1) Project Overview

This repo demonstrates four traditional segmentation pipelines:

- **Otsu Thresholding** – global, histogram-based binary segmentation.
- **K-Means Clustering** – color clustering for multi-region segmentation.
- **Contour Detection** – edge detection followed by contour extraction/visualization.
- **Watershed** – marker-based region growing for touching-object separation.

Each pipeline is a standalone Python script for clarity and quick experimentation.

---

2) Requirements & Installation

Supported on Python 3.9–3.12.

```
# (Optional) create and activate a virtual environment
python -m venv venv
# Windows
venv\Scripts\activate
# macOS / Linux
source venv/bin/activate

# Upgrade installer
python -m pip install -U pip

# Install dependencies
pip install -r requirements.txt
```

Minimal `requirements.txt`:

```
numpy
matplotlib
opencv-contrib-python
```

If you installed `opencv-python` and miss functions like `selectROIs`, switch to `opencv-contrib-python`.

---

3) Quickstart

Place a small test image in `data/` (e.g., `data/sample.jpg`) and run any script from `src/`:

```
python "src/Otsu Thresholding.py" --image data/sample.jpg
python "src/K-Means Algorithm.py" --image data/sample.jpg --k 3
python "src/Contour Detection.py" --image data/sample.jpg
python "src/Watershed Algorithm.py" --image data/sample.jpg
```

If your scripts don't yet accept CLI arguments, open them and set the input path near the top, then run without flags. GUI windows (`cv2.imshow`, ROI selector, or `plt.show`) require a desktop environment.

---

4) Pipelines & Notes

4.1 Otsu Thresholding

- **Idea:** choose a global threshold that maximizes inter-class variance.
- **Typical steps:** grayscale → Gaussian blur (optional) → `cv2.threshold(..., OTSU)`.
- **Output:** binary mask + histogram visualization.
- **Good for:** bimodal histograms; quick foreground/background separation.

4.2 K-Means Clustering

- **Idea:** cluster pixels (RGB or features) into *k* groups.
- **Typical steps:** reshape image → `cv2.kmeans` → map labels back to image.
- **Key params:** k (clusters), criteria/iterations.
- **Good for:** coarse region grouping, color-based segmentation.

4.3 Contour Detection

- **Idea:** detect edges, then find and draw/measure contours.
- **Typical steps:** blur → Canny → `cv2.findContours` → filter by area/shape.
- **Outputs:** contour overlay, count/area/perimeter stats.
- **Good for:** object boundaries, simple shape analysis.

4.4 Watershed

- **Idea:** treat image as a topographic surface and flood from markers.
- **Typical steps:** denoise → threshold → morphology → distance transform → markers → `cv2.watershed`.
- **Good for:** splitting touching objects (cells, coins, seeds).

---

5) Troubleshooting

- **`ModuleNotFoundError: cv2`** → `pip install opencv-contrib-python`
- **No GUI/Windows open** (headless server) → run locally with a desktop environment, or modify code to save figures instead of showing them.
- **`selectROIs` missing** → uninstall `opencv-python`, install `opencv-contrib-python`.
- **Unicode path issues on Windows** → avoid non-ASCII paths; wrap paths in quotes.

---

6) License & Citation

Add a license file (e.g., MIT) at repo root. If this is for a class or paper, include a short citation here:

```
@software{image_segmentation_demo,
  title={Image Segmentation – Classic Pipelines},
  author={Your Name},
  year={2025},
  url={https://github.com/lxl-max/image-segmentation}
}
```

Last updated: 2025-11-07