OpenCV Week 1 Assignment – Report

1. Project Overview

The objective of this assignment is to understand **OpenCV fundamentals** and perform practical exercises using Python. This project includes:

- Learning **core modules** and image processing techniques.
- Performing **image and video operations**.
- Implementing **drawing, filtering, and edge detection**.
- Managing **assets dynamically** (random images/videos).
- Preparing a reproducible environment in **VS Code**.

This project forms a foundation for advanced computer vision tasks.

2. Folder Structure

```
opencv-week1/
│
├── notes/                          # Module summaries
│   ├── introduction.md
│   ├── core_module.md
│   ├── imgproc_module.md
│   └── misc_research.md
│
├── exercises/                      # Python exercise scripts
│   ├── read_display_image.py
│   ├── basic_drawing.py
│   ├── video_capture.py
│   └── filtering.py
│
├── research/                       # Research notes
│   ├── opencv_applications.md
│   └── os_window_differences.md
│
├── assets/                         # Images and videos downloaded/generated
├── report.pdf                      # This report
├── README.md                       # Project overview and instructions
├── requirements.txt                # Required Python packages
```

3. Notes Summary

3.1 Introduction to OpenCV

- OpenCV is an **open-source computer vision library**.
- Supports **image/video processing**, machine learning, object detection, and more.
- Core concepts: matrices (images), arrays, utility functions.

3.2 Core Module

- Provides **basic data structures**, **array operations**, and **utility functions**.
- Example: Using **NumPy arrays** for image manipulation.

3.3 Image Processing (imgproc module)

- **Transformations:** Resize, rotate, warp.
- **Filtering:** Gaussian blur, median filtering.
- **Edge detection:** Canny, Sobel.
- **Histograms:** Intensity analysis.

3.4 Misc Research

- OpenCV applications: robotics, medical imaging, surveillance, autonomous vehicles, augmented reality.
- OS differences: Linux may block multiple windows if not in main thread; Windows allows multi-window easily.

---

4. Exercises

4.1 read_display_image.py

- **Purpose:** Read and display a random image.
- **Method:** Randomly download an image from URLs → display in OpenCV window → save copy.
- **Expected Result:** Image window shows random image.
- **Screenshot Placeholder:** assets/output_random.jpg

4.2 basic_drawing.py

- **Purpose:** Draw shapes (line, rectangle, circle) on a canvas.
- **Method:** Create a black canvas with NumPy → draw colored shapes using OpenCV functions.
- **Expected Result:** Window shows shapes. Saved as assets/basic_drawing_output.jpg.
- **Screenshot Placeholder:** *(insert here)*

4.3 video_capture.py

- **Purpose:** Play a random video.
- **Method:** Randomly select video → download if missing → play with cv2.VideoCapture.
- **Expected Result:** Video plays in window. Press q to exit.

4.4 filtering.py

- **Purpose:** Apply filters and edge detection.
- **Method:** Gaussian blur → Canny edge detection → save result.
- **Expected Result:** Edge-highlighted image displayed. Saved as assets/output_edges.jpg.
- **Screenshot Placeholder:** *(insert here)*

---

5. Research Findings

5.1 OpenCV Applications

1. **Robotics:** Object detection and navigation.
2. **Medical Imaging:** MRI/X-ray analysis.
3. **Autonomous Vehicles:** Lane and pedestrian detection.
4. **Surveillance:** Face detection, motion tracking.
5. **Augmented Reality:** Marker detection and overlays.

5.2 OS Window Differences

- **Windows:** Multiple windows open without main thread issues.
- **Linux:** Windows must run in main thread to avoid freezes.

---

6. Key Learnings

- OpenCV integrates **theory and practice** in Python.
- Random image/video handling makes scripts dynamic and reusable.
- Proper window handling (imshow, waitKey, destroyAllWindows) is essential.
- Folder organization and asset automation improve reproducibility.

---

7. Challenges

- Handling random images/videos of different resolutions.
- Ensuring scripts run in **VS Code and Colab**.
- Managing OS-specific window behavior.

---

8. Conclusion

This assignment demonstrates **OpenCV fundamentals**, including:

- Core modules and array handling.
- Image transformations, filtering, and edge detection.
- Video playback and dynamic asset handling.
- Practical understanding of OpenCV applications.

All scripts are modular, automated, and ready to run in **VS Code** or **Colab**, providing a strong foundation for future computer vision projects.

---

9. References

- OpenCV Official Documentation
- YouTube Playlist: OpenCV Week 1