COMPUTER VISION (AM3209PE)

UNIT-I

Computer Vision Introduction

Computer vision is a subfield of artificial intelligence that deals with acquiring, processing, analyzing, and making sense of visual data such as digital images and videos. It is one of the most compelling types of artificial intelligence that we regularly implement in our daily routines.

Computer vision helps to understand the complexity of the human vision system and trains computer systems to interpret and gain a high-level understanding of digital images or videos. In the early days, developing a machine system having human-like intelligence was just a dream, but with the advancement of artificial intelligence and machine learning, it also became possible. Similarly, such intelligent systems have been developed that can "see" and interpret the world around them, similar to human eyes. The fiction of yesterday has become the fact of today. In this tutorial, **"Computer Vision Introduction",** we will discuss a few important concepts of computer vision, such as:

- **What is Computer Vision?**
- **How does Computer Vision Work?**
- **The evolution of computer vision**
- **Applications of computer vision**
- **Challenges of computer vision**

What is Computer Vision?

Computer vision is one of the most important fields of artificial intelligence (AI) and computer science engineering that makes computer systems capable of extracting meaningful information from visual data like

videos and images. Further, it also helps to take appropriate actions and make recommendations based on the extracted information.

Further, Artificial intelligence is the branch of computer science that primarily deals with creating a smart and intelligent system that can behave and think like the human brain. So, we can say if artificial intelligence enables computer systems to think intelligently, computer vision makes them capable of seeing, analyzing, and understanding.

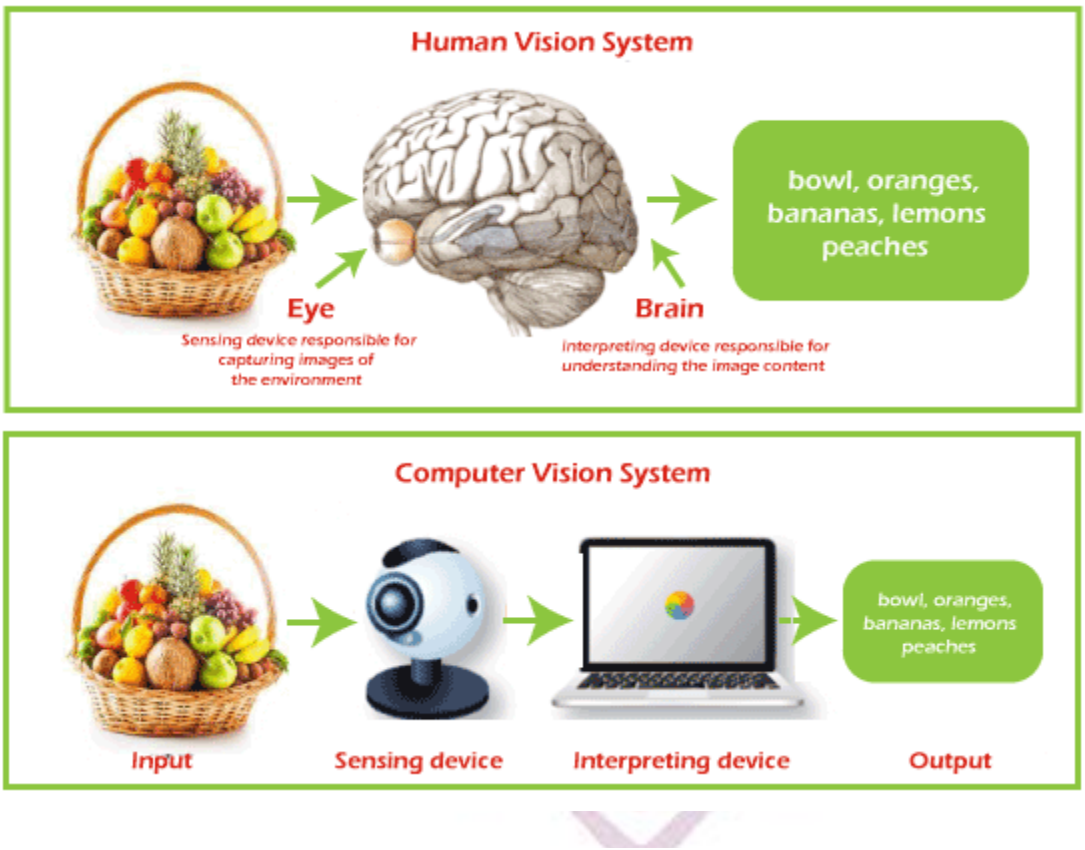
History of Computer Vision

Computer vision is not a new technology because scientists and experts have been trying to develop machines that can see and understand visual data for almost six decades. The evolution of computer vision is classified as follows:

- **1959:** The first experiment with computer vision was initiated in 1959, where they showed a cat as an array of images. Initially, they found that the system reacts first to hard edges or lines, and scientifically, this means that image processing begins with simple shapes such as straight edges.
- **1960:** In 1960, artificial intelligence was added as a field of academic study to solve human vision problems.
- **1963:** This was another great achievement for scientists when they developed computers that could transform 2D images into 3-D images.
- **1974:** This year, optical character recognition (OCR) and intelligent character recognition (ICR) technologies were successfully discovered. The OCR has solved the problem of recognizing text printed in any font or typeface, whereas ICR can decrypt handwritten text. These inventions are one of the greatest achievements in document and invoice processing, vehicle number plate recognition, mobile payments, machine translation, etc.
- **1982:** In this year, the algorithm was developed to detect edges, corners, curves, and other shapes. Further, scientists also developed a network of cells that could recognize patterns.
- **2000:** In this year, scientists worked on a study of object recognition.
- **2001:** The first real-time face recognition application was developed.
- **2010:** The ImageNet data set became available to use with millions of tagged images, which can be considered the foundation for recent Convolutional Neural Network (CNN) and deep learning models.
- **2012:** CNN has been used as an image recognition technology with a reduced error rate.
- **2014:** COCO has also been developed to offer a dataset for object detection and support future research.

How does Computer Vision Work?

Computer vision is a technique that extracts information from visual data, such as images and videos. Although computer vision works similarly to human eyes with brain work, this is probably one of the biggest open questions for IT professionals: How does the human brain operate and solve visual object recognition?

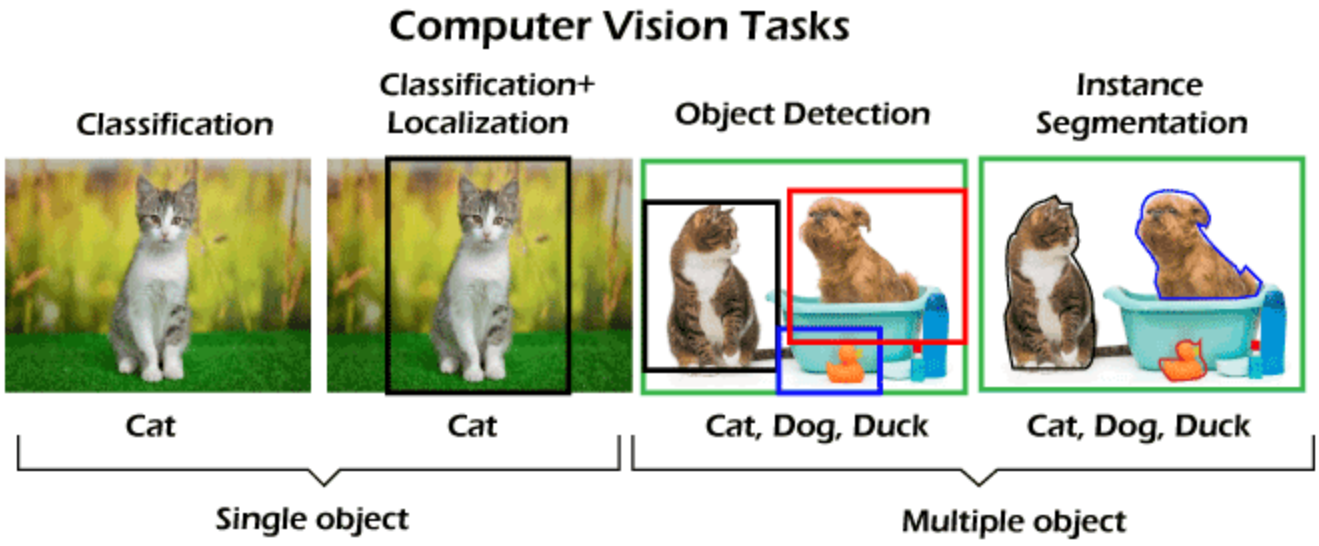


On a certain level, computer vision is all about pattern recognition which includes the training process of machine systems for understanding the visual data such as images and videos, etc.

Firstly, a vast amount of visual labeled data is provided to machines to train it. This labeled data enables the machine to analyze different patterns in all the data points and can relate to those labels. E.g., suppose we provide visual data of millions of dog images. In that case, the computer learns from this data, analyzes each photo, shape, the distance between each shape, color, etc., and hence identifies patterns similar to dogs and generates a model. As a result, this computer vision model can now accurately detect whether the image contains a dog or not for each input image.

Task Associated with Computer Vision

Although computer vision has been utilized in so many fields, there are a few common tasks for computer vision systems. These tasks are given below:



- **Object classification: Object classification is a** computer vision technique/task used to classify an image, such as whether an image contains a dog, a person's face, or a banana. It analyzes the visual content (videos & images) and classifies the object into the defined category. It means that we can accurately **predict the class of an object present in an image with image classification.**
- **Object Identification/detection:** Object identification or detection uses image classification to identify and locate the objects in an image or video. With such detection and identification technique, the system can count objects in a given image or scene and determine their accurate location and labeling. For example, in a given image, one dog, one cat, and one duck can be easily detected and classified using the object detection technique.
- **Object Verification:** The system processes videos, finds the objects based on search criteria, and tracks their movement.
- **Object Landmark Detection:** The system defines the key points for the given object in the image data.
- **Image Segmentation:** Image segmentation not only detects the classes in an image as image classification; instead, it classifies each pixel of an image to specify what objects it has. It tries to determine the role of each pixel in the image.
- **Object Recognition:** In this, the system recognizes the object's location with respect to the image.

How to learn computer Vision?

Although, computer vision requires all basic concepts of machine learning, deep learning, and artificial intelligence. But if you are eager to learn computer vision, then you must follow below things, which are as follows:

1. **Build your foundation:**

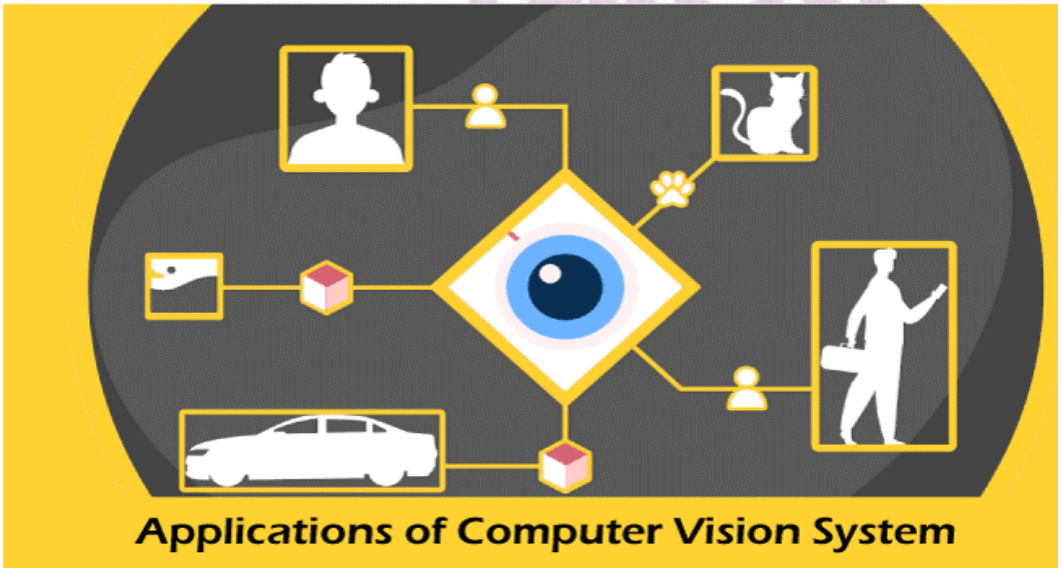
- Before entering this field, you must have strong knowledge of advanced mathematical concepts such as Probability, statistics, linear algebra, calculus, etc.
- The knowledge of programming languages like Python would be an extra advantage to getting started with this domain.

2. **Digital Image Processing:**
   It would be best if you understood image editing tools and their functions, such as histogram equalization, median filtering, etc. Further, you should also know about compressing images and videos using JPEG and MPEG files. Once you know the basics of image processing and restoration, you can kick-start your journey into this domain.
3. **Machine learning understanding**
   To enter this domain, you must deeply understand basic machine learning concepts such as **CNN, neural networks, SVM, recurrent neural networks, generative adversarial neural networks, etc.**
4. **Basic computer vision:** This is the step where you need to decrypt the mathematical models used in visual data formulation.

These are a few important prerequisites that are essentially required to start your career in computer vision technology. Once you are prepared with the above prerequisites, you can easily start learning and make a career in Computer vision.

Applications of computer vision

Computer vision is one of the most advanced innovations of artificial intelligence and machine learning. As per the increasing demand for AI and Machine Learning technologies, computer vision has also become a center of attraction among different sectors. It greatly impacts different industries, including retail, security, healthcare, automotive, agriculture, etc.



Below are some most popular applications of computer vision:

- **Facial recognition:** Computer vision has enabled machines to detect face images of people to verify their identity. Initially, the machines are given input data images in which computer vision algorithms detect facial features and compare them with databases of fake profiles. Popular social media platforms like Facebook also use facial recognition to detect and tag users. Further, various government spy agencies are employing this feature to identify criminals in video feeds.
- **Healthcare and Medicine:** Computer vision has played an important role in the healthcare and medicine industry. Traditional approaches for evaluating cancerous tumors are time-consuming and have less accurate predictions, whereas computer vision technology provides faster and more accurate chemotherapy response assessments; doctors can identify cancer patients who need faster surgery with life-saving precision.
- **Self-driving vehicles:** Computer vision technology has also contributed to its role in self-driving vehicles to make sense of their surroundings by capturing video from different angles around the car and then introducing it into the software. This helps to detect other cars and objects, read traffic signals, pedestrian paths, etc., and safely drive its passengers to their destination.
- **Optical character recognition (OCR)** Optical character recognition helps us extract printed or handwritten text from visual data such as images. Further, it also enables us to extract text from documents like invoices, bills, articles, etc.
- **Machine inspection:** Computer vision is vital in providing an image-based automatic inspection. It detects a machine's defects, features, and functional flaws, determines inspection goals, chooses lighting and material-handling techniques, and other irregularities in manufactured products.
- **Retail (e.g., automated checkouts):** Computer vision is also being implemented in the retail industries to track products, shelves, wages, record product movements into the store, etc. This AI-based computer vision technique automatically charges the customer for the marked products upon checkout from the retail stores.
- **3D model building:** 3D model building or 3D modeling is a technique to generate a 3D digital representation of any object or surface using the software. In this field also, computer vision plays its role in constructing 3D computer models from existing objects. Furthermore, 3D modeling has a variety of applications in various places, such as Robotics, Autonomous driving, 3D tracking, 3D scene reconstruction, and AR/VR.
- **Medical imaging:** Computer vision helps medical professionals make better decisions regarding treating patients by developing visualization of specific body parts such as organs and tissues. It helps them get more accurate diagnoses and a better patient care system. E.g., Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scanner to diagnose pathologies or guide medical interventions such as surgical planning or for research purposes.
- **Automotive safety:** Computer vision has added an important safety feature in automotive industries. E.g., if a vehicle is taught to detect objects and dangers, it could prevent an accident and save thousands of lives and property.

- **Surveillance:** It is one of computer vision technology's most important and beneficial use cases. Nowadays, CCTV cameras are almost fitted in every place, such as streets, roads, highways, shops, stores, etc., to spot various doubtful or criminal activities. It helps provide live footage of public places to identify suspicious behavior, identify dangerous objects, and prevent crimes by maintaining law and order.
- **Fingerprint recognition and biometrics:** Computer vision technology detects fingerprints and biometrics to validate a user's identity. Biometrics deals with recognizing persons based on physiological characteristics, such as the face, fingerprint, vascular pattern, or iris, and behavioral traits, such as gait or speech. It combines Computer Vision with knowledge of human physiology and behavior.

How to become a computer vision engineer?

Computer vision is one of the world's most popular & high-demand technologies. Although starting your career in this domain is not easy, if you have a good command of machine learning basics, advanced mathematics concepts, and the basics of computer vision, you can easily start your career as a computer vision engineer.

There are some roles and responsibilities required to become a computer vision engineer, which is as follows

- To create and implement a vision algorithm for working with image and video content pixels
- To develop a data-based approach for better problem solutions.
- Whenever required, you have to work on various AI and ML tasks required for computer vision, such as image processing.
- Experience in working on various real-time project scenarios for problem-solving.
- Hierarchical problem decomposition, implementation of solutions, and integration with other sub-systems.
- Hierarchical problem decomposition, implementation of solutions, and integration with other sub-systems.
- Should be capable of understanding business objectives and can connect to technical solutions through effective system design and architecture.

Job description (JD) for Computer vision engineer

- The candidate must have cumulative work experience in visual data processing and analysis using machine learning and deep learning.
- Hands-on experience with various AI/ML frameworks such as Python, TensorFlow, PyTorch, Keras, CPP, etc.
- Candidates must have good experience in implementing AI techniques.
- Must have good written and verbal communication skills.
- Candidates should be aware of object detection techniques and models such as YOLO, RCNN, etc.

Which programming language is best for computer vision?

Computer vision engineers require in-depth knowledge of machine learning and deep learning concepts with strong command over at least one programming language. There are so many programming languages that can be used in this domain, but Python is among the most popular. However, one can also choose OpenCV with Python, OpenCV with C++, or MATLAB to learn and implement computer vision applications.

OpenCV with Python could be the most preferred choice for beginners due to its flexibility, simple syntax, and versatility. Various reasons make Python the best programming language for computer vision, which is as follows:

- **Easy-to-use:** Python is very famous as it is easy to learn for entry-level persons and professionals. Further, Python is also easily adaptable and covers all business needs.
- **Most used programming language:** Python is one of the most popular programming languages as it contains complete learning environments to get started with machine learning, artificial intelligence, deep learning, and computer vision.
- **Debugging and visualization:** Python has an in-built facility for debugging via **'PDB'** and visualization through Matplotlib.

Computer Vision Challenges

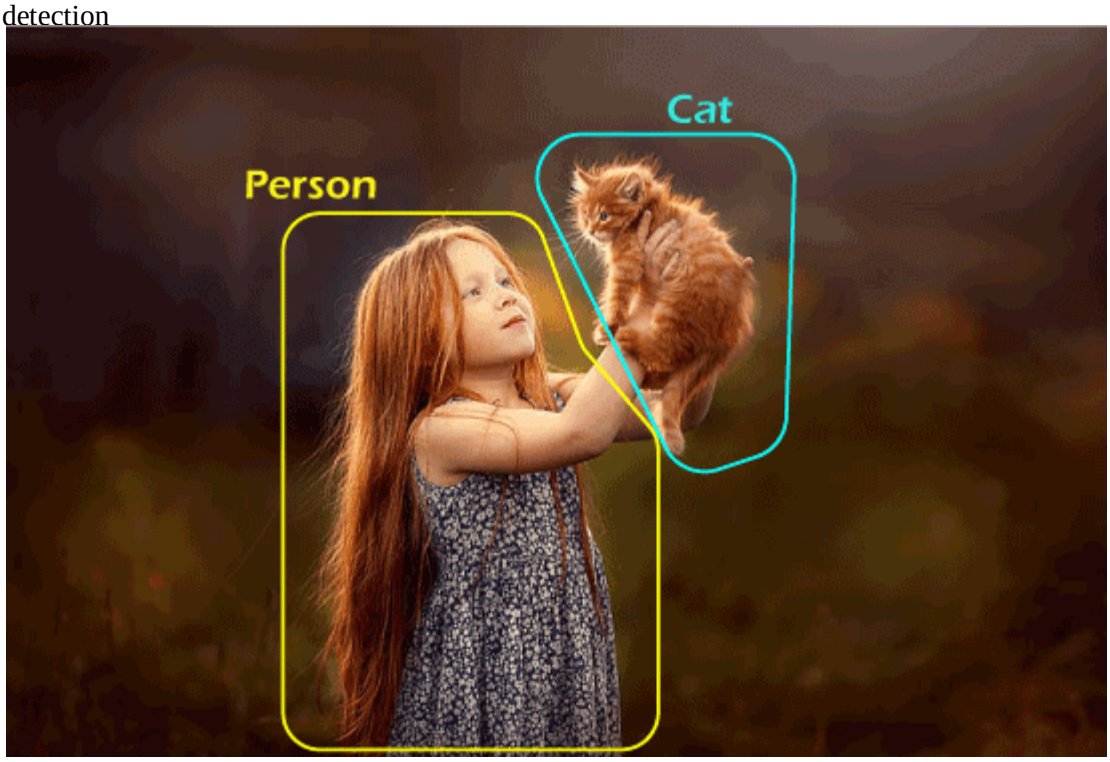
Computer vision has emerged as one of the most growing domains of artificial intelligence, but it still has a few challenges to becoming a leading technology. There are a few challenges observed while working with computer vision technology.

- **Reasoning and analytical issues**All programming languages and technologies require the basic logic behind any task. To become a computer vision expert, you must have strong reasoning and analytical skills. If you don't have such skills, then defining any attribute in visual content may be a big problem.
- **Privacy and security:** Privacy and security are among the most important factors for any country. Similarly, vision-powered surveillance is also having various serious privacy issues for lots of countries. It restricts users from accessing unauthorized content. Further, various countries also avoid such face recognition and detection techniques for privacy and security reasons.
- **Duplicate and false content:** Cyber security is always a big concern for all organizations, and they always try to protect their data from hackers and cyber fraud. A data breach can lead to serious problems, such as creating duplicate images and videos over the internet.

Below are some common tasks for which computer vision can be used:

- **Image Classification: Image classification is a** computer vision technique used to classify an image, such as whether an image contains a dog, a person's face, or a banana. It means that with image classification, we can accurately **predict the class of an object present in an image.**

- **Object Detection:** Object detection uses image classification to identify and locate the objects in an image or video. With such detection and identification technique, the system can count objects in a given image or scene and determine their accurate location, along with their labelling. For example, in a given image, there is one person and one cat, which can be easily detected and classified using the object detection technique.



- **Object Tracking:** Object tracking is a computer vision technique used to follow a particular object or multiple items. Generally, object tracking has applications in videos and real-world interactions, where objects are firstly detected and then tracked to get observation. Object tracking is used in applications such as Autonomous vehicles, where apart from object classification and detection such as pedestrians, other vehicles, etc., tracking of real-time motion is also required to avoid accidents and follow the traffic rules.
- **Semantic Segmentation**: Image segmentation is not only about detecting the classes in an image as image classification. Instead, it classifies each pixel of an image to specify what objects it has. It tries to determine the role of each pixel in the image.

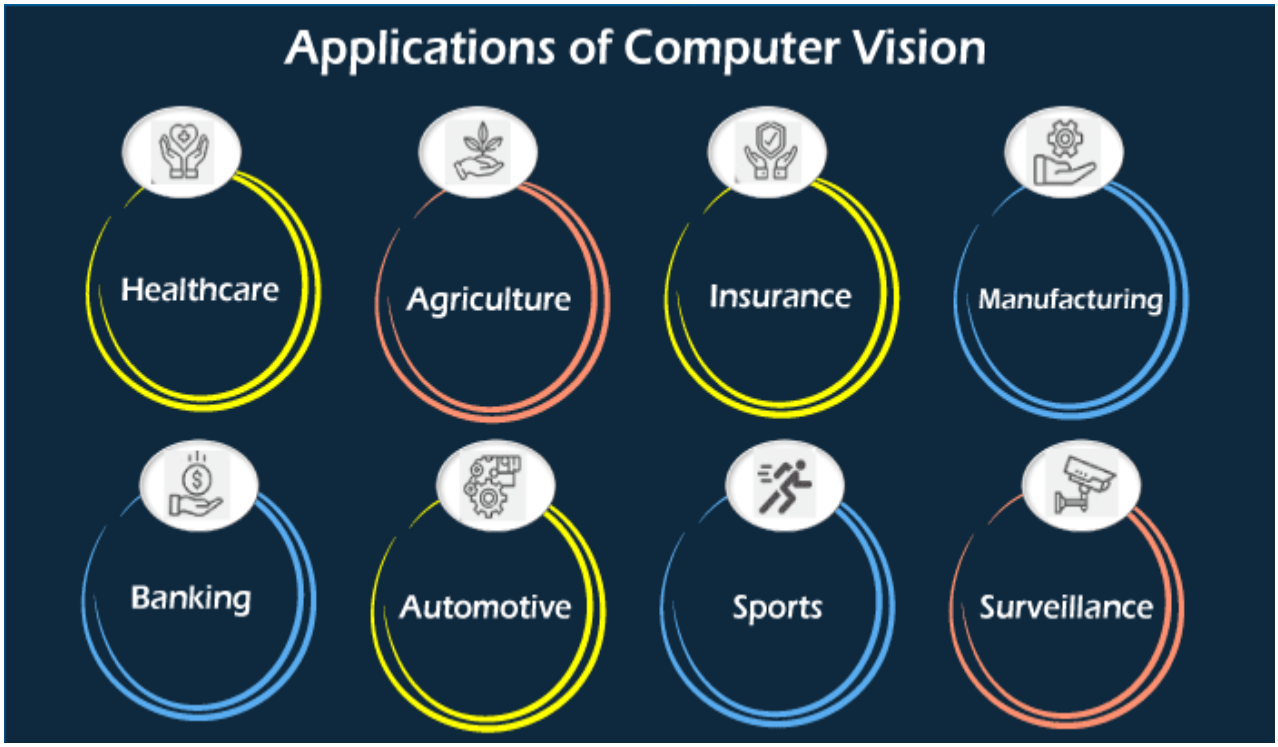
Computer Vision Applications

As per the increasing demand for AI and Machine Learning technologies, computer vision also has a great demand among different sectors. It has a massive impact on different industries, including retail, security, healthcare, automotive, agriculture, etc. Below are some most popular applications of computer vision:

- Defect detection using Computer Vision

- OCR using Computer vision
- Crop Monitoring
- Analysis of X-rays, MRI, and CT scans using Computer Vision
- Road Condition Monitoring
- 3D model Building using Computer vision
- Cancer Detection using Computer Vision
- Plant Disease Detection using Computer Vision
- Traffic Flow Analysis

Above are some most common applications of Computer vision. Now let us discuss applications of computer vision across different sectors such as Retail, healthcare, etc.



1. Computer Vision in Healthcare

The Healthcare industry is rapidly adopting new technologies and automation solutions, one of which is computer vision. In the healthcare industry, computer vision has the following applications:

- **X-Ray Analysis**
  Computer vision can be successfully applied for medical X-ray imaging. Although most doctors still prefer manual analysis of X-ray images to diagnose and treat diseases, with computer vision, X-ray analysis can be automated with enhanced efficiency and accuracy. *The state-of-art image recognition algorithm* can be used to detect patterns in an X-ray image that are too subtle for the human eyes.

- **CancerDetection**
  Computer vision is being successfully applied for breast and skin cancer detection. With image recognition, doctors can identify anomalies by comparing cancerous and non-cancerous cells in images. With automated cancer detection, doctors can diagnose cancer faster from an MRI scan.
- **CTScanandMRI**
  Computer vision has now been greatly applied in CT scans and MRI analysis. AI with computer vision designs such a system that analyses the radiology images with a high level of accuracy, similar to a human doctor, and also reduces the time for disease detection, enhancing the chances of saving a patient's life. It also includes deep learning algorithms that enhance the resolution of MRI images and hence improve patient outcomes.

2. Computer Vision in Transportation

With the enhanced demand for the transportation sector, there has occurred various technological development in this industry, and one of such technologies is Computer vision. Below are some popular applications of computer vision in the transportation industry:

- **Self-driving cars**
  Computer vision is widely used in self-driving cars. It is used to detect and classify objects (e.g., road signs or traffic lights), create 3D maps or motion estimation, and plays a key role in making autonomous vehicles a reality.
- **Pedestrian detection**
  Computer vision has great application and research in Pedestrian detection due to its high impact on the designing of pedestrian systems in various smart cities. With the help of cameras, pedestrian detection automatically identifies and locate the pedestrians in image or video. Moreover, it also considers the variations among pedestrians related to attire, body position, and illuminance in different scenarios. This pedestrian detection is very helpful in different fields such as traffic management, autonomous driving, transit safety, etc.
- **Road Condition Monitoring & Defect detection**
  Computer vision has also been applied for monitoring the road infrastructure condition by accessing the variations in concrete and tar. A computer vision-enabled system automatically senses pavement degradation, which successfully increases road maintenance allocation efficiency and decreases safety risks related to road accidents.
  To perform road condition monitoring, CV algorithms collect the image data and then process it to create automatic crack detection and classification system.

3. Computer Vision in Manufacturing

In the manufacturing industry, the demand for automation is at its peak. Many tasks have already been automated, and other new technology innovations are in trend. For providing these automatic solutions, Computer vision is also widely used. Below are some most popular applications

- **Defect Detection**
  This is perhaps, the most common application of computer vision. Until now, the detection of defects has been carried out by trained people in selected batches, and total production control is usually impossible. With computer vision, we can detect defects such as cracks in metals, paint defects, bad prints, etc., in sizes smaller than 0.05mm.
- **Analyzing text and barcodes (OCR)**
  Nowadays, each product contains a barcode on its packaging, which can be analyzed or read with the help of the computer vision technique OCR. Optical character recognition or OCR helps us detect and extract printed or handwritten text from visual data such as images. Further, it enables us to extract text from documents like invoices, bills, articles, etc. and verifies against the databases.
- **Fingerprint recognition and Biometrics**
  Computer vision technology is used to detect fingerprints and biometrics to validate a user's identity. Biometrics is the measurement or analysis of physiological characteristics of a person that make a person unique such as Face, Finger Print, iris Patterns, etc. It makes use of computer vision along with knowledge of human physiology and behaviour.
- **3D Model building**
  3D model building or 3D modelling is a technique to generate a 3D digital representation of any object or surface using the software. Computer vision plays its role here also in constructing 3D computer models from existing objects. Furthermore, 3D modelling has a variety of applications in various places, such as Robotics, Autonomous driving, 3D tracking, 3D scene reconstruction, and AR/VR.

4. Computer Vision in Agriculture

In the agriculture sector, Machine Learning has made a great contribution with its models, including Computer vision. It can be used in areas such as crop monitoring, weather analysis, etc. Below are some popular cases of computer vision applications in Agriculture:

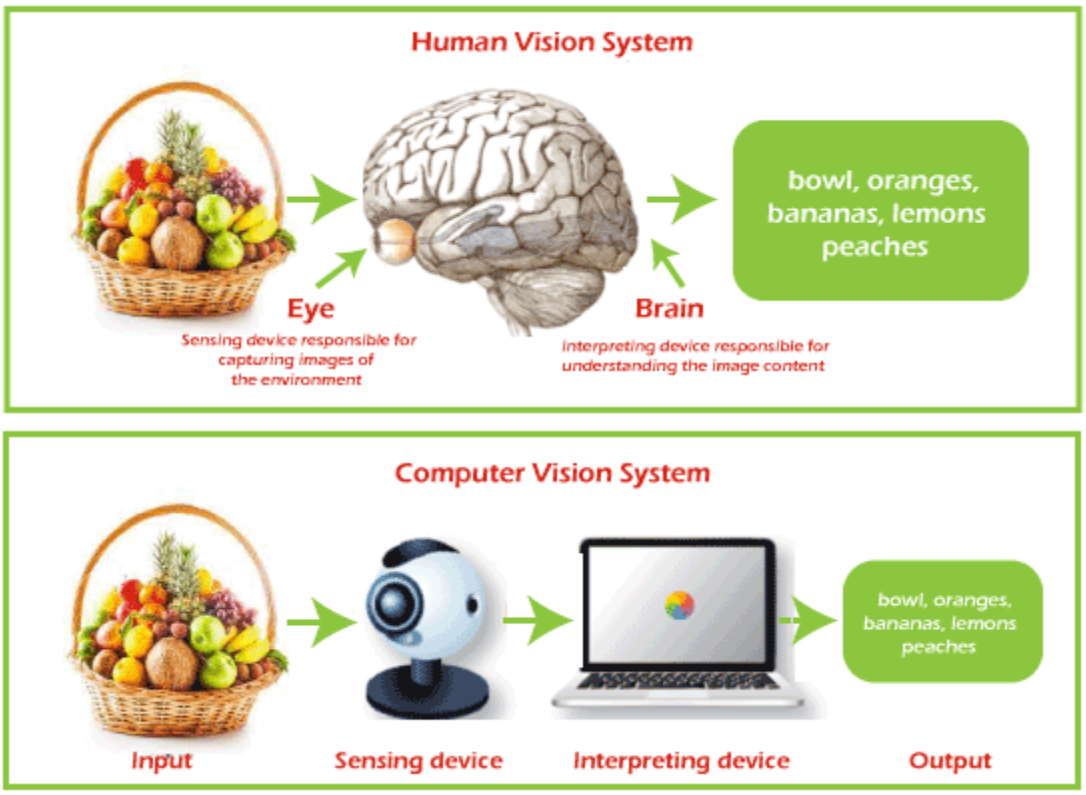
- **Crop Monitoring**
  In the agriculture sector, crop and yield monitoring are the most important tasks for better agriculture. Traditionally, it depends on subjective human judgment, but that is not always accurate. With computer vision systems, real-time crop monitoring and identification of any crop variation due to any disease or deficiency of nutrition can be made.
- **Automatic Weeding**
  An automatic weeding machine is an intelligent project enabled with AI and computer vision that removes unwanted plants or weeds around the crops. Traditionally weeding methods require human labour, which is costly and inefficient compared to automatic weeding systems. Computer vision enables the intelligent detection and removal of weeds using robots, which reduces costs and ensures higher yields.
- **PlantDiseaseDetection**
  Computer vision is also used in automated plant disease detection, which is important at an early stage of

plant development. Various deep-learning-based algorithms use computer vision to identify plant diseases, estimate their severity and predict their impact on yields.

5. Computer Vision in Retail

In the retail sector, computer vision system enables retailers to collect a huge volume of visual data and hence design better customer experiences with the help of cameras installed in stores. Some popular applications of computer vision in the retail industry are given below:

- **Self-checkout**
  Self-checkout enables the customers to complete their transactions from a retailer without the need for human staff, and this becomes possible with computer vision. Self-checkouts are now helping retailers in avoiding long queues and manage customers.
- **Automaticreplenishment**
  Automated stock replenishment is a leading technology innovation in retail sectors. Traditionally, stock replenishment is performed by store staff, who check selves to track the items for inventory management. But now, automatic replenishment with computer vision systems captures the image data and performs a complete inventory scan to track the shelves item at regular intervals.
- **PeopleCounting**
  Nowadays, various situations occur where we may need the count of people or customers entering and leaving the stores. This foot count or people counting can be done by computer vision systems that analyze the image or video data captured by the in-store cameras. People counting is helpful in managing the people and allowing the limited people for cases such as Covid social distancing.
- Computer Vision Techniques
- As human beings, we can see, process, understand, and act on anything that we can see or any visual input; in other words, we have the ability to see and understand any visual data. But how we can implement the same thing in machines? So, here Computer Vision comes into the picture. Although there are still various limitations in machines to visualise similar to humans, they are very close to analysing, understanding, and extracting meaningful information from any visual input. Nowadays, Computer vision is one of the trending research areas with deep learning.
- In this topic, we will have a deep understanding of different computer vision techniques that are currently being used in several applications. However, before starting, let's first understand the basic introduction of computer vision.
- What is Computer Vision?
- Computer vision is a sub-field of AI and machine learning that enables the machine to see, understand, and interpret the visuals such as images, video, etc., and extract useful information from them that can be helpful in the decision-making of AI applications. It can be considered as an eye for an AI application. With the help of computer vision technology, such tasks can be done that would be impossible without this technology, such as Self Driving Cars.
- Computer Vision Process



A typic process of Computer vision is illustrated in the above image. It mainly performs three steps, which are:

Capturing an Image

A computer vision software or application always includes a digital camera or CCTV to capture the image. So, firstly it captures the image and puts it as a digital file that consists of Zero and one's.

2. Processing the image

In the next step, different CV algorithms are used to process the digital data stored in a file. These algorithms determine the basic geometric elements and generate the image using the stored digital data.

3. Analyzing and taking required action

Finally, the CV analyses the data, and according to this analysis, the system takes the required action for which it is designed.

Top Computer Vision Techniques

1. Image Classification

Image classification is the simplest technique of Computer Vision. The main aim of image classification is to classify the image into one or more different categories. Image classifier basically takes an image as input and

tells about different objects present in that image, such as a person, dog, tree, etc. However, it would not give you other more information about the image data, such as how many persons are there, tree colour, item positions, etc., and for this, we need to go for any other CV technique.

mage classification is basically of two types, Binary classification and multi-class classification. As the name suggests, binary image classification looks for a single class in the given image and provides results based on if the image has that object or not. For example, we can achieve superhuman performance in detecting skin cancer in humans by training an AI system on both images that have skin cancer and images that do not have skin cancer.

2. Object Detection

Object detection is another popular technique of computer vision that can be performed after Image classification or which uses image classification to detect the objects in visual data. It is basically used to recognize the objects within the boundary boxes and find the class of the objects in the image. Object detection makes use of deep learning and machine learning technology to generate useful results.

As human beings, whenever we see a visual or look at an image or video, we can immediately recognize and even locate the objects within a moment. So, the aim of object detection is to replicate the same human intelligence into machines to identify and locate the objects.

Object detection has several applications, including *object tracking, retrieval, video surveillance, image captioning, etc.*

A variety of techniques can be used to perform object detection, which includes **R-CNN, YOLO v2**, etc.

3. Semantic Segmentation

Semantic Segmentation is not only about detecting the classes in an image as image classification. Instead, it classifies each pixel of an image to specify what objects it has. It tries to determine the role of each pixel in the image. It basically classifies pixelS in a particular category without differentiating the object instances. Or we can say it classifies similar objects as a single class from the pixel levels. For example, if an image contains two dogs, then semantic segmentation will put both the dogs under the same label. It tries to understand the role of each pixel in an image.

4. Instance Segmentation

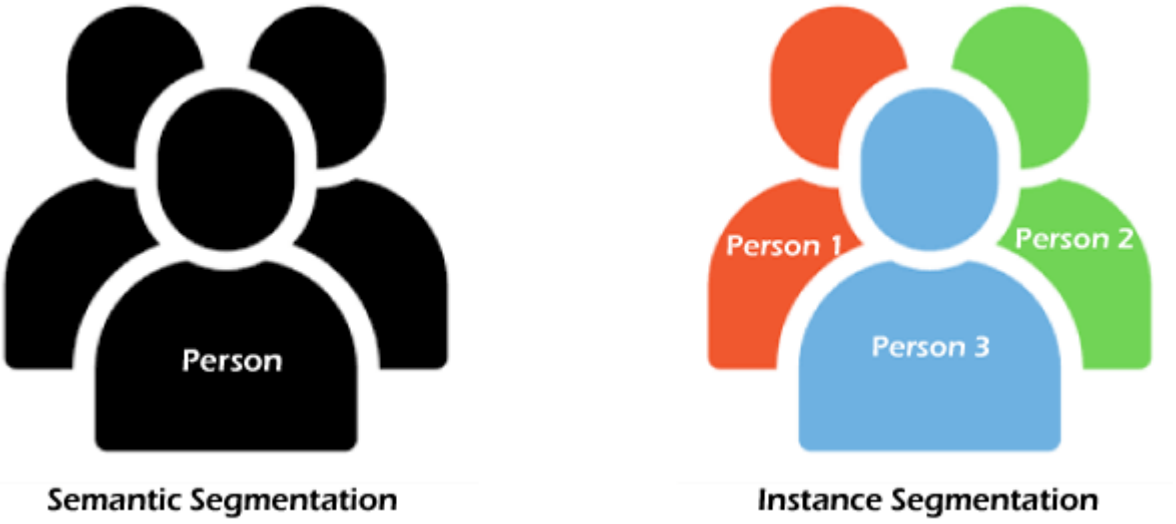
Instance segmentation can classify the objects in an image at pixel level as similar to semantic segmentation but with a more advanced level. It means Instance Segmentation can classify similar types of objects into different categories. For example, if visual consists of various cars, then with semantic segmentation, we can tell that there are multiple cars, but with instance segmentation, we can label them according to their colour, shape, etc.

Instance segmentation is a typical computer vision task compared to other techniques as it needs to analyse the difference within visual data with different overlapping objects and different backgrounds.

In Instance segmentation, CNN or Convolutional Neural Networks can be effectively used, where they can locate the objects at pixels level instead of just bounding the boxes. A well-known example of CNN and instance segmentation is **Facebook AI**. This application can detect or differentiate two colours of the same object, and the

architecture of CNN used in this is known as **Mask R-CNN** or **Mask Region-Based Convolutional Neural Network.**

Using the below image, we can analyse the difference between semantic segmentation and instance segmentation, where semantic segmentation classified all the persons as singly entities, whereas instance segmentation classified all the persons as different by considering colours also.



5. Panoptic Segmentation

Panoptic Segmentation is one of the most powerful computer vision techniques as it combines the Instance and Semantic Segmentation techniques. It means with Panoptic Segmentation, you can classify image objects at pixel levels and can also identify separate instances of that class.

6. Keypoint Detection

Keypoint detection tries to detect some key points in an image to give more details about a class of objects. It basically detects people and localizes their key points. There are mainly two keypoint detection areas, which are **Body Keypoint Detection** and **Facial Keypoint Detection.**

For example, Facial keypoint detection includes detecting key parts of the human face such as the nose, eyes, corners, eyebrows, etc. Keypoint detection mainly has applications, including face detection, pose detection, etc.

With Pose estimation, we can detect what pose people have in a given image, which usually includes where the head, eyes, nose, arms, shoulders, hands, and legs are in an image. This can be done for a single person or multiple people as per the need.

7. Person Segmentation

Person segmentation is a type of image segmentation technique which is used to separate the person from the background within an image. It can be used after the pose estimation, as with this, we can closely identify the exact location of the person in the image as well as the pose of that person.

8. Depth Perception

Depth perception is a computer vision technique that provides the visual ability to machines to estimate the 3D depth/distance of an object from the source. Depth Perception has wide applications, including the Reconstruction of objects in Augmented Reality, Robotics, self-driving cars, etc. **LiDAR**(Lights Detection and Ranging) is one of the popular techniques that is used for in-depth perception. With the help of laser beams, it measures the relative distance of an object by illuminating it with laser light and then measuring the reflections using sensors.

9. Image Captioning

Image captioning, as the name suggests, is about giving a suitable caption to the image that can describe the image. It makes use of neural networks, where when we input an image, then it generates a caption for that image that can easily describe the image. It is not only the task of Computer vision but also an NLP task.

10. 3D Object Reconstruction

As the name suggests, 3D object reconstruction is a technique that can extract 3D objects from a 2D image. Currently, it is a much-developing field of computer vision, and it can be done in different ways for different objects. On this technique, one of the most successful papers is **PiFuHD**, which tells about 3D human digitization.

Introduction to Image Pre-processing | What is Image Pre-processing?

By Great Learning Team 38706
Contributed to: Sreekanth
LinkedIn profile: https://www.linkedin.com/in/sreekanth-tadakaluru-3301649b/

Introduction to Image Pre-Processing

As a Machine Learning Engineer, data pre-processing or data cleansing is a crucial step and most of the ML engineers spend a good amount of time in data pre-processing before building the model. Some examples for data pre-processing includes outlier detection, missing value treatments and remove the unwanted or noisy data.

Similarly, Image pre-processing is the term for operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task.

There are 4 different types of Image Pre-Processing techniques and they are listed below.

1. Pixel brightness transformations/ Brightness corrections

2. Geometric Transformations
3. Image Filtering and Segmentation
4. Fourier transform and Image restauration

Let's discuss each type in detail.

Pixel brightness transformations(PBT)

Brightness transformations modify pixel brightness and the transformation depends on the properties of a pixel itself. In PBT, output pixel's value depends only on the corresponding input pixel value. Examples of such operators include brightness and contrast adjustments as well as colour correction and transformations.

Contrast enhancement is an important area in image processing for both human and computer vision. It is widely used for medical image processing and as a pre-processing step in speech recognition, texture synthesis, and many other image/video processing applications

There are two types of Brightness transformations and they are below.

1. Brightness corrections
2. Gray scale transformation

The most common Pixel brightness transforms operations are

1. Gamma correction or Power Law Transform
2. Sigmoid stretching
3. Histogram equalization

Two commonly used point processes are multiplication and addition with a constant.

$g(x)=\alpha f(x)+\beta$

The parameters $\alpha>0$ and β are called the gain and bias parameters and sometimes these parameters are said to control contrast and brightness respectively.

cv.convertScaleAbs(image, alpha=alpha, beta=beta)

for different values of alpha and beta, the image brightness and contrast varies.

Gamma Correction

Gamma correction is a non-linear adjustment to individual pixel values. While in image normalization we carried out linear operations on individual pixels, such as scalar multiplication and addition/subtraction, gamma correction carries out a non-linear operation on the source image pixels, and can cause saturation of the image being altered.

How To Apply Machine Learning to Recognise Handwriting

Histogram equalization

Histogram equalization is a well-known contrast enhancement technique due to its performance on almost all types of image. Histogram equalization provides a sophisticated method for modifying the dynamic range and contrast of an image by altering that image such that its intensity histogram has the desired shape. Unlike contrast stretching, histogram modelling operators may employ non-linear and non-monotonic transfer functions to map between pixel intensity values in the input and output images.

The normalized histogram.

P(n) = number of pixels with intensity n/ total number of pixels

Sigmoid stretching

Sigmoid function is a continuous nonlinear activation function. The name, sigmoid, is obtained from the fact that the function is "S" shaped. Statisticians call this function the logistic function.

g (x,y) is Enhanced pixel value

c is Contrast factor

th is Threshold value

fs(x,y) is original image

By adjusting the contrast factor 'c' and threshold value it is possible to tailor the amount of lightening and darkening to control the overall contrast enhancement

Geometric Transformations

The earlier methods in this article deal with the colour and brightness/contrast. With geometric transformation, positions of pixels in an image are modified but the colours are unchanged.

Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured. The normal Geometric transformation operations are rotation, scaling and distortion (or undistortion!) of images.

There are two basic steps in geometric transformations:

1. Spatial transformation of the physical rearrangement of pixels in the image

2. Grey level interpolation, which assigns grey levels to the transformed image

change the perspective of a given image or video for getting better insights about the required information. Here the points needs to be provided on the image from which want to gather information by changing the perspective.

Interpolation Methods :

After the transformation methods, the new point co-ordinates (x',y') were obtained. Lets suppose these new points do not in general fit the discrete raster of the output image. So Each pixel value in the output image raster can be obtained by interpolation methods.

The brightness interpolation problem is usually expressed in a dual way. The brightness value of the pixel (x',y') in the output image where x' and y' lie on the discrete raster and it is

Different types of Interpolation methods are

1. Nearest neighbor interpolation is the simplest technique that re samples the pixel values present in the input vector or a matrix

2. Linear interpolation explores four points neighboring the point (x,y), and assumes that the brightness function is linear in this neighborhood.

3. Bicubic interpolation improves the model of the brightness function by approximating it locally by a bicubic polynomial surface.sixteen neighboring points are used for interpolation.

Image Filtering and Segmentation

The goal of using filters is to modify or enhance image properties and/or to extract valuable information from the pictures such as edges, corners, and blobs. A filter is defined by a kernel, which is a small array applied to each pixel and its neighbors within an image

Some of the basic filtering techniques are

1. Low Pass Filtering (Smoothing) : A low pass filter is the basis for most smoothing methods. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels
2. High pass filters (Edge Detection, Sharpening) : High-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image – the opposite of the low-pass filter. High-pass filtering works in the same way as low-pass filtering; it just uses a different convolution kernel.
3. Directional Filtering : Directional filter is an edge detector that can be used to compute the first derivatives of an image. The first derivatives (or slopes) are most evident when a large change occurs between adjacent pixel values.Directional filters can be designed for any direction within a given space
4. Laplacian Filtering : Laplacian filter is an edge detector used to compute the second derivatives of an image, measuring the rate at which the first derivatives change. This determines if a change in adjacent pixel values is from an edge or continuous progression. Laplacian filter kernels usually contain negative values in a cross pattern, centered within the array. The corners are either zero or positive values. The center value can be either negative or positive.

Computer Vision: Low-level Vision

Image Segmentation

Image segmentation is a commonly used technique in digital image processing and analysis to partition an image into multiple parts or regions, often based on the characteristics of the pixels in the image. Image segmentation could involve separating foreground from background, or clustering regions of pixels based on similarities in colour or shape.

Image Segmentation mainly used in

- Face detection
- Medical imaging
- Machine vision
- Autonomous Driving

There are two types of image segmentation techniques.

1. Non-contextual thresholding : Thresholding is the simplest non-contextual segmentation technique. With a single threshold, it transforms a greyscale or colour image into a binary image considered as a binary region map. The binary map contains two possibly disjoint regions, one of them containing pixels with input data values smaller than a threshold and another relating to the input values that are at or above the threshold. The below are the types of thresholding techniques

1. Simple thresholding
2. Adaptive thresholding
3. Colour thresholding

1. Contextual segmentation : Non-contextual thresholding groups pixels with no account of their relative locations in the image plane. Contextual segmentation can be more successful in separating individual objects because it accounts for closeness of pixels that belong to an individual object. Two basic approaches to contextual segmentation are based on signal discontinuity or similarity. Discontinuity-based techniques attempt to find complete boundaries enclosing relatively uniform regions assuming abrupt signal changes across each boundary. Similarity-based techniques attempt to directly create these uniform regions by grouping together connected pixels that satisfy certain similarity criteria. Both the approaches mirror each other, in the sense that a complete boundary splits one region into two. The below ate the types of Contextual segmentation.

1. Pixel connectivity
2. Region similarity
3. Region growing
4. Split-and-merge segmentation

1. Texture Segmentation : Texture is most important attribute in many image analysis or computer vision applications. The procedures developed for texture problem can be subdivided into four categories.
    1. structural approach
    2. statistical approach
    3. model based approach
    4. filter based approach

Fourier transform

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

The DFT(Discrete Fourier Transform) is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size.

For a square image of size N×N, the two-dimensional DFT is given by:

In this article, the attempt is made to list down the different image pre-processing techniques. In the next articles, I will explain each and every technique with the Math and Python codes by using OpenCV and Neural Networks.

If you want to learn more about other artificial intelligence and machine learning techniques, study artificial intelligence today. Upskilling in this domain can help you advance your career. Join now! Feel free to leave your queries in the comments below.

Types of Machine Vision Lenses

This is **Section 6.1** of the Imaging Resource Guide.

‹ Previous SectionNext Section ›

Throughout the prior six sections of the resource guide, an understanding of imaging lenses has been slowly built up, piece by piece. From understanding the basics of angular field of view (AFOV) and resolution to learning about modulation transfer function (MTF) and what variables impact lens performance, the story of how a lens fundamentally behaves has been laid out. This section aims to bring everything from the prior sections together for the confident selection of an imaging lens for any given machine vision problem.

Imaging lenses are a complicated and nuanced component in imaging systems, and it is not always straightforward which decisions to make when it comes time to choose a lens and what tradeoffs are made as a direct result of those decisions. Lens specification sheets (or datasheets) vary between manufacturers, which can make comparisons a daunting task. Oftentimes, however, the problem is not as complicated as that, as it can be challenging enough to determine even the type of lens that is required for a particular application. Is a fixed focal length lens the best choice? What about a zoom lens? Or a telecentric?

This lens selection guide is broken into three distinct parts. This app note, which is the first section, Types of Machine Vision Lenses, answers the question: what kind of lens do I need for my application? For the purposes of this text, lenses will be divided into one of two buckets: a variable magnification lens or a fixed magnification lens. It describes these different types of lenses and what they are useful for. The second section, Basic Lens Selection explains how to make an informed lens selection when a camera has already been chosen, which is often the case. This section focuses on lens selection by emphasizing the importance of comparing the needed field of view (FOV), set by application requirements, to the FOV specification on the lens data sheet, given the working distance (WD) or other constrained parameters. The final piece, Advanced Lens Selection, explains how to choose a lens alongside the camera, which is important if the application is to be properly optimized in terms

of cost and performance. It discusses pixel mapping and contrast reproduction of features on the pixel level of a camera.

It is important to read (or have prior understanding of) the previous six sections of this text before jumping into this section, as it will discuss previously learned concepts without going into any background detail. An understanding of the prior sections is particularly important for Advanced Lens Selection.

Types of Variable Magnification Lenses

Fixed Focal Length Lenses

Fixed focal length lenses known by many different names: prime lenses (common in photography or cinematography), FA lenses (where FA typically stands for factory automation), or simply as machine vision lenses. They are the most common types of lenses that exist in machine vision; as a good rule of thumb, if a lens is referred to as a single focal length (e.g., a 25mm lens), it is typically a fixed focal length lens. As explained in Understanding Focal Length and Field of View, fixed focal length lenses have a fixed AFOV. These lenses can still focus at different WDs, which is most often achieved by moving all of the individual lens elements together such that the relative spacing between them does not change.

Figure 1 shows a 75mm focal length fixed focal length lens focused at two different distances. While the spacing between each element does not change as it focuses, the distance between the image plane and the last lens element varies a great deal. The top lens is focused at optical infinity, and the bottom lens is focused at a 200mm WD.

It is important to remember that true fixed focal length lenses will always behave as in *Figure 1*, though some lenses exist that have a "floating element focus," where the relative element spacing does change through focus. This spacing change does impart a change in the focal length of the system, though it is usually not enough to classify them differently.

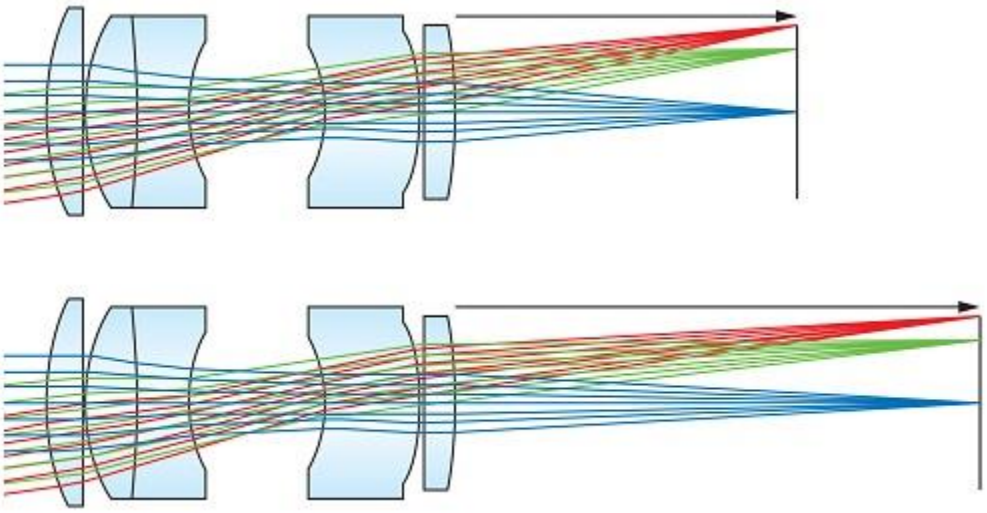


Figure 1: A 75mm double gauss-type fixed focal length lens focused at two different WDs. Note that the spacing between each element did not change as WD shifts.

Fixed focal length lenses should be used for the vast majority of machine vision applications, as they are flexible and have great performance. General parts inspection, barcode reading, biometric and document scanning, license plate reading, and other types of optical character recognition (OCR) or optical character verification (OCV) are all best suited for fixed focal length lenses most of the time. The AFOV that a fixed focal length lens has is a product of its focal length matched with its sensor size. As the focal length of a lens goes down, its AFOV increases. This occurs in a linear fashion; therefore, a 25mm focal length lens will have an AFOV that is twice as large as lens with a 50mm field of view FOV. Because these lenses can be focused at different distances and have different magnifications, they are classified as a variable magnification lens for the purpose of this document.

Zoom Lenses

Where fixed focal length lenses are designed to have a fixed AFOV, zoom lenses are designed to change their focal length, and hence their AFOV. **Zoom lenses are ideal for applications that require the ultimate amount of flexibility during use and do not require high resolution; unless a FOV actively needs to change while imaging, it is likely not the best choice.** When this is the case, stepper motors are required to change the focal length quickly and accurately.

Zoom lenses are specified as having particular zoom ratios, which can be found by dividing the longest focal length option by the smallest for any given lens. For example, if a zoom lens varies between an 8mm and a 48mm focal length, it is said to be a 6X zoom lens (48mm/8mm = 6X). This can also be expressed as a ratio: for the aforementioned lens it would be a 6:1 zoom ratio.

Figure 2 shows the same zoom lens set to two different focal lengths. Note that both relative element spacing and distance to the image plane change, despite the fact that the WD has not changed. These complicated mechanics add to the cost of the lens system, as precise movements are required to simultaneously change the lens's focal length and keep it in focus. Also, zoom lenses cannot have as high a resolution as in a comparably priced fixed focal length lens, as the complex mechanics and optical elements are multitasking.

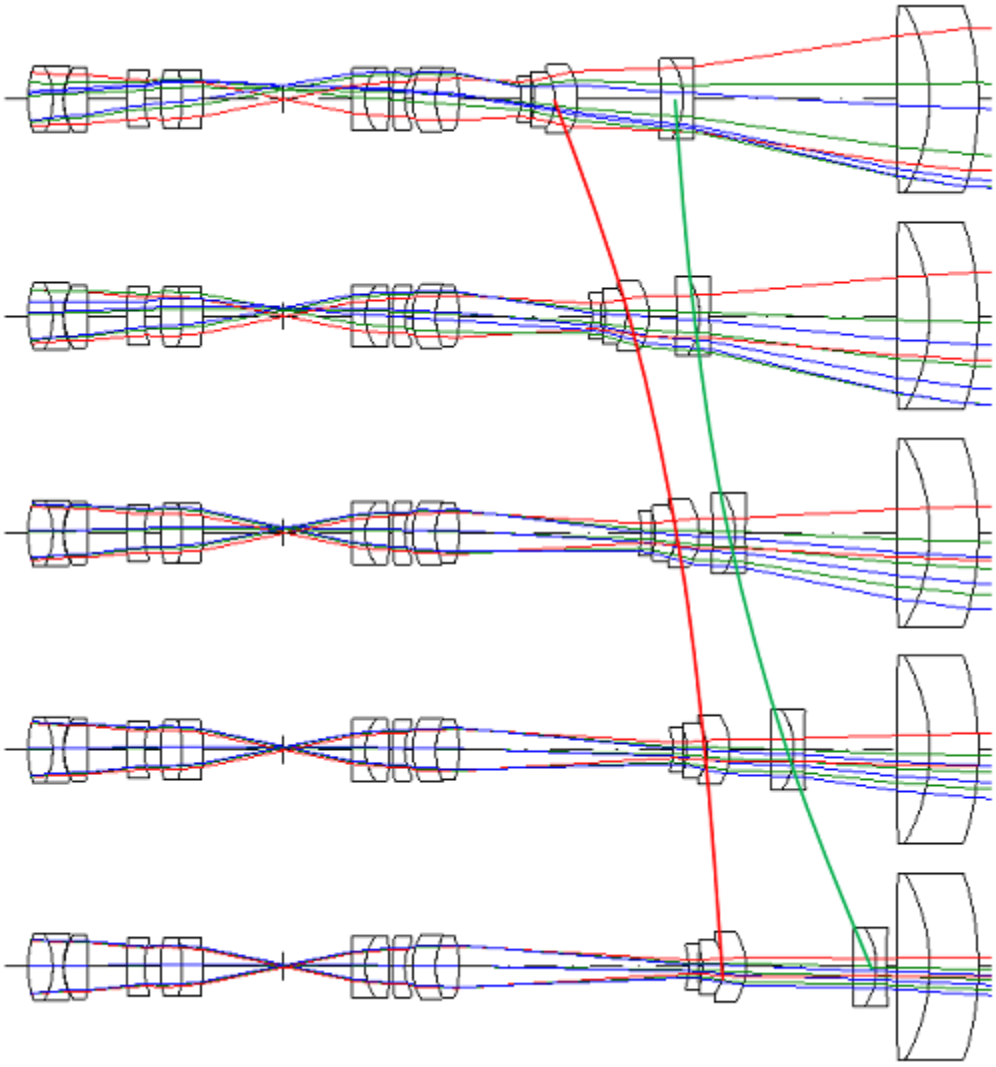


Figure 2: A zoom lens at multiple optical magnifications.

Zoom lenses not only attempt to get the best performance at a single focal length, but they are required to function over a broad range of focal lengths, which lowers their overall performance.

Other interesting optical properties may arise as a direct result of the complex movements in a zoom lens; depending on how the lens was designed, the f/# can change as the focal length changes. This type of design is typically avoided for photography or videography lenses, but for machine vision lenses this is often not the case. It is also important to recall from System Throughput, f/#, and Numerical Aperture, that the working f/# will still change as the magnification changes, resulting in different exposures.

By definition, as zoom lenses change their FOV, they remain in focus. If a lens is defocused as its focal length is changed, it is more accurately referred to as a varifocal lens, not a zoom lens. A zoom lens is chosen in much the same way as a fixed focal length lens, with the additional caveat that the focal length is a variable as opposed to a fixed parameter, which will alter the AFOV.

Types of Fixed Magnification Lenses

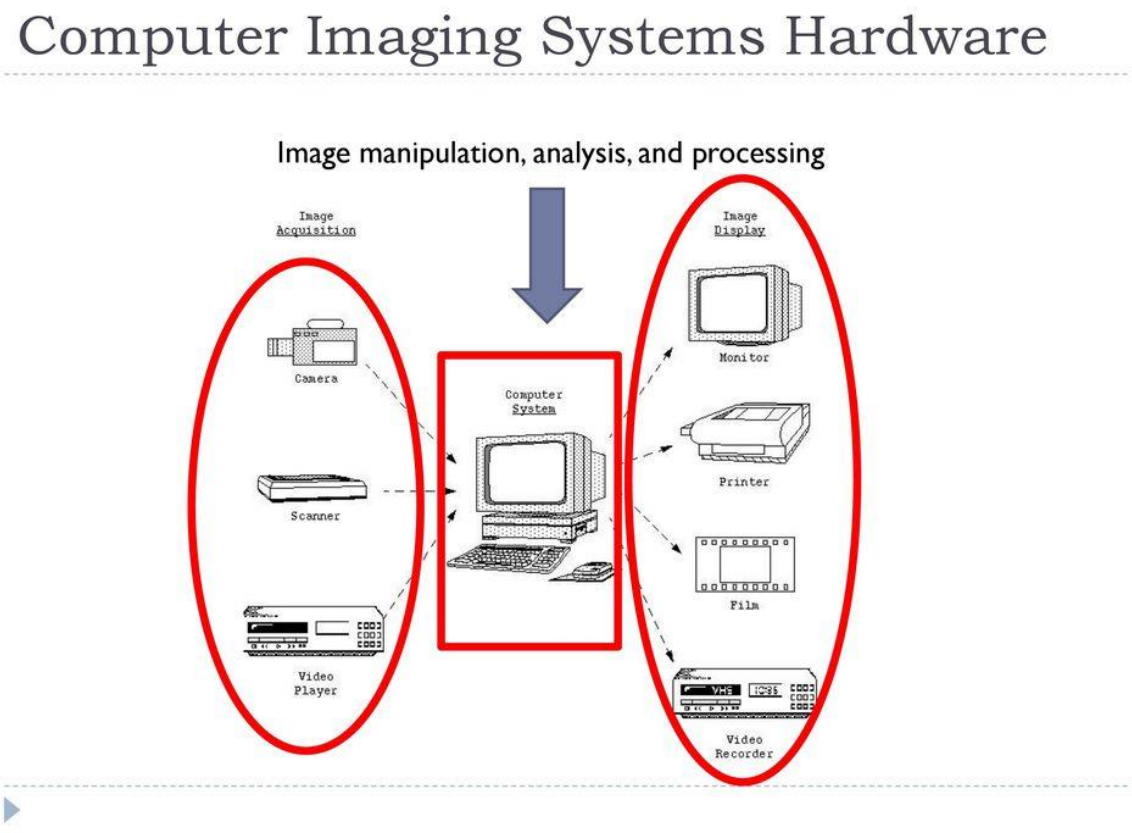
Telecentric Lenses

Telecentric lenses should be used any time a high accuracy measurement in a system needs to take place. They are highly specialized, fixed magnification lenses that come with many powerful optical capabilities. The working principles and advantages of telecentric lenses are discussed in detail in Section 4: Telecentricity and Perspective Error.

The selection of a telecentric lens is often thought of as more challenging than that of a fixed focal length lens, though this is almost always not the case, as will be seen in Basic Lens Selection.

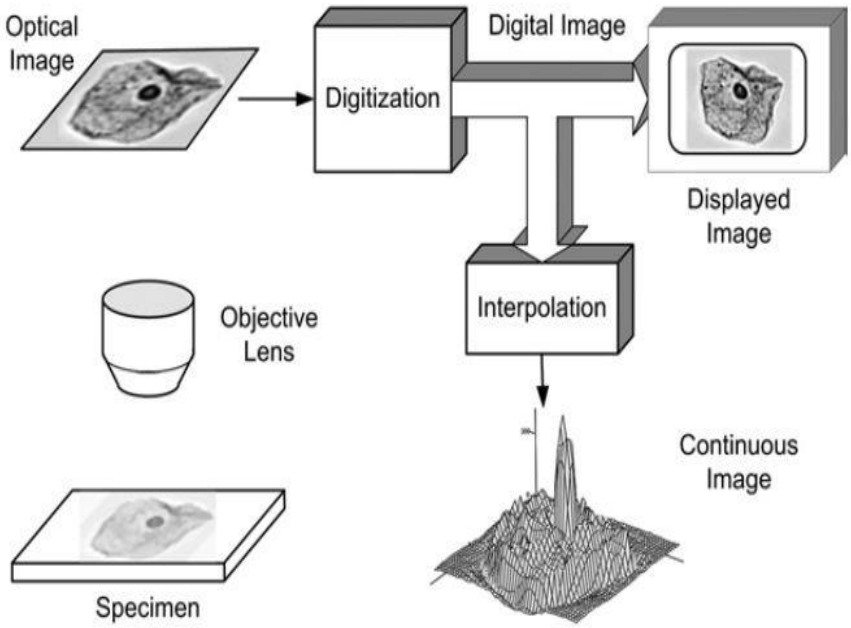
Microscope Objectives

Microscope objectives are used to image very small objects, generally with magnifications much greater than 1X. They are fixed magnification optics that only function properly at a single WD, which is generally quite small relative to other imaging lenses. **Microscope objectives should be used when a high magnification image is required and there are no strict minimum WD constraints**



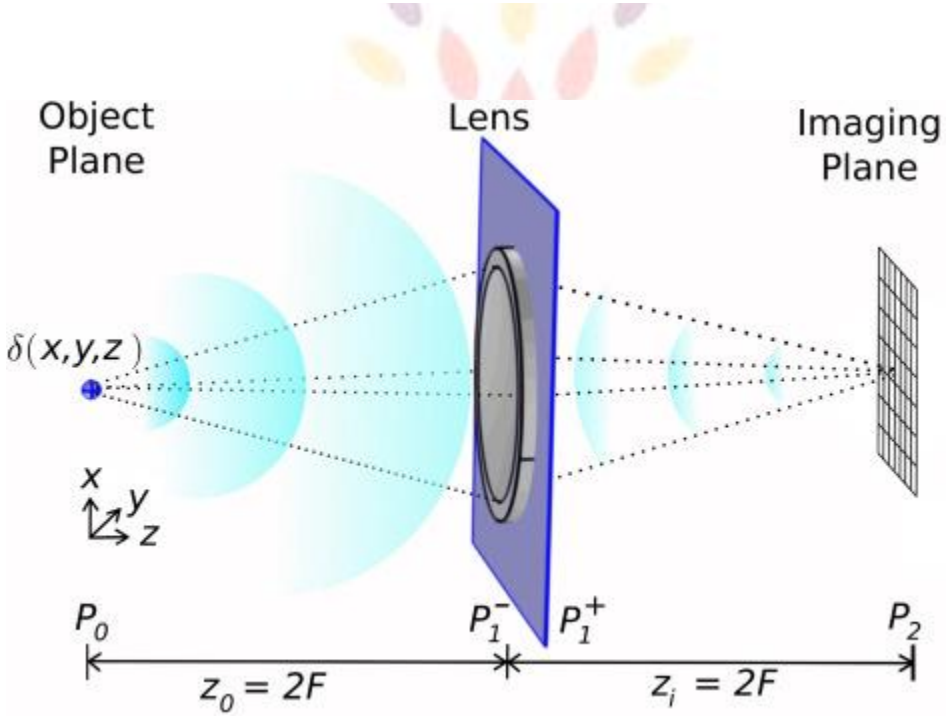
Computer Imaging Systems

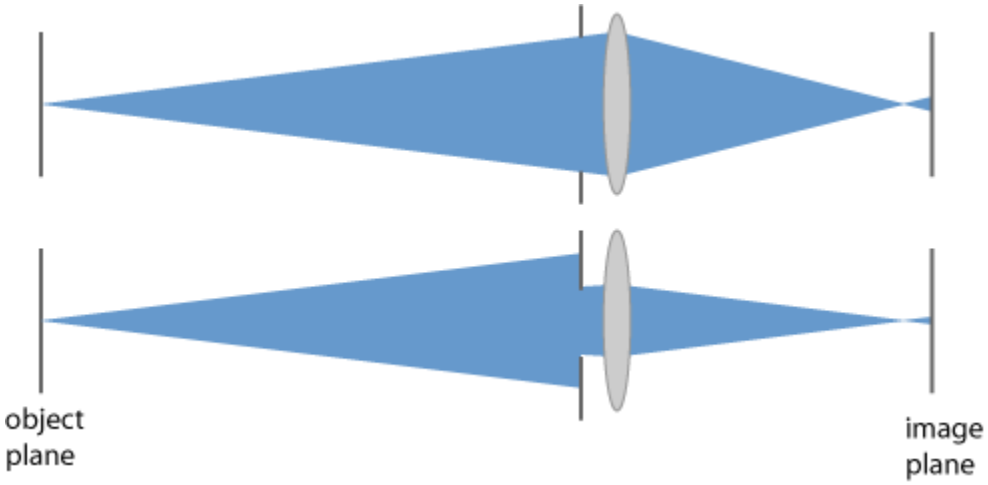
Computer imaging systems are comprised of two primary components types, hardware and software. The hardware components can be divided into image acquiring sub system (computer, scanner, and camera) and display devices (monitor, printer).The software allows us to manipulate the image and perform any desired processing on the image data.

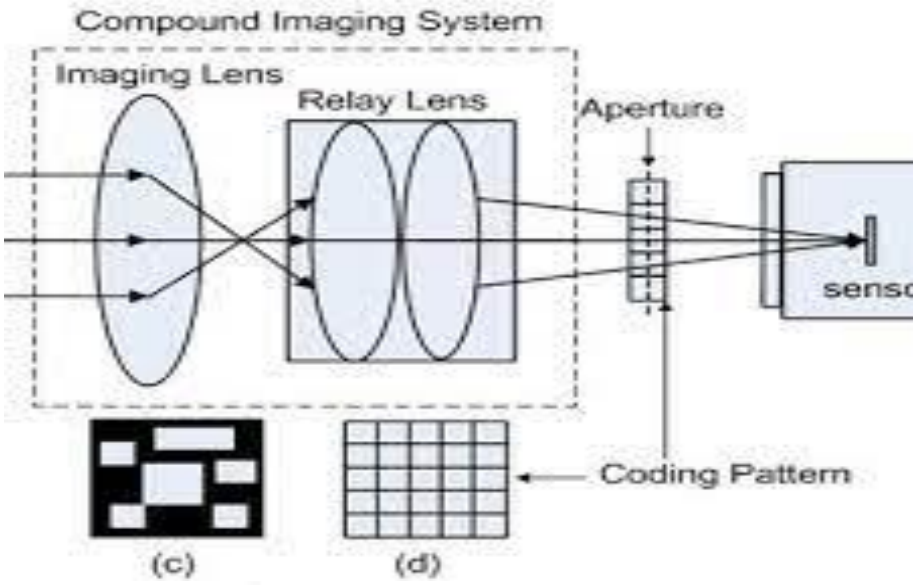


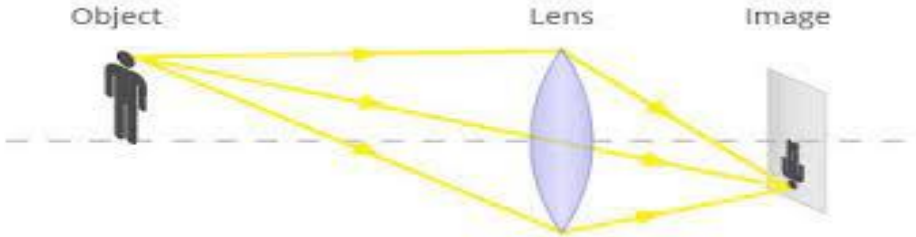
Imaging Systems Overview

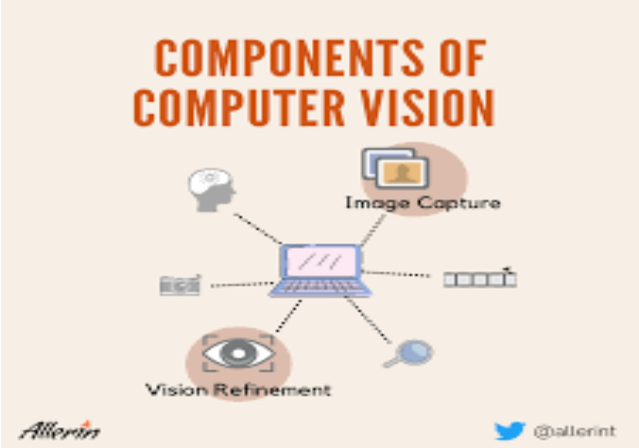
- 2 primary component – hardware & software
- Hardware – image acquisition subsystem, computer, display devices
- Software - allows us to manipulate the image and perform any desired processing on the image data and also use software to control the image acquisition and storage process.




object
plane
image
plane

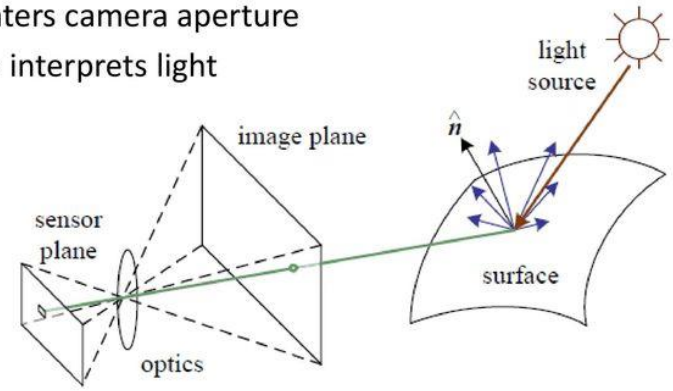

Compound Imaging System
Imaging Lens
Relay Lens
Aperture
Coding Pattern
(c)
(d)


Object
Lens
Image



How are images formed

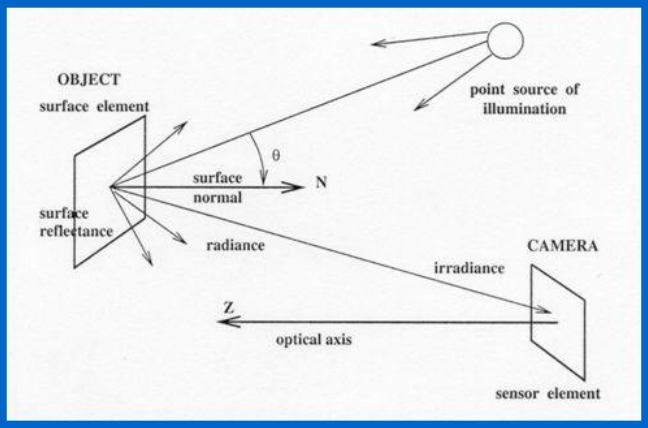
1. Light is emitted from a light source
2. Light hits a surface
3. Light interacts with the surface
4. Reflected light enters camera aperture
5. Sensor of camera interprets light



Szeliski Ch 2.2 Don't worry about all the details of the math
Shapiro & Stockman Ch. 6, Ch. 2
(https://courses.cs.washington.edu/courses/cse576/99sp/book.html

A Simple model of image formation

- The scene is illuminated by a single source.
- The scene reflects radiation towards the camera.
- The camera senses it via solid state cells (CCD cameras)



Light and Optics

- Pinhole camera model
- Perspective projection
- Thin lens model
- Fundamental equation
- Distortion: spherical & chromatic aberration, radial distortion
- Reflection and Illumination: color, Lambertian and specular surfaces, Phong, BRDF
- Sensing Light

■ Conversion to Digital Images

■ Sampling Theorem

■ Other Sensors: frequency, type

Task	Human vision	Machine vision
Visualization	Passive, mainly by reflection of light from opaque surfaces	Passive and active (controlled illumination) using electromagnetic, particulate, and acoustic radiation (Volume 1, Chapters 3 and 6)
Image formation	Refractive optical system	Various systems (see Volume 1, Chapter 4)
Control of irradiance	Muscle-controlled pupil	Motorized apertures, filter wheels, tunable filters
Focusing	Muscle-controlled change of focal length	Autofocus systems based on various principles of distance measurements
Irradiance resolution	Logarithmic sensitivity	Linear sensitivity, quantization between 8- and 16-bits; logarithmic sensitivity, for example, HDRC-sensors (Volume 1, Chapter 8)
Tracking	Highly mobile eyeball	Scanner and robot-mounted cameras (Chapter 9)
Processing and analysis	Hierarchically organized massively parallel processing	Serial processing still dominant; parallel processing not in general use

Steps of medical image analysis

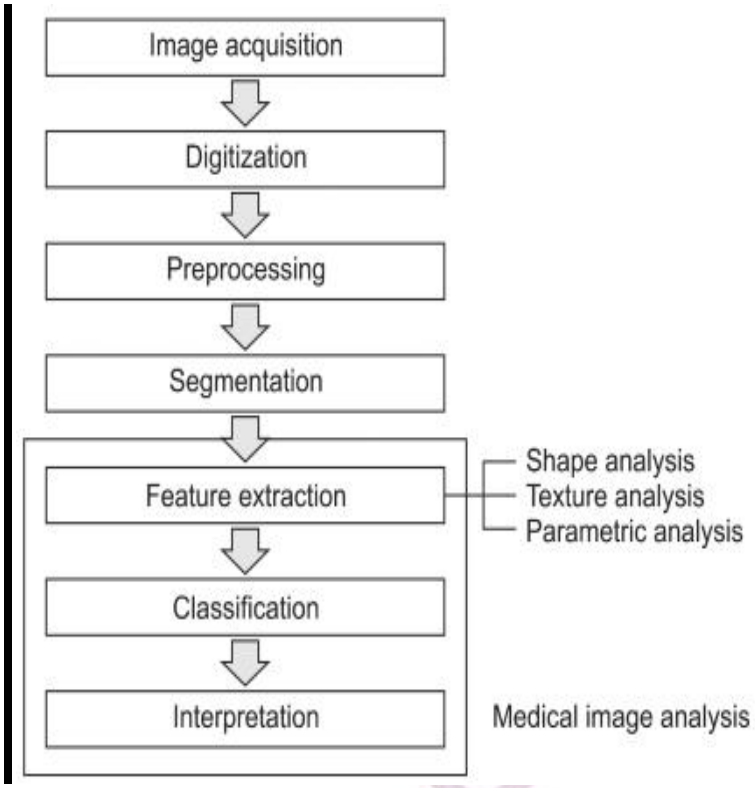


Image acquisition
Digitization
Preprocessing
Segmentation
Feature extraction
Classification
Interpretation
Shape analysis
Texture analysis
Parametric analysis
Medical image analysis

Edge detection

A useful technique in computer vision is edge detection, where the boundaries between objects are automatically identified. Having these boundaries makes it easy to segment the image (break it up into separate objects or areas), which can then be recognised separately.

For example, here's a photo where you might want to recognise individual objects:

And here's a version that has been processed by an edge detection algorithm:

Notice that the grain on the table above has affected the quality; some pre-processing to filter that would have helped!

Earlier, we looked at how we could use a *convolutional kernel* to blur an image. One of the common techniques in edge detection also requires a convolutional kernel. If we multiply the values of pixels on one side of each point on the image by a negative amount, and pixels on the other side by a positive amount, then combine the results, we'll discover a number which represents the difference between pixels on the two sides. This technique is called finding the *image gradient*. The following interactive allows you to do that, then apply a threshold to the result so that you can begin to spot likely edges in a picture.

Edge Detection

There are a few commonly used convolutional kernels that people have come up with for finding edges. After you've had a go at coming up with some of your own, have a look at the Prewitt operator, the Roberts cross and Sobel operator on wikipedia. Try these out in the interactive. What results do you get from each of these?

There are a number of good edge detections out there, but one of the more famous ones is the Canny edge detection algorithm. This is a widely used algorithm in computer vision, developed in 1986 by John F. Canny. You can read more about Canny edge detection on Wikipedia.

You could extend the techniques used in the above interactive by adding a few more processing stages. If you were to apply a gaussian filter to the image first, then do some work to favour edges that were connected to other edges, then you would be on your way to implementing the Canny edge detector.

14.6.1.+

Activity: Edge detection evaluation

With the edge detection interactive above, try uploading different images and determining how good different convolutional kernels are at detecting boundaries in the image. Capture images to put in your report as examples to illustrate your experiments with the detector.

- Can the detector find all edges in the image? If there are some missing, why might this be?
- Are there any false edge detections? Why did they system think that they were edges?
- Does the lighting on the scene affect the quality of edge detection?
- Does the system find the boundary between two colours? How similar can the colours be and still have the edge detected?
- How fast can the system process the input? Does this change with more complex kernels?
- How well does the system deal with an image with text on it?

Edges are significant local changes of intensity in a digital 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. It reduces the amount of data in an image and preserves the structural properties of an image.

What are edges

We can also say that sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

Types of edges

Generally edges are of three types:

- Horizontal edges
- Vertical Edges
- Diagonal Edges

Why detect edges

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Here are some of the masks for edge detection that we will discuss in the upcoming tutorials.

- Prewitt Operator
- Sobel Operator
- Robinson Compass Masks
- Krisch Compass Masks
- Laplacian Operator.

Above mentioned all the filters are Linear filters or smoothing filters.

Prewitt Operator

Prewitt operator is used for detecting edges horizontally and vertically.

Sobel Operator

The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. It also calculates edges in both horizontal and vertical direction.

Robinson Compass Masks

This operator is also known as direction mask. In this operator we take one mask and rotate it in all the 8 compass major directions to calculate edges of each direction.

Kirsch Compass Masks

Kirsch Compass Mask is also a derivative mask which is used for finding edges. Kirsch mask is also used for calculating edges in all the directions.

Laplacian Operator

Laplacian Operator is also a derivative operator which is used to find edges in an image. Laplacian is a second order derivative mask. It can be further divided into positive laplacian and negative laplacian.

All these masks find edges. Some find horizontally and vertically, some find in one direction only and some find in all the directions. The next concept that comes after this is sharpening which can be done once the edges are extracted from the image

Sharpening

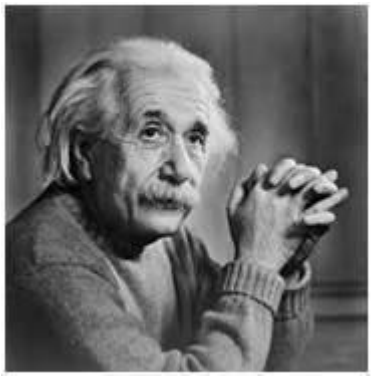
Sharpening is opposite to the blurring. In blurring, we reduce the edge content and in Sharpening, we increase the edge content. So in order to increase the edge content in an image, we have to find edges first.

Edges can be find by one of the any method described above by using any operator. After finding edges, we will add those edges on an image and thus the image would have more edges, and it would look sharpen.

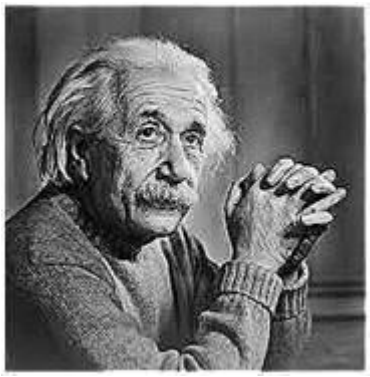
This is one way of sharpening an image.

The sharpen image is shown below.

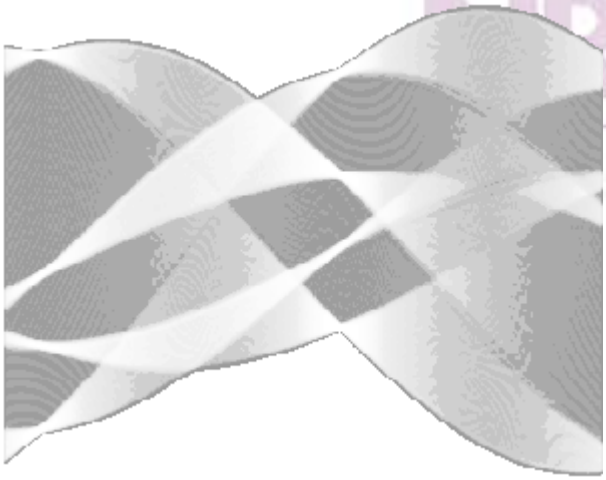
Original Image



Sharpen Image



Hough Transform



Common Names: Hough transform

Brief Description

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the *classical* Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, *etc.* A *generalized* Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the *classical* prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

How It Works

The Hough technique is particularly useful for computing a global description of a feature(s) (where the number of solution classes need not be known *a priori*), given (possibly noisy) local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (*e.g.* coordinate point) indicates its contribution to a globally consistent solution (*e.g.* the physical line which gave rise to that image point).

As a simple example, consider the common problem of fitting a set of line segments to a set of discrete image points (*e.g.* pixel locations output from an edge detector). Figure 1 shows some possible solutions to this problem. Here the lack of *a priori* knowledge about the number of desired line segments (and the ambiguity about what constitutes a line segment) render this problem under-constrained.

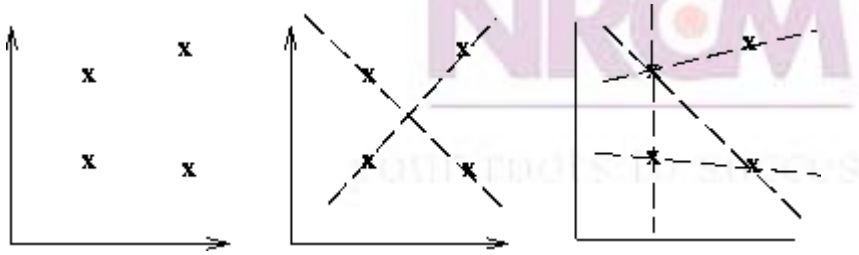


Figure 1 a) Coordinate points. **b)** and **c)** Possible straight line fittings.

We can analytically describe a line segment in a number of forms. However, a convenient equation for describing a set of lines uses *parametric* or *normal* notion:

$$x\cos\theta + y\sin\theta = r$$

where r is the length of a normal from the origin to this line and θ is the orientation of r with respect to the X-axis. (See Figure 2.) For any point (x, y) on this line, r and θ are constant.

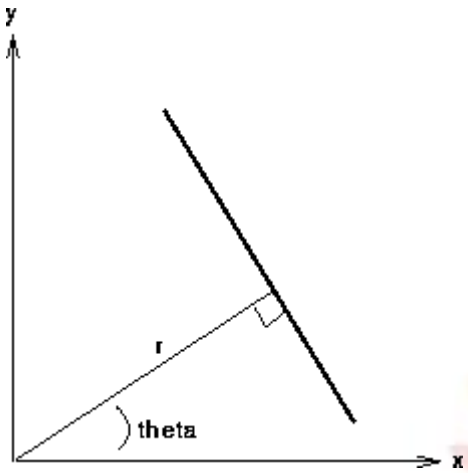


Figure 2 Parametric description of a straight line.

In an image analysis context, the coordinates of the point(s) of edge segments (*i.e.* (x_i, y_i)) in the image are known and therefore serve as constants in the parametric line equation, while r and θ are the unknown variables we seek. If we plot the possible (r, θ) values defined by each (x_i, y_i), points in cartesian image space map to curves (*i.e.* sinusoids) in the polar Hough parameter space. This *point-to-curve* transformation is the Hough transformation for straight lines. When viewed in Hough parameter space, points which are collinear in the cartesian image space become readily apparent as they yield curves which intersect at a common (r, θ) point.

The transform is implemented by quantizing the Hough parameter space into finite intervals or *accumulator cells*. As the algorithm runs, each (x_i, y_i) is transformed into a discretized (r, θ) curve and the accumulator cells which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.

We can use this same procedure to detect other features with analytical descriptions. For instance, in the case of *circles*, the parametric equation is

$$(x-a)^2+(y-b)^2=r^2$$

where a and b are the coordinates of the center of the circle and r is the radius. In this case, the computational complexity of the algorithm begins to increase as we now have three coordinates in the parameter space and a 3-D accumulator. (In general, the computation and the size of the accumulator array increase polynomially with the number of parameters. Thus, the basic Hough technique described here is only practical for simple curves.)

Guidelines for Use

The Hough transform can be used to identify the parameter(s) of a curve which best fits a set of given edge points. This edge description is commonly obtained from a feature detecting operator such as the Roberts Cross, Sobel or Canny edge detector and may be noisy, *i.e.* it may contain multiple edge fragments corresponding to a single whole feature. Furthermore, as the output of an edge detector defines only *where* features are in an image, the work of the Hough transform is to determine both *what* the features are (*i.e.* to detect the feature(s) for which it has a parametric (or other) description) and *how many* of them exist in the image.

In order to illustrate the Hough transform in detail, we begin with the simple image of two occluding rectangles,

The Canny edge detector can produce a set of boundary descriptions for this part, as shown in

Here we see the overall boundaries in the image, but this result tells us nothing about the identity (and quantity) of feature(s) within this boundary description. In this case, we can use the Hough (line detecting) transform to detect the eight separate straight lines segments of this image and thereby identify the true geometric structure of the subject.

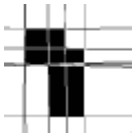
If we use these edge/boundary points as input to the Hough transform, a curve is generated in polar (r,θ) space for each edge point in cartesian space. The accumulator array, when viewed as an intensity image, looks like

Histogram equalizing the image allows us to see the patterns of information contained in the low intensity pixel values, as shown in

Note that, although r and θ are notionally polar coordinates, the accumulator space is plotted rectangularly with θ as the abscissa and r as the ordinate. Note that the accumulator space wraps around at the vertical edge of the image such that, in fact, there are only 8 real peaks.

Curves generated by collinear points in the gradient image intersect in peaks (r, θ) in the Hough transform space. These intersection points characterize the straight line segments of the original image. There are a number of methods which one might employ to extract these bright points, or *local maxima*, from the accumulator array. For example, a simple method involves thresholding and then applying some thinning to the isolated clusters of bright spots in the accumulator array image. Here we use a *relative threshold* to extract the unique (r, θ) points corresponding to each of the straight line edges in the original image. (In other words, we take only those local maxima in the accumulator array whose values are equal to or greater than some fixed percentage of the global maximum value.)

Mapping back from Hough transform space (*i.e. de-Houghing*) into cartesian space yields a set of line descriptions of the image subject. By overlaying this image on an inverted version of the original, we can confirm the result that the Hough transform found the 8 true sides of the two rectangles and thus revealed the underlying geometry of the occluded scene



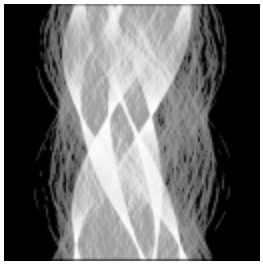
Note that the accuracy of alignment of detected and original image lines, which is obviously not perfect in this simple example, is determined by the quantization of the accumulator array. (Also note that many of the image edges have several detected lines. This arises from having several nearby Hough-space peaks

with similar line parameter values. Techniques exist for controlling this effect, but were not used here to illustrate the output of the standard Hough transform.)

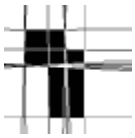
Note also that the lines generated by the Hough transform are infinite in length. If we wish to identify the actual line segments which generated the transform parameters, further image analysis is required in order to see which portions of these infinitely long lines actually have points on them.

To illustrate the Hough technique's robustness to noise, the Canny edge description has been corrupted by 1% salt and pepper noise

before Hough transforming it. The result, plotted in Hough space, is



De-Houghing this result (and overlaying it on the original) yields



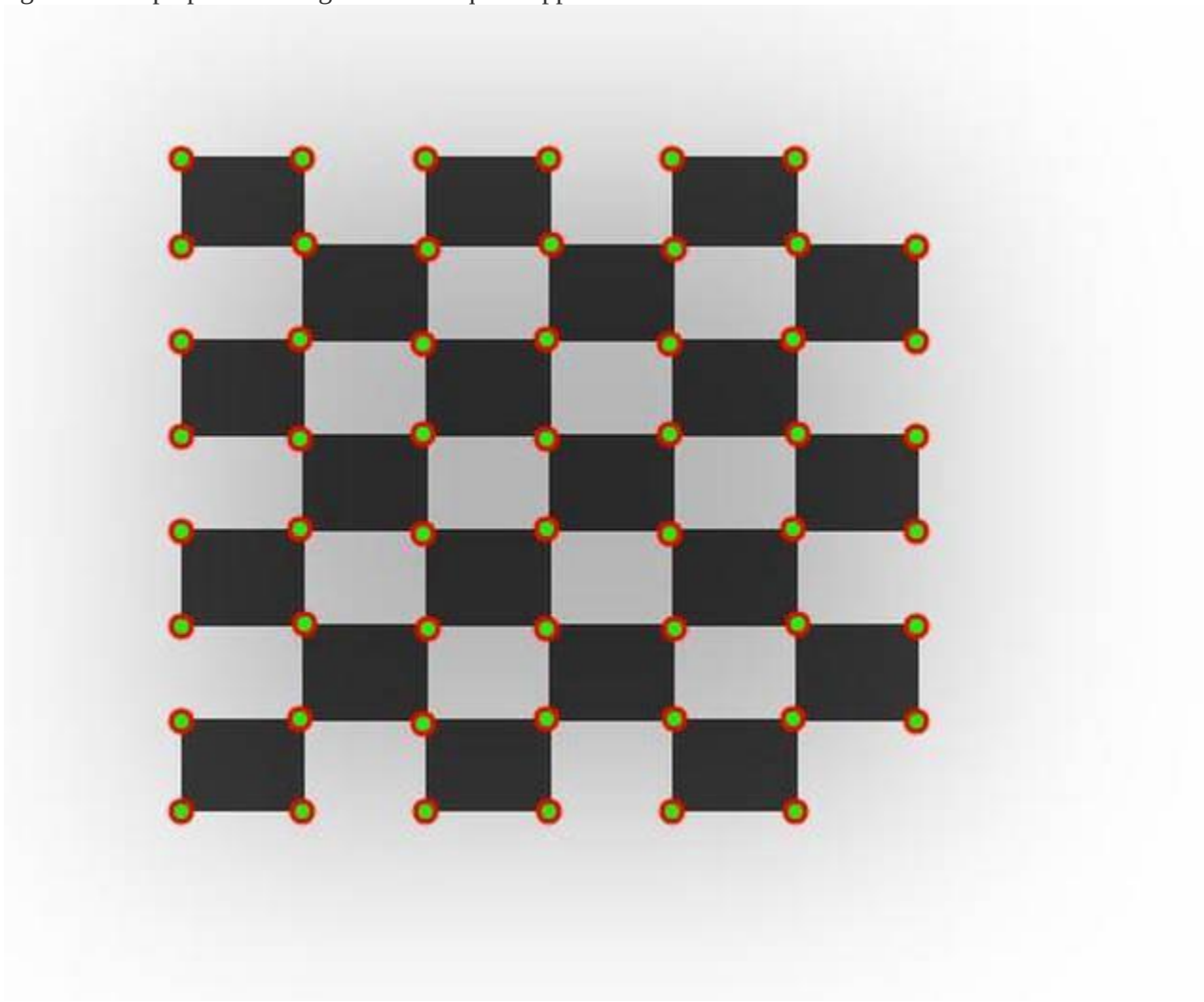
(As in the above case, the relative threshold is 40%.)

The sensitivity of the Hough transform to gaps in the feature boundary can be investigated by transforming the image

, which has been edited using a paint program. The Hough representation is

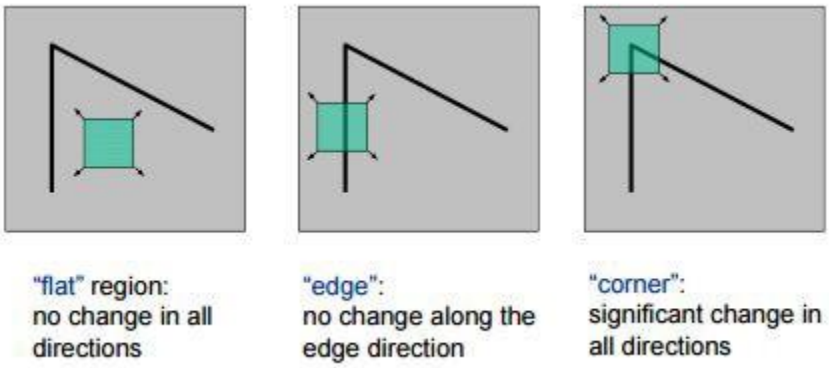
Introduction to Harris Corner Detector

Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. It was first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector. Compared to the previous one, Harris' corner detector takes the differential of the corner score into account with reference to direction directly, instead of using shifting patches for every 45-degree angles, and has been proved to be more accurate in distinguishing between edges and corners. Since then, it has been improved and adopted in many algorithms to preprocess images for subsequent applications.



Corner

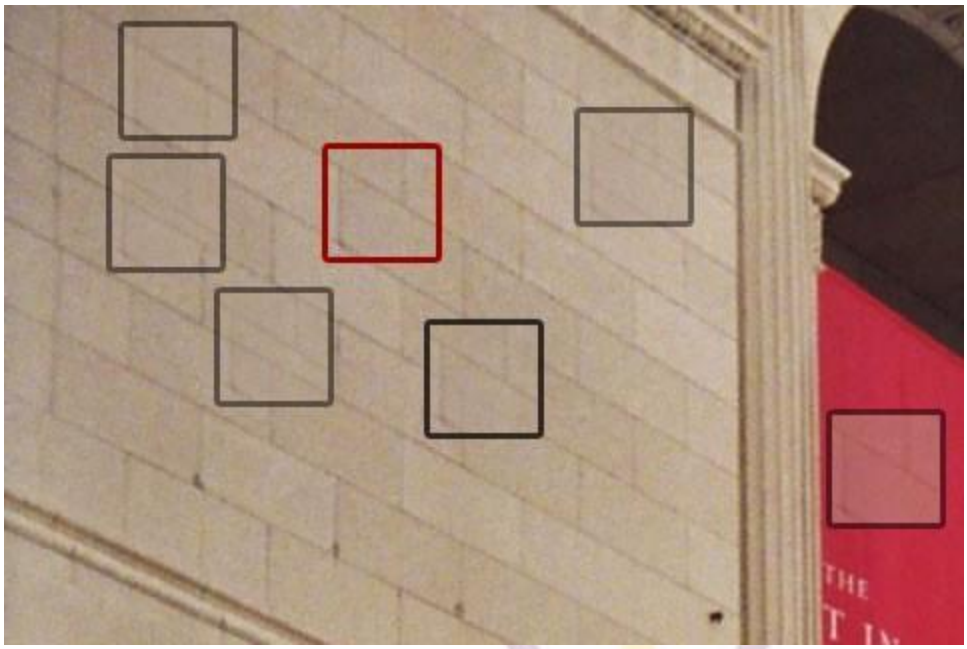
A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.



So let's understand why corners are considered better features or good for patch mapping. In the above figure, if we take the flat region then no gradient change is observed in any direction. Similarly, in the edge region, no gradient change is observed along the edge direction. So both flat region and edge region are bad for patch matching since they not very distinctive (there are many similar patches in along edge in edge region). While in corner region we observe a significant gradient change in all direction. Due this corners are considered good for patch matching(shifting the window in any direction yield a large change in appearance) and generally more stable over the change of viewpoint.

Corner Detection

The idea is to consider a small window around each pixel p in an image. We want to identify all such pixel windows that are unique. Uniqueness can be measured by shifting each window by a small amount in a given direction and measuring the amount of change that occurs in the pixel values.



More formally, we take the sum squared difference (SSD) of the pixel values before and after the shift and identifying pixel windows where the SSD is large for shifts in all 8 directions. Let us define the change function E(u,v) as the **sum** of all the sum squared differences (SSD), where u,v are the x,y coordinates of every pixel in our 3 x 3 window and I is the intensity value of the pixel. The features in the image are all pixels that have large values of E(u,v), as defined by some threshold.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

The equation

We have to maximize this function $E(u,v)$ for corner detection. That means, we have to maximize the second term. Applying Taylor Expansion to the above equation and using some mathematical steps, we get the final equation as:

$$E(u,v) \approx [u \quad v]\left(\sum \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix}$$

Now, we rename the summed-matrix, and put it to be M:

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}$$

So the equation now becomes:

$$E(u,v) \approx [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

Remember that we want the SSD to be large in shifts for all eight directions, or conversely, for the SSD to be small for none of the directions. By solving for the eigenvectors of M, we can obtain the directions for both the largest and smallest increases in SSD. The corresponding eigenvalues give us the actual value amount of these increases. A score, R, is calculated for each window:

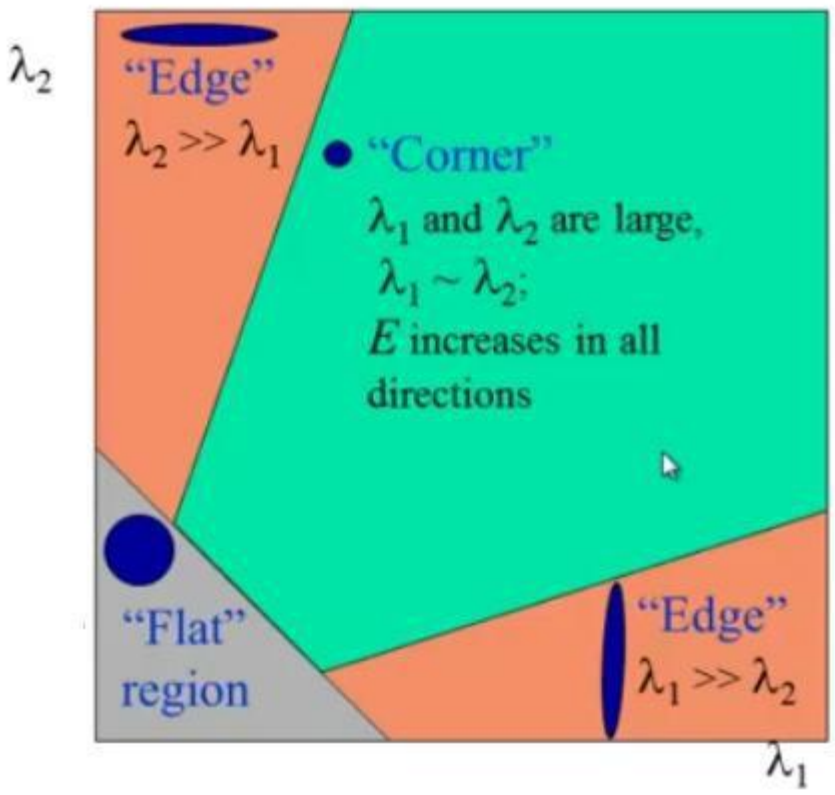
$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

$\lambda 1$ and $\lambda 2$ are the eigenvalues of M. So the values of these eigenvalues decide whether a region is a corner, edge or flat.

- When $|R|$ is small, which happens when $\lambda 1$ and $\lambda 2$ are small, the region is flat.
- When $R<0$, which happens when $\lambda 1 >> \lambda 2$ or vice versa, the region is an edge.
- When R is large, which happens when $\lambda 1$ and $\lambda 2$ are large and $\lambda 1 \sim \lambda 2$, the region is a corner.



High-level pseudocode

1. Take the grayscale of the original image

2. Apply a Gaussian filter to smooth out any noise

3. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image

4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.

5. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features)

6. For each pixel that meets the criteria in 5, compute a feature descriptor.

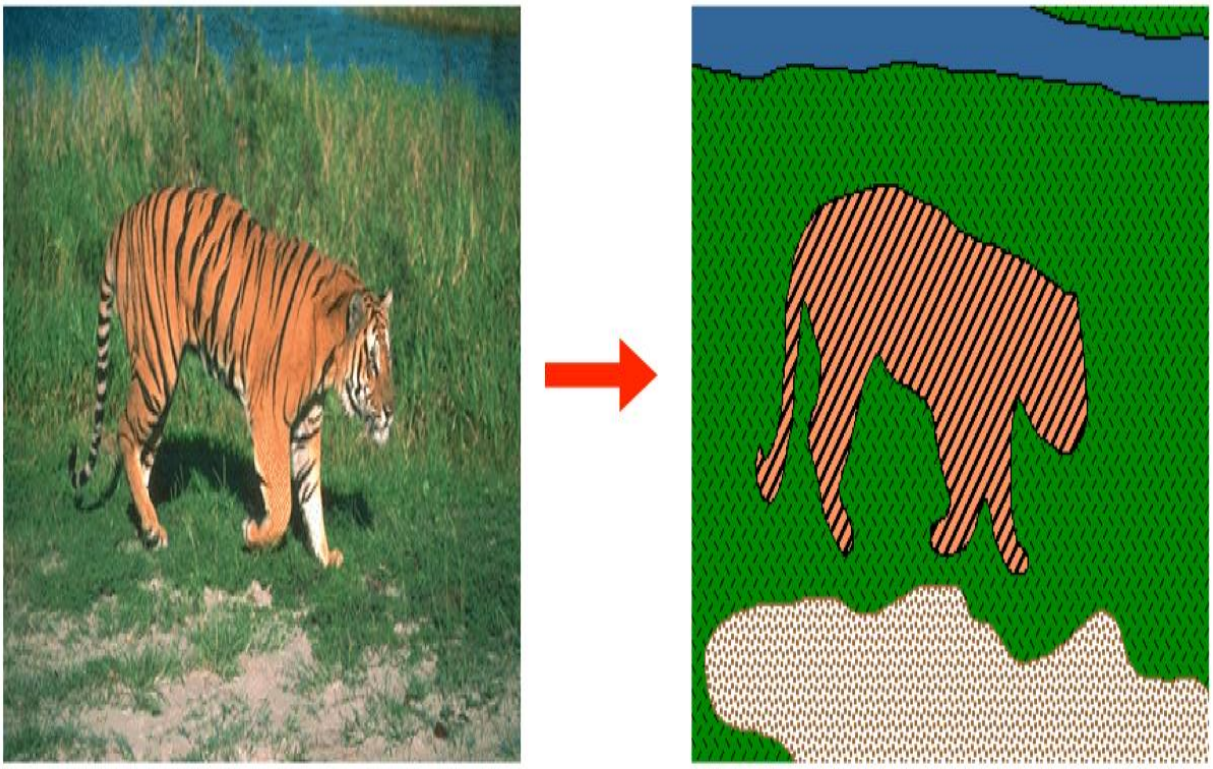
Implementation

I was able to implement the Harris Corner Detector using OpenCV. Here's how I did it:

UNIT-III

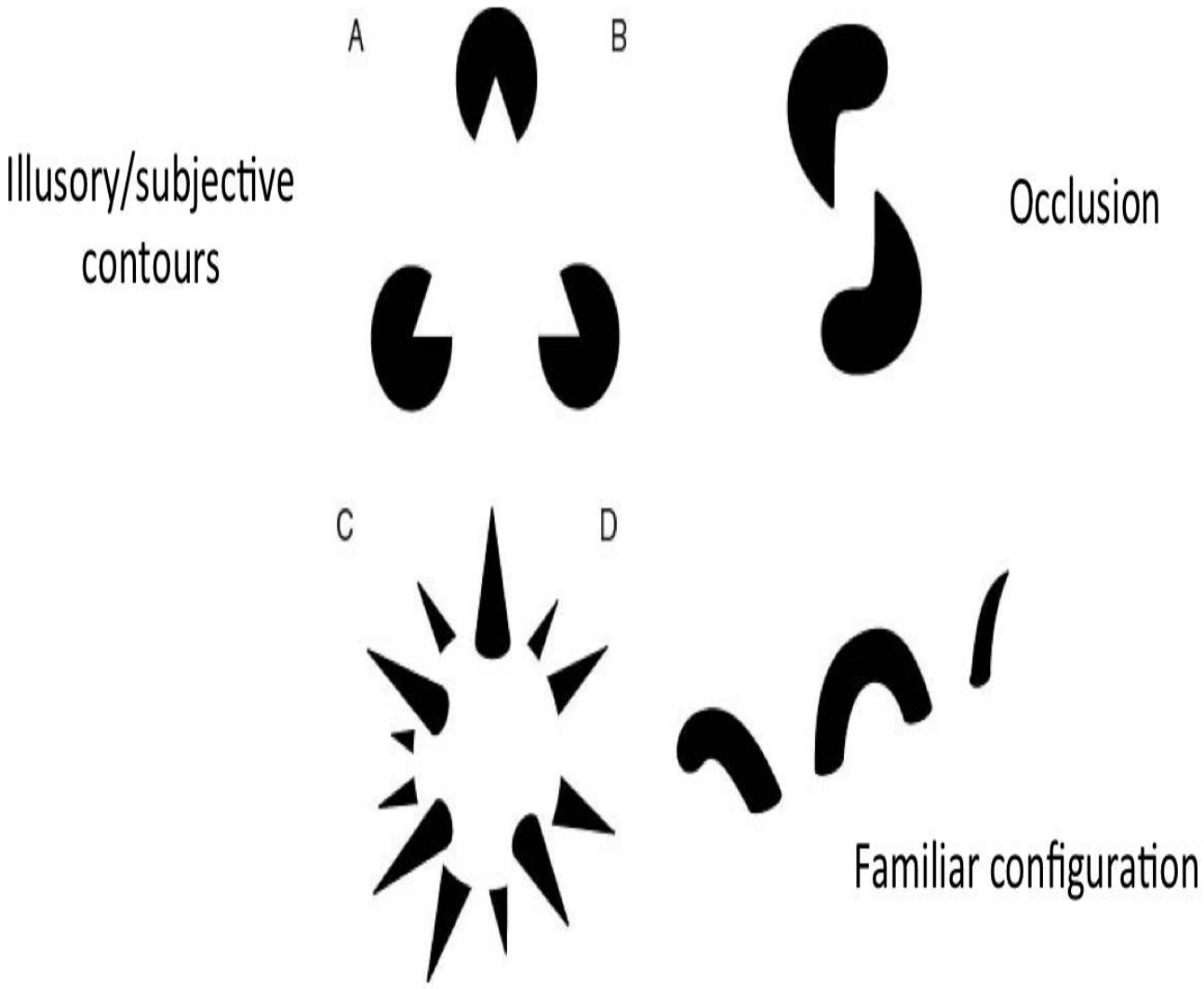
Tutorial 3: Image Segmentation

Another important subject within computer visionis image segmentation. It is the process of dividing an image into different regions based on the characteristics of pixels to identify objects or boundaries to simplify an image and more efficiently analyze it. Segmentation impacts a number of domains, from the filmmaking industry to the field of medicine. For instance, the software behind green screens implements image segmentation to crop out the foreground and place it on a background for scenes that cannot be shot or would be dangerous to shoot in real life. Image segmentation is also used to track objects in a sequence of images and to classify terrains, like petroleum reserves, in satellite images. Some medical applications of segmentation include the identification of injured muscle, the measurement of bone and tissue, and the detection of suspicious structures to aid radiologists (Computer Aided Diagnosis, or CAD).



One way to view segmentation is clustering, where pixels sharing certain features such as color, intensity, or texture are grouped together and represented as a single entity.

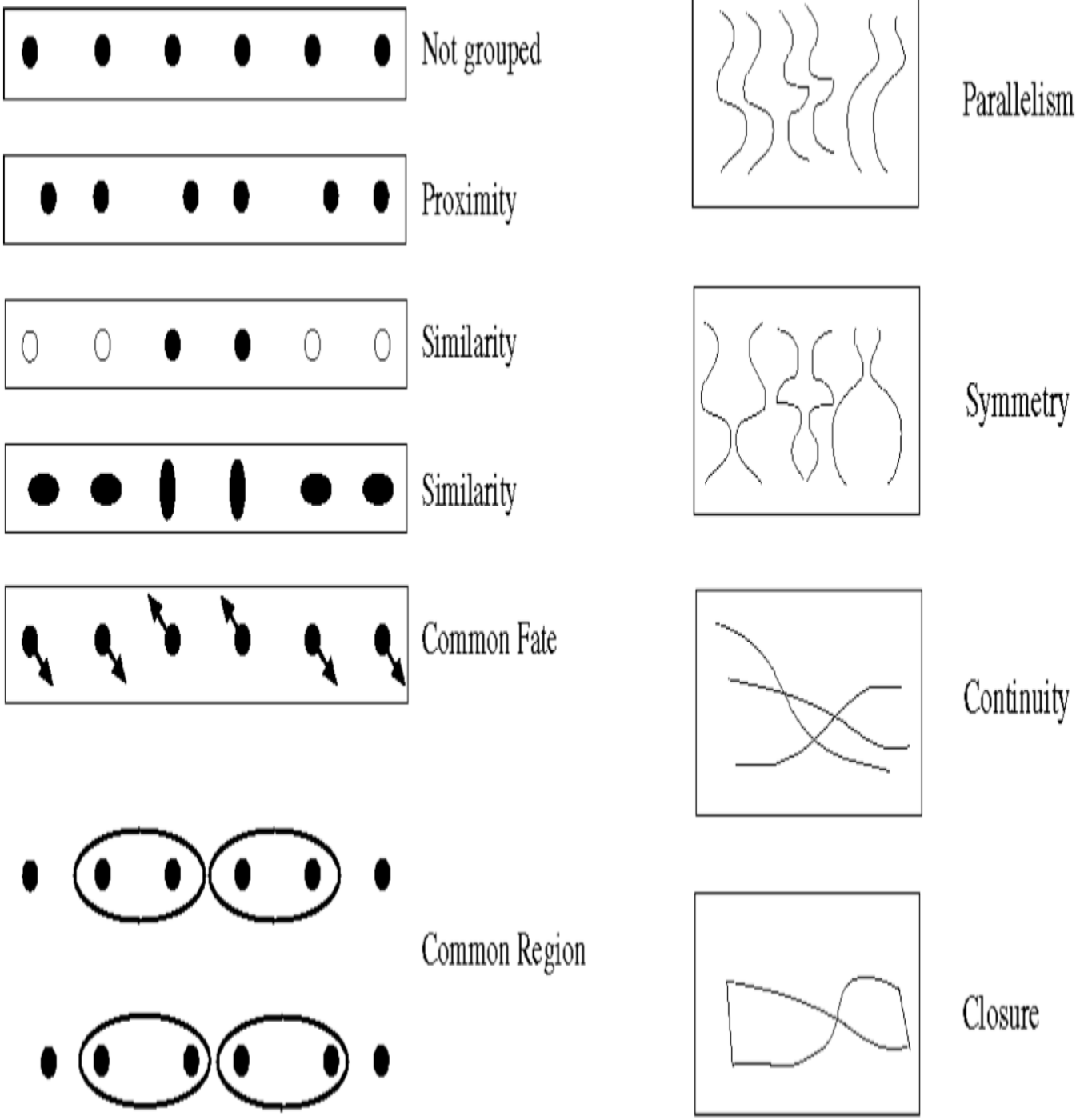
The Gestalt theory provided an approach to image segmentation with a basis in psychology, accounting for factors that cause people to see things as a "unified whole" to extract meaningful information. The fundamental viewpoint of the German Gestalt theorists can be summarized by what Gestalt psychologist Kurt Koffka stated, "The whole is other than the sum of the parts" (often incorrectly translated into "The whole is greater than the sum of the parts").



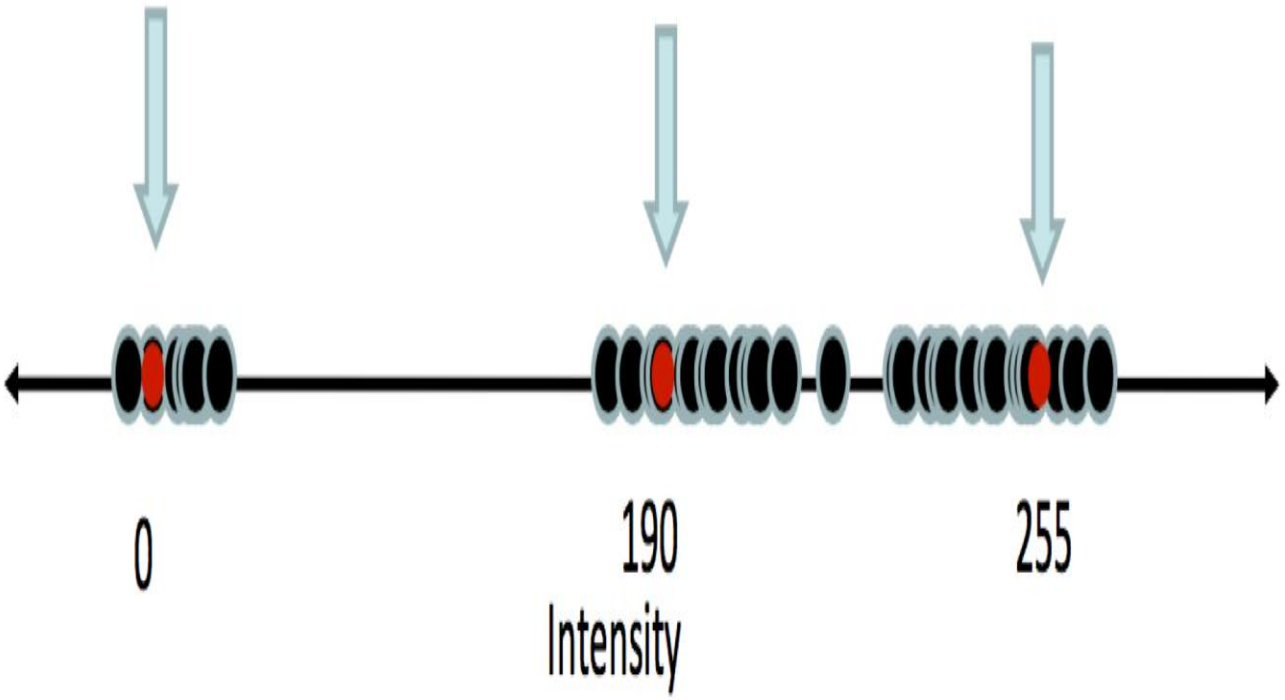
They identified proximity, similarity, common fate, common region, parallelism, symmetry, continuity,

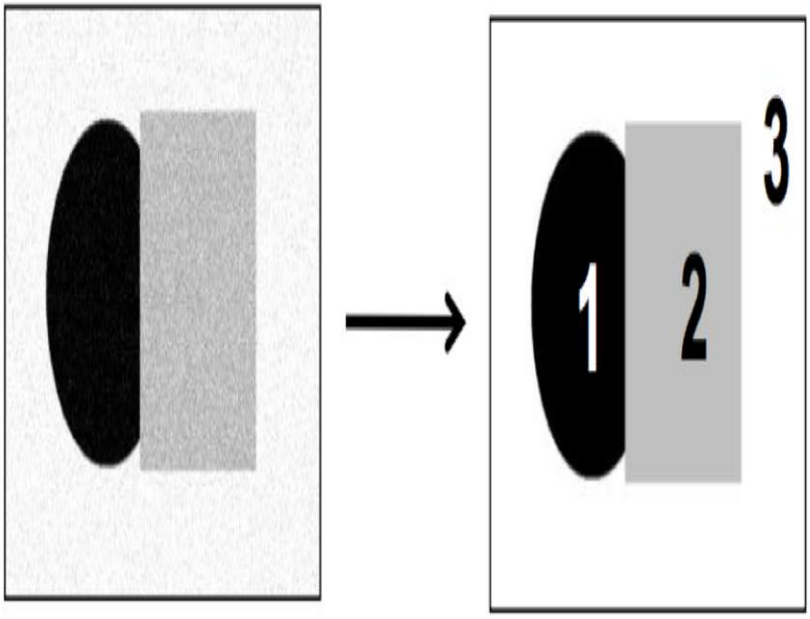
and closure as the factors that make people group certain visual elements together.

However, it is difficult to translate these factors into algorithms.



The goal of clustering is to group the pixels in an image into "clusters" that are similar in some respect. For example, clustering could involve choosing a certain number of cluster centers, which are representative of the different pixel intensities in the image and assigning each pixel in the image to the center it is the closest or the most similar to.



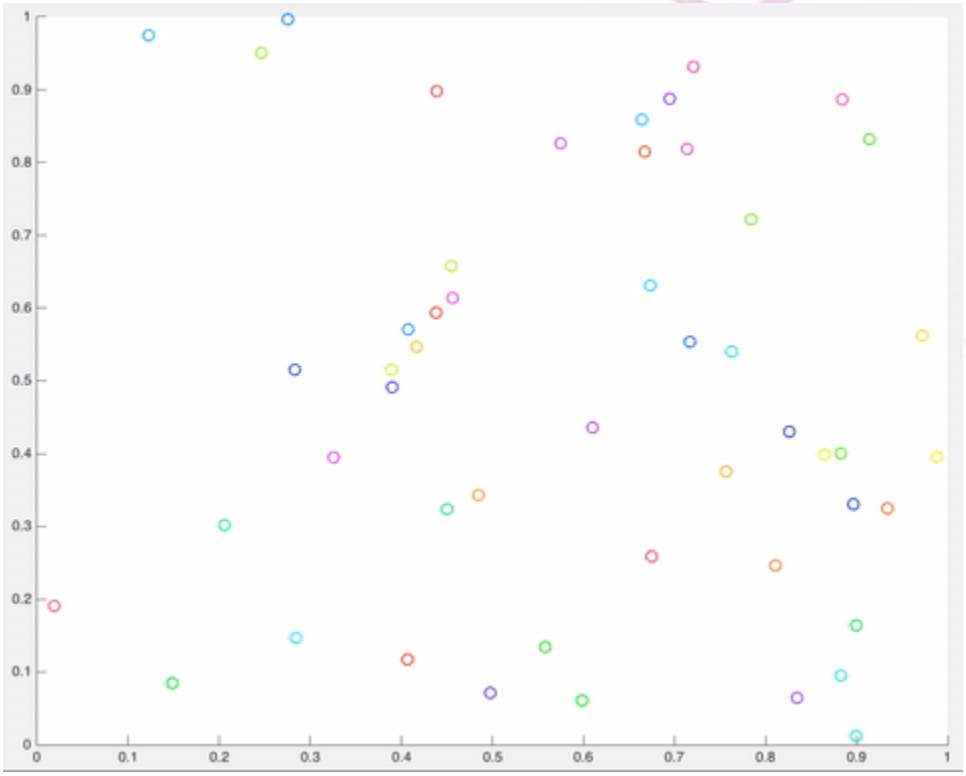


There are several clustering methods, three of which are introduced below. In the examples, we are measuring "similarity" by Euclidean distance.

Hierarchical Agglomerative Clustering (HAC): a bottom-up algorithm

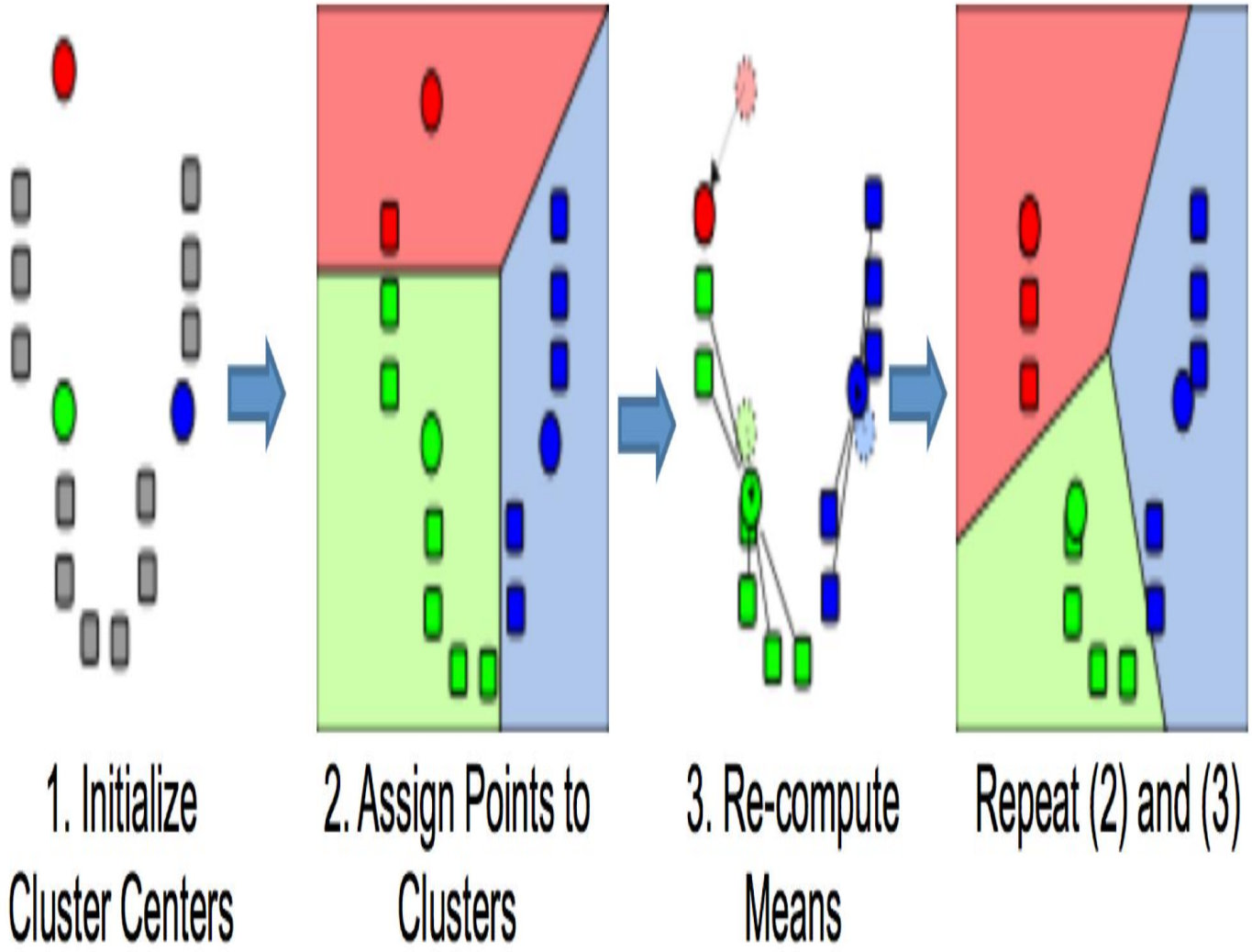
1. Say "Every point is its own cluster"

With K = 5 clusters:

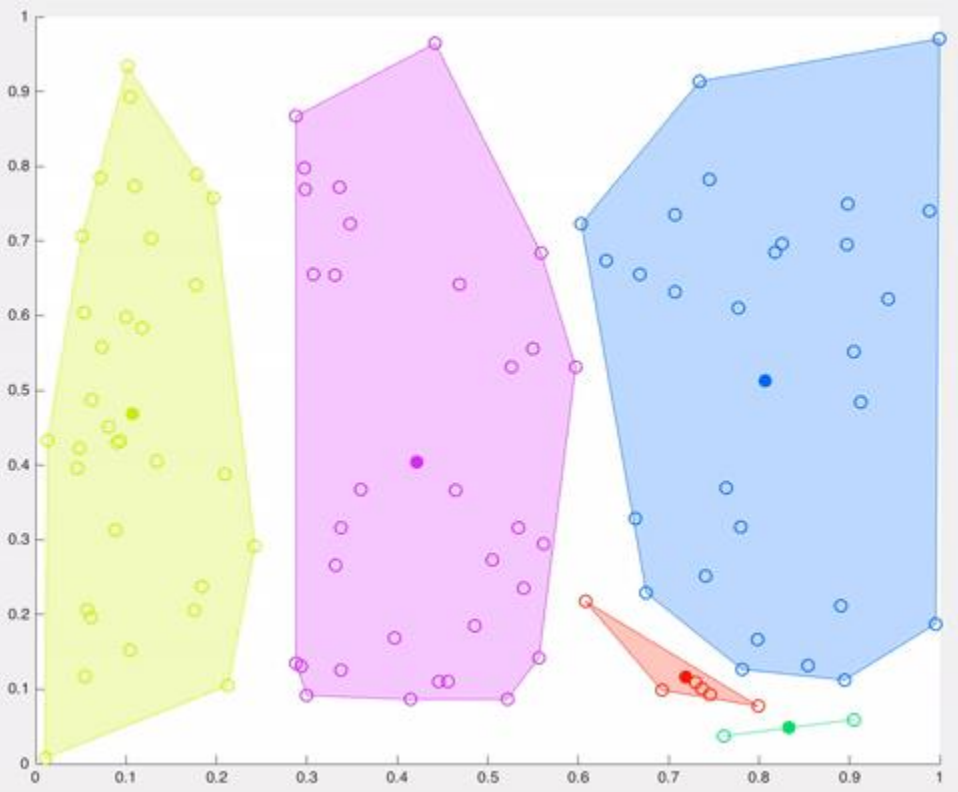


K means: a top-down algorithm

1. Initialize K clusters randomly, or greedily choose K.
2. Assign each pixel to the closest center.
3. Update cluster centers by computing the average of the pixels in the cluster.
4. Repeat steps 2 and 3 until no pixels change cluster centers.

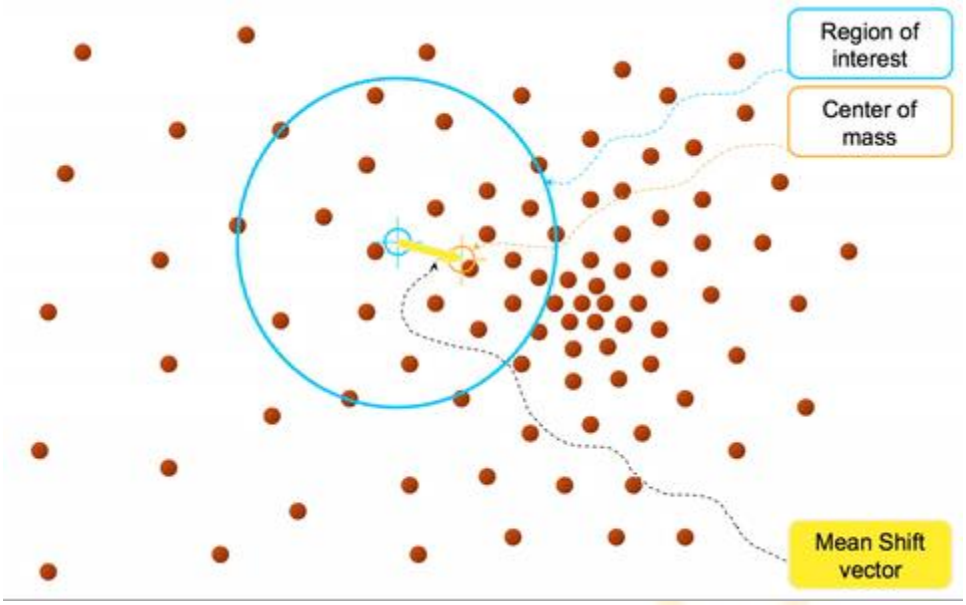


With K = 5 clusters:



Mean Shift

1. Initialize a random seed and window W.
2. Calculate the center of gravity (the "mean") of W.
3. Shift W to the mean.
4. Repeat steps 2 and 3 until convergence.



To read more about the mean shift algorithm, click here.

HAC vs. K means vs. Mean Shift

- Unlike K means and mean shift, HAC provides a hierarchy of clusters. However, the clusters may be imbalanced, and you still need to specify the number of clusters, as is the case with K means.
- K means finds cluster centers that are a good representation of the data. However, it is prone to being affected by outliers and local minima and can be slow in runtime. Because of this, K means is rarely used for pixel segmentation.
- Contrary to K means, mean shift is robust to outliers. However, the output depends on window size, and similar to K means, mean shift can be computationally expensive.

Grab Cat

A successful segmentation of an image should allow us to separate objects from the background and transfer them from one image to another. Here, we can segment pictures of cats using the K means algorithm (K = 5) and transfer the adorable felines onto different backgrounds.Picture and background:

Segmented: The pixels are partitioned into five groups, shown below. You can select the groups that form the kitten portion, discard the background (image 5, in this case), and move the segmented parts onto a separate background.

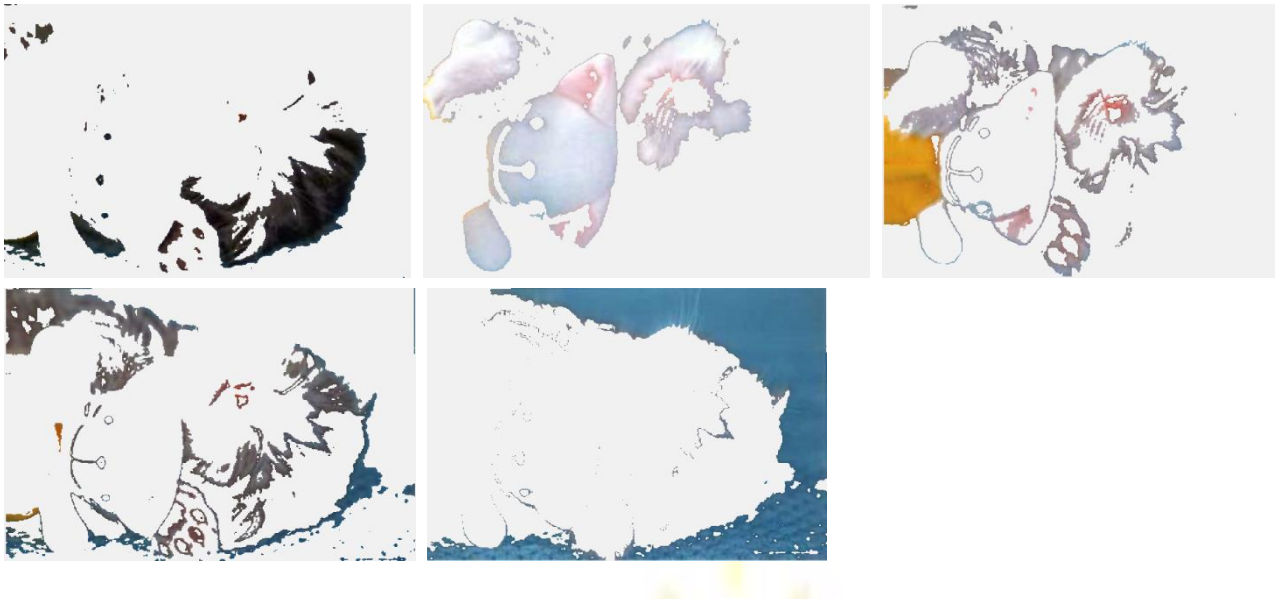


Image Processing Class #6 — Morphological Filter

This article is for sum up the lesson that I have learned in medical image processing class (EGBE443).

This article is about basic image processing. If you are new in this field, you can read my first post by clicking on the link below. :)

Image Processing Class (EGBE443) #0.1 — Image Aquisition
Hello everyone! My name is Pitchaya Thipkham. I'm a 4th-year student of Biomedical Engineering Department, Mahidol…
towardsdatascience.com

In the previous chapter, I've talked about a method to remove noise using median filter. Median filter makes image structure change a lot. A figure below shows the result of applying median filter to a binary image. The small structures, single line, and dot, are removed and small size holes are filled.

(a) (b) (c)

So in this chapter, I will introduce an idea which overcomes this problem. It is called "***Morphological Filter***".

Shrink and grow process

Morphological Filter

The idea of the morphological filter are shrink and let grow process. The word "shrink" means using median filter to round off the large structures and to remove the small structures and in grow process, remaining structures are grow back by the same amount.

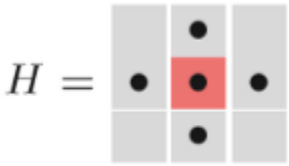
The morphological operation of the binary image is described first and will talk in the following outline.

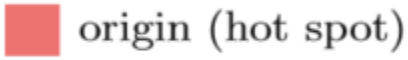
Outlines

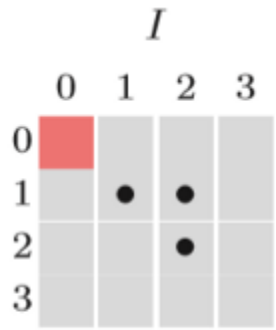
- The structuring element of a binary filter
- Dilation and Erosion
- Composite Operation

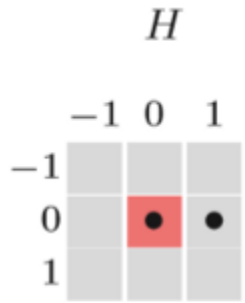
The structuring element

In morphological filter, each element in the matrix is called "structuring element" instead of coefficient matrix in the linear filter. The structuring elements contain only value 0 and 1. And the hot spot of the filter is the dark shade element.









$$I \equiv \mathcal{Q}_I = \{(1,1),(2,1),(2,2)\}$$

$$H \equiv \mathcal{Q}_H = \{(0,0),(1,0)\}$$

The binary image is described as sets of two-dimensional coordinate point. This is called ***"Point Set"*** Q and point set consist of the coordinate pair p = (u,v) of all foreground pixels. Some operations of point set are similar to the operation in others image. For inverting binary image is complement operation and combining two binary image use union operator. Shifting binary image I by some coordinate vector d by adding vector d to point p. Or reflection of binary image I by multiply -1 to point p.

Dilation and Erosion

- ***Dilation*** is a morphological operator which works for the grow process as I mentioned before. The equation of this operator is defined as

$$I \oplus H \equiv \{(\boldsymbol{p}+\boldsymbol{q}) \mid \text{for every } \boldsymbol{p} \in I, \boldsymbol{q} \in H\}.$$

- ***Erosion*** is a morphological operator which works for the shrink process as I mentioned before as well and the equation is defined as

$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid (\boldsymbol{p}+\boldsymbol{q}) \in I, \text{ for every } \boldsymbol{q} \in H\}.$$

Properties of dilation and erosion

- Commutative: only in dilation
- Associative: only in dilation

Note that: in erosion is in contrast to dilation, not have commutative property.

In addition, erosion and dilation are ***duels***, for a dilation of the foreground can be accomplished by an erosion of background and subsequent of the result in two different properties but work similarity

$$I \oplus H \equiv \overline{(\bar{I} \ominus H^*)},$$

$$I \ominus H \equiv \overline{(\bar{I} \oplus H^*)},$$

Composite Operation

In morphological process, dilation and erosion work together in composite operation. There are common way to represent the order of these two operations, opening and closing. Opening denotes an erosion followed by dilation and closing work in opposite way.

$$I \circ H = (I \ominus H) \oplus H.$$

$$I \bullet H = (I \oplus H) \ominus H.$$

Opening and Closing process respectively

The opening and closing also are dual in sense that opening the foreground is equal to closing the background.

$$I \circ H = \overline{(\bar{I} \bullet H)} \quad \text{and} \quad I \bullet H = \overline{(\bar{I} \circ H)}.$$

Morphological Filter can also apply to gray-scale image, but in the different definition. It is a generalization with MIN and MAX operators. I will describe in following outline.

Outlines

- Structuring Elements
- Dilation and Erosion
- Opening and Closing

Structuring Element

In gray-scale morphology, structuring elements are defined as real-value 2D functions instead of point sets

$$H(i,j) \in \mathbb{R}\,, \qquad \text{for } \; (i,j) \in \mathbb{Z}^2.$$

The value in H can be negative or zero value. But it contrast to linear convolution, zero elements are used to compute the result. And if you do not want to use the elements in some location, you can put no element in that location.

Dilation and Erosion

The result of dilation and erosion in gray-scale morphology is contributed from maximum and minimum operation.

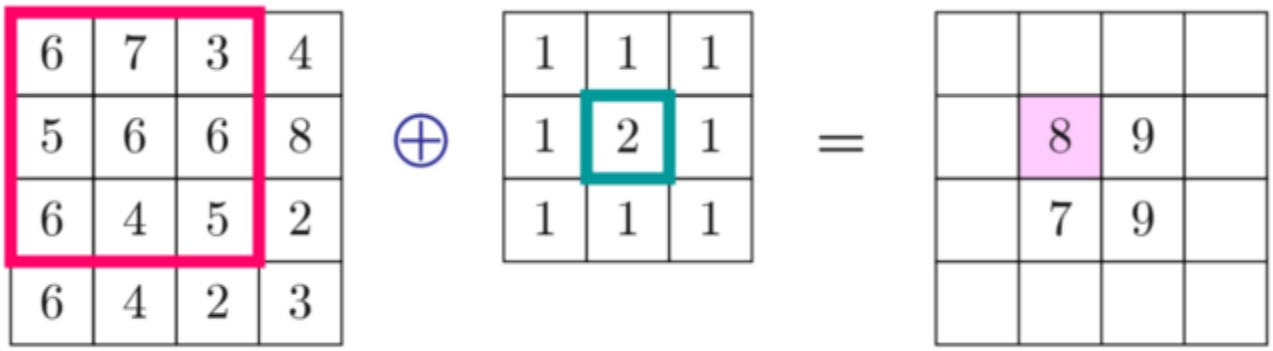
For dilation, the result is the maximum value of the value in H add to the current sub-image.

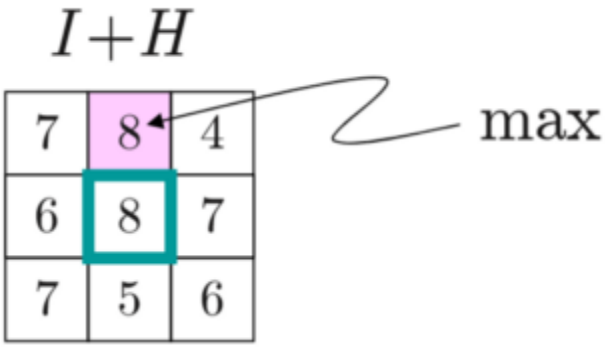
$$(I \oplus H)(u,v) = \max_{(i,j)\in H} \{I(u+i, v+j) + H(i,j)\}.$$

For erosion, the result is the minimum value of the difference.

$$(I \ominus H)(u,v) = \min_{(i,j)\in H} \{I(u+i, v+j) - H(i,j)\}.$$

These operations can cause the negative value, so we need to clamping the result after calculation.





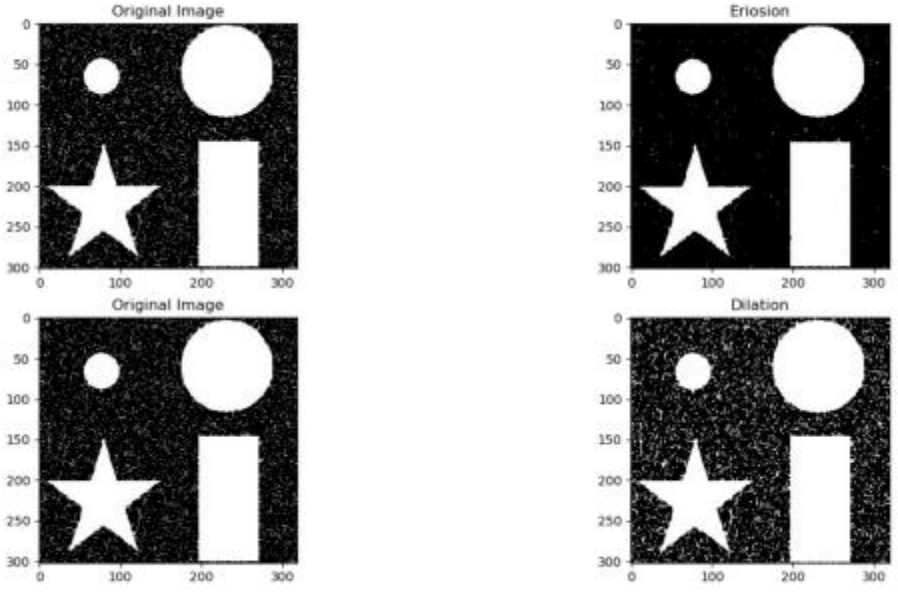
Example of dilation in gray-scale morphology

Opening and Closing

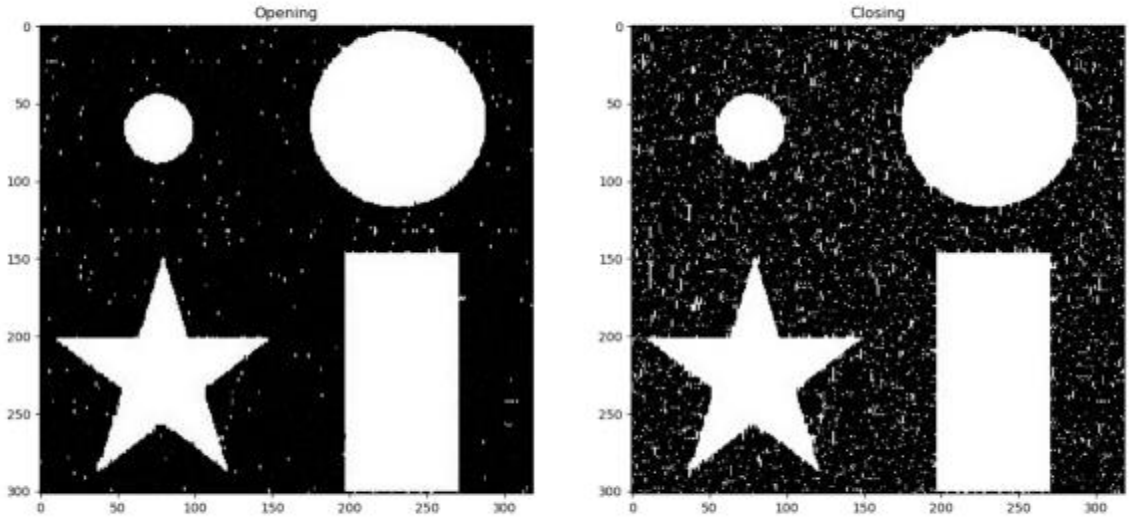
Opening and closing in gray-scale morphology work in the same way as in binary morphology. The difference is just the operator in dilation and erosion.

For implementation in Python 3 using OpenCV module, you can use the function cv2.erode(input,size) and cv2.dilate(input,size)

This is the result of the program, erosion and dilation, opening and closing.



The result of erosion and dilation of the program.



The result of opening and closing from the program.

This is my last article on image processing. Thanks for reading and following my post. Keep enjoying ***image processing****! I'll be back soon, Good luck. :)*

Fourier Series and Transform

In the last tutorial of Frequency domain analysis, we discussed that Fourier series and Fourier transform are used to convert a signal to frequency domain.

Fourier

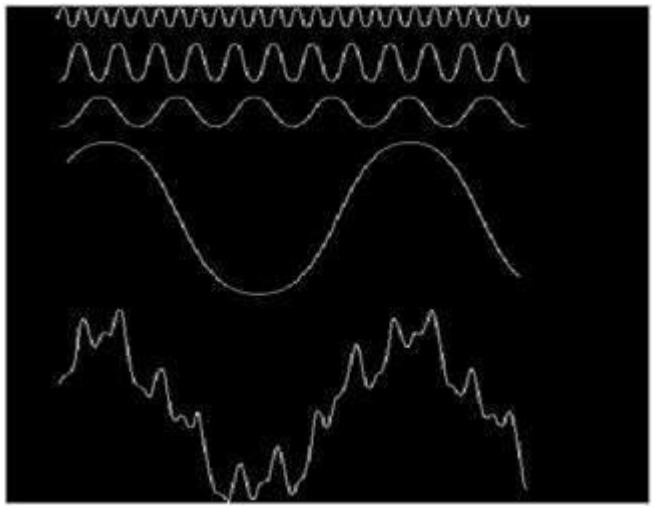
Fourier was a mathematician in 1822. He give Fourier series and Fourier transform to convert a signal into frequency domain.

Fourier Series

Fourier series simply states that , periodic signals can be represented into sum of sines and cosines when multiplied with a certain weight.It further states that periodic signals can be broken down into further signals with the following properties.

- The signals are sines and cosines
- The signals are harmonics of each other

It can be pictorially viewed as



In the above signal , the last signal is actually the sum of all the above signals. This was the idea of the Fourier.

How it is calculated.

Since as we have seen in the frequency domain , that in order to process an image in frequency domain , we need to first convert it using into frequency domain and we have to take inverse of the output to convert it back into spatial domain. That's why both Fourier series and Fourier transform has two formulas. One for conversion and one converting it back to the spatial domain.

Fourier series

The Fourier series can be denoted by this formula.

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} \, dx \, d$$

The inverse can be calculated by this formula.

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} \, du \, dv.$$

Fourier transform

The Fourier transform simply states that that the non periodic signals whose area under the curve is finite can also be represented into integrals of the sines and cosines after being multiplied by a certain weight.

The Fourier transform has many wide applications that include , image compression (e.g JPEG compression) , filtrering and image analysis.

Difference between Fourier series and transform

Although both Fourier series and Fourier transform are given by Fourier , but the difference between them is Fourier series is applied on periodic signals and Fourier transform is applied for non periodic signals

Which one is applied on images.

Now the question is that which one is applied on the images , the Fourier series or the Fourier transform. Well , the answer to this question lies in the fact that what images are. Images are non – periodic. And since the images are non periodic , so Fourier transform is used to convert them into frequency domain.

Discrete fourier transform.

Since we are dealing with images, and infact digital images , so for digital images we will be working on discrete fourier transform

Consider the above Fourier term of a sinusoid. It include three things.

- Spatial Frequency
- Magnitude
- Phase

The spatial frequency directly relates with the brightness of the image. The magnitude of the sinusoid directly relates with the contrast. Contrast is the difference between maximum and minimum pixel intensity. Phase contains the color information.

The formula for 2 dimensional discrete Fourier transform is given below.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)}$$

The discrete Fourier transform is actually the sampled Fourier transform, so it contains some samples that denotes an image. In the above formula f(x,y) denotes the image , and F(u,v) denotes the discrete Fourier transform. The formula for 2 dimensional inverse discrete Fourier transform is given below.

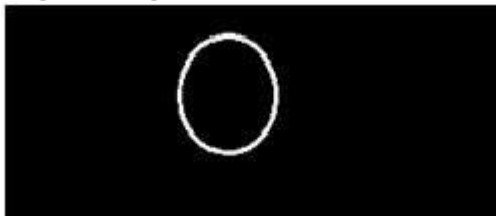
$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

The inverse discrete Fourier transform converts the Fourier transform back to the image

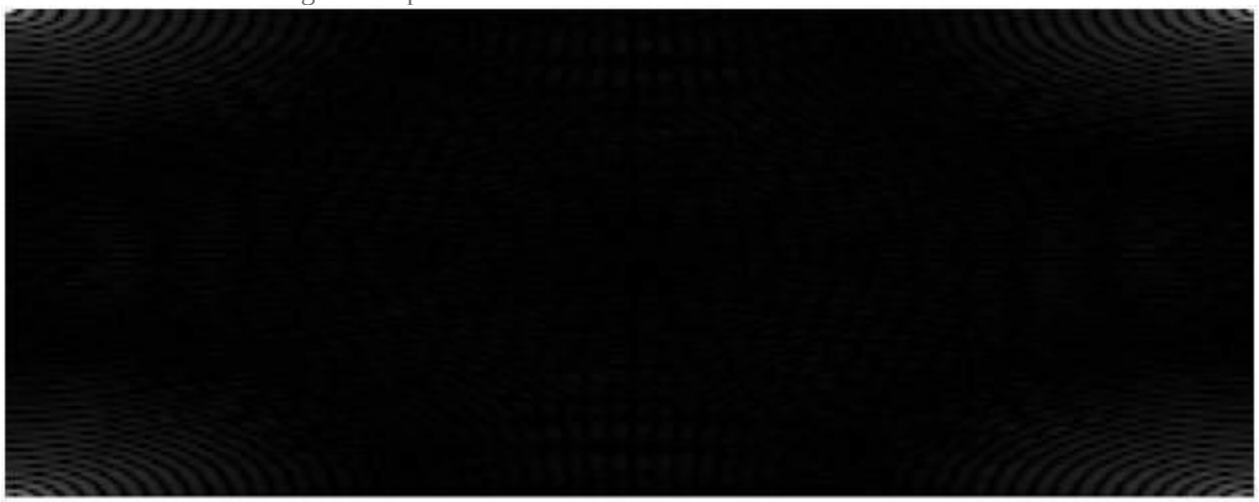
Consider this signal.

Now we will see an image , whose we will calculate FFT magnitude spectrum and then shifted FFT magnitude spectrum and then we will take Log of that shifted spectrum.

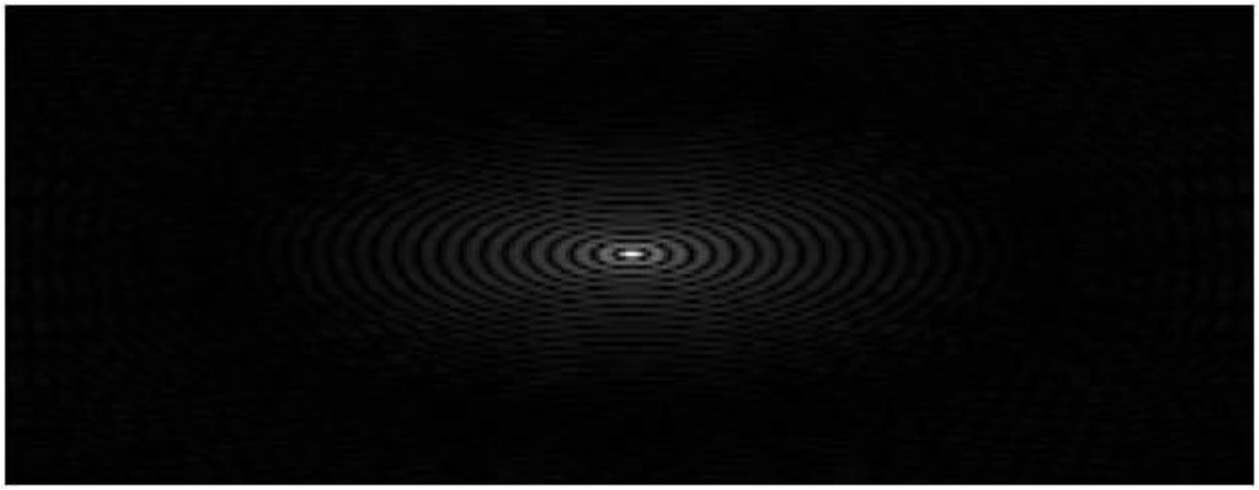
Original Image



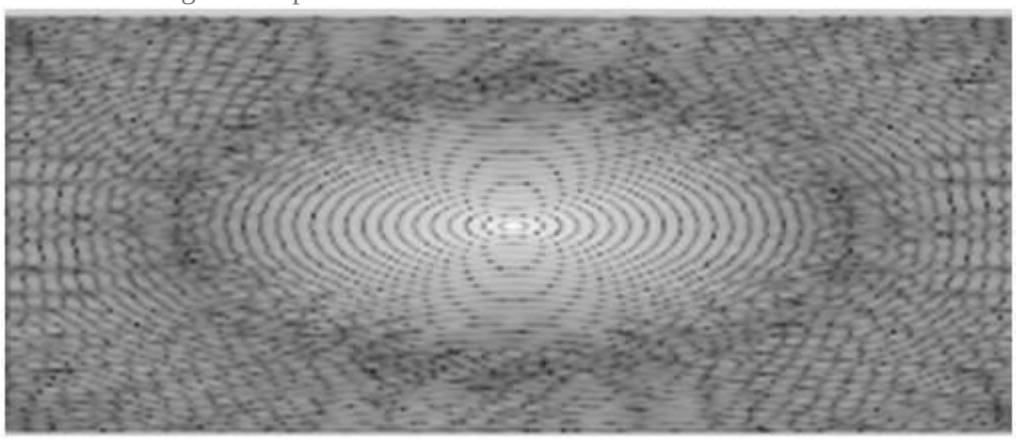
The Fourier transform magnitude spectrum



The Shifted Fourier transform



The Shifted Magnitude Spectrum



Modern Baby Names

UNIT-IV

The Computer Vision Pipeline, Part 4: feature extraction

Feature extraction

Feature extraction is a core component of the computer vision pipeline. In fact, the entire deep learning model works around the idea of extracting **useful features** which clearly define the objects in the image. We're going to spend a little more time here because it's important that you understand what a feature is, what a vector of features is, and why we extract features.

A feature in machine learning is an individual measurable property or characteristic of a phenomenon being observed. Features are the input that you feed to your machine learning model to output a prediction or classification. Suppose you want to predict the price of a house, your input features (properties) might include: square_foot, number_of_rooms, bathrooms, etc. and the model will output the predicted price based on the values of your features. Selecting good features that clearly distinguish your objects increases the predictive power of machine learning algorithms.

What is a feature in computer vision?

In computer vision, a feature is a measurable piece of data in your image which is unique to this specific object. It may be a distinct color in an image or a specific shape such as a line, edge, or an image segment. A good feature is used to distinguish objects from one another. For example, if I give you a feature like a wheel, and ask you to guess whether the object is a motorcycle or a dog. What would your guess be? A motorcycle. Correct! In this case, the wheel is a strong feature that clearly distinguishes between motorcycles and dogs. If I give you the same feature (a wheel) and ask you to guess whether the object is a bicycle or a motorcycle. In this case, this feature isn't strong enough to distinguish between both objects. Then we need to look for more features like a mirror, license plate, maybe a pedal that collectively describes an object.

In machine learning projects, we want to transform the raw data (image) into a *features vector* to show our learning algorithm how to learn the characteristics of the object.

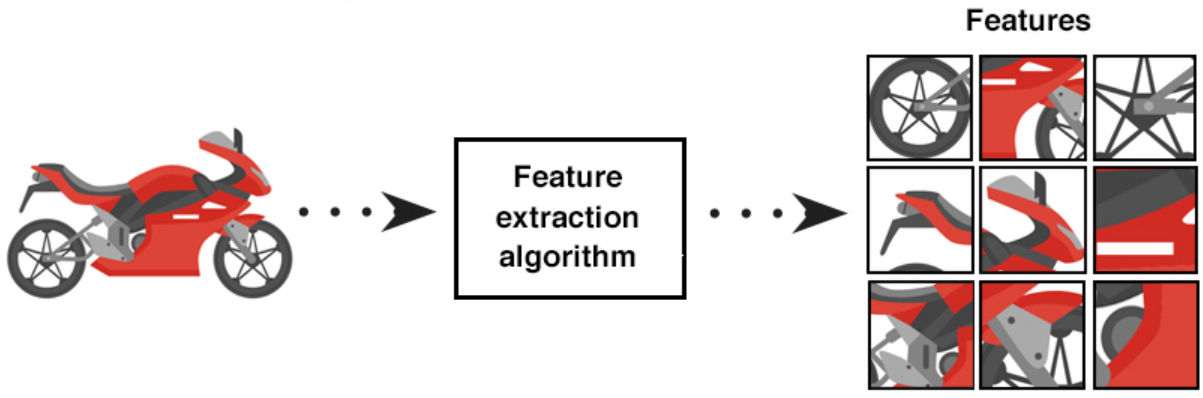


Figure 1

In the image above, we feed the raw input image of a motorcycle to a feature extraction algorithm. Let's treat the feature extraction algorithm as a black box for now and we'll come back to it soon. For now, we need to know that the extraction algorithm produces a ***vector*** that contains a list of features. This is called **features vector** which is a 1D array that makes a robust representation of the object.

It is important to call out that the image above reflects features extracted from just one motorcycle. A very important characteristic of a feature is **repeatability**. As in the feature should be able to detect the motorcycles in general not just this specific one. So, in real world problems, the feature will not be an exact copy of the piece in the input image.

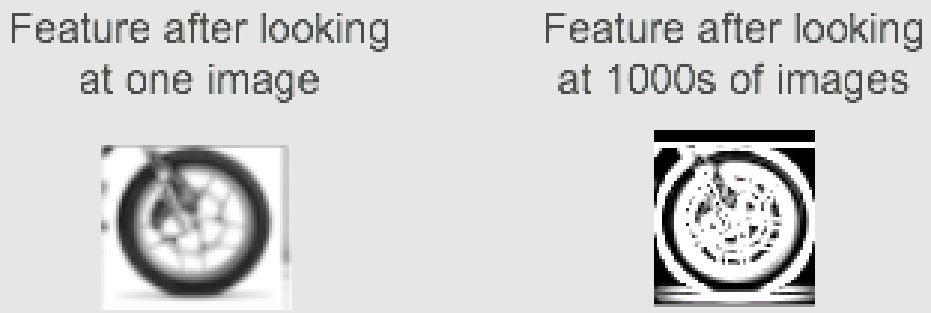


Figure 1.2

If we take the wheel feature for example, the feature will not look exactly like the wheel on just one motorcycle. Instead, it looks like a circular shape with some patterns that identify wheels in all images in the training dataset. When the feature extractor sees thousands of images of motorcycles, it recognizes patterns that define wheels in general regardless of where they appear in the image and what type of motorcycle it is.

What makes a good (useful) feature?

Machine learning models are only as good as the features you provide. That means coming up with good features is an important job in building ML models. But what makes a good feature? And how can you tell?

Let's discuss this by an example: Suppose we want to build a classifier to tell the difference between two types of dogs, Greyhound and Labrador. Let's take two features and evaluate them: 1) the dogs' height and 2) their eye color.

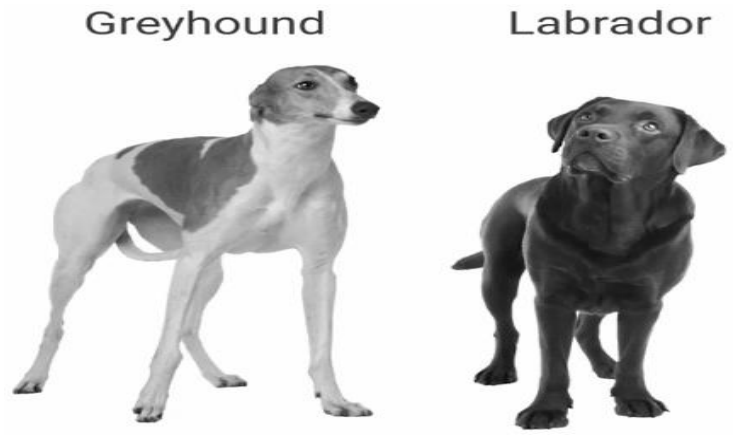


Figure 3

Let's begin with height. How useful do you think this feature is? Well, on average, Greyhounds tend to be a couple of inches taller than Labradors, but not always. A lot of variation exists in the world. Let's evaluate this feature across different values in both breeds population. We can visualize the height distribution on a toy example in the histogram below:

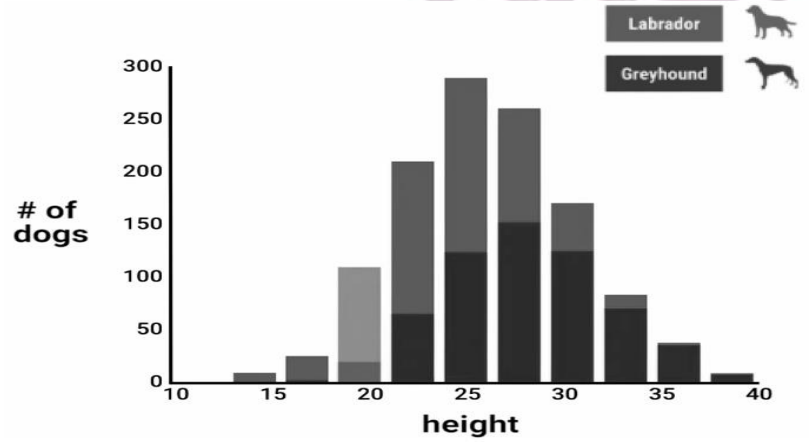


Figure 4

From the histogram above, we can see that if the dog's height is twenty inches or less, there's more than an 80% probability that this dog is a Labrador. On the other side of the histogram, if we look at dogs which are taller than thirty inches, we can be pretty confident that the dog is a greyhound. Now, what about the data in the middle of the histogram (heights from twenty to thirty inches)? We can see that the probability of each type of dog is pretty close. The thought process in this case is as follows:

if height <=20: return higher probability to Labrador
if height >=30: return higher probability to greyhound
if 20 < height >30: look for other features to classify the object

The "height" of the dog in this case is a useful feature because it helps (adds information) distinguish between both dog types. We can keep it, but it doesn't distinguish between Greyhounds and Labradors in all cases, which is fine. In ML projects, there's usually no one feature which can classify all objects on its own. This is why with machine learning we almost always need multiple features where each feature captures a different type of information. If only one feature does the job, we can write if-else statements instead of bothering with training a classifier.

Similar to what we did earlier with color conversion (color vs grayscale), to figure out which features you should use for a specific problem, do a thought experiment. Pretend you are the classifier. If you want to differentiate between greyhounds and labradors, what information you would need to know? You might ask about the hair length, or the body size, to color, and so on.

Another quick example of a non-useful feature to drive this idea home. Let's look at eye color. For this toy example, imagine that we have only two eye colors, blue and brown. Here's what a histogram might look like for this example:

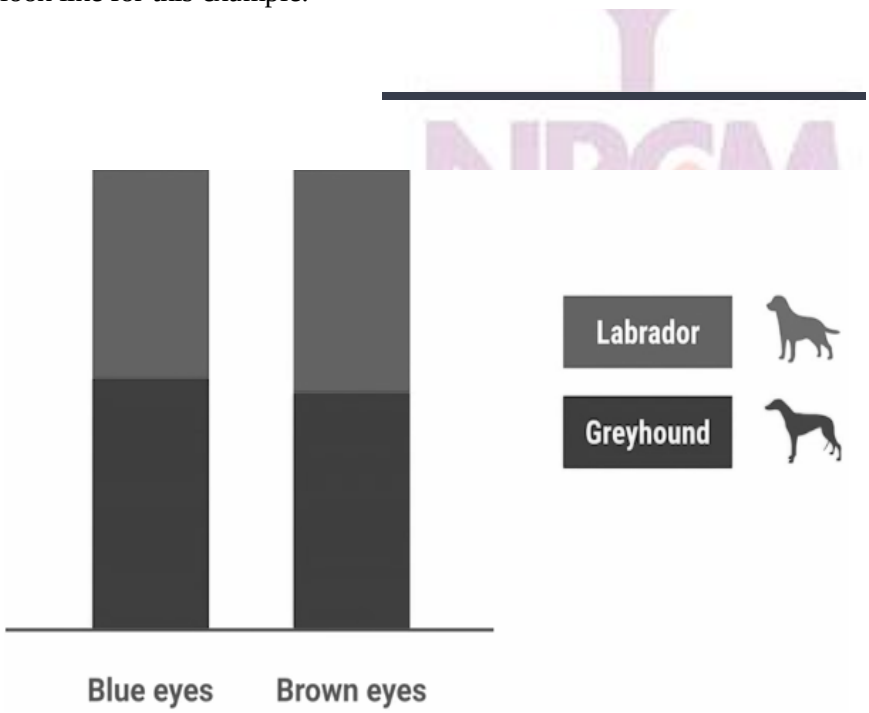


Figure 5

It's clear that for most values, the distribution is about 50/50 for both types. Practically this feature tells us nothing because it doesn't correlate with the type of dog. Hence, it doesn't distinguish between Greyhounds and Labradors.

A good feature will help us recognize an object in all the ways it may appear.

Characteristics of a good feature:

- Identifiable
- Easily tracked and compared
- Consistent across different scales, lighting conditions, and viewing angles
- Still visible in noisy images or when only part of an object is visible

Extracting features (hand-craft vs automatic extracting)

Okay, this a can be a large topic in machine learning that needs an entire book to discuss. Typically described in the context of a topic called *feature engineering*. In this section we're only concerned with extracting features in images. I'm going to touch on the idea quickly.

Traditional machine learning uses hand-crafted features

In traditional machine learning problems, we spend a good amount of time in **manual** features selection and engineering. In this process we rely on our domain knowledge (or partnering with domain experts) to create features which make machine learning algorithms work better. We then feed the produced features to a classifier like Support Vector Machines (SVM) or Adaboost to predict the output. Some of the handcrafted feature sets are:

- Histogram of Oriented Gradients (HOG)
- Haar Cascades
- Scale-Invariant Feature Transform (SIFT)
- Speeded Up Robust Feature (SURF)

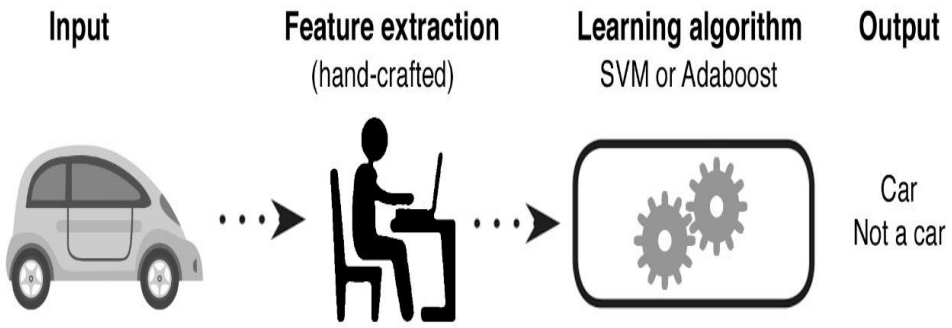


Figure 6

Deep learning automatically extracts features

In deep learning, we don't need to manually extract features from the image. The network **automatically** extracts features and learns their importance on the output by applying weights to its connections. You feed the raw image to the network and, as it passes through the network layers, it identifies patterns within the image to create features. Neural networks can be thought of as feature extractors + classifiers which are end-to-end trainable as opposed to traditional ML models that use hand-crafted features.

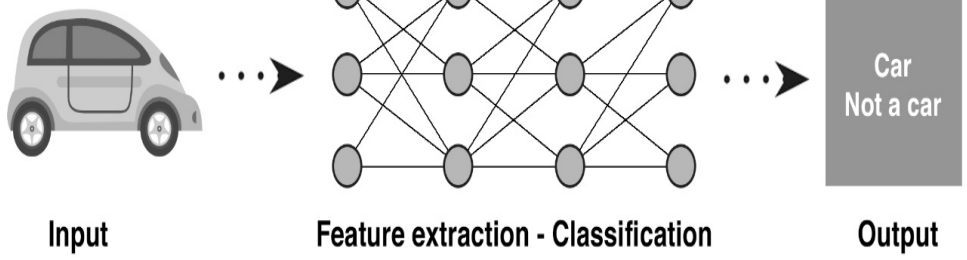


Figure 7

How do neural networks distinguish useful features from the non-useful features?

You might get the impression that neural networks only understands the useful features but that's not entirely true. Neural Networks scoop all the features available and give them random weights. During the training process it adjusts these weights to reflect their importance and how they should impact the output prediction. The patterns with the highest appearance frequency will have higher weights and in turn are considered more useful features. Whereas, features with lowest weights will have very low impact on the output. This learning process is going to be discussed in deep details in the next chapter.

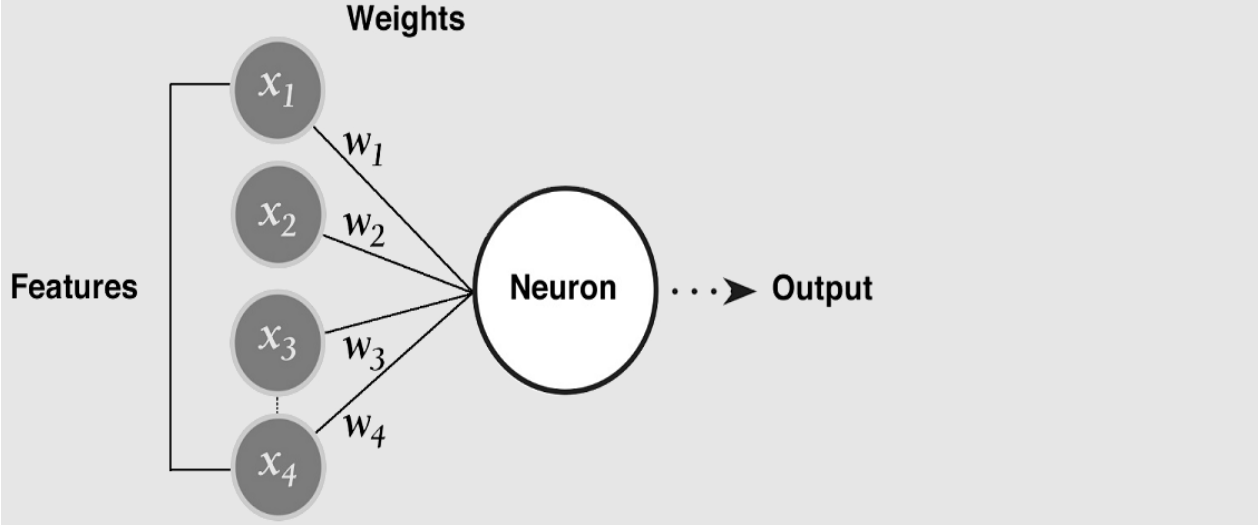
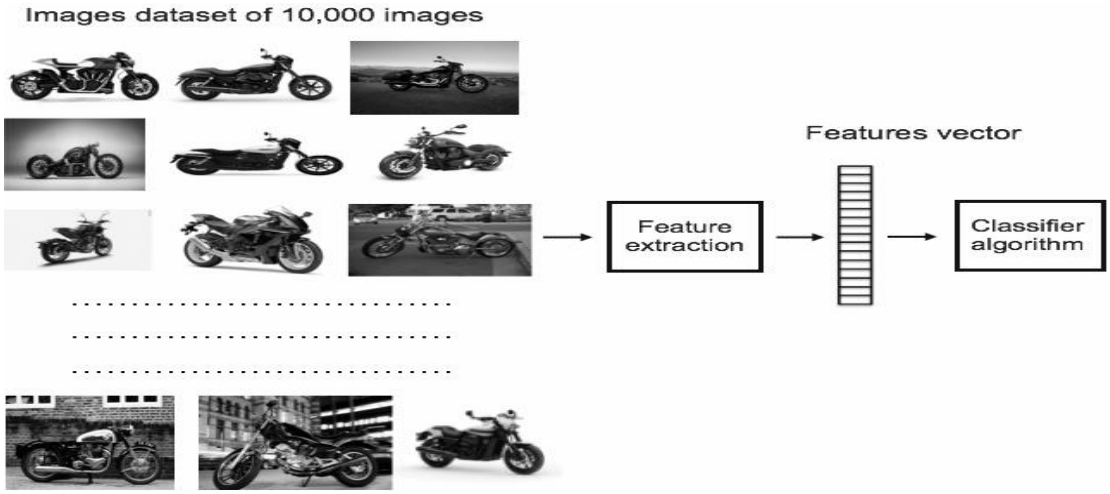


Figure 8

Why use features?

The input image has too much extra information which isn't necessary for classification. Therefore, the first step after preprocessing the image is to simplify the image by extracting the important information and throwing away non-essential information. By extracting important colors or image segments, we can transform complex and large image data into smaller sets of features. This makes the task of classifying images based on their features done simpler and faster.

Consider the example below. Suppose we're given a dataset of 10,000 images of motorcycles each of 1,000 width x 1,000 height. Some images have solid backgrounds and others have busy backgrounds of unnecessary data. When these thousands of images are fed to the feature extraction algorithms, we lose all the unnecessary data that isn't important to identify motorcycles and we only keep a consolidated list of useful features which can then be fed directly to the classifier. This process is a lot simpler than having the classifier look at a dataset of 10,000 images to learn the properties of motorcycles.



HOG (Histogram of Oriented Gradients): An Overview

Mathematics, Paper Explanation and Code from scratch for beginners

What is it?

Histogram of Oriented Gradients, also known as HOG, is a feature descriptor like the Canny Edge Detector, SIFT (Scale Invariant and Feature Transform) . It is used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in the localized portion of an image. This method is quite similar to Edge Orientation Histograms and Scale Invariant aFeature Transformation (SIFT). The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features. For the regions of the image it generates histograms using the magnitude and orientations of the gradient.

Steps to calculate HOG Features

1. Take the input image you want to calculate HOG features of. Resize the image into an image of 128x64 pixels (128 pixels height and 64 width). This dimension was used in the paper and was suggested by the authors as their primary aim with this type of detection was to obtain better results on the task of pedestrian detection. As the authors of this paper were obtaining exceptionally perfect results on the MIT pedestrian database, they decided to produce a new and significantly more challenging dataset called the 'INRIA' dataset (http://pascal.inrialpes.fr/data/human/), containing 1805 (128x64) images of humans cropped from a varied set of personal photos.

Figure 1

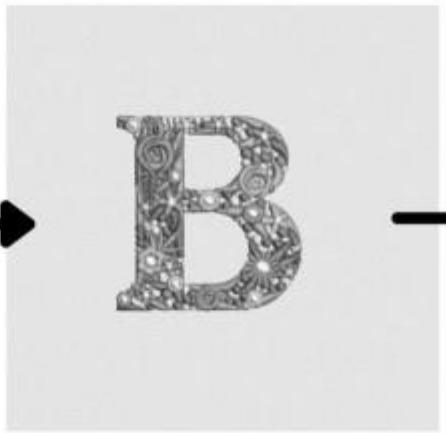


Figure 2

Figure 3

Figure 1 : The image imported to get HOG features of. Figure 2 : The imported image grayscale for the process. Figure 3 : Resized and grayscale image of the imported image. (Image by author)

2. The gradient of the image is calculated. The gradient is obtained by combining magnitude and angle from the image. Considering a block of 3x3 pixels, first Gx and Gy is calculated for each pixel. First Gx and Gy is calculated using the formulae below for each pixel value .

$$G_x(r,c) = I(r,c+1) - I(r,c-1) \quad G_y(r,c) = I(r-1,c) - I(r+1,c)$$

where r, c refer to rows and columns respectively. (Image by author)

After calculating Gx and , magnitude and angle of each pixel is calculated using the formulae mentioned below.

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \quad Angle(\theta) = |\tan^{-1}(G_y/G_x)|$$

Figure 4

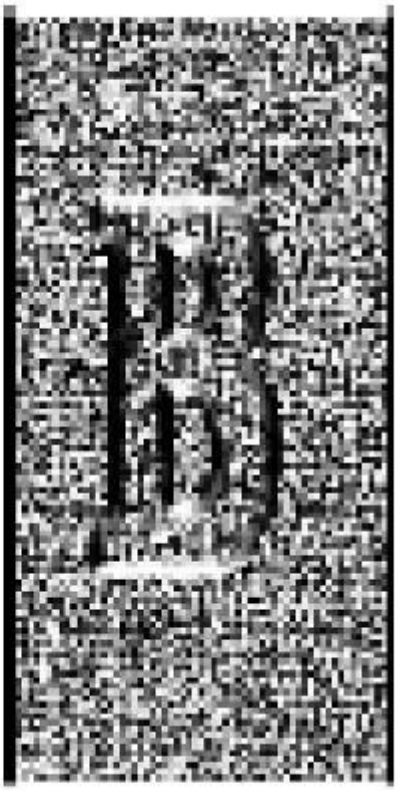


Figure 5

Figure 4 : Visualization of magnitude of the image. Figure 5 : Visualization of angle of the image. (Image by author)

3. After obtaining the gradient of each pixel, the gradient matrices (magnitude and angle matrix) are divided into 8x8 cells to form a block. For each block, a 9-point histogram is calculated. A 9-point histogram develops a histogram with 9 bins and each bin has an angle range of 20 degrees. Figure 8 represents a 9 bin histogram in which the values are allocated after calculations. Each of these 9-point histograms can be plotted as histograms with bins outputting the intensity of the gradient in that bin. As a block contains 64 different values, for all 64 values of magnitude and gradient the following calculation is performed. As we are using 9 point histograms, hence :

$$Number\ of\ bins = 9(ranging\ from\ 0^\circ\ to\ 180^\circ)$$
$$Step\ size(\Delta\theta) = 180^\circ\ /\ Number\ of\ bins = 20^\circ$$

(Image by author)

Each Jth bin, bin will have boundaries from :

$$[\Delta\theta \cdot j, \Delta\theta \cdot (j+1)]$$

(Image by author)

Value of the centre of each bin will be :

$$C_j = \Delta\theta(j + 0.5)$$

(Image by author)

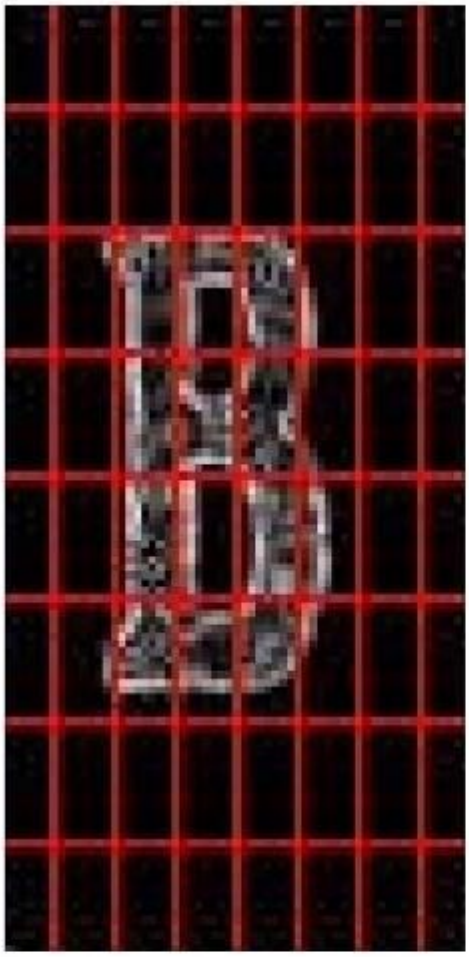


Figure 6

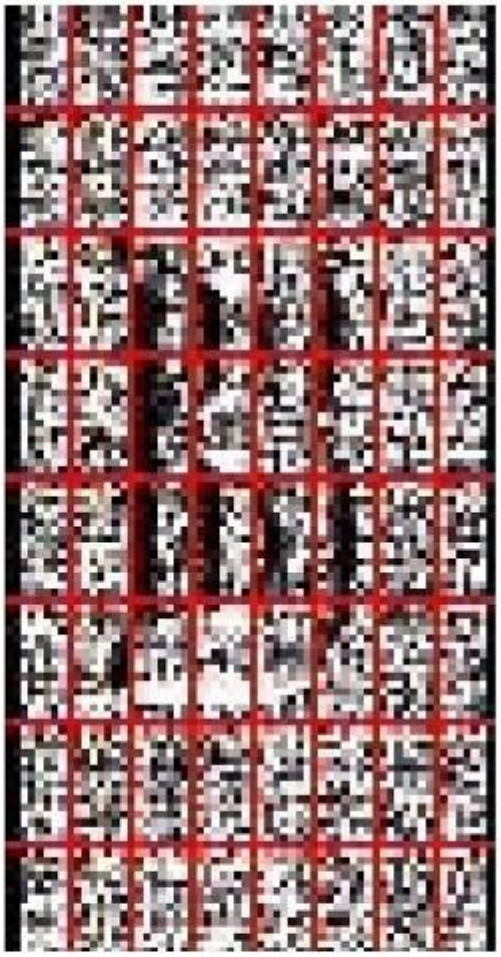


Figure 7

Figure 6 : 8x8 blocks on the magnitude image. Figure 7 : 8x8 blocks on an angle image. (Image by author)

Value									
Bins	0	20	40	60	80	100	120	140	160

Figure 8 : Representation of a 9 bin histogram. This one single histogram will be unique for one 8x8 block made up of 64 cells. All 64 cells will add their Vj and Vj+1 value to the jth and (j+1)th index of the array respectively. (Image by author)

4. For each cell in a block, we will first calculate the jth bin and then the value that will be provided to the jth and (j+1)th bin respectively. The value is given by the following formulae :

$$j = \lfloor \left(\frac{\theta}{\Delta\theta} - \frac{1}{2} \right) \rfloor$$

$$V_j = \mu \cdot \left[\frac{\theta}{\Delta\theta} - \frac{1}{2} \right]$$

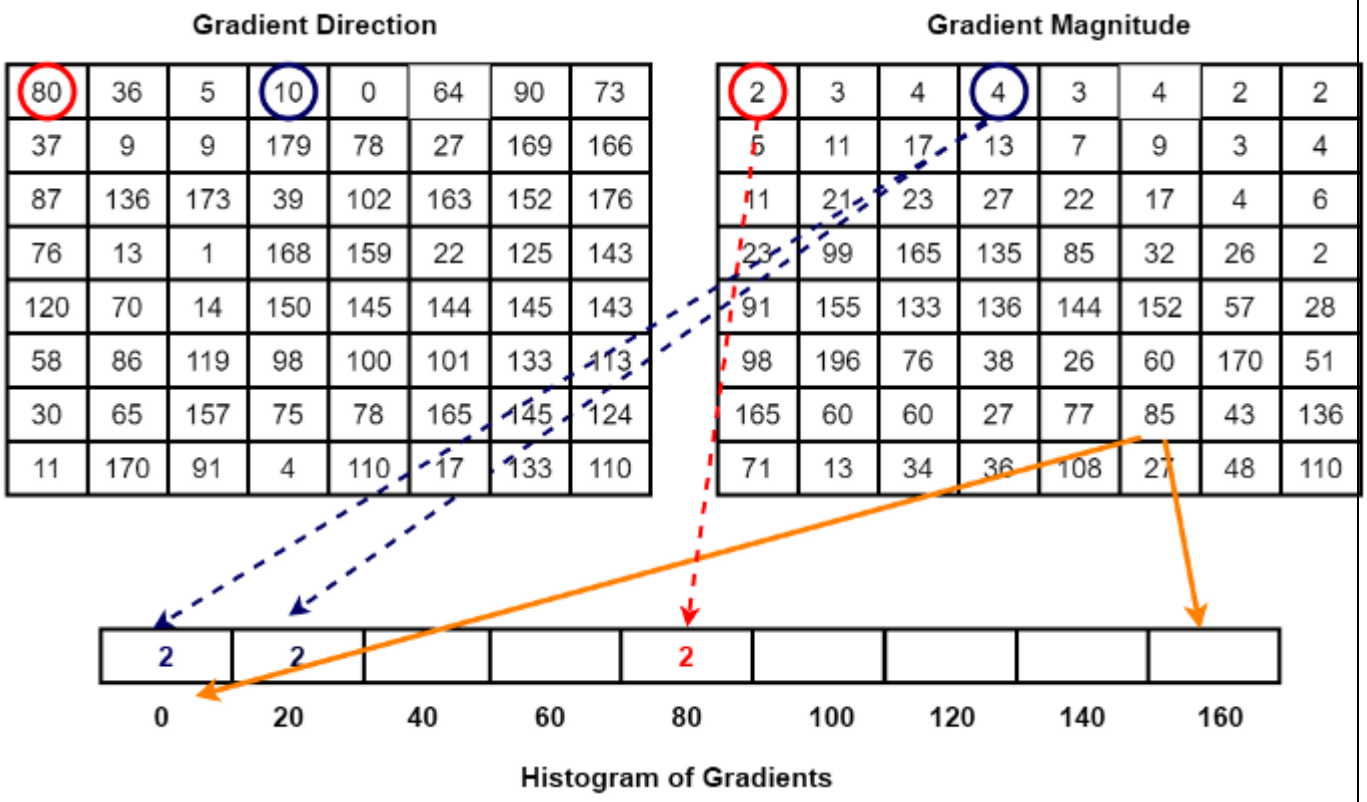
$$V_{j+1} = \mu \cdot \left[\frac{\theta - C_j}{\Delta\theta} \right]$$

(Image by author)

5. An array is taken as a bin for a block and values of Vj and Vj+1 is appended in the array at the index of jth and (j+1)th bin calculated for each pixel.

6. The resultant matrix after the above calculations will have the shape of 16x8x9.

7. Once histogram computation is over for all blocks, 4 blocks from the 9 point histogram matrix are clubbed together to form a new block (2x2). This clubbing is done in an overlapping manner with a stride of 8 pixels. For all 4 cells in a block, we concatenate all the 9 point histograms for each constituent cell to form a 36 feature vector.



Method for calculation of 9 bin histograms is illustrated in the above image. (Image by author) Inspired by [https://www.programmersought.com/article/42554276349/]

$$f_{bi} = [b_1, b_2, b_3, ..., b_{36}]$$

(Image by author)

Traversing of 2x2 grid box around the image in order to make a combined fbi from 4 blocks. (Image by author)

8. Values of fb for each block is normalized by the L2 norm :

$$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{\|f_{bi}\|^2 + \varepsilon}}$$

Where ε is a small value added to the square of fb in order to avoid zero division error. In code value taken of is 1e-05. (Image by author)

9. To normalize, the value of k is first calculated by the following formulae :

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + ... + b_{36}^2}$$

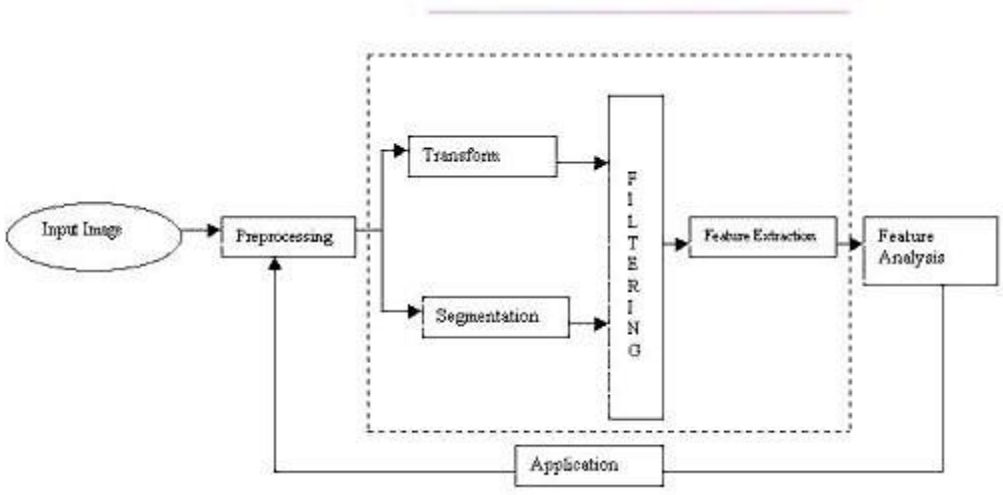
$$f_{bi} = \left[\left(\frac{b_1}{k} \right), \left(\frac{b_2}{k} \right), \left(\frac{b_3}{k} \right), ..., \left(\frac{b_{36}}{k} \right) \right]$$

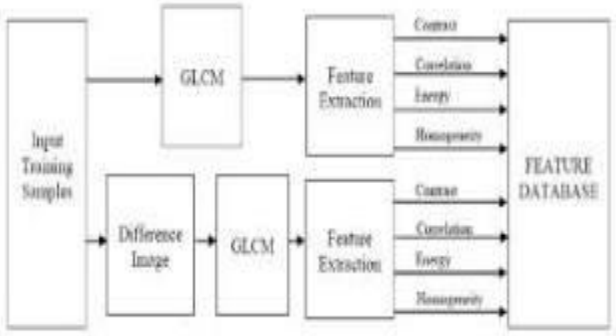
(Image by author)

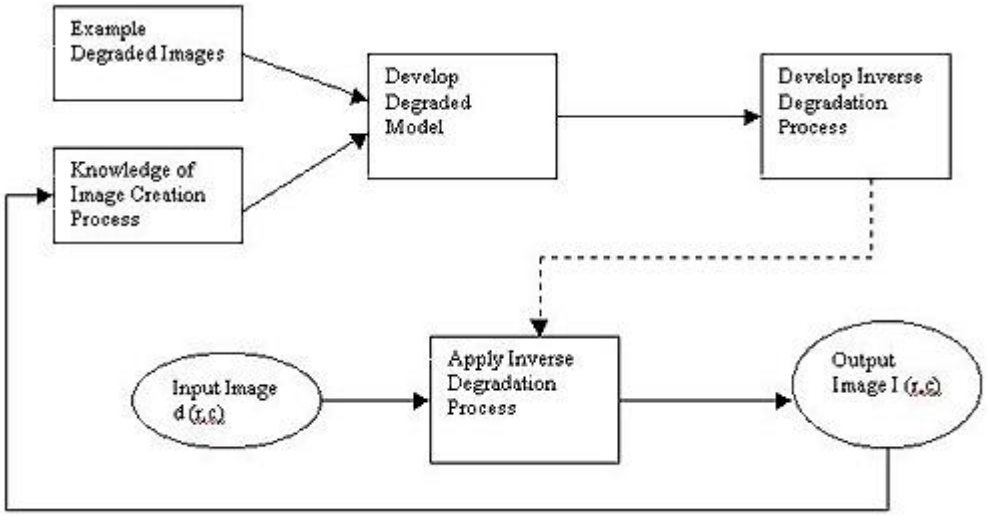
10. This normalization is done to reduce the effect of changes in contrast between images of the same object. From each block. A 36 point feature vector is collected. In the horizontal direction there are 7 blocks and in the vertical direction there are 15 blocks. So the total length of HOG features will be : 7 x 15 x 36 = 3780. HOG features of the selected image are obtained.

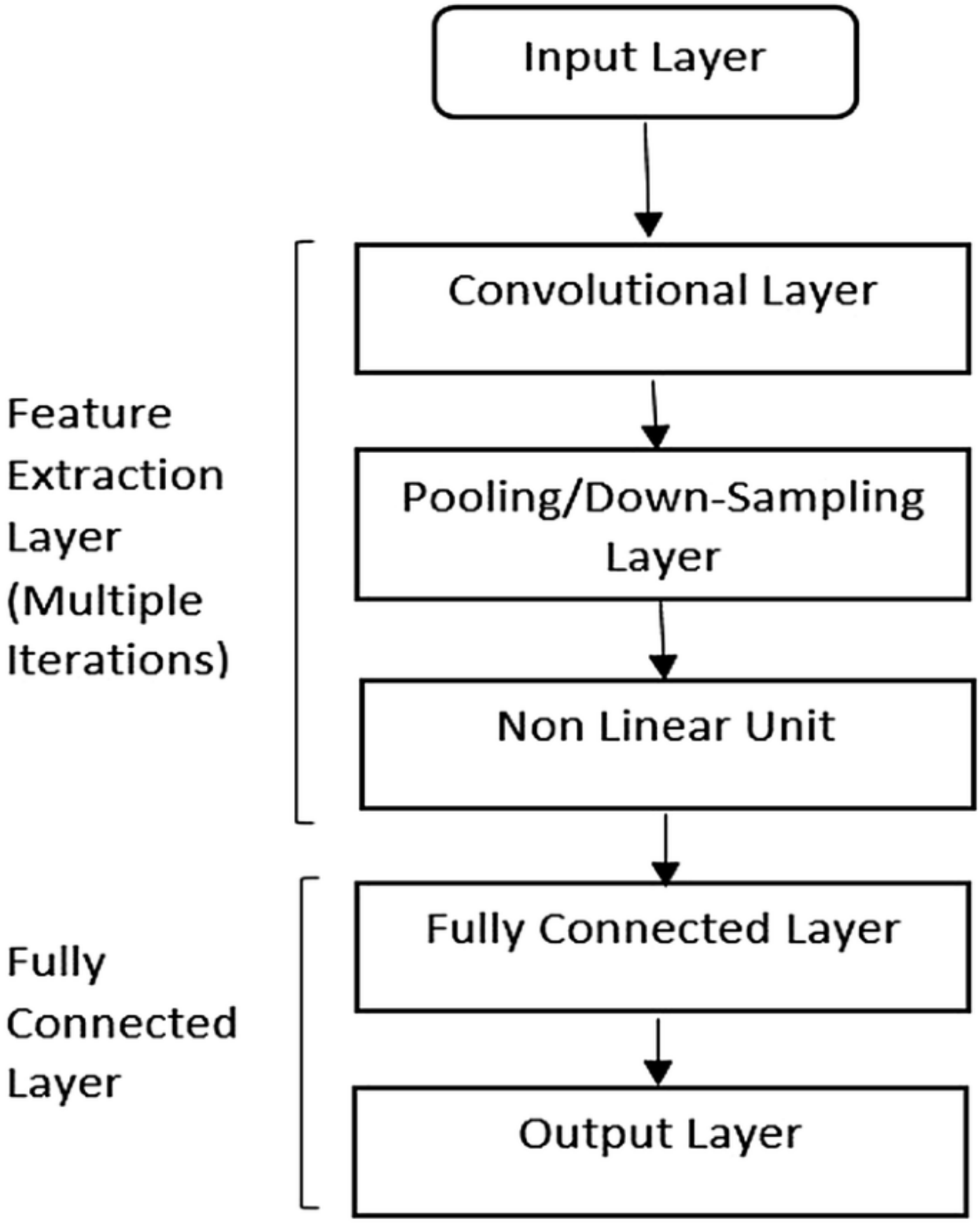
Visualization of HOG features on the same image using skimage library. (Image by author)

CVI TOOLS

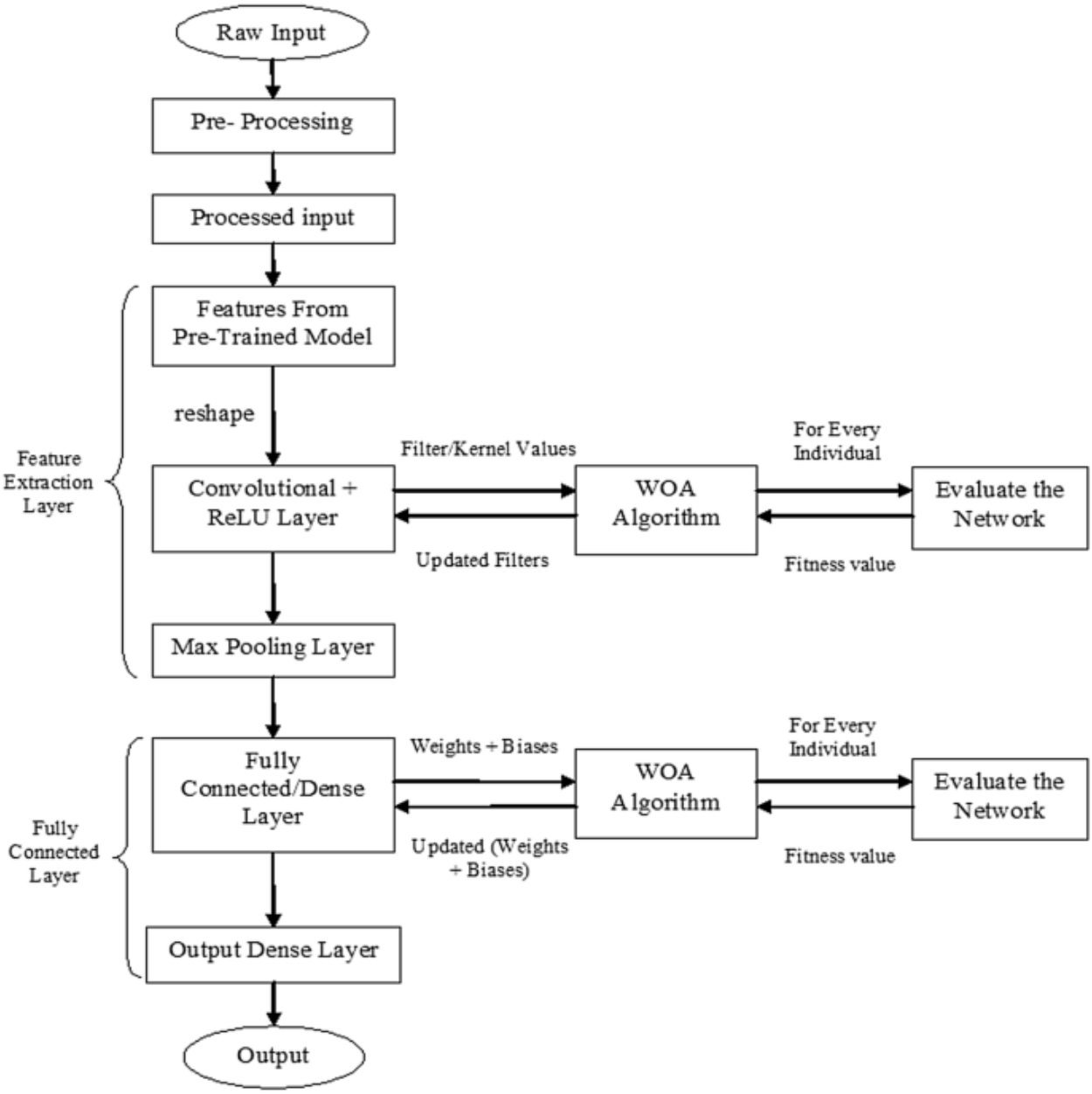







Input Layer
Convolutional Layer
Pooling/Down-Sampling Layer
Non Linear Unit
Feature Extraction Layer (Multiple Iterations)
Fully Connected Layer
Fully Connected Layer
Output Layer

NRCM
your roots to success...


Raw Input
Pre- Processing
Processed input
Features From Pre-Trained Model
reshape
Feature Extraction Layer
Convolutional + ReLU Layer
Filter/Kernel Values
Updated Filters
WOA Algorithm
For Every Individual
Fitness value
Evaluate the Network
Max Pooling Layer
Fully Connected/Dense Layer
Weights + Biases
Updated (Weights + Biases)
WOA Algorithm
For Every Individual
Fitness value
Evaluate the Network
Fully Connected Layer
Output Dense Layer
Output

UNIT-V

Introduction to Dimensionality Reduction Technique

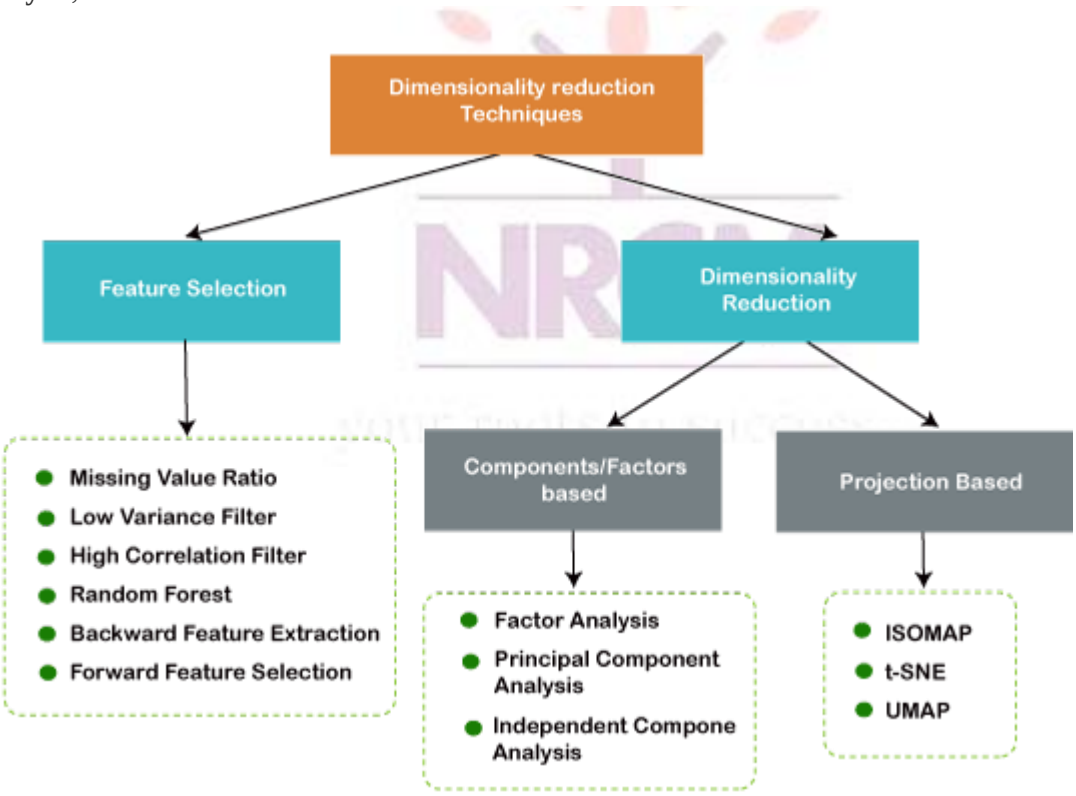
What is Dimensionality Reduction?

The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.

A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."*** These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.



The Curse of Dimensionality

Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Benefits of applying Dimensionality Reduction

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- **I**t removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of dimensionality Reduction

There are also some disadvantages of applying the dimensionality reduction, which are given below:

- Some data may be lost due to dimensionality reduction.
- In the PCA dimensionality reduction technique, sometimes the principal components required to consider are unknown.

Approaches of Dimension Reduction

There are two ways to apply the dimension reduction technique, which are given below:

Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation**
- **Chi-Square Test**

- **ANOVA**
- **Information Gain, etc.**

2. Wrappers Methods

The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection
- Bi-directional Elimination

3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

a. Principal Component Analysis
b. Linear Discriminant Analysis
c. Kernel PCA
d. Quadratic Discriminant Analysis

Common techniques of Dimensionality Reduction

a. **Principal Component Analysis**
b. **Backward Elimination**
c. **Forward Selection**
d. **Score comparison**
e. **Missing Value Ratio**
f. **Low Variance Filter**

g. **High Correlation Filter**
h. **Random Forest**
i. **Factor Analysis**
j. **Auto-Encoder**

Principal Component Analysis (PCA)

Principal Component Analysis is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***

Backward Feature Elimination

The backward feature elimination technique is mainly used while developing Linear Regression or Logistic Regression model. Below steps are performed in this technique to reduce the dimensionality or in feature selection:

- In this technique, firstly, all the n variables of the given dataset are taken to train the model.
- The performance of the model is checked.
- Now we will remove one feature each time and train the model on n-1 features for n times, and will compute the performance of the model.
- We will check the variable that has made the smallest or no change in the performance of the model, and then we will drop that variable or features; after that, we will be left with n-1 features.
- Repeat the complete process until no feature can be dropped.

In this technique, by selecting the optimum performance of the model and maximum tolerable error rate, we can define the optimal number of features require for the machine learning algorithms.

Forward Feature Selection

Forward feature selection follows the inverse process of the backward elimination process. It means, in this technique, we don't eliminate the feature; instead, we will find the best features that can produce the highest increase in the performance of the model. Below steps are performed in this technique:

- We start with a single feature only, and progressively we will add each feature at a time.
- Here we will train the model on each feature separately.
- The feature with the best performance is selected.
- The process will be repeated until we get a significant increase in the performance of the model.

Missing Value Ratio

If a dataset has too many missing values, then we drop those variables as they do not carry much useful information. To perform this, we can set a threshold level, and if a variable has missing values more than that threshold, we will drop that variable. The higher the threshold value, the more efficient the reduction.

Low Variance Filter

As same as missing value ratio technique, data columns with some changes in the data have less information. Therefore, we need to calculate the variance of each variable, and all data columns with variance lower than a given threshold are dropped because low variance features will not affect the target variable.

High Correlation Filter

High Correlation refers to the case when two variables carry approximately similar information. Due to this factor, the performance of the model can be degraded. This correlation between the independent numerical variable gives the calculated value of the correlation coefficient. If this value is higher than the threshold value, we can remove one of the variables from the dataset. We can consider those variables or features that show a high correlation with the target variable.

Random Forest

Random Forest is a popular and very useful feature selection algorithm in machine learning. This algorithm contains an in-built feature importance package, so we do not need to program it separately. In this technique, we need to generate a large set of trees against the target variable, and with the help of usage statistics of each attribute, we need to find the subset of features.

Random forest algorithm takes only numerical variables, so we need to convert the input data into numeric data using **hot encoding**.

Factor Analysis

Factor analysis is a technique in which each variable is kept within a group according to the correlation with other variables, it means variables within a group can have a high correlation between themselves, but they have a low correlation with variables of other groups.

We can understand it by an example, such as if we have two variables Income and spend. These two variables have a high correlation, which means people with high income spends more, and vice versa. So, such variables are put into a group, and that group is known as the **factor**. The number of these factors will be reduced as compared to the original dimension of the dataset.

Auto-encoders

One of the popular methods of dimensionality reduction is auto-encoder, which is a type of ANN or artificial neural network, and its main aim is to copy the inputs to their outputs. In this, the input is compressed into latent-space representation, and output is occurred using this representation. It has mainly two parts:

- **Encoder:** The function of the encoder is to compress the input to form the latent-space representation.
- **Decoder:** The function of the decoder is to recreate the output from the latent-space representation.

Linear Discriminant Analysis (LDA) in Machine Learning

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).

This can be used to project the features of higher dimensional space into lower-dimensional space in order to reduce resources and dimensional costs. In this topic, "**Linear Discriminant Analysis (LDA) in machine learning"**, we will discuss the LDA algorithm for classification predictive modeling problems, limitation of logistic regression, representation of linear Discriminant analysis model, how to make a prediction using LDA, how to prepare data for LDA, extensions to LDA and much more. So, let's start with a quick introduction to Linear Discriminant Analysis (LDA) in machine learning.

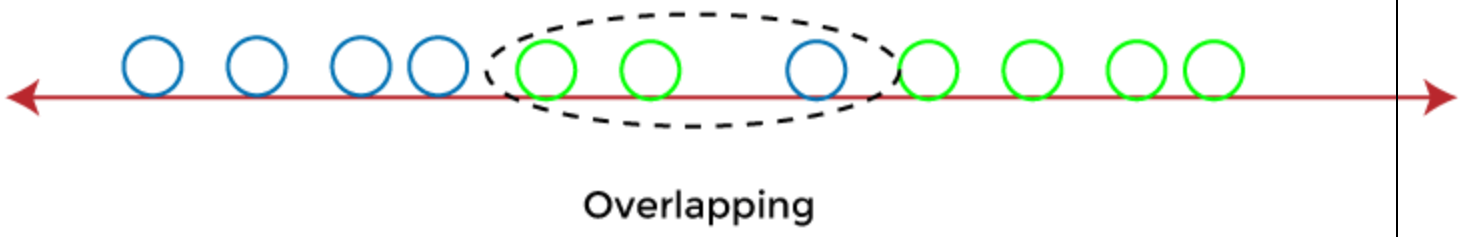
Note: Before starting this topic, it is recommended to learn the basics of Logistic Regression algorithms and a basic understanding of classification problems in machine learning as a prerequisite

What is Linear Discriminant Analysis (LDA)?

Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.

Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification

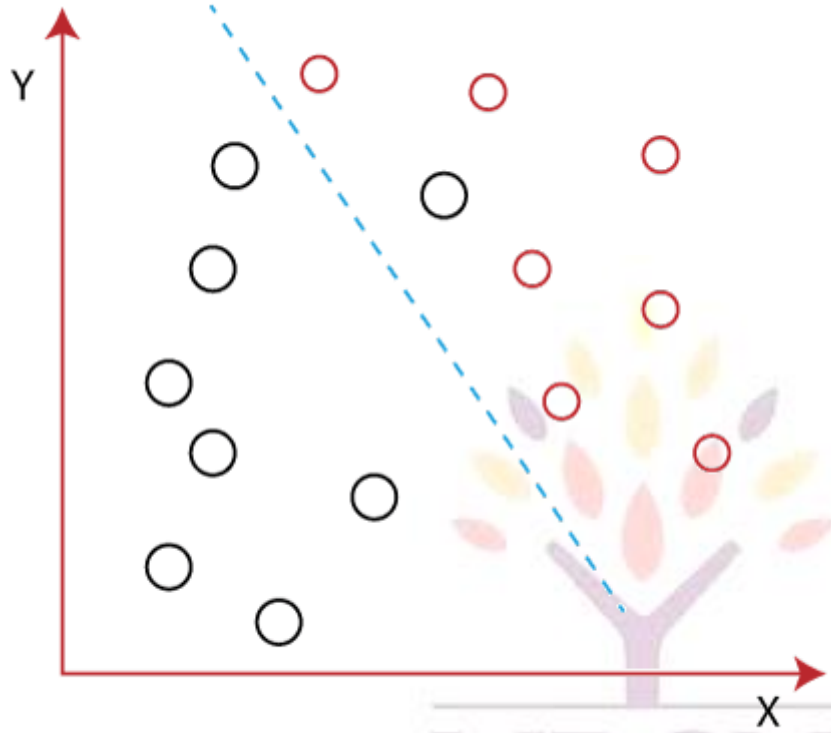
Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.



To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

Example:

Let's assume we have to classify two different classes having two sets of data points in a 2-dimensional plane as shown below image:



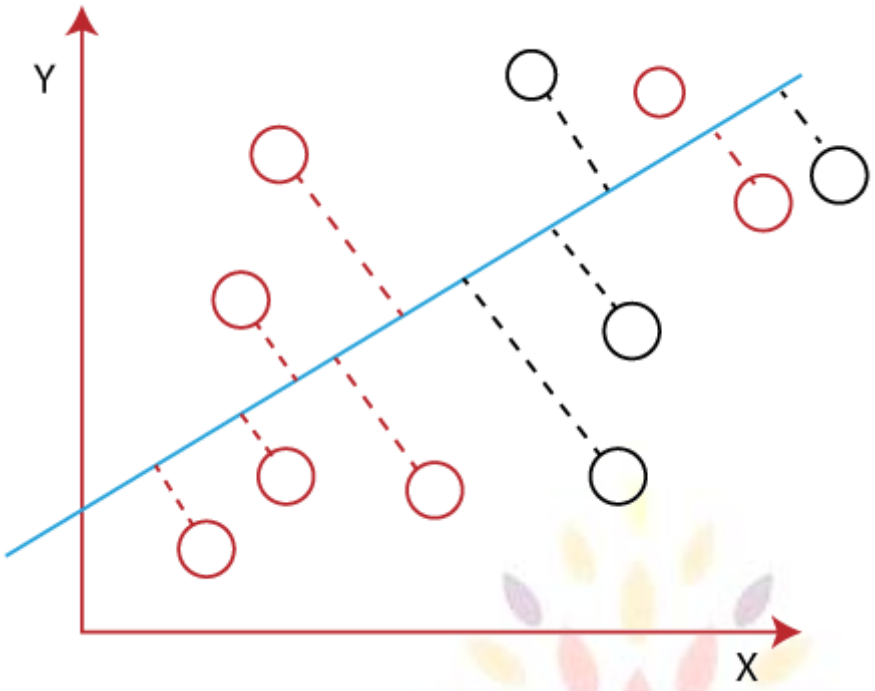
However, it is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis; we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

How Linear Discriminant Analysis (LDA) works?

Linear Discriminant analysis is used as a dimensionality reduction technique in machine learning, using which we can easily transform a 2-D and 3-D graph into a 1-dimensional plane.

Let's consider an example where we have two classes in a 2-D plane having an X-Y axis, and we need to classify them efficiently. As we have already seen in the above example that LDA enables us to draw a straight line that can completely separate the two classes of the data points. Here, LDA uses an X-Y axis to create a new axis by separating them using a straight line and projecting data onto a new axis.

Hence, we can maximize the separation between these classes and reduce the 2-D plane into 1-D.



To create a new axis, Linear Discriminant Analysis uses the following criteria:

- It maximizes the distance between means of two classes.
- It minimizes the variance within the individual class.

Using the above two conditions, LDA generates a new axis in such a way that it can maximize the distance between the means of the two classes and minimizes the variation within each class.

In other words, we can say that the new axis will increase the separation between the data points of the two classes and plot them onto the new axis.

Why LDA?

- Logistic Regression is one of the most popular classification algorithms that perform well for binary classification but falls short in the case of multiple classification problems with well-separated classes. At the same time, LDA handles these quite efficiently.
- LDA can also be used in data pre-processing to reduce the number of features, just as PCA, which reduces the computing cost significantly.
- LDA is also used in face detection algorithms. In Fisherfaces, LDA is used to extract useful data from different faces. Coupled with eigenfaces, it produces effective results.

Drawbacks of Linear Discriminant Analysis (LDA)

Although, LDA is specifically used to solve supervised classification problems for two or more classes which are not possible using logistic regression in machine learning. But LDA also fails in some cases where the Mean of the distributions is shared. In this case, LDA fails to create a new axis that makes both the classes linearly separable.

To overcome such problems, we use **non-linear Discriminant analysis** in machine learning.

Extension to Linear Discriminant Analysis (LDA)

Linear Discriminant analysis is one of the most simple and effective methods to solve classification problems in machine learning. It has so many extensions and variations as follows:

1. **Quadratic Discriminant Analysis (QDA):** For multiple input variables, each class deploys its own estimate of variance.
2. **Flexible Discriminant Analysis (FDA):** it is used when there are non-linear groups of inputs are used, such as splines.
3. **Flexible Discriminant Analysis (FDA):** This uses regularization in the estimate of the variance (actually covariance) and hence moderates the influence of different variables on LDA.

Real-world Applications of LDA

Some of the common real-world applications of Linear discriminant Analysis are given below:

- **FaceRecognition**
  Face recognition is the popular application of computer vision, where each face is represented as the combination of a number of pixel values. In this case, LDA is used to minimize the number of features to a manageable number before going through the classification process. It generates a new template in which each dimension consists of a linear combination of pixel values. If a linear combination is generated using Fisher's linear discriminant, then it is called Fisher's face.
- **Medical**
  In the medical field, LDA has a great application in classifying the patient disease on the basis of various parameters of patient health and the medical treatment which is going on. On such parameters, it classifies disease as mild, moderate, or severe. This classification helps the doctors in either increasing or decreasing the pace of the treatment.
- **CustomerIdentification**
  In customer identification, LDA is currently being applied. It means with the help of LDA; we can easily identify and select the features that can specify the group of customers who are likely to purchase a specific product in a shopping mall. This can be helpful when we want to identify a group of customers who mostly purchase a product in a shopping mall.
- **ForPredictions**
  LDA can also be used for making predictions and so in decision making. For example, "will you buy this product" will give a predicted result of either one or two possible classes as a buying or not.
- **InLearning**
  Nowadays, robots are being trained for learning and talking to simulate human work, and it can also

be considered a classification problem. In this case, LDA builds similar groups on the basis of different parameters, including pitches, frequencies, sound, tunes, etc.

Difference between Linear Discriminant Analysis and PCA

Below are some basic differences between LDA and PCA:

- PCA is an unsupervised algorithm that does not care about classes and labels and only aims to find the principal components to maximize the variance in the given dataset. At the same time, LDA is a supervised algorithm that aims to find the linear discriminants to represent the axes that maximize separation between different classes of data.
- LDA is much more suitable for multi-class classification tasks compared to PCA. However, PCA is assumed to be an as good performer for a comparatively small sample size.
- Both LDA and PCA are used as dimensionality reduction techniques, where PCA is first followed by LDA.

How to Prepare Data for LDA

Below are some suggestions that one should always consider while preparing the data to build the LDA model:

- **Classification Problems:** LDA is mainly applied for classification problems to classify the categorical output variable. It is suitable for both binary and multi-class classification problems.
- **Gaussian Distribution:** The standard LDA model applies the Gaussian Distribution of the input variables. One should review the univariate distribution of each attribute and transform them into more Gaussian-looking distributions. For e.g., use log and root for exponential distributions and Box-Cox for skewed distributions.
- **Remove Outliers:** It is good to firstly remove the outliers from your data because these outliers can skew the basic statistics used to separate classes in LDA, such as the mean and the standard deviation.
- **Same Variance:** As LDA always assumes that all the input variables have the same variance, hence it is always a better way to firstly standardize the data before implementing an LDA model. By this, the Mean will be 0, and it will have a standard deviation of 1.

Difference between Parametric and Non-Parametric Methods

Parametric Methods: The basic idea behind the parametric method is that there is a set of fixed parameters that uses to determine a probability model that is used in Machine Learning as well. Parametric methods are those methods for which we priory knows that the population is normal, or if not then we can easily approximate it using a normal distribution which is possible by invoking the Central Limit Theorem. Parameters for using the normal distribution is as follows:

- Mean
- Standard Deviation

Eventually, the classification of a method to be parametric is completely depends on the presumptions that are made about a population. There are many parametric methods available some of them are:

- Confidence interval used for – population mean along with known standard deviation.
- The confidence interval is used for – population means along with the unknown standard deviation.
- The confidence interval for population variance.
- The confidence interval for the difference of two means, with unknown standard deviation.

Nonparametric Methods: The basic idea behind the parametric method is no need to make any assumption of parameters for the given population or the population we are studying. In fact, the methods don't depend on the population. Here there is no fixed set of parameters are available, and also there is no distribution (normal distribution, etc.) of any kind is available for use. This is also the reason that nonparametric methods are also referred to as distribution-free methods. Nowadays Non-parametric methods are gaining popularity and an impact of influence some reasons behind this fame is:

- The main reason is that there is no need to be mannered while using parametric methods.
- The second important reason is that we do not need to make more and more assumptions about the population given (or taken) on which we are working on.
- Most of the nonparametric methods available are very easy to apply and to understand also i.e. the complexity is very low.

There are many nonparametric methods are available today but some of them are as follows:

- Spearman correlation test
- Sign test for population means
- U-test for two independent means

Difference between Parametric and Non-Parametric Methods are as follows:

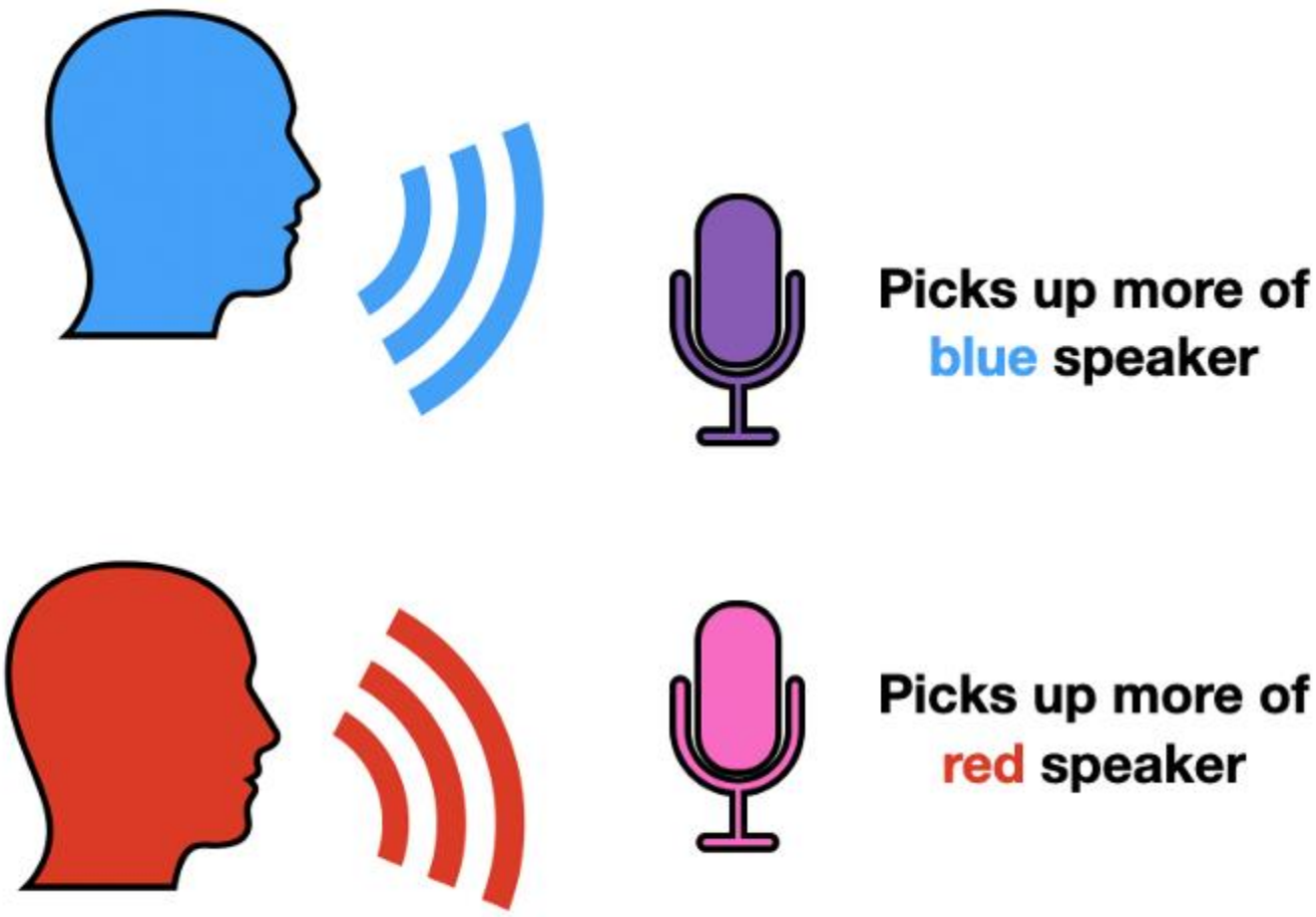
Parametric Methods	Non-Parametric Methods
Parametric Methods uses a fixed number of parameters to build the model.	Non-Parametric Methods use the flexible number of parameters to build the model.
Parametric analysis is to test group means.	A non-parametric analysis is to test medians.
It is applicable only for variables.	It is applicable for both – Variable and Attribute.
It always considers strong assumptions about data.	It generally fewer assumptions about data.
Parametric Methods require lesser data than Non-Parametric Methods.	Non-Parametric Methods requires much more data than Parametric Methods.
Parametric methods assumed to be a normal distribution.	There is no assumed distribution in non-parametric methods.
Parametric data handles – Intervals data or ratio data.	But non-parametric methods handle original data.
Here when we use parametric methods then the result or outputs generated can be easily affected by outliers.	When we use non-parametric methods then the result or outputs generated cannot be seriously affected by outliers.
Parametric Methods can perform well in many	Similarly, Non-Parametric Methods can perform well

Parametric Methods	Non-Parametric Methods
situations but its performance is at peak (top) when the spread of each group is different.	in many situations but its performance is at peak (top) when the spread of each group is the same.
Parametric methods have more statistical power than Non-Parametric methods.	Non-parametric methods have less statistical power than Parametric methods.
As far as the computation is considered these methods are computationally faster than the Non-Parametric methods.	As far as the computation is considered these methods are computationally slower than the Parametric methods.
Examples: Logistic Regression, Naïve Bayes Model, etc.	Examples: KNN, Decision Tree Model, etc.

ndependent Component Analysis (ICA)

Finding hidden factors in data

This is the final post in a two-part series on **Principal Component Analysis (PCA)** and **Independent Component Analysis (ICA)**. Although the techniques are similar, they are in fact different approaches and perform different tasks. In this post, I will provide a high level introduction to ICA, compare it to PCA, and give an example of using ICA to remove blink artifacts from EEG data.



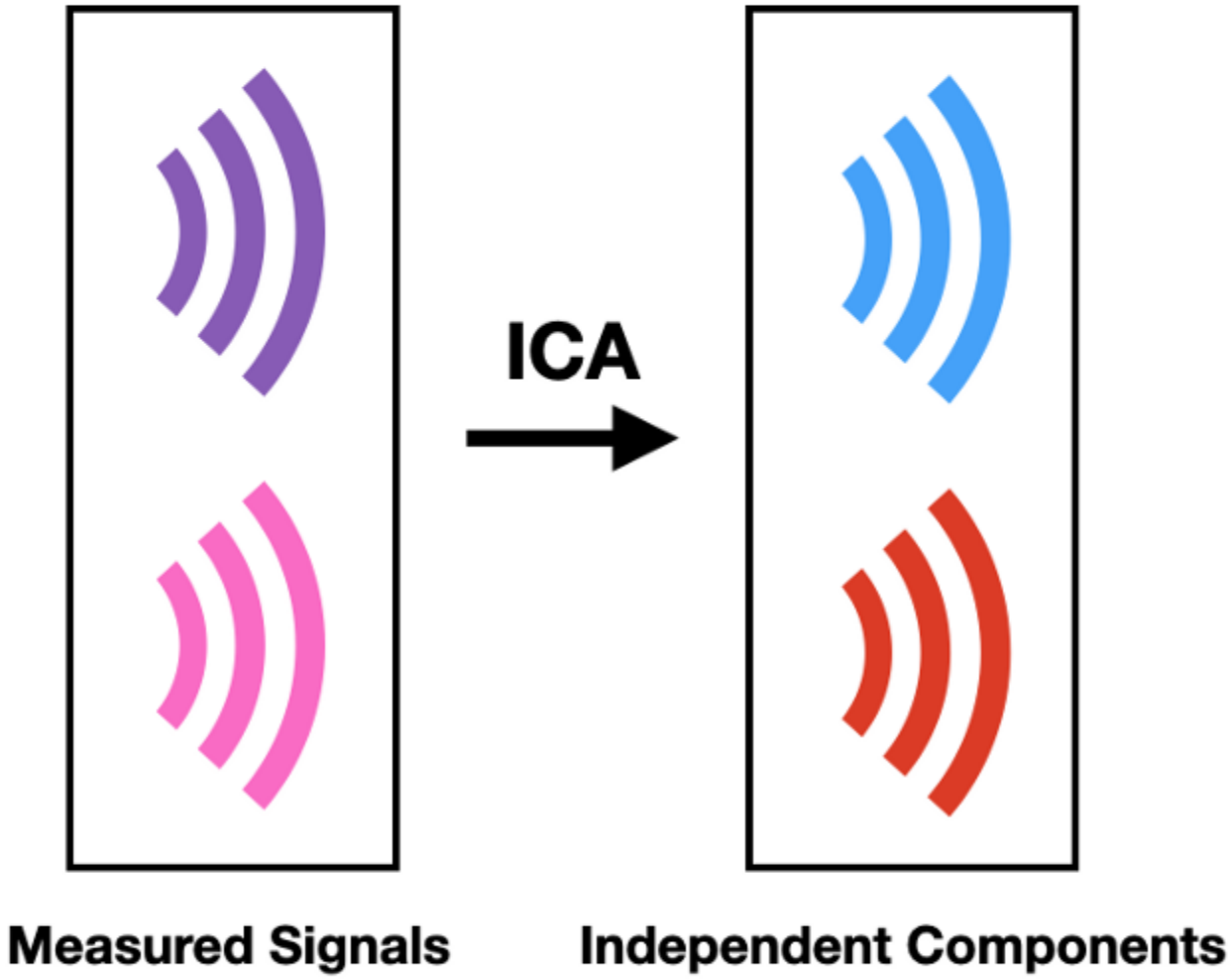
The simplest version of the "Cocktail Party Problem". Image by author.

ICA

The standard problem used to describe ICA is the "Cocktail Party Problem". In its simplest form, imagine two people having a conversation at a cocktail party (like the red and blue speakers above). For whatever reason, you have two microphones placed near both party-goers (like the purple and pink microphones above). Both voices are heard by *both* microphones at different volumes based on the distance between the person and the microphone. In other words, we record two files that include audio from the two party-goers mixed together. The problem then is, *how can we separate the two voices in each file to obtain isolated recordings of each speaker?*

This problem is solved easily with I**ndependent Component Analysis (ICA)** which **transforms a set of vectors into a maximally independent set**. Returning to our "Cocktail Party Problem", ICA will convert the

two mixed audio recordings (represented by purple and pink waveforms below) into two unmixed recordings of each individual speaker (represented by blue and red waveforms below). Notice, that the **number of inputs and outputs are the same**, and since the outputs are mutually independent there is no obvious way to drop components like in **Principal Component Analysis (PCA)**.



Converting mixed signals to independent components using ICA. Image by author.

How it works

There are **two key assumptions** made in ICA. The hidden independent components we are trying to uncover must be one, **statistically independent**, and two, **non-Gaussian**. Semantically, by independent I mean information about **x** does not give you information about **y** and vice versa. Mathematically, this translates to,

$$p(x, y) = p(x)p(y)$$

Mathematical definition of statistical independence. Image by author.

Where *p(x)* represents the probability distribution of x. *p(x,y)* represents the joint distribution of x and y. The non-Gaussian assumption simply means the independent components have distributions that are not Gaussian, meaning it doesn't look like a bell curve.

Non-Gaussianity is a key assumption for ICA. Image by author.

The first assumption is the starting point of ICA. We want to disentangle information to derive a set of independent factors. If there are not multiple independent generators of information to uncover, there really isn't a need for ICA. For example, imagine using ICA for the "Cocktail Party Problem", but with only one partygoer. What one could call the COVID birthday party problem. It wouldn't make much sense.

The need for the second assumption lies in the mathematics. ICA uses the idea of **non-Gaussianity** to uncover independent components. Non-Gaussianity **quantifies how *far* the distribution of a random variable is from being Gaussian**. Example measures of non-Gaussianity are kurtosis and negentropy. Why such a measure is helpful follows from the **Central Limit Theorem**. Specifically, a result that states, the sum

of two independent random variables has a distribution that is *closer* to Gaussian than either of the original variables. ICA combines this idea, non-Gaussianity measures, and the non-Gaussian assumption to uncover independent components hidden in data.

To illustrate this, consider a dataset with two variables **x_1** and **x_2**. These variables serve as a basis that defines a space i.e. we can use them to plot points in 2 dimensions. Suppose, we know the two independent components underlying the data, **s_1,** and **s_2**. These two components serve as an alternative basis to describe the same space. Therefore, any point **y** in this space could be written as both a linear combination of variables **x_1** and **x_2** or components **s_1** and **s_2**.

$$\mathbf{y} = w_1\mathbf{x}_1 + w_2\mathbf{x}_2$$

Linear combination of measured signals i.e. input variables. Image by author.

$$\mathbf{y} = a_1\mathbf{s}_1 + a_2\mathbf{s}_2$$

Linear combination of independent components. Image by author.

Going back to the Central Limit Theorem, the distribution of the sum of two random variables will be *more Gaussian* than either individual variable. Thus, when a_1 and a_2 are both non-zero, the distribution of **y** will be *more Gaussian* than either **s_1** or **s_2**. The reverse of that is, if either a_1 or a_2 is zero, then the distribution of **y** will be *less Gaussian* than the former case. And, if the non-Gaussian assumption of **s_1** and **s_2** holds, it will not be Gaussian at all since **y** will be exactly equal to one of the independent components!

In other words, the non-Gaussianity of **y** is maximized when it is directly proportional to one of the independent components. This allows us to frame ICA as an optimization problem. For example,

$$\max \quad kurt(w_1\mathbf{x}_1 + w_2\mathbf{x}_2)$$

Framing ICA as an optimization problem for a single independent component. Image by author.

Where we want to find the values of w_1 and w_2 that maximize the kurtosis of a linear combination of our known input variables. These optimal values of w_1 and w_2 will define an independent component.

$$\mathbf{s} = w_1^* \mathbf{x}_1 + w_2^* \mathbf{x}_2$$

Solutions to ICA optimization problem define independent components.

More generally, we can solve for the matrix of weights, **W**, which maximizes the non-Gaussianity of the matrix multiplication of **W** and a data matrix, **X**.

$$\max \quad kurt(\mathbf{WX})$$

Framing ICA as an optimization problem for multiple independent components. Image by author.

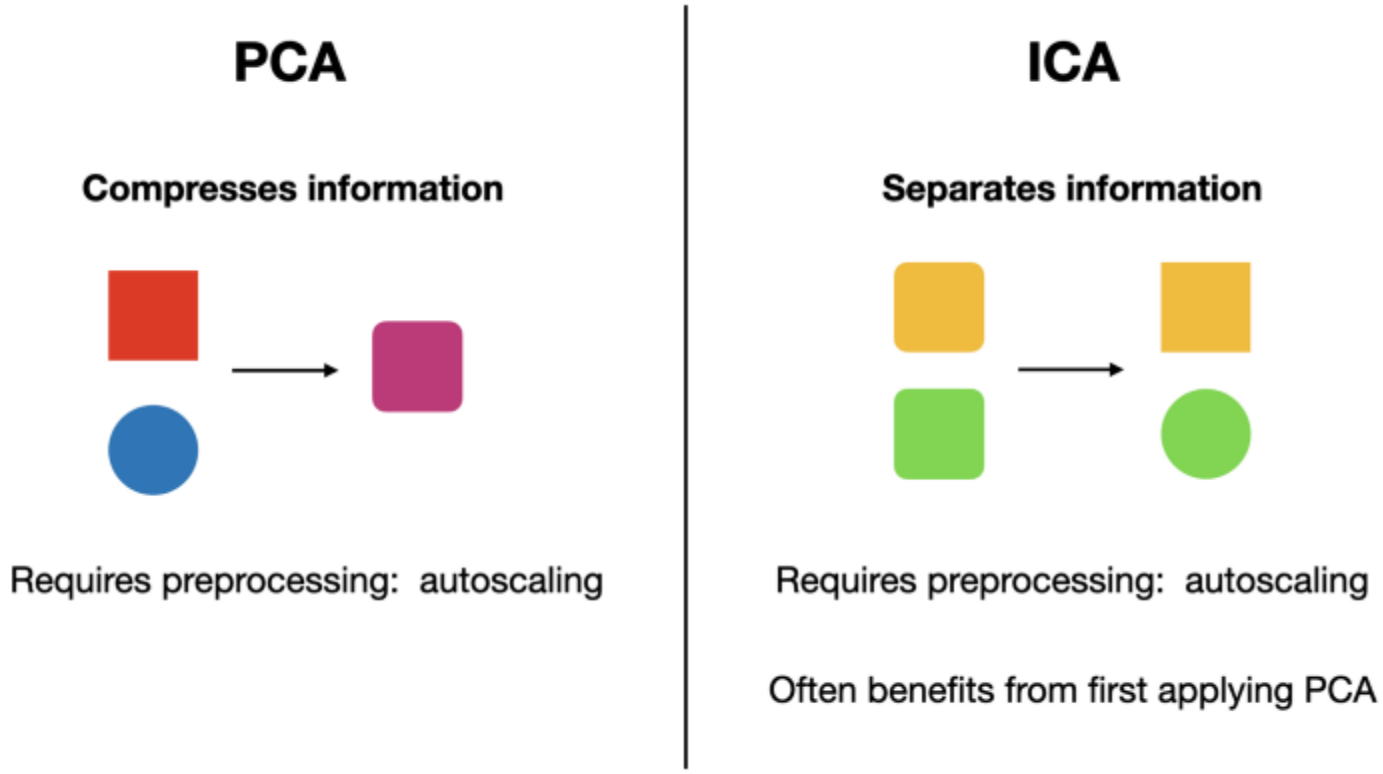
Key Points

I may have once again gone too far into the mathematical weeds. As a takeaway I will just highlight three key points of ICA:

- The number of inputs equals the number of outputs
- Assumes independent components are statistically independent
- Assumes independent components are non-Gaussian

PCA vs ICA

Before moving on to an example I will briefly compare PCA and ICA. Although the two approaches may seem related, they perform different tasks. Specifically, **PCA** is often used to **compress information** i.e. dimensionality reduction. While **ICA** aims to **separate information** by transforming the input space into a maximally independent basis. A commonality is both approaches require input data to be **autoscaled** i.e. **subtract each column by its mean and divide by its standard deviation**. This is one reason why PCA is usually a good thing to do before performing ICA.



Comparison of PCA and ICA. Image by author.

Principal Component Analysis (PCA)
Intuition, math, and stonks
towardsdatascience.com

Example: Blink Removal from EEG

As always I will close with a concrete practical example. In this example I will use ICA to remove blink artifacts from EEG data, code is available in the **GitHub repository**.

Electroencephalography (EEG) is a technique that measures electrical activity resulting from the brain. A major disadvantage of EEG is its sensitivity to motion and other non-brain artifacts. One such artifact occurs whenever participants blink. In the below figure, blink artifacts can plainly be seen via spikes in the voltage vs time plot of the Fp1 electrode (near the front of the head).