THE UNIVERSITY OF TENNESSEE CHATTANOOGA

Cameras and OpenCV

Mckenzie Mack
GenCyber Workshop

Agenda

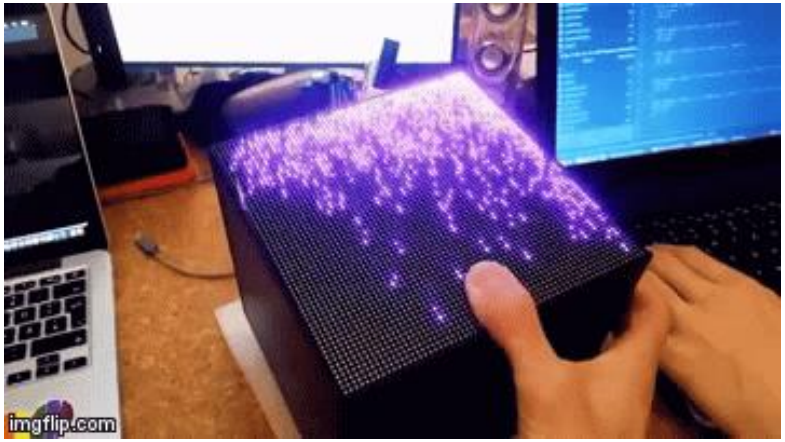
- Camera Introduction
- Camera Setup
- OpenCV
- Basic Functions of OpenCV

Learning Objectives

- Describe how to attach and set up a camera to take photos and videos with the Raspberry Pi
- Explore the many uses of OpenCV
- Explain how to use simple functions included in the OpenCV library to load and view an image or video

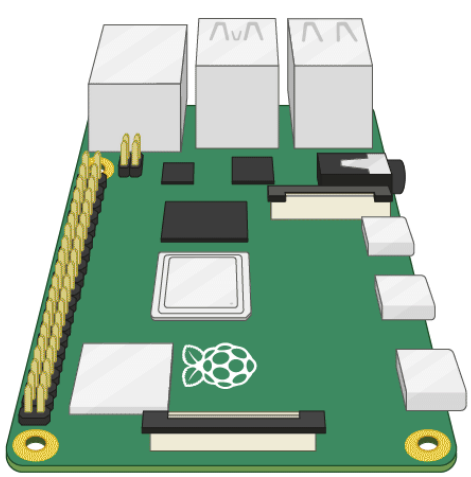
Camera Introduction

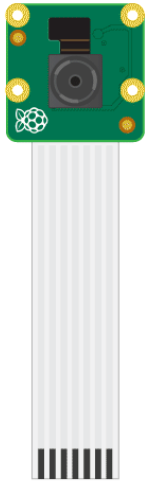
- Add-on components can be used along with the Raspberry Pi to enhance its capabilities
  - LED screens
  - motion sensors
  - cameras
- When combined with Python scripts, the Arducam Auto Focus Camera can be used to capture both photos and videos



Camera Setup

- To install the camera:
  - Lift the black tab next to the A/V port
  - Insert the camera ribbon into the opening
    - **be sure the blue side is facing the USB ports**
  - press down evenly on the tabs to secure the ribbon in place





Camera Setup

- From there, use **sudo raspi-config** to access the Pi Software Configuration Tool
  - Choose Option 3, then P1 to enable connection to the camera
  - Reboot the computer
- The camera can now be used to take pictures and videos through the command line or through a Python script

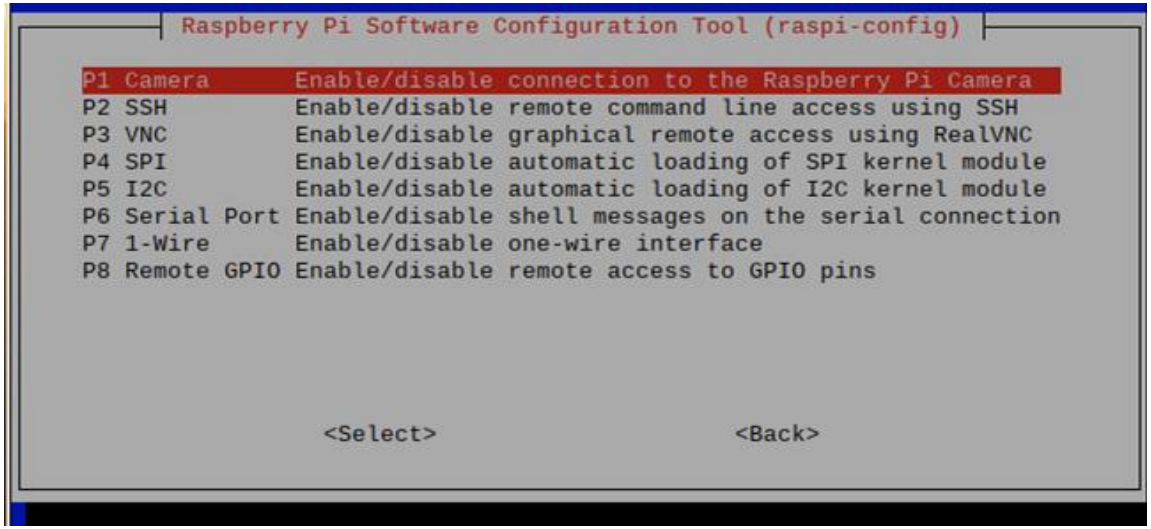


Image Capture

- A simple way to capture an image using Python is by utilizing the **PiCamera package**

```
1  from picamera import PiCamera
2  import time
3
4  camera = PiCamera()              #creates camera object to control camera
5
6  camera.start_preview()           #starts camera preview before image capture
7  time.sleep(2)                    #tells camera to wait 2 seconds before capturing picture
8
9  camera.capture("newPhoto.jpg")   #camera takes picture and saves it as newPhoto.jpg
10
11 camera.stop_preview()            #stops camera preview
```

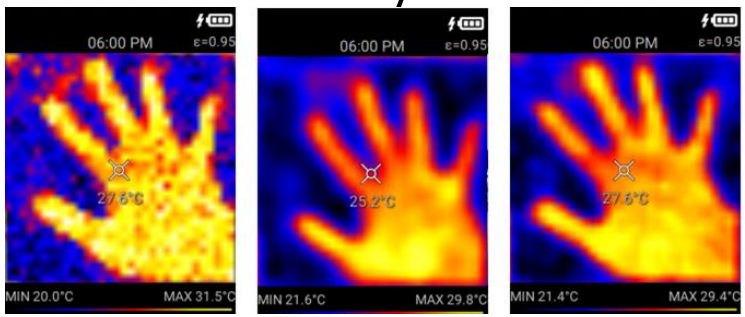
Video Capture

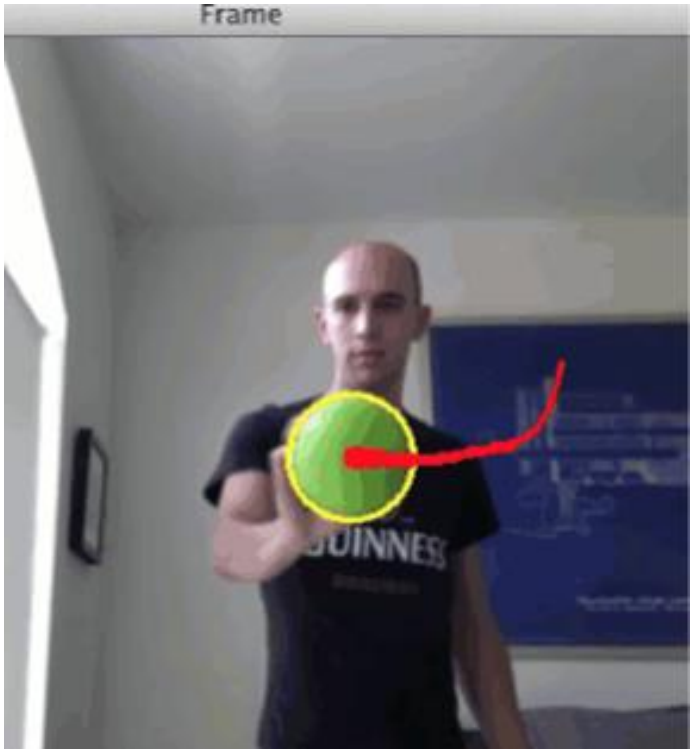
- Videos can be captured in a similar way:

```
1  from picamera import PiCamera                #imports PiCamera package
2  import time                                  #imports time package
3
4  camera = PiCamera()                          #creates camera object to control camera
5
6  camera.start_preview()                       #starts camera preview before video capture
7  camera.start_recording("Desktop/newVid2.h264") #starts recording video and saves
8                                               #it as newVid2.h264 on the Desktop
9  time.sleep(10)                               #tells program to wait 10 seconds before
10                                              #executing next line
11 camera.stop_recording()                      #stops video recording
12 camera.stop_preview()                        #stops camera preview
```

OpenCV

- These images and videos can then be analyzed using **OpenCV**
  - OpenCV: library of functions aimed at automating tasks that the human visual system performs
  - applications of OpenCV include:
    - image filtering
    - facial detection
    - facial recognition
    - motion tracking
    - object recognition
    - and many more!





Basic Functions of OpenCV

- **imread()** can be used to load an image from a specific file
  - format: **cv2.imread(*path to file*, *flag for color*)**
    - flags for color include:
      - color (any transparency is ignored): 1
      - grayscale: 0
      - unchanged: -1
- To resize the image, **resize()** can be used
  - format: **cv2.resize(*image*, (*width in pixels*, *height in pixels*))**
  - to resize by a certain scale: **cv2.resize(*file path*, *(0,0)*, *fx = scale to resize x-axis*, *fy = scale to resize y-axis*)**
    - ex: **newImg = cv2.resize("Desktop/me.jpeg", (0,0), fx = 0.75, fy = 0.75)**

      ^resizes image to ¾ its original size

Basic Functions of OpenCV

- Once the image has been loaded, any changes made to the image can be saved using **imwrite()**
  - format: **cv2.imwrite(*path to output file*, *modified source image*)**

```
import cv2

# read image as grey scale
grey_img = cv2.imread('/home/img/python.png', cv2.IMREAD_GRAYSCALE)

# save image
status = cv2.imwrite('/home/img/python_grey.png',grey_img)

print("Image written to file-system : ",status)
```

Basic Functions of OpenCV

- After the filters or modifications have been saved, **imshow()** can be used to show the image to the user
  - format: **cv2.imshow(*title of window*, *image*)**
  - **cv2.waitKey(0)** shows the image to the user an infinite amount of time until the user hits any key on the keyboard
  - **cv2.destroyAllWindows()** closes the window that holds the

```
cv2.imshow('Image',source)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Basic Functions of OpenCV

- Viewing videos using OpenCV is a little more complicated
  - **VideoCapture()** is used to load the video
    - format: **cv2.VideoCapture(*file path*)**
  - **release()** is used to close the video
  - instead of displaying the entire video at once, the video is displayed on a frame-by-frame basis
    - You will look at how to code this in the lab!

```
source = cv2.VideoCapture("Desktop/newVid.h264")
```

```
source.release()
cv2.destroyAllWindows()
```

Resources

- https://www.youtube.com/watch?v=nx8gDSS1vO4
- https://opencv.org/
- https://www.tutorialkart.com/opencv/python/opencv-python-save-image-example/
- https://www.youtube.com/watch?v=_snMNl5dplc

LAB

complete the questions for the lab - Images, Filters, and Cybersecurity