SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

18CSE481T - APPLIED MACHINE LEARNING

Unit-IV

Unit IV - Syllabus

- Image Content Analysis
- Computer Vision
- Operating on images using OpenCV- Python
- learn to extract and load the image
- Detecting edges Histogram equalization
- Sobel filter, Laplacian edge detector, Canny edge detector
- Histogram equalization
- Visualize gray scale image
- Detecting corners
- Understand the output corner detection image
- Detecting SIFT feature points
- SIFT feature detection

Computer Vision

Introduction

- Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.

- If AI enables computers to think, computer vision enables them to see, observe and understand.

Image Content Analysis

Introduction

- Computer Vision is a field that studies how to process, analyze, and understand the contents of visual data.
- In image content analysis, we use a lot of Computer Vision algorithms to build our understanding of the objects in the image.
- Computer Vision covers various aspects of image analysis, such as object recognition, shape analysis, pose estimation, 3D modeling, visual search, and so on.
- Humans are really good at identifying and recognizing things around them!
- The ultimate goal of Computer Vision is to accurately model the human vision system using computers.

Computer Vision Hierarchy

Computer vision is divided into three basic categories that are as following:

- **Low-level vision**: includes process image for feature extraction.
- **Intermediate-level vision**: includes object recognition and 3D scene Interpretation
- **High-level vision**: includes conceptual description of a scene like activity, intention and behavior.

Computer Vision - Related Fields

Computer Vision overlaps significantly with the following fields:

- **Image Processing**: it focuses on image manipulation.
- **Pattern Recognition:** it studies various techniques to classify patterns.
- **Photogrammetry:** it is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

- Image processing studies image to image transformation. The input and output of image processing are both images.
- Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Examples of established computer vision tasks

- **Image classification** sees an image and can classify it (a dog, an apple, a person's face). More precisely, it is able to accurately predict that a given image belongs to a certain class. For example, a social media company might want to use it to automatically identify and segregate objectionable images uploaded by users.

- **Object detection** can use image classification to identify a certain class of image and then detect and tabulate their appearance in an image or video. Examples include detecting damages on an assembly line or identifying machinery that requires maintenance.

Examples of established computer vision tasks

- **Object tracking** follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, other cars and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.

- **Content-based image retrieval** uses computer vision to browse, search and retrieve images from large data stores, based on the content of the images rather than metadata tags associated with them. This task can incorporate automatic image annotation that replaces manual image tagging. These tasks can be used for digital asset management systems and can increase the accuracy of search and retrieval.

OpenCV

- We will use a library, called OpenCV, to analyze images.
- **OpenCV** is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human.

- Machine Learning - Image Content Analysis | The University of AI (exploreai.org)

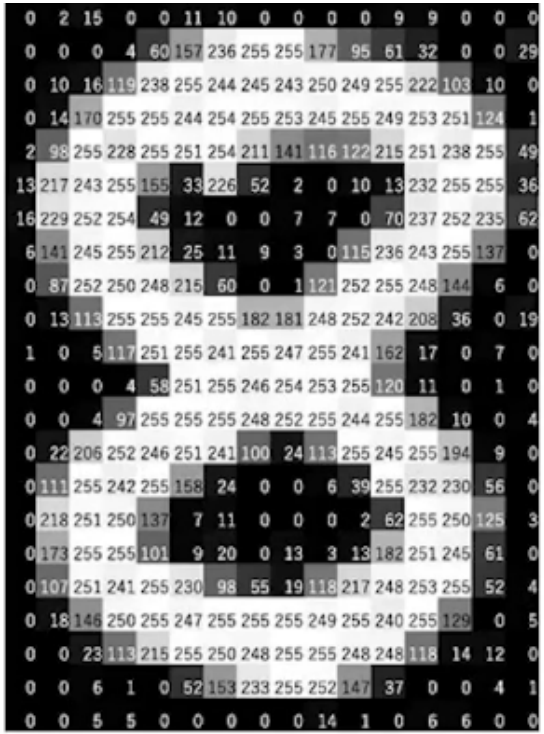
How Images are stored in the computer

- This is the image of a number 8.

- Now, if we zoom in further and if you look closely you can see that the images getting distorted and you would see some small square boxes on this image.

OpenCV

- These small boxes are called Pixels. We often use the term- the dimension of the image is X x Y
- This means that the dimension of the image is simply the number of pixels across the height(x) and width(y) of the image
- In this case, if you count, it would be 24 pixels across the height and 16 pixels across the width. Hence the dimension of this image will be 24 x 16. Although we see an image in this format the computer store image in the form of numbers

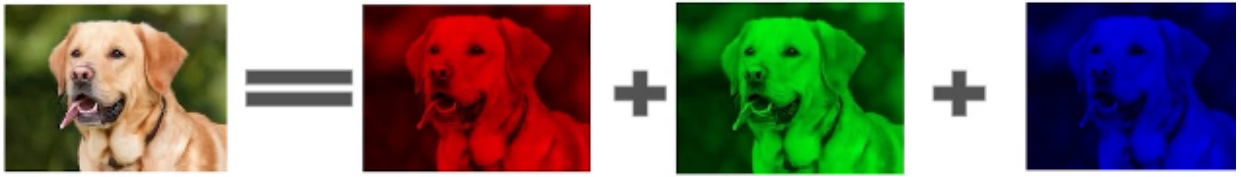


- Each of these pixels is denoted as the numerical value and these numbers are called **Pixel Values**. These pixel values denote the intensity of the pixels. For a grayscale or b&w image, we have pixel values ranging from 0 to 255.
- The smaller numbers closer to zero represent the darker shade while the larger numbers closer to 255 represent the lighter or the white shade.
- So every image in a computer is saved in this form where you have a matrix of numbers and this matrix is also known as a Channel-

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

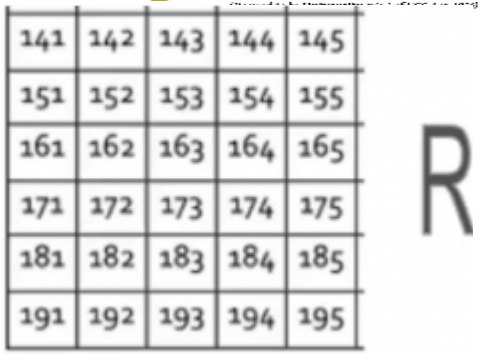
- **How colored Images are stored**

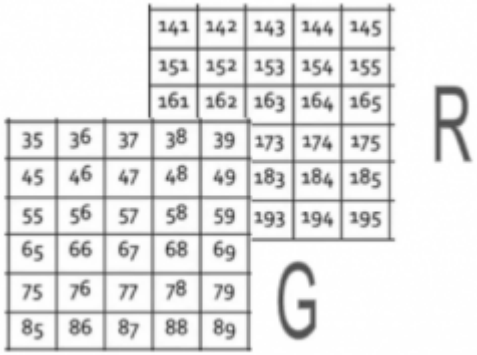
- This image is composed of many colors and almost all colors can be generated from the three primary colors- Red, Green, and Blue. We can say that each colored image is composed of these three colors or 3 channels- Red, Green, and Blue-

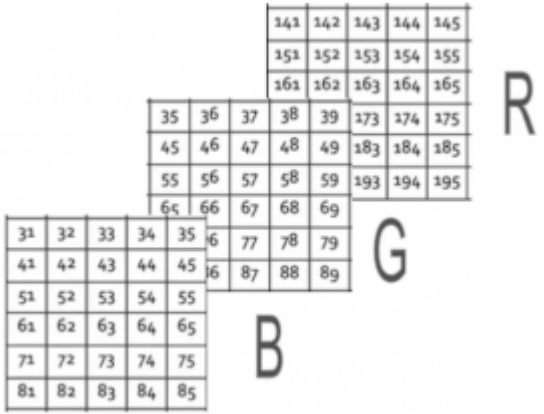


- This means that in a colored image the number of matrices or the number of channels will be more. In this particular example, we have 3 matrices- 1 matrix for red known as Red channel-

- another metrics for green known as the Green channel- & Blue Channel







Each of these metrics would again have values ranging from 0 to 255 where each of these numbers represents the intensity of the pixels or you can say that the shades of red, green, and blue. Finally, all of these channels or all of these matrices are superimposed so the shape of the image, when loaded in a computer, will be-

$$N \times M \times 3$$

where N is the number of pixels across the height, M would be the number of pixels across the width, and 3 is representing the number of channels.

Introduction to Imagenet

- To work with computer vision problems such as object recognition, action detection etc the primary requirement is **suitable dataset** to train our model .
- Due to lack of proper dataset to Machine learning and AI algorithms, it was limited research only; the business world did not find much interest in AI back then.
- In 2006, Fei Fei Li started ImageNet under the hood of WordNet.
- ImageNet is the biggest image dataset containing more than 14 million images of more than 20000 different categories having 27 high-level subcategories containing at least 500 images each. All of these images are manually annotated by the ImageNet developers, and over 1million images contain the bounding boxes around the object in the picture. In 1.2 million pictures SIFT(Scale-Invariant Feature Transform) is provided, which gives a lot of information regarding features in an image.

ImageNet Challenge

- From 2010 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which is a global annual contest held where software programs(mostly these are Convnets) compete for image classification and detection of objects and scenes.
- ILSVRC uses the smaller portion of the ImageNet consisting of only 1000 categories. The total count of **training images is 1.3 million**, accompanied by **50,000 validation images**, and **1,00,000 testing images**.
- The models participating in this competition have to perform object detection and image classification tasks at large scale and models that achieve the **minimal top-1 and top-5 error rates**
- Within six years, the error rate came down from 26% to 2.25%, which is a huge achievement

Edge Detection

- Edge detection is an image processing technique for finding the boundaries of an object in the given image
- An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. There are three types of edges:
    - Horizontal edges
    - Vertical edges
    - Diagonal edges

Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing like

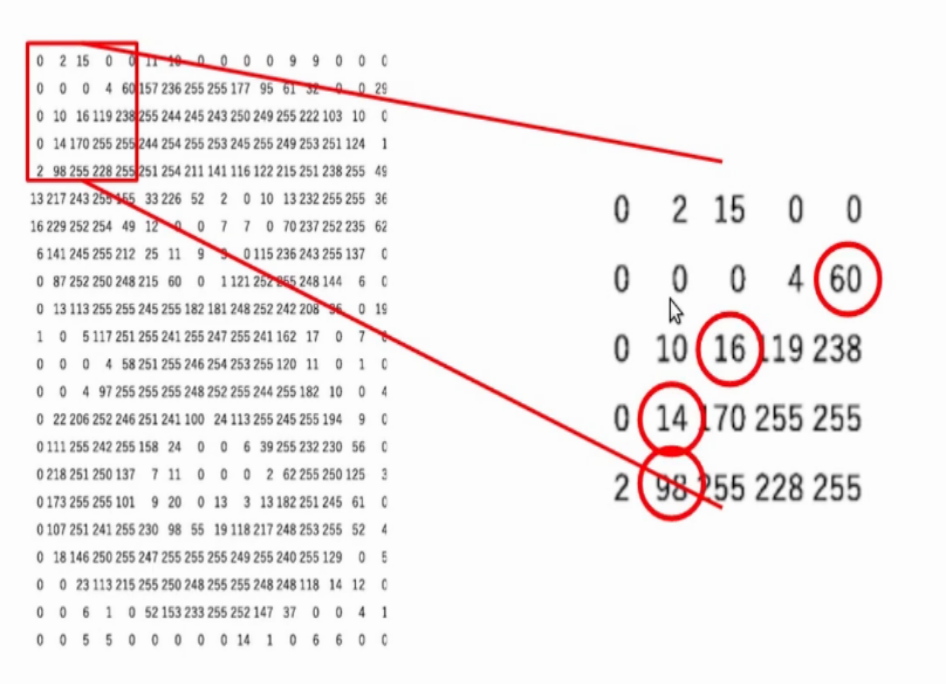
- pattern recognition
- image morphology
- feature extraction

- Edge detection allows users to observe the features of an image for a significant change in the gray level. This texture indicating the end of one region in the image and the beginning of another.

How to Extract the Edges From An Image?

- we take a small part of the image. We can compare the pixel values with its surrounding pixels, to find out if a particular pixel lies on the edge.

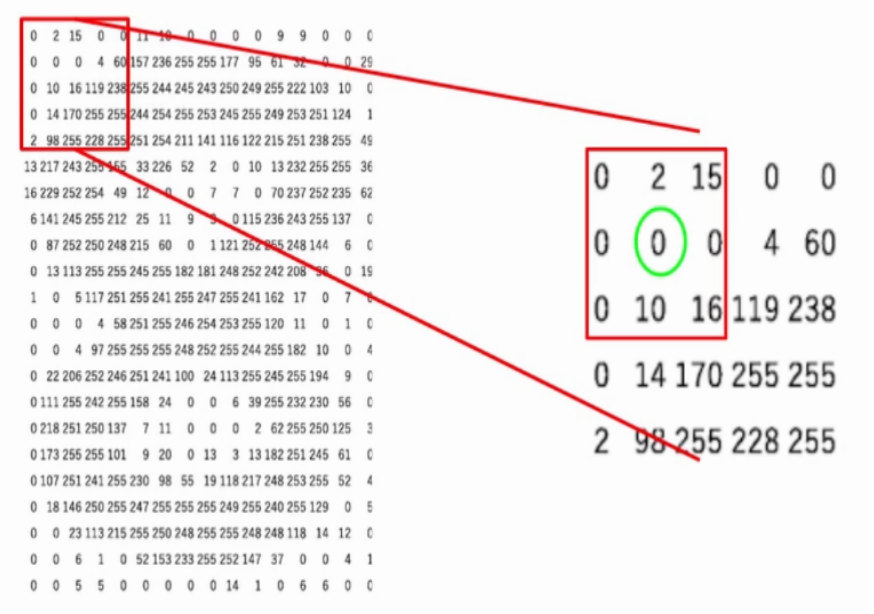
- **Example1**, take the target pixel 16 and compare the values at its left and right. Here the values are 10 and 119 respectively. Clearly, there is a significant change in the pixel values. So, we can say the pixel lies on the edge.



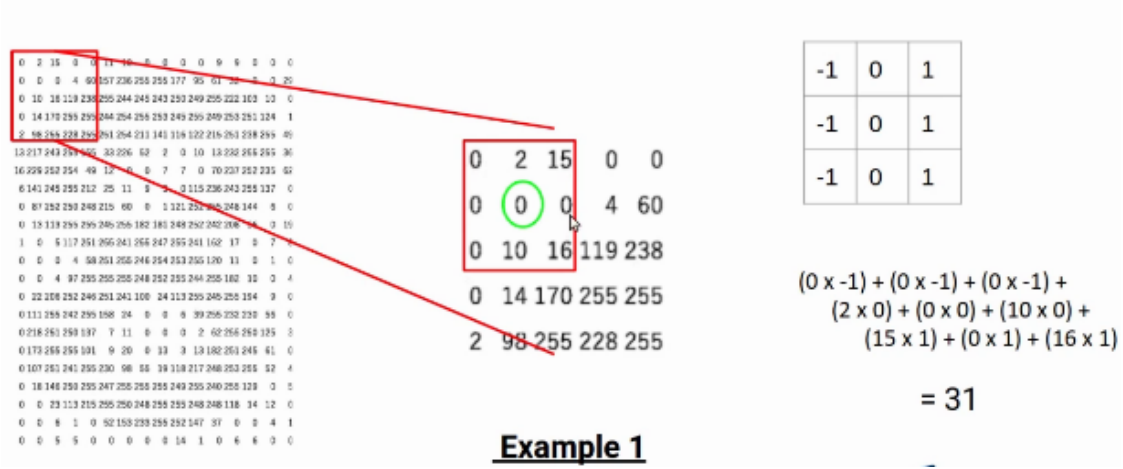
How to Extract the Edges From An Image?

Example 2: look at the pixels in the image. The pixel values to the left and the right of the selected pixel don't have a significant difference. Hence we can say that this pixel is not at the edge.

Question: How to compare these values?
Answer: Use a matrix known as **the kernel** and perform the element-wise multiplication.



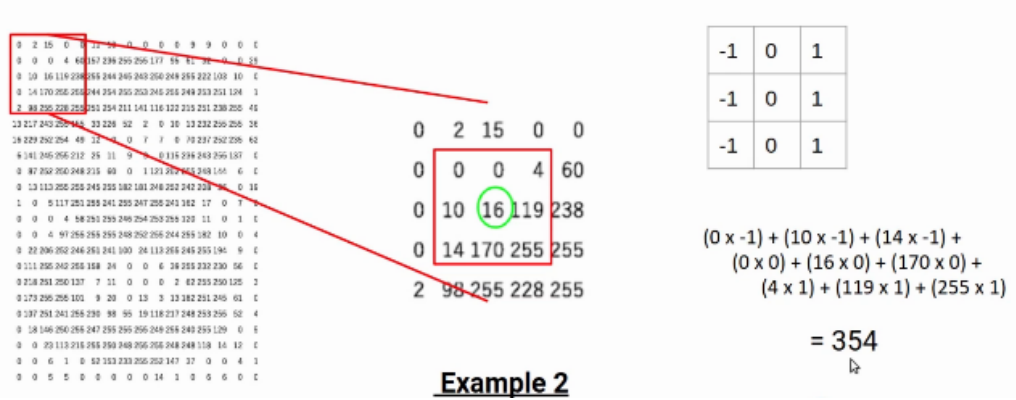
How to Extract the Edges From An Image?



In the selected portion of the image, for example, all the numbers on left are multiplied with -1, all the numbers on right with 1. Also all the numbers in the middle row with 0. In simple terms, we are trying to find the difference between the left and right pixels. When this difference is higher than a threshold, we can conclude it's an edge.

In the above case, the number is 31 which is not a large number. Hence this pixel doesn't lie on edge.

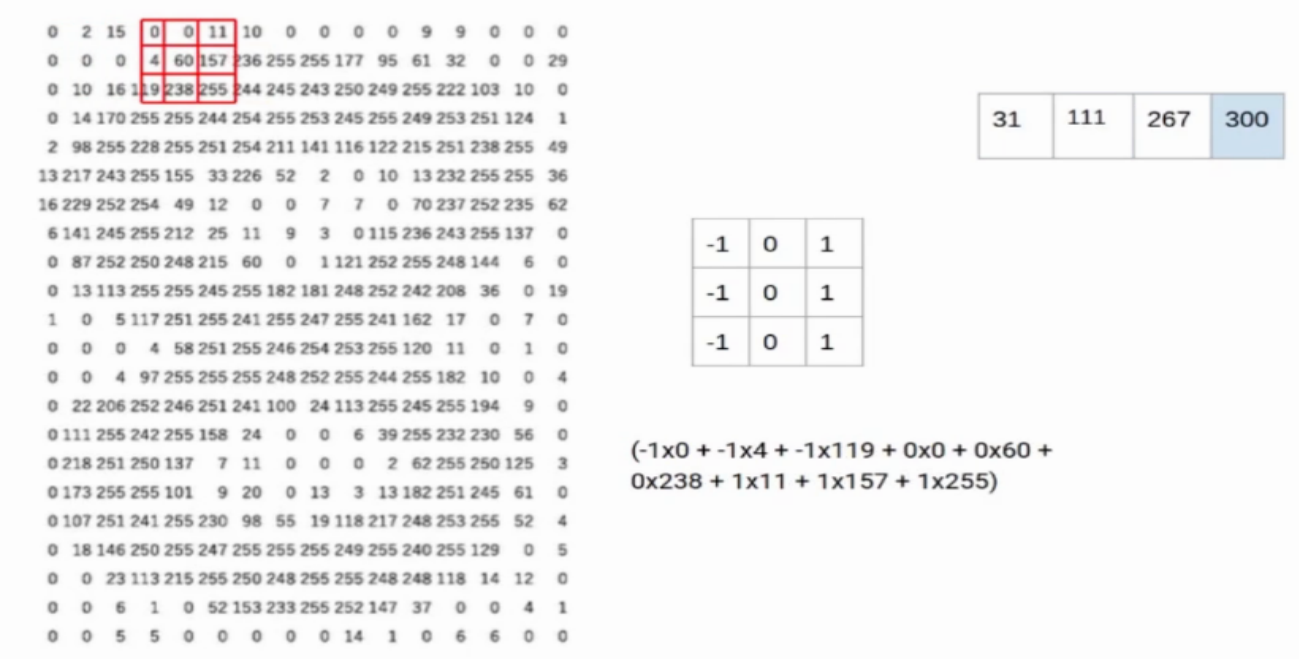
How to Extract the Edges From An Image?



In this example, the result is 354 which is significantly high. Hence, we can say that the given pixel lies on edge.

Filter/kernel

This matrix, that we use to calculate the difference is known as the filter or the kernel. This filter slides through the image to generate a new matric called a **feature map**. The values of the feature map tell, the particular pixel lies on the edge or not.



The kernel we used in the above example is called the **Prewitt kernel** in the X-direction. Since it compares the values on the horizontal axis. Similarly, have a Prewitt kernel in Y-direction. Also, we have the Sobel kernel in X and Y directions.

-1	0	1
-1	0	1
-1	0	1

Prewitt Kernel
X Direction

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

Sobel Kernel
X Direction

-1	-1	-1
0	0	0
1	1	1

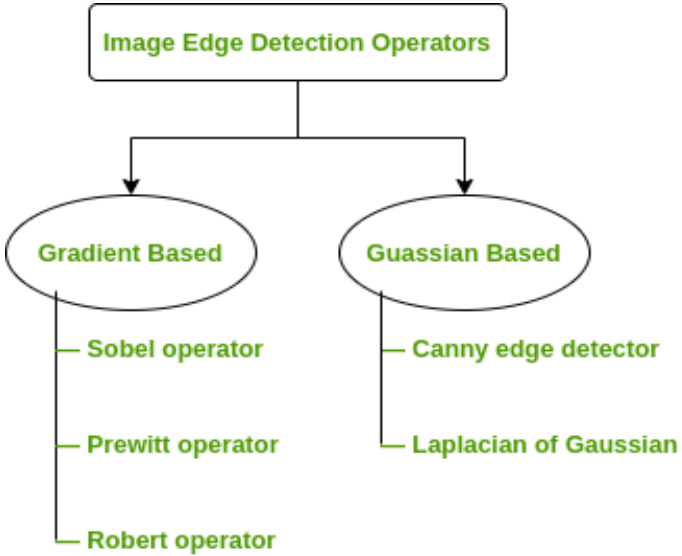
Prewitt Kernel
Y Direction

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

Sobel Kernel
Y Direction

In the case of Sobel kernels, higher importance is given to the pixel values right next to the target pixel.

Edge Detection Operators

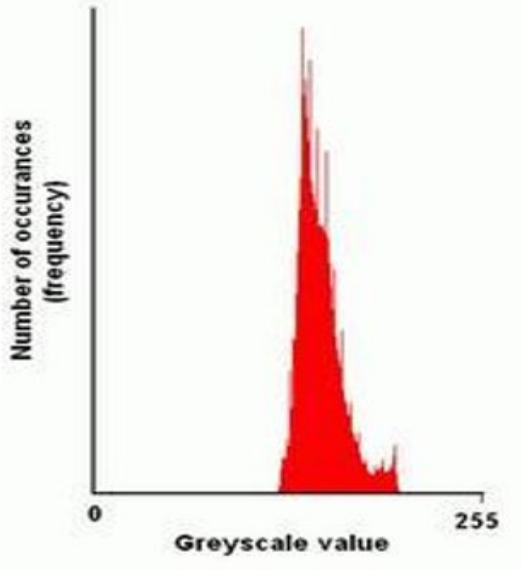


- **Gradient –** based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator, Robert operator
- **Gaussian –** based operator which computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian

Histogram Equalization

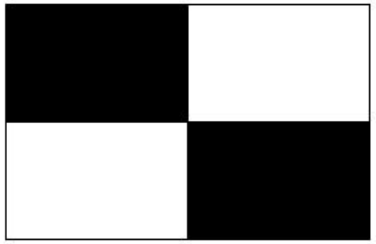
What is Histogram

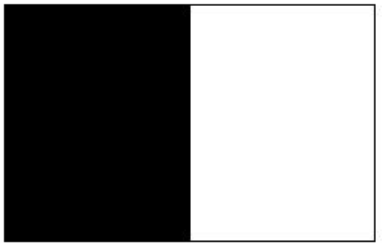
- A histogram shows the number of pixels for each intensity value in a given image
- A histogram is a statistical representation of an image.
- It doesn't show any information about where the pixels are located in the image.
- Therefore, two different images can have equivalent histograms.



What is Histogram

For example, the two images below are different but have identical histograms because both are 50% white (grayscale value of 255) and 50% black (grayscale value of 0).

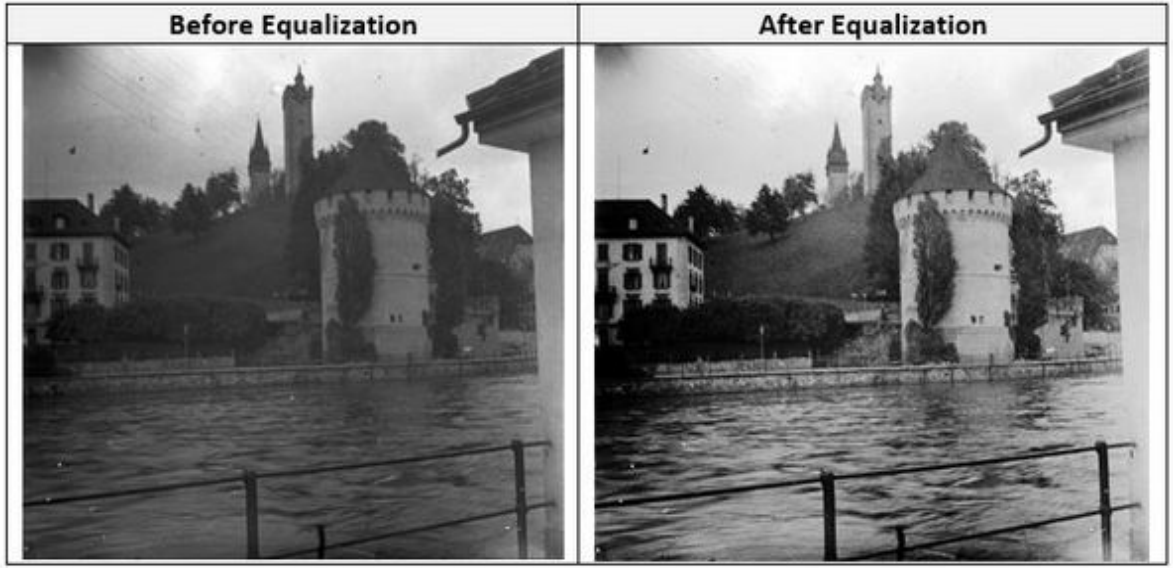




Histogram Equalization

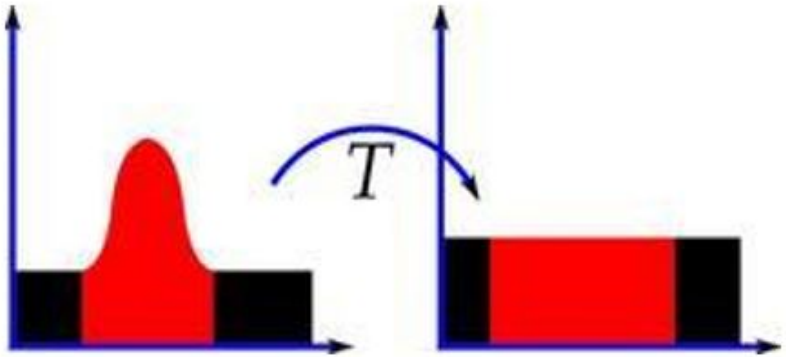
Histogram equalisation is a nonlinear process aimed to highlight image brightness in a way particularly suited to human visual analysis

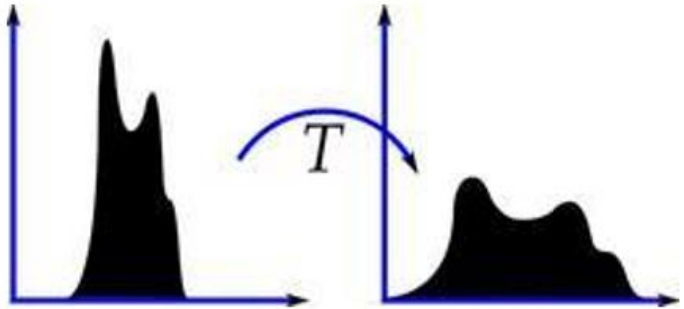
In **histogram equalization** (also known as histogram flattening), the goal is to improve contrast in images that might be either blurry or have a background and foreground that are either both bright or both dark. Histogram equalization helps sharpen an image.



Histogram Equalization

- The two images below are two examples of what the histogram for an input image might look like before and after it goes through histogram equalization.
- Histogram equalization is useful in a number of real-world use cases, such as x-rays, thermal imagery, and satellite photos.





Histogram Equalization Example

Perform histogram equalization on the following 8 X 8 image. The gray level distribution of the image is given below

Gray Level (r_k)	0	1	2	3	4	5	6	7
No.of Pixels(p_k)	8	10	10	2	12	16	4	2

Total number of pixels = 8+10+10+2+12+16+4+2 = 64

Histogram Equalization

Gray Level (k=8 Total Levels)	Count	Probability Density Function (Count/Total Count)	Cumulative Distribution Function (CDF)	(k-1)*CDF	Floor(k-1)*CDF
0	8	0.125	0.125	0.875	1
1	10	0.15625	0.28125	1.96875	2
2	10	0.15625	0.4375	3.0625	3
3	2	0.03125	0.46875	3.28125	3
4	12	0.1875	0.65625	4.59375	5
5	16	0.25	0.90625	6.34375	6
6	4	0.0625	0.96875	6.78125	7
7	2	0.03125	1	7	7

Histogram Equalization Example

Resultant Equalized Histogram

Gray Level (r_k)	0	1	2	3	4	5	6	7
No.of Pixels(p_k)	0	8	10	10+2	0	12	16	4+2

Types

- Global histogram equalization (GHE)
- Local histogram equalization (LHE)

CLAHE (Contrast Limited Adaptive Histogram Equalization) - considers the global contrast of the image

Histogram Equalization of Color Images

- Histogram of the color images follows a different procedure.
- Histogram equalization only applies to the intensity channel.
- An RGB image consists of three color channels, and we cannot apply the histogram equalization process on these channels separately.
- We need to separate the intensity information from the color information before we do anything.
- So, we convert it to YUV colorspace first, equalize the Y channel, and then convert it back to RGB to get the output.

Histogram Equalization of Color Images

The Y component determines the brightness of the color (referred to as luminance or luma)

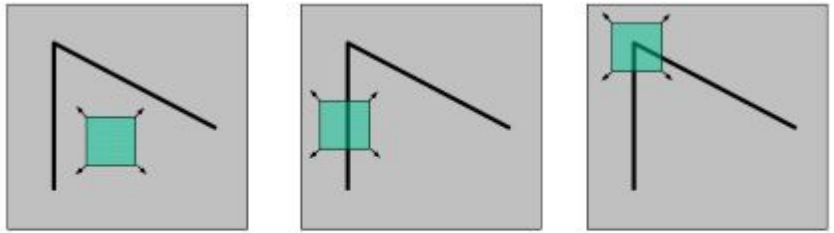
The U and V components determine the color itself (the chroma).

Y ranges from 0 to 1 (or 0 to 255 in digital formats), while U and V range from -0.5 to 0.5 (or -128 to 127 in signed digital form, or 0 to 255 in unsigned form).

One neat aspect of YUV is that we can throw out the U and V components and get a grey-scale image

Corner Detection

Corner:
A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness.



"flat" region:
no change in all directions

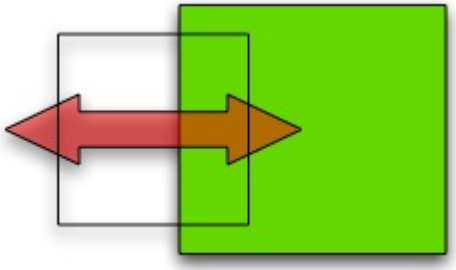
"edge":
no change along the edge direction

"corner":
significant change in all directions

Corner Detection

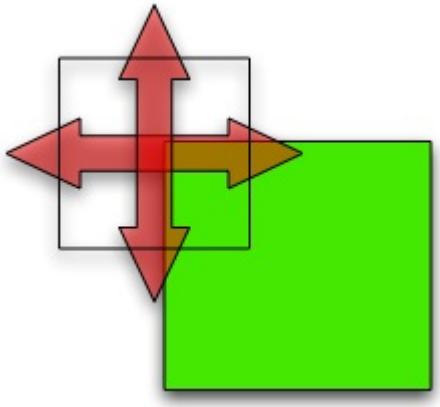
Corner detection works on the principle that if we place a small window over an image, if that window is placed on a corner then if it is moved in any direction there will be a large change in intensity.

This is illustrated below with some diagrams.
If the window is over a flat area of the image then there will be obviously be no intensity change when the window moves. If the window is over an edge there will only be an intesity change if the window moves in one direction.



Corner Detection

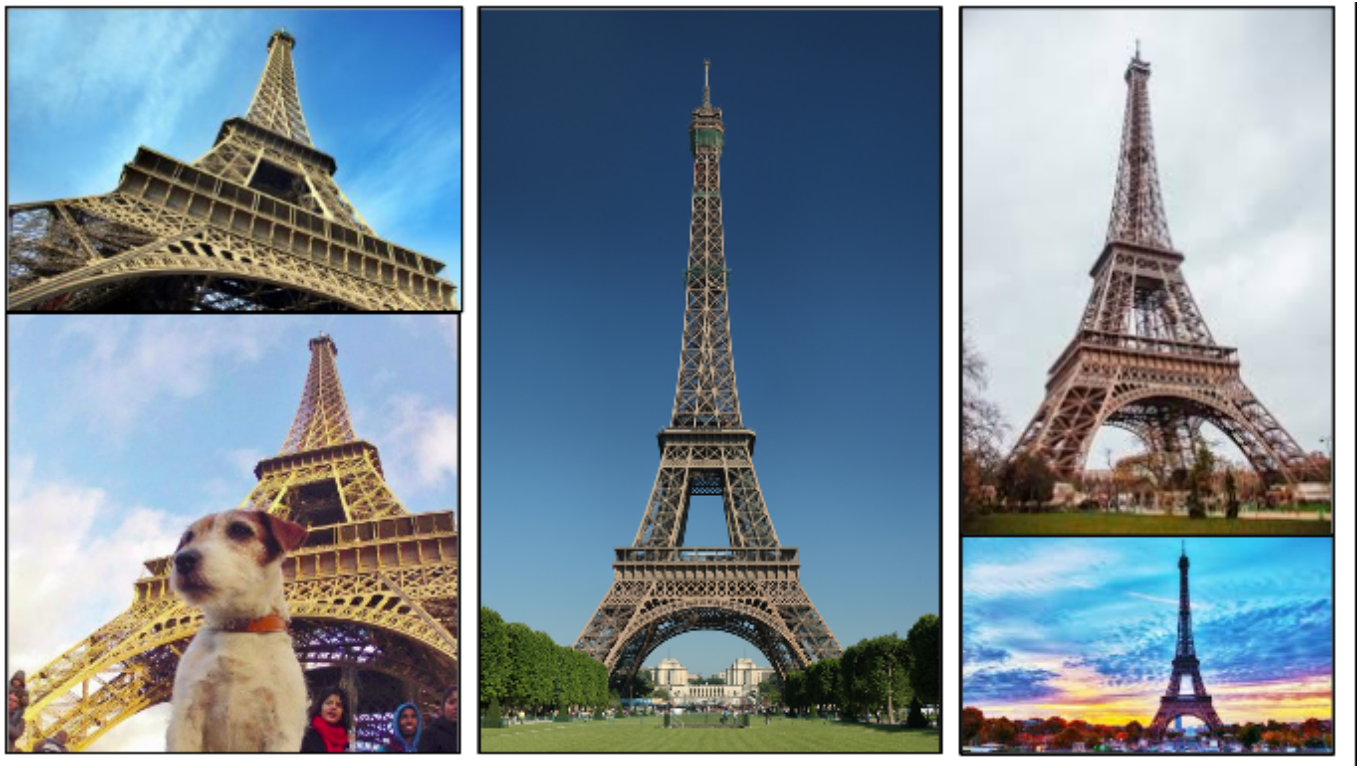
If the window is over a corner then there will be a change in all directions, and therefore we know there must be a corner.
The Harris corner detector, measures the strength of detected corners, and only marks those above a given strength as actual corners. The number detected can be altered by varying the value of k.



Week 6: Corner Detection (sbme-tutorials.github.io)

SIFT (Scale Invariant Feature Transform)

Introduction



SIFT (Scale Invariant Feature Transform)

Introduction:

- The collection of images of resplendent Eiffel Tower is shown here. Image has a different background, is captured from different angles, and also has different objects in the foreground. Since we have seen the images of the Eiffel Tower multiple times no matter the image is rotated at a weird angle or zoomed in to show only half of the Tower, our memory easily recalls its features.
- But for machines it is challenge for them to identify the object in an image if we change certain things (like the angle or the scale).
- SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm in Computer Vision
- SIFT helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

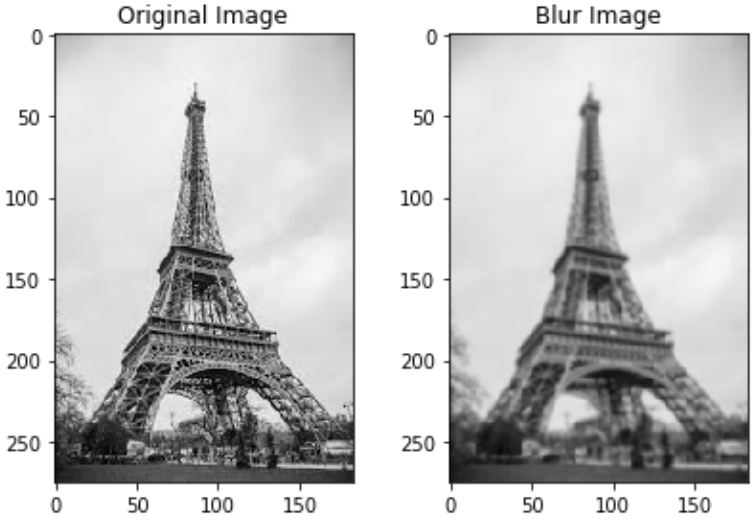
SIFT (Scale Invariant Feature Transform)

- keypoints generated using SIFT can be used as features for the image during model training.
- **The major advantage of SIFT features, over edge features or corner features, is that they are not affected by the size or orientation of the image.**
- The entire process can be divided into 4 parts:
  - **Constructing a Scale Space:** To make sure that features are scale-independent
  - **Keypoint Localisation:** Identifying the suitable features or keypoints
  - **Orientation Assignment:** Ensure the keypoints are rotation invariant
  - **Keypoint Descriptor:** Assign a unique fingerprint to each keypoint
- Finally, we can use these keypoints for feature matching!
- *Ref: original paper by David G. Lowe. Link: Distinctive Image Features from Scale-Invariant Keypoints*

Constructing the Scale Space

- Identify the most distinct features in a given image while ignoring any noise
- Additionally, we need to ensure that the features are not scale-dependent.
- We use the **Gaussian Blurring technique** to reduce the noise in an image
- for every pixel in an image, the Gaussian Blur calculates a value based on its neighboring pixels
- we need to ensure that these features must not be scale-dependent.
- we will be searching for these features on multiple scales, by creating a 'scale space'.

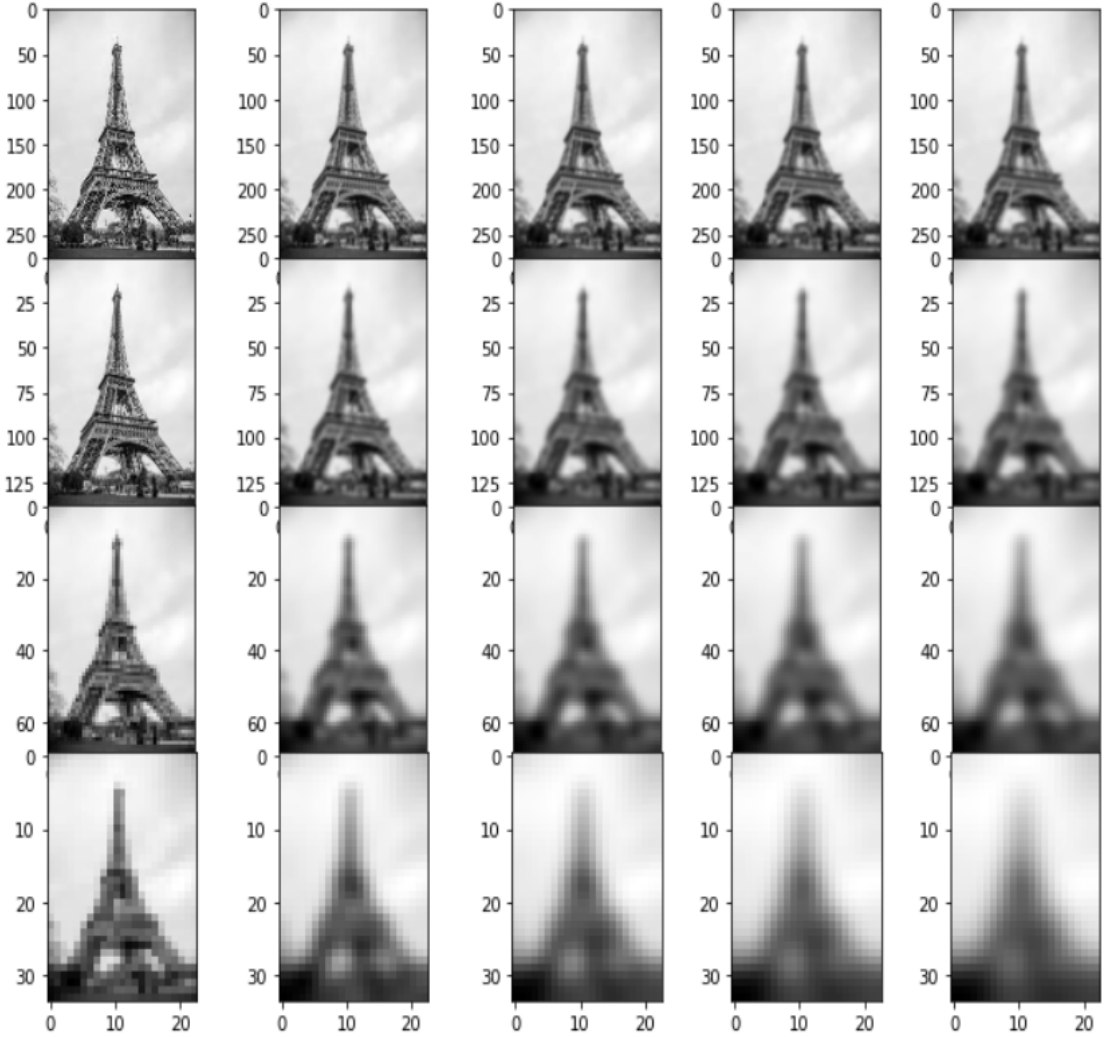
Example image before and after Gaussian Blur



The texture and minor details are removed from the image and only the relevant information like the shape and edges remain

Constructing the Scale Space

- Scale space is a collection of images having different scales, generated from a single image.
- To create a new set of images of different scales, we will take the original image and reduce the scale by half.
- The ideal number of octaves should be four, and for each octave, the number of blur images should be five.
- We have the original image of size (20, 250) and a scaled image of dimension (20, 30).



First Octave

we have created images of multiple) and used Gaussian blur for each of them to reduce the noise in the image

Second Octave

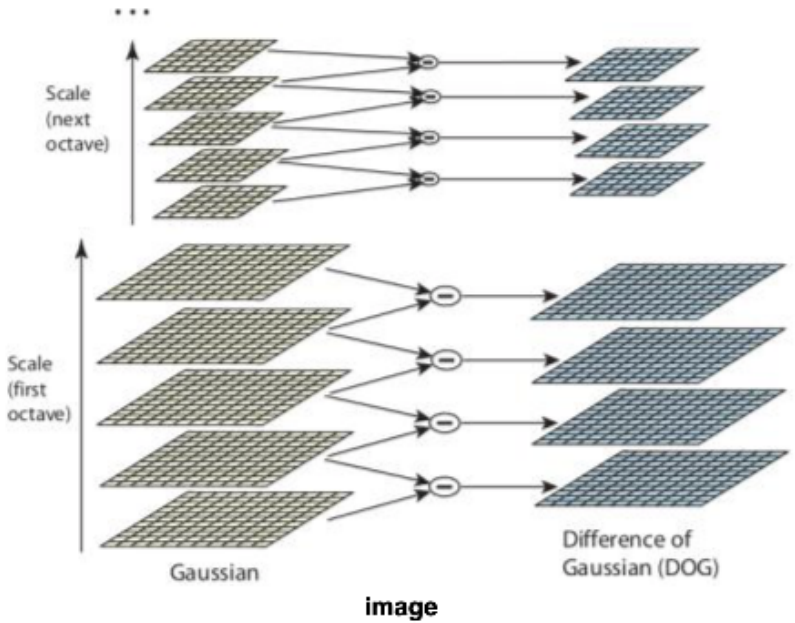
Third Octave

Fourth Octave

Constructing the Scale Space

Difference of Gaussian:

It is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original.



image

Constructing the Scale Space

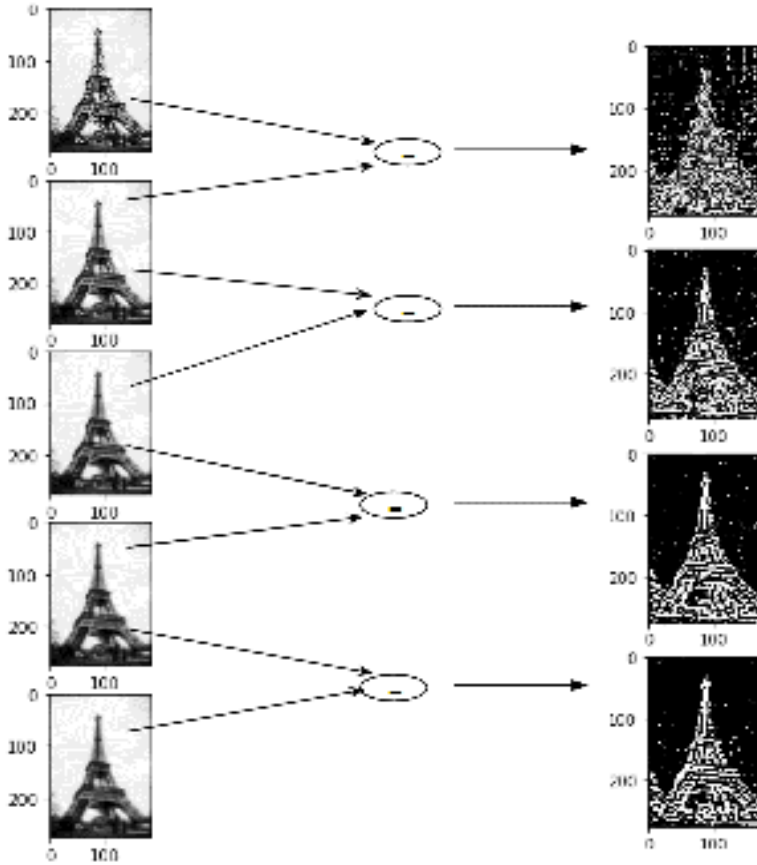
Difference of Gaussian:

we have 5 images, all from the first octave (thus having the same scale). Each subsequent image is created by applying the Gaussian blur over the previous image.

On the left, we have 5 images, all from the first octave (thus having the same scale). Each subsequent image is created by applying the Gaussian blur over the previous image.

On the right, we have four images generated by subtracting the consecutive Gaussians.

Shown here is only for the first octave but the same process happens for all the octaves.



Keypoint Localization

Next step is to find the important keypoints from the image that can be used for feature matching. The idea is to find the local maxima and minima for the images. This part is divided into two steps:

- Find the local maxima and minima
- Remove low contrast keypoints (keypoint selection)

Local Maxima and Local Minima

To locate the local maxima and minima, we go through every pixel in the image and compare it with its neighboring pixels

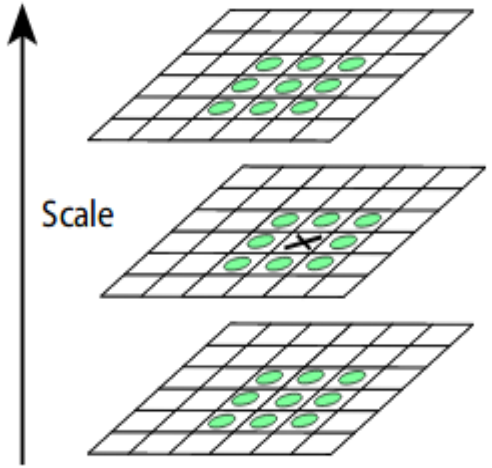
'neighboring' - this not only includes the surrounding pixels of that image (in which the pixel lies), but also the nine pixels for the previous and next image in the octave.

Keypoint Localization

Every pixel value is compared with 26 other pixel values to find whether it is the local maxima/minima.

Example: , we have three images from the first octave. The pixel marked x is compared with the neighboring pixels (in green) and is selected as a keypoint if it is the highest or lowest among the neighbors

We now have potential keypoints that represent the images and are scale-invariant. We will apply the last check over the selected keypoints to ensure that these are the most accurate keypoints to represent the image.



Keypoint Selection

- we will eliminate the keypoints that have low contrast, or lie very close to the edge
- To deal with the low contrast keypoints, a second-order Taylor expansion is computed for each keypoint. If the resulting value is less than 0.03 (in magnitude), it is rejected
- Remaining keypoints whether they are poorly located keypoints. These are the keypoints that are close to the edge and have a high edge response but may not be robust to a small amount of noise. A second-order Hessian matrix is used to identify such keypoints.
- We have performed both the contrast test and the edge test to reject the unstable keypoints, we will now assign an orientation value for each keypoint to make the rotation invariant.

Orientation Assignment

we have a set of stable keypoints for the images. We will now assign an orientation to each of these keypoints so that they are invariant to rotatio
can again divide this step into two smaller steps:

1. Calculate the magnitude and orientation
2. Create a histogram for magnitude and orientation

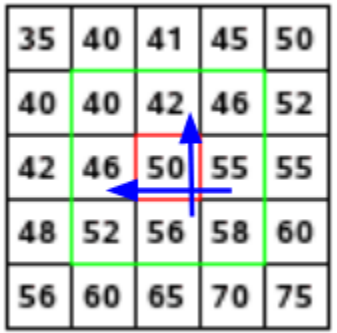
Calculate Magnitude and Orientation

Consider the sample image shown :

find the magnitude and orientation for the pixel value in red

For this, we will calculate the gradients in x and y directions by taking th
difference between 55 & 46 and 56 & 42.

This comes out to be Gx = 9 and Gy = 14 respectively.

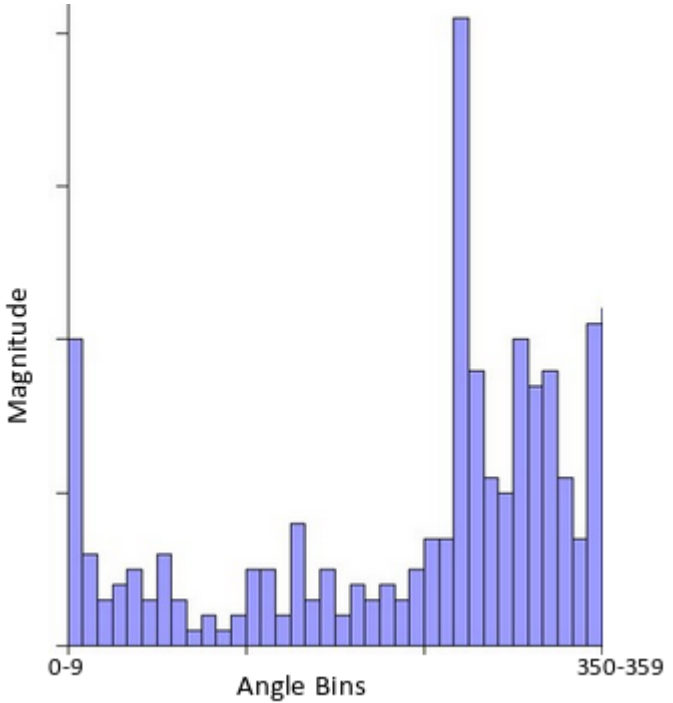


- Once we have the gradients, we can find the magnitude and orientation using the following formulas:Magnitude = √[(Gx)2+(Gy)2] = 16.64
- Φ = atan(Gy / Gx) = atan(1.55) = 57.17

The magnitude represents the intensity of the pixel and the orientation gives the direction for the same.

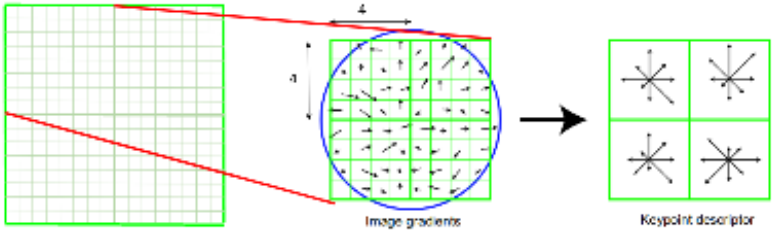
Creating a Histogram for Magnitude and Orientation

- This histogram would peak at some point. The bin at which the peak will be the orientation for the keypoint. Additionally, if there is another significant peak (seen between 80 – 100%), then another keypoint is generated with the magnitude and scale the same as the keypoint used to generate the histogram. And the angle or orientation will be equal to the new bin that has the peak.
- Effectively at this point, we can say that there can be a small increase in the number of keypoints.



Keypoint Descriptor

- This is the final step for SIFT. So far, we have stable keypoints that are scale-invariant and rotation invariant. In this section, we will use the neighboring pixels, their orientations, and magnitude, to generate a unique fingerprint for this keypoint called a 'descriptor'.
- Additionally, since we use the surrounding pixels, the descriptors will be partially invariant to illumination or brightness of the images.
- We will first take a 16×16 neighborhood around the keypoint. This 16×16 block is further divided into 4×4 sub-blocks and for each of these sub-blocks, we generate the histogram using magnitude and orientation.
- At this stage, the bin size is increased and we take only 8 bins (not 36). Each of these arrows represents the 8 bins and the length of the arrows define the magnitude. So, we will have a total of 128 bin values for every keypoint.



Building a Star feature detector

- Star Feature Detector is derived from CenSurE (Center Surrounded Extrema) detector. While CenSurE uses polygons such as Square, Hexagon and Octagons as a more computable alternative to circle.
- Star mimics the circle with 2 overlapping squares: 1 upright and 1 45-degree rotated.
- * It works in gray scale only *