VIT-AP UNIVERSITY

# Computer Vision

(Course Code: 4047)

## Module-2:Lecture-5: Speeded Up Robust Features (SURF)

Gundimeda Venugopal, Professor of Practice, SCOPE

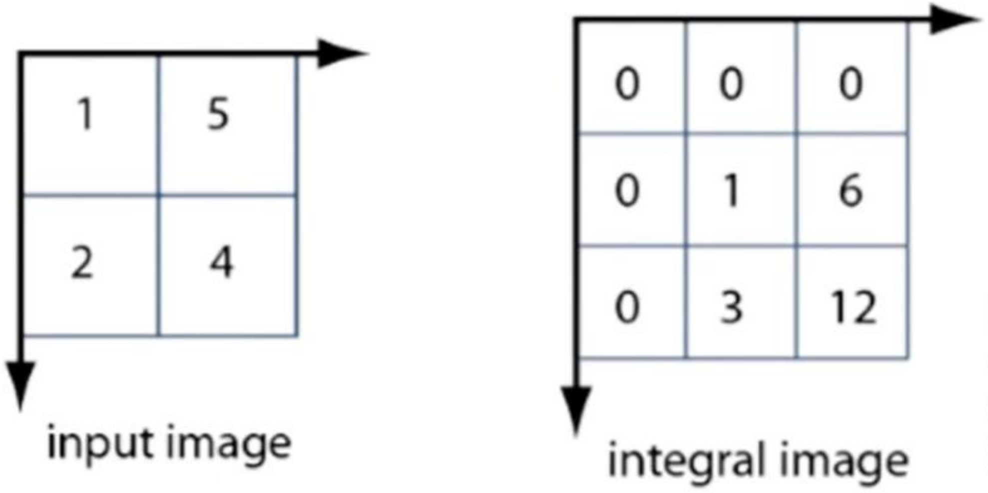
# Motivation

- Fast Interest Point **Detection**
- Distinctive interest point **Description**
- Speed-up Descriptor **Matching**
- Invariant to common Image transformations:
    - Image Rotation
    - Scale changes
    - Illumination Change
    - Scale Change in View point

# Integral Images

# Integral Image - Methodology

## Integral Images!




### Integral Image Formula

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

Integral image at location (x,y) contains the sum of the pixels above and to the left of (x,y), inclusive

# Integral Image

❖ A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

| | | | | | |
|---|---|---|---|---|---|
| 98 | 110 | 121 | 125 | 122 | 129 |
| 99 | 110 | 120 | 116 | 116 | 129 |
| 97 | 109 | 124 | 111 | 123 | 134 |
| 98 | 112 | 132 | 108 | 123 | 133 |
| 97 | 113 | 147 | 108 | 125 | 142 |
| 95 | 111 | 168 | 122 | 130 | 137 |
| 96 | 104 | 172 | 130 | 126 | 130 |

Image $I$

| | | | | | |
|---|---|---|---|---|---|
| 98 | 208 | 329 | 454 | 576 | 705 |
| 197 | 417 | 658 | 899 | 1137 | 1395 |
| 294 | 623 | 988 | 1340 | 1701 | 2093 |
| 392 | 833 | 1330 | 1790 | 2274 | 2799 |
| 489 | 1043 | 1687 | 2255 | 2864 | 3531 |
| 584 | 1249 | 2061 | 2751 | 3490 | 4294 |
| 680 | 1449 | 2433 | 3253 | 4118 | 5052 |

Integral Image $II$

| | | | | | |
|---|---|---|---|---|---|
| 98 | 110 | 121 | 125 | 122 | 129 |
| 99 | 110 | 120 | 116 | 116 | 129 |
| 97 | 109 | 124 | 111 | 123 | 134 |
| 98 | 112 | 132 | 108 | 123 | 133 |
| 97 | 113 | 147 | 108 | 125 | 142 |
| 95 | 111 | 168 | 122 | 130 | 137 |
| 96 | 104 | 172 | 130 | 126 | 130 |

Image $I$

| | | | | | |
|---|---|---|---|---|---|
| 98 | 208 | 329 | 454 | 576 | 705 |
| 197 | 417 | 658 | 899 | 1137 | 1395 |
| 294 | 623 | 988 | 1340 | 1701 | 2093 |
| 392 | 833 | 1330 | 1790 | 2274 | 2799 |
| 489 | 1043 | 1687 | 2255 | 2864 | 3531 |
| 584 | 1249 | 2061 | 2751 | 3490 | 4294 |
| 680 | 1449 | 2433 | 3253 | 4118 | 5052 |

Integral Image $II$

# Summation within a Rectangle

Fast summations of arbitrary rectangles using integral images

| | | | | | |
|---|---|---|---|---|---|
| 98 | 110 | 121 | 125 | 122 | 129 |
| 99 | 110 | 120 | 116 | 116 | 129 |
| 97 | 109 | 124 | 111 | 123 | 134 |
| 98 | 112 | 132 | 108 | 123 | 133 |
| 97 | 113 | 147 | 108 | 125 | 142 |
| 95 | 111 | 168 | 122 | 130 | 137 |
| 96 | 104 | 172 | 130 | 126 | 130 |

Image $I$

| | | | | | |
|---|---|---|---|---|---|
| 98 | 208 | 329 | 454 | 576 | 705 |
| 197 | 417 | 658 | 899 | 1137 | 1395 |
| 294 | 623 | 988 | 1340 | 1701 | 2093 |
| 392 | 833 | 1330 | 1790 | 2274 | 2799 |
| 489 | 1043 | 1687 | 2255 | 2864 | 3531 |
| 584 | 1249 | 2061 | 2751 | 3490 | 4294 |
| 680 | 1449 | 2433 | 3253 | 4118 | 5052 |

Integral Image $II$

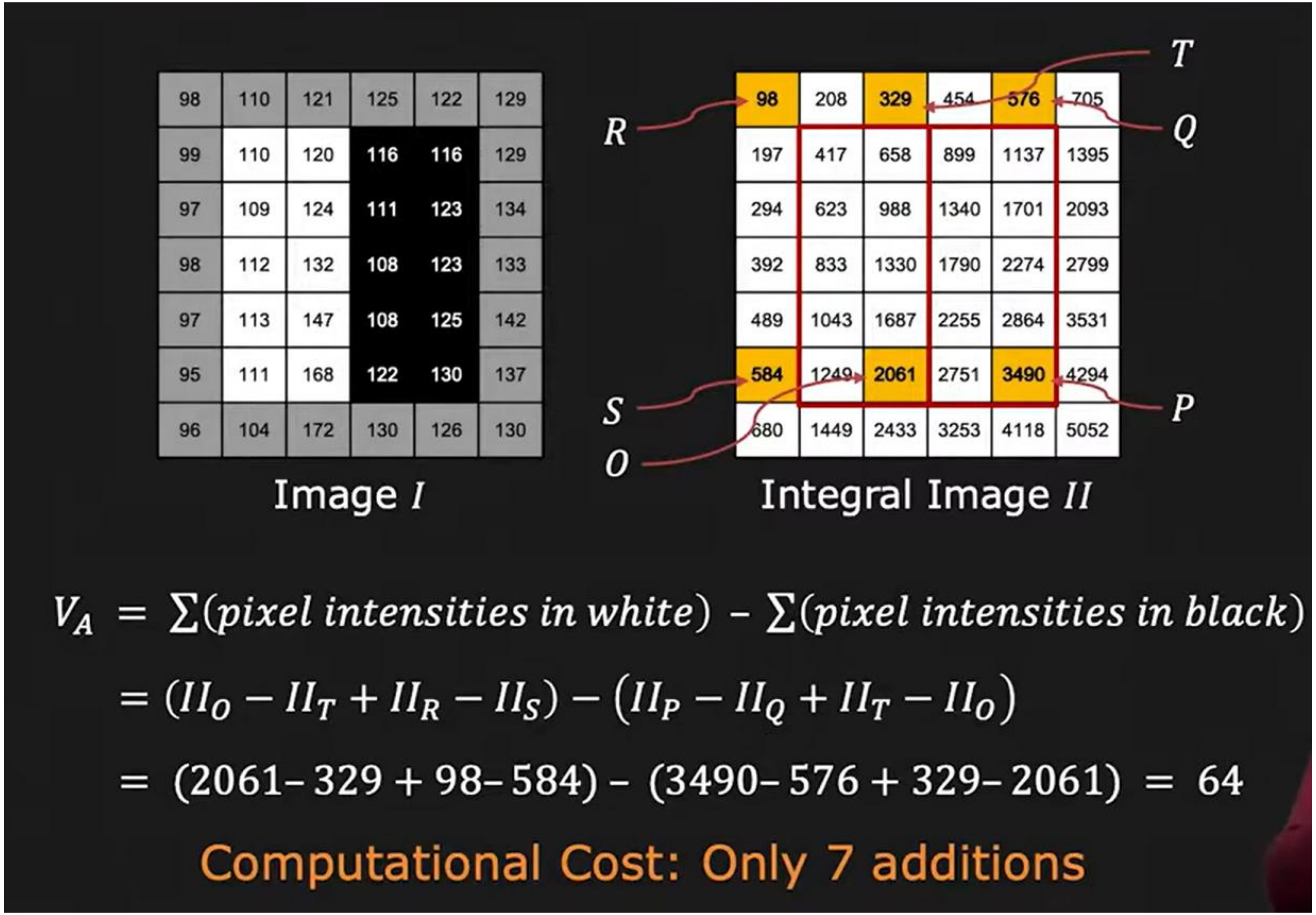
R Q S P

$$Sum = II_P - II_Q - II_S + II_R$$

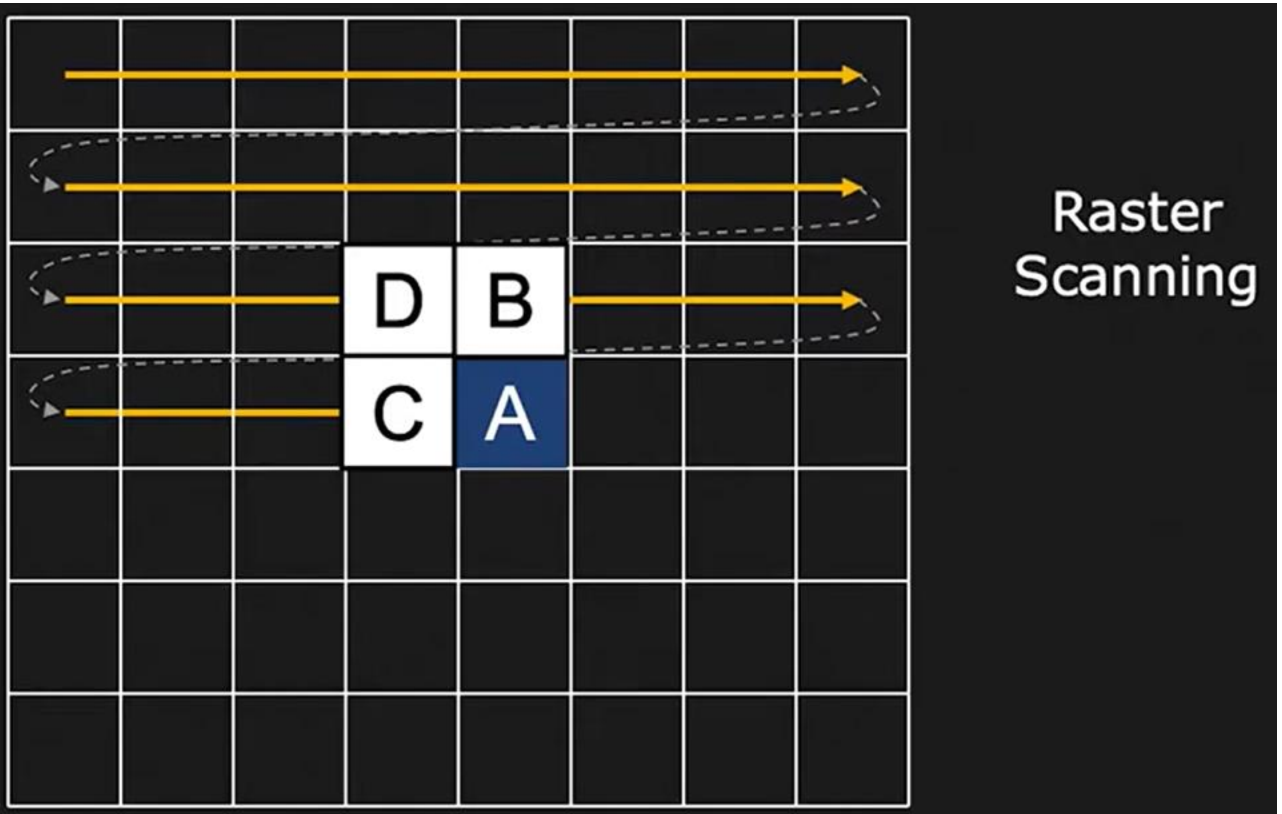
$$= 3490 - 1137 - 1249 + 417 = 1521$$

Computational Cost: Only 3 additions

# Haar Response using Integral Image



# Computing the Integral Image

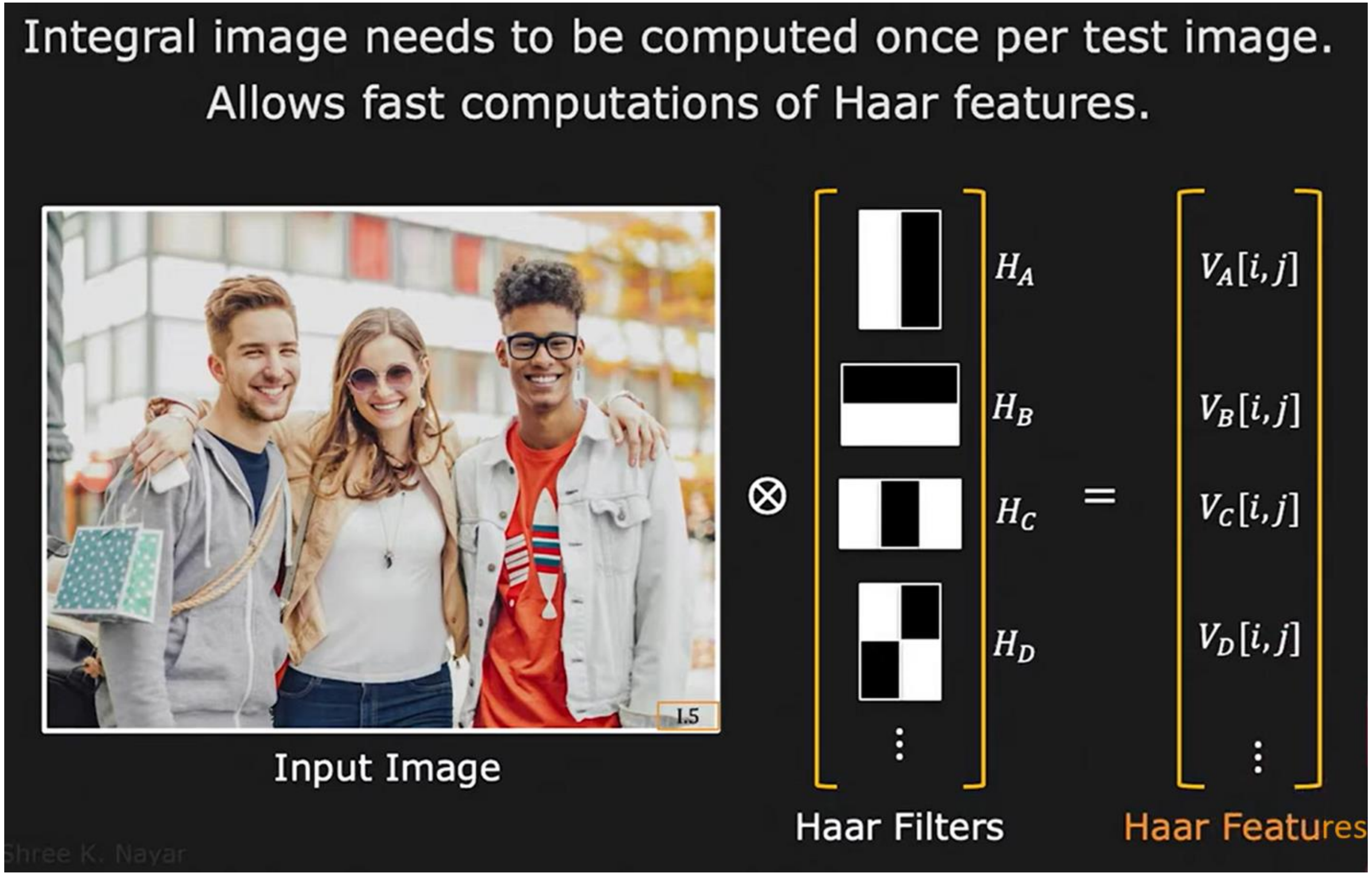



Let $I_A$ and $II_A$ be the values of Image and Integral Image, respectively, at pixel $A$.

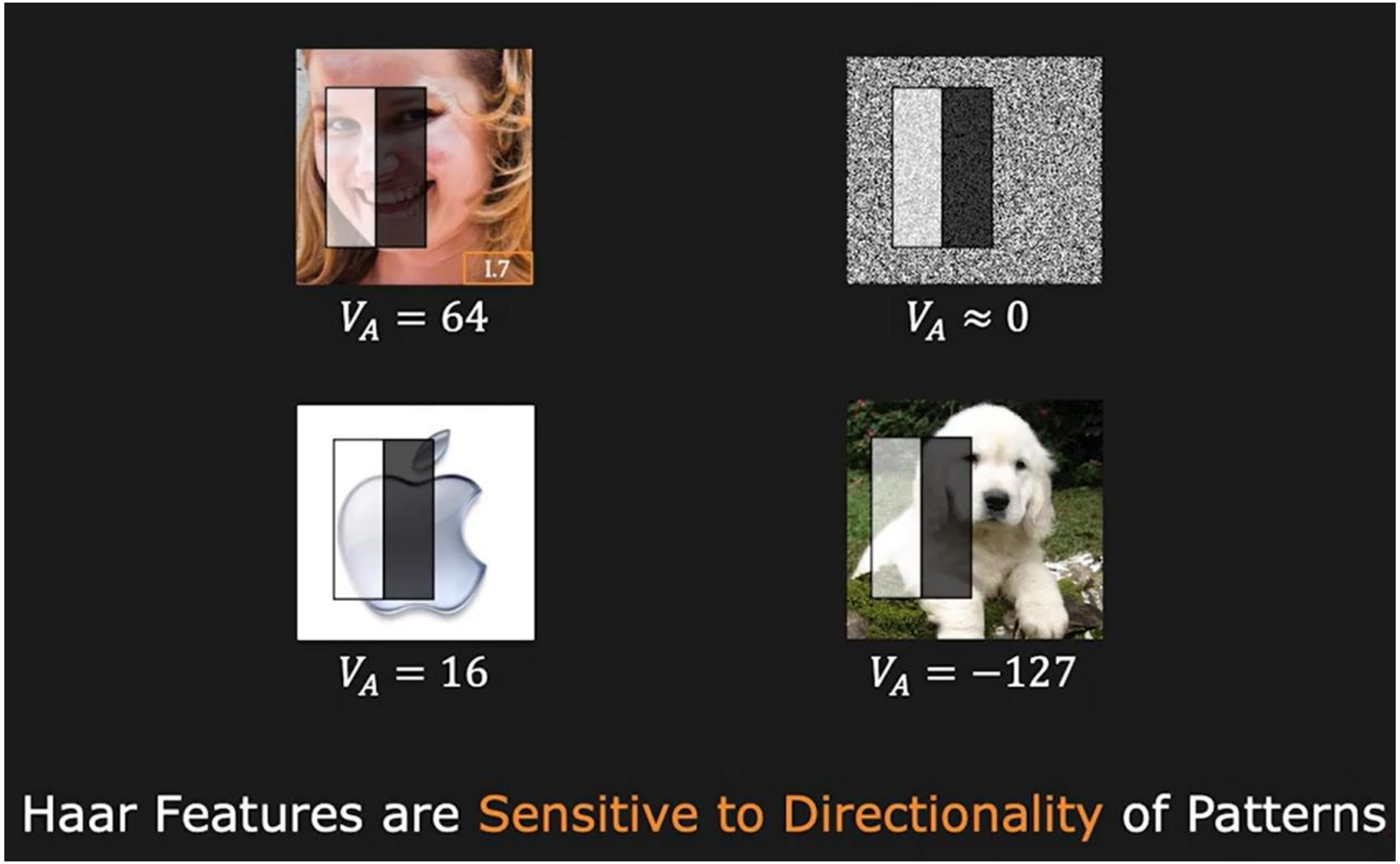
$$\boxed{II_A = II_B + II_C - II_D + I_A}$$

# Haar Features using Integral Images

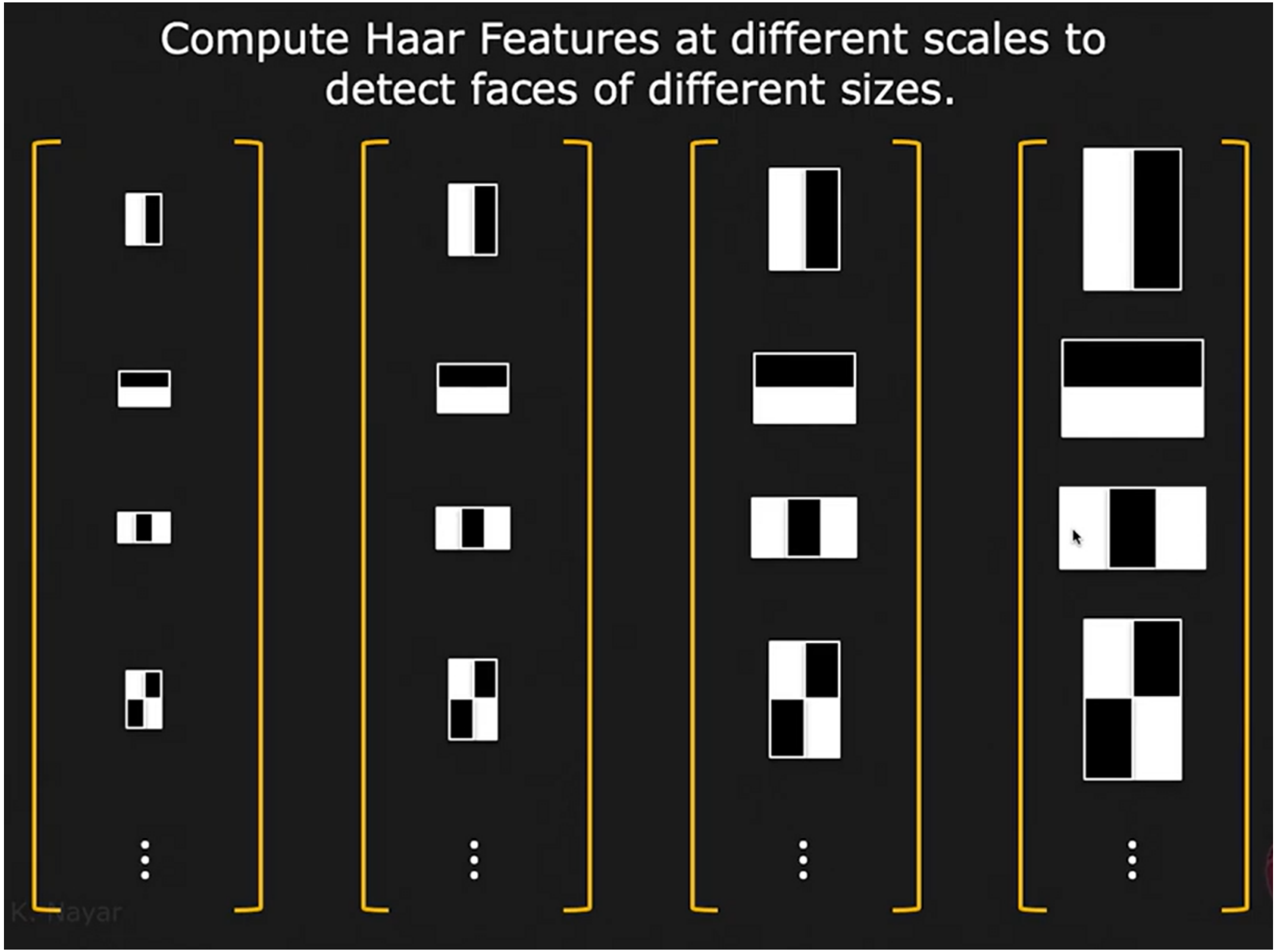
Haar feature Vector at a point in Image. For feature comparison, you may have to use different vectors for different scale.



# Discriminative Ability of Haar Feature



# Detecting Faces of Different Size



# Computing a Haar Feature

$\otimes$  $H_A$

White = 1, Black = -1

Response to Filter $H_A$ at location $(i,j)$:

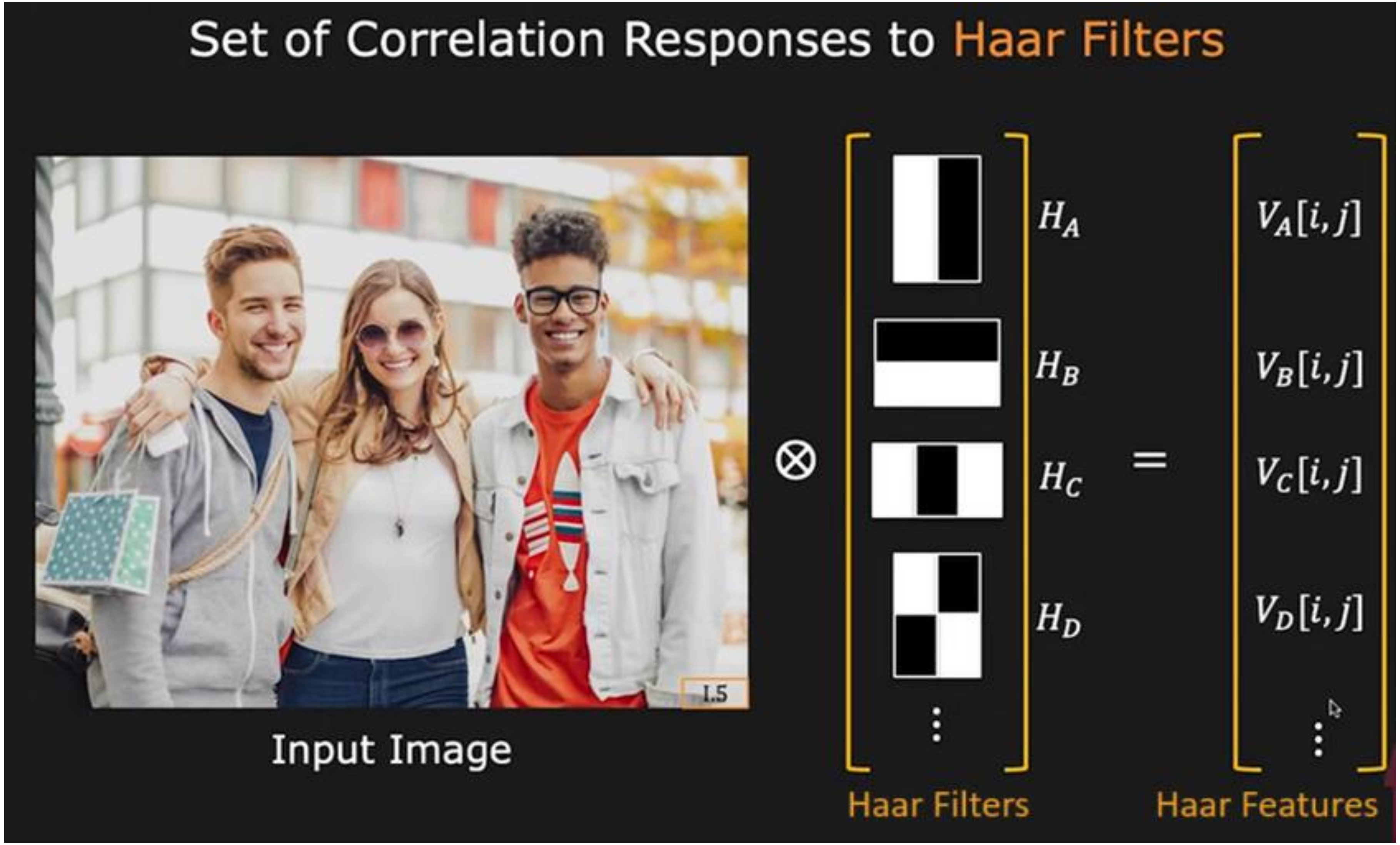
$$V_A[i,j] = \sum_m \sum_n I[m-i, n-j] H_A[m,n]$$

$$V_A[i,j] = \sum(\text{pixel intensities in white area}) - \sum(\text{pixels intensities in black area})$$

# Haar Feature: Computation Cost


Computation cost = ( N X M – 1) additions per pixel per filter per scale

# Haar Feature Vector (per pixel per scale)



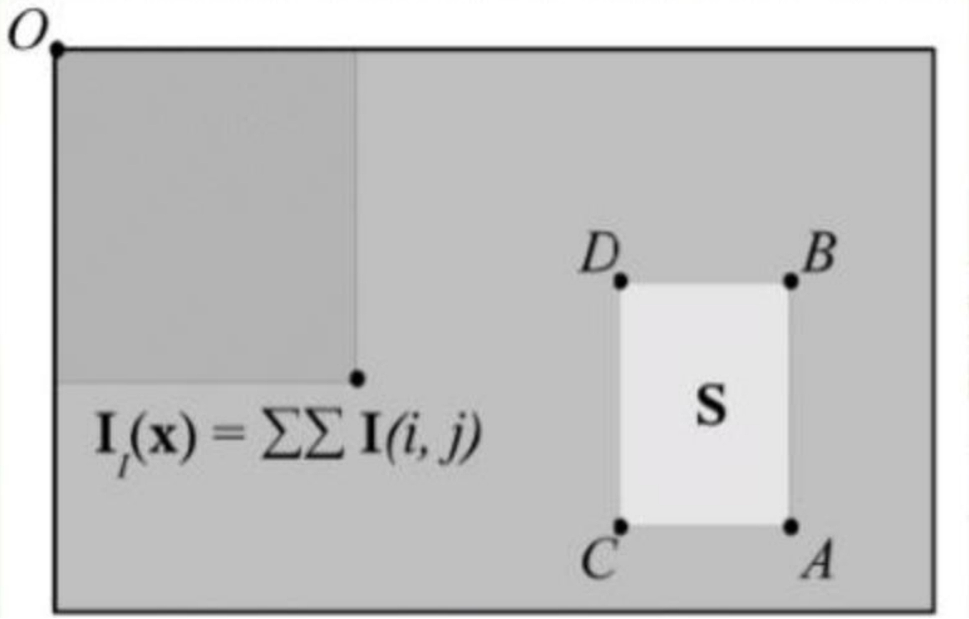
# SURF Methodology

- ❖ SURF (Speeded Up Robust Features) is a well-known local feature detector and descriptor algorithm and can be used for several task such as image registration, object recognition, object tracking, etc.
- ❖SURF it presented by H. Bay et.al, at 2006 European conference on computer vision.
- ❖SURF is several time faster than other algorithms (e.g. SIFT) and its more robust against such image transformations issues like illumination change, scale space, rotation and partial occlusion.
- ❖General Overview of the Approach:

| Detector | Descriptor |
|---|---|
| Based on the Hessian matrix | Describes a distribution of Haar-wavelet responses within the interest point neighbourhood |
| Uses a further approximated DoG | Only uses 64 dimensions |
| Uses Integral Image | Introduced a new indexing step that is based on the sign of the laplacian |

# SURF Methodology

- Using integral images for major speed up
  - **Integral Image (summed area tables)** is an intermediate representation for the image and contains the sum of gray scale pixel values of image
  - Second order derivative and Haar-wavelet response




$$\boxed{\mathbf{S} = A - B - C + D}$$

Cost *four* additions operation only

# Keypoint Detection

## Keypoint detection

Gaussians are considered to be ideal for scale space analysis [Lindeberg, T., Bretzner, L(2003)].

The continuous gaussian functions are being cropped and discretized.

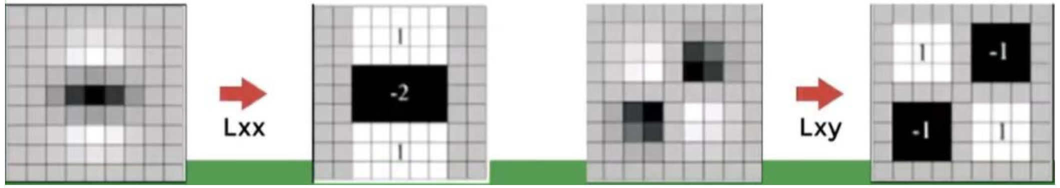
Aliasing still occurs when the filtered images are sub-sampled.

Gaussians are overrated!!

## Keypoint detection

The author's of this paper pushed approximation to another level!!

They proposed box filters for gaussian second order derivatives.




## Keypoint detection

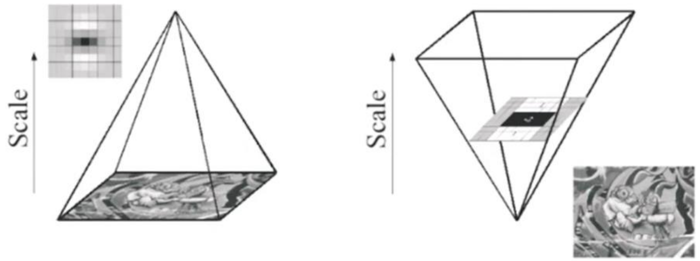
The performance of approximated box filter is comparable to the one using the discretized and cropped Gaussians.

Earlier for finding gaussian derivative response we had to convolve the input image with the gaussian derivative, thus the computational complexity for this process used to be O(N^(2)).

Compute the sum of the pixel intesnsities in the rectangular area and multiply with respective coefficient and then compute the total sum, all this in O(1) time.

## Key-point detection using box filters

The scale space is analysed by upscaling the filter size.




The filter of size 27 x 27 corresponds to σ = 1.2 x3 = 3.6, and as given earlier filter of 9 x 9 corresponds to σ = 1.2 (intial scale)

# Feature Point Detection

- SURF uses The first step in SURF algorithm is to create an Integral image to speeded up the calculation and decreasing the time complexity in efficient manner. ) as a base to locate the interest points in an image, in which the blob structure that has maximum values than neighbors selected as interest points.

This can be calculated directly according to (1), but it is not computationally recommended.

Thus, it can be calculated by the approximation given in (2), where the Hessian matrix is accomplished the function space x = (x, y) and σ scale.

- Surf allows to find interest points in different scales. The second derivative of gaussian kernel is approximated by the box filters. (see next slide).
- The idea of scale-space is implemented by changing the box filter size (e.g., 9 × 9, 15 × 15, 21 × 21 and 27 × 27) of the kernel used in the convolutional process.
- To locate and map the interest points in both space and scale, the 3D Non Maximum Suppression (NMS) technique is applied to a (3 × 3 × 3) neighborhood

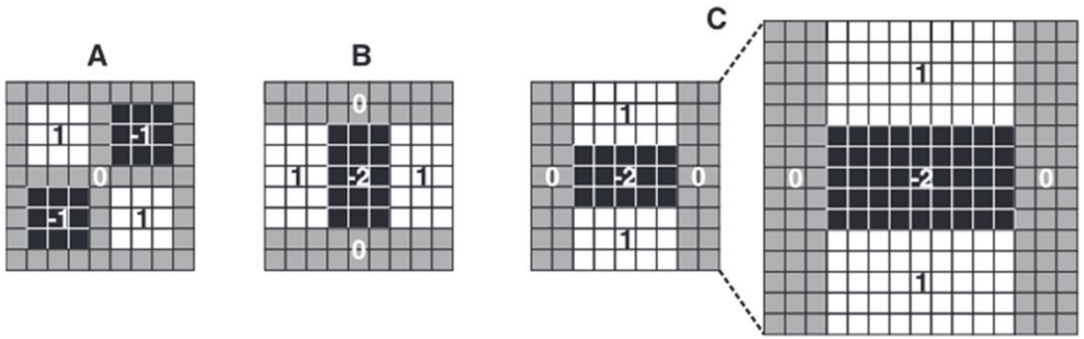



**Figure Filters used by SURF.** **(A)**, **(B)**, and **(C)** Approximation of Gaussian second order partial derivatives using box filters: $D_{xy}$ in xy-direction, $D_{xx}$ in x-direction and y-direction in $D_{yy}$, respectively, with increased kernel size from (9 × 9) to (15 × 15) in **(C)**.

For the given image I (M×N) pixels (points), the point P (x, y) in the image I can be an interest point if the H (p, σ) has maximum value, the Hessian matrix of H (p, σ) at point (P) and scale (σ) can be calculated as follows:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \quad (1)$$

$$H(\mathbf{x},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{yx}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix} \quad (2)$$

and

$$L_{xx}(\mathbf{x},\sigma) = I(\mathbf{x}) * \frac{\partial^2}{\partial x^2} g(\sigma) \quad (3)$$

$L_{xx}(\mathbf{x},\sigma)$ in (3) is the Convolution of the second order Gaussian derivative with the image at point I(x) with scale value ($\sigma$) = 1.2

similarly to $L_{yy}$ and $L_{xy}$.

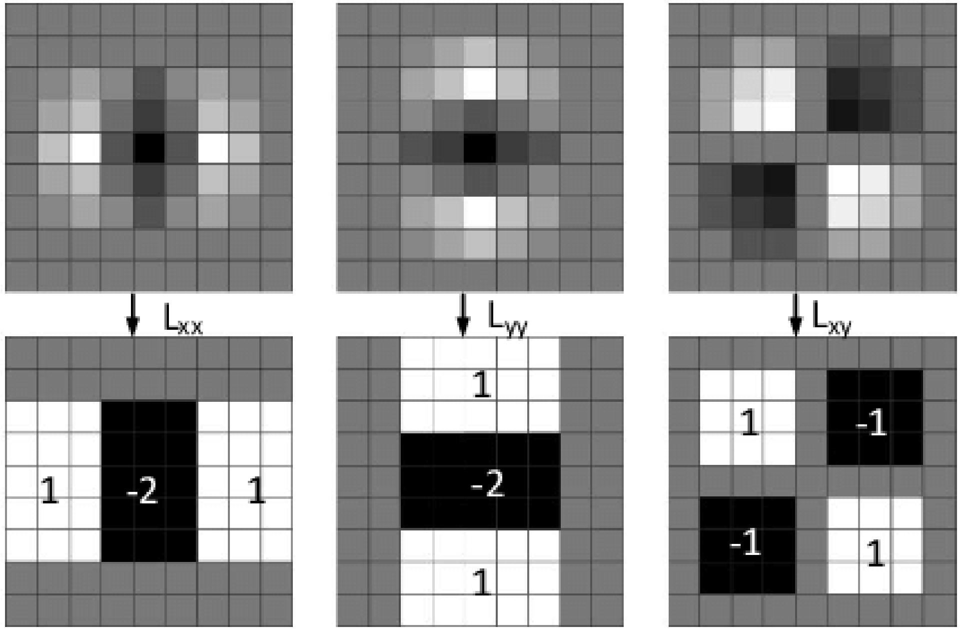
Determinant of the Hessian matrix can be calculated by (4), and w is a weight used to balance the equation. That is, the blob response at location I(x).

$$\det(\mathcal{H}_{\text{approx}}) = D_{xx} D_{yy} - (w D_{xy})^2 \quad (4)$$

The value of w is:

$$w = \frac{|L_{xy}(1.2)|_F |D_{yy}(9)|_F}{|L_{yy}(1.2)|_F |D_{xy}(9)|_F} = 0.912... \simeq 0.9$$

# Laplacian of Gaussian approx. with Box filters + Scale Space



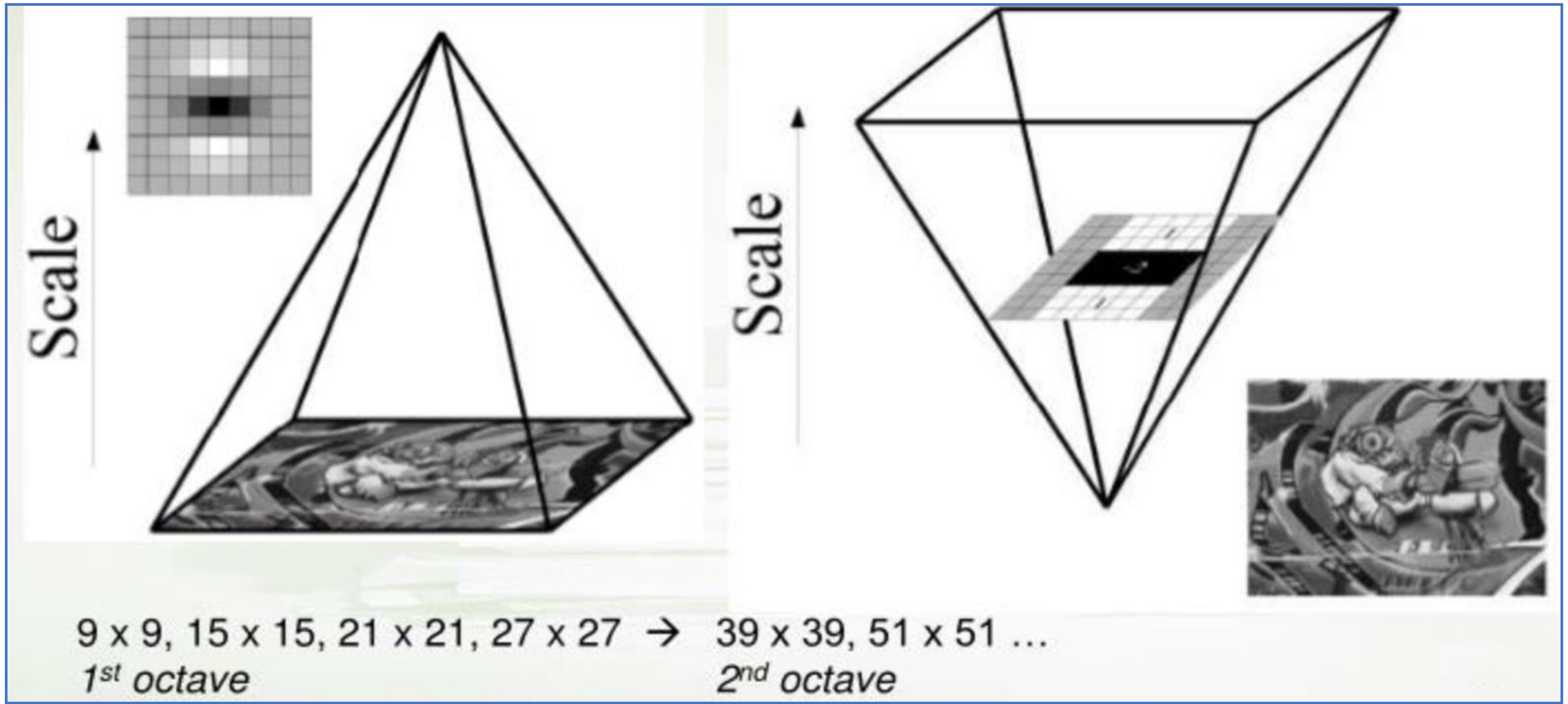

- Laplacian of Gaussian approximation with box filters SURF keeps the size of the input image constant and upscales the size of box filters to construct the image pyramid.
- The initial scale layer of the image pyramid is the convolution of the original image and the 99 box filters which correspond to scale.
- Subsequent layers are obtained by filtering the original image with the box filters (of different sizes)

# Scale Analysis with Constant image size

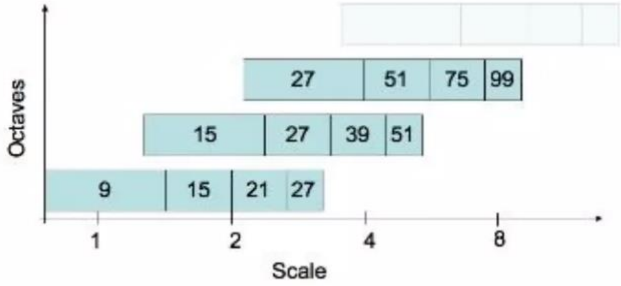
- In SIFT, images are repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve a higher level of the pyramid.

- Alternatively, we can use filters of larger size on the original image.

- Due to using integral images, filters of any size can be applied at exactly the same speed!




The scale space is divided into octaves. An octave represents a series of increasing filter response maps.




**It is selected as the interest point only if it is larger than all of the neighbors.**
**For each new octave, the filter size increase is doubled.**

# Scale-space Representation

- Scale spaces are usually implemented as image pyramids. The images are repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve a higher level of the pyramid. (**SIFT approach**)

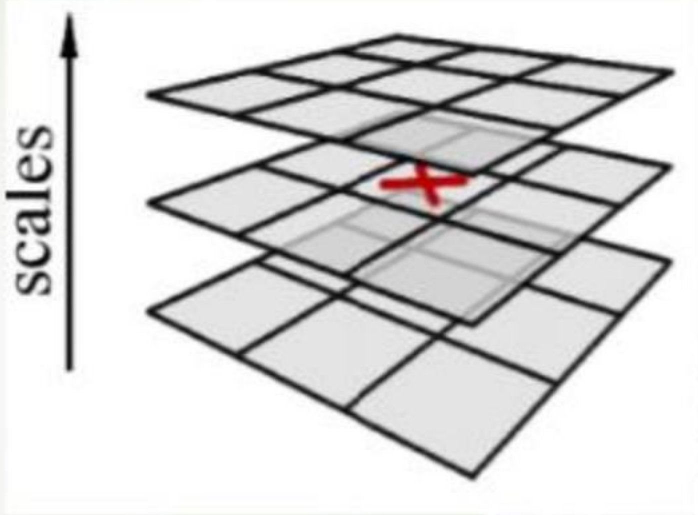
- Due to the use of box filters and integral images, surf does not have to iteratively apply the same filter to the output of a previously filtered layer but instead **can apply such filters of any size at exactly the same speed directly on the original image, and even in parallel**.

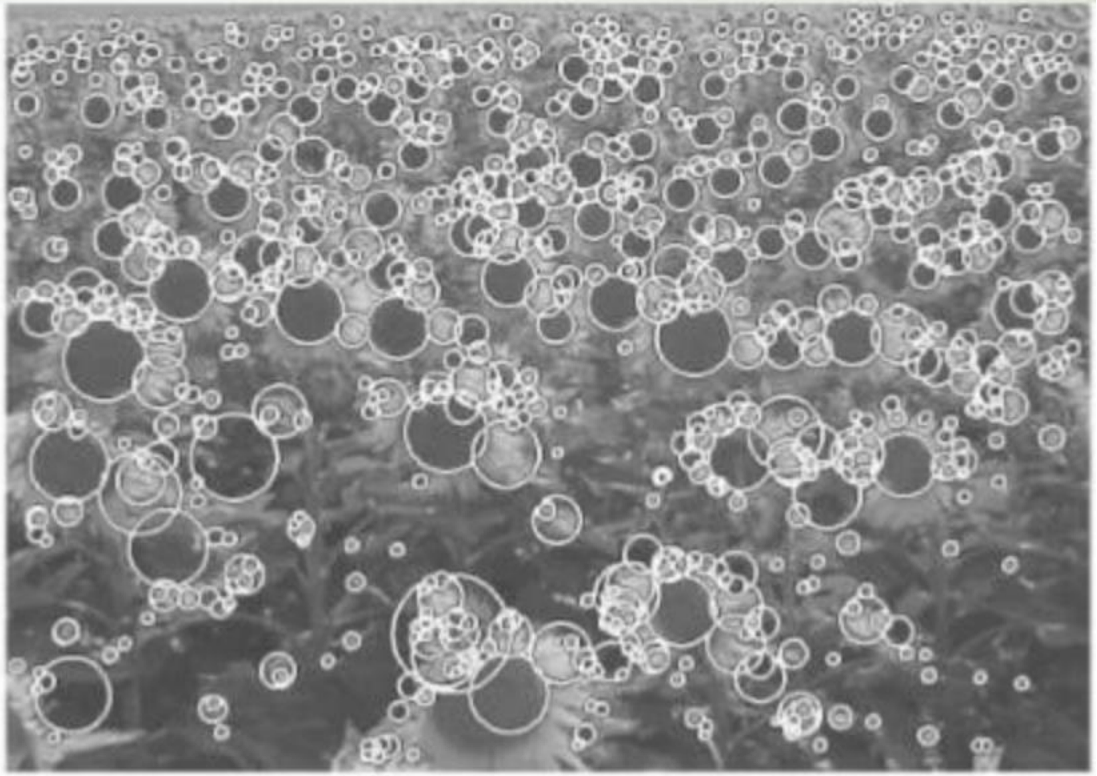
- The scale space is analyzed by **up-scaling the filter size(9×9 → 15×15 → 21×21 → 27×27, etc.)** rather than iteratively reducing the image size.

- For each new octave, **the filter size increase is doubled simultaneously the sampling intervals for the extraction of the interest points(σ) can be doubled** as well which allow the up-scaling of the filter at constant cost.

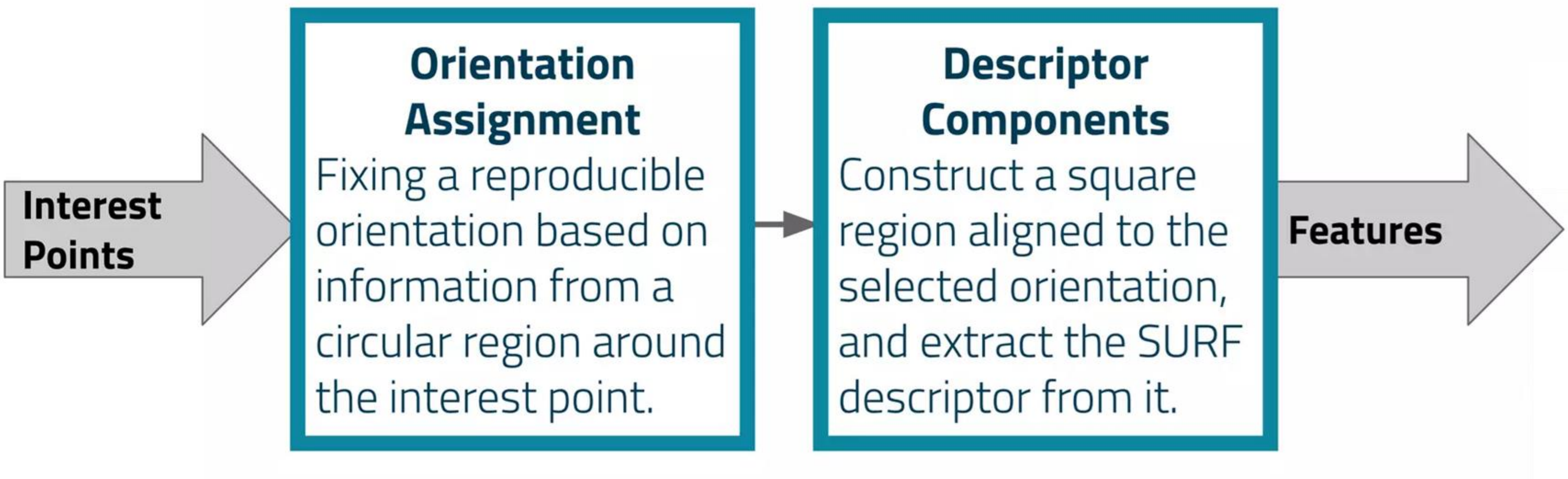
# Feature Point Detection

- Non-maximum suppression and interpolation
  - Blob-like feature detector

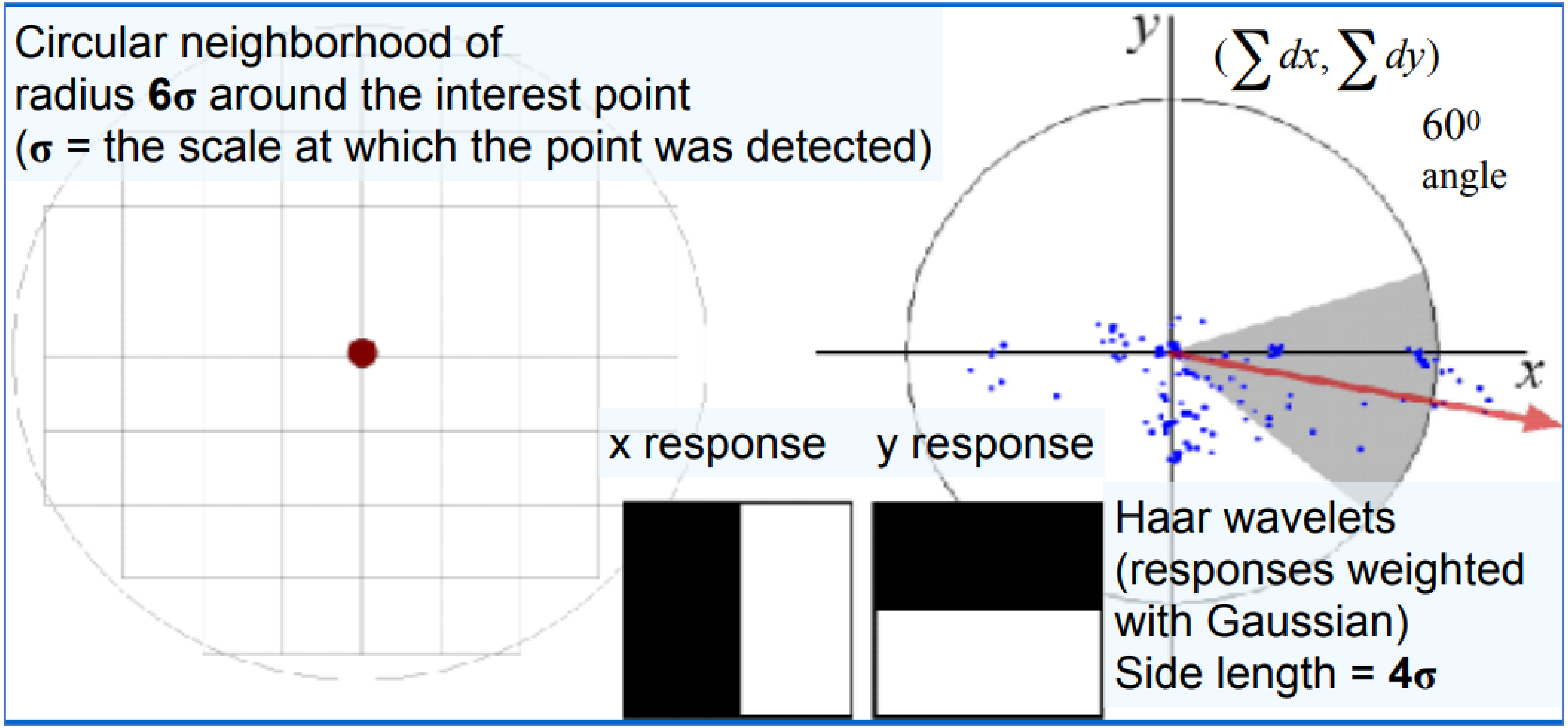


# SURF Descriptor

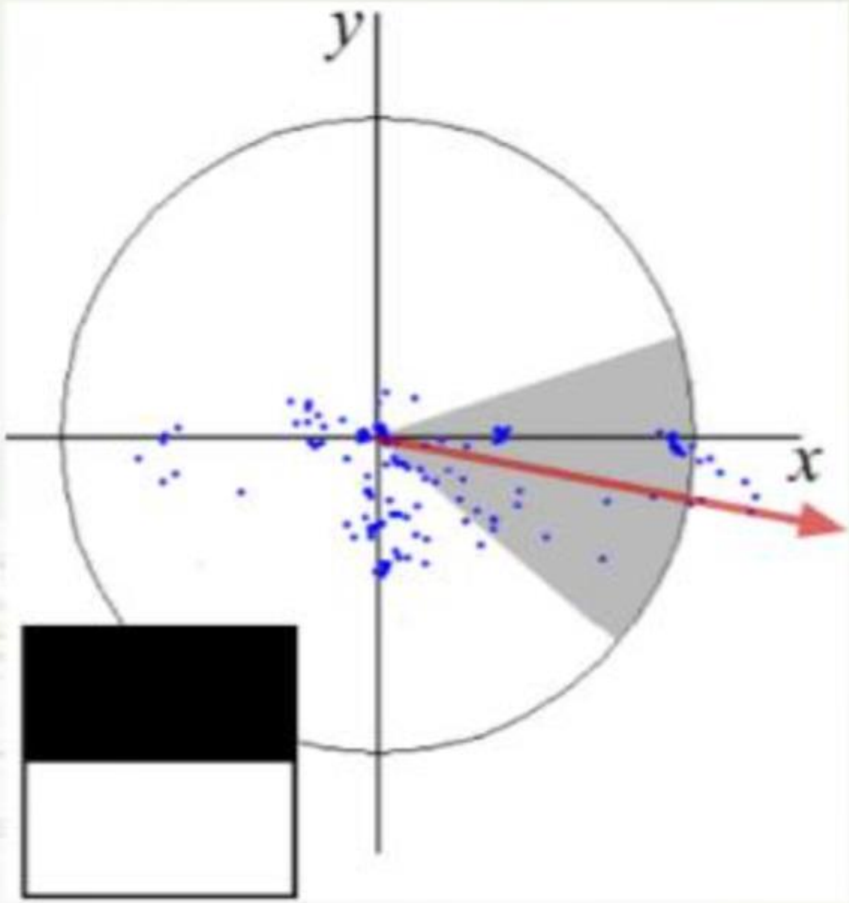


# Orientation Assignment



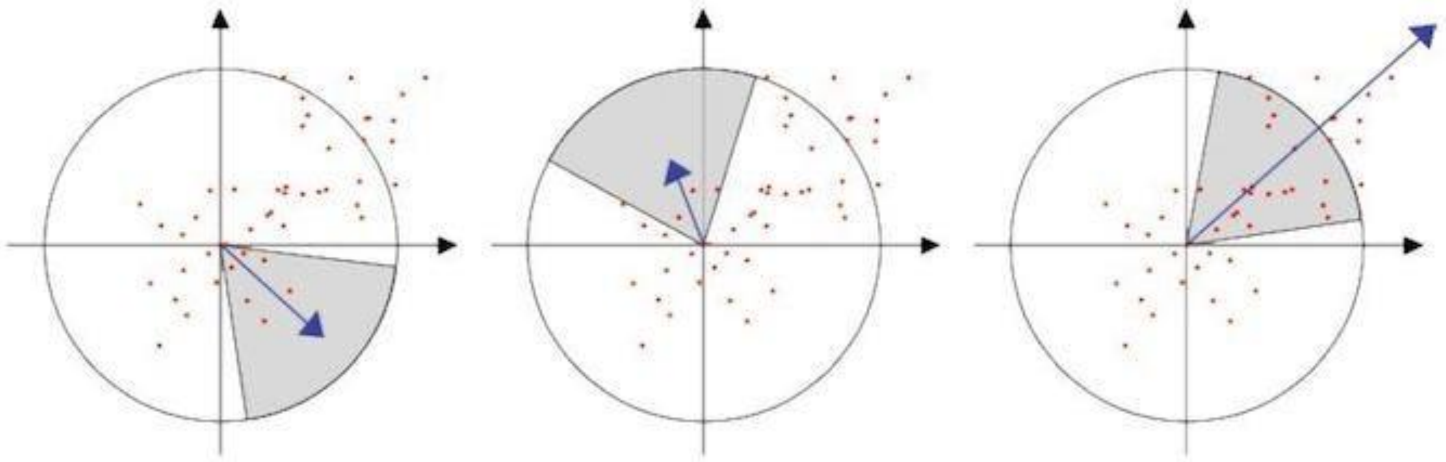
# Dominant Orientation

- Dominant orientation
  - The Haar wavelet responses are represented as vectors
  - Sum all responses within a sliding orientation window covering an angle of 60 degree
  - The two summed response yield a new vector
  - The longest vector is the dominant orientation
  - Second longest is … ignored



# Dominant Orientation - details

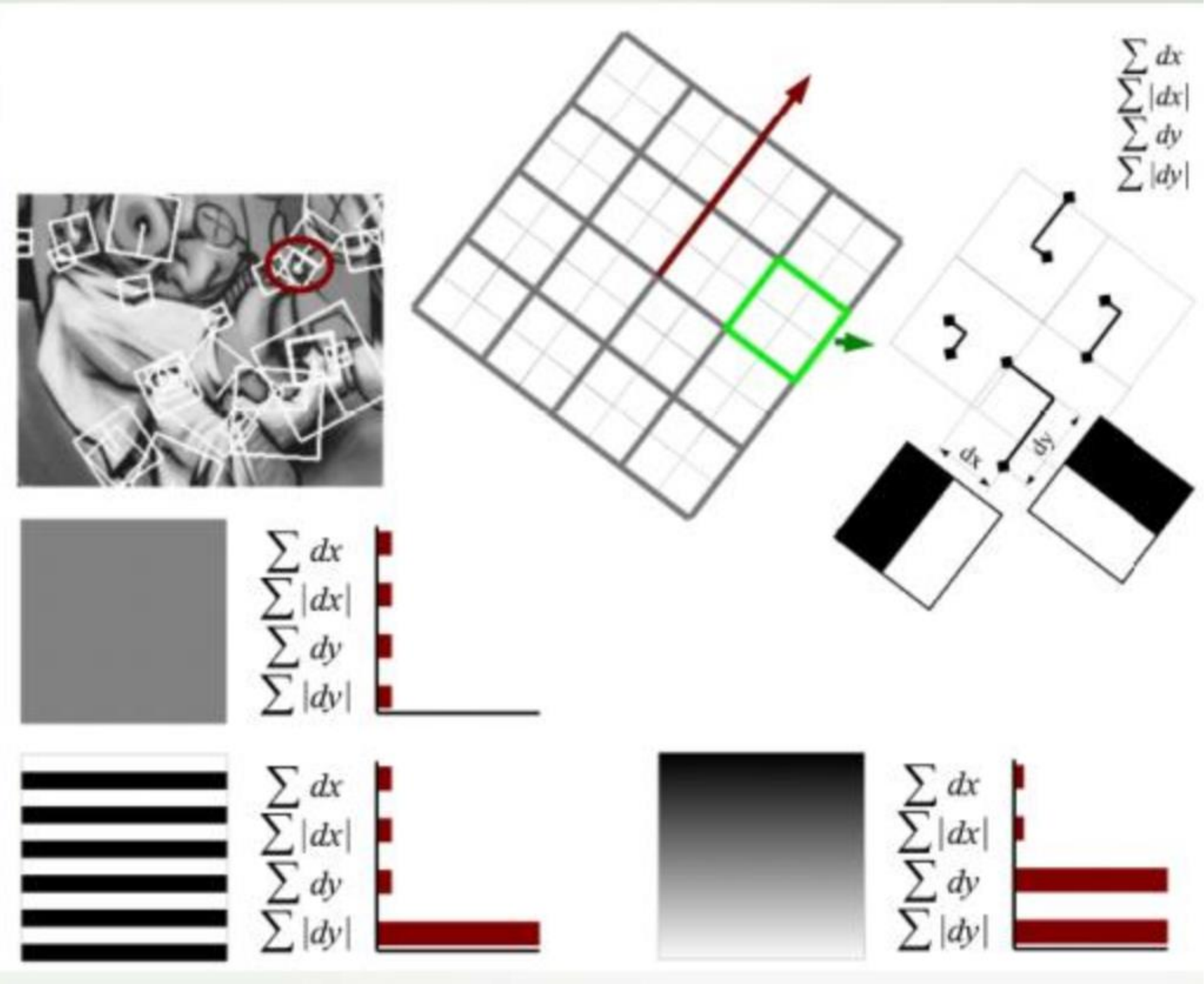
1. Surf first calculate the Haar-wavelet responses in x and y-direction, and this in a circular neighborhood of radius 6s around the keypoint, with s the scale at which the keypoint was detected. Also, the sampling step is scale dependent and chosen to be s, and the wavelet responses are computed at that current scale s. Accordingly, at high scales the size of the wavelets is big. Therefore integral images are used again for fast filtering.
2. Then we calculate the sum of vertical and horizontal wavelet responses in a scanning area, then change the scanning orientation (add π/3), and re-calculate, until we find the orientation with largest sum value, this orientation is the main orientation of feature descriptor.



# Description

- Split the interest region up into 4 x 4 square sub-regions with 5 x 5 regularly spaced sample points inside
- Calculate Haar wavelet response $d_x$ and $d_y$
- Weight the response with a Gaussian kernel centered at the interest point
- Sum the response over each sub-region for $d_x$ and $d_y$ separately → feature vector of length 32
- In order to bring in information about the polarity of the intensity changes, extract the sum of absolute value of the responses → feature vector of length 64
- Normalize the vector into unit length

# Description



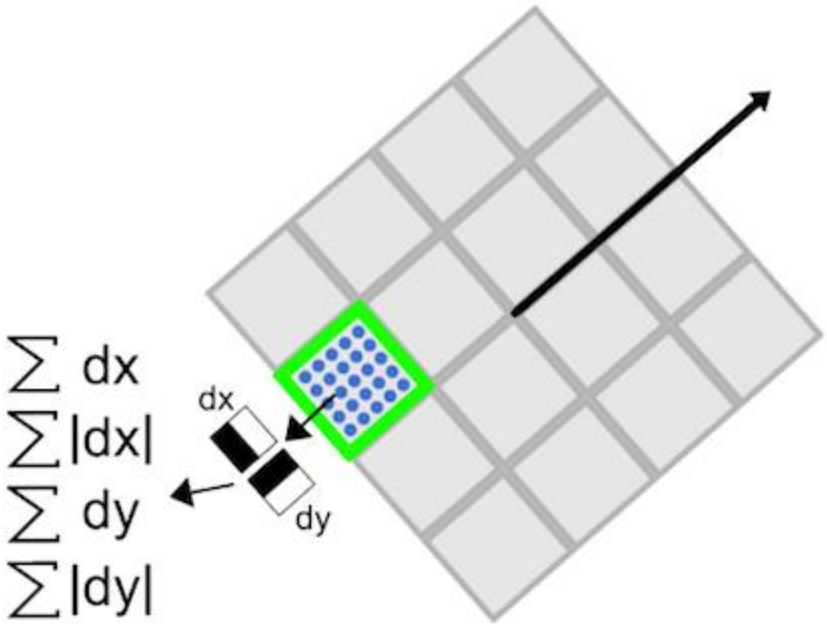
# Description

- SURF-128
  - The sum of $d_x$ and $|d_x|$ are computed separately for $d_y < 0$ and $d_y > 0$
  - Similarly for the sum of $d_y$ and $|d_y|$
  - This doubles the length of a feature vector

# SURF Descriptor

1. The first step consists of constructing a square region centered around the keypoint and oriented along the orientation we already got above. The size of this window is 20s.
2. Then the region is split up regularly into smaller 4 × 4 square sub-regions. For each sub-region, we compute a few simple features at 5×5 regularly spaced sample points.
3. For reasons of simplicity, we call dx the Haar wavelet response in the horizontal direction and dy the Haar wavelet response in the vertical direction (filter size 2s). To increase the robustness towards geometric deformations and localization errors, the responses dx and dy are first weighted with a Gaussian (σ = 3.3s) centered at the keypoint.
4. Then, the wavelet responses dx and dy are summed up over each subregion and form a first set of entries to the feature vector. In order to bring in information about the polarity of the intensity changes, we also extract the sum of the absolute values of the responses, |dx| and |dy|.
5. Hence, each sub-region has a four-dimensional descriptor vector v for its underlying intensity structure V = (∑ dx, ∑ dy, ∑|dx|, ∑|dy|).

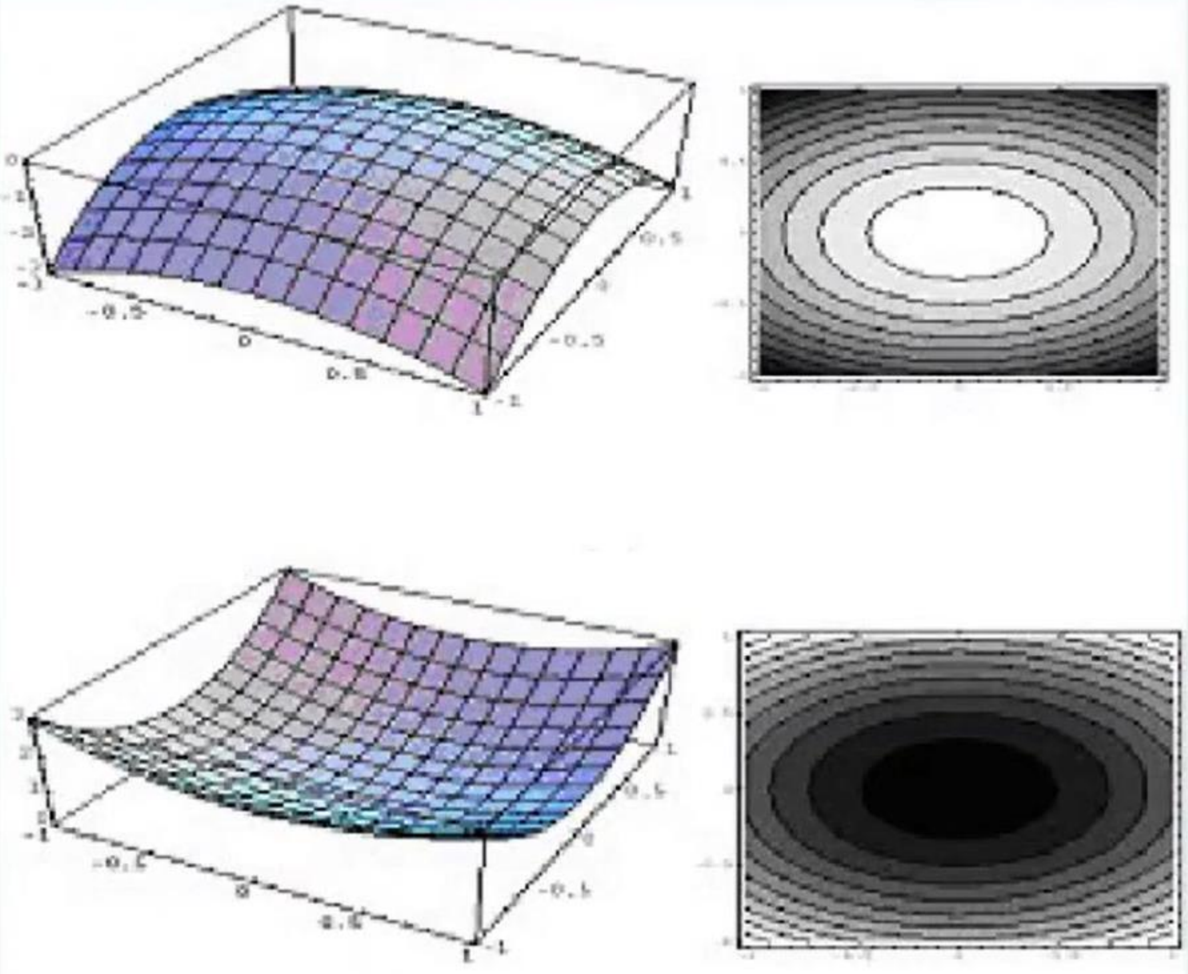
This results in a descriptor vector for all 4×4 sub-regions of length 64(In Sift, our descriptor is the 128-D vector, so this is part of the reason that SURF is faster than Sift).



# Sign of Laplacian

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

$\alpha$ : largest eigenvalue($\lambda_{max}$)
$\beta$ : smallest eigenvalue($\lambda_{min}$)



Sign of Laplacian is nothing but The sign of the trace of the Hessian Matrix.

$$H = \begin{pmatrix} Dxx & Dxy \\ Dyx & Dyy \end{pmatrix}$$

Sign of Laplacian = Dxx + Dyy

Typically interest points are found at blob type structures.

The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse situation.

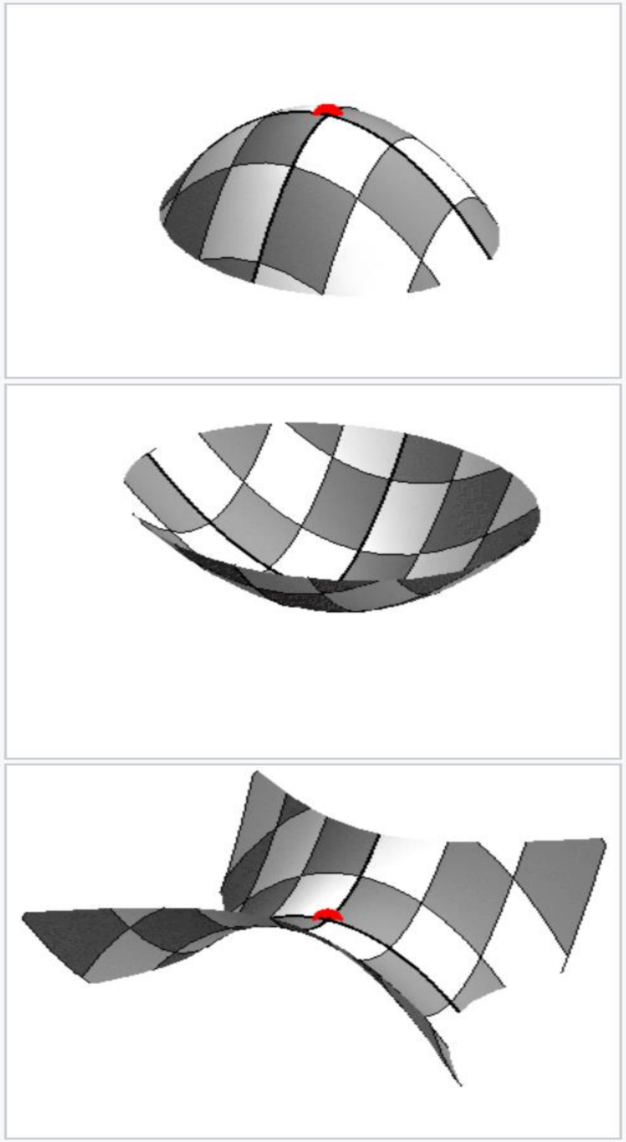
With time Complexity = O(0).

# Determinant of Hessian

$$\det(Hessian(I)) = I_{xx} I_{yy} - I_{xy}^2$$

- "Determinant of the Hessian" summarizes the curvature at a point.
  - *If D*>0, the point looks like either a bowl upward or downward,
  - a) If the first term in the upper left corner of our Hessian matrix ($I_{xx}$) is a **positive** number, we are dealing with a **minimum**. (dark blob)
  - b) If the first term in the upper left corner of our Hessian matrix ($I_{xx}$) is **negative**, we are dealing with a **maximum**. (bright blob)
  - If *D*<0, the point looks like a saddle point.

- As the edge looks more like a saddle than a bowl, this is one way that you can separate good, stable interest points from edges using the Hessian.



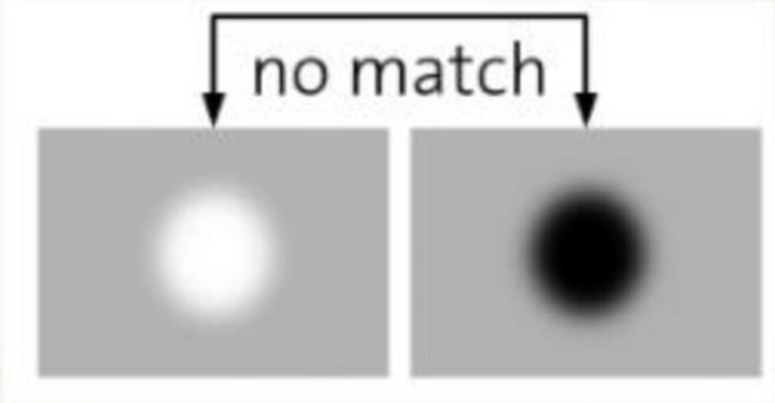
The Hessian approximates the function at a critical point with a second degree polynomial.

# Matching

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

$\alpha$ : largest eigenvalue($\lambda_{max}$)
$\beta$ : smallest eigenvalue($\lambda_{min}$)

- Fast indexing through the sign of the Laplacian for the underlying interest point
  - The sign of trace of the Hessian matrix
  - Trace = $L_{xx} + L_{yy}$




- Either 0 or 1 (Hard thresholding, may have boundary effect ...)
- In the matching stage, compare features if they have the same type of contrast (sign)

# SURF Feature vector Creation and Matching

**Feature Vector Creation:**

- The creation of a feature vector (V ) 64-dimensional (5) is based on information from descriptors of a area around the interest point.

$$V = \left(\sum dx, \sum dy, \sum |dx|, \sum |dy|\right) \qquad (5)$$

- These descriptors are the result of applying filters (Haar wavelets) Figure 2D centered around (4 × 4 regular subregions) the interest points of the image, and provide the gradient in x and y directions.
- This allows invariance to rotation, translation and brightness during matching.

**Matching Feature Vectors:**

- In matching step, it is verified the sign of the Laplacian matrix (6) with no extra computational cost, since this information is previously known.

$$\nabla^2 L = tr(H) = L_{xx}(\mathbf{x}, \sigma) + L_{yy}(\mathbf{x}, \sigma) \qquad (6)$$

- This allows the comparison of two similar types of contrast (i.e., dark or clear blob-types)
- The Laplacian of a Hessian matrix is the sum of the diagonal elements. Finally, the algorithm seeks for the smallest Euclidean distance between the vectors, that is, the pairs most likely to be similar.

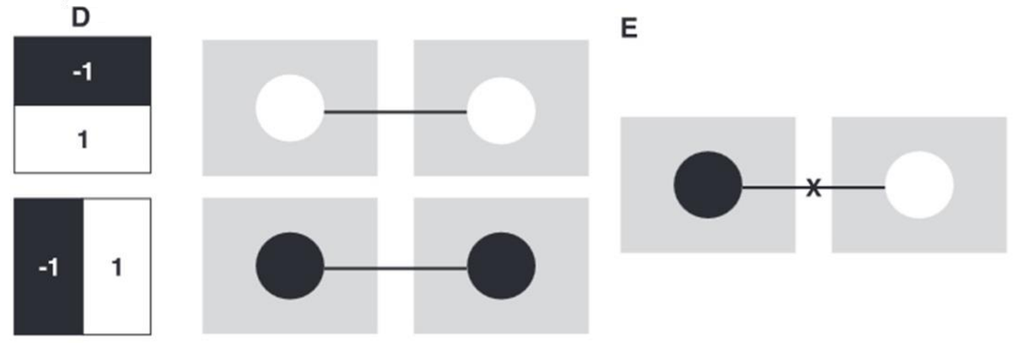



**Figure types of matching used by SURF.**

**(D)** Haar wavelet filters in response to $dx$ (top) and $dy$ (bottom) directions with ($\sigma = 2s$) kernel size.

**(E)** Candidate interest points to perform matching are combinations between circular areas with the same contrast and, the oposite are not considered.

# SURF Algorithm Highlights

- Create an Integral image (to speeded up the calculations and save time)
- Uses box filters instead of Difference of Gaussians to approximate Laplacians
- Use Haar wavelets to get key point orientations
- SURF is good at Blur and rotation variations
- SURF is not as good as SIFT invariance to illumination and viewpoint changes
- Surf is ~3 times faster than SIFT

# Text Books and References

## ❖ Text Books

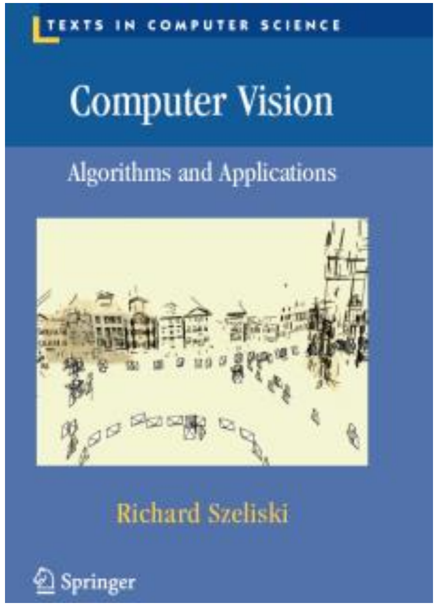
- Richard Szeliski. "Computer vision: Algorithms and Applications. Springer Nature, Second Edition", 2022
- E. R. Davies, Computer Vision Principles, Algorithms, Applications, Learning, Elsevier,5th Edition, 2017

## ❖ References

- Rafael C. Gonzales, Richard E. Woods, "Digital Image Processing", Fourth Edition, Pearson Education, 2018.
- Richard Szeliski ,"Computer Vision: Algorithms and Applications" , Springer 2015
- Richard Szeliski. "Computer Vision: Algorithms and Applications 2nd Edition – final draft", 2021
- Intro to Digital Image Processing
- Digital Image Processing 2nd edition

## ❖ Credits: Slides and Video content borrowed from:

- First Principles of Computer Vision
- https://robotacademy.net.au/
- MIT 6.S094: Computer Vision
- MIT Introduction to Deep Learning | 6.S191
- Digital Image Processing 2nd edition
- CSE/EE486 Computer Vision I (Penn State University)
- https://www.cs.umd.edu/class/fall2019/cmsc426-0201/files/21_SURF.pdf
- https://www.slideserve.com/dolf/surf-speeded-up-robust-features-eccv-2006-herbert-bay-tinne-tuytelaars-and-luc-van-gool (SURF: original presentation)
- https://www.tarjomefa.com/wp-content/uploads/2016/09/5349-English.pdf (SURF: original paper)
- https://medium.com/@deepanshut041/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e
- SURF - Speeded Up Robust Features




PDF online