⭐ Major Concept Explanations

1. Color Format Difference

- OpenCV loads images as **BGR**
- Matplotlib displays images as **RGB**

If you use an image loaded directly by OpenCV in Matplotlib, the colors will be incorrect.

2. How OpenCV Windows Work

OpenCV manages image windows using:

- `namedWindow()`
- `imshow()`
- `waitKey()`
- `destroyWindow()` or `destroyAllWindows()`

`waitKey()` is essential:

- It keeps the window responsive
- It processes keyboard events
- It prevents the window from freezing
- It controls timing for video loops

3. Matplotlib is NOT real-time

Matplotlib:

- Renders slowly
- Blocks execution until the figure closes
- Is designed for analysis, not high-speed CV

🔥 **Matplotlib imshow() vs OpenCV imshow() — Complete Comparison**

✓ 1. Color Format

Library	Color Space
OpenCV	BGR
Matplotlib	RGB

✓ 2. Display Location

Library	Where the image appears
OpenCV	OS window (fast, real-time)
Matplotlib	Inside a plotted figure window or notebook

✓ 3. Speed

Library	Performance
OpenCV	Very fast, usable for live video and real-time CV
Matplotlib	Much slower, not suitable for real-time loops

✓ 4. Blocking Behavior

Library	Behavior
Matplotlib	Blocks execution until `plt.show()` completes
OpenCV	Blocks depending on `waitKey()` value

✓ 5. Supported Data Types

Library	Flexibility
OpenCV	Strict (expects uint8 image arrays)
Matplotlib	Flexible (supports float, normalized values, heatmaps, etc.)

🎯 Real-World Use Cases

Use Matplotlib when:

- You’re analyzing results (segmentation maps, bounding boxes)
- You're debugging the image processing output
- You're working in Jupyter notebooks
- You want titles, axes, annotations, subplots

Use OpenCV when:

- You need a real-time frame display (webcam, video, robotics)
- You're building a CV pipeline or application
- You need keyboard interactions (`q` to quit)
- You need a high FPS display