Feature Extraction

PART 1: OUVERTURE

Feature matching

Feature: measured characteristic of (part of) a pattern / object

Goal : efficient matching for

registration (i.e. mosaicking / overlaying),
correspondences for 3D,
tracking,
recognition,
…

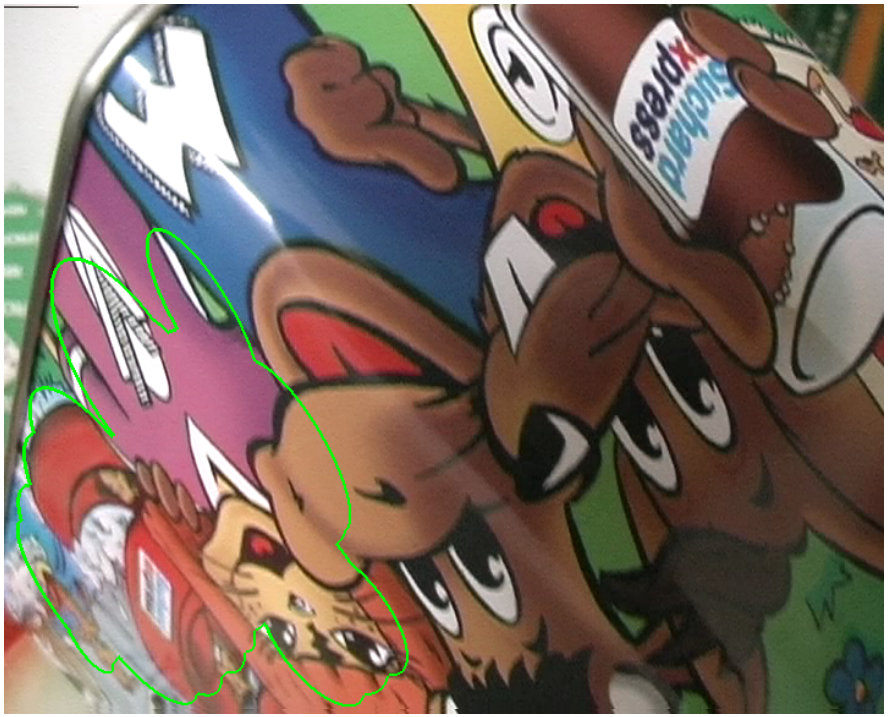
Computer Vision

Feature matching – examples for recognition

Computer Vision

Feature matching

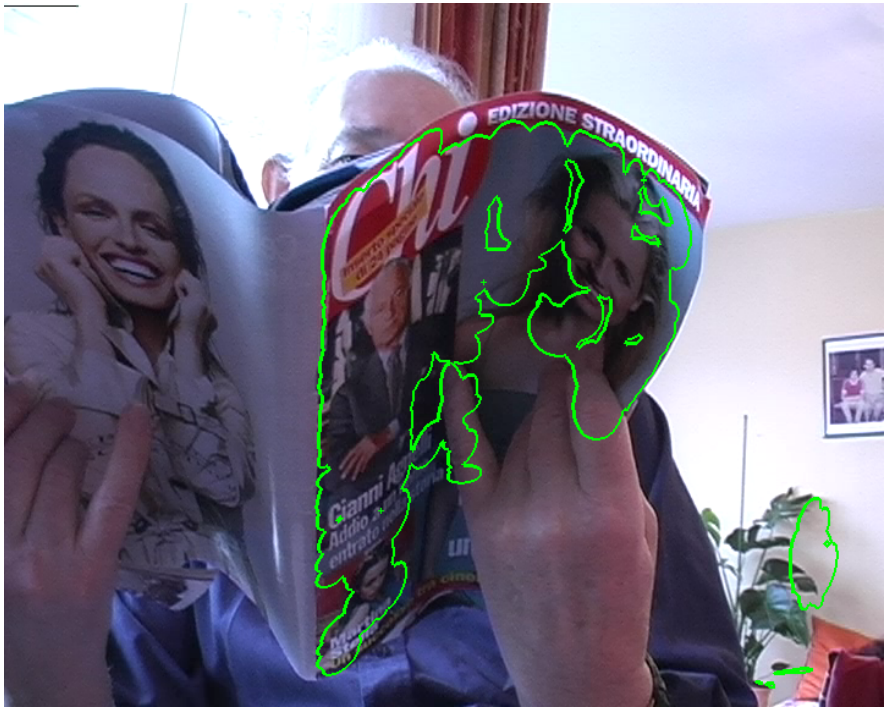
Feature matching



Large scale change, heavy occlusion

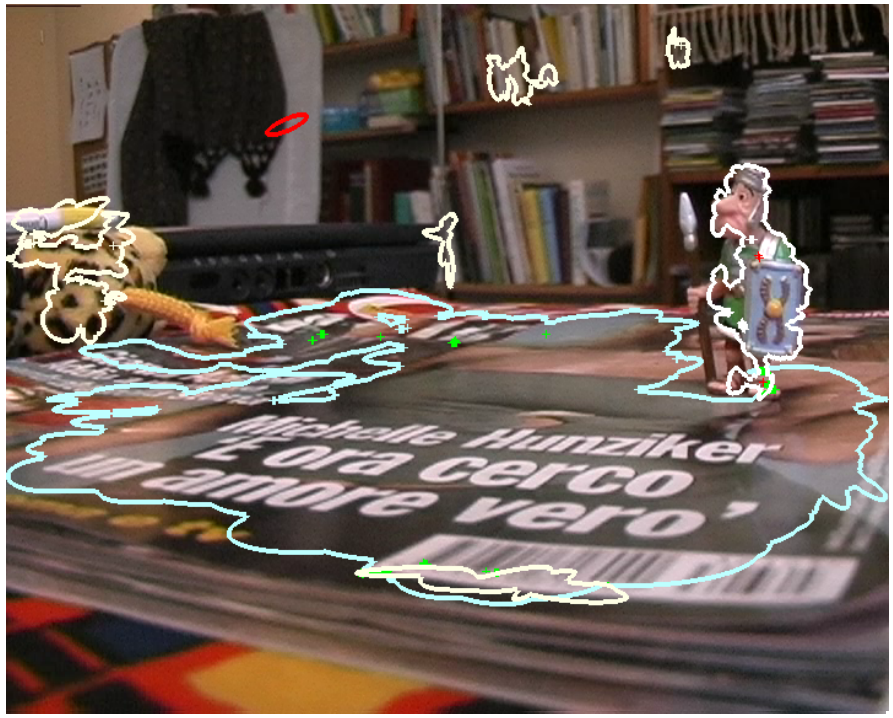
Computer Vision

Feature matching



Deformation, illumination change, occlusion

Feature matching



Large scale change, perspective deformation, extensive clutter

Feature matching

Extensive clutter, scale, occlusion, blur

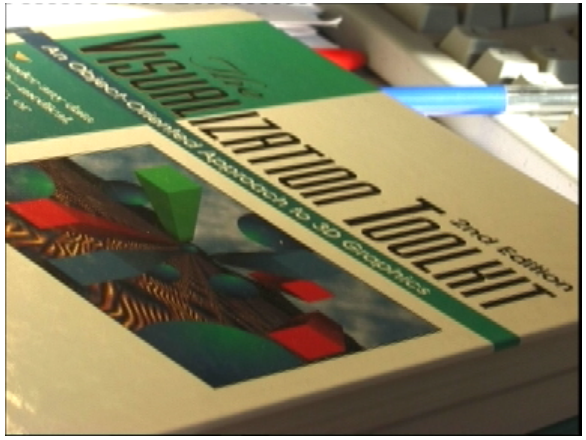
Matching: a challenging task

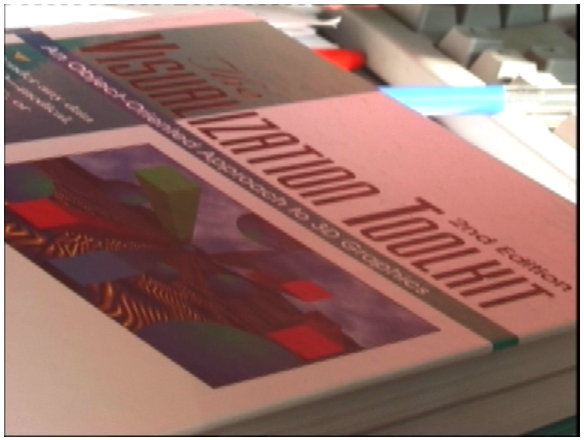
Re-iterating the difficulties with matching highlighted thus far:

- features to deal with **large variations** in
  - Viewpoint

Matching: a challenging task

- features to deal with **large variations** in
  - Viewpoint
  - Illumination





Matching: a challenging task

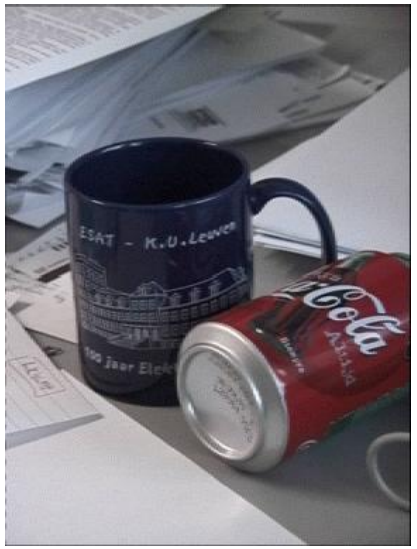
- features to deal with **large variations** in
  - Viewpoint
  - Illumination
  - Background

Matching: a challenging task

- features to deal with **large variations** in
  - Viewpoint
  - Illumination
  - Background

And with

occlusions



Considerations when selecting features

1. Complete (describing pattern unambiguously) or not
2. Robustness of extraction
3. Ease of extraction
4. Global vs. local

Considerations when selecting features

1. Complete (describing pattern unambiguously) or not
2. Robustness of extraction
3. Ease of extraction
4. Global vs. local

PART 2: `INTEREST POINTS’

Additional requirement:

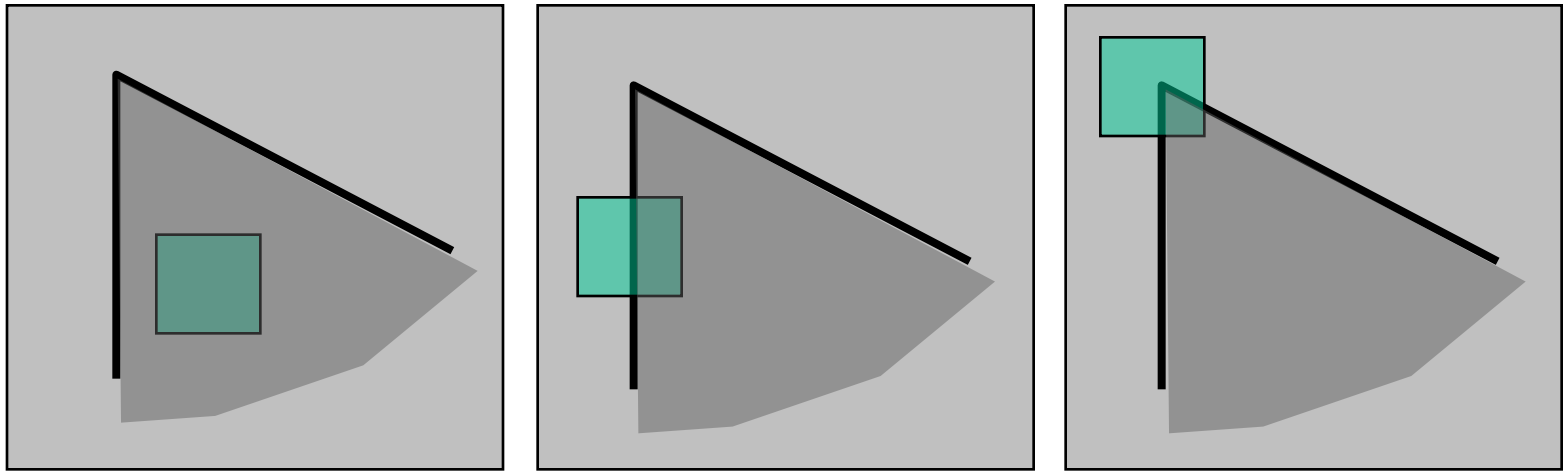
A feature should capture something *discriminative* about a well *localisable* patch of a surface

We start with the well localisable bit:

Shifting the patch a bit should make a big difference in terms of the underlying pattern

Uniqueness of a patch

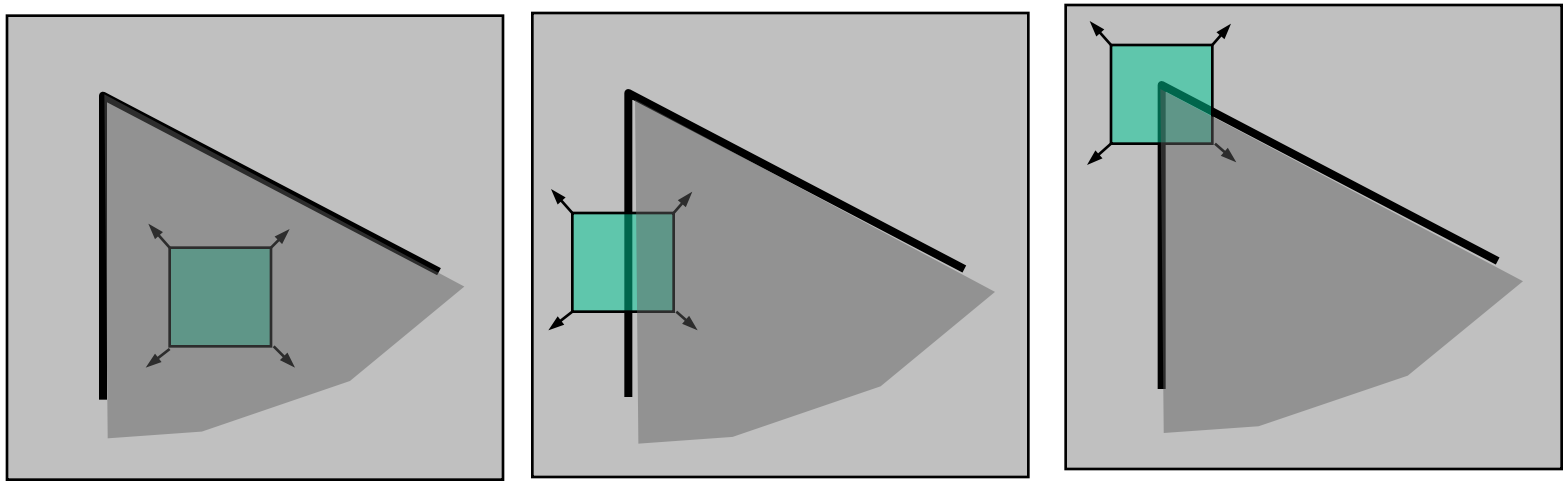
consider the pixel pattern within the green patches:



Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

How do the patterns change upon a shift?



"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

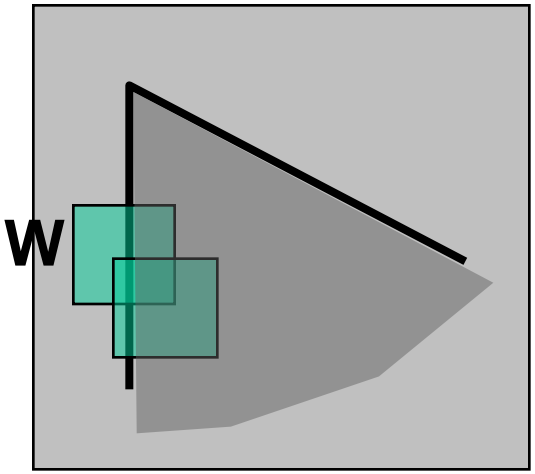
Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

Now that we know what we are looking for, we need to find a good way to do it in practice…

Uniqueness of a patch

Consider shifting the patch or `window' **W** by (u,v)



- how do the pixels in **W** change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" or *E(u,v)*:

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

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

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

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order}$$

If the motion (u,v) is small, then 1st order appr. is good

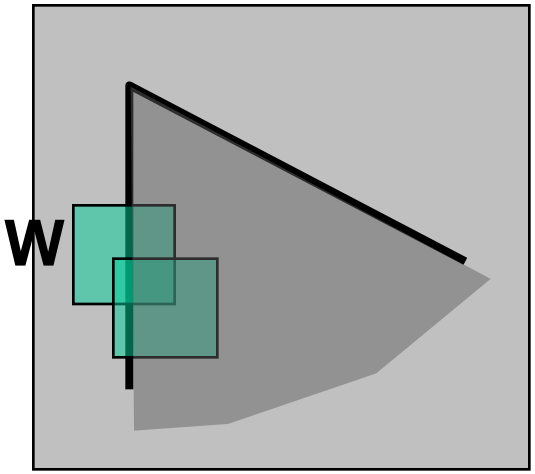
$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

Plugging this into the formula at the top…

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch



Then, with shorthand: $I_x = \frac{\partial I}{\partial x}$

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

$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} \left[[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

This can be rewritten further as:

$$E(u,v) = \sum_{(x,y)\in W} [u \; v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Which directions $[u,v]$ will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of ***H***

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch - PCA

In order to get the eigenvectors/values of ***H*** we need to apply PCA to ***H***

PCA will be covered in the lecture on Unitary Transf. So, we give a sneak preview only here.
H is basically is a covariance matrix of (I_x, I_y) ; that will later clarify the link with the part on PCA

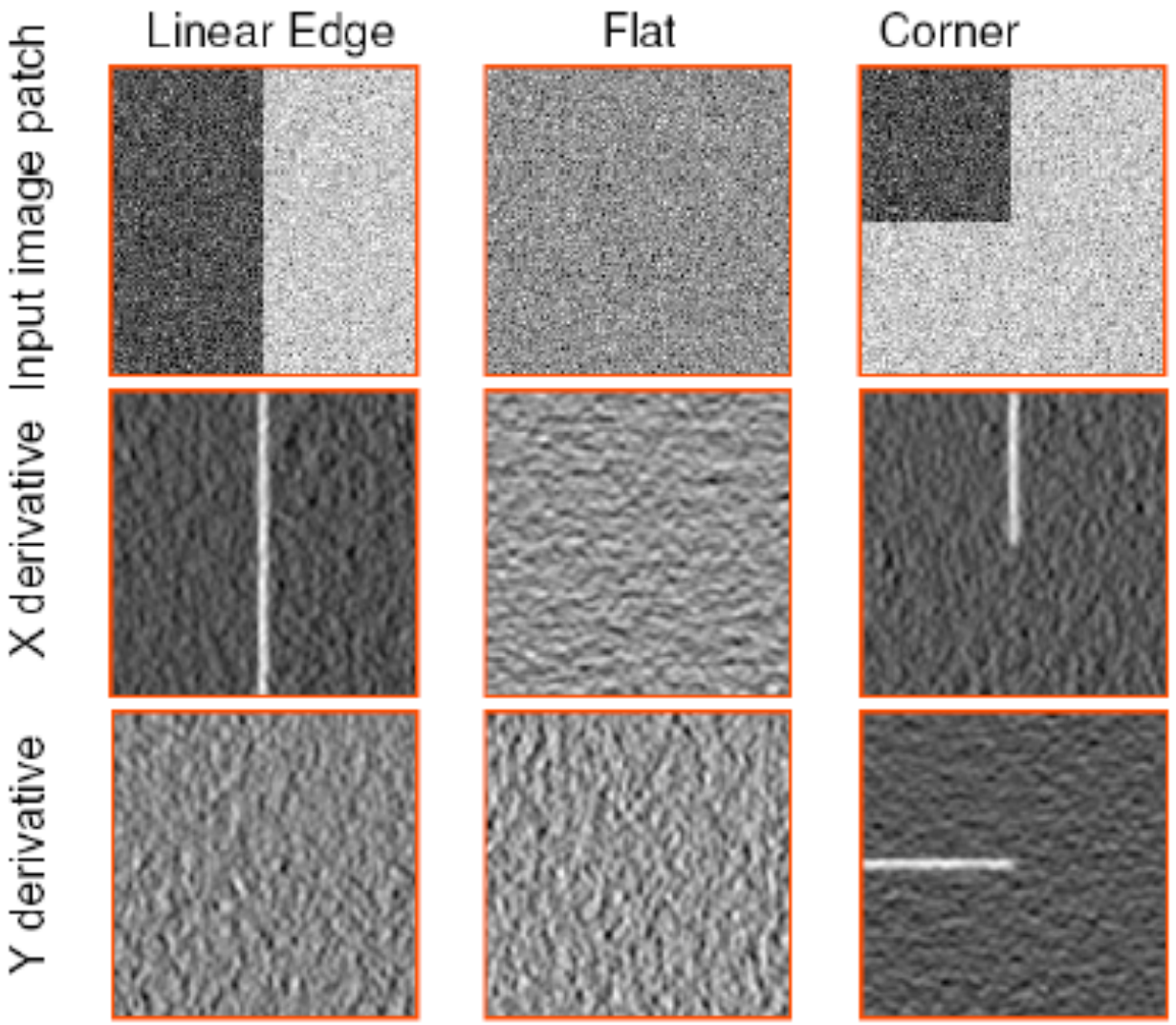
PCA assumes the distribution of (I_x, I_y) to be Gaussian.

Hence, it fits and ellipse to that distribution.

The eigenvectors (`principal components' in PCA parlance) correspond to the directions of the long and the short axis of the ellipse. The eigenvalue are given by the ellipse size in those direction.

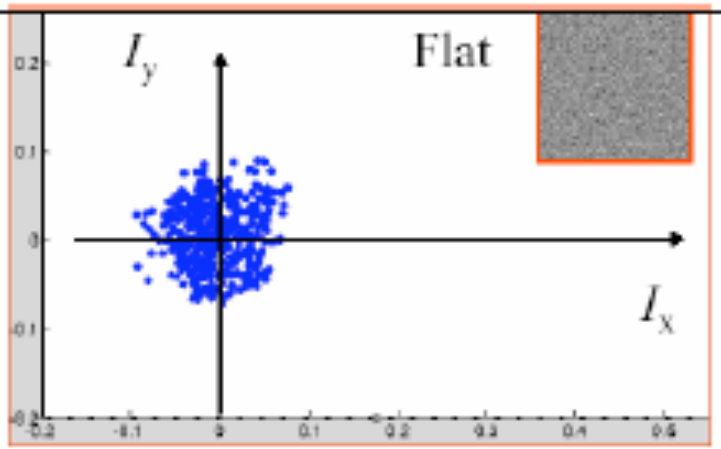
Uniqueness of a patch - PCA

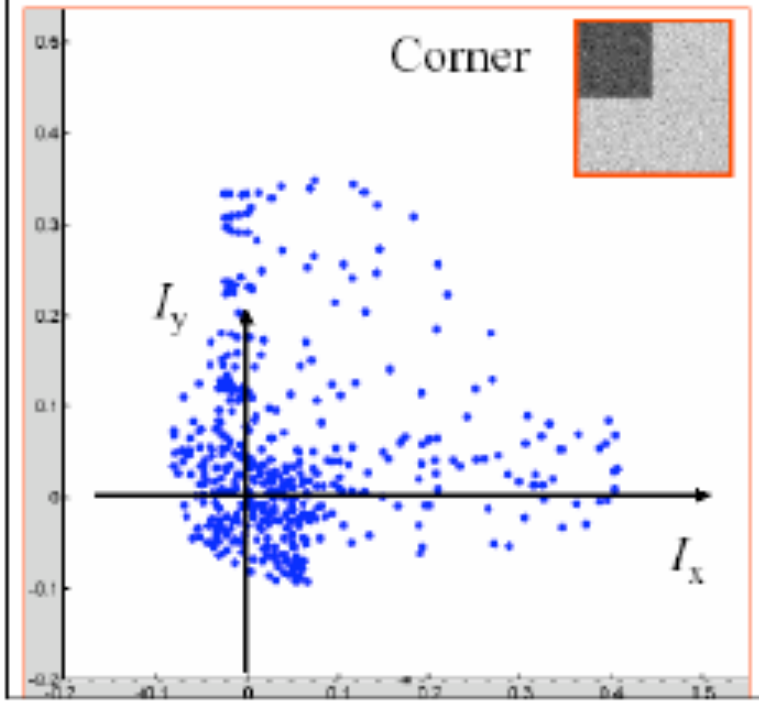
Let's look at those (I_x, I_y) *distributions for:*

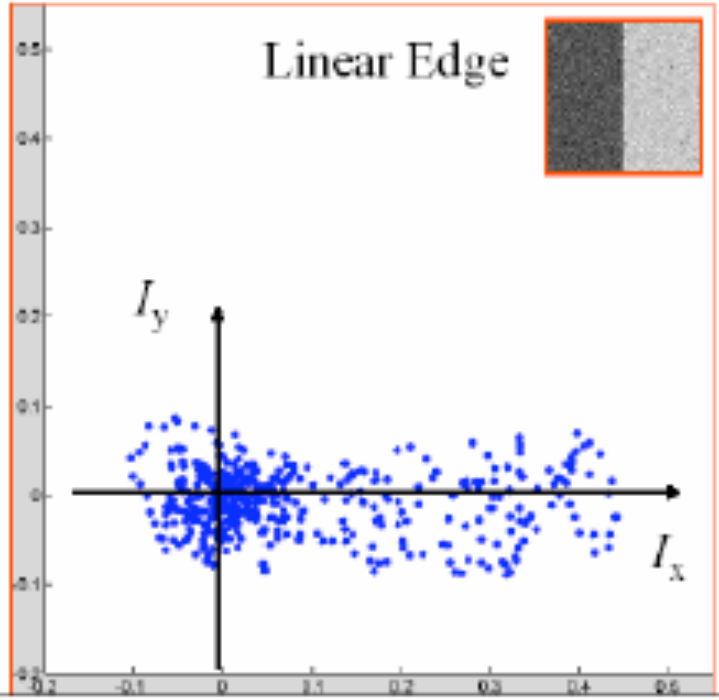


Uniqueness of a patch - PCA

The distribution of the x and y derivatives is very different for all three types of patches

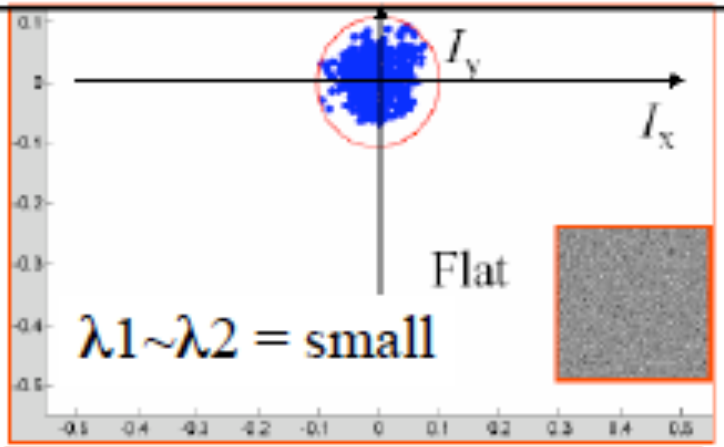


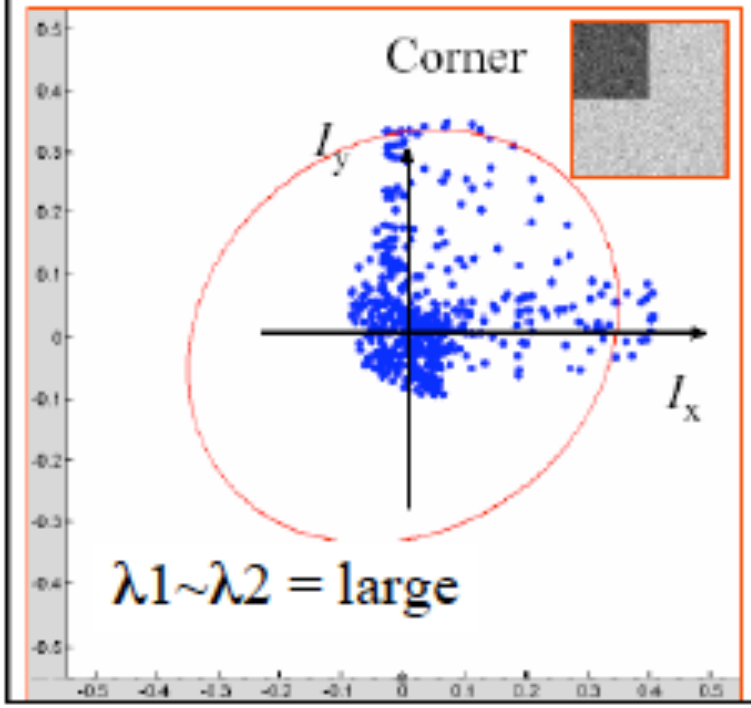


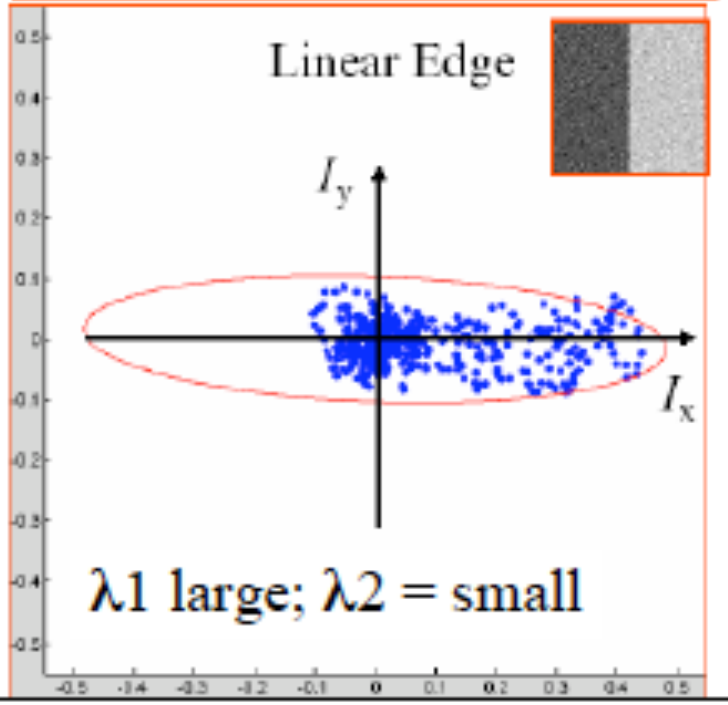


Uniqueness of a patch - PCA

The distribution of x and y derivatives can be characterized by the shape and size of the principal component ellipse







Uniqueness of a patch

This can be rewritten further as:

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

Eigenvalues and eigenvectors of H

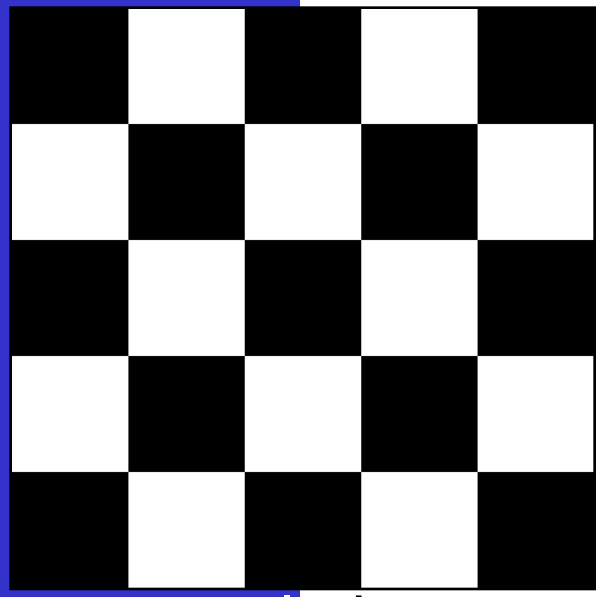
- Define shifts with the smallest and largest change (E value)
- x_+ = direction of **largest** increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of **smallest** increase in E.
- λ_- = amount of increase in direction x_+

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

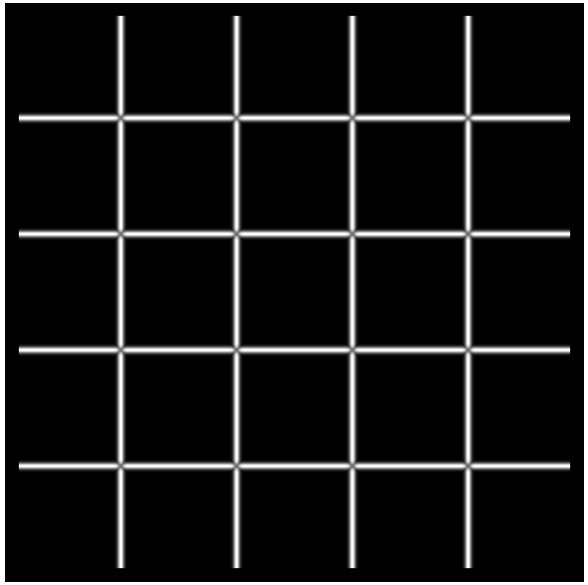
Uniqueness of a patch

Want *E(u,v)* to be ***large*** for small shifts in ***all*** directions

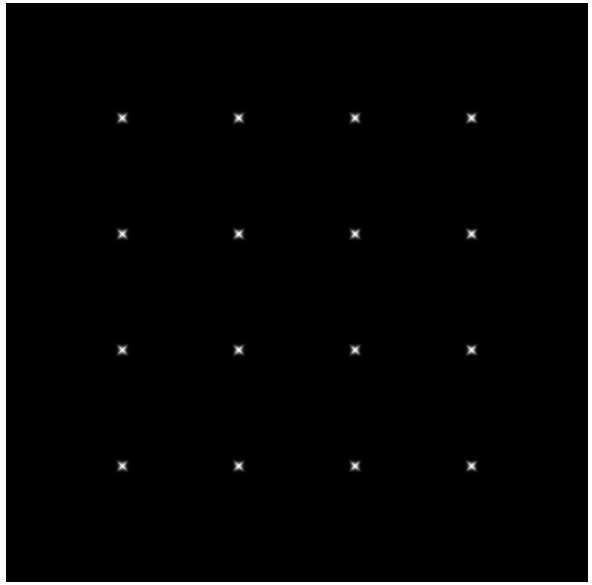
- the *minimum* of *E(u,v)* should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_-) of ***H***



example image



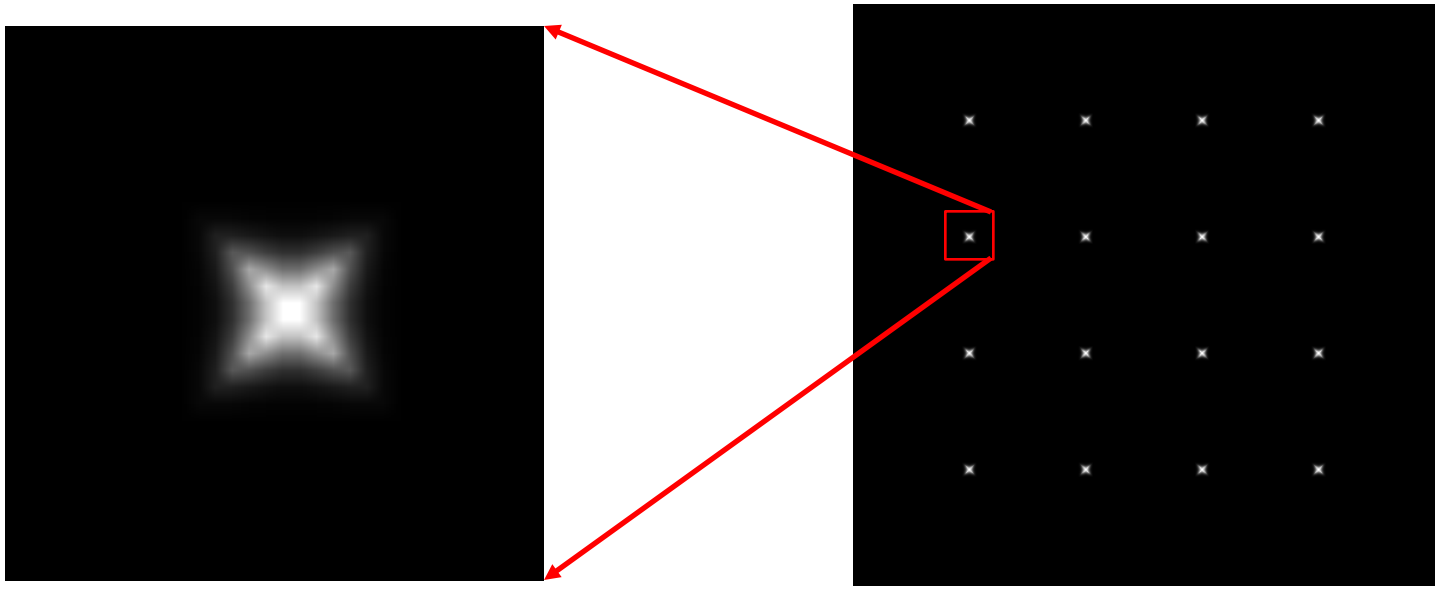
λ_+



λ_-

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch



λ_-

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

The Harris corner detector

λ_- is a variant of the "Harris operator" for feature detection

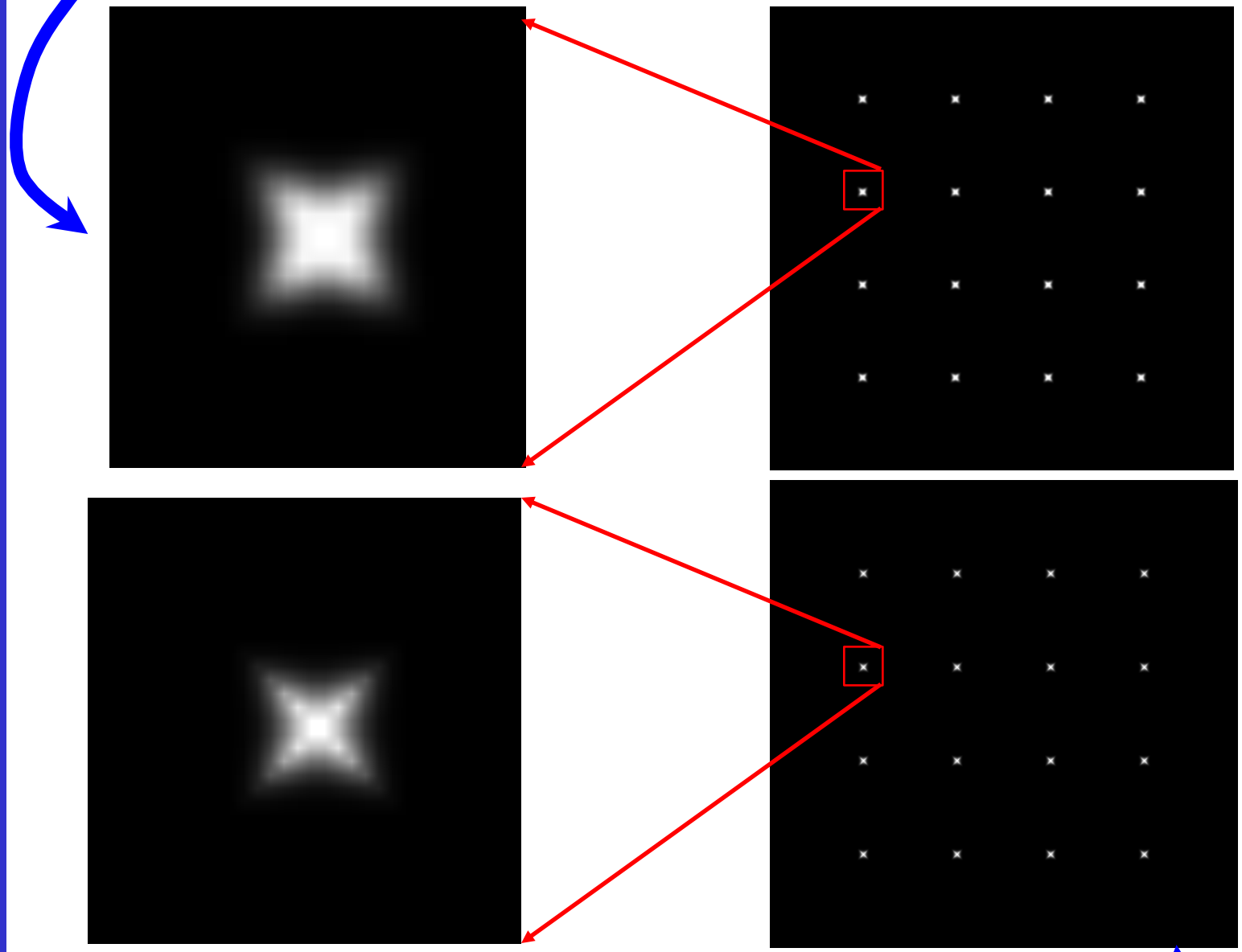
$$Determinant - k\,(Trace)^2$$

(k is empirically chosen, typically 0.04 to 0.06)

- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Det(H) = product of eigenvalues, Trace(H) = sum eigenval.
- Thus, related to eigenvalues but cheaper to compute
- AND more refined detection of corners
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other corner detectors, this is one of the most popular ones

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

The Harris corner detector

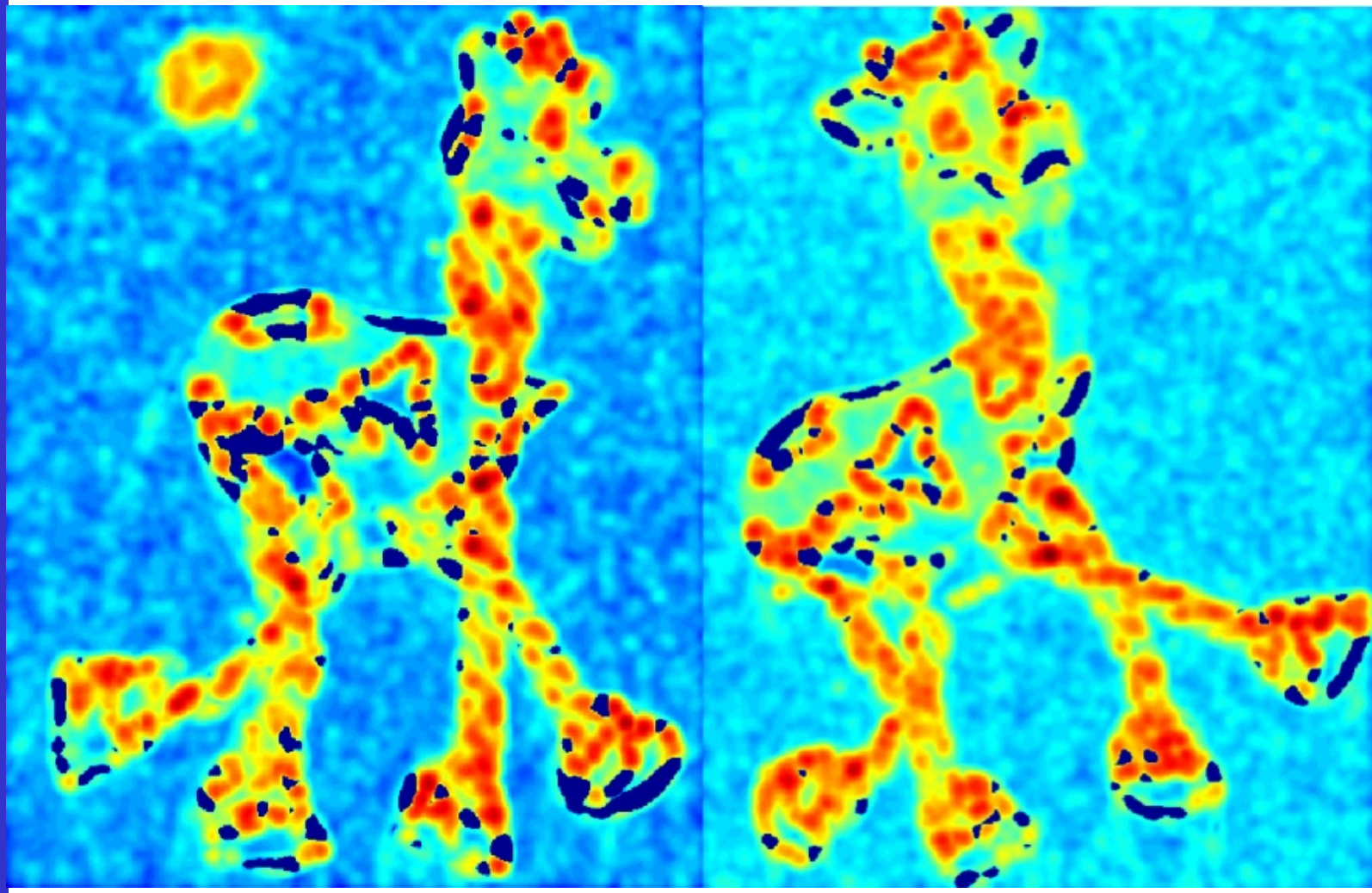


Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

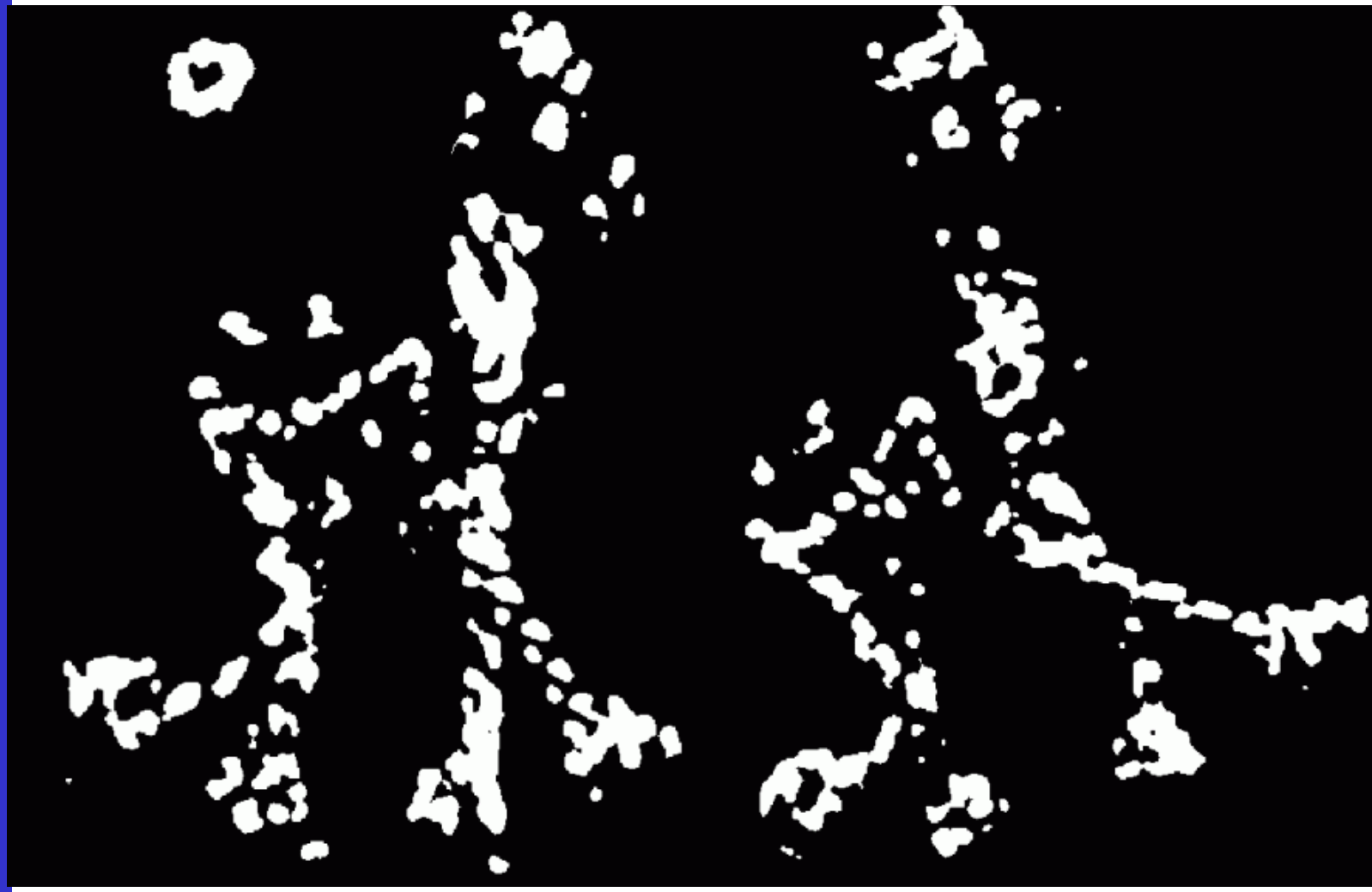
The Harris corner detector

2 views of an object… are the corners stable?

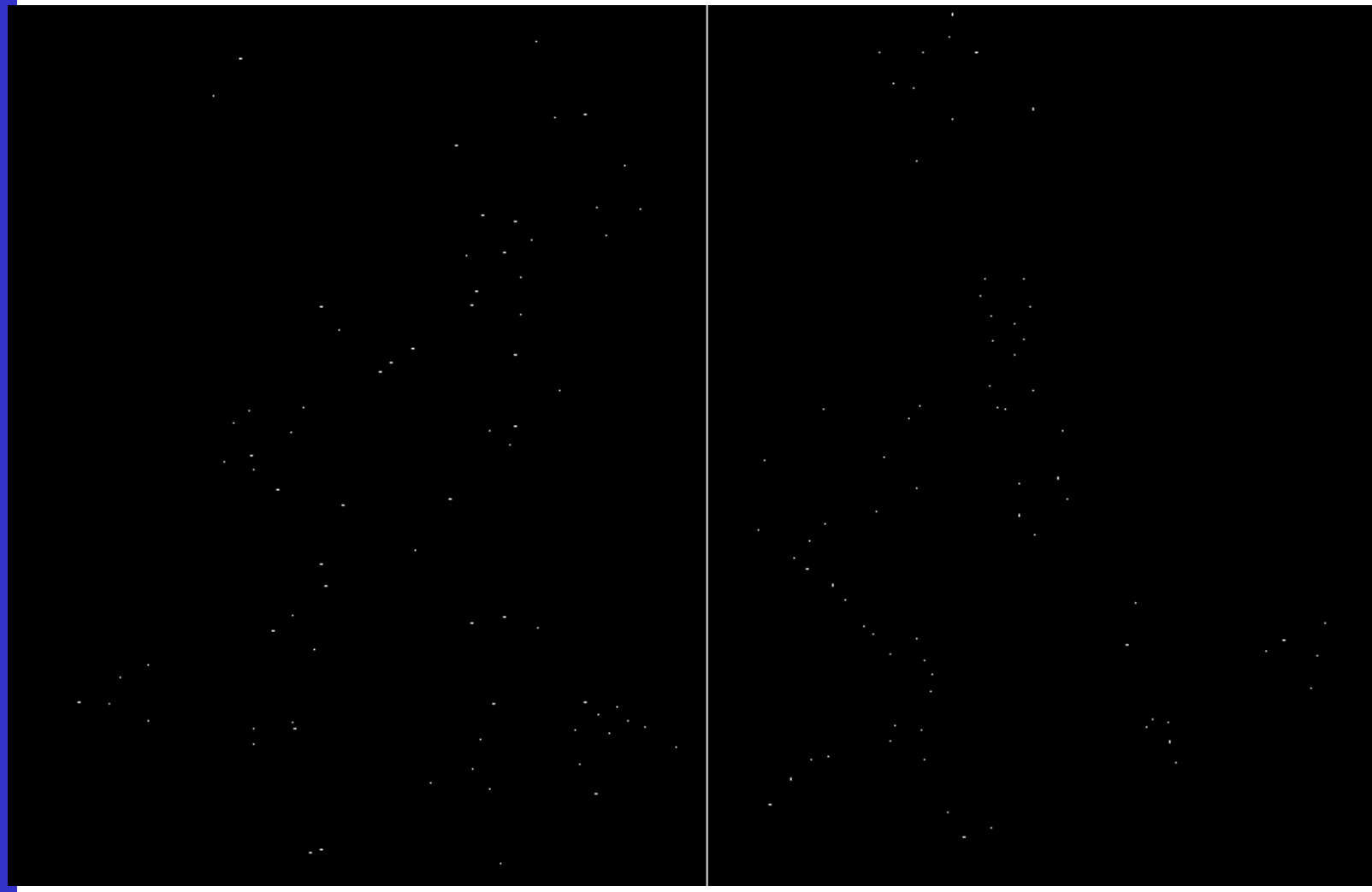
The Harris corner detector



The Harris corner detector



The Harris corner detector



The Harris corner detector

Interest points

Corners are the most prominent example of so-called **`Interest Points',** i.e. points that can be well localised in different views of a scene

`Blobs' are another, as we will see… but also a blob is a region with intensity changes in multiple directions

PART 3:
A TALE OF INVARIANCE

Need for an invariant descriptor:

There are many corners coming out of our **DETECTOR**, but they still cannot be told apart

We need to describe their surrounding image patch such we can discriminate between them, i.e. we need to build a feature vector for the patch, a so-called **DESCRIPTOR**

During a **MATCHING** step, the descriptors can then be compared. In order for that to be easy, the descriptors for corresponding, detected points must be similar in different views. i.e. *invariant* under the changes between these views.

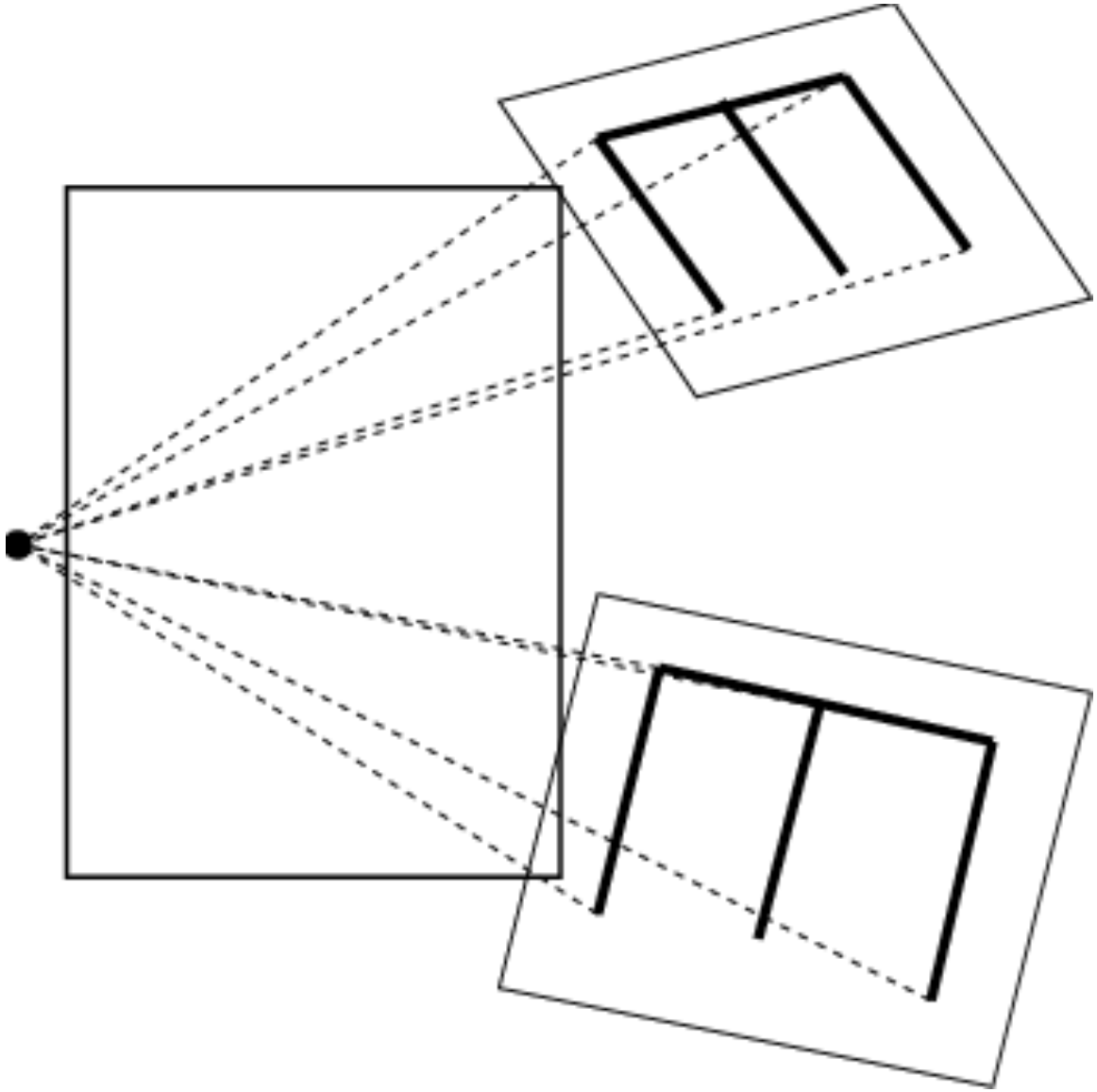
Additional requirement on the descriptor:

Invariance under geom./phot. change

A local patch is small,
hence probably rather *planar.*

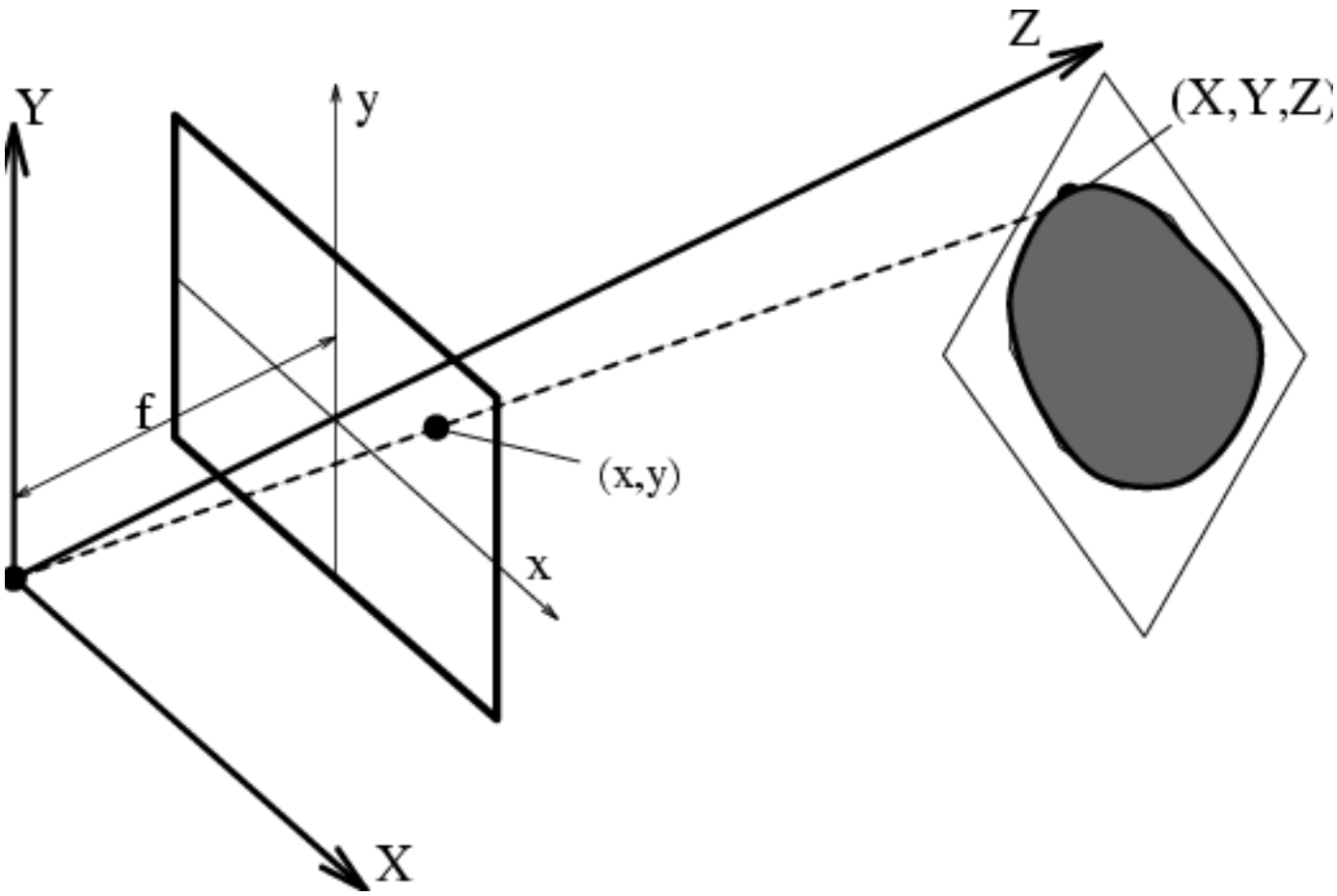
But how do planar patches deform when looking at their image projection?
i.e. we determine the *geometric* changes the descriptor should remain invariant under

Planar pattern projections to be compared



Projection : a camera model

Reversed center-image pinhole model :



Projection : equations

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

an approximation...

special cases :

1. Z constant, or
2. object small compared to its distance to the camera

$$\left.\begin{aligned} x &= kX \\ y &= kY \end{aligned}\right\}$$ pseudo-perspective

Deformations under projection

In particular, how do different views of the same planar shape / contour differ ?

we consider 3 cases :

1. viewed from a perpendicular direction
2. viewed from any direction but at sufficient distance to use pseudo - perspective
3. viewed from any direction and from any distance

Deformations : case 1

$$X' = \cos\theta\, X - \sin\theta\, Y + t_1$$
$$Y' = \sin\theta\, X + \cos\theta\, Y + t_2$$
$$Z' = Z + t_3$$

$$(X(t), Y(t), Z(t)) \rightarrow (x(t), y(t)) \qquad x = kX, \quad y = kY,$$
$$(X'(t), Y'(t), Z'(t)) \rightarrow (x'(t), y'(t)) \qquad x' = k'X', \; y' = k'Y',$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{k'}{k} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + k' \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

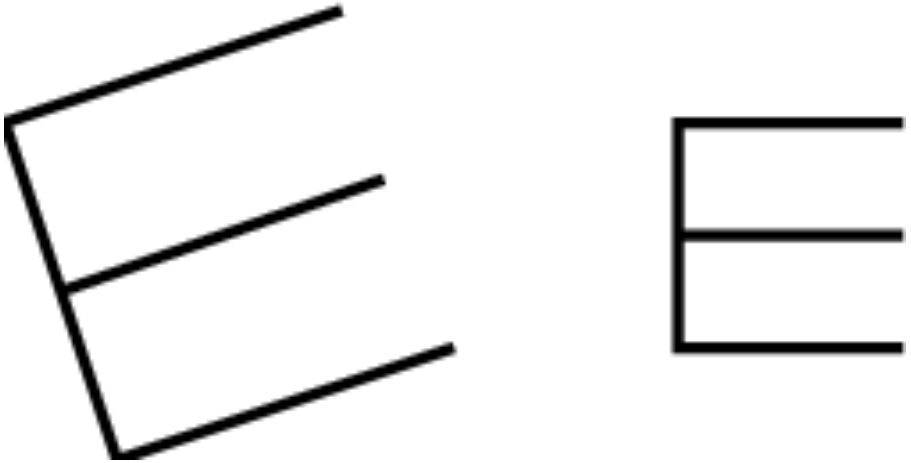
Image projection undergoes a similarity transformation

Euclidean motions if $k = k'$ (i.e. $Z = Z'$)

Summary case 1

Case of fronto-parallel viewing :

SIMILARITY



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathrm{s} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Deformations : case 2

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$x = kX, \qquad y = kY,$$
$$x' = k'X', \qquad y' = k'Y',$$

we easily derive

$$x' = \frac{k'}{k} r_{11} x + \frac{k'}{k} r_{12} y + k' r_{13} Z + k' t_1$$

and should eliminate Z

Deformations : case 2 cont' d

We use the planarity restriction :

$$aX + bY + cZ + d = \frac{a}{k}x + \frac{b}{k}y + cZ + d = 0$$

Hence

$$Z = -\frac{a}{kc}x - \frac{b}{kc}y - \frac{d}{c}$$

and thus

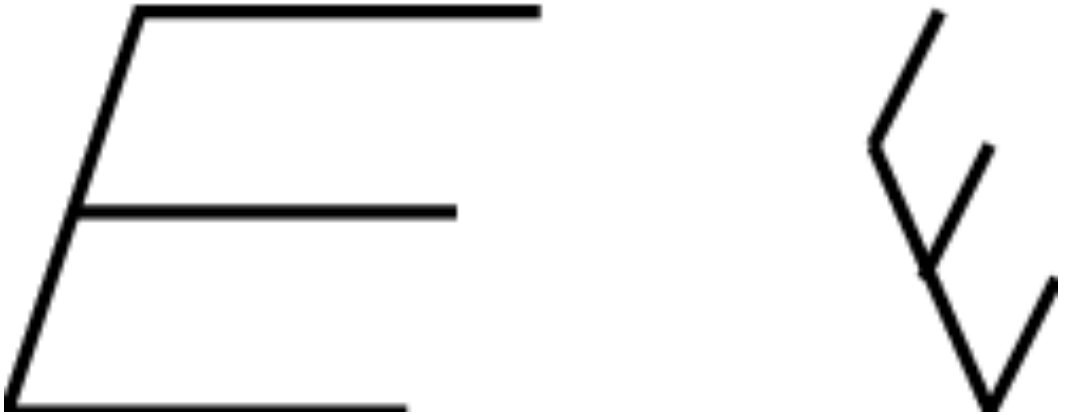
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

we find a 2D affine transformation

Summary case 2

Viewing from a distance :

AFFINITY



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Deformations : case 3

3D Euclidean motion, but perspective projecton

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$x = f\frac{X}{Z}, \qquad y = f\frac{Y}{Z},$$

$$x' = f\frac{X'}{Z'} \qquad y' = f\frac{Y'}{Z'}.$$

Deformations : case 3

3D Euclidean motion, but perspective projecton

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$x' = f\frac{X'}{Z'}$$

$$= f\frac{r_{11}X + r_{12}Y + r_{13}Z + t_1}{r_{31}X + r_{32}Y + r_{33}Z + t_3}$$

$$= f\frac{r_{11}\frac{fX}{Z} + r_{12}\frac{fY}{Z} + r_{13}\frac{fZ}{Z} + t_1\frac{f}{Z}}{r_{31}\frac{fX}{Z} + r_{32}\frac{fY}{Z} + r_{33}\frac{fZ}{Z} + t_3\frac{f}{Z}}$$

Deformations : case 3

Thus,

$$x' = f\,\frac{r_{11}x + r_{12}y + r_{13}f + (ft_1)/Z}{r_{31}x + r_{32}y + r_{33}f + (ft_3)/Z}$$

$$y' = f\,\frac{r_{21}x + r_{22}y + r_{23}f + (ft_2)/Z}{r_{31}x + r_{32}y + r_{33}f + (ft_3)/Z}$$

Deformations : case 3 cont'd

Using the planarity restriction : $Z = \dfrac{-df}{ax + by + cf}$.

$$x' = f\,\frac{r_{11}x + r_{12}y + r_{13}f + (ft_1)/Z}{r_{31}x + r_{32}y + r_{33}f + (ft_3)/Z}$$

$$y' = f\,\frac{r_{21}x + r_{22}y + r_{23}f + (ft_2)/Z}{r_{31}x + r_{32}y + r_{33}f + (ft_3)/Z}$$

Deformations : case 3 cont'd

Using the planarity restriction : $Z = \dfrac{-df}{ax+by+cf}$.

$$x' = f\frac{\left(r_{11}-\frac{t_1}{d}a\right)x+\left(r_{12}-\frac{t_1}{d}b\right)y+\left(r_{13}-\frac{t_1}{d}c\right)f}{\left(r_{31}-\frac{t_3}{d}a\right)x+\left(r_{32}-\frac{t_3}{d}b\right)y+\left(r_{33}-\frac{t_3}{d}c\right)f}$$

$$y' = f\frac{\left(r_{21}-\frac{t_2}{d}a\right)x+\left(r_{22}-\frac{t_2}{d}b\right)y+\left(r_{23}-\frac{t_2}{d}c\right)f}{\left(r_{31}-\frac{t_3}{d}a\right)x+\left(r_{32}-\frac{t_3}{d}b\right)y+\left(r_{33}-\frac{t_3}{d}c\right)f}$$

$$x' = \frac{p_{11}x+p_{12}y+p_{13}}{p_{31}x+p_{32}y+p_{33}} \qquad y' = \frac{p_{21}x+p_{22}y+p_{23}}{p_{31}x+p_{32}y+p_{33}}$$

we find a 2D projective transf. = homography

Summary case 3

General viewing conditions:

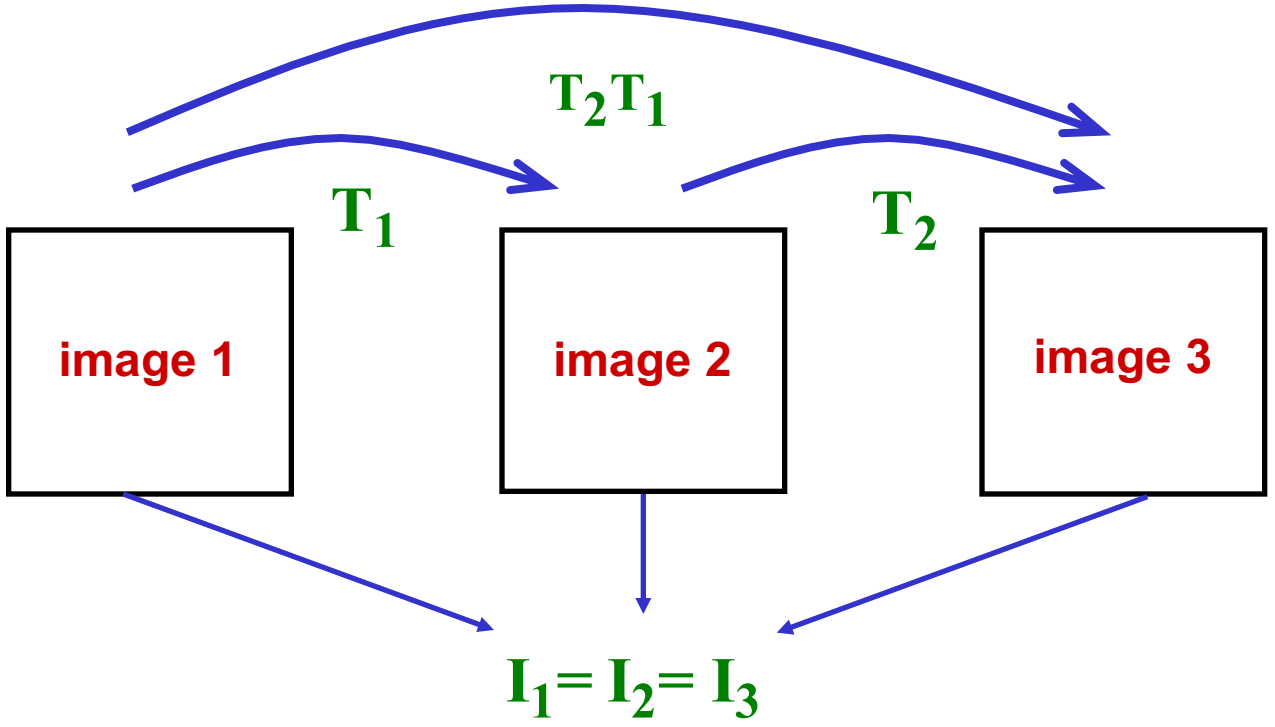
PROJECTIVITY

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

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

Invariance and groups

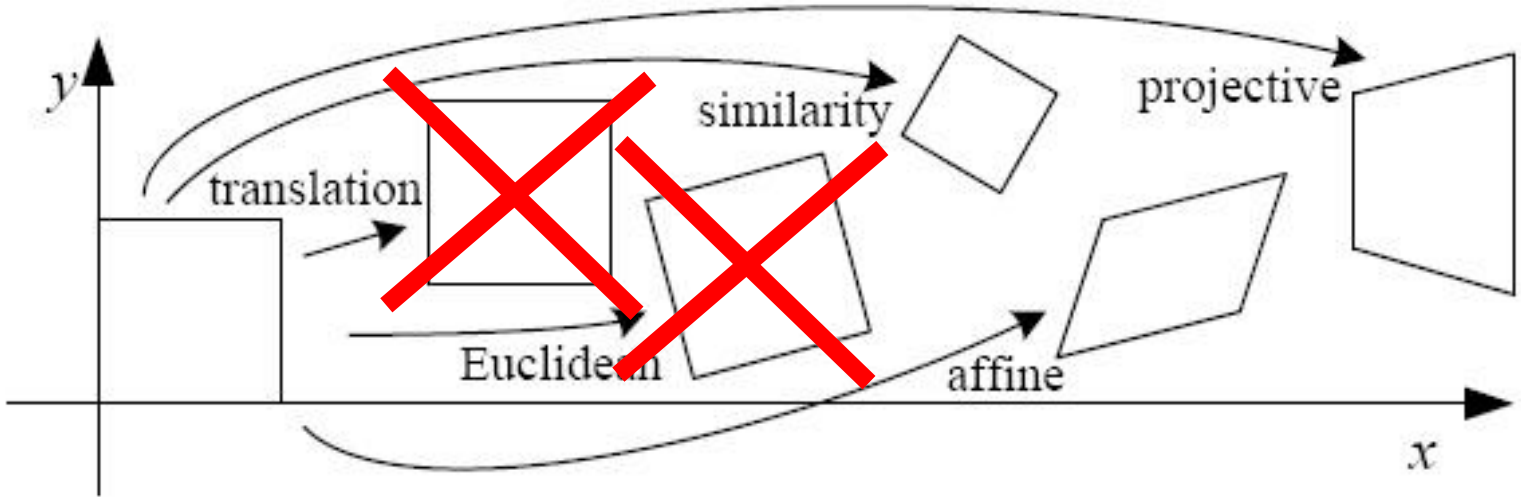
Invariance w.r.t. group actions



Invariance under transformations implies invariance under the smallest encompassing group

Remarks

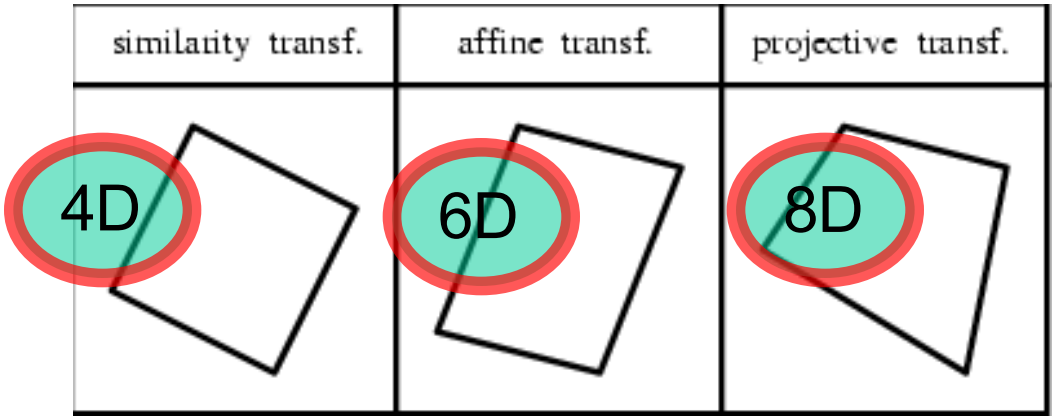
There are more groups, but the ones described are the most relevant for us



Remarks

Complexity of the groups goes up with the generality of the viewing conditions, and so does the complexity of the group's invariants

e.g. effects on a square:



Fewer invariants are found going from left to right

Similarities $\subset$ affinities $\subset$ projectivities

Invar. Proj. $\subset$ invar. Aff. $\subset$ invar. Sim.

Remarks

Many types of invariants can be generated, e.g. for points, for lines, for moments, etc.

notations for point coordinates:

Image coordinates of a point: $X = (x,y)^T$

Subscript indentifiers: $X_i = (x_i,y_i)^T$ for point i

Remarks

Many types of invariants can be generated, e.g. for points, for lines, for moments, etc.

Examples for points

Similarities: $\dfrac{\|X_1 - X_2\|}{\|X_1 - X_3\|}$

Affinities: $\dfrac{\begin{vmatrix} X_1 - X_2 & X_1 - X_3 \end{vmatrix}}{\begin{vmatrix} X_1 - X_2 & X_1 - X_4 \end{vmatrix}}$

Projectivities: $\dfrac{\begin{vmatrix} X_1 - X_2 & X_1 - X_5 \end{vmatrix}}{\begin{vmatrix} X_1 - X_3 & X_1 - X_5 \end{vmatrix}} \Big/ \dfrac{\begin{vmatrix} X_2 - X_4 & X_2 - X_5 \end{vmatrix}}{\begin{vmatrix} X_3 - X_4 & X_3 - X_5 \end{vmatrix}}$

Comment:

As our local patches are by definition small, we expect that invariance under affine or even similarity transformations will do, also for case 3 viewing conditions.

That invariants under similarities can serve us well under case 3 viewing conditions should come as a surprise… their deviation from strict geometric requirements is compensated by their computational simplicity compared to affine invariants, if viewing changes are not too extreme.

Computer Vision

photometric changes

photometric changes

- A reasonable model for typical illumination changes is given by

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} s_R & 0 & 0 \\ 0 & s_G & 0 \\ 0 & 0 & s_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} o_R \\ o_G \\ o_B \end{bmatrix}$$

- linear part caters for changes of color and/or intensity of the illumination;
  the 3 scale factors are the same if the illumination changes intensity only

- the non-linear part caters for specularities

Thus:

As our patches cover part of a surface containing some surface texture, we will also have to be invariant under photometric changes if we want to use that texture.

photometric changes

- Contrasts (intensity differences) let the non-linear offsets cancel; hence gradients are good !

- Moreover the orientation of gradients in the color bands is invariant under their linear changes, as is the intensity gradient orientation in case the scale factors are identical;
this is indeed relevant if the illumination changes its intensity, but not its color, which is typically assumed.

- But even under changing color of the illumination, in practice edge orientations tend to remain the same.

Our goal

Define good interest points, i.e.

DETECTORS + **DESCRIPTORS**

Our goal

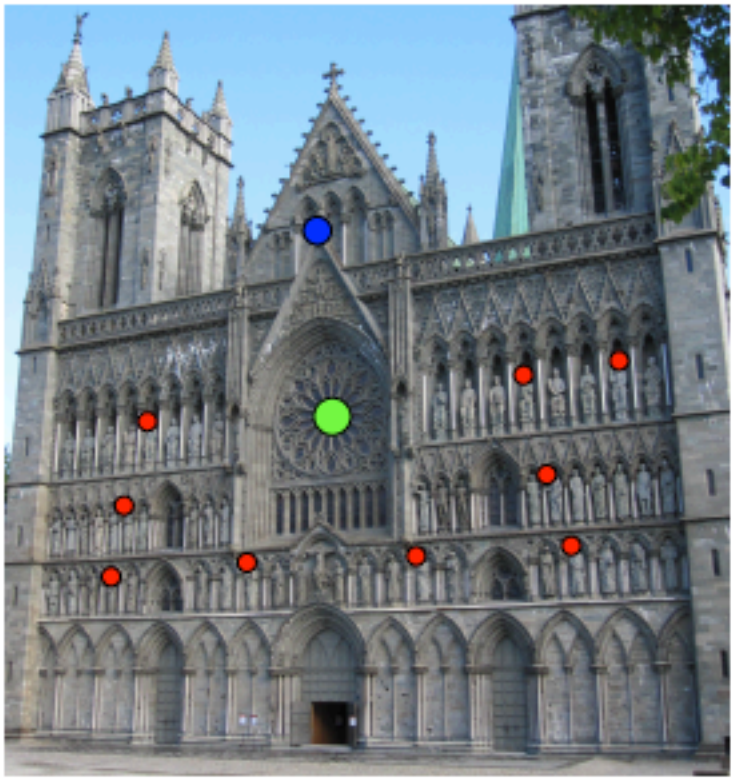
Define good interest points, i.e.

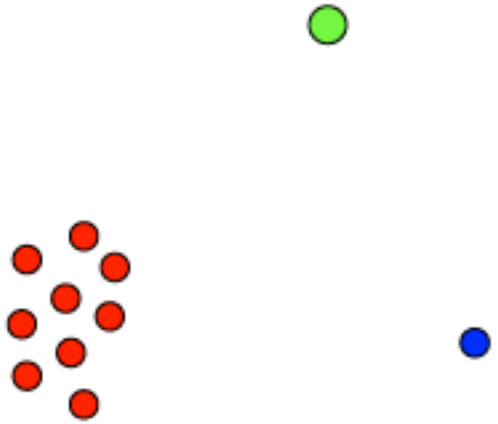
DETECTORS + **DESCRIPTORS**

The detector typically yields image points

Descriptors then are a vector of measurements taken around each such point

Descriptor Matching





Matching of the descriptors based on closeness according to an appropriate metric…

Hierarchical matching can help to speed up

Not discussed in detail here

notes on matching

- Interest points are matched on the basis of their descriptors
- E.g. nearest neighbour, based on some distance like Euclidean or Mahalanobis; good to compare against 2nd nearest neighbour: OK if difference is big; or fuzzy matching w. multiple neighbours
- Speed-ups by using lower-dim. descriptor space (PCA) or through some coarse-to-fine scheme (very fast schemes exist to date!)
- Matching of individual points typically followed by some consistency check, e.g. epipolar geometry, homograpy, or topological

PART 3: THE PATCH…

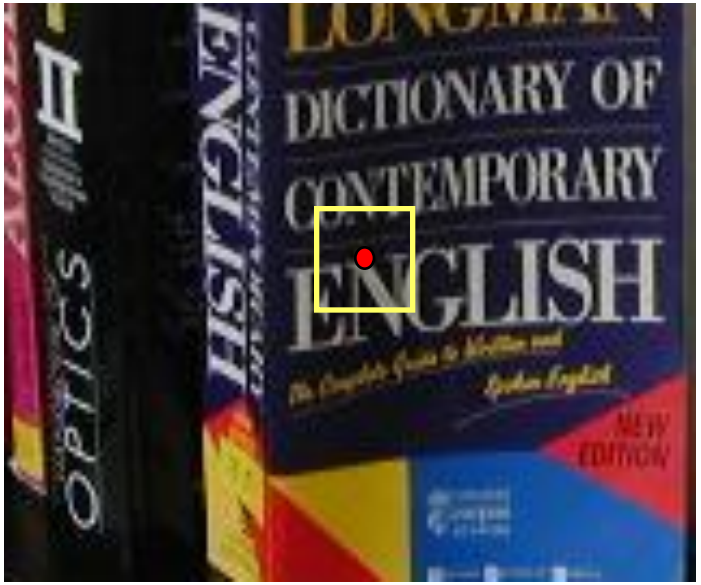
Our goal

BUT, so far we have glossed over an important issue:

The shape of the patch should change with viewpoint

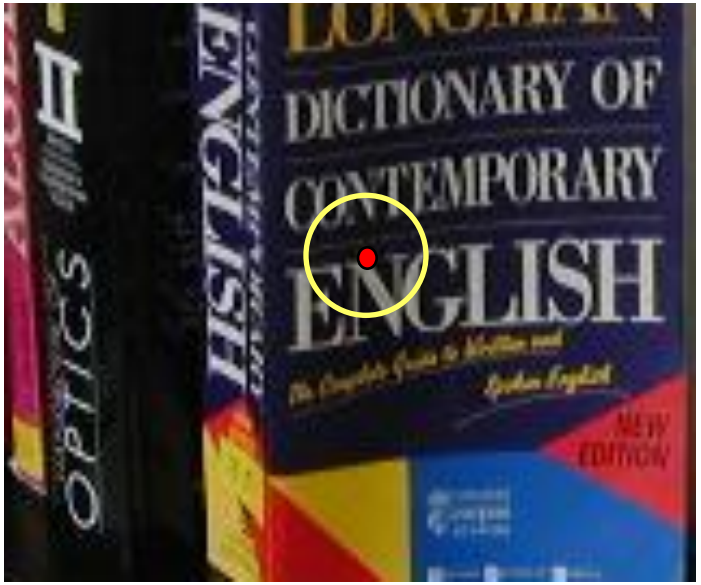
The need for variable patch shape

The need for variable patch shape



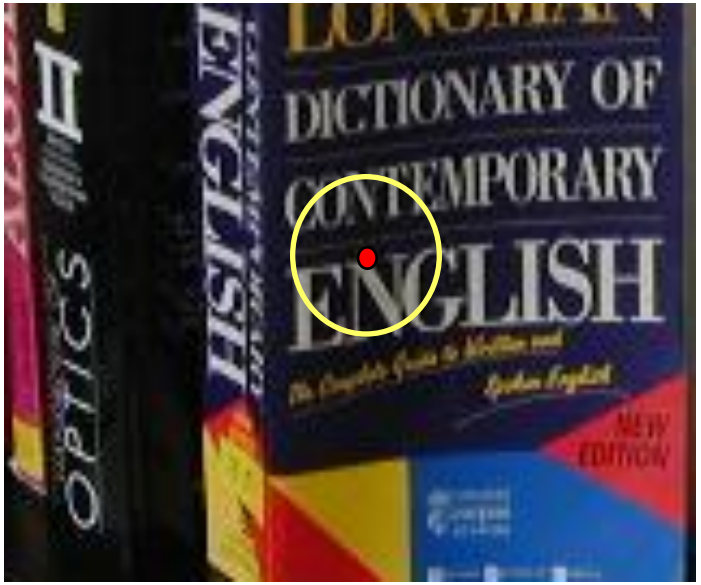
Taking the same square patch around corresponding interest points leads to a very different content of the patches… hence the matching will become hard.

The need for variable patch shape



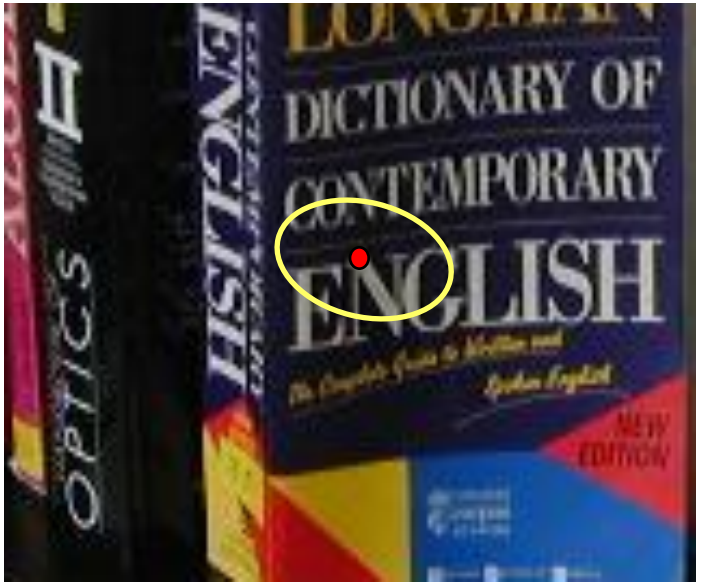
Replacing the squares by identical circles does not really help much…

The need for variable patch shape



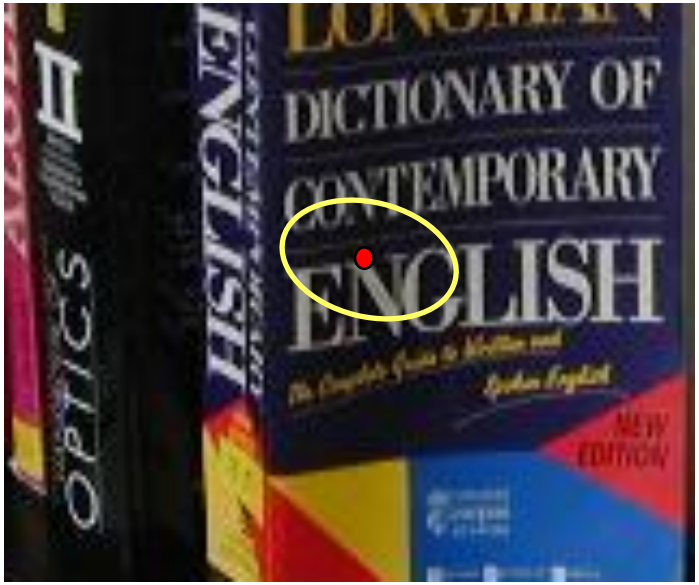
Allowing the diameters to differ helps somewhat, but contents still quite different (look at the top regions in the circles)

The need for variable patch shape



Using ellipses works much better: the circle on the left transforms into an ellipse under the affine transf. between the local view change

The need for variable patch shape



The important thing is to achieve such change in patch shape without having to compare the images, i.e. this should happen on the basis of information in one

The need for variable patch shape

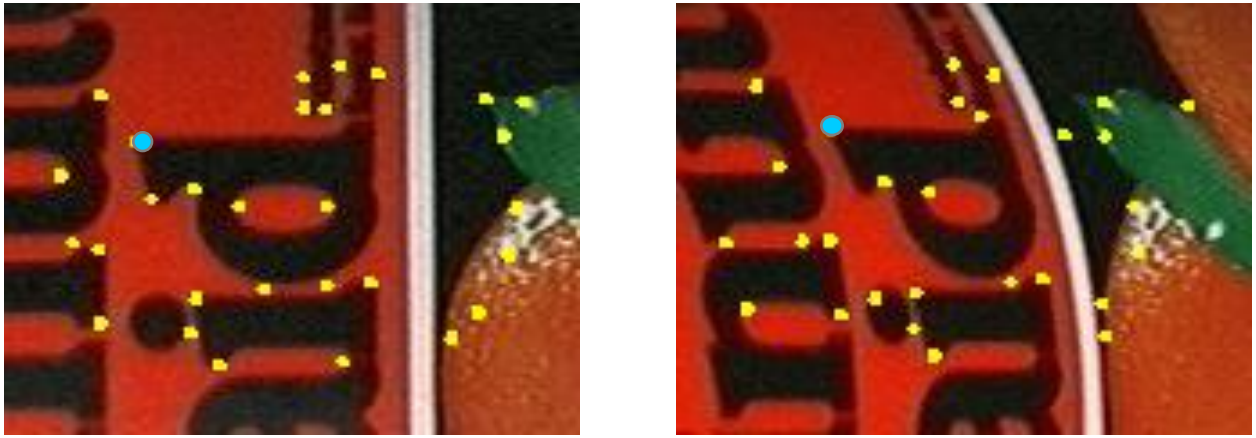
An image that we will use as test case…

Note the global perspective/projective distortion!

Example: parallelogram next to edge corner

Example: parallelogram next to edge corner

1. Harris corner detection

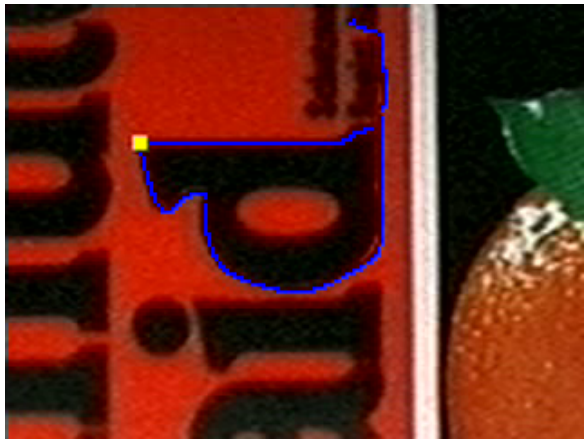


Example: parallelogram next to edge corner

2. (Canny) edge detection

Example: parallelogram next to edge corner

3. Evaluation relative affine invariant parameter along two edges

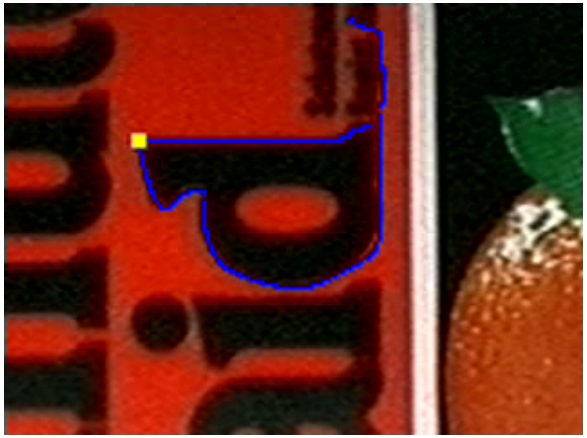


$$l_i = \int abs(|\, p_i^{(1)}(s_i) \quad p - p_i(s_i)\,|) ds_i$$

Letting 2 points evolve away from the corner in opposite directions, such that the above expression is the same for both, yields in the 2 images to corresponding parallelograms

Example: parallelogram next to edge corner

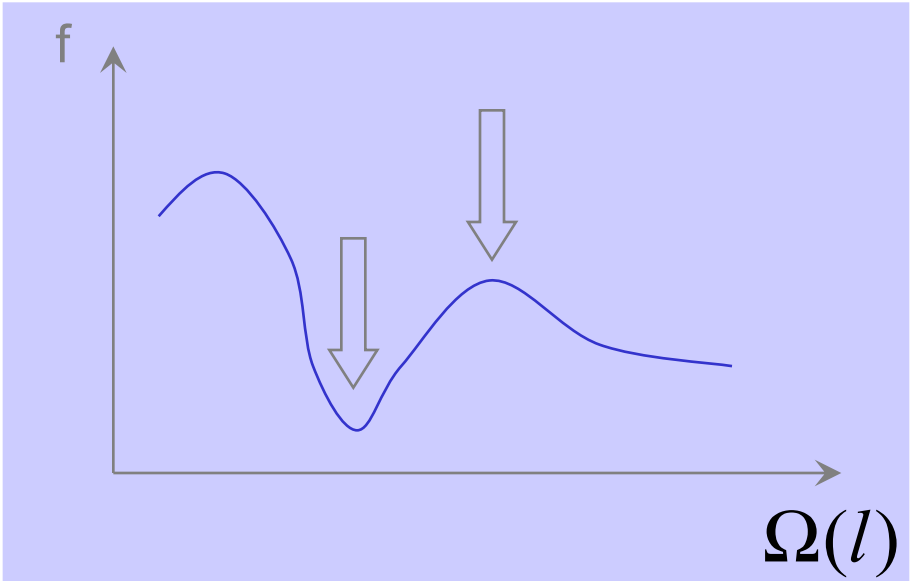
4. Construct 1-dimensional family of parallelogram shaped regions



Example: parallelogram next to edge corner

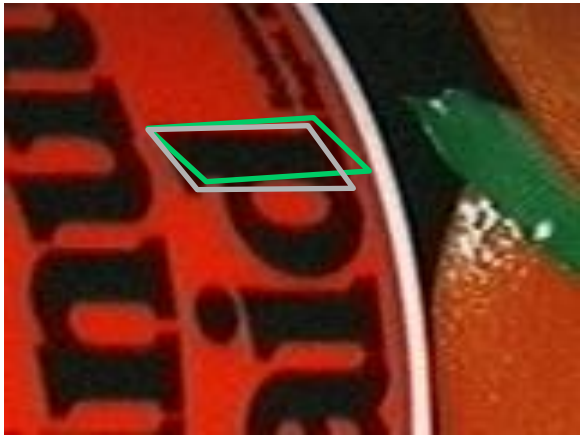
5. Select parallelograms based on invariant extrema of function

For instance:
extrema of average value of a color band within the patch



Example: parallelogram next to edge corner

5. Select parallelograms based on local extrema of invariant function

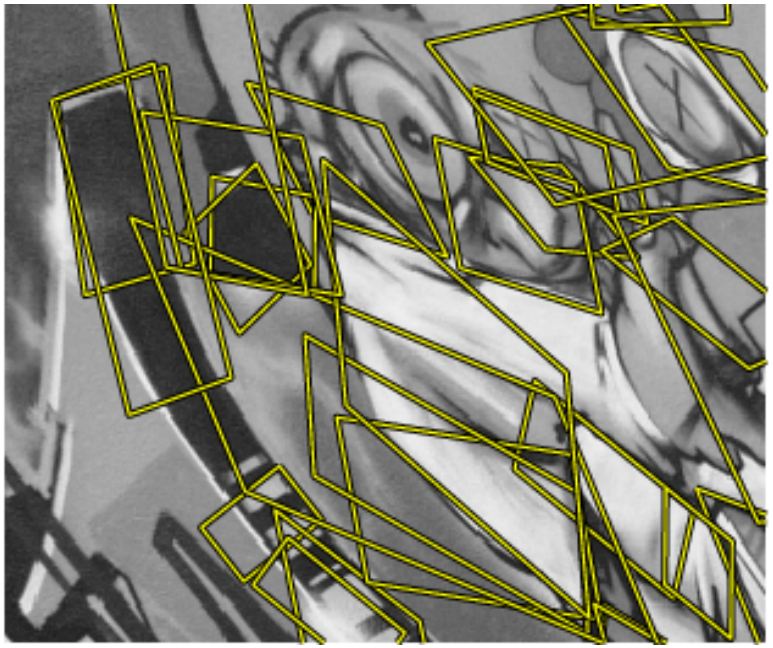


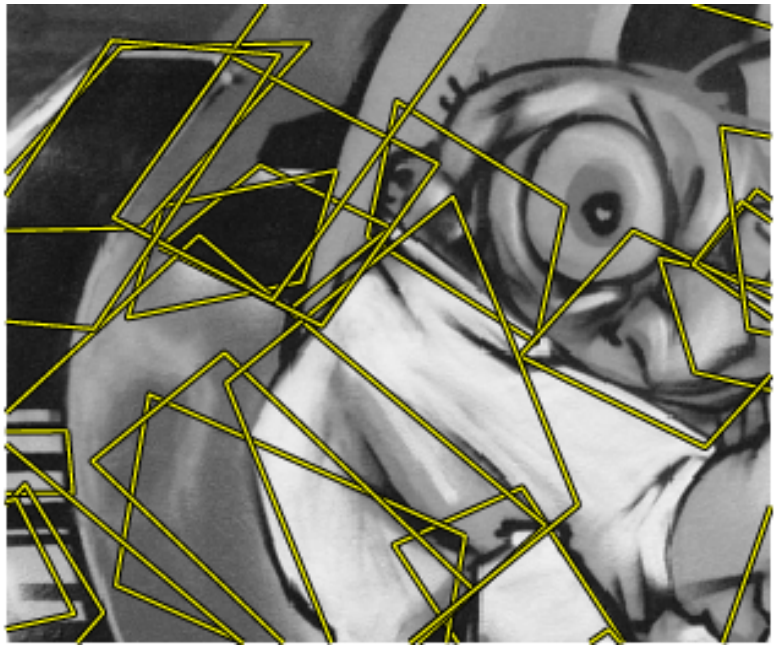
Example: parallelogram next to edge corner

Augmented Reality where an interest point's patch is filled with the texture `BIWI'. The stability shows how well the patch adapts to the viewpoint

PART 4: EXAMPLES

Example 1: edge corners + affine moments





Example 1: edge corners + affine moments

6. Describe the pattern within the parallelogram with affine invariant moments

Geometric/photometric moment invariants based on generalised colour moments:

$$M_{p,q}^{a,b,c} = \int x^p y^q \, r^a(x,y) \, g^b(x,y) \, b^c(x,y) \, dxdy$$

M_{pq}^{abc} *are not invariant themselves, need to be combined*

Example 1: edge corners + affine moments

6. Describe the pattern within the parallelogram with affine invariant moments

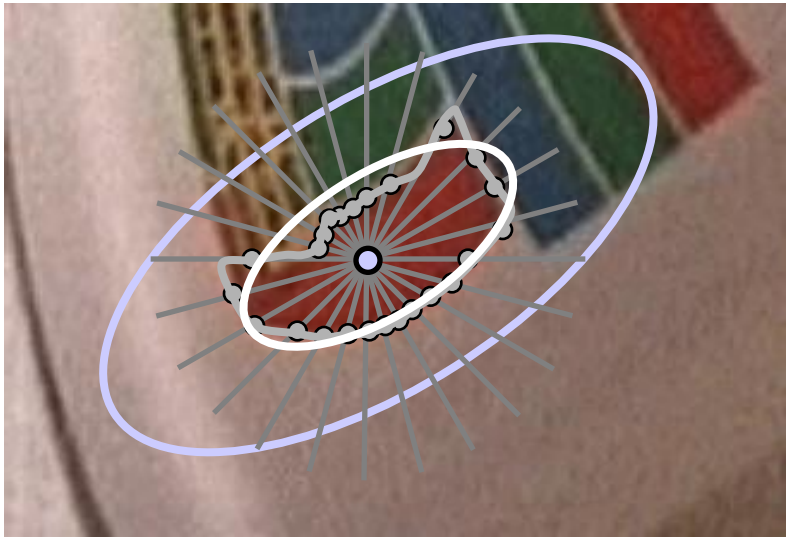
Example moment invariant from only 2 color bands:

$$D_{02} = \frac{\left[M_{00}^{11} M_{00}^{00} - M_{00}^{10} M_{00}^{01}\right]^2}{\left[M_{00}^{20} M_{00}^{00} - (M_{00}^{10})^2\right]\left[M_{00}^{02} M_{00}^{00} - (M_{00}^{01})^2\right]}$$

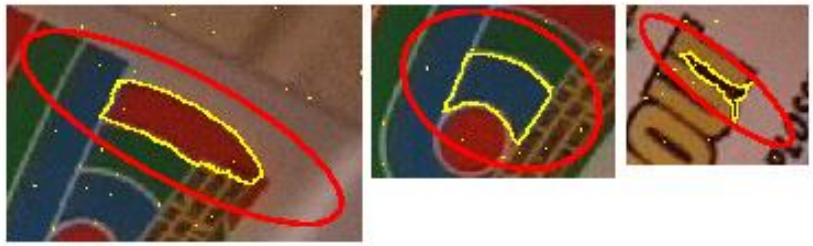
Ex. 2: intensity extrema + affine moments

1. Search intensity extrema
2. Observe intensity profile along rays
3. Search maximum of invariant function f(t) along each ray
4. Connect local maxima
5. Fit ellipse
6. Double ellipse size
7. Describe elliptical patch with moment invariants

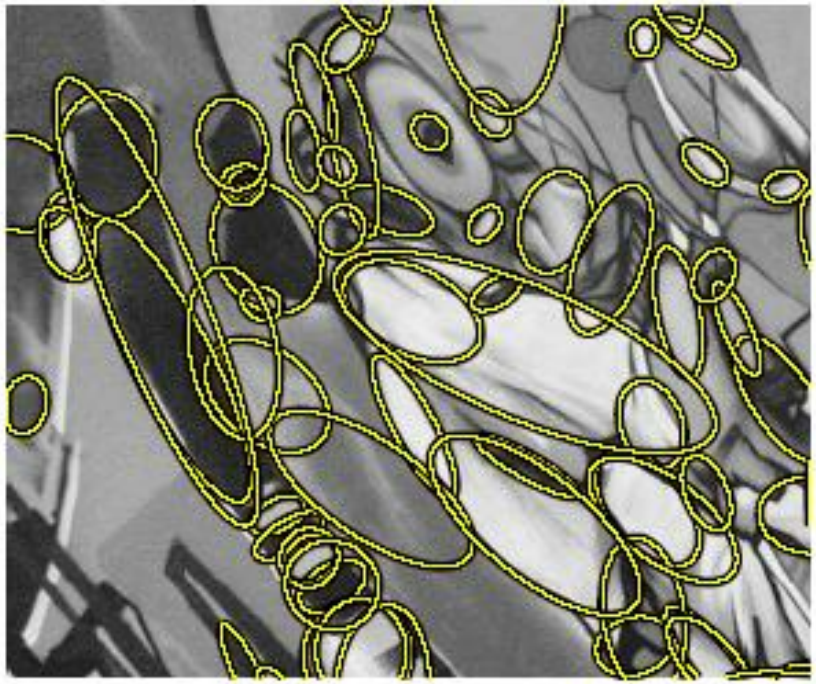
$$f(t) = \frac{abs(I_0 - I)}{\max\left(\frac{\int abs(I_0 - I)dt}{t}, d\right)}$$

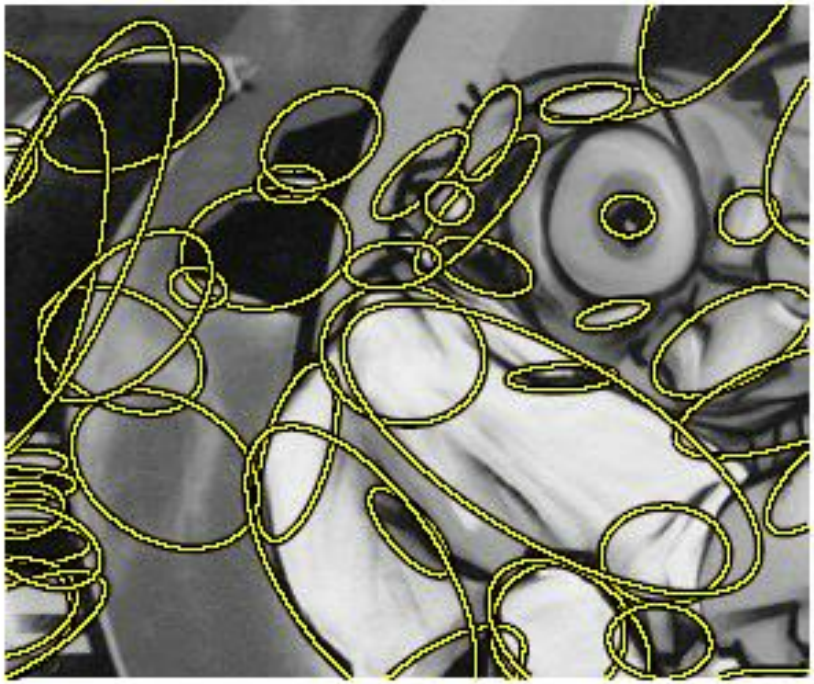


Ex. 2: intensity extrema + affine moments



Ex. 2: intensity extrema + affine moments



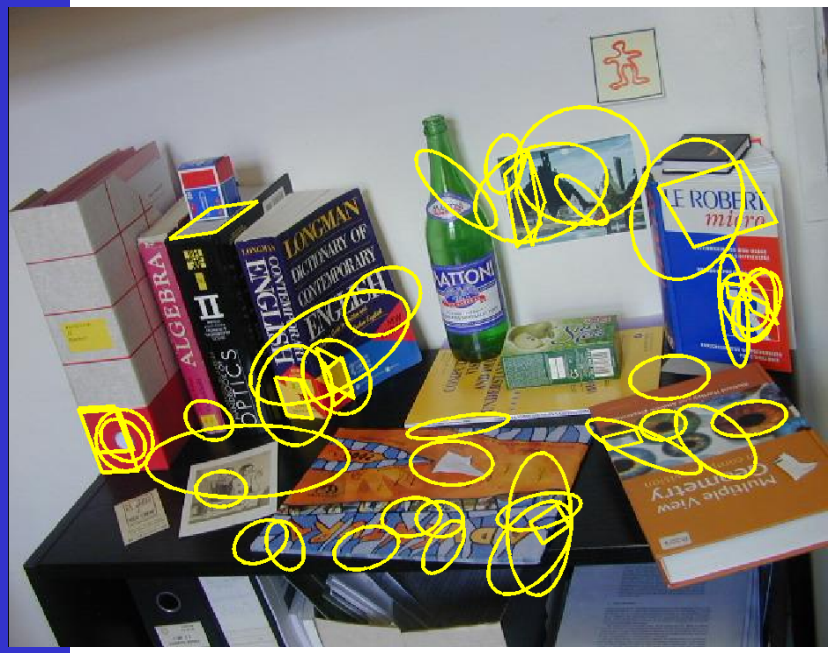


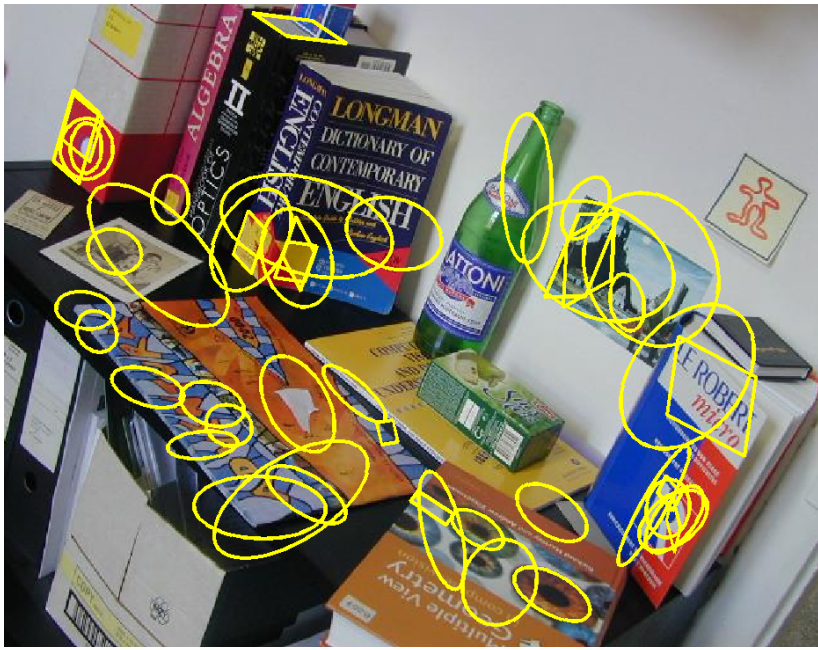
Remark

In practice different types of interest points are often combined

Wide baseline stereo matching example.

based on ex.1 and ex.2 interest points





MSER interest points

MSER = Maximally Stable Extremal Regions

- Similar to the Intensity-Based Regions we just saw
- Came later, but is more often used
- Start with intensity extremum
- Then move intensity threshold away from its value and watch the super/sub-threshold region grow
- Take regions at thresholds where the growth is slowest (happens when region is bounded by strong edges)

MSER

MSER

MSER

MSER

MSER

Computer Vision

MSER

MSER

MSER

MSER

MSER

MSER

MSER

MSER

Computer Vision

MSER

Computer Vision

MSER

Computer Vision

MSER

MSER

- *Extremal region*: region such that

$$\forall p \in Q, \forall q \in \delta Q : \begin{matrix} I(p) > I(q) \\ I(p) < I(q) \end{matrix}$$

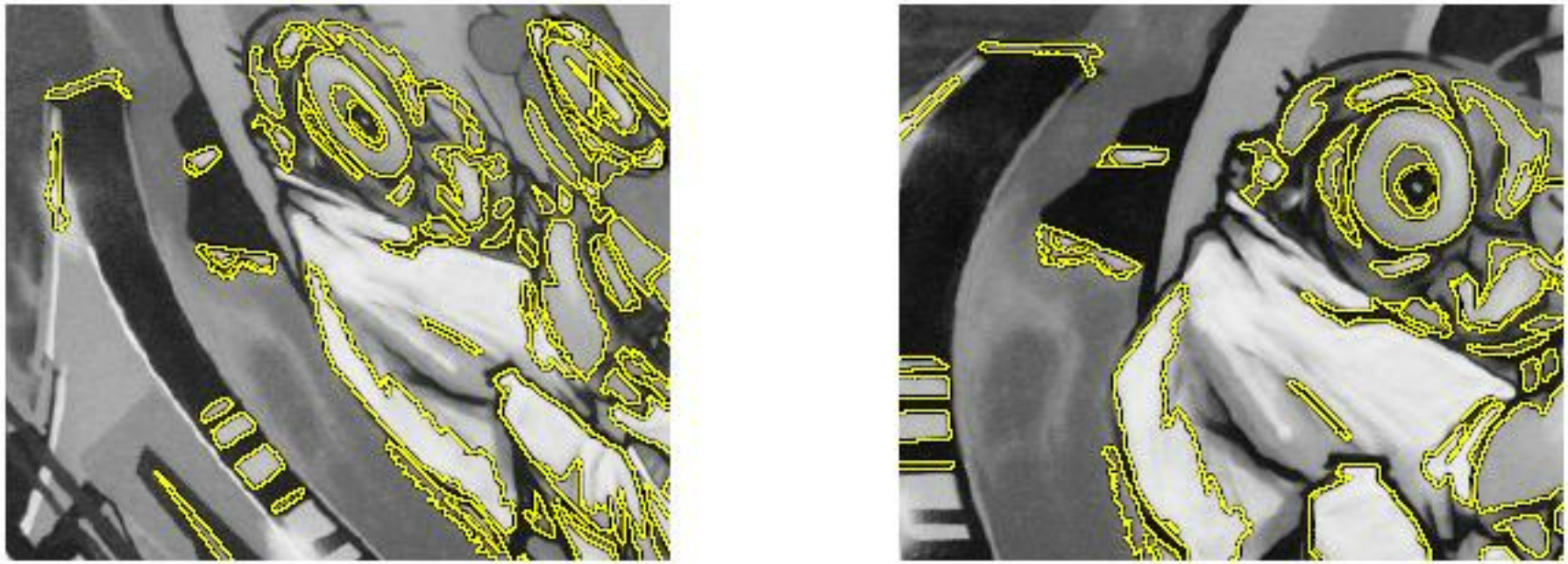
- Order regions, following increasing or decreasing threshold

$$Q_1 \subset \ldots \subset Q_i \subset Q_{i+1} \subset \ldots Q_n$$

- *Maximally Stable Extremal Region*: local minimum of

$$q(i) = |Q_{i+\Delta} \setminus Q_{i-\Delta}| / Q_i$$

MSER



(a) **MSER**

Example 3: SIFT

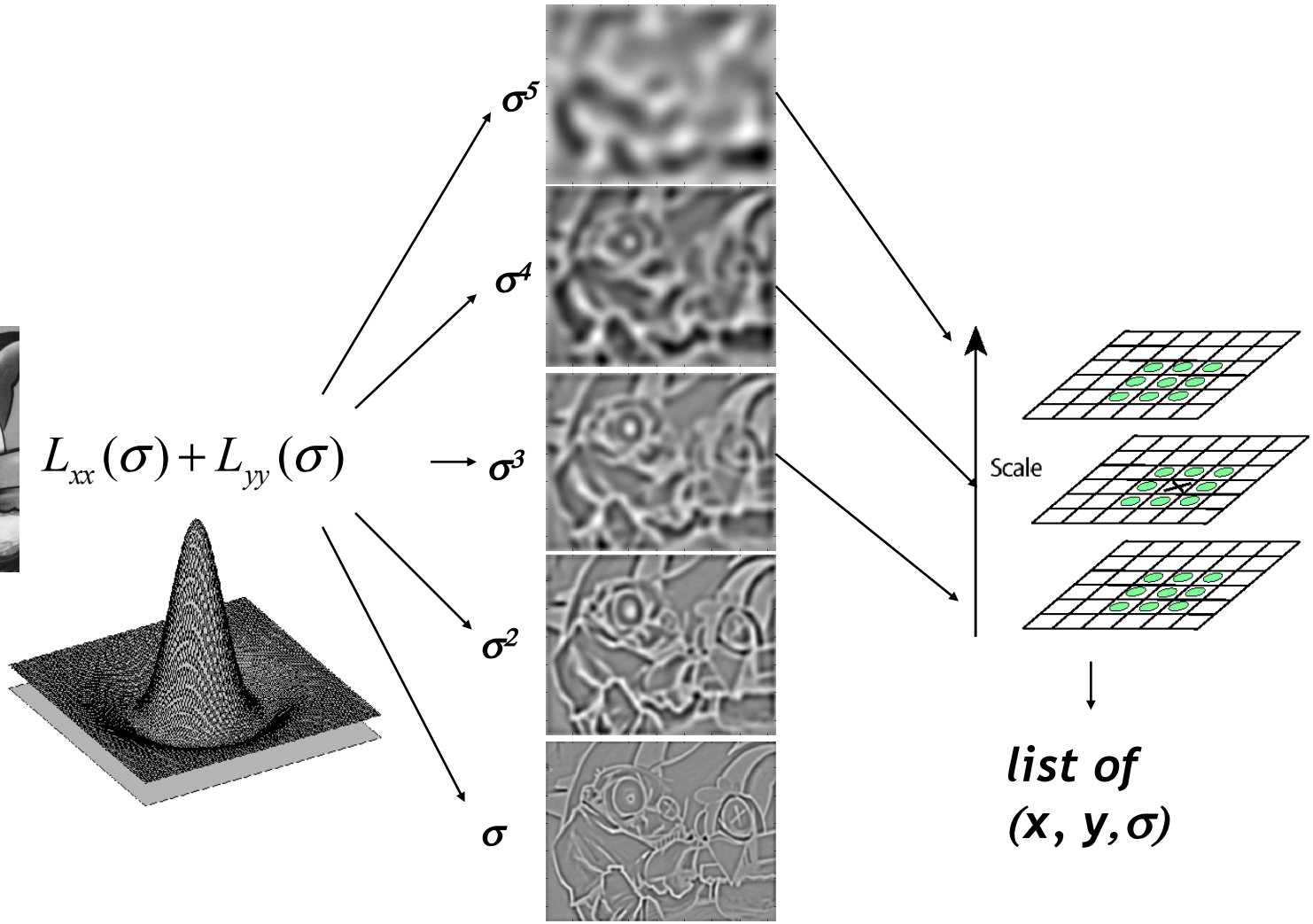
SIFT = Scale-Invariant Feature Transform

SIFT, developed by David Lowe (Un. British Columbia, Canada), is a carefully crafted interest point detector + descriptor, based on intensity gradients (cf. our comment on photometric invariance) and invariants under similarities, not affine like so far

Our summary is a simplified account!

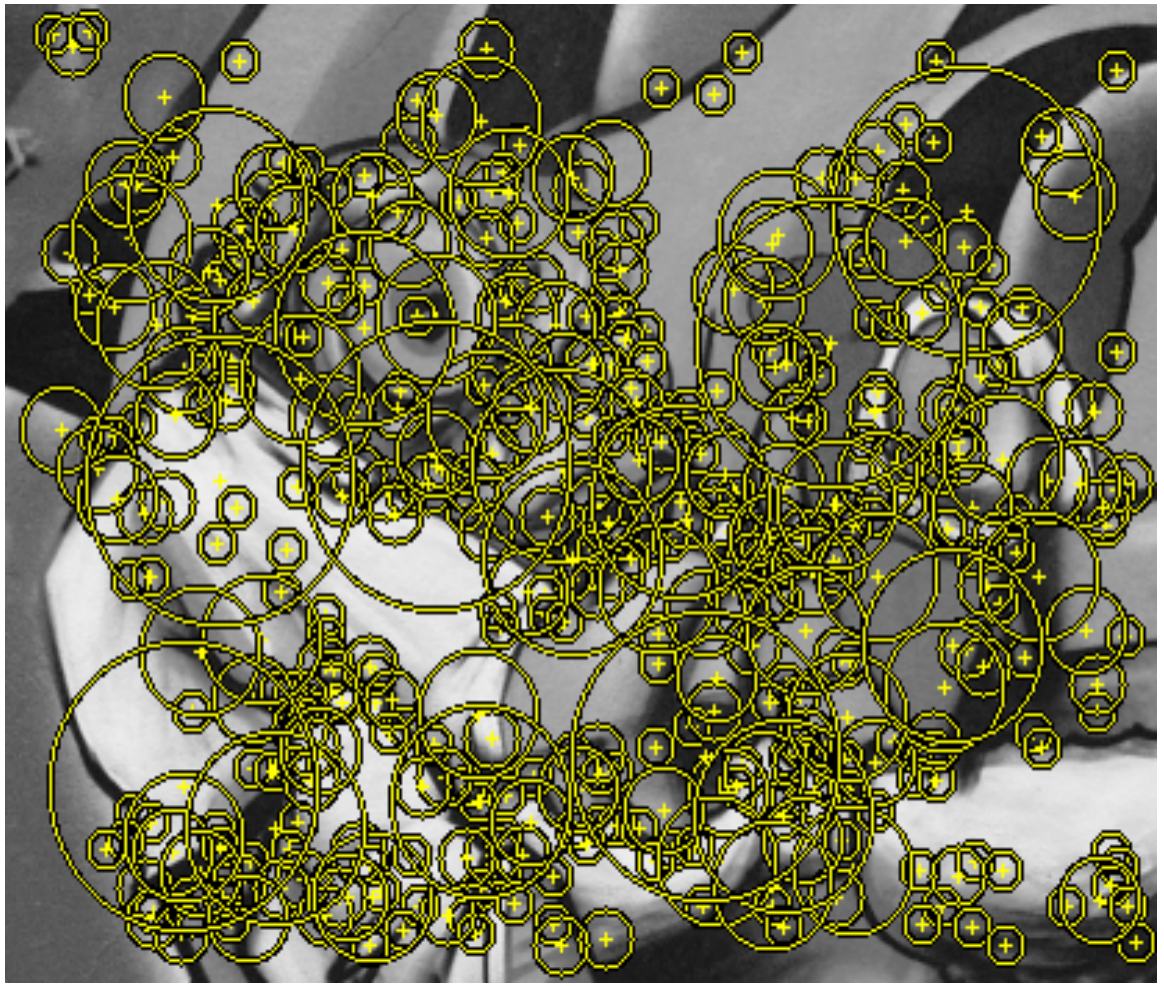
SIFT

Descriptor is based on blob detection, at several scales, i.e. local extrema of the Laplacian-of-Gaussian, or LoG



Computer Vision

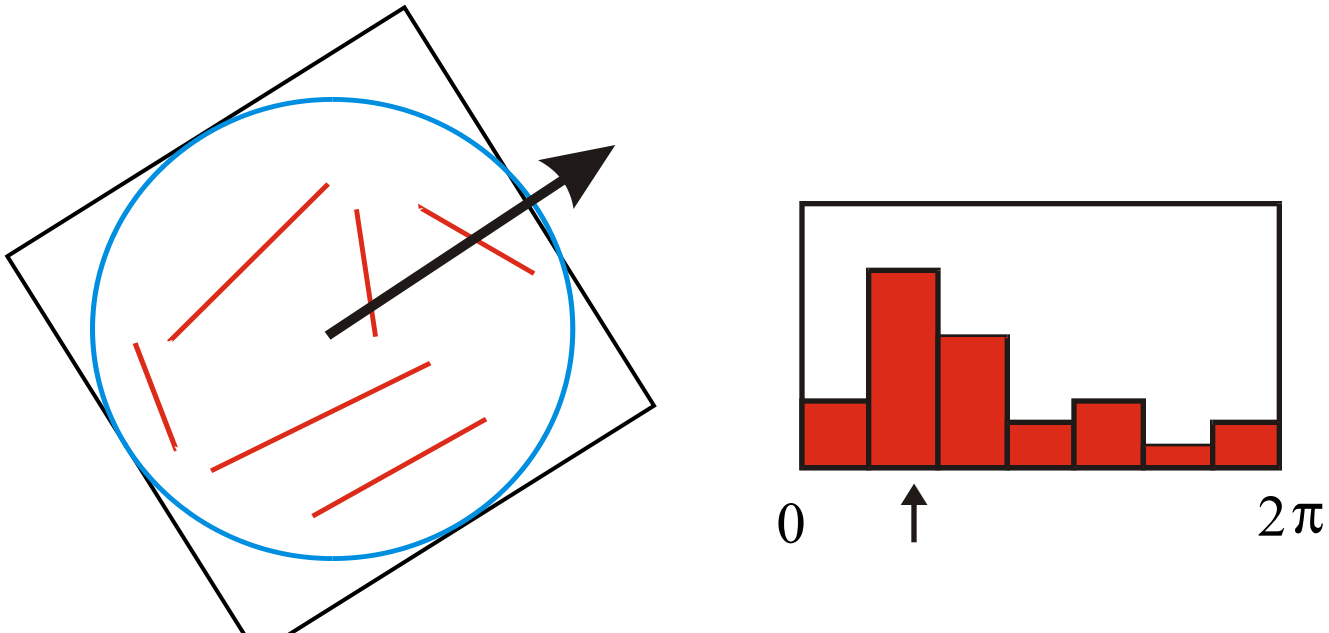
SIFT



SIFT

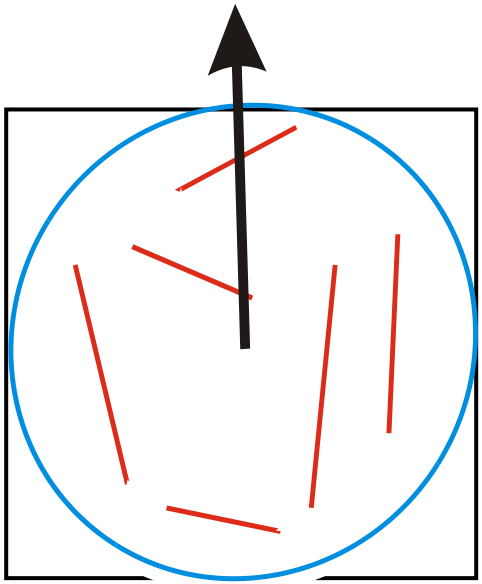
Dominant orientation selection

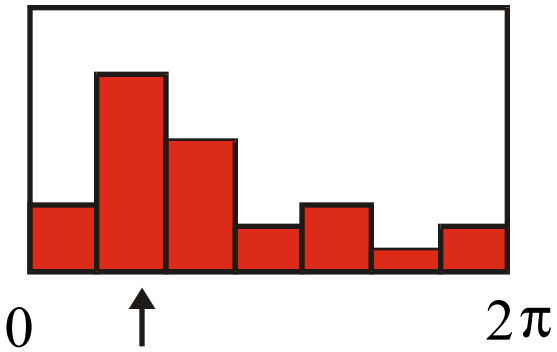
- Compute image gradients
- Build orientation histogram
- Find maximum



SIFT

- Dominant orientation selection
  - Compute image gradients
  - Build orientation histogram
  - Find maximum





SIFT

- Thresholded image gradients are sampled over a grid
- Create array of orientation histograms within blocks
- 8 orientations x 4x4 histogram array = 128 dimensions
- Apply weighting with a Gaussian located at the center
- Normalized to unit vector

(Fig. shows 2x2, actually 4x4)

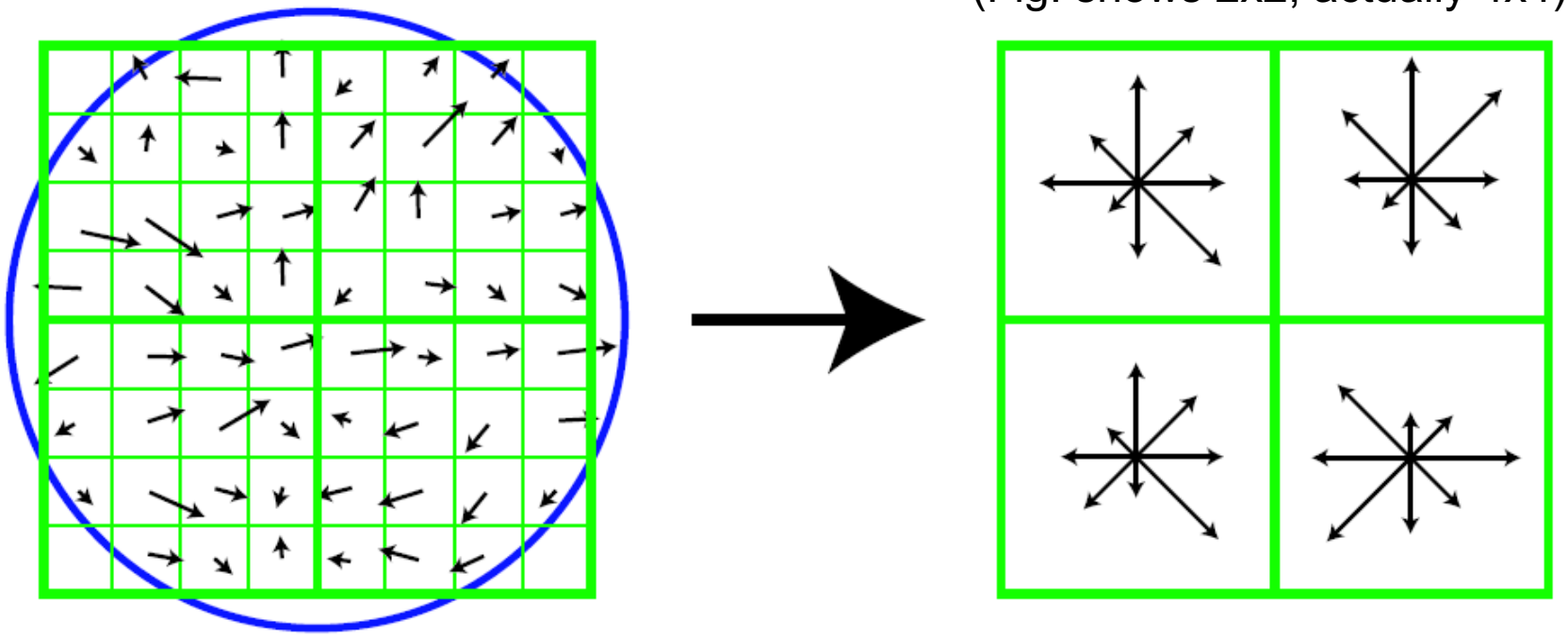
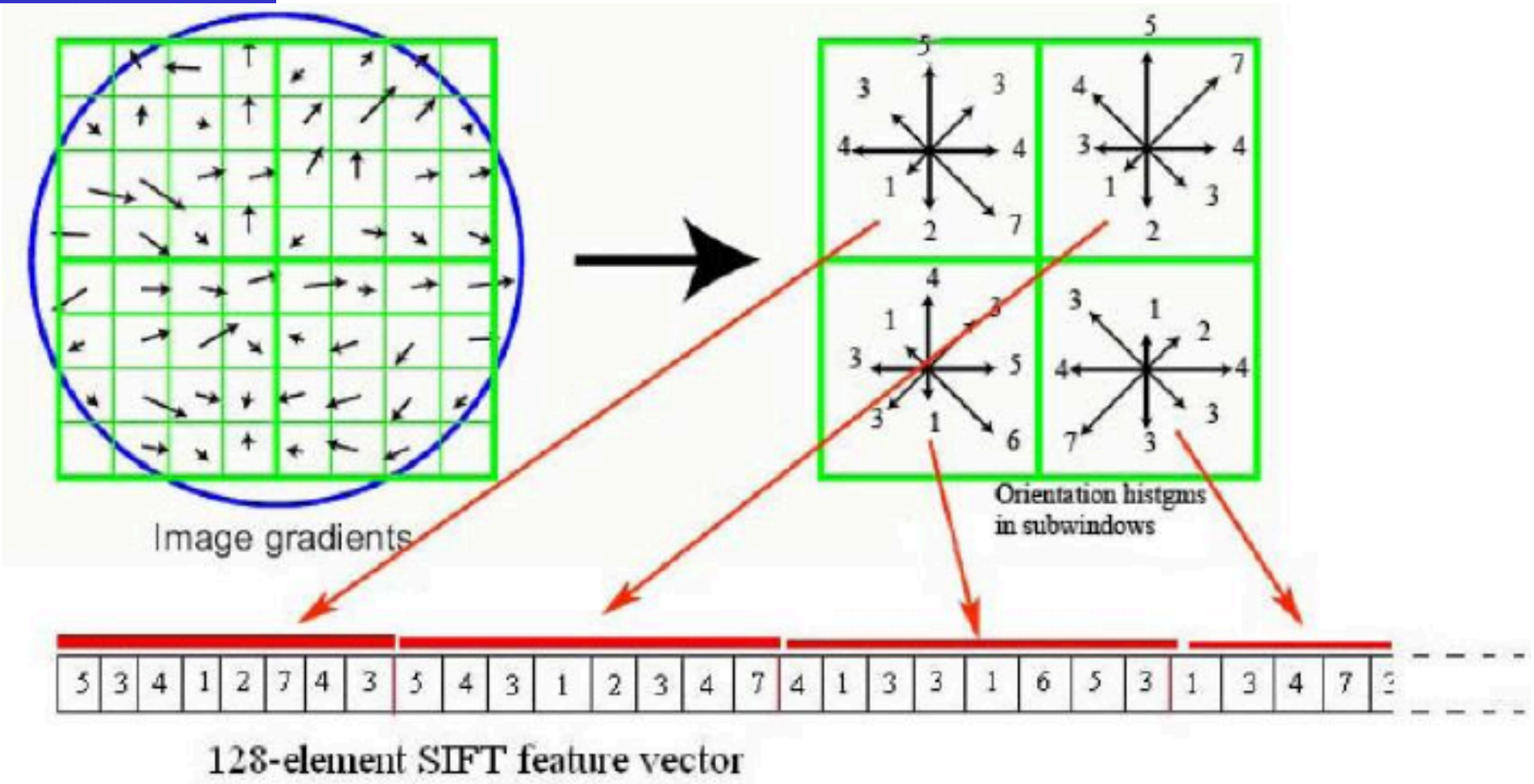


Image gradients

Keypoint descriptor

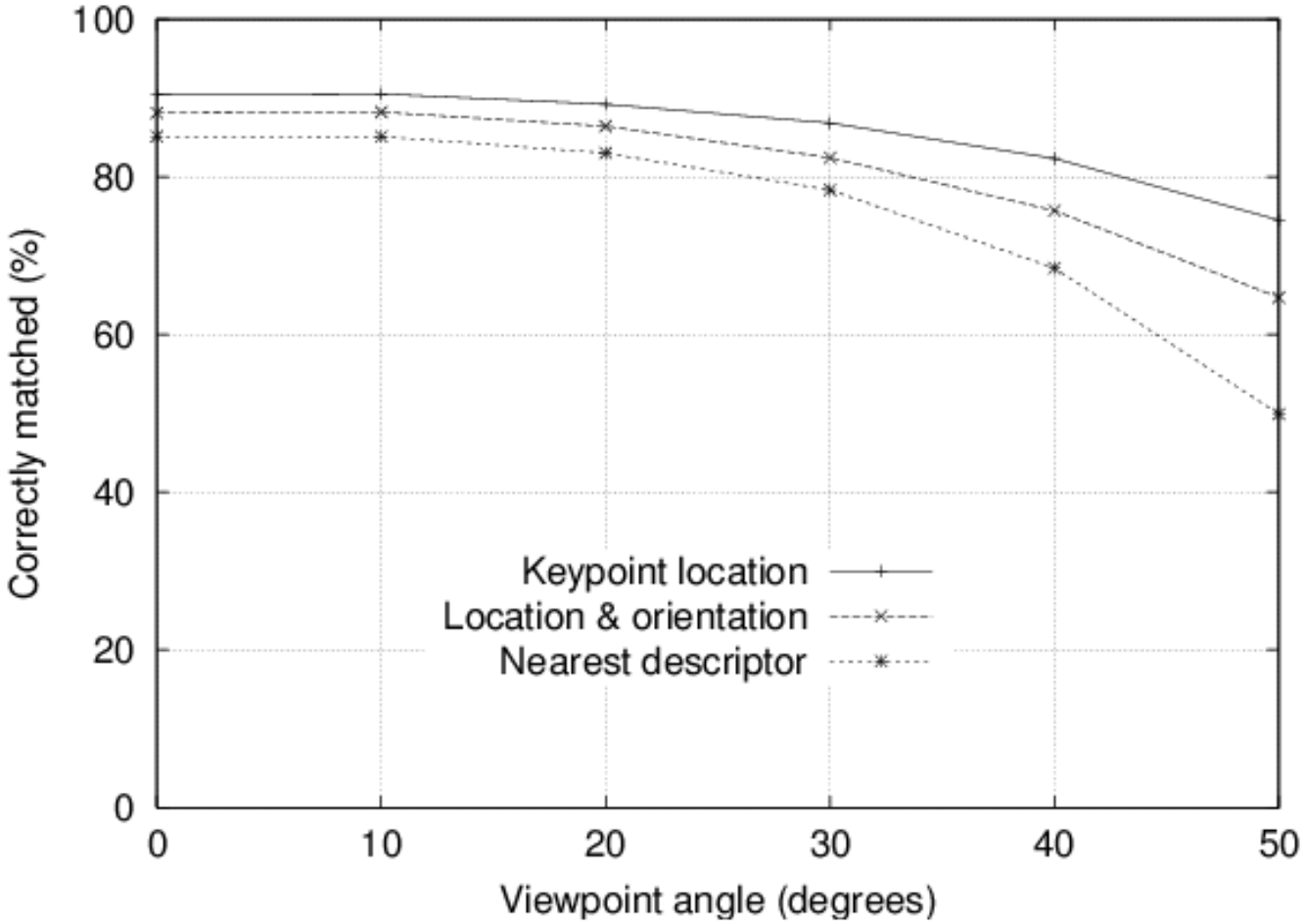
SIFT

The descriptor is a vector concatenating the cell histograms…
Its total dimension is 128



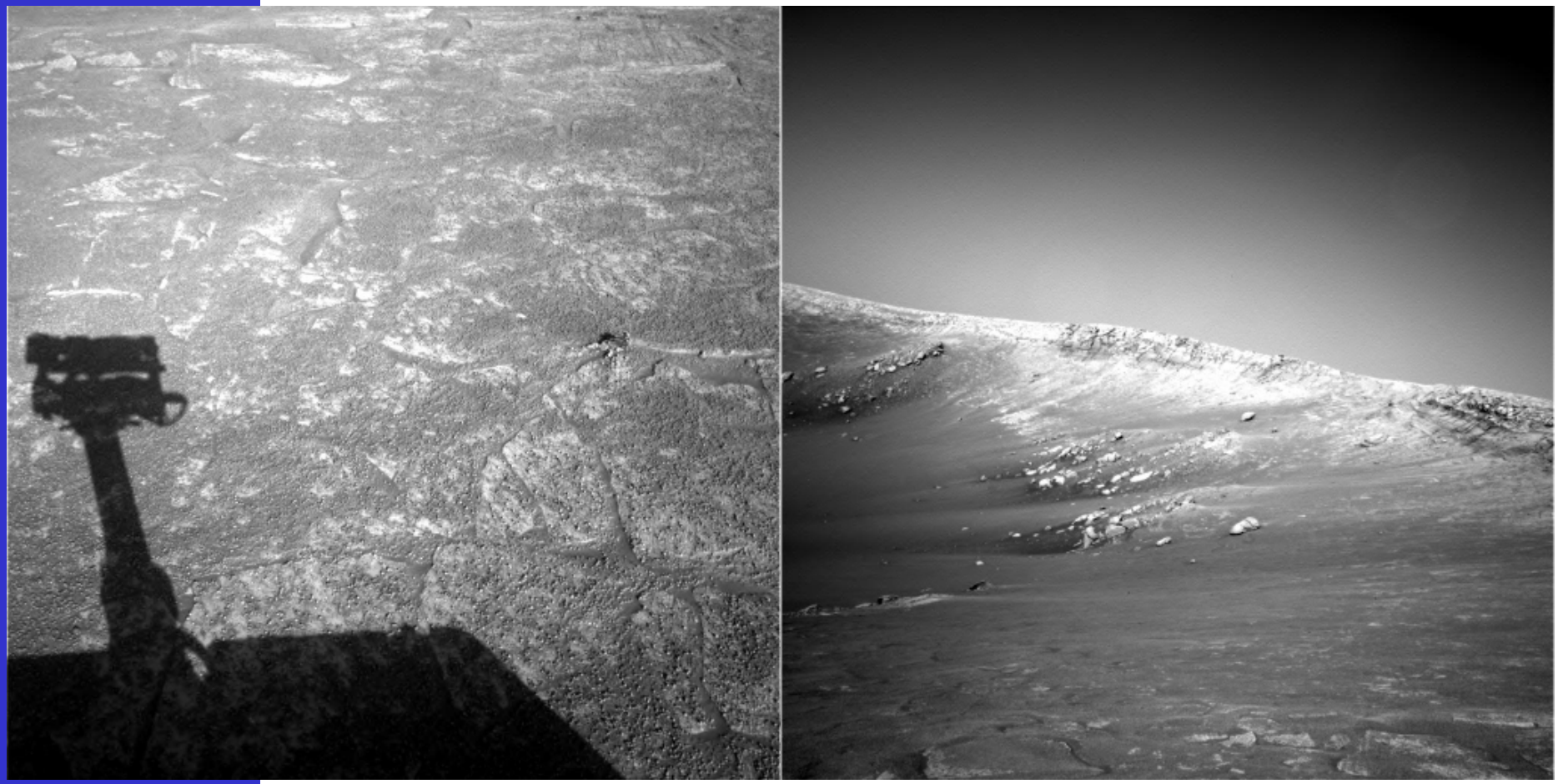
SIFT

Sensitivity to affine changes… quite good !!!



SIFT

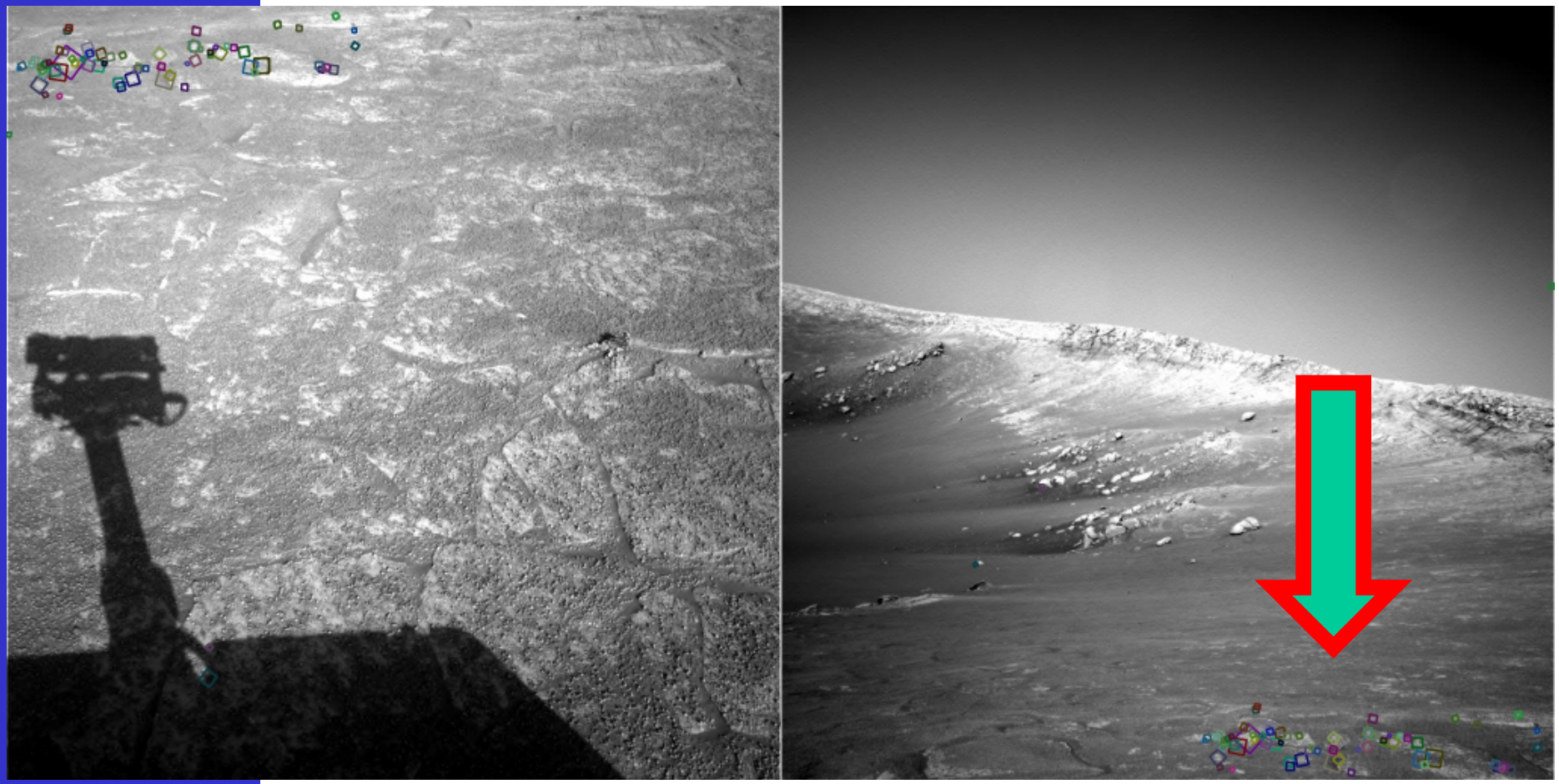
Can you find correspondences ?



NASA Mars Rover images

SIFT

SIFT can…

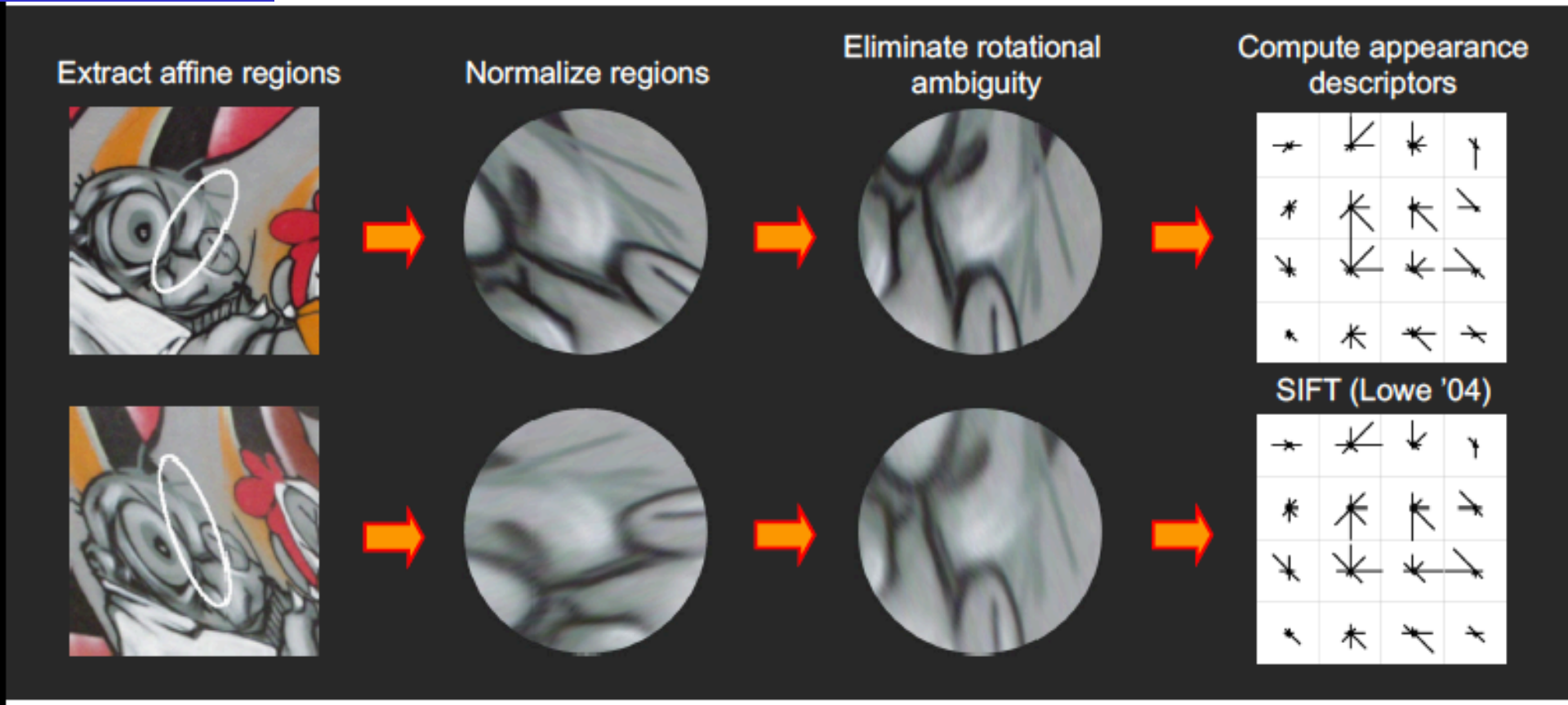


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

SIFT

NB: detectors and descriptors can be mixed, through normalization

e.g. create an affinely invariant region (affinely inv. detector), then describe content with SIFT:



SURF efficient alternative to SIFT

Application: robot navigation

SURF

- Robot equipped with omnidirectional camera

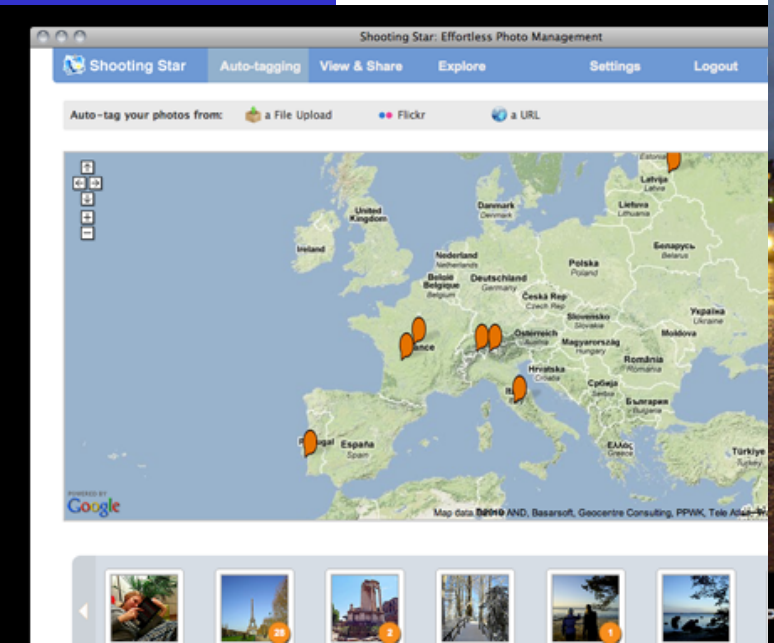
Computer Vision

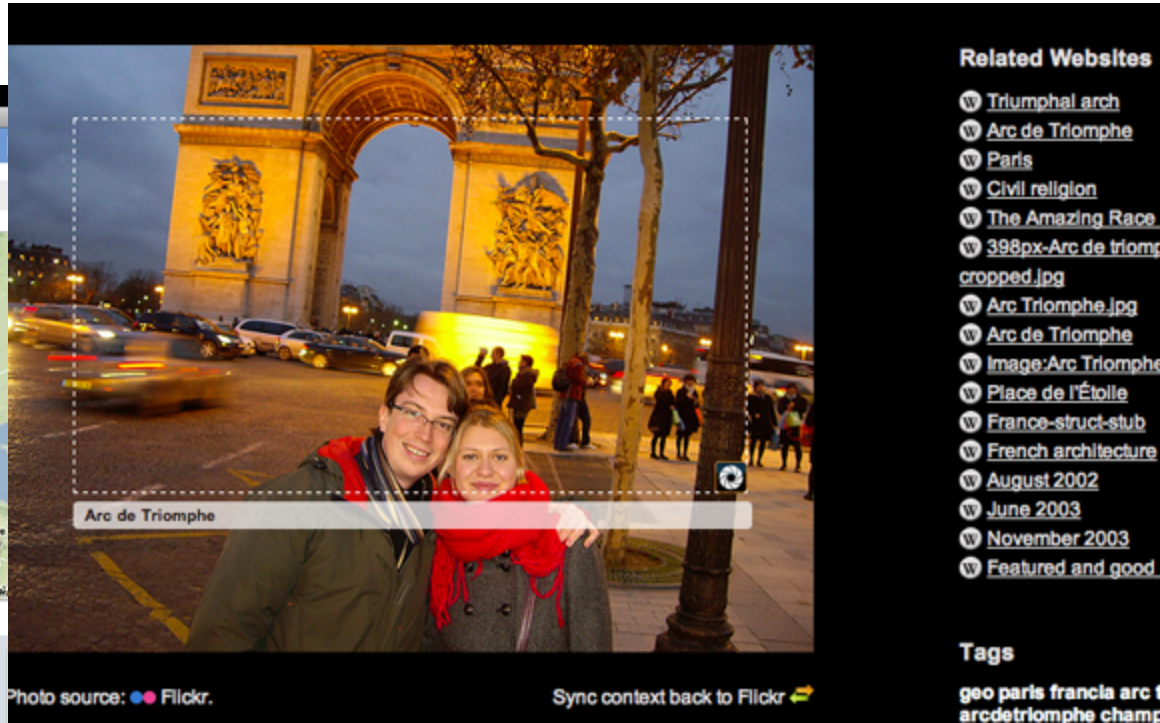
SURF

Computer Vision

SURF

Landmark, label, cover,… recognition also used for prior kooaba Déjà Vu app (then Qualcomm) for the iPhone, Android, and Symbian platforms





Automatic mosaicing

http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Computer Vision