DAILY ASSESSMENT FORMAT

Date:	30/05/2020	Name:	Shilpa N
Course:	Logic Design	USN:	4AL16EC071
Topic:	DAY 3	Semester & Section:	8 “B”
Github Repository:	Shilpan-test		

FORENOON SESSION DETAILS

Report – Report can be typed or hand written for up to two pages.

PLC Analog I/O and Network I/O

In the early days of programmable logic controllers, processor speed and memory were too limited to support anything but discrete (on/off) control functions.
Consequently, the only I/O capability found on early PLCs were discrete in nature. Modern PLC technology, though, is powerful enough to support the measurement, processing, and output of analog (continuously variable) signals.

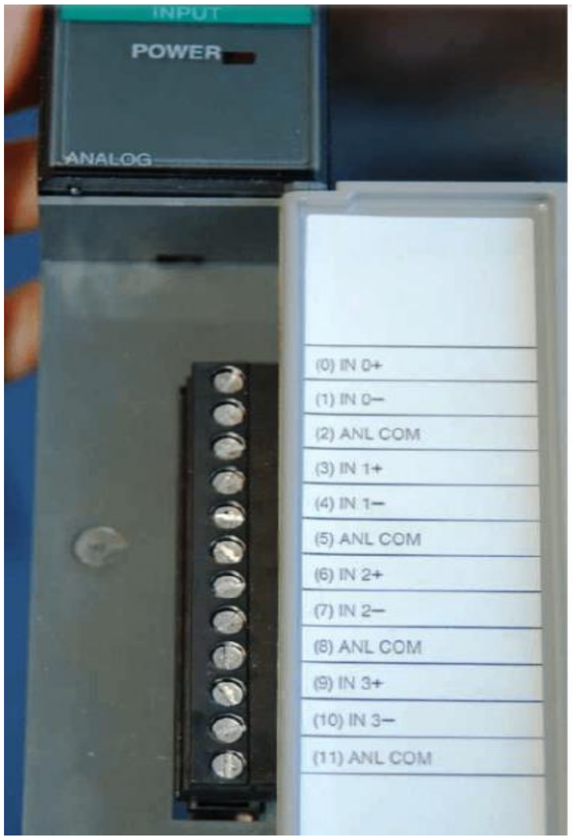
All PLCs are digital devices at heart. Thus, in order to interface with an analog sensor or control device, some “translation” is necessary between the analog and digital worlds. Inside every analog input module is an ADC, or Analog-to-Digital Converter, circuit designed to convert an analog electrical signal into a multi-bit binary word.
Conversely, every analog output module contains a DAC, or Digital-to-Analog Converter, circuit to convert the PLC’s digital command words into analog electrical quantities.

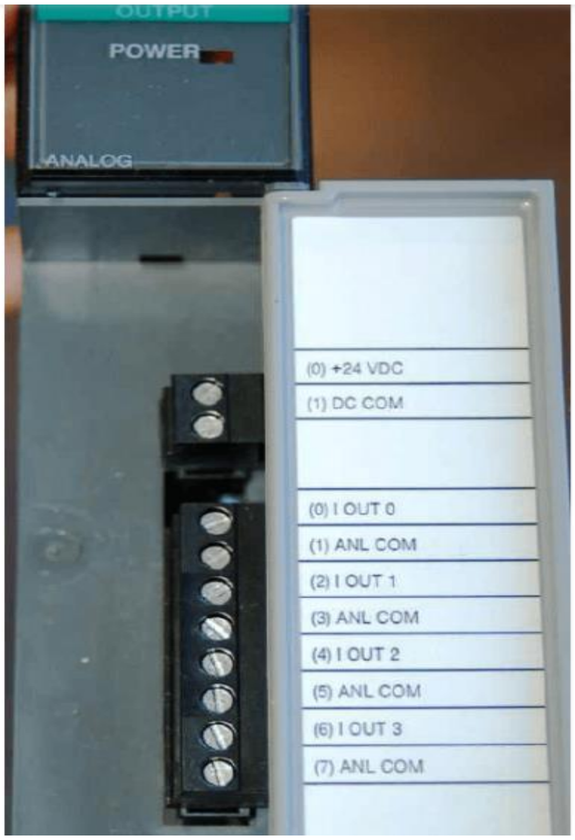
Analog I/O is commonly available for modular PLCs for many different analog signal types, including:

- Voltage (0 to 10 volt, 0 to 5 volt)
- Current (0 to 20 mA, 4 to 20 mA)
- Thermocouple (millivoltage)
- RTD
- Strain gauge
-

PLC Analog I/O

The following photographs show two analog I/O cards for an Allen-Bradley SLC 500 modular PLC system, an analog input card and an analog output card. Labels on the terminal cover doors indicate screw terminal assignments:





PLC Network I/O

Many different digital network standards exist for PLCs to communicate with, from PLC to PLC and between PLCs and field devices.

One of the earliest digital protocols developed for PLC communication was Modbus, originally for the Modicon brand of PLC.

Modbus was adopted by other PLC and industrial device manufacturers as a de facto standard, and remains perhaps the most universal digital protocol available for industrial digital devices today.

Another digital network standard developed by a particular manufacturer and later adopted as a de facto standard is Profibus, originally developed by Siemens.

A Programmable Logic Controller, also called a PLC or programmable controller, is a computer-type device used to control equipment in an industrial facility.
The kinds of equipment that PLCs can control are as varied as industrial facilities themselves. Utility Plants, Batch Control Application, Chemical Processing, Conveyor systems, food processing machinery, auto assembly lines etc…you name it and there's probably a PLC out there controlling it.

Modbus was adopted by other PLC and industrial device manufacturers as a de facto standard, and remains perhaps the most universal digital protocol available for industrial digital devices today. Another digital network standard developed by a particular manufacturer and later adopted as a de facto standard is Profibus, originally developed by Siemens.

A Programmable Logic Controller, also called a PLC or programmable controller, is a computer-type device used to control equipment in an industrial facility.
The kinds of equipment that PLCs can control are as varied as industrial facilities themselves. Utility Plants, Batch Control Application, Chemical Processing, Conveyor systems, food processing machinery, auto assembly lines etc…you name it and there's probably a PLC out there controlling it.

PLC Advantages

In addition to the programming flexibility we just mentioned, PLCs offer other advantages over traditional control systems.

These advantages include:

- high reliability
- small space requirements
- computing capabilities
- reduced costs
- ability to withstand harsh environments
- expandability

A PLC basically consists of two elements:

1. CENTRAL PROCESSING UNIT
2. INPUT/OUTPUT SYSTEM

The Central Processing Unit

The central processing unit (CPU) is the part of a programmable controller that retrieves, decodes, stores, and processes information. It also executes the control program stored in the PLC's memory. In essence, the CPU is the "brains" of a programmable controller.

It functions much the same way the CPU of a regular computer does, except that it uses special instructions and coding to perform its functions.

The CPU has three parts:

- the processor
- the memory system
- the power supply

The processor is the section of the CPU that codes, decodes, and computes data. Memory system is the section of the CPU that stores both the control program and data from the equipment connected to the PLC. Power supply is the section that provides the PLC with the voltage and current it needs to operate.

The Input/ Output System

The input/output (I/O) system is the section of a PLC to which all of the field devices are connected. If the CPU can be thought of as the brains of a PLC, then the I/O system can be thought of as the arms and legs. The I/O system is what actually physically carries out the control commands from the program stored in the PLC's memory.

The I/O system consists of two main parts:

- **the Rack**
- **I/O modules**

The rack is an enclosure with slots in it that is connected to the CPU. I/O modules are devices with connection terminals to which the field devices are wired. Together, the rack and the I/O modules form the interface between the field devices and the PLC.

When set up properly, each I/O module is both securely wired to its corresponding field devices and securely installed in a slot in the rack. This creates the physical connection between the field equipment and the PLC. In some small PLCs, the rack and the I/O modules come prepackaged as one unit.

Date:	29/05/2020	Name:	Shilpa N
Course:	PYTHON	USN:	4AL16EC071
Topic:	DAY 12	Semester & Section:	8 “B”
AFTERNOON SESSION DETAILS			

Python – Process images of a video using OpenCV

Processing a video means, performing operations on the video frame by frame. Frames are nothing but just the particular instance of the video in a single point of time. We may have multiple frames even in a single second. Frames can be treated as similar to an image.

So, whatever operations we can perform on images can be performed on frames as well. Let us see some of the operations with examples.

Adaptive Threshold –

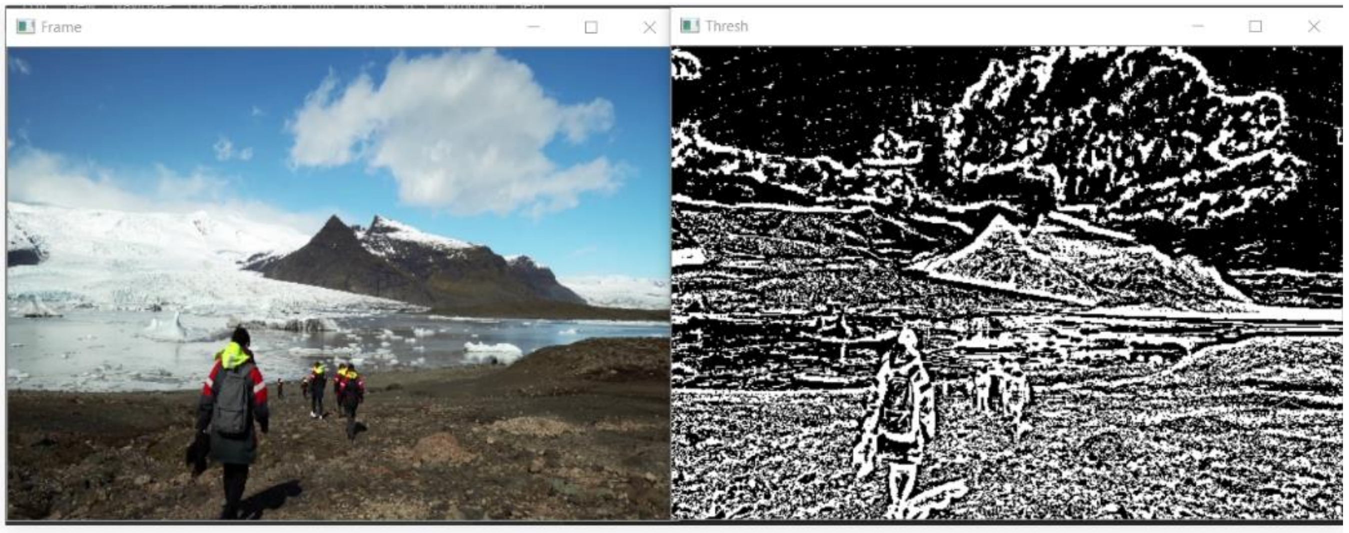
By using this technique we can apply thresholding on small regions of the frame. So the collective value will be different for the whole frame.

```
# importing the necessary libraries
import cv2
import numpy as np
```

```
# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture('sample.mp4')



# Loop untill the end of the video
while (cap.isOpened()):

        # Capture frame-by-frame
        ret, frame = cap.read()
        frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                                         interpolation = cv2.INTER_CUBIC)

        # Display the resulting frame
        cv2.imshow('Frame', frame)

        # conversion of BGR to grayscale is necessary to apply this operation
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # adaptive thresholding to use different threshold
        # values on different regions of the frame.
        Thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,

        cv2.THRESH_BINARY_INV, 11, 2)

        cv2.imshow('Thresh', Thresh)
        # define q as the exit button
        if cv2.waitKey(25) & 0xFF == ord('q'):
                break

# release the video capture object
cap.release()
```

```
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

Output:



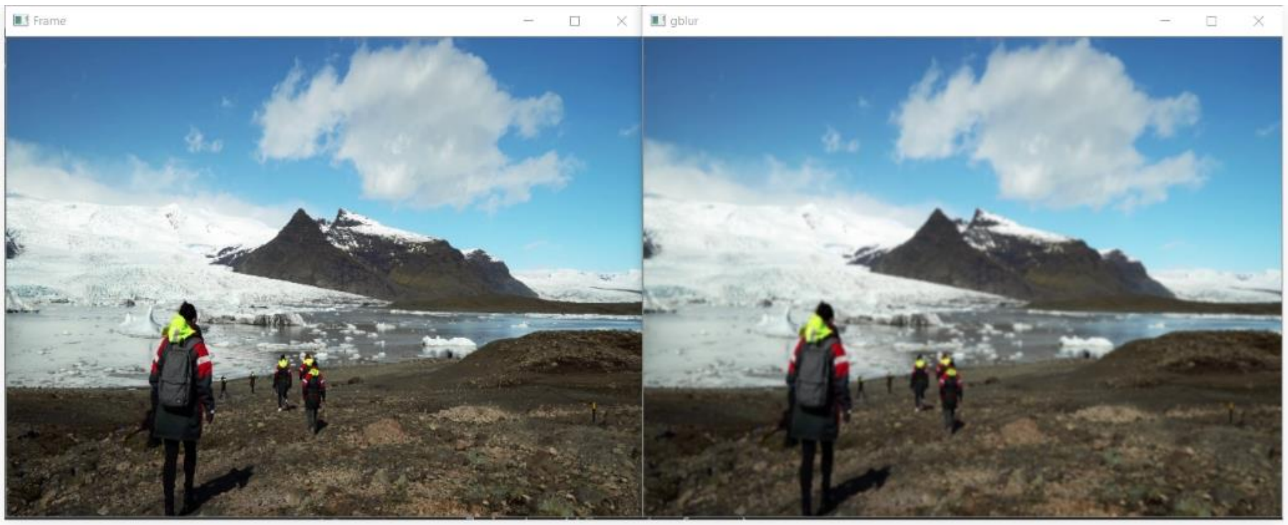
Smoothing

Smoothing a video means removing the sharpness of the video and providing a blurriness to the video. There are various methods for smoothing such as cv2.Gaussianblur(), cv2.medianBlur(), cv2.bilateralFilter(). For our purpose, we are going to use cv2.Gaussianblur().

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture('sample.mp4')



# Loop untill the end of the video
while (cap.isOpened()):
	# Capture frame-by-frame
	ret, frame = cap.read()
```

```
        frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                                                     interpolation = cv2.INTER_CUBIC)

        # Display the resulting frame
        cv2.imshow('Frame', frame)

        # using cv2.Gaussianblur() method to blur the video

        # (5, 5) is the kernel size for blurring.
        gaussianblur = cv2.GaussianBlur(frame, (5, 5), 0)
        cv2.imshow('gblur', gaussianblur)

        # define q as the exit button
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

# release the video capture object
cap.release()

# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

Output:



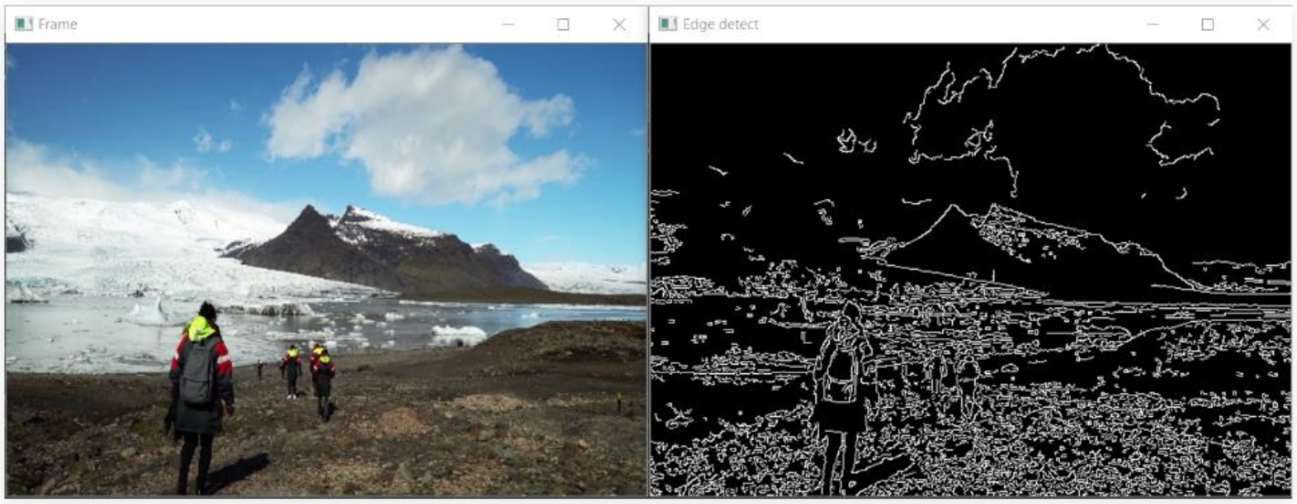
Edge Detection –

Edge detection is a useful technique to detect he edges of surfaces and objects in the video. Edge detection involves the following steps:

1. Noise reduction
2. Gradient calculation
3. Non-maximum suppression
4. Double threshold
5. Edge tracking by hysteresis

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture('sample.mp4')



# Loop untill the end of the video
while (cap.isOpened()):
        # Capture frame-by-frame
        ret, frame = cap.read()

        frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                                                    interpolation = cv2.INTER_CUBIC)

        # Display the resulting frame
        cv2.imshow('Frame', frame)

        # using cv2.Canny() for edge detection.
        edge_detect = cv2.Canny(frame, 100, 200)
        cv2.imshow('Edge detect', edge_detect)

        # define q as the exit button
        if cv2.waitKey(25) & 0xFF == ord('q'):
                break

# release the video capture object
cap.release()
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

Output:


Frame
Edge detect