Vision Programming for Intelligent Autonomous Robot Using ROS

IR DR ZOOL HILMI ISMAIL & INNOKAI TEAM

OUR SCHEDULE

DAY 1

Session 1 : Revision and Simple Command on Linux

- Quick Flashback on Phase 1
- Task Given, Creating package, workspace
- Revise back some useful command in Linux

Session 2: Revise back about Python

- Revise back basic of python
- Some exercises to test programming skills

Session 3: Revise back about OpenCV

- Detecting Edges and Applying Image Filters
- Blurring
- Edge detection
- Converting between different color spaces

OUR SCHEDULE

DAY 2

Session 4: Continue OpenCV

- Motion blur
- Sharpening
- Embossing
- Object Tracking

Session 5: Installation of YOLO

- Introduction to YOLO Bbox Annotation Tool
- Installation the Tool on LinuxOS (Laptop)

Session 6: : Labelling and Training model

- Do the labelling
- Train the model by using Laptop

OUR SCHEDULE

DAY 3

Session 7: Programming with ROS

- Follow-bot
- Hands-on on the TurtleBot3

LET'S START CODING... SESSION 1

Task

- ✓ Create a workspace in home
- ✓ Workspace name is innokai_opencv_ws

- ✓ Create a package with stds_msgs, rospy and roscpp
- ✓ Package must be ros_basic

Getting into Linux

Installation of packages and libraries

- From Github
- Git clone into src folder in workspace

Revise back about Python

SESSION 2


Python
2.7
3
Python

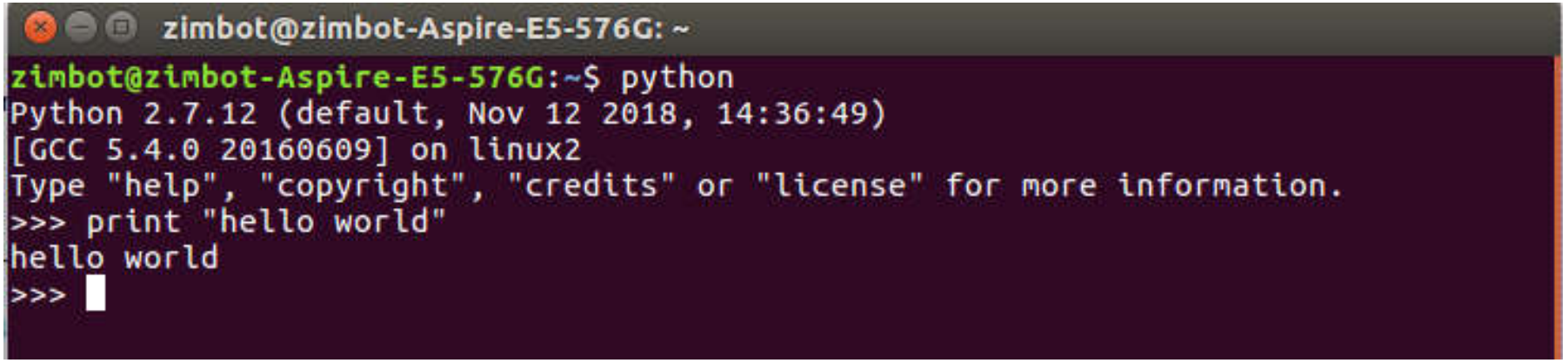
Basic of Programming

1. Pseudo Code
   - A simple description of a computer algorithms using a combination of natural language and programming language.
   - In other words, this is description of your code before you start coding
2. Flowchart
   - Flowcharts are written with program flow from the top of a page to the bottom

Basic of Python

- Classes
- Function
- Variable
- Loop
- Mathematical operationn

First Try in Python Programming
HELLO World



Types of Data

- String
- Char
- Number
  - Float
  - Double
  - Int
- Boolean

Class

1. Classes are a way of combining information and behavior.

```
class Rocket():
    # Rocket simulates a rocket ship for a game,
    #  or a physics simulation.

    def __init__(self):
        # Each rocket has an (x,y) position.
        self.x = 0
        self.y = 0

    def move_up(self):
        # Increment the y-position of the rocket.
        self.y += 1

# Create a Rocket object.
my_rocket = Rocket()
print(my_rocket)
```

Exercise

Use **Class** to create a simple calculator which can perform:

- Additional
- Subtraction

OpenCV

OpenCV

OpenCV is a library of programming functions mainly aimed at real-time computer vision

We may use OpenCV without ROS.

In this workshop, we will discuss about the usage of OpenCV

Revision to OpenCV

Resizing images

>> resized = cv2.resize(image, (200, 200))

Resizing in ratio

- Using imutils

In a new terminal to install imutils

>> pip install imutils --user

Rotating an image

```
>> rotated = imutils.rotate(image, -45)
```

Rotate without clipping

```
>> rotated = imutils.rotate_bound(image, 45)
```

Smoothing an image

```
>> blurred = cv2.GaussianBlur(image, (11, 11), 0)
```

Detecting Edges and Image Filtering

SESSION 3

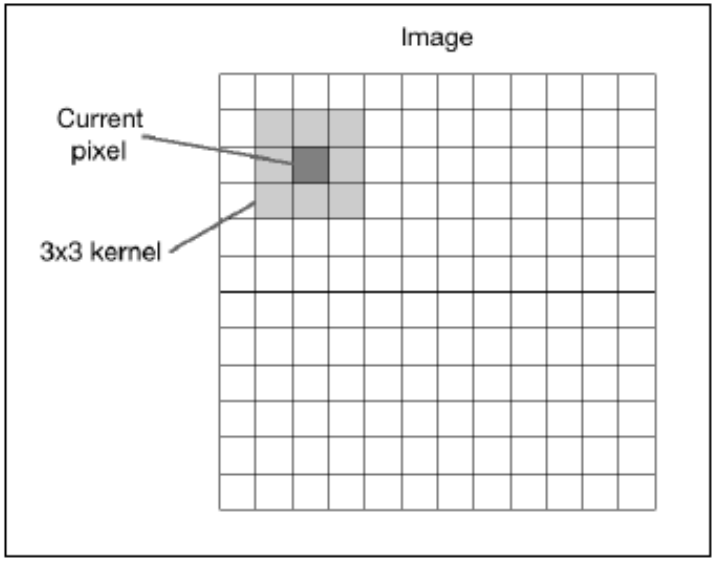
In this session, we are going to see

- How to **blur** an image
- How to **detect edges** in an image
- How to **apply motion blur** to an image
- How to **sharpen** an image
- How to **erode** and **dilate** an image
- How to create a **vignette filter**
- How to enhance **image contrast**

2D convolution

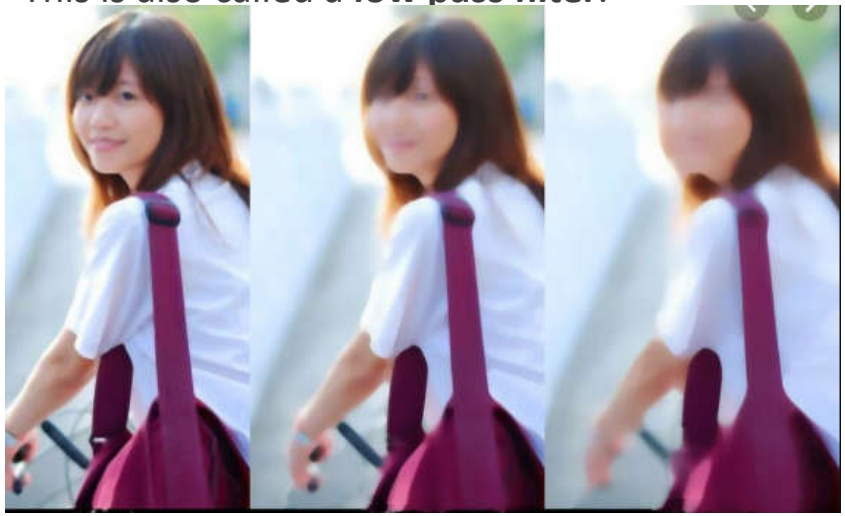
- Convolution is a fundamental operation in image processing.
- **Kernel = matrix**.
- Here, the kernel is called the "image filter" and the process of applying this kernel to the given image is called "image filtering".

$$I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Blurring

This is also called a **low pass filter**.



blur(simple average) medianBlur GaussianBlur()

Several ways to blur an image

Averaging

blur = cv.blur(img,(5,5))

Gaussian Blurring

blur = cv.GaussianBlur(img,(5,5),0)

Median Blurring

median = cv.medianBlur(img,5)

Bilateral Filtering

blur = cv.bilateralFilter(img,9,75,75)

Converting between different color spaces

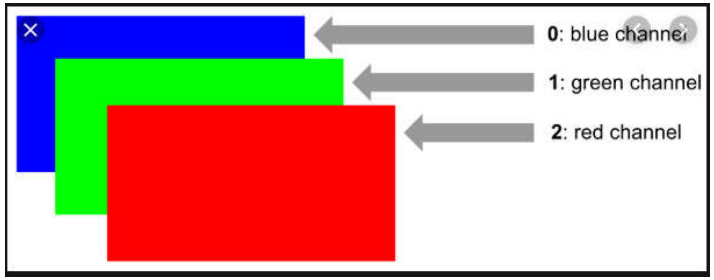
In general, three color spaces are prevalent in modern day computer vision: gray, BGR, and **Hue, Saturation, Value** (**HSV**).

Gray :

- ✓ eliminates color information translating to shades of gray:
- ✓ useful for intermediate processing, such as face detection.

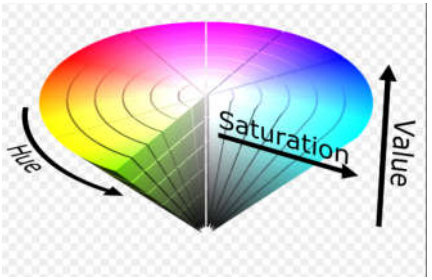
BGR : blue-green-red color space

- ✓ each pixel is a three-element array
- ✓ web developers would be familiar with a similar definition of colors, except the order of colors is RGB.



HSV :

- ✓ hue is a color tone, saturation is the intensity of a color, and value represents its darkness (or brightness at the opposite end of the spectrum).



COLOR CALCULATOR

Click link to open the website

https://www.tydac.ch/color/