COMPUTER VISION
Features

Emanuel Aldea <emanuel.aldea@u-psud.fr>
http://hebergement.u-psud.fr/emi/

Computer Science and Multimedia Master - University of Pavia

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

A fundamental step for :

- estimating how cameras are located relatively to each other

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

A fundamental step for :

- estimating how cameras are located relatively to each other
- recovering scene depth

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

A fundamental step for :

- estimating how cameras are located relatively to each other
- recovering scene depth
- estimating ego-movement (visual odometry)

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

A fundamental step for :

- estimating how cameras are located relatively to each other
- recovering scene depth
- estimating ego-movement (visual odometry)
- matching image content in general

Why do we need invariant features in CV ?

Multiple views require reliable correspondences

- how do we *usually* get multiple views ?
  - we use multiple cameras simultaneously
  - one camera is moving while acquiring data - and the scene is static

A fundamental step for :

- estimating how cameras are located relatively to each other
- recovering scene depth
- estimating ego-movement (visual odometry)
- matching image content in general

The foundations of Computer Vision are based on these tasks, and features play thus a significant role in this field.

Why do we need invariant features in CV ?

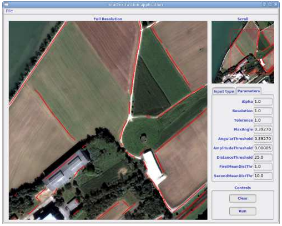
Why not use contours ?

- the processing effort is relatively low
- parametric curves may be extracted relatively easy as well (Hough)

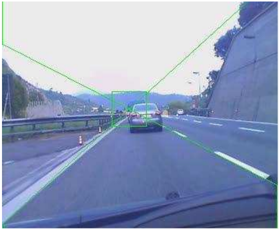
Why do we need invariant features in CV ?

Why not use contours ?

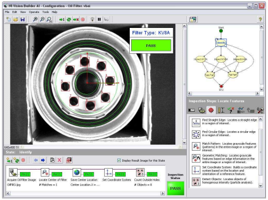
- the processing effort is relatively low
- parametric curves may be extracted relatively easy as well (Hough)
- various applications for specific environments :
  - road / panel / text detection
  - medical and satellite imagery
  - inspection for industrial vision



Aerial imagery



Lane detection



Industrial vision

Why do we need invariant features in CV ?

Why not use contours ?

- the processing effort is relatively low
- parametric curves may be extracted relatively easy as well (Hough)
- various applications for specific environments :
  - road / panel / text detection
  - medical and satellite imagery
  - inspection for industrial vision

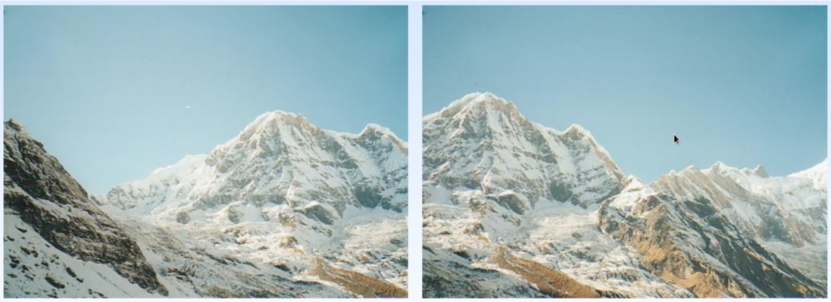
✓ Fast, specialized tasks

✓ Intensity variation invariant

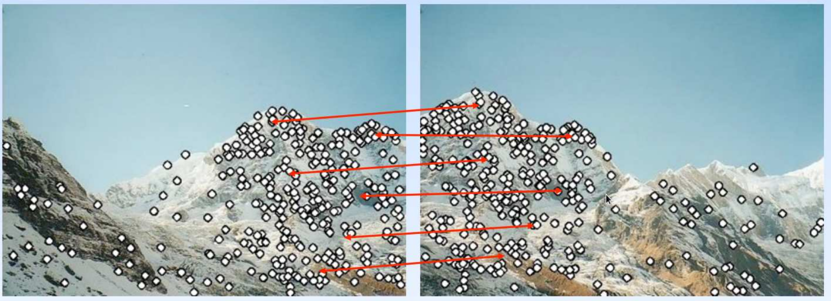
✗ Sensitive to other geometric transforms

✗ Problem for pattern recognition

Simple motivator - panoramic images

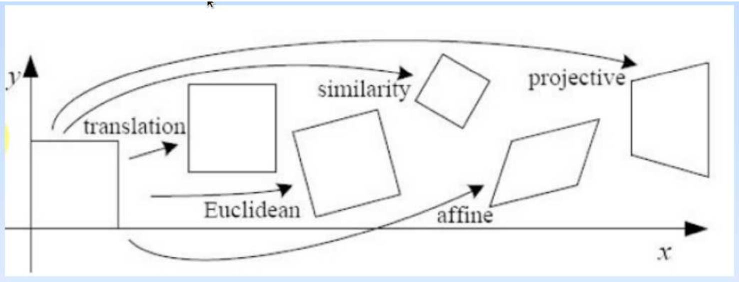


Simple motivator - panoramic images



Simple motivator - panoramic images

The core of the problem



- translation
- Euclidean (translation + rotation)
- similarity transform (tr. + rot. + scale)
- affine (rot. + scale + shear + translation)
- projective

Why we need invariance in CV

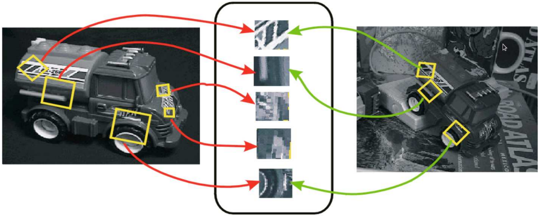
Objective

- identify structures which are invariant with respect to rotation, rescaling, etc.
- these structures are currently called interest points or corners

Why we need invariance in CV

Objective

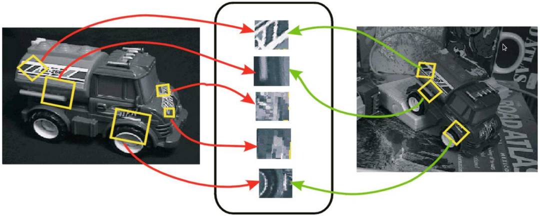
- identify structures which are invariant with respect to rotation, rescaling, etc.
- these structures are currently called interest points or corners



Why we need invariance in CV

Objective

- identify structures which are invariant with respect to rotation, rescaling, etc.
- these structures are currently called interest points or corners



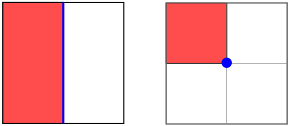
How to :

- identify them in a non supervised manner ?
- associate them robustly ?

Corner detectors : the basics

Definition

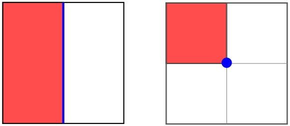
Corner : a location in the image which is characterized by strong intensity variation along two different directions.



Corner detectors : the basics

Definition

Corner : a location in the image which is characterized by strong intensity variation along two different directions.

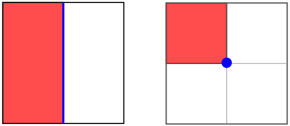


We will still need to compute the local image gradients

Corner detectors : the basics

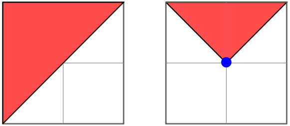
Definition

Corner : a location in the image which is characterized by strong intensity variation along two different directions.



We will still need to compute the local image gradients

- but it is not enough (to do it only in the image reference system) !



Corner detectors : the basics

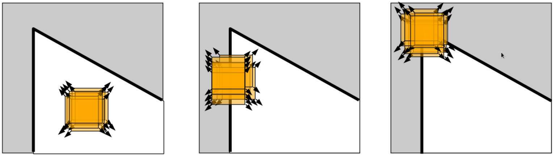
Definition

Strategy : the content of a patch centered in the corner should vary across all possible directions

Corner detectors : the basics

Definition

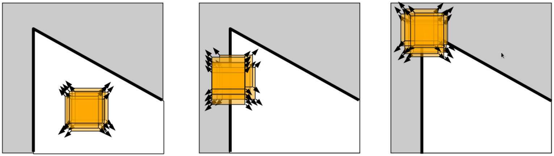
Strategy : the content of a patch centered in the corner should vary across all possible directions



Corner detectors : the basics

Definition

Strategy : the content of a patch centered in the corner should vary across all possible directions



Typical behavior :

- homogeneous regions : no change in patch content
- contours : no change along the contour
- corners : important change across all directions
- corner quality : defined by the smallest possible change
- proposed by Moravec in 1980

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x, y, \Delta x, \Delta y) = \sum_x \sum_y w(x, y) \big[I(x, y) - I(x + \Delta x, y + \Delta y) \big]^2$$

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x,y,\Delta x,\Delta y) = \sum_x \sum_y w(x,y)\big[\underbrace{I(x,y)}_{\text{intensity}} - I(x+\Delta x, y+\Delta y)\big]^2$$

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x, y, \Delta x, \Delta y) = \sum_x \sum_y w(x, y) \Big[\underbrace{I(x, y)}_{\text{intensity}} - \underbrace{I(x + \Delta x, y + \Delta y)}_{\text{shifted intensity}} \Big]^2$$

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x, y, \Delta x, \Delta y) = \sum_x \sum_y \underbrace{w(x, y)}_{\text{support}} \big[\underbrace{I(x, y)}_{\text{intensity}} - \underbrace{I(x + \Delta x, y + \Delta y)}_{\text{shifted intensity}} \big]^2$$

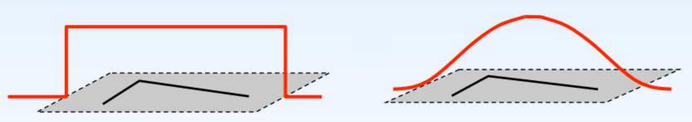


FIGURE – Possible choices for the support function $w(x, y)$

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x,y,\Delta x,\Delta y) = \sum_x \sum_y \underbrace{w(x,y)}_{\text{support}} \Big[\underbrace{I(x,y)}_{\text{intensity}} - \underbrace{I(x+\Delta x, y+\Delta y)}_{\text{shifted intensity}} \Big]^2$$

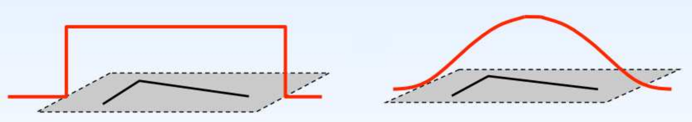


FIGURE – Possible choices for the support function $w(x,y)$

$E(x,y)$ large highlights a potential corner.

Corner detectors : the basics

Intensity change by shift of $(\Delta x, \Delta y)$

$$E(x,y,\Delta x,\Delta y) = \sum_x \sum_y \underbrace{w(x,y)}_{\text{support}} \Big[\underbrace{I(x,y)}_{\text{intensity}} - \underbrace{I(x+\Delta x, y+\Delta y)}_{\text{shifted intensity}} \Big]^2$$

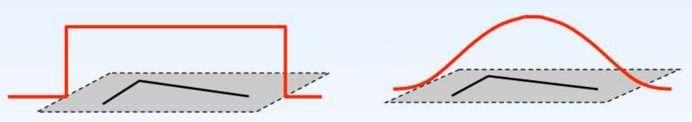


FIGURE – Possible choices for the support function $w(x,y)$

Costly if we do not use any tricks

- what is approximately the computational cost for an image of side N if we implement this method naively using a patch of side K ?

Corner detectors : the basics

First order approximation by Taylor series development

$$f(x+\Delta x, y+\Delta y) = f(x,y) + f_x(x,y)\Delta x + f_y(x,y)\Delta y$$

Corner detectors : the basics

First order approximation by Taylor series development

$$f(x+\Delta x, y+\Delta y) = f(x,y) + f_x(x,y)\Delta x + f_y(x,y)\Delta y$$

We use this approximation to rewrite the intensity variation due to shift :

$$\begin{aligned}
\sum \left[I(x+\Delta x, y+\Delta y) - I(x,y)\right]^2 &\approx \sum \left[I(x,y) + \Delta x I_x(x,y) + \Delta y I_y(x,y) - I(x,y)\right]^2 \\
&\approx \sum \Delta x^2 I_x^2 + 2\Delta x \Delta y I_x I_y + \Delta y^2 I_y^2 \\
&\approx \sum [\Delta x \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
&\approx [\Delta x \Delta y] \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
\end{aligned}$$

Corner detectors : the basics

First order approximation by Taylor series development

$$f(x+\Delta x, y+\Delta y) = f(x,y) + f_x(x,y)\Delta x + f_y(x,y)\Delta y$$

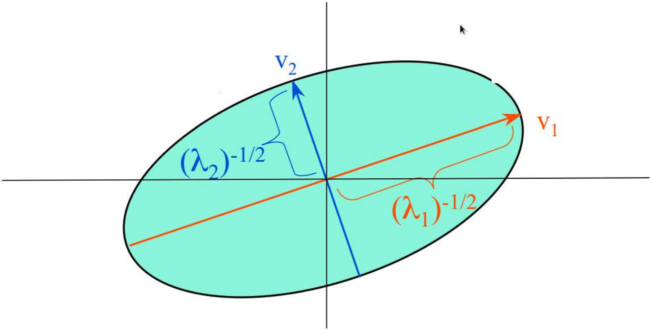
We use this approximation to rewrite the intensity variation due to shift :

$$\begin{aligned}
\sum \left[I(x+\Delta x, y+\Delta y) - I(x,y)\right]^2 &\approx \sum \left[I(x,y) + \Delta x I_x(x,y) + \Delta y I_y(x,y) - I(x,y)\right]^2 \\
&\approx \sum \Delta x^2 I_x^2 + 2\Delta x \Delta y I_x I_y + \Delta y^2 I_y^2 \\
&\approx \sum [\Delta x \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
&\approx [\Delta x \Delta y] \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
E(x,y,\Delta x,\Delta y) &\approx [\Delta x \Delta y] \left(\sum g(\sigma_I) \star \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
&\approx [\Delta x \Delta y] \underbrace{\begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}}_{\text{structure tensor}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
\end{aligned}$$

Corner detectors : the structure tensor

Properties

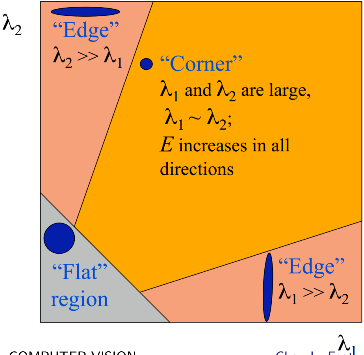
- the eigenvectors highlight the main directions of gradient variation around the location we consider (see the ellipse of constant change)
- ex. : if $\lambda_2 > \lambda_1$, strong variation along v_2 and smaller variation in the direction of v_1
- if corner, λ_1, λ_2 are large



Corner detectors : the structure tensor

Properties

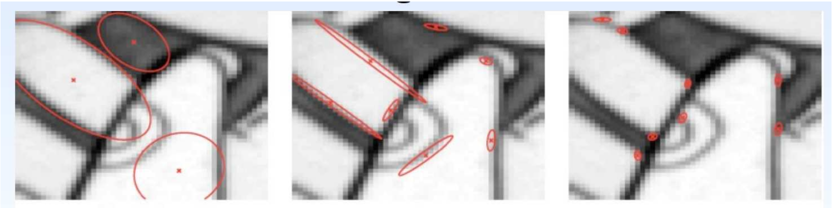
- the eigenvectors highlight the main directions of gradient variation around the location we consider (see the ellipse of constant change)
- ex. : if $\lambda_2 > \lambda_1$, strong variation along v_2 and smaller variation in the direction of v_1
- if corner, λ_1, λ_2 are large



Corner detectors : the structure tensor

Properties

- the eigenvectors highlight the main directions of gradient variation around the location we consider (see the ellipse of constant change)
- ex. : if $\lambda_2 > \lambda_1$, strong variation along v_2 and smaller variation in the direction of v_1
- if corner, λ_1, λ_2 are large



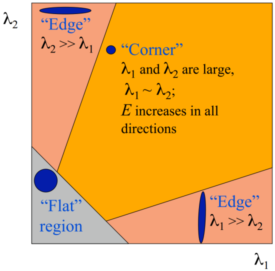
Corner detectors : the structure tensor

Decision based on the tensor eigenvalues

- one may compute λ_1, λ_2 explicitly, but too costly
- prefered method :

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

- the value of parameter α is usually 0.04 - 0.06
- interesting eigenvalues = local maxima of R



Corner detectors : Harris detector

Main algorithm steps

1. compute gradients $I_x = \frac{\partial}{\partial x} g(\sigma_D) \star I$, $I_y = \frac{\partial}{\partial y} g(\sigma_D) \star I$
2. compute the structure tensor :

$$M = g(\sigma_I) \star \left[\begin{array}{cc} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{array} \right]$$

3. compute the response function R :

$$R = det(M) - \alpha trace^2(M)$$

4. apply thresholding to R
5. non maximal suppression on the values of R

Corner detectors : example

FIGURE – Initial pair

Corner detectors : example

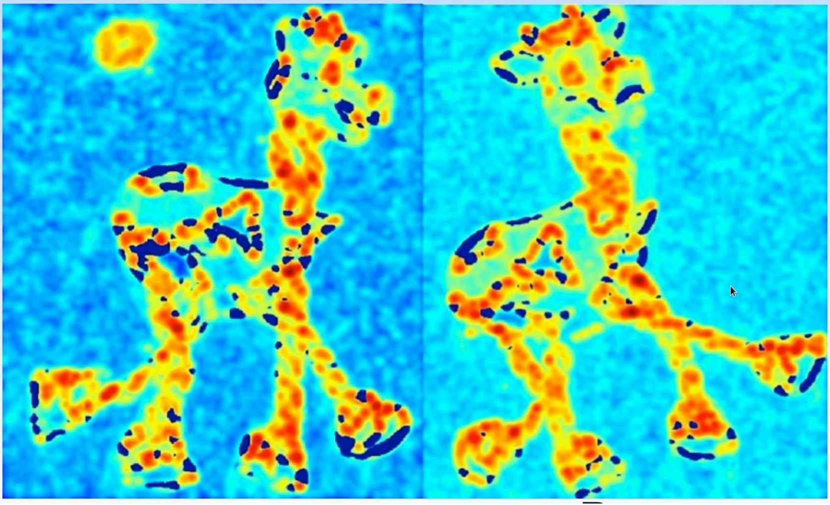


FIGURE – response function R

Corner detectors : example

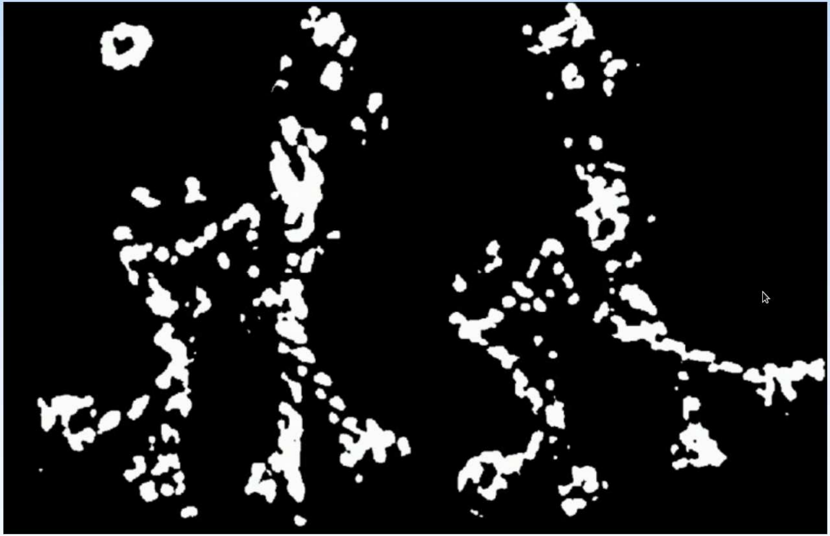


FIGURE – Thresholding R

Corner detectors : example

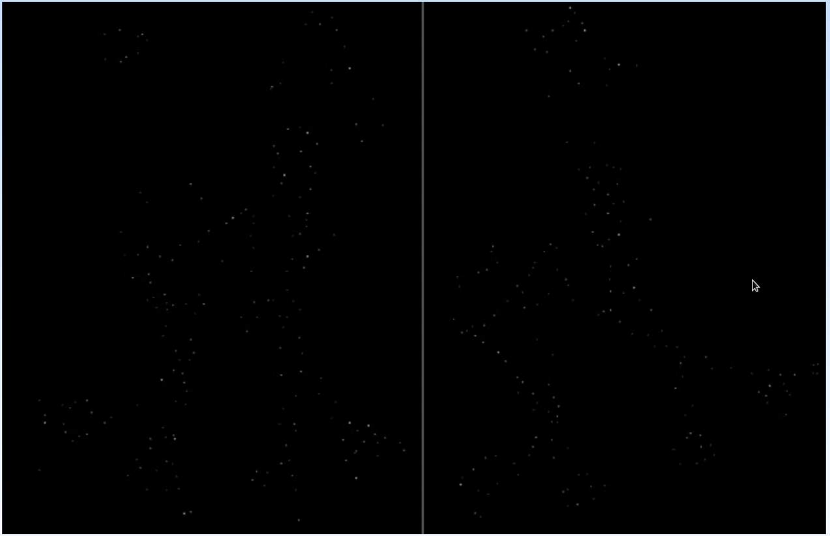


FIGURE – Non maximal suppression on R

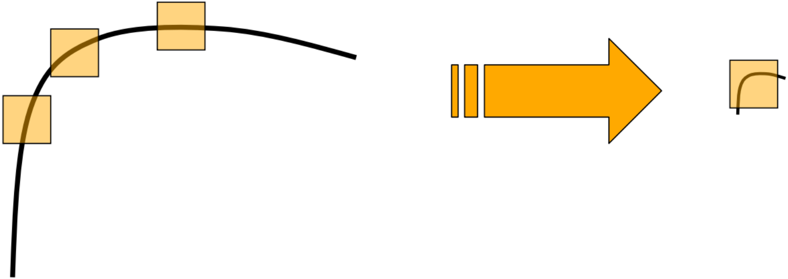
Corner detectors : example

FIGURE – Detector results

Conclusion : Harris detector

Conclusions

- ✓ rotation invariant detector
- ✓ intensity change invariant
- ✗ not robust to scale change
- ✗ no descriptor provided for matching



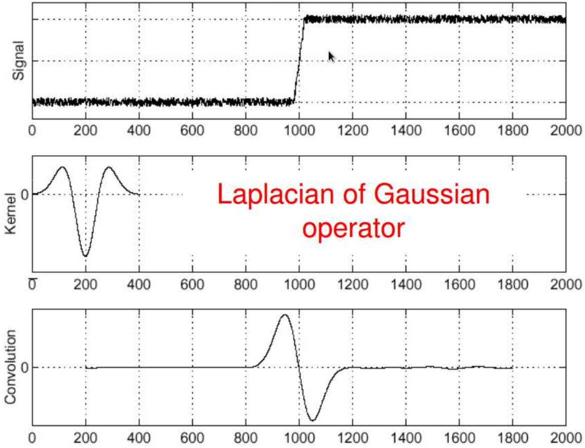
The characteristic scale

Short intro to Laplacian filtering :

$$f$$

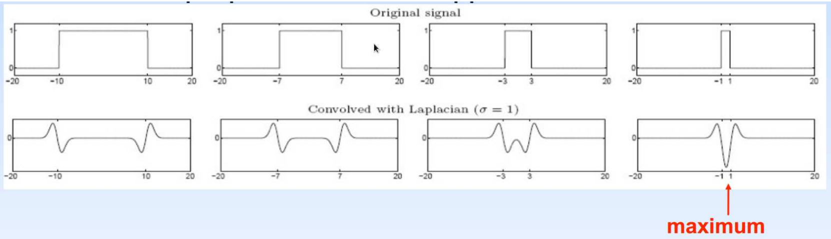
$$\frac{\partial^2}{\partial x^2}h$$

$$\left(\frac{\partial^2}{\partial x^2}h\right) \star f$$



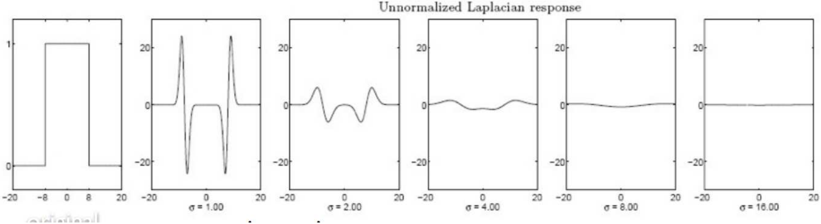
Gaussian filter + Laplace (LoG) - zero crossing

The characteristic scale



The Laplacian response - maximal if the Laplacian scale corresponds to the scale of the variation in the image space

The characteristic scale



If one varies σ, the Laplacien response varies as well ; the operation has to be normalized by a multiplication by σ^2

The characteristic scale

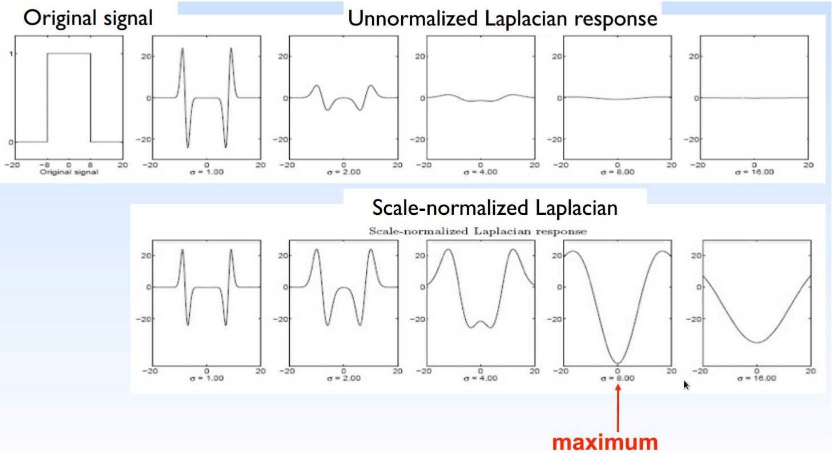
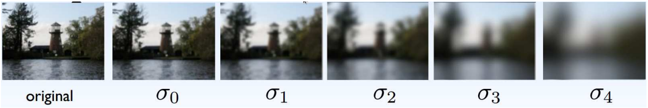
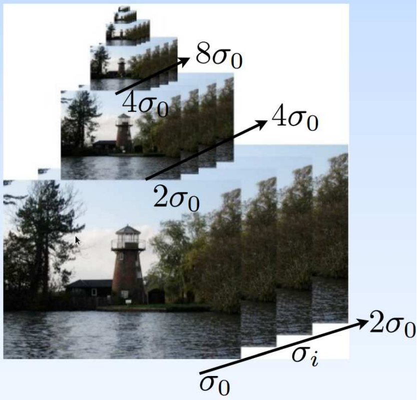


FIGURE – Multi scale normalized Laplacian response

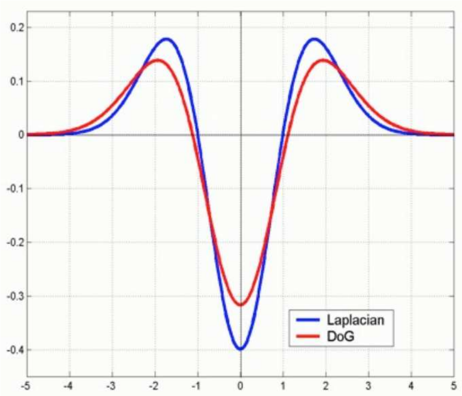
The pyramid representation





$$\sigma_i = 2^{\frac{i}{s-1}} \sigma_0$$

Approximating the Laplacian



Laplacian :

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

Difference of Gaussians :

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

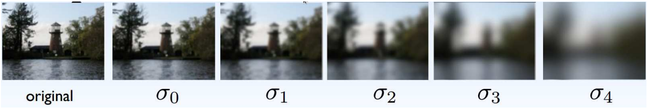
The SIFT detector

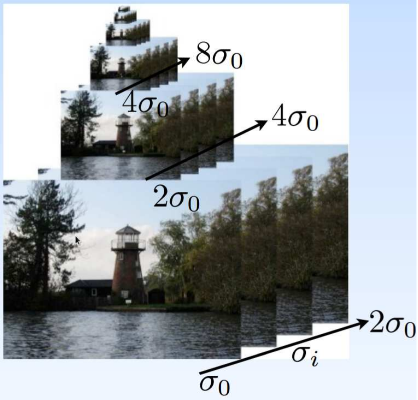
Scale Invariant Feature Transform

- high performance
- very costly
- the descriptor is integrated (it is also provided by the algorithm)

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

The pyramid representation



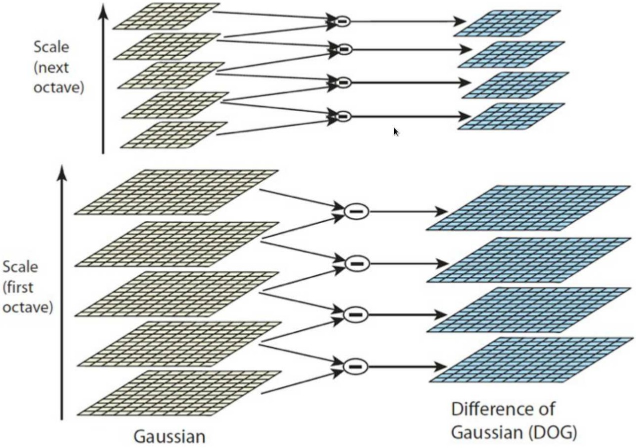


$$\sigma_i = 2^{\frac{i}{s-1}} \sigma_0$$

The SIFT detector

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

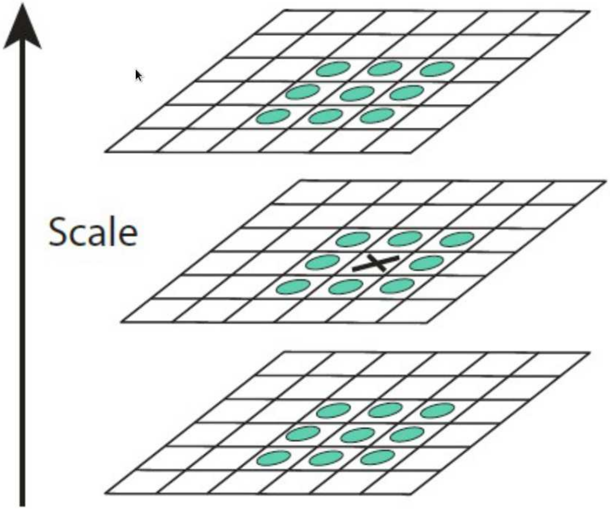
Computing the DoGs



The SIFT detector

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

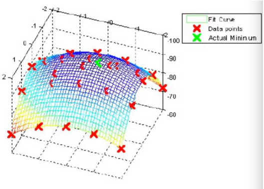
Identifying the extrema



The SIFT detector

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

Sub-pixel localization



Interpolation of discrete values of $D(x, y, \sigma)$. Use of the Taylor series second order development :

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

Solution :

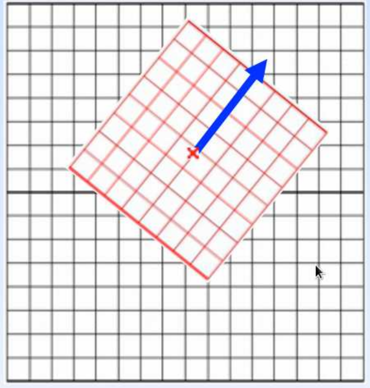
$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

The SIFT detector

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

Computing the orientation

1. Compute local gradients at the characteristic scale
2. Compute local gradient histogram
3. The canonic orientation is the maximal direction
4. Each corner is characterized by : location, scale, orientation
5. Local coordinate system for building up the descriptor

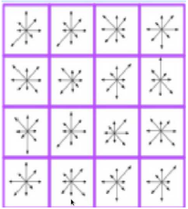


The SIFT detector

1. Construction of the scale space
2. Computing the DoGs
3. Computing the characteristic scale
4. Sub-pixel localization
5. Eliminating contour responses
6. Computing the orientation
7. Computing the descriptor

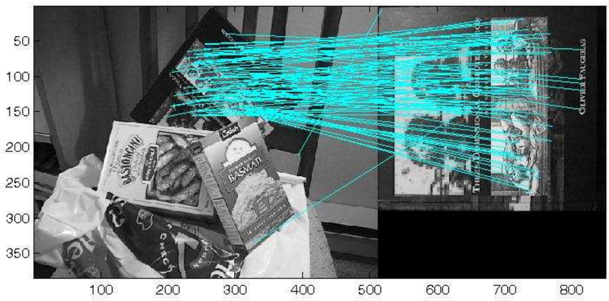
Computing the descriptor

1. Local gradient orientations in 16 neghboring regions
2. Coordinate system defined by the corner
3. 4*4*8 orientations = 128 (descriptor dimension)



Conclusions about SIFT

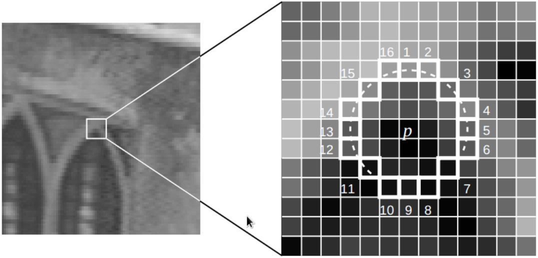
- Scale invariant
- Rotation invariant
- Illumination invariant
- Perspective invariant
- Costly



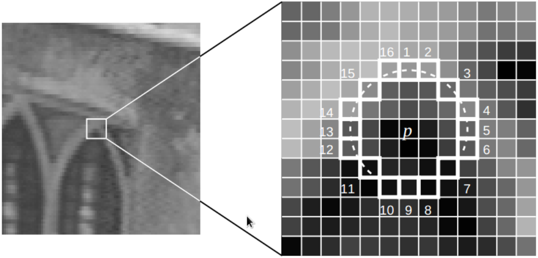
The FAST detector

Features from Accelerated Segment Test

- extremely fast
- no complex operations (convolution, gradient computation etc.)
- not too robust
- no descriptor



The FAST detector - strategy



$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \\ s, & I_p - t < I_{p\to x} < I_p + t \\ b, & I_p + t \leq I_{p\to x} \end{cases}$$

The FAST detector

Question 1

Sketch a naive implementation in order to test whether a pixel is a FAST corner or not.

The FAST detector

Question 2

How many possible configurations are in total ?
How many coin configurations $c \in Q$ are there ?
What does the following function :

$$H(Q) = (c + \bar{c}) \log(c + \bar{c}) - c \log c - \bar{c} \log \bar{c}$$

represent ?

The FAST detector

Question 3

Given that the entropy gain is :

$$H_g = H(Q) - H(A) - H(B)$$

where $Q = A \cup B$, think of a trick in order to improve the test that you proposed for Question 1.

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching
- otherwise, a simple solution is patch matching : a patch is extracted around the corner, and matched against a candidate in the destination image using a correlation, SSD or SAD function

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching
- otherwise, a simple solution is patch matching : a patch is extracted around the corner, and matched against a candidate in the destination image using a correlation, SSD or SAD function
- other solutions exist (BRIEF, FREAK etc.)

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching
- otherwise, a simple solution is patch matching : a patch is extracted around the corner, and matched against a candidate in the destination image using a correlation, SSD or SAD function
- other solutions exist (BRIEF, FREAK etc.)

Tricks used commonly in order to improve matching quality

- these tricks usually increase the computation time but remove false matches (and also some good matches sometimes)

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching
- otherwise, a simple solution is patch matching : a patch is extracted around the corner, and matched against a candidate in the destination image using a correlation, SSD or SAD function
- other solutions exist (BRIEF, FREAK etc.)

Tricks used commonly in order to improve matching quality

- these tricks usually increase the computation time but remove false matches (and also some good matches sometimes)
- married matching : the best candidate has to pick up the initial corner as best candidate as well

Corner association (matching)

How to do it ?

- matching needs to be fast and reliable
- if the detector provides a descriptor (i.e. SIFT), use it for matching
- otherwise, a simple solution is patch matching : a patch is extracted around the corner, and matched against a candidate in the destination image using a correlation, SSD or SAD function
- other solutions exist (BRIEF, FREAK etc.)

Tricks used commonly in order to improve matching quality

- these tricks usually increase the computation time but remove false matches (and also some good matches sometimes)
- married matching : the best candidate has to pick up the initial corner as best candidate as well
- ranking : the second match must have a significantly larger distance/lower similarity than the best match, in order to avoid confusion between similarly looking corner

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image
- plenty other detectors which provide some advantage in terms of either computational time or some invariance : SURF, AGAST, ORB, HOOFR etc.

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image
- plenty other detectors which provide some advantage in terms of either computational time or some invariance : SURF, AGAST, ORB, HOOFR etc.

Which detector to choose ?

- the choice is application dependent

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image
- plenty other detectors which provide some advantage in terms of either computational time or some invariance : SURF, AGAST, ORB, HOOFR etc.

Which detector to choose ?

- the choice is application dependent
- FAST : great for real time robotic navigation

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image
- plenty other detectors which provide some advantage in terms of either computational time or some invariance : SURF, AGAST, ORB, HOOFR etc.

Which detector to choose ?

- the choice is application dependent
- FAST : great for real time robotic navigation
- SIFT : useful when quality is important

Detectors - conclusion

Overview

- FAST : not so robust, no descriptor provided - but runs in 1ms on a regular image ;
- Harris : slightly more robust, no descriptor provided - runs in 25-40ms on a regular image
- SIFT : very robust, descriptor provided - runs in 2-5 seconds on a regular image
- plenty other detectors which provide some advantage in terms of either computational time or some invariance : SURF, AGAST, ORB, HOOFR etc.

Which detector to choose ?

- the choice is application dependent
- FAST : great for real time robotic navigation
- SIFT : useful when quality is important
- most other descriptors provide a compromise between robustness and cost