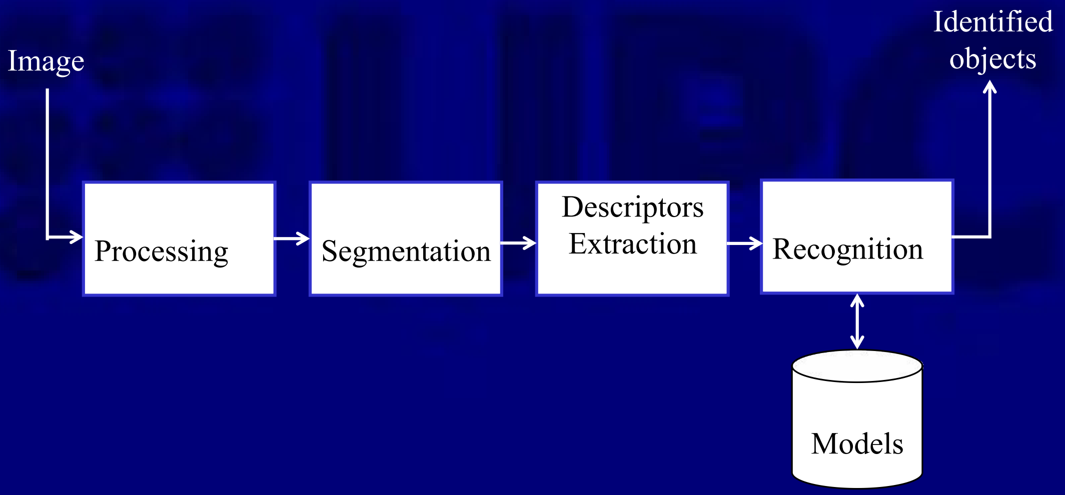
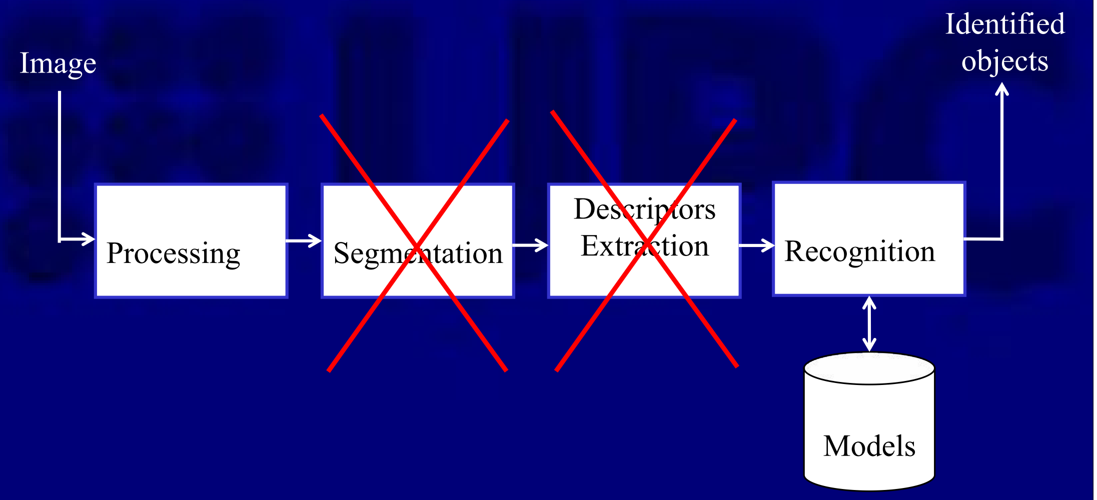
Local Features

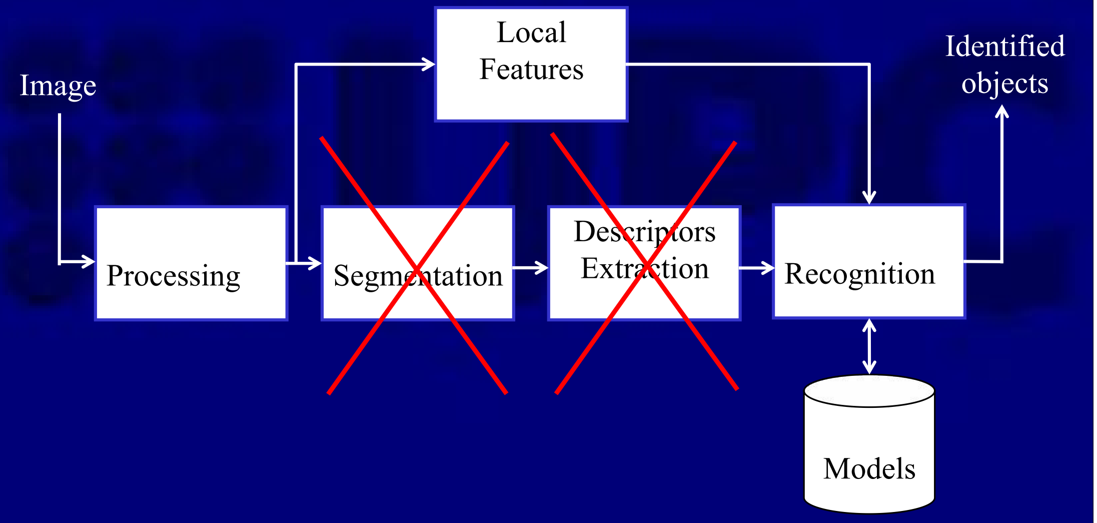
A computer vision system



A computer vision system



A computer vision system



Local Features

- Histograms
- Hough transform
- Corners
- Scale Invariant Feature transform (SIFT)
- Haar Features (face detection)

Keypoints. Etapes

1. Detecció de Keypoints (Harris)
2. Descripció de Keypoints (SIFT)
3. Aparellament de keypoints
4. Matching d'objectes

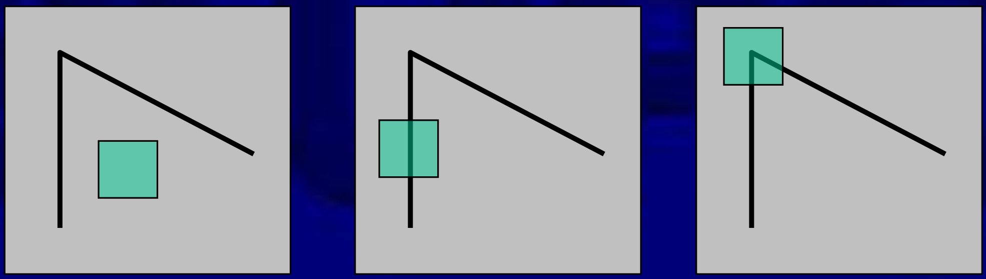
1. Detecció de keypoints

Vèrtexos

Saliency

Cal buscar parts de la imatge "inusuals"

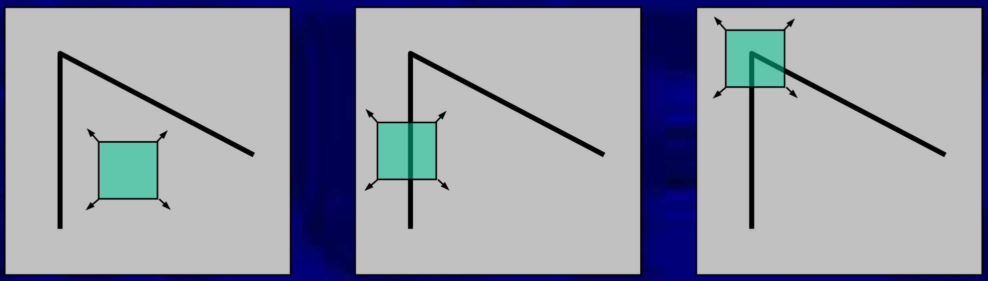
Què vol dir "inusuals"?



Saliency

Mesura d'inusualitat d'una feature

- Com varia el descriptor quan ens movem?
- Cal que al moure's els paràmetres variïn considerablement



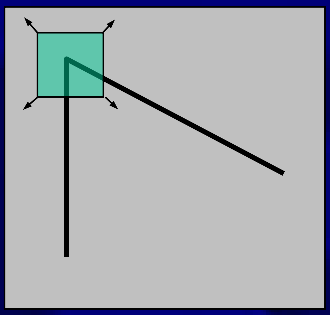
Regió plana:
cap canvi en
cap direcció

Contorn:
Sense canvis al
llarg de la direcció
del contorn

Vértex:
Canvis significatius
en totes direccions

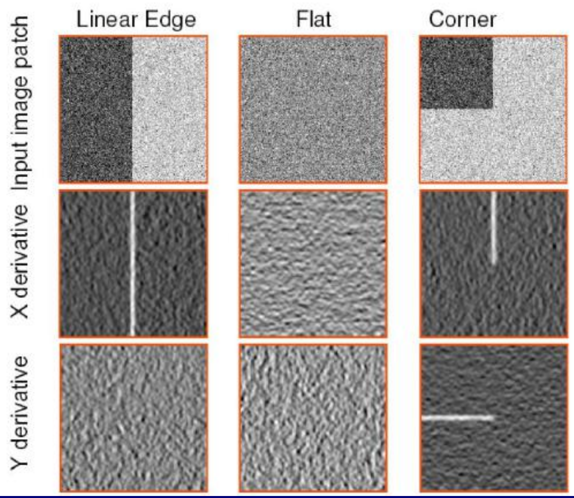
Vèrtexs

Punts on es troben dos contorns



Les derivades I_x i I_y son importants prop del vèrtex.

Corner Detector : Intuitive Example



Treat gradient vectors as a set of (dx, dy) points with center of mass at (0,0)

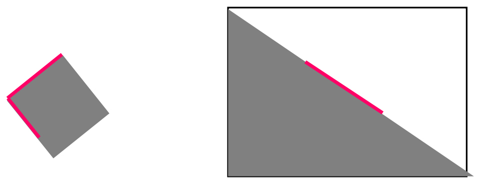
Detecció de vèrtexs

- Trobarem els vèrtexs usant la matriu:

$$C = \begin{bmatrix} \sum {I_x}^2 & \sum I_x I_y \\ \sum I_x I_y & \sum {I_y}^2 \end{bmatrix}$$

Detecció de vèrtexs

- Què passa si el vèrtex no està aliniat amb el sistema de coordenades?



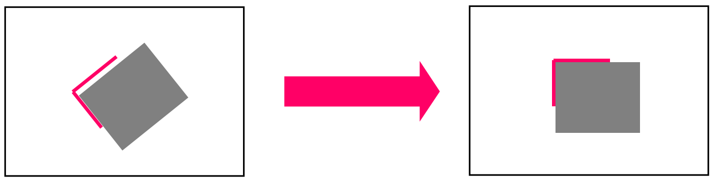
$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Les derivades I_x i I_y són importants prop d'un vertex

Però també són importants en un contorn diagonal !!!

Detecció de vèrtexs

- Solució: aliniar el vèrtex amb el sistema de coordenades



$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

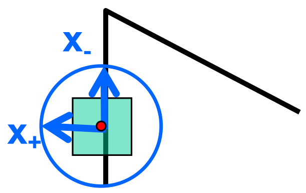
$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Detection of Corner Features

- How do we do this rotation?
  - Since C is symmetric, it can be diagonalized;
  - the diagonalization is done by the rotation we need!

Feature detection: the math

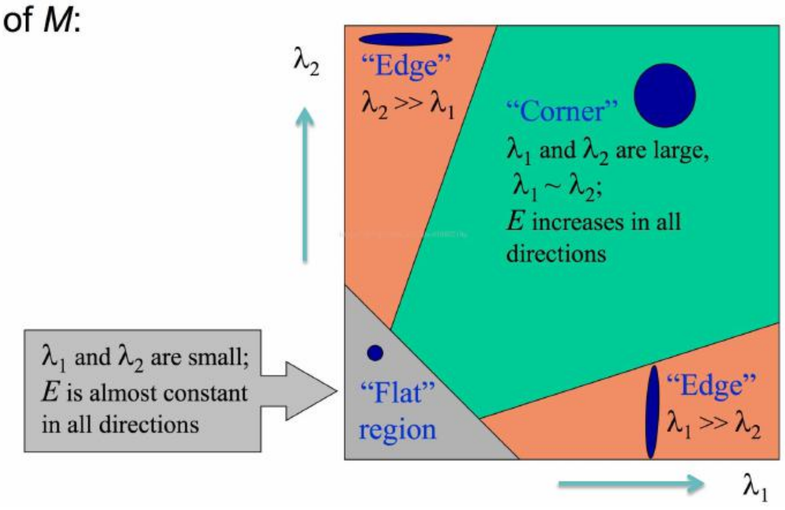
$$C = \begin{bmatrix} \sum I_x^{\,2} & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^{\,2} \end{bmatrix}$$



Eigenvalues and eigenvectors of C

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

Classification of image points using eigenvalues of *M*:



λ_2
"Edge"
$\lambda_2 >> \lambda_1$
"Corner"
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;
E increases in all directions
λ_1 and λ_2 are small;
E is almost constant in all directions
"Flat" region
"Edge"
$\lambda_1 >> \lambda_2$
λ_1

Harris Corner Detector: Cornerness Measure

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

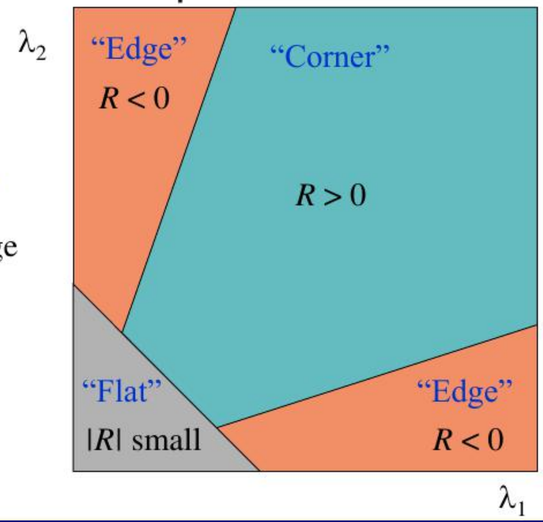
(k – empirical constant, k = 0.04-0.06)

The computation of the eigenvalues is computationally expensive. Harris proves that:

$$R=(\sum I_x^2 \cdot \sum I_y^2 - (\sum I_x \cdot I_y)^2) - k \cdot (\sum I_x^2 + \sum I_y^2)^2$$

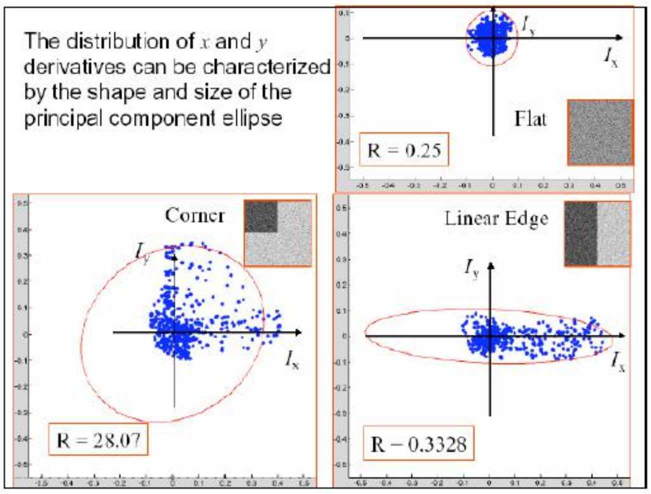
Harris Corner Detector: Corner Response

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Plotting Derivatives as 2D Points for PCA

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

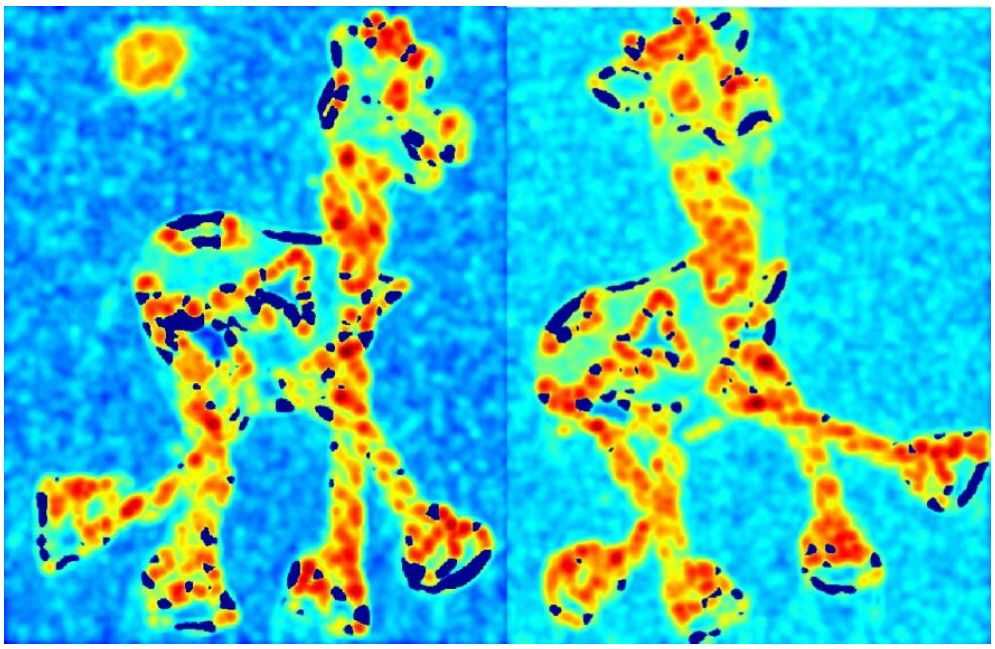


Algorithm : Harris Corner Detector

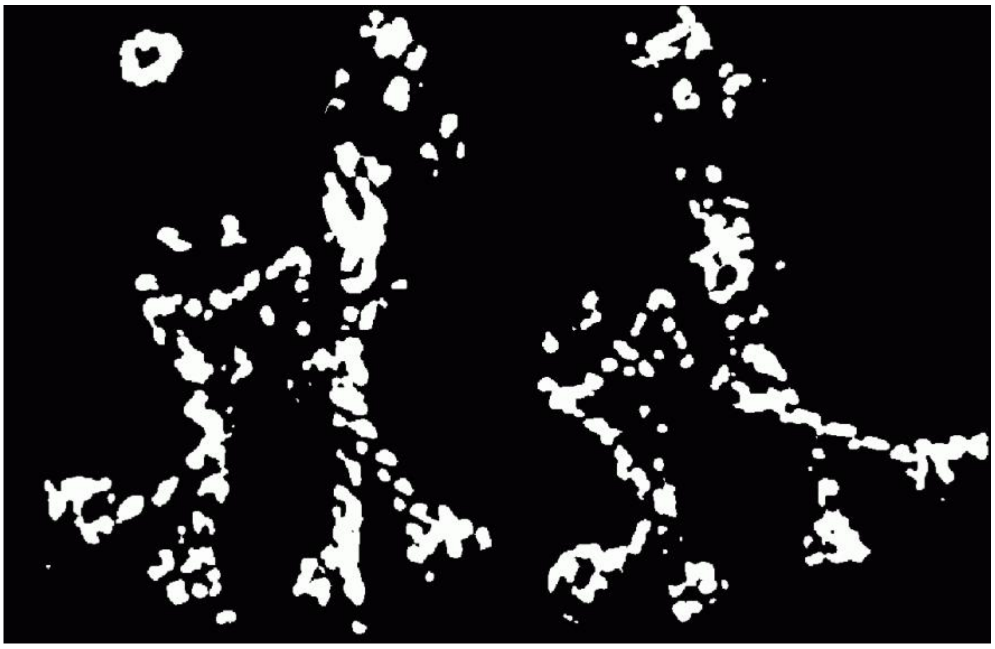
1. Computer x and y derivatives I_x and I_y of the input image
2. Computer products of derivatives I_xI_x, I_xI_y and I_yI_y
3. For each pixel, compute the matrix M in a local neighborhood
4. Compute the corner response R at each pixel
5. Threshold the value of R to select corners
6. Perform non-maximum suppression

Harris detector example

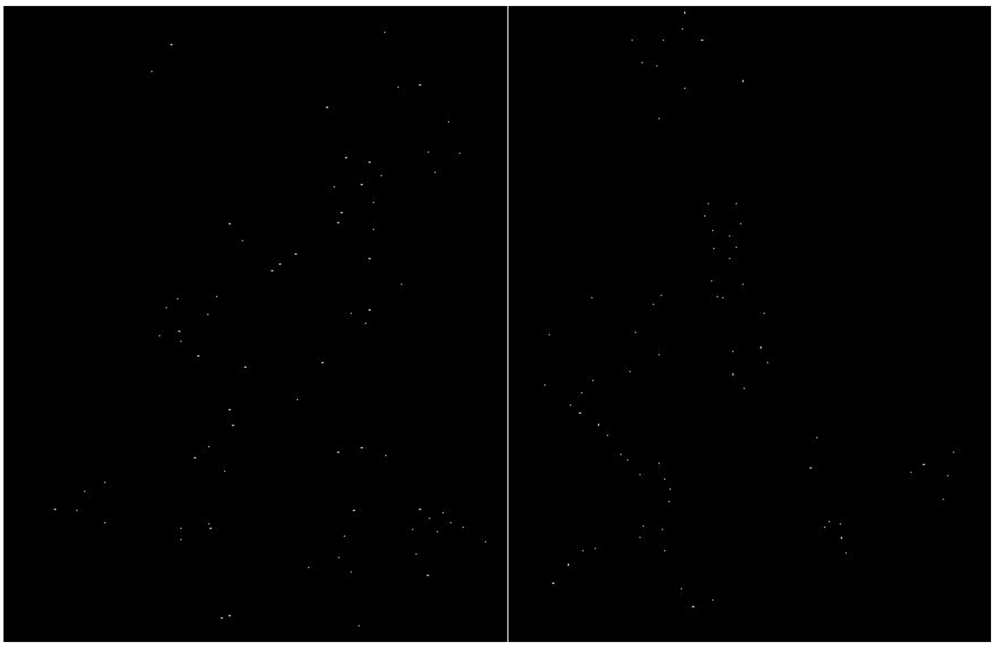
R value (red high, blue low)



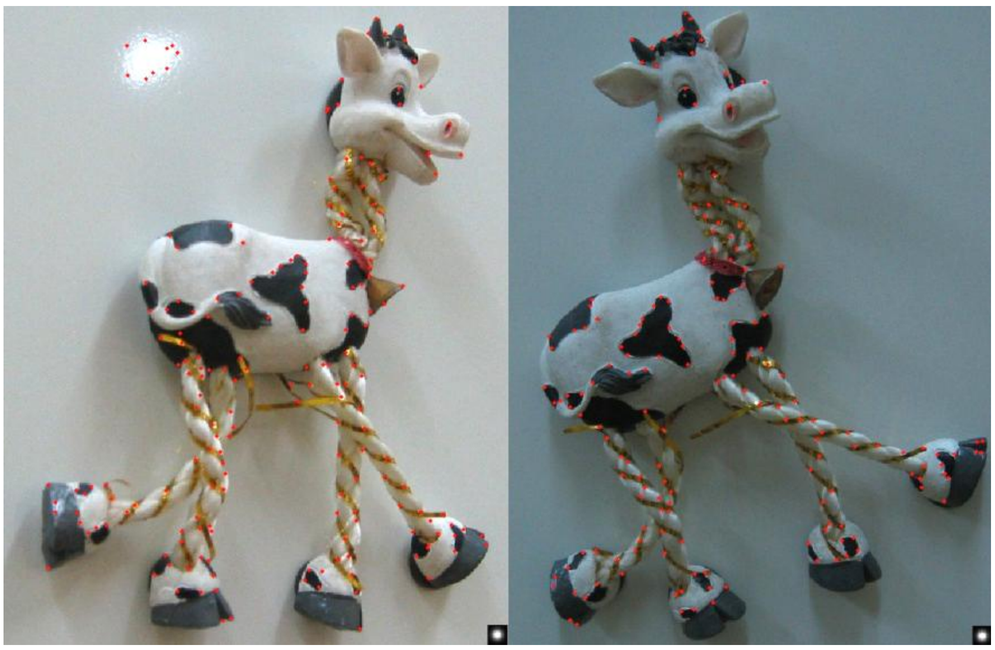
Threshold (R > value)



Find local maxima of R



Harris features (in red)



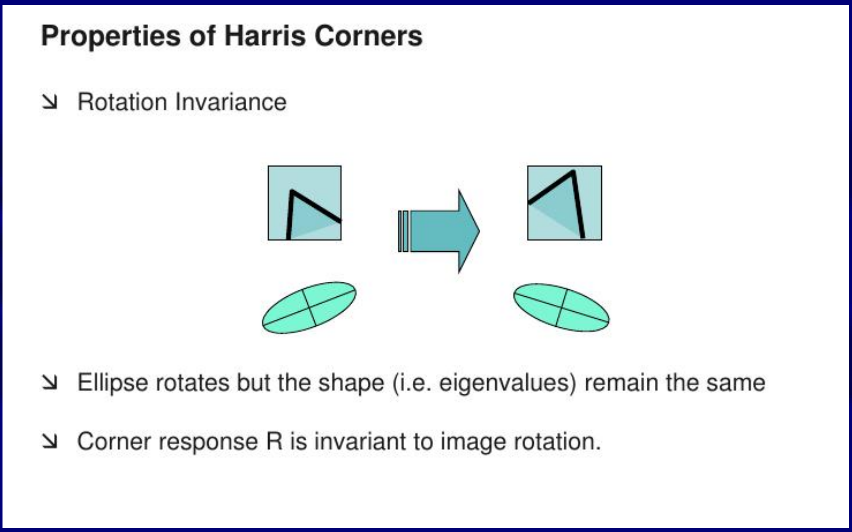
Detecció de vèrtexs. Invariància

El detector de Harris és invariant a la rotació?

I als canvis d'il.luminació?

I als canvis d'escala?

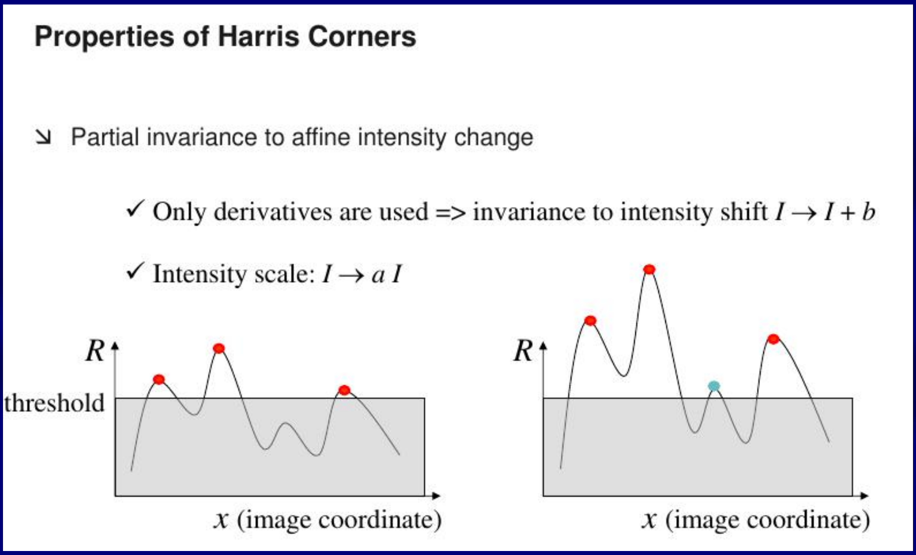
Invariància a la rotació



Properties of Harris Corners

- Rotation Invariance
- Ellipse rotates but the shape (i.e. eigenvalues) remain the same
- Corner response R is invariant to image rotation.

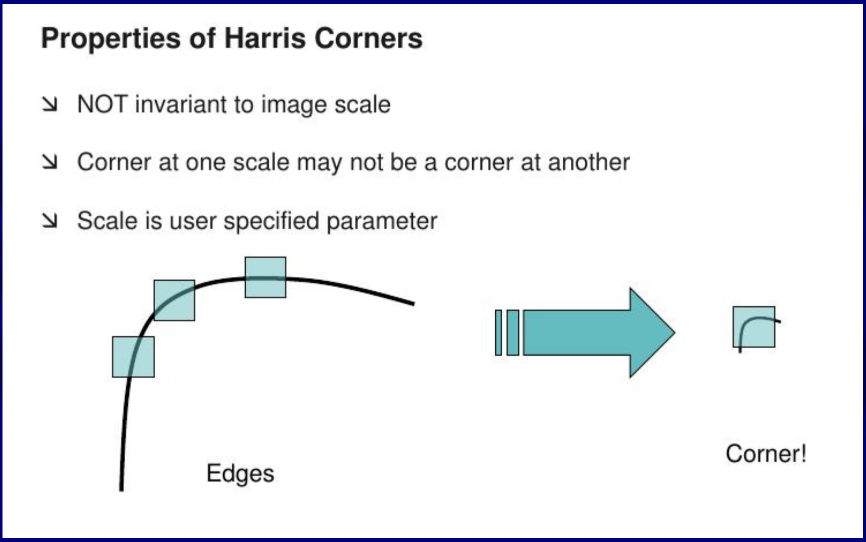
Invariància als canvis d'il.luminació


Properties of Harris Corners

- Partial invariance to affine intensity change
  - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
  - ✓ Intensity scale: $I \rightarrow a\,I$

R
threshold
x (image coordinate)
R
x (image coordinate)

Invariància als canvis d'escala

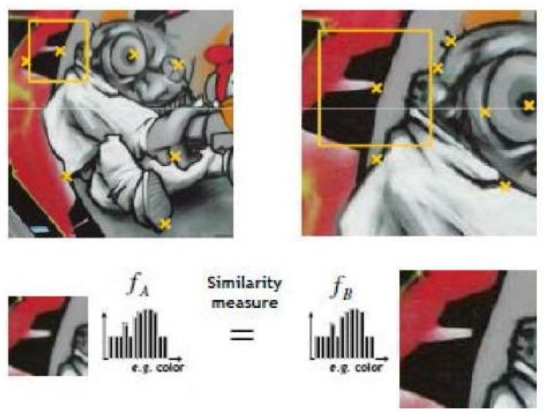

Properties of Harris Corners

- NOT invariant to image scale
- Corner at one scale may not be a corner at another
- Scale is user specified parameter

Edges
Corner!

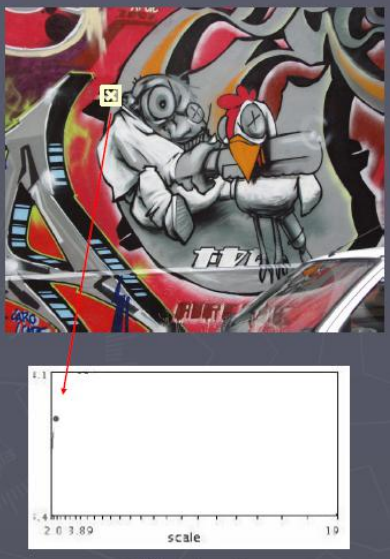
Multiples escales

Try Different Window Sizes



Automatic scale selection

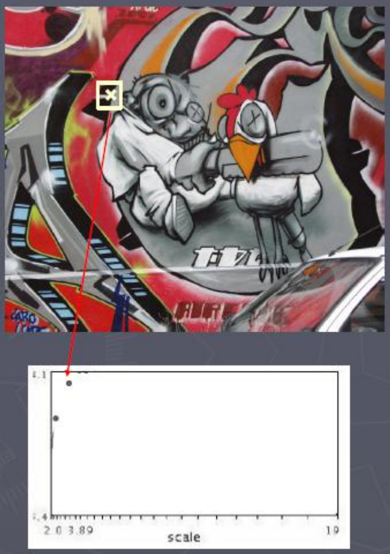
Lindeberg et al., 1996



$f(I_{i_1 \ldots i_m}(x,\sigma))$

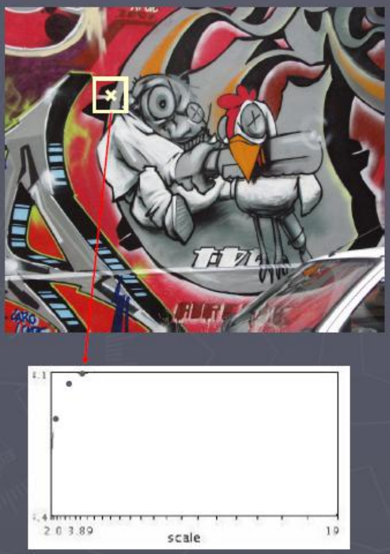
Slide from Tinne Tuytelaars

Automatic scale selection



$f(I_{i_1 \dots i_m}(x,\sigma))$

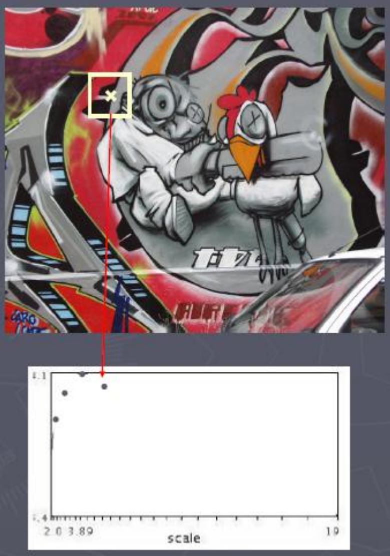
Automatic scale selection



$f(I_{i_1 \dots i_m}(x,\sigma))$

Automatic scale selection

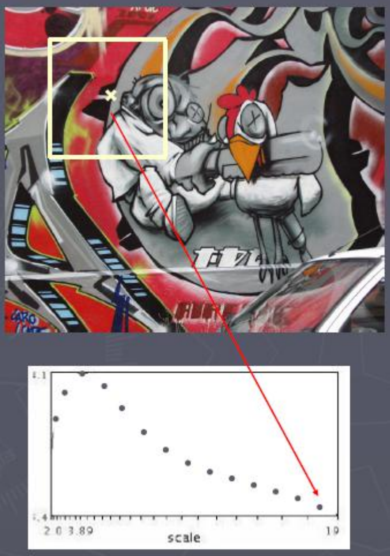
Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x,\sigma))$

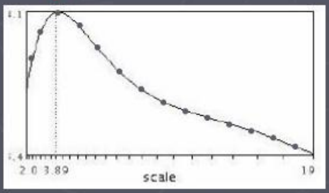
Automatic scale selection

Function responses for increasing scale (scale signature)



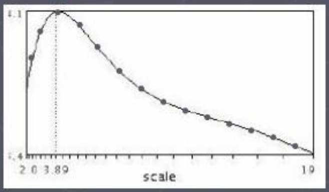
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

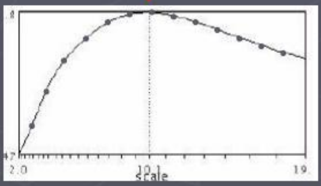


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

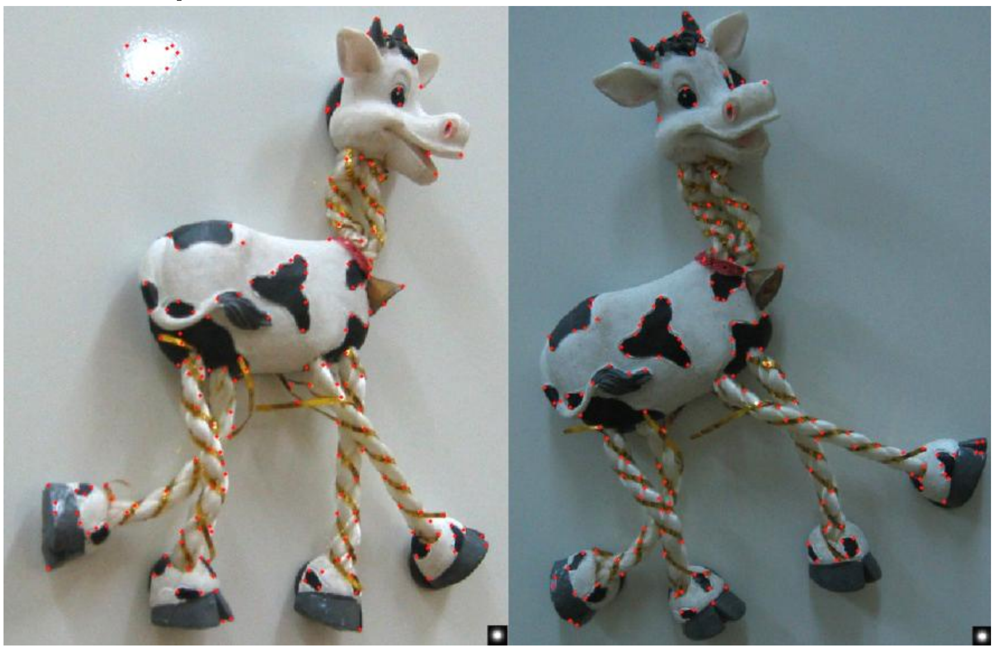


$f(I_{i_1 \ldots i_m}(x,\sigma))$

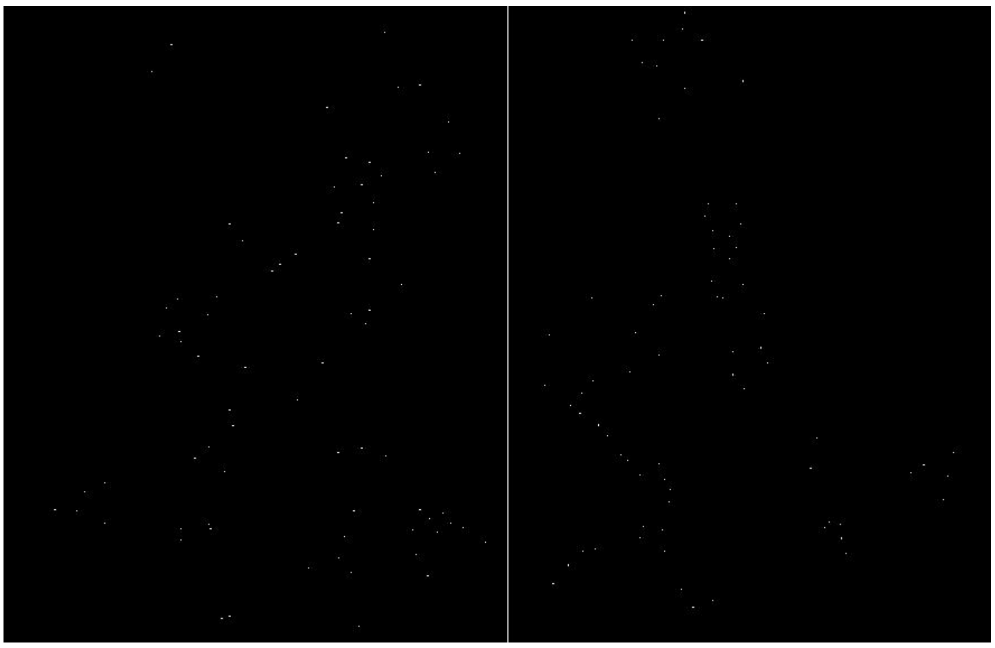


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

We know how to detect good points
Next question: **How to match them?**



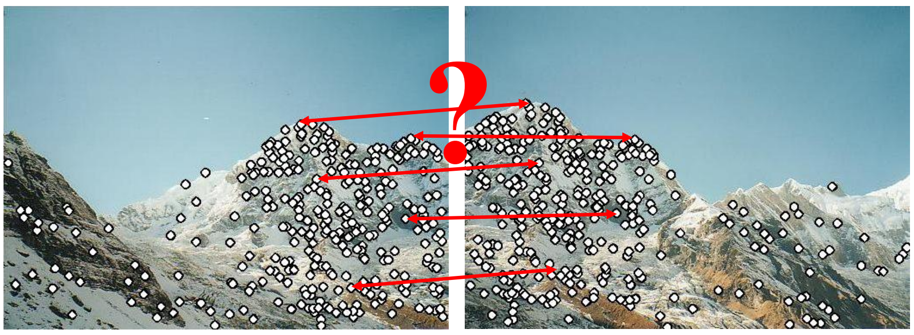
We know how to detect good points
Next question: **How to match them?**



2. Descripció de keypoints

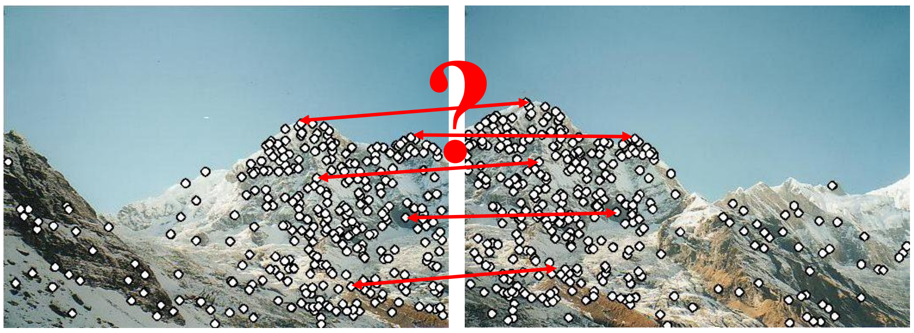
SIFT

Feature descriptors



Feature descriptors

How to match them?



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point (pattern matching)
- State of the art approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Keypoint descriptors

Cal fer-los Invariants !!!

Cal que els descriptors del mateix keypoint siguin similars en imatges diferents. Robustos a:

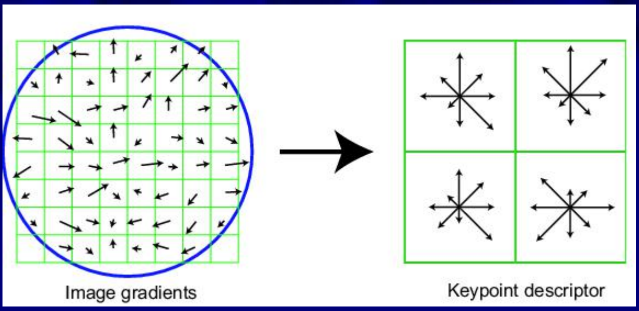
- Traslacions
- Rotacions
- Canvis d'escala
- Canvis de perspectiva
- Canvis d'il.luminació
- etc

Definitivament usar template matching NO és una bona idea

Scale Invariant Feature Transform

Idea: usar HOGS

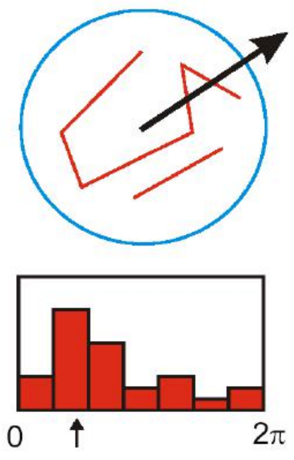
- Treballar amb una finestra al voltant del keypoint
- Calcular mòdul i orientació del gradient dels píxels veïns
- Desestimar píxels amb poc gradient
- Usar l'histograma d'orientacions com a descriptor



SIFT. Orientació principal

Orientation Assignment : Concept

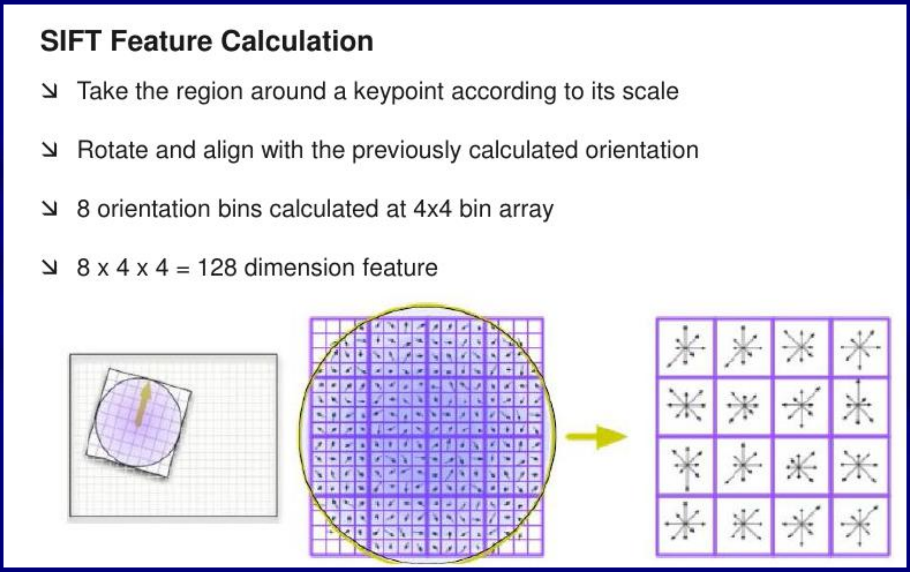
- Create histogram of local gradient directions at the selected scale
- Assign canonical orientation at the peak of the smoothed histogram
- If two major orientations, use both



SIFT. Orientació principal

Shiftem el descriptor en funció de l'orientació principal

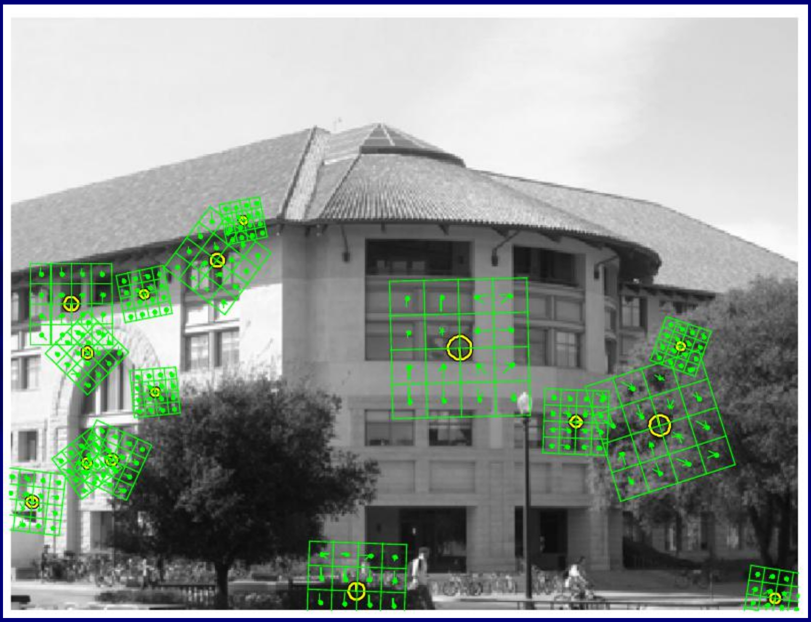
SIFT. Detalls d'implementació



SIFT Feature Calculation

- Take the region around a keypoint according to its scale
- Rotate and align with the previously calculated orientation
- 8 orientation bins calculated at 4x4 bin array
- 8 x 4 x 4 = 128 dimension feature

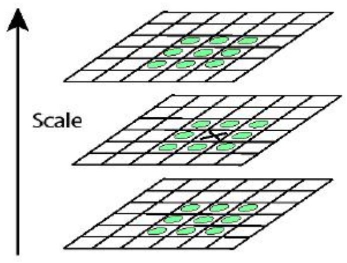
SIFT. Exemple



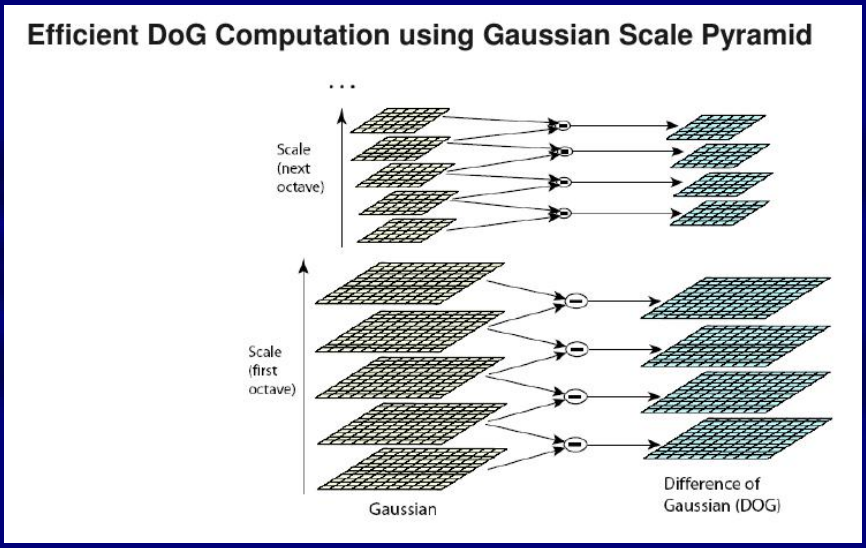
SIFT. Keypoint detector

Local Extrema in DoG Images

- Minima
- Maxima
- 26 neighbourhood



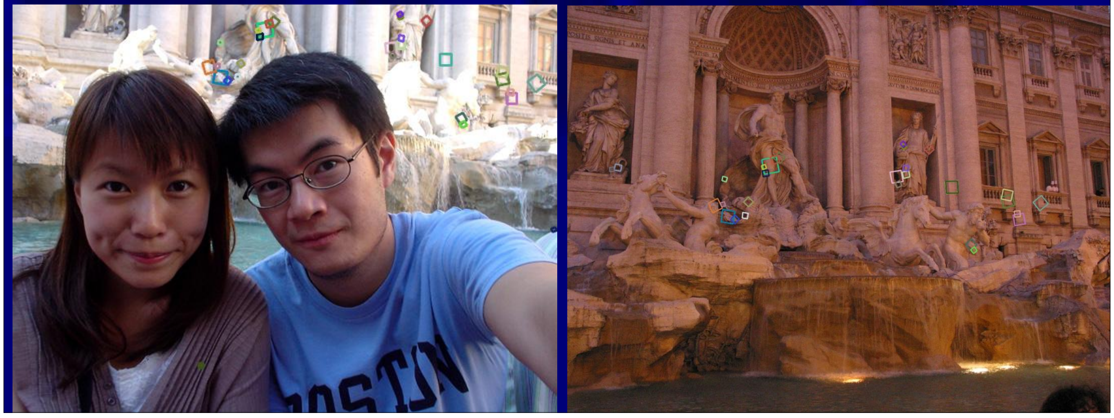
SIFT. Keypoint detector


Efficient DoG Computation using Gaussian Scale Pyramid
Scale (next octave)
Scale (first octave)
Gaussian
Difference of Gaussian (DOG)

SIFT. Propietats

- Descriptor de keypoints molt robust (el detector no gaire)
    - Robust a canvis d'il.luminació
    - Robust a rotacions i canvis de punt de vista

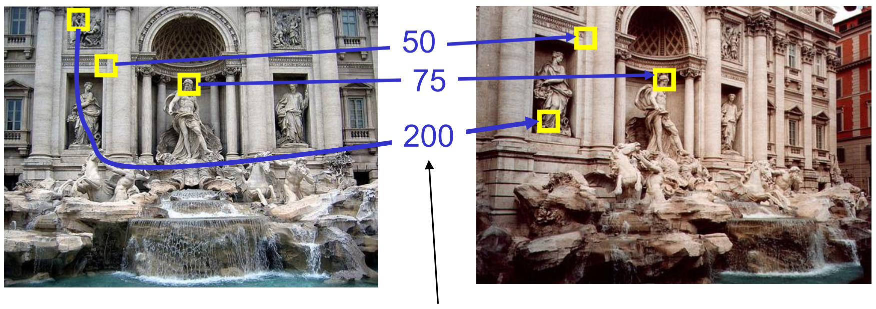
- Ràpid i eficient (temps real)
- Centenars d'implementacions disponibles a la xarxa



3. Aparellament de keypoints

Feature matching

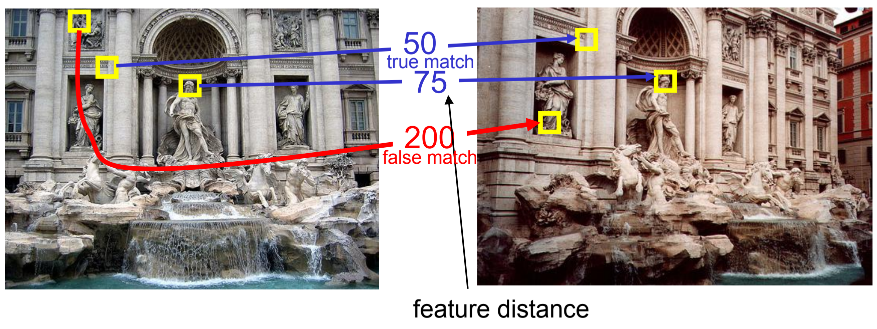
Cal definir una mètrica per comparar descriptors



feature distance

P.ex: SSD, Suma de diferències absolutes,...

True/false positives

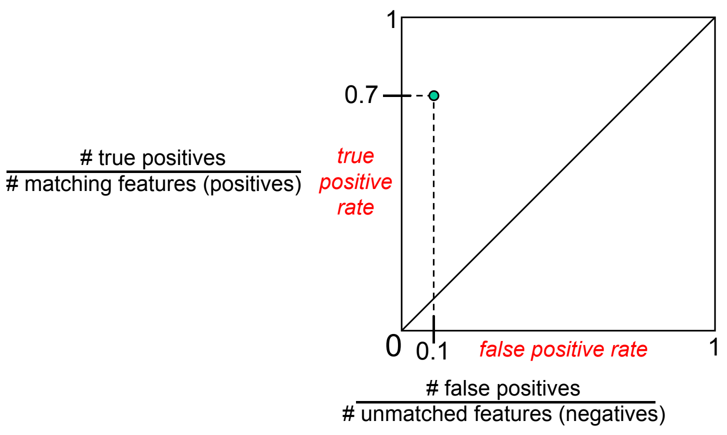


The distance threshold affects performance

- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

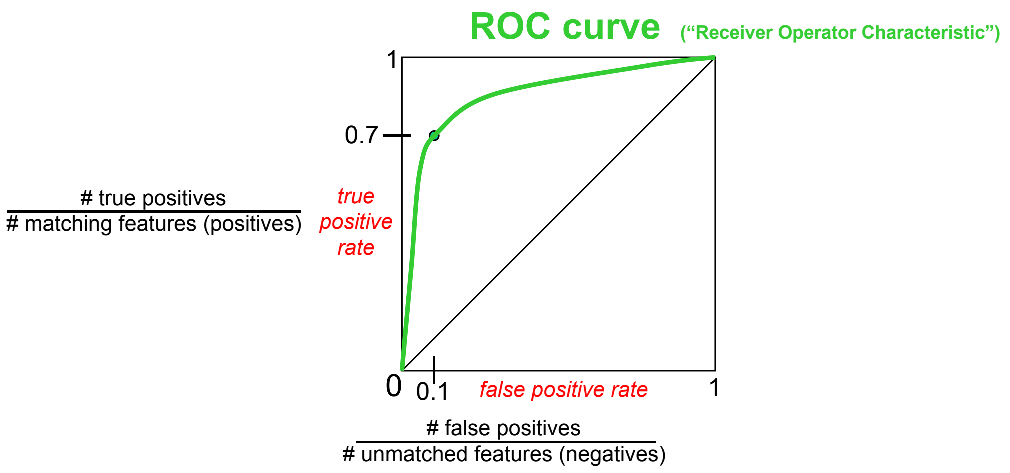
Evaluating the results

How can we measure the performance of a feature matcher?



Evaluating the results

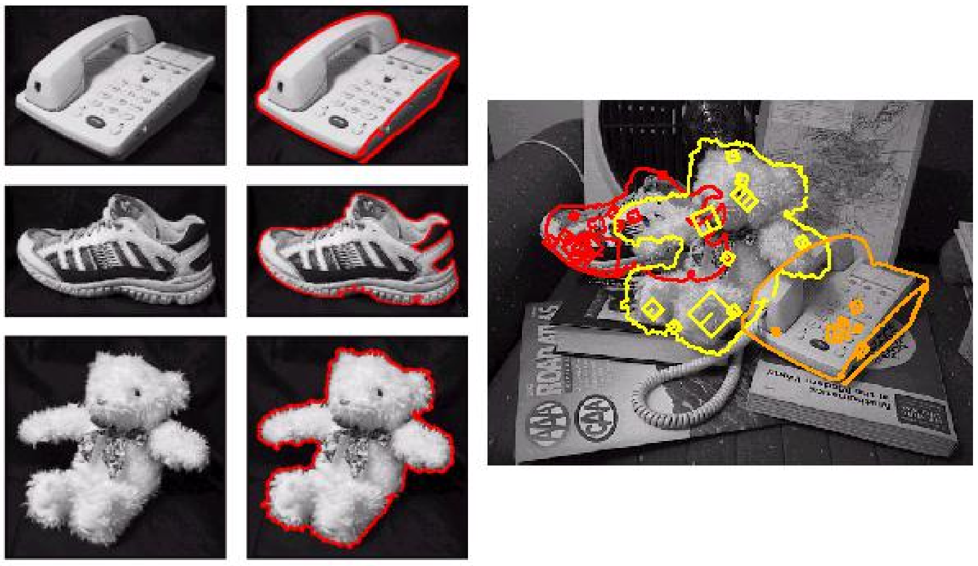
How can we measure the performance of a feature matcher?

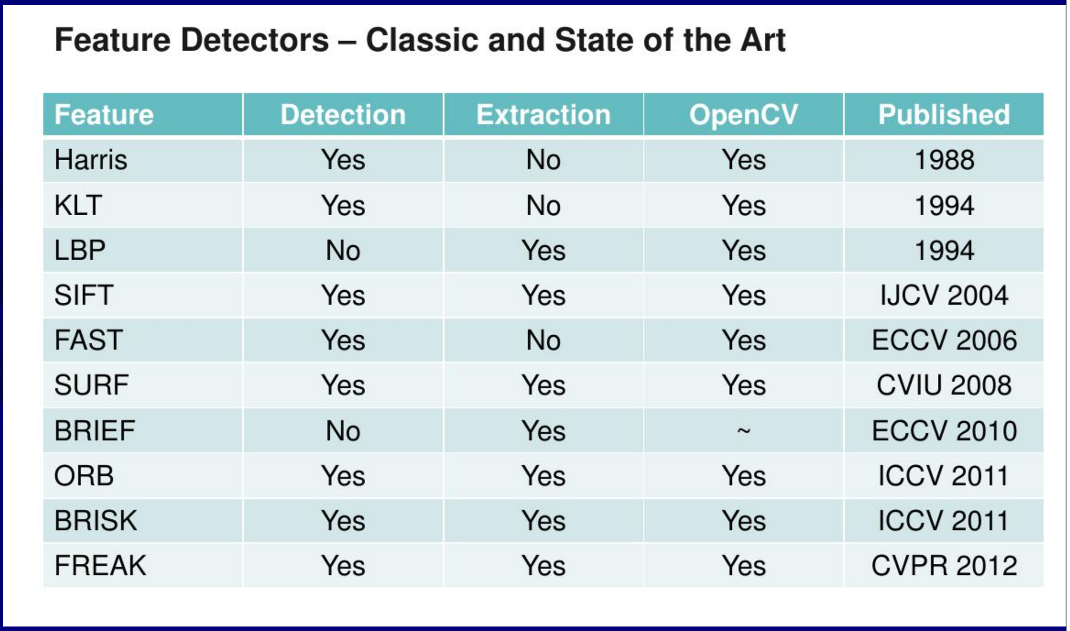


ROC Curves

- Generated by counting # current/incorrect matches, for different thresholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods

Object recognition (David Lowe)



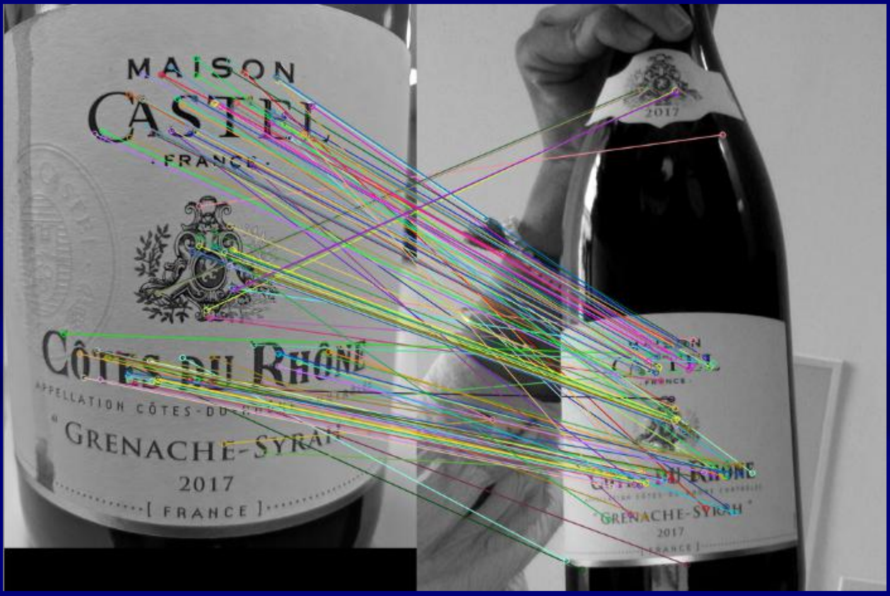


Feature Detectors – Classic and State of the Art

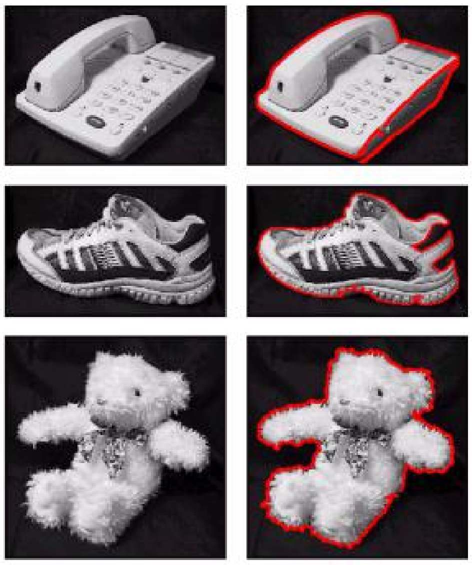
Feature	Detection	Extraction	OpenCV	Published
Harris	Yes	No	Yes	1988
KLT	Yes	No	Yes	1994
LBP	No	Yes	Yes	1994
SIFT	Yes	Yes	Yes	IJCV 2004
FAST	Yes	No	Yes	ECCV 2006
SURF	Yes	Yes	Yes	CVIU 2008
BRIEF	No	Yes	~	ECCV 2010
ORB	Yes	Yes	Yes	ICCV 2011
BRISK	Yes	Yes	Yes	ICCV 2011
FREAK	Yes	Yes	Yes	CVPR 2012

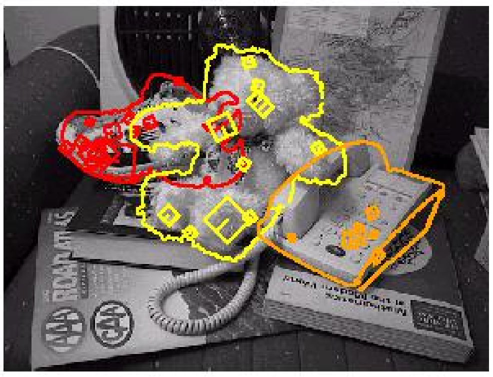
4. Matching

Cal un matching global?



Cal un matching global?

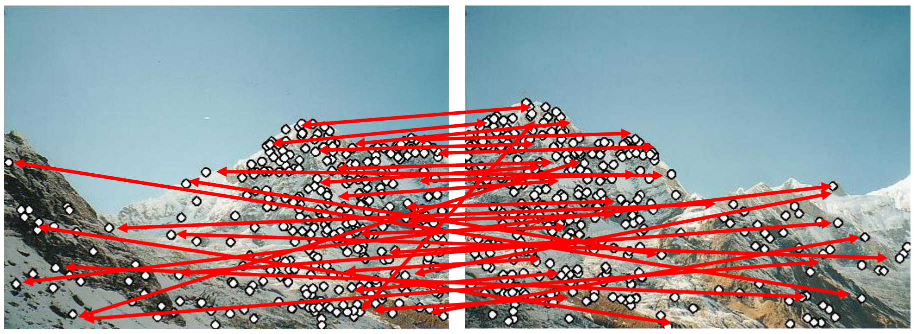




Com construïm la imatge panoràmica？

- We need to match (align) images

Com aliniem les dues imatges?



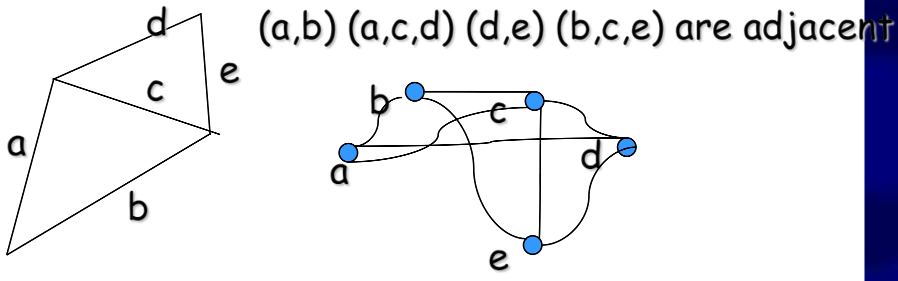
1. Buscar keypoints i els seus descriptors
2. Aparellar els keypoints de la imatge A amb els que tinguin descriptors semblants a la imatge B
3. *Buscar una matriu de transformació geomètrica T que mapegi els keypoints de la imatge A a la B*

Mosaic d'imatges

- Com trobem la matriu T ?

Relational Graphs

- Features and their relationships can be organized by using a *relational graph.*

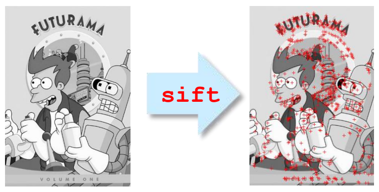


- Graph matching algorithms ➔ Exponential cost !!!!!
- Cal buscar alguna solució més eficient

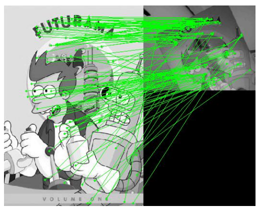
Once detected...
How do we match an object in an image?

Object matching in three steps

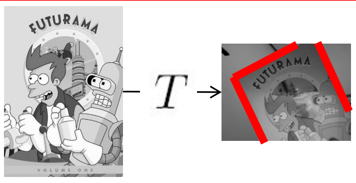
1. Detect features in the template and search images



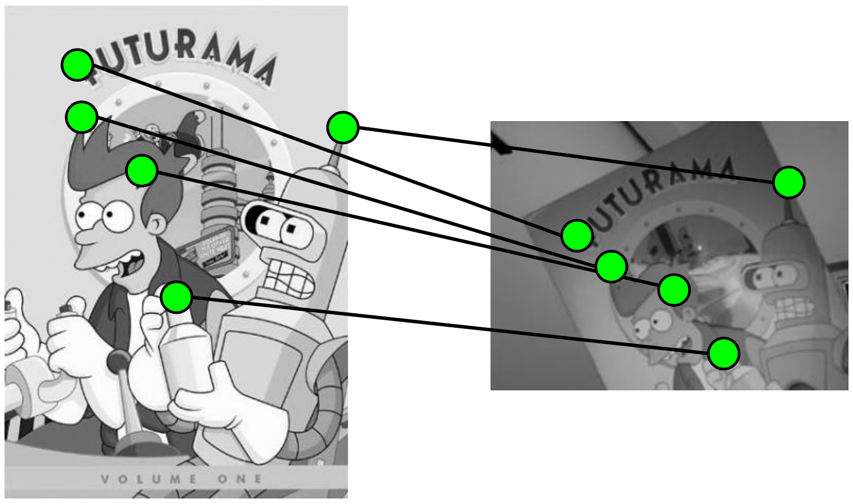
2. Match features: find "similar-looking" features in the two images



3. Find a transformation $\boldsymbol{T}$ that explains the movement of the matched features



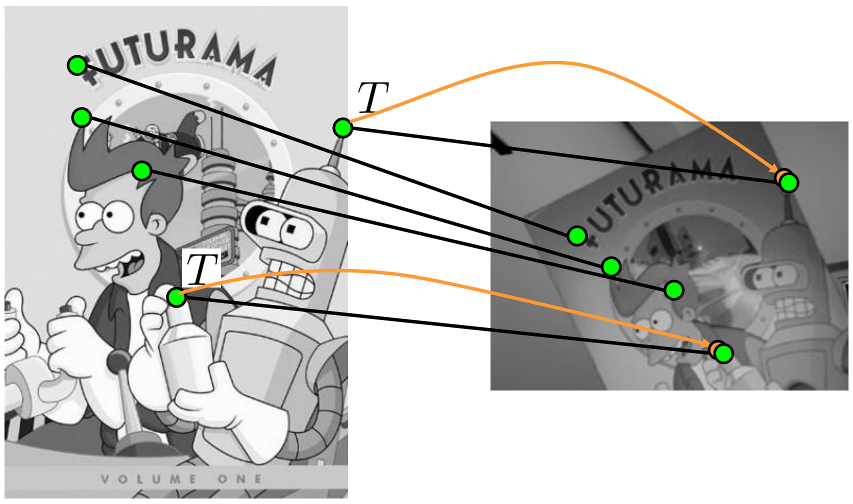
Fitting an affine transformation



- Given two images with a set of feature matches, how do we compute an affine transform between the two images?

Cal trobar T

- In other words:
  - Find 2D affine xform *T* that maps points in image 1 as close as possible to their matches in image 2



Affine transformations

- A *2D affine transformation* has the form:

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

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Multi-variable fitting

- Let's consider 2D affine transformations
  - maps a 2D point to another 2D point

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

- We have a set of n putative matches

$[\ x_1\ y_1\ ] \rightarrow [\ x_1'\ y_1'\ ]$
$[\ x_2\ y_2\ ] \rightarrow [\ x_2'\ y_2'\ ]$
$[\ x_3\ y_3\ ] \rightarrow [\ x_3'\ y_3'\ ]$
…
$[\ x_n\ y_n\ ] \rightarrow [\ x_n'\ y_n'\ ]$

Fitting an affine transformation

- Consider just one match

$[\ x_1\ y_1\ ] \rightarrow [\ x_1'\ y_1'\ ]$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix}$$

$$ax_1 + by_1 + c = x_1'$$
$$dx_1 + ey_1 + f = y_1'$$

- 2 equations, 6 unknowns → we need at least 3 matches

How do we solve for T given 3 matches?

- Three matches give a linear system with six equations:

$[\ x_1\ y_1\ ] \rightarrow [\ x_1'\ y_1'\ ]$
$$ax_1 + by_1 + c = x_1'$$
$$dx_1 + ey_1 + f = y_1'$$

$[\ x_2\ y_2\ ] \rightarrow [\ x_2'\ y_2'\ ]$
$$ax_2 + by_2 + c = x_2'$$
$$dx_2 + ey_2 + f = y_2'$$

$[\ x_3\ y_3\ ] \rightarrow [\ x_3'\ y_3'\ ]$
$$ax_3 + by_3 + c = x_3'$$
$$dx_3 + ey_3 + f = y_3'$$

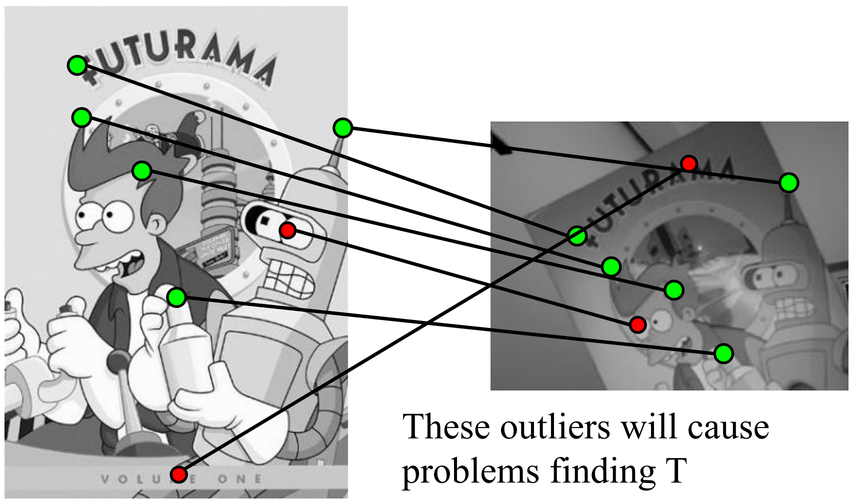
- This is just a linear system, easy to solve
- Really just two linear systems with 3 equations each (one for a,b,c, the other for d,e,f)

Fitting affine transformations

- We will fit an affine transformation to a set of feature matches
  - Problem: there are many incorrect matches

Incorrect putative matches

- We have some bad data (incorrect matches)



These outliers will cause problems finding T

Dealing with outliers

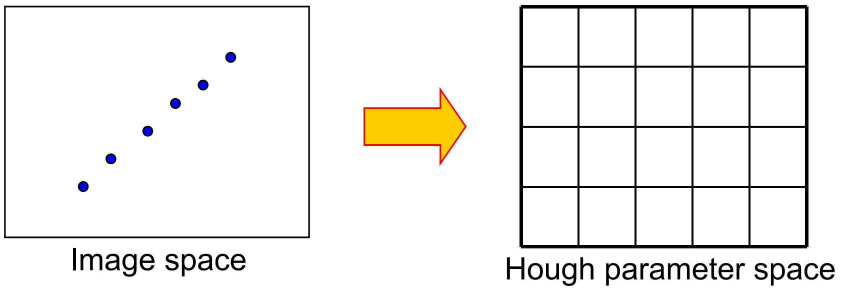
- The set of putative matches contains a very high percentage of outliers
- Geometric fitting strategies:
  - Hough transform
  - RANSAC

Hough Transform (Voting schemes)

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn’t matter as long as there are enough features remaining to agree on a good model

Hough transform

- An early type of voting scheme
- General outline:
  - Discretize parameter space into bins
  - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  - Find bins that have the most votes



P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Fitting an affine transformation

Consider just one match

$[\, x_1\ y_1 \,] \rightarrow [\, x_1'\ y_1' \,]$

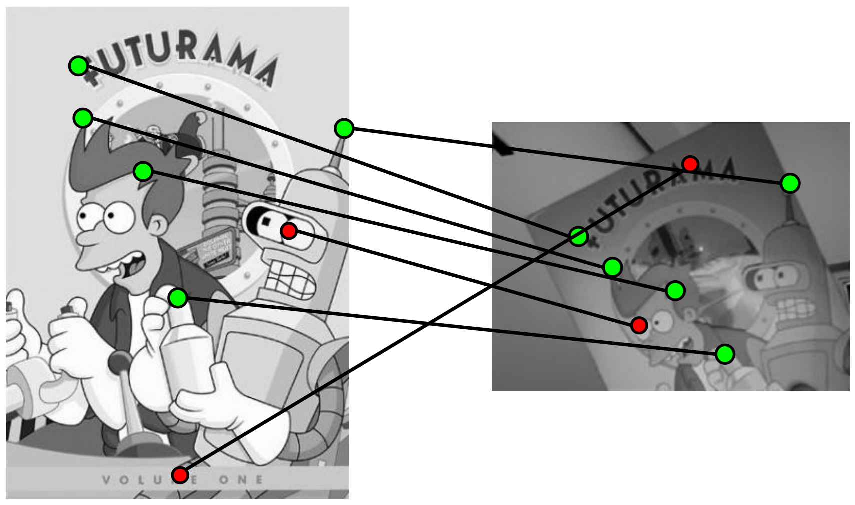
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix}$$

$$ax_1 + by_1 + c = x_1'$$
$$dx_1 + ey_1 + f = y_1'$$

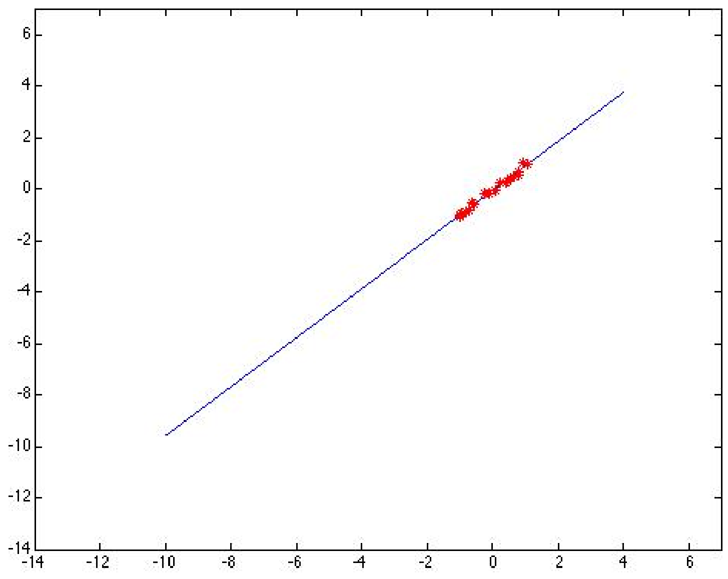
Acumule votes in the [a,b,c] and the [d,e,f] Hough arrays.

Remember the curse of dimensionality!!!

RANSAC

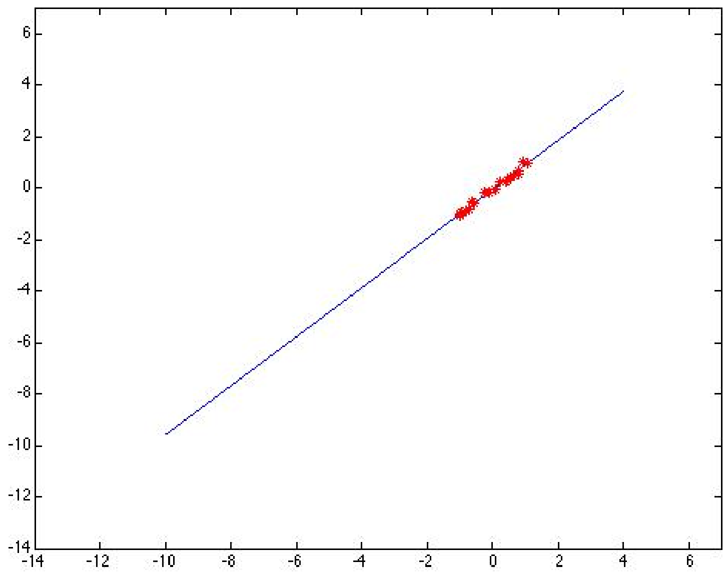


A toy example: fitting a line



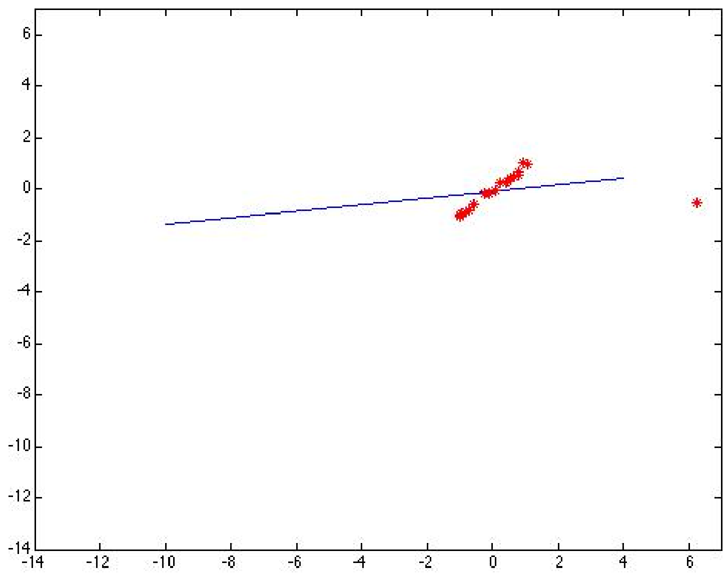
Least squares: Robustness to noise

Least squares fit to the red points:



Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

RANSAC

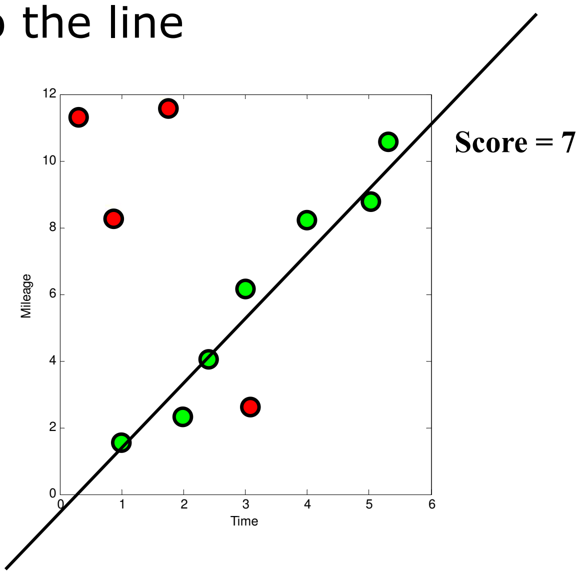
- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers
- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are "close" to the model and reject the rest as outliers
  - Do this many times and choose the best model

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

Slide: S. Lazebnik

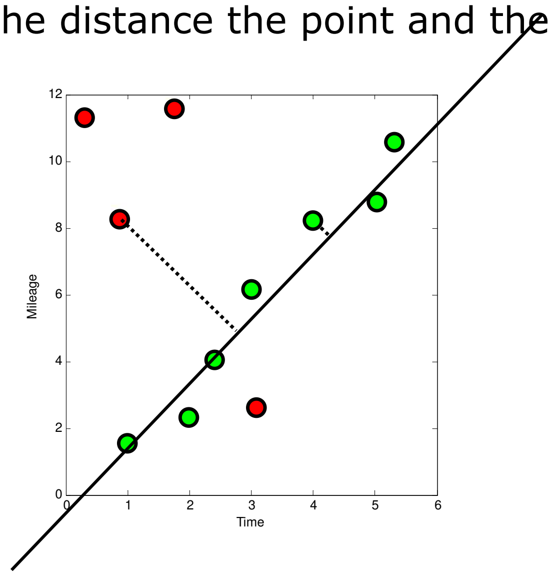
Testing goodness

- Idea: *count* the number of points that are "close" to the line



Testing goodness

- How can we tell if a point agrees with a line?
- Compute the distance the point and the line, and threshold



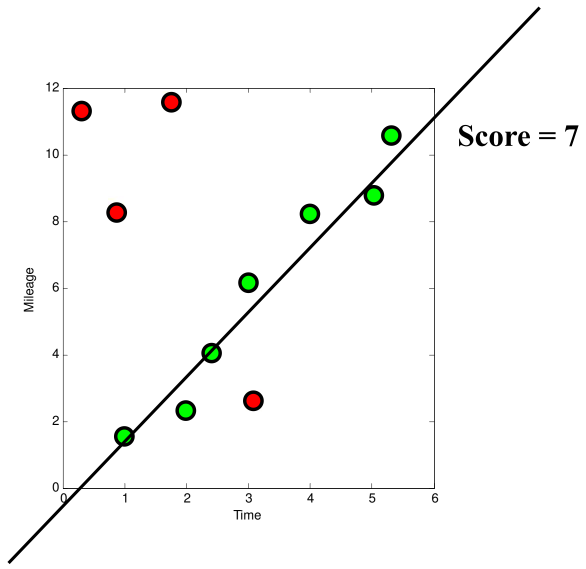
Testing goodness

- If the distance is small, we call this point an *inlier* to the line
- If the distance is large, it's an *outlier* to the line
- For an inlier point and a good line, this distance will be close to (but not exactly) zero
- For an outlier point or bad line, this distance will probably be large

- Objective function: find the line with the most inliers (or the fewest outliers)

Optimizing for inlier count

- How do we find the best possible line?



RANSAC for line fitting

Repeat ***N*** times:

- Pick ***s*** points uniformly at random (s=2)
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
- Probability *p*, that at least one random sample is free from outliers after N iterations.(e.g: p=0’99)
- Outlier ratio e.

$$\boxed{1 \quad 1 \quad e^{s^{\;N}} \quad 1 \quad p}$$

$$N \quad \log 1 \quad p \;/ \log 1 \quad 1 \quad e^{s}$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

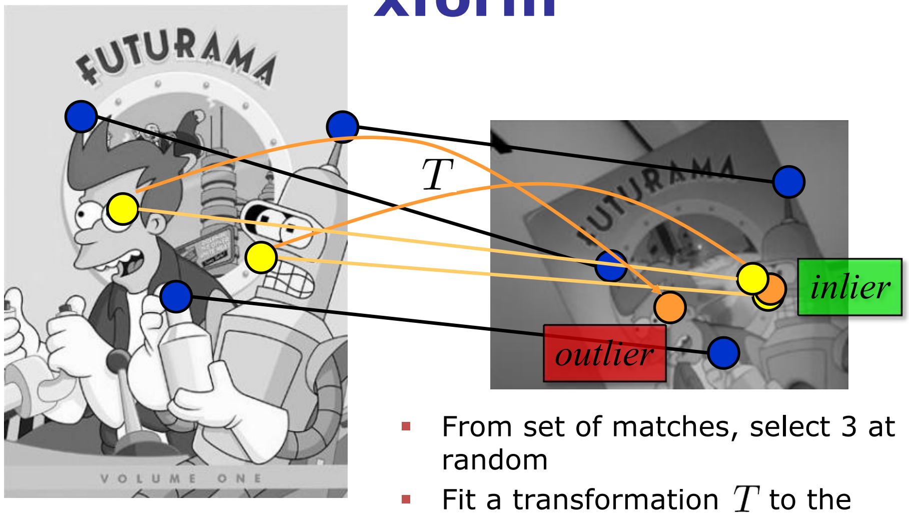
Adaptively determining the number of samples

- Inlier ratio *e* is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield *e*=0.2
- Adaptive procedure:
  - *N*=∞, *sample_count* =0
  - While *N* >*sample_count*
    - – Choose a sample and count the number of inliers
    - – Set e = 1 – (number of inliers)/(total number of points)
    - – Recompute *N* from *e:*

$$N \quad \log 1 \quad p \;/ \log 1 \quad 1 \quad e^{s}$$

    - – Increment the *sample_count* by 1

Generating and testing an xform



- From set of matches, select 3 at random
- Fit a transformation T to the selected matches
- Count inliers

Transform Fitting Algorithm (RANSAC)

1. Select 3 putative matches at random
2. Solve for the affine transformation T
3. Count the number of matches that are inliers to *T*
4. If *T* has the highest number of inliers so far, save it
5. Recompute N
6. Repeat for N rounds, return the best *T*

How do we solve for T given 3 matches?

- Three matches give a linear system with six equations:

$[\ x_1\ y_1\ ] \rightarrow [\ x_1'\ y_1'\ ]$

$$ax_1 + by_1 + c = x_1'$$
$$dx_1 + ey_1 + f = y_1'$$

$[\ x_2\ y_2\ ] \rightarrow [\ x_2'\ y_2'\ ]$

$$ax_2 + by_2 + c = x_2'$$
$$dx_2 + ey_2 + f = y_2'$$

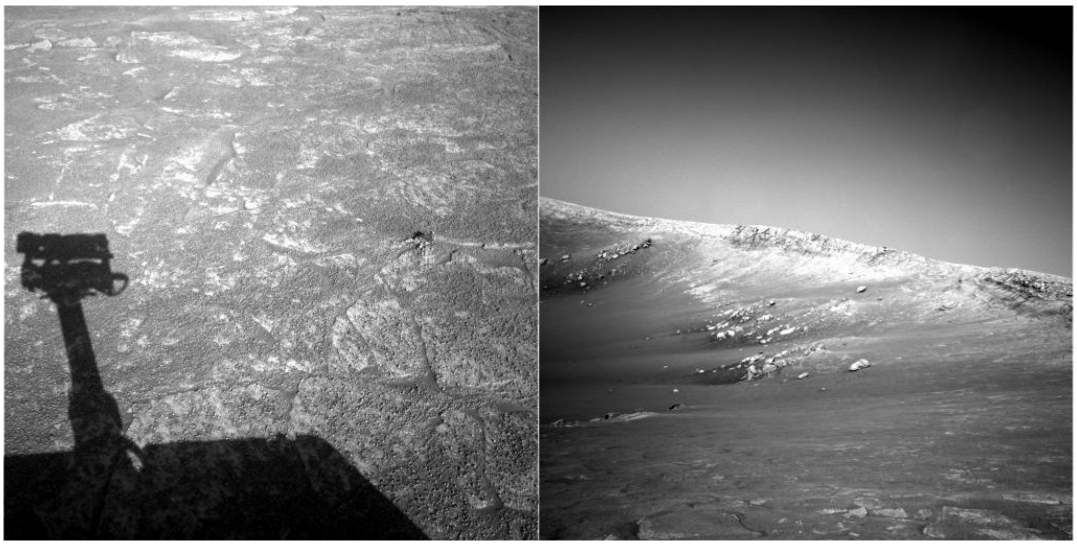
$[\ x_3\ y_3\ ] \rightarrow [\ x_3'\ y_3'\ ]$

$$ax_3 + by_3 + c = x_3'$$
$$dx_3 + ey_3 + f = y_3'$$

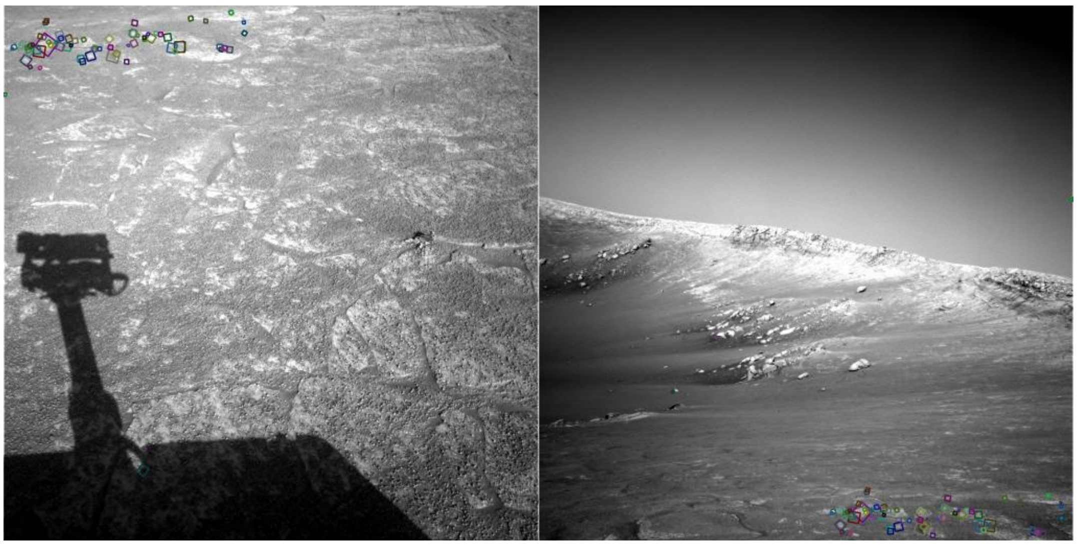
Randomized algorithms

- Very common in computer science
  - In this case, we avoid testing an infinite set of possible lines, or all $O(n^2)$ lines generated by pairs of points
- These algorithms find the right answer with some probability
- Often work very well in practice

Do these two images overlap?


NASA Mars Rover images

Answer below


NASA Mars Rover images

UPC