RASPBERRY PI

PART 3

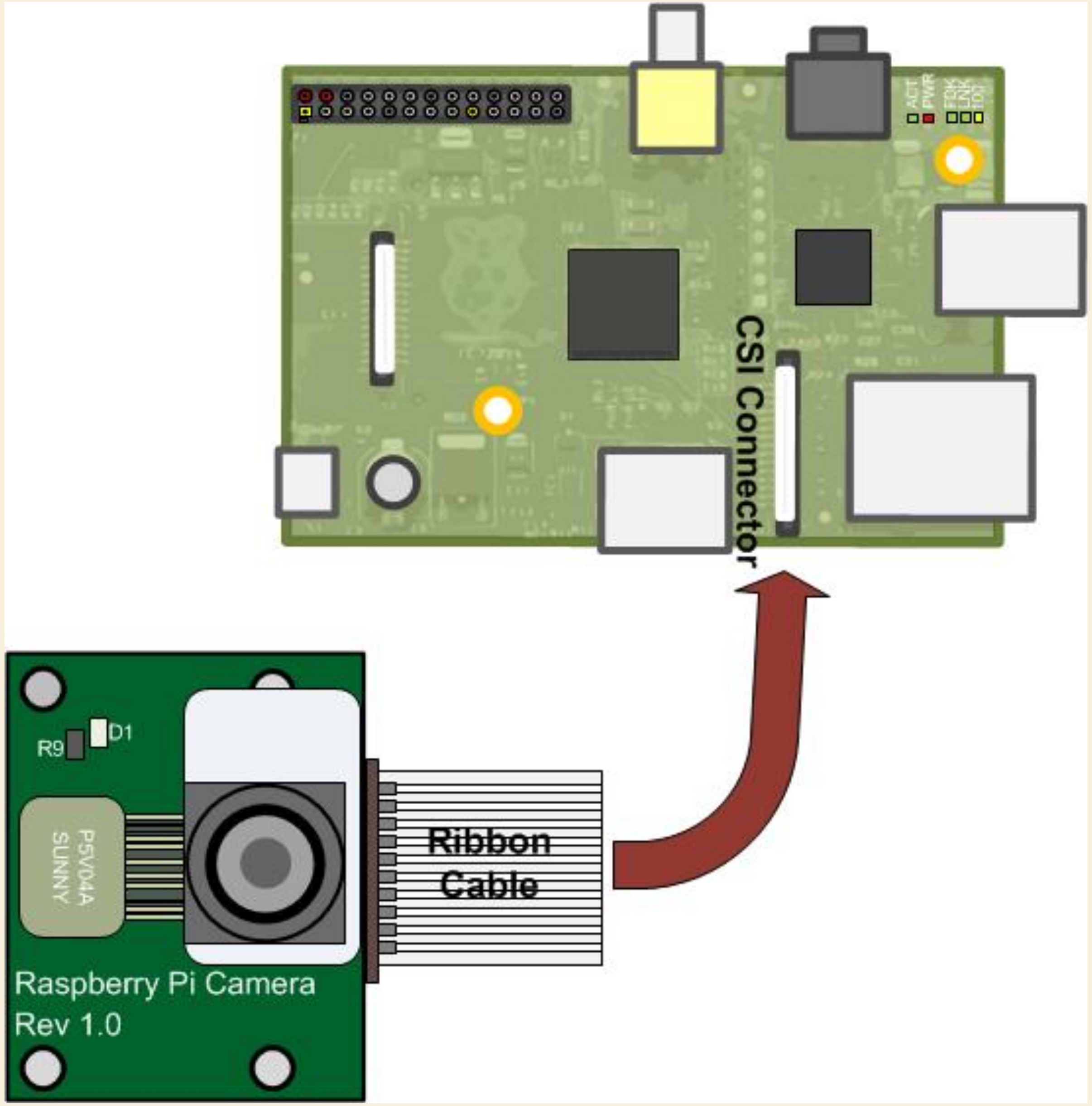
EMILY WENG

LAST TIME…

- We typed the final part of the code and to those who still need it, I'll show it later.
- We attached everything and managed to make it move. For some people, if you need time to finish, please finish as well.

CAMERA INTRO

CAMERA



CSI Connector
Ribbon Cable
Raspberry Pi Camera
Rev 1.0
P5V04A
SUNNY
R9
D1
ACT
PWR
FDX
LNK

FEATURES:

- Still Picture Resolution: 2592 x 1944
- Sensor:OmniVision OV5647 (5MP)
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- Size: 20 x 25 x 9mm
- Pixel Size:1.4 x 1.4 µm
- Lens:f=3.6 mm, f/2.9

NO IR CUT CAMERA

- Also called: Infrared cut-off filters
- No IR: designed to reflect or block near-infrared(shorter) wavelengths while passing visible light
- There are also filters which are used in solid state (CCD or CMOS) video cameras to block IR due to the high sensitivity of many camera sensors to near-infrared light
- So No IR camera with light source = night camera

CAMERA

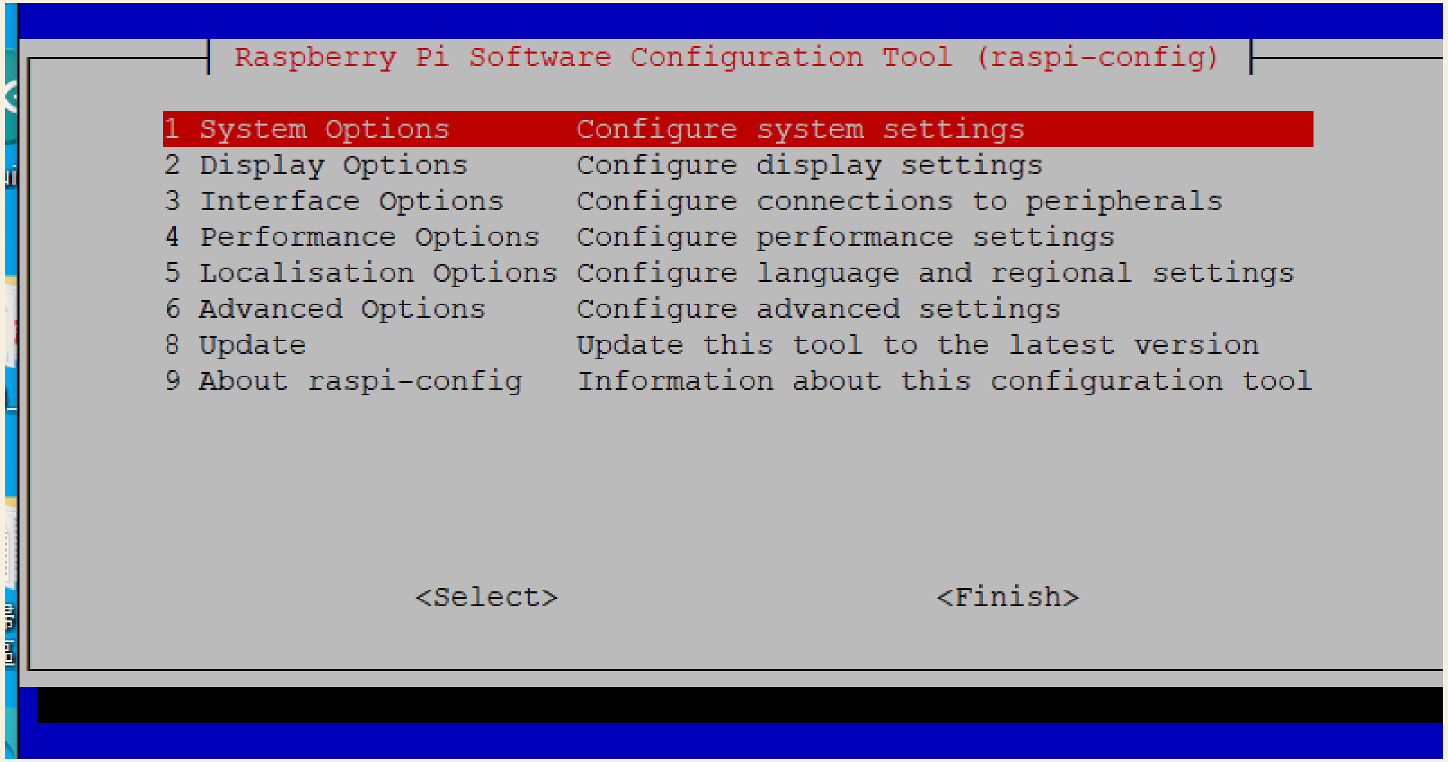
CAMERA

- First, we power off our raspberry pi
  - Sudo poweroff
- Insert the tip of the camera into your raspberry pi
- Today, we're using the raspberry pi camera module v1

SETTING UP

- The camera should light up when you power on
- Next we can go back to VNC and open up our Raspberry window
- Enter the command:
  - sudo raspi-config
- You'll be taken to a settings menu

FIRST WE GO TO INTERFACING OPTIONS AND ENTER (THIRD ONE)



NEXT WE'LL SEE A CAMERA OPTIONS

```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC           Enable/disable graphical remote access using RealVNC
I4 SPI           Enable/disable automatic loading of SPI kernel module
I5 I2C           Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins

                 <Select>                     <Back>
```

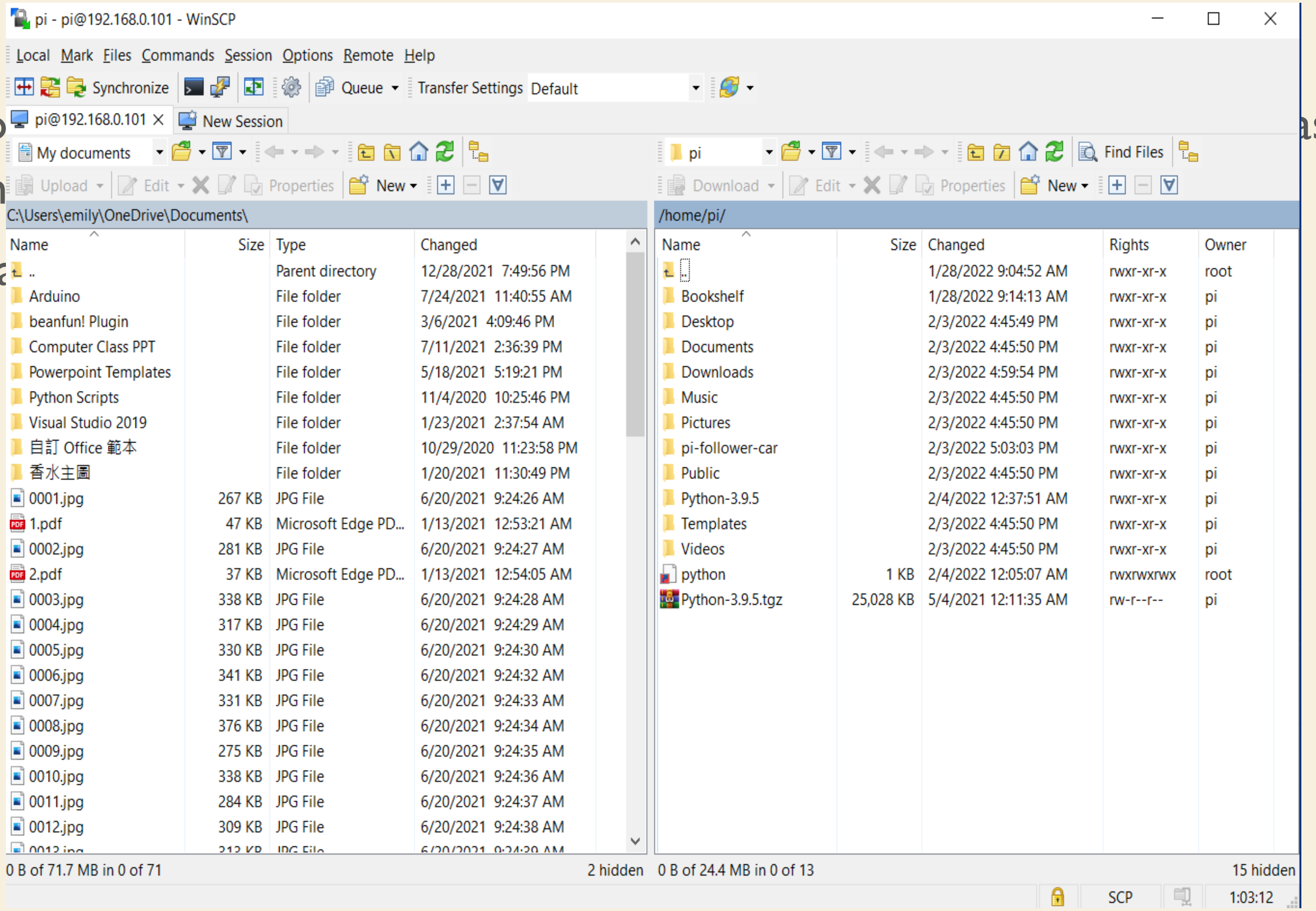
PRESS ENTER AND IT WILL ASK IF YOU WANT TO ENABLE IT

SETTING UP

- Once you enter yes, it will confirm
- Next we close it and reboot
- For our camera, we need to download winscp from this link:
- https://winscp.net/eng/download.php

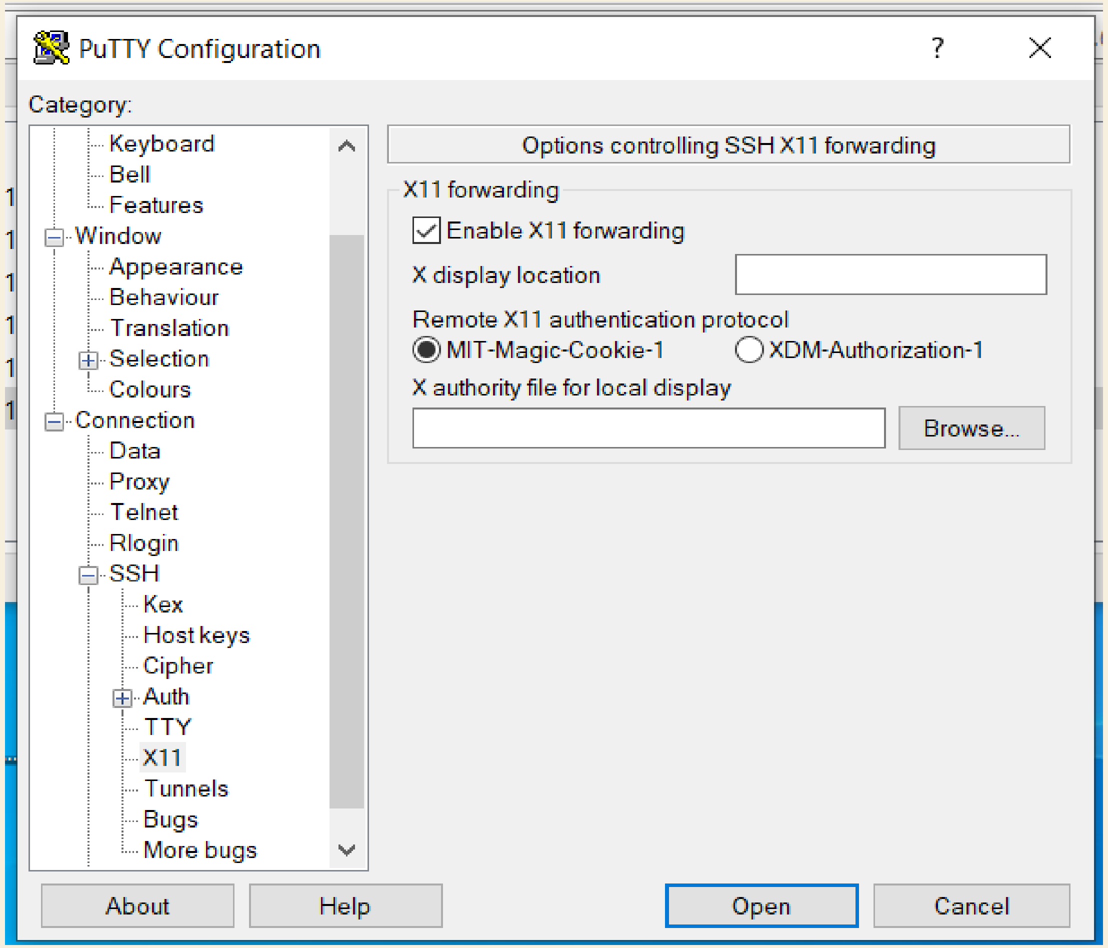
SENDING PI'S FILE BACK TO YOUR OWN PC

- After do … aspberry pi and usernam …
- After tha …



USING X11 FORWARDING

- Next we need to download another program
- https://sourceforge.net/projects/xming/
- After download, go to putty and on the left side, there is a menu
- Click on SSH, then X11, and check enable X11 forwarding

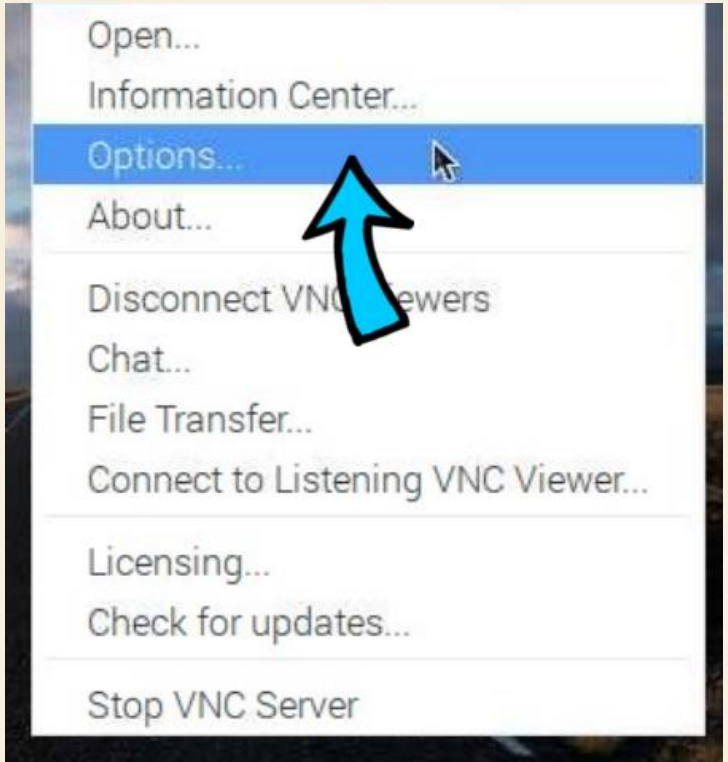


USING X11 FORWARDING

- Putty should open to pi's login page, enter your user and password
- Next type in the command:
  - gpicview test.jpg
- But technically, if you don't have a test.jpg then there won't be a picture. You can use a picture to test it out

DIRECT CAPTURE SET UP

- Reboot raspberry pi
- Use VNC to log in but this time add 5900 behind your ip address
  - Ex: 192.168.0.101:5900
- Go to options in your raspberry pi desktop
- Right click and you should be able to see it
- Go to troubleshoot on the left side of the menu
- Enable direct capture mode



DIRECT CAPTURE MODE FOR CAMERA

- Then, we can run a command called:
  - Raspistill
- We'll try two commands:
  - raspistill -o test.jpg
  - raspistill -t 3000 -o test.png -e png -w 640 -h 480
- This will allow your camera to take pictures
- To check you can go to files and you'll see the images you took

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

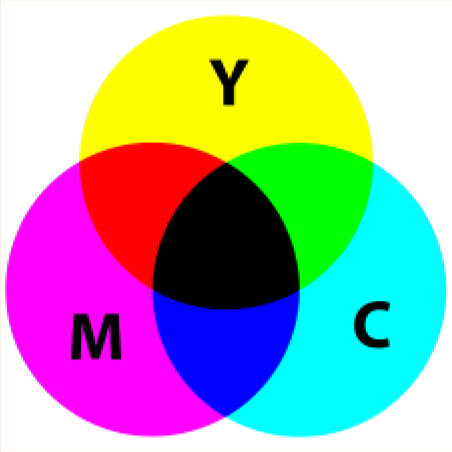
- Specific organization of colors
- RGB
- CMY
- YUV
- HSV

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)

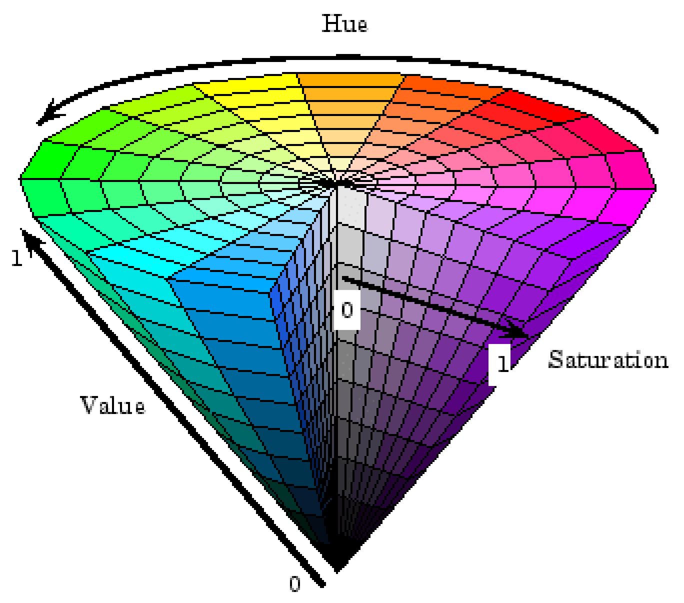
CMY(K)

- 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

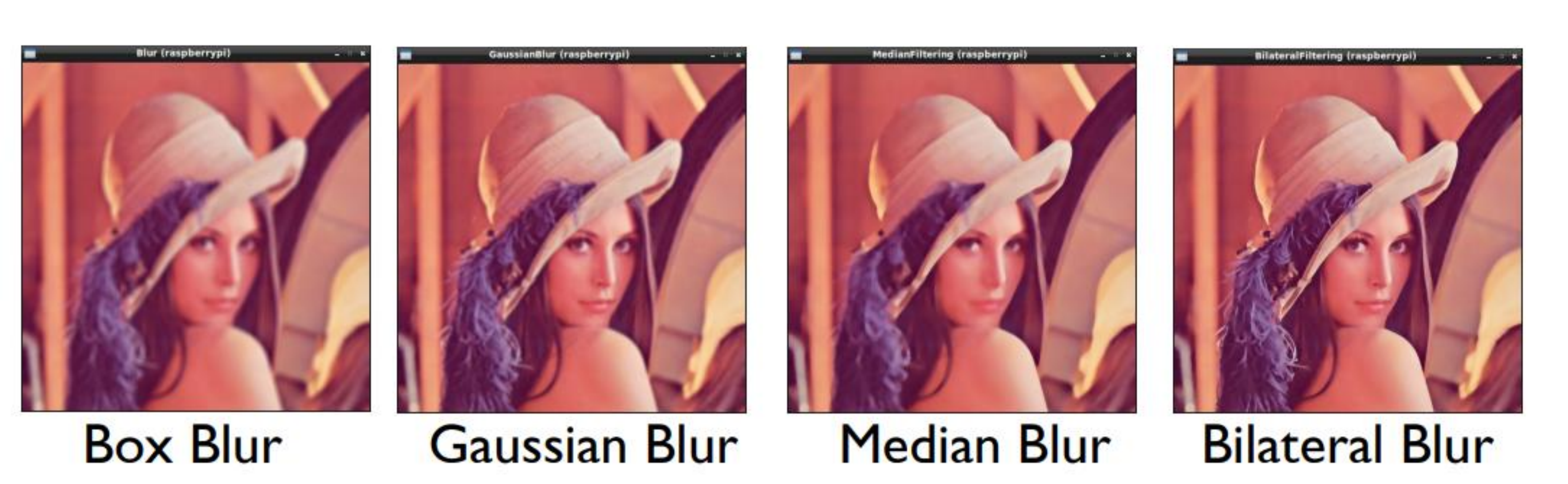
```
# Convert BGR to Gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)
cv2.waitKey(0)

# 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.

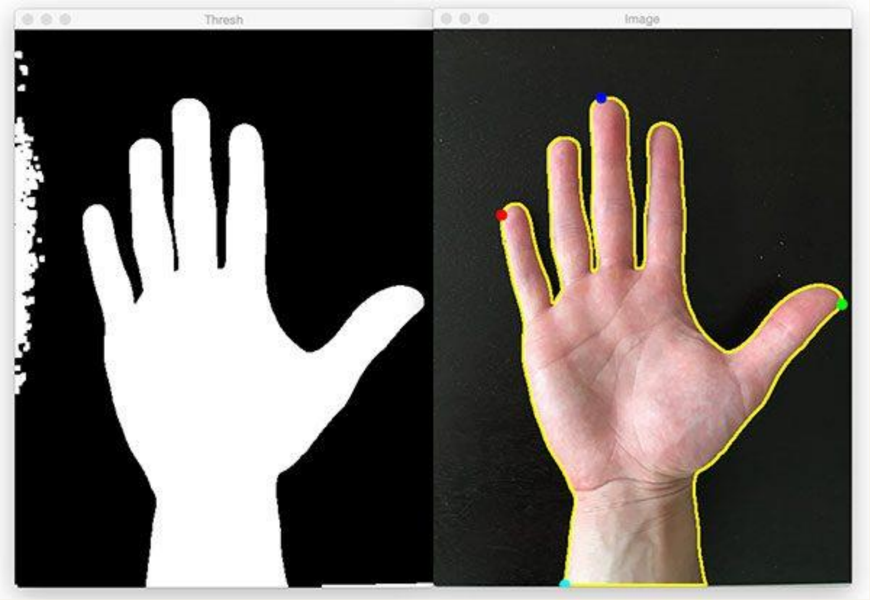
FILTERS

- 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