Computer Vision

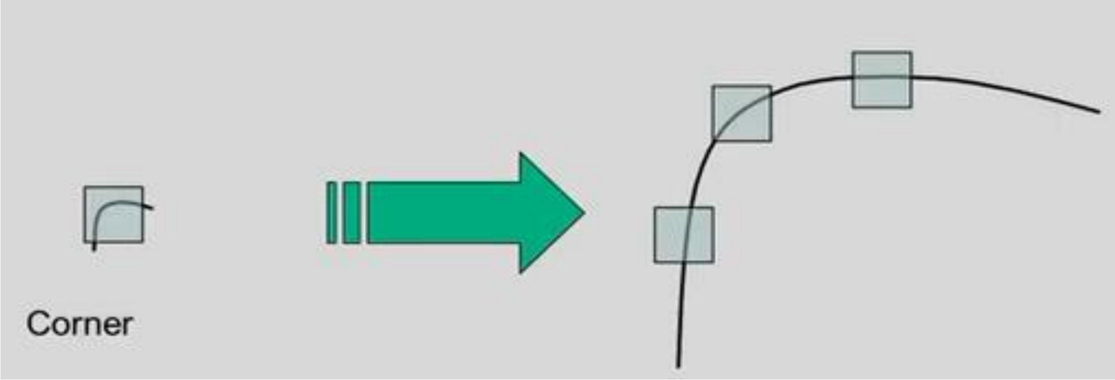
Feature Extraction
(SIFT, SURF)

Contents

- SIFT
- SURF

Limitations of Harris and Hessian Corner Detectors

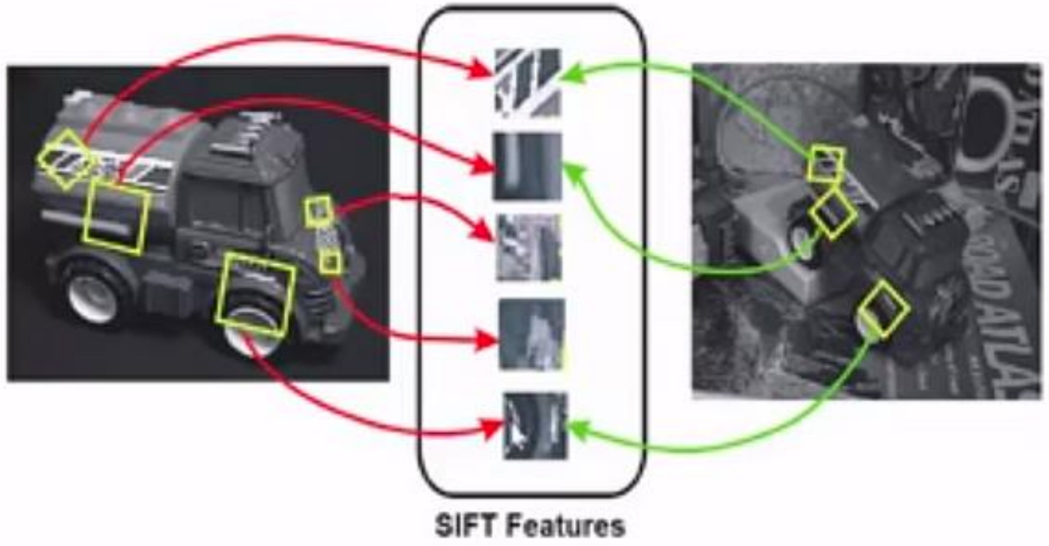
- Image scaling



- 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 (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

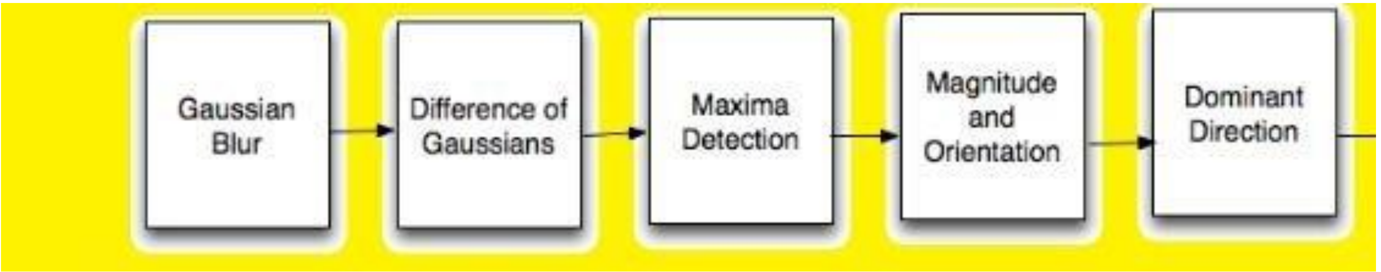
Advantages of SIFT Detector

- 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

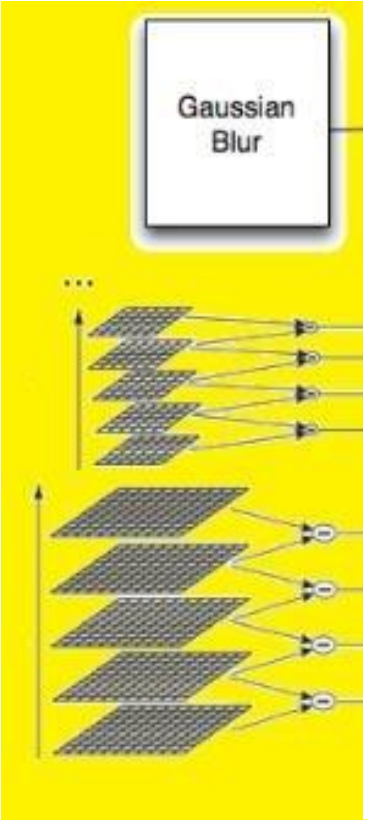
SIFT Algorithm



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

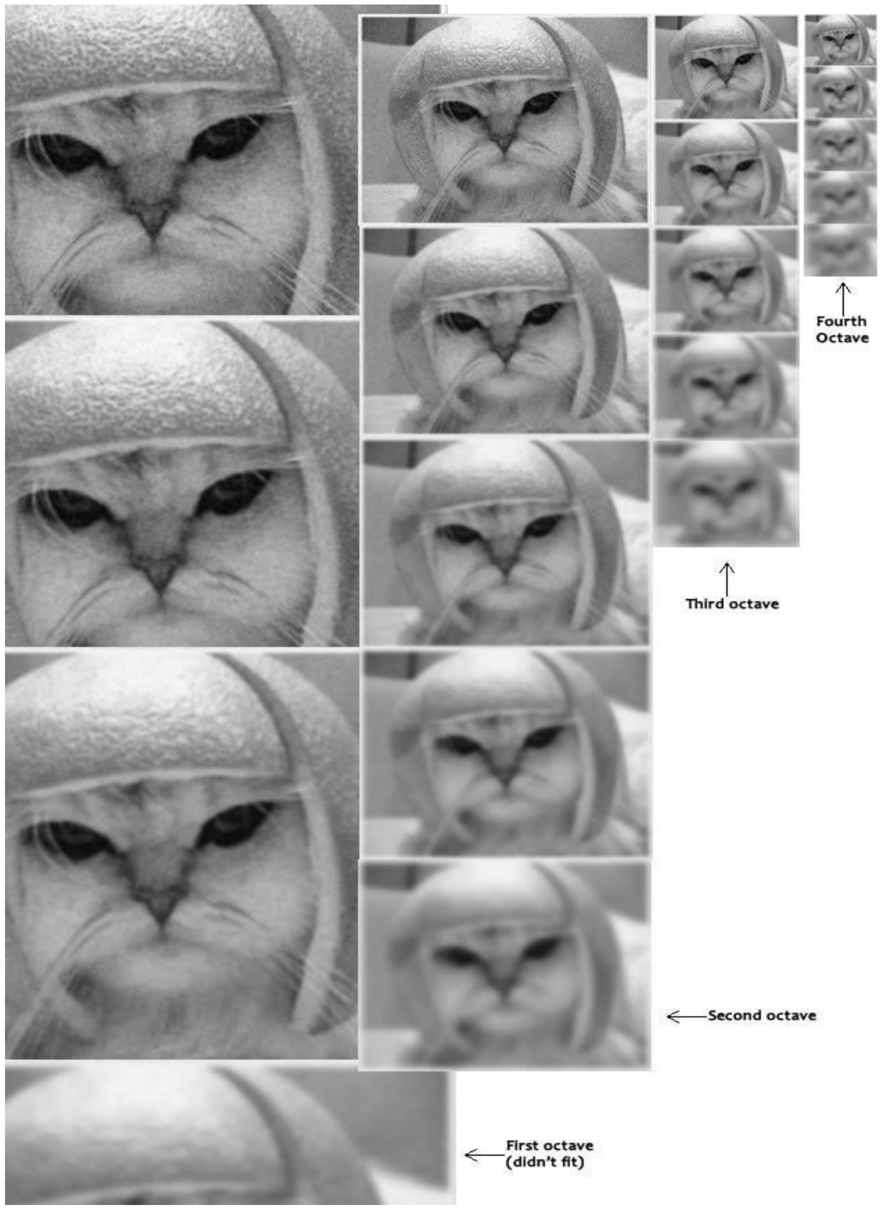


SIFT Algorithm (Construct a Scale Space)

- 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

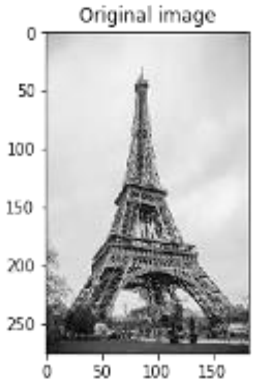


SIFT Algorithm (Construct a Scale Space)

- 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)

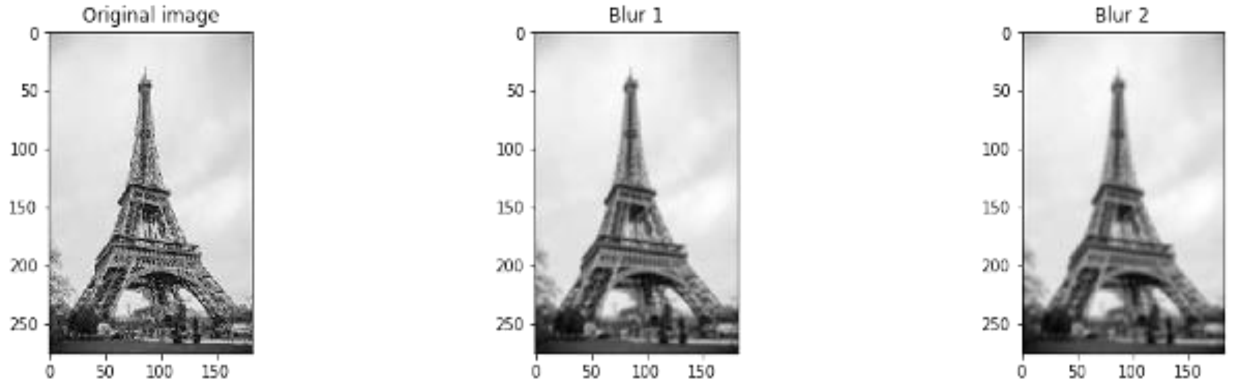
Original image of size (275, 183)



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

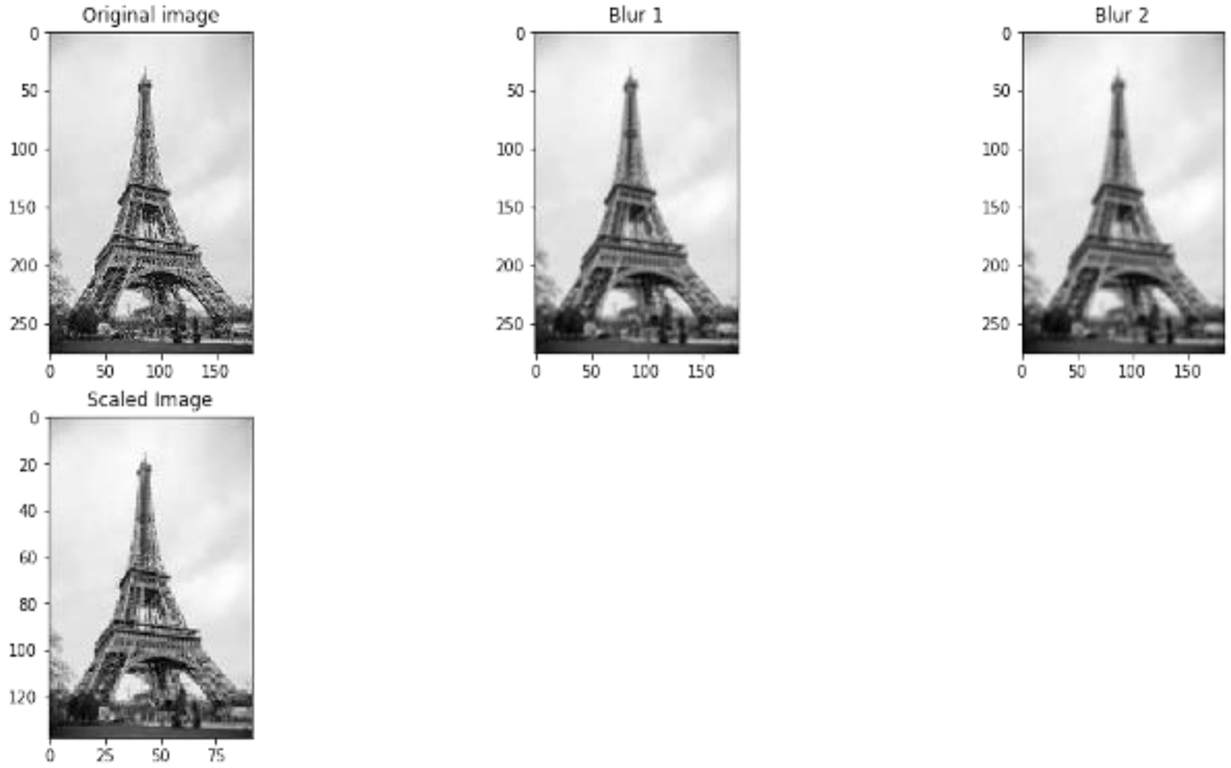
Original image of size (275, 183)



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)

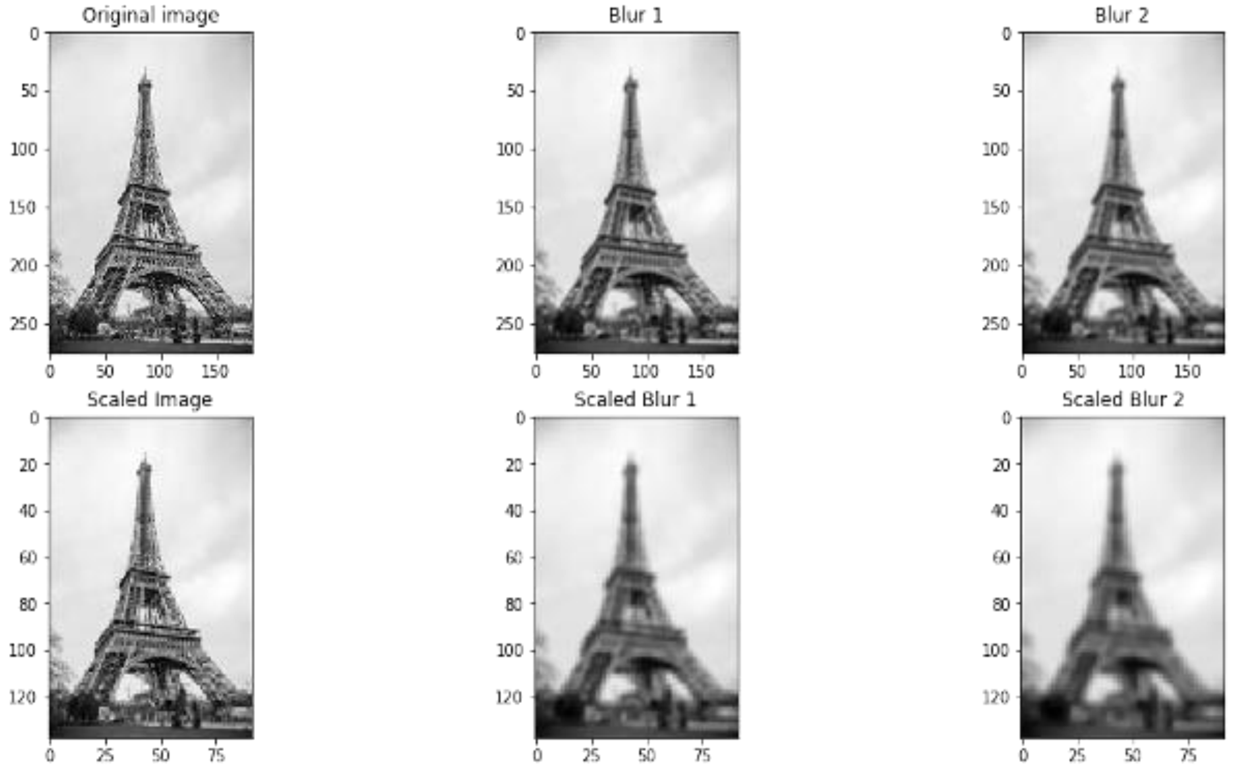


scaled image of dimension (138, 92)

SIFT Algorithm (Construct a Scale Space)

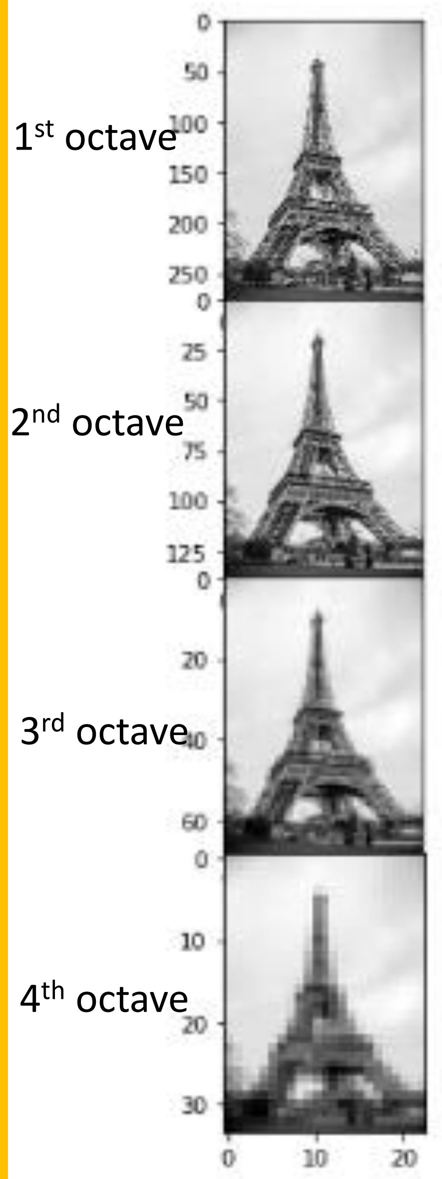
- Create a new set of images
- Take the original image and down sample it by rate 1/2
- Reduces the resolution of image
- For each new image, create blurred versions

Original image of size (275, 183)



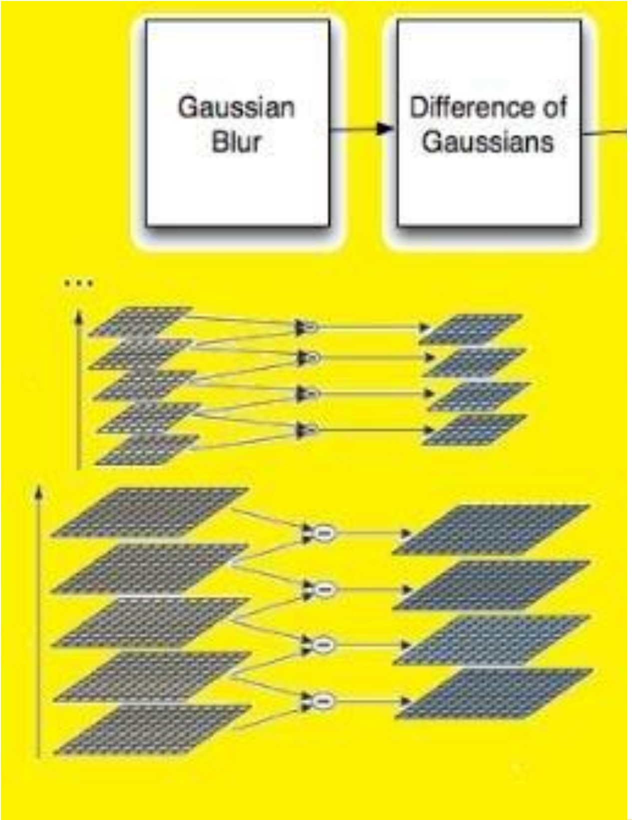
scaled image of dimension (138, 92)

SIFT Algorithm (Construct a Scale Space)



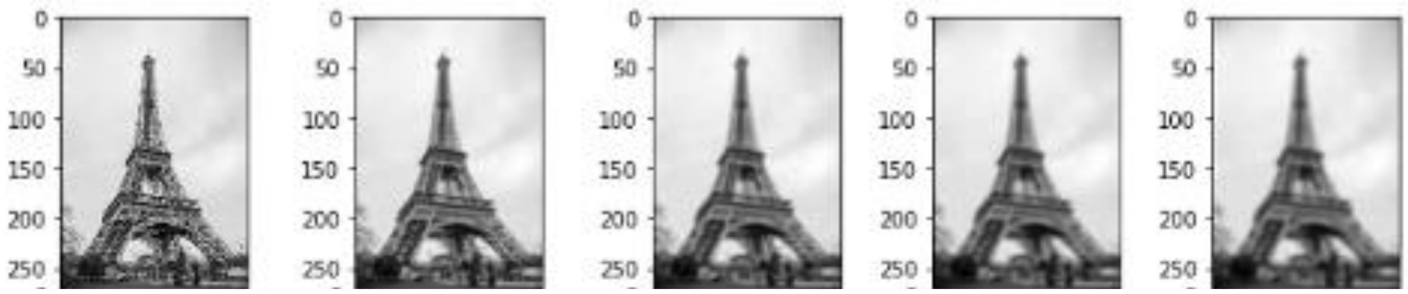
- Scale space is a collection of images having resolutions and blurring generated from a single image
- Ideal number of low resolution versions (octaves) is four
- Octave is different levels of image resolutions
- Each octave is down sampled by 2 to generate next octave to reduce image size by 1/4
-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

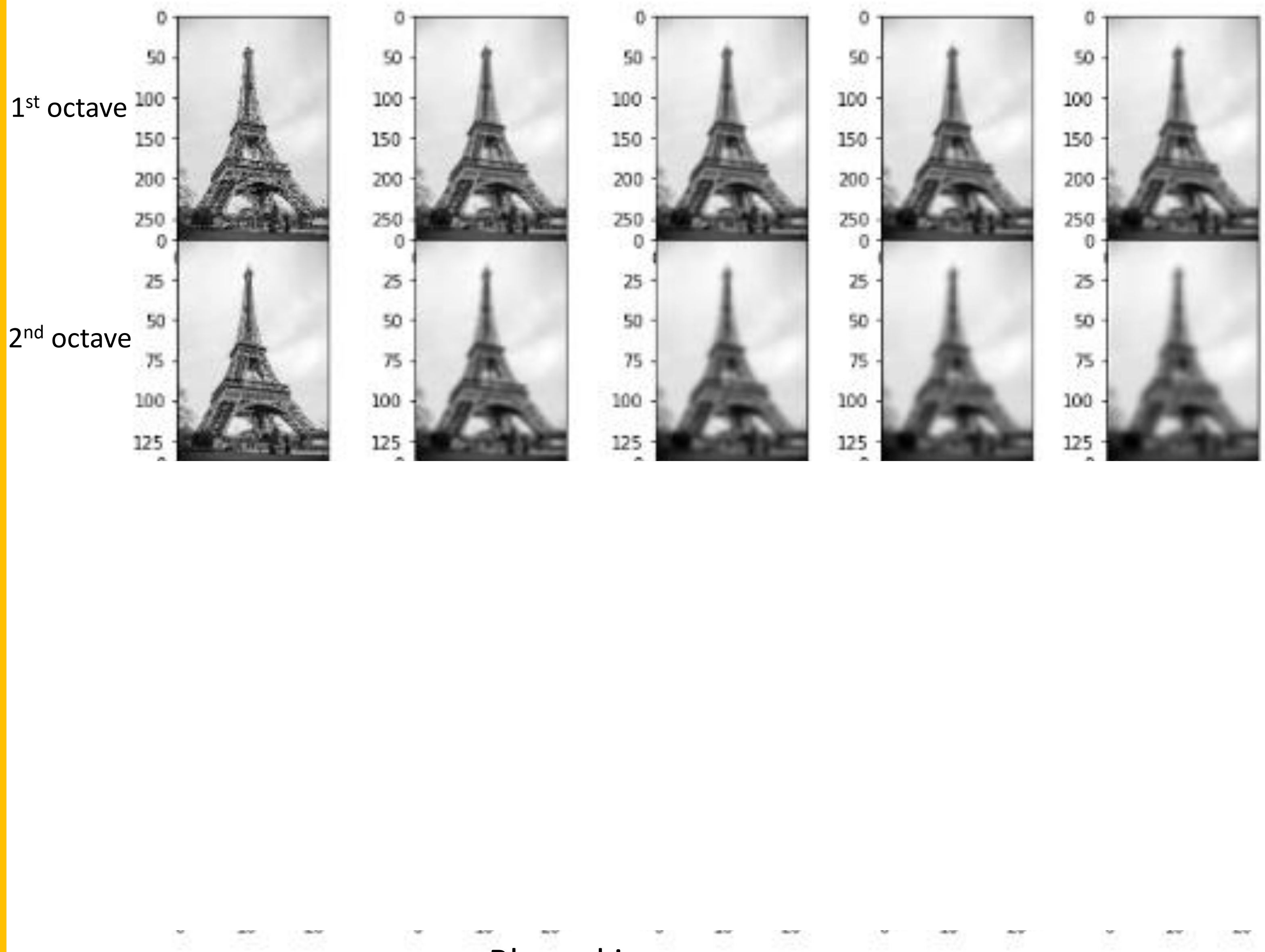


Blurred images

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)

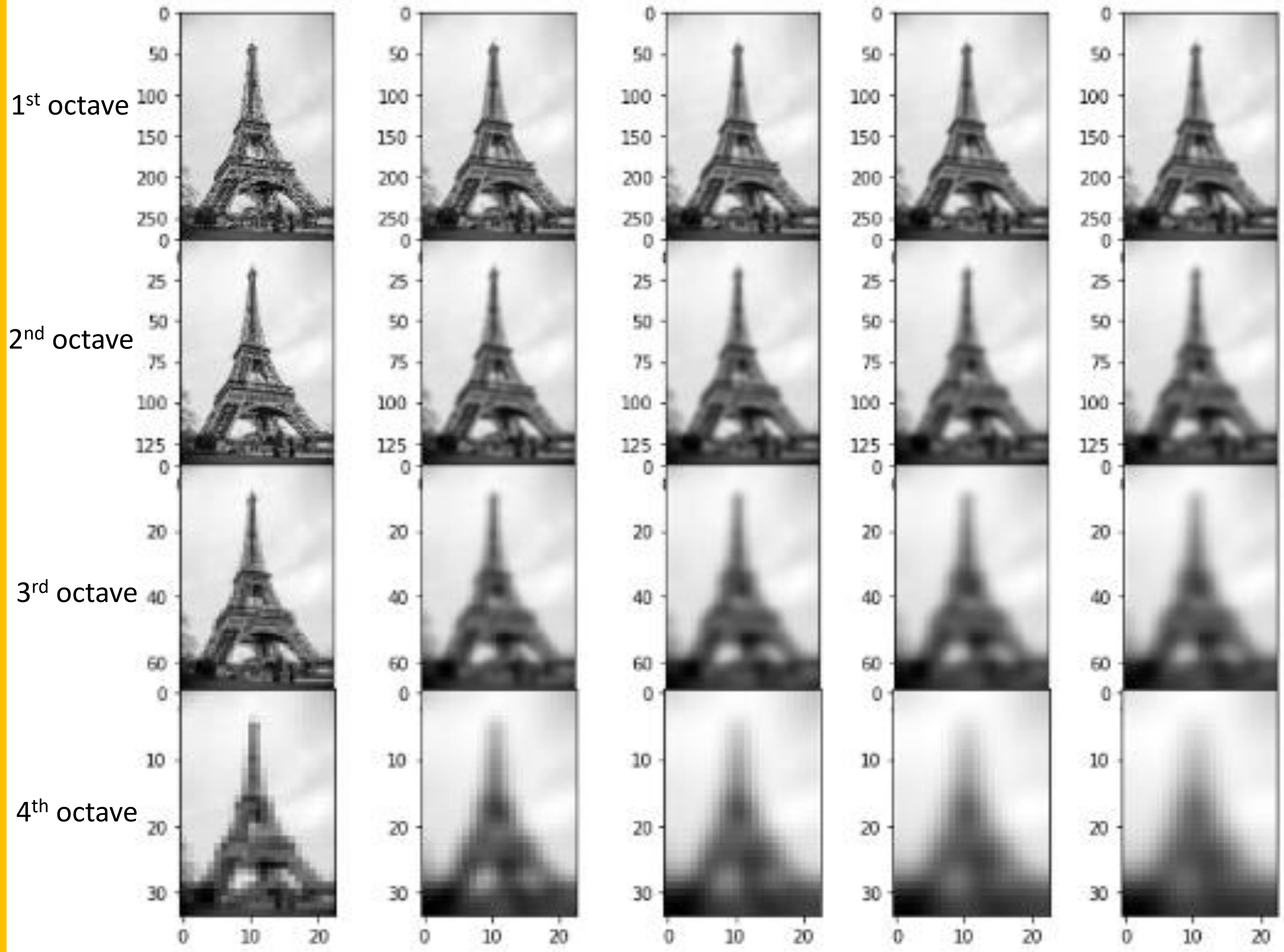


Blurred images

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)

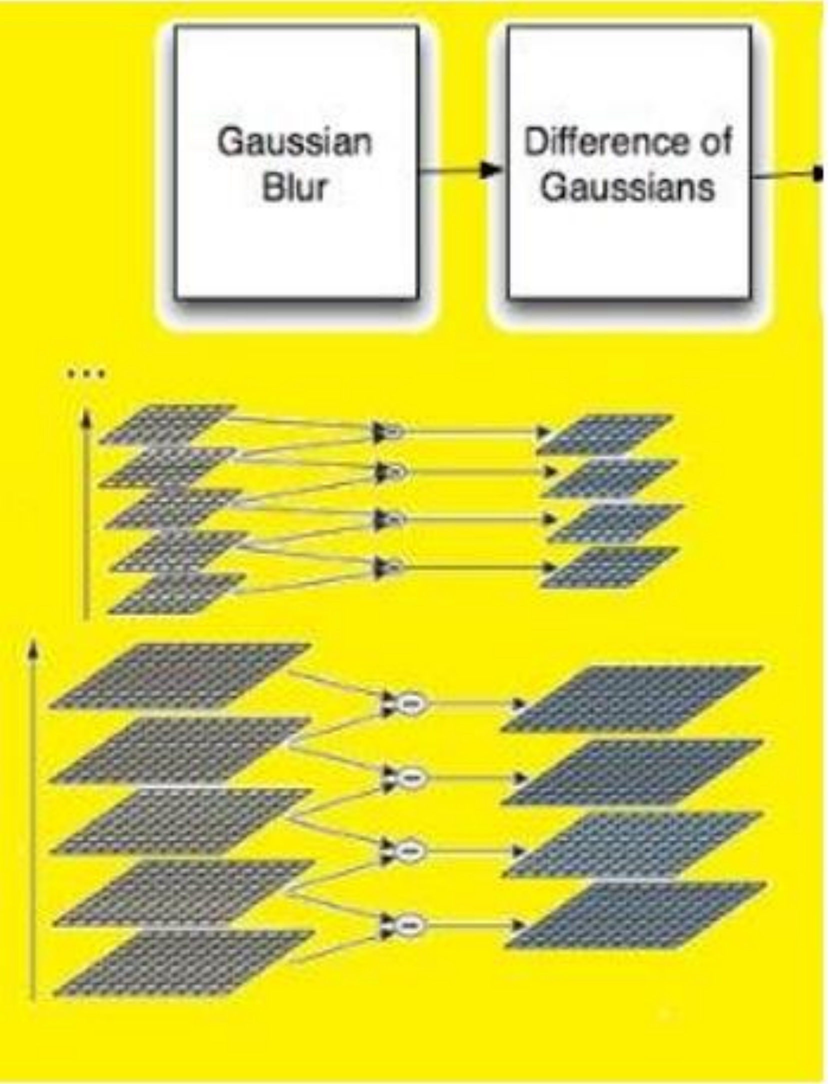


Blurred images

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)

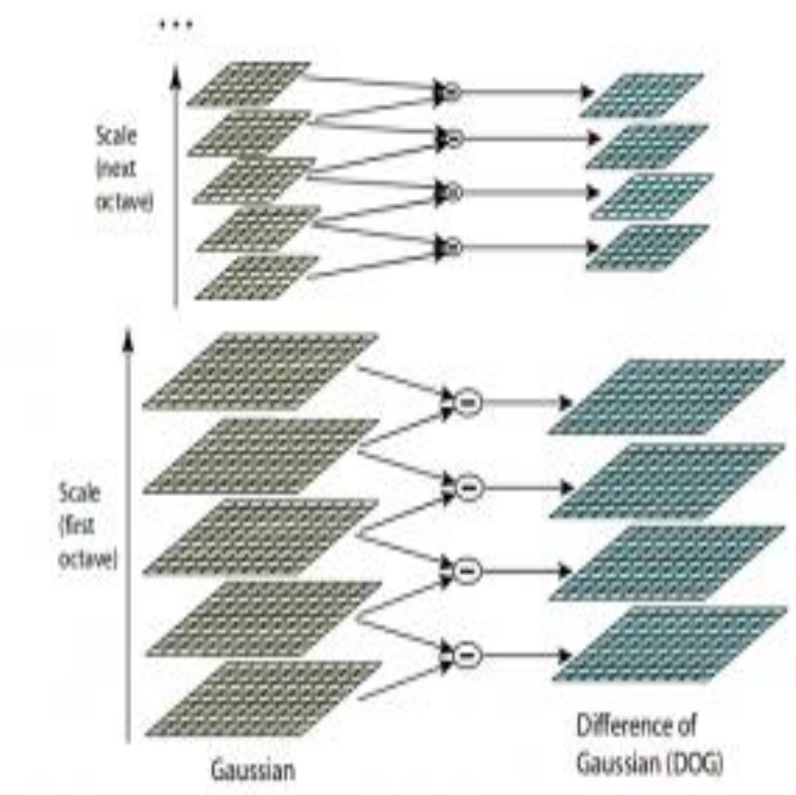


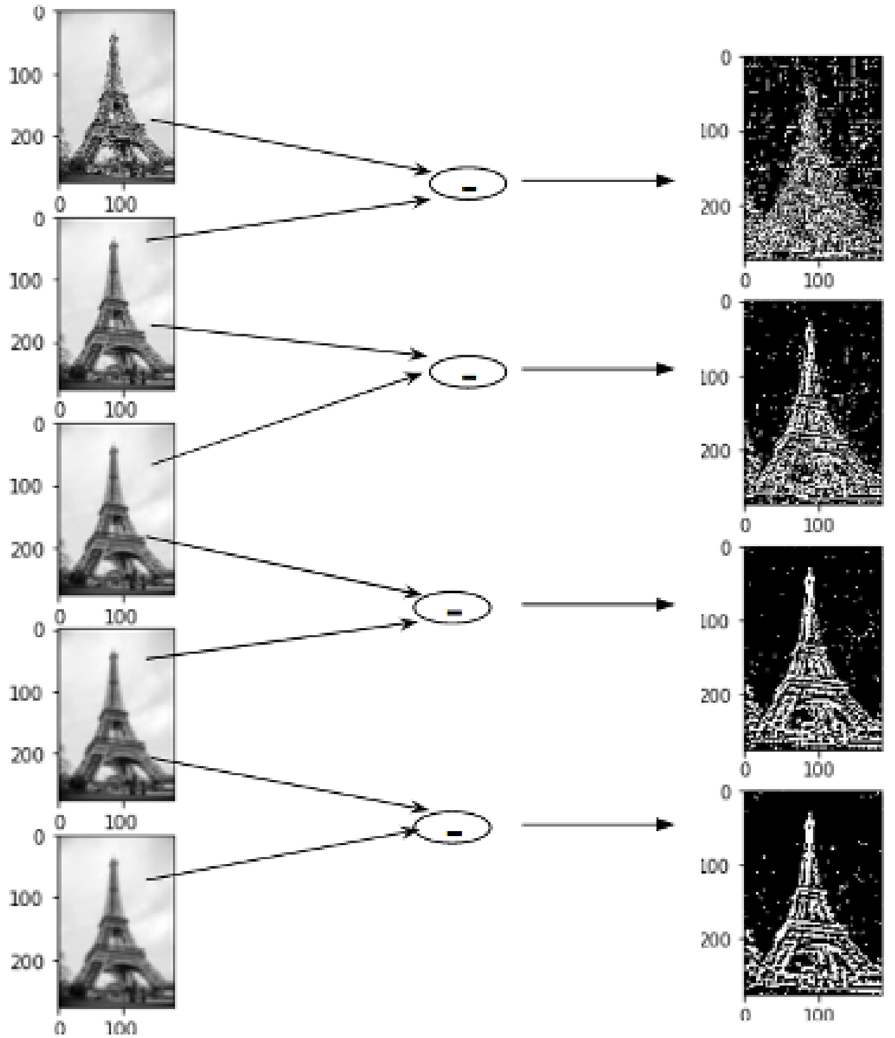
- Determine Difference of Gaussian (DoG) of two consecutive blurred images in each octave

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

- 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)



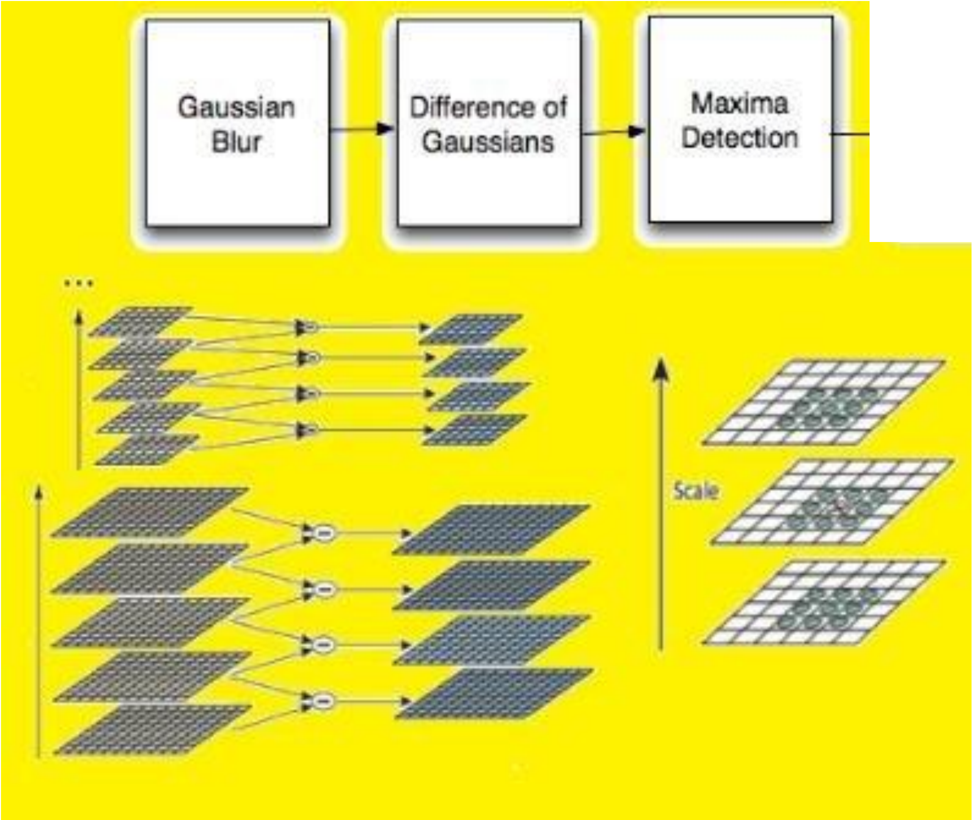


Same process is used for all the octaves

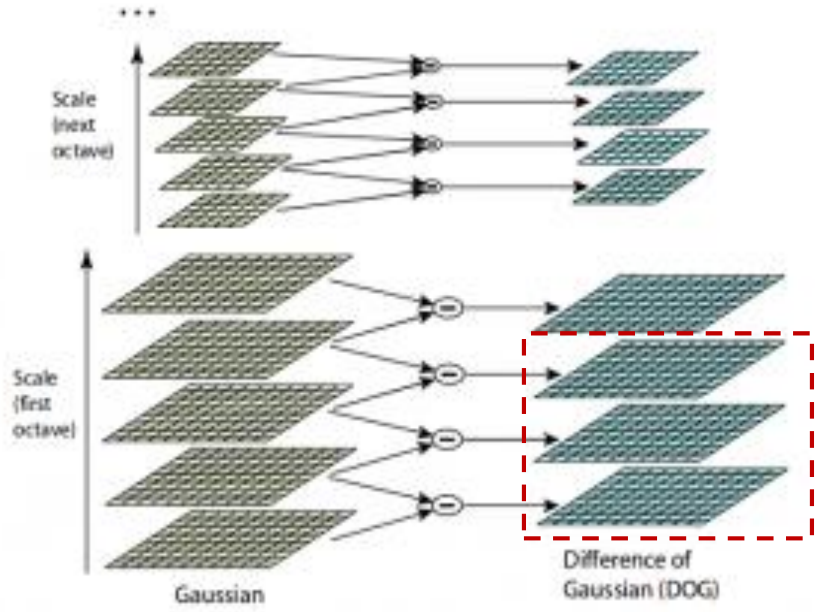
SIFT Algorithm

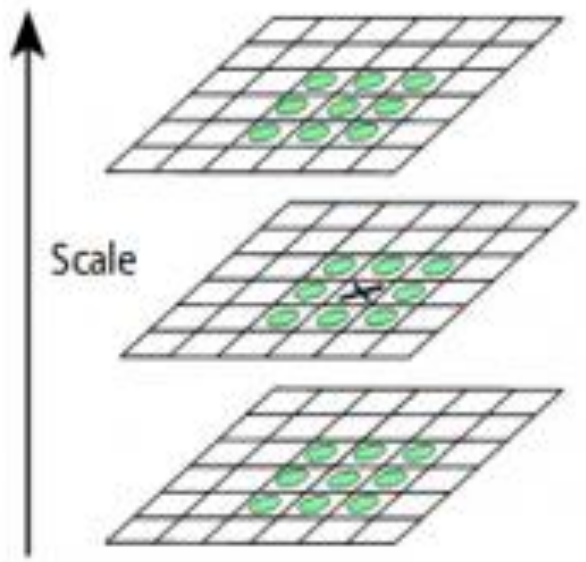
1. Construct a Scale Space:
    - Generate images over multiple scales and image locations
    - To ensures that features are scale-independent
2. Key point Localisation:
    - Select key points based on measure of stability
    - Ignore other key points to avoid false keypoints
3. Orientation Assignment:
    - 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)





Three DoGs of first octave

- 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

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

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

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

  - Select key point if

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

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

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