Computer Vision

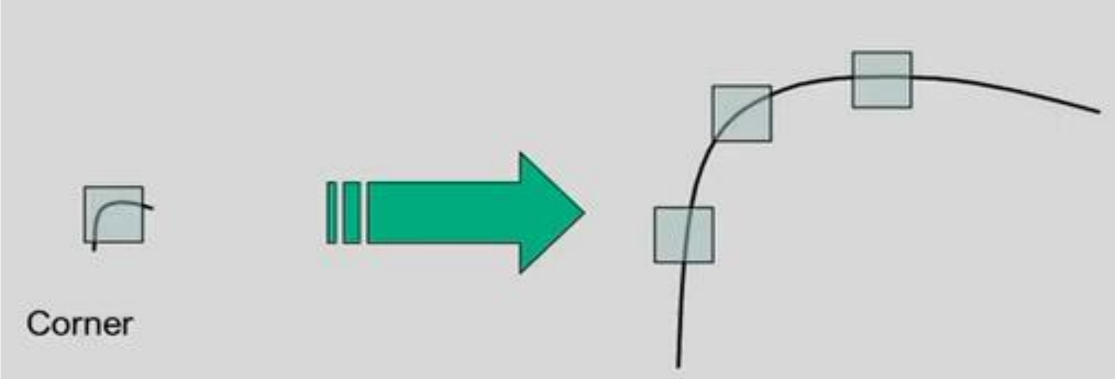
Feature Extraction
(SIFT, SURF)

Contents

- SIFT
- SURF

Limitations of Harris and Hessian Corner Detectors

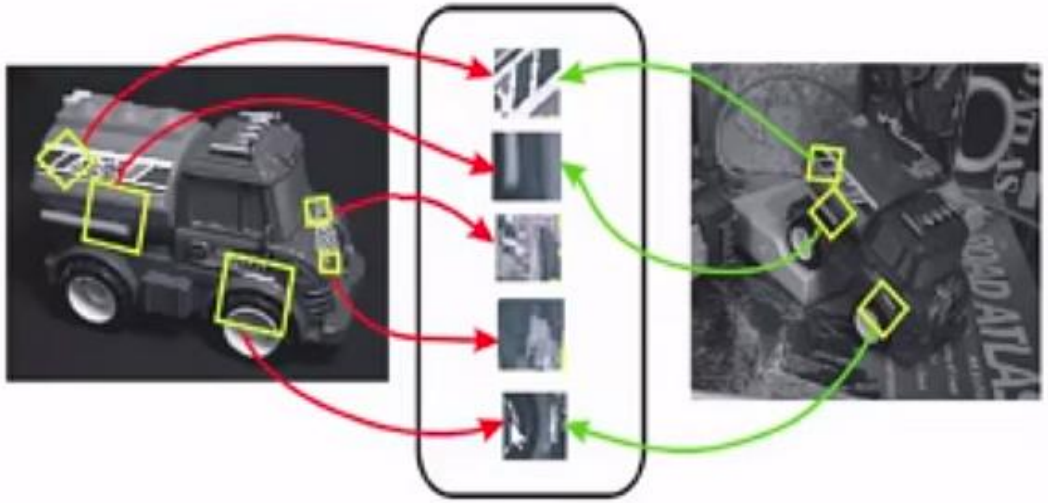
- Image scaling



- Corner gets magnified and becomes bigger than the size of the window by zooming
- Herris and hessian detect classify corner points as edges
- They can not detect corners if image is up scaled
- That is they are not covariant to scaling

SIFT (Scale-Invariant Feature Transform) Detector

- Proposed by David Lowe
- Detects distinct key points/features in an image
- Key points are robust to changes in scale, rotation, and affine transformations


SIFT Features

SIFT (Scale-Invariant Feature Transform) Detector

- Each image has a different background
- Is captured from different angles
- Size is different
- Has different objects in the foreground

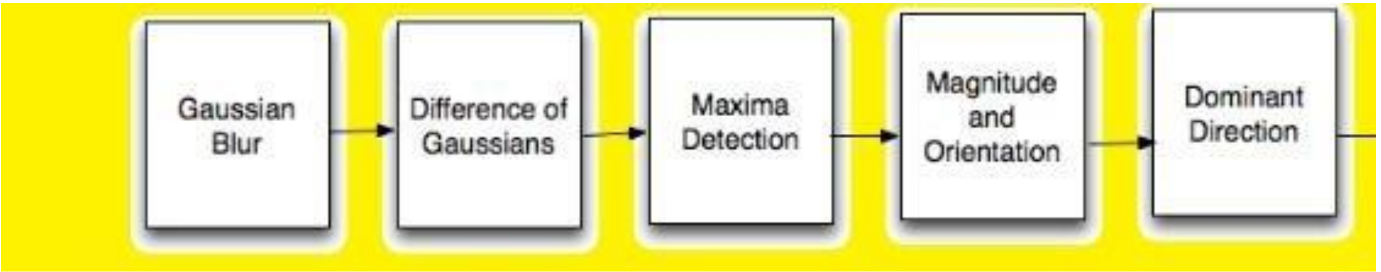
Advantages of SIFT Detector

- Locality
  - Features are local, robust to occlusion
  - Does not require segmentation of objects
- Distinctiveness
  - Features can be matched to a large database of objects
- Quantity
  - Many features can be generated even if objects are small
- Efficiency
  - Close to real-time performance
- Extensibility
  - Can easily be extended to a wide range of different feature types

SIFT Algorithm

1. Construct a Scale Space:
   - Generate images over multiple scales
   - Ensures that features are scale-independent
2. Key point Localisation:
   - Select key points based on measure of stability
   - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
   - Compute best orientations for each key point region
   - To ensure that keypoints are rotation invariant
4. Keypoint Descriptor:
   - Use local image gradients at selected scale and rotation

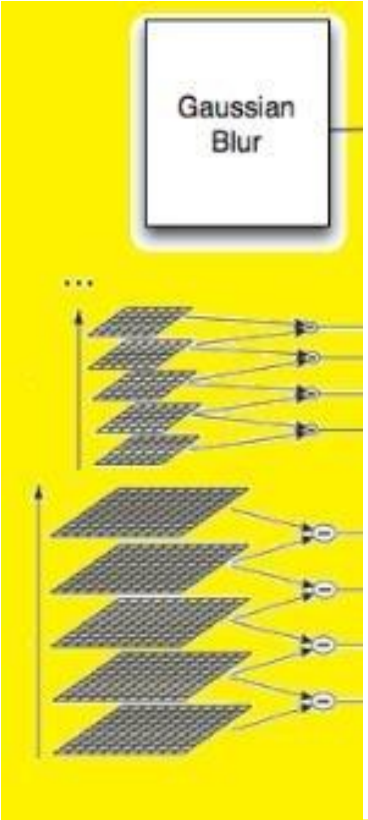
SIFT Algorithm



SIFT Algorithm

1. Construct a Scale Space:
   - Generate images over multiple scales
   - To ensures that features are scale-independent
2. Key point Localisation:
   - Select key points based on measure of stability
   - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
   - Compute best orientations for each key point region
   - To ensure that keypoints are rotation invariant
4. Keypoint Descriptor:
   - Use local image gradients at selected scale and rotation

SIFT Algorithm

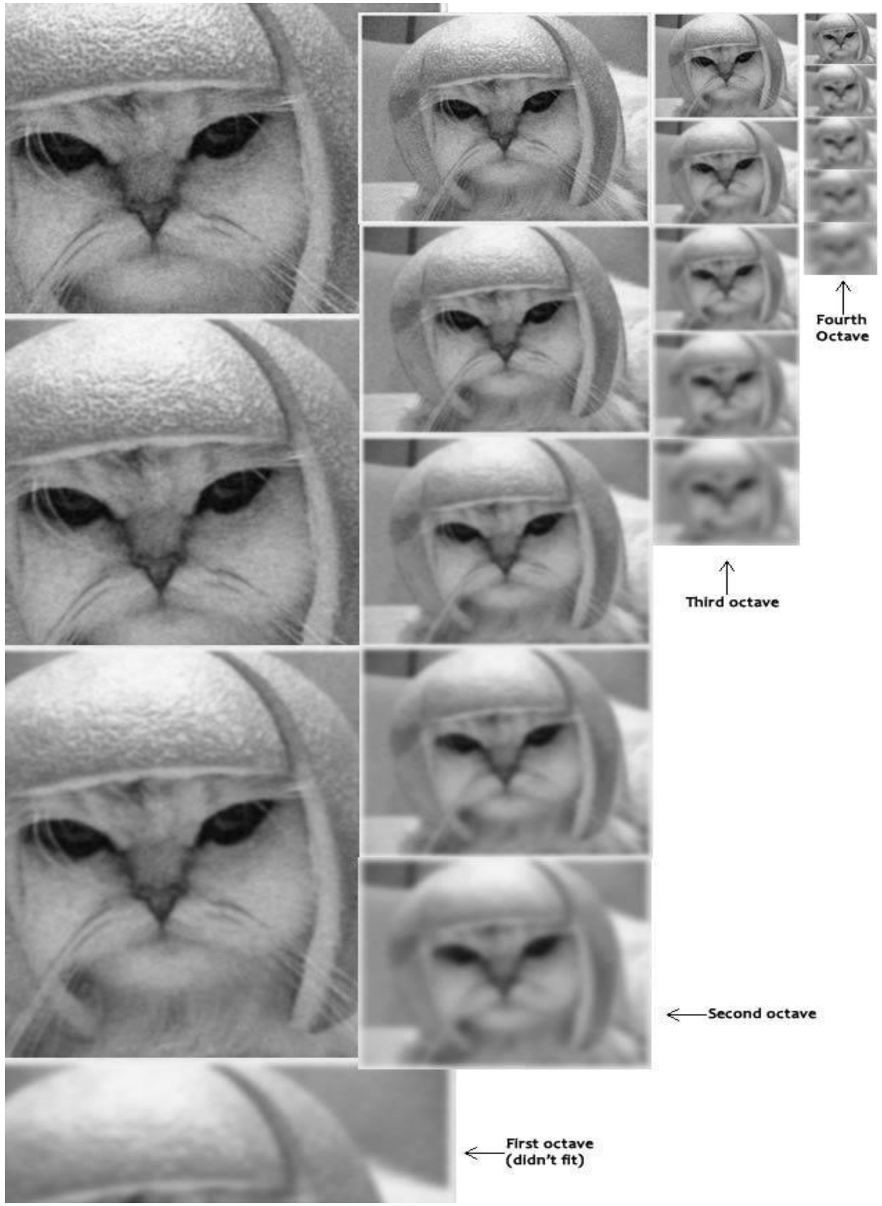


SIFT Algorithm (Construct a Scale Space)

- Real world objects are meaningful only at a certain scale
- A small object kept on a table can be easily seen
- Same object may not be prominent if seen from far
- Therefore key points are searched at multiple scales by creating a ‘scale space’
- Ensures that features are scale-independent

SIFT Algorithm (Construct a Scale Space)

- Real world objects are meaningful only at a certain scale
- A small object kept on a table can be easily seen
- Same object may not be prominent if seen from far
- Key points are searched at multiple scales by creating a ‘scale space’
- Ensures that features are scale-independent

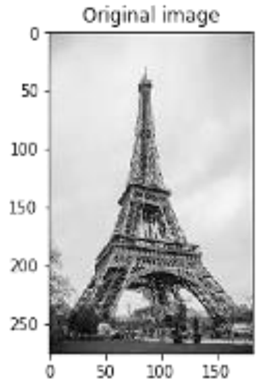


SIFT Algorithm (Construct a Scale Space)

- In each octave, images are progressively blurred using the Gaussian Blur filter
- Blurring removes texture and minor details from the image
- Information, like the shape and edges of the image exists
- Scale space is a collection of blurred images which are generated by Gaussian filter with different standard deviations
- These image are generated from a single image in an octave
- Same process is repeated for each octave

SIFT Algorithm (Construct a Scale Space)

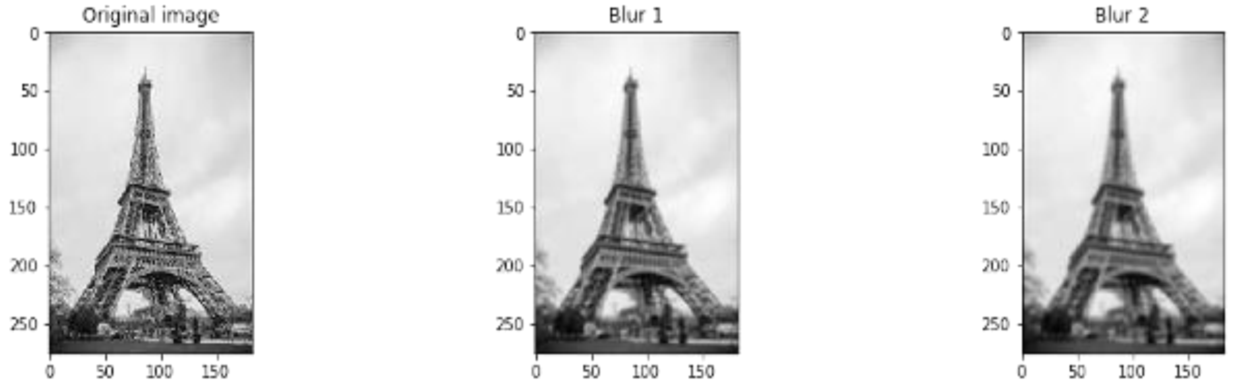
Original image of size (275, 183)



SIFT Algorithm (Construct a Scale Space)

- Create a new set of images
- Take the original image and down sample it by rate 1/2
- Reduces the resolution of image
- For each new image, create blurred versions

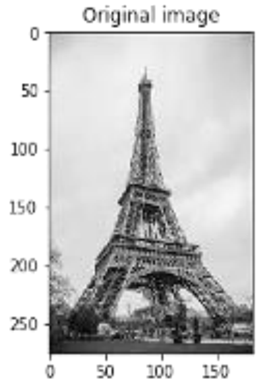
Original image of size (275, 183)

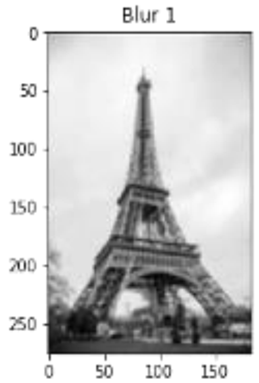


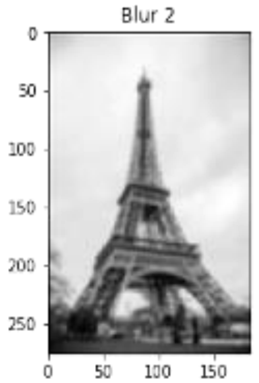
SIFT Algorithm (Construct a Scale Space)

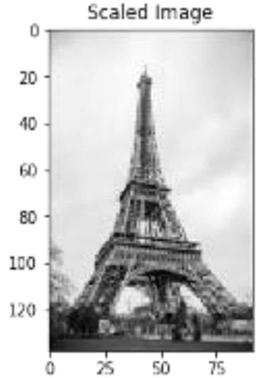
- Create a new set of images
- Take the original image and down sample it by rate 1/2
- Reduces the resolution of image
- For each new image, create blurred versions

Original image of
size (275, 183)







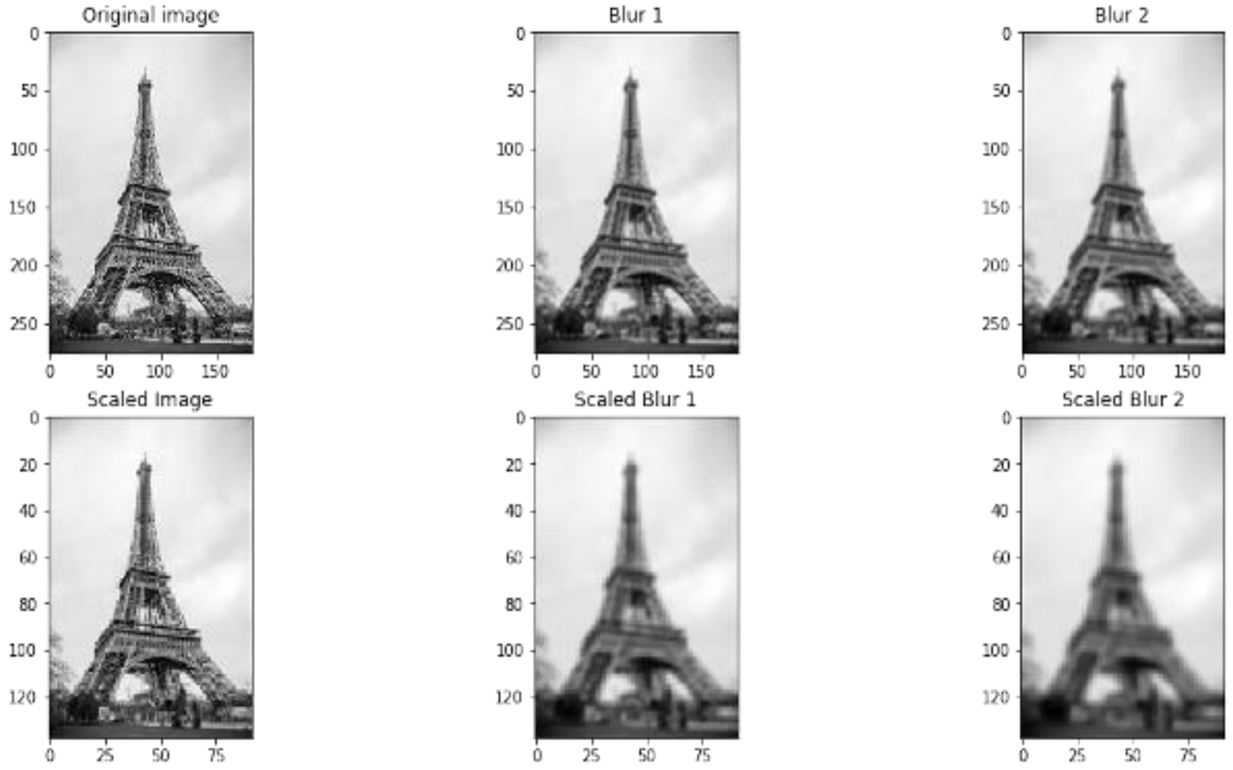


scaled image of
dimension (138, 92)

SIFT Algorithm (Construct a Scale Space)

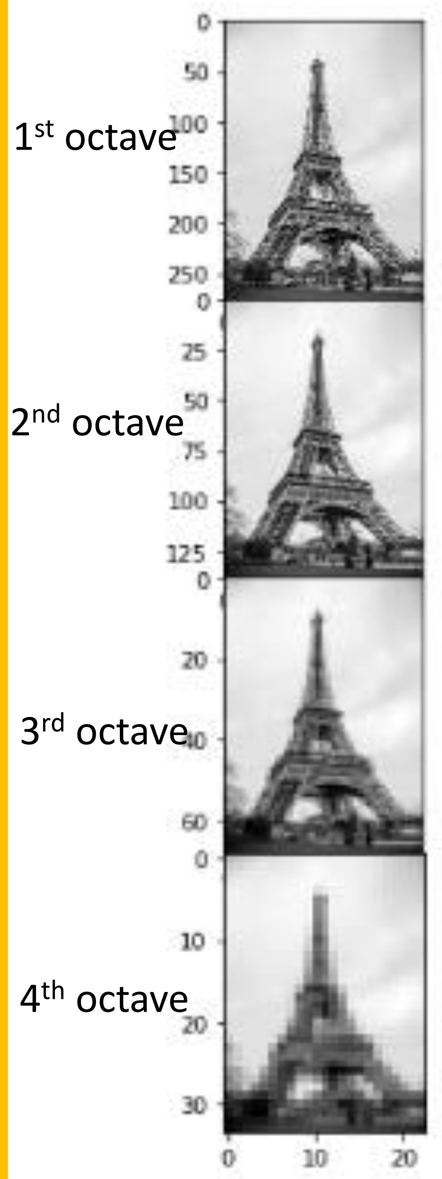
- Create a new set of images
- Take the original image and down sample it by rate 1/2
- Reduces the resolution of image
- For each new image, create blurred versions

Original image of size (275, 183)



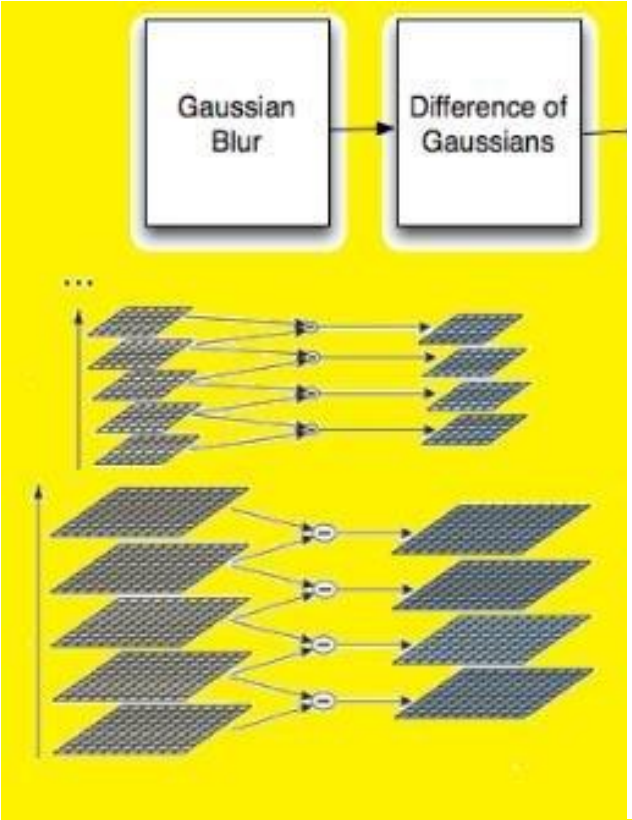
scaled image of dimension (138, 92)

SIFT Algorithm (Construct a Scale Space)



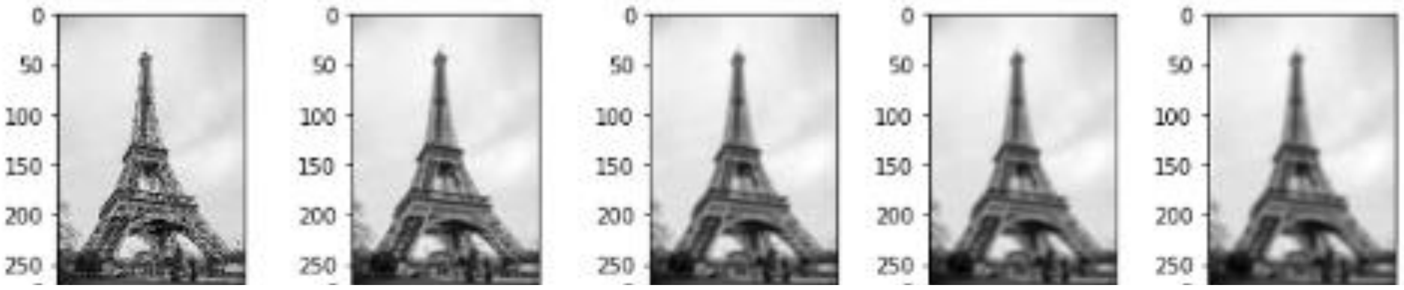
- Scale space is a collection of images having resolutions and blurring generated from a single image
- Ideal number of low resolution versions (octaves) is four
- Octave is different levels of image resolutions
- Each octave is down sampled by 2 to generate next octave to reduce image size by 1/4
- Each octave has five blurred images
- Gaussian filter of different scales (variance) blur the images

SIFT Algorithm



SIFT Algorithm (Construct a Scale Space)

1st octave

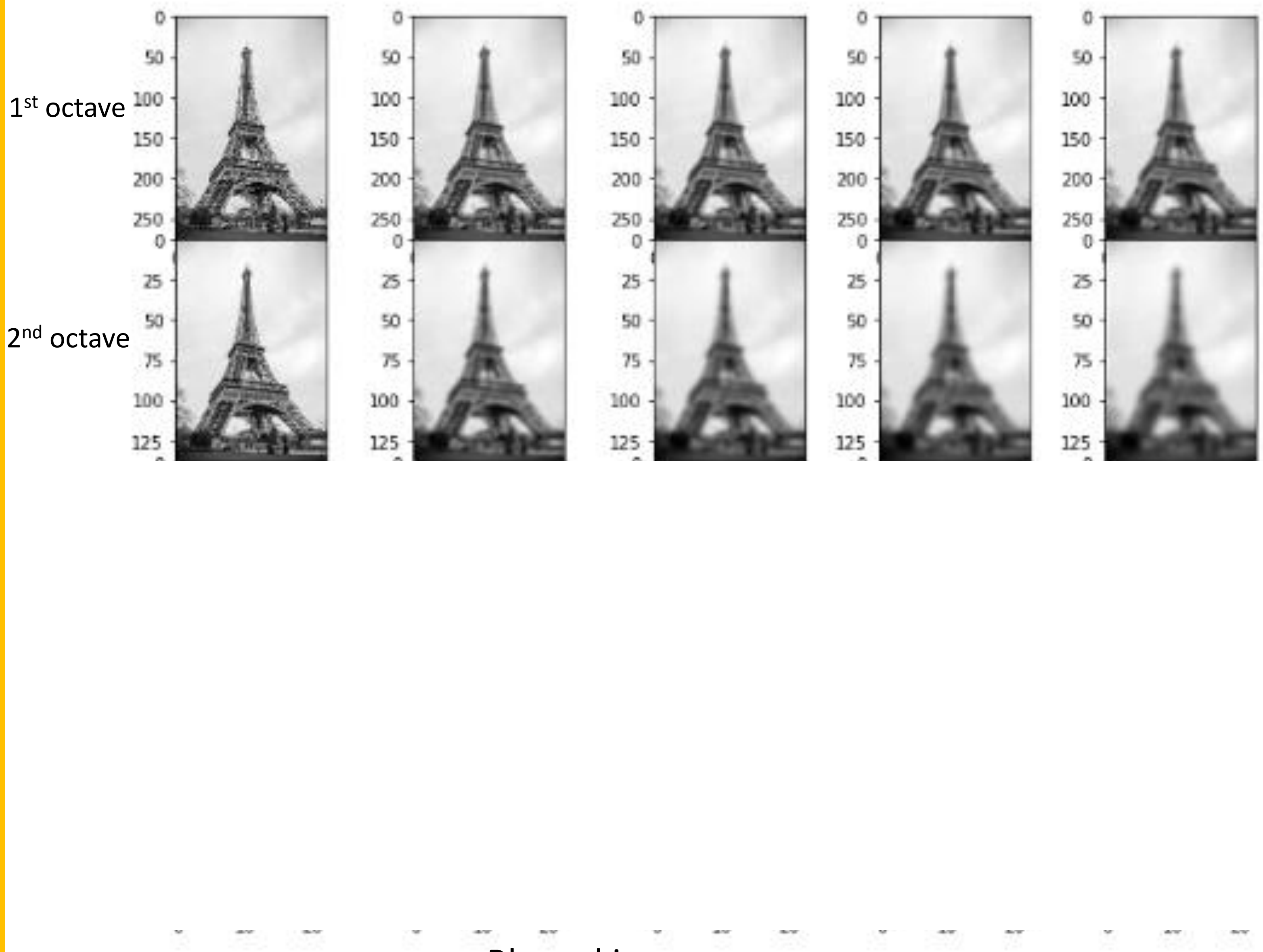


Blurred images

Filter Images using Gaussian filter of different sigma values

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

SIFT Algorithm (Construct a Scale Space)

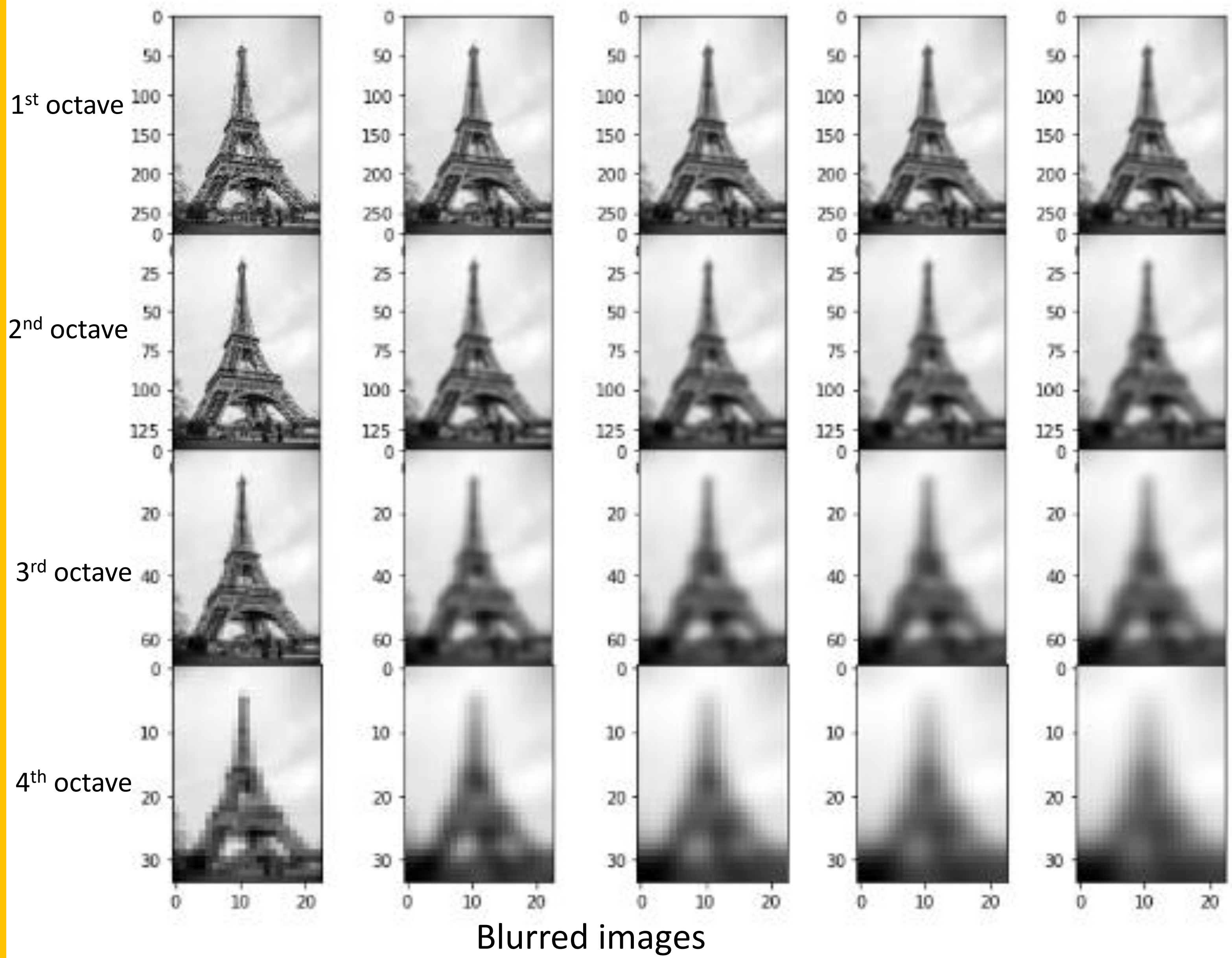


Blurred images

Filter Images using Gaussian filter of different sigma values

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

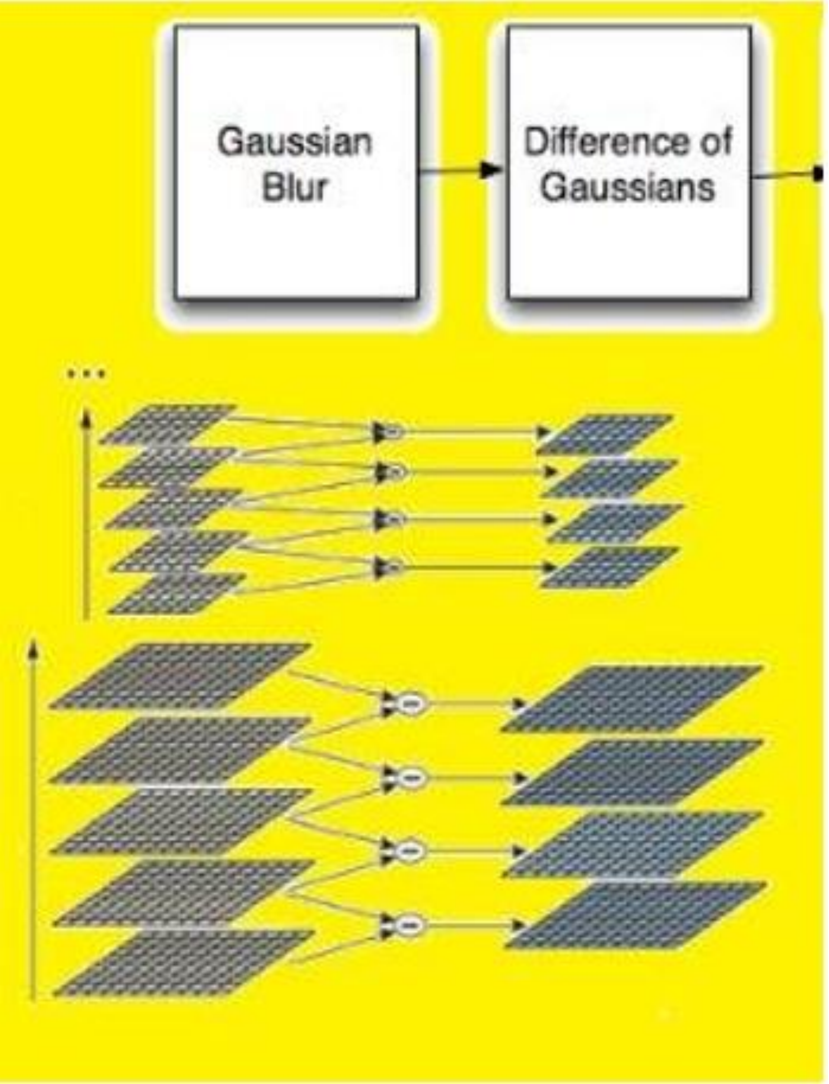
SIFT Algorithm (Construct a Scale Space)



Filter Images using Gaussian filter of different sigma values

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

SIFT Algorithm (Construct a Scale Space)

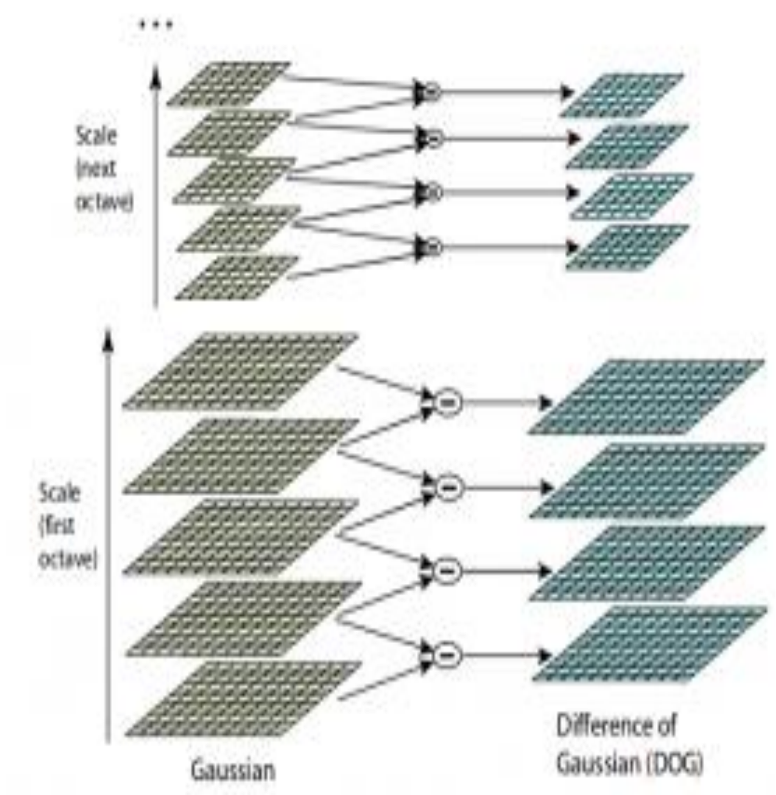


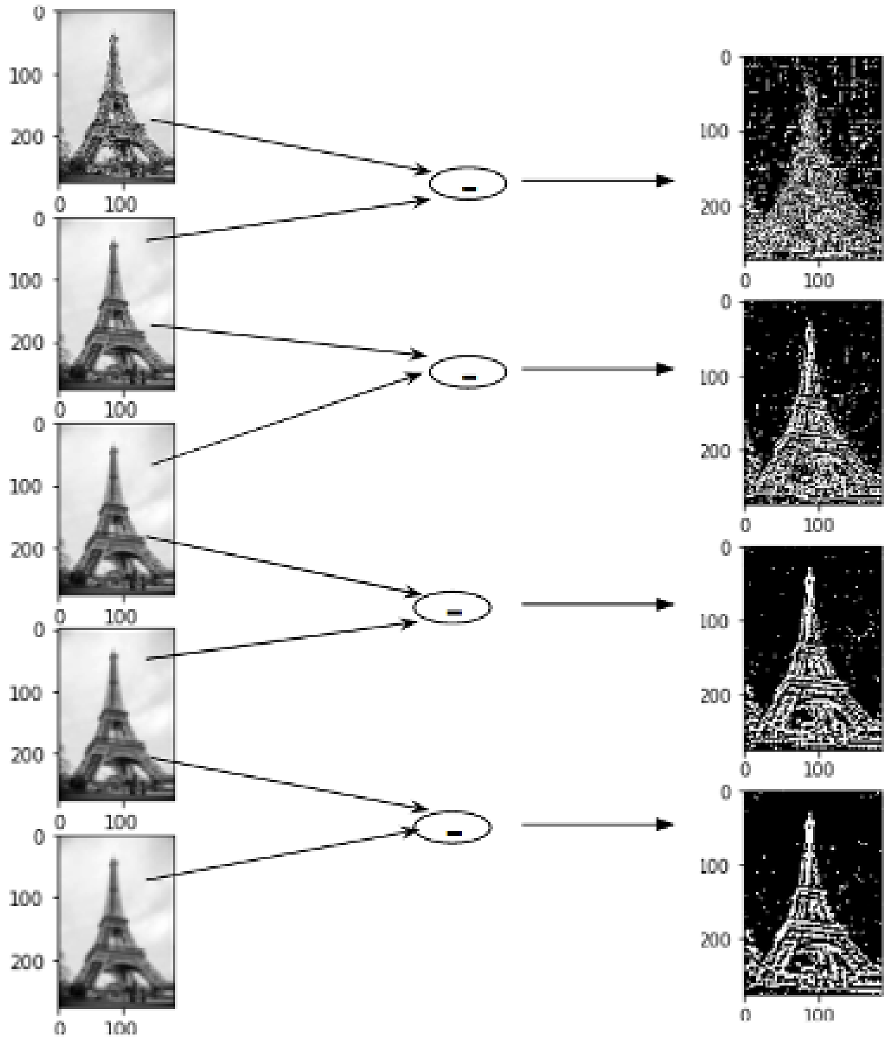
- Determine Difference of Gaussian (DoG) of two consecutive blurred images in each octave

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned}$$

- Initial value of σ is 1.6 and k = $2^{1/2}$
- For each octave, a set of DoG images are generated
- Dog enhances features (edges and corners) of image

SIFT Algorithm (Difference of Gaussian, DoG)



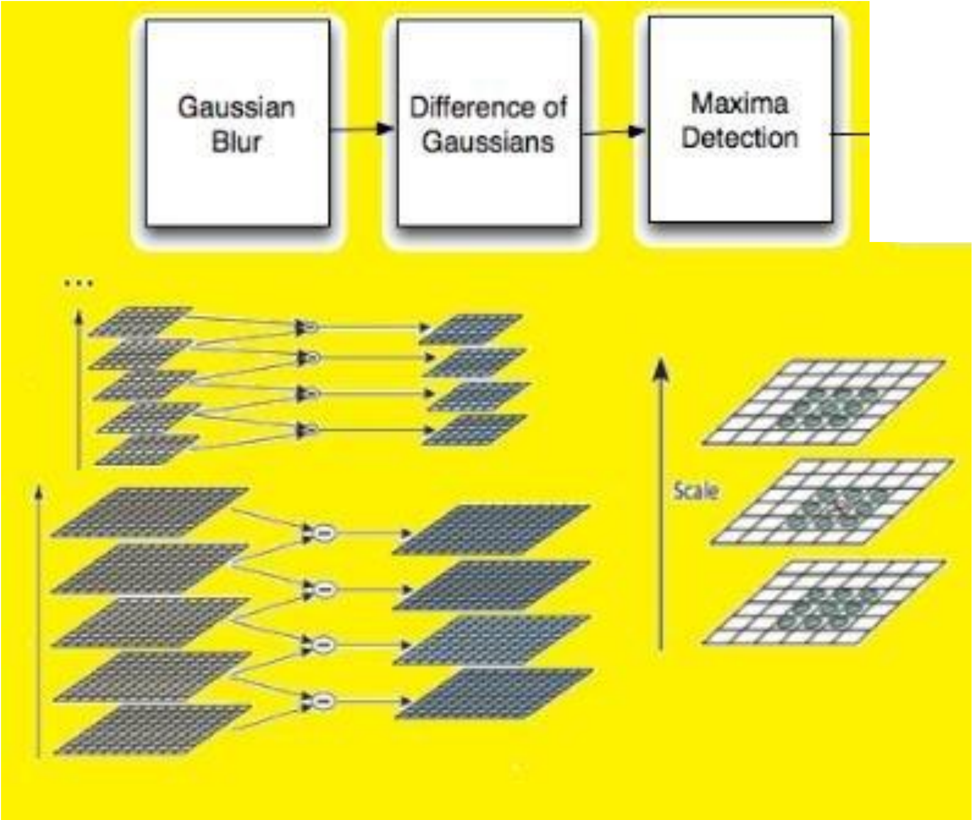


Same process is used for all the octaves

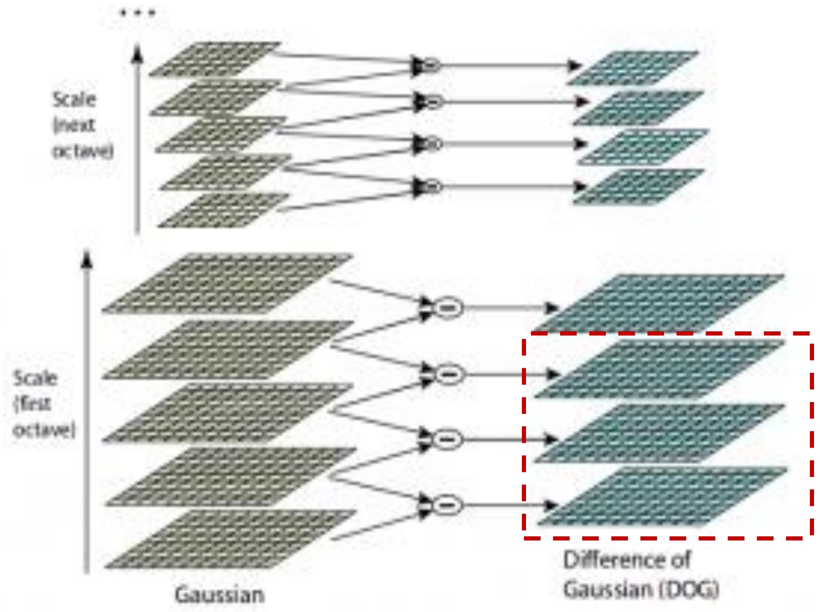
SIFT Algorithm

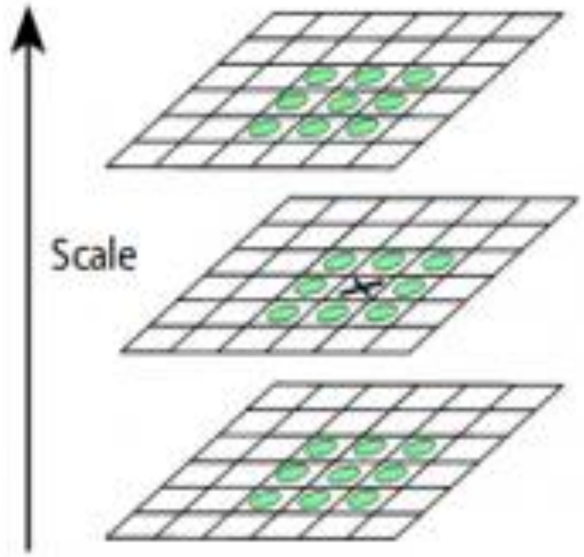
1. Construct a Scale Space:
   - Generate images over multiple scales and image locations
   - To ensures that features are scale-independent
2. Key point Localisation:
   - Select key points based on measure of stability
   - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
   - Compute best orientations for each key point region
   - To ensure that keypoints are rotation invariant
4. Keypoint Descriptor:
   - Use local image gradients at selected scale and rotation

SIFT Algorithm



SIFT Algorithm (Keypoint localization)





- Each pixel is compared with 26 other pixel values
- Pixel marked *x* is selected as a potential keypoint if it is the highest positive or lowest negative among 26 neighbors

Three DoGs of first octave

SIFT Algorithm (Keypoint localization)

- Discard weak key points
  - Normalize DoGs to get values in 0-1 range
  - Set threshold to 0.03
  - For DOG(x), x is coordinates of extrema (minima/maxima)
    - If DOG(x) < Threshold, discard it
    - Else retain it
  - Construct Hessian matrix at each potential keypoint to check whether these points are good key points

$$H = \sum \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

I_{xx} and I_{yy} are second order derivative in x and y directions

I_{xy} is first order derivative in x direction and then in y direction

SIFT Algorithm (Keypoint localization)

- Discard weak key points

$$H = \sum \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

- $\text{Det}(H) = I_{xx} I_{yy} - I_{xy}^2$
  $\text{Trace}(H) = I_{xx} + I_{yy}$

  - Select key point if

$$\frac{\text{Tr}^2(\mathbf{H})}{\text{Det}(\mathbf{H})} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} < \frac{(r+1)^2}{r}$$

  - For SIFT, r = 10
  - $\text{Tr}^2(H)/\text{Det}(H) < 12.1$

SIFT Algorithm (Keypoint localization)

233x189 image

832 DOG extrema (maxima/minima)

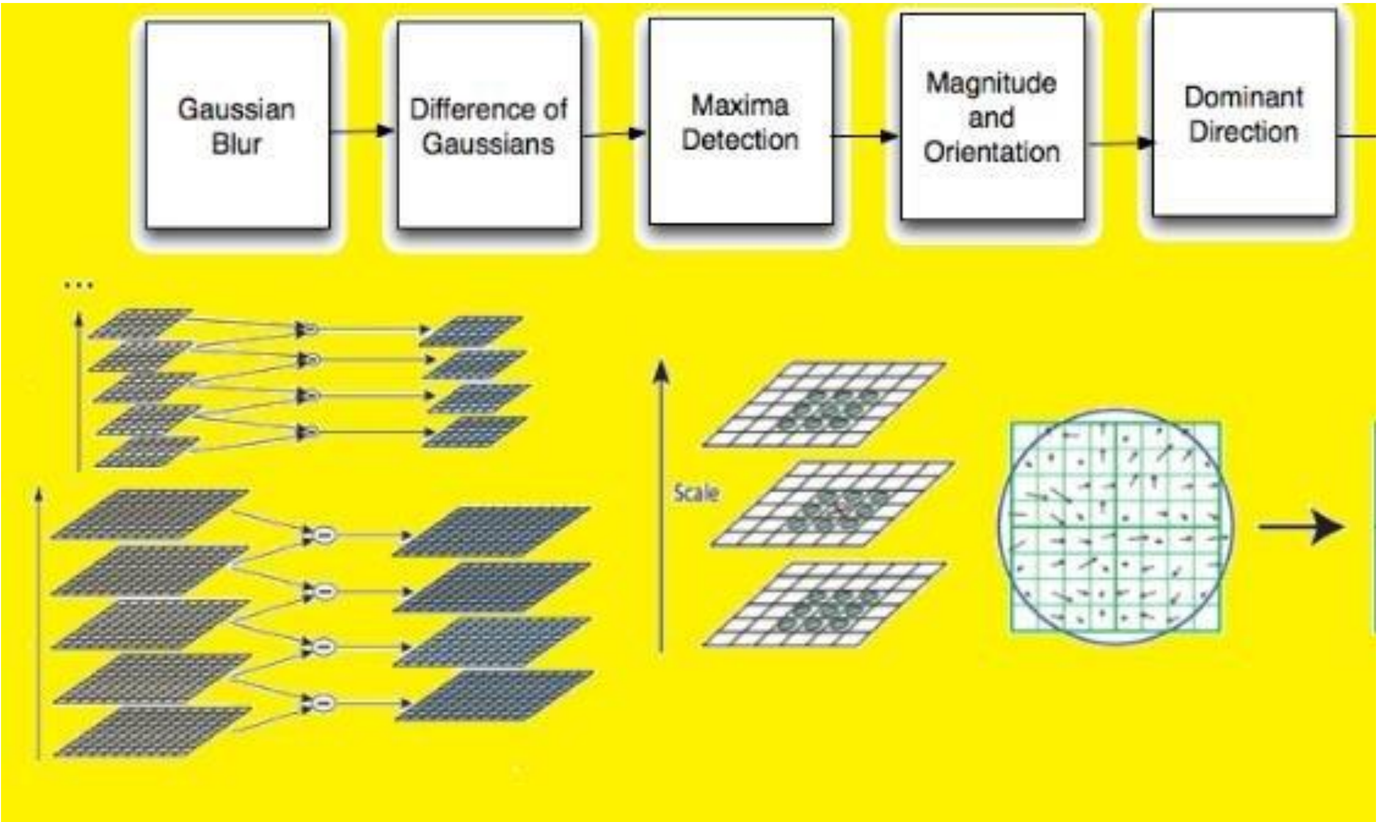
729 after peak value threshold (=0.03)

536 after edge point removal using ‘r’ parameter

SIFT Algorithm

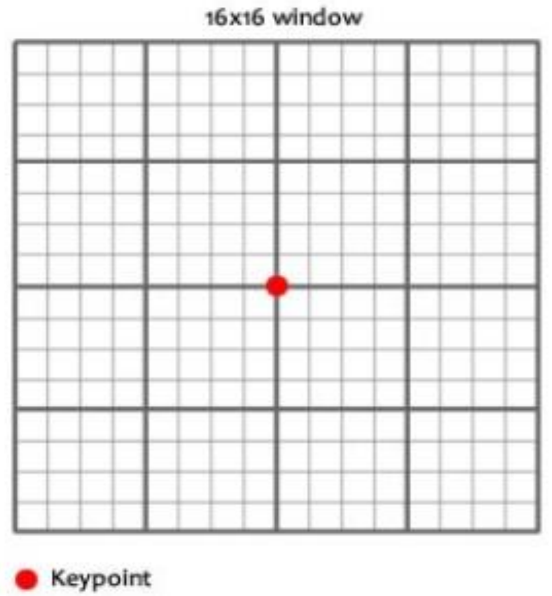
1. Construct a Scale Space:
   - Generate images over multiple scales and image locations
   - To ensures that features are scale-independent
2. Key point Localisation:
   - Select key points based on measure of stability
   - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
   - Compute best orientations for each key point region
   - To ensure that keypoints are rotation invariant
4. Keypoint Descriptor:
   - Use local image gradients at selected scale and rotation

SIFT Algorithm



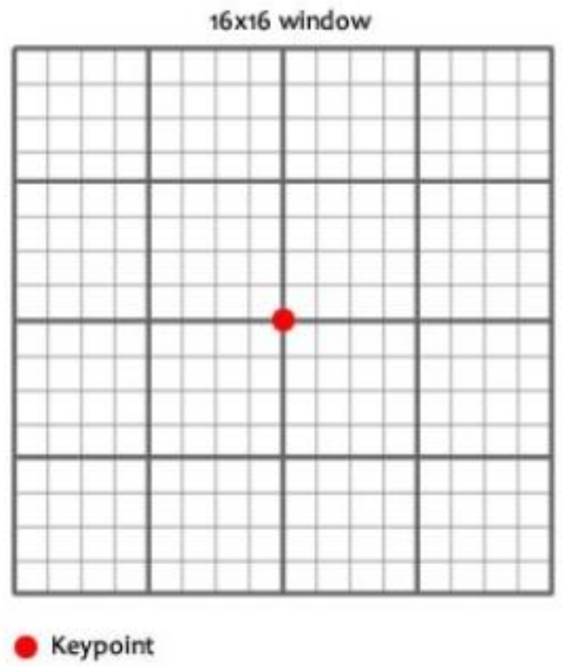
SIFT Algorithm (Orientation Assignment)

- Each keypoint has a
  - location in image
  - Scale at which key point was selected
  - Orientation
- Use a 16x16 pixels (window) around each key point in the original image

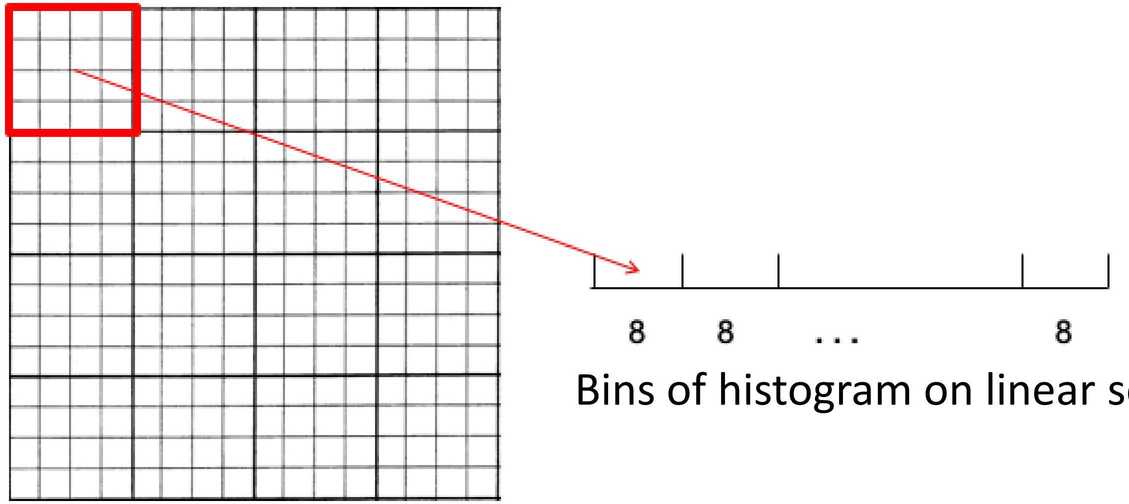


SIFT Algorithm (Orientation Assignment)

- Window is divided into 16 cells of 4x4 pixels
- Determine gradients of each pixel in X and Y direction
- Determine magnitudes and orientations of gradients for each pixel of each cell
- Divide orientation (0 to 360^o) into 8 bins
- Draw Histogram on circular scale

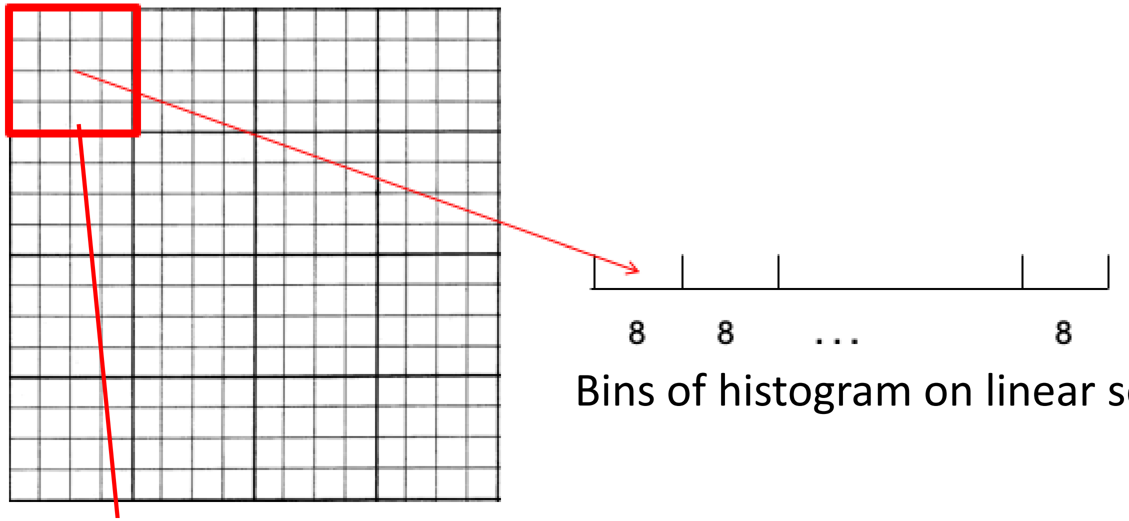


SIFT Algorithm (Orientation Assignment)

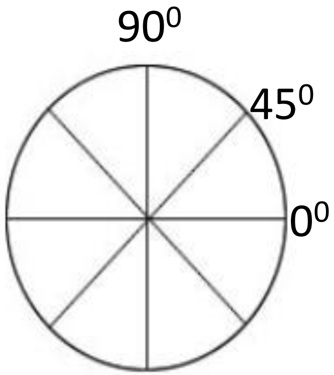


Bins of histogram on linear scale

SIFT Algorithm (Orientation Assignment)

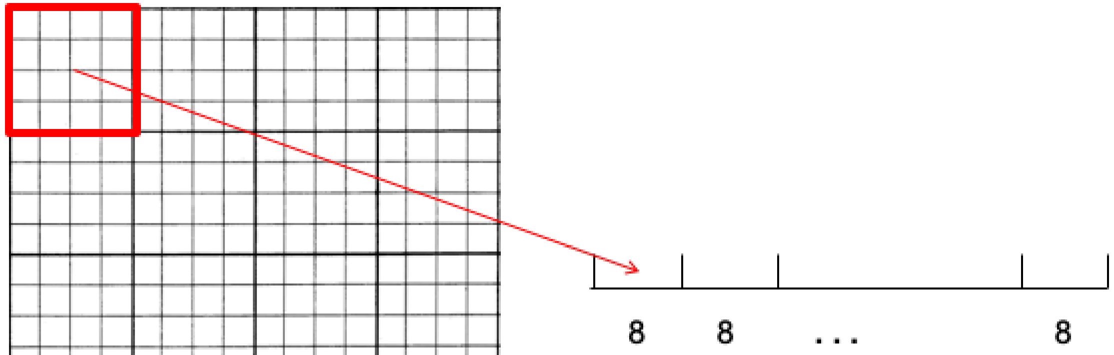


Bins of histogram on linear scale

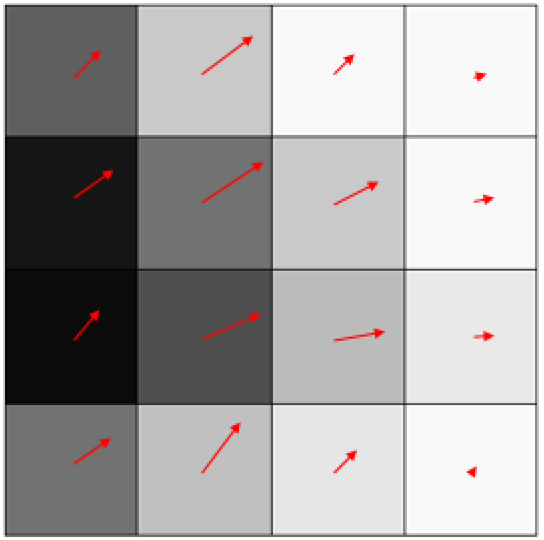


Bins of histogram on circular scale

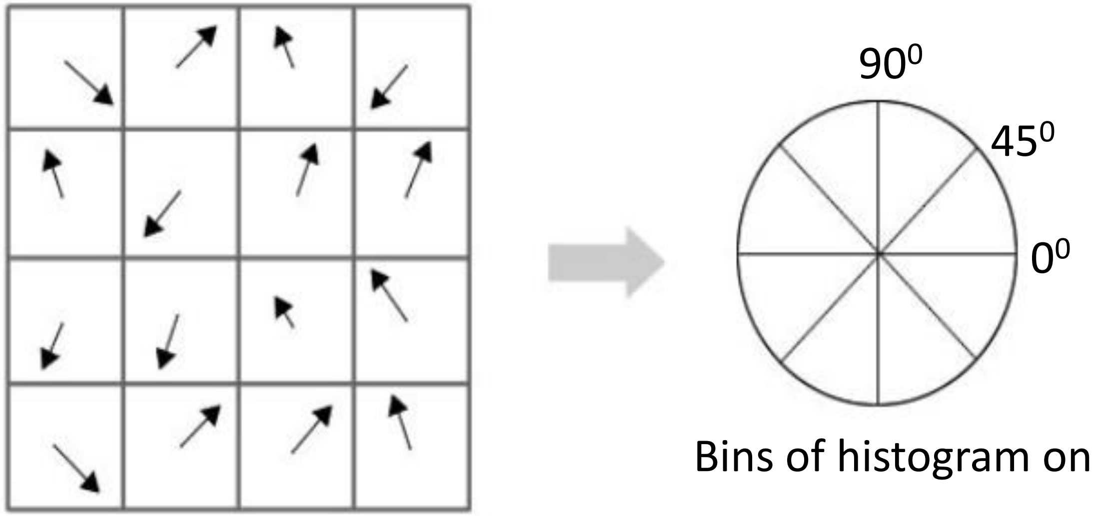
SIFT Algorithm (Orientation Assignment)



Bins of histogram on linear scale



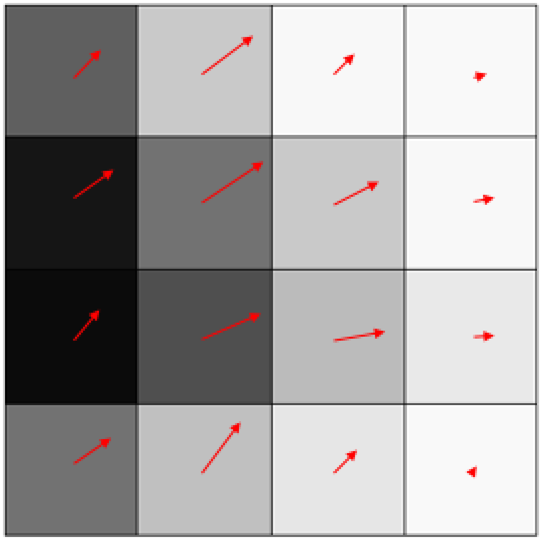
Orientations in each of the 16 pixels of the cell



Bins of histogram on circular scale

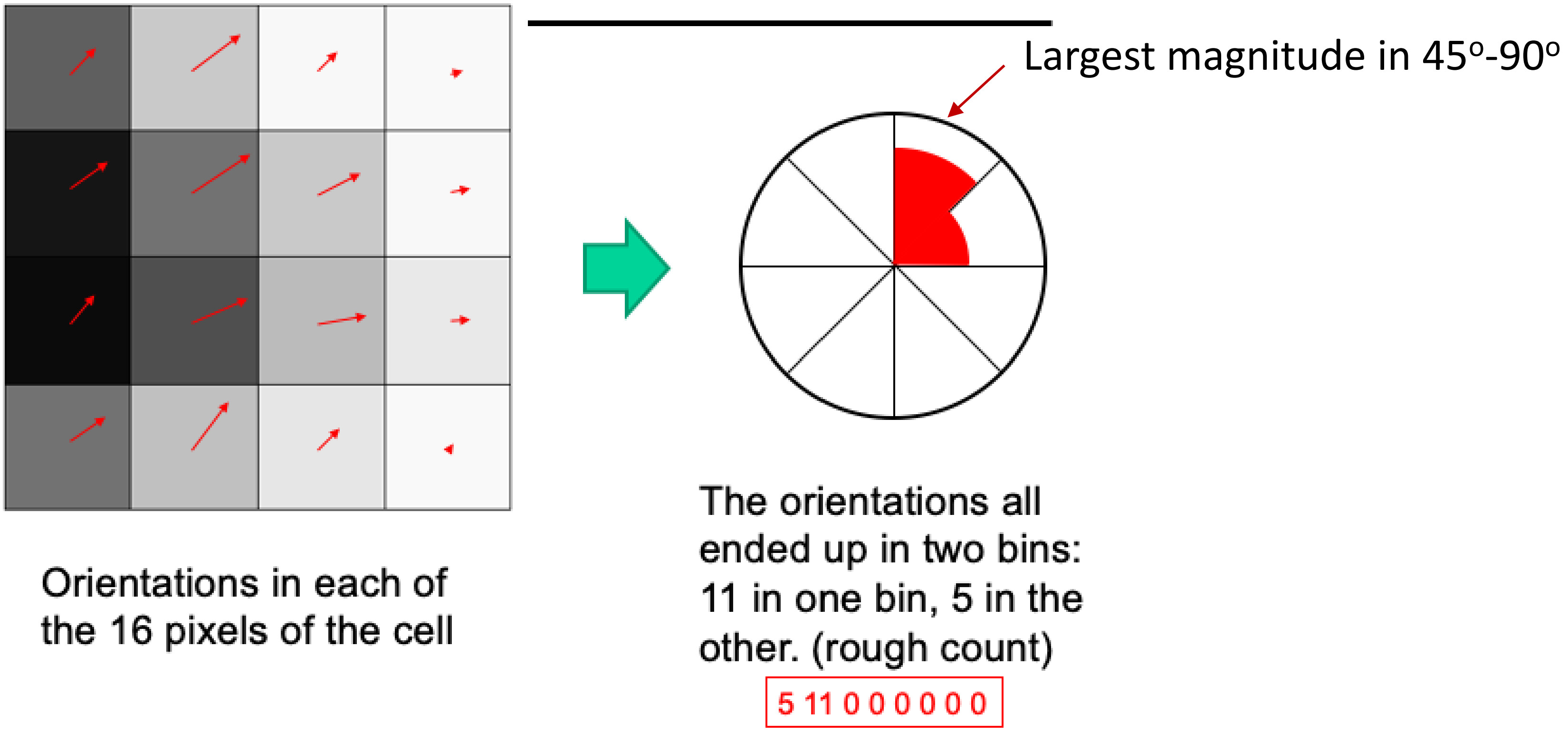
Magnitude and angle of gradients

SIFT Algorithm (Orientation Assignment)



Orientations in each of
the 16 pixels of the cell

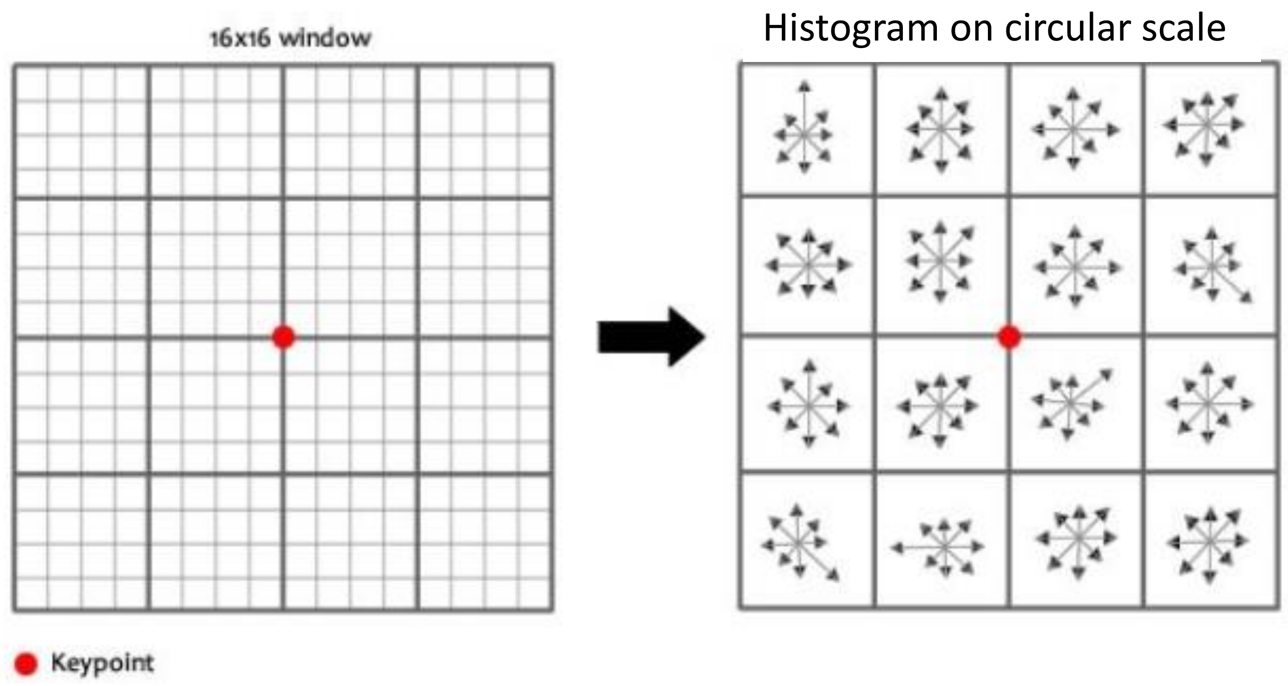
SIFT Algorithm (Orientation Assignment)



Direction of this cell is 45º-90º

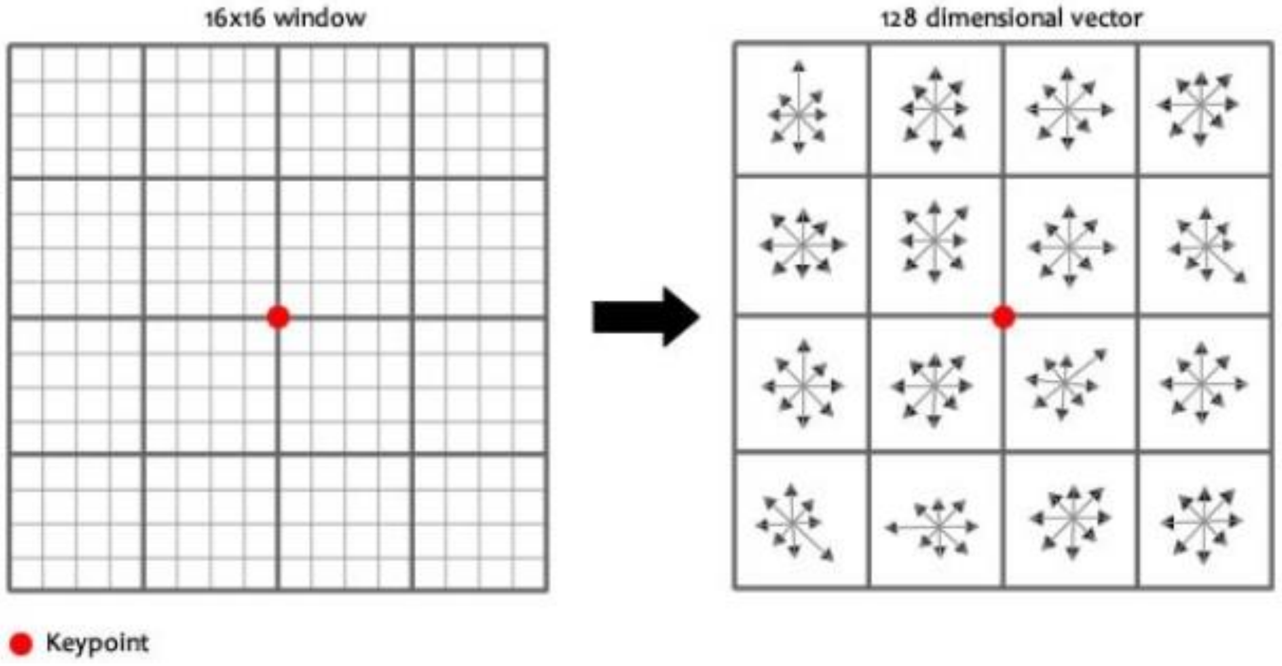
SIFT Algorithm (Orientation Assignment)

- Length of each arrow of histogram is proportional to the accumulated magnitudes in the bin



SIFT Algorithm (Orientation Assignment)

- Each cell has 8 orientations (8 bins/ cell)
- 16 cells and 8 orientations = 16x8 = 128 bins with magnitude of gradients as bin value
- Feature vector of each key point contains 128 elements

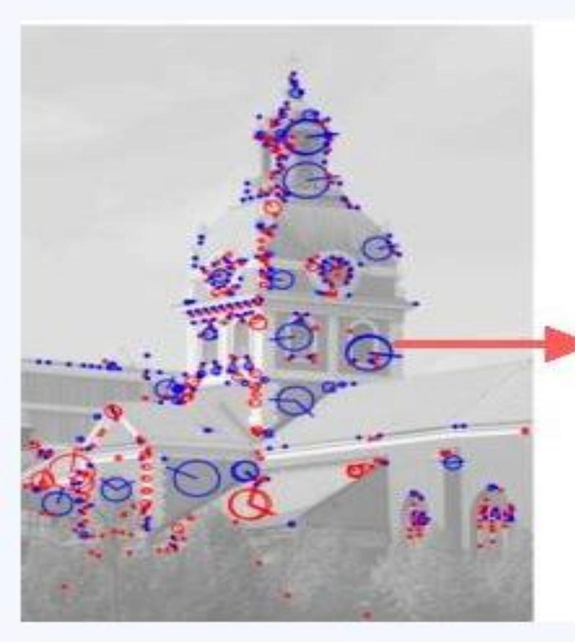


SIFT Algorithm

1. Construct a Scale Space:
    - Generate images over multiple scales and image locations
    - To ensures that features are scale-independent
2. Key point Localisation:
    - Select key points based on measure of stability
    - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
    - Compute best orientations for each key point region
    - To ensure that keypoints are rotation invariant
4. Keypoint Descriptor:
    - Use local image gradients at selected scale and rotation

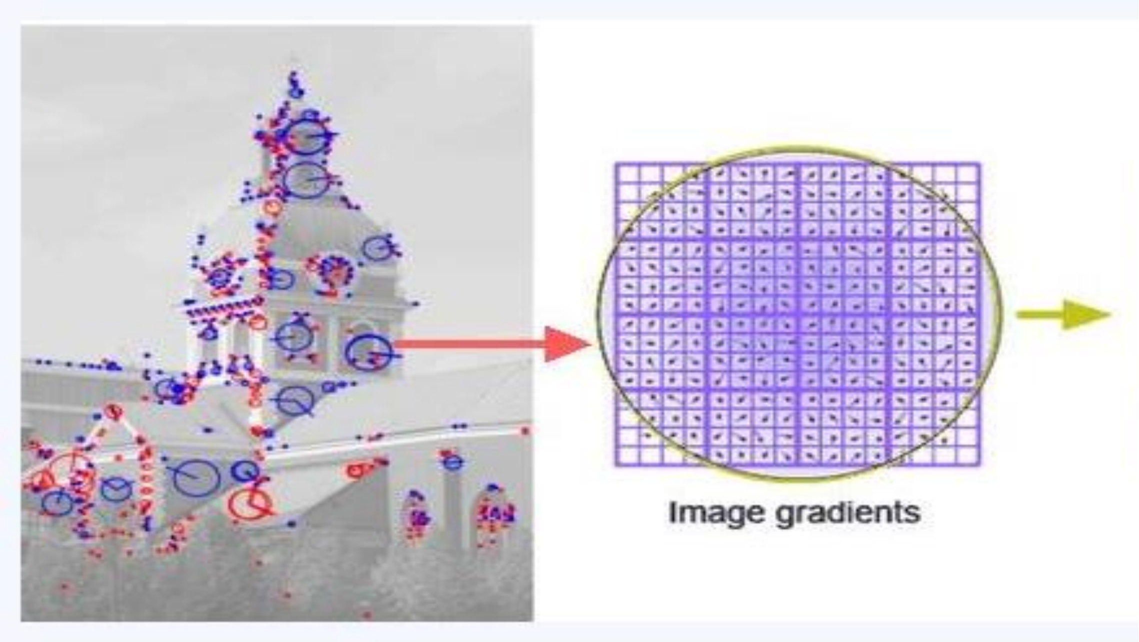
SIFT Algorithm (Keypoint Descriptor)

- Use Gaussian-weighted circular window with σ equals to 1.5 times the scale of the image at which key point is selected
- Gaussian window is used to give less emphasis to gradients that are far from the key point



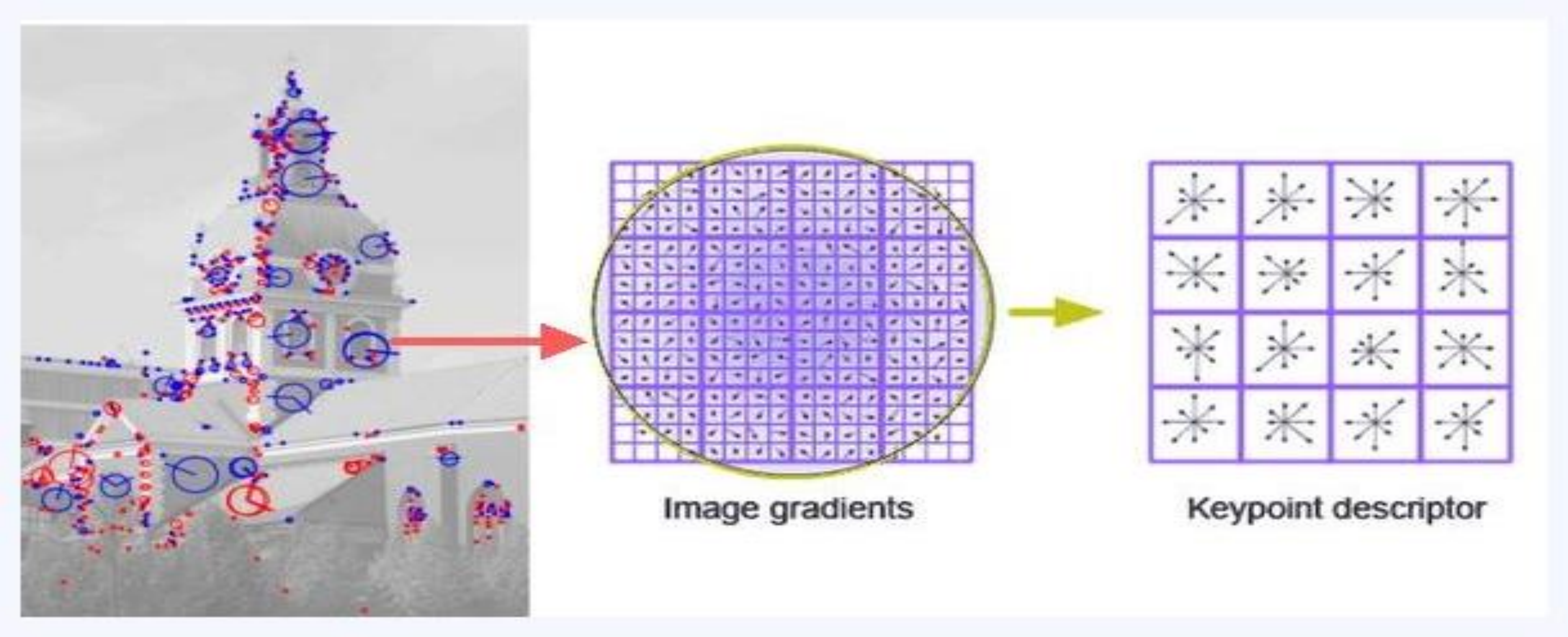
SIFT Algorithm (Keypoint Descriptor)

- Use Gaussian-weighted circular window with σ equals to 1.5 times the scale of the image at which key point is selected
- Gaussian window is used to give less emphasis to gradients that are far from the key point



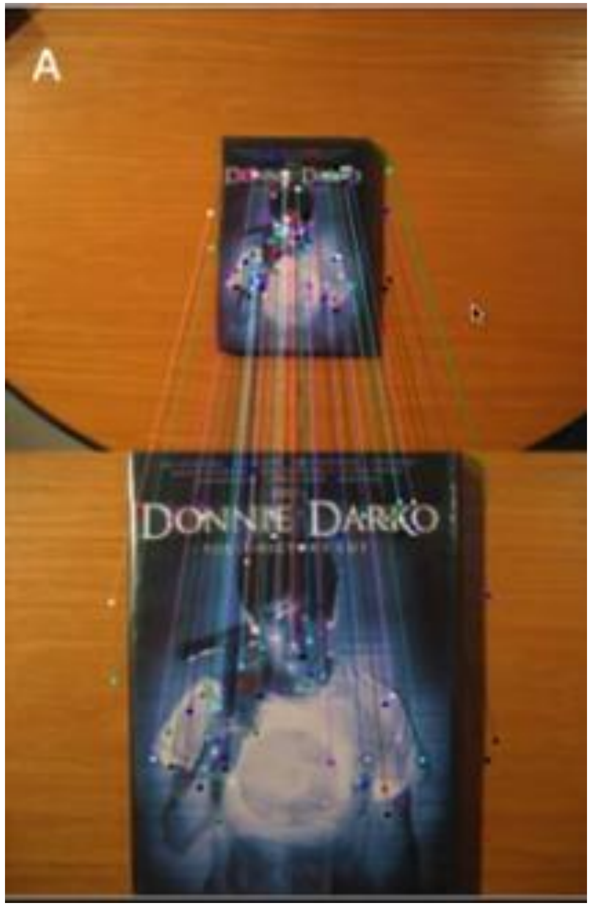
SIFT Algorithm (Keypoint Descriptor)

- Use Gaussian-weighted circular window with σ equals to 1.5 times the scale of the image at which key point is selected
- Gaussian window is used to give less emphasis to gradients that are far from the key point

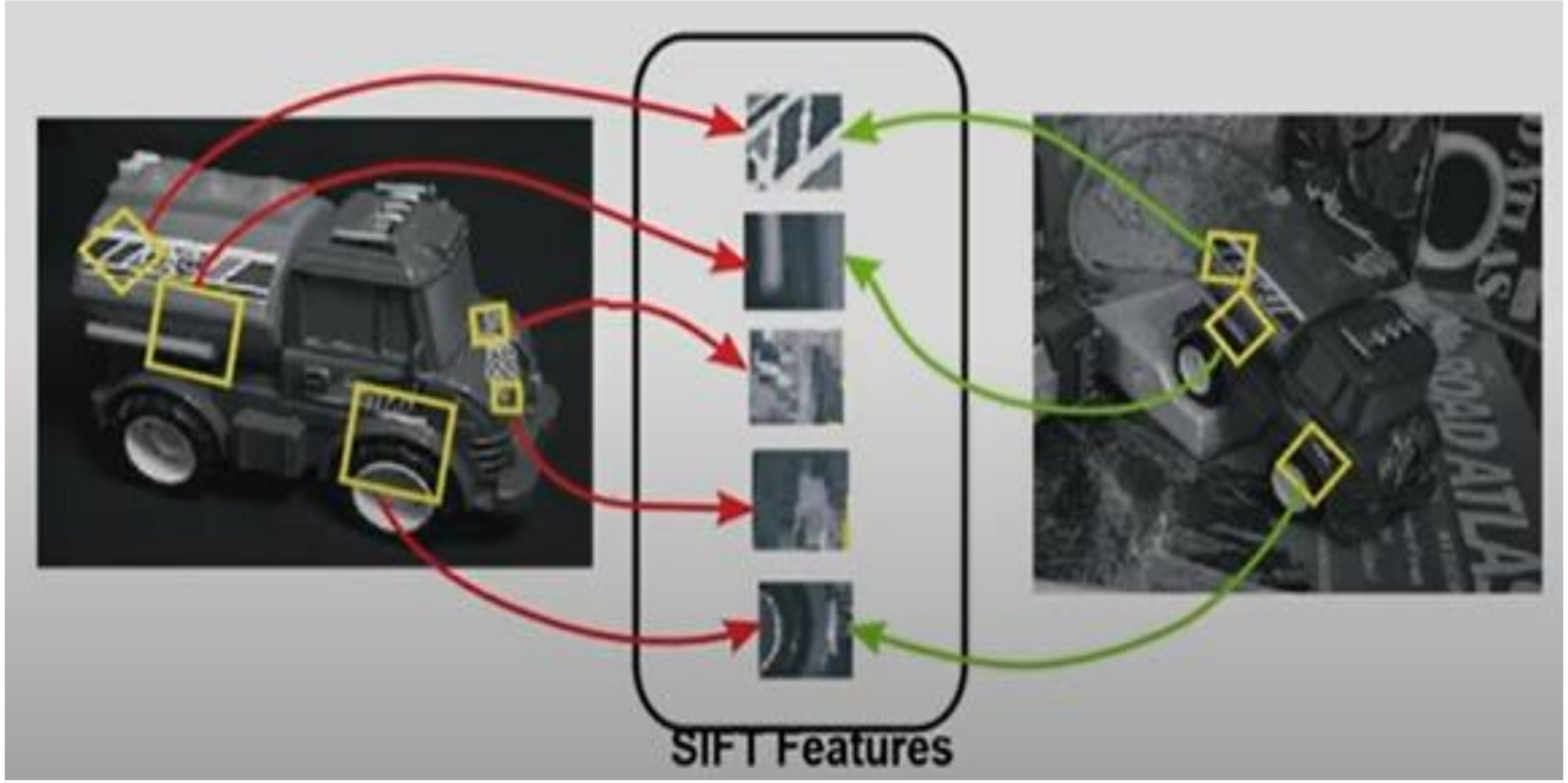


SIFT Descriptor for image matching

Keypoints of two images are matched by identifying their nearest neighbors



Examples: Image matching using SIFT



Invariant to all rotation, illumination and scaling changes

Integral Image

- Used as a quick way of calculating the sum of pixel values in part of the given image
- Useful for box type (most of the values are same) filters
- Box filter on integral image offers fast computation
- Element of an integral image, $I_{Int}(x,y)$ at a location (x,y) represents

  sum of all pixels in the input image I(x,y) within a rectangular region formed by the origin (0,0) and (x,y)

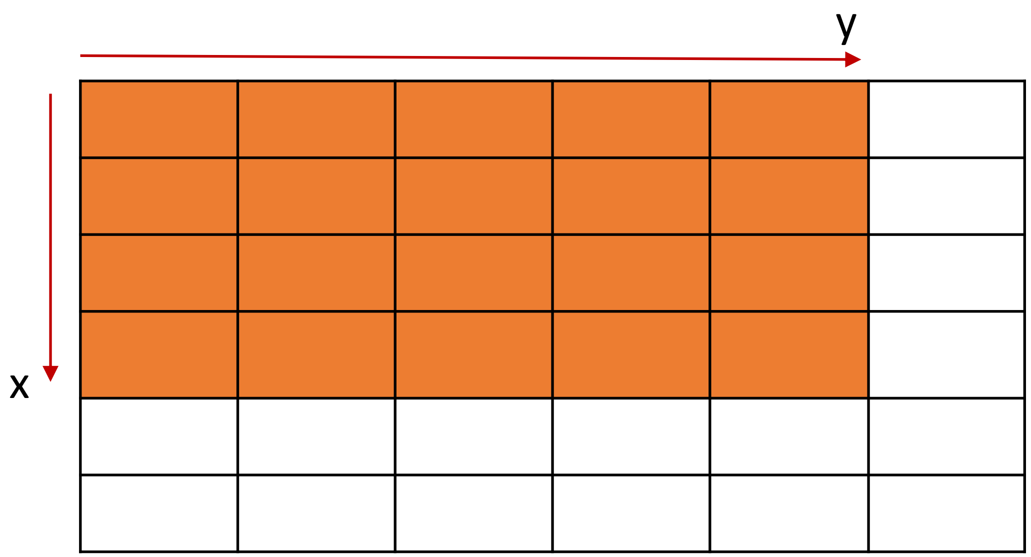
$$I_{Int}(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j)$$

Integral Image

- Used as a quick way of calculating the sum of pixel values in part of the given image
- Useful for box type (most of the values are same) filters
- Box filter on integral image offers fast computation
- Element of an integral image, $I_{Int}(x,y)$ at a location (x,y) represents

  sum of all pixels in the input image I(x,y) within a rectangular region formed by the origin (0,0) and (x,y)

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



Integral Image

- Element of an integral image $I_{Int}(x,y)$ at a location (x,y) represents the sum of all pixels in the input image I(x,y) within a rectangular region formed by the origin (0,0) and (x,y)

1	5
2	4

Input Image

0	0	0
0	1	5
0	2	4

Add row and column of zeroes

0	0	0
0	1	
0		

Part of Integral Image

0	0	0
0	1	6
0	3	12

Integral Image

1	6
3	12

Integral Image

Example: Integral Image

- Cell/elements of integral image are not pixels
- Elements are sum of all values in a corresponding area of the original image
- Consider an 8-bit image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

		?							

Integral Image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

		51							

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

							80		
		51							

Integral Image

Original Image

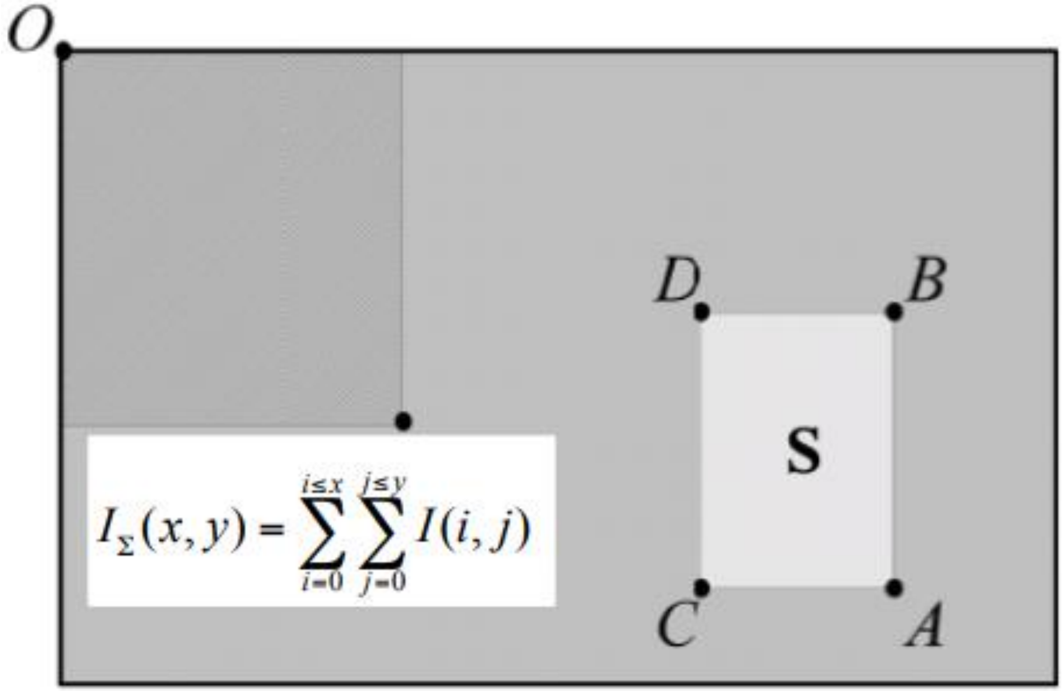
0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Use of Integral Image

Use of Integral Image



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

Integral Image

Determine sum of elements within a region

A, B, C and D are elements of Integral image at the corners of region

sum, S= A-B-C+D

Use of Integral Image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Add elements within red box

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Use of Integral Image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	$\boxed{215}$	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	$\boxed{80}$	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	$\boxed{215}$	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Use of Integral Image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Use of Integral Image

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

A

B

D

C

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Find sum of element

Use of Integral Image

Original Image

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Integral Image

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

- Sum of elements for region
- S = 215-80-72+20 = 83
- Only four numbers are required for computation if integral image is considered
- This process drastically reduces the complexity of computation of S even if area of region is large

Ex: Use of Integral Image

0	1	1
0	1	1
0	1	1

Box Filter

1	1	1
1	**<u>1</u>**	1
1	1	1

Input Image

1	1	1
1	1	1
1	1	1

Input Image

1	2	3
2	4	6
3	6	9

Integral Image

1_0	1_1	1_1
1_0	$\underline{\mathbf{1}}_1$	1_1
1_0	1_1	1_1

Filter on Image

1	2	3
2	4	6
3	6	9

Integral Image

Filter response = 9-0-3+0

= 6

	<u>6</u>	

Filtered Image

Even if size of filter increases, number of computations (3 additions/subtractions) does not increase

Comparison of SIFT and SURF

SIFT

- High dimensionality
- Reduction in dimensionality leads to reduction in accuracy
- Approximates Laplacian of Gaussian with DoG
- Uses determinant and trace of Hessian matrix

SURF

- It is a speeded-up version of SIFT (Fast and robust algorithm)
- Approximates LoG with Box Filter
- Convolution with box filter using integral images speeds up the process
- Convolution can be done in parallel for different scales
- Thus enabling real-time applications such as tracking and object recognition
- Uses determinant of Hessian matrix for both scale and location not trace

Speeded Up Robust Features (SURF)

Detector

- Based on Hessian matrix
- Uses box filter which is approximation of Gaussian
- Uses integral image for filtering images

Descriptor

- Describes distribution of Haar wavelet responses for the neighbourhood of interest point
- Length of descriptor is 64
- Uses sign of Laplacian as a first step for matching key points

Speeded Up Robust Features (SURF) algorithm

Detector

1. Construct Hessian matrix using box filters at each pixel and determine keypoints
2. Increase the size of box filters and repeat step 1
3. For each key point, select points within radius of 6s, where s is standard deviation of filter for key point
4. Apply Gaussian filter of 2.5s on key point
5. Use Haar wavelet of size 4s to determine magnitude and direction of points
6. Draw histogram of orientation for 6 bins to identify orientation of key point

Descriptor

1. Around each key point, choose points within region of square with size (20s, 20s)
2. Divide region into 16 sub-regions
3. For each sub-region, determine vector of length 4
4. Length of descriptor vector for region is 64

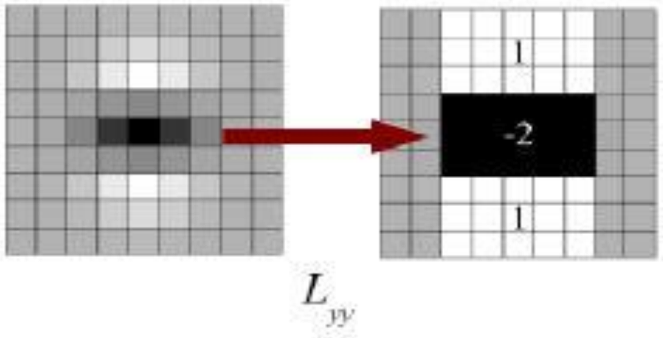
SURF (Key Point Detection)

- Detection of interest points is based on Hessian matrix
- For each pixel, determinant of Hessian matrix is calculated
- Computation requires constant time irrespective of size of filters as integral image is used for computation
- For a point p, in an image I, and with scale, σ
- Filter point, p at I(x,y) with Gaussian filter of standard deviation, σ
- Approximate Gaussian filter with Box filter

Gaussian filters as box filters

- Use discretized version of Gaussian filters (with integer values)
- Approximate discretized Gaussian filters to construct Box filters

Discretized Box



Box Filter for L_{yy}

0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0

Box Filter for L_{xx}

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Gaussian box filters of σ=1.2

Filter response of box filters

Determine Hessian matrix for the given image

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Image

Filter response of box filters

- Determine integral image

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

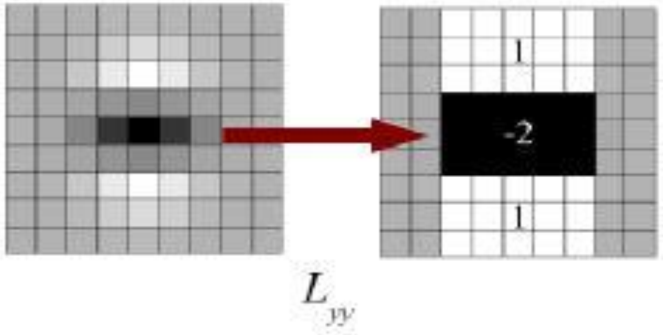
Image

1	2	3	4	5	6	7	8	9	10	11
2	4	6	8	10	12	14	16	18	20	22
3	6	10	14	18	22	26	30	34	37	40
4	8	14	20	26	32	38	44	50	54	58
5	10	18	26	34	42	50	58	66	71	76
6	12	22	32	42	52	62	72	82	88	94
7	14	26	38	50	62	74	86	98	105	112
8	16	30	44	58	72	86	100	114	122	130
9	18	34	50	66	82	98	114	130	139	148
10	20	37	54	71	88	105	122	139	149	159
11	22	40	58	76	94	112	130	148	159	170

Integral Image

Filter response of box filters

- Determine Hessian matrix for the given image



Gaussian box filters of σ=1.2

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Image

Filter response of box filters

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Lyy filter on Image

Filter response of box filters

Black = -2, White = 1, Grey = 0

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Filter on Image

Lyy?

1	2	3	4	5	6	7	8	9	10	11
2	4	6	8	10	12	14	16	18	20	22
3	6	10	14	18	22	26	30	34	37	40
4	8	14	20	26	32	38	44	50	54	58
5	10	18	26	34	42	50	58	66	71	76
6	12	22	32	42	52	62	72	82	88	94
7	14	26	38	50	62	74	86	98	105	112
8	16	30	44	58	72	86	100	114	122	130
9	18	34	50	66	82	98	114	130	139	148
10	20	37	54	71	88	105	122	139	149	159
11	22	40	58	76	94	112	130	148	159	170

Integral Image

Filter response of box filters

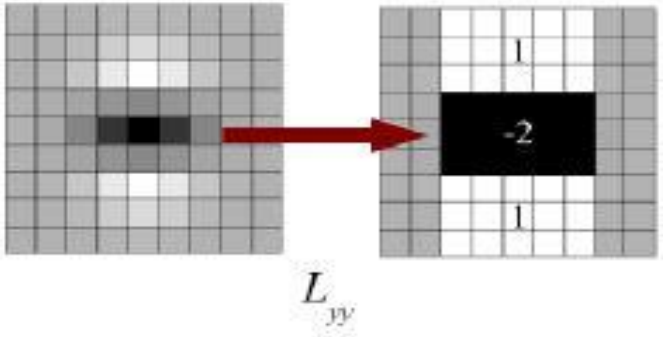
L_{yy}= -2(62-26-12+6)+(26-6) + (98-62-18+12)
= -10

Black = -2, White = 1, Grey = 0

1	2	3	4	5	6	7	8	9	10	11
2	4	6	8	10	12	14	16	18	20	22
3	6	10	14	18	22	26	30	34	37	40
4	8	14	20	26	32	38	44	50	54	58
5	10	18	26	34	42	50	58	66	71	76
6	12	22	32	42	52	62	72	82	88	94
7	14	26	38	50	62	74	86	98	105	112
8	16	30	44	58	72	86	100	114	122	130
9	18	34	50	66	82	98	114	130	139	148
10	20	37	54	71	88	105	122	139	149	159
11	22	40	58	76	94	112	130	148	159	170

Integral Image

Filter response of box filters

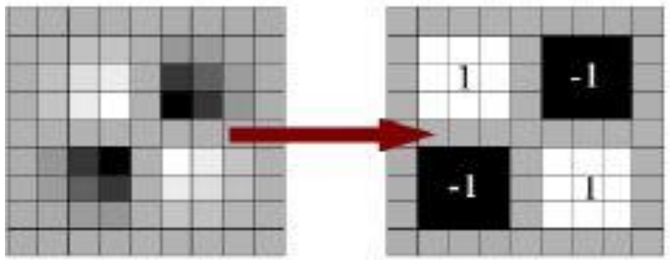


Gaussian box filters of σ=1.2

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	-10	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Lyy at a location

Filter response of box filters



L_{xy}

Gaussian box filters of σ=1.2

Black = -1, White = 1, Grey = 0

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

L_{xy} filter on Image

Filter response of box filters

Black = -1, White = 1, Grey = 0

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

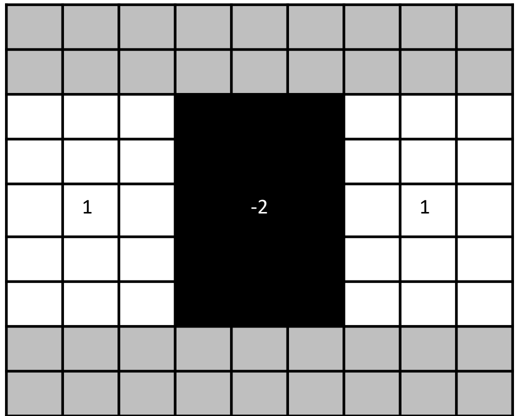
Lxy filter on image

1	2	3	4	5	6	7	8	9	10	11
2	4	6	8	10	12	14	16	18	20	22
3	6	10	14	18	22	26	30	34	37	40
4	8	14	20	26	32	38	44	50	54	58
5	10	18	26	34	42	50	58	66	71	76
6	12	22	32	42	52	62	72	82	88	94
7	14	26	38	50	62	74	86	98	105	112
8	16	30	44	58	72	86	100	114	122	130
9	18	34	50	66	82	98	114	130	139	148
10	20	37	54	71	88	105	122	139	149	159
11	22	40	58	76	94	112	130	148	159	170

Integral Image

Filter response of box filters

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1



Filter for Lxx

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Lxx Filter on Image

Filter response of box filters

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Lxx Filter on Image

1	2	3	4	5	6	7	8	9	10	11
2	4	6	8	10	12	14	16	18	20	22
3	6	10	14	18	22	26	30	34	37	40
4	8	14	20	26	32	38	44	50	54	58
5	10	18	26	34	42	50	58	66	71	76
6	12	22	32	42	52	62	72	82	88	94
7	14	26	38	50	62	74	86	98	105	112
8	16	30	44	58	72	86	100	114	122	130
9	18	34	50	66	82	98	114	130	139	148
10	20	37	54	71	88	105	122	139	149	159
11	22	40	58	76	94	112	130	148	159	170

Integral Image

Lxx at point, p = ?

SURF (Key Point Detection)

- Determine Hessian matrix

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

- $L_{xx}(p, \sigma)$ is convolution of image by second order derivative in x direction
- Similarly, $L_{yy}(p, \sigma)$, $L_{xy}(p, \sigma)$ are defined
- Points is chosen as a corner point if determinant at this point is large

Filter response of box filters

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Image

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

Lxx = -10, Lyy = -10, Lxy = 1

Hessian matrix at p

$$H = \begin{bmatrix} -10 & 1 \\ 1 & -10 \end{bmatrix}$$

Det(H) = $L_{xx}L_{yy}-(Wl_{xy})^2$, W is a constant

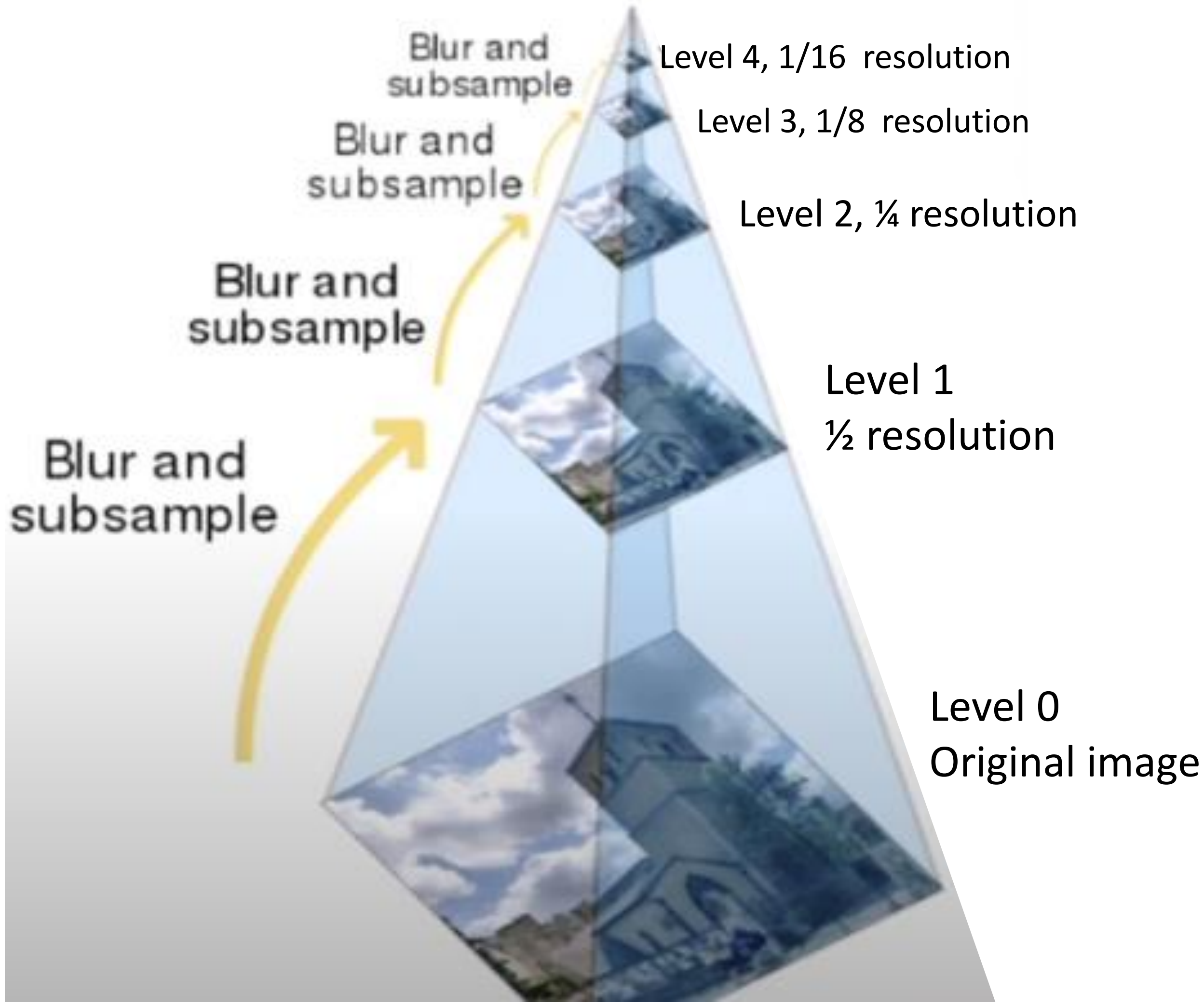
= $L_{xx}L_{yy}-(0.9l_{xy})^2$

= 100 – 0.81x1

- Points is chosen as a corner point if determinant > threshold
- Same process is repeated for all pixels of the image
- Above step is repeated for box filters with different sigma values

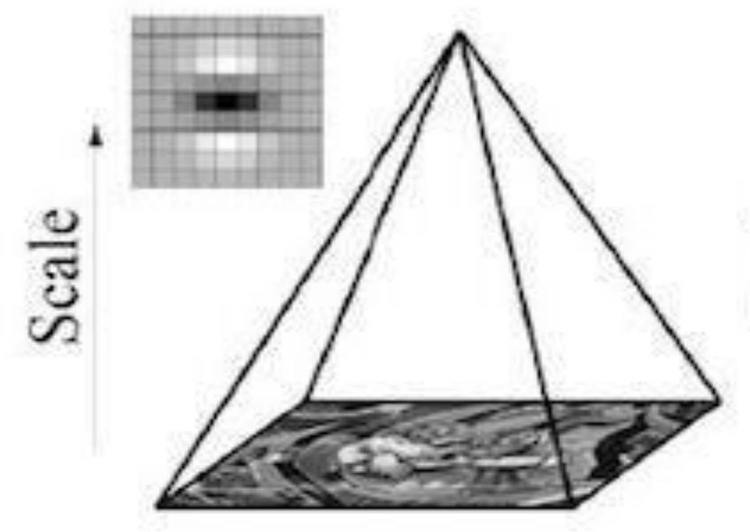
Image Pyramids for SIFT to generate scale space

- For SIFT, images are repeatedly smoothed by Gaussian filters and subsequently down sampled by rate 1/2



Speeded Up Robust Features SURF (Scale Space)

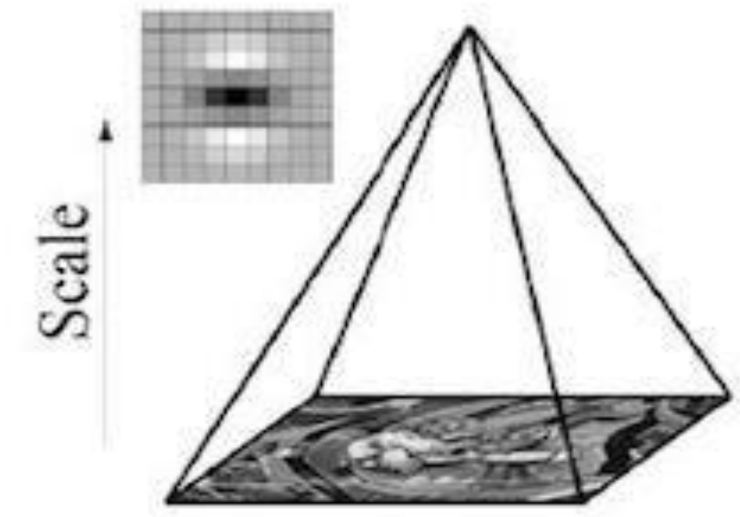
- For SURF, instead of reducing the size of image, filter size is increased
- Computational complexity does not increase with the size of filter
- Because integral images and box filters are used



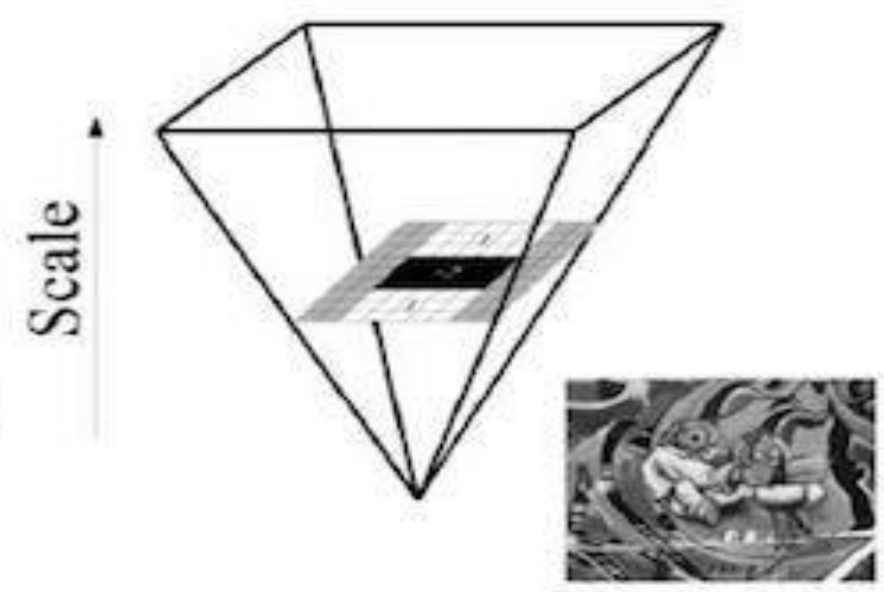
Gaussian filters on image pyramid
for SIFT

Speeded Up Robust Features SURF (Scale Space)

- Instead of reducing the size of image, filter size is increased
- Computational complexity does not increase with the size of filter
- Because integral images and box filters are used



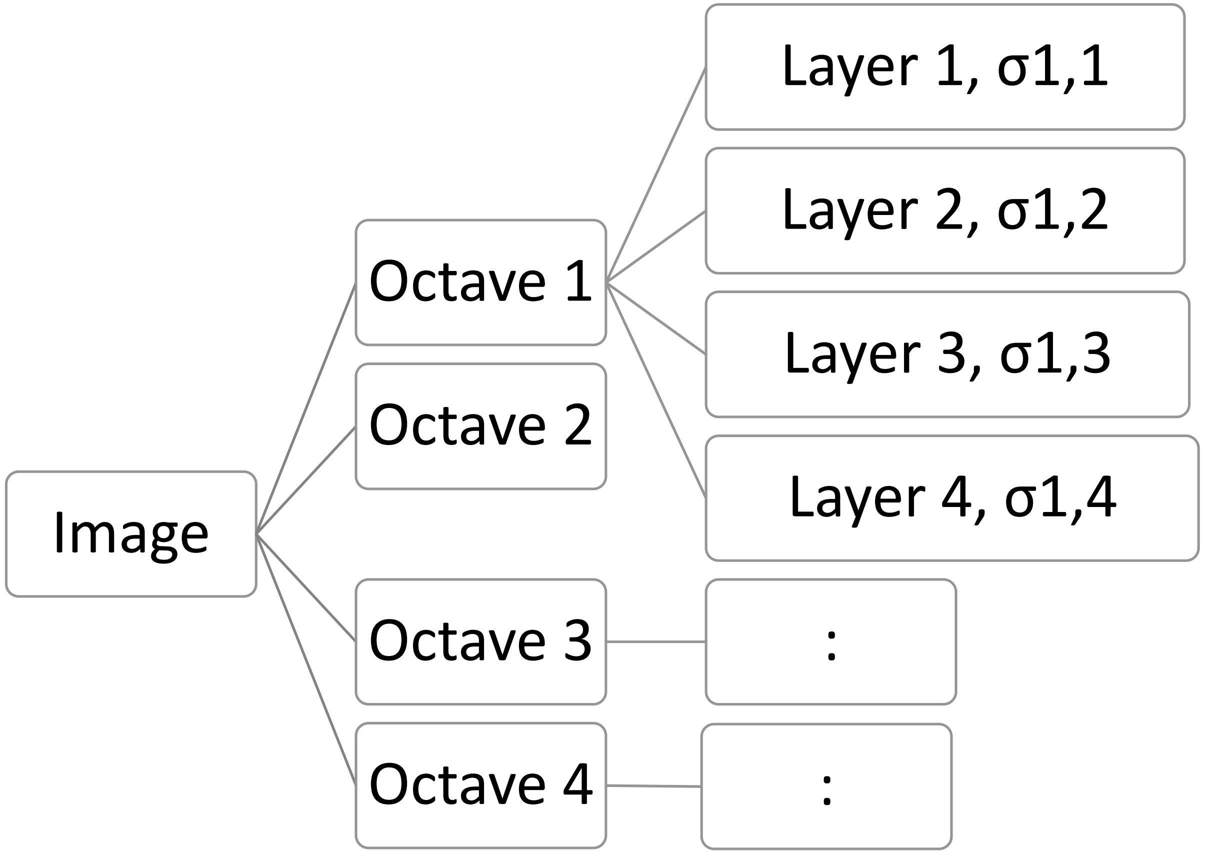
Gaussian filters on image pyramid for SIFT



Box filters on filter pyramid for SURF

Speeded Up Robust Features SURF (Scale Space and location of interest points)

- Uses a number of octaves
- Each octave has a number of layers
- For each octave and layer, image is filtered with second order Gaussian Filter with higher and higher standard deviation



Speeded Up Robust Features SURF (Scale Space and location of interest points)

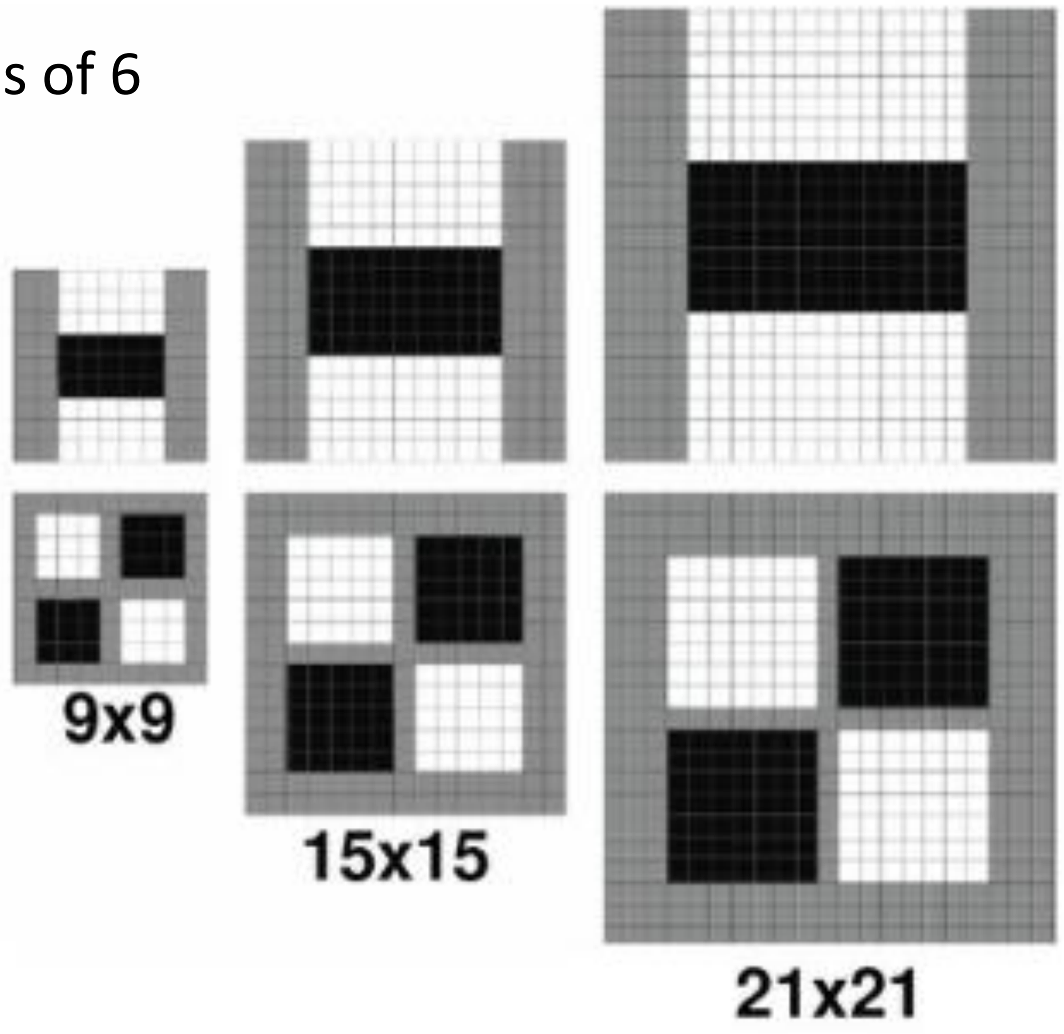
- Instead of reducing the size of image, filter size is increased for the same image

Size of filter	scale(σ) of filter
9×9	1.2 (initial scale)
15×15	1.6
21×21	3.2
27×27	3.6

SURF (Octave and layers)

Size of filter in first octave increases by multiples of 6

Size of filters for first octave	**scale(σ) of filter**	**Layer**
9×9	1.2	1
15×15 (= 9+6x1)	1.6	2
21×21 (= 9+6x2)	3.2	3
27×27 (= 9+6x3)	3.6	4



SURF (Octave and layers)

Size of filters for first octave	**Scale (σ) of filter**	**Layer**
9×9	1.2	1
15×15 (= 9+6x1)	1.6	2
21×21 (= 9+6x2)	3.2	3
27×27 (= 9+6x3)	3.6	4

- Range of filter sizes in the middle is 12 to 24
- First octave,
  - Layer 1, $\sigma = 1.2$
  - Layer 2, $\sigma = 1.2 \times 12/9 = 1.6$
  - Layer 3, $\sigma = 1.2 \times 24/9 = 3.2$
  - Layer 4, $\sigma = 1.2 \times 27/9 = 3.6$

SURF (Octave and layers)

For the second octave, filter size increases by multiples of 12

Size of filters for second octave	**scale(σ) of filter**	**Layer**
15×15	1.8	1
27×27 (= 15+12x1)	2.8	2
39×39 (= 15+12x2)	6	3
51×51 (= 15+12x3)	6.8	4

- Range of filter sizes in the middle is 21 to 45
- Second octave,
  - Layer 1, $\sigma = 1.8$
  - Layer 2, $\sigma = 1.2 \times 21/9 = 2.8$
  - Layer 3, $\sigma = 1.2 \times 45/9 = 6.0$
  - Layer 4, $\sigma = 1.2 \times 51/9 = 6.8$

SURF (Octave and layers)

Size of filter in third octave increases by multiples of 24

Size of filter for third octave	**scale(σ) of filter**	**Layer**
27×27	2.5	1
51×51 (= 27+24x1)	5.2	2
75×75 (= 27+24x2)	11.6	3
99×99 (= 27+24x3)	13.2	4

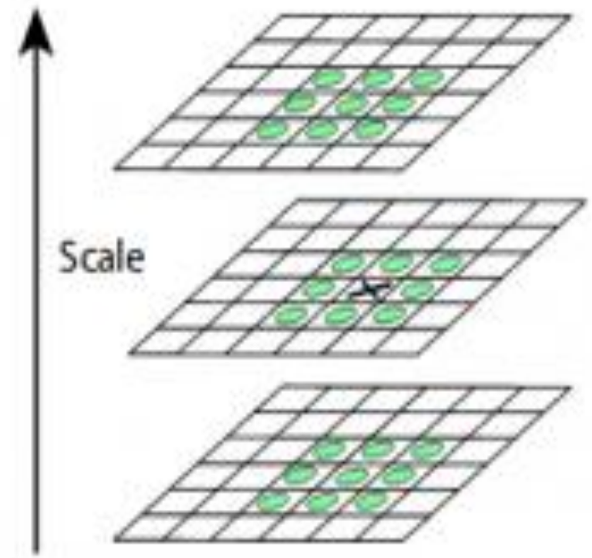
- Range of filter sizes in the middle is 39 to 87
- Second octave,
  - Layer 1, $\sigma = 2.5$
  - Layer 2, $\sigma = 1.2 \times 39/9 = 5.2$
  - Layer 3, $\sigma = 1.2 \times 87/9 = 11.6$
  - Layer 4, $\sigma = 1.2 \times 99/9 = 13.2$

SURF (Octave and layers)

- Increasing the octave number gives the ability to detect
  - both smaller and larger sized objects (blobs) in the image
- Increasing the number of octave layers give the ability to detect
  - Detailed finer features of different sizes in a blob

SURF (Key point detection)

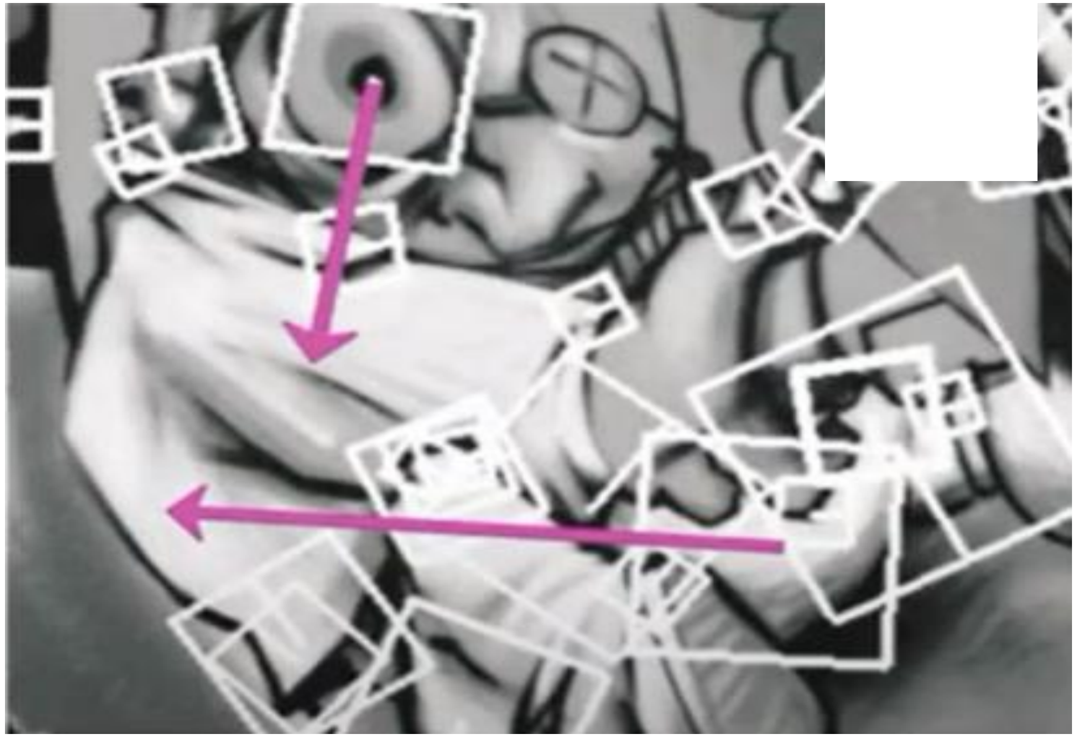
- Determine determinant of Hassian matrix for each point at different scales
- Choose only those points for which determinant > threshold
- These are interest points
- Apply Non Maxima suppression in 3x3x3 region



- A point is retained only if it is maximum among 26 neighbors
- These points are classified as key points

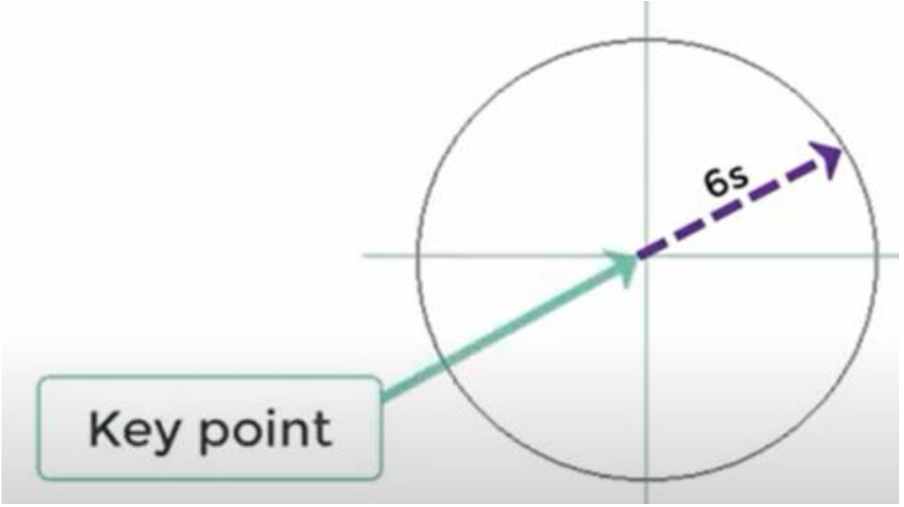
SURF (Key point orientation and descriptor)

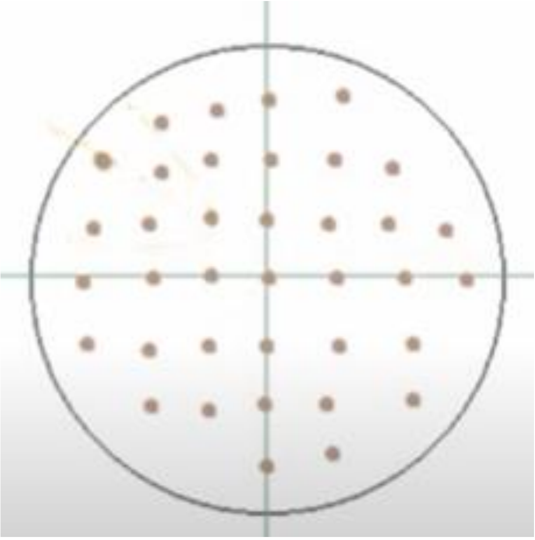
- Two step process
  1. Find dominant orientation of key point
  2. Compute descriptor vector in dominant orientation



SURF (Key point orientation)

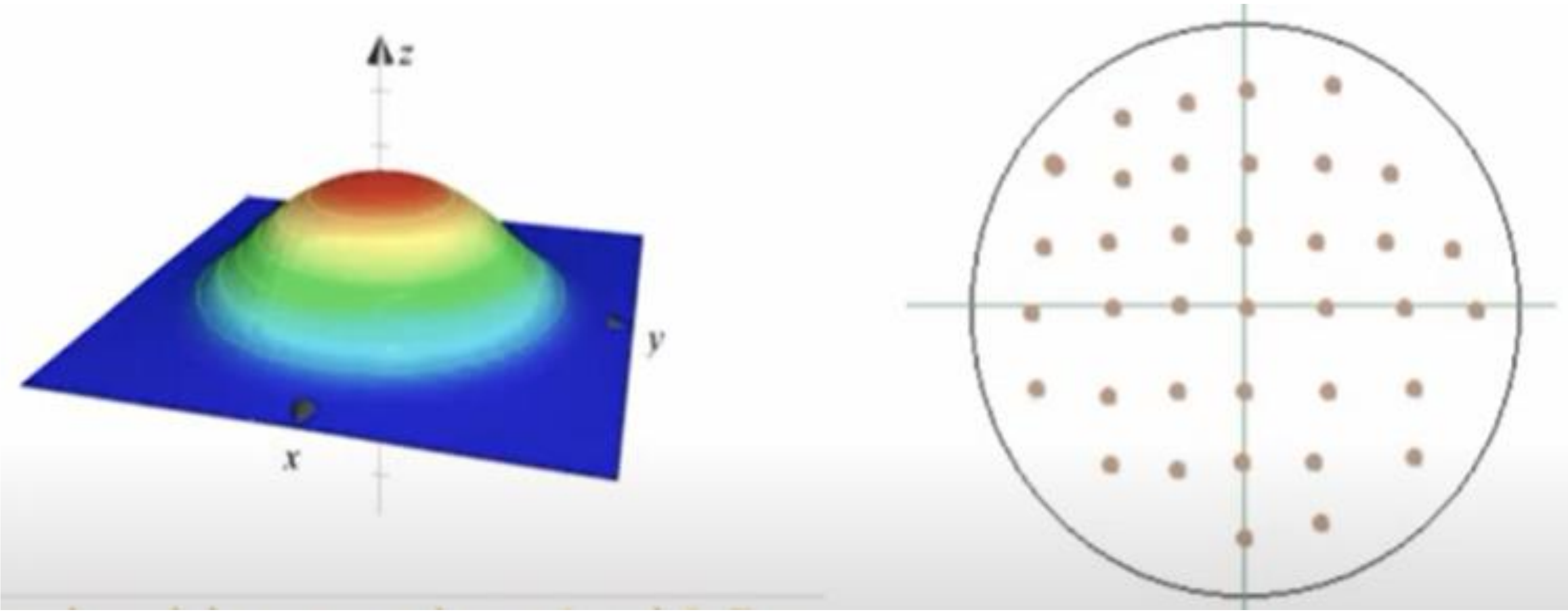
- Consider neighbourhood of radius 6s around the interest point
- Where, ‘s’ is scale (standard deviation) of filter for detected interest point
- Generate grid of points with distance of ‘s’ between them
- There are approximately 100 points/key point





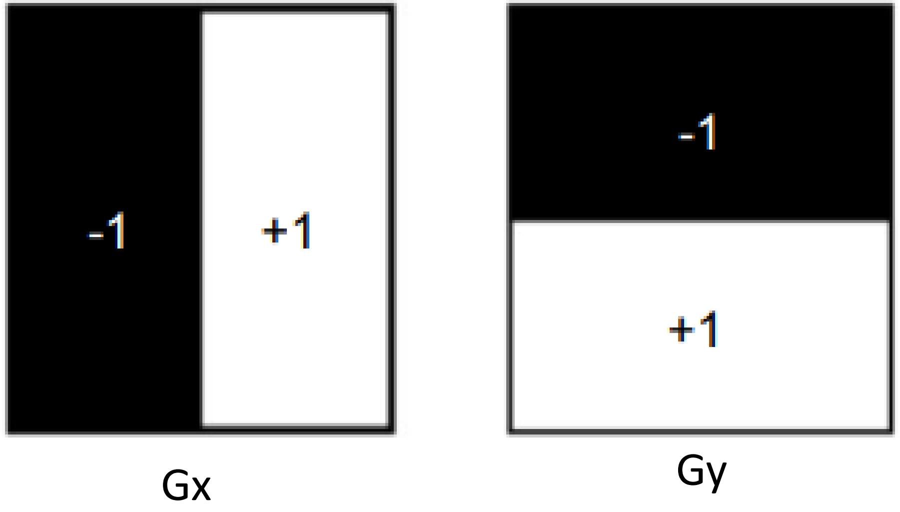
SURF (Key point orientation)

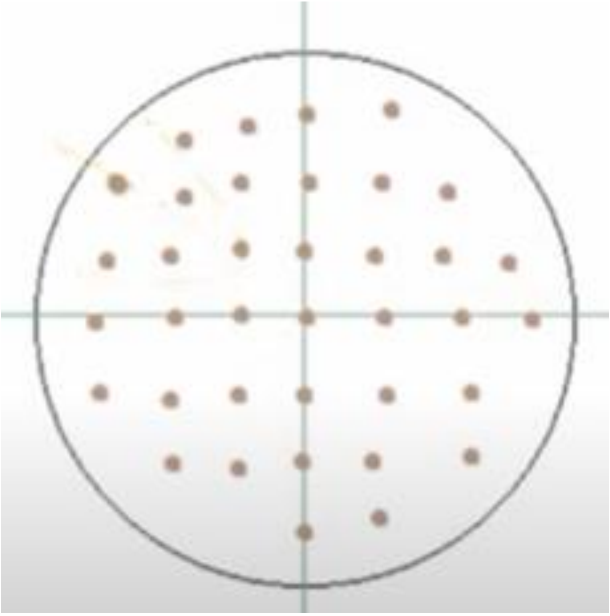
- Multiply grid of point with Gaussian filter of σ=3.3 s
- This is to give more importance to the point near the center



SURF (Key point orientation)

- Use Haar wavelets (filter) to compute gradient in x and y direction for each of 100 points





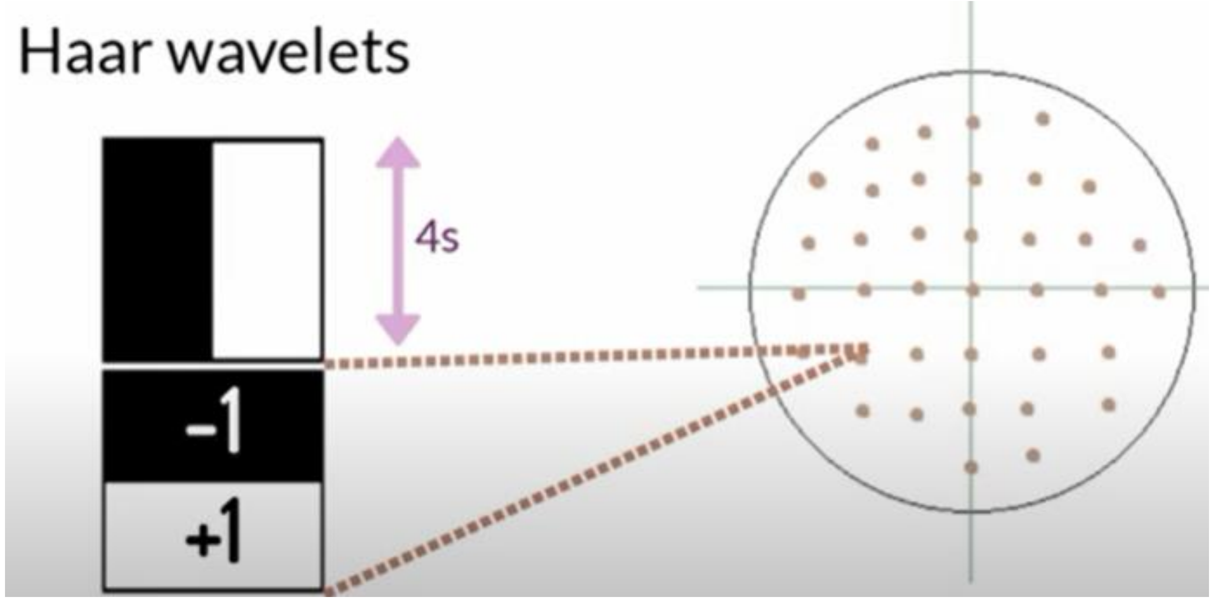
- There are fixed number of gradient (100 each) irrespective of sigma value, s
- It makes SURF scale invariant

SURF (Key point orientation)

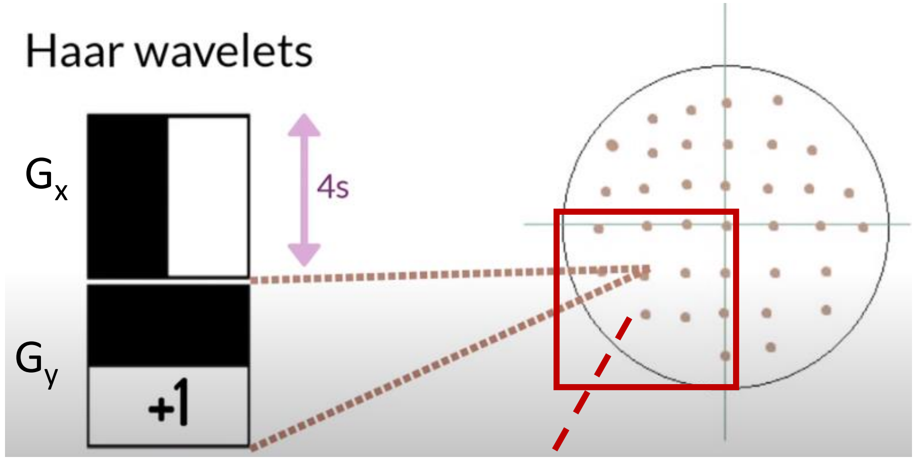
- Output of Haar filter
  $= 1/ns\ (\Sigma I_{white} - \Sigma I_{black}),\ n = 4$
- Complexity of computation is low
- Because sum can be calculated using integral image

Gx

Gy



SURF (Key point orientation)



- Output of Haar filter,

$\Delta = (\Sigma I_{white} - \Sigma i_{black})/4s$

15	15	6	6	6
15	15	6	6	6
15	15	6	6	6
15	15	6	6	6
15	15	6	6	6
15	15	6	6	6

Image

-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1

G_x, (for s=1)

-15	-15	6	6	6
-15	-15	6	6	6
-15	-15	6	6	6
-15	-15	6	6	6
15	15	6	6	6
15	15	6	6	6

Filter on Image

$[(15\times8) - (6\times8)]/4 = 18$

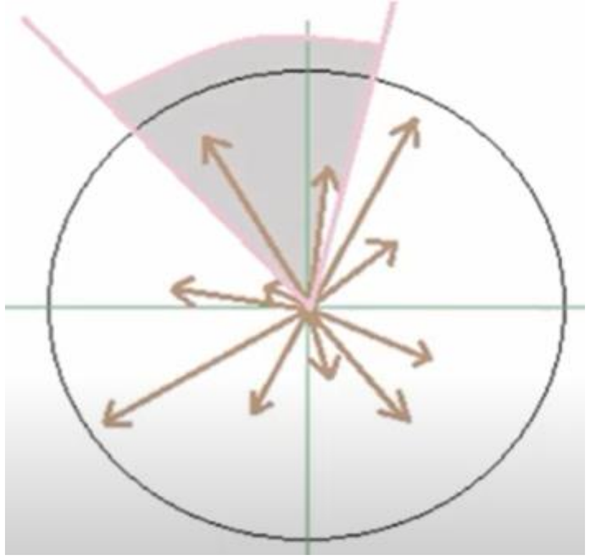
15	15	6	6	6
15	?	6	6	6
15	15	6	6	6
15	15	6	6	6
15	15	6	6	6
15	15	6	6	6

Filtered pixel in Image

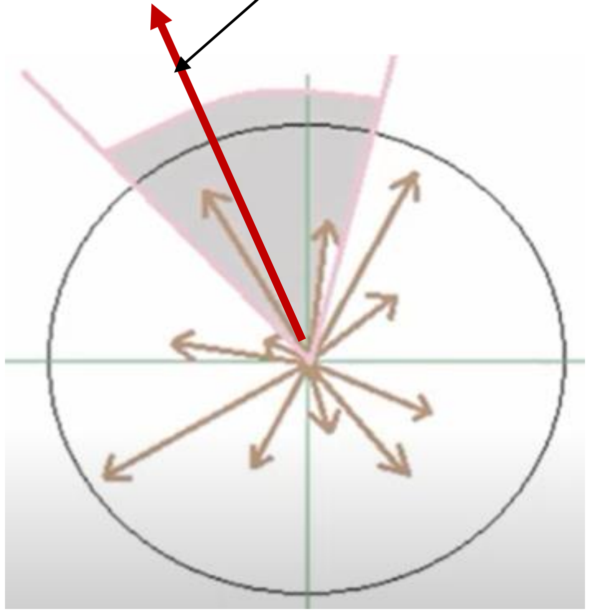
Filtering can use integral image as Haar filter is box filter

SURF (Key point orientation)

- Apply Haar wavelet to calculate Gx and Gy for each of 100 points
- Determine magnitude and orientation of gradient for 100 points
- And plot histogram of orientations on circular scale
- Orientations are in the sectors of 60^0
- Add magnitudes of points if their angles are in the same segment



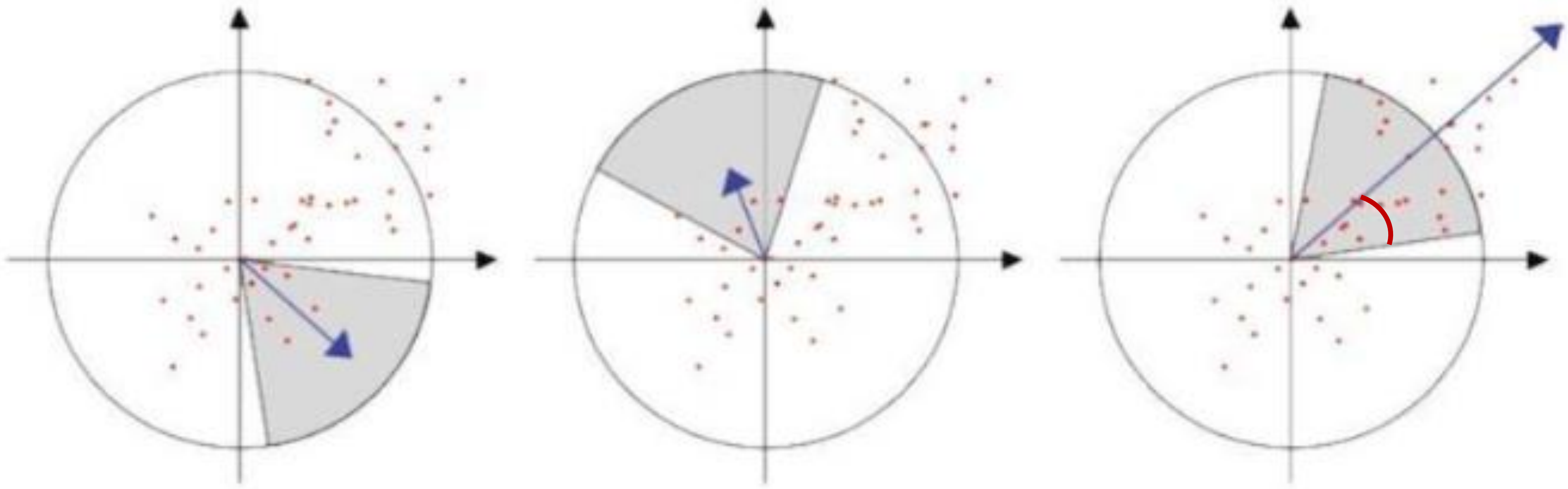
- Addition of magnitudes
- Orientation direction of segment



SURF (Key point orientation)

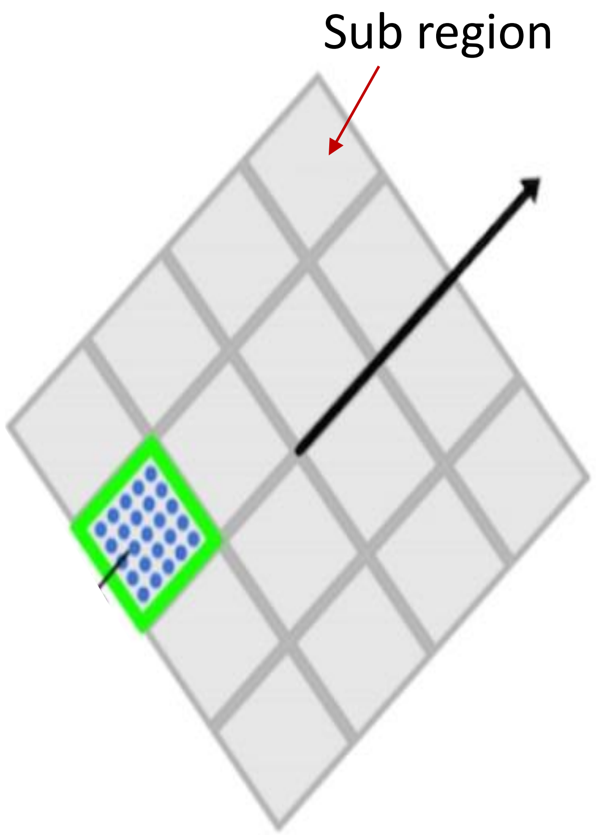
- Choose orientation for which magnitude is maximum
- This step ensures that descriptor is rotation invariant

- Orientation of key point



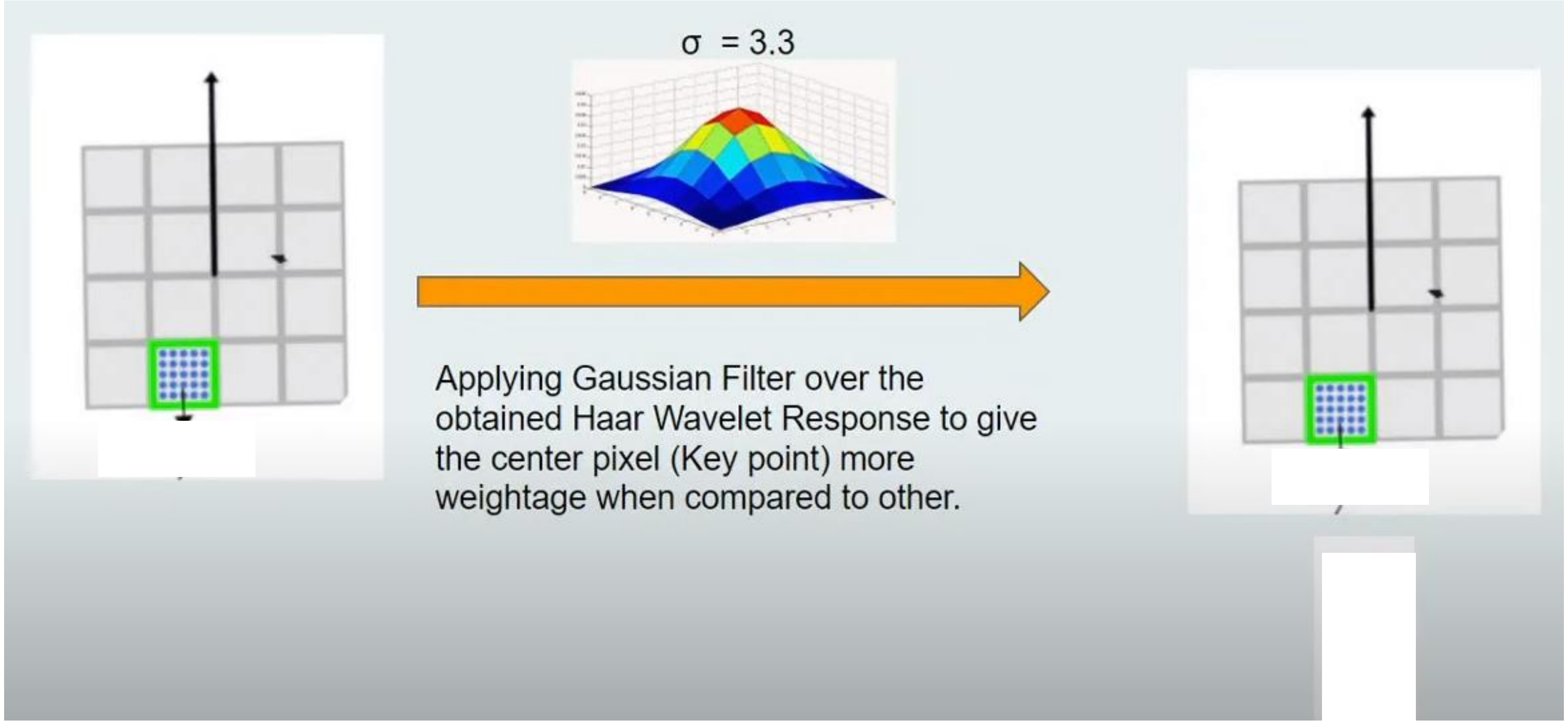
SURF (Key point descriptor)

- Compute descriptor vector in dominant direction
- Take a square window of size 20sx20s with key point at the center
- Split window into 4x4 subregions
- Each subregion is divided into 5x5 equal spaced points
- Space between points is s



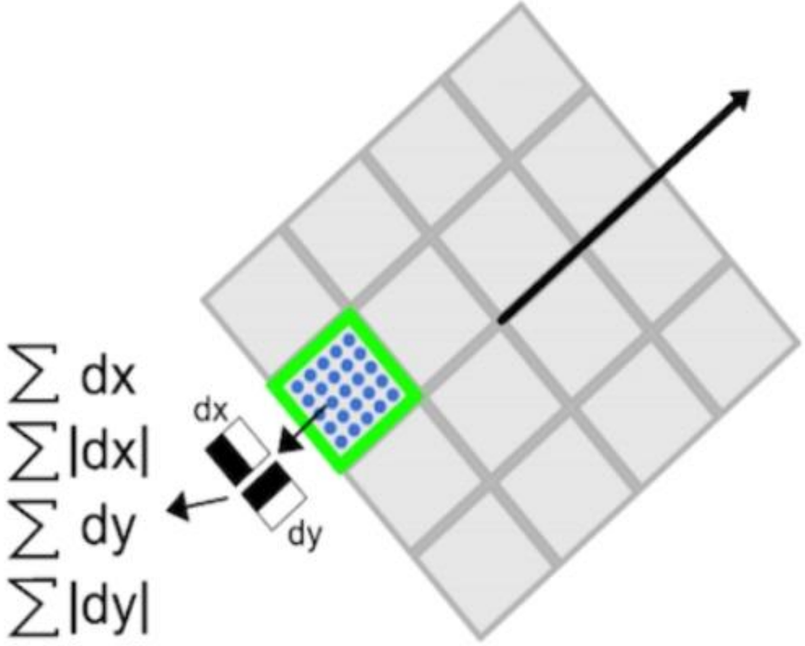
SURF (Key point descriptor)

- Gradients of sample points (dx and dy) are scaled by Gaussian with std = 3.3s centered at key point
- This ensures invariance against geometric deformation and localization errors



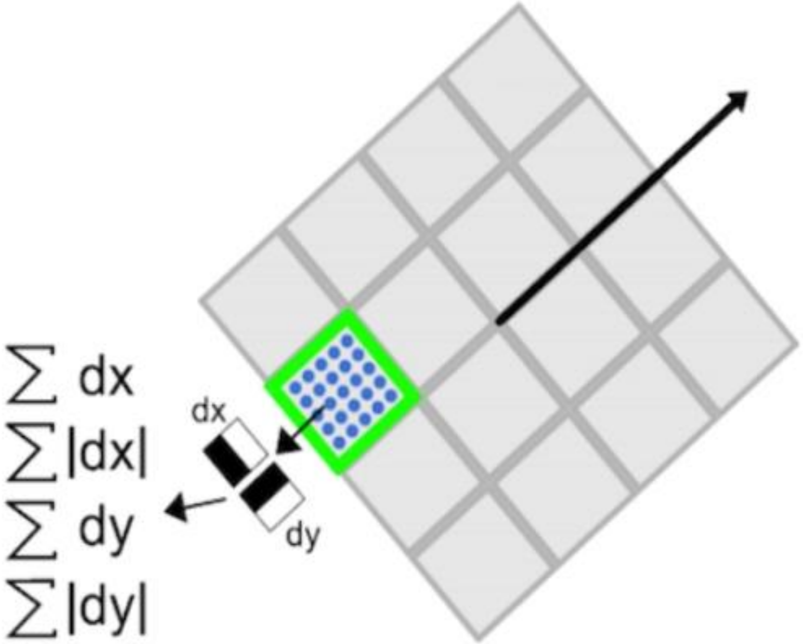
SURF (Key point descriptor)

- Apply Haar filter to determine dx and dy for each pixel in subregion
- Add gradients of 25 pixels to determine the features
  $\sum dx$, $\sum dy$, $\sum|dx|$, $\sum|dy|$

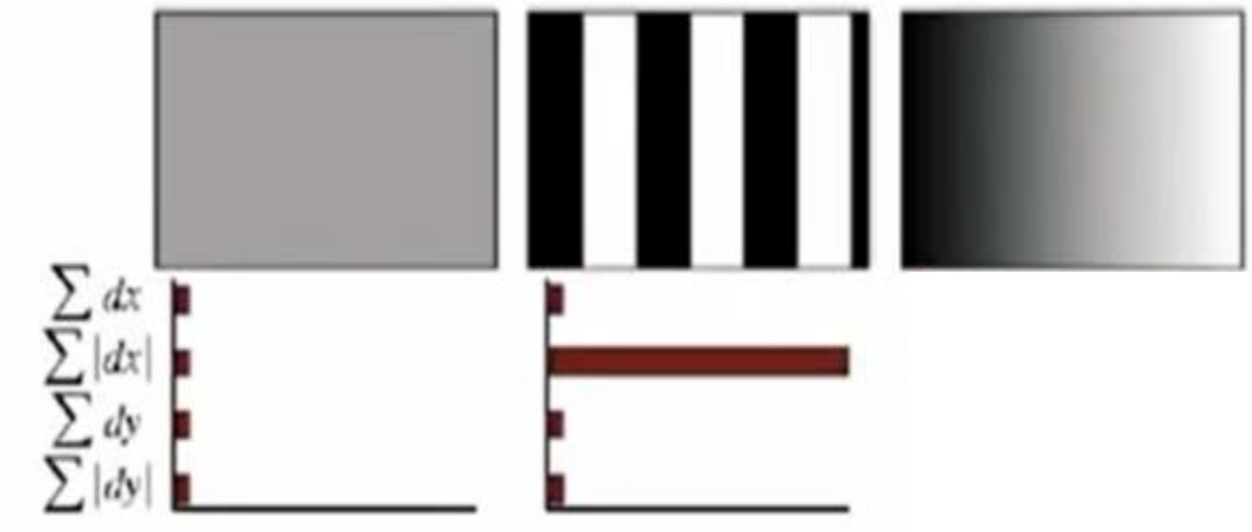


- Example:

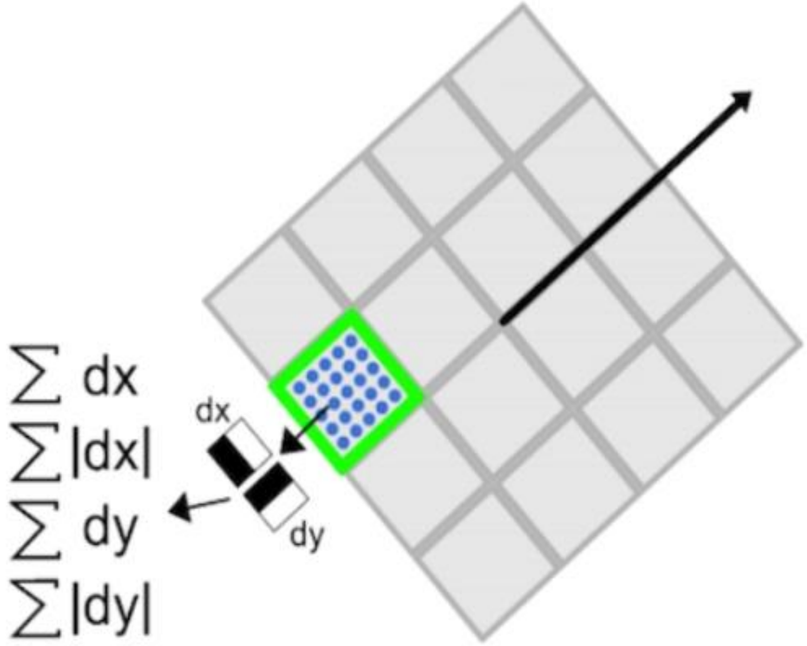
SURF (Key point descriptor)



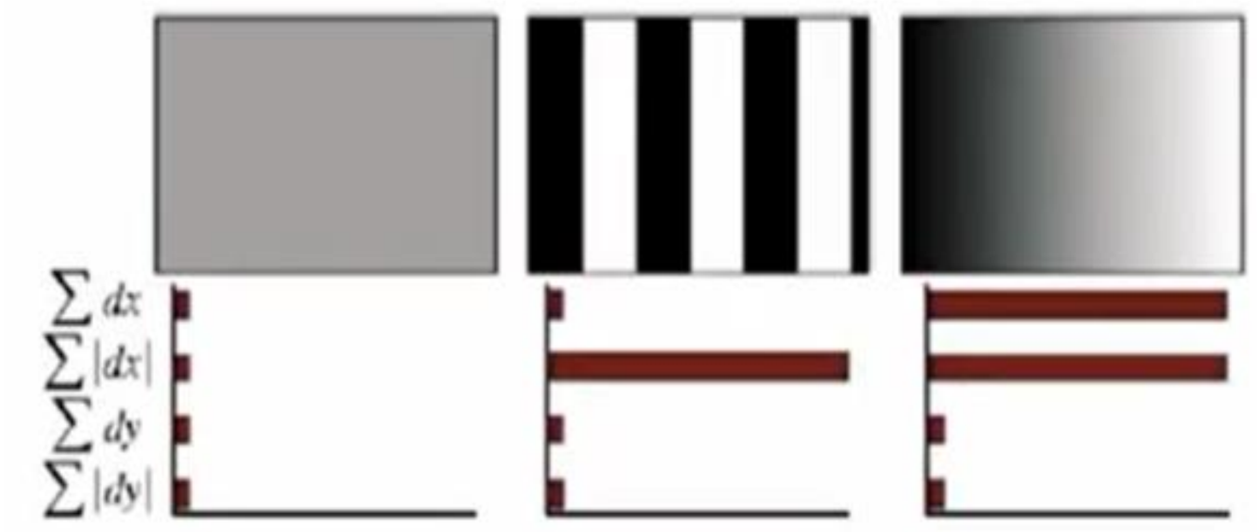
- Example:



SURF (Key point descriptor)

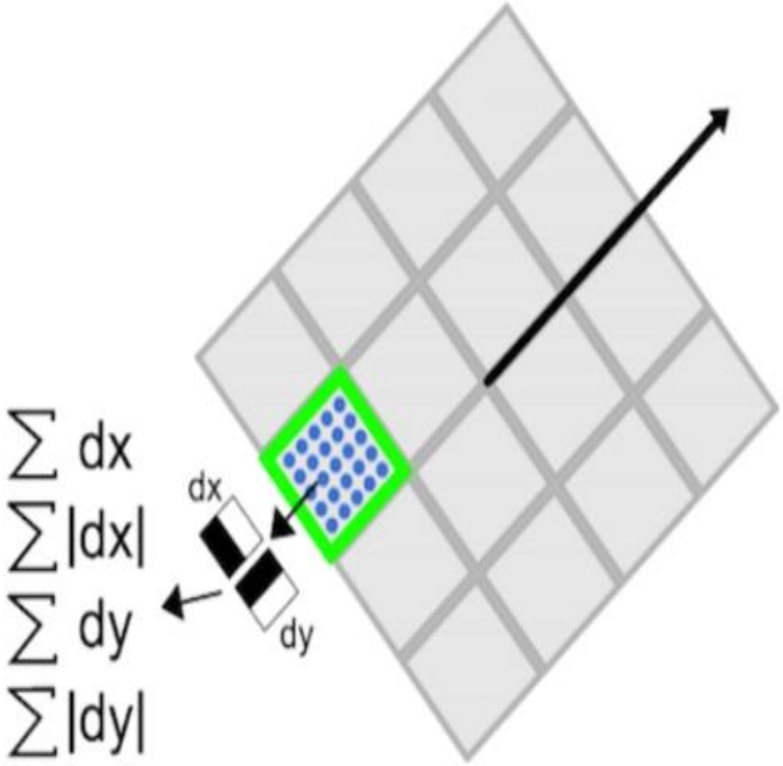


- Example:



SURF (Key point descriptor)

- Each subregion has 4 features
- ∑dx, ∑dy, ∑|dx|, ∑|dy|
- There are 16 sub regions
- Feature vector has 16x4=64 dimensions or length



SURF (Key point descriptor)

- Vector has 16x4=64 dimensions
- SURF has less number of features than SIFT which has 128 features
- Thus faster in matching
- Also faster in computation of feature vector at various stages
  - because integral image is used
- To speed up key point matching process, SURF has one more feature

SURF feature matching

- Check sign of Laplacian (trace of Hessian Matrix) for interest point
- It adds no computation cost since it is already computed during detection of interest points
- If sign of trace of Hessian matrix at key points in two images is same then use matching method to check whether key points have low Euclidean distance
- This information allows for faster matching, without reducing the descriptor's performance
- Example: Hessian matrices at corner points in images are

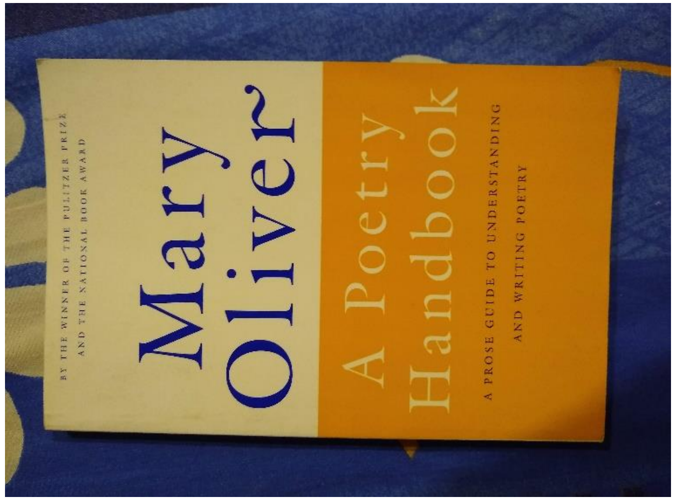
$$H1 = \begin{bmatrix} 10 & -1 \\ -1 & 15 \end{bmatrix} \qquad H2 = \begin{bmatrix} 10 & -2 \\ -2 & -15 \end{bmatrix} \qquad H3 = \begin{bmatrix} 10 & -1 \\ -1 & -5 \end{bmatrix}$$

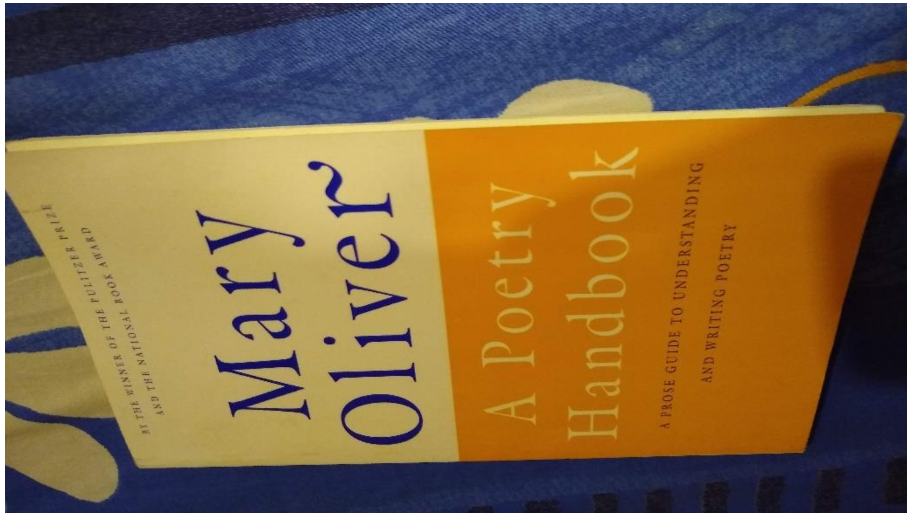
Trace1 = 25, trace2=-5, trace3=5

Check Euclidean distance between key point 1 and key point 3

Image Alignment/ Registration using SIFT/ SURF descriptors

- Align different images of the same scene
- Ex: click the picture of a book from various angles
- Image registration algorithm aligns an images with reference image





How does image registration work?

- Keypoint detectors and descriptors are SIFT, SURF, ORB(Oriented FAST and Rotated BRIEF)
- Match the key points between the two images
- Match features from the image to be aligned, to the reference image
- Brute force matcher is used to retrieve the best match
- Pick the top matches, and remove the noisy matches
- Store the coordinates of the corresponding key points
- Determine relation between keypoints of two images (rotation, scaling and shifting)
- Compute 2X2 homography transform matrices to represent the above relation
- Apply homograhy transform to unaligned image to align original image called warping
- Aligned images can be stitched to form bigger image

Image Alignment

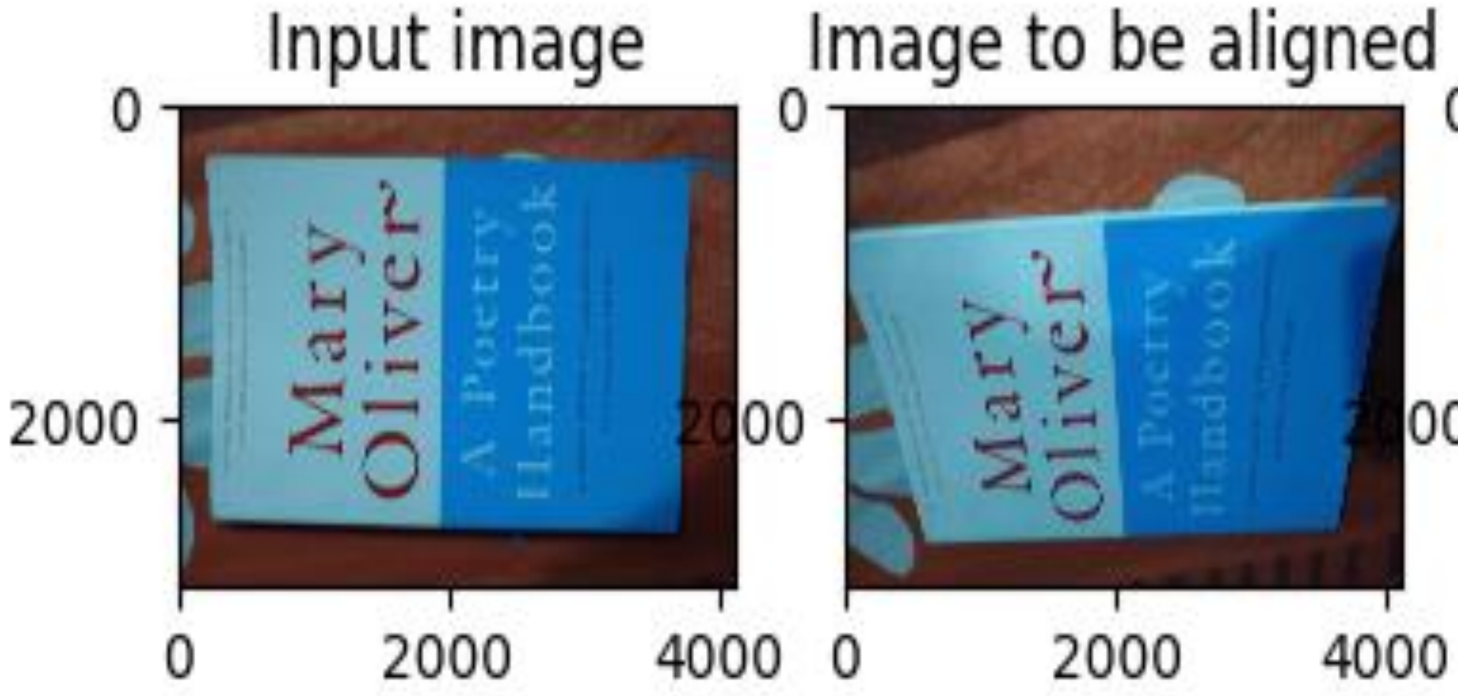
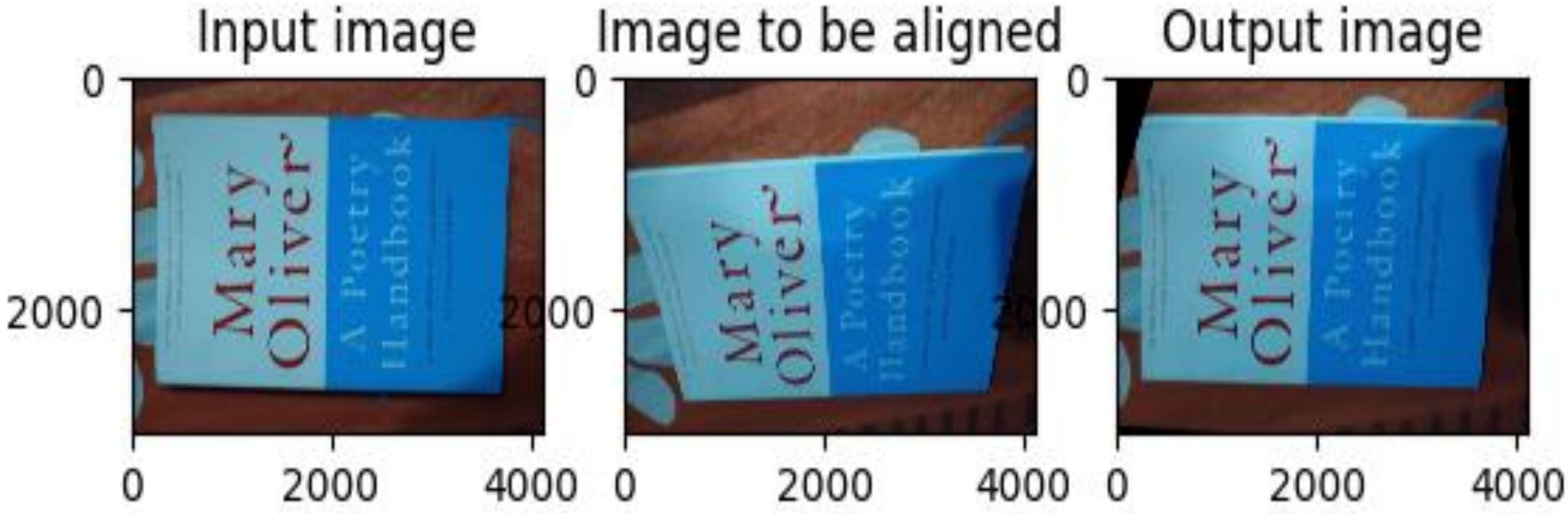
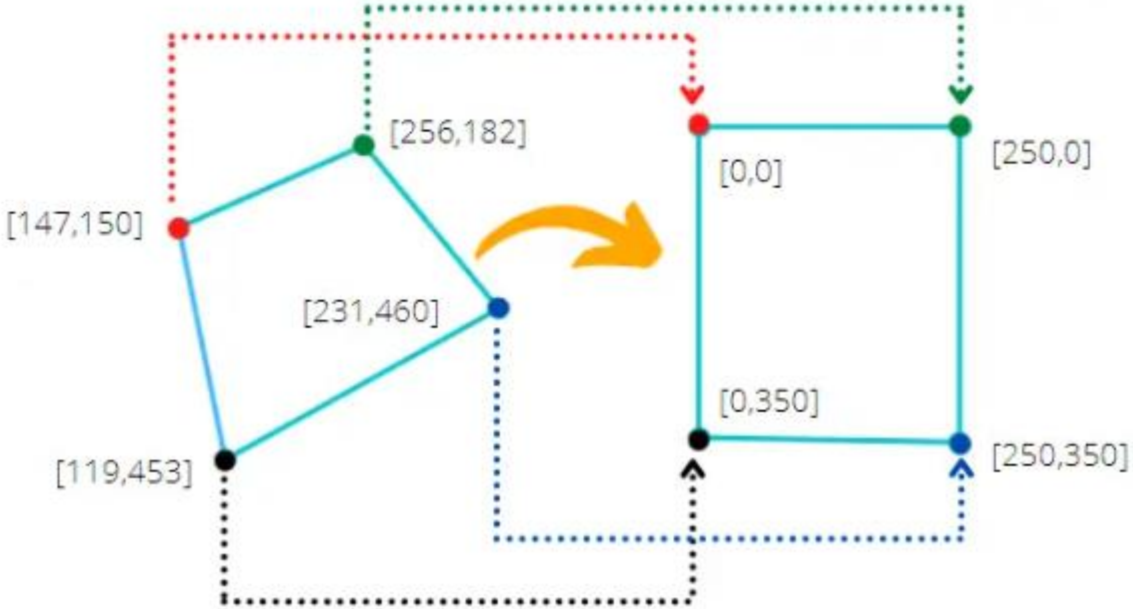


Image Alignment



Example: Image warping

- Apply warping to unaligned image to get final aligned image output



Applications of Feature Descriptors

Powerful tools in computer vision for identifying and describing local features in images

- **Image Matching**: Match keypoints between different images of the same scene or object.
  - Useful in applications like panorama stitching, where multiple images are combined to form a wide-angle view
- **Object Recognition**: Match features between a known object and a scene
  - Identify and locate objects within images
  - Widely used in robotics and automated inspection systems
- **3D Reconstruction**: Help in matching images taken from different viewpoints
  - which is essential for reconstructing 3D models of objects or environments
  - Useful in fields like augmented reality (AR) and virtual reality (VR)
- **Image Retrieval**: In content-based image retrieval systems, features can be used to search for and retrieve images
  - that are similar to a query image based on visual content rather than metadata

Applications of Feature Descriptors

- **Scene Recognition**: By analyzing the spatial arrangement of features,
  - systems can recognize and categorize scenes or environments, which is useful in autonomous navigation and contextual understanding in AI systems
- **Robotic Vision**: Robots use features for visual recognition tasks
  - helping them to navigate, identify objects, and interact with their environment more effectively
- **Video Tracking**: keypoints can be applied to track objects or people in video sequences by matching features frame-to-frame
  - which is useful in surveillance and motion analysis.
- **Forgery Detection**: In digital forensics, features can help detect tampered or forged images by identifying inconsistencies in local features