Scott Smith
Advanced Image Processing
March 15, 2011

Speeded-Up Robust Features

SURF

Overview

- Why SURF?
- How SURF works
  - Feature detection
  - Scale Space
  - Rotational invariance
  - Feature vectors
- SURF vs Sift

Assumptions

- We are only looking at grey scale images
- We will only discuss 2-d (there are 3-d extensions)

SURF Applications

- Essentially, the same as SIFT
- Generate Feature Vectors
  - Interest Point descriptor
- Registration points
- Feature detection
- Feature matching
  - Object identification

SURF Roadmap

1. Find image interest points
   - Use determinant of Hessian matrix
2. Find major interest points in scale space
   - Non-maximal suppression on scaled interest point maps
3. Find feature “direction”
   - We want rotationally invariant features
4. Generate feature vectors

SURF Roadmap

1. Find image interest points
   - Use determinant of Hessian matrix
2. Find major interest points in scale space
   - Non-maximal suppression on scaled interest point maps
3. Find feature “direction”
   - We want rotationally invariant features
4. Generate feature vectors

Hessian Matrix for feature detection

- A Hessian matrix in 2-dimensions consists of a 2 x 2 matrix containing the second-order partial derivatives as follows:

$$\begin{bmatrix} \frac{\partial I^2}{\partial x^2} & \frac{\partial I^2}{\partial x \partial y} \\ \frac{\partial I^2}{\partial y \partial x} & \frac{\partial I^2}{\partial y^2} \end{bmatrix}$$

- Symmetric matrix

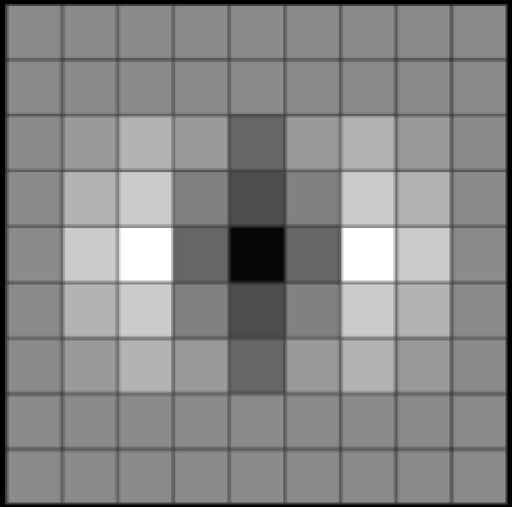
Hessian Matrix for feature detection

- For any square matrix, the determinant of the matrix is the product of the eigenvalues
- For the Hessian matrix, the eigenvectors form an orthogonal basis showing the direction of curve (gradient) of the image
  - If both eigen values are positive, local min
  - If both eigen values are negative, local max
  - If eigen values have mixed sign, saddle point

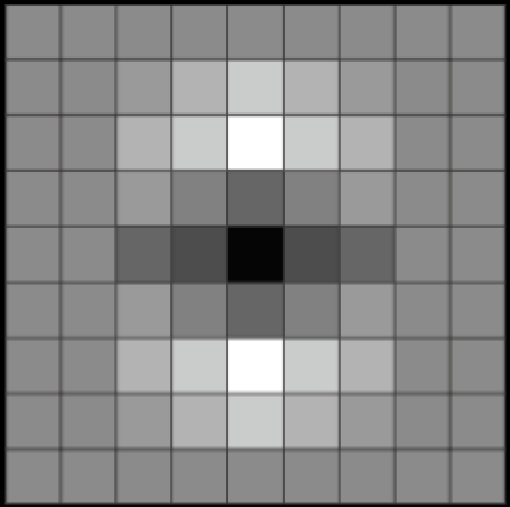
Hessian Matrix for feature detection

- Therefore, if the product of the eigen values is positive, then they were either both positive or both negative and we are at a local extremum
- Typically, we apply some kind of thresholding to the determinant value so we only detect major features.
  - You can control the number of interest points this way
- Some algorithms save the trace of the hessian to remember whether a min or a max

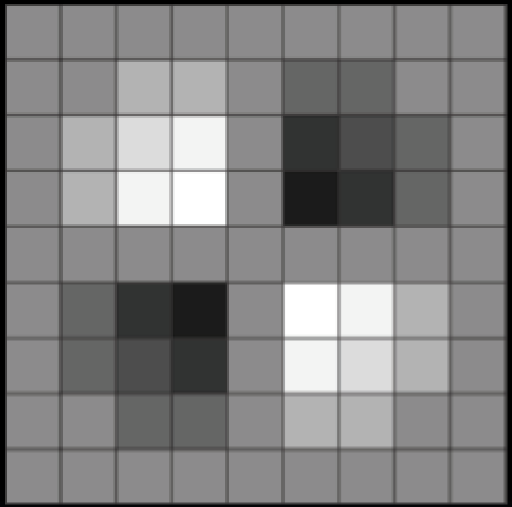
Laplacian of Gaussian (9x9 filters)



L_{xx}

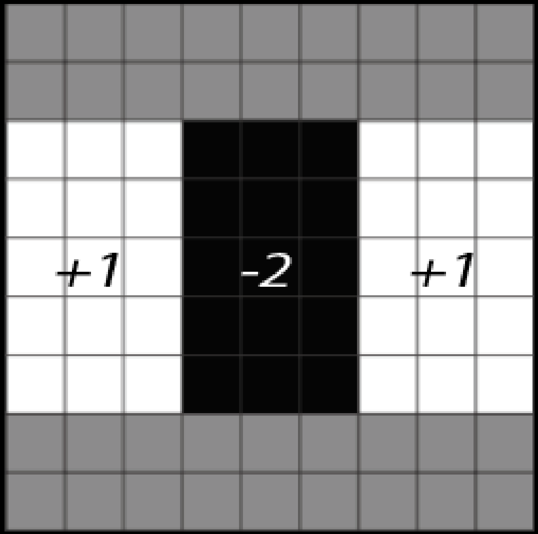


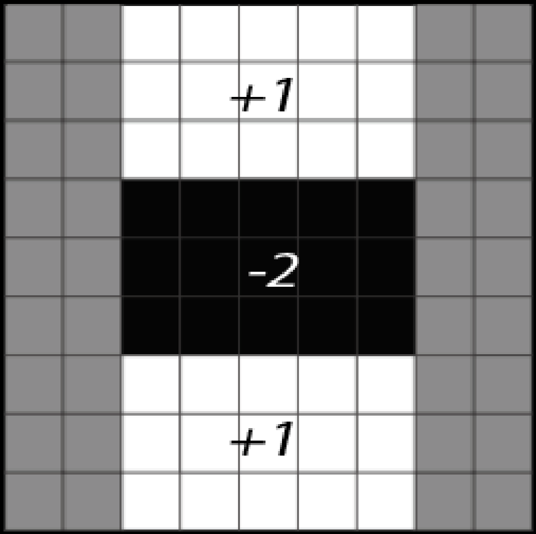
L_{yy}

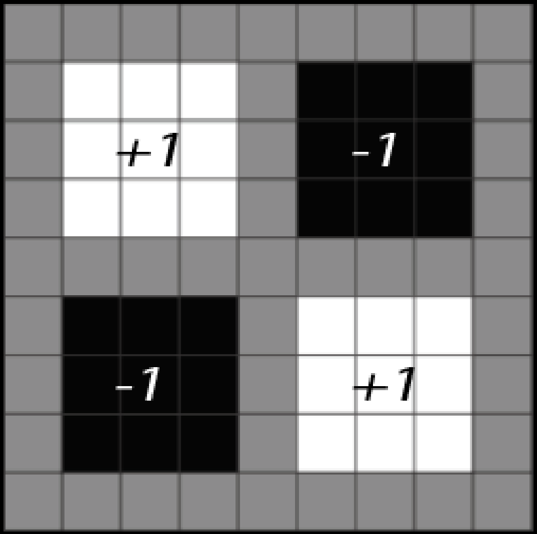


L_{xy}

LoG Approximations



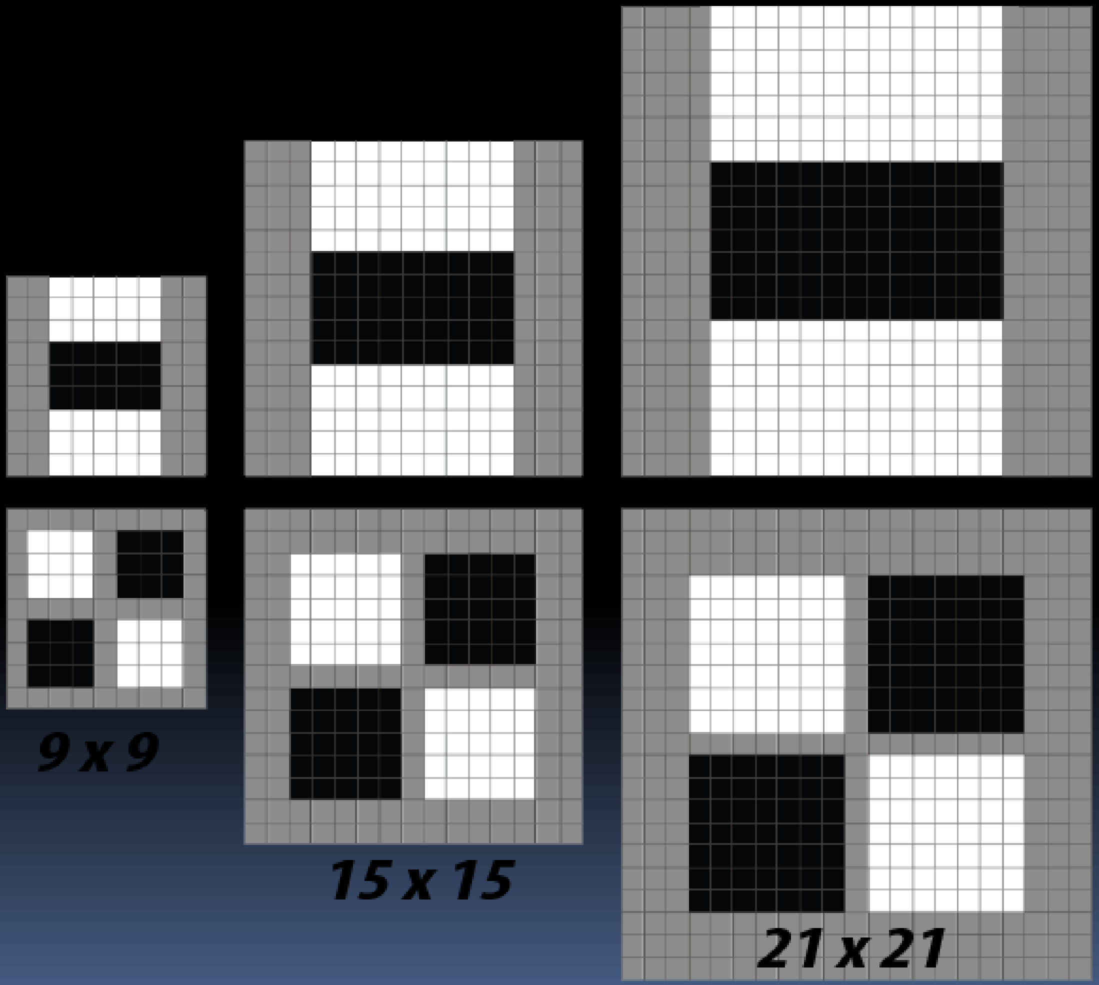




D_{xx} D_{yy} D_{xy}

- In practice, these approximations are very close to LoG.
- Need to normalize for filter size
- We can take advantage of the large areas of constant weighting to speed up the algorithm.

How filters grow



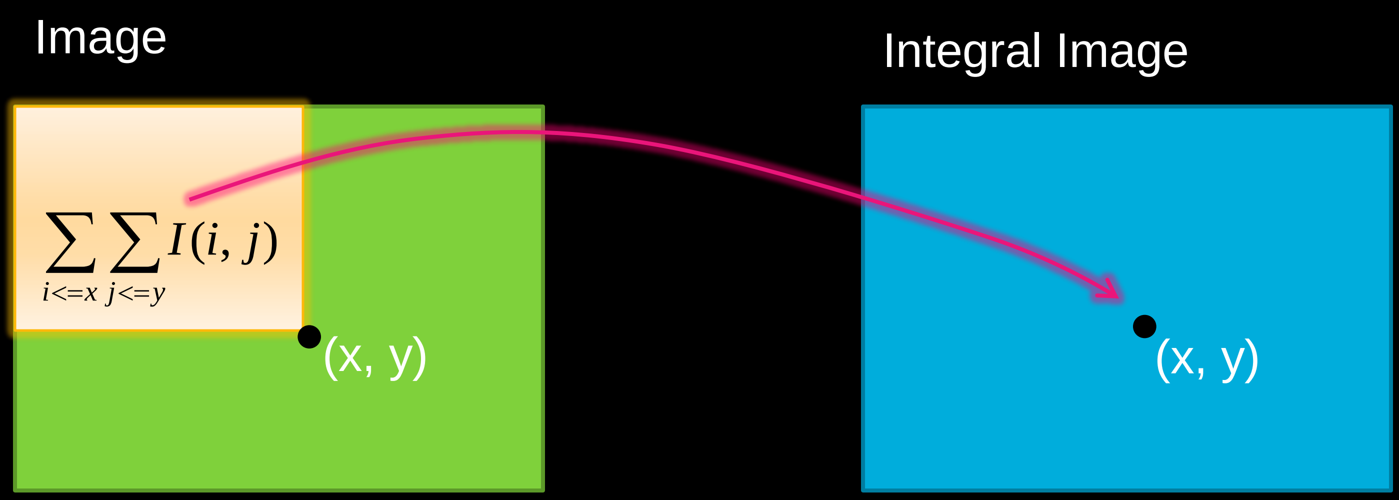
Integral Image - Overview

- Goal of Integral Images
  - Fast computation of box convolutions
  - We need a fast way to compute the intensities for any rectangle within the image which isn't sensitive to rectangle size
- Computation time is independent of the size of the filter!
- Can be used for any box filter application
- Originally from "Rapid object detection using a boosted cascade of simple features" by Viola and Jones (2001)

Integral Image – How it works

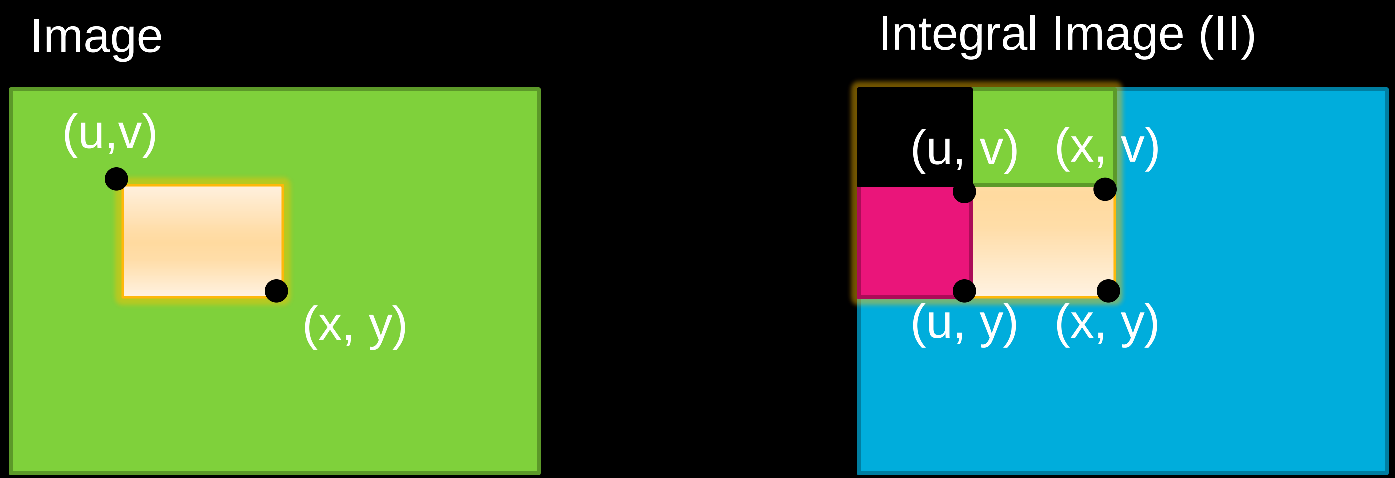
- Create an "Integral Image"
  - An Integral image has the same size as the image you are analyzing
  - The value of the <u>integral image</u> at any point (x, y) is the sum of the intensity values for all points in the <u>image</u> with location less than or equal to (x, y)
- Use the integral image to compute the intensities for any rectangle in the image

Integral Image – Creating



The value of the Integral Image at point (x, y) is the sum of all of the intensities in the gold box. An integral image can be created in a "recursive" manner to minimize computations. Start at the top-left corner and work down a row at a time.

Integral Image – Using it



The sum of the intensities in the gold box is simply:

$$II(x, y) - II(x, v) - II(u, y) + II(u, v)$$

Integral Images and Hessian Matrices

- Recall that the Hessian box filters consisted of squares with a common weight
- We can use the Integral Image to get the sum of the intensities for the square, multiply by the weight factor and add the resulting sums for the box filter together.
  - Don't forget to normalize for filter size
- The matrix with the thresholded determinants for a particular filter size is called the "blob response map"

Blob Response

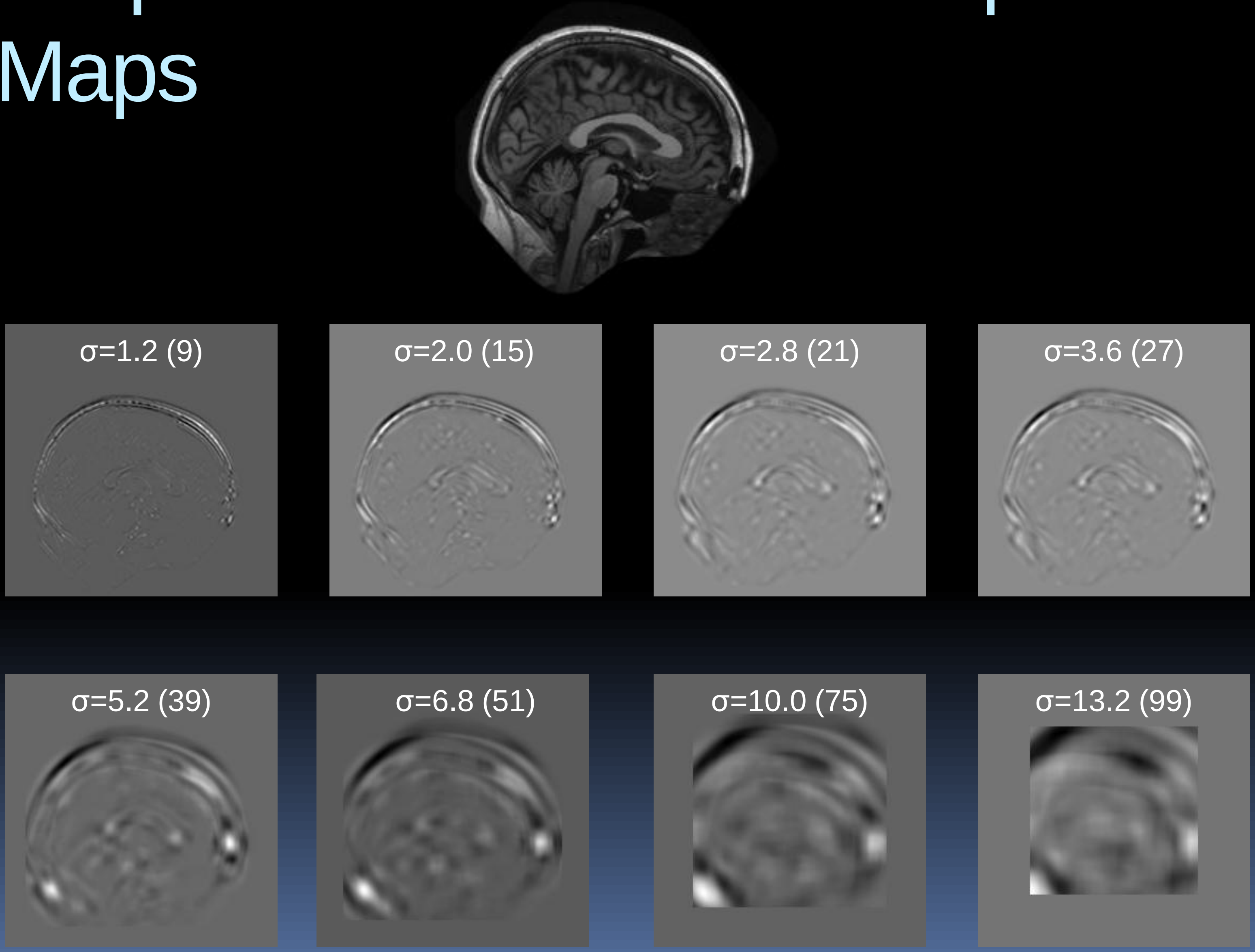
- Blob Response at location x = (x, y, σ)

$$\det(H_{approx}) = D_{xx}D_{yy} - (.9D_{xy})^2$$

- For a 9 x 9 matrix, σ = 1.2; In general:

$$\sigma = (filtersize/9)*1.2$$

Corpus Callosum Blob Response Maps



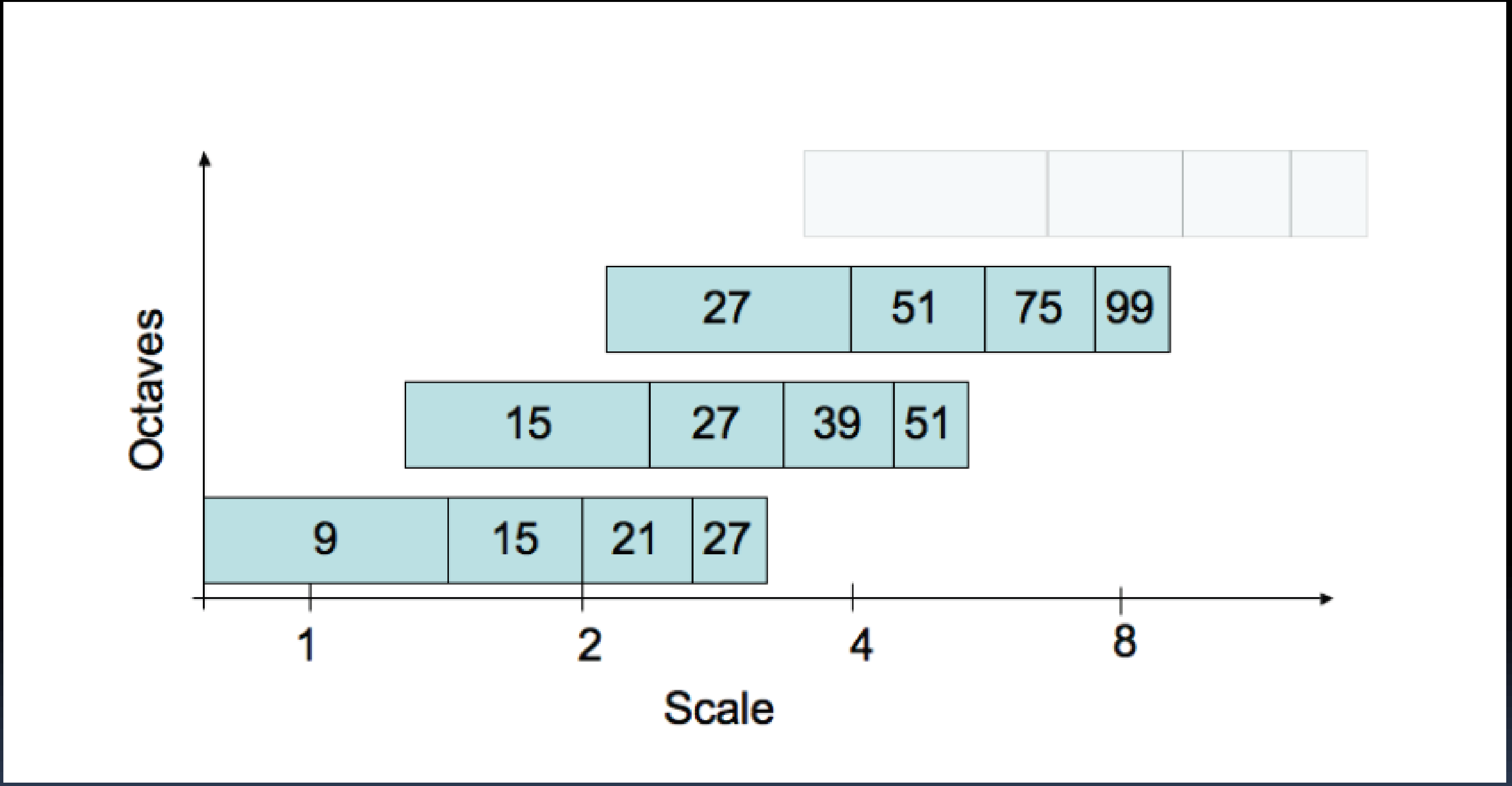
SURF Roadmap

1. Find image interest points
   - Use determinant of Hessian matrix
2. Find major interest points in scale space
   - Non-maximal suppression on scaled interest point maps
3. Find feature “direction”
   - We want rotationally invariant features
4. Generate feature vectors

Octaves

- An octave is defined as a series of filters which have a range which approximates a doubling of scale.
- Bay computes 3 octaves with the option of going to 4 octaves
- The octaves overlap to ensure full coverage of each scale

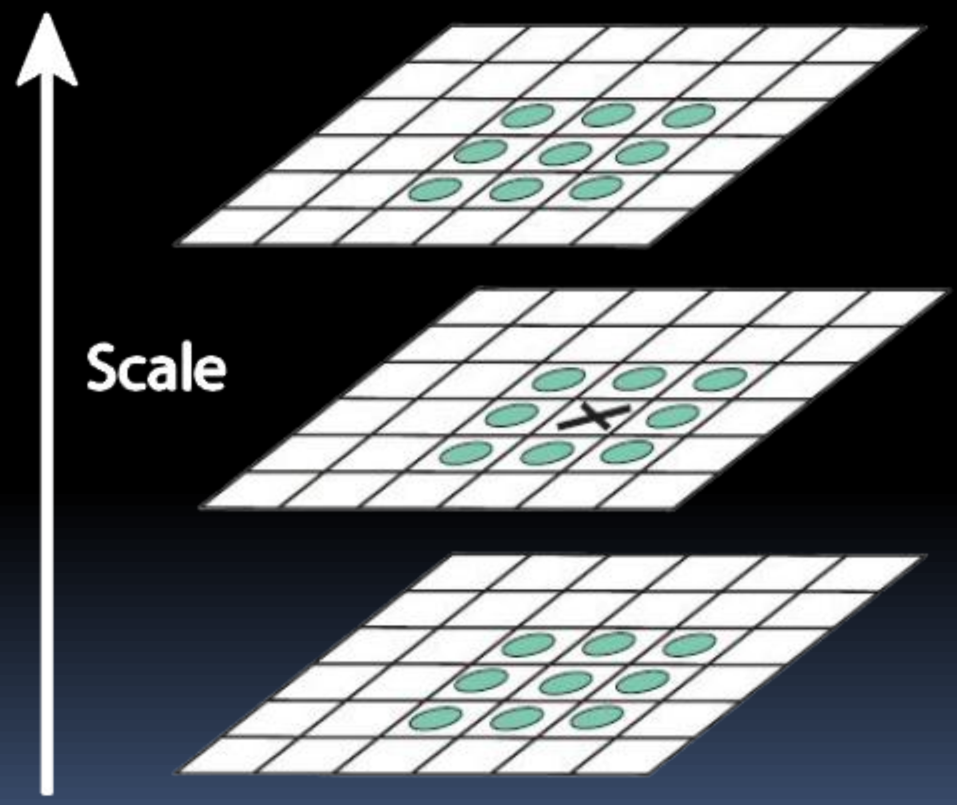
Octaves



Finding the main features

- Non-maximal suppression is applied
- We do normal 3x3 non-maximal suppression within the same blob response map
- We also do non-maximal suppression with the blob response map above and below the image in scale space for each octave
  - This means that we only use the middle two blob response maps for each octave

Non-Maximal Suppression in 3D



Interpolate the interest points

- Because of the coarse scale of the scale space, we need to interpolate the interest point to arrive at the correct scale (σ); express the hessian as a Taylor expansion

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

- Differentiating and set to 0 gives

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

Interpolate the interest points

$$\frac{\partial^2 H}{\partial \mathbf{x}^2} = \begin{bmatrix} d_{xx} & d_{yx} & d_{sx} \\ d_{xy} & d_{yy} & d_{sy} \\ d_{xs} & d_{ys} & d_{ss} \end{bmatrix}$$

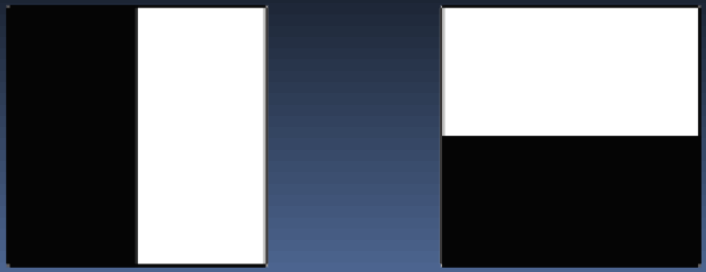
$$\frac{\partial H}{\partial \mathbf{x}} = \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix}.$$

SURF Roadmap

1. Find image interest points
   - Use determinant of Hessian matrix
2. Find major interest points in scale space
   - Non-maximal suppression on scaled interest point maps
3. Find feature “direction”
   - We want rotationally invariant features
4. Generate feature vectors

Haar Transforms

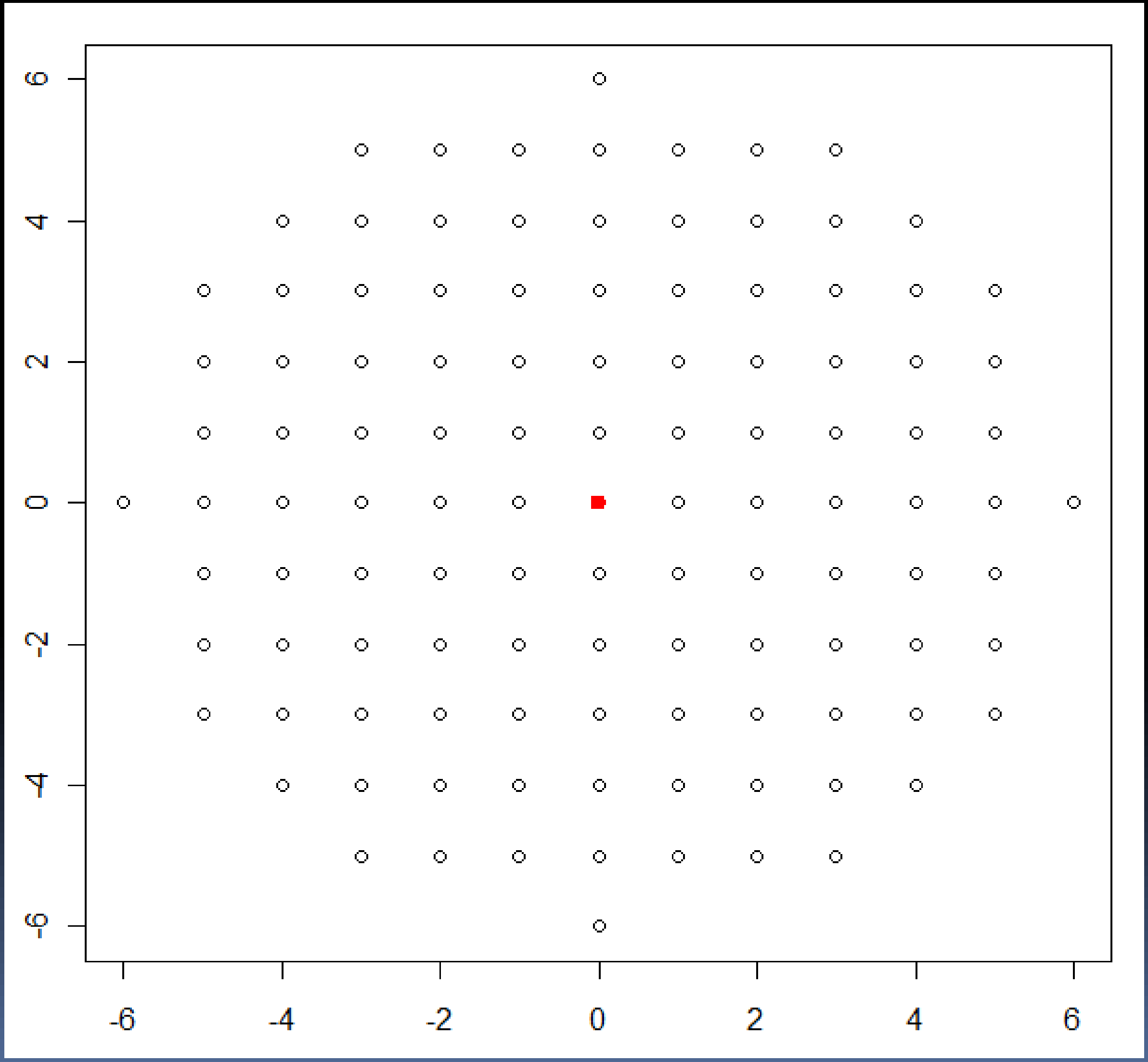
- We use Haar Transforms to assess the primary direction of the feature.
- The intuition is that they give you a sense of the direction of the change in intensity.
  - They are resistant to overall luminance changes
- Simple box filters (=>Integral images)



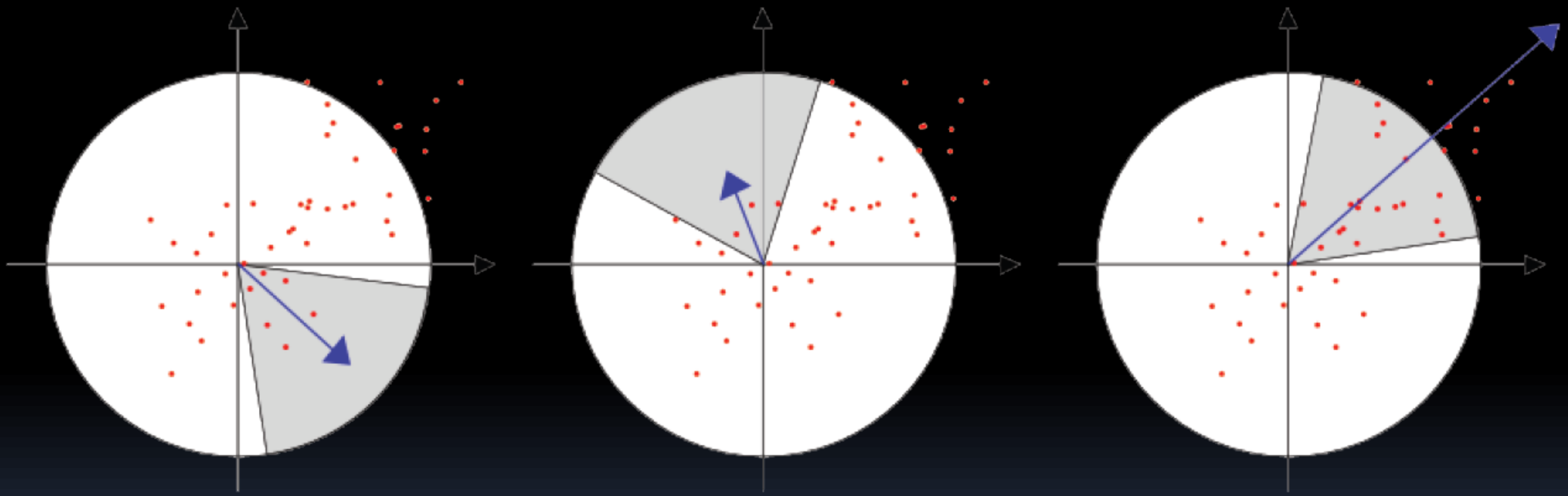
Computing the rotation

- For each feature
  - Look at pixels in a circle of 6*σ radius
  - Compute the x and y Haar transform for each point
  - Use the resulting values as x and y coordinates in a Cartesian map.
    - Weight each point with a Gaussian of 2*σ based on the distance from the interest point.
  - Rotate a wedge of π/3 radians around the circle.
    - Choose direction of maximum total weight

Circle Points



Rotation



SURF Roadmap

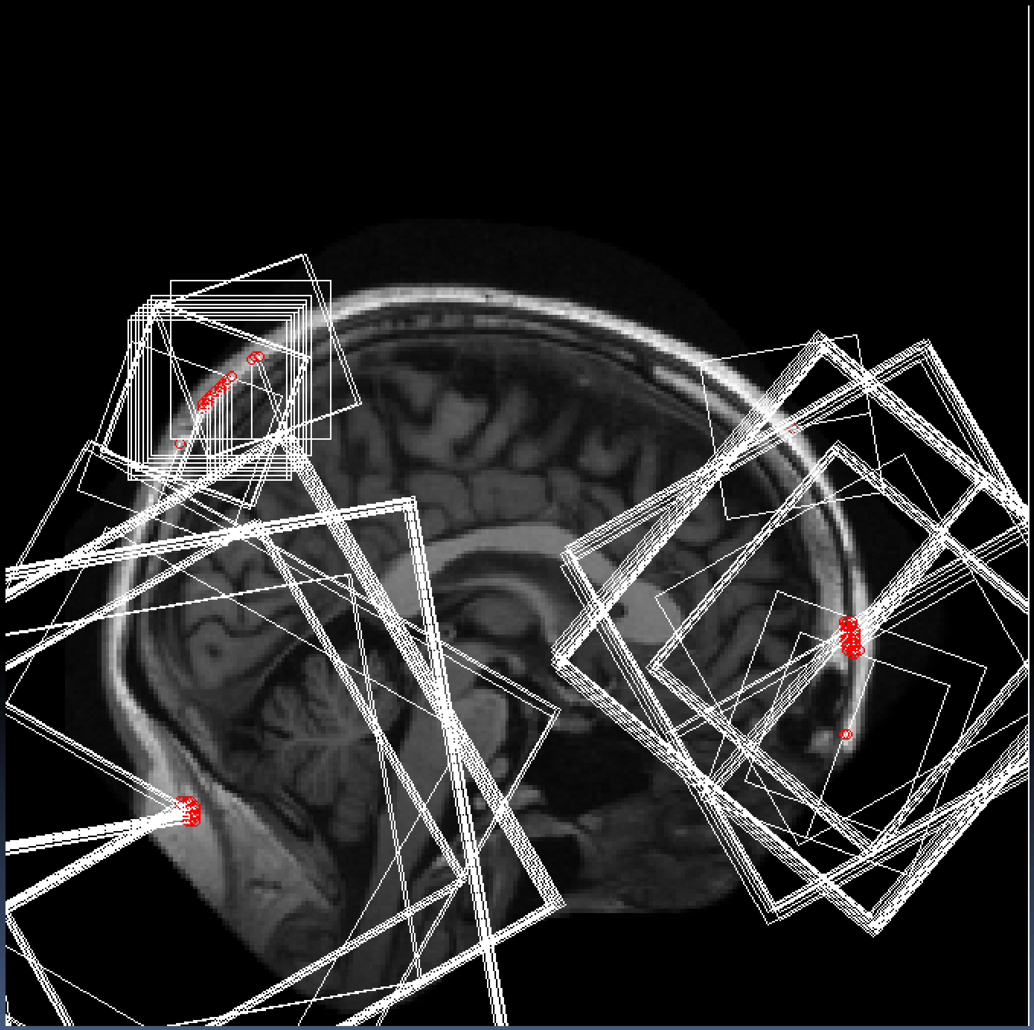
1. Find image interest points
   - Use determinant of Hessian matrix
2. Find major interest points in scale space
   - Non-maximal suppression on scaled interest point maps
3. Find feature “direction”
   - We want rotationally invariant features
4. Generate feature vectors

Computing Feature Vectors

- A square descriptor window is constructed with a size of 20*σ centered on each interest point and orientation based on the derived rotation
- Divide the descriptor window into 4 x 4 sub-regions
  - Each sub-region is 5*σ square
  - Haar wavelets of size 2*σ are computed for 25 regularly spaced points in each sub-region
    - dx and dy are computed at each point in the rotated direction (=> no integral images ☹)

Feature Windows

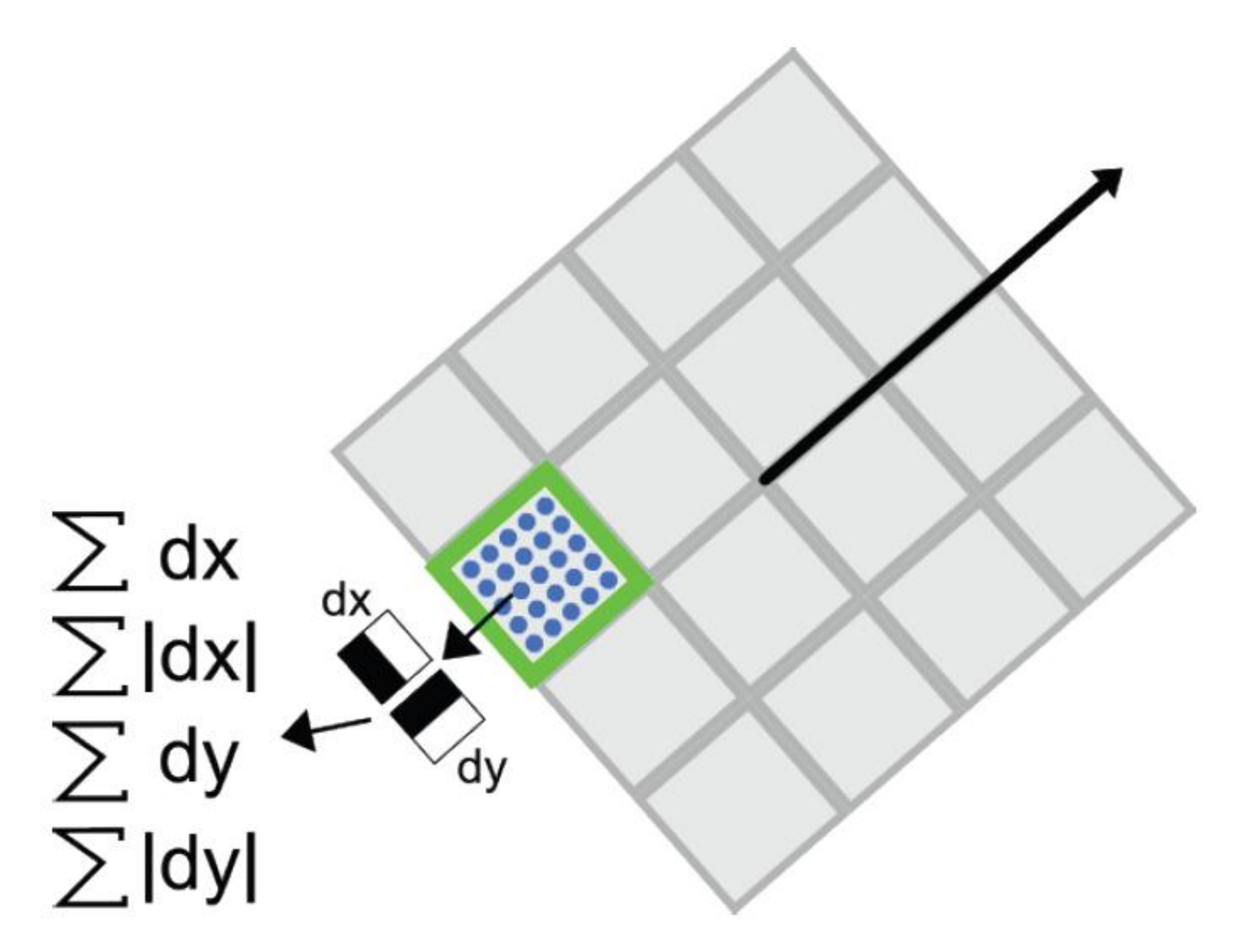
Feature Windows



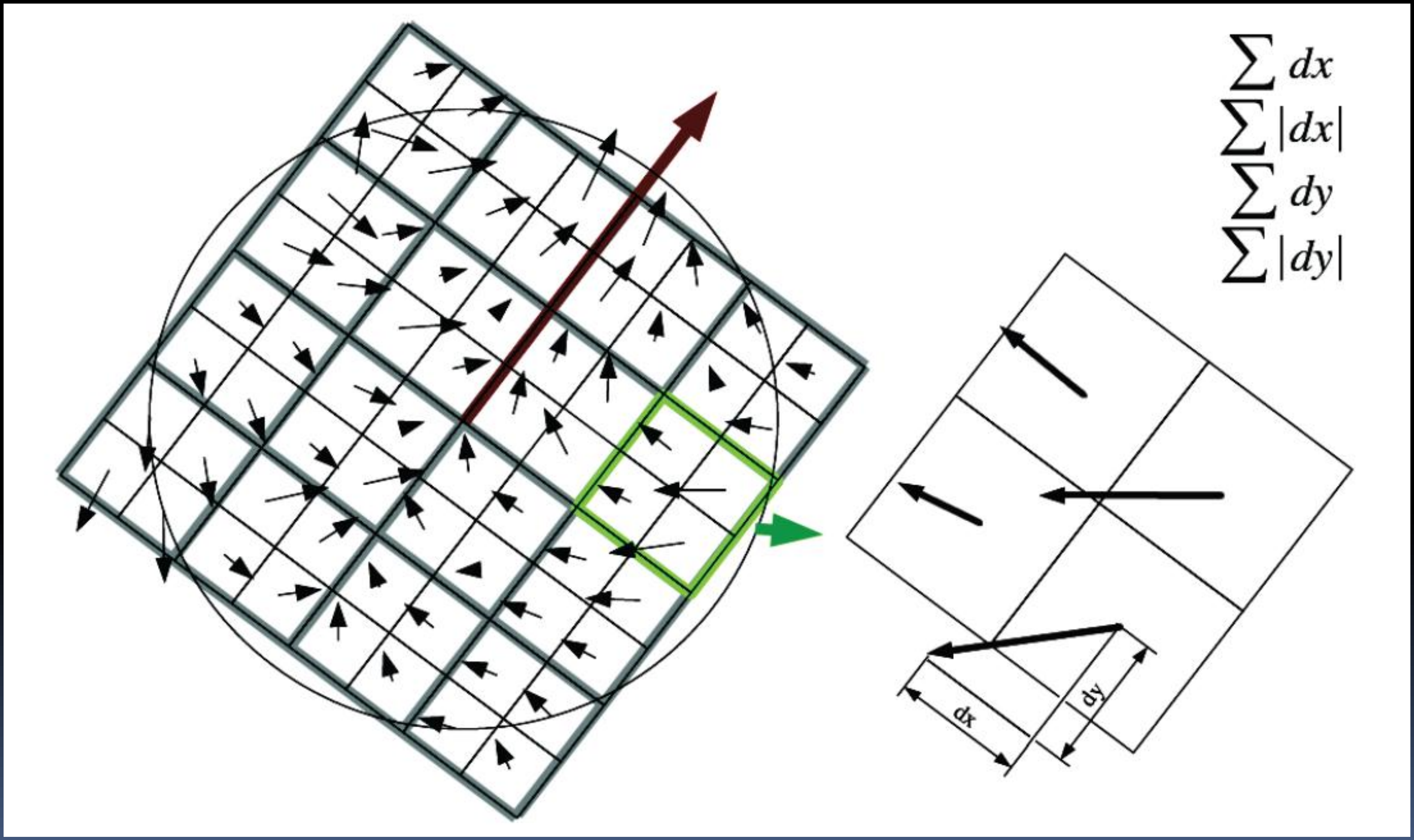
Computing Feature Vectors

- For each of the 16 sub-regions we compute 4 values
  - Sum of dx
  - Sum of dy
  - Sum of abs(dx)
  - Sum of abs(dy)
- Feature vector is a 64 dimensional vector consisting of the above 4 values for each of the 16 sub-regions

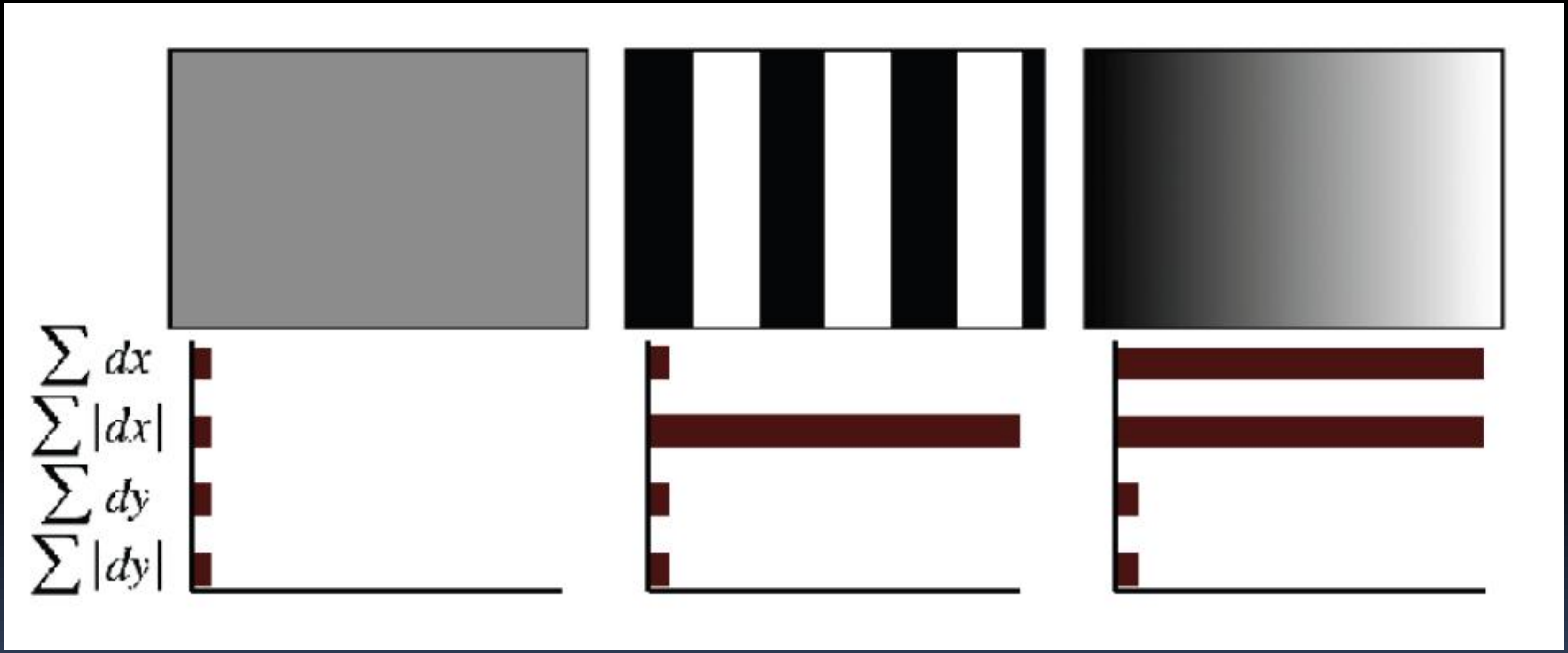
Feature Vectors



Feature Vectors



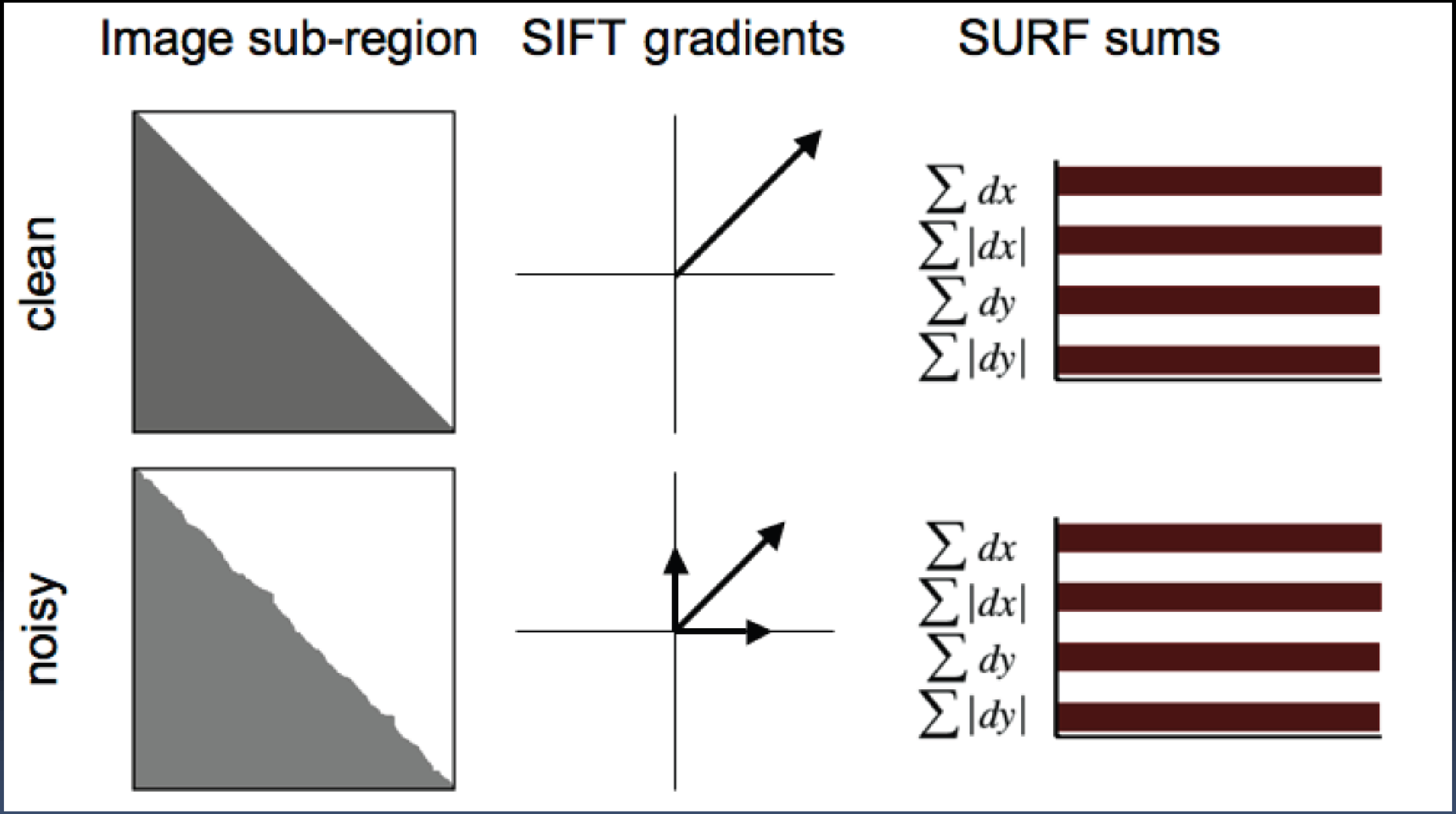
Feature Vectors



SURF vs SIFT

- SURF is roughly 3-5 times faster than SIFT
  - More resilient to noise than SIFT.
  - Also, more easily adapted to parallel processing since each Hessian image can be independently generated (unlike SIFT)
- Some loss of accuracy from SIFT in certain situations
  - Authors claim it is minimal
  - Perhaps not as invariant to illumination and viewpoint change as SIFT

SURF vs SIFT with noise



References

- “SURF: Speeded Up Robust Features” – H. Bay, et. al., 2006
- “Notes on the OpenSURF library” - http://www.chrisevansdev.com/computer-vision-opensurf.html