Unit 1

Computer vision is a subfield of artificial intelligence that deals with acquiring, processing, analysing, 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.

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.

How does Computer Vision Work?

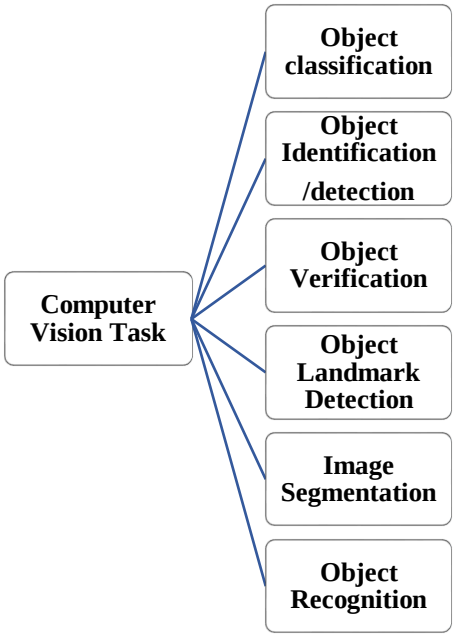
Computer vision is a technique that extracts information from visual data, such as images and videos. Although computer vision works similarly to human eyes with brain work, this is probably one of the biggest open questions for IT professionals: How does the human brain operate and solve visual object 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 labelled data is provided to machines to train it. This labelled data enables the machine to analyse 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, labelled each photo, shape, the distance between each shape, colour, 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 in computer vision categorizes visual content, like images or videos, by identifying and labelling specific objects, enabling accurate prediction of an object's class within an image.

- **Object Identification/detection:** Object identification employs image classification to locate and label objects in images or videos. This technique enables counting and precise localization of various objects within a scene, such as identifying a dog, cat, and duck in an image.

- **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.

Applications of Computer Vision:

Below are some most popular applications of computer vision:

1. **Facial Recognition:** Computer vision allows machines to identify faces by analysing facial features, aiding in identity verification. This technology is utilized on social media platforms for user tagging, while government agencies use it in video feeds to identify individuals, including potential criminals.

2. **Healthcare and Medicine:** Computer vision revolutionizes cancer treatment by offering faster, more precise tumor evaluations. It enables quicker identification of patients requiring urgent surgery, improving survival rates.

3. **Self-driving Vehicles:** Computer vision powers self-driving cars by analysing multi-angle video feeds to detect vehicles, interpret traffic signals, identify pedestrians, and navigate safely to destinations.

4. **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.

5. **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.

Image Representation:

Image representation in computer vision transforms images into computer-readable formats by converting them into numerical or symbolic data. Typically composed of pixels, where each pixel holds colour or intensity values, this process aims to extract key features and information. It enables computers to perform tasks like object recognition, image classification, and segmentation.

Techniques in Image Representation:

1. **Grayscale Representation:** Uses a single channel with grayscale values (0 to 255) per pixel. Ideal for tasks not reliant on colour information.

2. **Colour Representation (RGB):** Utilizes three channels (Red, Green, Blue) per pixel, with values ranging from 0 to 255 for each channel.

3. **Feature Extraction:** Algorithms extract edges, corners, textures, or complex features like HOG and CNN-based deep features rather than raw pixel values.

4. **Histograms:** Represent the frequency distribution of pixel intensities, offering insights into contrast, brightness, and image content.

5. **Local Descriptors:** Descriptions of specific image regions (e.g., SIFT, SURF) used for object recognition and matching.

6. **Deep Learning-based Representations (CNNs):** CNNs learn hierarchical representations from images, capturing intricate patterns and features effectively.

Need of Image Representation:

- **Efficient Storage and Processing:** Image representation condenses high-dimensional pixel data, making it easier to store and process by reducing memory requirements and computational complexity.

- **Feature Extraction:** These methods extract essential image features like edges, textures, shapes, and colours, crucial for recognizing objects and scenes.

- **Dimensionality Reduction:** Image representation reduces feature numbers while retaining critical information, simplifying data for analysis and visualization.

- **Invariance and Robustness:** Some techniques remain unaffected by transformations like rotation or scaling, enhancing algorithm robustness in recognizing objects under varying conditions.

- **Machine Learning Compatibility:** Representations as feature vectors enable machine learning algorithms for tasks like classification and object detection, facilitating diverse algorithm applications.

- **Generalization:** Effective representations enhance a model's capability to recognize similar patterns in new data, improving performance across diverse images through learned features.

Image Processing In Computer Vision:

What Is Image Processing In Computer Vision?

Image processing in computer vision involves using various techniques and algorithms to manipulate, enhance, analyse, and extract information from images. It encompasses a wide range of methods aimed at improving image quality, detecting patterns, recognizing objects, and preparing images for further analysis or interpretation by machines.

What Is Image Processing?

Image processing refers to the manipulation and analysis of images using various techniques and algorithms. It involves modifying or enhancing images to extract useful information, improve quality, recognize patterns, and perform tasks such as object detection, image restoration, and feature extraction. Image processing techniques can range from basic operations like resizing and filtering to complex tasks involving machine learning and computer vision algorithms.

Where Is Image Processing Used?

- **Medical Imaging:** Utilized for diagnosing illnesses and planning treatments, employing techniques like MRI scans, X-rays, and CT scans. Image processing aids in enhancing images for better clarity and assists in identifying anomalies or abnormalities in tissues or organs.

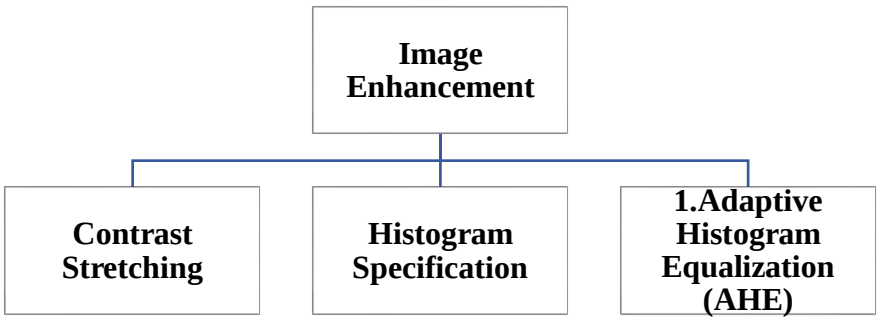
- **Remote Sensing:** Involves analysing satellite and aerial images for various purposes such as monitoring environmental changes, assessing land use, predicting weather patterns, and aiding in disaster response and recovery efforts.

- **Entertainment:** Image processing contributes to creating captivating visual effects in movies, television, and gaming. It encompasses tasks like colour correction, compositing, and CGI, enhancing the overall visual appeal of entertainment media.

- **Manufacturing and Quality Control:** Vital for inspecting and ensuring the quality of products during manufacturing processes. Image processing techniques detect defects, measure dimensions, and validate product integrity, ensuring adherence to quality standards and minimizing production flaws.

Image Enhancement:

Image enhancement is a set of techniques used to improve the visual quality of a digital image for better interpretation and analysis by humans or computer algorithms. The goal of image enhancement is to emphasize certain features or details in the image, reduce noise, increase contrast, or otherwise make the image more visually appealing or informative. Image enhancement is widely used in various fields, including computer vision, medical imaging, remote sensing, and photography. Here are some key aspects and methods of image enhancement:



1. Contrast Stretching:

Contrast stretching, or enhancement, boosts image contrast and visibility by widening the pixel value range to cover the entire dynamic range. This enhances visual appeal and accentuates details within different image regions. Here's how contrast stretching works:

a. Understanding Dynamic Range:

- In digital images, each pixel has a specific intensity value, often represented as a grayscale value ranging from 0 (black) to 255 (white) for 8-bit images (the most common format).
- The dynamic range of an image represents the full range of intensity values it can contain. For an 8-bit image, this is from 0 to 255.

b. Stretching the Dynamic Range:

- Contrast stretching involves mapping the original pixel values of an image to new values within the full dynamic range, typically from 0 to 255.
- The process of stretching is often linear, meaning that each pixel value in the original range is mapped to a new value in the full range in a proportional manner.

c. Formula:

- The formula for linear contrast stretching is as follows:

$$\text{NewPixelValue} = \frac{(\text{OldPixelValue} - \text{minPixelValue}) \times (\text{NewMax} - \text{NewMin})}{\text{maxPixelValue} - \text{minPixelValue}} + \text{NewMin}$$

- **OldPixelValue** is the original intensity value of a pixel.
- **minPixelValue** and **maxPixelValue** are the minimum and maximum intensity values in the original image, respectively.
- **NewMin** and **NewMax** are the desired minimum and maximum intensity values for the stretched image, usually 0 and 255 in an 8-bit image.

d. Benefits:

- **Increases visibility:** By expanding the range of pixel values, contrast stretching makes subtle details and features more visible.
- **Enhances image quality:** It improves the overall appearance of an image, making it more visually appealing.
- **Supports subsequent image processing:** Enhanced images are often easier to work with for tasks like object detection, recognition, or segmentation in computer vision.

e. Limitations:

- Contrast stretching assumes that the full dynamic range is not utilized in the original image. If the image already contains a wide range of pixel values, contrast stretching may not yield significant improvements.
- It is essential to choose appropriate values for **NewMin** and **NewMax** to avoid over-enhancement or under-enhancement, which can lead to image artifacts or loss of information.
- Extreme values or outliers in the original image can significantly affect the stretching process, so it's crucial to account for such cases.

f. Applications:

- Contrast stretching is used in various applications, including:
- Improving the visibility of medical images (X-rays, MRIs, etc.) to assist in diagnosis.
- Enhancing satellite or aerial imagery for land cover classification or object detection.
- Enhancing photographs for better visual quality.
- Preprocessing images in computer vision tasks to make objects or features more distinguishable.

2. Histogram Specification:

Histogram specification in computer vision adjusts image contrast and brightness by aligning the image histogram to a desired target histogram. It's an advanced form of histogram equalization that not only enhances contrast but also matches the image histogram to a specific desired shape or distribution. It's valuable for imposing specific intensity characteristics on an image.

Here's how histogram specification works:

a. Original Image Histogram:

- The first step is to compute the histogram of the original image. The histogram represents the distribution of pixel intensities in the image, showing how many pixels have each intensity value.

b. Desired Histogram:

- In histogram specification, you set a target histogram representing the desired intensity distribution for the processed image. Typically defined as a probability distribution or cumulative distribution function (CDF).

c. Transformation Function:

- Creating a transformation function aims to map original pixel intensities to new values, aligning the image histogram with a desired one. This involves matching cumulative distribution functions of the original image histogram and the desired histogram.

d. Applying the Transformation:

- Each pixel in the original image is processed by applying the transformation function. This maps the original pixel values to new values based on the desired distribution.

e. Resulting Image:

- The transformed pixel values form the resulting image, which is expected to have the desired intensity distribution specified in the target histogram.

f. Limitations:

- Histogram specification's success isn't guaranteed; outcomes vary based on chosen histograms and the original image's traits..
- Extreme changes in the histogram specification can result in image artifacts or loss of information.

3. Adaptive Histogram Equalization (AHE):

Adaptive Histogram Equalization (AHE) enhances image contrast and detail visibility. Unlike traditional methods, AHE divides the image into smaller regions and processes their histograms individually. This localized approach effectively improves contrast, especially in images with varying illumination and detail levels.

Here's a detailed explanation of Adaptive Histogram Equalization:

a. Image Division into Tiles:

- AHE begins by dividing the input image into non-overlapping tiles or windows.
- These tiles can be of various sizes depending on the application and the desired level of local adaptability. Common tile sizes range from 8x8 to 128x128 pixels.

b. Histogram Equalization for Each Tile:

- For each tile, a histogram equalization operation is performed independently. Histogram equalization enhances the contrast by redistributing the pixel intensity values within the local histogram of the tile to cover the full dynamic range.
- The cumulative distribution function (CDF) of each tile's histogram is calculated and used to map pixel values.

c. Local Transformations:

- As a result of the local histogram equalization, the pixel values within each tile are stretched to cover the full intensity range. This enhances the contrast within each tile independently.

d. Interpolation:

- After equalizing the histograms of individual tiles, the transformed tiles are reassembled into the final output image. This may involve interpolation or blending at tile boundaries to ensure smooth transitions between regions.

e. Limitations and Considerations:

- AHE can create visual artifacts at tile boundaries due to contrast enhancements.
- Tile size selection is critical; too small causes noise, too large limits adaptability. CLAHE mitigates these issues by limiting contrast enhancement.

4. Wavelet-Based Enhancement:

Wavelet-based enhancement in computer vision and image processing utilizes wavelet transform methods to boost digital image quality. It decomposes images into wavelet coefficients, tweaks these coefficients for enhancement, and reconstructs the image. This method effectively improves images by adjusting details and noise across various scales.

Let's delve into wavelet-based enhancement in more detail:

a. Wavelet Transform:

- The wavelet transform dissects images into different frequency parts using localized basis functions called wavelets, allowing detailed analysis at various scales and orientations.
- It's a key tool in image processing, offering a multi-resolution view of an image's content.

b. Steps in Wavelet-Based Enhancement:

- **Decomposition:** The image breaks down into scales and orientations via wavelet transform, producing wavelet coefficients.
- **Enhancement:** Operations tweak coefficients for amplification, noise reduction, and contrast adjustment.
- **Reconstruction:** Modified coefficients rebuild the image, enhancing visual quality.

c. Types of Enhancements:

- **Noise Reduction:** Wavelet-based enhancement selectively reduces high-frequency noise while preserving key image elements.
- **Edge Enhancement:** Focused enhancement prioritizes edges and intricate details, vital in fields like medical imaging and object recognition.
- **Contrast Enhancement:** Modifying wavelet coefficients can enhance an image's overall contrast, rendering it more visually striking and informative.
- **Feature Enhancement:** Wavelet coefficient adjustments can highlight specific features like texture or shapes, emphasizing their prominence in the image.

d. Advantages:

- Wavelet-based enhancement operates across various scales, enriching both intricate details and overall features.
- It's adept at managing noise by isolating it in distinct frequency bands, allowing for its adjustment or removal.
- **Adaptability is a strength:** wavelet basis functions and parameters can be tailored to match image traits and enhancement objectives.

e. Applications:

- **Medical Imaging:** Enhances X-rays, MRIs, and CT scans for improved quality.
- **Remote Sensing:** Enhances satellite imagery for better land classification.
- **Astronomy:** Sharpens celestial images, revealing faint objects.

Image Filtering:

Image filtering in computer vision involves applying mathematical operations or convolutional processes to modify or enhance images. Filters, essentially matrices or kernels, are passed over an image to perform operations such as blurring, sharpening, edge detection, noise reduction, and more.

Purpose of Image Filtering:

- **Enhancement:** Filters can amplify certain features or characteristics in an image to make them more prominent or visually appealing.
- **Noise Reduction:** Filtering helps in reducing noise, such as random variations or disturbances, to improve image clarity.
- **Feature Detection:** Specific filters detect edges, corners, textures, or other distinctive features within an image.
- **Image Restoration:** Filters aid in restoring degraded or damaged images by removing unwanted artifacts.

Types of Filters:

1. **Linear Filters:**
   - **Smoothing Filters:** Blur the image, reducing noise or fine details. (e.g., Gaussian filter)
   - **Sharpening Filters:** Emphasize edges and details, enhancing image clarity. (e.g., Laplacian filter)
   - **Edge Detection Filters:** Detect and highlight edges or boundaries in an image. (e.g., Sobel, Prewitt filters)

2. **Non-Linear Filters:**
   - **Median Filter:** Replaces each pixel's value with the median value of its neighborhood, useful for noise reduction.
   - **Bilateral Filter:** Preserves edges while reducing noise, balancing smoothing and edge preservation.

How Image Filtering Works:

1. **Convolution Operation:**
   - **Kernel:** A small matrix used in filtering that defines the operation to be performed.
   - **Sliding Window:** The kernel moves over the image, and at each position, a mathematical operation (such as multiplication and addition) occurs between the kernel and corresponding image pixels.

2. **Process:**
   - For each pixel, the kernel is placed over its neighborhood, and the operation specified by the kernel is applied.
   - The resulting value becomes the new value for the pixel under the centre of the kernel.
   - This process is repeated for all pixels, adjusting their values based on the specified operation.

Applications:

1. **Preprocessing:** Filtering prepares images for further analysis by enhancing or denoising them.
2. **Computer Vision Tasks:** Filtering aids in feature extraction, object recognition, image segmentation, and more.
3. **Medical Imaging:** Used for noise reduction, feature enhancement, and diagnostics.
4. **Image Restoration:** Helps in reconstructing images affected by various factors like noise, blur, or low resolution.

Limitations:

1. **Parameter Tuning:** Selecting appropriate filter sizes and types is crucial.
2. **Artifacts:** Improper filtering can introduce artifacts or unwanted effects.
3. **Computational Cost:** Some filters, especially complex ones, might be computationally intensive.

Image filtering is a fundamental technique in computer vision, enabling the extraction of meaningful information from images, enhancing their quality, and facilitating subsequent analysis and understanding.

Smoothing:

Smoothing in computer vision is a technique used to reduce noise, suppress fine details, and create a more uniform appearance in images. It's often achieved through various filters or algorithms that blur or average neighbouring pixel values. Here's a detailed breakdown:

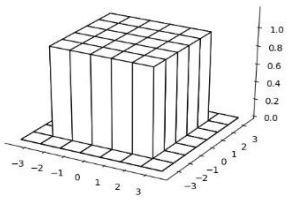
Purpose of Smoothing:

- **Noise Reduction:** Smoothing filters eliminate random variations or disturbances (noise) present in an image, improving its overall quality.
- **Detail Suppression:** They diminish small or insignificant features, reducing their impact on subsequent analysis or visual perception.
- **Preprocessing:** Smoothing prepares images for tasks like edge detection, segmentation, or feature extraction by simplifying complex structures.

Types of Smoothing Filters:

1. **Linear Filters for Smoothing:**
   - Linear smoothing filters belong to the subclass of linear filters, allowing simple application through linear operations, frequency domain analysis, and often efficient implementation.
   - These filters are crucial for levelling out regions with varying intensities caused by noise or inherent to the image, such as edges or fine textures.
   - Different types of linear smoothing filters are defined based on how weights are assigned and spatially arranged within the kernel.
   - Commonly used filters will be discussed, highlighting their specific characteristics and applications.

   a. **Box Filter:**
      - **Operation:** The box filter works by computing the average pixel value within a defined neighborhood or window surrounding each pixel. This average value replaces the original pixel value.
      - **Effect:** The box filter effectively blurs the image, reducing high-frequency components and smoothing abrupt transitions, but it might cause visible blurring, especially around edges and details.
      - **Usage:** This filter is simple and fast, making it suitable for preliminary noise reduction but might not be ideal for preserving fine details.



b. Gaussian Filter:

- **Operation:** Utilizes a weighted average derived from the Gaussian distribution to compute the new pixel value within the filter's window.
- **Effect:** The Gaussian filter is effective in reducing noise while preserving more detail compared to the box filter. It achieves smoother results owing to the distribution of weights.
- **Usage:** Widely employed due to its ability to effectively blur while maintaining better edge preservation, making it suitable for various computer vision tasks.

$$K = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} * \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

2. Non-Linear Filters:

Non-linear filter is a filter whose output is not a linear function of its input. Non-linear filtering cannot be done with convolution or Fourier multiplication. Median filter is a simple example of a non-linear filter.

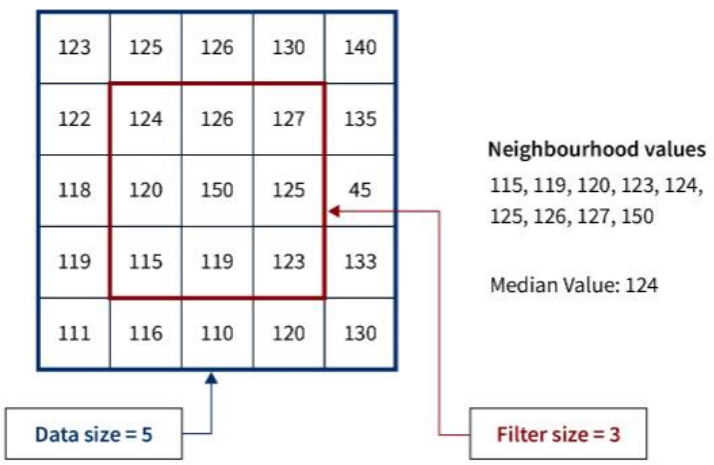
a. Minimum Filter:

- The minimum filter is one of the morphological filters .It is also called as **dilation filter**. The dark values present in an image are enhanced by the minimum filter. When the minimum filter is applied to a digital image it picks up the minimum value of the neighbourhood pixels under the filter window and assigns it to the current pixel. A pixel with the minimum value means the darkest pixel from the window.
- When minimum filter is applied, the object boundaries present in an image are extended.

b. Maximum Filter:

- Maximum filter is also called **erosion filter**. It works opposite to minimum filter. When a maximum filter is applied, the darker objects present in the image are eroded. The maximum filter replaces each pixel value of a digital image with the maximum value(i.e., the value of the brightest pixel) of its neighbourhoods.
- Applying the maximum filter removes the negative outlier noise present in a digital Image.

c. **Median Filter:**
   - The median filter is a type of nonlinear smoothing filter that replaces each pixel in the image with the median value of its neighbouring pixels. The size of the neighbourhood is also defined by the filter kernel. Unlike the mean filter, the median filter does not blur edges and details in the image. Instead, it preserves them while removing the noise. Median filters are commonly used in image processing tasks that involve removing salt and pepper noise from the image.



How Smoothing Works:

1. **Convolution Process:**
   - **Kernel Application:** A filter kernel moves over the image, and at each position, a mathematical operation (like averaging) occurs between the kernel and neighbouring pixels.
   - **Blurring Effect:** The resulting value becomes the new value for the pixel under the centre of the kernel, causing blurring or smoothing effects.

2. **Effect on Image:**
   - High-frequency components, such as noise or fine details, are attenuated or removed.
   - Lower-frequency components, like larger structures or objects, are retained to varying degrees based on the filter type and parameters.

Applications:

1. **Object Recognition:** Enhances the detectability of objects by simplifying image structures.
2. **Medical Imaging:** Used to reduce noise and improve the visibility of structures in medical images.

Sharpening:

Sharpening using a Laplacian filter is a technique in image processing that enhances edges and details in images by emphasizing high-frequency components, such as edges. The Laplacian filter highlights rapid intensity changes in an image, making edges more pronounced.

Purpose of Sharpening with Laplacian Filter:

- **Enhancement of Edges:** Emphasizes boundaries and edges in images, making them more distinct and visually appealing.
- **Highlighting Details:** Brings out fine details and high-frequency components, enhancing image clarity.
- **Improving Image Quality:** Sharpens images to improve overall visual quality and detail perception.

Operation of the Laplacian Filter:

1. **Laplacian Kernel:**
   - The Laplacian filter is a second-derivative filter used to detect regions of rapid intensity change or discontinuities in an image.
   - Its kernel represents a discrete approximation of the Laplacian operator, which computes the sum of second derivatives in both horizontal and vertical directions.

2. **Convolution Process:**
   - **Kernel Application:** The Laplacian kernel is applied to the image through convolution.
   - **Effect on Image:** The resulting values represent the degree of intensity change. High positive or negative values indicate significant changes, which correspond to edges.

3. **Sharpening Effect:**
   - The Laplacian filter's output highlights areas with rapid intensity changes, effectively enhancing edges and details.
   - To obtain a sharpened image, the Laplacian-filtered image is added back to the original image, boosting the contrast at edges.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Steps for Sharpening with Laplacian Filter:

1. **Obtain the Laplacian of the Image:**
   - Apply the Laplacian kernel to the image to compute the second derivatives, emphasizing edge information.

2. **Enhance Edges:**
   - Add the Laplacian-filtered image to the original image. This step increases the contrast at edges, making them appear sharper.

Applications:

- **Image Enhancement:** Used to improve image quality by increasing the clarity and sharpness of edges and details.
- **Preprocessing:** Often used as a step-in image processing pipelines before subsequent analysis or feature extraction.
- **Visual Effects:** Applied in photography and graphics to create visually striking or crisp images.

Sharpening using a Laplacian filter is a powerful technique for enhancing image edges and details, but careful consideration of its impact on noise and parameter selection is essential for optimal results.

Unit 2

Image Transformation:

In computer vision, image transformation refers to the process of altering the appearance of an image to extract meaningful information, enhance features, or prepare it for further analysis. Image transformations are fundamental operations in computer vision as they can help improve the quality of input images, correct distortions, and make them suitable for various tasks such as object detection, image recognition, and more. Here, I will explain image transformations in the context of computer vision, with a focus on scaling and rotation, along with their properties:

1. Scaling: Scaling in computer vision involves resizing an image while preserving its content and spatial relationships. It can be either upscaling (making the image larger) or downscaling (making it smaller).

Properties:

- **Aspect Ratio Preservation:** To maintain the image's integrity, it's essential to preserve its aspect ratio during scaling. This ensures that objects in the image are not distorted.

- **Interpolation:** During scaling, interpolation methods are used to estimate the values of pixels in the new image. Common interpolation techniques include bilinear and bicubic interpolation, which help create smoother results.

- **Anti-aliasing:** When downscaling an image, anti-aliasing techniques are often applied to reduce aliasing artifacts (jagged edges) that can occur due to the reduced resolution.

- **Scaling Factor:** The scaling factor determines the degree of scaling. For instance, a scaling factor of 2 doubles the image's dimensions.

- **Applications:** Scaling is used in computer vision for tasks such as resizing input images to a consistent size for deep learning models, pyramid image representations for object detection at multiple scales, and real-time video processing.

2. Rotation: Rotation in computer vision involves changing the orientation of an image by a specific angle. It's often used to align objects of interest or correct for misalignment.

Properties:

- **Angle of Rotation:** The angle by which the image is rotated is specified, typically in degrees or radians. Positive angles represent counterclockwise rotation, while negative angles indicate clockwise rotation.

- **Interpolation:** Just like with scaling, interpolation methods (e.g., bilinear or bicubic) are used to estimate pixel values in the rotated image.

- **Center of Rotation:** The point around which the image is rotated is crucial. In computer vision, this point is often determined based on the application's requirements.

- **Angle Units:** Depending on the software or library used, angles can be specified in degrees or radians.

- **Applications:** Rotation is employed in computer vision for aligning objects in images, registering images from different viewpoints, and correcting images that are not level or have orientation issues.

3. Discrete Fourier Transform (DFT):

In computer vision, the Discrete Fourier Transform (DFT) is a mathematical operation used for analyzing and processing images in the frequency domain. It is a fundamental tool for understanding the frequency content of an image, which can be valuable for various image processing tasks. Here's an explanation of DFT in computer vision:

1. Definition:

- The Discrete Fourier Transform (DFT) is a mathematical transformation that converts a spatial domain image into its frequency domain representation.
- It is a discrete counterpart of the Continuous Fourier Transform (CFT) and is used for digital data, such as images, where the data is sampled at discrete points.

2. Working Principle:

- DFT decomposes an image into a sum of sinusoidal components, each with a specific frequency, amplitude, and phase. These sinusoidal components are called "frequency components" or "frequencies."
- Each frequency component corresponds to a particular spatial frequency in the original image. Low-frequency components represent smooth variations, while high-frequency components represent rapid changes or fine details.

3. Mathematical Representation:

- The DFT of a 2D image I(x, y) with dimensions MxN is typically represented as a complex-valued matrix F(u, v) of the same dimensions:

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

- Here, (u, v) represents the spatial frequency coordinates, (x, y) are the spatial coordinates of the image, and j is the imaginary unit.

4. Applications:

- **Frequency Analysis:** DFT is used to analyze the frequency content of an image, which can be helpful in tasks like edge detection, texture analysis, and pattern recognition.
- **Image Enhancement:** By modifying the amplitude or phase of specific frequency components in the frequency domain and then applying the Inverse

DFT (IDFT), you can perform operations like filtering and image enhancement.

- **Image Compression:** Techniques like the Discrete Cosine Transform (DCT), which is closely related to DFT, are used in image compression methods like JPEG.
- **Feature Extraction:** DFT can be used to extract features from an image in the frequency domain, which can be used for object recognition and classification.
- **Noise Removal:** Filtering in the frequency domain can be used to remove noise from images.

5. Fast Fourier Transform (FFT):

- Computing the DFT directly using the above formula can be computationally expensive, especially for large images. In practice, the Fast Fourier Transform (FFT) algorithm is often used to efficiently calculate the DFT.
- FFT reduces the computational complexity from O(N^2) to O(N log N), making it practical for real-time or large-scale image processing.

In summary, the Discrete Fourier Transform (DFT) is a valuable tool in computer vision for analyzing and processing images in the frequency domain. It helps reveal the frequency characteristics of an image, which can be leveraged for various tasks, including image enhancement, feature extraction, and image compression. The efficient implementation of DFT through FFT is crucial for practical applications in computer vision.

4. Discrete Cosine Transform (DCT):

In computer vision, the Discrete Cosine Transform (DCT) is a mathematical transformation used for various image processing tasks, particularly in the context of image compression and feature extraction. It is closely related to the Discrete Fourier Transform (DFT) and is used to analyze and represent the frequency content of an image. Here's an explanation of DCT in computer vision:

1. Definition:

- The Discrete Cosine Transform (DCT) is a mathematical transformation that converts an image from its spatial domain into its frequency domain representation.
- Unlike the Discrete Fourier Transform (DFT), which uses complex sinusoids, the DCT uses only real-valued cosine functions.

2. Working Principle:

- DCT decomposes an image into a sum of cosine components, each with a specific frequency, amplitude, and phase.
- The primary idea behind the DCT is to represent the image data as a linear combination of a small number of low-frequency cosine functions, as most image information is concentrated in the lower frequencies.

3. Mathematical Representation:

- The DCT of a 2D image I(x, y) with dimensions MxN is typically represented as a matrix of the same dimensions containing the DCT coefficients C(u, v):

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y) \cos\left[\frac{(2x+1)u\pi}{2M}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

- Here, (u, v) represents the frequency coordinates, (x, y) are the spatial coordinates of the image, and α(u) and α(v) are scaling factors (usually $1/\sqrt{2}$ for u or v = 0, and 1 for others).

4. Applications:

- **Image Compression:** DCT is widely used in image compression methods such as JPEG (Joint Photographic Experts Group). By quantizing and discarding the coefficients corresponding to high-frequency components (which typically contain less visual information), DCT-based compression reduces the storage size of the image.
- **Feature Extraction:** DCT can be used for feature extraction in various computer vision tasks. By analyzing the DCT coefficients of different image blocks, distinct texture or pattern features can be captured.

- **Image Watermarking:** DCT is used in watermarking techniques to embed information into the frequency domain of an image while minimizing visual distortion.
- **Image Denoising:** DCT-based denoising methods exploit the fact that noise in images often has a higher frequency than the underlying signal. Filtering or thresholding DCT coefficients can reduce noise while preserving image details.
- **Image Analysis:** DCT can be employed in various analysis tasks where frequency information is relevant, such as image segmentation and object recognition.

5. DCT Variants:

- There are several variants of DCT, including DCT types (e.g., DCT-I, DCT-II, DCT-III, DCT-IV), each with different mathematical formulations. DCT-II (the most common) is used in JPEG compression.
- The 2D DCT can also be extended to higher dimensions for video and multi-dimensional data analysis.

In summary, the Discrete Cosine Transform (DCT) is a valuable tool in computer vision for representing image data in the frequency domain. It has various applications, with image compression being one of the most well-known. DCT allows for efficient representation of image information in terms of cosine functions, which makes it a critical component in many image and video processing algorithms.

5. Discrete Sine Transform (DST):

In computer vision, the Discrete Sine Transform (DST) is a mathematical transformation used for various image processing and analysis tasks. The DST is closely related to the Discrete Cosine Transform (DCT) and the Discrete Fourier Transform (DFT), and it plays a role in applications such as image compression and feature extraction. Here's an explanation of DST in the context of computer vision:

1. Definition:

- The Discrete Sine Transform (DST) is a mathematical transformation that converts an image from its spatial domain into its frequency domain representation.
- Similar to the DCT, the DST decomposes an image into a sum of sine components, each with a specific frequency, amplitude, and phase.
- Unlike the DCT, which uses cosine functions, the DST uses only real-valued sine functions.

2. Working Principle:

- DST operates on an image by decomposing it into sine waves of varying frequencies and amplitudes.
- The primary idea behind the DST, like the DCT, is to represent the image data as a linear combination of a small number of low-frequency sine functions, with the majority of image information concentrated in the lower frequencies.

3. Mathematical Representation:

- The DST of a 2D image I(x, y) with dimensions MxN is typically represented as a matrix of the same dimensions containing the DST coefficients S(u, v):

$$S(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\sin\left[\frac{(2x+1)u\pi}{2M}\right]\sin\left[\frac{(2y+1)v\pi}{2N}\right]$$

- Here, (u, v) represents the frequency coordinates, (x, y) are the spatial coordinates of the image, and α(u) and α(v) are scaling factors (usually $1/\sqrt{2}$ for u or v = 0, and 1 for others).

4. Applications:

- **Image Compression:** Although less commonly used than the DCT in image compression standards like JPEG, DST-based compression methods exist. These methods use DST to represent and compress image data in the frequency domain.

- **Feature Extraction:** DST can be used for feature extraction in computer vision tasks. Analyzing the DST coefficients of image blocks can capture distinct texture or pattern features.
- **Image Analysis:** DST can be employed in various image analysis tasks where frequency information is relevant, such as texture classification and object recognition.
- **Signal Processing:** DST is also used in signal processing applications beyond images, including audio processing and speech recognition.

5. DST Variants:

- Similar to the DCT, there are different variants of the DST, including DST-I, DST-II, DST-III, and DST-IV, each with different mathematical formulations. DST-II is the most commonly used variant in applications similar to those of the DCT.

In summary, the Discrete Sine Transform (DST) is a mathematical tool used in computer vision for representing image data in the frequency domain using sine functions. While it is less commonly used than the DCT in image compression standards, it finds applications in feature extraction and various image analysis tasks where frequency information is relevant. Like the DCT, DST can efficiently represent image information in terms of sine waves, making it valuable in certain computer vision applications.

6. Walsh-Hadamard Transform:

The Walsh-Hadamard Transform (WHT) is a mathematical operation used in signal processing and computer vision to analyze and manipulate images. It's a type of Fourier-related transform, like the Discrete Fourier Transform (DFT) or Discrete Cosine Transform (DCT), but with unique properties that make it particularly useful in certain applications.

Basics of the Walsh-Hadamard Transform:

1. Orthogonal Transform: The WHT is an orthogonal transform, meaning its basis functions are orthogonal to each other. This property simplifies the inverse transform, as it involves the same set of basis functions used in the forward transform but with some scaling adjustments.

2. Binary Nature: Unlike the continuous values in the Fourier Transform, the WHT deals with binary values (typically ±1). This binary nature makes it computationally efficient for digital systems, especially in hardware implementations.

3. Fast Implementation: Similar to the Fast Fourier Transform (FFT) for the DFT, there are algorithms for fast computation of the WHT, such as the Fast Walsh-Hadamard Transform (FWHT), which reduces the computational complexity significantly.

Applications:

In computer vision, the WHT finds applications in several areas:

- **Image Compression:** Similar to the DCT in JPEG compression, the WHT can efficiently represent images with fewer coefficients. The transform concentrates most of the signal energy into a few significant coefficients, allowing for compression with minimal loss of quality.

- **Feature Extraction:** WHT can be used for feature extraction, where it analyzes different frequency components in an image. For example, it can help identify edges or patterns in an image by highlighting specific Walsh-Hadamard coefficients associated with those features.

- **Pattern Recognition:** In pattern recognition tasks, the WHT can be used to extract discriminative features from images. These features can then be used in classification or recognition algorithms.

4. Noise Reduction: It can also be utilized in noise reduction by suppressing noise in certain frequency components, similar to how Fourier-based methods are employed in filtering.

Steps in Walsh-Hadamard Transform:

1. Image Preparation: The image is typically converted into a square matrix, usually of size 2^n for ease of computation.

2. Transform Matrix: The WHT is applied by multiplying the image matrix with the Walsh-Hadamard matrix or its transpose.

3. Coefficient Calculation: The resulting transformed matrix contains coefficients that represent different frequency components of the image.

4. Compression or Analysis: Depending on the application, these coefficients can be used for compression or further analysis.

In summary, the Walsh-Hadamard Transform offers a unique way to analyze and process images, particularly suitable for tasks like compression, feature extraction, noise reduction, and pattern recognition in computer vision applications. Its binary nature and orthogonal properties make it a valuable tool in certain scenarios, offering advantages over other transform methods.

Image Restoration:

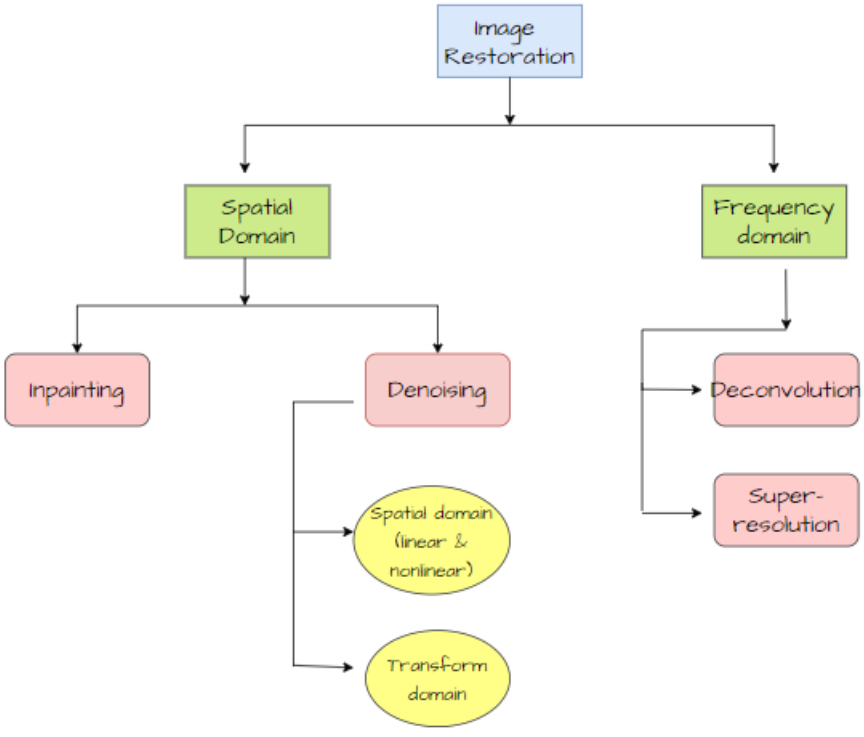
Image restoration refers to the process of improving the quality and appearance of a degraded or damaged image. It involves recovering lost or degraded information to restore the image to its original or desired state.

Let's visualize image restoration with an image.

Image restoration example

Image restoration techniques can be classified into two main categories:

1. Frequency domain (transform-based).
2. Spatial domain restoration (filtering based).



Classification of the denoising techniques

Frequency domain:

Frequency domain restoration techniques are image restoration methods that operate in the frequency domain. They involve transforming the image from the spatial domain to the frequency domain using mathematical transformations like the Fourier Transform or Wavelet Transform.

- **Deconvolution**: It is used to recover sharpness and detail in blurred images. It involves estimating and compensating for the blurring effects by deconvolving the degraded image with an estimated blur kernel in the frequency domain, resulting in the restoration of high-frequency details.
- **Super-resolution:** This technique uses frequency domain operations, such as interpolation or extrapolation, to predict or reconstruct high-frequency components in the frequency domain to enable the generation of a higher-resolution image with improved visual quality and finer details.

Spatial domain:

Spatial domain is an image restoration method that operates directly on the pixel values of an image, meaning it manipulates the pixel values within the image itself. It includes inpainting and denoising.

- **Inpainting**: It is a technique used to fill in missing or damaged regions of an image with plausible content. It involves estimating the missing information based on the surrounding pixels or structures.

Now let's discuss denoising in image restoration in detail.

Denoising an Image:

Before moving on to denoising techniques, let's understand what is meant by noise in an image.

What is noise in an image?

Noise in an image refers to unwanted and random variations in pixel values that deviate from the true content of the image. It can be seen as a disturbance that disrupts the desired information in the image and degrades the quality of an image.

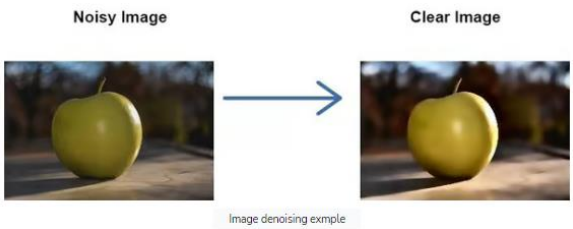
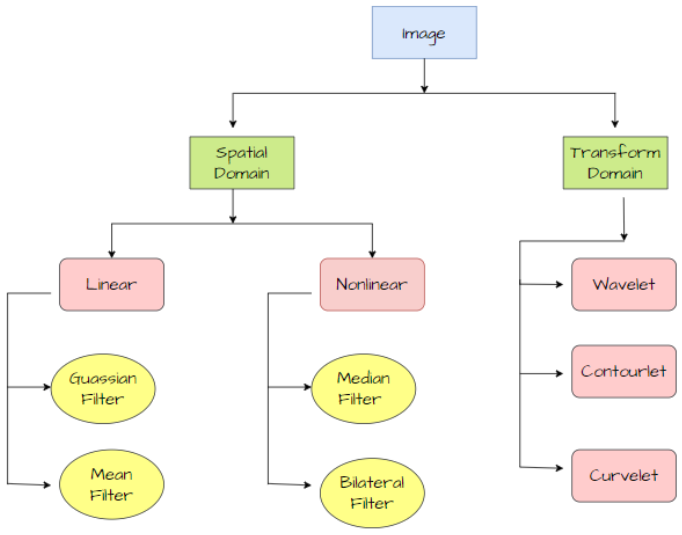


Image denoising exmple

Some common sources of noise include sensor noise and image processing noise in digital images. Sensor noise arises from the image sensor itself during the image capture process from factors such as thermal noise, which results in random fluctuations and variations in pixel values. Image processing noise can occur during various stages, including analog-to-digital conversion and transmission over networks.

Denoising techniques:

Image denoising techniques can be classified based on the domains in which they operate. The two primary domains used for denoising are the spatial domain and the transform domain. Below is a classification of denoising filters based on these domains.



Classification of the denoising techniques

Spatial domain:

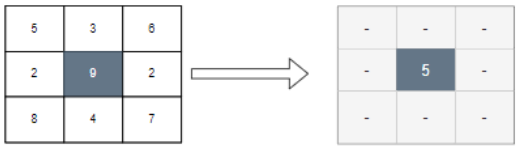
Spatial domain filters, also known as direct or pixel-based filters, operate directly on the pixel values of an image. These filters consider the values of neighbouring pixels to estimate the denoised value for each pixel. They are further classified into linear and nonlinear filters:

Linear spatial filters:

- **Gaussian filter:** It applies a weighted average to each pixel based on a Gaussian kernel. It reduces noise in an image by smoothing the pixel values with the amount of smoothing controlled by the standard deviation of the Gaussian kernel.

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- **Mean filter:** It is a simple and widely used filter for reducing noise by smoothing the image. The mean filter replaces each pixel with the average value of its neighbouring pixels.



Mean Filtered

Mean filter example

$$M(x,y) = \frac{1}{N^2} \sum_i \sum_j I(i,j)$$

Nonlinear spatial filters

- **Median filter:** The median filter replaces the value of each pixel with the median value of the pixel intensities within a defined neighborhood.

$$M(x,y) = \text{median}\left(I(i,j) \mid i,j \text{ within window}\right)$$

- **Bilateral filter:** It applies a weighted average to each pixel by taking into account the differences in spatial distance and pixel intensity, which results in noise reduction while maintaining important image features and sharp edges.

$$B(x,y) = \frac{1}{W} \sum_i \sum_j I(i,j) \cdot W(i,j)$$

Transform domain:

Transform domain denoising techniques, such as wavelet denoising, contourlet denoising, and curvelet denoising, apply specific transforms to effectively reduce noise in images. Here's a brief explanation of each technique:

- **Wavelet denoising:** It applies the wavelet transform to decompose an image into different frequency bands as the wavelet coefficients corresponding to noise are comparatively of higher magnitude. Therefore, it thresholds the wavelet coefficients and suppresses the noise while retaining the significant image features.
- **Contourlet denoising:** It is similar to wavelet denoising and utilizes the contourlet transform to decompose an image into subbands at multiple scales and orientations. It preserves image structure along contours and edges.
- **Curvelet denoising:** It is designed to handle images with curve-like structures and utilizes the curvelet transform to decompose an image into curvelet coefficients that provide high directional sensitivity.

Unit 3

Segmentation:

What Is Image Segmentation?

Image segmentation is a fundamental technique in computer vision that involves dividing an image into multiple segments or regions to simplify the representation of an image and make it more meaningful and easier to analyse. The goal is to partition an image into semantically meaningful parts or objects, enabling machines to understand, interpret, and process visual information more effectively.

There are various methods used for image segmentation, including:

1. **Thresholding:** Dividing pixels based on intensity values (brightness or colour) to create binary masks.

2. **Clustering:** Grouping pixels with similar attributes together using algorithms like K-means clustering or mean shift.

3. **Edge Detection:** Identifying boundaries or edges between different objects in an image using techniques like Sobel, Canny, or Prewitt edge detectors.

4. **Region-based segmentation:** Dividing an image into regions based on similarities in colour, texture, or other features.

5. **Semantic segmentation:** Assigning semantic labels (e.g., person, car, building) to each pixel in an image, providing a detailed understanding of the image content.

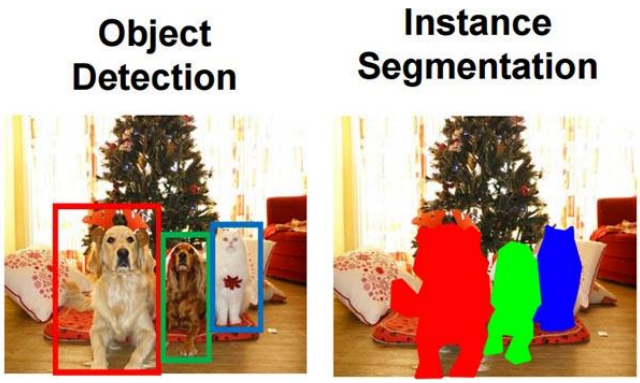
6. **Instance segmentation:** Distinguishing individual objects of the same class and assigning unique labels to them.

Image segmentation is crucial in various applications, such as medical imaging (tumor detection, organ segmentation), autonomous driving (object detection and tracking), image editing, satellite imaging, and many more, as it forms the basis for higher-level image understanding and analysis by machines.

How Does Image Segmentation Work?

We can divide or partition the image into various parts called segments. It's not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works.

An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation.



Object detection builds a bounding box corresponding to each class in the image. But it tells us nothing about the shape of the object. We only get the set of bounding box coordinates.
Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a far more granular understanding of the object(s) in the image.

Techniques for Segmentation:

1. **Thresholding:** This is the simplest form of segmentation, where a threshold value is chosen, and all pixels with values above or below this threshold are classified into separate segments. Binary thresholding, for example, separates objects from the background.
2. **Region Growing:** Starting from seed points, this technique groups adjacent pixels with similar characteristics to form segments. It's useful for regions with consistent properties.
3. **Edge-based Segmentation:** Edge detection can be used to identify boundaries between different objects or regions in an image. Once edges are detected, they can be used to separate different regions.
4. **Clustering:** Techniques like k-means clustering group pixels with similar feature values together, forming segments.
5. **Watershed Segmentation:** This technique treats the image as a topographic surface, and regions are defined based on the watershed lines. It's especially useful for separating touching or overlapping objects.

Edge Detection:

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

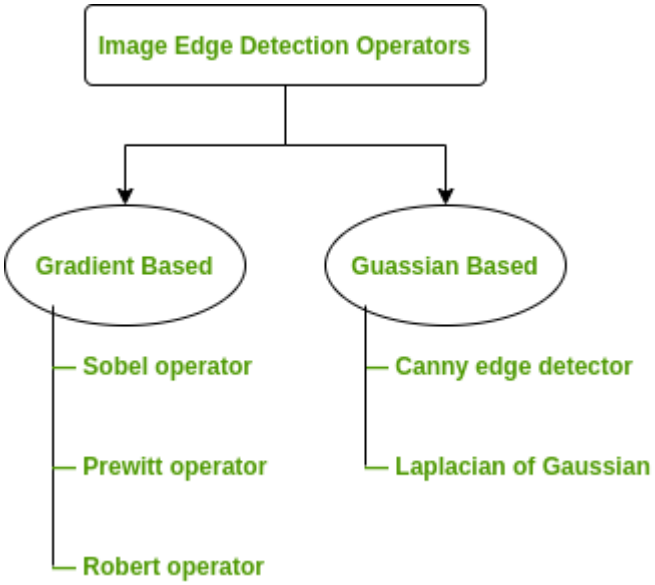
Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing like

- 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.

Edge Detection Operators are of two types:

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



Gradient Based:

1. **Sobel Operator:** It is a discrete differentiation operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3 x 3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

   Advantages:
   1. Simple and time efficient computation.
   2. Very easy at searching for smooth edges.

   Limitations:
   1. Diagonal direction points are not preserved always.
   2. Highly sensitive to noise.
   3. Not very accurate in edge detection.
   4. Detect with thick and rough edges does not give appropriate results.

2. **Prewitt Operator:** This operator is almost similar to the sobel operator. It also detects vertical and horizontal edges of an image. It is one of the best ways to detect the orientation and magnitude of an image. It uses the kernels or masks:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

   Advantages:
   1. Good performance on detecting vertical and horizontal edges.
   2. Best operator to detect the orientation of an image.

   Limitations:
   1. The magnitude of coefficient is fixed and cannot be changed.
   2. Diagonal direction points are not preserved always.

3. Robert Operator: This gradient-based operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation. Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks:

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Advantages:

1. Detection of edges and orientation are very easy.
2. Diagonal direction points are preserved.

Limitations:

1. Very sensitive to noise.
2. Not very accurate in edge detection.

Gaussian Based:

1. **Marr-Hildreth Operator or Laplacian of Gaussian (LoG):**
   It is a gaussian-based operator which uses the Laplacian to take the second derivative of an image. This really works well when the transition of the grey level seems to be abrupt. It works on the zero-crossing method i.e. when the second-order derivative crosses zero, then that particular location corresponds to a maximum level. It is called an edge location. Here the Gaussian operator reduces the noise and the Laplacian operator detects the sharp edges. The Gaussian function is defined by the formula:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \left(\frac{x^2+y^2}{2\sigma^2}\right)$$

Where

σ

is the standard deviation.

And the LoG operator is computed from

$$\text{LoG} = \frac{\partial^2}{\partial x^2} G(x,y) + \frac{\partial^2}{\partial y^2} G(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Advantages:

1. Easy to detect edges and their various orientations.
2. There is fixed characteristics in all directions.

Limitations:

1. Very sensitive to noise.
2. The localization error may be severe at curved edges.
3. It generates noisy responses that do not correspond to edges, so-called "false edges".

2. Canny Operator: It is a gaussian-based operator in detecting edges. This operator is not susceptible to noise. It extracts image features without affecting or altering the feature. Canny edge detector have advanced algorithm derived from the previous work of Laplacian of Gaussian operator. It is widely used an optimal edge detection technique. It detects edges based on three criteria:

1. Low error rate
2. Edge points must be accurately localized
3. There should be just one single edge response

Advantages:

1. It has good localization.
2. It extract image features without altering the features.
3. Less Sensitive to noise.

Limitations:

1. There is false zero crossing.
2. Complex computation and time consuming.

Some Real-world Applications of Image Edge Detection:

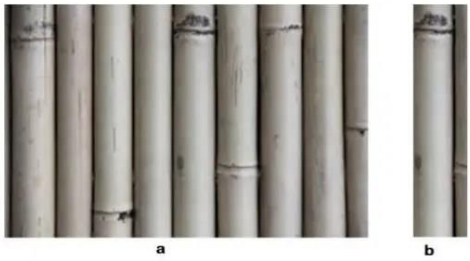
- Medical imaging, study of anatomical structure.
- Locate an object in satellite images.
- Automatic traffic controlling systems.
- Face recognition, and fingerprint recognition.

Texture Analysis:

What is Texture?

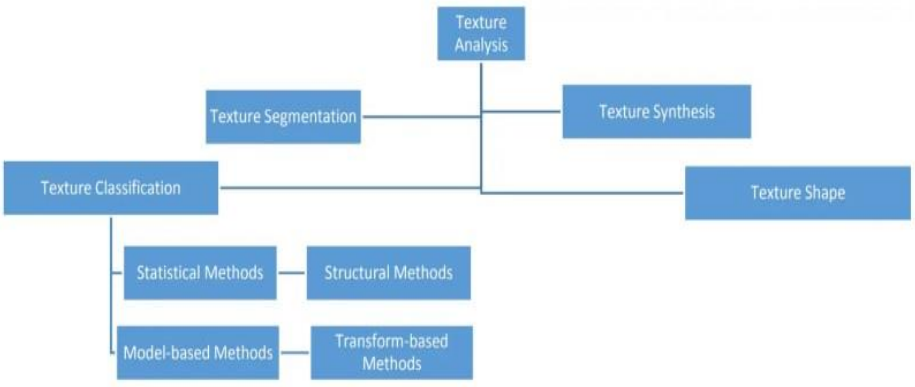
There are two kinds of texture: one is tactile and the other one is optical, where we can feel the tactile texture by touching or seeing the surface. When we talk about the optical or visual texture, it refers to the shape and content of the image. Humans can easily diagnose the texture of the image but making a machine to analyse the texture of the image has its complexity. In the field of image processing, we can consider the spatial changes of the brightness intensity of the pixel as the texture of the image.

In image processing, textural images are those images in which a specific pattern of texture distribution is repeated sequentially throughout the image. The below image can be a representation of the textural image where part (b), represents the repeated pattern on the texture.



What is Texture Analysis?

Till now we have got an understanding of the texture in the image data. The aim of the article is to discuss how machines understand the texture of images so that machines can be capable of performing different tasks of image processing. Understanding the texture of the images requires texture analysis and we can consider texture analysis as a whole subject. In the general view of the textural analysis, we can find the areas like the following.



Considering the above representation, the texture analysis can be categorized into four categories: texture segmentation, texture synthesis, texture classification, texture shape.

Let's have a small discussion about the categories so that we will have a proper vision about the areas of image processing where they can be used.

Texture Segmentation: In image data, we can find out the difference between the image areas in the context of the texture. By texture segmentation, we find different boundaries of the different textures in the image. We can also say that, in texture segmentation, we compare different areas of the images if the textual characteristics are different and define them by assigning the boundaries.

Texture Synthesis: In image synthesis, we use methods to make images that have a similar texture as the images we have as input. This part of the texture analysis is being used in the creation of computer games and image graphics.

Texture Shape Extraction: In this section, we try to extract the 3D view and areas of the images. Normally these areas are covered with a unique or specific texture. This section is useful in analysing the shape and structure of the objects in the image using the image's textual properties and spatial relationship of textures with each other.

Texture classification: We can consider it as the most important lesson of texture analysis which is responsible for describing the type of image texture. Texture classification is the process of assigning an unknown sample of textures from the image to any predefined texture class.

Here we have seen a basic introduction to the texture analysis and the parts of the texture analysis. Now we are required to know the challenges we may face in the texture analysis.

Where is texture analysis used?

Nowadays, texture analysis is a significant part of many tasks, from **medical imaging** to **remote sensing**, and is also used in large image databases to query the content.

In **industrial inspection** texture analysis is a powerful tool where the current technology fails to work. Let's take an instance of **wood manufacturing**, in this case, it is difficult to detect a crack without the use of texture analysis.

Texture detection is also used for **grading carpets** based on the appearance changes due to wearing. Texture analysis is used for **leather inspection**, which is done by assessing the colour, thickness, and grey-level variations. The defective pieces often have scars or folds on the skin.

The applications of texture analysis range from **texture classification** like **remote sensing** (*fig 5*) to **texture segmentation** tasks like **biomedical imaging** (*fig 6*). It is also used in **image synthesis** and **pattern recognition** tasks such as recognizing a painting from a photograph.

When objects in an image get classified by their texture property and not by intensity, or when threshold techniques cannot classify them properly, in such cases texture analysis plays a significant role.

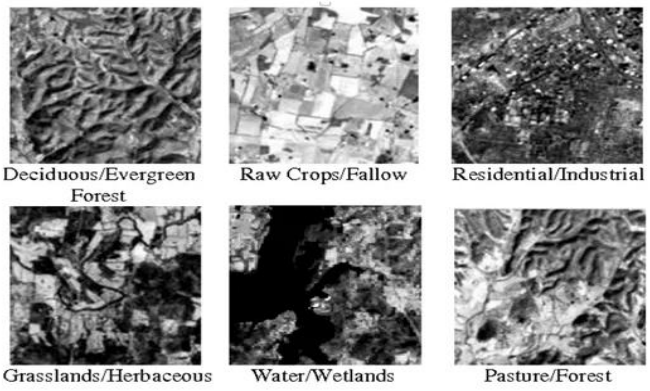


Fig 5. Remote sensing using texture analysis. These texture patterns are recognized by clustering and labeled by an expert in the domain of remote sensing

The below figure (*fig 6*) shows an ultrasound picture of the second-rate vena cava (a dull district in the lower third of the picture). A segment of the liver, encased by white specks for simple distinguishing proof, shows a texture that is unique in relation to that of the surrounding tissue.

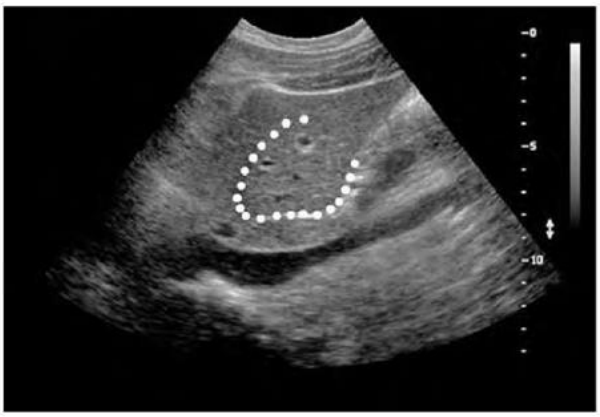


Fig 6. Biomedical Imaging using texture analysis

Nowadays texture analysis is also being used in the **food manufacturing** industry to understand the quality of food. A variety of textures are found in foods such as hard candy, chewy chocolate chip cookies, crisp crackers, sticky toffee, crunch celery, tender steak, and many more. Texture analysis is used to a great extent in this domain, such as the *mouthfeel* property of food can be easily measured using texture analysis.

It is also used in a study called **Rheology**, which is the science of disfigurement and flow of matter or in other words, the response of an object when the application of force is made on it.

Apart from all of these, texture analysis can be used to measure/ assess the quality of many products such as adhesives, medicines, skin/hair care products, polymers, etc.

How can texture analysis be used in classification problems and why is it important?

So far we have understood diverse types of textures and have seen real-life examples of places where texture analysis is useful. Let us understand how it can be used in a classification problem, where the main objective of the classifier is to categorize textural images by providing descriptors for each image. In other words,

> Allocating an unknown sample to a pre-defined texture class is known as texture classification.

While doing texture classification, patterns and texture content of the image are taken into consideration. Texture-based classification is done based on textual features such as roughness, irregularity, uniformity, smoothness, and so on. Every class in any image dataset is most likely to have a different texture, which makes it a unique attribute that helps the model classify the images more accurately.

Different techniques and methods to extract texture:

There are multiple methods that are used to extract texture from an image. In this blog, we will discuss the most used and significant methods for texture extraction.

GLCM (Grey Level Co-occurrence Matrix) is a common and basic statistical approach for texture analysis. The GLCM features are based on second-order statistics, and are used to gain insight into the average degree of correlation between pixels from the view of uniformity, homogeneity, etc.

LBP is an approach that combines structural and statistical methods to make texture analysis more efficient. An important feature of LBP in the real world is its sturdiness towards the monotonic grey-scale changes caused by different lighting conditions. Its simple computation allows for use in real-time settings.

1. Grey Level Co-occurrence Matrix (GLCM):

GLCM gives information about how pixels of an image are related to each other and this relationship helps us classify the texture based on multiple features that are extracted from GLCM. The matrix gives information about the position of pixels which have similar intensity. The set of possible intensity values are rows and columns labels of the 2-D array *(P).*

A GLCM $P_d[i,j]$ is first initialized by specifying a displacement vector $d = (dx, dy)$ and counting all pairs of pixels separated by d in the angle of the displacement vector, having grey levels *j* and *i* (where *j* is a column and *i* is a row).

In general, GLCM is represented as $P_d[i,j] = n_{ij}$, where n_{ij} is the number of occurrences of the pixel values *(i,j)* lying at distance d in the image. The co-occurrence matrix P_d has dimension *n*n*, where *n* is the number of grey levels in the image.

GLCM is calculated based on the distance and angle as mentioned in the displacement vector. For a pixel *'x'* we can compute the GLCM value in 8 different directions as shown in *fig 7*.

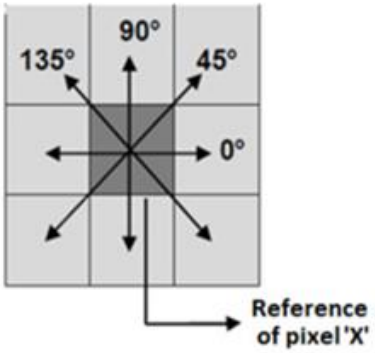


Fig 7. Closest neighbors of pixel 'x' [3]

For better understanding check the below example (*fig 8*) where a 4x4 image is composed using 4 grey levels. Here, in *fig 8* the value in GLCM for an angle of 0°, with *i*=2, *j* =3, and *d*=1 is 4. The matrix at the top in *fig* 8 shows that there are 4 instances in our image where a pixel with grey level 3 is separated horizontally (i.e. 0°) from a pixel of grey level 2.

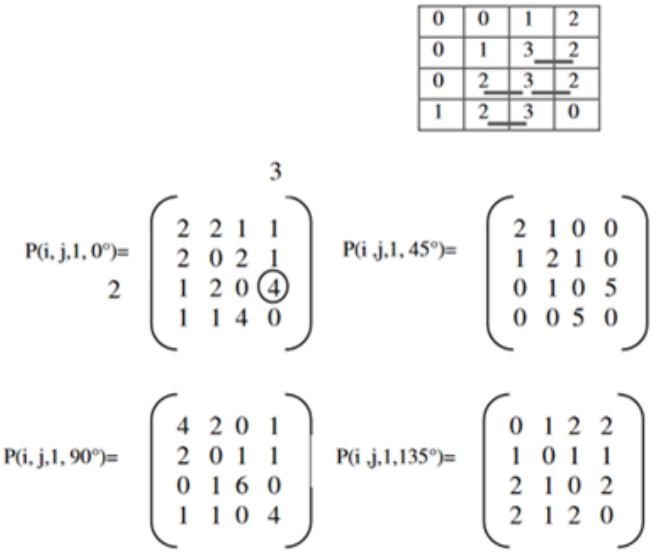


Fig 8. Example of co-occurrence matrix built from 4x4 image with 4 intensities [3]

This matrix can further be used to numerically compute the global textual features such as **correlation, energy, entropy, homogeneity, contrast, prominence, and shade**.

2. Local Binary Pattern (LBP):

While GLCM focuses on capturing the information from the whole image, LBP focuses more on the local features than the global features. Since the texture is the repetition of patterns, LBP tries to classify texture based on these patterns. The local representation of texture is computed by comparing a pixel with all the pixels in its neighbourhood.

Before constructing the LBP, we need to convert the image to grayscale. For each pixel in grayscale, we select a neighbourhood of size *r* around the central pixel. The LBP value is determined for the central pixel, by marking the neighbouring pixels as 0 and 1, wherever the intensity of the pixel is equal or greater than the central pixel it is marked as 1 otherwise as 0 (as shown in *fig* 9 which has a fixed neighbourhood of 3 x 3).

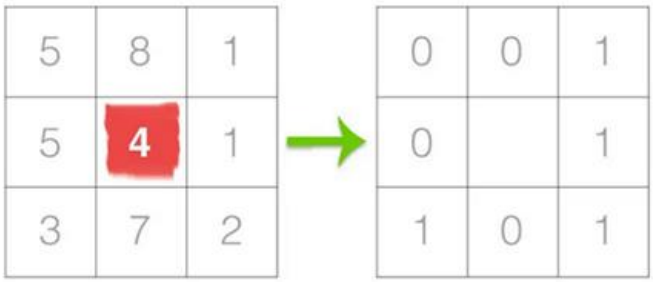


Fig 9. First step of constructing LBP [4]

In the next step to calculate the LBP we start from any of the adjoining pixels and work our direction clockwise or counterclockwise, this order must be kept the same for all pixels of all pictures in the dataset. This output is saved in an 8-bit array, which is transformed to decimal as shown in *fig 10*. With 8 surrounding pixels, there will be a total of $2^8 = 256$ potential combinations of LBP codes.

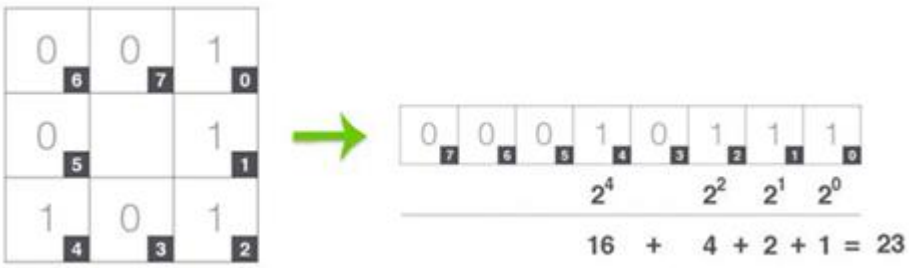


Fig 10. 8-bit binary neighborhood of the centre pixel and converting it into decimal representation[4]

Once we repeat the above method for all the pixels of an image, we get the LBP image. Just for reference see *fig 11* for an example of LBP representation (right) of an original image (left).

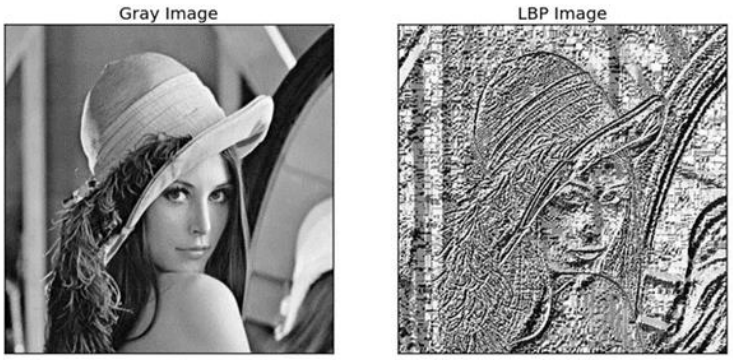


Fig 11. original image(left) LBP transformation(right) [5]

After using LBP, the texture is extracted so that the extremely fine-grained details of the texture are captured and it's much simpler to classify the texture.

Texture Segmentation:

The primary objective of Texture Segmentation is to partition an image into distinct regions based on differences in texture. Any texture measure that describes the texture in a local neighbourhood of each pixel can be used to segment an image into regions of similar texture by providing a value or a vector of values at each pixel. This segmentation typically involves a two-step process.

1. The first stage is the feature extraction phase, where textural properties are extracted. The objective is to create a model for each texture in the training dataset.

2. In the second stage, the classification phase, the test sample image's texture is analysed using the same technique as in the previous step. Then, a classification algorithm is used to compare the extracted features of the test image with the training imagery, determining its class.

Unit 4

Feature Extraction:

Feature extraction in computer vision is the process of identifying and extracting relevant and distinctive information or patterns from raw image data. These extracted features serve as a foundation for various computer vision tasks, including object recognition, image classification, image retrieval, and more.

Process of Feature Extraction:

1. Key Point Detection:

- **Identifying Distinctive Points:** Algorithms search for unique points or areas within an image that possess specific characteristics like corners, edges, or regions with high texture or contrast. These points are often called keypoints or interest points.
- **Scale-Space Representation:** As images can have features at various scales, feature extraction techniques often involve analyzing images across different scales to detect features at multiple resolutions.

2. Feature Description:

- **Extracting Feature Descriptors:** Once keypoints are identified, descriptors or representations are generated for these points. These descriptors encode information about the local image structure around each keypoint. They capture details such as gradients, color histograms, texture information, or other relevant characteristics.
- **Invariance Properties:** Effective feature descriptors should be invariant to certain transformations such as rotation, scaling, and changes in lighting conditions to ensure robustness in different scenarios.

3. Feature Matching:

- **Matching Corresponding Features:** Extracted features from different images are compared to find correspondences. This matching process involves comparing the descriptors of keypoints to identify similarities or matches between images. This step is crucial for tasks like image stitching, object recognition, or image retrieval.

Techniques Used in Feature Extraction:

1. SIFT (Scale-Invariant Feature Transform):

- SIFT detects keypoints at various scales, ensuring scale invariance.
- It localizes keypoints accurately, assigns orientations, and generates descriptive vectors robust to transformations.
- Suitable for tasks requiring high accuracy but can be computationally intensive.

2. SURF (Speeded-Up Robust Features):

- SURF employs integral images and approximations for Hessian matrix computation, making it faster than SIFT.
- It generates descriptors using gradient information and is robust to transformations, sacrificing a bit of accuracy for speed.

3. BRISK (Binary Robust Invariant Scalable Keypoints):

- BRISK focuses on speed by using a scale-space pyramid and generating binary descriptors.
- While sacrificing some accuracy compared to SIFT and SURF, BRISK is efficient for real-time applications.

Importance of Feature Extraction:

- **Dimensionality Reduction:** Extracted features reduce the complexity of image data, providing a more manageable and meaningful representation for subsequent processing.
- **Robustness and Generalization:** Effective feature extraction techniques enable robust performance across different images, lighting conditions, and variations in the environment.
- **Facilitating Higher-Level Vision Tasks:** Extracted features serve as inputs for higher-level computer vision tasks like object detection, image matching, and scene understanding.

Feature extraction is a foundational step in computer vision, influencing the accuracy, robustness, and efficiency of subsequent processing and analysis of visual data. The choice of feature extraction technique depends on the specific requirements of the task, balancing factors such as computational complexity, accuracy, and real-time performance.

1. SIFT (Scale-Invariant Feature Transform):

The Scale-Invariant Feature Transform (SIFT) is a powerful and widely used feature extraction technique in computer vision, primarily known for its robustness to changes in scale, rotation, illumination, and viewpoint. It was introduced by David Lowe in 1999 and has since become a fundamental tool for various computer vision applications, including object recognition, image stitching, and 3D reconstruction.

Key Components of SIFT:

1. Scale-space Extrema Detection:

- SIFT operates on multiple scales of an image to detect keypoints irrespective of their size. This is crucial because objects in images can appear at different scales.
- It achieves scale invariance by creating a Gaussian pyramid of the image. This pyramid consists of progressively down sampled versions of the original image, allowing SIFT to detect features at different scales.

2. Keypoint Localization:

- Once the scale-space representation is obtained, SIFT uses the Difference of Gaussians (DoG) technique to identify potential keypoints.
- DoG is computed by taking the difference between adjacent scales in the Gaussian pyramid. Keypoints are detected at locations where the difference in Gaussian-blurred images is maximal.

3. Orientation Assignment:

- SIFT assigns an orientation to each keypoint to achieve rotational invariance. It computes the dominant orientation of gradients in the local neighborhood of the keypoint.
- This orientation is used to normalize the keypoint's descriptor, ensuring consistency in feature representation even if the image is rotated.

4. Descriptor Generation:

- Around each keypoint, SIFT constructs a descriptor that captures the local image gradient information.
- This descriptor is typically a vector of gradient magnitudes and orientations in a spatial region around the keypoint. SIFT divides this region into sub-regions and creates histograms of gradient orientations within each sub-region.
- These histograms are concatenated to form a high-dimensional descriptor vector, which represents the keypoint's distinctive features.

Robustness and Applications:

- **Robustness to Transformations:** SIFT's strength lies in its robustness to various image transformations, including scale changes, rotations, changes in lighting conditions, and partial occlusions. This makes it highly reliable for object recognition and matching tasks.

- **Object Recognition and Matching:** SIFT features are widely used for matching objects across images. They enable the identification of correspondences between keypoints in different images, facilitating tasks like image alignment and object recognition.

- **Image Stitching and Panorama Creation:** In applications like creating panoramas or stitching multiple images together, SIFT features help identify common keypoints across images, allowing for accurate alignment and blending.

However, SIFT does have some limitations, notably its computational intensity and memory requirements. Generating SIFT features can be resource-intensive, making it less suitable for real-time applications on devices with limited computational capabilities. Despite these limitations, SIFT remains a cornerstone in computer vision due to its robustness and effectiveness in various applications requiring distinctive feature extraction.

2. SURF (Speeded-Up Robust Features):

SURF, which stands for Speeded-Up Robust Features, is a feature detection and description method in computer vision. It was introduced as an improvement over SIFT (Scale-Invariant Feature Transform) with a focus on computational efficiency while maintaining robustness in handling various image transformations.

Key Components of SURF:

1. **Integral Images:**
   - SURF utilizes integral images, which are precomputed data structures enabling fast calculations of rectangular area sums within an image. This drastically speeds up convolution-like operations and allows for rapid feature detection and descriptor computation.

2. **Hessian Matrix Approximation:**
   - Instead of directly computing the Hessian matrix, which characterizes the second-order derivatives of an image, SURF uses a box filter-based approximation to estimate the determinant of the Hessian matrix.
   - The determinant of the Hessian matrix at multiple scales and positions in an image is used to identify interest points or keypoints.

3. **Fast Keypoint Detection:**
   - SURF identifies interest points based on the locations where the determinant of the Hessian matrix exceeds a predefined threshold. This method efficiently identifies regions with distinctive structures, edges, or textures across different scales.

4. **Orientation Assignment:**
   - To achieve rotation invariance, SURF computes the dominant orientation for each keypoint using the responses in the wavelet-based image approximation. This orientation assignment helps in aligning the descriptors for consistent matching.

5. **Descriptor Generation:**
   - SURF constructs feature descriptors based on the distribution of gradients in circular regions around each detected keypoint.
   - These descriptors are generated by considering the sums of Haar wavelet responses over rectangular regions, which are calculated using the integral images. The response is then represented in a feature vector.

Advantages of SURF:

1. **Speed and Efficiency:**
   - Integral images and the approximation of the Hessian matrix allow SURF to perform computations faster than SIFT.
   - The use of box filters and integral images speeds up feature detection and descriptor computation, making it suitable for real-time applications.

2. **Robustness:**
   - SURF maintains a level of robustness against various image transformations, including scaling, rotation, and changes in lighting conditions.

3. **Scalability:**
   - SURF's approach allows for efficient computation across different scales, enabling the detection of features at multiple levels of granularity.

Use Cases:

- **Object Recognition:** SURF features can be used to identify and match objects in images, enabling applications like object tracking and augmented reality.

- **Image Stitching:** It helps in aligning and merging multiple images by detecting and matching common features.

- **Robotics and Autonomous Systems:** SURF features are used for navigation, localization, and mapping in robotics applications.

SURF's balance between speed and robustness makes it a popular choice in scenarios where real-time performance is essential, such as video analysis, robotics, and other applications requiring rapid image processing. However, it's worth noting that while SURF is faster than SIFT, it may sacrifice a bit of accuracy compared to the more computationally intensive SIFT algorithm.

3. BRISK (Binary Robust Invariant Scalable Keypoints):

BRISK, standing for Binary Robust Invariant Scalable Keypoints, is a feature detection and description method used in computer vision. It's designed to offer a compromise between speed, robustness, and efficiency in extracting distinctive features from images.

Key Components of BRISK:

1. Feature Detection:

- **Scale-space Pyramid:** BRISK uses a scale-space representation, similar to SIFT and SURF, to detect keypoints across multiple scales. This scale-space pyramid helps in identifying features at different levels of detail.

2. Feature Description:

- **Binary Descriptors:** Unlike SIFT and SURF, which use floating-point descriptors, BRISK employs binary descriptors. These descriptors encode local image properties in a binary form. Instead of representing gradients with real-valued numbers, BRISK uses a pattern comparison strategy to produce binary strings, making them more compact and efficient for computation.

3. Rotation Invariance:

- **Sampling Pattern:** BRISK uses a sampling pattern around each detected keypoint to ensure rotation invariance. This pattern captures points in the neighborhood of the keypoint at various orientations, allowing the generation of descriptors that are invariant to image rotation.

How BRISK Works:

- **Scale-space Pyramid:** BRISK begins by constructing a scale-space pyramid by applying Gaussian blurring at various levels. This pyramid enables the detection of keypoints at different scales.

- **Feature Detection:** The scale-space pyramid helps in identifying areas of interest where keypoints are likely to be found. BRISK uses a difference-of-Gaussian (DoG) approach to detect these regions across the scale-space pyramid.

- **Descriptor Generation:** Once keypoints are identified, BRISK samples points around these keypoints using a predetermined pattern. By comparing intensities of these sampled points, it generates binary descriptors. These descriptors are constructed to maintain robustness against various transformations such as scaling and rotation.

- **Binary Descriptors:** BRISK constructs binary strings as descriptors by comparing intensities at sampled points and encoding these comparisons as binary values. These binary strings represent the local image properties around each keypoint in a concise and efficient manner.

Advantages of BRISK:

1. **Speed:** BRISK is known for its computational efficiency, primarily due to its use of binary descriptors. The binary nature of descriptors allows for faster matching and processing compared to methods using floating-point descriptors like SIFT or SURF.

2. Efficiency: The binary representation of descriptors makes BRISK more memory-efficient compared to methods that use higher-dimensional descriptors, reducing storage requirements.

3. Real-time Applications: BRISK's speed and efficiency make it suitable for real-time applications where processing speed is critical, such as robotics, augmented reality, or real-time object tracking.

Limitations:

1. Accuracy: While BRISK is efficient, it may sacrifice a bit of accuracy and robustness compared to methods like SIFT or SURF, especially in scenarios with significant viewpoint changes or challenging lighting conditions.

2. Descriptor Length: The fixed length of binary descriptors might limit their descriptive capacity compared to the variable-length descriptors of other methods.

BRISK's ability to offer a good compromise between speed, efficiency, and reasonable robustness makes it a popular choice in scenarios where real-time performance is a priority and where computational resources are constrained.

Feature Representation:

Feature representation in computer vision involves transforming raw data, such as images or videos, into a format that highlights meaningful and discriminative information. It aims to extract relevant characteristics or patterns that are essential for understanding, analysing, and making decisions about visual data.

What is Feature Representation?

1. **Features:** In the context of computer vision, features are distinctive, measurable, and often abstract representations of data. They can be simple, like edges or corners, or complex, like textures, shapes, or higher-level semantic attributes.

2. **Representation:** Feature representation involves the process of capturing these features in a structured format, making them suitable for computational analysis and interpretation by algorithms.

Types of Feature Representations:

1. **Low-Level Features:** These encompass basic visual attributes like colour, texture, edges, corners, and gradients. They serve as building blocks for more complex representations.

2. **Mid-Level Features**: These features encapsulate more intricate patterns and structures by combining low-level features. Examples include shapes, contours, textures, and regions of interest.

3. **High-Level Features:** These are abstract, semantic, or contextually rich representations. They might represent objects, scenes, or specific attributes such as facial features or object categories.

Techniques:

1. **Handcrafted Features:** Traditionally, engineers used manually designed feature extraction techniques like Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), or Local Binary Patterns (LBP). These methods involve predefined algorithms to capture specific visual characteristics.

2. **Deep Learning Representations:** With the rise of deep neural networks, especially Convolutional Neural Networks (CNNs), feature learning has become more automated. CNNs learn hierarchical representations through successive layers, automatically extracting features at different levels of abstraction.

Challenges and Considerations:

1. **Dimensionality:** Feature representation often involves handling high-dimensional data. Techniques like dimensionality reduction (e.g., Principal Component Analysis - PCA, t-Distributed Stochastic Neighbor Embedding - t-SNE) help manage and visualize these complex representations.

2. **Robustness and Generalization:** Effective feature representations should be robust to variations like changes in lighting, viewpoint, scale, and occlusions. They should also generalize well across different datasets or scenarios.

3. **Interpretability vs. Complexity:** The trade-off between interpretability and complexity is essential. Deep learning models might learn highly effective representations but could lack interpretability due to their black-box nature.

Importance:

- **Object Recognition:** Features aid in identifying objects within images or videos.

- **Image Classification:** Effective representations are crucial for accurately categorizing images into predefined classes or categories.

- **Segmentation and Localization:** Features help in segmenting images into meaningful parts and localizing specific objects within an image.

In summary, Feature representation in computer vision: extracting key details from visuals for machines to interpret images/videos effectively.

Feature representation in computer vision involves transforming raw data, such as images, into a format that machine learning algorithms can understand and process effectively. Here's a detailed explanation of feature representation, focusing on building a dataset with extracted features, feature vector representation by Bag-of-words, and vector quantization:

Building a dataset with Extracted Features:

1. **Feature Extraction:** Initially, raw images are processed using feature extraction techniques (like SIFT, SURF, or CNN-based methods) to identify distinctive and meaningful patterns or keypoints within the images.

2. **Feature Descriptor Generation:** These techniques generate descriptors or representations of these keypoints. Each descriptor encapsulates information about the local region around a keypoint, capturing its unique characteristics.

3. **Dataset Creation:** A dataset is then constructed by collecting these extracted features and their corresponding descriptors across multiple images. Each image contributes a set of keypoints with associated descriptors to the dataset.

Feature Vector representation by Bag-of-words:

1. **Vocabulary Construction:** The next step involves creating a visual vocabulary or dictionary by applying clustering algorithms (e.g., K-means clustering) to group similar descriptors together. This process results in a set of representative visual words or clusters.

2. **Quantization:** Each descriptor in the dataset is then quantized by assigning it to its closest visual word in the constructed vocabulary. This quantization effectively maps each descriptor to a specific visual word.

3. **Histogram Generation:** For each image in the dataset, a histogram is created based on the frequency of occurrence of visual words. This histogram represents the image in a Bag-of-words (BoW) fashion, where the order or spatial information of words is disregarded, and only the frequency matters.

Vector Quantization:

1. **Dimensionality Reduction:** Vector quantization aims to reduce the dimensionality of the descriptors while preserving their discriminative information. It achieves this by replacing each descriptor with a representative codebook entry or centroid from a pre-defined codebook.

2. **Clustering and Codebook Creation:** Methods like K-means clustering are employed to create a codebook by clustering the descriptors into a set of representative codewords. Each descriptor is then assigned to its nearest centroid in the codebook.

3. **Encoding:** Descriptors are encoded by representing them with the index or label of the nearest centroid in the codebook. This encoding effectively transforms continuous descriptors into discrete representations using the codebook entries.

Vector quantization helps in reducing the memory footprint and computational complexity by representing high-dimensional descriptors with a reduced set of representative codewords.

These techniques in feature representation are pivotal in enabling efficient and effective processing of visual data for various computer vision tasks, including object recognition, image classification, and scene understanding. They bridge the gap between raw pixel data and meaningful representations that machine learning algorithms can utilize for analysis and decision-making.

Feature Classification:

Feature classification in computer vision involves assigning or categorizing extracted features into different classes or categories. Machine learning algorithms play a vital role in this process, where the goal is to train a model that can accurately classify features based on their characteristics. Let's explore three popular classification algorithms used in computer vision: Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Random Forest.

1. Support Vector Machines (SVM):

- **Feature Representation and Hyperplane Optimization:** SVM works by transforming feature data into a high-dimensional space. It aims to find the hyperplane that best separates different classes while maximizing the margin between the nearest data points of each class.

- **Kernel Trick:** SVM can efficiently handle non-linearly separable data by using kernel functions such as polynomial, radial basis function (RBF), or sigmoid. These kernels map the input space into a higher-dimensional space, allowing SVM to perform complex classifications.

- **Handling Imbalanced Data:** SVM can handle imbalanced datasets by assigning different weights to different classes, ensuring better performance in scenarios where classes are not equally represented.

- **Complexity and Scalability:** While SVMs are powerful for high-dimensional data, they might become less efficient for very large datasets due to their computational complexity, especially when using non-linear kernels.

2. k-Nearest Neighbors (KNN):

- **Feature Space and Proximity-based Classification:** KNN relies on the idea that similar features tend to belong to the same class. It doesn't build an explicit model but rather classifies new instances based on their similarity to existing instances in the feature space.

- **Parameter 'k' Selection:** The choice of 'k', the number of nearest neighbors considered, is crucial. A smaller 'k' might lead to overfitting, while a larger 'k' might introduce bias.

- **Distance Metrics:** KNN uses distance metrics (Euclidean, Manhattan, etc.) to measure the similarity between feature points. Choosing an appropriate distance metric is crucial for accurate classification.

- **Scalability:** KNN can suffer from computational inefficiency, especially with larger datasets, as it requires computing distances for every new data point against all existing points during classification.

3. Random Forest:

- **Ensemble of Decision Trees:** Random Forest constructs multiple decision trees by randomly selecting subsets of features and data samples. Each tree independently classifies input features, and the final prediction is determined through voting or averaging of individual tree predictions.

- **Bootstrap Aggregating (Bagging):** Random Forest uses bagging, a technique that introduces randomness by training each tree on a different subset of the dataset. This reduces overfitting and enhances the model's generalizability.

- **Feature Importance:** Random Forest can evaluate feature importance based on how much each feature contributes to reducing classification error across trees. This information can help identify relevant features in the dataset.

- **Robustness:** Random Forest is robust against overfitting and noise in the data. It can handle large datasets efficiently and generally performs well without much hyperparameter tuning.

Comparison:

- **SVM** is effective in high-dimensional spaces, suitable for scenarios with clear margins of separation between classes.

- **KNN** is simple and versatile but might be computationally expensive for large datasets due to its reliance on distance calculations for every prediction.

- **Random Forest** is robust, handles large datasets well, and tends to offer high accuracy due to its ensemble approach.

The choice among these algorithms depends on factors such as dataset size, feature characteristics, computational resources, and desired accuracy. Each algorithm has its strengths and weaknesses, making them suitable for different types of feature classification tasks in computer vision.

Unit 5

Image Classification:

Image Classification: It's the process of extracting information from the images and labelling or categorizing the images. There are two types of classification:-

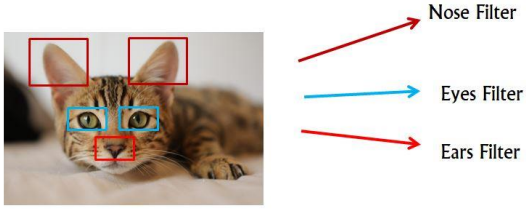
1. **Binary classification**: In this type of classification our output is in binary value either 0 or 1, let's take an example that you're given an image of a cat and you have to detect whether the image is of cat or non-cat.
2. **Multi-class classification**: In this type of classification our output will be multi-class, so let's take an example that you're given an image and you have to detect the breed of dog among 37 classes.

In **computer vision**, we have a **convolutional neural network** that is very popular for computer vision tasks like **image classification, object detection, image segmentation** and a lot more.
Image classification is one of the most needed techniques in today's era, it is used in various domains like healthcare, business, and a lot more, so knowing and making your own state of the art computer vision model is a must if you're in a domain of AI.

1. Convolution Neural Network (CNN):

A convolutional neural network (CNN) is a type of neural network for working with images, This type of neural network takes input from an image and extract features from an image and provide learnable parameters to efficiently do the classification, detection and a lot more tasks.

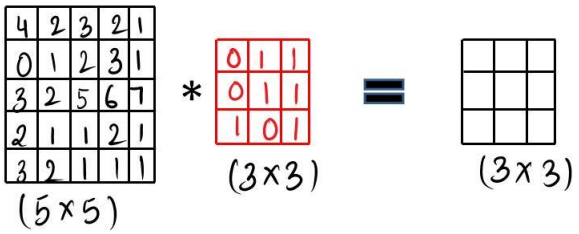
We extract the features from the images using something called "filters", we have different filters used to extract different features from the images.
Let's take an example, you are building a classification model which detects whether an image is of cat or non-cat. So we have different filters used to extract different features from an image like in this case, one filter may learn to detect the eyes of a cat another learn to detect ears and etc.



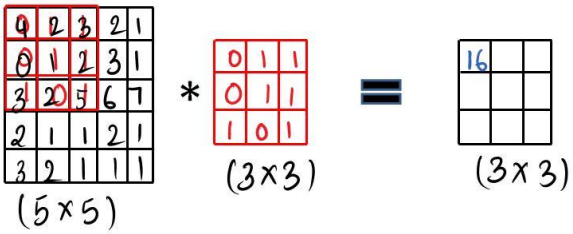
How do we extract the information using these filters?

We convolute our image using filters using convolution operations, Confused??, let's see in detail with some visualization.

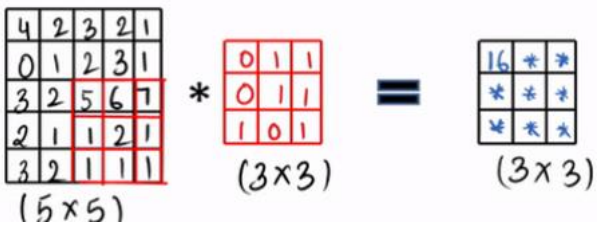
Given:- We take our image (5 x 5), here we have greyscale image and then we take our learnable filters (3 x3) and then we do the convolution operation.



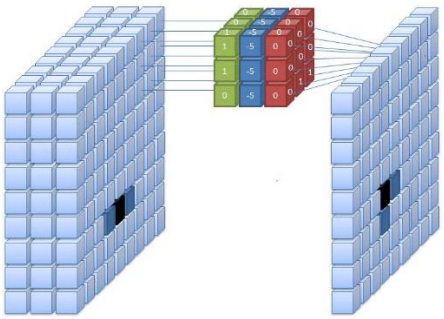
Step:1:- You do the element wise product and then you sum it all up and then you fill the first cell. [4 0 + 1 2 + 1 3 + 0 0 + 1 1 + 2 1 + 3 1 + 2 0 + 5 1 = 16]



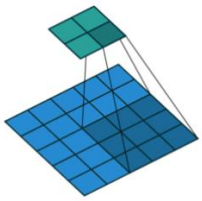
Then you slide by a factor of 1 and again you do the same thing which is called the convolution operation by just doing element-wise product and sum it up.



You may ask a question how can we do for RGB scale or colourful image, you have to do the same instead of your number of channels.

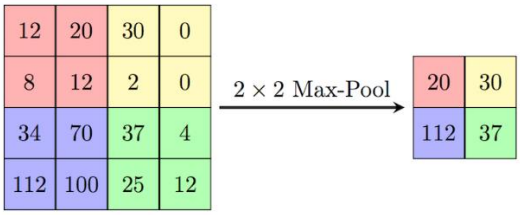


1. Stride Convolutions:
In the above examples, we are sliding over our images with the factor of 1, so for faster computation over the images, so in the below example we are sliding over the image with the factor of 2.



2. Padding:
In convolution operation, we often lose some information, so for preserving the information, we pad our image with zeros and then we start convoluting our image.

3. Pooling Layer: In order to down-sample images while preserving information, we use pooling layers, we have two types of pooling layers which are max-pooling and average pooling.



In the above image, we are doing max-pooling, and also if you want to use average pooling then you can take average instead of max.

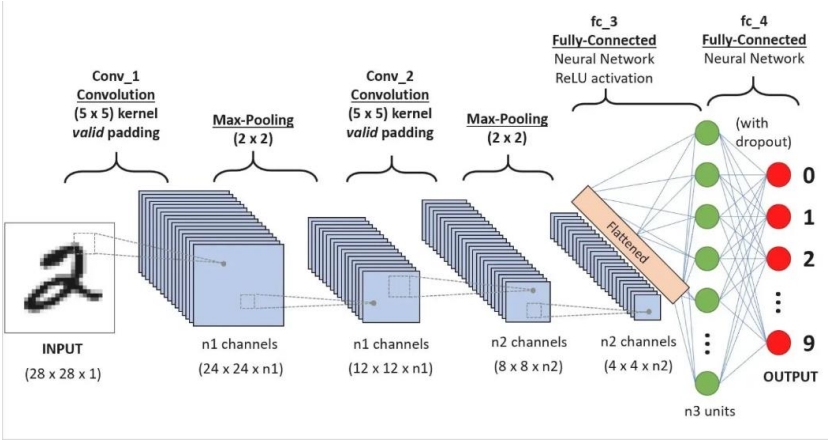
4. Up-Sampling layer:
In order to Up-Sample or make your image large you use these types of layers, it often sometimes blur your image or other disadvantages.

5. Get to know the dimensions: After your image is convoluted then how you will get to know the dimension, so here is the formula for calculating the dimension of your image after convoluting:

$$((n-f+2p)/s) + 1$$

- n is the size of the input, if you have a 32x32x3 image then n will be 32.
- f is the size of the filer, if your size of the filter is 3×3, then f will be 3.
- p is the padding.
- s is the factor by which you want to slide

Working:



- **Convolutional Layers:** CNNs are designed to automatically learn hierarchical representations from images. Convolutional layers apply learnable filters across the input image, capturing features like edges, textures, and patterns. These filters slide (convolve) over the image, performing element-wise multiplications and aggregating values to create feature maps. Deeper layers learn complex features by combining lower-level representations.

- **Pooling Layers:** After convolution, pooling layers down-sample the feature maps, reducing their spatial dimensions. Max pooling, for instance, retains the maximum value within each pooling window, preserving important features while reducing computation. Pooling helps in achieving translation invariance by focusing on essential features and discarding less important ones.

- **Fully Connected Layers:** The final layers of a CNN typically comprise fully connected layers responsible for classification. These layers take the flattened output of the previous layers and map them to output classes using activation functions like SoftMax. They learn to distinguish between different classes based on the features extracted in earlier layers.

CNNs have revolutionized image classification by automatically learning hierarchical features from raw pixels, eliminating the need for handcrafted feature extraction and achieving state-of-the-art performance in various computer vision tasks.

2. **Attention Models:**

- **Self-Attention Mechanism:** Attention mechanisms, especially in models like Transformers, allow the network to weigh different parts of the input image differently during processing. Self-attention enables the network to focus more on relevant regions by assigning varying importance to different locations within the image, aiding in capturing long-range dependencies and relationships.

- **Multi-Head Attention:** Multi-head attention employs multiple sets of attention mechanisms to extract different perspectives or representations of the input image. These multiple heads help in capturing various aspects of the image simultaneously, enhancing the model's understanding and improving performance.

- **Hierarchical Attention:** Some attention models incorporate hierarchical attention to capture both local and global context within an image. This enables the model to discern relationships between different regions at multiple scales, facilitating better understanding and classification.

Attention models excel in capturing complex dependencies and contextual information within images, allowing for more precise and nuanced classification by focusing on relevant parts of the image.

3. **Vision Transformation:**

- **Data Augmentation:** This technique involves generating additional training data by applying various transformations to the images, such as rotation, scaling, translation, and flipping. Data augmentation helps in increasing the diversity of the training dataset, thereby improving the model's ability to generalize to unseen data and reducing overfitting.

- **Normalization and Preprocessing:** Preprocessing steps, including normalization (scaling pixel values to a standard range) and mean subtraction (subtracting mean pixel values), ensure that the input data is standardized. This preprocessing aids in better convergence during training and improves the model's ability to learn useful features.

- **Transfer Learning:** Transfer learning involves leveraging pre-trained models that have been trained on large datasets like ImageNet. These models possess learned features that can be fine-tuned or used as feature extractors for new image classification tasks. This approach helps in situations where labeled data is limited, allowing the model to benefit from previously learned representations.

Vision transformation techniques play a vital role in improving the robustness, generalization, and efficiency of image classification models by providing variations in the training data, optimizing input data, and leveraging knowledge learned from pre-trained models.

Each of these components—CNNs, attention models, and vision transformation techniques—adds significant value to image classification tasks, contributing distinct strengths and capabilities to enhance the accuracy and efficiency of classification models in computer vision.

Generative Adversarial Network (GAN):

A Generative Adversarial Network (GAN) is a type of generative deep learning model that consists of two neural networks: a generator network and a discriminator network.

The generator network takes a random noise vector as input and generates a new sample in the data space, while the discriminator network takes a sample from either the training data or the generator network and classifies it as real or fake.

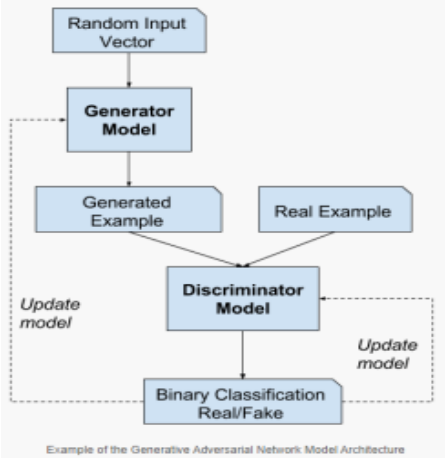
The two networks are trained together in an adversarial process, where the generator network tries to generate samples that are indistinguishable from real data samples, while the discriminator network tries to correctly distinguish between real and fake samples.

The generator network is trained to maximize the error of the discriminator network on the generated samples, while the discriminator network is trained to minimize the error between the real and generated samples.

As the two networks are trained together, the generator network learns to generate increasingly realistic samples, while the discriminator network becomes better at distinguishing between real and fake samples. The ultimate goal of the GAN is to generate new samples that are indistinguishable from the real data distribution.

GANs have been successfully applied to a wide range of tasks, including image and video synthesis, text generation, and data augmentation. Some examples of GAN applications include generating photorealistic images, creating realistic 3D models, and generating synthetic data to augment training sets for machine learning models.

GANs are a powerful tool in the deep learning arsenal, but they can also be challenging to train due to their complex architecture and the instability of the training process. However, with careful tuning and regularization techniques, GANs can produce impressive results.



Example of the Generative Adversarial Network Model Architecture

Advantages of GANs in Deep Learning:

1. **Generative Modelling:**
   GANs excel at generative modelling tasks by learning to capture complex patterns and distribution of the training data. They can generate new samples that closely resemble the original data distribution, providing a valuable tool for data augmentation, creativity, and synthesis.

2. **Realistic Outputs:**
   GANs are known for producing outputs that are highly realistic and indistinguishable from real samples. This makes them suitable for applications like image synthesis, video generation, or audio generation, where generating high-quality and visually appealing content is crucial.

3. **Unsupervised Learning:**
   GANs operate in an unsupervised learning framework, meaning they don't require labelled data during training. They learn directly from the unlabelled training data and can discover underlying patterns and structure without explicit supervision. This makes GANs useful in scenarios where labelled data is scarce or costly to obtain.

4. **Feature Learning:**
   GANs learn to extract meaningful representations and features from the data during the training process. The generator and discriminator networks develop intricate internal representations, which can be leveraged for transfer learning or fine-tuning on downstream tasks. This feature learning capability of GANs is valuable for various computer vision and natural language processing tasks.

Disadvantages of GANs in Deep Learning:

1. **Training Instability:**
   GANs can be challenging to train and prone to instability. Finding the right balance between the generator and discriminator networks is crucial. The training process often involves fine-tuning hyperparameters, such as learning rates, network architectures, and regularization techniques. In some cases, GANs can suffer from mode collapse, where the generator fails to capture the full diversity of the target distribution.

2. **Mode Dropping:**
   GANs may produce samples that only represent a subset of the target distribution, ignoring some important modes or variations present in the training data. This can limit their diversity and generate biased outputs that do not fully capture the complexity of the real data distribution.

3. **Evaluation Challenges:**
   Evaluating GANs is a non-trivial task. Traditional evaluation metrics like accuracy or loss functions may not capture the quality or diversity of the generated samples effectively. Metrics such as Inception Score or Fréchet Inception Distance (FID) have been proposed, but they have their own limitations and may not provide a comprehensive evaluation.

4. **Computationally Intensive:**
   GANs, especially deep convolutional architectures, can be computationally demanding and require significant computational resources, memory, and training time. Training large-scale GANs on high-resolution images or complex datasets can be resource-intensive and may require specialized hardware or distributed computing setups.

5. **Lack of Interpretability:**
   GANs are considered black box models, meaning it can be challenging to interpret the internal workings of the generator or discriminator networks. Understanding the specific features or representations that lead to the generation of specific outputs is a complex task, limiting the interpretability of the model.

Object Detection:

Object detection in computer vision involves identifying and locating objects within an image or video. Various methods have evolved to achieve this, each with its own approach and trade-offs. Let's delve into the key concepts of several object detection techniques:

Regions with CNN (R-CNN):

1. **Region Proposal:**
   - R-CNN operates through a two-stage process. Initially, it employs a selective search algorithm to propose potential regions within an image that might contain objects. These regions are known as region proposals.

2. **CNN Feature Extraction:**
   - Once the regions are proposed, R-CNN crops each region from the image and passes it through a convolutional neural network (CNN) to extract features. These features serve as representations for the objects within each proposed region.

R-CNN was a groundbreaking method for object detection, although it's computationally expensive due to its two-stage approach, making it slower for real-time applications.

Fast R-CNN:

1. **Region of Interest (RoI) Pooling:**
   - Fast R-CNN addresses the inefficiency of R-CNN by introducing RoI pooling. It enables feature extraction from region proposals without individually cropping regions. Instead, it aligns features within a fixed-size feature map, reducing computational overhead.

2. Single Network:
   - Unlike R-CNN, Fast R-CNN shares the CNN computation across all proposed regions, making it more efficient and faster.

Fast R-CNN significantly improves speed compared to R-CNN by eliminating redundant computations, resulting in a more streamlined and faster object detection process.

Faster R-CNN:

1. **Region Proposal Network (RPN):**
   - Faster R-CNN introduces the RPN, a fully convolutional network that generates region proposals directly from the image. It replaces the external algorithms used for region proposal generation in R-CNN and Fast R-CNN.

2. **Integration of RPN and Fast R-CNN:**
   - Faster R-CNN integrates the RPN and the Fast R-CNN detection network into a single unified architecture, enabling end-to-end training and improving both speed and accuracy.

Faster R-CNN further optimizes object detection by unifying the region proposal and object detection stages, resulting in a more efficient and accurate system.

Mask R-CNN:

1. **Mask Prediction:**
   - Mask R-CNN extends Faster R-CNN by adding a mask prediction branch. In addition to predicting bounding box coordinates and class labels, it also predicts segmentation masks for each detected object, providing precise pixel-level segmentation.

2. **Three Outputs:**
   - Mask R-CNN outputs three pieces of information for each object: bounding box coordinates, class labels, and segmentation masks.

Mask R-CNN is especially valuable in applications requiring precise segmentation alongside object detection, such as instance segmentation tasks.

SSD (Single Shot MultiBox Detector):

1. **Default Boxes at Multiple Scales:**
   - SSD predicts bounding boxes and class probabilities by applying convolutional filters to multiple feature maps at different scales. It uses default boxes of various aspect ratios to detect objects of different sizes.

2. **Efficient Single-Shot Detection:**
   - SSD performs object detection in a single pass through the network, directly predicting object locations and categories without the need for separate region proposal stages.

SSD is known for its efficiency and is capable of real-time object detection, suitable for applications where speed is crucial.

YOLO (You Only Look Once):

1. **Grid-based Prediction:**
   - YOLO divides the image into a grid and directly predicts bounding boxes and class probabilities within each grid cell. This approach allows the model to make predictions at different spatial locations in a single pass.

2. **Single Pass Detection:**
   - YOLO processes the entire image in one pass through the network, which makes it extremely fast compared to methods that involve multiple stages or regions.

YOLO stands out for its speed and has been widely adopted in applications requiring real-time object detection, such as video analysis and surveillance systems.

These object detection techniques vary in their approach, trade-offs, and suitability for different applications. They've been developed to address challenges like speed, accuracy, and computational efficiency, catering to diverse requirements in various computer vision tasks. The choice of method depends on the specific needs of the application, considering factors like speed, accuracy, and available computational resources.

Semantic Segmentation using U-Net:

What is Semantic Segmentation?

There are various levels of granularity in which the computers can gain an understanding of images. For each of these levels there is a problem defined in the Computer Vision domain. Starting from a coarse grained down to a more fine-grained understanding, let's describe these problems below:

a. Image classification:

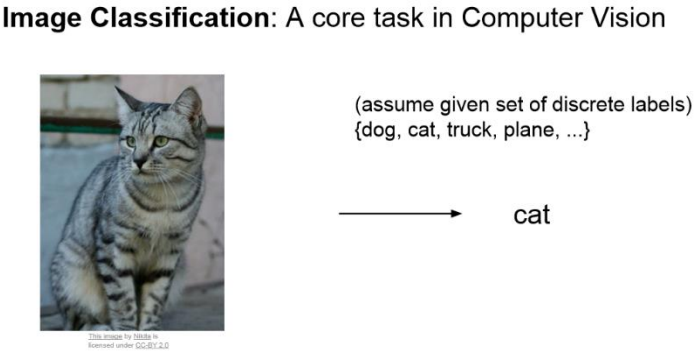
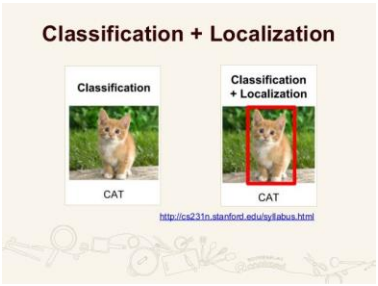


Image Classification

The most fundamental building block in Computer Vision is the Image classification problem where given an image, we expect the computer to output a discrete label, which is the main object in the image. In image classification we assume that there is only one (and not multiple) object in the image.

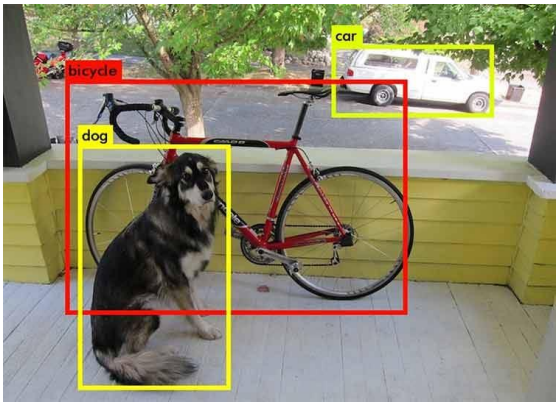
b. Classification with Localization:



Classification with localization

In localization along with the discrete label, we also expect the compute to localize where exactly the object is present in the image. This localization is typically implemented using a bounding box which can be identified by some numerical parameters with respect to the image boundary. Even in this case, the assumption is to have only one object per image.

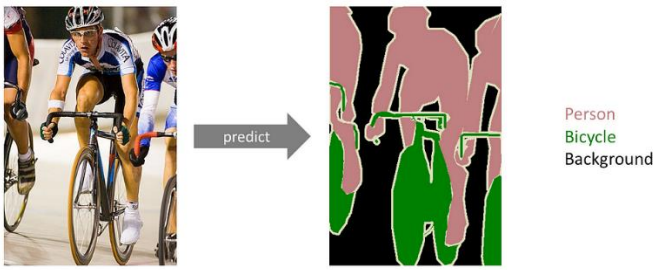
c. Object Detection:



Object Detection

Object Detection extends localization to the next level where now the image is not constrained to have only one object, but can contain multiple objects. The task is to classify and localize all the objects in the image. Here again the localization is done using the concept of bounding box.

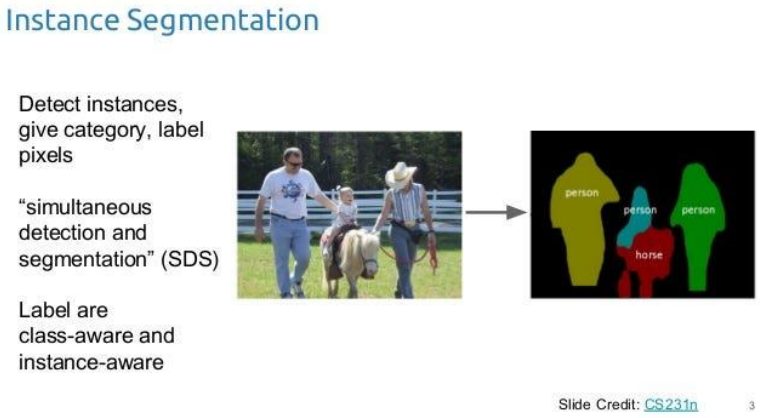
d. Semantic Segmentation:



Semantic Segmentation

The goal of semantic image segmentation is to label each **pixel** of an image with a corresponding **class** of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as **dense prediction**. Note that unlike the previous tasks, the expected output in semantic segmentation are not just labels and bounding box parameters. The output itself is a high-resolution image (typically of the same size as input image) in which each pixel is classified to a particular class. Thus it is a pixel level image classification.

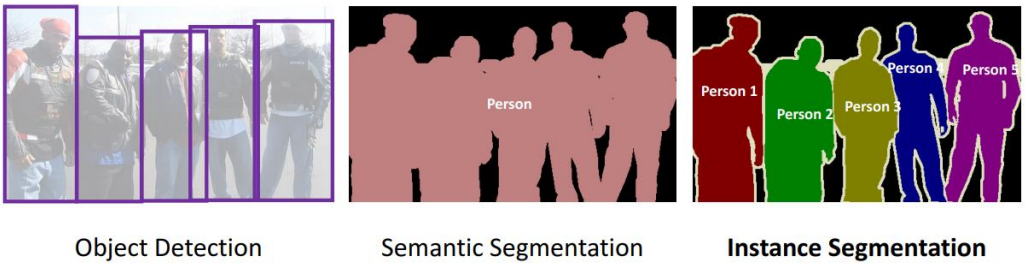
e. Instance Segmentation:



Instance Segmentation

Instance segmentation is one step ahead of semantic segmentation wherein along with pixel level classification, we expect the computer to classify each instance of a class separately. For example in the image above there are 3 people, technically 3 instances of the class "Person". All the 3 are classified separately (in a different colour). But semantic segmentation does not differentiate between the instances of a particular class.

If you are still confused between the differences of object detection, semantic segmentation and instance segmentation, below image will help to clarify the point:



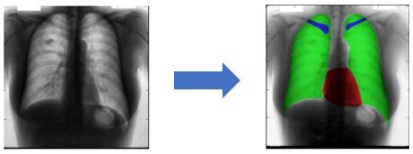
Object Detection vs Semantic Segmentation vs Instance Segmentation

In this post we will learn to solve the Semantic Segmentation problem using Fully Convolutional Network (FCN) called UNET.

Applications:
If you are wondering, whether semantic segmentation is even useful or not, your query is reasonable. However, it turns out that a lot of complex tasks in Vision require this fine-grained understanding of images. For example:

a. Autonomous vehicles:
Autonomous driving is a complex robotics tasks that requires perception, planning and execution within constantly evolving environments. This task also needs to be performed with utmost precision, since safety is of paramount importance. Semantic Segmentation provides information about free space on the roads, as well as to detect lane markings and traffic signs.

b. Bio Medical Image Diagnosis:
Machines can augment analysis performed by radiologists, greatly reducing the time required to run diagnostic tests.



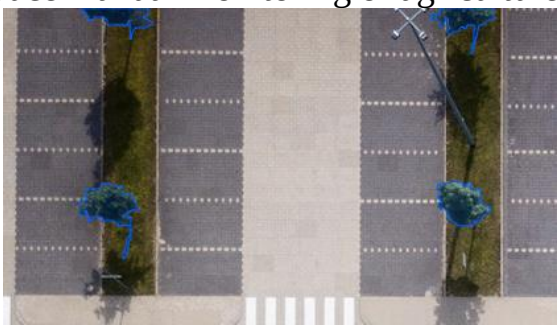
c. Geo Sensing:
Semantic Segmentation problems can also be considered classification problems, where each pixel is classified as one from a range of object classes. Thus, there is a use case for land usage mapping for satellite imagery. Land cover information is important for various applications, such as monitoring areas of deforestation and urbanization.

To recognize the type of land cover (e.g., areas of urban, agriculture, water, etc.) for each pixel on a satellite image, land cover classification can be regarded as a multi-class semantic segmentation task. Road and building detection is also an important research topic for traffic management, city planning, and road monitoring.

There are few large-scale publicly available datasets (Eg : SpaceNet), and data labelling is always a bottleneck for segmentation tasks.

d. Precision Agriculture

Precision farming robots can reduce the amount of herbicides that need to be sprayed out in the fields and semantic segmentation of crops and weeds assist them in real time to trigger weeding actions. Such advanced image vision techniques for agriculture can reduce manual monitoring of agriculture.



We will also consider a practical real world case study to understand the importance of semantic segmentation. The problem statement and the datasets are described in the below sections.

UNET Architecture and Training:

U-Net Architecture:

The U-Net architecture consists of a contracting path (encoder) followed by an expansive path (decoder). It's characterized by a U-shaped structure, which is where its name originates.

1. Encoder (Contracting Path):

- **Convolutional Blocks:** The encoder consists of several convolutional layers organized in blocks. Each block typically contains multiple convolutional layers followed by pooling layers (like max-pooling) to down-sample the spatial dimensions while increasing the number of feature channels.
- **Down-sampling:** These operations help extract hierarchical features by progressively reducing the spatial resolution of the input image.

2. Decoder (Expansive Path):

- **Up-sampling:** The decoder path begins by performing up-sampling operations, which increase the spatial dimensions while decreasing the number of channels. This process aims to recover the spatial information lost during the down-sampling phase.

- **Concatenation:** At each up-sampling step, the decoder merges feature maps from the encoder's corresponding contracting path. This step allows the decoder to leverage low-level and high-level feature representations for accurate segmentation.

- **Convolutional Blocks:** The decoder contains convolutional layers that help refine and learn the fine-grained details necessary for precise pixel-level classification.

3. Skip Connections:

- U-Net introduces skip connections between the encoder and decoder paths. These connections concatenate feature maps from the contracting path to the expanding path. They enable the network to combine low-level and high-level feature information, aiding in preserving spatial details during the up-sampling process.

4. Final Layer:

- The final layer of the U-Net typically uses a convolutional layer with a SoftMax activation function to produce a segmentation map. Each pixel in the output corresponds to a specific class, providing a pixel-wise classification of the input image.

Training U-Net for Semantic Segmentation:

Training U-Net for semantic segmentation involves:

- **Loss Function:** Often, cross-entropy loss is used to measure the discrepancy between the predicted segmentation map and the ground truth labels.

- **Optimization:** Gradient-based optimization techniques like Adam or SGD are employed to update the network's weights and minimize the loss function.

- **Data Augmentation:** To prevent overfitting and enhance generalization, data augmentation techniques like random rotations, flips, and scaling are applied to increase the diversity of the training dataset.

Advantages of U-Net:

- **Preservation of Spatial Information:** U-Net's skip connections aid in retaining spatial details during the up-sampling process, enabling accurate segmentation.

- **Efficient Learning:** The architecture efficiently learns both global and local features, crucial for semantic segmentation tasks.

- **Applicability:** U-Net's versatility makes it adaptable to various domains and types of imagery, including medical imaging and satellite imagery analysis.

U-Net has proven effective in numerous segmentation tasks, especially where pixel-level accuracy and preservation of spatial information are paramount. Its architecture and design principles have served as a basis for many subsequent segmentation models.

Centroid-Based Object Tracking:

Centroid-based object tracking is a method used in computer vision to follow and monitor objects within a video or a sequence of images. It focuses on identifying and continuously updating the centroids, or centre points, of objects of interest throughout successive frames.

Here's a detailed breakdown of how centroid-based object tracking typically works:

Object Detection:

1. Initialization: The process begins with an initial step of detecting objects in the first frame of the video or image sequence. This can be achieved using various object detection algorithms like Haar cascades, HOG (Histogram of Oriented Gradients), or deep learning-based methods like YOLO (You Only Look Once) or SSD (Single Shot Multibox Detector).

2. Bounding Box Extraction: Once an object is detected, a bounding box is placed around it to define its spatial extent within the frame. The centroid of this bounding box is initially considered as the object's centroid.

Centroid Computation:

1. **Centroid Calculation:** The centroid of an object is determined by computing the average position of all pixels within the bounding box. Mathematically, it's the average of x and y coordinates of all the pixels within the bounding box.

Tracking Across Frames:

1. **Frame-to-Frame Tracking:** As the video progresses, subsequent frames are analysed to track the movement of the objects. The bounding boxes around the objects are updated in each frame, and the centroids are recalculated based on the updated bounding boxes.

2. **Centroid Association:** To maintain continuity in tracking, the centroids from the previous frame are associated with the closest centroids in the current frame. Techniques like nearest neighbor search or the Hungarian algorithm may be used for this association.

3. **Motion Prediction:** Some tracking algorithms incorporate motion prediction models to estimate the likely positions of centroids in the next frame based on the object's velocity or trajectory. This prediction helps in reducing errors caused by sudden movements or occlusions.

4. **Handling Occlusions and Object Changes:** Challenges like occlusions (where objects are temporarily hidden) or object appearance/disappearance are addressed using methods like Kalman filters, particle filters, or more advanced deep learning-based tracking models to maintain track continuity.

Challenges and Considerations:

- **Occlusions:** When an object is partially or completely hidden by another object, the tracker might lose the object's position. Various methods like multi-object tracking or re-detection can help in recovering the track.

- **Object Deformation or Appearance Changes:** Objects might change shape, appearance, or size, posing challenges for trackers that rely heavily on appearance-based features. Robust tracking algorithms are needed to handle these variations.

Applications:

Centroid-based object tracking finds applications in:

- **Surveillance:** Tracking individuals or vehicles in surveillance videos.

- **Augmented Reality:** Keeping track of objects in real-time for AR applications.

- **Autonomous Vehicles:** Monitoring and tracking other vehicles or pedestrians for safe navigation.

Centroid-based object tracking serves as a fundamental approach in computer vision, forming the basis for more advanced tracking algorithms. Its simplicity and efficiency make it a foundational method for various real-world applications.