Feature detection and description

MVA/IMA – Vision 3D artificielle

Mathieu Aubry
http://imagine.enpc.fr/~aubrym/

10/31/2017

With many slides from Renaud Marlet

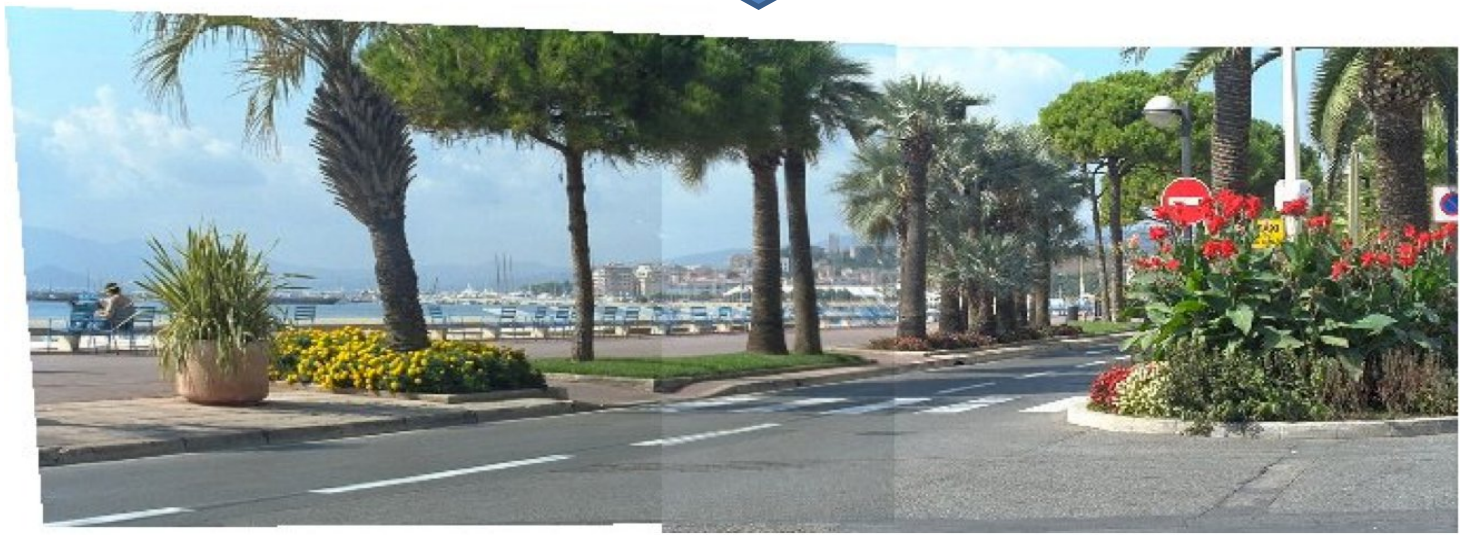
Goal/Key concepts

Goal: “Draw” correspondences between images

Mathematical tools and practice for feature:

1. Detection: Harris/Laplacian/Hessian
2. Description: HOG/BRIEF/CNNs
3. Practice: SIFT/SURF

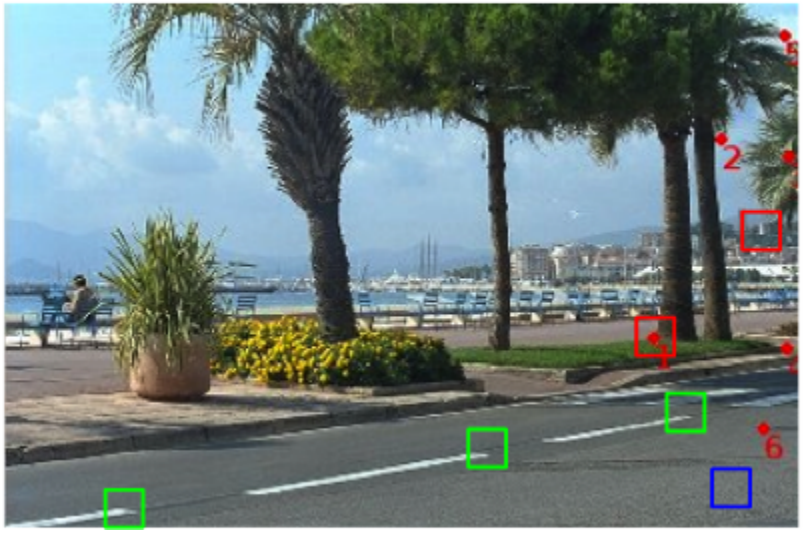
Motivation: panorama

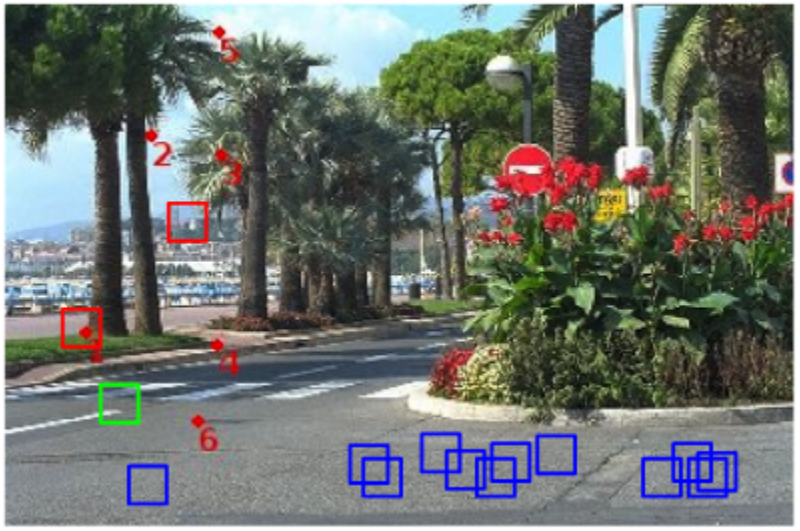


Motivation: panorama

1. Which features?
2. How to describe them?
3. How to match them?

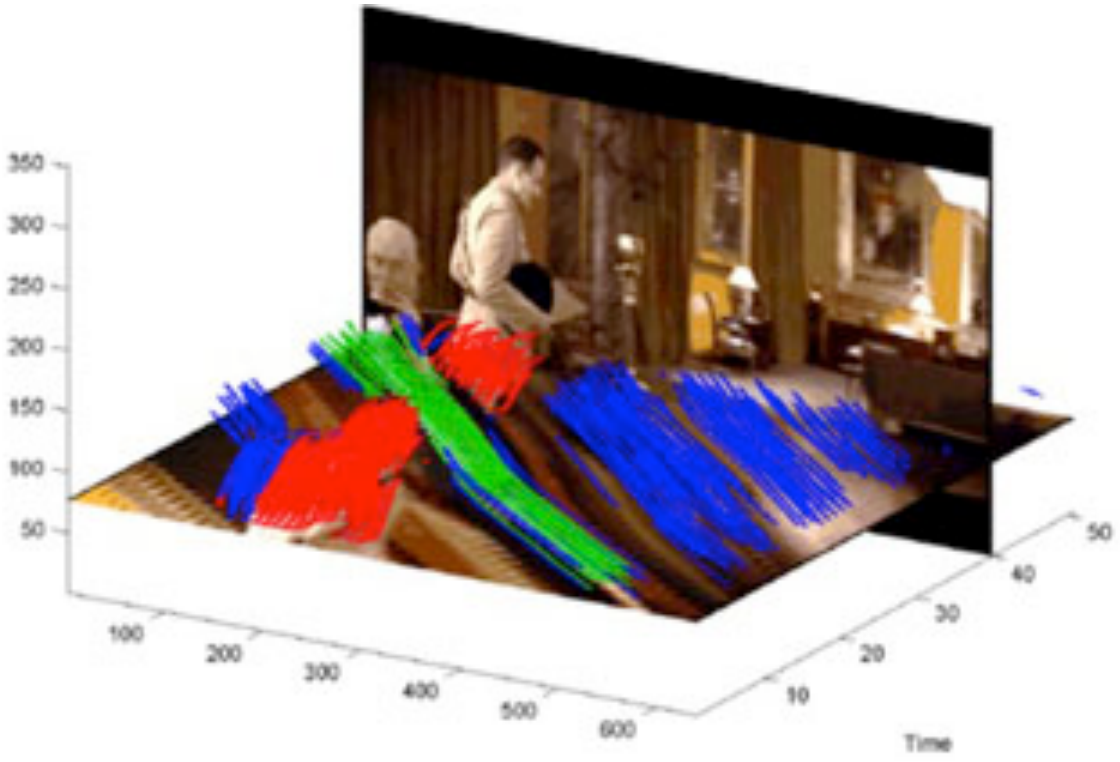
Motivation: panorama





1. Find distinctive features
2. Match similar features
3. Compute the Fundamental matrix (last lecture)

Motivation: tracking

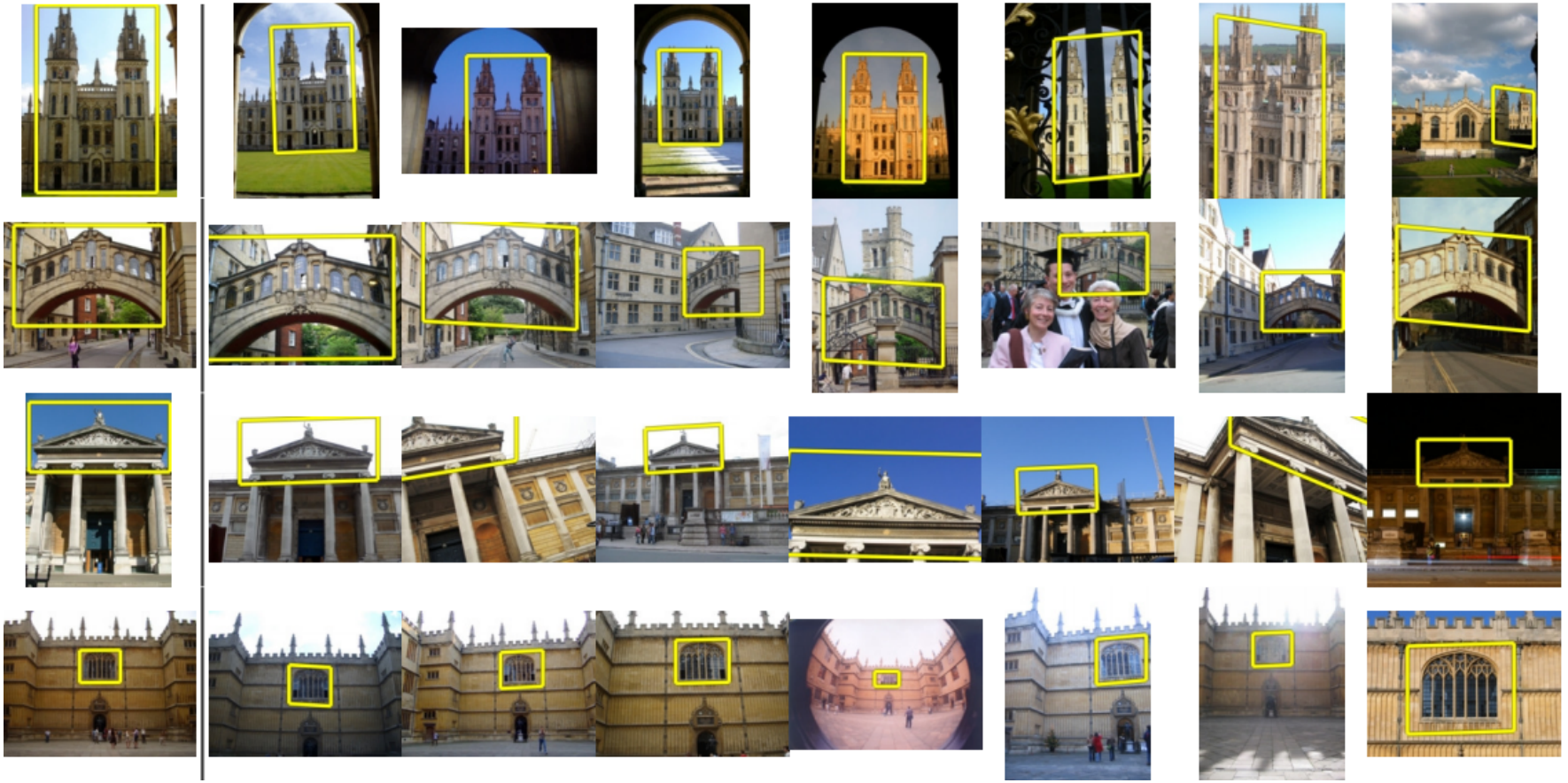


J. Lezama, K. Alahari, J. Sivic, I. Laptev

Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues

CVPR 2011

Motivation: content-based image retrieval



Philbin, J. , Chum, O. , Isard, M. , Sivic, J. and Zisserman, A.
Object retrieval with large vocabularies and fast spatial matching
CVPR 2007

Motivation: instance retrieval

1. Identify salient points

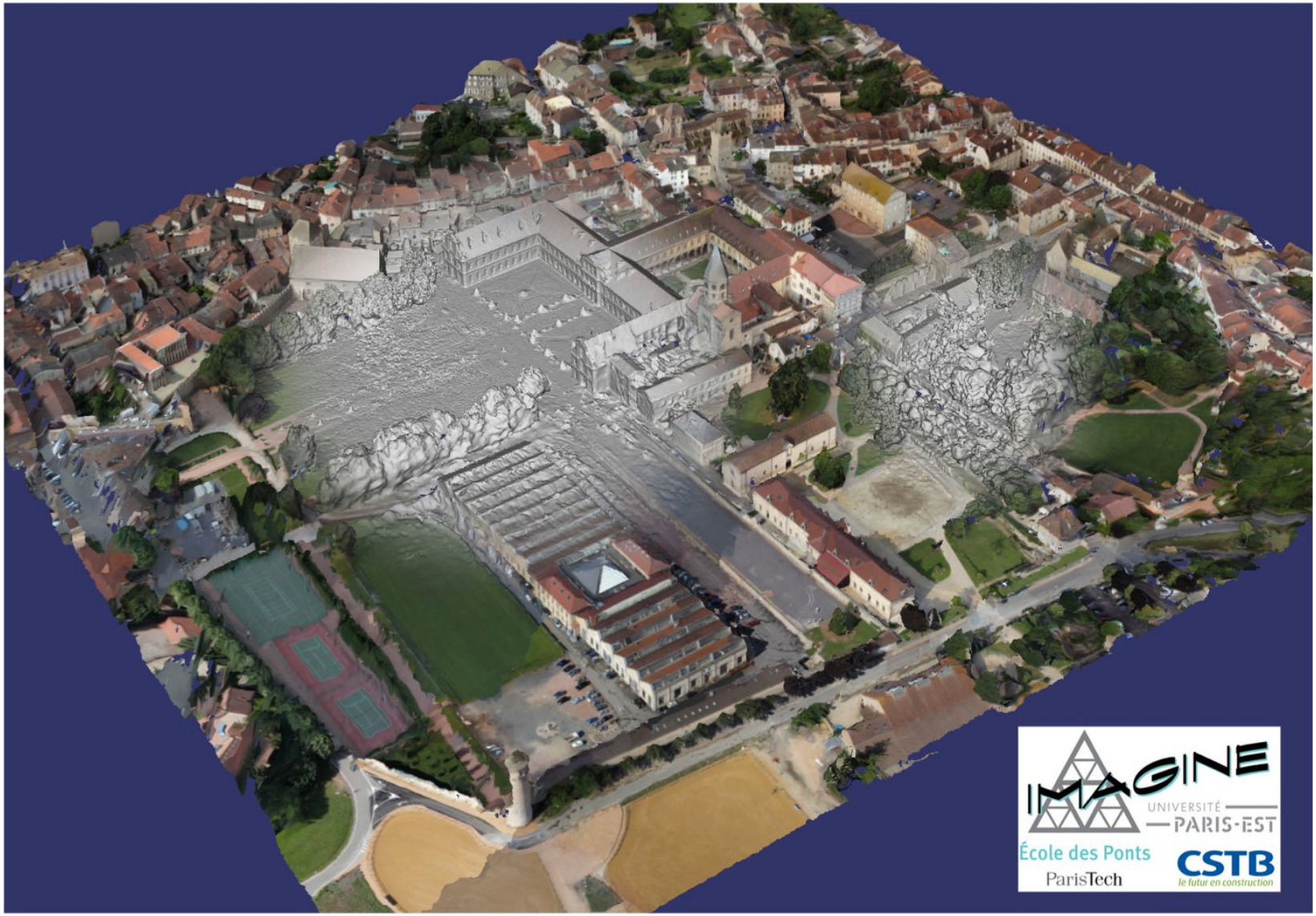
2. Look for similar salient points in other image

3. Check geometrical consistency (rigid or deformable)

Motivation: 3D reconstruction

1. Identify salient points
2. Match similar points
3. Estimate camera pose
4. 3D reconstruct by triangulation

Motivation: 3D reconstruction



Application: virtual reality

reconstitution © ENSAM Cluny

Largest religious building (187m) in Europe in the Middle Age, until St Peter's construction. Demolished during the French revolution.

Application: virtual reality

Application: virtual reality

Advertisement

If you are interested in historical/artistic data

Looking for interns/PhD students on
Computer Vision for Humanities

Including:

- 3D reconstruction from historical depictions
- Deep Learning based

3D reconstruction

- External camera calibration

  = determination of pose (i.e., location and orientation) of each camera in a common coordinate system

  - requires enough corresponding points in several images

  → **detection** and **matching** of salient points

- Dense 3D reconstruction

  = by triangulation, given camera pose
  (!) not restricted to salient points only

  - requires **matching** image patches in several images

Difficulty

Viewpoint changes

Difficulty

Illumination changes

Difficulty

Season changes

Difficulty

Clutter and occlusion

Difficulty

- change of scale (camera motion or change of focal length)
- change of orientation (rotation)
- change of viewpoint (affine, projective transformations)
- change of illumination (time of day, weather, flash...)

And also

- change of camera parameters (speed/aperture ...)
- non-rigid scene (objects in motion, deformable surface)
- surface reflectance (Lambertian or not, reflection,transpar.)
- repetitive patterns (windows, road marks...)

Problematic

- How to extract informative features consistently?
- How to compare features?

Outline

Preliminaries: convolutions, correlations, derivatives

1. Feature detection:
   Harris (Corner)
   Laplacian, Hessian (Blob)

2. Feature description and comparison:
   SSD, ZNCC, HOG, BRIEF, CNN

3. SIFT

Outline

Preliminaries: convolutions, correlations, derivatives

1. Feature detection:
   Harris (Corner)
   Laplacian, Hessian (Blob)

2. Feature description and comparison:
   SSD, ZNCC, HOG, BRIEF, CNN

3. SIFT

Convolutions, correlations, derivatives

Convolution

- Continuous convolution (1D)
  - $f, g : \mathbb{R} \to \mathbb{R}$
  - $(f * g)(x) = \int_{-\infty}^{+\infty} f(u)\, g(x-u)\, du = \int_{-\infty}^{+\infty} f(x-u)\, g(u)\, du$
- Discrete convolution (1D)
  - $f, g : \mathbb{Z} \to \mathbb{R}$
  - $(f * g)(n) = \sum_{m=-\infty}^{+\infty} f(m)\, g(n-m) = \sum_{m=-\infty}^{+\infty} f(n-m)\, g(m)$

... if integral/sum exists

- sufficient: f compactly supported, f integrable and g bounded...

Convolution

- Extension to dimension d
  - $f, g : \mathbb{R}^d$ or $\mathbb{Z}^d \to \mathbb{R}$ (or with values in $\mathbb{C}$)
- Ex. 2D continuous convolution

$$(f*g)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(u,v)\,g(x-u,y-v)\,du\,dv = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x-u,y-v)\,g(u,v)\,du\,dv$$

- Ex. 2D discrete convolution

$$(f*g)(i,j) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} f(i,j)\,g(i-k,j-l) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} f(i-k,j-l)\,g(k,l)$$

... if integral/sum exists

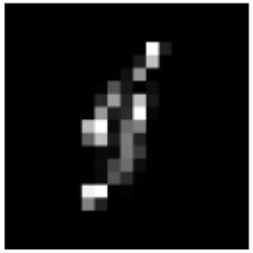
- sufficient: f compactly supported, f integrable and g bounded...

- Efficient convolution in Fourier

Blur-Convolution

- Blurred image: $O = K * I$

e.g. uniform motion blur



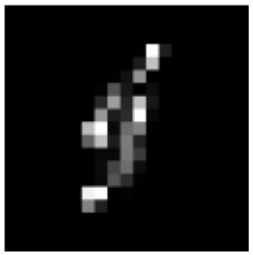
Levin, A., Weiss, Y., Durand, F., & Freeman, W. T. (2009, June). Understanding and evaluating blind deconvolution algorithms. *CVPR 2009*

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$*$

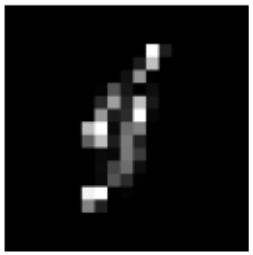


K

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

$*$

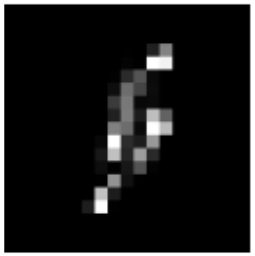


I K

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

$*$



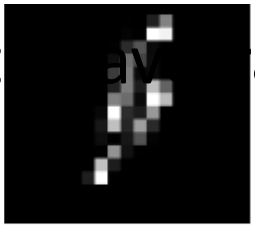
I $\qquad\qquad$ $K(-.,-.)$

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$(K * I)(i,j)$

$*$ Weight  age

$(K * I)(i,j)$

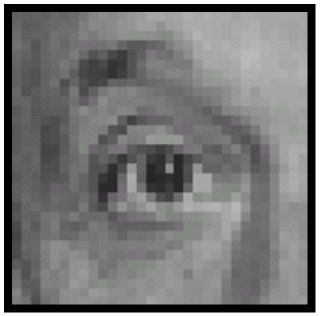
Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$(K * I)(i,j)$

Practice with linear filters



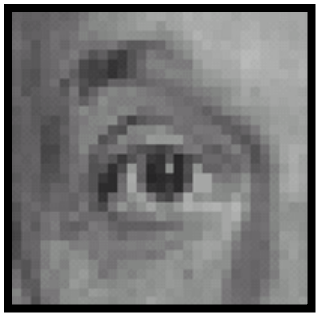
Original

0	0	0
0	1	0
0	0	0

?

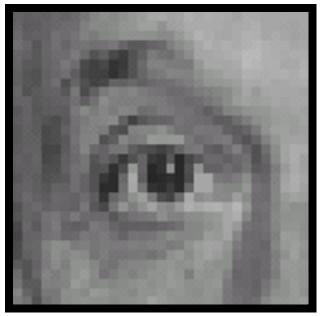
Source: D. Lowe

Practice with linear filters



Original

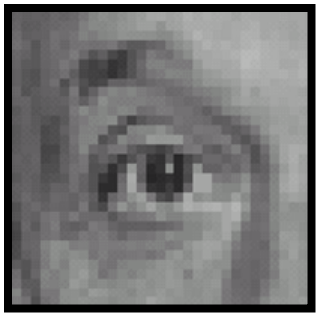
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



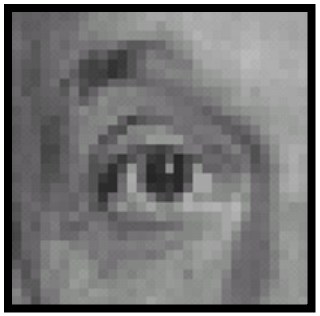
Original

0	0	0
0	0	1
0	0	0

?

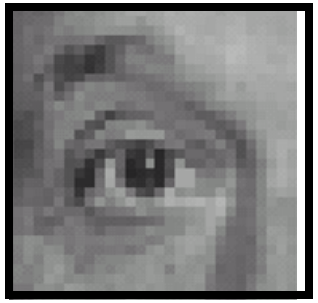
Source: D. Lowe

Practice with linear filters



Original

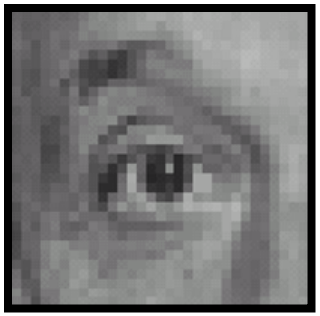
0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters



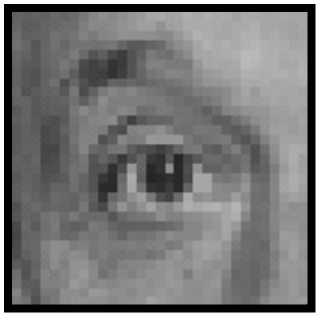
Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

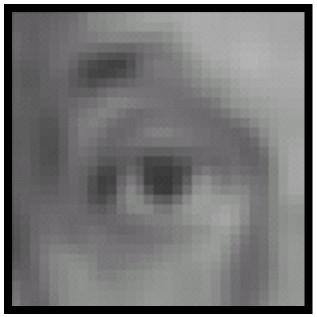
Source: D. Lowe

Practice with linear filters



Original

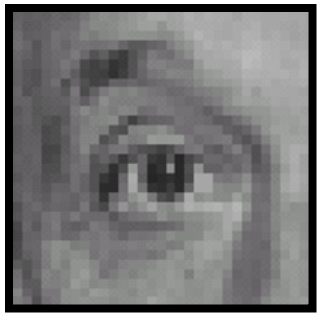
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



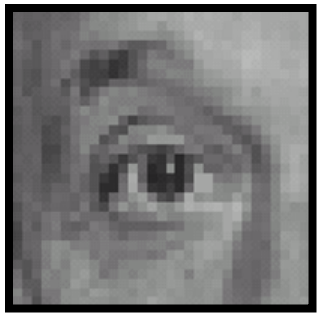
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters



Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

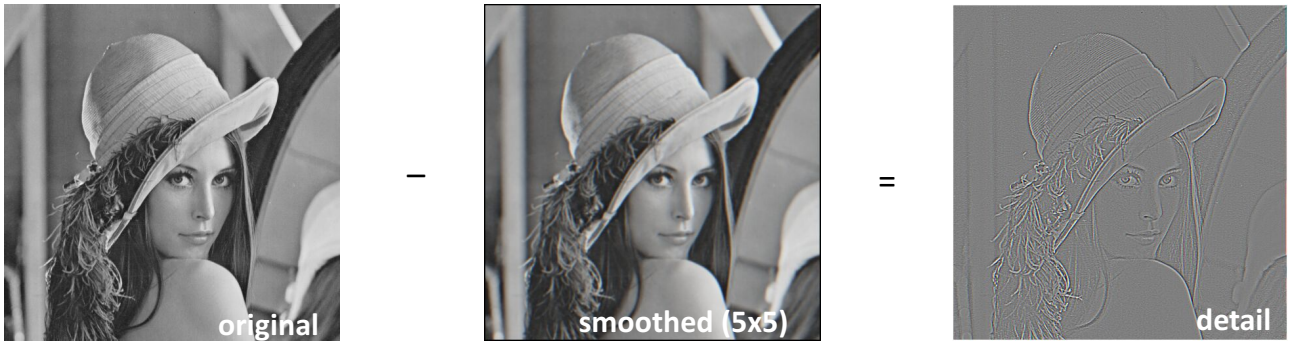
Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

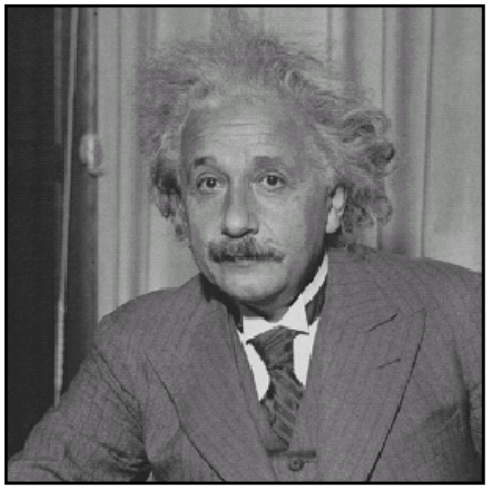
Sharpening

- What does blurring take away?

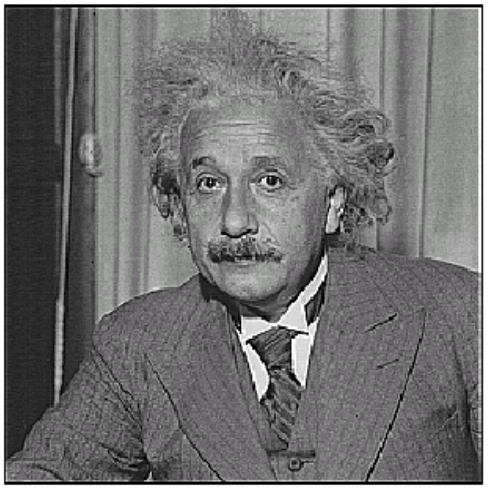


Let’s add it back:

Sharpening



before

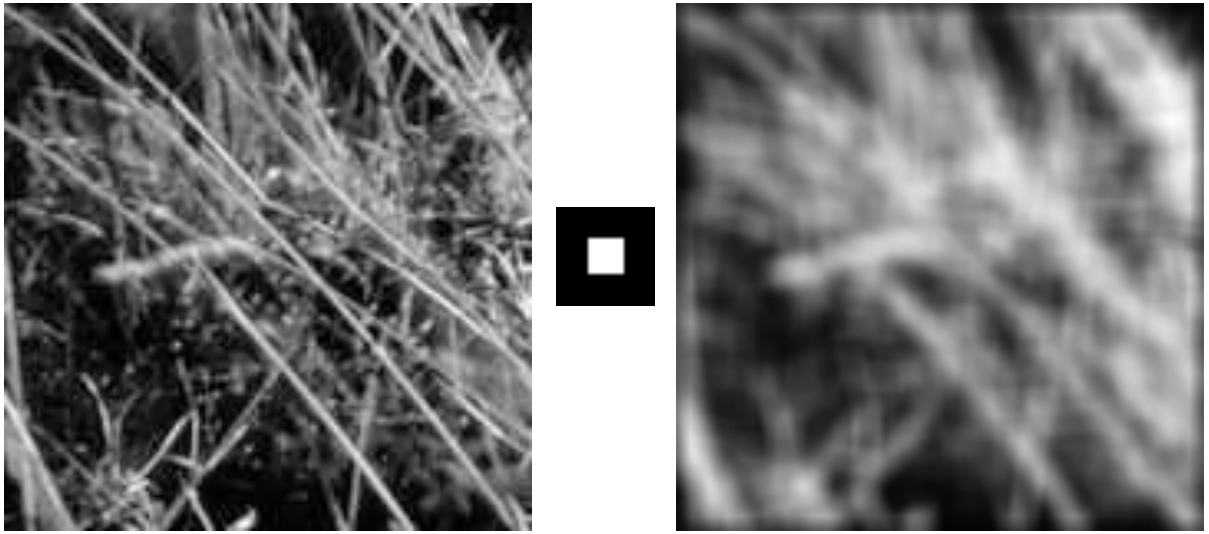


after

Source: D. Lowe

Smoothing with box filter revisited

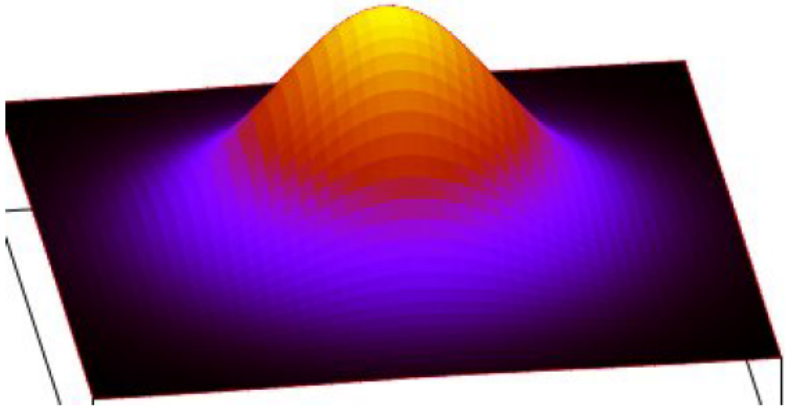
- What's wrong with this picture?
- What's the solution?

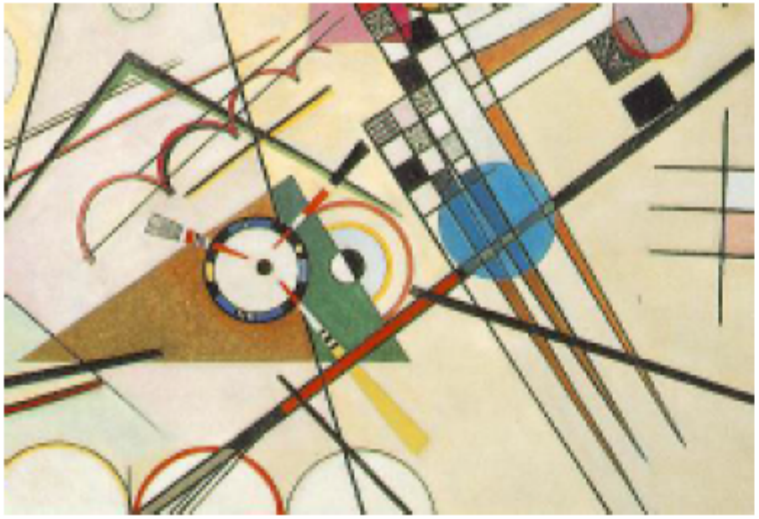


Source: D. Forsyth

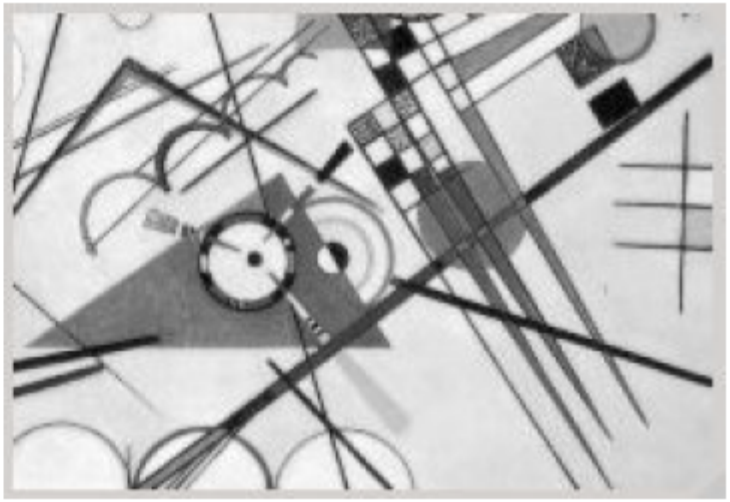
Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$





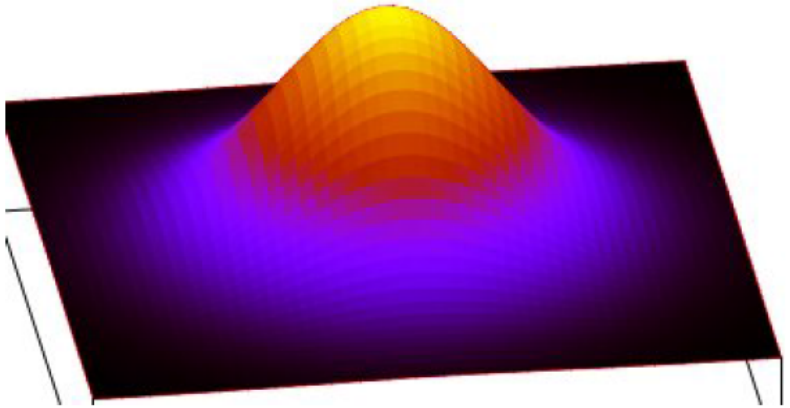
extrait de Kandinsky

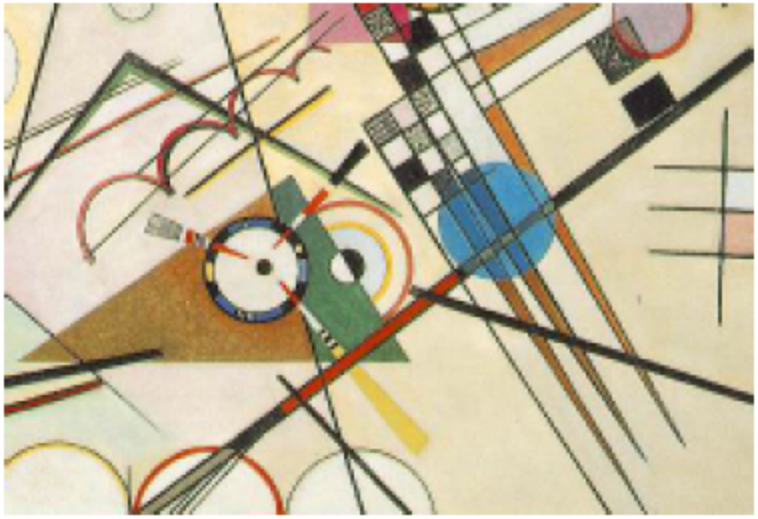


$\sigma = 0.5$

Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$



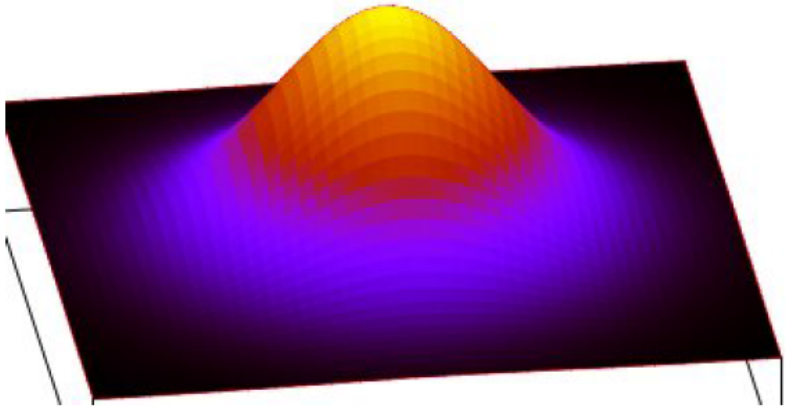


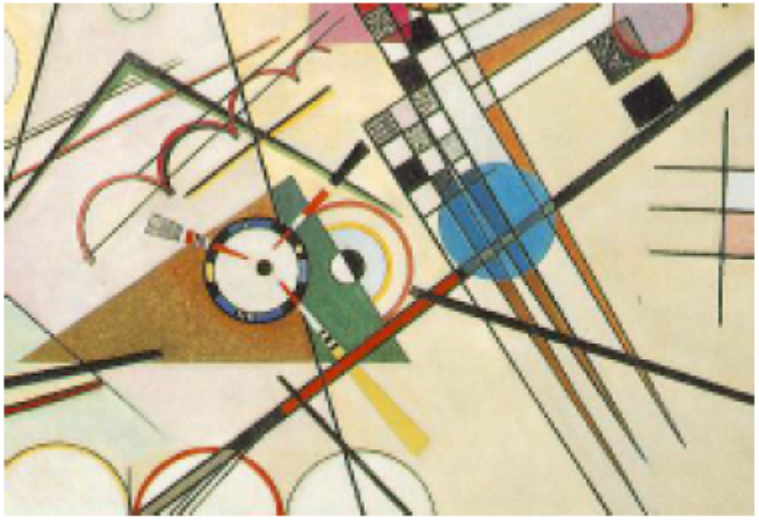
extrait de Kandinsky

$\sigma = 1.0$

Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$



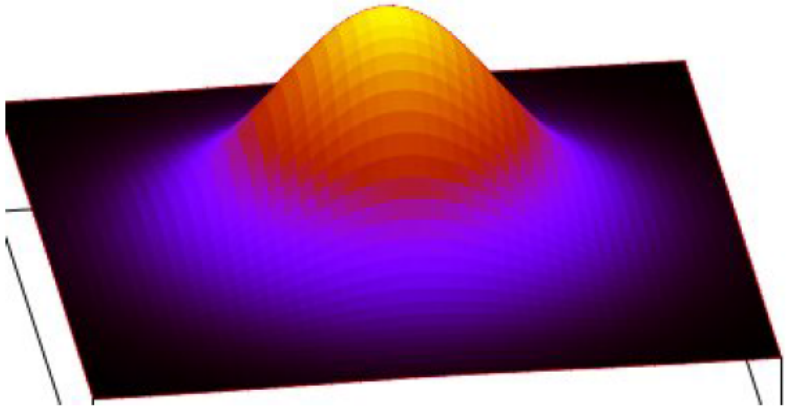


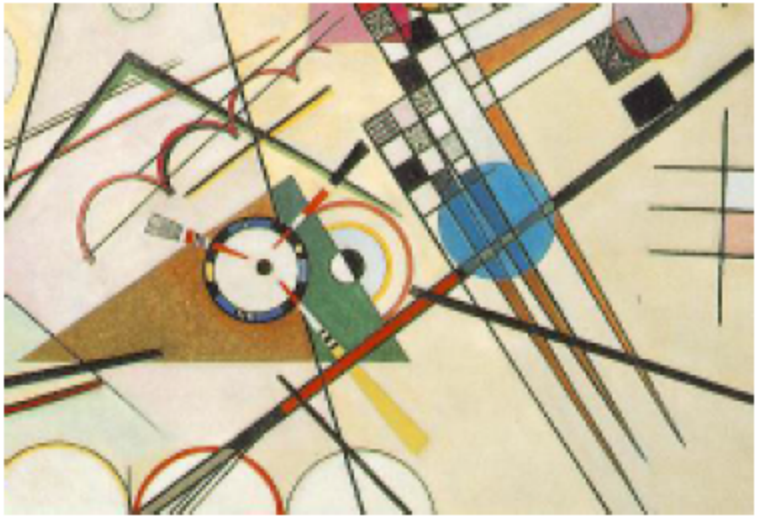
extrait de Kandinsky

$\sigma = 1.5$

Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$



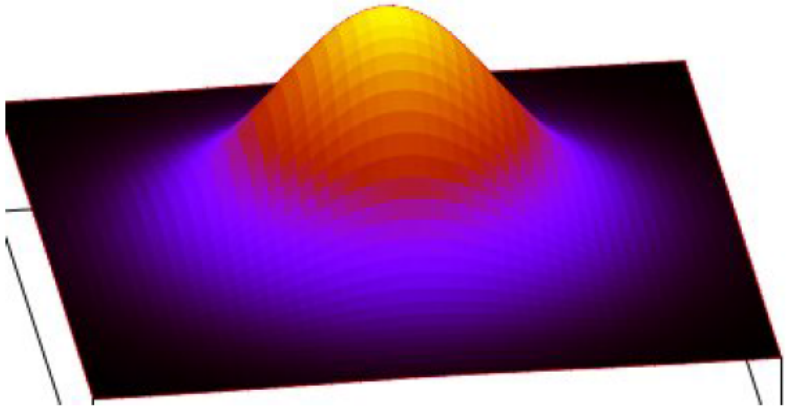


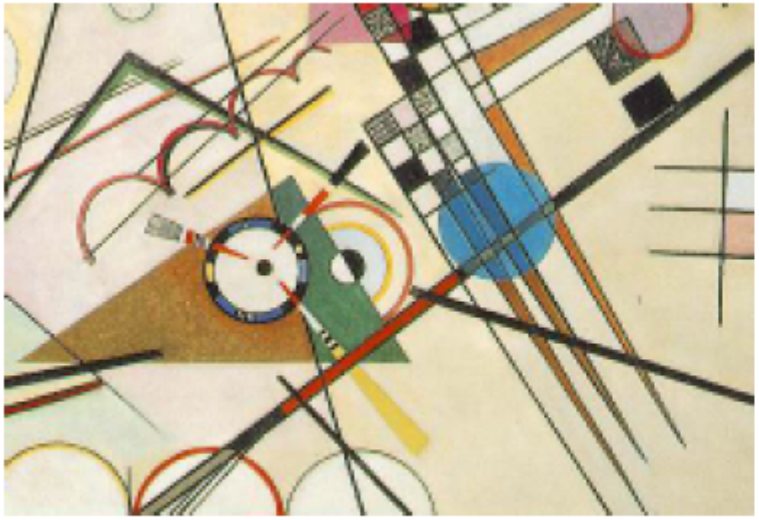
extrait de Kandinsky

$\sigma = 2.0$

Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$



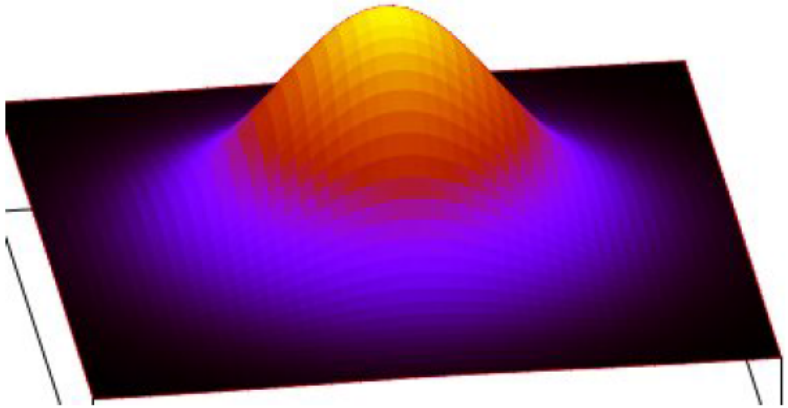


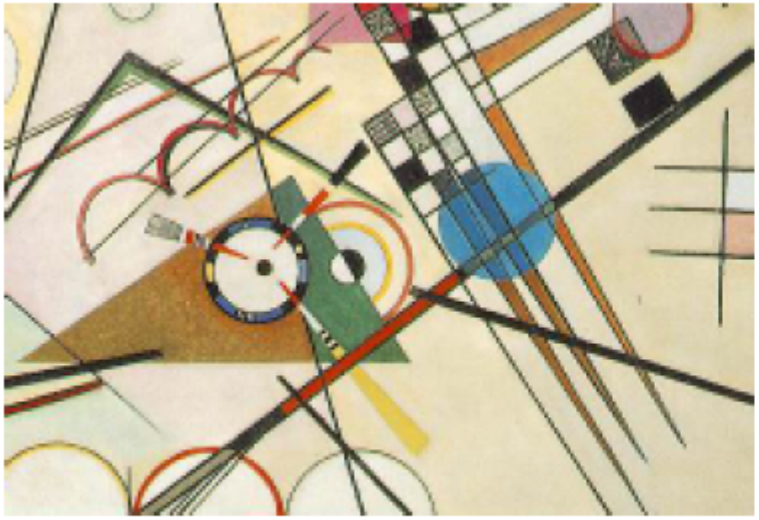
extrait de Kandinsky

$\sigma = 2.5$

Gaussian convolution

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$





extrait de Kandinsky

$\sigma = 3.0$

Gaussian convolution

Defining a Gaussian Kernel

```
1     function f=gaussian_filter_2d(sigma)
2 -       x=-ceil(3*sigma):ceil(3*sigma);
3 -       v=exp(-x.^2./(2*sigma^2));
4 -       f=v'*v;
5 -       f=f./sqrt(sum(f(:).^2));
6 -   end
```

Separability

- Let $h : \{1,...,M\} \times \{1,...,N\} \rightarrow \mathbb{R}$ be a kernel/matrix
  - usual convolution of h with image f requires $M \times N$ multiplications and additions per pixel of f
- Separable kernel
  - if there exist vectors h_1, h_2 such that $h = h_1^T h_2$ then $h = h_1^T h_2 = h_1^T * h_2 = h_2 * h_1^T$
- Separate convolution
  - $f * h = f * (h_1^T * h_2) = (f * h_1^T) * h_2$ [or $(f * h_2) * h_1^T$]
  - perform two 1D-convolutions instead of one 2D-convolution
  - only $M+N$ multiplications and additions per pixel of f

-> The Gaussian Kernel is separable!

Gaussian Kernel: common approximations

- 3×3 Gaussian ($\sigma \approx 3/4$)

$$G_{3\times3} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \begin{bmatrix}1/4\\ 2/4\\ 1/4\end{bmatrix}.\begin{bmatrix}1/4 & 2/4 & 1/4\end{bmatrix} = \begin{bmatrix}1/4\\ 2/4\\ 1/4\end{bmatrix} * \begin{bmatrix}1/4 & 2/4 & 1/4\end{bmatrix}$$

$$= \begin{bmatrix}1/4 & 2/4 & 1/4\end{bmatrix} * \begin{bmatrix}1/4\\ 2/4\\ 1/4\end{bmatrix}$$

- 5×5 Gaussian ($\sigma \approx 1$)

$$G_{5\times5} = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix} = \begin{bmatrix}1/16\\ 4/16\\ 6/16\\ 4/16\\ 1/16\end{bmatrix} * \begin{bmatrix}1/16 & 4/16 & 6/16 & 4/16 & 1/16\end{bmatrix}$$

Convolution

-> Can be used for template matching

(eg. object detection, blob detection)

-> Can be used on more than 1 channel

(eg. Color image, Convolutional Neural Networks)

Convolution vs. correlation

- Continuous convolution (1D): $f * g$

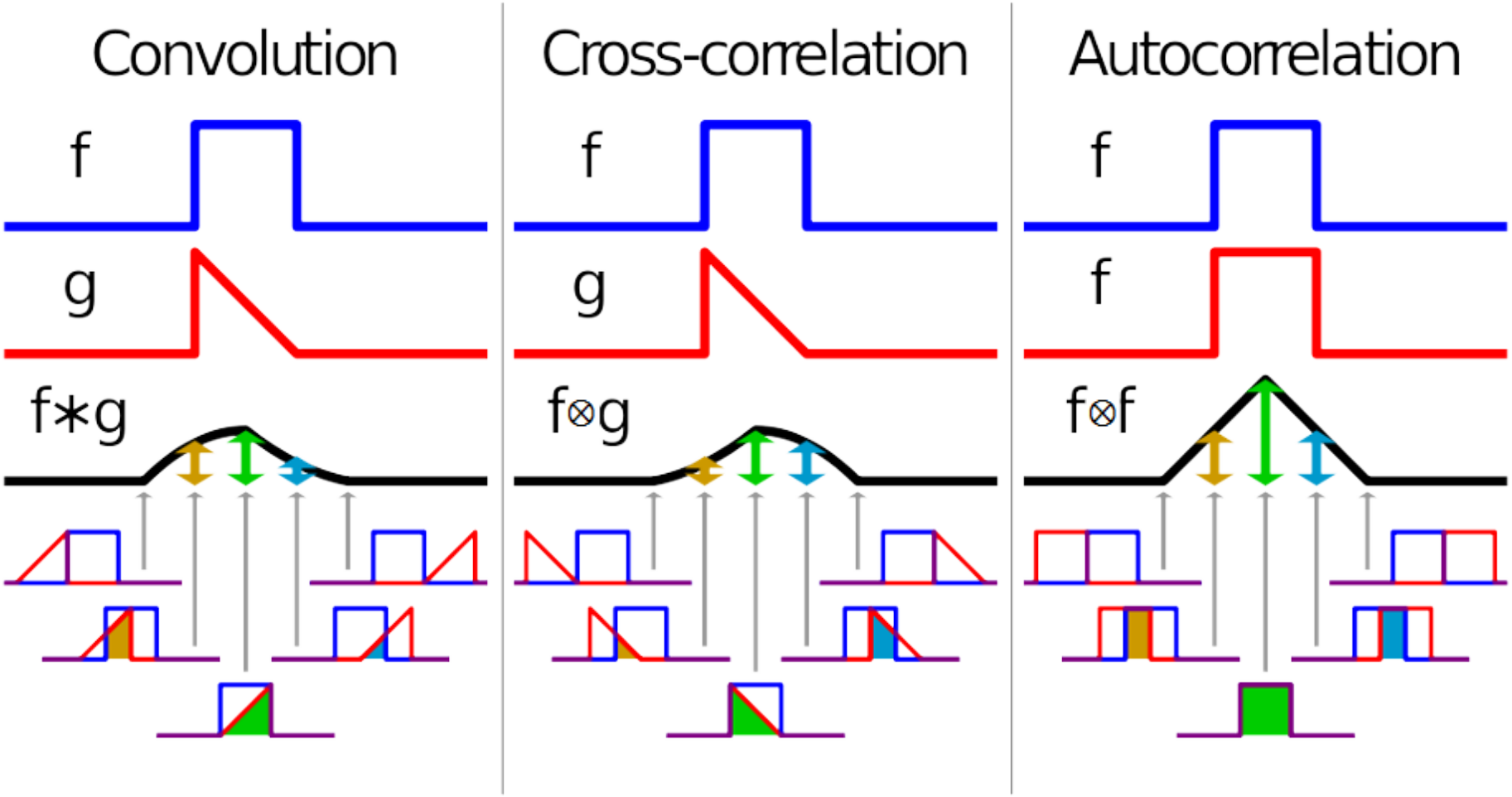
$$(f*g)(x) = \int_{-\infty}^{+\infty} f(u)\,g(x-u)\,du = \int_{-\infty}^{+\infty} f(x-u)\,g(u)\,du = (g*f)(x)$$

- Continuous correlation (1D): $f \otimes g$ [see variant later]

$$(f\otimes g)(x) = \int_{-\infty}^{+\infty} f(u)\,g(x+u)\,du = \int_{-\infty}^{+\infty} f(u-x)\,g(u)\,du \neq (g\otimes f)(x)$$

- If f symmetric, i.e., $f(x) = f(-x)$, then $f \otimes g = f * g$

Convolution vs. correlation



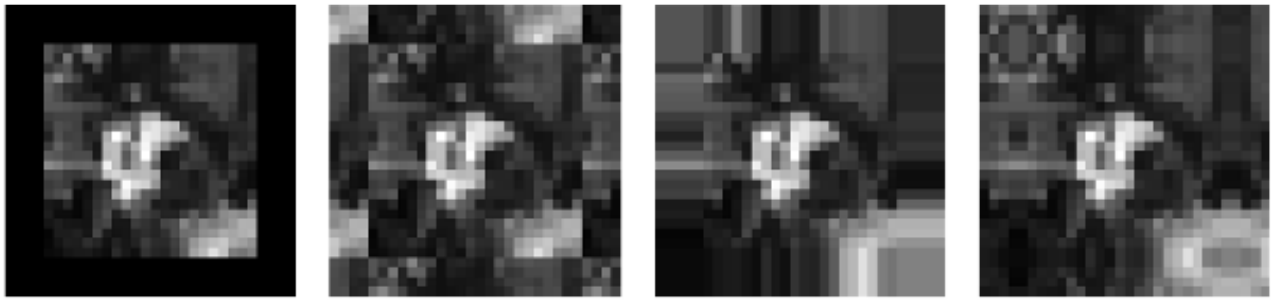
by CMG Lee

Convolution: properties

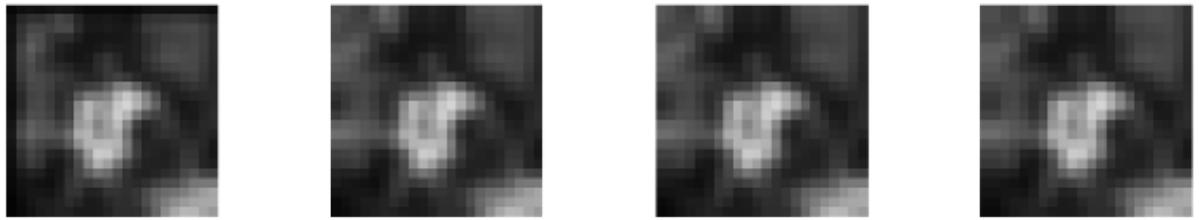
- Linearity: distributivity + associativity with scalar mult.
  - $f * (h_1 + h_2) = f * h_1 + f * h_2$
  - $(f_1 + f_2) * h = f_1 * h + f_2 * h$ (also true of $\otimes$)
  - $c.(f * h) = (c.f) * h = f * (c.h)$
- Shift-invariance
  - translation: $(T_u f)(x) = f(x-u)$
  - $(T_u f) * h = T_u(f * h)$ (also true of $\otimes$)
    - $g(i,j) = f(i+k, j+l) \Leftrightarrow (g * h)(i,j) = (f * h)(i+k, j+l)$
- Associativity
  - $(f * g) * h = f * (g * h)$ (also true of $\otimes$)
- Commutativity
  - $f * h = h * f$ (***not*** true of $\otimes$)

Image boundary effects

- **Padding strategies** (aka wrapping mode, texture addressing mode)
  - pad with 0 (or constant), wrap (loop around), clamp (replicate edge pixel), mirror (reflect pixels across edge)



Blurring examples :



Szeliski 2010 © Springer

☛ or discard results close to boundary

Derivatives with convolution

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

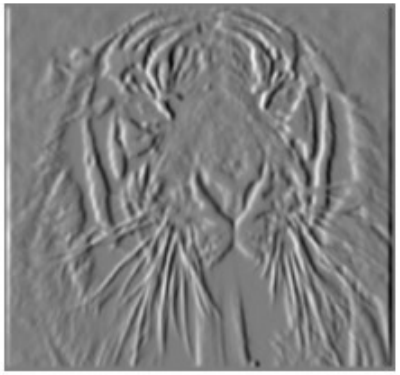
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

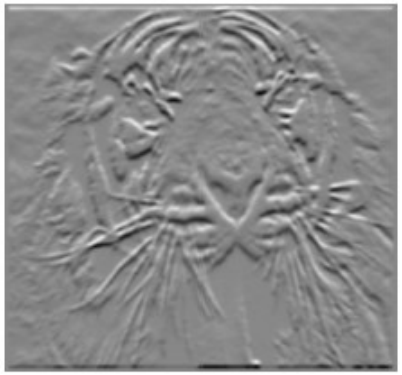
To implement the above as convolution, what would be the associated filter?

Source: K. Grauman

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$





$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

-1
1

or

1
-1

Which shows changes with respect to x?

Source: L. Lazebnik

Finite difference filters

- Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

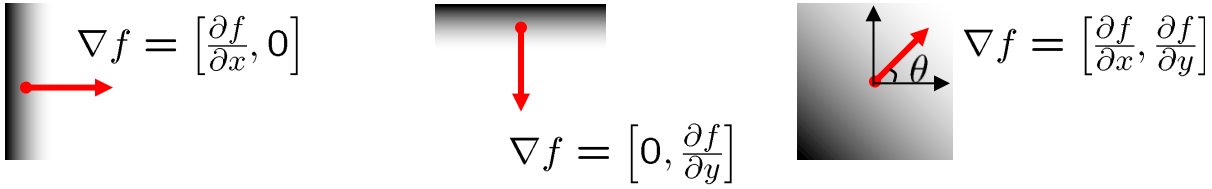
Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



- The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Image gradient

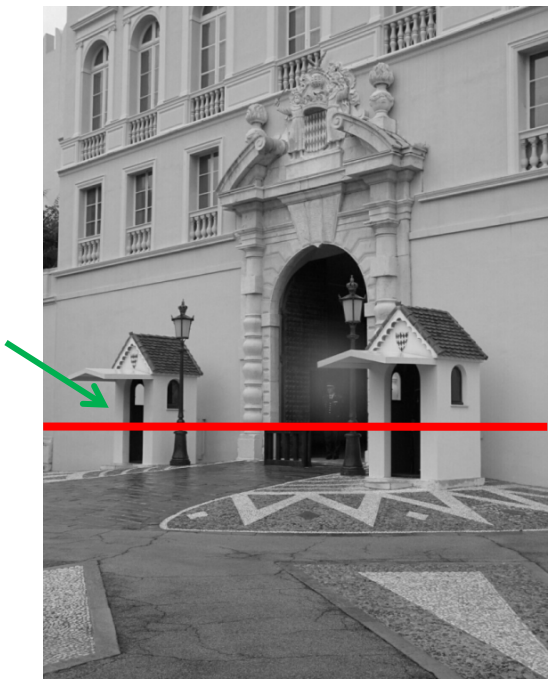
- Directional derivative along $n = (n_x, n_y)$:

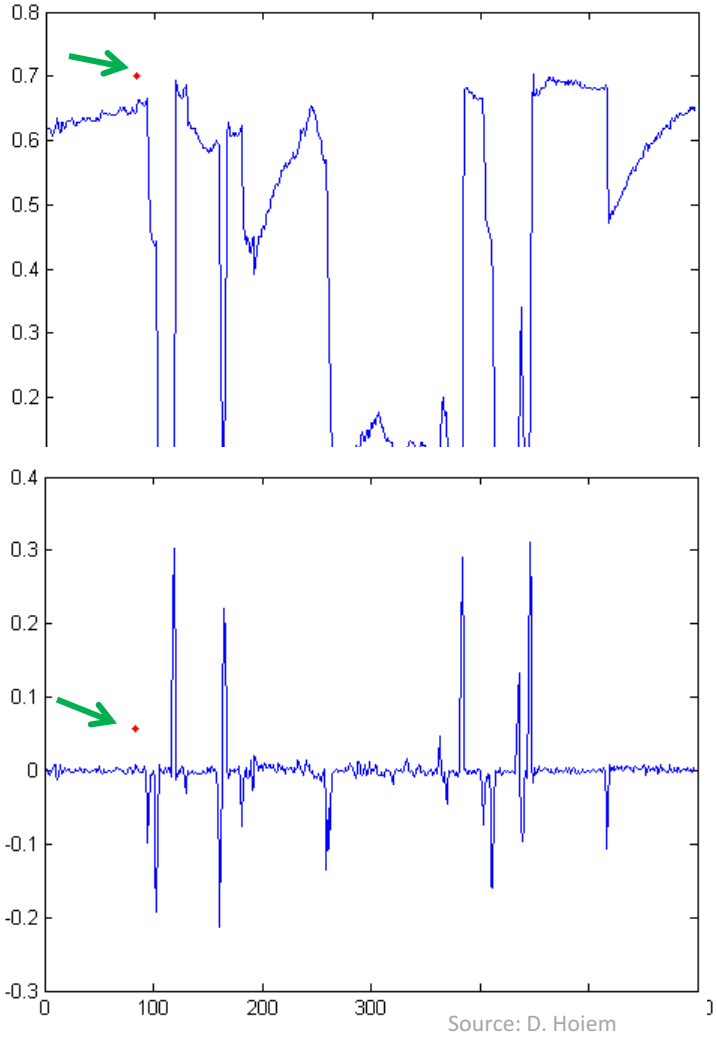
$$\frac{\partial u}{\partial n}(x, y) = \lim_{h \to 0} \frac{u(x + hn_x, y + hn_y) - u(x, y)}{h}$$

- Relation between directional derivative and gradient:

$$\frac{\partial u}{\partial n}(x, y) = \frac{\partial u}{\partial x}(x, y)n_x + \frac{\partial u}{\partial y}(x, y)n_y = \nabla u(x, y) \cdot n$$

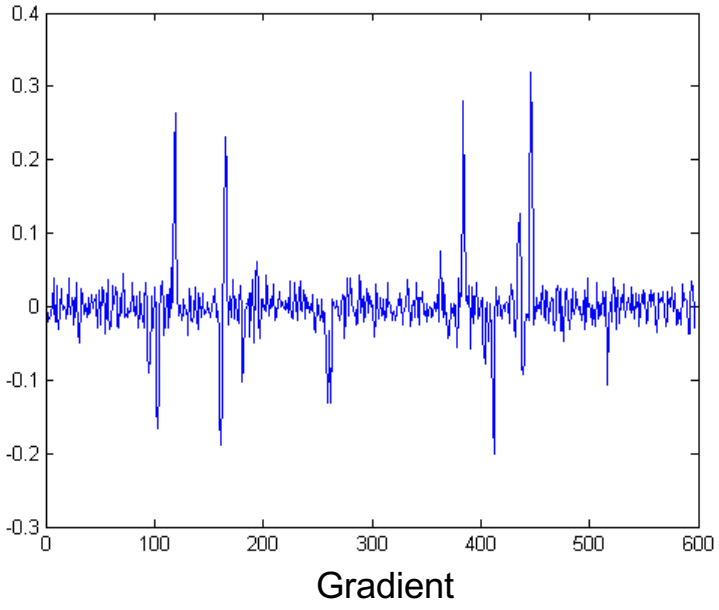
Intensity profile





Source: D. Hoiem

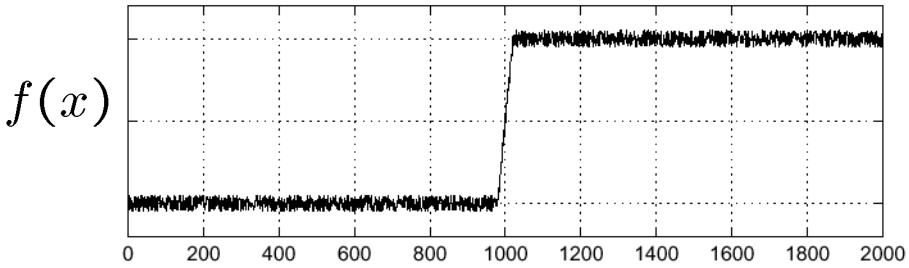
With a little Gaussian noise

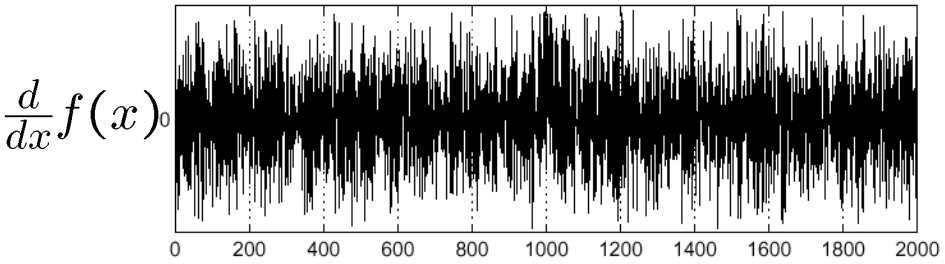


Source: D. Hoiem

Effects of noise

- Consider a single row or column of the image

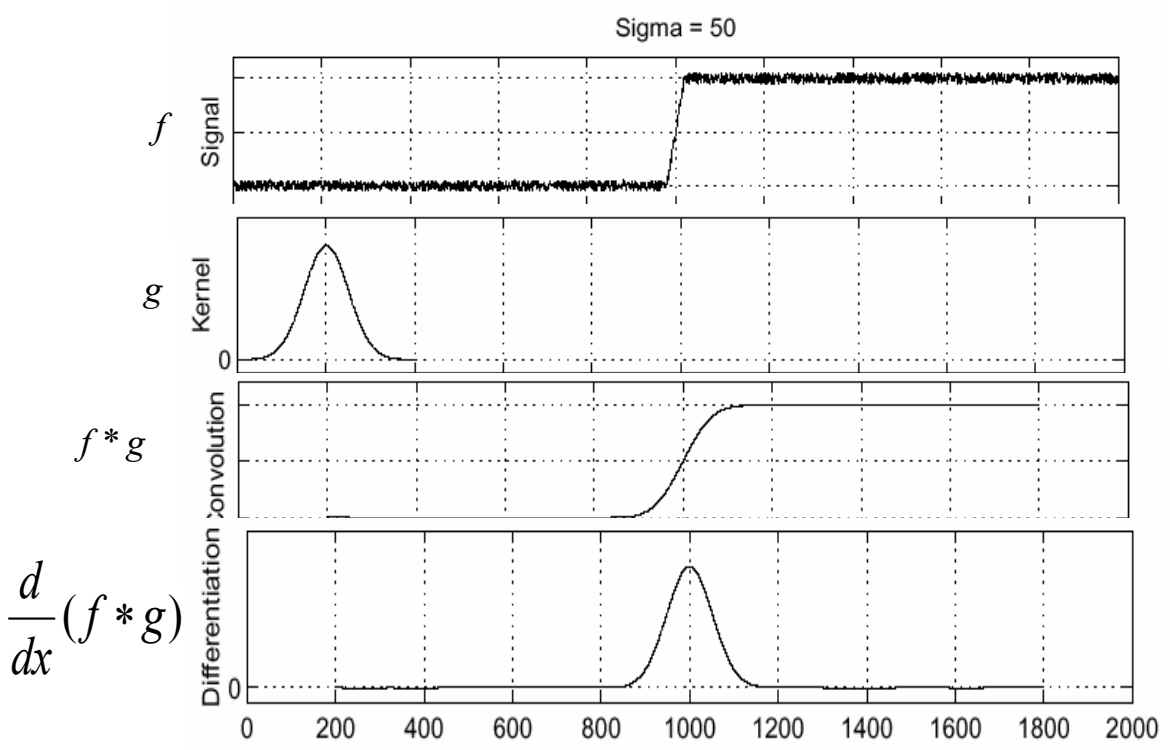




Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

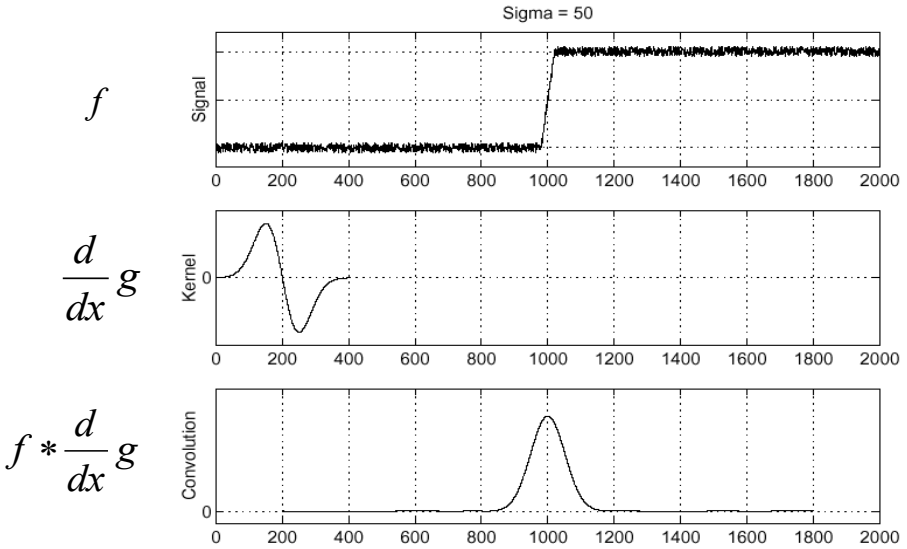
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

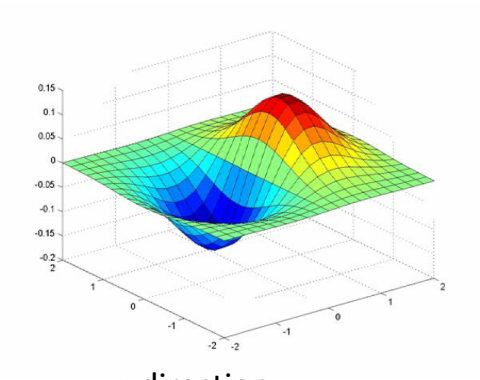
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

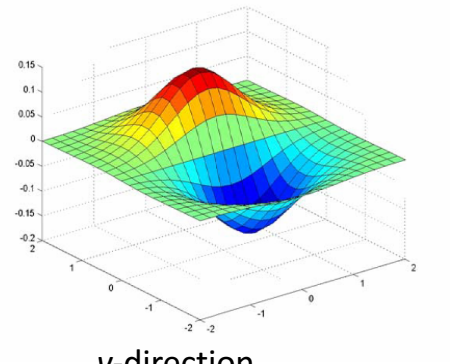


Source: S. Seitz

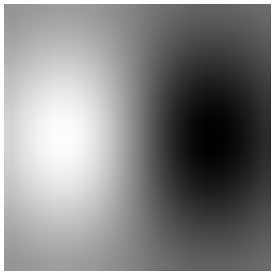
Derivative of Gaussian filters

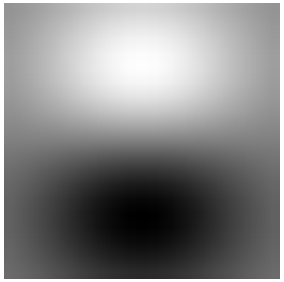


x-direction



y-direction

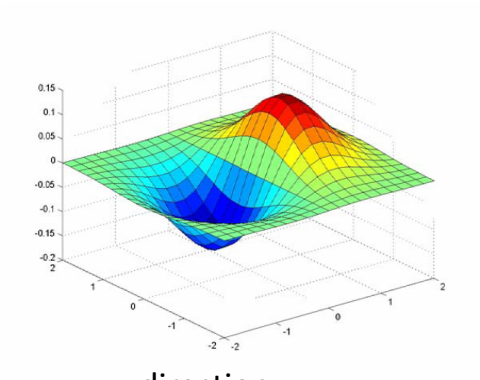




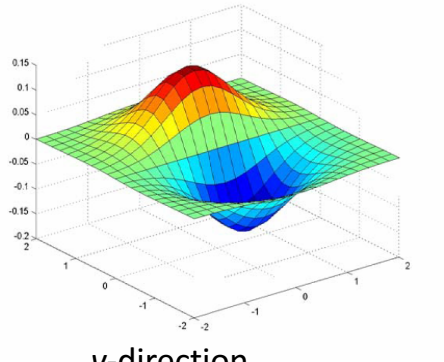
- Which one finds horizontal/vertical edges?

Source: L. Lazebnik

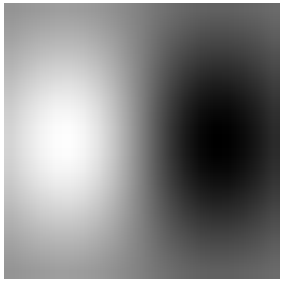
Derivative of Gaussian filters

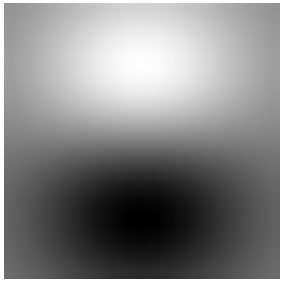


x-direction



y-direction





- Are these filters separable?

Source: L. Lazebnik

Outline

Preliminaries: convolutions, correlations, derivatives

1. Feature detection:
   Harris (Corner)
   Laplacian, Hessian (Blob)

2. Feature description and comparison:
   SSD, ZNCC, HOG, BRIEF, CNN

3. SIFT

Which features?

- Regions: eg. MSER

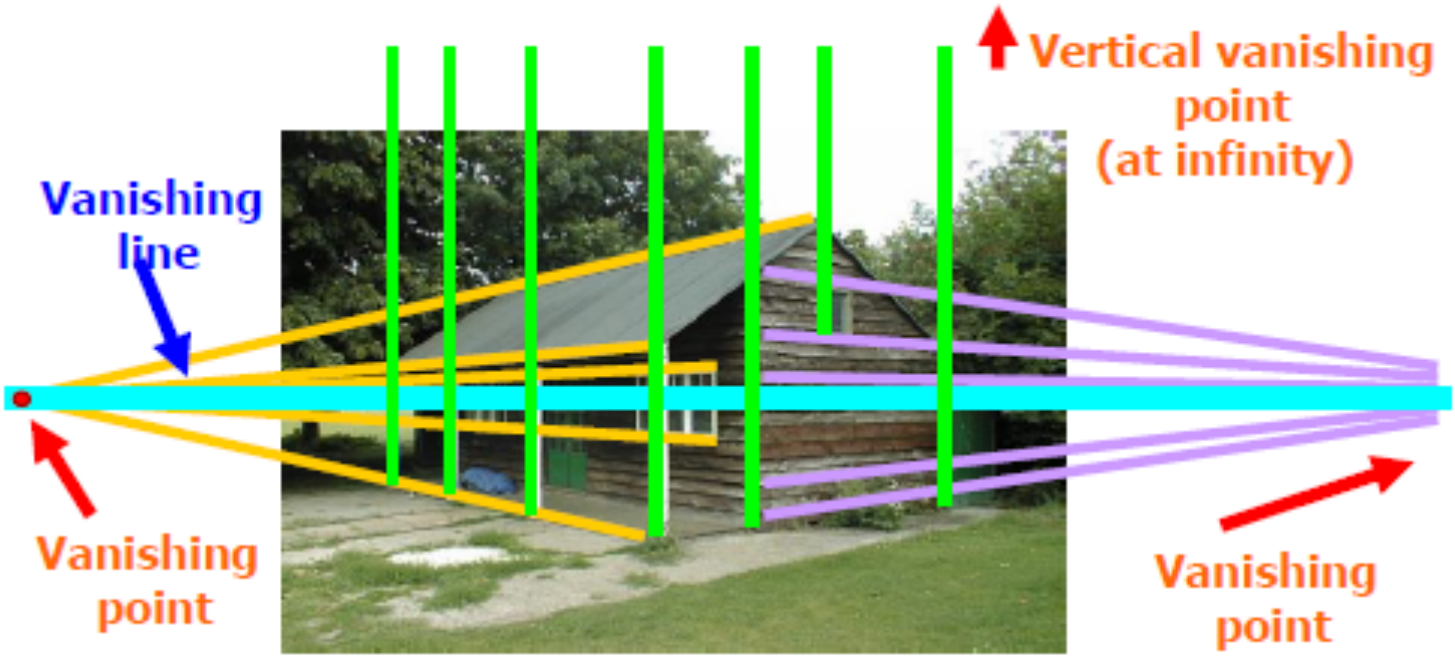
J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing, 22(10):761{ 767, 2004.

Which features?

- Simple region, blobs: eg. Harris-affine

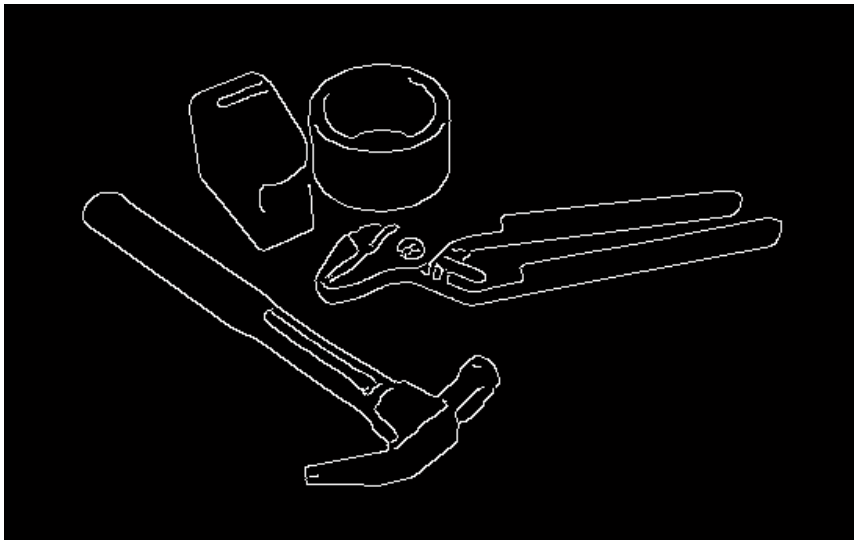
Which features?

- Edges: could be an important clue for 3D geometry



Which features?

- Edges: eg. Canny edges



Which features?

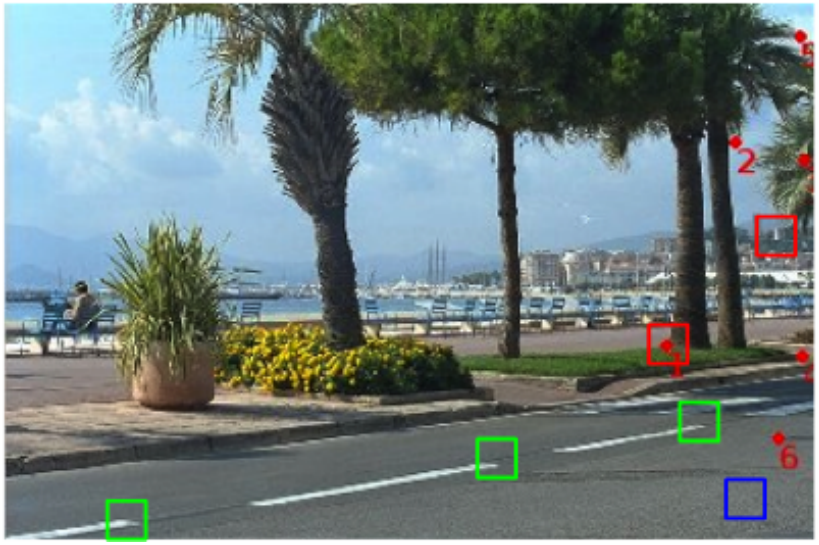
- Points

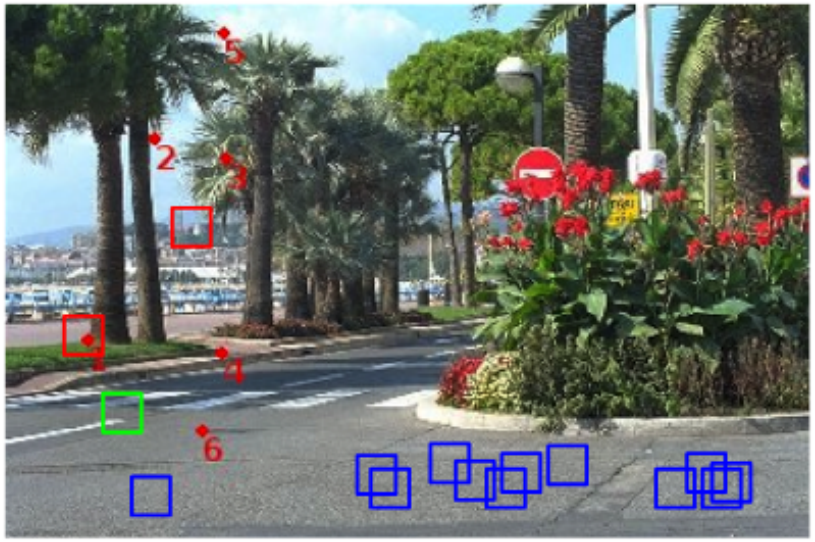
Which features?

- Informative/distinctive/repeatable

Which features?

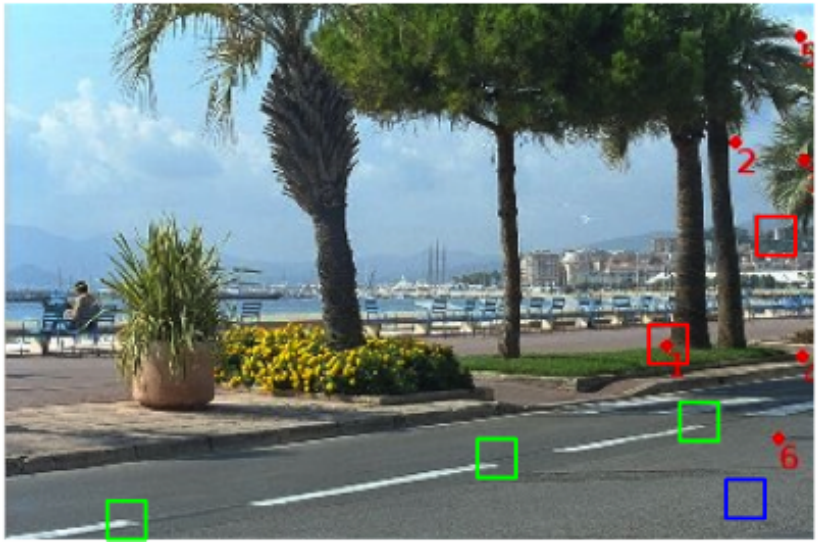
- Informative/distinctive/repeatable

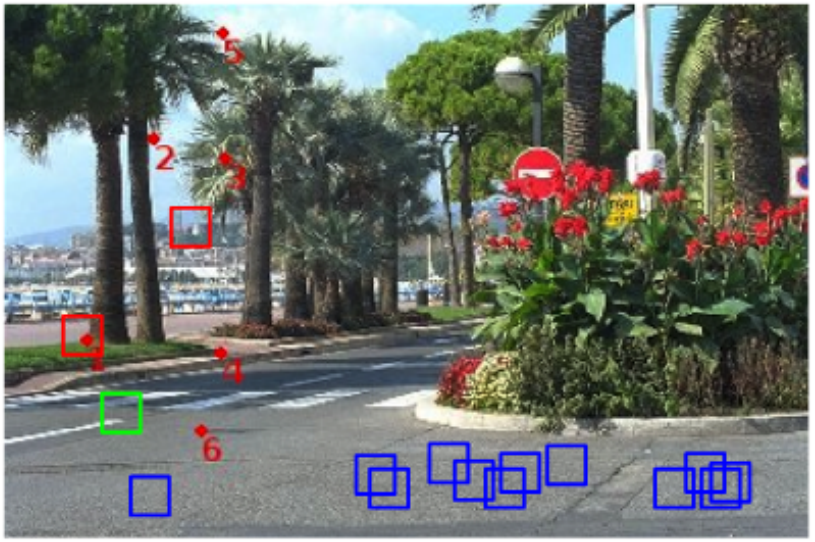




Which features?

- Informative/distinctive/repeatable





Can you think of good candidates?

Harris Corner

Harris Corner

Idea: compare a patch centered in *(0,0)* defined by the weights *w* to a patch centered in *(x,y)* using pixel intensity

$$S(x,y) = \sum_u \sum_v w(u,v)\ (I(u+x, v+y) - I(u,v))^2$$

-> if the difference is large for all *(x,y)* the patch is distinctive.

Initial idea Moravec 1980

Auto-correlation for corner detection (Moravec 1980)

- Corner?

A. Interior Region
Little intensity variation
in any direction

B. Edge
Little intensity variation
along edge, large
variation perpendicular
to edge

C. Edge
Large intensity variation
in all directions

D. Edge
Large intensity variation
in all directions

Harris Corner

Idea: compare a patch centered in *(0,0)* defined by the weights *w* to a patch centered in *(x,y)* using pixel intensity

$$S(x,y) = \sum_u \sum_v w(u,v)\ (I(u+x, v+y) - I(u,v))^2$$

Harris Corner

Idea: compare a patch centered in *(0,0)* defined by the weights *w* to a patch centered in *(x,y)* using pixel intensity

$$S(x,y) = \sum_u \sum_v w(u,v)\ (I(u+x, v+y) - I(u,v))^2$$

Use Taylor extension of *I* (Harris and Stephen 1988):

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

$$S(x,y) \approx \sum_u \sum_v w(u,v)\ (I_x(u,v)x + I_y(u,v)y)^2$$

Harris Corner

Idea: compare a patch centered in *(0,0)* defined by the weights *w* to a patch centered in *(x,y)* using pixel intensity

$$S(x,y) = \sum_u \sum_v w(u,v)\ (I(u+x, v+y) - I(u,v))^2$$

Use Taylor extension of *I* (Harris and Stephen 1988):

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Harris Corner

- *S* large in all direction <-> the two eigenvalues of $\sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ are large

Harris Corner

- *S* large in all direction <-> the two eigenvalues of $\sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ are large

- Harris and Stephens suggest to use

$$M_c = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2 = \det(A) - \kappa \ \mathrm{trace}^2(A)$$

with $A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

the *auto-correlation* or *second moment* matrix

w Gaussian -> invariant to in plane rotation

Harris Corner

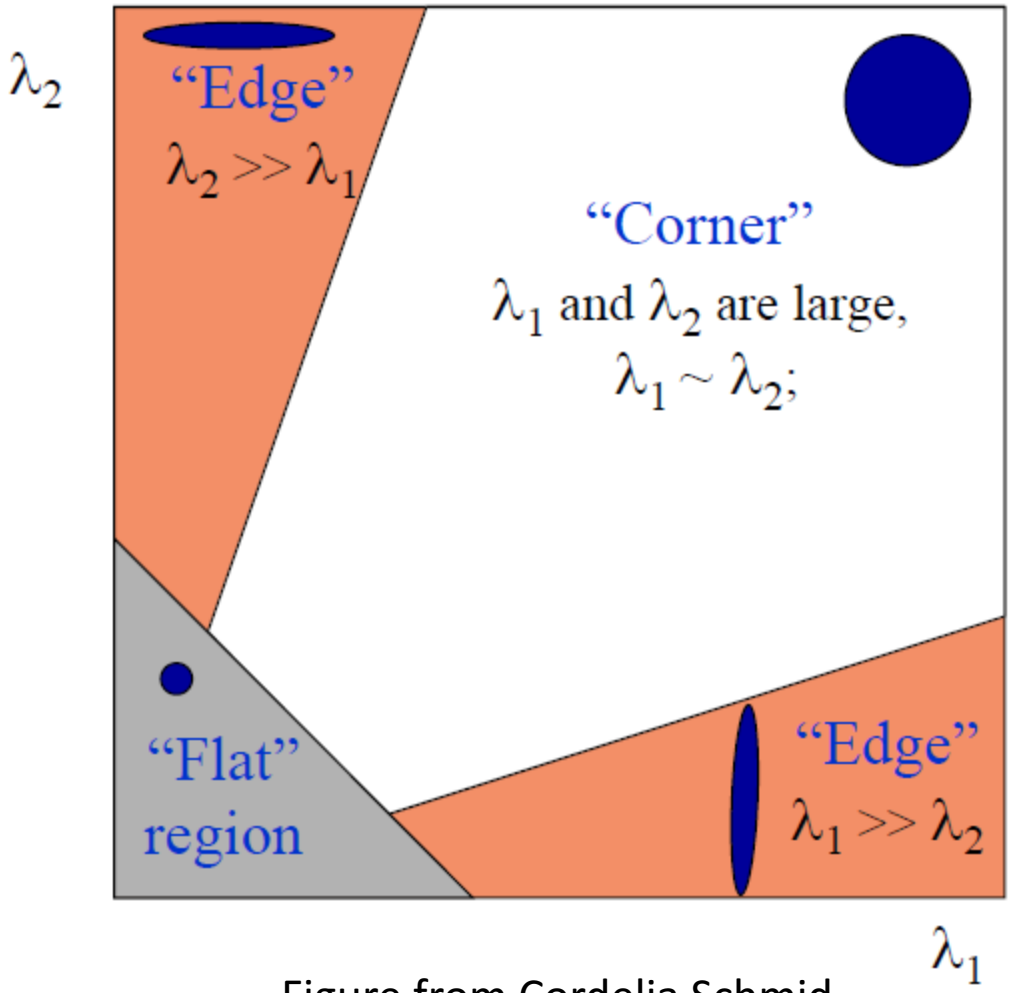


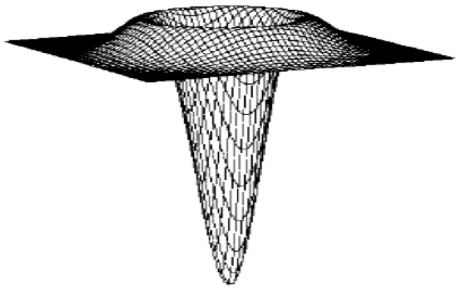
Figure from Cordelia Schmid

Harris Corner: algo

- Compute images derivatives $I_x(\boldsymbol{x}_q)$ and $I_y(\boldsymbol{x}_q)$ for each pixel q of I
  - compute smooth derivation operators: e.g., convolve $d_x = [-½\ 0\ ½]$ with 1D Gaussian G (e.g., $\sigma_d = 1$) → "mask" G_x ; then define $G_y = G_x^T$
  - compute "image derivatives" I_x and I_y: convolve I with masks G_x and G_y
- Compute "product images" I_x^2, I_xI_y, I_y^2 (not matrix products!)
  - then add extra smoothing using an "integration" Gaussian (e.g., $\sigma_i = 2$) (again using two 1D-convolutions rather than one 2D-convolution)
- Consider auto-correlation matrix $\boldsymbol{A} = [I_x^2\ I_xI_y;\ I_xI_y\ \ I_y^2]$
  - compute corner response (or strength) for each q
  - response above threshold and local maximum (8 neighbors) → detection
  - possibly: only keep locally significant responses (see ANMS below)

Blob detection

Blob detection: LoG



- Idea: convolve image with Laplacian of Gaussian and look for extrema
- Laplacian of Gaussian (LoG)
  - $L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$
  - $\nabla^2 L = \nabla^2 (G * I) = (\nabla^2 G) * I$
  - $$\nabla^2 G(x, y; \sigma) = -\frac{1}{\pi \sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$
  - strongest response for compact blobs of extent $\sqrt{2}\sigma$

Scale-normalized LoG

- Need for normalization (Lindeberg 1994)
  - scale-space smoothing $\sigma \Rightarrow$ spatial derivatives $\searrow$, factor σ

    -> or looking at the formula by homogeneity

    -> or writing the desired scale invariance

    -> or thinking of DoG

  - $\nabla^2 G$ second derivatives $\Rightarrow \searrow$ factor σ^2
  - scale-normalized LoG: $\nabla^2_{\text{norm}} G = \sigma^2 \nabla^2 G$

Blob detection: Hessian

Hessian: its eigen-values/vector give principal curvatures of the image

$$H = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix}$$

$Tr(H) = L_{xx} + Lyy = \lambda_0 + \lambda_1$ also large around edges

-> use $Det(H) = \lambda_0 \lambda_1$

Affine invariance/covariance

- Idea: use the eigen-decomposition of the second moment matrix $$M = \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix}$$
  - Give direction of maximum and minimum variation of the image and a characteristic scale

-> Normalize the image $$x' \rightarrow M^{\frac{1}{2}} x$$

Non-Max suppression

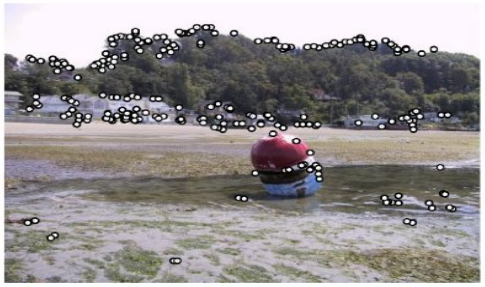
Non-Maximum Suppression

- Problem:

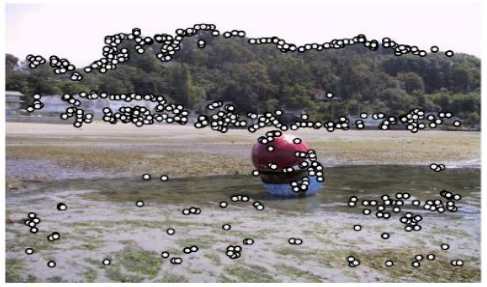
  maximality in 3x3 neighborhood

  → uneven distribution
  (dense where high contrast)

  → poor robustness
  (sensitive to noise)



(a) Strongest 250



(b) Strongest 500

- Solution 1 (NMS):
  - check in larger region around p (e.g., disk of given radius r):

    check maximality w.r.t. all points q such that $|| \boldsymbol{x}_p - \boldsymbol{x}_q || \leq r$

  - check almost largest response (e.g., within 10%):

    $\forall q \;\; || \boldsymbol{x}_p - \boldsymbol{x}_q || \leq r \;\Rightarrow\; c\, f(\boldsymbol{x}_q) \leq f(\boldsymbol{x}_p)$ [e.g., $c = 0.9$]

Adaptive non-maximal suppression (ANMS)

- Problem:

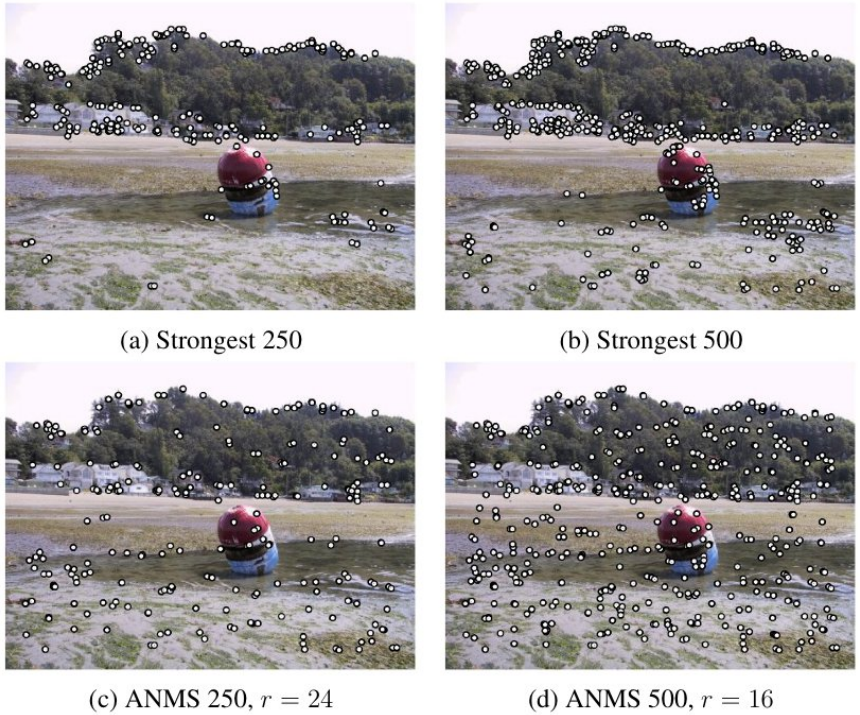
  maximality with NMS

  → distribution still uneven

  → hard-to-tune: radius r

- Solution 2 (ANMS):

  adapt r w.r.t. number n
  of requested detections



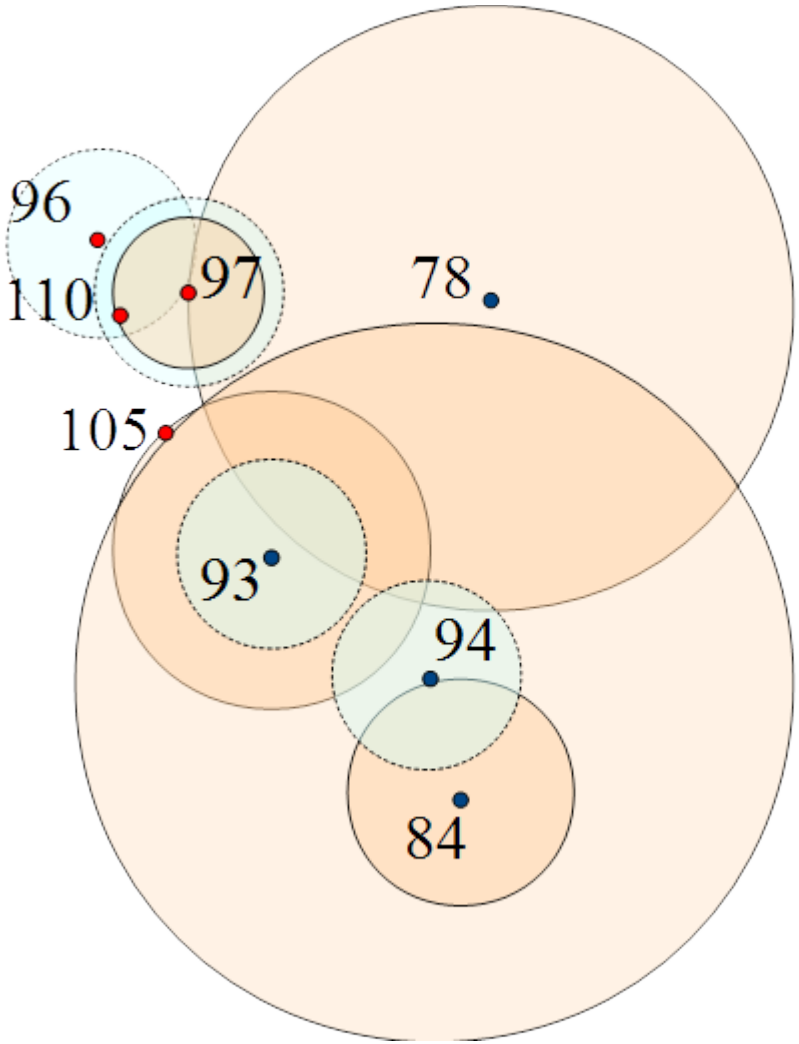
(a) Strongest 250 (b) Strongest 500

(c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

[property: largest in r-disk $\Rightarrow$ largest in r'-disk for $r' < r$]

- start NMS with $r = 0$ and grow r until number of detections $= n$
- or start with $r = +\infty$ and point with strongest response, then decrease r (thus adding detections) until number of detections $= n$

ANMS : example

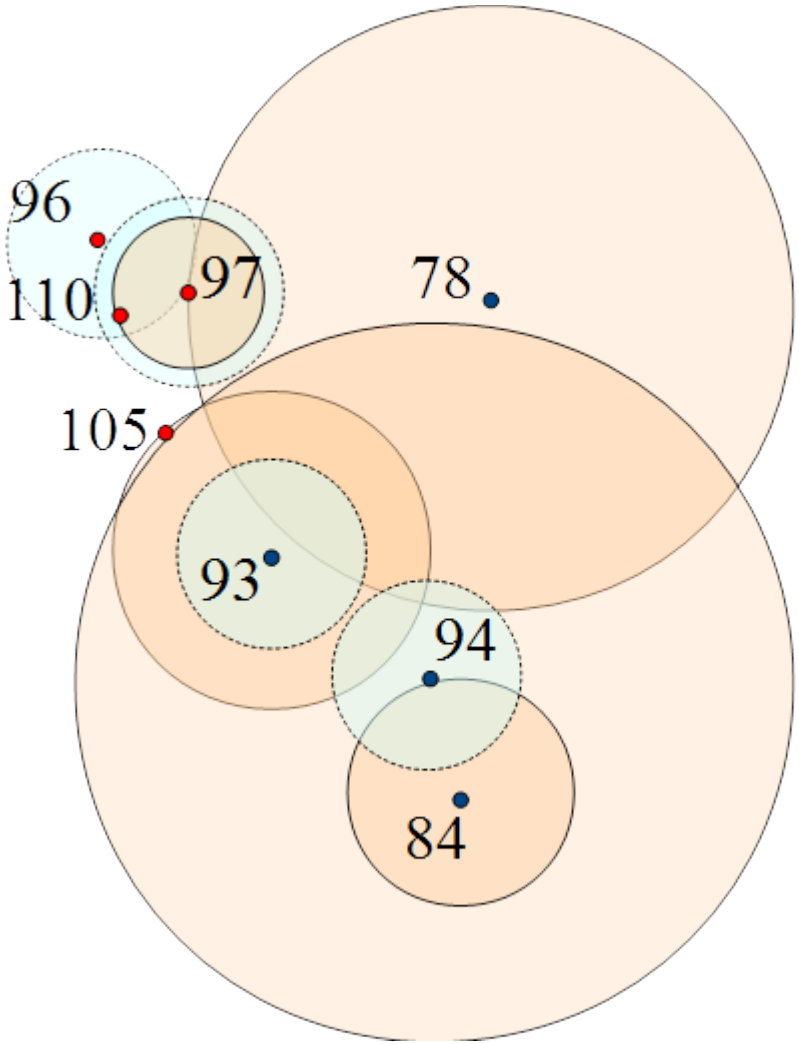


4 strongest points:

4 strongest points with NMS:

4 strongest points with ANMS:

ANMS : example



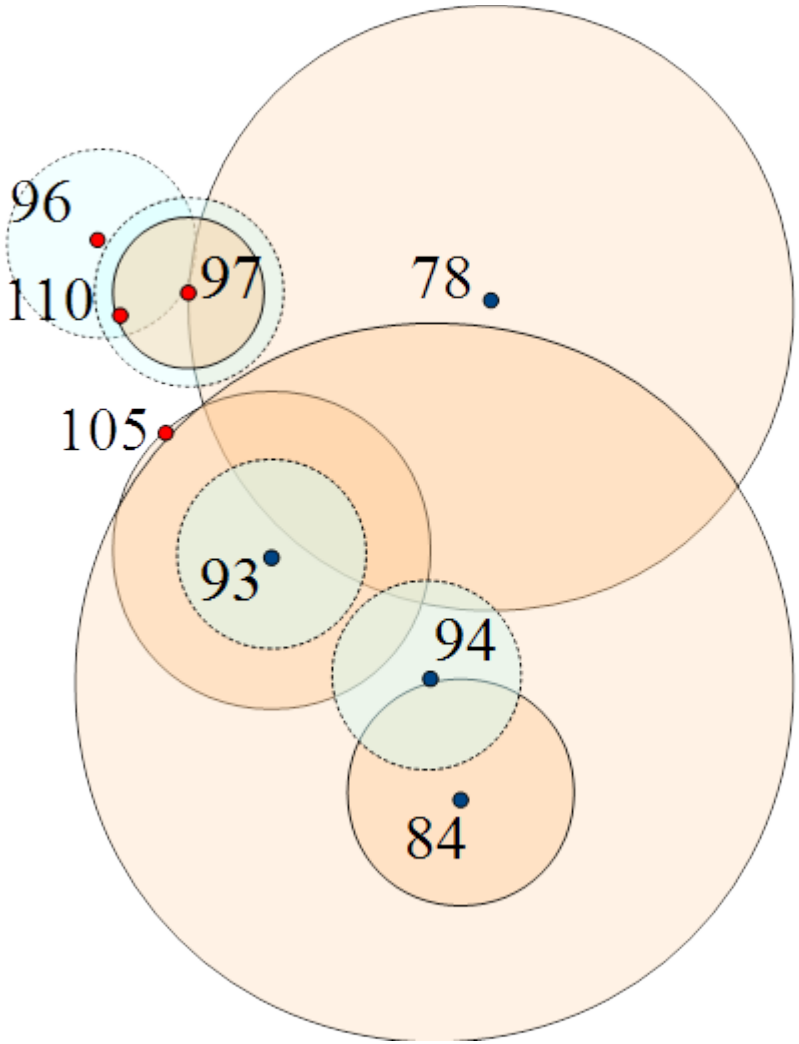
4 strongest points:

110, 105, 97, 96

4 strongest points with NMS:

4 strongest points with ANMS:

ANMS : example



4 strongest points:

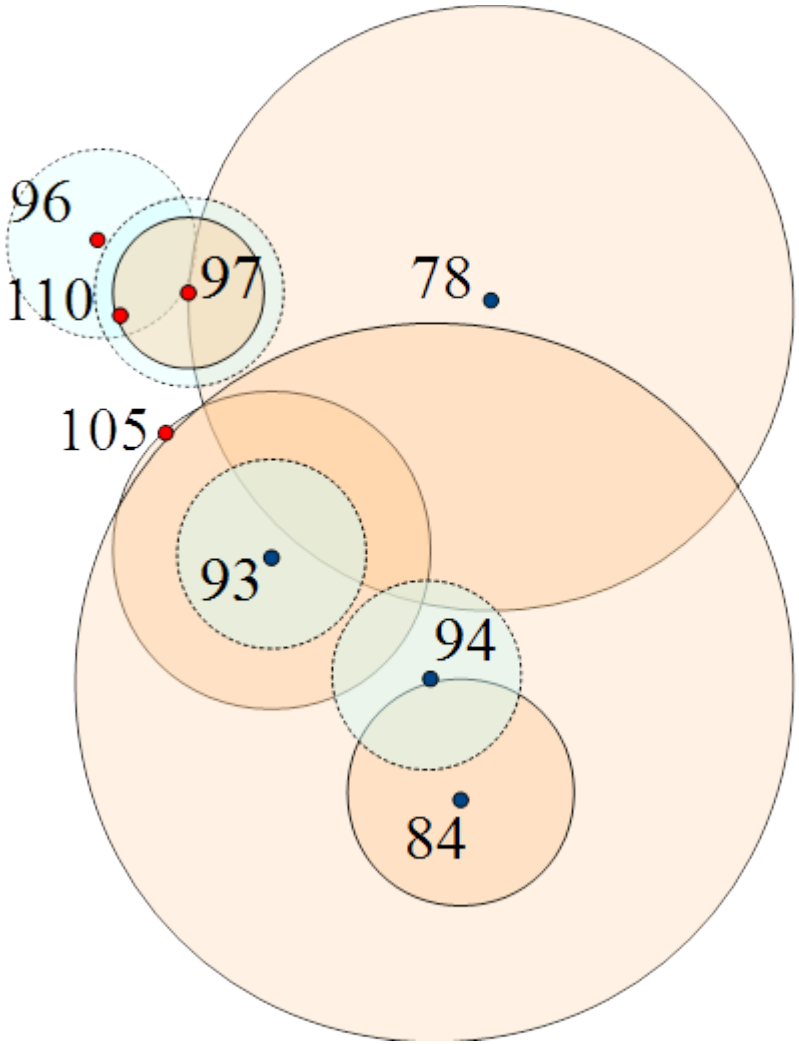
110, 105, 97, 96

4 strongest points with NMS:

110, 105, 94, 93

4 strongest points with ANMS:

ANMS : example



4 strongest points:

110, 105, 97, 96

4 strongest points with NMS:

110, 105, 94, 93

4 strongest points with ANMS:

110, 105, 94, 78

ANMS : algo

Sort $DetectedPoints$ by decreasing response

$p_1 \leftarrow$ point with highest response

$r_{p1} \leftarrow +\infty$

$ProcessedPoints \leftarrow \{p_1\}$, and remove p_1 from $DetectedPoints$

For each detection $p \in DetectedPoints$, in decreasing strength order

$$r_p \leftarrow \min_{q \in ProcessedPoints} \| \boldsymbol{x}_p - \boldsymbol{x}_q \| \text{ such that } f(\boldsymbol{x}_p) < c\, f(\boldsymbol{x}_q)$$

[as $f(\boldsymbol{x}_p) > c\, f(\boldsymbol{x}_q)$ guaranteed for $q \notin ProcessedPoints$]

add p to $ProcessedPoints$

Return n first points p with the highest suppression radius r_p

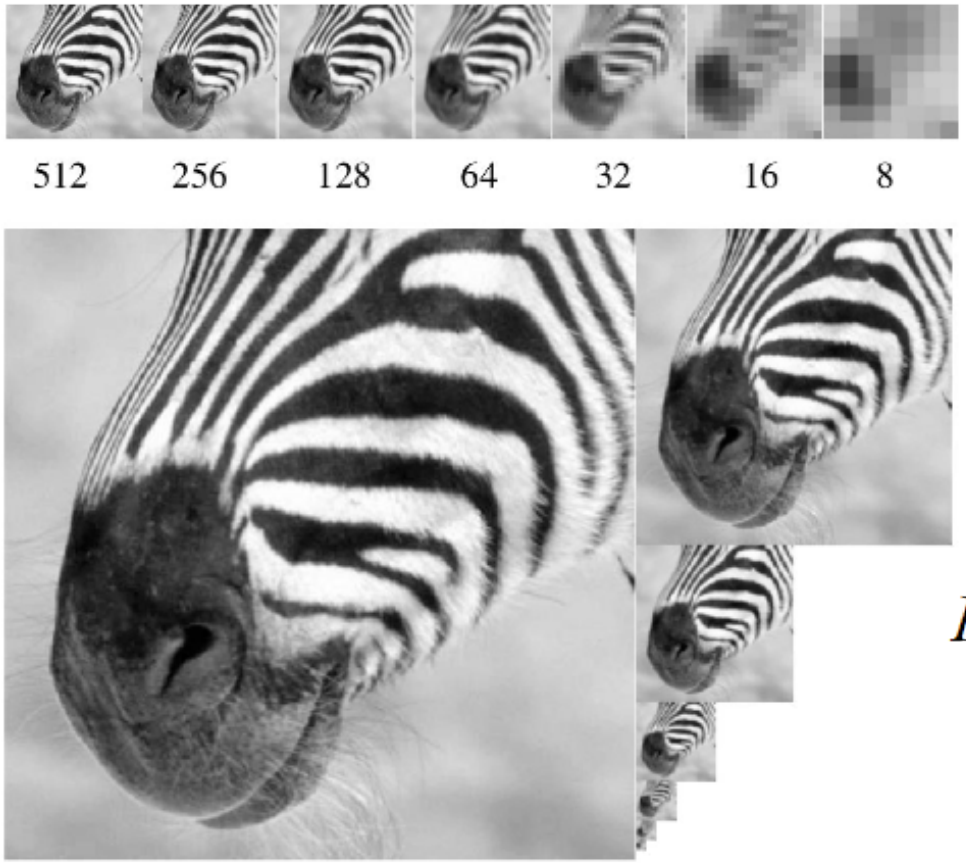
// Still quadratic in number of points. (But there are subquadratic algorithms.)

// Compute, store and compare r^2 rather than r to avoid computing a square root for $r = \| \boldsymbol{x}_p - \boldsymbol{x}_q \|$

Multi-scale

- Problem
  - match objects viewed at different scales/distances
  - detect features in low frequency images (e.g., clouds)
- Solution
  - extract features that are “stable” (correspond to extrema) in **both location and scale**

Multi-scale



- Convolution with Gaussian of varying σ

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Scale-space representation

$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$$

- Scale pyramid

Space: *x,y* dimensions (location)
Scale-space: σ dimension

Multi-scale

σ = 0 (original image)

σ = 1

σ = 4

σ = 16

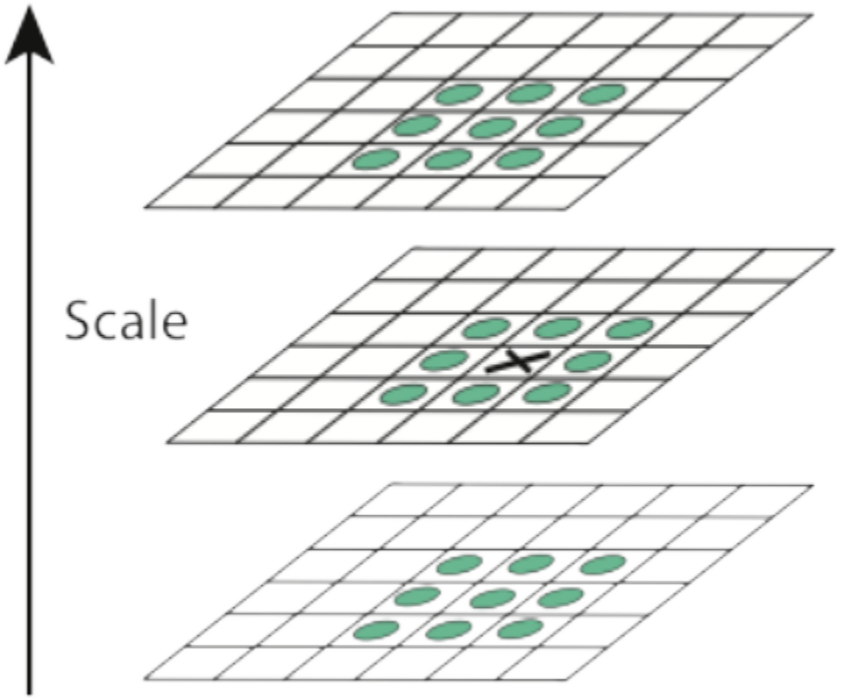
σ = 64

σ = 256

Scale-invariant Harris corner: extremum in scale-space

Idem, replace *I* by *L*

NMS in scale as well



Lowe 2004 © Springer

Evaluation

Evaluation

- The main criteria for a detector is to detect the same features in two different views of the same scene.
- ”Same” can mean different things depending on the feature type (location, scale, orientation…)

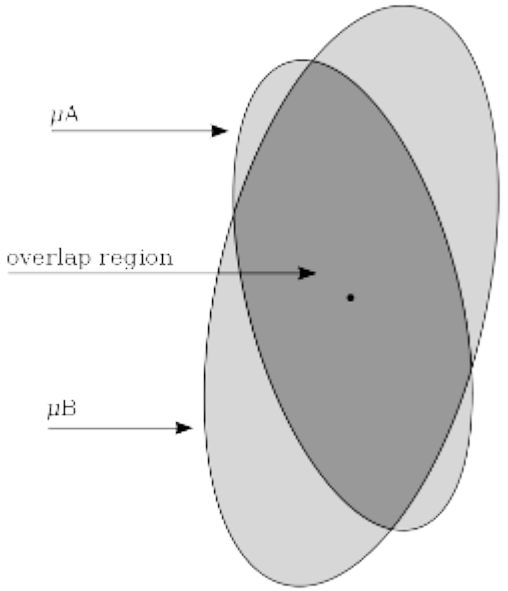
Evaluation

- Setting (Schmid et al. 2000, Mikolajczyk & Schmid 2001, 2002)
  - images of planar scenes
  - known homography and scale transformations
- Location error
  - detected points $\boldsymbol{x}_a$ in I, $\boldsymbol{x}_b$ in I'
  - I and I' related by homography H: $I = H(I')$
  - $\epsilon_{\text{pos}} = \| \boldsymbol{x}_a - H\boldsymbol{x}_b \| < 1.5$ pixels means success (e.g.)
- Scale error (for detectors that compute a feature scale)
  - scale ratio within given factor, e.g. 1.2, means success

Evaluation

- Affinity error
  - $\hat{H}$ local affine approximation of H at point $\boldsymbol{x}_b$
  - μ_A and μ_B elliptical regions defined by $\mu_M = \{x \mid x^T M x \leq 1\}$ corresponding to Harris correlation matrices A and B
  - Jaccard distance

$$\epsilon_{\text{surf}} = 1 - \frac{\mu_A \cap (\hat{H}^T \mu_B \hat{H})}{\mu_A \cup (\hat{H}^T \mu_B \hat{H})}$$



  - $\epsilon_{\text{surf}} < 0.2$ means success (e.g.)

Outline

1. Feature detection:

   Harris (Corner)

   Laplacian, Hessian (Blob)

2. Feature description and comparison:

   SSD, ZNCC, HOG, BRIEF, CNN

3. SIFT

Descriptors

Many type of descriptors

- Pixel values
- Based on local image statistics (local derivatives, answer to filters…)
- Based on local histograms
- Binary comparisons
- CNN-based
- …

Evaluation for sparse matching

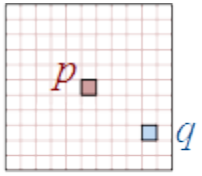
- Can only be evaluated together with a detector.
- The main criteria is the proportion of valid matches between the local features.
- The measure can be mAP, FP rate at a given recall (e.g. 95%), area under ROC…

Patches as descriptors

Idea: normalize/reshape the detected regions and compare the resulting patches

- How to compare patches?
  - Directly compare pixels
  - Look at a more meaningful embedding of images
- Which distance/similarity?

Similarity measures



patch P
centered on p

P : patch of pixels around p in image I
$\boldsymbol{x}_q = (x,y)$: position of pixel $q \in P$ in image I
$\boldsymbol{u} = (u,v)$: displacement of patch P in image I'

- Sum of square difference (SSD) [similar ↘]

$$E_{SSD}(P;\boldsymbol{u}) = \sum_{q \in P} [I'(\boldsymbol{x}_q+\boldsymbol{u}) - I(\boldsymbol{x}_q)]^2$$

- Cross correlation (CC) [similar ↗]

$$E_{CC}(P;\boldsymbol{u}) = \sum_{q \in P} [I'(\boldsymbol{x}_q+\boldsymbol{u})\, I(\boldsymbol{x}_q)]$$

-> meaningful only if normalized!

Similarity measures

- Normalization w.r.t. patch size
  - robustness to patch size variation
    - $E_{SSD}(P\,;\,\boldsymbol{u}) = 1/|P| \sum_q [I'(\boldsymbol{x}_q+\boldsymbol{u}) - I(\boldsymbol{x}_q)]^2$

- Zero-mean normalization: subtract average intensity
  - robustness to constant intensity change: $I \longrightarrow I + c$
    - $\bar{I}_P = 1/|P| \sum_{q\in P} I(\boldsymbol{x}_q)$ and $\bar{I}'_P = 1/|P| \sum_{q\in P} I'(\boldsymbol{x}_q)$
    - $E_{ZSSD}(\boldsymbol{x}_q\,;\,\boldsymbol{u}) = 1/|P| \sum_q [(I'(\boldsymbol{x}_q+\boldsymbol{u}) - \bar{I}'_P) - (I(\boldsymbol{x}_q) - \bar{I}_P)]^2$

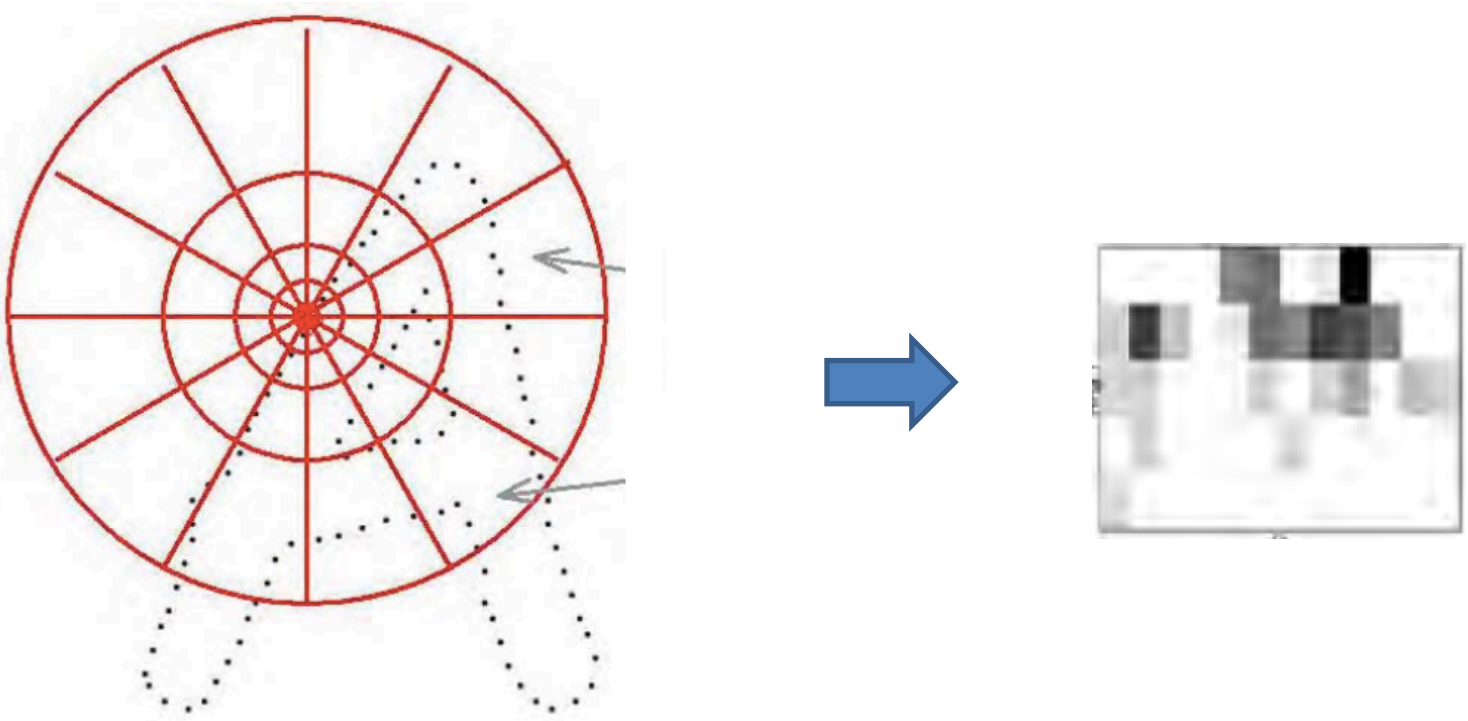
Similarity measures

- Normalization w.r.t. patch intensity variance
  - robustness to affine intensity change: $I \rightarrow aI + b$
    - $\bar{I}_P = 1/|P| \sum_{q \in P} I(\boldsymbol{x}_q)$ and likewise for $\bar{I}'_P$
    - $\sigma = \sigma_I = [\ 1/|P| \sum_{q \in P} (I(\boldsymbol{x}_q) - \bar{I}_P)^2\ ]^{1/2}$ and likewise for $\sigma' = \sigma_{I'}$
    - $E_{ZNSSD}(P\,;\boldsymbol{u}) = 1/|P| \sum_{q \in P} [(I'(\boldsymbol{x}_q + \boldsymbol{u}) - \bar{I}'_P) / \sigma' - (I(\boldsymbol{x}_q) - \bar{I}_P) / \sigma]^2$
      $= E_{ZSSD}(P\,;\boldsymbol{u}) / \sigma\sigma'$
    - sometimes also called E_{NSSD} or E_{SSD}
      - ☛ often confusion between E_{SSD}, E_{ZSSD}, E_{NSSD}, E_{ZNSSD}

-> Problem: limited robustness

Shape context

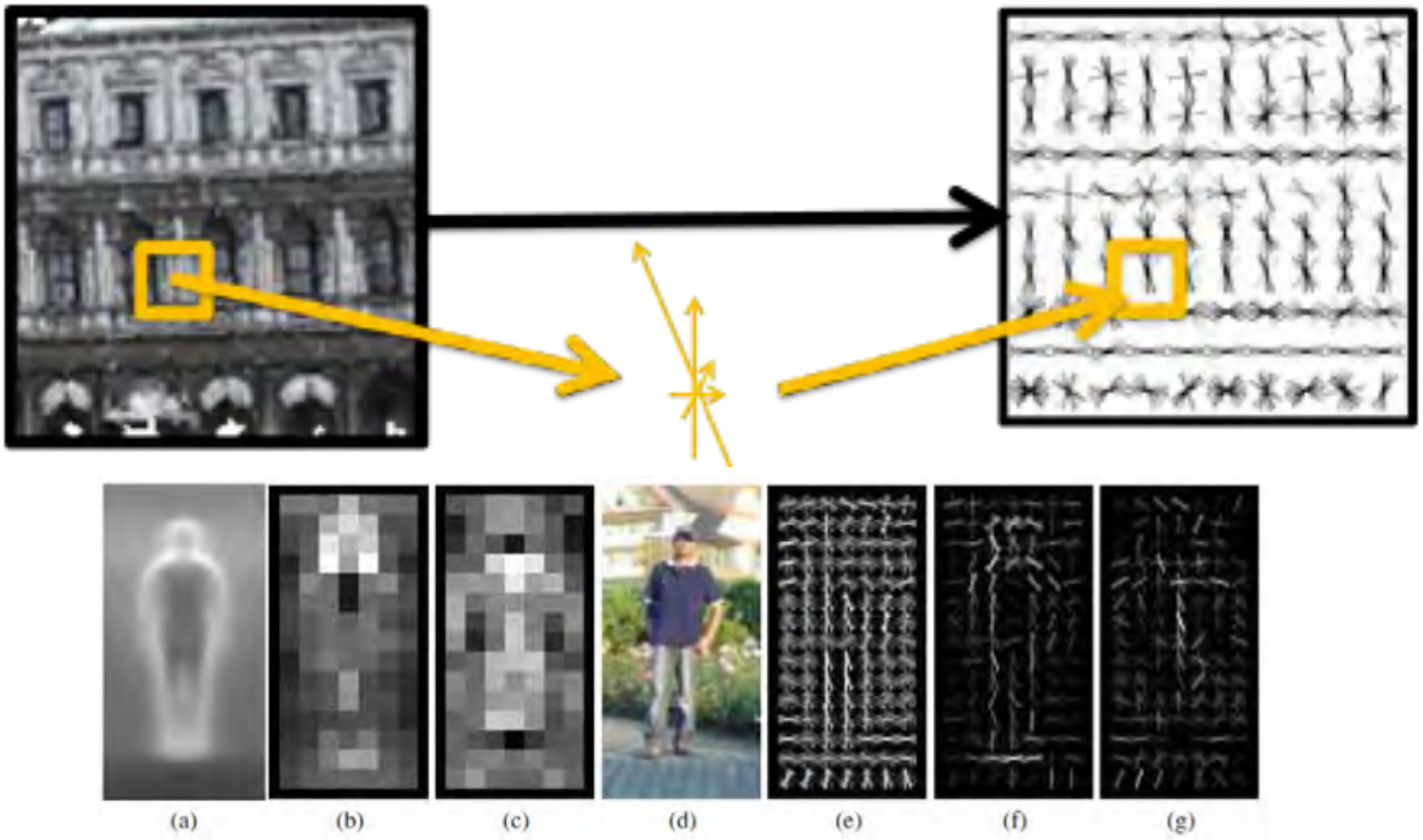
1. Detect edges ; 2. Sample points ; 3. Build histogram



Belongie, S., Malik, J., & Puzicha, J.
Shape matching and object recognition using shape contexts. *PAMI 2002*

Histograms of Oriented Gradients

- Use Histograms



Dalal, N., & Triggs, B. Histograms of oriented gradients for human detection. *CVPR 2005*.

BRIEF

- Using binary comparisons between random locations

$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(\mathbf{p};\mathbf{x}_i,\mathbf{y}_i)$$

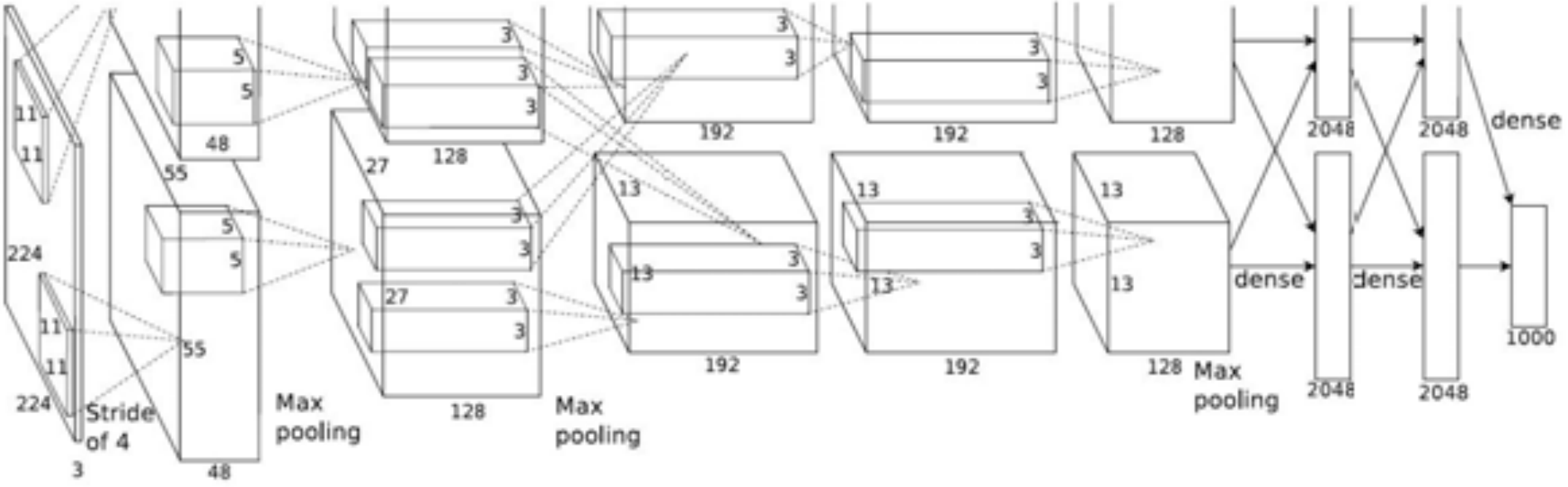
Calonder, M., Lepetit, V., Strecha, C., & Fua, P. Brief: Binary robust independent elementary features. *ECCV 2010*

See also Local Binary patterns (LBP)

CNNs/Deep features

Standard CNNs (eg. AlexNet):

- Succession of convolutions, non linearities (ReLu) and max-poolings
- Trained for image classification (1 million images from ImageNet)



Fischer, P., Dosovitskiy, A., & Brox, T. , Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint 2014*

Conv 4 features seem generic and outperform SIFTs

Descriptor learning

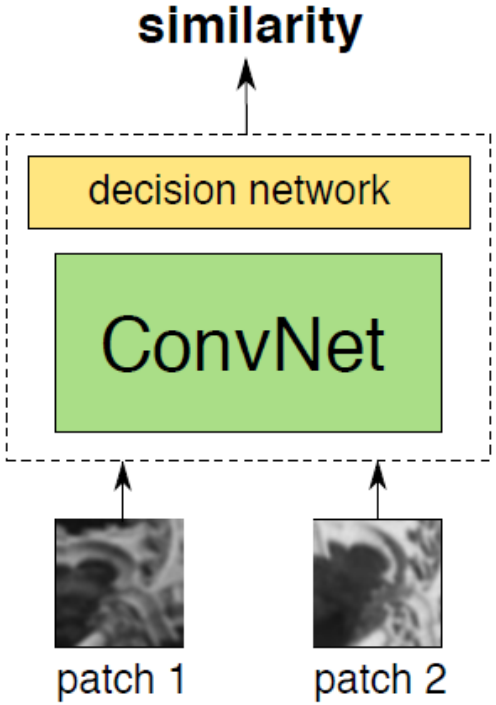
Idea: create a large database of ground truth local feature matches using the 3D of reconstructed scenes and use it to learn the parameter of a descriptor.

M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. PAMI 2011

-> 0.5 million pairs

CNNs/Deep features

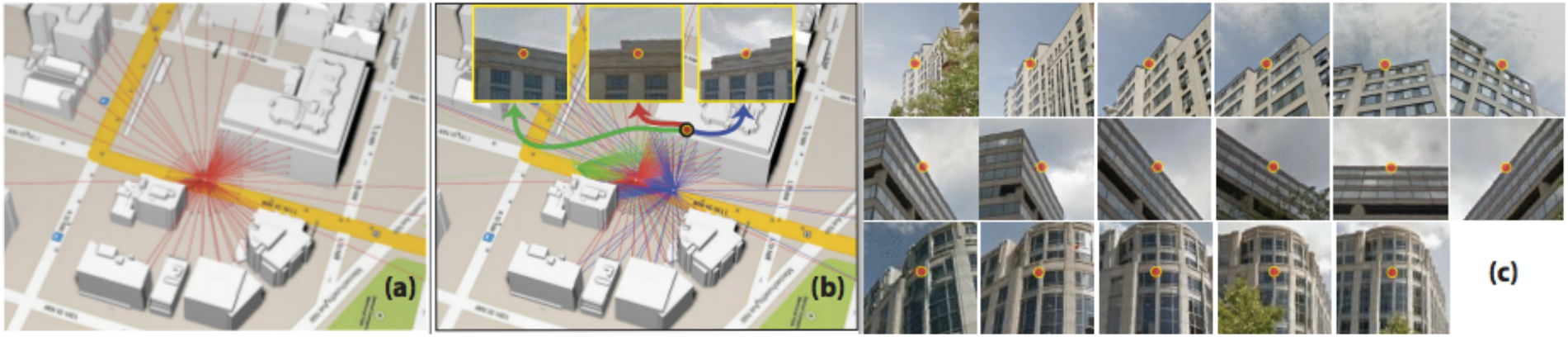
Idea: learning to compare features using a large database of ground truth correspondences



Zagoruyko, S., & Komodakis, N. Learning to Compare Image Patches via Convolutional Neural Networks. *CVPR 2015*

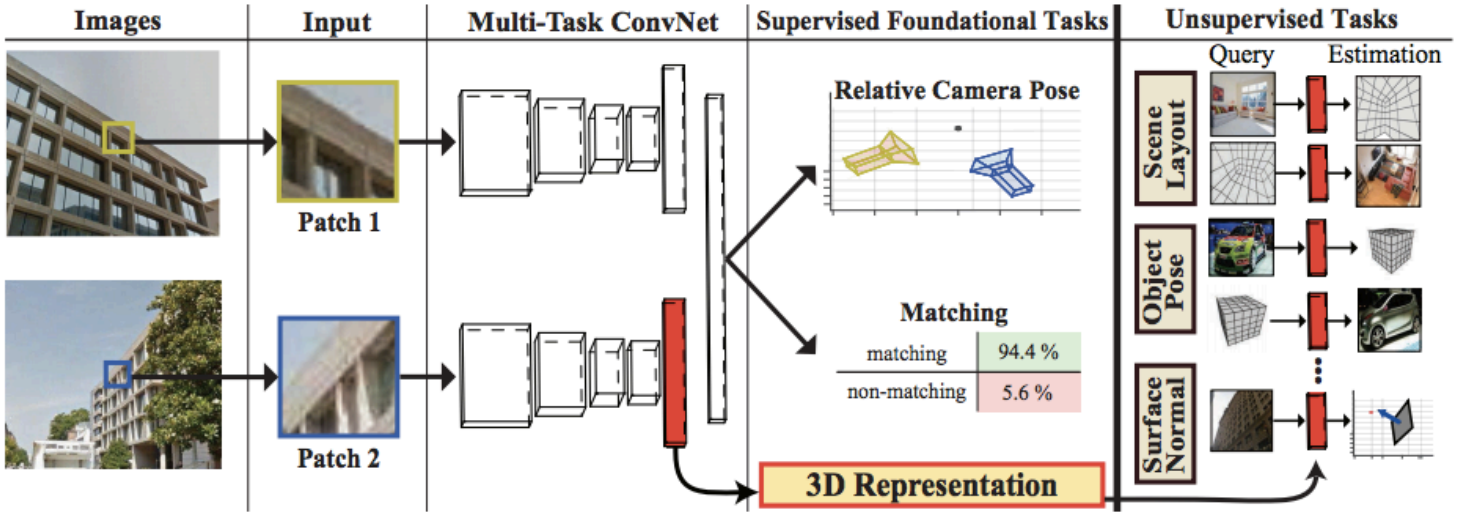
CNNs: going larger scale

- 0.5 billion correspondences from Google Street View



Zamir, A. R., Wekel, T., Agrawal, P., Wei, C., Malik, J., & Savarese, S.
Generic 3D Representation via Pose Estimation and Matching, ECCV 2016

CNNs: going larger scale



EVALUATION ON MIKOLAJCZYK & SCHMID'S FEATURE MATCHING BENCHMARK

Transf. Magnitude	1	2	3	4	5
SIFT [22]	40.1	28.0	24.3	29.0	17.1
Zagor. [29]	43.2	37.5	29.2	28.0	16.8
Fischer et al [30]	42.3	33.9	26.1	22.1	14.6
Ours-rectified	46.4	**41.3**	29.5	23.7	17.9
Ours-unrectified	**51.4**	37.8	**34.2**	**30.8**	**20.8**

Outline

1. Feature detection:

   Harris (Corner)

   Laplacian, Hessian (Blob)

2. Feature description and comparison:

   SSD, ZNCC, HOG, BRIEF, CNN

3. SIFT

SIFT detector and descriptor

- SIFT: Scale-Invariant Feature Transform
- Robustness across
  - a substantial range of affine distortions
  - moderate changes in 3D viewpoint
  - addition of noise
  - change of illumination

Approximating LoG with DoG

- Property of Gaussian:
  - $\partial G / \partial \sigma = \sigma \nabla^2 G$

  $\leftrightarrow$ solution to heat diffusion equation
- Finite difference approximation
  - $\partial G / \partial \sigma \approx (G(x,y;k\sigma) - G(x,y;\sigma)) / (k\sigma - \sigma)$ for $k \approx 1$
  - hence $G(x,y;k\sigma) - G(x,y;\sigma) \approx (k-1)\sigma^2 \nabla^2 G$

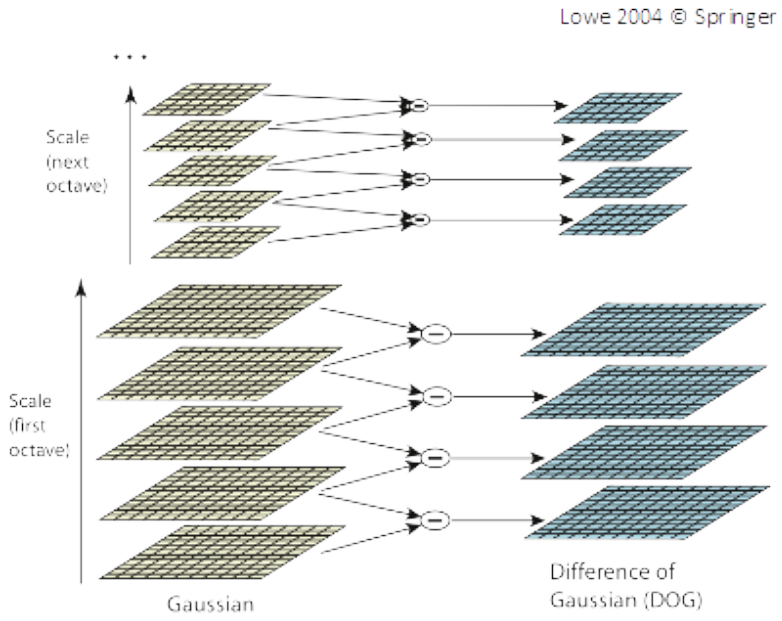
☛ Instead of searching extrema via $\sigma^2 \nabla^2 G$ convolution,
look for extrema via $G(x,y;k\sigma) - G(x,y;\sigma)$
→ simpler, faster

Approximating LoG with DoG

- Difference of Gaussian (DoG)
  - $D(x,y;\sigma) = (G(x,y;k\sigma) - G(x,y;\sigma)) * I(x,y)$
    $= L(x,y;k\sigma) - L(x,y;\sigma)$
- k: constant for all scales
  - approximation OK when $k \to 1$
  - in practice, even OK for $k = \sqrt{2}$
- Basic idea:
  - sample scales for scale-space exploration
  - subtract blurred images at successive scales

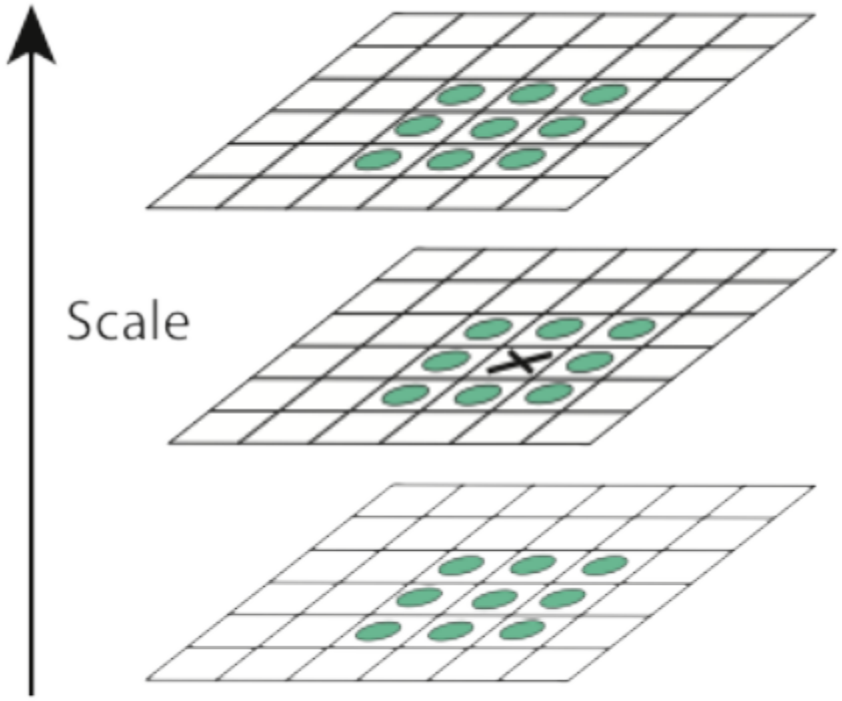
Efficient computation of scale-space: DoG

- Geometric progression of scales with ratio $k = 2^{1/s}$
- Successive convolutions
- At each octave (i.e., every sample $s \rightarrow$ scale factor of 2), resample image
  - every second pixel in each row and column
  - no accuracy loss
  - space & time efficient



Gaussian and DoG pyramids

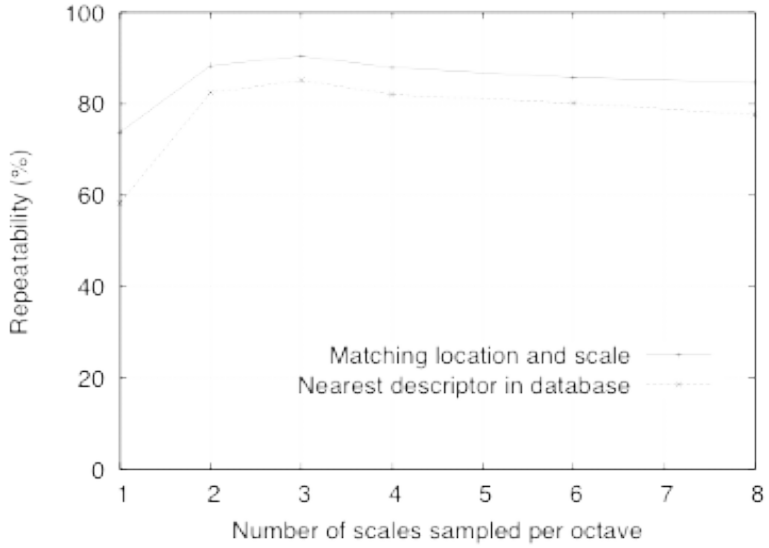
Extremum in scale-space

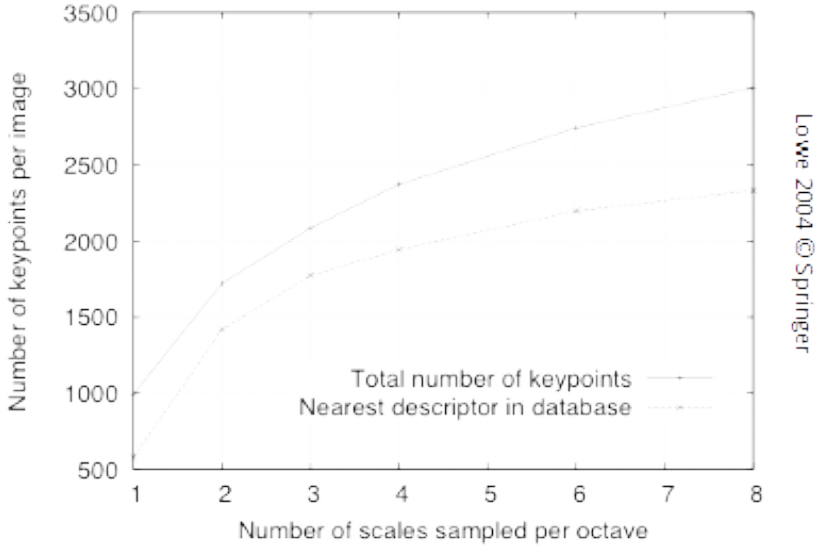


Lowe 2004 © Springer

Scale discretization parameter

- Highest repeatability: $s = 3$ samples per octave





- reason: more samples → more features, but less stable
- depends on use (and test images):
  - OK if s greater for object recognition (quantity vs quality)

Accurate localization

- Subpixel-subscale optimization (Brown and Lowe 2002)
  - fit a 3D quadratic function to local sample points
  - interpolate location of extremum
  - → substantial improvement to matching and stability
- Taylor expansion up to 2nd order at sample point
  - $\boldsymbol{x} = (x, y, \sigma)^T$: offset from sample point
  - D and its derivatives evaluated at sample point

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

Accurate localization

- Extremum $\hat{x}$ such that $\frac{\partial \tilde{D}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}})=0$ i.e., $\hat{\boldsymbol{x}}=-\frac{\partial^2 D}{\partial \boldsymbol{x}^2}^{-1}\frac{\partial D}{\partial \boldsymbol{x}}$
  - Hessian and derivative via finite difference at neighborhood
  - solve 3x3 linear system

- If any dimension of offset $\hat{x} > 0.5$
  - extremum is closer to a different sample point
  - change sample point for interpolation

Rejection of unstable keypoints with low contrast

- DoG value at extremum
  - using above equations

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

- Discard extrema such that $|\tilde{D}(\hat{\boldsymbol{x}})| < 0.03$ (e.g.)
  (assuming pixels range in $[0,1]$)

Eliminating edge responses

- Problem: strong response along edges
  - large principal curvature across edge
  - small one in the perpendicular direction
  - unstable, location poorly determined
- Solution: check curvatures (cf. Harris and Stephens 1988)
  - use trace and determinant rather than explicit eigenvalues

$$\boldsymbol{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$$

$$\text{trace}(\boldsymbol{H}) = D_{xx} + D_{yy} = \lambda_0 + \lambda_1$$

$$\det(\boldsymbol{H}) = D_{xx} D_{yy} - D_{xy}^2 = \lambda_0 \lambda_1$$

Eliminating edge responses

- Assume $\lambda_0 \leq \lambda_1$, let $r = \lambda_1 / \lambda_0$, then

$$\frac{\text{trace}(\boldsymbol{H})^2}{\det(\boldsymbol{H})} = \frac{(\lambda_0 + \lambda_1)^2}{\lambda_0 \lambda_1} = \frac{(r+1)^2}{r}$$

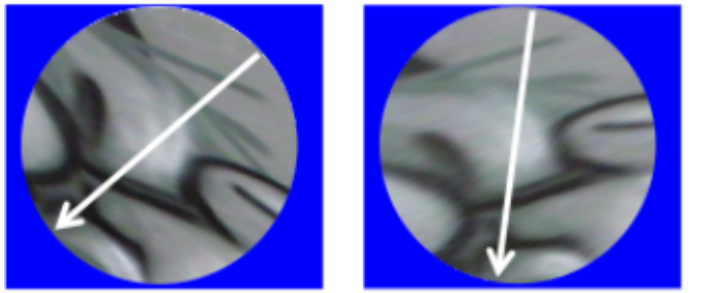
- Ratio $(r+1)^2 / r$
  - is at minimum when eigenvalues are equal
  - increases with r
- Discard any extremum such that $\frac{\text{trace}(\boldsymbol{H})^2}{\det(\boldsymbol{H})} > \frac{(r_{\max}+1)^2}{r_{\max}}$
  - in practice (Lowe 2004): $r_{\max} = 10$

Orientation assignment

- Principle:

  find stable orientation at keypoint

  normalize w.r.t. orientation for rotation invariance



Schmid @ INRIA

Orientation assignment

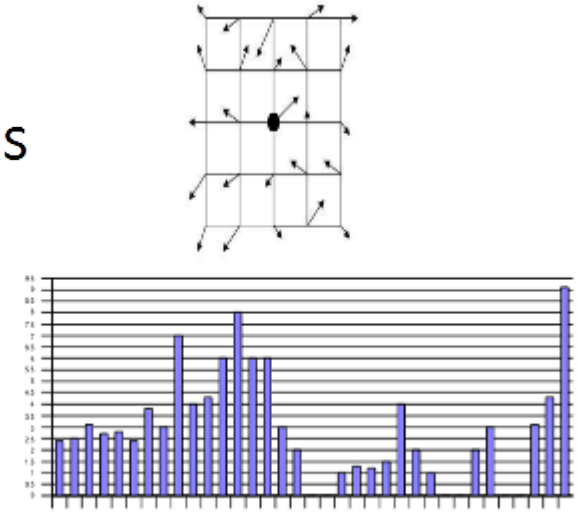
- Orientation = dominant direction of local gradient

$$I_x = \tfrac{1}{2}(L(x+1,y) - L(x-1,y)), \quad I_y = \tfrac{1}{2}(L(x,y+1) - L(x,y-1))$$

$$m(x,y) = \sqrt{I_x^2 + I_y^2}, \quad \theta(x,y) = \arctan(I_y / I_x)$$

- Solution 1: average gradient in region near keypoint
  - can be small, unreliable indicator of orientation
- Solution 2: histogram of orientations near keypoint
  - 36 bins covering 360°
  - weighted by magnitude m
  - weighted by Gaussian window with scale $1.5 \times \sigma_{\text{keypoint}}$

Orientation assignment

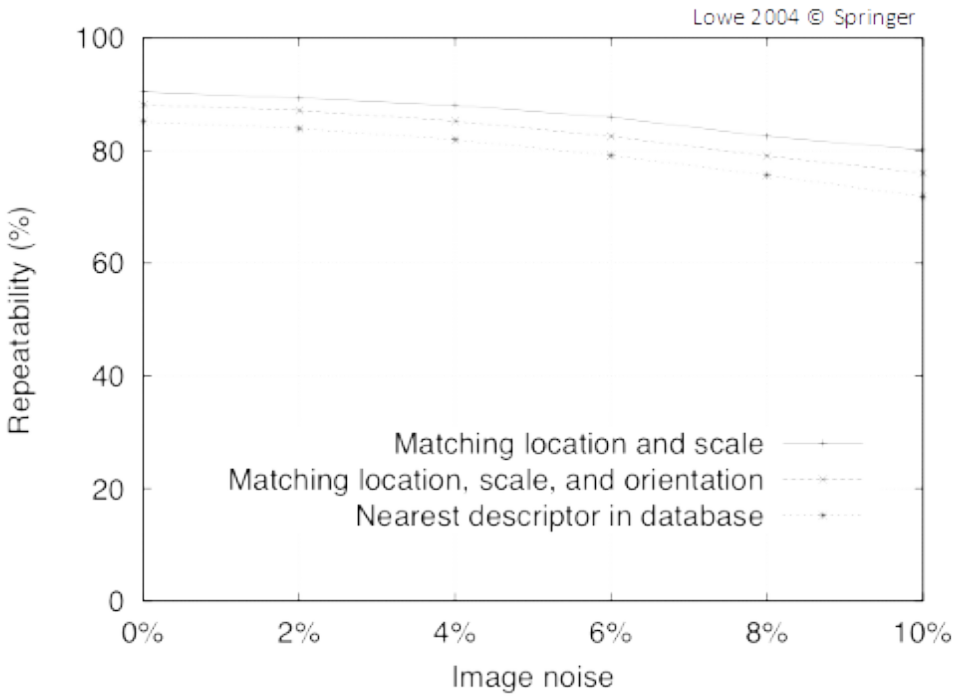


- Dominant directions = histogram peaks
  - keep highest peak
  - also keep all directions within 80% of dominant one (→ +15%)
    - contributes to matching stability
    - in practice, extra directions implemented as additional features

      (Vedaldi: up to 4 directions, Lowe: ?)
- Accurate orientation
  - interpolate peak position, i.e., gradient orientation
    - use 3 histogram values closest to peak, fit parabola

Orientation repeatability

- Repeatability measure :
  - images randomly rotated and scaled + random pixel noise
  - 2nd line of graph: success if orientation difference < 15°
  - standard deviation
    - 2.5° when no noise
    - 3.9° when 10% noise
      - only 5% loss in repeat.
  - major cause of error
    - imprecision in feature location and scale



SIFT descriptor: use local gradients

- Inspired by biological vision (Edelman et al. 1997)

  complex neurons in primary visual cortex respond to a gradient at a particular orientation and spatial frequency

☛ Sample gradient orientation around keypoint

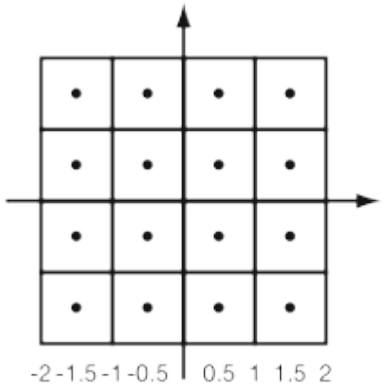
  + some normalization

  + rotate relative to keypoint orientation

  at keypoint scale

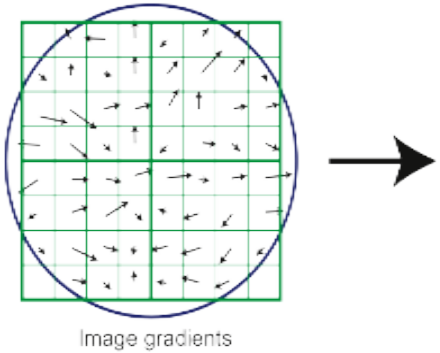
SIFT descriptor

- Descriptor sampling region
  - gradients at keypoint scale σ_{keypoint}
  - grid size around keypoint (Lowe: 4x4)
  - nb of (scaled) pixels / grid cell (Lowe: 4x4, Vedaldi: 3x3)



Vedaldi 2006 © UCLA

- Gaussian weight on gradient magnitude
  - less emphasis far from center (affected by misregistration)
  - prevention of sudden changes
  - deviation related to grid width



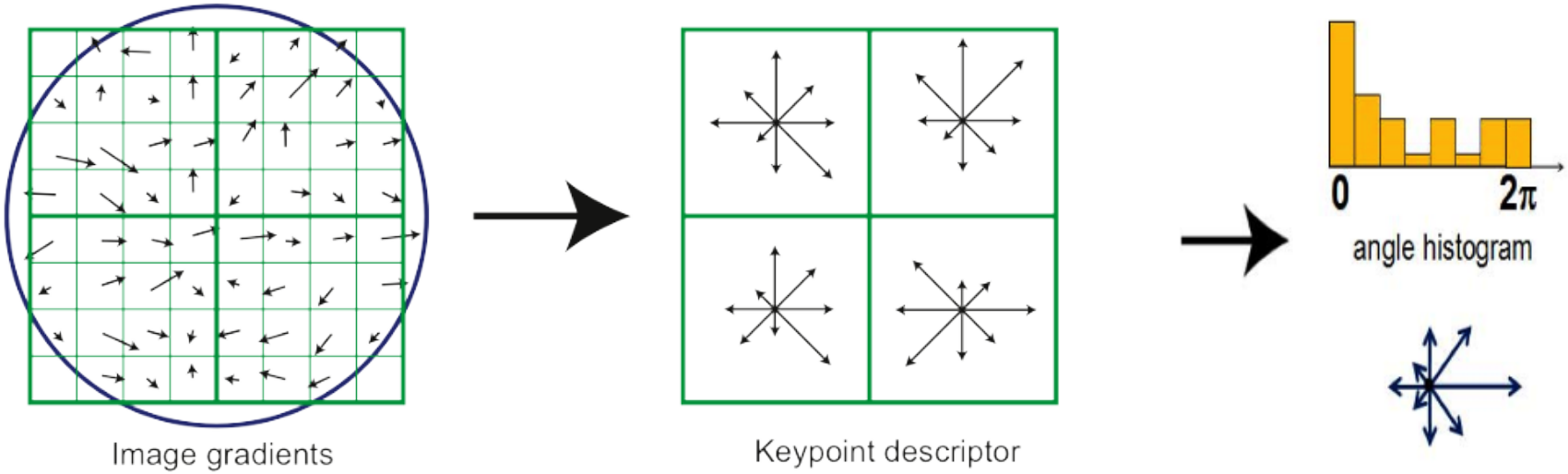
Lowe 2004 © Springer

[0.5 grid width, shown here on 2x2 grid of 4x4-pixel cells]

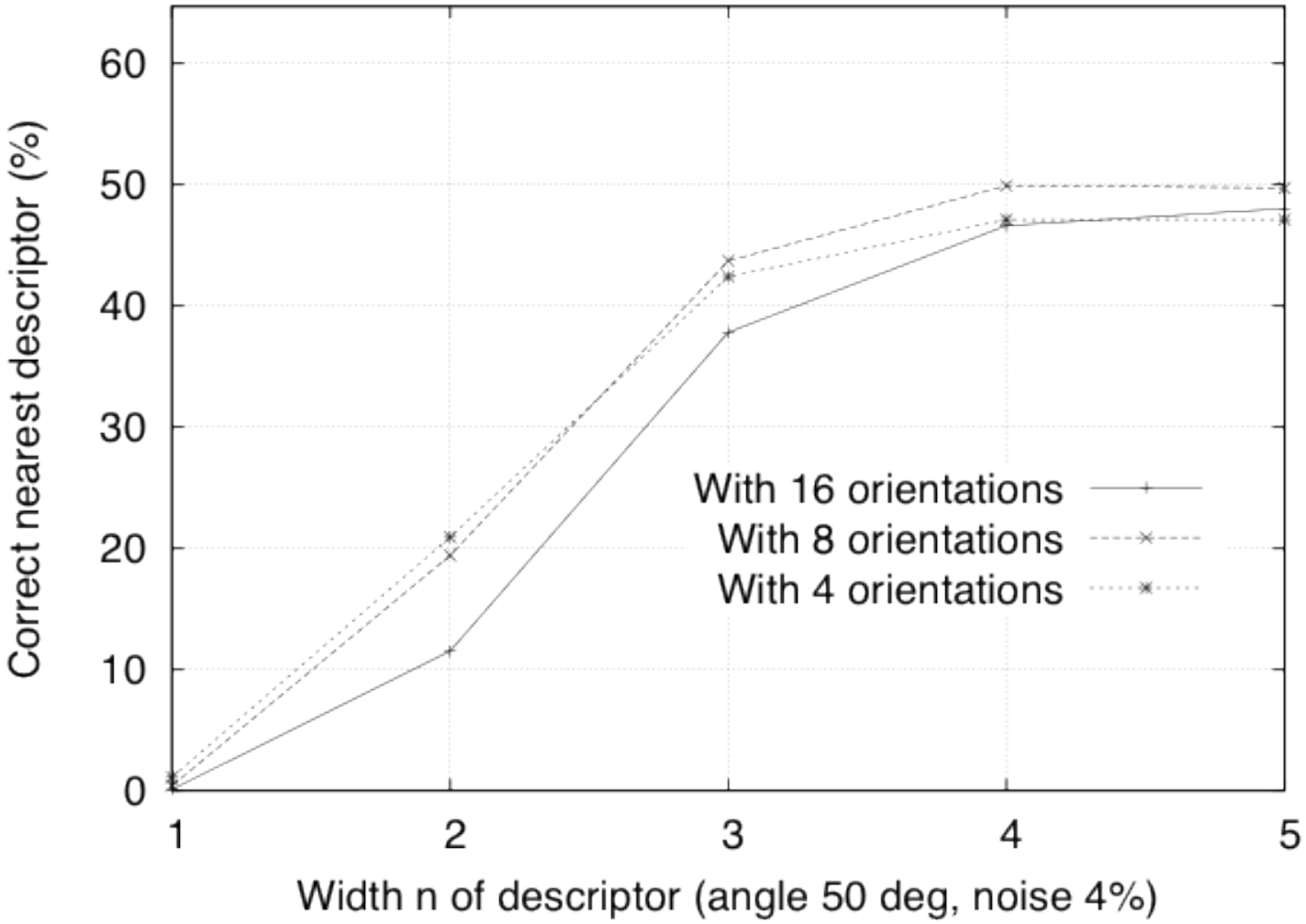
SIFT: gradient orientation histogram

- orientation bins (Lowe: 8), weighted by magnitude
- for each pixel of a given grid cell (Lowe: 4x4)

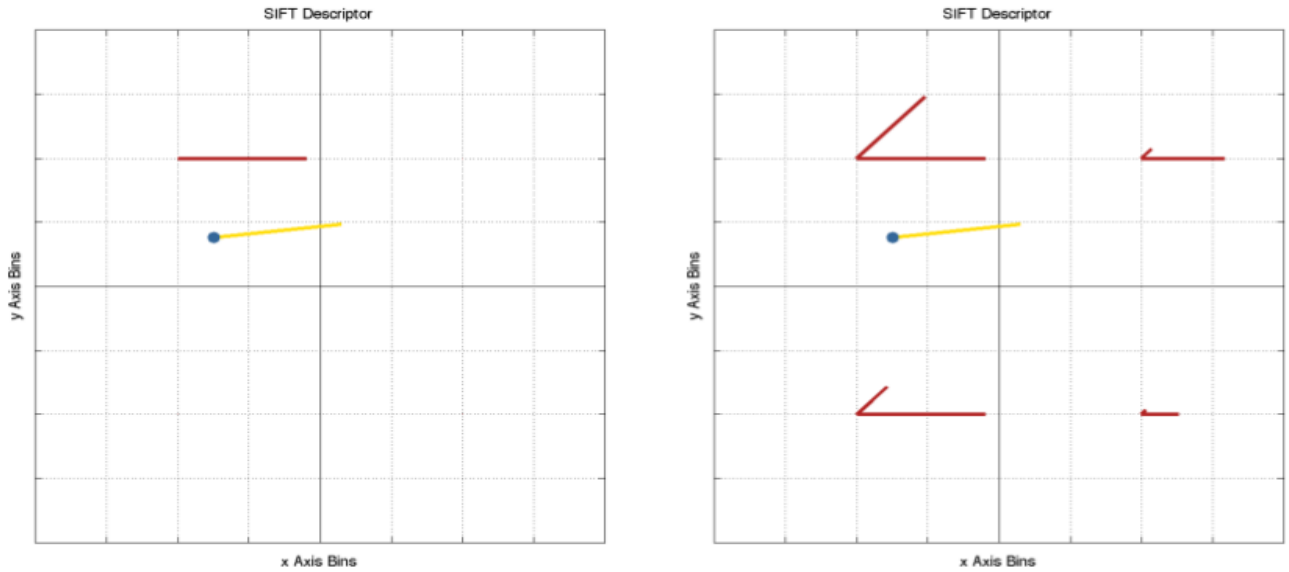
[shown: 2x2 grid of 4x4-pixel cells]



SIFT: gradient orientation histogram



SIFT: Histogram smoothing



(a) Descriptor without Interpolation (b) Descriptor with Interpolation

Figure 5: The effect of interpolation when generating the SIFT descriptor. *A single pixel's gradient*

- Avoid bin boundary effects: histogram smoothing
  - distribute each gradients into adjacent bins
  - trilinear interpolation (for each dimension: x, y, orientation)
    - multiply by weight $1-d$ where d distance to the central bin value

SIFT: robustness to illumination changes

Reduce effects of illumination change

- affine contrast change:
  - normalize vector to unit length
- non linear illumination (camera saturation, 3D surfaces): need to reduce the influence of large gradient magnitudes
  - max threshold value in normalized vector: not larger than 0.2
  - renormalize feature vector

-> greater emphasis on distribution of orientations

SIFT matching

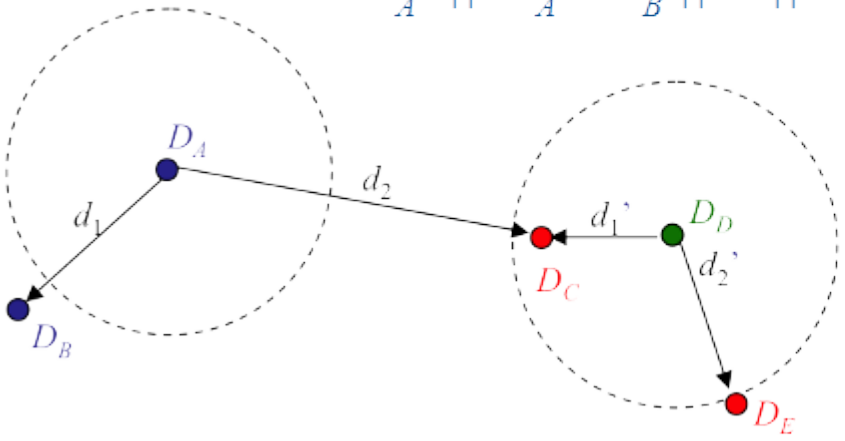
- Measure of similarity between descriptors
  - Euclidian distance

- Descriptor matching test
  - fixed threshold on distance ?
  - difficult to set (but can be learnt)
  - varies a lot depending on feature space regions

-> comparison with neighbors !

SIFT matching

- Matching with nearest neighbor (in 128D-space)
  - OK if 2nd nearest neighbor at least 20% (e.g.) more distant
    - score of D_A: $|| D_A - D_B || \; / \; || D_A - D_C ||$, OK if score < 0.8 (e.g.)



D_A matches D_B
D_C does not match D_D
although
$||D_A - D_B|| > ||D_C - D_D||$

    - 2nd nearest neighbor ~ rough estimation of false match density
    - discards (some) ambiguous matches
    - typically eliminates 90% false matches and 5% correct matches
  - relative threshold easier to set than absolute parameter

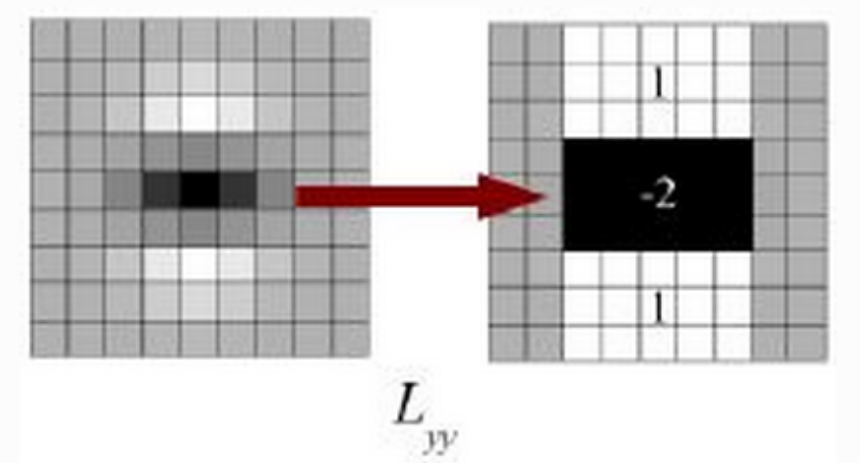
SIFT

- Tons of parameters (sizes, thresholds, etc.)
  - "good" parameters found by experimentation
    - ☛ possible bias towards used image database
- Many tiny details, some unsaid at all

☛ Most likely no 2 implementations give the same result

SURF

- Inspired by SIFT
- Faster, using approximations and integral images



Bay, H., Tuytelaars, T., & Van Gool, Surf: Speeded up robust features, *ECCV 2006*

Measuring matching performance

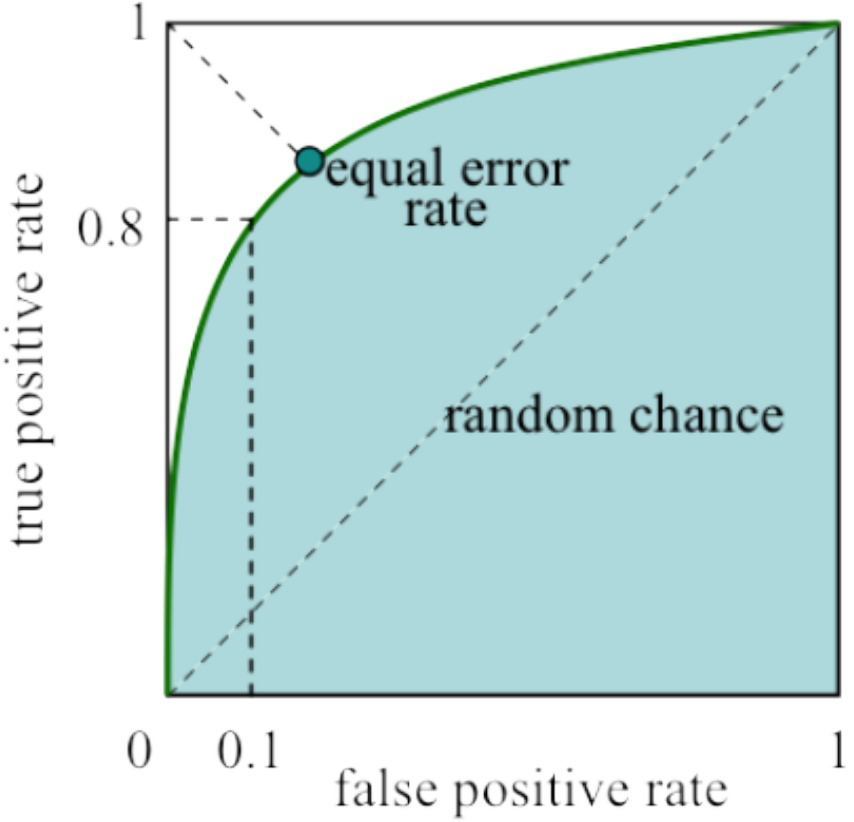
- true positives (TP): correct matches
- false positives (FP): proposed matches that are incorrect
- true negatives (TN): non-matches correctly rejected
- false negatives (FN): matches considered not matching

Combinations

- positive predictive value (precision) : PPV = TP/(TP+TN)
- true positive rate (recall): TPR = TP/(TP+FN) = TP/P
- false positive rate: FPR = FP/(FP+TN) = FP/N
- accuracy: ACC = (TP+TN)/(P+N)
- F-measure: 2 (PPV TPR)/(PPV + TPR) [harmonic mean]

ROC curve

- Plot (FPR, TPR) for one varying parameter of the matching strategy
- Good performance:
  - close to the upper left corner
  - large area under the curve (AUC)



TP: report and analysis

- **Find or — better! — implement** a Harris detector.
- Try various scenes. **Comment.**
- Try various parameters. **Comment.**
- Try with different smoothing kernels. **Comment.**
- Test sensitivity to rotation, noise, rescale. **Comment.**
- Test sensitivity to viewpoint changes. **Comment.**
- Look for image boundary effects. **Comment**.
- Implement and test ANMS. **Comment.**
- ☛ Beware of rapid conclusions with a single image or with a single change
- Send report + implemented code (C/C++ or Matlab)