SIFT, SURF GLOH descriptors

SIFT Scale Invariant Feature Transform

- SIFT method has been introduced by D. Lowe in 2004 to represent visual entities according to their local properties. The method employs local features taken in correspondence of salient points (referred to as *keypoints* or *SIFT points*)

- The original Lowe's algorithm:
  Given a grey-scale image:
  - Build a Gaussian-blurred image pyramid
  - Subtract adjacent levels to obtain a Difference of Gaussians (DoG) pyramid (so approximating the Laplacian of Gaussians)
  - Take local extrema of the DoG filters at different scales as keypoints

  For each keypoint:
  - Evaluate local gradients in a neighbourhood of the keypoint with orientations relative to the keypoint orientation and normalize
  - Build a descriptor as a feature vector with the salient keypoint information

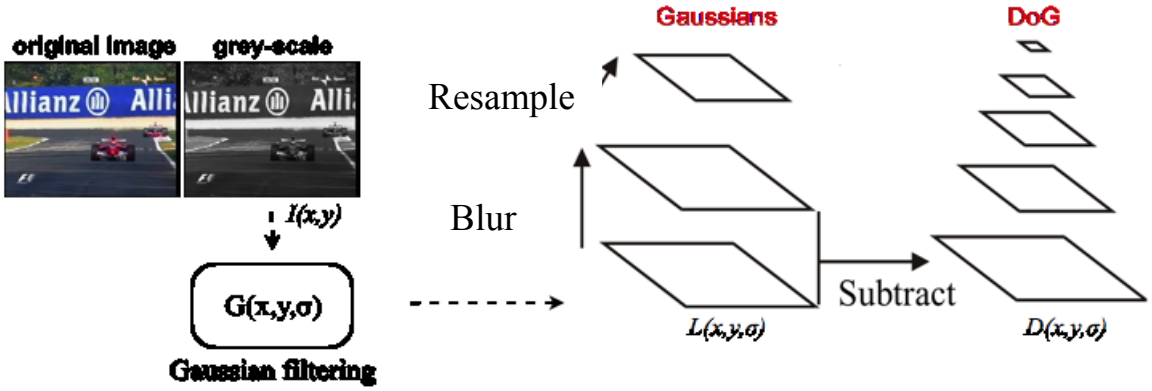
- SIFT descriptors are therefore obtained in the following three steps:
  A. Keypoint detection using local extrema of DoG filters
  B. Computation of keypoint descriptor
  C. SIFT descriptor derivation

- Keypoints (their SIFT descriptors) are used to characterize shapes with invariant properties
- Image points selected as keypoints and their SIFT descriptors are robust under:
  - Luminance change (due to difference-based metrics)
  - Rotation (due to local orientations wrt the keypoint canonical)
  - Scale change (due to scale-space)

SIFT Keypoint detection

Keypoints are local scale-space maxima of the Differences of Gaussians. They correspond to local min/max points in image *I(x,y)* that keep stable at different scales σ

A1. Obtain a Difference of Gaussians pyramid



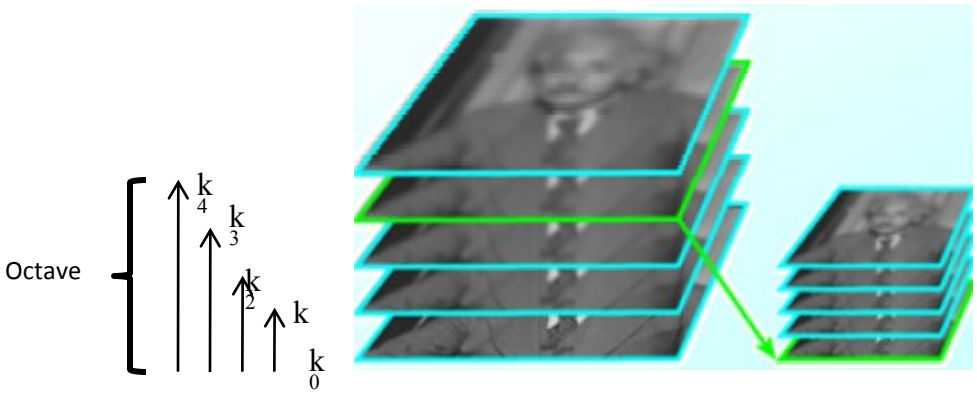
$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

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

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

Blur: σ is doubled from the bottom to top of each pyramid
Resample: pyramid images are sub-sampled from scale to scale
Subtract: adjacent levels of pyramid images are subtracted

- In order to build the pyramids, the original image is convoluted with a set of Gaussians, so as to obtain a set of images that differ by k in the scale space. Each of these sets is usually called *octave*.



- Each *octave* is divided into a number of intervals such as $k = 2^{1/s}$. For each *octave* s + 3 images must be calculated to obtain a correct extraction of local extrema.

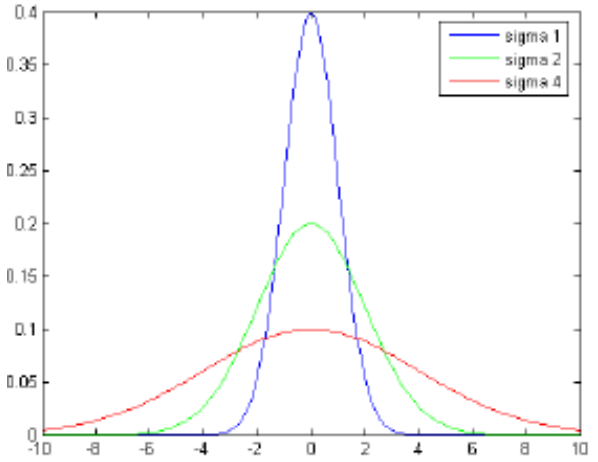
- For example if s = 2 then $k = 2^{½}$ and we will have 5 images at different scales:

$\sigma_0 = (2^{½})^0 \sigma = \sigma$
$\sigma_1 = (2^{½})^1 \sigma = \kappa \sigma$
$\sigma_2 = (2^{½})^2 \sigma = 2 \sigma$
$\sigma_3 = (2^{½})^3 \sigma = 2 \kappa \sigma$
$\sigma_4 = (2^{½})^4 \sigma = \kappa^2 \sigma$

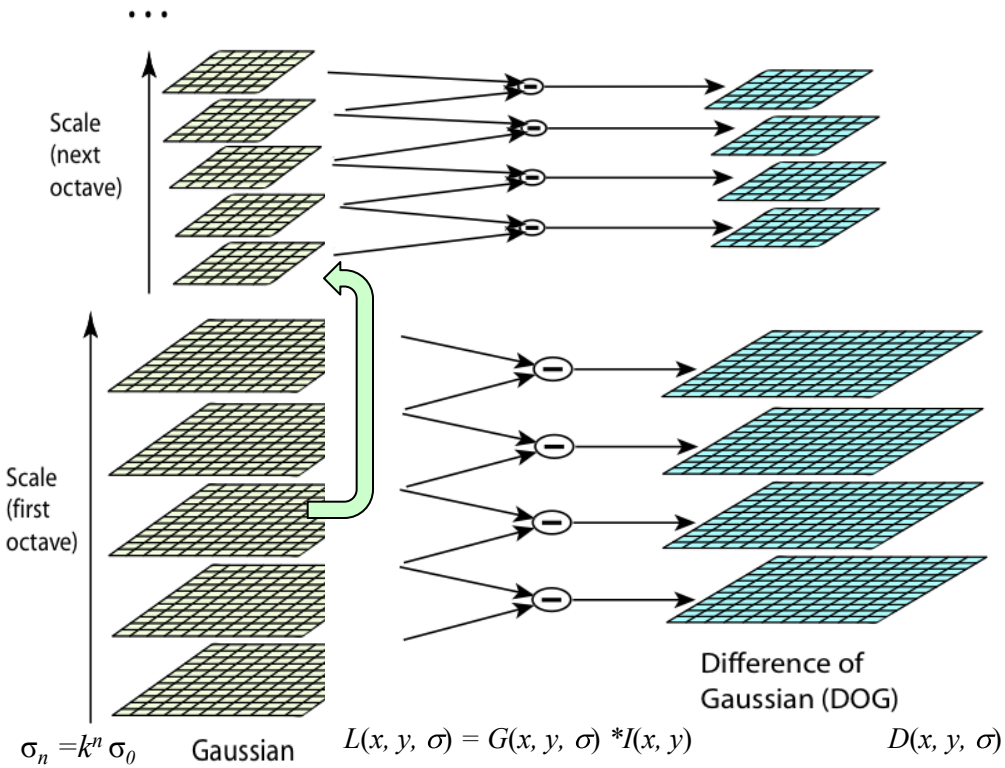
Gaussian smoothing

- The number of samples increases as σ increases. The number of operations that are needed are (N^2 -1) sums and N^2 products. They grow as σ grows.
- A good compromise is to use a sample interval of [-3σ, 3σ]

σ	Kernel [-3σ, 3σ]	Sums (N^2 – 1)	Products (N^2)
1	7x7	48	49
2	13x13	168	169
4	25x25	624	625
8	49x49	2400	2401



Building pyramids in detail



- A first pyramid is obtained by the convolution operation at different σ
- $L(x,y,\sigma)$ are grouped into a first "octave"
- After the first octave is completed the image such that $\sigma = 2\,\sigma_0$ is subsampled by a factor equal to 2 and the next pyramid is obtained in the same way
- The procedure is iterated for the next levels

$\sigma_n = k^n \sigma_0$

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

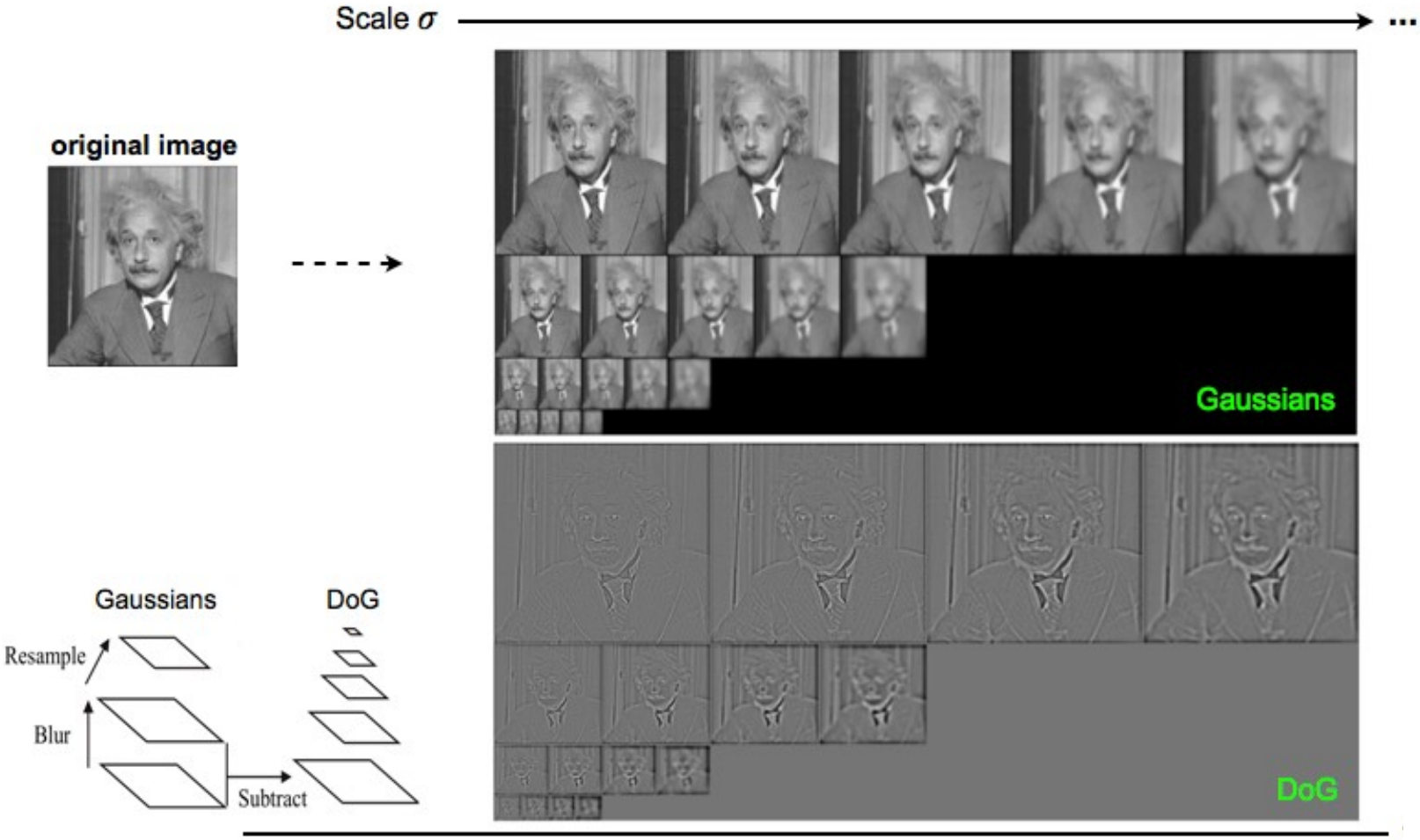
$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$

$\sigma_0 = (2^{1/2})^0 \sigma = \sigma$
$\sigma_1 = (2^{1/2})^1 \sigma = \kappa\,\sigma$
$\sigma_2 = (2^{1/2})^2 \sigma = 2\,\sigma$
$\sigma_3 = (2^{1/2})^3 \sigma = 2\,\kappa\,\sigma$
$\sigma_4 = (2^{1/2})^4 \sigma = \kappa^2\sigma$

The DoG at a scale σ is given by the difference of two nearby scales separated by a constant k

An octave corresponds to doubling the value of σ

Example


Scale σ
...
original image
Gaussians
DoG
Gaussians
DoG
Resample
Blur
Subtract

- Computational savings can be obtained considering that the Gaussian kernel is separable into the product of two one-dimensional convolutions (2N) products and (2N – 2) sums. This makes computational complexity O(N).

- Moreover convolution of two gaussians of σ_1^2 and σ_2^2 is a Gaussian with variance: $\sigma_3^2 = (\sigma_1^2 + \sigma_2^2)$. This property can be exploited to build the scale space, so to use convolutions already calculated

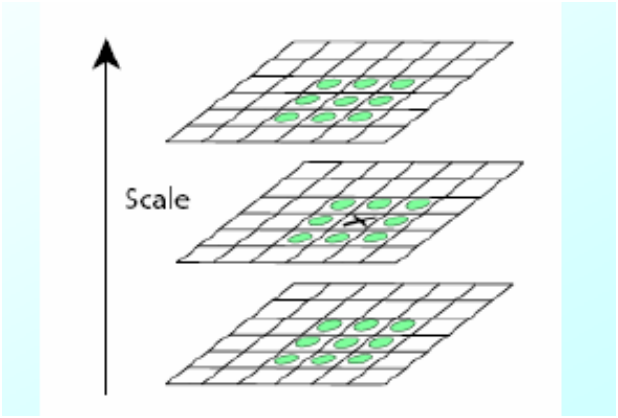
Detect maxima and minima of DoGs in scale space

- Local extrema of D(x,y,σ) are the local interest points. To detect the interest points at each level of scale of the DoG pyramid every pixel *p* is compared to its 8 neighbours:
  - if *p* is a local extrema (local minimum or maximum) it is selected as a *candidate keypoint*
  - each candidate is compared to the 9 neighbours in the scale above and below

    Only pixels that are local extrema in 3 adjacent levels are promoted as keypoints

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

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



- A quadratic function is adapted to the local points in order to determine the interpolated position, taking the Taylor series of D(x,y,σ) up to the second order, calculated at the local extrema.
- If the shift is higher than 0.5 the extrema is closer to another point. If so, the point is discarded and interpolation is performed with a different sample.

Keypoint stability

- The many points extracted from maxima+minima of DoGs must be filtered considering that low contrast keypoints are generally less reliable than high contrast and that keypoints that respond to edges are unstable. Filtering can be performed respectively by:
  - thresholding on simple contrast
  - thresholding based on principal curvature

- The local contrast can be directly obtained from $D(x,y,\sigma)$ calculated at the extrema:

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

Unstable extrema with low contrast can be discarded according to Lowe' rule: $|D(x) < 0{,}03|$

- To discard keypoints in correspondence with edges, it is appropriate to consider the Hessian of D(x,y,σ) and that the eigenvalues of H are proportional to the principal curvature of D(x,y,σ):

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

- If r is the ratio between the highest and the lowest eigenvalue, then:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

$(r+1)^2 / r$ is minimum when the two eigenvalues have the same value and increases as r increases. In order to have the ratio between the two principal curvatures below a threshold it must be:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

Example

Keypoint detection

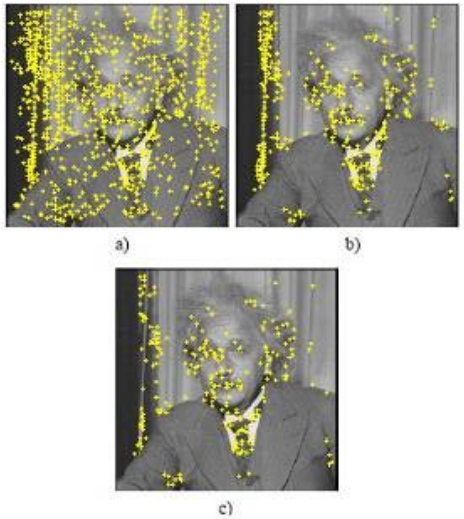


Figure 1 a) Maxima of DoG across scales. b) Remaining keypoints after removal of low contrast points. C) Remaining keypoints after removal of edge responses (bottom).

Final keypoints with selected orientation and scale

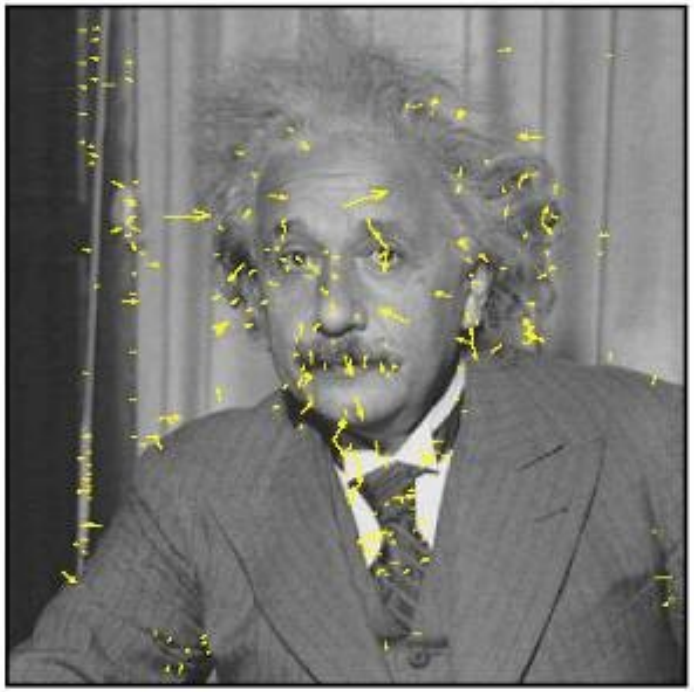


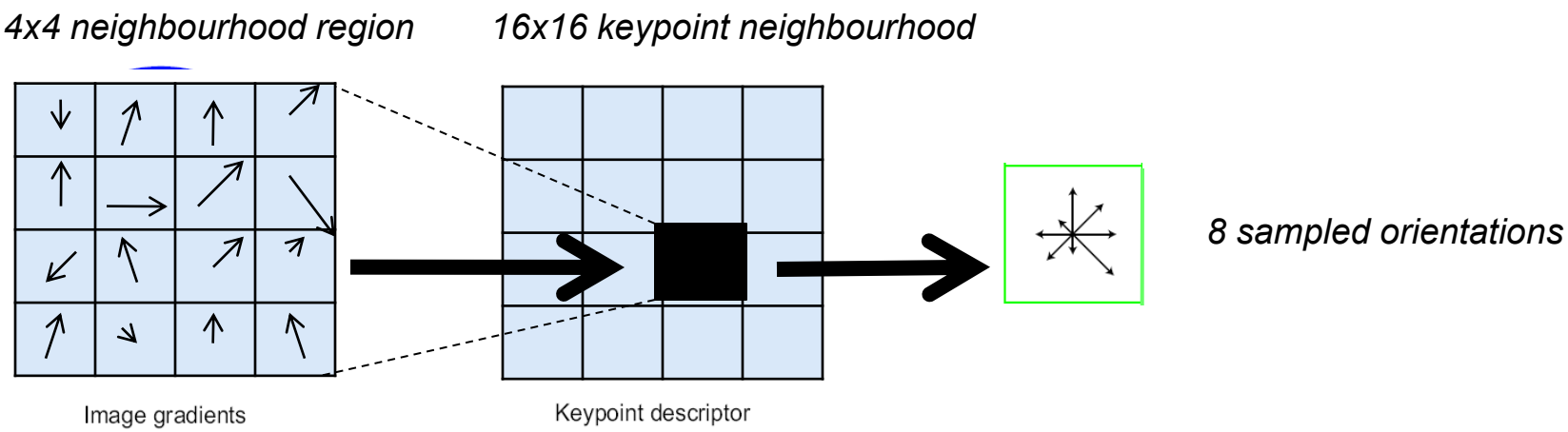
Figure 2 Extracted keypoints, arrows indicate scale and orientation.

Computation of SIFT descriptor

In order to makekeypoint estimate more reliable, it is usually preferable to use a larger aggregation window (Gaussian kernel size) than detection window.

The SIFT keypoint descriptor is obtained from thresholded image gradients sampled over a 16x16 array of locations in the neighbourhood of the detected keypoint, using the level of the Gaussian pyramid at which the keypoint was detected. For each 4x4 region samples they are accumulated into an 8-bin orientation histogram.

Gradient orientations are relative to the keypoint orientation



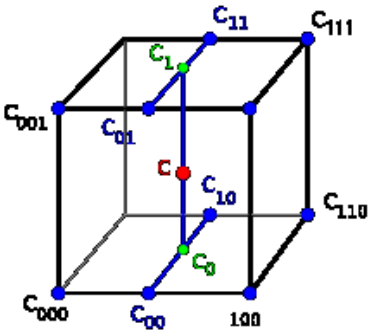
- A keypoint feature vector contains 16 x 8 = 128 elements. Empirically found to be best.

Gradient orientation histogram

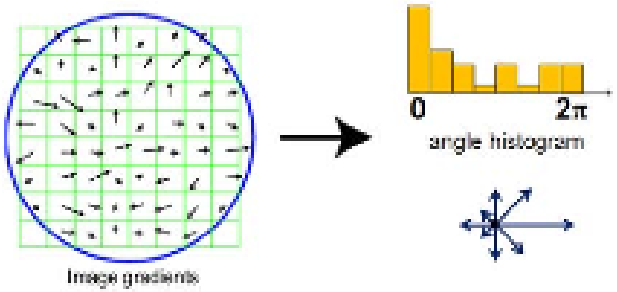
- The gradient *amplitude and direction* are computed from image differences.
- The gradient magnitudes are downweighted by a Gaussian function (blue circle) in order to reduce the influence of gradients far from the center, as these are more affected by small misregistrations (smoothed by a Gaussian function with s equal to 1.5 the keypoint scale).

- In each 4 × 4 quadrant, a 8-bin gradient orientation histogram is formed by *softly* adding the weighted gradient magnitudes of each of the original 256 to 2 × 2 × 2 histogram bins. The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.

Soft distribution of values to adjacent histogram bins is performed by trilinear interpolation to reduce the effects of location and dominant orientation misestimation

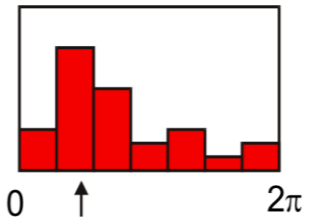


8x8 keypoint neighbourhood
(Lowe's original 16x16)



The SIFT dominant orientation at each detected keypoint can be obtained from the gradient orientation histogram : the peak (within 80% of the global maximum) of the histogram corresponds to the dominant direction of the local gradients (*canonical keypoint orientation*) .

Orientation histogram



Dominant direction

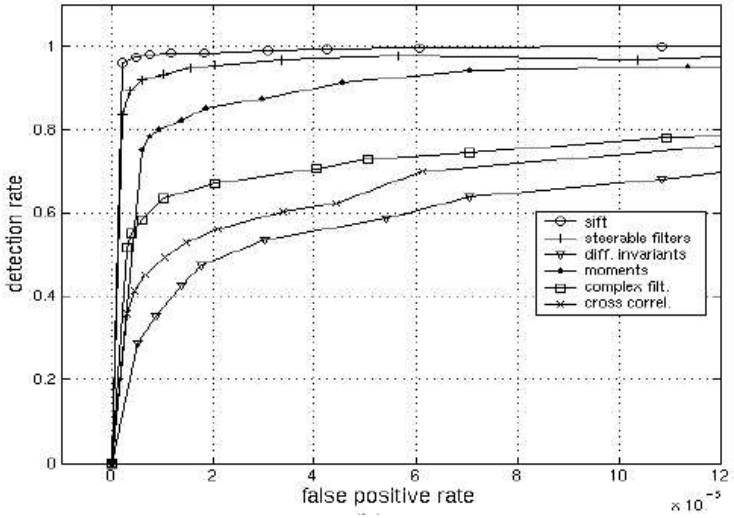
Once the local orientation and scale of a keypoint have been estimated, a scaled and oriented patch around the detected point can be extracted and used to form a feature descriptor

Illumination invariance of SIFT descriptor

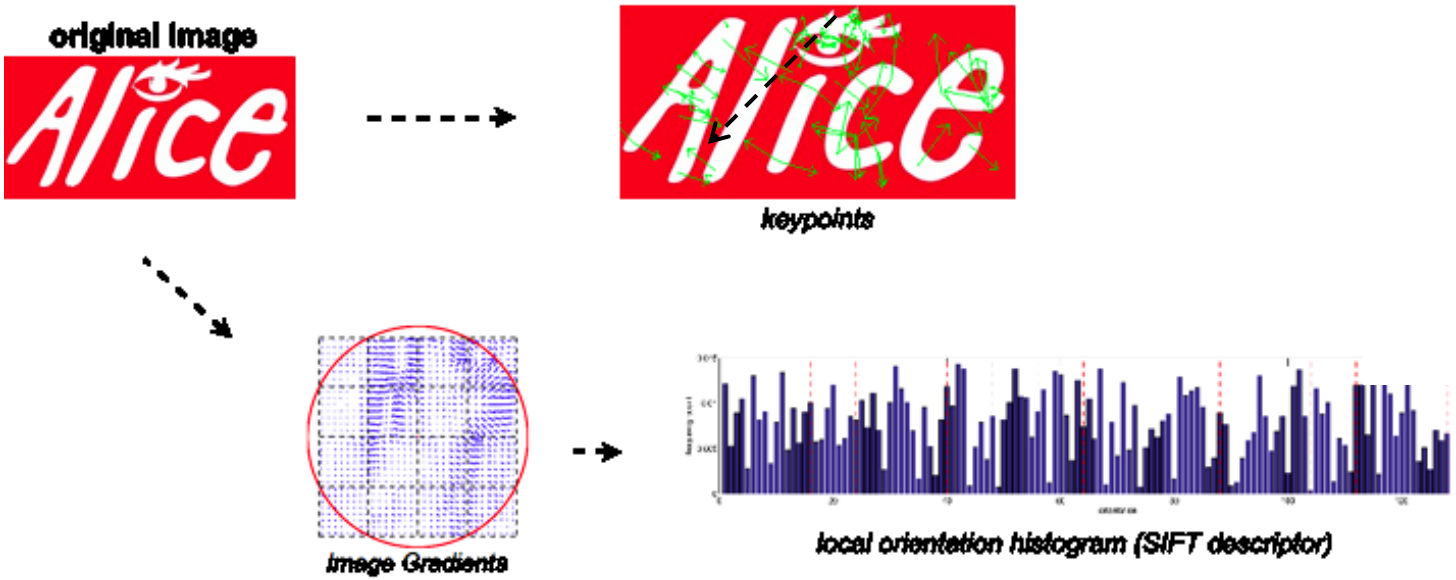
- The keypoint descriptor is normalized to unit length to make it invariant to intensity change, i.e. to reduce the effects of contrast or gain.

- To make the descriptor robust to other photometric variations, values are clipped to 0.2 and the resulting vector is once again renormalized to unit length.

- SIFT has been empirically found to show very good performance, invariant to *image rotation*, *scale*, *intensity change*, and to moderate *affine* transformations



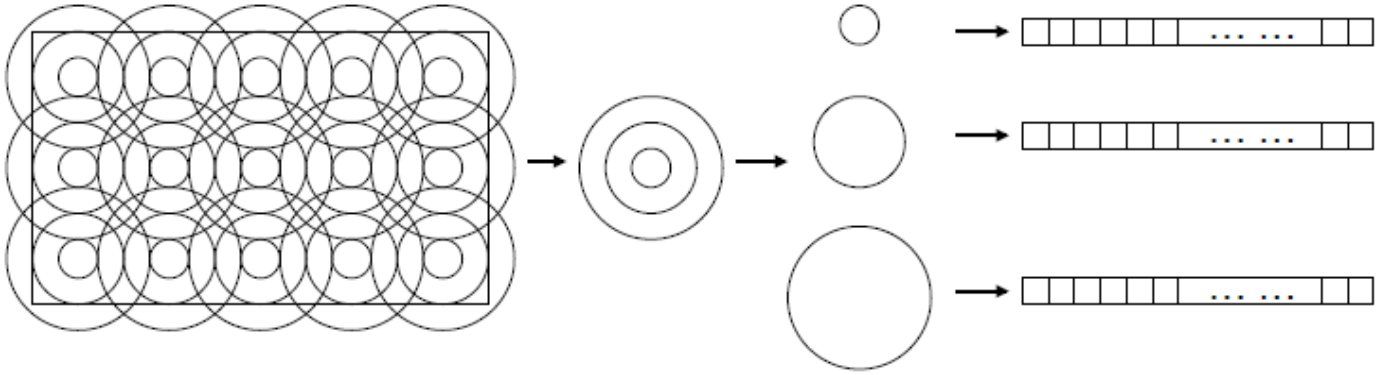
Example



- SIFT can be applied to image regions, independent of feature detectors. SIFT examples:
  - Gray SIFT (dense)
  - Color SIFT (dense)
  - Gray SIFT (sparse)

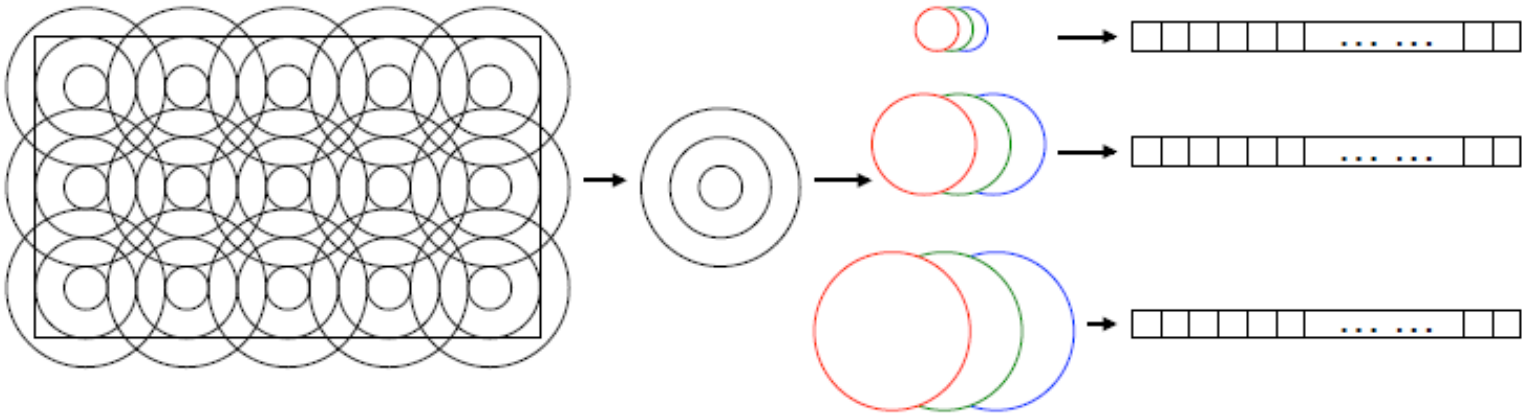
Gray SIFT

- Gray SIFT (dense)
  - At each center point, *n* 128-d SIFT descriptors are computed.
    (*n* = number of concentric circular regions)
  - Multiple descriptors are computed to allow for scale variation



HSV color SIFT

- HSV Color SIFT (dense)
  - At each center point, *n* 128x3-d SIFT descriptors are computed. (*n* = number of concentric circular regions)
  - Multiple descriptors are computed to allow for scale variation.



Considerations

- SIFT is the most used algorithm but it is quite slow (~6 sec for an image of size 1280x768): it is computationally expensive and copyrighted

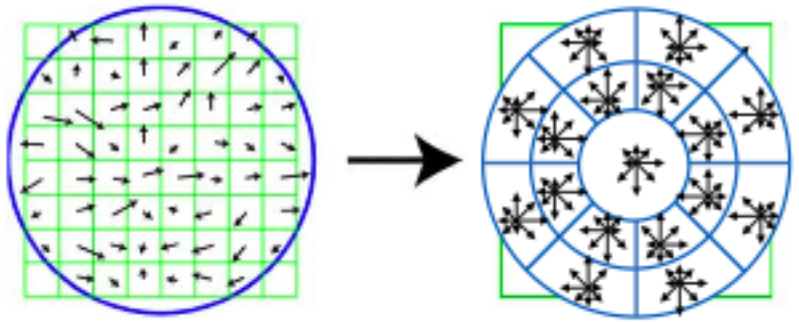
- Newest alternatives:
  - GLOH (Gradient Location and Orientation Histogram). TPAMI 2005.
    - larger initial descriptor + PCA
  - SURF (Speeded Up Robust Features)
    - faster than SIFT and sometimes more robust. ECCV 2006. (343, Google citations)

GLOH: Gradient Location and Orientation Histogram

- GLOH is a method for local shape description very similar to SIFT introduced by Miko in 2004

- Differently from SIFT it employs *a log-polar* location grid:
  - 3 bins in radial direction
  - 8 bins in angular direction
  - 16 bins for Gradient orientation quantization

- The GLOH descriptor is therefore a higher dimensional vector with a total of 17 (i.e. 2x8+1) * 16 = 272 bins. PCA dimension reduction is employed in the vector representation space



(a) image gradients (b) keypoint descriptor

Example

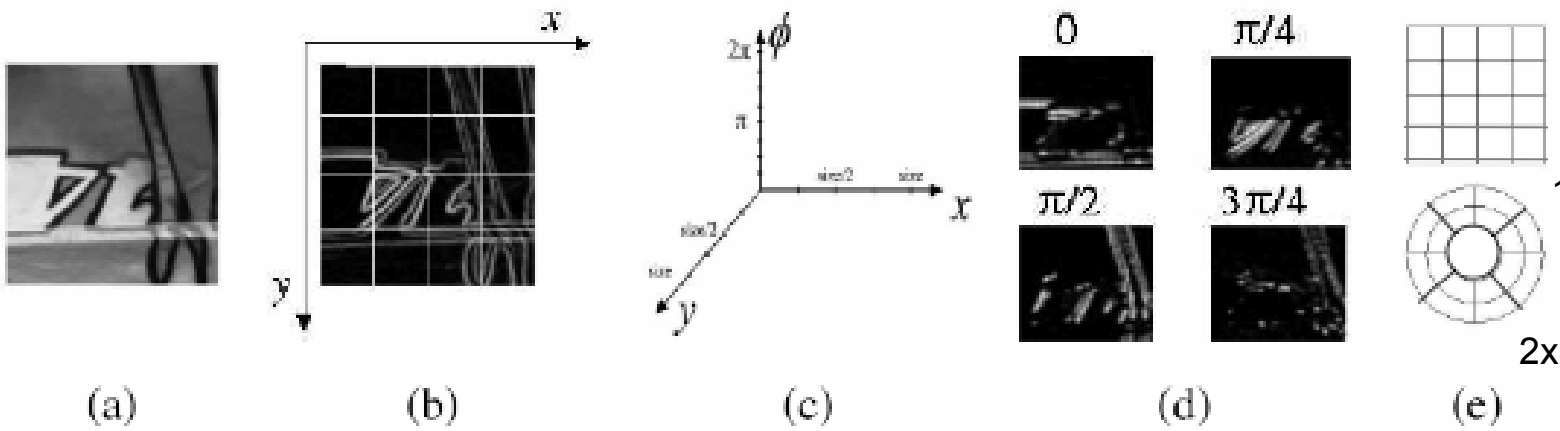


Fig. GLOH descriptor. (a) Detected region. (b) Gradient image and location grid. (c) Dimensions of the histogram. (d) 4 of 8 orientation planes. (e) Cartesian and log-polar location grids. Log-polar grid shows 9 location bins used in shape context (4 in angular direction). GLOH uses 17 bins (8 in angular direction)

SURF: Speed Up Robust Features

- SURF is a performant scale and rotation invariant interest point detector and descriptor. It approximates or even outperforms SIFT and other local descriptors with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster

- This is achieved by:
  - relying on integral images for image convolutions
  - building on the strengths of the leading existing detectors and descriptors (using a Hessian matrix-based measure for the detector, and a distribution-based descriptor)
  - simplifying these methods to the essential

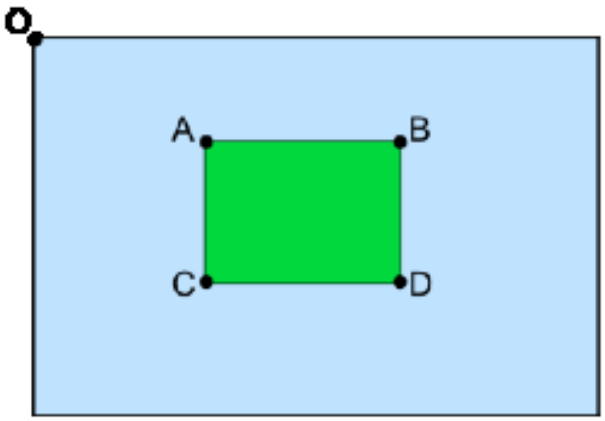
Integral images

- Much of the performance increase in SURF can be attributed to the usage of Integral Images:
  - Integral Image is computed rapidly from an input image *I*
  - Then it is used to speed-up the calculation of the area of any upright rectangular area

Given an input image *I* and a point (x,y) the *integral image* I_Σ is calculated by the sum of the pixel intensities between the point and the origin:

$$I_\Sigma(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(x,y)$$

Using this representation, the cost of computation for a rectangular area is only 4 additions



$$\sum = A + D - (C + B)$$

Fast-Hessian Detection

- SURF detector approximates the the second-order Gaussian derivatives of the image at point **x,** also referred to as Laplacian of Gaussians (LoG) by using box filter representations of the Gaussian kernels (differently from SIFT that uses instead Difference of Gaussians (DoG)).

- In the SURF approach interest points are detected at locations where the determinant of the Hessian Matrix is maximum. SURF permits a very efficient implementation, making good use of integral images to perform fast convolutions of varying size box filters (at near costant time)

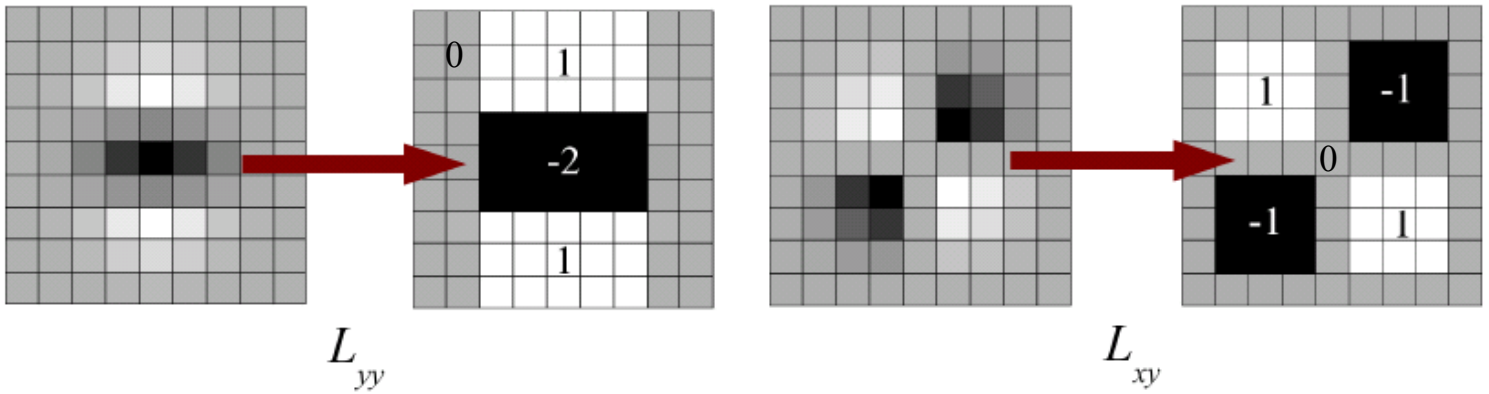
The Hessian matrix *H* is calculated as a function of both space and scale:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

$L_{xx}(x, \sigma)$, $L_{yy}(x, \sigma)$, $Lxy(x, \sigma)$ are the second-order Gaussian derivatives of the image at point **x** (LoGs)

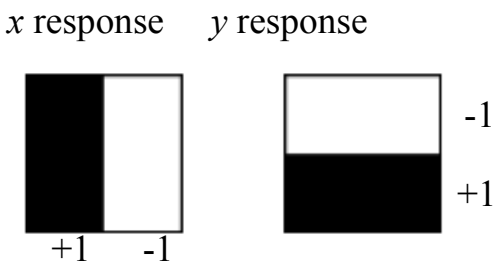
LoG approximation

- Approximated second order derivatives (LoGs) with box filters:



The 9x9 box filters in the figure are approximations of a Gaussian with σ=1.2 and represent the lowest scale

- Haar-Wavelets are simple filters which can be used to find gradients in *x* and *y* directions

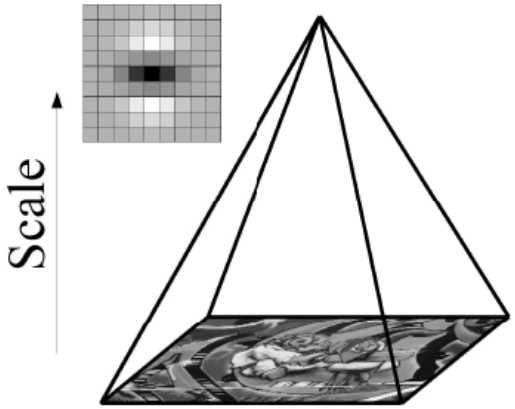


When used with integral images each wavelet requires 6 operations to compute

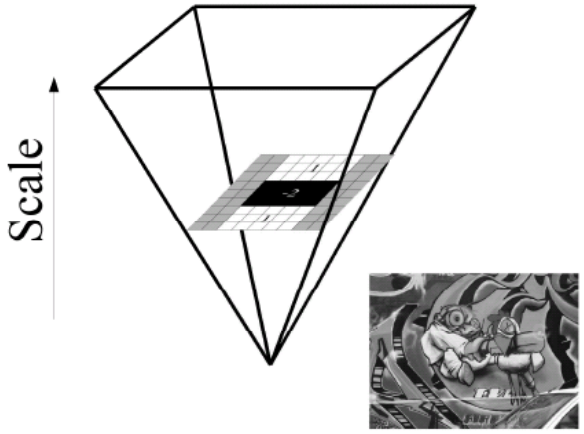
SURF descriptor

- The approach of SURF is similar to that of SIFT but ...
  - Integral images in conjunction with Haar Wavelets are used to increase robustness and decrease computation time

  - Instead of iteratively reducing the image size (like in the SIFT approach), the use of integral images allows the up-scaling of the filter at constant cost



"SIFT approach"



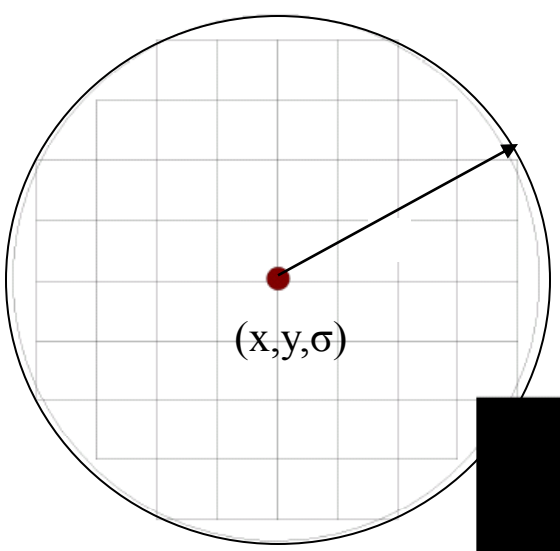
"SURF approach"

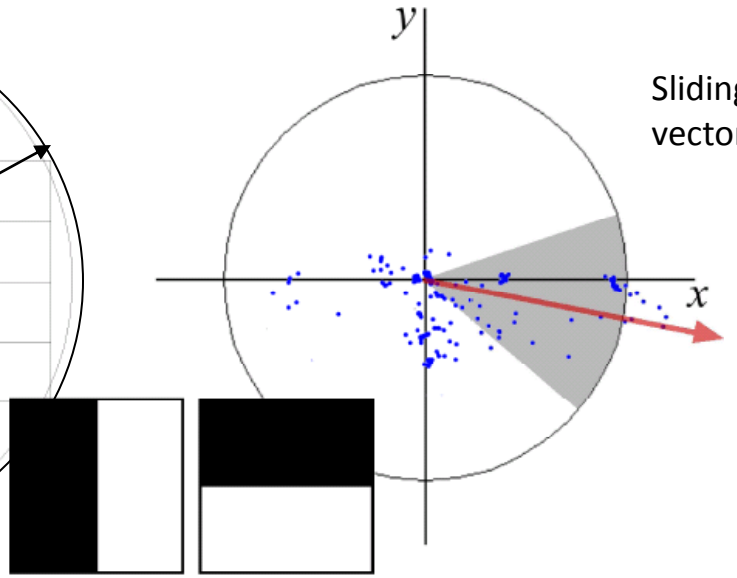
- The SURF descriptor computation can be divided into two tasks:
  - Orientation assignment
  - Extraction of descriptor components

Orientation assignment

- In order to achieve invariance to image rotation each detected interest point is assigned a reproducible orientation.To determine the orientation, Haar wavelet responses of 4σ are calculated for a set of pixels within radius 6σ of the detected point:
  - The Haar wavelet responses are represented as vectors.
  - The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of 60 degrees.

Circular neighborhood of radius 6σ





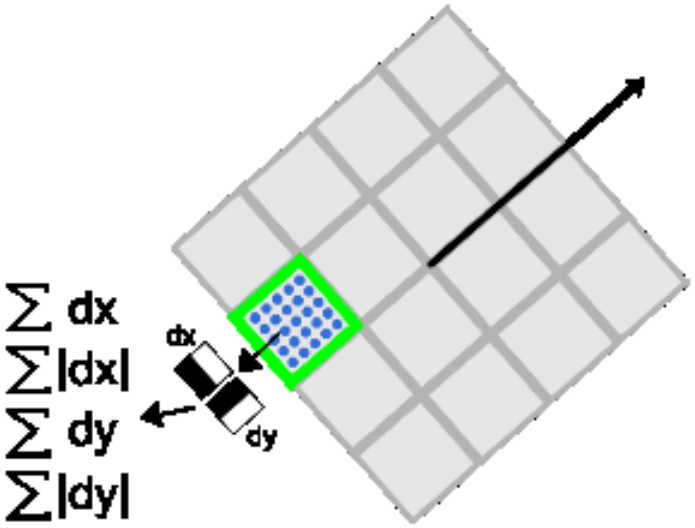
Sliding orientation window: the longest vector is the dominant orientation

Extraction of Descriptor components

- To extract descriptor components, a square window of size 20σ is taken around the interest point oriented along the dominant orientation.
- The window is divided into 4x4 regular subregions
- Haar wavelets of size 2σ are calculated for 25 regularly distributed sample points in each subregion
- The *x* and *y* wavelet responses (*dx*, *dy*) are collected for each subregion according to:

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]$$

- Each subregion therefore contributes 4 values to the SURF descriptor vector leading to an overall vector of length 4x4x4 = 64



The green square bounds one of the 16 subregions and blue circles represent the sample points at which we compute the wavelet responses

x ad y responses are calculated relative to the dominant orientation

Computational cost

- SURF computational cost (detector and descriptor) against the most-used detectors and the SIFT descriptor:

Table 1. Thresholds, number of detected points and calculation time for the detectors in our comparison. (First image of Graffiti scene, 800 × 640).

detector	threshold	nb of points	comp. time (msec)
Fast-Hessian	600	1418	120
Hessian-Laplace	1000	1979	650
Harris-Laplace	2500	1664	1800
DoG	default	1520	400

Table 2. Computation times for the joint detector - descriptor implementations, tested on the first image of the Graffiti sequence. The thresholds are adapted in order to detect the same number of interest points for all methods. These relative speeds are also representative for other images.

	U-SURF	SURF	SURF-128	SIFT
time (ms):	255	354	391	1036

SURF descriptor: alternative versions

- U-SURF (Upright version)
  - For some applications rotation invariance is not necessary.
  - U-SURF (Upright version) of SURF does not implement the *orientation assignment* step.
  - It is faster while maintaining a robustness to rotation of about +/- 15 degrees

- SURF-128
  - SURF-128 implements a descriptor vector of 128 length.
  - The sum of d_x and $|d_x|$ are computed separately for $d_y < 0$ and $d_y > 0$ (and so for d_y and $|d_y|$)
  - It is more precise and not much slower to compute, although slower to match

Other SIFT-like Implementations

- GIST: Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention, TPAMI 2007
- LESH: Head Pose Estimation In Face Recognition Across Pose Scenarios, VISAPP 2008
- PCA-SIFT: A More Distinctive Representation for Local Image Descriptors, CVPR 2004
- Spin image: Sparse Texture Representation Using Affine-Invariant Neighborhoods, CVPR 2003