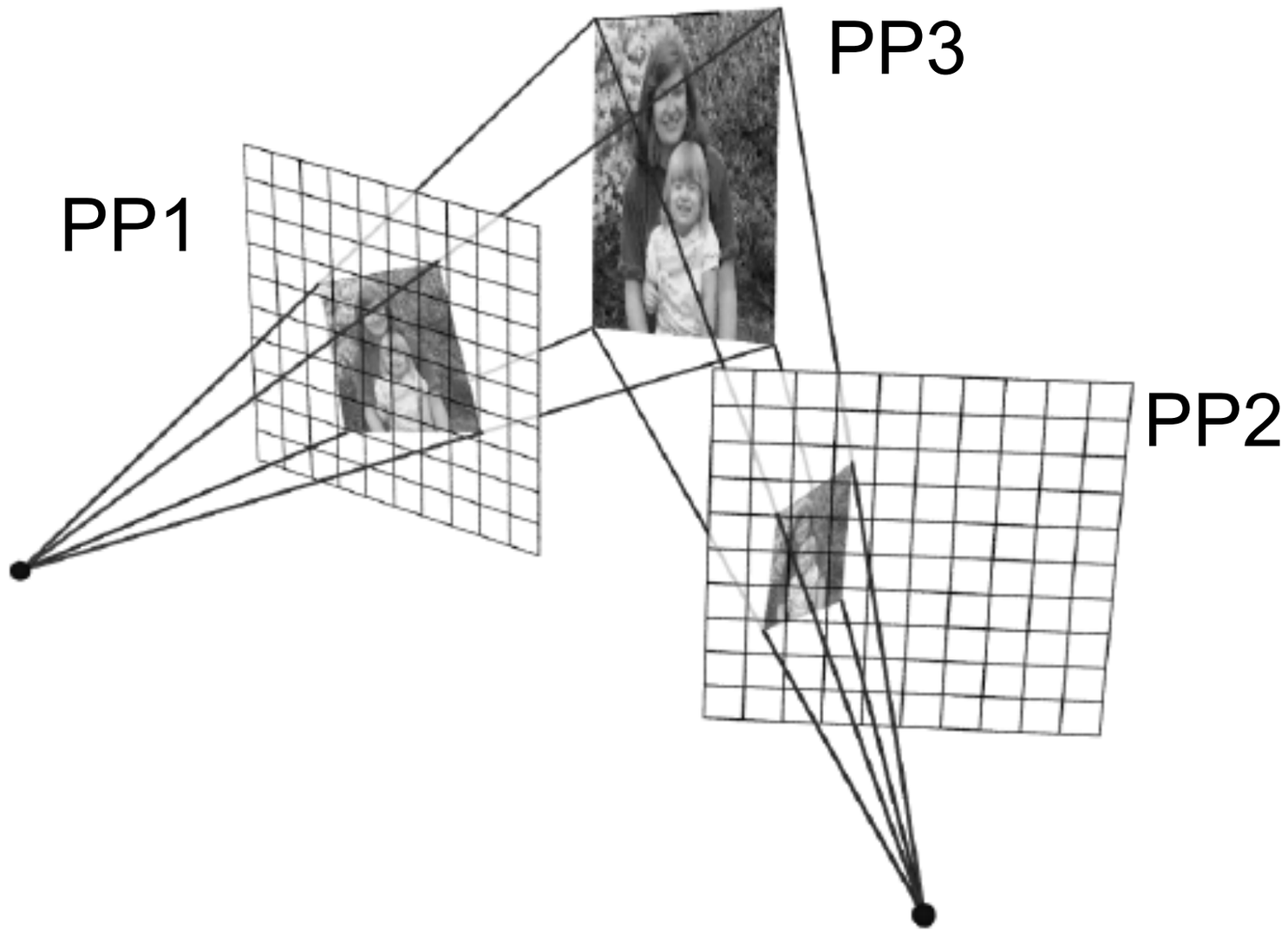
Recall: same camera center



Can generate synthetic camera view
as long as it has **the same center of projection**.

Source: Alyosha Efros

…Or: Planar scene (or far away)



PP3 is a projection plane of both centers of projection, so we are OK!

This is how big aerial photographs are made

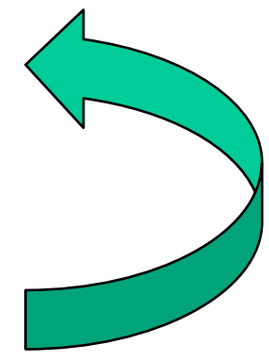
Source: Alyosha Efros


Map
Satellite
Hybrid
200 ft
100 m
©2007 Google - Terms of Use

RANSAC for estimating homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers*
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

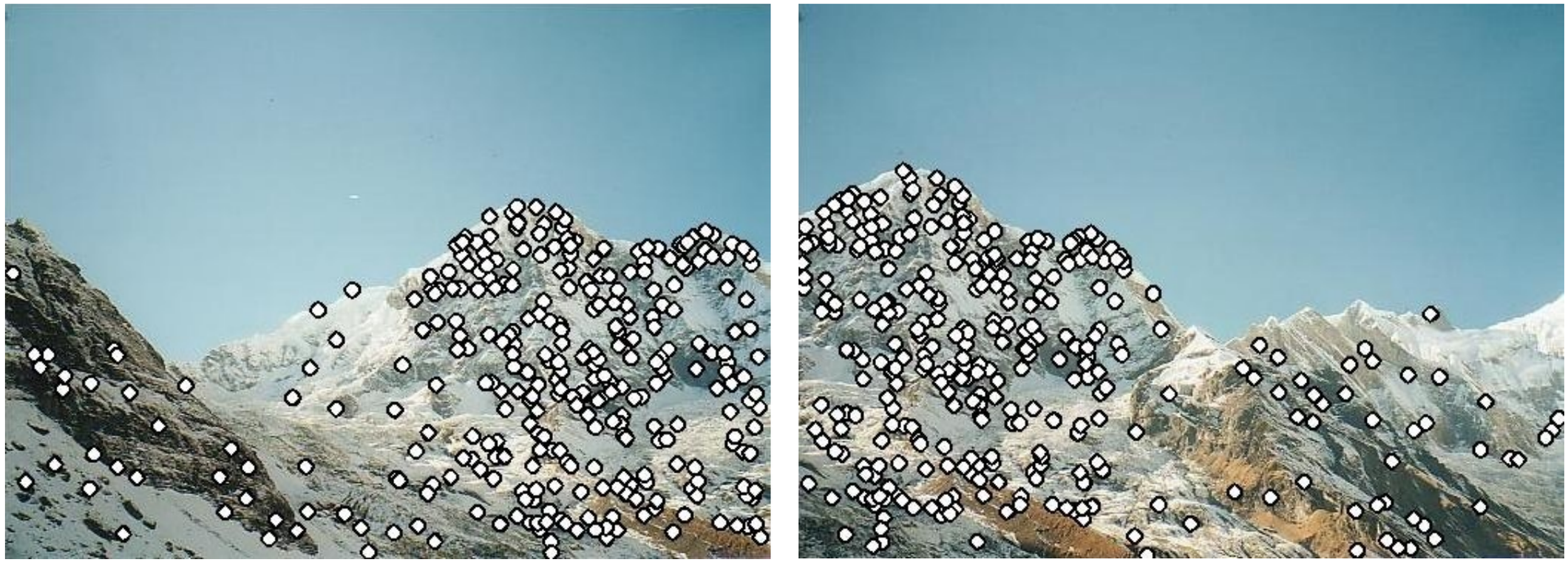


Slide credit: Steve Seitz

Robust feature-based alignment

Source: L. Lazebnik

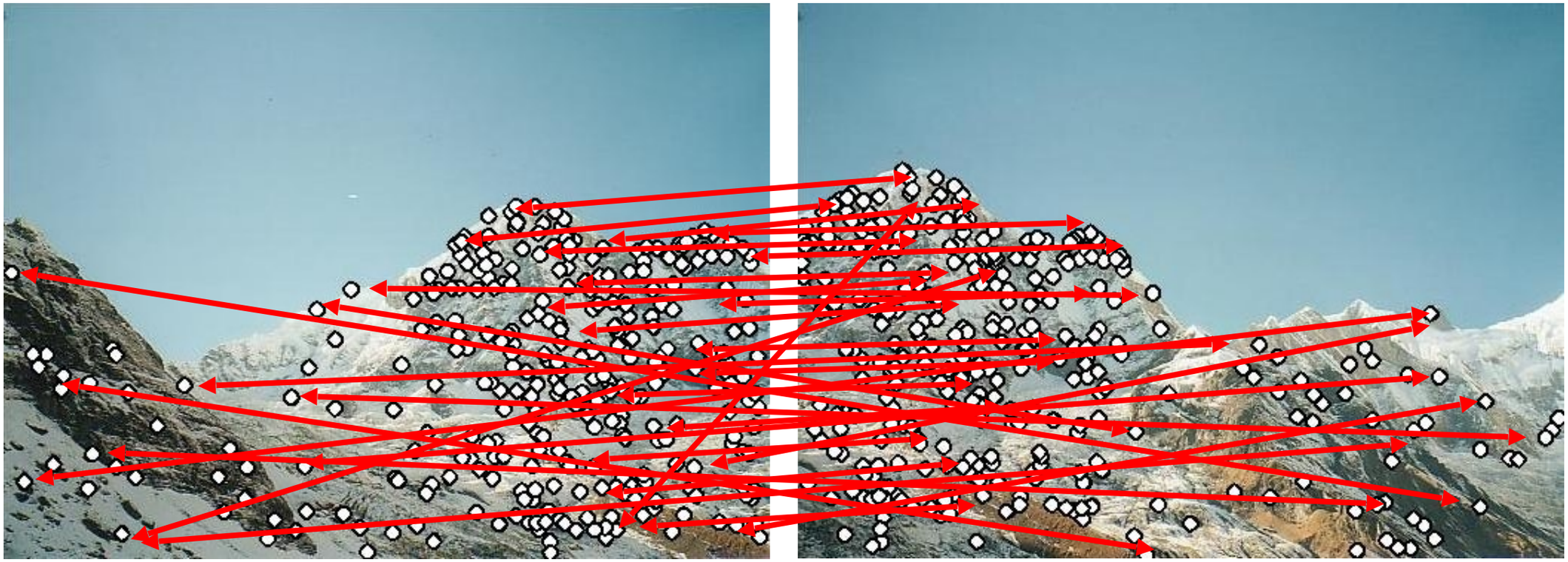
Robust feature-based alignment



- Extract features

Source: L. Lazebnik

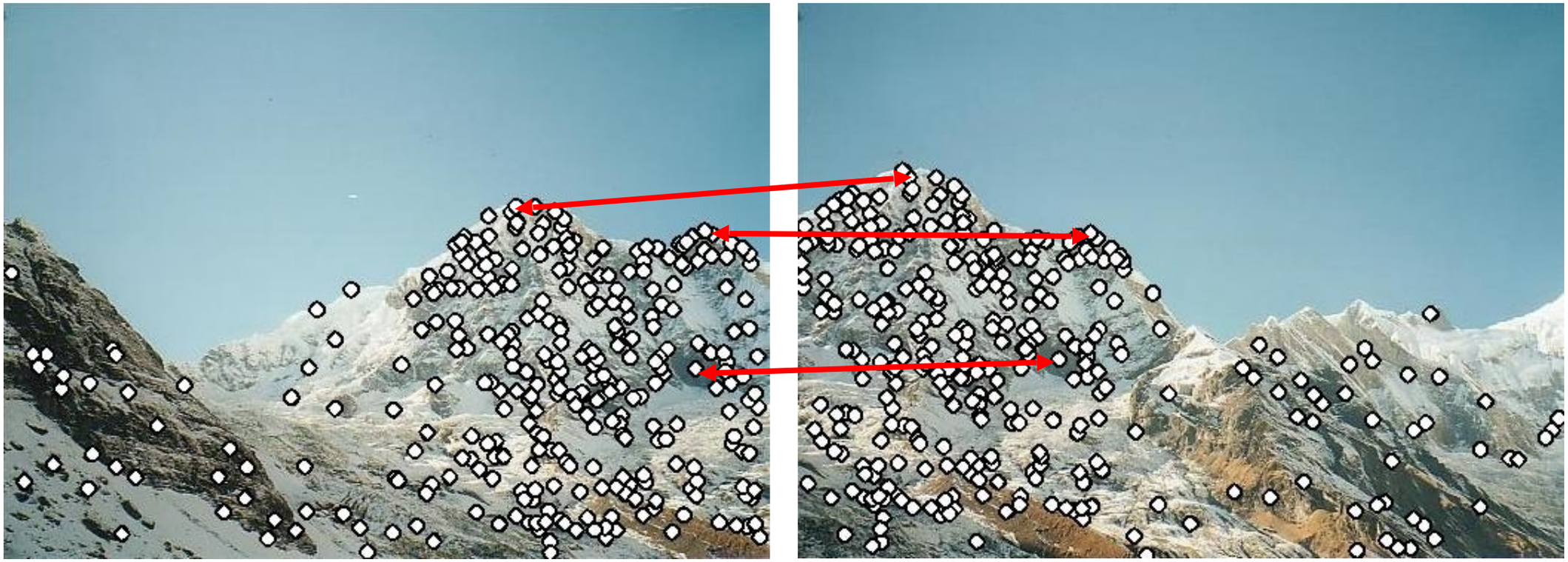
Robust feature-based alignment



- Extract features
- Compute *putative matches*

Source: L. Lazebnik

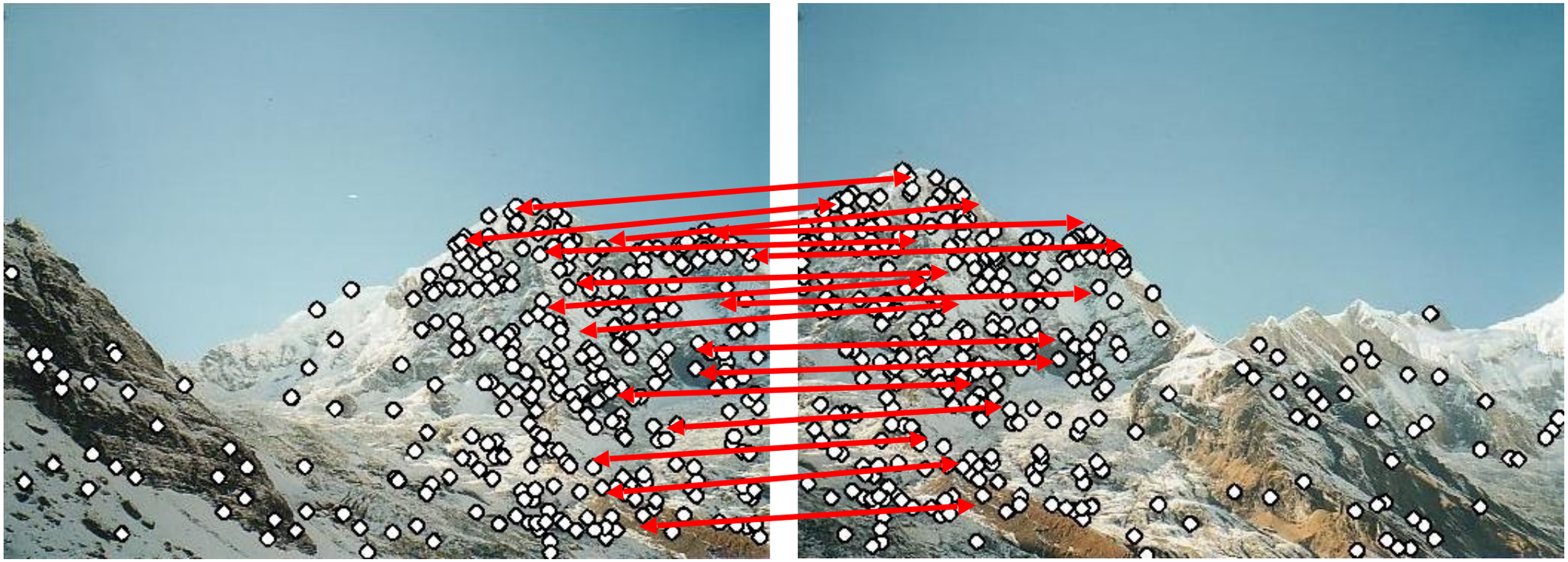
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

Source: L. Lazebnik

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

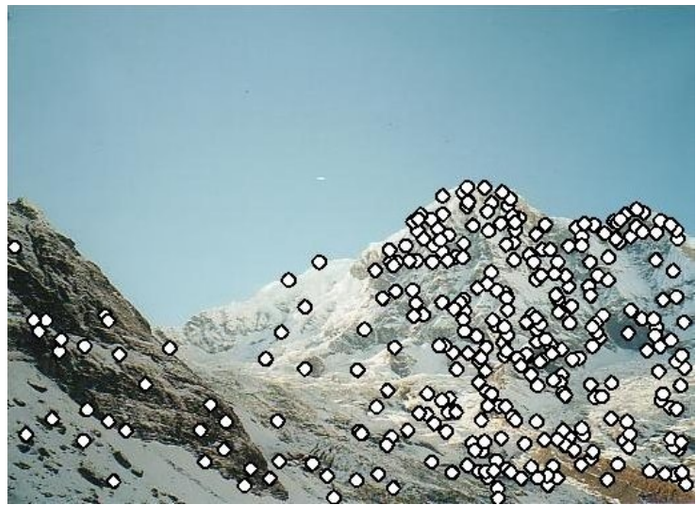
Source: L. Lazebnik

Summary: alignment & warping

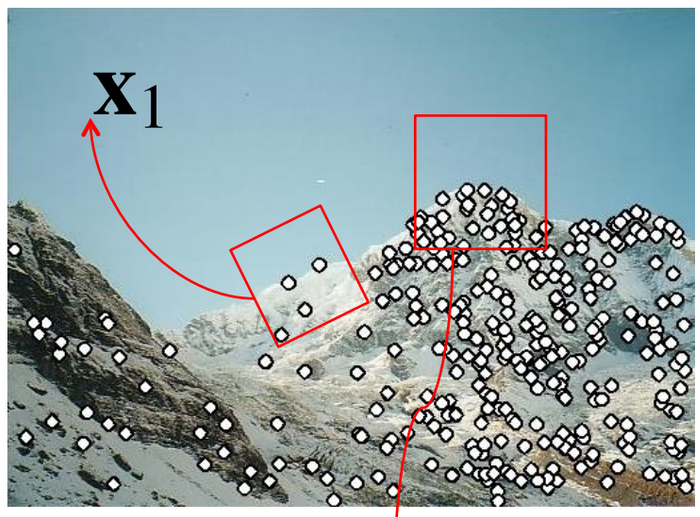
- Write **2d transformations** as matrix-vector multiplication (including translation when we use homogeneous coordinates)
- Perform **image warping** (forward, inverse)
- **Fitting transformations**: solve for unknown parameters given corresponding points from two views (affine, projective (homography)).
- **Mosaics**: uses homography and image warping to merge views taken from same center of projection.

Local features: main components

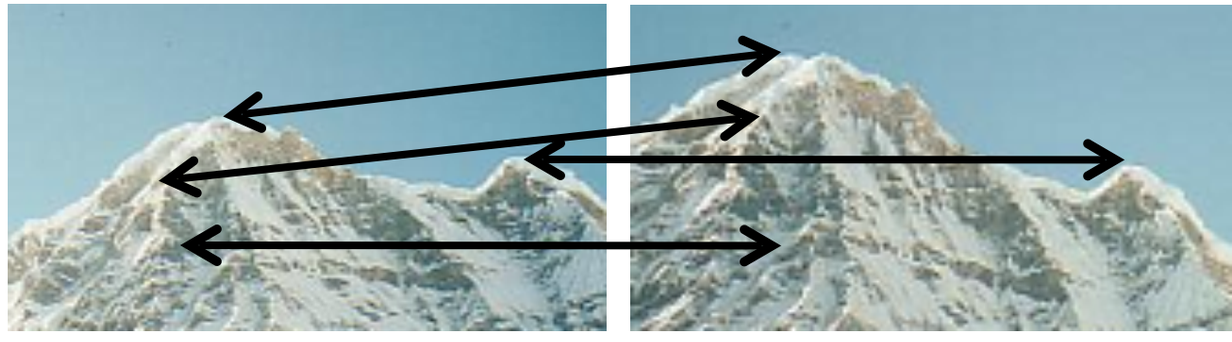
1) Detection: Identify the interest points



2) Description: Extract feature descriptor vector surrounding each interest point.



3) Matching: Determine correspondence between descriptors in two views



Local features: desired properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

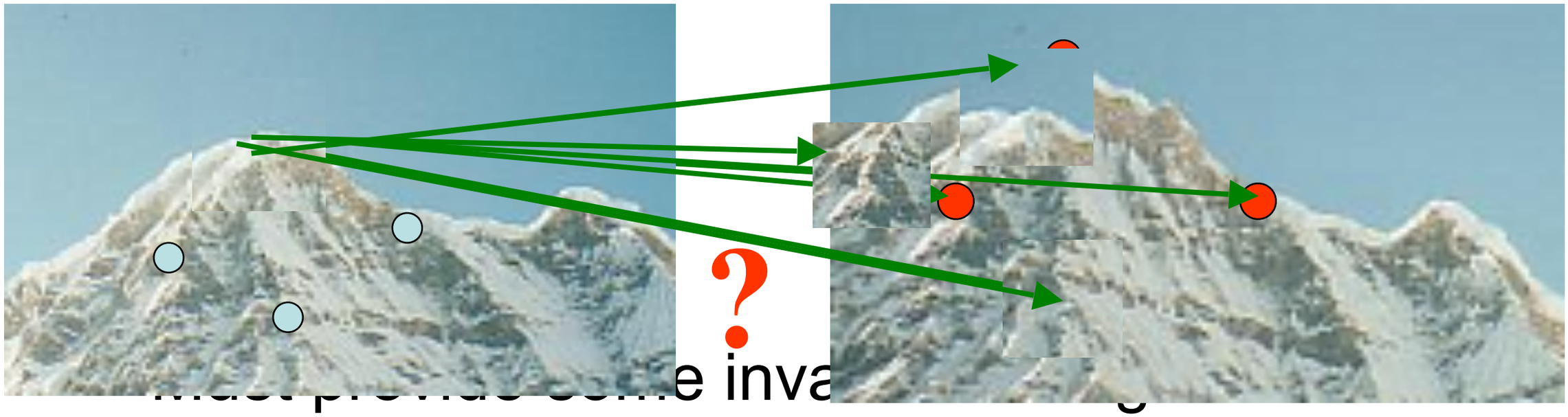
Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.
- Yet we have to be able to run the detection procedure *independently* per image.

No chance to find true matches!

Goal: descriptor distinctiveness

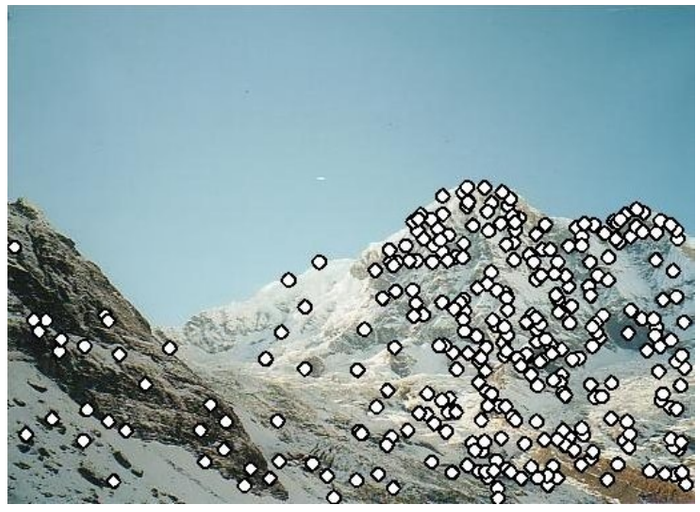
- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components

1) Detection: Identify the interest points



2) Description:Extract vector feature descriptor surrounding each interest point.

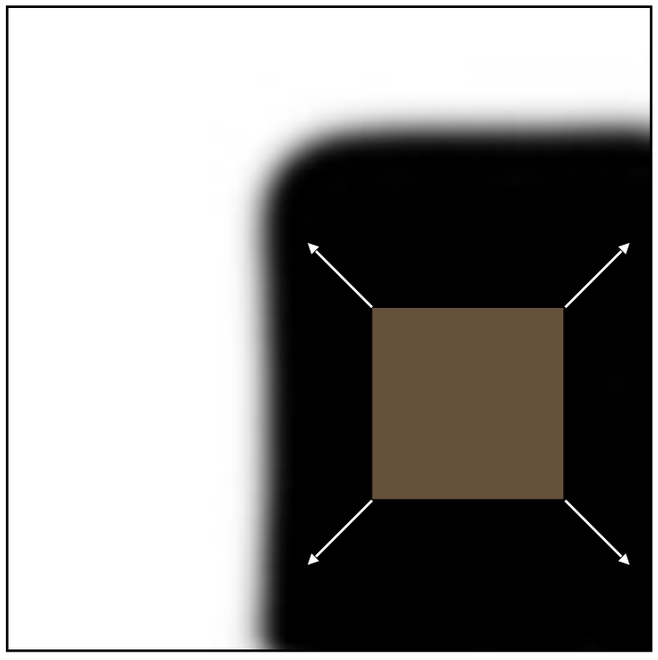
3) Matching: Determine correspondence between descriptors in two views

- What points would you choose?

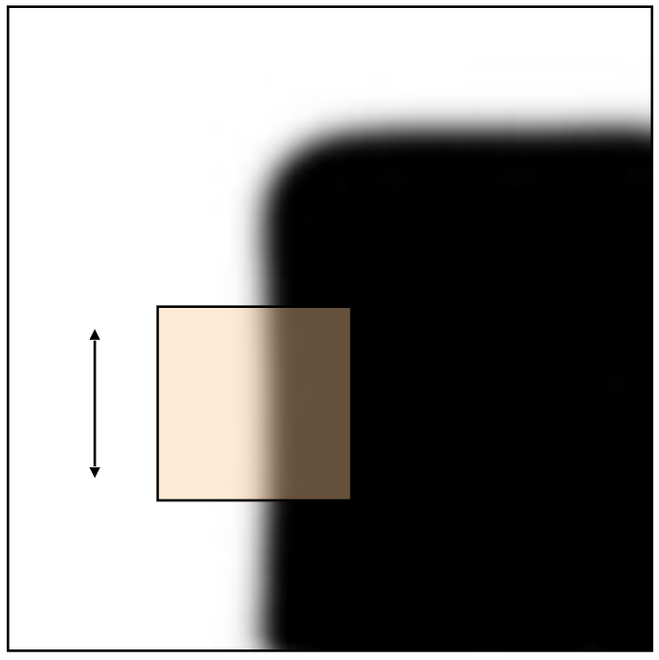
Corners as distinctive interest points

We should easily recognize the point by looking through a small window

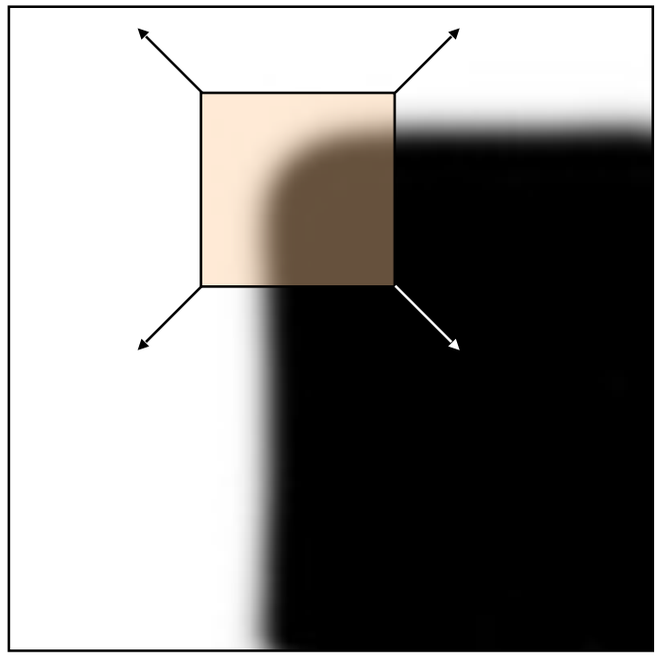
Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction



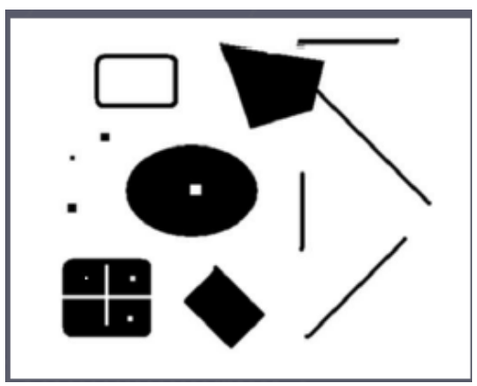
"corner": significant change in all directions

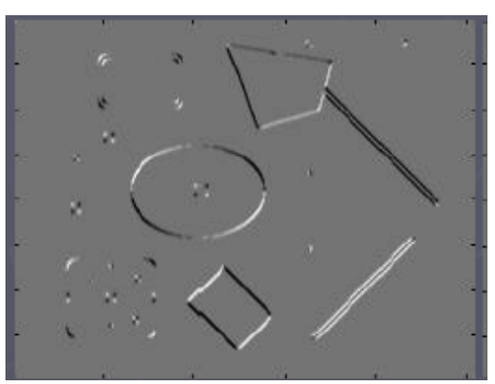
Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Corners as distinctive interest points

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

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



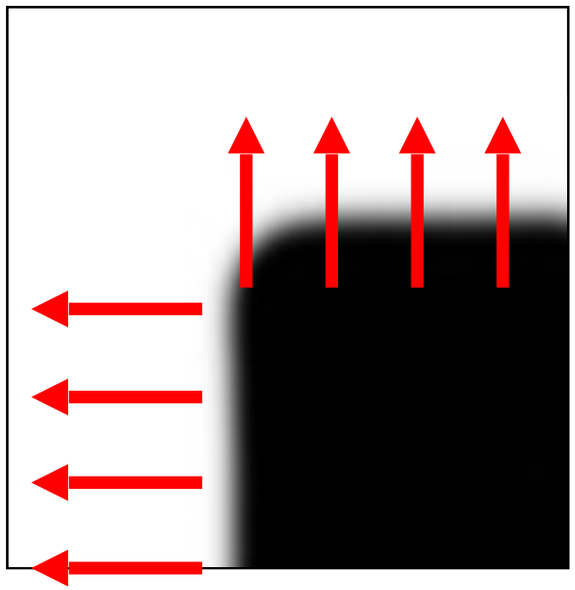


Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

Look for locations where **both** λ's are large.

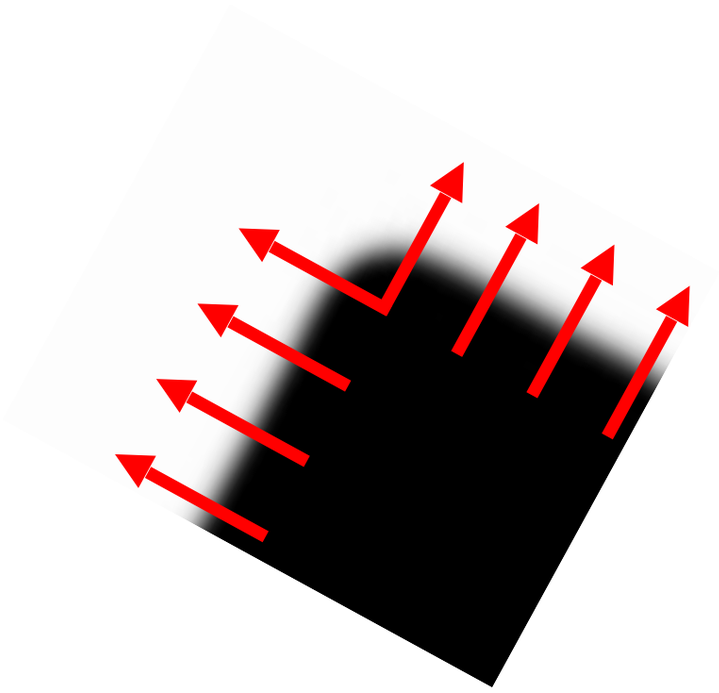
If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

What does this matrix reveal?

Since *M* is symmetric, we have

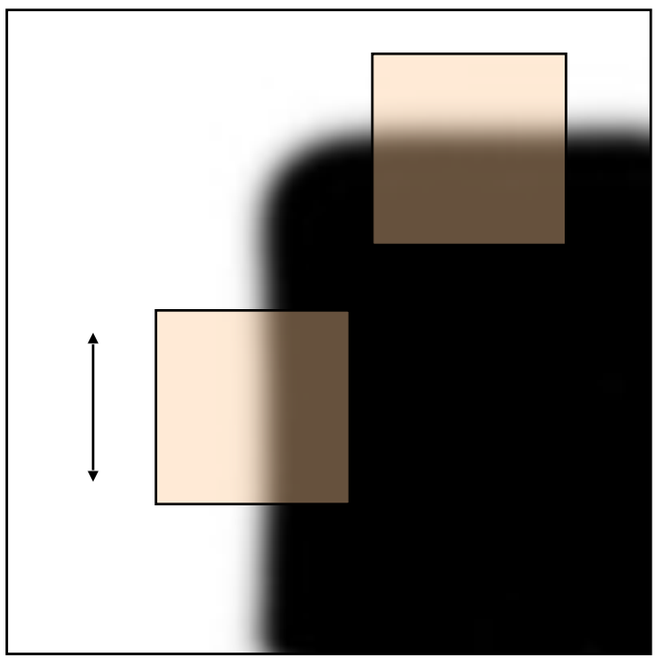
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

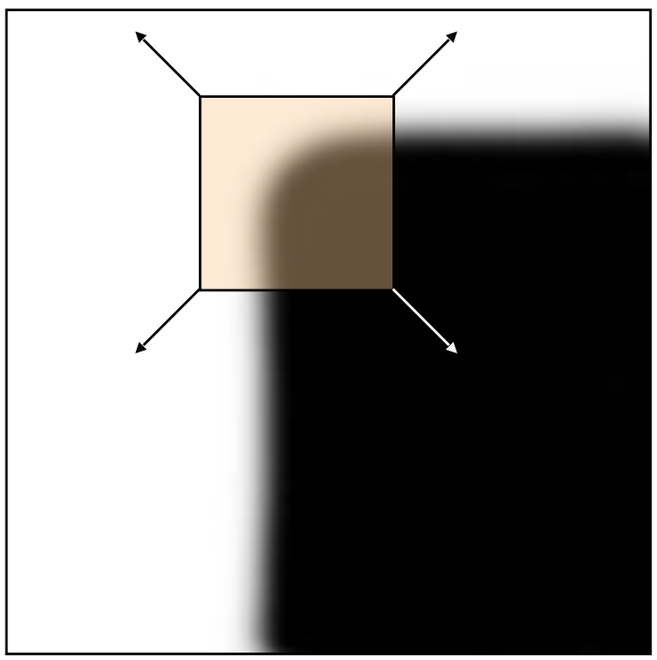


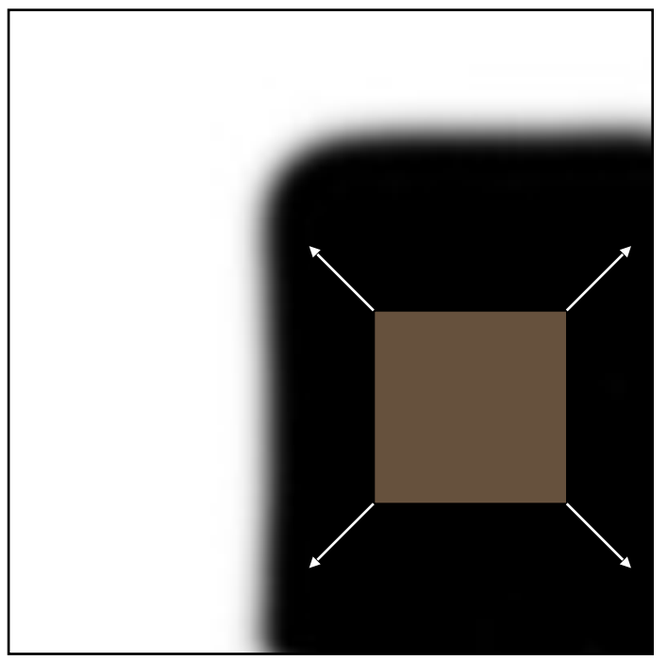
$$Mx_i = \lambda_i x_i$$

The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

Corner response function







"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$

"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

"flat" region
λ_1 and λ_2 are sma

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Example of Harris application

Example of Harris application

Compute corner response at every pixel.

Example of Harris application

Properties of the Harris corner detector

Rotation invariant? **Yes**

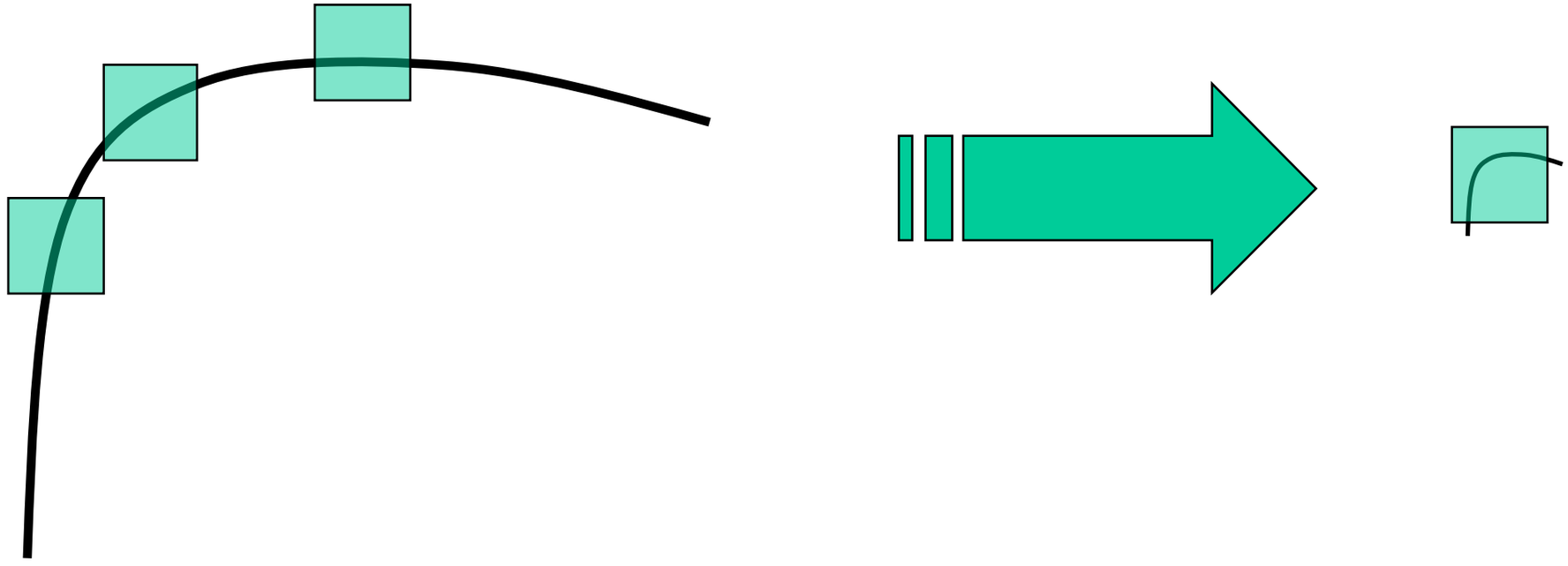
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

Scale invariant?

Properties of the Harris corner detector

Rotation invariant? **Yes**

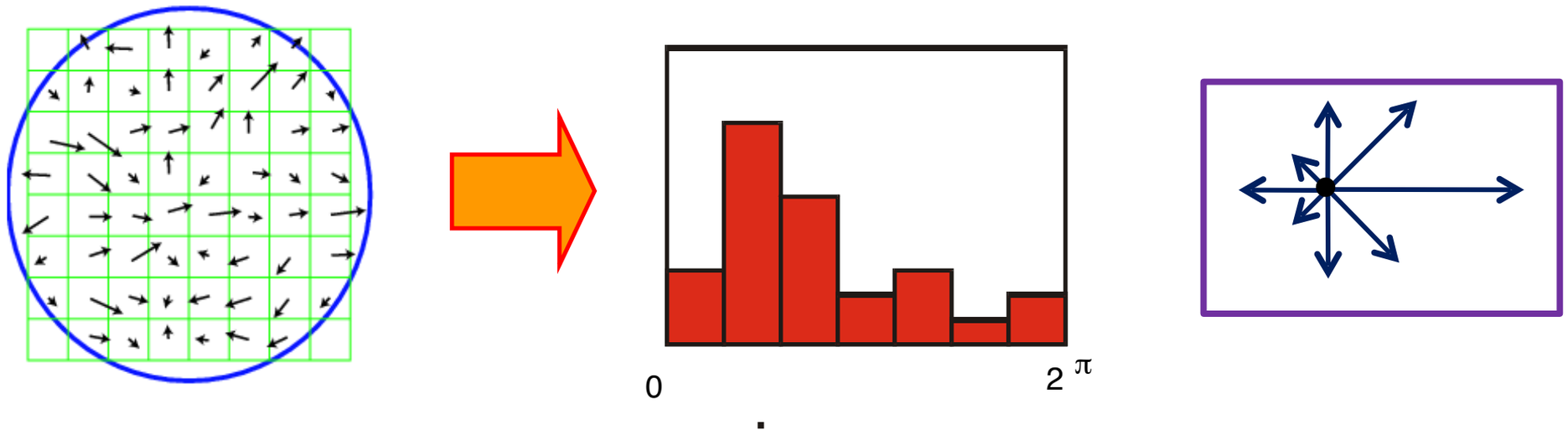
Scale invariant? **No**

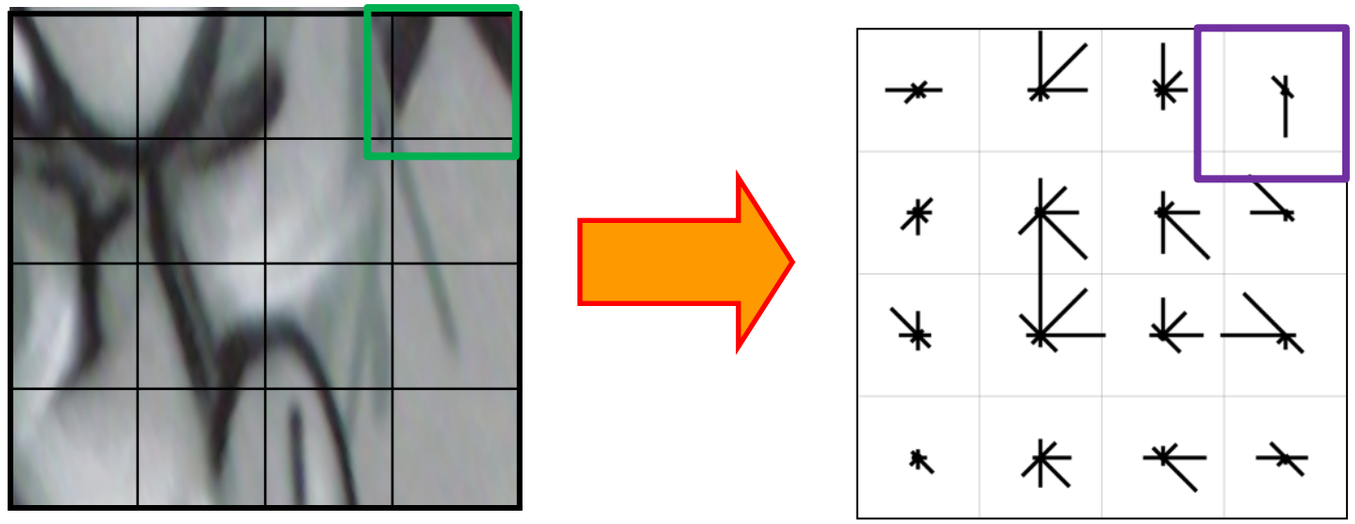


All points will be classified as edges

Corner !

Next time: invariant features and matching





Worktime: Project 1

9 September 2014 Consultations

Bryan

	Group # and Location
10:05	
10:15	
10:25	
10:35	

Pengxiang

	Group # and Location
10:05	
10:15	
10:25	
10:35	