Date / /
Page L&W

* Computer Vision
- is a subfield of Artificial Intelligence (AI) and machine learning focused on enabling machines to understand and interpret visual data

Goal:
- To automate tasks the human visual system can do like detecting objects, reading text, recognizing faces, or understanding scenes.

• How does it works?
- Input: Image or video
- Processing: Algorithms (statistical or deep learning) extract features
- Output: Classification, detection, segmentation, or understanding.

* The Three R's of computer vision
These are fundamental tasks that CV systems perform:

a] Recognition
- Task: Identify what is present in an image
ex: Image classification, Object detection, face recognition
algo:- CNN, YOLO, ResNet

b] Reconstruction
Task: Rebuild a 3D model or scene from 2D images.

ex. Depth estimation, 3D object reconstruction from multiple views, SLAM (Simultaneous Localization and mapping) in robotics

c] Reorganization (or Reorganization)
- Task: Divide and structure an image into meaningful components.
ex. Semantic segmentation (classifying each pixel), Image parsing, Background-foreground separation
algo:- k-means clustering, U-net

* Image Basics

- A digital image can be considered as 2D array of numbers, with each number representing a pixel - the smallest unit of an image

- Grayscale image :- A grayscale image is an image in which each pixel carries only intensity information - no color, It contains shades of gray ranging from 0 to 255

- Color Image: A color image contains multiple color channels (typically Red, Green, Blue - RGB), and each pixel is represented by a combination of color intensities.

- Normalization: in image processing refers to scaling pixel values to a standard range, typically 0 to 1, -1 to 1. It helps improve the performance of machine learning models by making data consistent and comparable.

- Image shape (720, 400, 3)
  - 720 ↑ height. no. of rows
  - 400 ↑ width no. of cols
  - 3 ↑ color channel. (RGB)

- cv2.imread()
  - The function reads these coded instructions
  - It decodes them into something that python can understand
  - Each no. group in this array is an instruction for one pixel.

- plt.imshow()
  - Takes the Numpy array and tells the graphics card to paint each pixel according to its values

* Pillow.
- Initially developed as a friendly fork of PIL (Python Imaging Library)
- Actively maintained by the Python Imaging Library community. • written in python with some extension of C
- Features
  - Image processing: crop, resize, rotate, flip, Filter, convert formats

- Image enhancement: brightness, contrast, sharpness adjustments
- Image creation and drawing: add text, lines, shapes (via ImageDraw)
- Supports many formats: JPG, PNG, GIF etc.
- lightweight and easy to integrate in GUI or webapps

* openCV

- written in C++, with bindings for Python, Java, MATLAB etc.
- Highly optimized with CUDA and OpenCL support for real-time applications

Features
- Image processing
- video analysis
- feature detection
- object detection
- Camera calibration and 3D reconstruction
- machine-learning support.

* HSV (Hue, saturation and value.
- It is an alternative color representation to RGB, designed to be more intuitive for how humans perceive and describe ~~below~~ colors

a) Hue (H): Represents the pure color itself
ex 0° = Red 60° = Yellow 120° = Green
180° = Cyan 240° = Blue 300° = Magenta

b] saturation (S) : Represents the purity or intensity of the color. Indicates how much of the color is mixed with white or gray
0 = (gray) 255 = full color.

c] Value (V) : - Brightness of the color
(0 = pure black, 255 = maximum brightness)

* Image Resizing
- means changing the dimensions (width x height) of an image)
- when resizing an image, especially enlarging or shrinking it, you need to estimate pixel values at new positions - this is done using interpolation

- Interpolation Methods
(a) cv2.INTER_NEAREST - copies the color of the closest original pixel.
(b) cv2.INTER_LINEAR - Bilinear Interpolation - computes weighted avg of 4 nearest pixels
(c) cv2.INTER_CUBIC - Bicubic Interpolation - considers 16 pixels and perform cubic convolution
(d) cv2.INTER_AREA - Resampling using pixel area relation - avg all input pixels contributing to output pixels

Date / /
Page

Nearest Neighbor Interpolation

ex original 3×3 → 6×6

for pixel (3×2) in the new image map to original co-ordinates.

$$x = \frac{\text{new } x}{\text{Scale of } x'} \qquad y = \frac{\text{new } y}{\text{Scale of } y}$$

$$x = \frac{3}{2} = 1.5 \qquad y = \frac{2}{2} = 1$$

rounded (2, 1)

Pixel value at (2,1) from the original is copied to (3,2) in the new image

* Image cropping

- Image cropping is the process of removing unwanted outer areas from an image by selecting a specific rectangular region
- Numpy arrays allow us to select portions (slices) of the array using the [start : stop] notation.
- In open cv: (height, width, channels) = (rows, columns, channels)
  - Dimension 1 = rows (y-axis or height)
  - Dimension 2 = colums (x-axis or width)
  - Dimension 3 (if present) = color channels

Date / /
Page L&W

* Image Rotating
- Image rotation is the process of turning or spinning an image around a central point (typically its center) by a specific angle, measured in degrees

- Why image rotation
  - For data augmentation in machine learning (to train on varied image orientations)
  - To apply geometric transformations in object tracking or graphics
  - To align scanned documents or photos

- In 2D rotation (about the origin), a point (x, y) is rotated by angle θ using the matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- cv2.rotate() :- Rotates image by fixed angles (90, 180 or 270 degrees). No need for complex math or interpolation

- cv2.getRotationMatrix2D + cv2.warpAffine() :- getRotationMatrix2D(center, angle, scale) creates a 2x3 matrix M, then warpAffine() applies it to image.

* Affine Transformation
- An Affine Transformation is a geometric transformation that preserves:
  • points • straight lines • parallelism (but not angles or lengths)
- A way to change the shape or position of an image or object without breaking its basic structure
- widely used in operations like:
  • Rotation
  • Translation (shifting)
  • Scaling (resizing)
  • Shearing (slanting)

* Image blurring
- is the process of reducing sharp edges and noise in an image by averaging the pixel values with neighboring pixels.
It is commonly used to:
  - Smooth images
  - Reduce noise
  - Preprocess for edge detection
  - Hide details or apply artistic effects

- Types

a) Average blur :- Replaces each pixel with the average of its neighbors
- cv2.blur()
- Each method applies a kernel (small matrix
- Be:

like 3x3 or 5x5) that moves across the image.
- In average mean it computes mean within kernel.
- Best for :- Basic smoothing, reducing uniform noise

b] Gaussian blur : Weighted average (gives more weight to center pixels); reduces noise better
- cv2.GaussianBlur()
- uses a bell-shaped Gaussian filter
- Best for :- Reducing gaussian noise, image smoothing

c] Median Blur : Replaces each pixel with the median of its neighborhood; effective for salt-and-pepper noise
- cv2.medianBlur()
- picks the median value.
- Best for :- Removing salt and pepper noise

d] Bilateral Filter : Blurs while preserving edges; good for facial filtering
- cv2.bilateralFilter()
- filters both spatially and across intensity differences

* Image sharpening
- Image sharpening is a process used to enhance the edges and fine details in an image by increasing the contrast betn neighboring pixels
- It makes images appear clearer and more efficient defined, especially useful when an image looks blurry or soft.

- Uses convolution, where a small matrix called a kernel (or filter) slides over each pixel of the image.
- The kernel contains weights. The center pixel and its neighbors are multiplied by these weights, and the results are summed up to determine the new value for the centre pixel.

Note:-

- Noise Amplification
- If the image already has noise (random dots or graininess), sharpening can make it look even more noisy. It might highlight the noise instead of improving the image.

- Unwanted Effects (Artifacts)
- Too much sharpening can create weired glow-like lines or halo effects around edges. This makes the image look unnatural.

- Try Different settings.
- There are many types of sharpening filters (called kernels).
- You may need to experiment with diff values and sizes to get the best result for each image.

Date / /
Page L&W

Thresholding

- Thresholding is a technique in image processing that converts a grayscale image into a binary image (only black and white) based on a threshold value.
- For each pixel:
  - If the pixel value is greater than the threshold, @set it to white (255)
  - If it's less, set it to black (0)
- Powerful form of image segmentation, helping to separate the foreground (objects of interest) from the background based on pixel intensity value.
- Types:
  - Simple thresholding: A single, global threshold values is applied to the entire image - works best when the image has uniform lighting
  - Adaptive thresholding: The threshold is calculated dynamically for smaller regions of the image, making it effective for images with varying lighting conditions.
- Main thresholding methods
  - Binary thresholding
    - convuerts a grayscale image into a binary image.
    - All pixel values above a certain threshold are set to a maximum value (255) and all pixel values below that threshold are set to 0.
  - Otsu's method:- Assume all possible thresholds:

- For each threshold t, divide the pixels into:
- For each threshold t, divide the pixels into:
  - class 1 (background pixels): pixels ≤ t
  - class 2 (foreground pixels): pixels > t

* Object Detection

- Object Detection is a computer vision technique that locates and identifies multiple objects in an image or video.
- It not only classifies what is in the image but also draws bounding boxes around each object

- Techniques
  - Template matching: Searches for an exact or near-exact match of a template image within a larger image
  - Edge detection: Detects significant transitions in intensity - highlights object boundaries (e.g., canny, sobel)
  - Corner detection: Identifies points in the image where edges intersect - useful for tracking and matching (e.g. Harris, shi-Tomasi)
  - Grid detection: combines edge and corner detection to find grid-like patterns or structures (e.g. chessboard calibration patterns)
  - Contour detection: Identifies continuous curves along boundaries of objects - useful for segmenting foreground vs. background and

internal parts (e.g. detecting eyes/mouth in a cartoon face)

- Feature matching: Matches local keypoints (e.g SIFT, ORB, AKAZE) to detect objects even under changes in scale, rotation or lighting

Template matching

- How it works:
  1. Take a template image (e.g. a small cut-out of an object)
  2. Slide it over the main image one pixel at a time
  3. At each position, compute a similarity score b/n the template and that region
  4. The best match is where the score is highest (or lowest depending on the method)

- Template matching methods
  1. cv2.TM_CCOEFF - Correlation coefficient (higher is better)
  2. cv2.TM_CCOEFF_NORMED - Normalized version (0 to 1)
  3. cv2.TM_CCORR - Cross-correlation
  4. cv2.TM_CCORR_NORMED - Normalized cross-correlation
  5. cv2.TM_SQDIFF - Squared difference (lower is better)
  6. cv2.TM_SQDIFF_NORMED - Normalized squared difference

- Template Matching limitations and solutions
  1. Same Size and Angle Needed

  Template matching works only when the object in the image is the same size and facing the same direction as the template.

2. Not Good for Real-World Images
It fails if the object is rotated, zoomed in/out, lit differently, or partially hidden.

3. No smart Understanding
- It just compares pixels, not what the object is. For ex, it won't recognizes different faces as the same kind of object

- Alternatives
  - Feature-based matching (ORB, SIFT, SURF)
  - Object classification (CNN, YOLO, SSD)
  - Fast but approximate (Haar Cascades)

b] Edge Detection.
- Edges are where the brightness (intensity) of the image changes sharply - like the outline of a face, object, or shape.

- Common Techniques.
(a) Sobel :- Detects edges in horizontal/vertical direction
cv2.Sobel()
(b) Prewitt :- Like Sobel but simpler (not in OpenCV by default())
(c) Laplacian :- Detects edges in all directions
cv2.Laplacian()
(d) Canny :- Most accurate and widely used
cv2.Canny()

Date / /
Page L&W

(i) Sobel Edge Detection.
- Detects edges in a specific direction.
- Uses first-order derivatives to calculate how pixel intensity changes.
- It applies convolution using two 3x3 kernels:
  - One for horizontal edges (x-direction)
  - One for vertical edges (y-direction)
- Sobel x (Horizontal gradient) $G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$
- Sobel Y (vertical gradient) $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$

(ii) Laplacian filter:
- Second-order derivative filter
- Looks for areas where the rate of change of intensity changes - typically edges, lines or corners
- 4 neighbors or 8-neighbor version is used
- 4 neighbor version: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- The center value is (+)ve and ~~(-)ve~~ large
- The surrounding values are (-)ve
- This highlights areas where the center pixel is very diff from its neighbors
- Work
  1. Applies the kernel over the image
  2. Positive response indicates a light-to-dark transition
  3. Negative response indicates a dark-to-light transition

4. zero-crossings often used to identify the exact edge location.

c) Canny Edge Detection
- A multi-step, smart algorithm that gives clean and precise edges.
- Uses gradient-based detection + noise filtering + edge tracking.

Steps involved
1. Noise reduction using gaussian blur
2. Gradient calculation (intensity change)
3. Non-maximum suppression (thin out edges)
4. Hysteresis thresholding (uses two thresholds for strong + weak edge control)

d) Corner detection
- Used to find sharp changes in direction or points where two edges meet - in short, corners
- corner are imp features in a image because they are:
  - Easy to detect
  - stable under transformation
  - Useful in tracking, stitching and 3D vision
- algorithms

(i) Harris: - Detects corners where intensity changes in both x and y directions using image gradients
- Sensitive to strong variations; may detect unstable corners.

(ii) Shi-Tomasi:- Uses the minimum eigenvalue of the gradient matrix to find strong and reliable corners

(iii) FAST (Feature from Accelerated Segment Test):- Tests a circle of 16 pixels around each pixel for brightness differences to identify corners
- Extremly fast; Not scale-invariant; great for real-time application

d] Contour Detection:
- use
  - Finding and outlining objects in an image (Object Detection and localization)
  - Image segmentation: Help in dividing an image into meaningful regions
  - Object Recognition
  - Motion Tracking
  - Foreground Extraction: Isolating objects from their background.

- Contour detection process

(i) preprocessing
  a) grayscale conversion: color image should convert to grayscale first. This simplifies the image by reducing it to a single channel, making it easier to analyze pixel intensity changes.
  b) ~~Edge detection~~ Binary Image conversion: Essential for clear distinction between objects and background:
     (i) Thresholding (ii) Edge detection

(ii) Contour finding algorithm: Use function (like cv2.findContours) to trace these white outlines. This gives you a list of points for each object's boundary.

(iii) Contour representation: Contours are returned as a list of Numpy arrays, where each array contains the (x,y) co-ordinates of the boundary points of an object.

- Drawing contours using cv2.drawContours to visualize the detected outlines on an image.

e] Feature Matching

- Match key features of an image (e.g. corners, edges, contours) rather than pixel-by-pixel comparison.

- Main approaches.

(a) Brute-Force Matching with ORB Descriptions:

- Compares every ORB feature (a binary pattern) from one image to every ORB feature in another, using Hamming distance (counting different bits). The closest match is taken.
- Best for speed and efficiency
- Good for real-time or resource-limited applicat^{n}

(b) Brute-Force Matching with SIFT Descriptors + Ratio Test:

- Compares every SIFT feature (a detailed numerical vector) from one image to every SIFT feature in another, using Euclidean distance. Then, it applies a Ratio Test: -

a match is kept only if its best match is significantly better than its second-best match, removing ambiguous ones.
Best for:- High accuracy and robustness to scale, rotation, and illumination changes.

(c) FLANN Based ~~Master~~ Matcher:
- This is a faster search engine for matches. Instead of comparing everything one-by one, it builds a special index (like a sorted list or tree) of features. It then quickly finds approximate best matches, especially good for large numbers of features. Works with both SIFT (Euclidean distance) and ORB (Hamming distance).
- Best For: Speed when you have many features.

* Cascade classifier.
- A machine-learning based approach that detects objects (e.g. faces) by applying a series of increasingly complex classifiers on image regions.
- Feature Detection :- Uses Haar-like features (rectangular patterns) to identify contrasts in a pixel intensity (e.g. edges, lines).
Note:- Haar feature is defined by two or more adjacent rectangular regions within a larger detection window. Its value is calculated as the diff between the sum of pixel intensities in the white rectangles and the sum of pixel intensities in the black rectangles.

- How it works
1. Training phase:
   - Uses many (+)ve images (with object) and negative images (without object).
   - Learns small features (like eyes, nose, edges) using a method called Haar features.
   - Builds a strong classifier by combining many weak ones using AdaBoost
2. Detection phase:
   - Scans the image at different scales and positions
   - Uses a cascade of stages:
     - If an area passes the first test → go to the next.
     - If it fails → immediately rejected.

* LBPH (Local Binary Patterns Histogram) ~~(LBPH)~~
- is a texture-based extraction method commonly used in face recognition and other image classification tasks
- It works by describing the local structure (patterns) around each pixel and then building a histogram from these patterns
- Key concepts:
1. Local Binary pattern (LBP)
   - For each pixel, compare it with its 8 neighboring pixels
   - If neighbor pixel ≥ center pixel → write 1, else
   - Form an 8-bit binary number (e.g. 11001010)
   - convert that binary number into a decimal value

2. LBP Image
- After converting each pixel to its LBP value, we get a new LBP image
- This image highlights textures and patterns

3. Histogram
- Divide the image into small grids (e.g. 8x8 cells)
- For each cells, calculate a histogram of LBP values (0-255)
- concatenate all histograms into a single feature vector

* keras
- keras is a high-level deep learning library written in python
- It allows you to ~~find~~ build and train neural networks easily - without worrying about the complex math happening in the background

* RNN
- RNN is a type of neural network designed for sequential data where the order of inputs matters, such as in text, speech, or time series. Unlike traditional networks, RNNs have a memory in the form of a hidden state that is passed from one time step to the next, allowing the model to learns patterns across sequences. At each step, the RNN processes the current input along with the previous hidden state to generate an updated state. While RNNs are effective for short-term dependencies, they struggle with long-term memory due to the

vanishing gradient problem, which is why improved versions like LSTM and GRU are often used

$$h_t = f(W_{hh} \cdot x_t + U h_{t-1} + b)$$

* LSTM (long short-term memory)
- LSTM is an advanced version of RNN that can remember information for a long time.
- It uses gates (input, forget and output gates) to control what to keep, update or forget from its memory. This makes LSTM great for handling long sequences like text, speech or time series without losing important information.

* GRU (Gated Recurrent Unit)
- GRU is a simpler and faster version of LSTM
- It combines the forget and input gate into a single update gate, and uses a reset gate to control memory. It's quicker to train and often performs just as well as LSTM for many tasks.

* Gated Feedback Recurrent Neural Network
- specialized type of recurrent neural network that introduces feedback connections between layers.
- Layers:
• Input layer: Receives the sequential input data.

- Hidden layers : Includes feedback and gating mechanisms
- Feedback Connections : Transmit high-level information back to lower layers
- Olp layer : produces predictions for the current time step

* CNN
- A CNN is a type of deep learning model specially designed for working with images.
- It automatically learns patterns like edges, textures, and shapes from images to perform tasks like classification, object detection and face recognition.

Main layers
1. Convolution layer
   - Applies filters to extract features
2. ReLU
   - Adds non-Linearity to learn complex patterns
3. Pooling layer
   - Reduces image size while keeping important features.
4. Fully connected layer (Dense)
   - Makes the final prediction (eg 'cat' or dog')

* Transfer learning
- is a technique in deep learning where a pre-trained model (trained on a large dataset like ImageNet) is reused for a

new, similar task
- Instead of training from scratch, you take an existing model and fine-tune it on your smaller dataset
- workflow
  - load a pre-trained model
  - Remove the top (o/p) layer
  - Add your own classification layers
  - Train only the top layers or fine-tune some base layers

* AlexNet
- was the first deep learning model that revolutionized image classification by winning the ImageNet competition in 2012 by a large margin
- key features:
  - 8 layers: 5 convolutional + 3 fully connected
  - Use ReLU activation
  - Introduced dropout to reduce overfitting

* ResNet
- is very deep network (upto 152 layers) that introduced skip connections, also known as residual connections
- key feature
  - solves the vanishing gradient problem
  - Adds "shortcut" paths.
  Instedd of learning everything from scratch, it learns the diff (residual)

Date / /
Page L&W

* Image segmentation
- predict a label for each pixel in an image, understant object boundaries and shape
- Types
(a) Semantic segmentation :- classifies each pixels into a category
(b) Instance Segmentation :- Identifies each object instance separately
(c) Panoptic Segmentation: Combines both semantic + instance segmentation

* Image Recreation
- Image Recreation refers to the process of rebuilding, restoring, or generating an image based on partial, degraded, or encoded data. It can mean different things depending on the context.

* Image Mapping
- Image mapping refers to the process of transforming one image to align with another using geometric transformations.
- It's used when you want to warp, align or project an image onto a different shape, surface or coordinate system.

* Video Concepts
- Video
- sequence of Images, called frames, played rapidly
- creates the illusion of motion, like a flip-book

* Video concept.

• Frame
- A single still image in a video sequence
- Importance: The building block of all video content
- Visual: show a single image extracted from a video

• Frames Per second (FPS)
- Low FPS = Choppy motion, High FPS = Smooth motion
- standard FPS: Movies: 24 FPS, Tv/online shows, videos: 30 FPS, sports/Gaming: 60 FPS

* video Resolution
- Number of pixels displayed in each frame of a video.

* Birate: The amount of data processed per unit of time in a video or audio file, usually measured in kilobits per second (kbps) or megabits per second (Mbps)

↑ birate = ↑ data/sec = Better quality = large file
↓ birate = ↓ data/sec = lower quality = small file

* Codec (compressor - Decompressor)
- compression / decompression algorithm used to encode and decode video or audio data.
- Reduces file size while trying to maintain

quality, making media easier to store and stream

* container: A file format that stores different multimedia data streams (video, audio, subtitles, metadata, etc) in a single file.

* YOLO (You Only Look once)
- is a deep learning-based object detection algorithm. It can detect multiple objects in an image in a single forward pass, making it very fast compared to older approaches.

Working
1. Single Neural Network processes the entire image.
2. It divides the image into S x S grid.
3. Each grid cell:
   - Predicts bounding boxes (object locations)
   - Predicts class probabilities (what object it is)
4. All predictions are done simultaneously. so it's fast.

* R-CNN.
- It was one of the first to combine region proposals with CNN-based feature extraction for accurate object detection.
- Rather than trying to ~~each~~ detect objects from the entire image in one go (like YOLO does), R-CNN works in two stages:
  Step 1: Find object-like regions (called region proposals)
  Step 2: Run a CNN on each of these regions to classify them and adjust their bounding boxes

- works:

1. Region proposal:
   - It uses a method like Selective Search to Find possible regions (usually ~2000 regions)

2. Feature Extraction:
   - Each region is passed through a CNN (like AlexNet) to extract features.

3. Classification:
   - A support vector Machine (SVM) classifies the object in each region.

4. Bounding Box regression:
   - A regressor refines the position of each bounding box.

* Fast R-CNN.

- Faster and more efficient object detection algorithm than R-CNN. It processes the entire image only once and performs region-based classification and bounding box prediction in one go.

working.

step 1: Input image

Step 2 :- RUN CNN on whole Image

step 3 : region proposals (still from selective search)

note :- Donot crop and resize them like RCNN

step 4: ROI Pooling
- converts diff sized regions into fixed-size Fixed size feature vectors
- makes batching and classificatn

steps: fully connected layers + prediction

step 6: final o/p
- all regions are processed
- apply Non-maximum suppression to eliminate duplicates

* Object tracking

- OpenCV's TrackerKCF library.
- KCF: kernelized correlation filters
- is the process of locating a moving object over time in a video or sequence of frames.