15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 10: Computer Vision with OpenCV

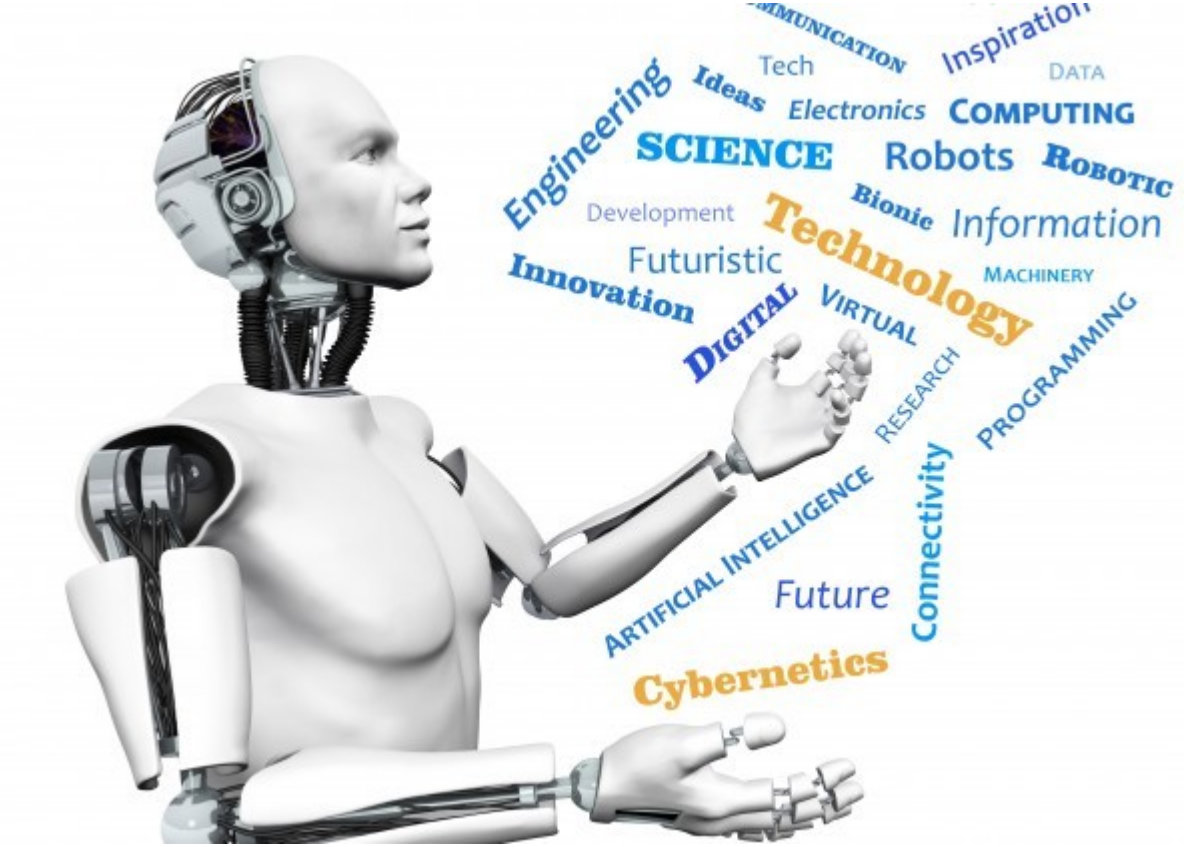


Image from http://www.futuristgerd.com/2015/09/10

Three Ways to Do Vision on VEX AIM

- On-board: the microcode detects barrels, balls, AprilTags, robots, and color blobs. vex-aim-tools maps objects to 3D space and puts them on the world map.

- AskGPT: “What kind of fruit do you see in the camera image?”

- Roll your own: write Python code, typically using OpenCV. We use this for ArUco markers.

Acessing Images on VEX AIM

- You can get the current camera image at robot.camera_image
- If you want to run your own vision code on every image, override the user_image method of StateMachineProgram
- user_image takes two arguments: the color image, and a grayscale version

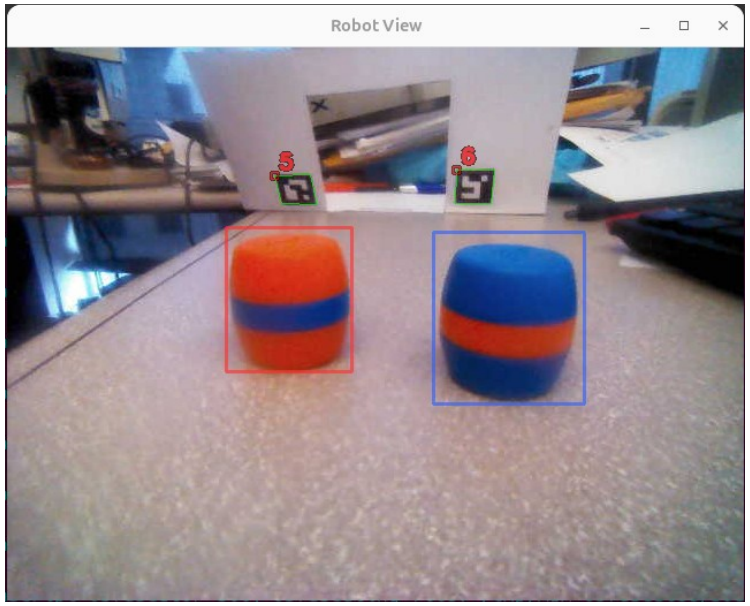
Color Segmentation Demo

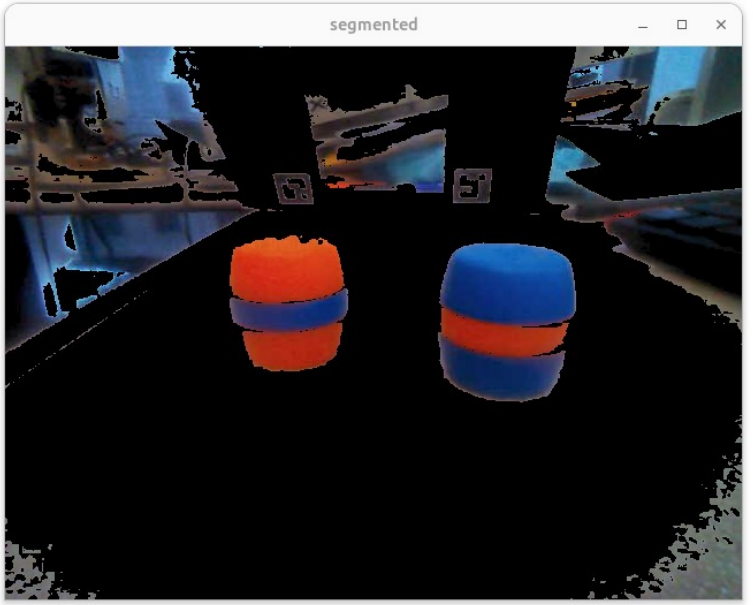
```
from aim_fsm import *


class ColorSeg(StateMachineProgram):
  def start(self):
    super().start()
    self.window = namedWindow('segmented')

  def user_image(self, image, gray):
    result = image.copy()
    orange_filter = (image[:,:,0] > 120)&(image[:,:,2] < 50)
    blue_filter = (image[:,:,0] < 120) & (image[:,:,2] > 10)
    result[~(orange_filter | blue_filter)] = 0
    imshow('segmented', result)
```

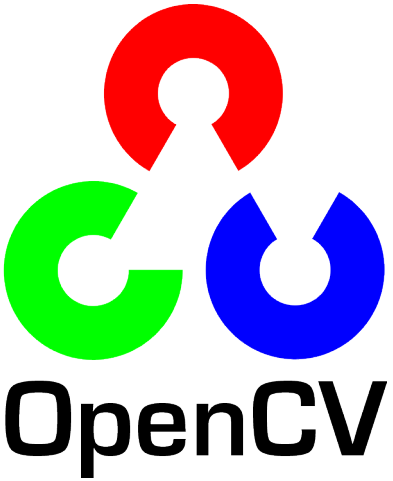
Color Segmentation





OpenCV

- Open source real-time computer vision library.
- Originally developed by Intel.
- Written in C++ and C.
- Includes support for GPU processing.



Online Documentation

- We are using OpenCV 4.11.0
- Documentation is at: https://docs.opencv.org/4.11.0
- OpenCV-Python tutorials linked from OpenCV.org.
- Note: OpenCV images use BGR byte order instead of the conventional RGB.

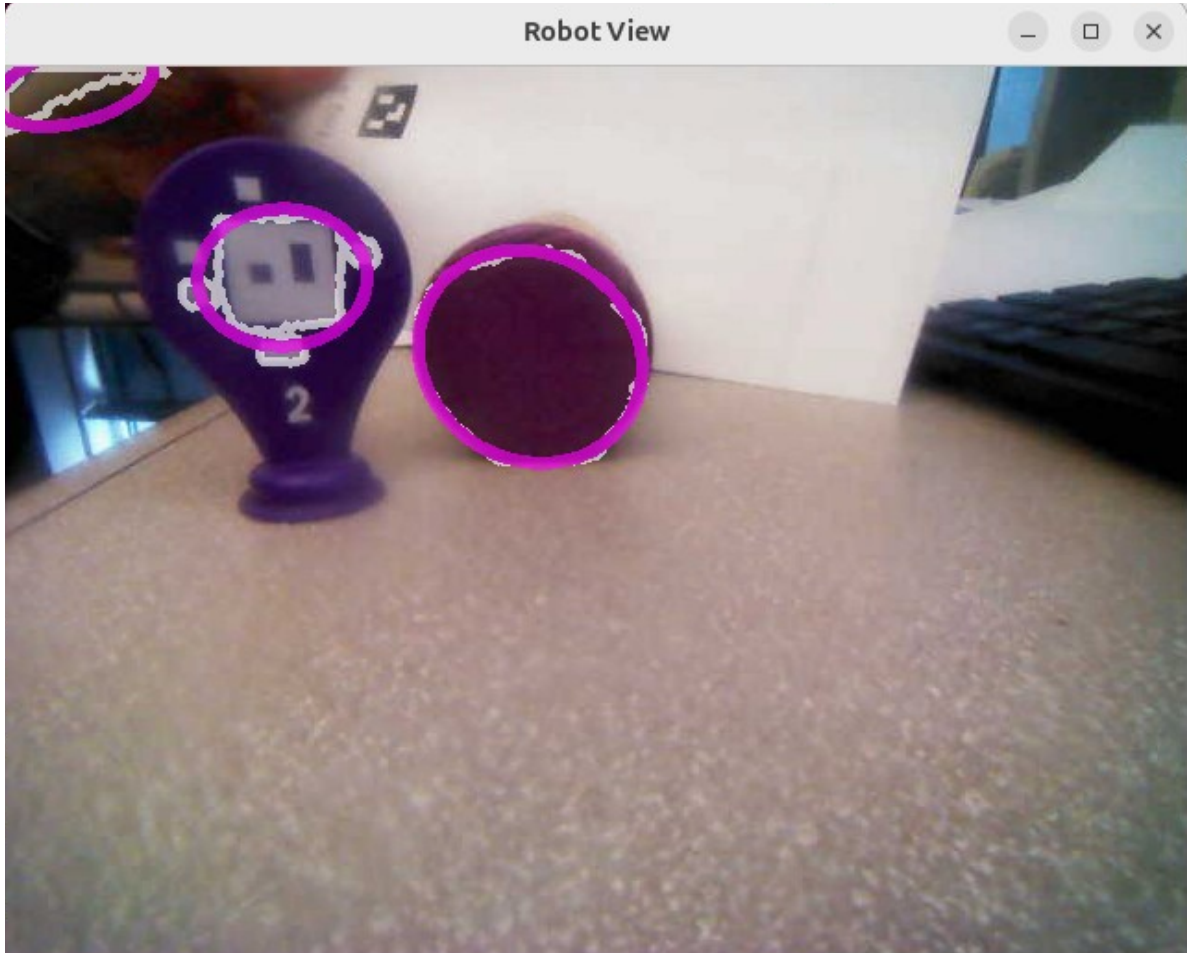
OpenCV in Python

- Python bindings allow you to call OpenCV library routines.
- The OpenCV module is called “cv2”.
- Data is passed as numpy arrays.
- vex-aim-tools uses OpenCV to detect ArUco markers.

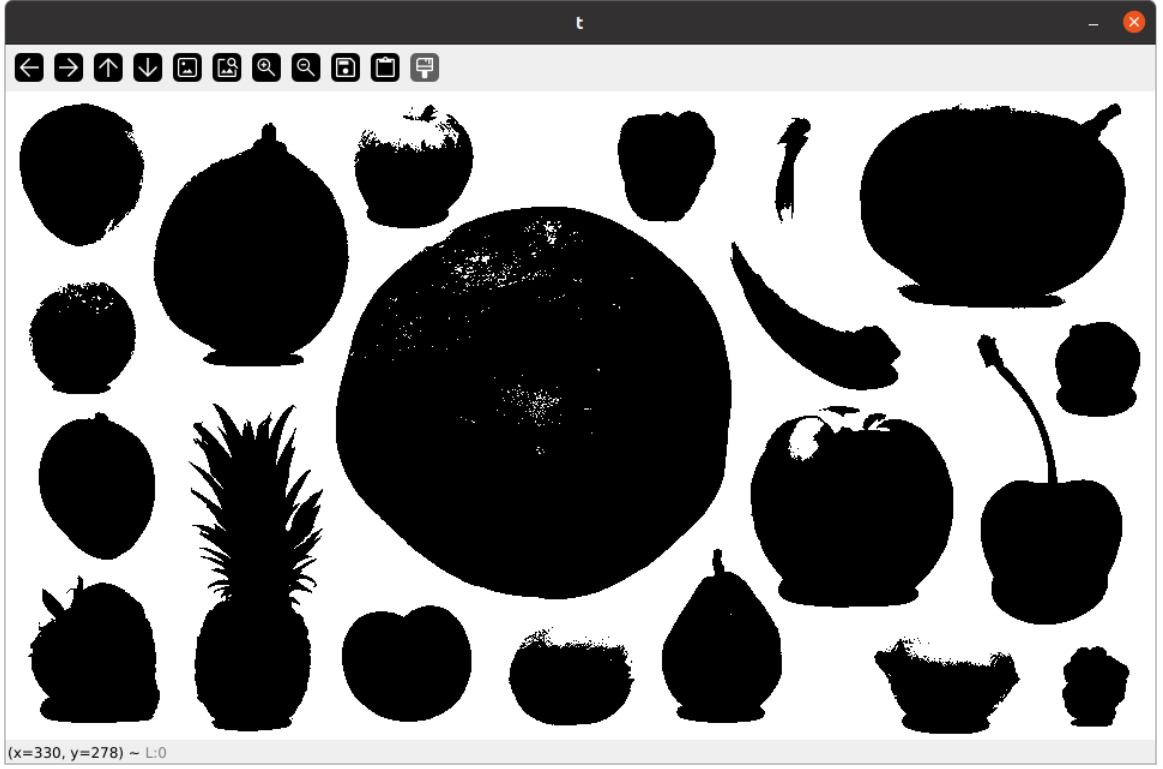
CircleWatcher demo

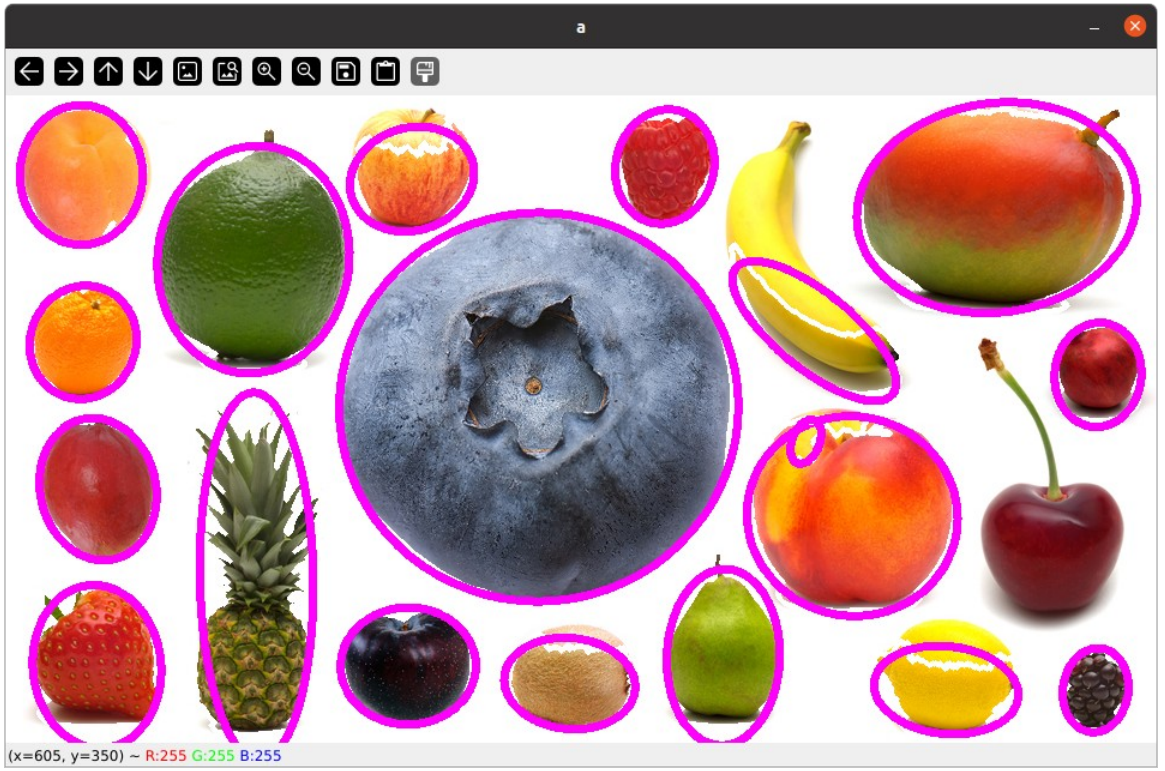
- Built on StateMachineProgram.
- user_image method processes each camera frame.
- cv2.threshold thresholds the image.
- cv2.findContours finds the boundaries of connected regions
- We filter them to find circular regions
- user_annotate method displays results in the camera viewer.

CircleWatcher Demo



Circle Extraction

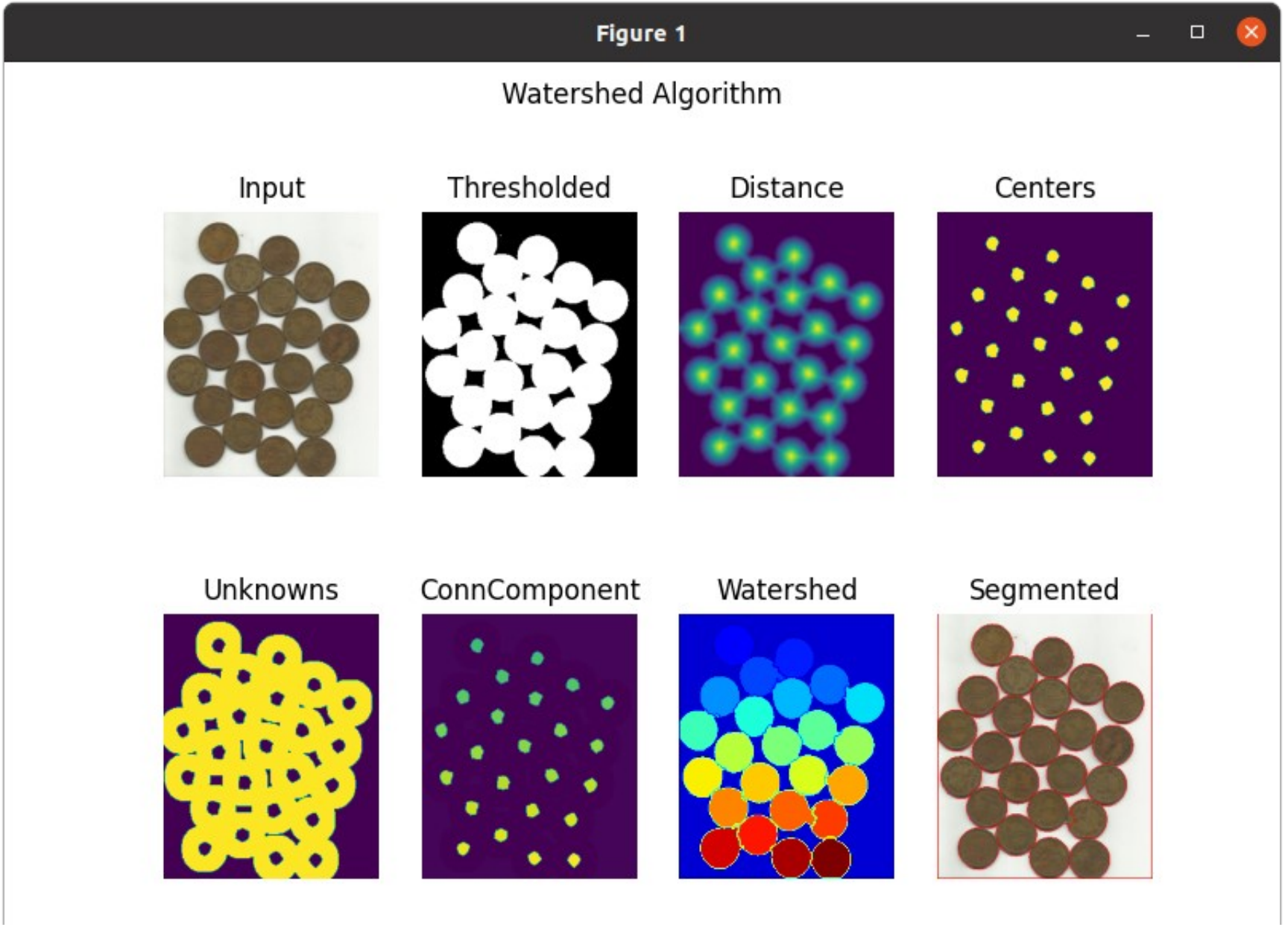




OpenCV Demos in vex-aim-tools/aim_fsm/examples

- CV_Canny – Canny edge detector
- CV_Contour – Find intensity contours
- CV_GoodFeatures – Find interest points
- CV_OpticalFlow – Track interest points
- CV_Hough – Find lines

Watershed Algorithm



Displaying Images

- In vex-aim-tools: uses PyQT6 and threading

  namedWindow('foo')

  imshow('foo', my_image)

- In OpenCV: uses Tkinter

  cv2.namedWindow('foo')

  cv2.imshow('foo', my_image)

  cv2.waitKey(1)

- In matplotlib: uses Tkinter
  - plt.imshow(my_image)
  - plt.pause(0.01)