OpenCV Fundamentals - Week 1

1. Introduction to OpenCV

Sources I used:

- OpenCV official docs – overview page
- YouTube video from freeCodeCamp on Computer Vision basics

What I learned:
OpenCV is a big library that helps with computer vision tasks. It was first made by Intel in 2000 and now is open-source. It works on many systems (Windows, Linux, Mac). We can use it in C++ or Python. Installing it in Python is just pip install opencv-python.

Key points:

- Can read, show, and save images easily.
- Works for both photos and live video.
- Has many modules for different work like image filtering, object detection, face recognition, etc.

---

2. Core Module

Sources I used:

- Docs for core module
- YouTube playlist section on core functions

What I learned:
This is like the heart of OpenCV. It has data structures (like Mat in C++ or NumPy arrays in Python) and helper functions. All the other modules depend on this one.

Key points:

- Handles basic math on images.
- Stores image data as arrays (each pixel has values).
- Has functions for random number generation, file storage, etc.

---

3. Image Processing Module (imgproc)

Sources I used:

- Docs for imgproc
- Blogs on Gaussian blur and edge detection

What I learned:
This module has the tools to actually change and process images. We can blur them, detect edges, change colors, and apply transformations like rotate, resize, warp. Some of the codes in code section is related to blurring and sharpening the image.

Key points:

- Color space changes.
- Filtering like blur, sharpen I have learnt.
- Detects shapes and contours.

---

4. Application Utils (highgui, imgcodecs, videoio)

Sources I used:

- Docs for highgui and imgcodecs
- Example codes from YouTube

What I learned:
These modules handle how we open windows, show images, read/write files, and handle camera/video input.

Key points:

- imshow() shows image.
- imread() reads image from file.
- VideoCapture() to read from webcam or video file.
- waitKey() is important for keeping window open.

---

5. Camera Calibration & 3D Reconstruction (calib3d)

Sources I used:

- OpenCV calibration tutorial page
- A YouTube tutorial with chessboard calibration example

What I learned:
This is for making cameras more accurate. Cameras have distortion, and calibration helps fix it. Also used for stereo vision to estimate depth.

Key points:

- The 3d distances are calculated with the position of the source and destination.
- Remove fisheye effect.
- Can measure distances and positions in 3D.

---

6. Object Detection (objdetect)

Sources I used:

- Docs for objdetect
- Example Haar cascade for face detection

What I learned:
This module is for detecting certain objects using pre-trained models like Haar cascades or LBP cascades. In the code present in code section, Haar cascades are used. It can detect multiple objects present in the image, video or screen.

Key points:

- Works for face, eyes, cars, etc.
- Fast but less accurate compared to deep learning.
- Needs XML cascade file to run.

---

7. 2D Features Framework (feature2d)

Sources I used:

- Docs on SIFT, ORB
- Blog on feature matching with FLANN

What I learned:
It's for finding unique points in images and matching them between two pictures. Useful for stitching, object recognition, etc. Stitching of image is an awesome concept that I came across after writing a code on it. It stitches two half pictures into a full one very accurately.

Key points:

- SIFT, SURF, ORB are famous algorithms.
- Works for tracking things even if rotated or scaled.

---

8. Deep Neural Networks (dnn)

Sources I used:

- Docs on dnn module
- PyImageSearch tutorial on loading models

What I learned:
You can run deep learning models inside OpenCV without heavy frameworks. It supports models from TensorFlow, Caffe, ONNX.

Key points:

- Load pre-trained .pb or .onnx files.
- Can do face recognition, object detection, segmentation.
- Faster if GPU is available.

---

9. Graph API (gapi)

Sources I used:

- Docs for G-API
- A GitHub example repo

What I learned:
This is for making image processing pipelines that are faster because OpenCV optimizes them. It is still new and evolving.

Key points:

- Works like a flow chart of operations.
- Good for repeated processing on many images.

---

10. Other Tutorials (ml, photo, stitching, video)

Sources I used:

- Docs for each
- Example codes from OpenCV samples

What I learned:
These are extra helpful modules:

- ml = Machine Learning
- photo = Image enhancement (denoising, inpainting)
- stitching = Joining multiple images
- video = Motion tracking

Key points:

- Stitching uses feature matching.
- Inpainting fills missing parts.
- Video module can track moving objects.

---

11. Theory of Image Processing

Sources I used:

- My college notes on Digital Image Processing
- OpenCV doc theory pages

What I learned:
Understanding pixels, color models (RGB), convolution, and frequency domain methods like Fourier Transform.

Key points:

- Pixel is smallest unit of an image. It actually tells us the image color.

- Filters can be applied in spatial or frequency domain.
- Edge detection finds object boundaries.

Research

Image Processing

What I learned:
Filtering smooths images, thresholding separates objects from background, and transformations change image position/shape. These are used in real life like face blurring in news, license plate detection, medical scans cleanup.

Research OpenCV Applications

Applications found:

1. Robotics - Detect obstacles and navigate.
2. Medical Imaging - Analyze MRI or X-rays.
3. Autonomous Vehicles - Detect lanes, cars, pedestrians.
4. Surveillance - Motion detection in CCTV.
5. Augmented Reality - Overlay objects on real video.
6. Industrial Inspection - Check defects in products.
7. Document Scanning - Detect and crop documents.
8. Face Recognition - For security systems.
9. Gesture Control - Control devices using hand movements.
10. Sports Analysis - Track players and ball.

Differences in OpenCV Windows

What I learned:
In Windows, OpenCV's GUI needs to run in the main thread, or the windows may freeze. In Linux, it's a bit more flexible but still not perfect.

Key Insights & Conclusion

- OpenCV is modular, but modules depend on the core. It is mainly used for processing images. Manipulate them, detecting objects
- Some algorithms are old but still useful.
- Deep learning in OpenCV is powerful but needs good hardware.
- Understanding theory makes coding easier.
- Practical problems include window freezing, image format issues, and slow processing for large images.

NOTE: For codes, install requirements.txt using pip install requirements.txt