RASPBERRY PI

EMILY WENG

DIRECT CAPTURE NOT WORKING

- There are a few ways we can solve this
- One is by using WINSCP and putty to execute the program
- Two is reinstalling VNC
  - If you do this, please backup your raspberry pi file and save it on your computer
- If all does not work, then we reinstall pi imager and rewrite all the files

Running directly rendered apps such as Minecraft remotely

KNOWN ISSUE: The "direct capture" option is not working when running VNC Server on Raspberry Pi OS Bullseye with the KMS driver. Enabling it will cause VNC Server to send a blank screen and you will lose remote access to the VNC Server. Our development team are investigating and hope to make a fix available in the near future. This issue affects all current versions of VNC Server, 6.8.0 and earlier.

DEBUGGING

- Check for spelling, capitalized letters or wrong used variables, indentation and space, programs not installed or modules not working
- Debugging takes a long time so BE PATIENT
- It doesn’t always work the first time

OPENCV

OPENCV

- This stands for Open Source Computer Vision Library
- It's a great tool for image processing and performing computer vision tasks
- It used to perform tasks like face detection, objection tracking, landmark detection and much more
- We also used it for face detection and picture detection

LET'S TRY GETTING AN IMAGE

- We have to import cv2 and sys (this lets us access system specific parameters and functions)
- We're going type:

```
imagePath = sys.argv[1]
```

- Sys.argv is a list in Python, which contains the command-line arguments passed to the script.
- We're going to read the image
  - image= cv2.imread(imagePath)
  - image path is the variable we set
  - cv2.imshow("preview")
  - cv2.waitkey(0)
  - cv2.detroyAllWindow

RUNNING THE FILE

- We need an image so find a random image on google and place it in the same folder as the rest of our files
- Run the program on terminal with the photo name
  - Python image_load.py test.jpg

VIDEO CAPTURE

- We can also view the results of the camera
- Import cv2 again
- This time, we'll set a variable for the function video capture
- a= cv2.VideoCapture(0)
- Then we set the frame width and height
  - a.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
  - a.set(cv2. CAP_PROP_FRAME_HEIGHT, 240)

CONT.

- And now the while true loop
  - Ret, fame =a.read() –(we’re going to read the camera for video taking)
  - Cv2.imshow(“preview”,frame)
  - If cv2.watikey(1) & 0xFF == ord(“q”) –(taking keyboard input)
    - Break
  - a.release()
  - Cv2.destroyAllWindows()

COLORS

- How does computer detect items?
  - Color
  - Shape
  - Properties
- Image detection:

COLOR SPACE

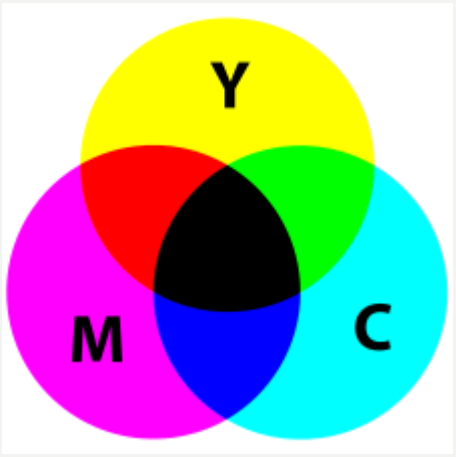
- Specific organization of colors
- RGB
- CMY
- YUV
- HSV

RGB

- Red, green, blue
- The original color of light and the most commonly used
- When we mix colors we get different colors
- Hex code: #800000 maroon
- Decimal code: (128,0,0)

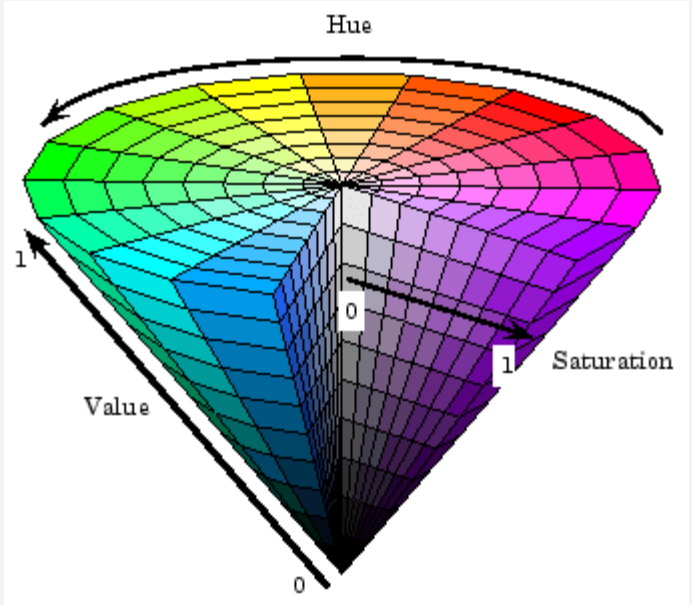
CMY(K)

- Cyan, magenta, yellow, black
- Normally used in printers
- The base colors for ink



HSV

- Hue, Saturation, Value
- Closest to what human sees
- Normally used in photography



CONVERTING WITH RGB AS CENTRE

- RGB to CMY

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1-R \\ 1-G \\ 1-B \end{pmatrix}$$

- CMY to RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1-C \\ 1-M \\ 1-Y \end{pmatrix}$$

- RGB to YUV

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

EXAMPLE

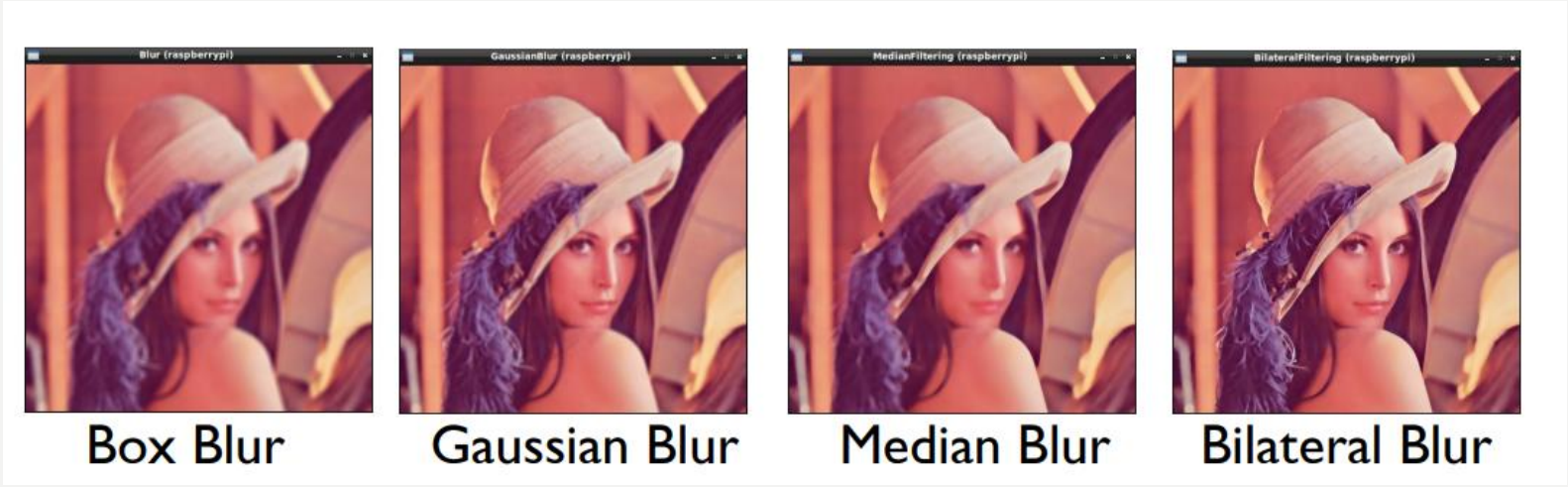
```
# Convert BGR to Gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)
cv2.waitKey(0)

# Threshold the gray image to binary image
(_, binary) = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
cv2.imshow("Binary", binary)
cv2.waitKey(0)

# Convert BGR to HSV
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV", hsv)
cv2.waitKey(0)

# Define range of purple color in HSV
lower = np.array([141, 0, 0])
upper = np.array([164, 145, 197])
```

GAUSSIAN BLUR

- In a Gaussian blur, the pixels nearest the center of the kernel are given more weight than those far away from the center.

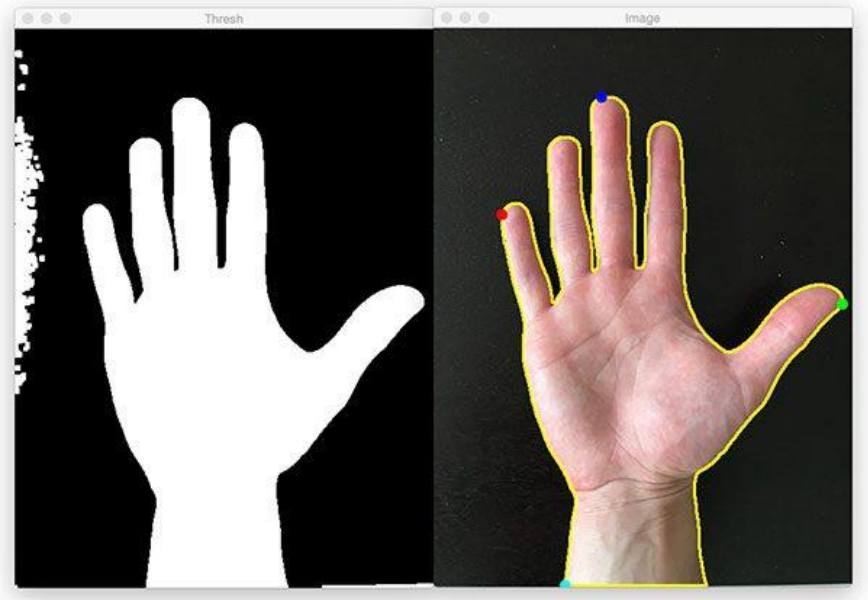
FILTERS

- There are different types of filter like
  - Blur
  - Gaussian blur
  - Fast blur
  - Median blur
  - Bilateral filter



FINDING THE MAIN POINT OF OBJECT

- The camera finds the edge of an object
- Then it contours which means the camera finds the light and dark items in the pictures



CONTOUR

- cv2.findContours(image, mode, method)
- cv2.drawContours(image,contours,contourIdx,color)
  - contourIdx = -1 means to contour all places

MAIN POINT

- The program finds the first contour:
  - (contours, _) = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
- The first contour point:
  - cnt = contours[0]
- Find the radius:
  - ((x, y), radius) = cv2.minEnclosingCircle(cnt)
  - M = cv2.moments(cnt)
- center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

FILES

- Move_car.py
- Follower_car.py
- Dwm_motor.py
- Pwr_motor.py
- Files with camera