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,
…

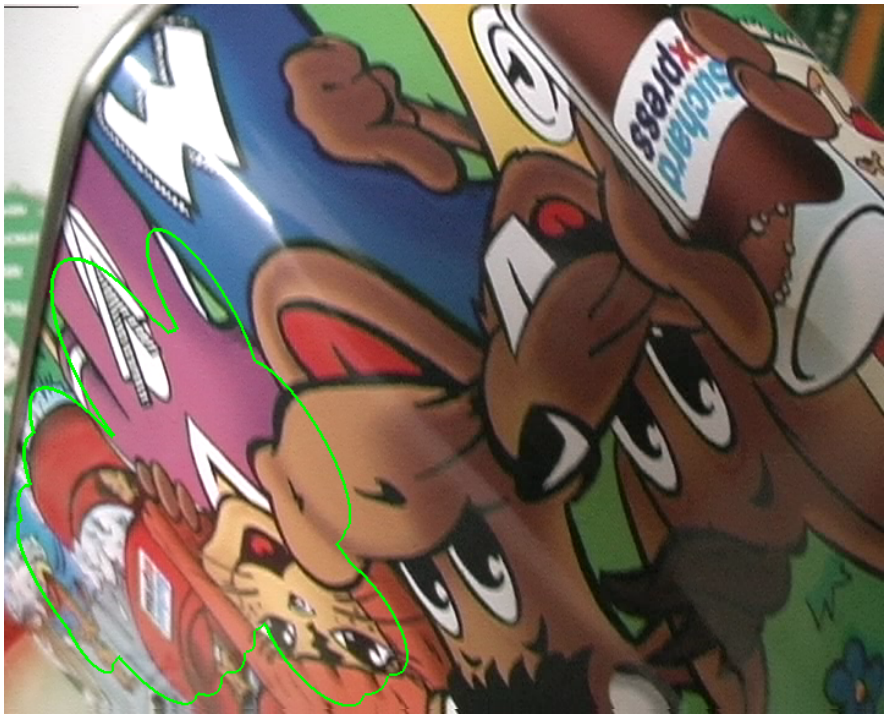
Computer Vision

Feature matching – examples for recognition

Computer Vision

Feature matching

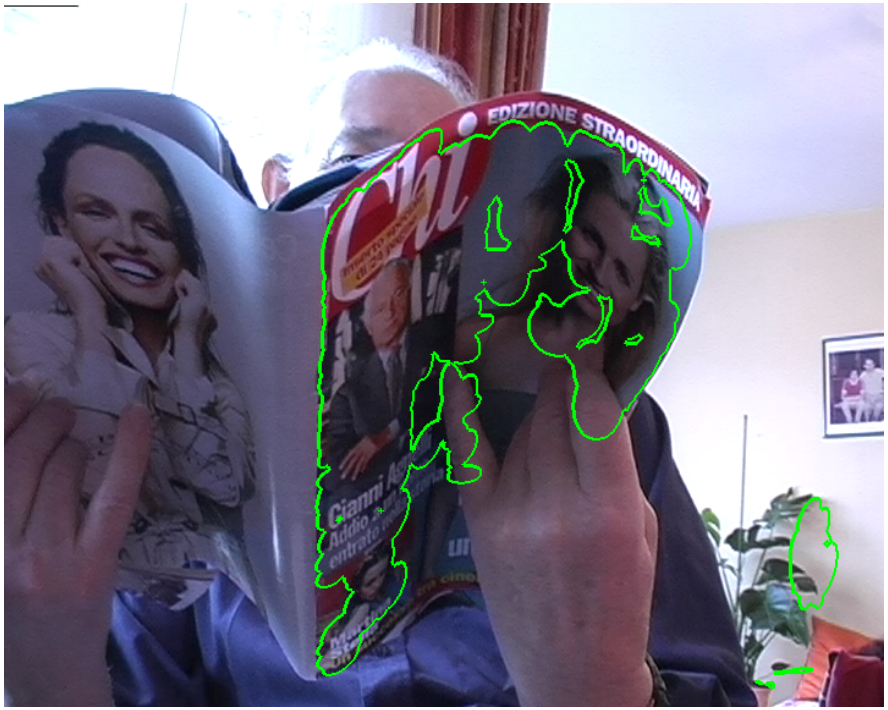
Feature matching



Large scale change, heavy occlusion

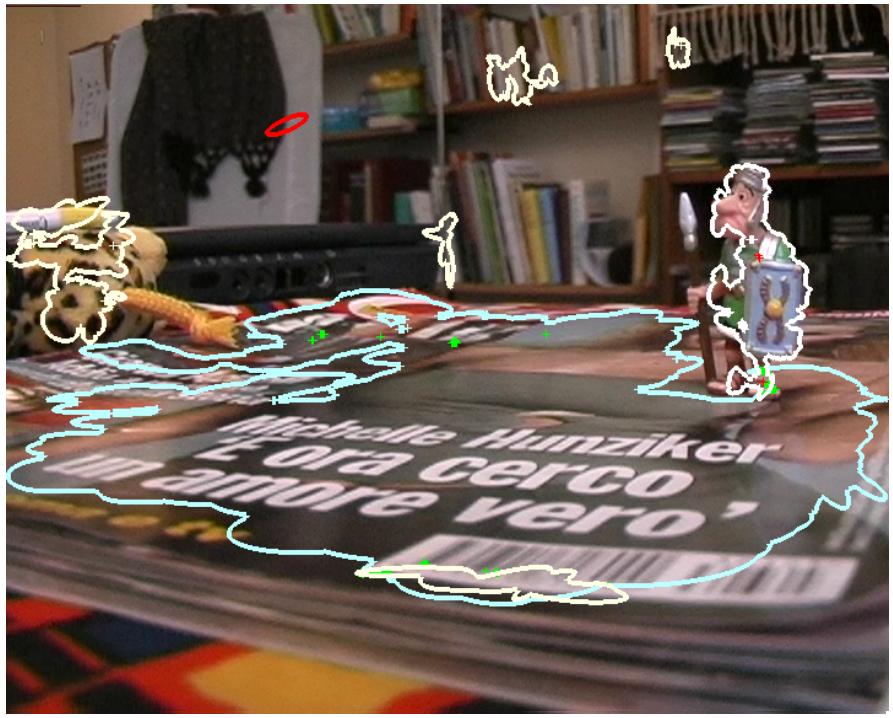
Computer Vision

Feature matching



Deformation, illumination change, occlusion

Feature matching



Large scale change, perspective deformation, extensive clutter

Feature matching

Extensive clutter, scale, occlusion, blur

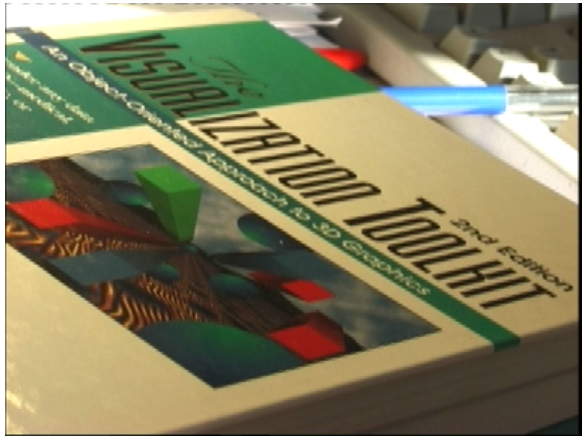
Matching: a challenging task

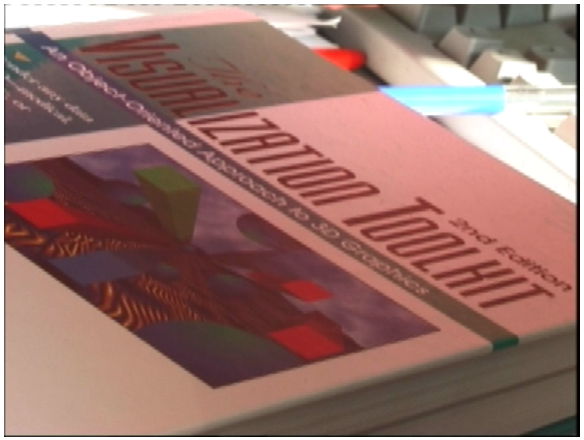
Re-iterating the difficulties with matching highlighted thus far:

- 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

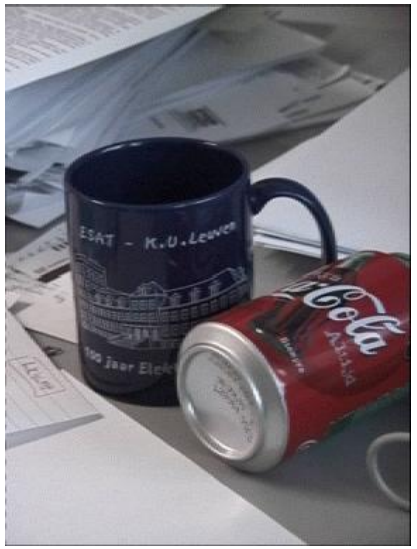
- features to deal with **large variations** in
  - Viewpoint
  - Illumination
  - Background

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:

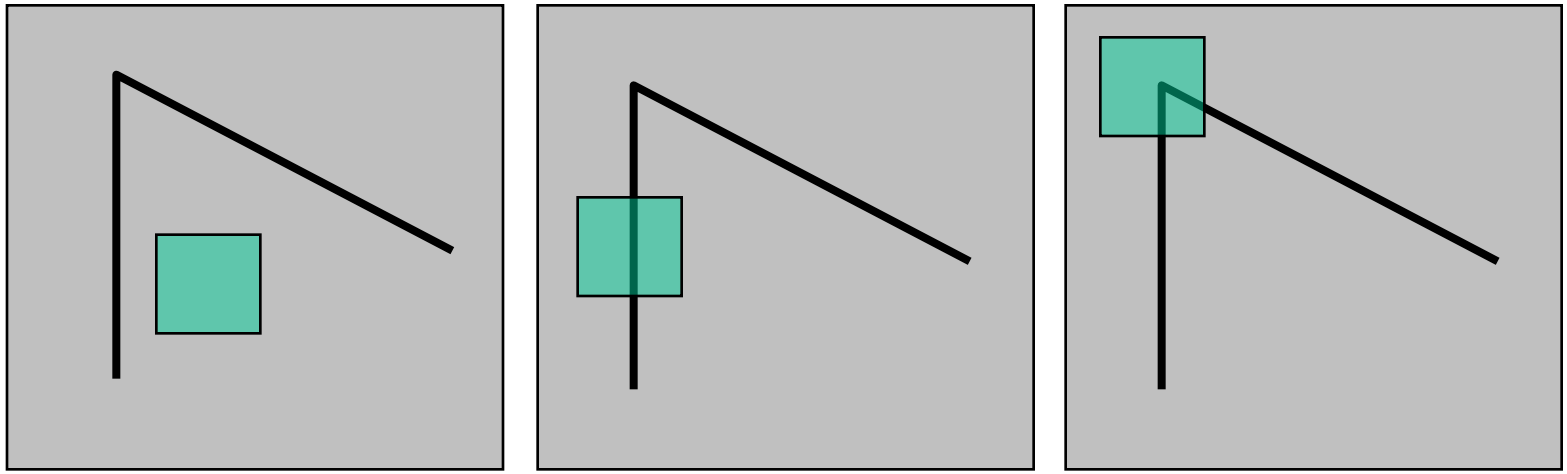
A feature should capture something *discriminative* about a well *localisable* patch of a surface

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

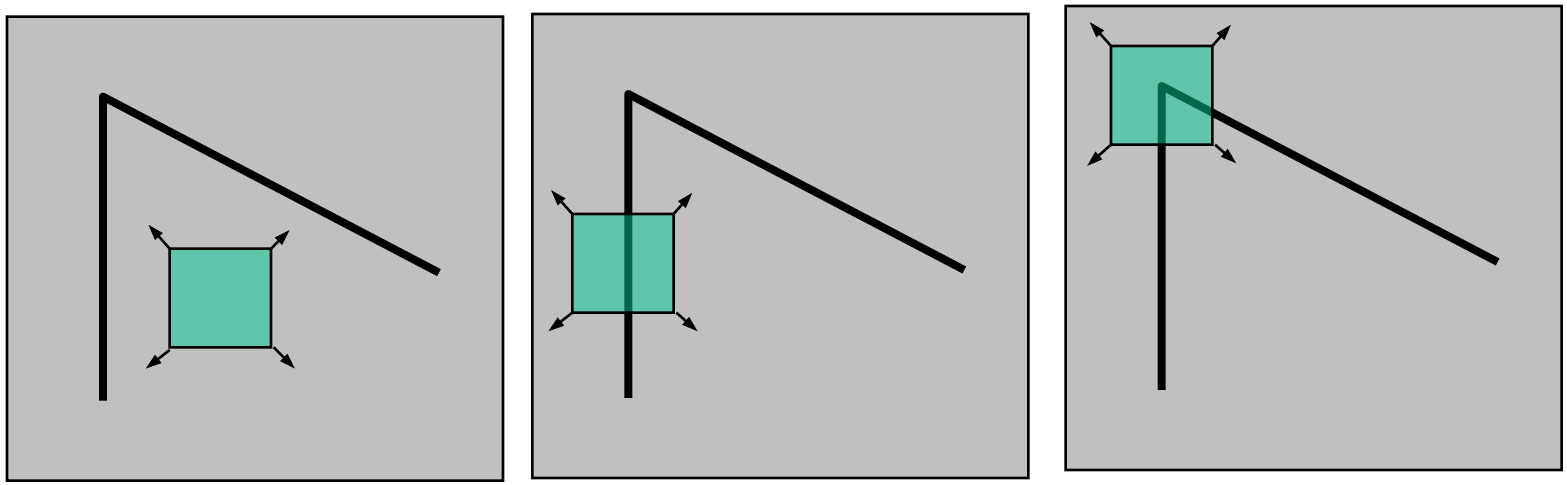
consider the pixel pattern within the green patches:



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

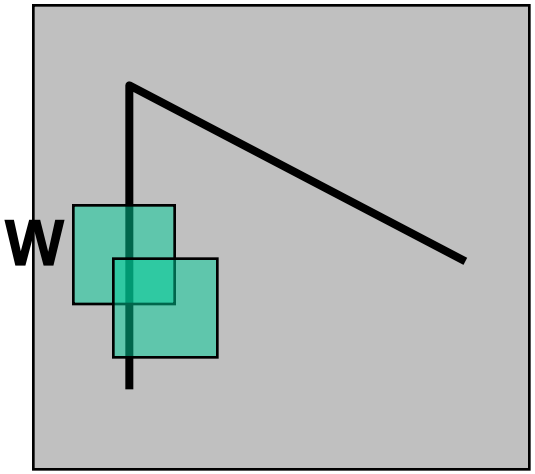
Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

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

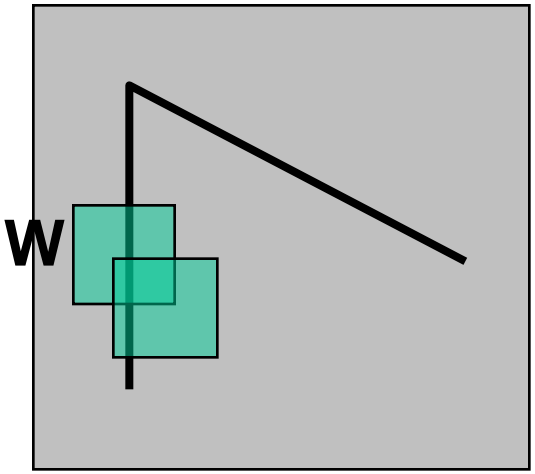
$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

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

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

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

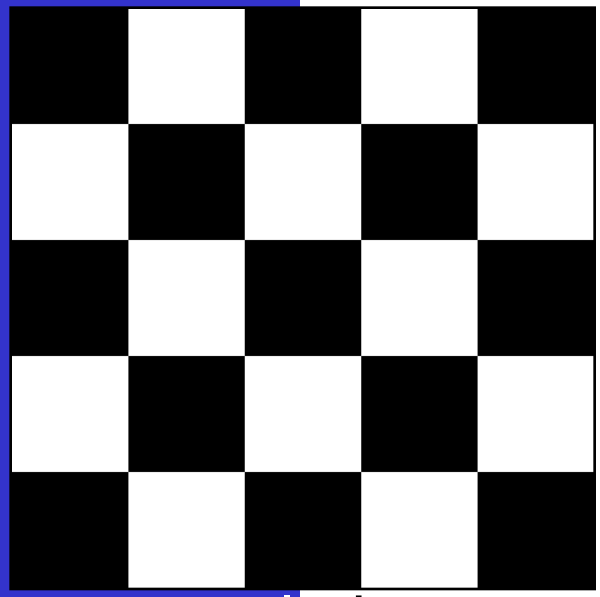
- Define shifts with the smallest and largest change (E value)
- x_+ = direction of **largest** increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of **smallest** increase in E.
- λ_- = amount of increase in direction x_+

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

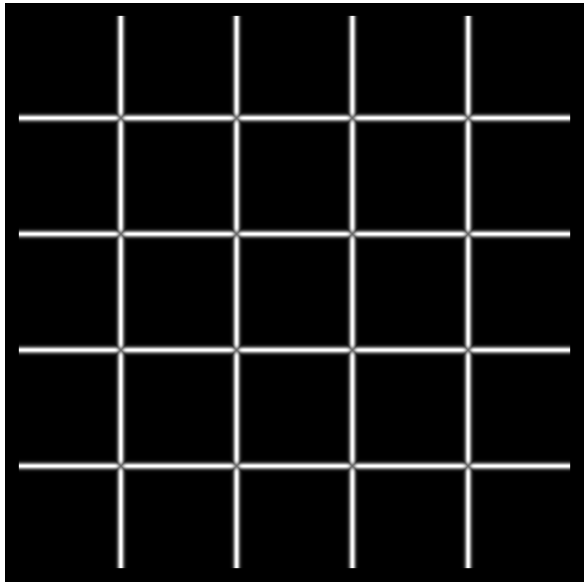
Uniqueness of a patch

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

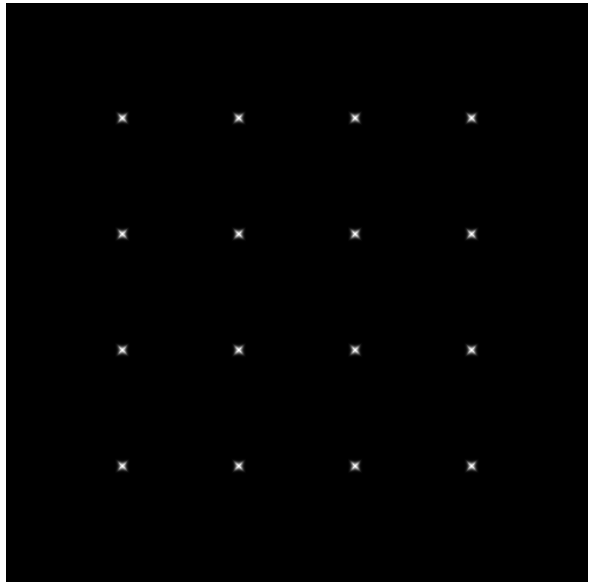
- the *minimum* of *E(u,v)* should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_-) of ***H***



example image



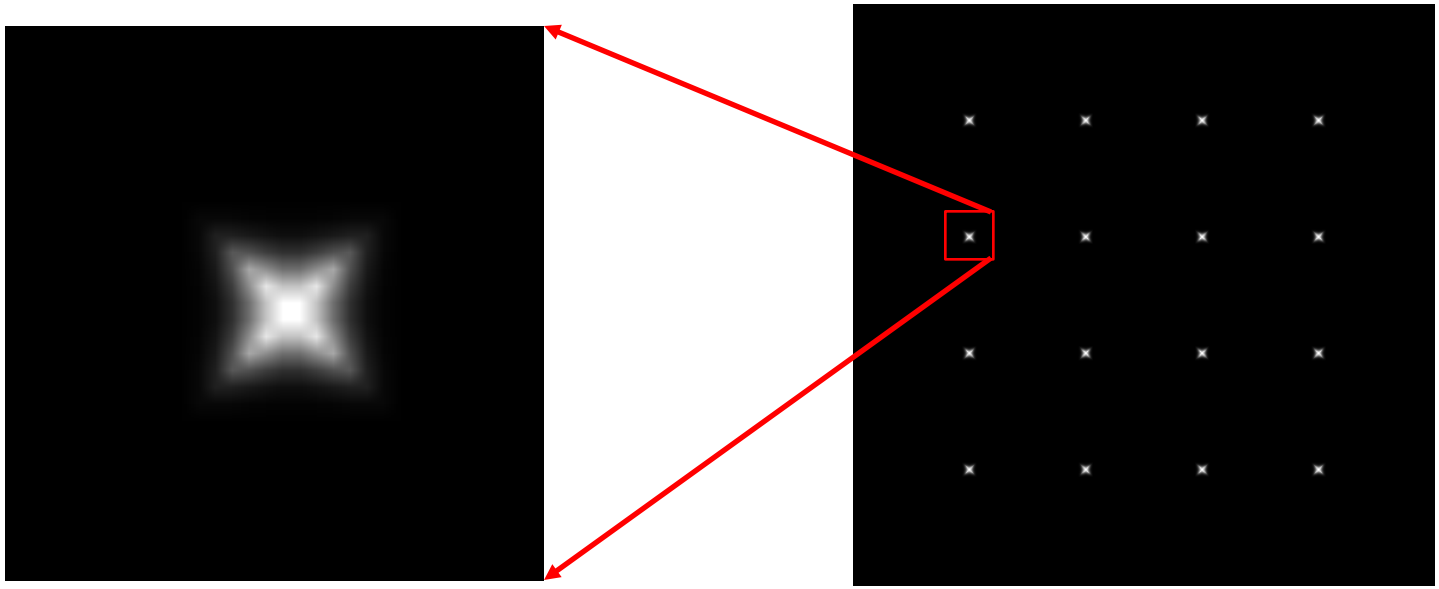
λ_+



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

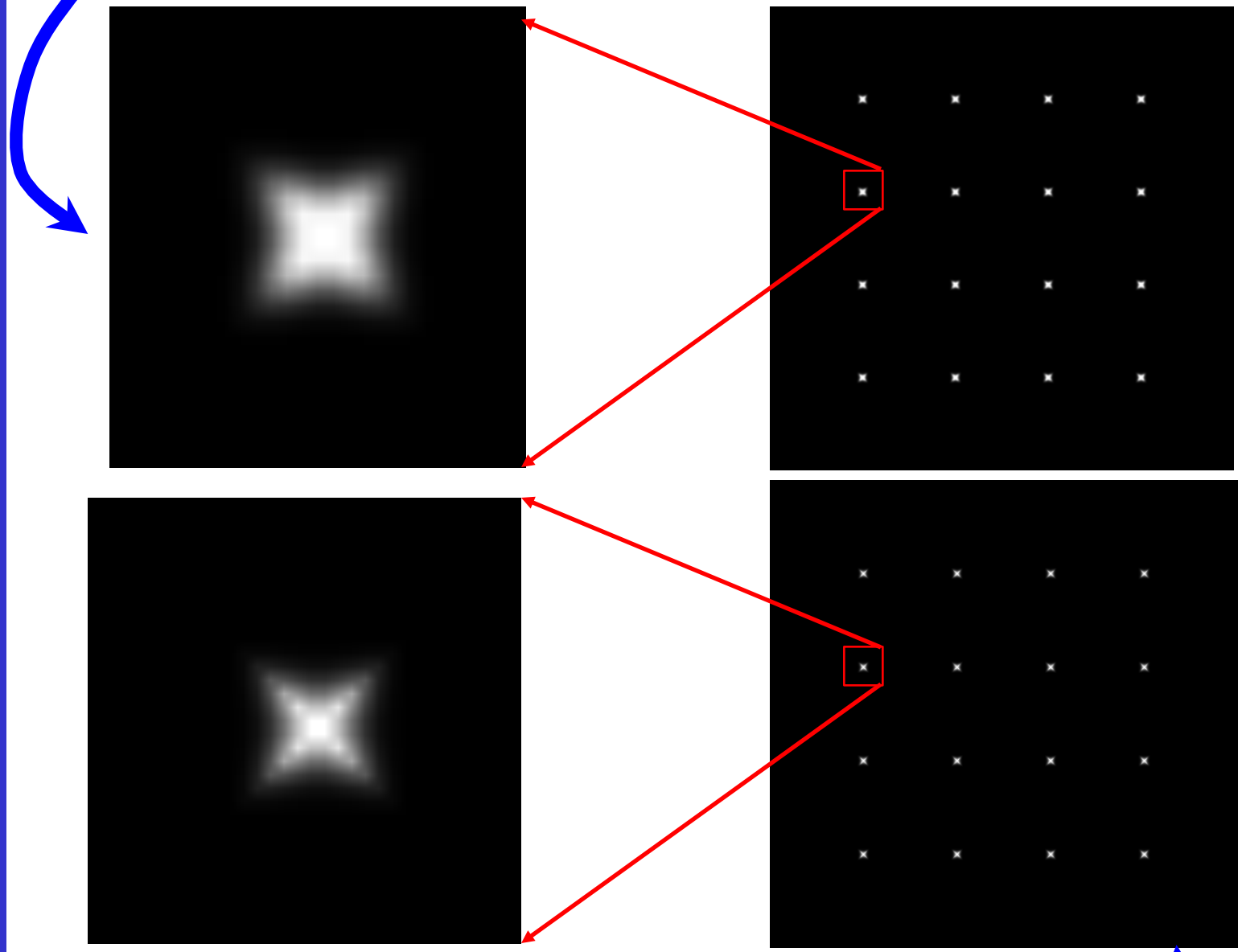
$$Determinant - k\,(Trace)^2$$

(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

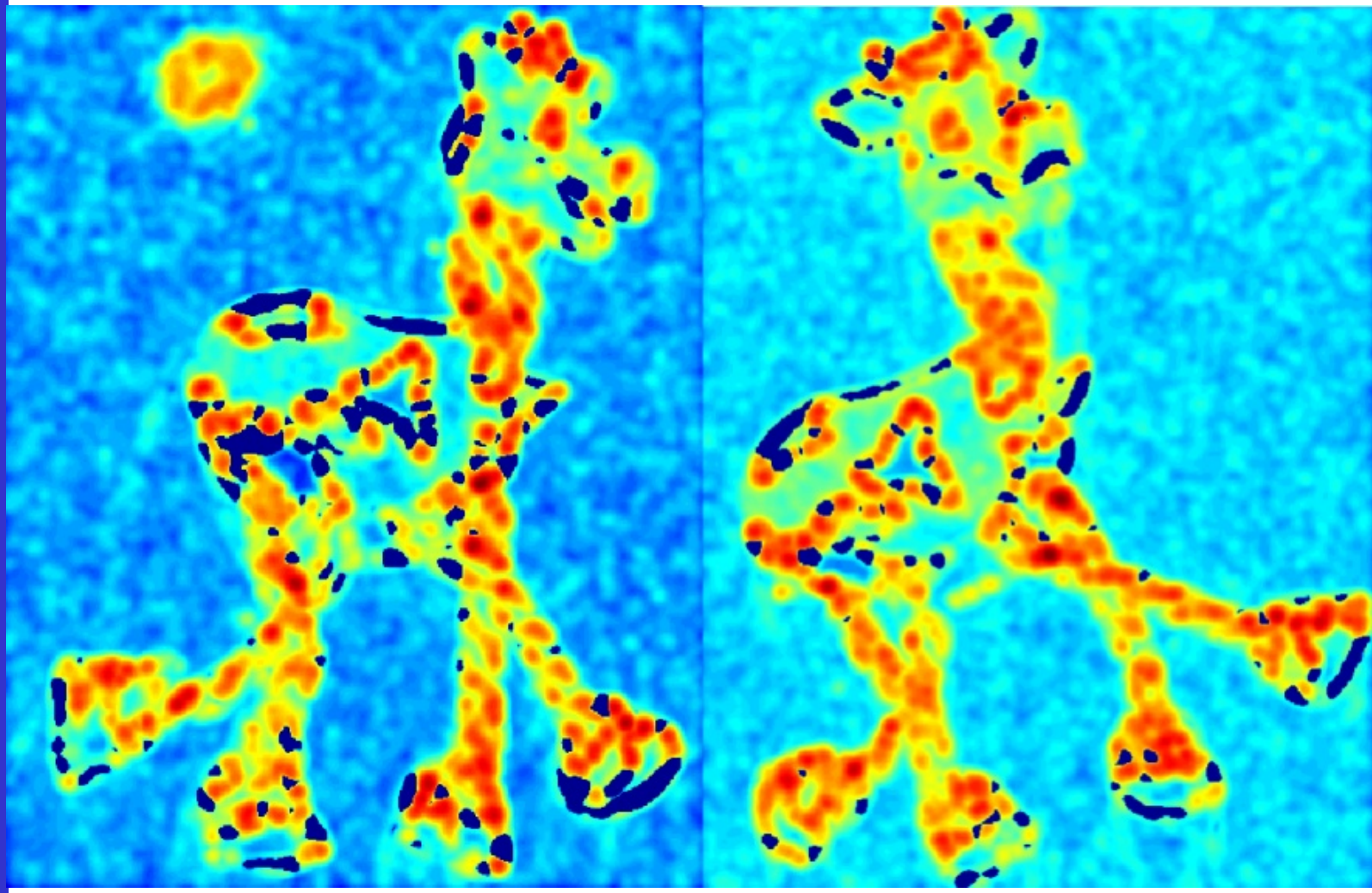


Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

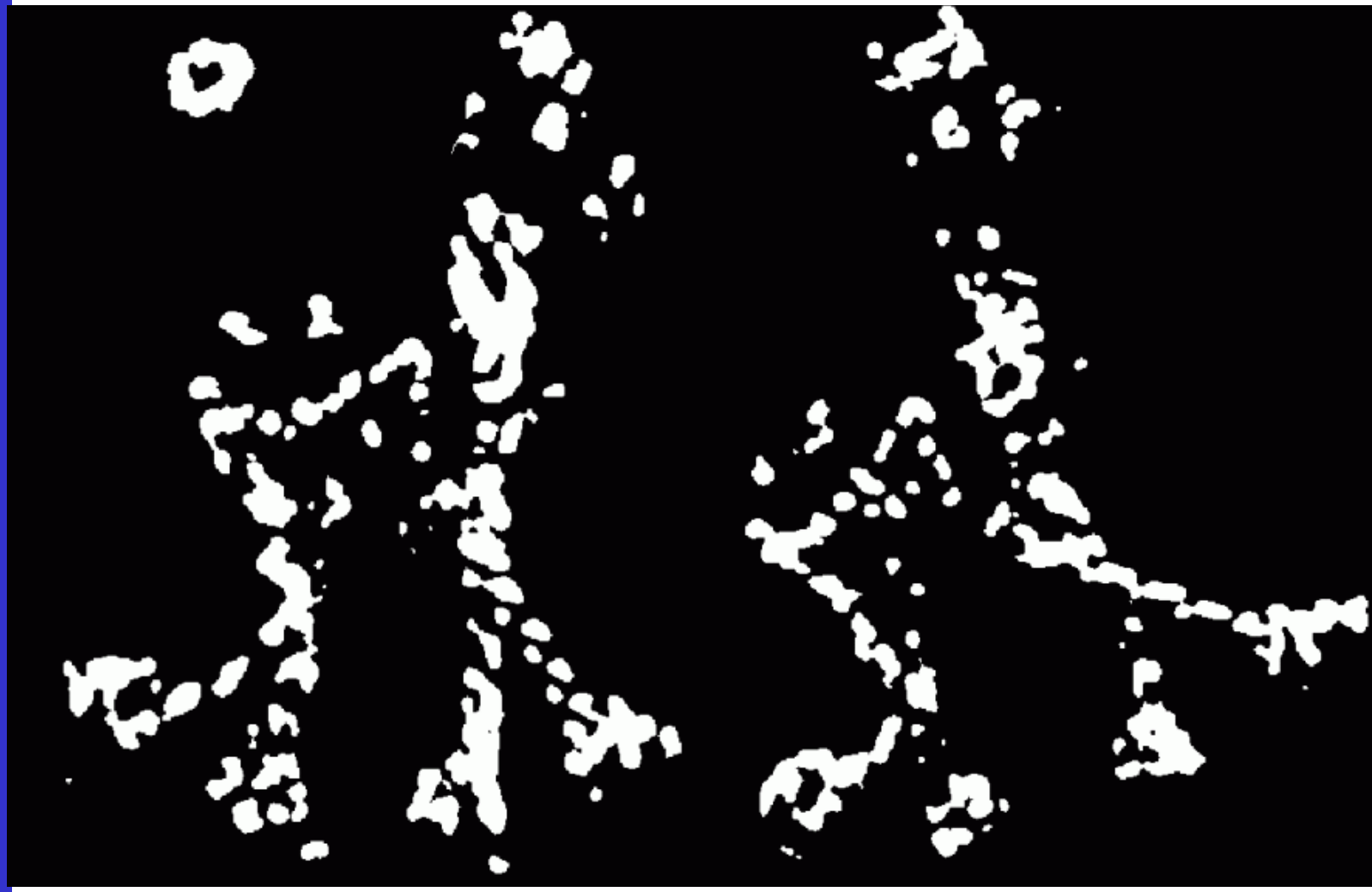
The Harris corner detector

2 views of an object… are the corners stable?

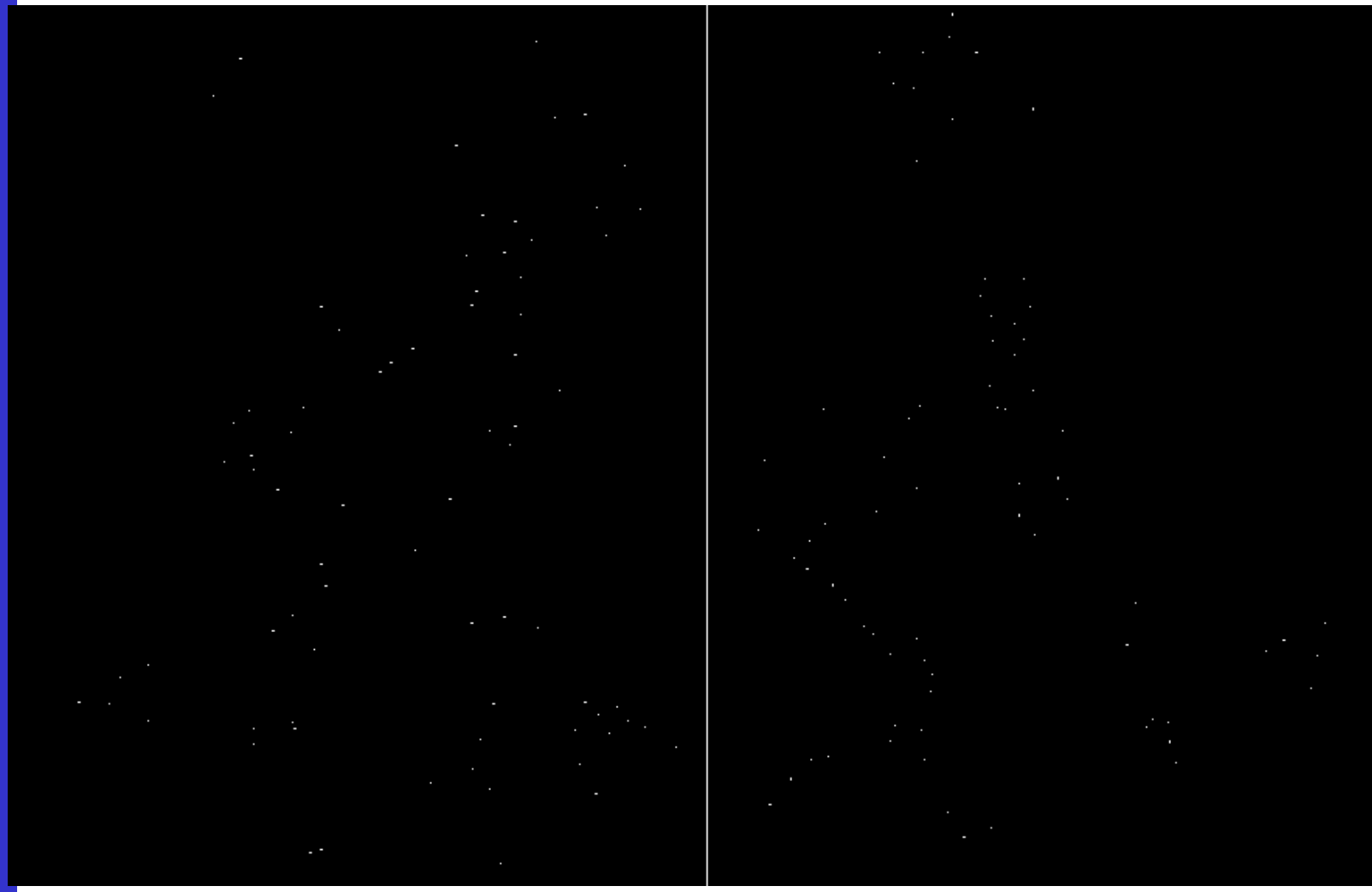
The Harris corner detector



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.

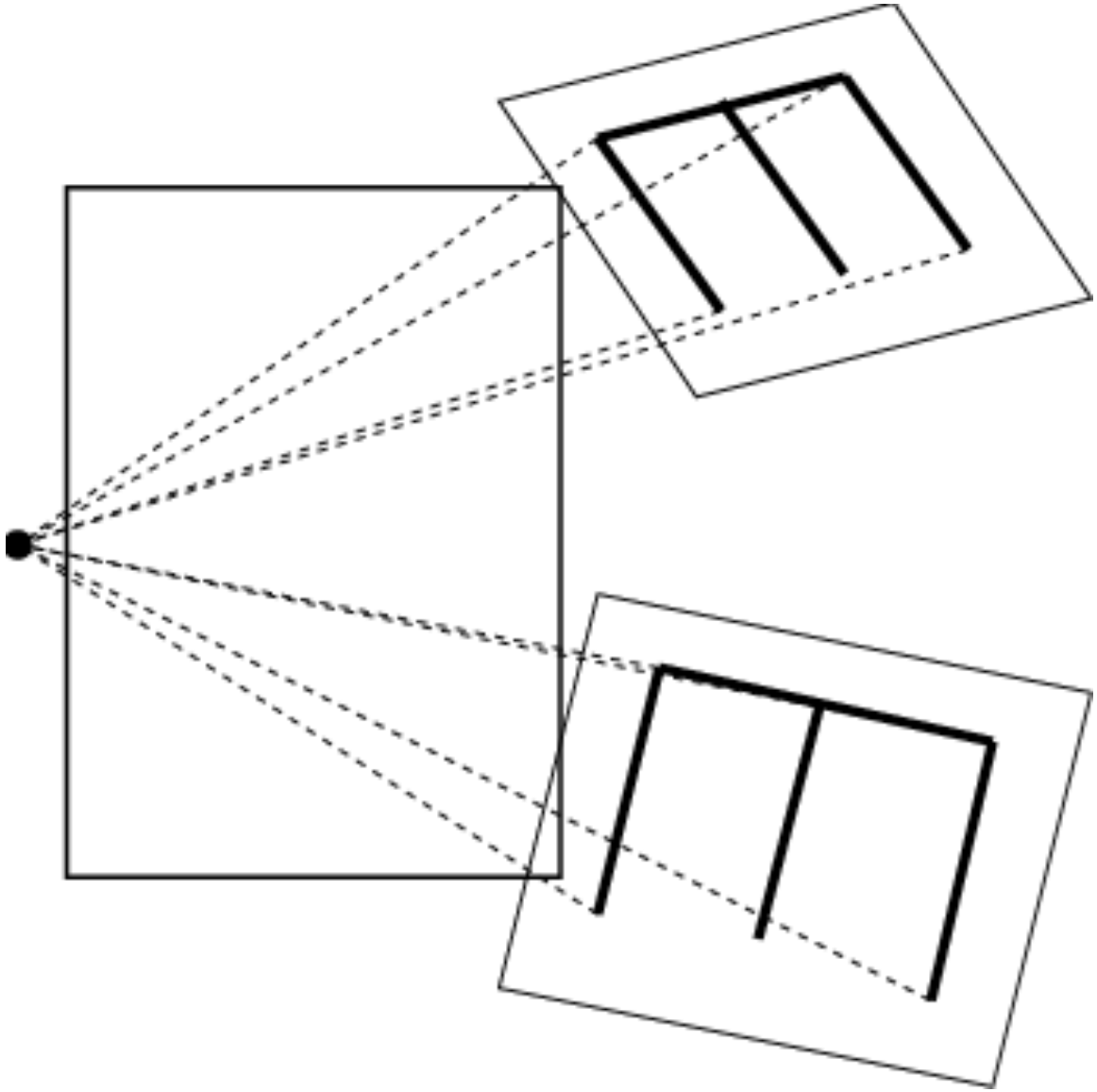
Additional requirement on the descriptor:

Invariance under geom./phot. change

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

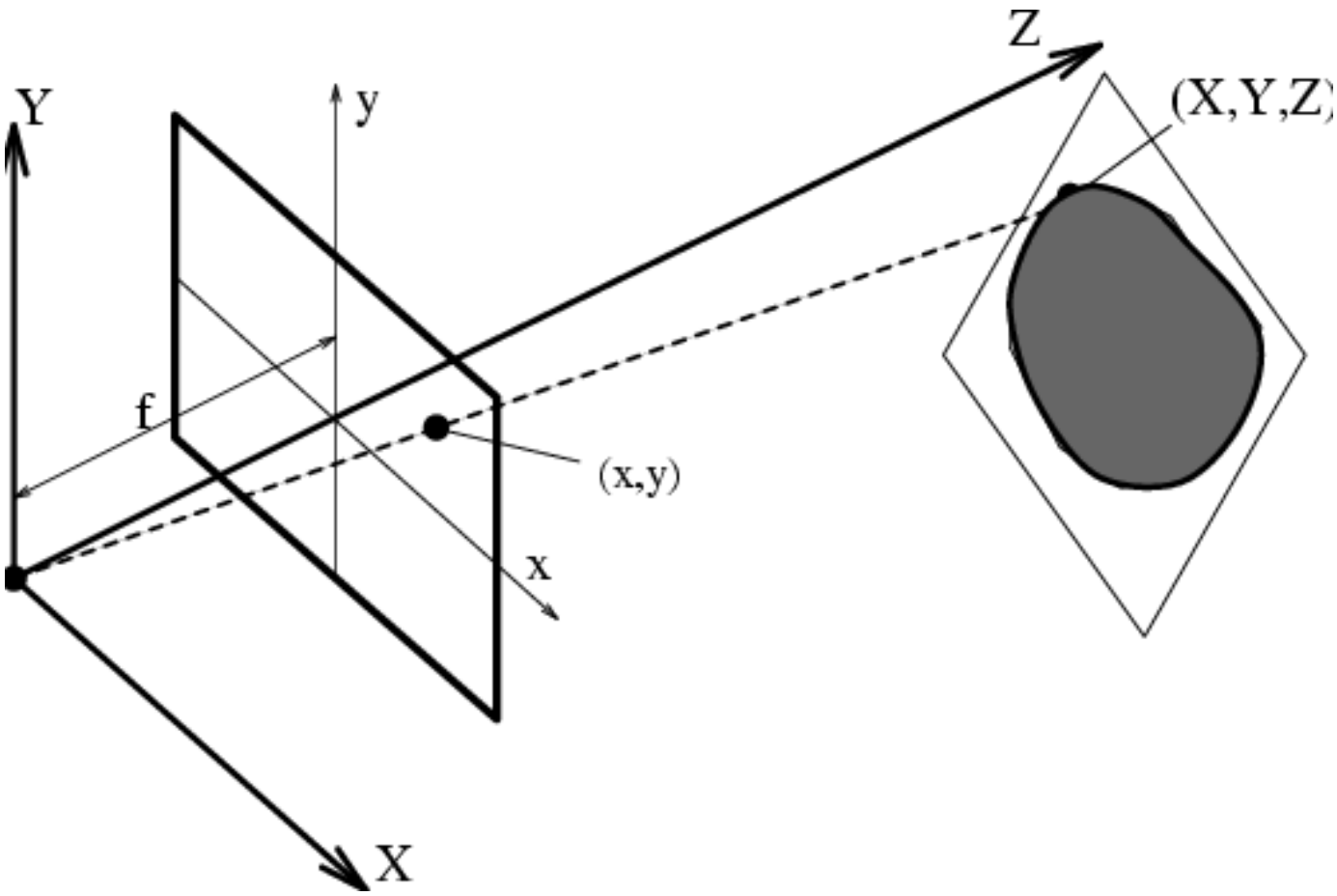
But how do planar patches deform when looking at their image projection?
i.e. we determine the *geometric* changes the descriptor should remain invariant under

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 under projection

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

we consider 3 cases :

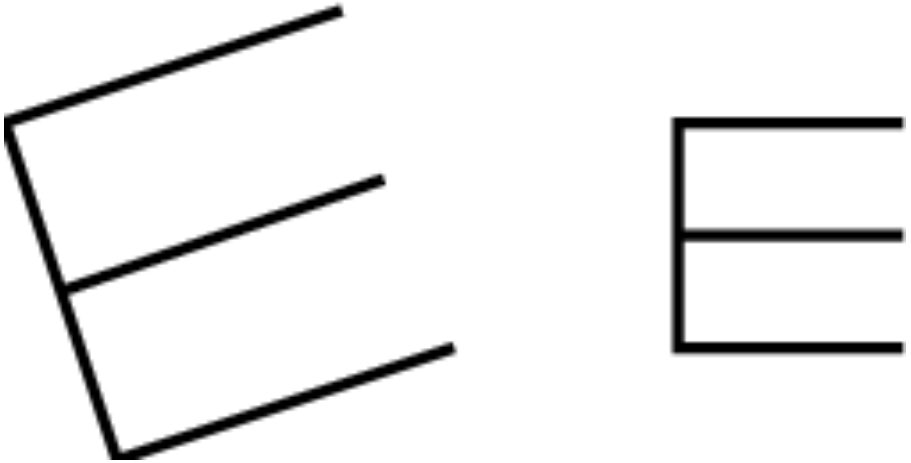
1. viewed from a perpendicular direction
2. viewed from any direction but at sufficient distance to use pseudo - perspective
3. viewed from any direction and from any distance

in the introductory lectures, it was found that…

Summary case 1

Case of fronto-parallel viewing :

SIMILARITY

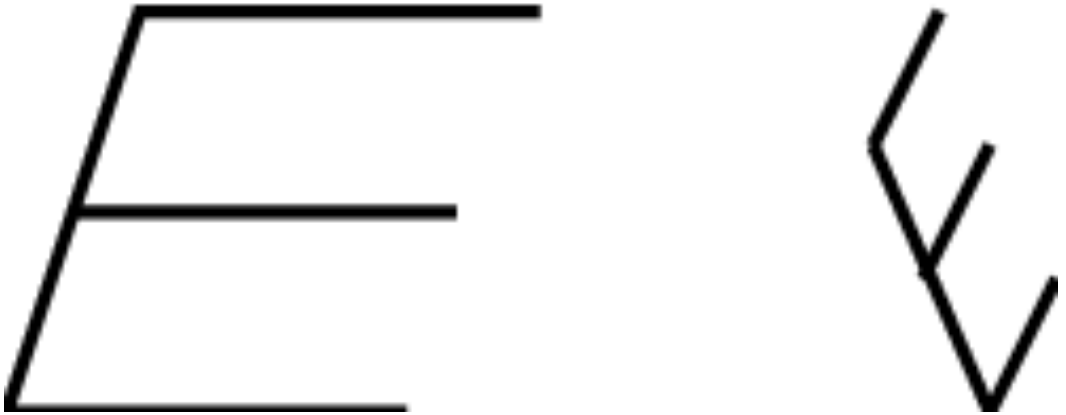


$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathrm{s} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Summary case 2

Viewing from a distance :

AFFINITY



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

Summary case 3

General viewing conditions:

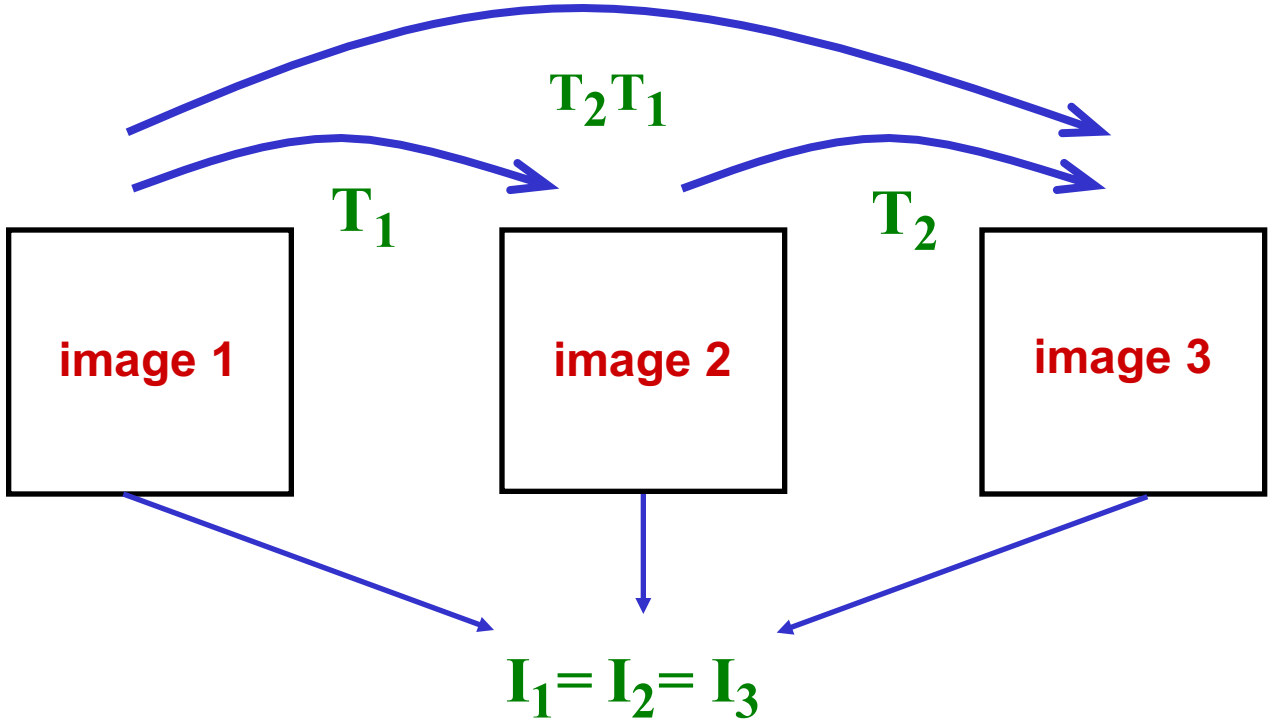
PROJECTIVITY

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

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

Invariance and groups

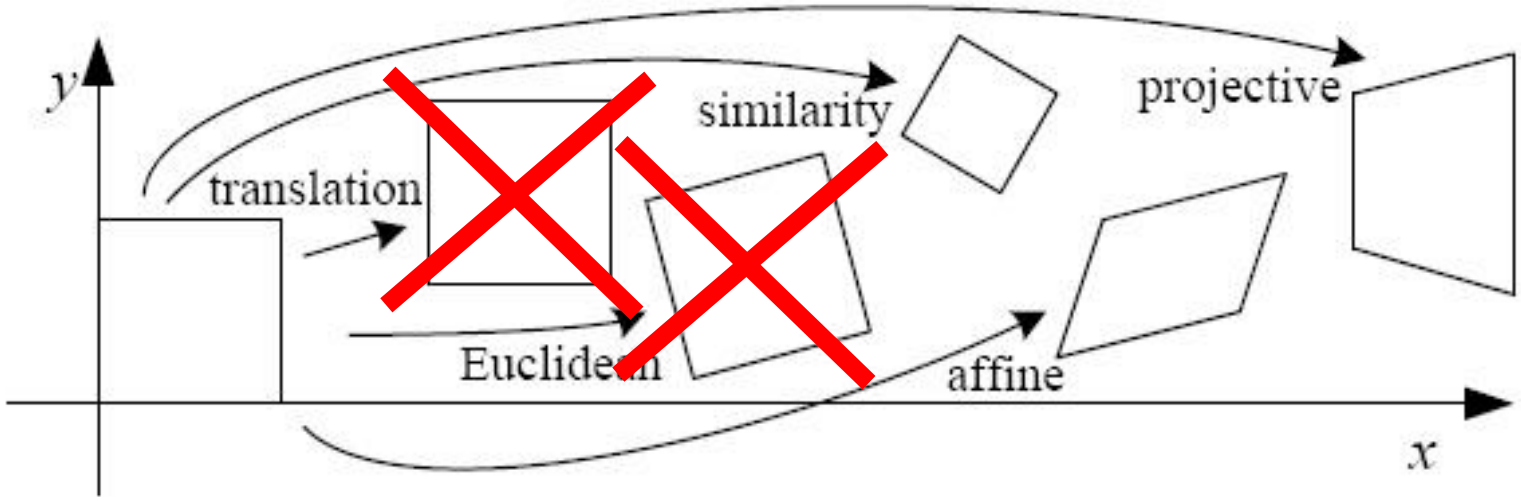
Invariance w.r.t. group actions



Invariance under transformations implies invariance under the smallest encompassing group

Remarks

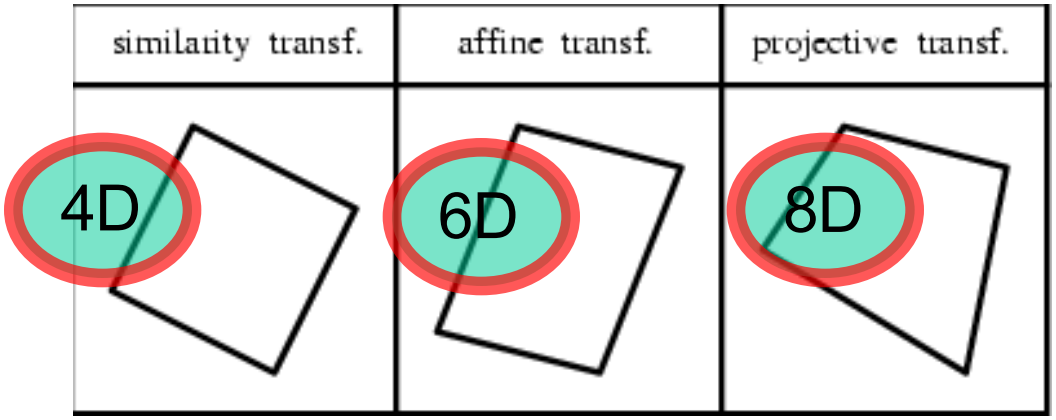
There are more groups, but the ones described are the most relevant for us



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

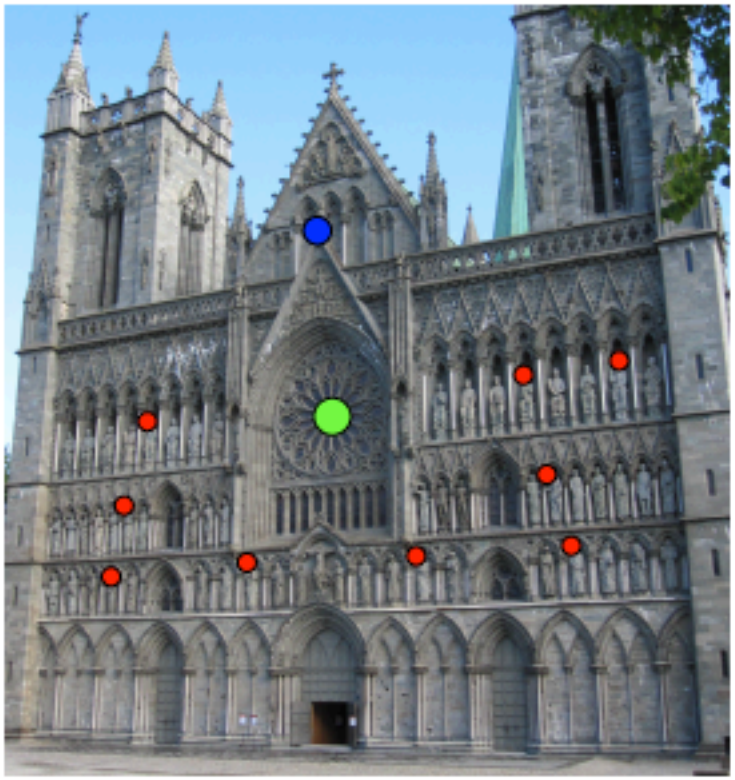
Note: matching simple intensities around interest points as descriptors, with NCC

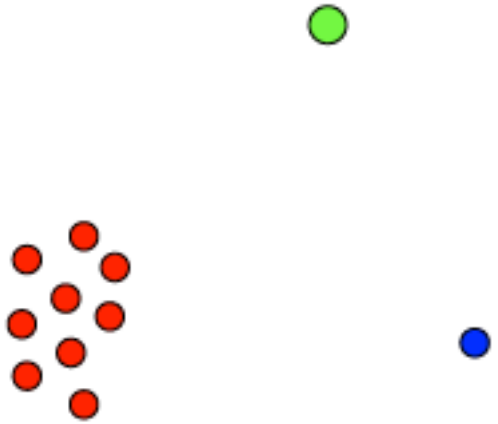
- NCC = Normalized Cross Correlation
- Invariant to affine intensity changes

$$\rho(u,v) = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\left(\sum_j (u_j - \bar{u})^2\right)\left(\sum_j (v_j - \bar{v})^2\right)}}$$

NCC basically follows a matched filtering approach, correlating normalized intensities instead of the original intensity values.

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