# Key properties of local features

- Locality, robust against occlusions
- Must be highly distinctive, a good feature should allow for correct object identification with low probability of mismatch
- Easy to extract and match, efficiency
- Quantity: many features from small objects
- Invariance to:
  - noise
  - changes in illumination
  - scaling
  - rotation
  - viewing direction (to a certain extent)

# Scale Invariant Feature Transform (SIFT)

- Well engineered local features, designed to address these requirements
- Proposed by David Lowe “Distinctive Image Features from Scale Invariant Features”, International Journal of Computer Vision, Vol. 60, No. 2, 2004, pp. 91-110

*Slides adapted from D.Lowe and Cordelia Schmid, Recognition and Matching based on local invariant features, CVPR 2003.*

# Steps

- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- Generation descriptors

# Finding keypoints

- Detect points in which the Laplacian (of Gaussian) of the image is large (Lindeberg)
- This has been experimentally shown to be a stable image feature compared to gradients or Harris corner detection functions
- To achieve invariance to scale, use the normalized LoG:

$$LoG_{norm} = \sigma^2 \left( \frac{\partial^2 G_\sigma}{\partial x^2} + \frac{\partial^2 G_\sigma}{\partial y^2} \right) = \sigma^2 \nabla^2 G$$

- Build a scale space, each octave is a collection of images smoothed in sequence:

$$\sigma, k\sigma, k^2\sigma, \ldots$$

- When we reach a new octave (sigma doubles), downsample and start a new octave
- If each octave is made of *s* intervals:

$$k = 2^{1/s}$$

- Adjacent image scales are subtracted to produce the difference-of-Gaussian images

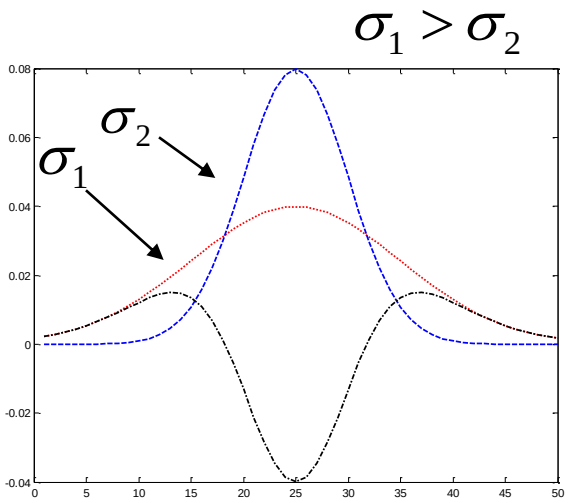
- Recall that the Laplacian of Gaussian can be approximated as a difference of two Gaussians
- Smoothed images L can be computed efficiently
- D is then computed as simple image subtraction
- It can be shown that the difference incorporates the normalization factor *t* ($\sigma^2$)

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

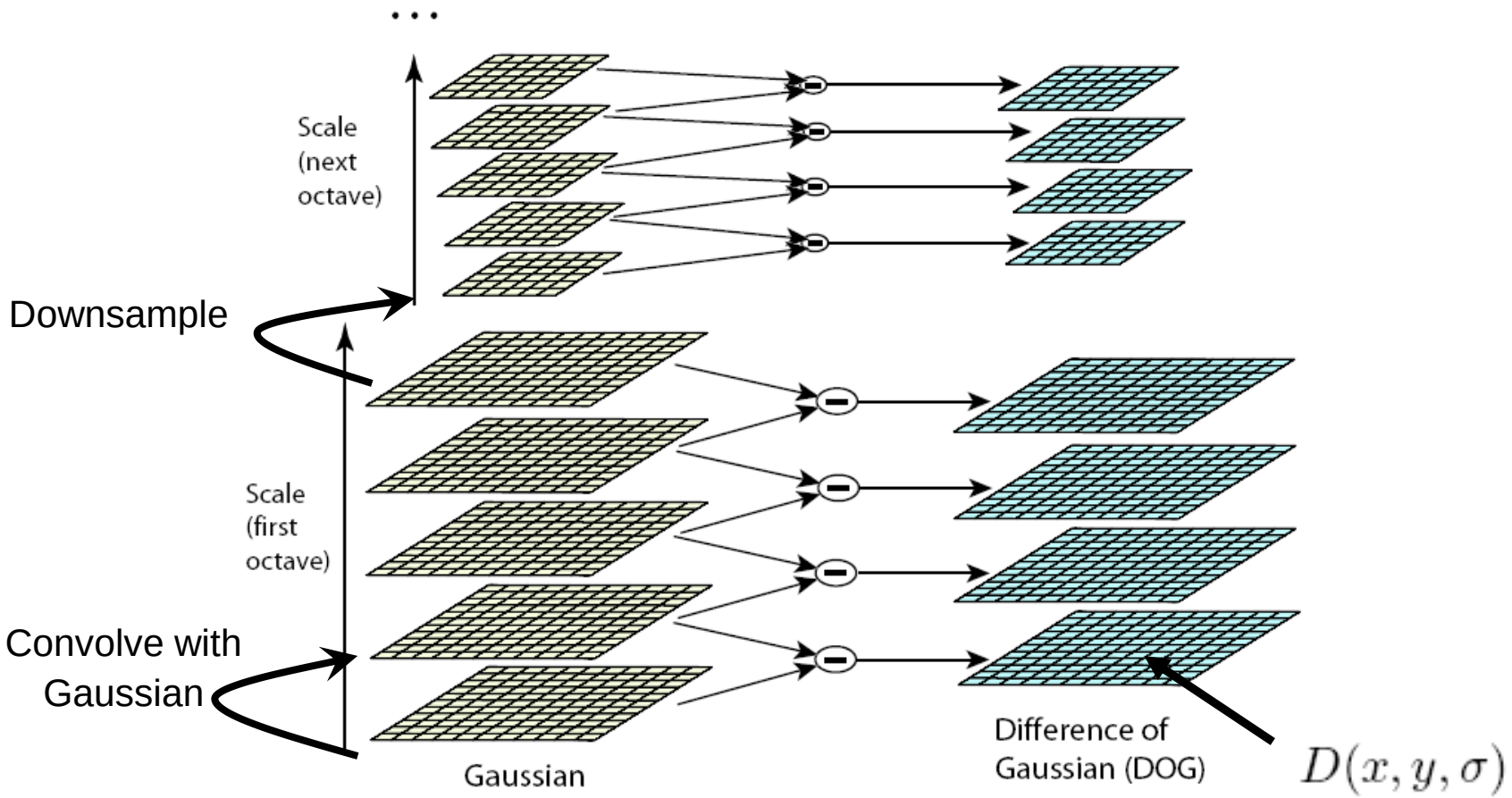
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

$$L(x, y, k\sigma) - L(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

K-1 is constant across scales



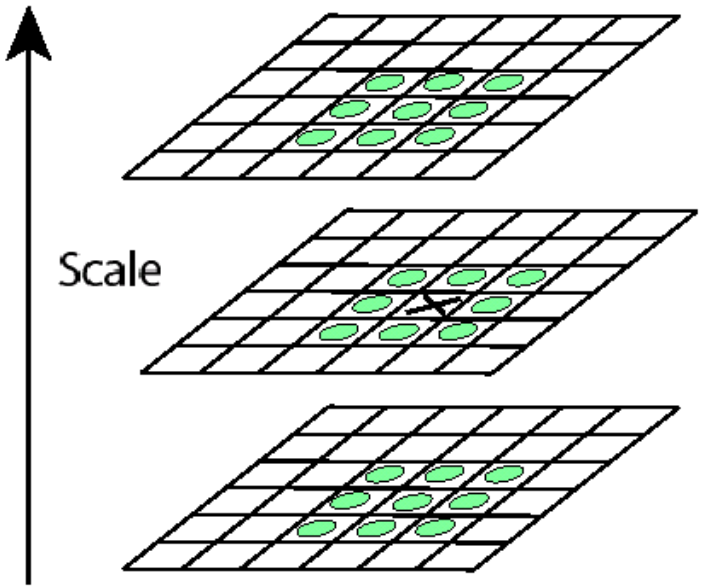
# Graphically…



# Local extrema detection

- Find local extrema (max or min) of DoG
- Compare each point with its eight neighbors at the same scale and nine neighbors at two nearby scales

$$\mathbf{x} = (x, y, \sigma)$$



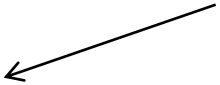
# Improve reliability

- For each candidate keypoint
- Discard low contrast keypoints
- Discard edges, difference-of-Gaussian has strong response to edges, unfortunately edges are unstable (due to translation or noise)

$$\left|D(x)\right| < t \Rightarrow \text{ low contrast point}$$

- Check principle curvature of D(), edges will have large principal curvature across edge but small one in the perpendicular direction

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



estimating using differences

- the eigenvalues of H are proportional to the principle curvatures of D, as for the Harris corner detector we can avoid computing the eigenvalues

$$th(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

$$\det(H) = D_{xx} \cdot D_{yy} - D_{xy}^2 = \lambda_1 \lambda_2$$

check that $\det(H) > 0$

$$\frac{tr(H)^2}{\det(H)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r+1)^2}{r}$$

$$r = \lambda_1 / \lambda_2$$

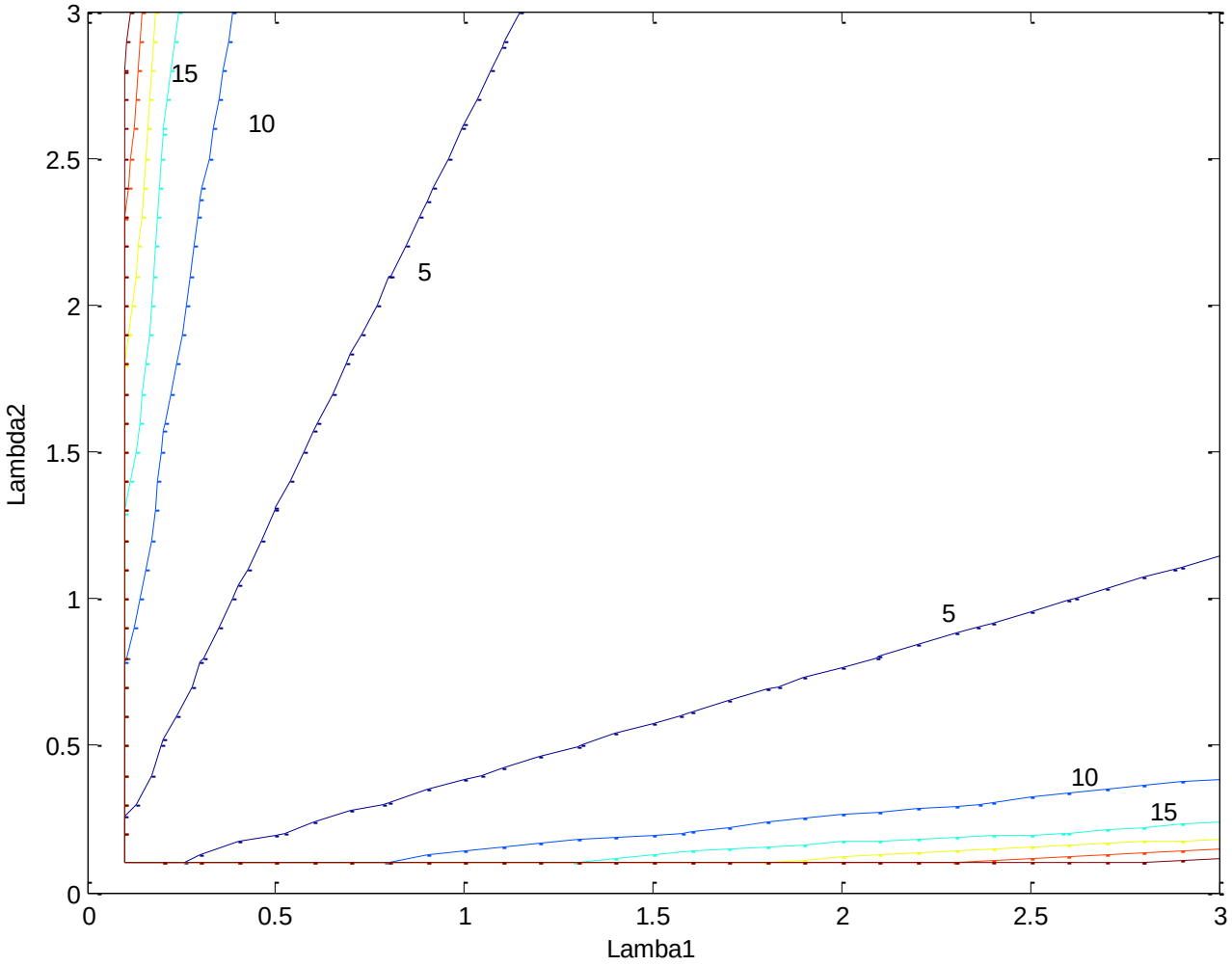
$\frac{tr(H)^2}{\det(H)}$ is min when $r = 1$, increases elewhere

pick $\hat{r} \Rightarrow t = \frac{(\hat{r}+1)^2}{\hat{r}}$

keep points for which $\frac{tr(H)^2}{\det(H)} < t$

e.g. $\hat{r} = 10$

$$\frac{tr(H)^2}{\det(H)}$$



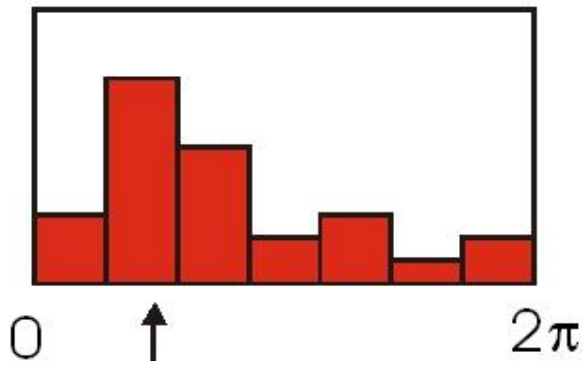
# Orientation assignment

For the selected keypoint, at the closest scale:

- compute a gradient orientation histogram
- determine dominant orientation ➔ assign this orientation to the keypoint

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \arctan((L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)))$$

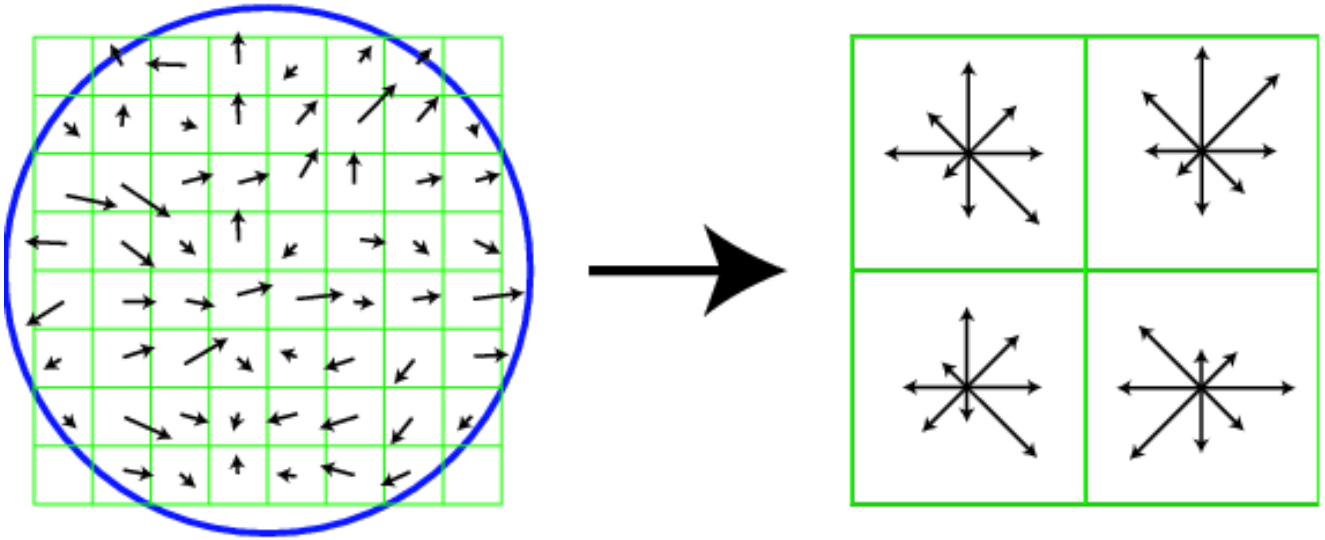


# Orientation assignment: technicalities

- Histogram size is 36 bins
- Use magnitude of gradient as weight in the histogram, multiplied by Gaussian weights
- Given scale *s*, use Gaussian with *sigma=1.5*s*
- Detect highest peak and any other local peak within 80% of the highest peak → these local peaks generate extra keypoints (experimentally this happens 15% of the times)
- Fit a parabola to 3 points in the histograms closest to the match, and re-detect maximum for better accuracy

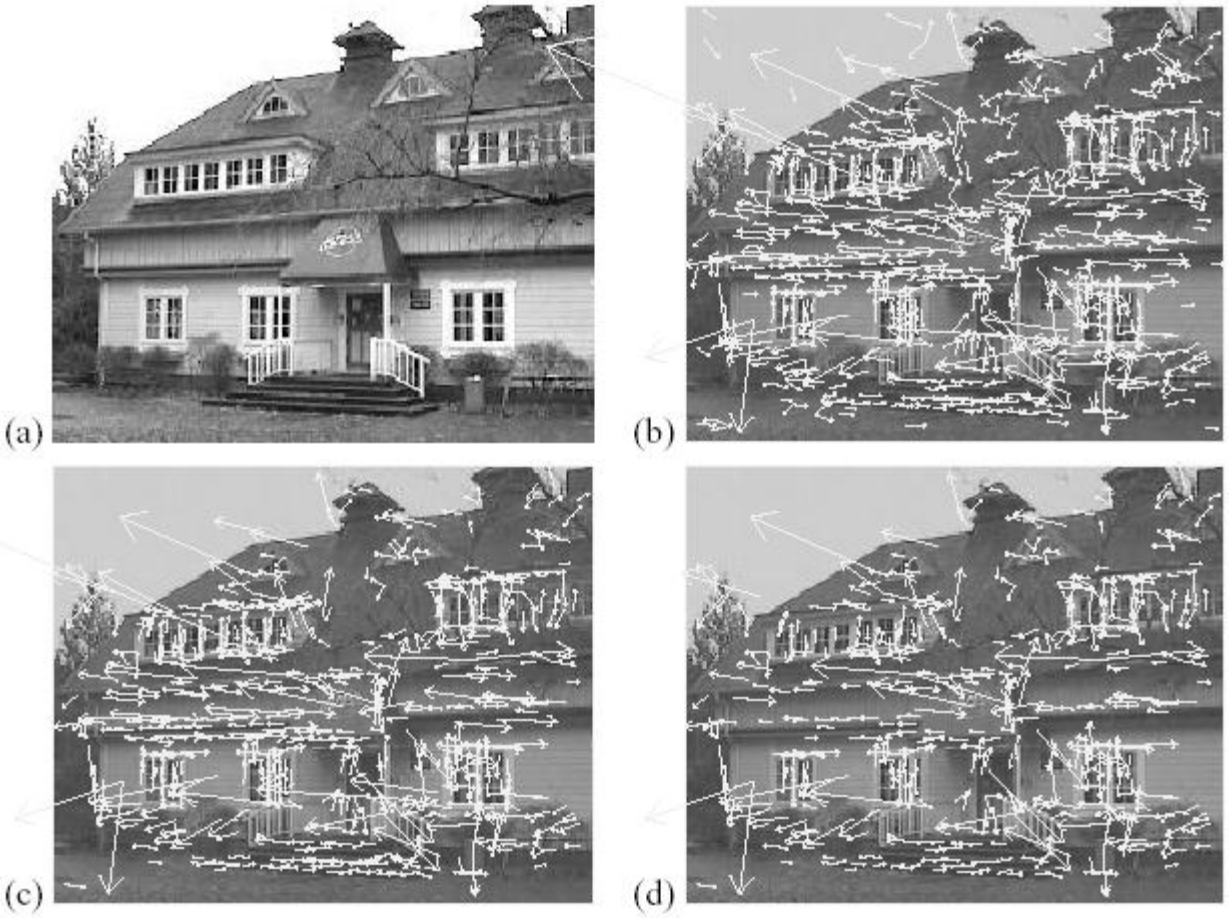
# Descriptor

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- To achieve orientation invariance, orientations are rotated relative to the keypoint orientation (prev. slide)
- 8 orientations x 4x4 histogram array = 128 dimensions
- Below: schematic representation  (in this case of a 2x2 descriptor from a 8x8 array of locations)



- Orientations are weighted with Gaussian sigma=scale/2
- Brightness change will not affect gradients, since they are computed from pixel differences
- A change in image contrast, in which all pixels of an image are multiplied by a constant, will multiply gradients of the same amount
  - To reduce the effect of illumination change, keypoint descriptors are normalized to unit length
- To reduce artifacts due to saturation: remove large gradients after first normalization (>0.2), and re-normalize the result

# Example of keypoint detection




(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

# Keypoint matching

- The best match for each keypoint is found as the nearest neighbor in a database of SIFT features from training images
- Use Euclidian distance between descriptors

$$k1(l), \quad l \in [1,128]$$

$$k2(l), \quad l \in [1,128]$$

$$d(k1,k2) = \left\| k1 - k2 \right\|^2$$

- How do we discard features that do not have a good match? Pick a global threshold?
- Lowe suggests using ratio of nearest neighbor to ratio of second nearest neighbor
- This measures performs well: correct matches need to have closest match significantly closer than the closest incorrect match
- False match: there is likely to be a number of other false matches within similar distances due to the high dimensionality of the feature space

- Matching:

$k_i(l) \quad \mathrm{i} = 1,...\mathrm{M}$

$\mathrm{compute:}$

$d_{i,j} = \left\| k_i - k_j \right\|$

$\mathrm{sort}\ k_{\mathrm{j}}\ \mathrm{depending\ on}\ d_{i,j},\ j \neq i$

$k_{1st}, k_{2\mathrm{nd}}, k_{3\mathrm{rd}} \ldots$

$$score = \frac{d_{i,1st}}{d_{i,2nd}}$$

$\mathrm{consider\ good\ match\ if}\ score > 0.8$

# Curiosity: real implementation

- To avoid exhaustive search Lowe proposes an approximate algorithm called Best-Bin-First, which returns the closest neighbor with high probability
- Priority search, with cut-off after checking 200 first candidates
- For a database of 100K keypoints speed up of 2 orders of magnitude, with less than 5% of correct match loss

# Conclusions

- SIFT features are reasonably invariant to rotation, scaling, and illumination changes
- We can use them for matching and object recognition among other things
- Robust to occlusion, as long as we can see at least 3 features from the object we can compute the location and pose
- Efficient on-line matching, recognition can be performed in close-to-real time (at least for small object databases)

# Empirical evaluation of the parameters

- In the paper most parameters are determined empirically
- 32 real images, outdoor scenes, human faces, aerial photographs, industrial images
- Range of “synthetic” transformation, rotation, scaling, affine stretch, change in brightness, contrast, noise
- Because transformations were synthetic, it was possible to know where each feature will be in the transformed image and compare the result of the match
- Check repeatability:
  - # of keypoints that are detected after the transformation, in the correct location and scale
  - # of keypoints that are successfully matched using the *nearest descriptor* technique



*details in Lowe 2004 IJCV*

# 1) Frequency of sampling in scale

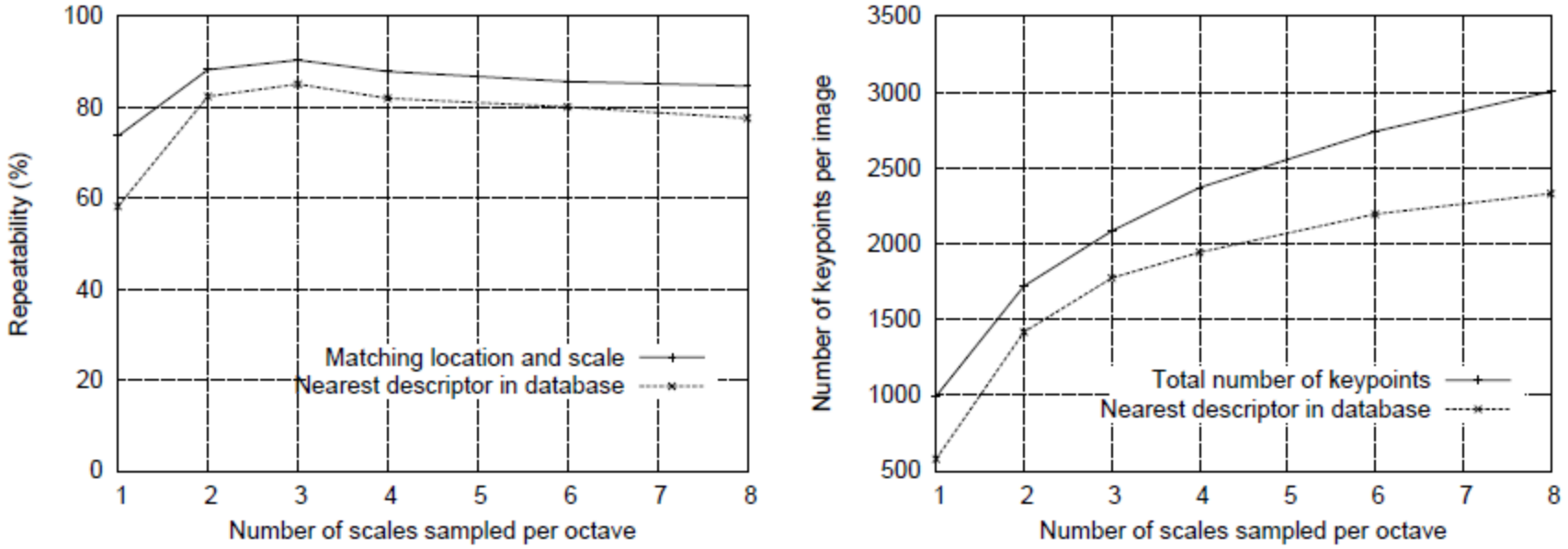



Figure 3: The top line of the first graph shows the percent of keypoints that are repeatably detected at the same location and scale in a transformed image as a function of the number of scales sampled per octave. The lower line shows the percent of keypoints that have their descriptors correctly matched to a large database. The second graph shows the total number of keypoints detected in a typical image as a function of the number of scale samples.

…best value appears to be 3 scales per octave

# 2) Smoothing for each octave (sigma)

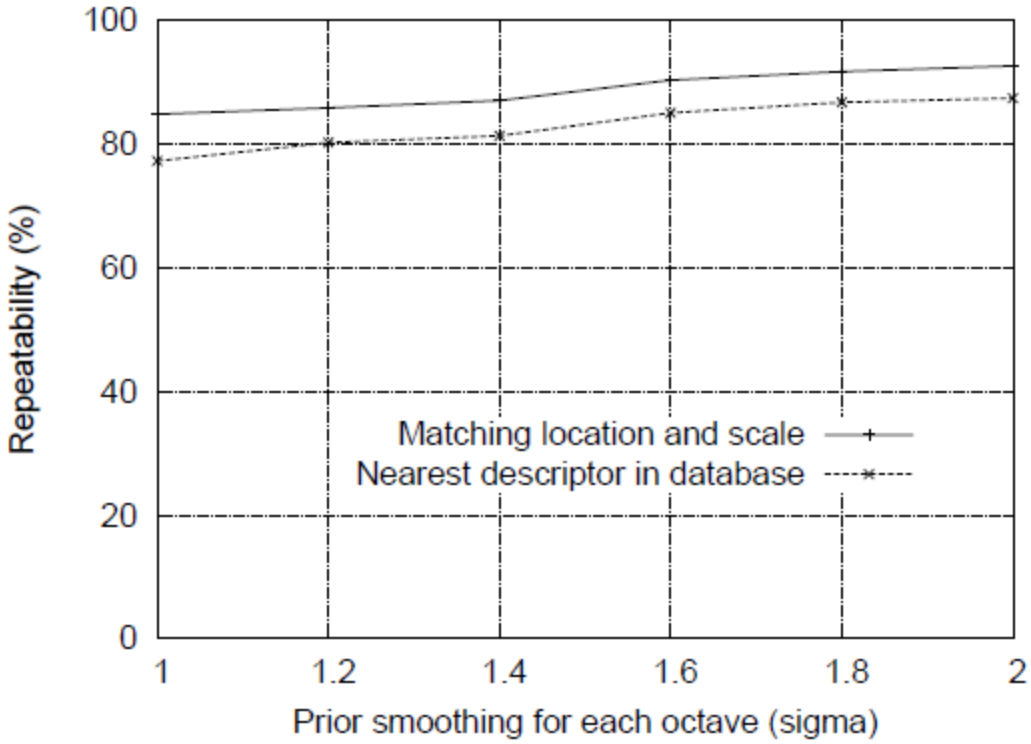



Figure 4: The top line in the graph shows the percent of keypoint locations that are repeatably detected in a transformed image as a function of the prior image smoothing for the first level of each octave. The lower line shows the percent of descriptors correctly matched against a large database.

$\sigma = 1.6$ (compromise bw performance and efficiency)

- Size of the descriptor:
  - *r* number of orientations in the histograms
  - *n* the width of the *nxn* array of orientation histograms
  - this gives a *rxnxn* descriptor vector
- Examples:
  - 8x2x2=32 …
  - 8x4x4=128,

# 3) Size of the descriptor

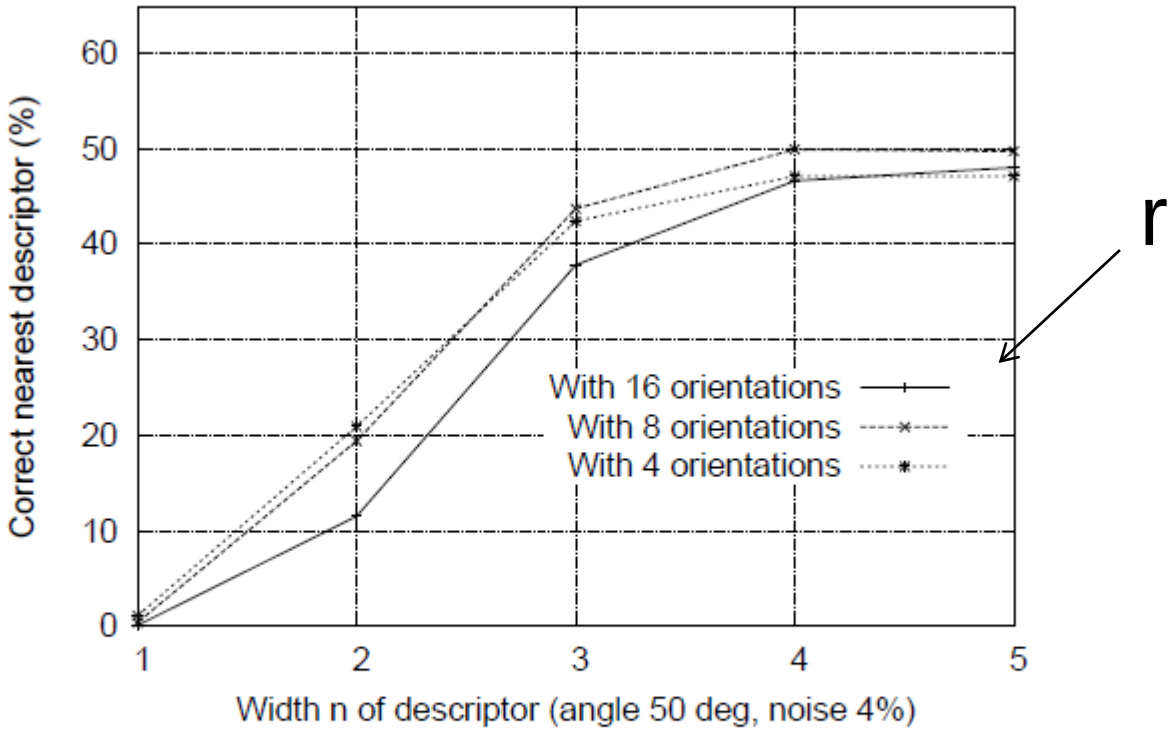



Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affi ne viewpoint change of 50 degrees and addition of 4% noise.

8 orientations 4x4 → 128 dimensions

# 4) Threshold

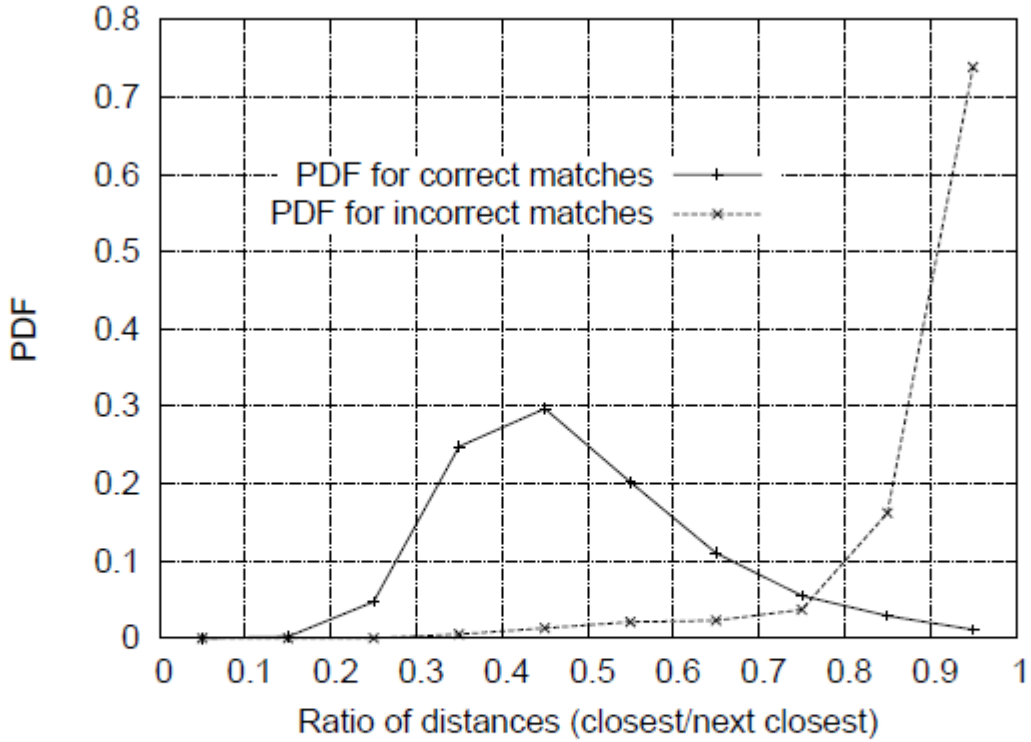
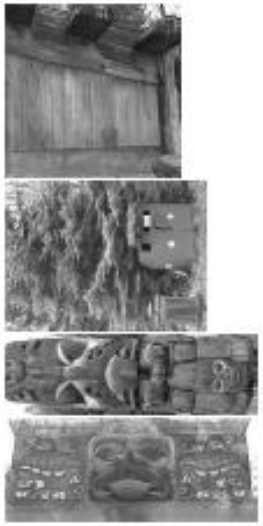


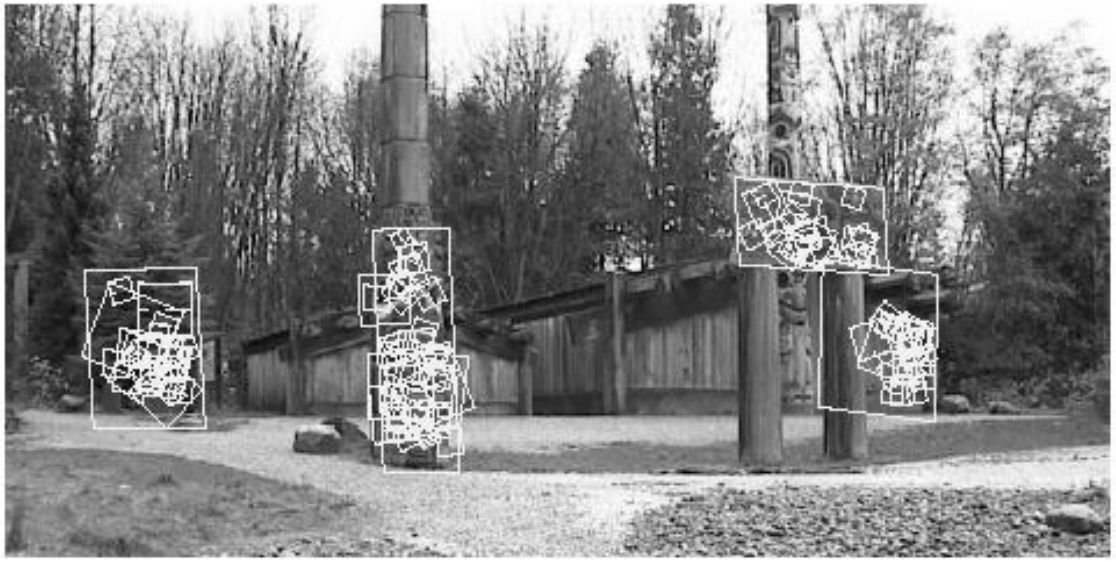

Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

0.8 is a threshold that eliminates 90% of false matches at the cost of discarding only 5% of good matches

# Results

# Location recognition





# 3D Object Recognition

# 3D Object Recognition

# Recognition under occlusion

# Improvements…

# Key point localization

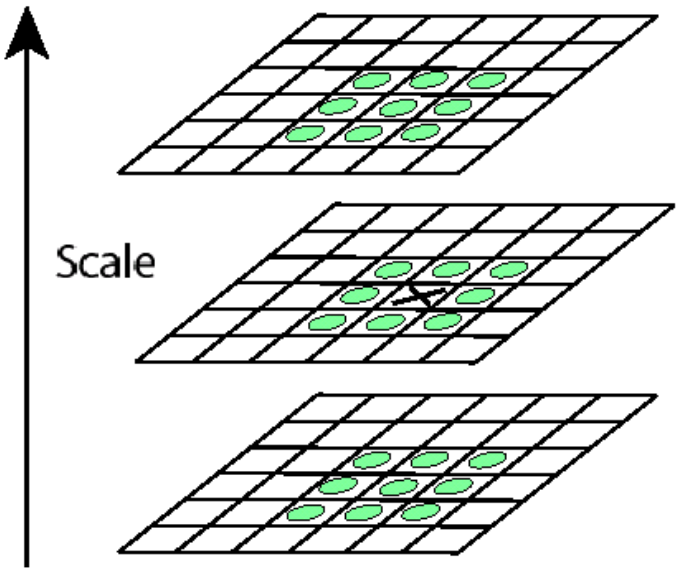
- Refine maxima and minima detection
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002)
- Taylor expansion around point:

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

$$\mathbf{x} = \left(x, y, \sigma\right)^T$$

- Offset of extremum (use finite differences for derivatives):

$$\mathbf{x} = -\mathbf{H}(\mathbf{x})^{-1} \nabla D\left(\mathbf{x}\right)$$




D and its derivatives are evaluated around the same point, x here is the offset

# Speeded-Up Robust Features (SURF)

- H.Bay, A.Ess, T.Tuytelaars, L. Van Gool, Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008
- Similar to SIFT but faster and more robust (according to the paper)
- Available in OpenCV

# SURF Keypoints

- Compute:

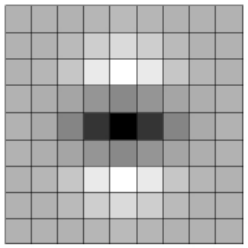
$$H(x,\sigma)=\begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}$$

$$L_{xx}=\frac{\partial^2}{\partial x^2} g * I,\ldots$$

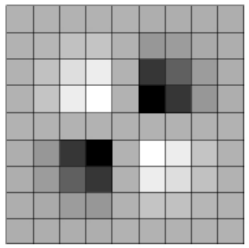
- …in practice compute an approximation
- Search maxima, solving:

$$\arg\max_{x,\sigma}\left\{\det(H(x,\sigma))\right\}$$

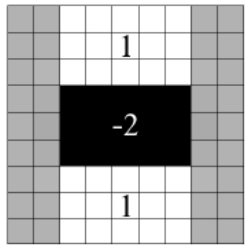
- Idea: gaussians are in practice discretized and cropped
- Compute an approximation of H using box filters

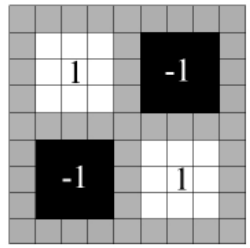


$$\frac{\partial^2 g}{\partial x^2}, \frac{\partial^2 g}{\partial y^2}$$

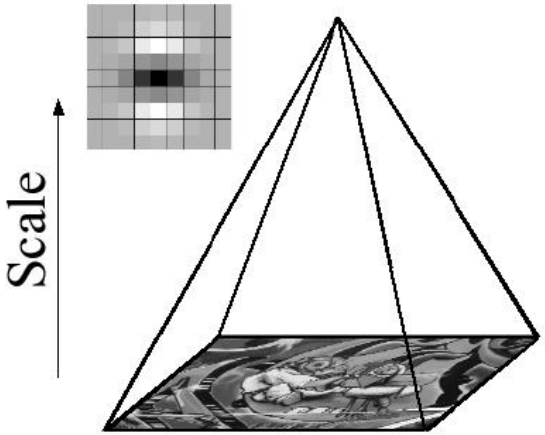


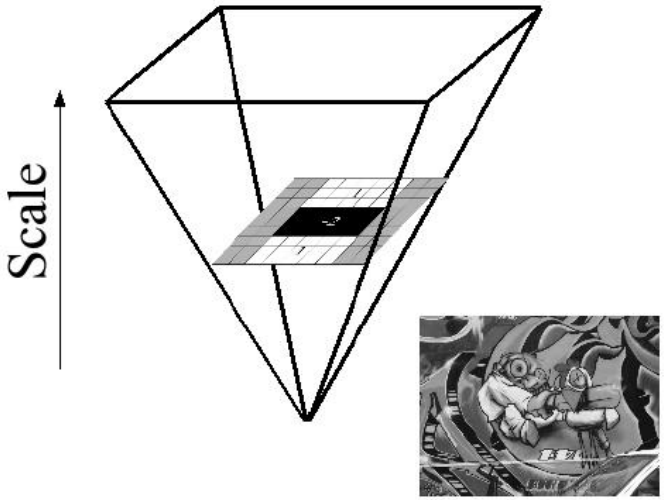
$$\frac{\partial^2 g}{\partial x \partial y}$$




- Box filters can be computed efficiently using the *integral image* (next)
- Computing these filters is fast irrespectively of the size, no need to perform pyramid decomposition and downsampling



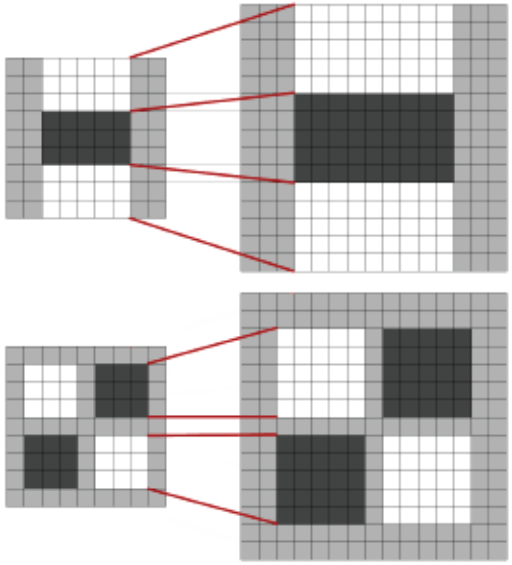



Fig. 5. Filters $D_{yy}$ (top) and $D_{xy}$ (bottom) for two successive scale levels ($9 \times 9$ and $15 \times 15$). The length of the dark lobe can only be increased by an even number of pixels in order to guarantee the presence of a central pixel (top).

Instead of down-sampling the image, up-scale the filters

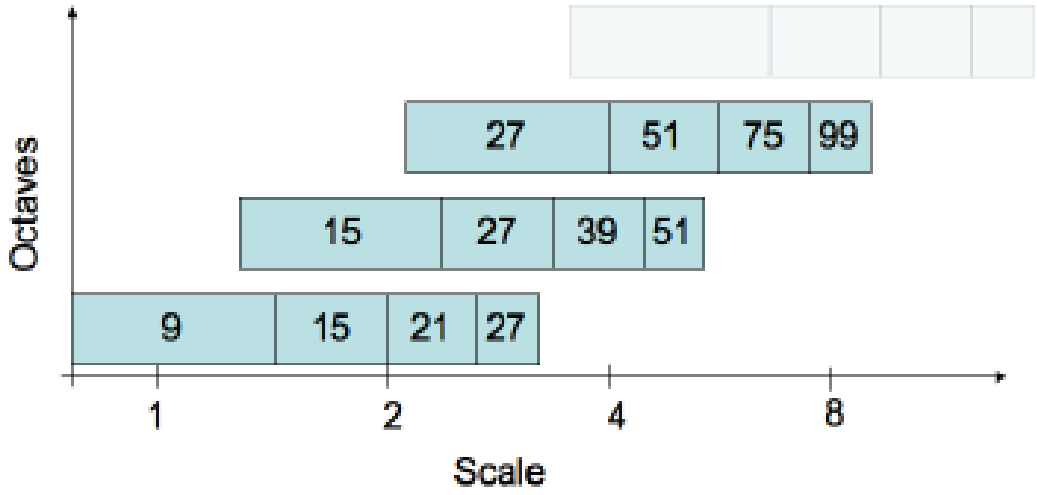


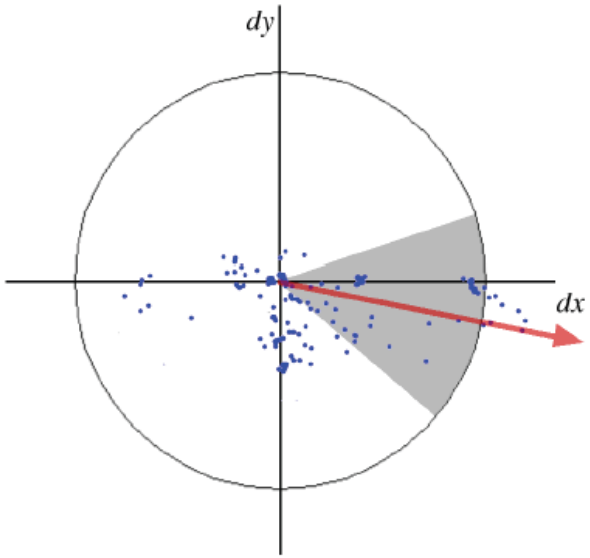

Fig. 6. Graphical representation of the filter side lengths for three different octaves. The logarithmic horizontal axis represents the scales. Note that the octaves are overlapping in order to cover all possible scales seamlessly.

# Descriptor

- Similar to SIFT, describe distribution of intensities
- Compute orientations using squared filters (called Haar wavelets)
- The dimension of the descriptor is 64, to speed up matching

Fig. 9. Haar wavelet filters to compute the responses in $x$ (left) and $y$ direction (right). The dark parts have the weight $-1$ and the light parts $+1$.

- From scale s of the keypoint
- Firstly compute descriptor principle orientation, compute response to Haar wavelets in neighborood of 6s points (size of filters 4s points)
- Weight responses using a Gaussian sigma=2s
- The response each filter dx,dy is represented in a x,y plane




slide a window of size pi/3

compute the sum of dx and dy within the windows

take the maximum

Fig. 10. Orientation assignment: A sliding orientation window of size $\frac{\pi}{3}$ detects the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood around the interest point.

# Descriptor

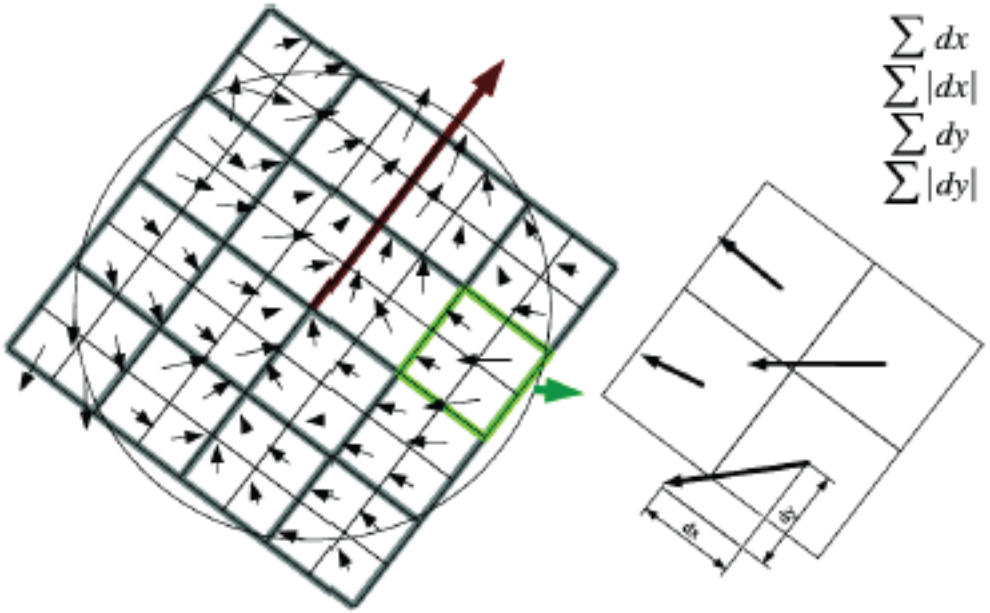



Fig. 12. To build the descriptor, an oriented quadratic grid with $4 \times 4$ square sub-regions is laid over the interest point (left). For each square, the wavelet responses are computed. The $2 \times 2$ sub-divisions of each square correspond to the actual fields of the descriptor. These are the sums $dx$, $|dx|$, $dy$, and $|dy|$, computed relatively to the orientation of the grid (right).

Size of the decriptor is 4x4x4=64

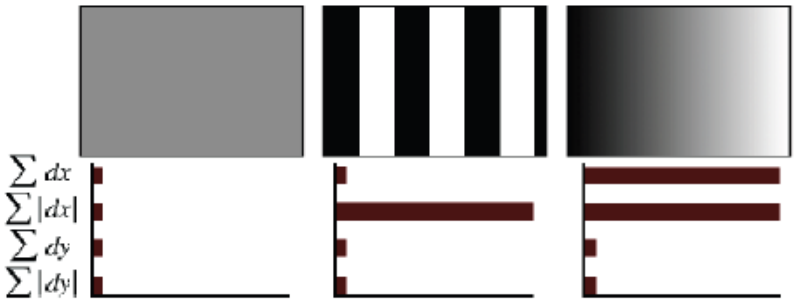



Fig. 13. The descriptor entries of a sub-region represent the nature of the underlying intensity pattern. Left: In case of a homogeneous region, all values are relatively low. Middle: In presence of frequencies in $x$ direction, the value of $\sum |d_x|$ is high, but all others remain low. If the intensity is gradually increasing in $x$ direction, both values $\sum d_x$ and $\sum |d_x|$ are high.

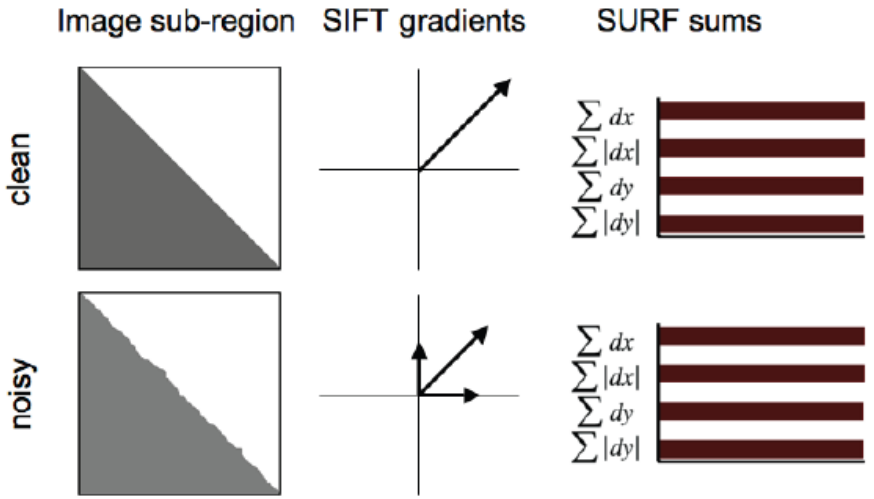
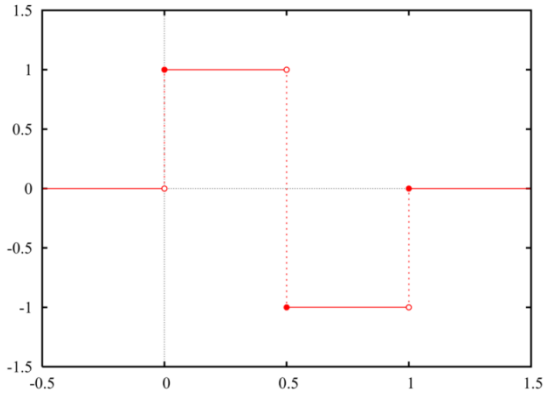


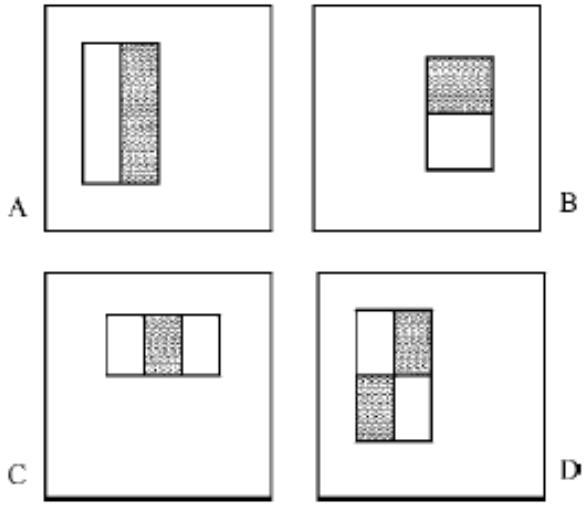

Fig. 14. Due to the global integration of SURF's descriptor, it stays more robust to various image perturbations than the more locally operating SIFT descriptor.

# Integral Image

- The *integral image* representation allows for fast computation of rectangular two-dimensional image features
- Originally proposed by Viola & Jones (2001) for face detection
- Used in the *SURF* algorithm to compute box filters at different scales

# Examples of features



# Integral Image

- The integral image at location x,y contains the sum of all pixels above and left of x,y:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

- Computed once, from a single pass:

$$s(x,y) = s(x,y-1) + i(x,y)$$

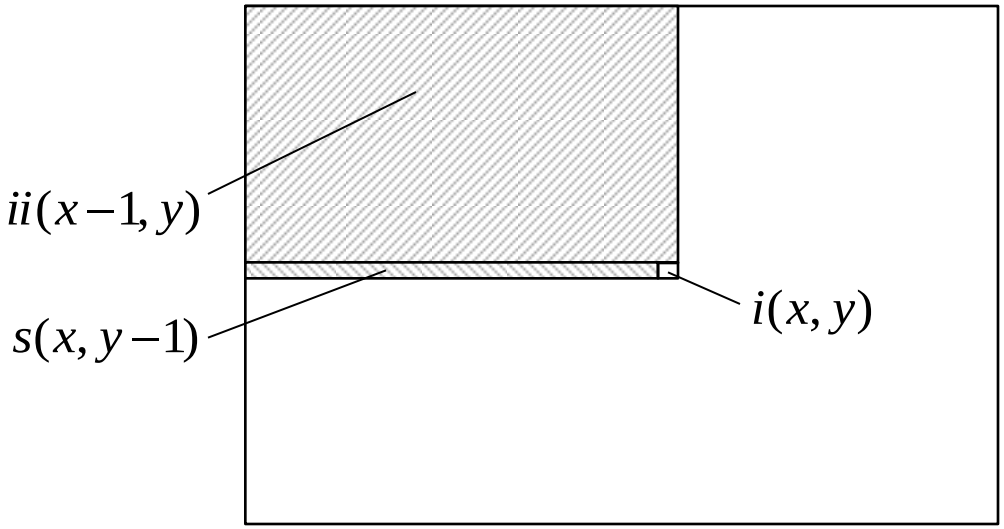
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

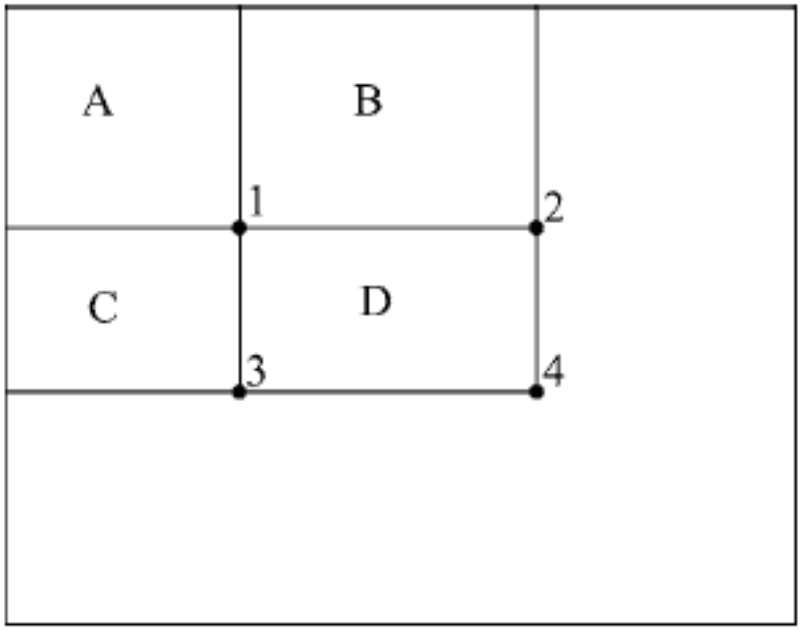
$$ii(-1,y) = 0$$

$$s(x,-1) = 0$$

at start

starting each row






The sum of the pixels within rectangle D can be computed with four array references:

$$D = ii(4) + ii(1) - (ii(2) + ii(3)) =$$

$$(A + B + C + D) + A - (B + A + C + A)$$