MOTION DETECTION USING OPENCV

A Mini Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

by

Namburi Sai Varun Kumar	**16N31A05F2**
Makutam Srujana Reddy	**16N31A05C2**
R.Ganga Krishnan	**16N31A05H7**

Under the esteemed guidance of

N. Siva Kumar
Assistant Professor

Department of Computer Science and Engineering

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2019-2020

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled " Online Attendance Management System", submitted by N. Sai Varun Kumar(16N31A05F2), M. Srujana Reddy(16N31A05C2), R. Ganga Krishnan(16N31A05H7) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2019-2020. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Mr. N. Siva kumar	**Dr. D. Sujatha**
Assistant Professor	**Professor**

External Examiner

DECLARATION

We hereby declare that the project titled "Motion Detection" submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Namburi Sai Varun Kumar - 16N31A05F2

Makutam Srujana Reddy - 16N31A05C2

R.Ganga Krishnan - 16N31A05H7

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) and our principal Dr. VSK Reddy who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. D. Sujatha for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide and Project Coordinator Dr. S. Shanthi for her regular guidance and constant encouragement. We are extremely grateful to her valuable suggestions and unflinching co-operation throughout project work.

We would like to thank our class in charge Mr. M. Samba Sivudu who in spite of being busy with his duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude

Namburi Sai Varun Kumar - 16N31A0F2

Makutam Srujana Reddy - 16N31A05C2

R. Ganga Krishnan - 16N31A05H7

ABSTRACT

Today with the increasing advancement in technology, a growing concern for safety and security is arising everywhere. To address this concern, the numbers of surveillance cameras have increased in the recent past. Data collected is nevertheless difficult to store and monitor manually on a continual basis. There are several approaches to do this job without human intervention. The underlying principle of all these methods is detection, segmentation and tracking objects in the live video.

Human brain can easily understand the visual frames which seen by eyes, but computers cannot do it without any special methodology. If we compare human and computer, camera module is the eye of the computer and the computer vision is the way that how brain understand the image. Computer vision is a field that includes some methods, processing and analyzing techniques for understanding images. Also known as Image Analysis, Scene Analysis and Image Understanding.

Computer vision is such kind of research field which tries to percept and represent the 3D information for world objects. Its essence is to reconstruct the visual aspects of 3D object by analyzing the 2D information extracted accordingly. 3D objects surface reconstruction and representation not only provide theoretical benefits, but also are required by numerous applications. Background subtraction is critical in many computer vision applications. We use it to count the number of cars passing through a toll booth. We use it to count the number of people walking in and out of a store.

TABLE OF CONTENTS

1. INTRODUCTION

Motion detection is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. Motion detection can be achieved by either mechanical or electronic methods. When motion detection is accomplished by natural organisms, it is called motion perception.

In video_surveillance, motion detection refers to the capability of the surveillance_system to detect motion and capture the events. Motion detection is usually a software-based monitoring algorithm which, when it detects motions will signal the surveillance camera to begin capturing the event. Also called activity detection. An advanced motion detection surveillance system can analyze the type of motion to see if it warrants an alarm.

1.1. Purpose:

This project called “motion detection using openCV” which is detects sense a human motion and then transmit the signal wirelessly. However, this project will relate to auto power ON light and fan system. When the sensor senses a human motion in the sensor’s detection area, sensor will be triggered and then the room’s light will automatically switch ON. It is useful for us when we cannot find the switch in the dark condition. [1] For the fan’s function, it is depends on the room temperature, when the temperature is higher, fan will run when the PIR had detect motion in the detection area. When the room temperature is low, fan will not run. Degree of temperature is measure by the temperature sensor and temperature will show on a LCD display (2x16). Light and fan will automatically OFF when the user was going out from the room. As long as PIR sensor does not detect motion in the detection area, light and fans are not function and the fan is depends on the room temperature. Once the sensor is triggered, system will have around 2 minutes to run the function. After 2 minutes and sensor does not detect any motion, light and fans will be switched OFF automatically.

Objective :

The main objective for this project is to ensure that the sensor’s signal will transmit wirelessly .When the sensor is triggered and the signal will transmit wirelessly and the light will switch ON automatically. Besides that, PIC also needs to ensure can function well in order to control the circuit for receive signal and switch ON the light or fan. It is also a green project since it can save a lot of electric energy in the long term. Users may forget to switch OFF the light or fan when they are going out. With this system, they do not have to worry about it because when they step out the room, light and fan will switch OFF automatically. This project can be implemented in several places, such as bedroom, office or others.

The remainder of this article will detail how to build a basic motion detection and tracking system for home surveillance using computer vision techniques. This example will work with both pre-recorded videos and live streams from your webcam; however, we'll be developing this system on our laptops/desktops.

1.2 A little bit about background subtraction:

Background subtraction is critical in many computer vision applications. We use it to count the number of cars passing through a toll booth. We use it to count the number of people walking in and out of a store.

Before we get started coding in this post, let me say that there are many, *many* ways to perform motion detection, tracking, and analysis in OpenCV. Some are very simple. And others are very complicated. The two primary methods are forms of Gaussian Mixture Model-based foreground and background segmentation.

1. An improved adaptive background mixture model for real-time tracking with shadow detection by KaewTraKulPong et al., available through thecv2. Background Subtractor MOG function.
2. Improved adaptive Gaussian mixture model for background subtraction by Zivkovic, and Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction, also by Zivkovic, available through the cv2.BackgroundSubtractorMOG2 function.

And in newer versions of OpenCV we have Bayesian (probability) based foreground and background segmentation, implemented from Godbehere et al.'s 2012 paper, Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation. We can find this implementation in the cv2.createBackgroundSubtractorGMG function (we'll be waiting for OpenCV 3 to fully play with this function though).

All of these methods are concerned with segmenting the background from the foreground (and they even provide mechanisms for us to discern between actual motion and just shadowing and small lighting changes)!

So why is this so important? And why do we care what pixels belong to the foreground and what pixels are part of the background?

Well, in motion detection, we tend to make the following assumption:

The background of our video stream is largely static and unchanging over consecutive frames of a video. Therefore, if we can model the background, we monitor it for substantial changes. If there is a substantial change, we can detect it — this change normally corresponds to motion on our video.

Now obviously in the real-world this assumption can easily fail. Due to shadowing, reflections, lighting conditions, and any other possible change in the environment, our background can look quite different in various frames of a video. And if the background appears to be different, it can throw our algorithms off. That's why the most successful background subtraction/foreground detection systems utilize fixed mounted cameras and in controlled lighting conditions.

The methods I mentioned above, while very powerful, are also computationally expensive. And since our end goal is to deploy this system to a Raspberry Pi at the end of this 2 part series, it's best that we stick to simple approaches. We'll return to these more powerful methods in future blog posts, but for the time being we are going to keep it simple and efficient.

In the rest of this blog post, I'm going to detail (arguably) the most basic motion detection and tracking system you can build. It won't be perfect, but it will be able to run on a Pi and still deliver good results.

Project overview:

The detection of stationary or moving targets, and tracking them on a real-time video streams is a very important and challenging task in order to protect fields from enemies. The enemies can be a human, an animal or even an object. The field is secures by drones or stationary sticks which include detection and tracking system. Video surveillance is a very popular research topic in computer vision applications that continuously tries to detect, recognize and track the targets.

1.3 Modules Description:

Motion detectors are used as security systems in banks, offices and shopping malls, and also as intruder alarm in home. The prevailing motion detectors can stop serious accidents by sensing the persons who are in close proximity to the detector. We can observe motion detectors in shopping malls or stores with automatic doors. The main element in the motion detector circuit is the dual infrared reflective sensor or any other detecting sensor.

2. SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of Audit Control including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS:

Processor	:	Intel or AMD.
Ram	:	1 GB or above.
Hard disk	:	40GB or above.

2.1.2 SOFTWARE REQUIREMENTS:

Technology/Language	:	Python,HTML5
Operating System	:	Windows, Linux, Mac.
IDE	:	Anaconda Navigator(Spyder)

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

2.2.1 SRS:

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

1) **Problem/Requirement Analysis:**

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

2) **Requirement Specification:**

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

2.2.2 ROLE OF SRS:

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

2.2.3 SCOPE:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also be the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

2.2.4 EXISTING SYSTEM:

In the existing system the monitoring is done by fixing tags in different location for identifying the exact position. The android terminal is connected to Bluetooth and wireless LAN which is slow when compared to 3G/4G distance and is limited to shorter distance. Mostly it is not specifying the exact location. Here only one model is implemented with one application.

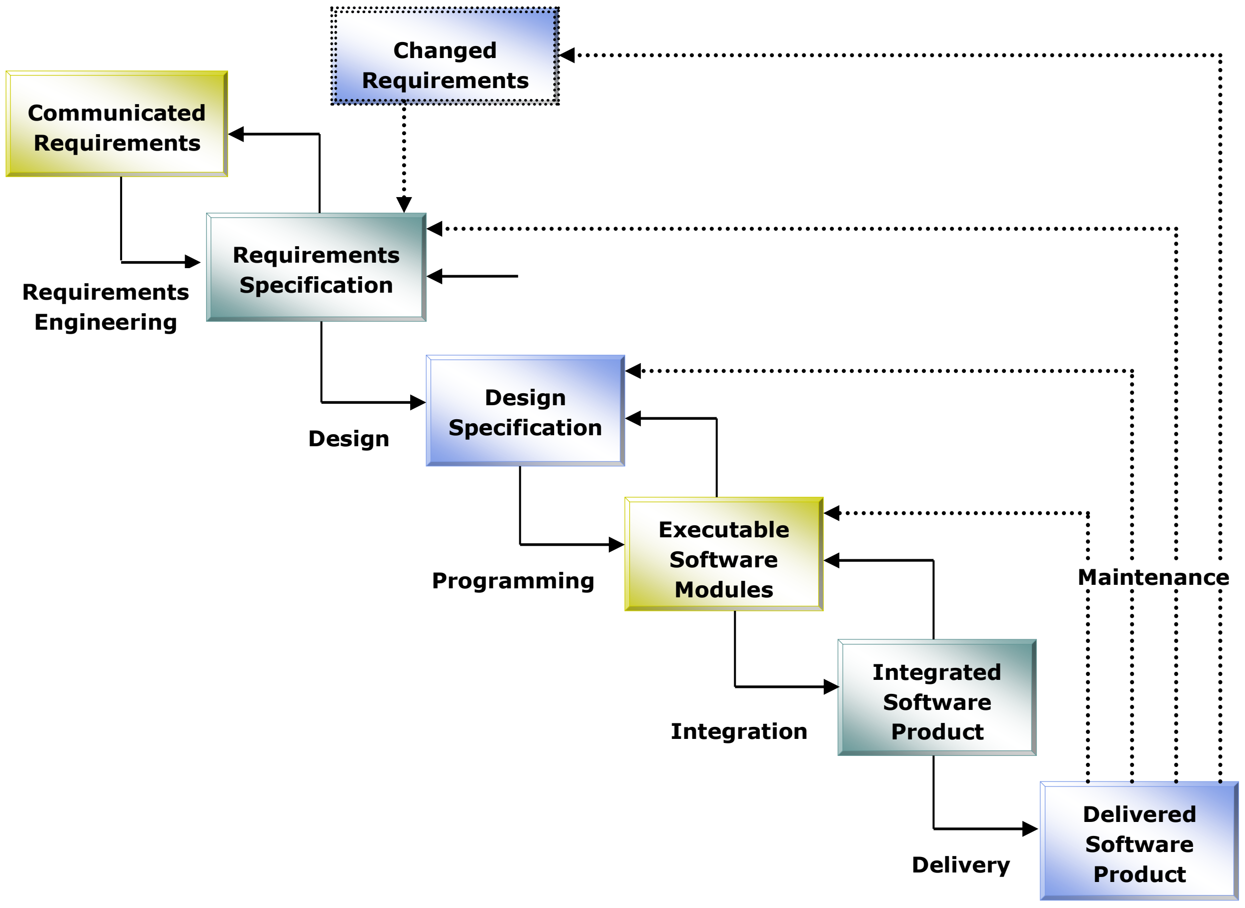

Changed Requirements
Communicated Requirements
Requirements Specification
Requirements Engineering
Design Specification
Design
Executable Software Modules
Programming
Maintenance
Integrated Software Product
Integration
Delivered Software Product
Delivery

Fig2.2: Water Fall Model

2.2.4 **DRAWBACKS OF EXISTING SYSTEM:**

- A person cannot know users (victims) current location.
- There is possibly of loss of data in between during the message transfer.

2.2.5 **PROPOSED SYSTEM:**

Different functions have been implemented for the new generation emergency system such as telephony manager and sms manager to call and send alert from the application. Android mobile terminal is connected to high speed 3G/4G/Wi-Fi network for effective data transfer. Monitoring can be made at a very high speed without any distortion in the network. Directions to near by locations can be easily obtained via GPS assistance and Google Maps with women safety features along side with audio, video functionalities. Sending the users current location as text sms and calling has become very easy in the new security system.

3. TECHNOLOGIES USED

Requirements:

1. Python3(Programming Language)
2. OpenCV (library)
3. Pandas (library)

Introduction to OpenCV-Python:

3.1 OpenCV:

OpenCV was started at Intel in 1999 by **Gary Bradsky** and the first release came out in 2000. **VadimPisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and VadimPisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Javaetc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

3.1.1 OpenCV-Python:

Python is a general purpose programming language started by **Guido van Rossum**, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpymakes the task more easier. **Numpy** is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy,

You can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

3.1.2 OpenCV-Python Tutorials:

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python. **This guide is mainly focused on OpenCV 3.x version** (although most of the tutorials will work with OpenCV 2.x also).

A prior knowledge on Python and Numpy is required before starting because they won't be covered in this guide. **Especially, a good knowledge on Numpy is must to write optimized codes in OpenCV-Python.**

This tutorial has been started by AbidRahman*K* as part of Google Summer of Code 2013 program, under the guidance of Alexander Mordvintsev.

OpenCV Needs You !!!

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also.

So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it.

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc.

As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV.

3.2 Install OpenCV-Python in Window:

3.2.1 Goals:

We will learn to setup OpenCV-Python in your Windows system. Below steps are tested in a Windows 7-64 bit machine with Visual Studio 2010 and Visual Studio 2012. The screenshots shows VS2012.

3.2.2. Installing OpenCV from prebuilt binaries:

1. Below Python packages are to be downloaded and installed to their default locations.

    1.1. Python-2.7.x.

    1.2. Numpy.

2. Install all packages into their default locations. Python will be installed to **C:/Python27/**.

3. After installation, open Python IDLE. Enter `import numpy` and make sure Numpy is working fine.

4. Download latest OpenCV release from sourceforge site and double-click to extract it.

5. Goto**OpenCv/build/python/2.7** folder.

6. Copy **cv2.pyd** to **C:/Python27/lib/site-packeges**.

7. Open Python IDLE and type following codes in Python terminal.

8. >>>importcv2

9. >>>printcv2.__version__

If the results are printed out without any errors, congratulations !!!. You have installed OpenCV-Python successfully.

3.2.3. Building OpenCV from source:

1. Download and install Visual Studio.

    1.1. Visual Studio 2012

    1.2. CMake

2. Download and install necessary Python packages to their default locations

    2.1. Python 2.7.x

    2.2. Numpy

In this case, we are using 32-bit binaries of Python packages. But if you want to use OpenCV for x64, 64-bit binaries of Python packages are to be installed. Problem is that, there is no official 64-bit binaries of Numpy. You have to build it on your own. For that, you have to use the same compiler used to build Python. When you start Python IDLE, it shows the compiler details. You can get more information here. So your system must have the same Visual Studio version and build Numpy from source.

Another method to have 64-bit Python packages is to use ready-made Python distributions from third-parties like Anaconda, Enthought etc. It will be bigger in size, but will have everything you need. Everything in a single shell. You can also download 32-bit versions also.

3. Make sure Python and Numpy are working fine.

4. Download OpenCV source. It can be from Sourceforge (for official release version) or from Github (for latest source).

5. Extract it to a folder, OpenCv and create a new folder build in it.

6. OpenCvMake-gui (Start > All Programs >CMake-gui)

3.3 PANDAS:

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

1. What problem does pandas solve?

Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R. Combined with the excellent IPython toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate. Pandas does not implement significant modeling functionality outside of linear and panel regression; for this, look to statsmodels and scikit-learn. More work is still needed to make Python a first class statistical modeling environment, but we are well on our way toward that goal.

2. Installing Pandas and Anaconda

Installing pandas and the rest of the NumPy and SciPy stack can be a little difficult for inexperienced users. The simplest way to install not only pandas, but Python and the most popular packages that make up the SciPy stack (IPython, NumPy, Matplotlib, …) is with Anaconda, a cross-platform (Linux, Mac OS X, Windows) Python distribution for data analytics and scientific computing. After running the installer, the user will have access to pandas and the rest of the SciPy stack without needing to install anything else, and without needing to wait for any software to be compiled. Installation instructions for Anaconda can be found here. A full list of the packages available as part of the Anaconda distribution can be found here. Another advantage to installing Anaconda is that you don’t need admin rights to install it. Anaconda can install in the user’s home directory, which makes it trivial to delete Anaconda if you decide (just delete that folder).

3. Library Highlights

A fast and efficient **Data Frame** object for data manipulation with integrated indexing. Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;

Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;

Flexible **reshaping** and pivoting of data sets;

Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;

Columns can be inserted and deleted from data structures for **size mutability**;

Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;

High performance **merging and joining** of data sets;

Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;

Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;

Highly **optimized for performance**, with critical code paths written in Cython or C.

Python with ***pandas*** is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

Time Series with Pandas

As someone who works with time series data on almost a daily basis, I have found the pandas Python package to be extremely useful for time series manipulation and analysis.

This basic introduction to time series data manipulation with pandas should allow you to get started in your time series analysis. Specific objectives are to show you how to:

create a date range

work with timestamp data

convert string data to a timestamp

index and slice your time series data in a data frame

resample your time series for different time period aggregates/summary statistics

compute a rolling statistic such as a rolling average

work with missing data

understand the basics of unix/epoch time understand common pitfalls of time series data analysis

Let’s get started. If you want to play with real data that you have, you may want to start by using pandas read_csv to read in your file to a data frame, however we’re going to start by playing with generated data.

First import the libraries we’ll be working with and then use them to create a date range.

```
import pandas as pd
from datetime import datetime
import numpy as np
```

```
date_rng = pd.date_range(start='1/1/2018', end='1/08/2018', freq='H')
```

This date range has timestamps with an hourly frequency. If we call date_rng we’ll see that it looks like the following:

```
DatetimeIndex(['2018-01-01 00:00:00', '2018-01-01 01:00:00',
               '2018-01-01 02:00:00', '2018-01-01 03:00:00',
               '2018-01-01 04:00:00', '2018-01-01 05:00:00',
               '2018-01-01 06:00:00', '2018-01-01 07:00:00',
               '2018-01-01 08:00:00', '2018-01-01 09:00:00',
               ...
               '2018-01-07 15:00:00', '2018-01-07 16:00:00',
               '2018-01-07 17:00:00', '2018-01-07 18:00:00',
               '2018-01-07 19:00:00', '2018-01-07 20:00:00',
               '2018-01-07 21:00:00', '2018-01-07 22:00:00',
               '2018-01-07 23:00:00', '2018-01-08 00:00:00'],
dtype='datetime64[ns]', length=169, freq='H')
```

We can check the type of the first element:

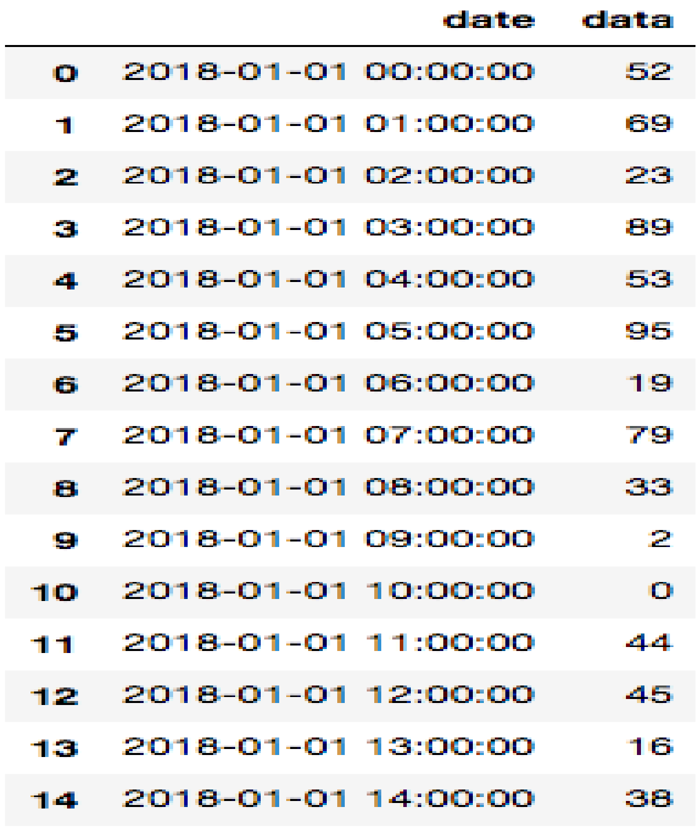
```
type(date_rng[0])#returnspandas._libs.tslib.Timestamp
```

Let’s create an example data frame with the timestamp data and look at the first 15 elements:

```
df  =  pd.DataFrame(date_rng, columns=['date'])df['data'] =
np.random.randint(0,100,size=(len(date_rng)))df.head(15)
```

Example data frame — df

If we want to do time series manipulation, we’ll need to have a date time index so that our data frame is indexed on the timestamp.



	date	data
0	2018-01-01 00:00:00	52
1	2018-01-01 01:00:00	69
2	2018-01-01 02:00:00	23
3	2018-01-01 03:00:00	89
4	2018-01-01 04:00:00	53
5	2018-01-01 05:00:00	95
6	2018-01-01 06:00:00	19
7	2018-01-01 07:00:00	79
8	2018-01-01 08:00:00	33
9	2018-01-01 09:00:00	2
10	2018-01-01 10:00:00	0
11	2018-01-01 11:00:00	44
12	2018-01-01 12:00:00	45
13	2018-01-01 13:00:00	16
14	2018-01-01 14:00:00	38

Convert the data frame index to a datetime index then show the first elements:

df['datetime'] = pd.to_datetime(df['date'])df = df.set_index('datetime')df.drop(['date'], axis=1, inplace=True)df.head.

4. SYSTEM DESIGN & UML DIAGRAMS

4.1 Architecture of simulation:

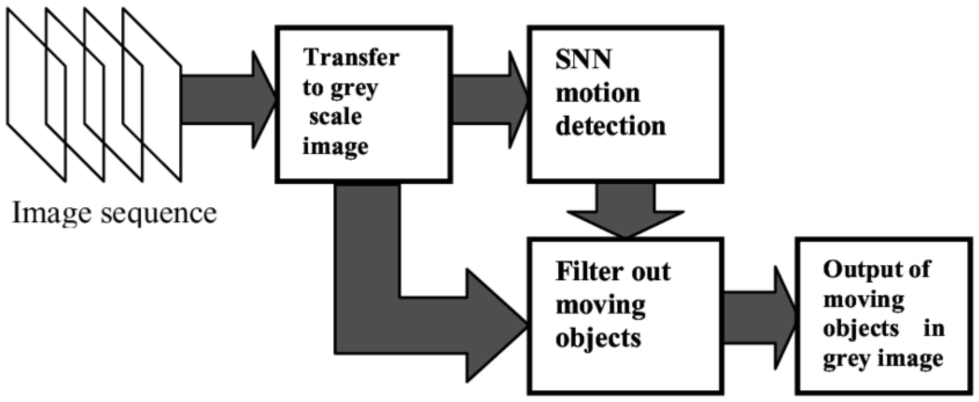
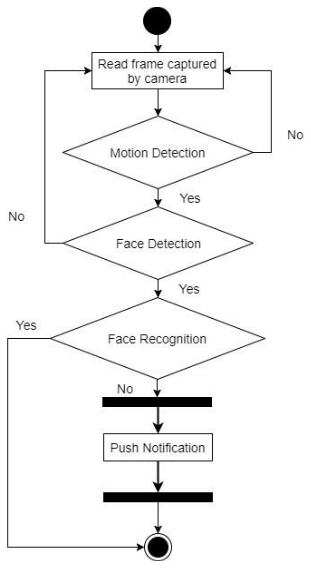


Image sequence
Transfer to grey scale image
SNN motion detection
Filter out moving objects
Output of moving objects in grey image

4.2 UML DIAGRAMS:

Activity diagram:

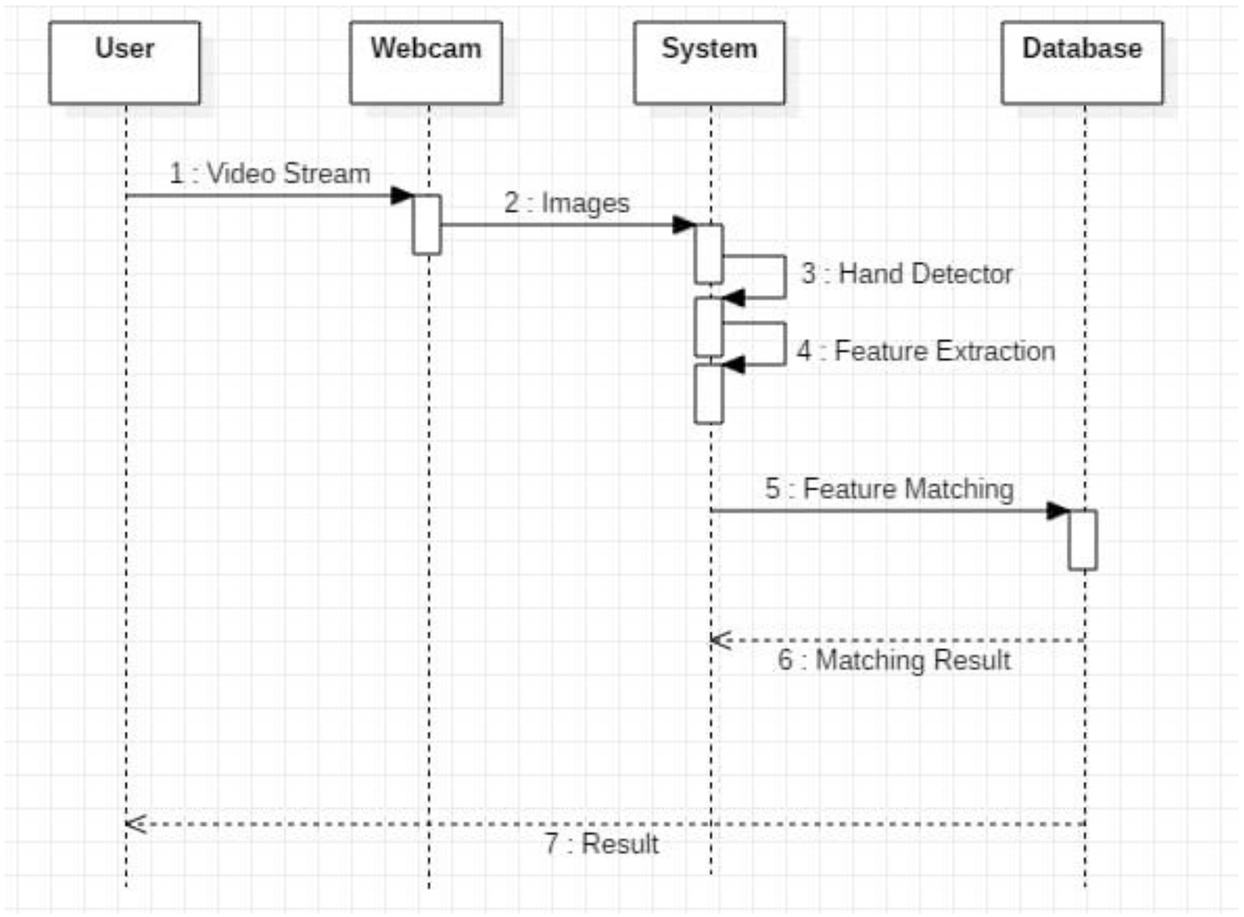
In Unified Modeling Language (UML), an activity diagram is a graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level.



Sequence diagram:

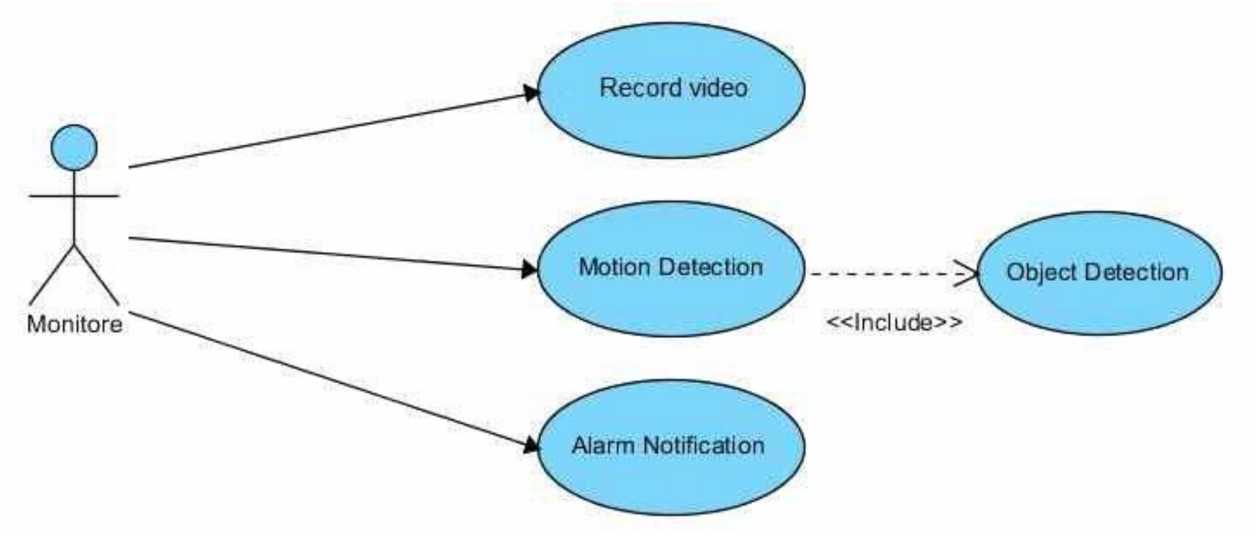
A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



Use Case diagram:

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The relationships between and among the actors and the usecases. The boundary, which defines the system of interest in relation to the world around it. The actors, usually individuals involved with the system defined according to their roles. The use cases, which are the specific roles played by the actors within and around the system. The relationships between and among the actors and the use cases.



5. INPUT/OUTPUT DESIGN

Getting Started with Images:

- Here, you will learn how to read an image, how to display it and how to save it back
- You will learn these functions : **cv2.imread()**, **cv2.imshow()** , **cv2.imwrite()**
- Optionally, you will learn how to display images with Matplotlib

5.1 INPUT DESIGN:

Read an image

Use the function **cv2.imread()** to read an image. The image should be in the working directory or a full path of image should be given.

Second argument is a flag which specifies the way image should be read.

- cv2.IMREAD_COLOR :Loads a color image. Any transparency of image will be neglected. It is the default flag.
- cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
- cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel

Instead of these three flags, you can simply pass integers 1, 0 or -1 respectively.

<u>**See the code below:**</u>

```
Import numpy as np
Import cv2
# Load an color image in grayscale
Img = cv2.imread('messi5.jpg',0)
```

Even if the image path is wrong, it won't throw any error, but `print img` will give you `None`

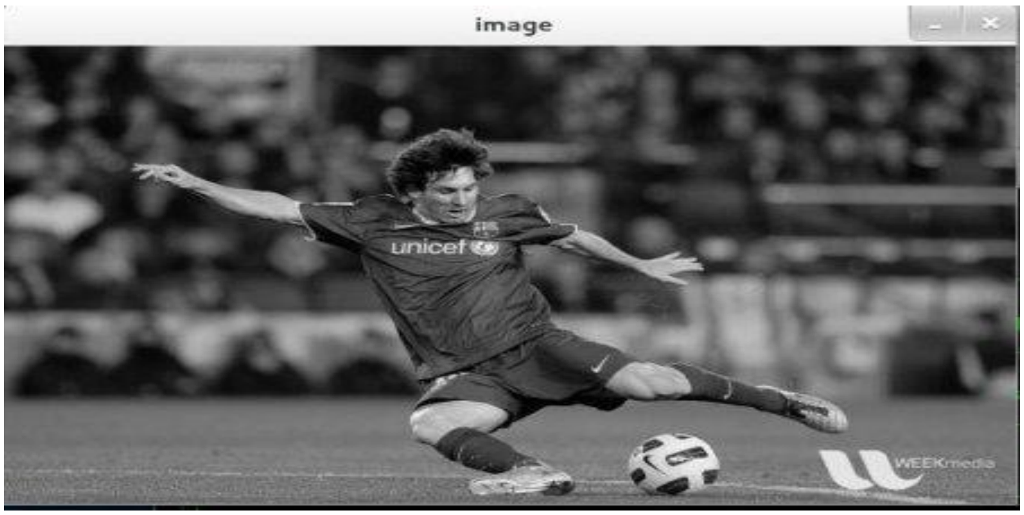
Display an image:

Use the function **cv2.imshow()** to display an image in a window. The window automatically fits to the image size.

First argument is a window name which is a string. second argument is our image. You can create as many windows as you wish, but with different window names.

```
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

A screenshot of the window will look like this (in Fedora-Gnome machine):



cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If **0** is passed, it waits indefinitely for a key stroke. It can also be set to detect specific key strokes like, if key *a* is pressed etc which we will discuss below.

cv2.destroyAllWindows() simply destroys all the windows we created. If you want to destroy any specific window, use the function **cv2.destroyWindow()** where you pass the exact window name as the argument.

There is a special case where you can already create a window and load image to it later. In that case, you can specify whether window is resizable or not. It is done with the function **cv2.namedWindow()**. By default, the flag is cv2.WINDOW_AUTOSIZE. But if you specify flag to be cv2.WINDOW_NORMAL, you can resize window. It will be helpful when image is too large in dimension and adding track bar to windows.

See the code below:

```
cv2.namedWindow('image',cv2.WINDOW_NORMAL)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Write an image

Use the function **cv2.imwrite()** to save an image.

First argument is the file name, second argument is the image you want to save.

```
cv2.imwrite('messigray.png',img)
```

This will save the image in PNG format in the working directory.

Sum it up

Below program loads an image in grayscale, displays it, save the image if you press ‘s’ and exit, or simply exit without saving if you press *ESC* key.

See below code:

```
import numpy as np
import cv2
img = cv2.imread('messi5.jpg',0)
cv2.imshow('image',img)
k = cv2.waitKey(0)
if k==27: # wait for ESC key to exit
cv2.destroyAllWindows()
elif k==ord('s'): # wait for 's' key to save and exit
cv2.imwrite('messigray.png',img)
cv2.destroyAllWindows()
```

5.2 OUTPUT DESIGN:

Getting Started with Videos

- Learn to read video, display video and save video.
- Learn to capture from Camera and display it.
- You will learn these functions : **cv2.VideoCapture()**, **cv2.VideoWriter()**

Capture Video from Camera

Often, we have to capture live stream with camera. OpenCV provides a very simple interface to this. Let’s capture a video from the camera (I am using the in-built webcam of my laptop), convert it into grayscale video and display it. Just a simple task to get started.

To capture a video, you need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don’t forget to release the capture.

<u>See below code:</u>

```
import numpy as np

import cv2

cap=cv2.VideoCapture(0)

while(True):

# Capture frame-by-frame
ret,frame=cap.read()
# Our operations on the frame come here
gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
# Display the resulting frame
cv2.imshow('frame',gray)
ifcv2.waitKey(1)&0xFF==ord('q'):
break
```

When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

cap.read() returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.

Sometimes, cap may not have initialized the capture. In that case, this code shows error. You can check whether it is initialized or not by the method **cap.isOpened()**. If it is True, OK. Otherwise open it using **cap.open()**.

You can also access some of the features of this video using **cap.get(propId)** method where propId is a number from 0 to 18. Each number denotes a property of the video (if it is applicable to that video) and full details can be seen here: Property Identifier. Some of these values can be modified using **cap.set(propId, value)**. Value is the new value you want.

For example, I can check the frame width and height by cap.get(3) and cap.get(4). It gives me 640x480 by default. But I want to modify it to 320x240. Just use ret = cap.set(3,320) and ret = cap.set(4,240).

If you are getting error, make sure camera is working fine using any other camera application (like Cheese in Linux).

Playing Video from file

It is same as capturing from Camera, just change camera index with video file name. Also while displaying the frame, use appropriate time for cv2.waitKey(). If it is too less, video will be very fast and if it is too high, video will be slow (Well, that is how you can display videos in slow motion). 25 milliseconds will be OK in normal cases.

See below code :

```
import numpy as np
import cv2
cap=cv2.VideoCapture('vtest.avi')
while(cap.isOpened()):
ret,frame=cap.read()
```

```
gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
cv2.imshow('frame',gray)
ifcv2.waitKey(1)&0xFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
```

Make sure proper versions of ffmpeg or gstreamer is installed. Sometimes, it is a headache to work with Video Capture mostly due to wrong installation of ffmpeg/gstreamer.

Saving a Video

So we capture a video, process it frame-by-frame and we want to save that video. For images, it is very simple, just use cv2.imWrite(). Here a little more work is required.

This time we create a **VideoWriter** object. We should specify the output file name (eg: output.avi). Then we should specify the **FourCC** code (details in next paragraph). Then number of frames per second (fps) and frame size should be passed. And last one is **isColor** flag. If it is True, encoder expect color frame, otherwise it works with grayscale frame.

FourCC is a 4-byte code used to specify the video codec. The list of available codes can be found in fourcc.org. It is platform dependent. Following codecs works fine for me.

- In Fedora: DIVX, XVID, MJPG, X264, WMV1, WMV2. (XVID is more preferable. MJPG results in high size video. X264 gives very small size video)
- In Windows: DIVX (More to be tested and added)
- In OSX : *(*I don't have access to OSX. Can some one fill this?)

as cv2.VideoWriter_fourcc('M','J','P','G') or cv2.VideoWriter_fourcc(*'MJPG) for MJPG.

Below code capture from a Camera, flip every frame in vertical direction and saves it.

Code:

```
import numpy as np
import cv2
cap=cv2.VideoCapture(0)
# Define the codec and create VideoWriter object
```

```
fourcc=cv2.VideoWriter_fourcc(*'XVID')
out=cv2.VideoWriter('output.avi',fourcc,20.0,(640,480))
while(cap.isOpened()):
ret,frame=cap.read()
if ret==True:
frame=cv2.flip(frame,0)
# write the flipped frame
out.write(frame)
cv2.imshow('frame',frame)
if cv2.waitKey(1)&0xFF==ord('q'):
break
else:
break
# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

Drawing Functions in OpenCV:

- Learn to draw different geometric shapes with OpenCV
- You will learn these functions : **cv2.line()**, **cv2.circle()** , **cv2.rectangle()**, **cv2.ellipse()**, **cv2.putText()** etc.

Code:

In all the above functions, you will see some common arguments as given below:

- img : The image where you want to draw the shapes
- color : Color of the shape. for BGR, pass it as a tuple, eg: (255,0,0) for blue. For grayscale, just pass the scalar value.
- thickness : Thickness of the line or circle etc. If **-1** is passed for closed figures like circles, it will fill the shape, default thickness = 1

- lineType : Type of line, whether 8-connected, anti-aliased line etc. By default, it is 8-connected. cv2.LINE_AA gives anti-aliased line which looks great for curves.

1. Drawing Line

To draw a line, you need to pass starting and ending coordinates of line. We will create a black image and draw a blue line on it from top-left to bottom-right corners.

See below code:

```
import numpy as np
import cv2
# Create a black image
img=np.zeros((512,512,3),np.uint8)
# Draw a diagonal blue line with thickness of 5 px
img=cv2.line(img,(0,0),(511,511),(255,0,0),5)
```

2. Drawing Rectangle

To draw a rectangle, you need top-left corner and bottom-right corner of rectangle. This time we will draw a green rectangle at the top-right corner of image.

```
img=cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)
```

3. Drawing Circle

To draw a circle, you need its center coordinates and radius. We will draw a circle inside the rectangle drawn above.

```
img=cv2.circle(img,(447,63),63,(0,0,255),-1)
```

4. Drawing Ellipse

To draw the ellipse, we need to pass several arguments. One argument is the center location (x,y). Next argument is axes lengths (major axis length, minor axis length). angle is the angle of rotation of ellipse in anti-clockwise direction. startAngle and endAngle denotes the starting and ending of ellipse arc measured in clockwise direction from major axis. i.e. giving values 0 and

360 gives the full ellipse. For more details, check the documentation of **cv2.ellipse()**. Below example draws a half ellipse at the center of the image.

Img = cv2.ellipse(img,(256,256),(100,50),0,0,180,255,-1)

5. Drawing Polygon

To draw a polygon, first you need coordinates of vertices. Make those points into an array of shape ROWSx1x2 where ROWS are number of vertices and it should be of type int32. Here we draw a small polygon of with four vertices in yellow color.

pts = np.array([[10,5],[20,30],[70,20],[50,10]],np.int32)

pts = pts.reshape((-1,1,2))

img = cv2.polylines(img,[pts],True,(0,255,255))

If third argument is False, you will get a polylines joining all the points, not a closed shape.

cv2.polylines() can be used to draw multiple lines. Just create a list of all the lines you want to draw and pass it to the function. All lines will be drawn individually. It is more better and faster way to draw a group of lines than calling cv2.line() for each line.

Adding Text to Images:

1. To put texts in images, you need specify following things.

- Text data that you want to write
- Position coordinates of where you want put it (i.e. bottom-left corner where data starts).
- Font type (Check **cv2.putText()** docs for supported fonts)
- Font Scale (specifies the size of font)
- regular things like color, thickness, lineType etc. For better look, lineType = cv2.LINE_AA is recommended.

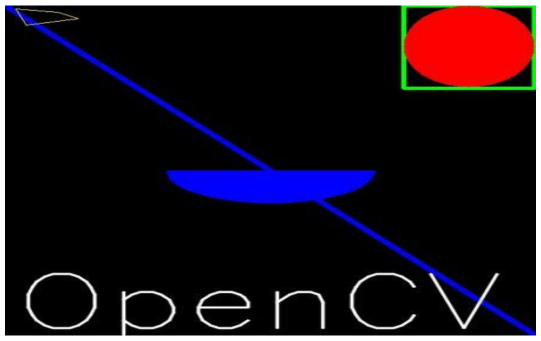
We will write **OpenCV** on our image in white color.

```
font=cv2.FONT_HERSHEY_SIMPLEX
```

```
cv2.putText(img,'OpenCV',(10,500),font,4,(255,255,255),2,cv2.LINE_AA)
```

2. Result:

So it is time to see the final result of our drawing. As you studied in previous articles, display image to see it.



6. IMPLEMENTATION

6.1 IMPORTING OF LIBRARIES:

```
# Import OpenCV for processing of images

# Import Time for window time logging

# Import DateTime for saving time stamp of motion detection events

# Import Pandas for dataframe and CSV creation

import cv2, time, pandas

from datetime import datetime

import datetime
```

6.2 CREATING FIRST FRAME AND VIDEO OBJECT:

```
# reference background frame against which to compare the presence of object/motion

first_frame = None

# Capture video feed from webcam (0), use video filename here for pre-recorded video

video = cv2.VideoCapture(0)

statusList = [-1, -1]  # stores the presence/absence of object in the present frame. -1 for absent and 1 for present

times = []  # stores timestamps of the entry and exit of object

df = pandas.DataFrame(columns=["Start", "End"])  # Pandas dataframe for exporting timestamps to CSV file

# the following loop continuously captures and displays the video feed until user prompts an exit by pressing Q

while True:

# the read function gives two outputs. The check is a boolean function that returns if the video is being read
```

```
check, frame = video.read()
#print(check,frame)
status = -1  # initialise status variable. This stores the presence/absence of object in the current frame
grayImg = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Grayscale conversion of the frame
# Gaussian blur to smoothen the image and remove noise.
# The tuple is the Kernel size and the 0 is the Std Deviation of the blur function
grayImg = cv2.GaussianBlur(grayImg, (21, 21), 1)
if first_frame is None:
first_frame = grayImg  # collect the reference frame as the first video feed frame
continue
# calculates the absolute difference between current frame and reference frame
deltaFrame = cv2.absdiff(first_frame, grayImg)
# convert image from grayscale to binary. This increases the demarcation between object and background by using a threshold function that
# converts everything above threshold to white
threshFrame = cv2.threshold(deltaFrame, 30, 255, cv2.THRESH_BINARY)[1]
cv2.imshow("thres", threshFrame)
# dilating the threshold removes the sharp edges at the object/background boundary and makes it smooth.
# More the iterations, smoother the image. Too smooth and you lose valuable data
threshFrame = cv2.dilate(threshFrame, None, iterations=0)
# Contour Function
# The contour function helps identify the closed object areas within the background.
```

# Afterthresholding, the frame has closed shapes of the objects against the background

# The contour function identifies and creates a list (cnts) of all these contours in the frame

# The RETR_EXTERNAL ensures that you only get the outermost contour details and all child contours inside it are ignored

# The CHAIN_APPROX_SIMPLE is the approximation method used for locating the contours. The simple one is used here for our trivial purpose

# Simple approximation removes all the redundant points in the description of the contour line

(cnts,_) = cv2.findContours(threshFrame.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

#(_, cnts, _) = cv2.findContours(threshFrame.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in cnts:

if cv2.contourArea(contour) < 10000:

# excluding too small contours. Set 10000 (100x100 pixels) for objects close to camera

continue

status = 1

# obtain the corresponding bounding rectangle of our detected contour

(x, y, w, h) = cv2.boundingRect(contour)

# superimpose a rectangle on the identified contour in our original colour image

# (x,y) is the top left corner, (x+w, y+h) is the bottom right corner

# (0,255,0) is colour green and 3 is the thickness of the rectangle edges

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

# do the above for all contours greater than our set size

# add the present status to our list

statusList.append(status)

```
# Detecting the entry and exit of objects
# Every entry/exit is identified by a sign change of the last two elements in our list, hence product is -1
if (statusList[-1] * statusList[-2]) == -1:
times.append(datetime.now())
# unitTesting
#cv2.imshow("Capturing", grayImg)
#cv2.imshow("DeltaFrame", deltaFrame)
#cv2.imshow("Threshold Frame", threshFrame)
# print(status)
# displays the continous feed with the green frame for any foreign object in frame
cv2.imshow("Colour Frame", frame)
# picks up the key press Q and exits when pressed
key = cv2.waitKey(1)
if key == ord('q'):
# if foreign object is in frame at the time of exiting, it stores the timestamp
if status == 1:
times.append(datetime.now())
break
# print(statusList)
# print(times)
# take every 2 timestamps in the list and store them as startTime and endTime in the Pandas dataframe
for i in range(0, len(times), 2):
df = df.append({"Start": times[i], "End": times[i + 1]}, ignore_index=True)
```

```
# Export to csv

df.to_csv("Times.csv")

# Closes all windows

cv2.destroyAllWindows()

# Releases video file/webcam

video.release()

result = motion_detection1()
```

6.3 PROGRAM EXECUTION FLOW CHART:

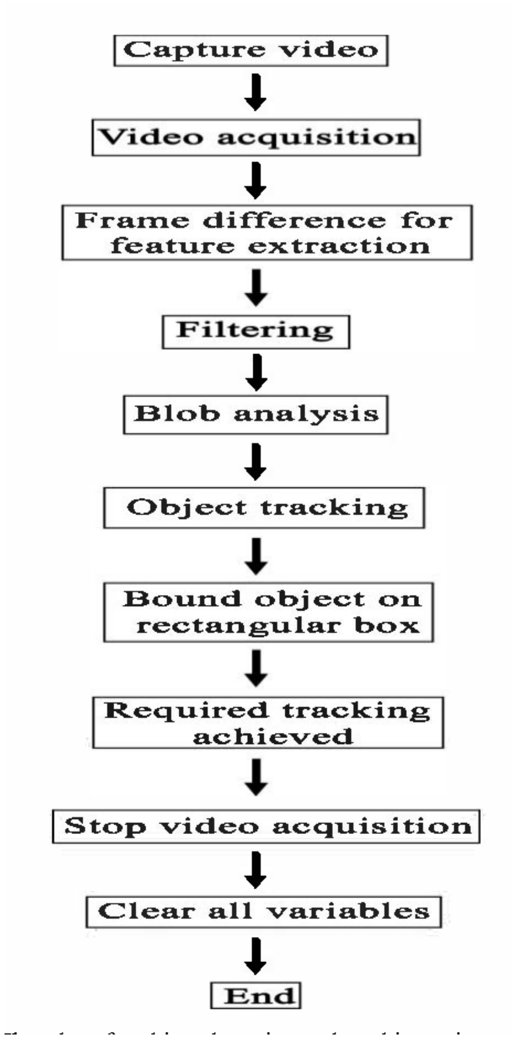


Fig: Flow chart of program execution

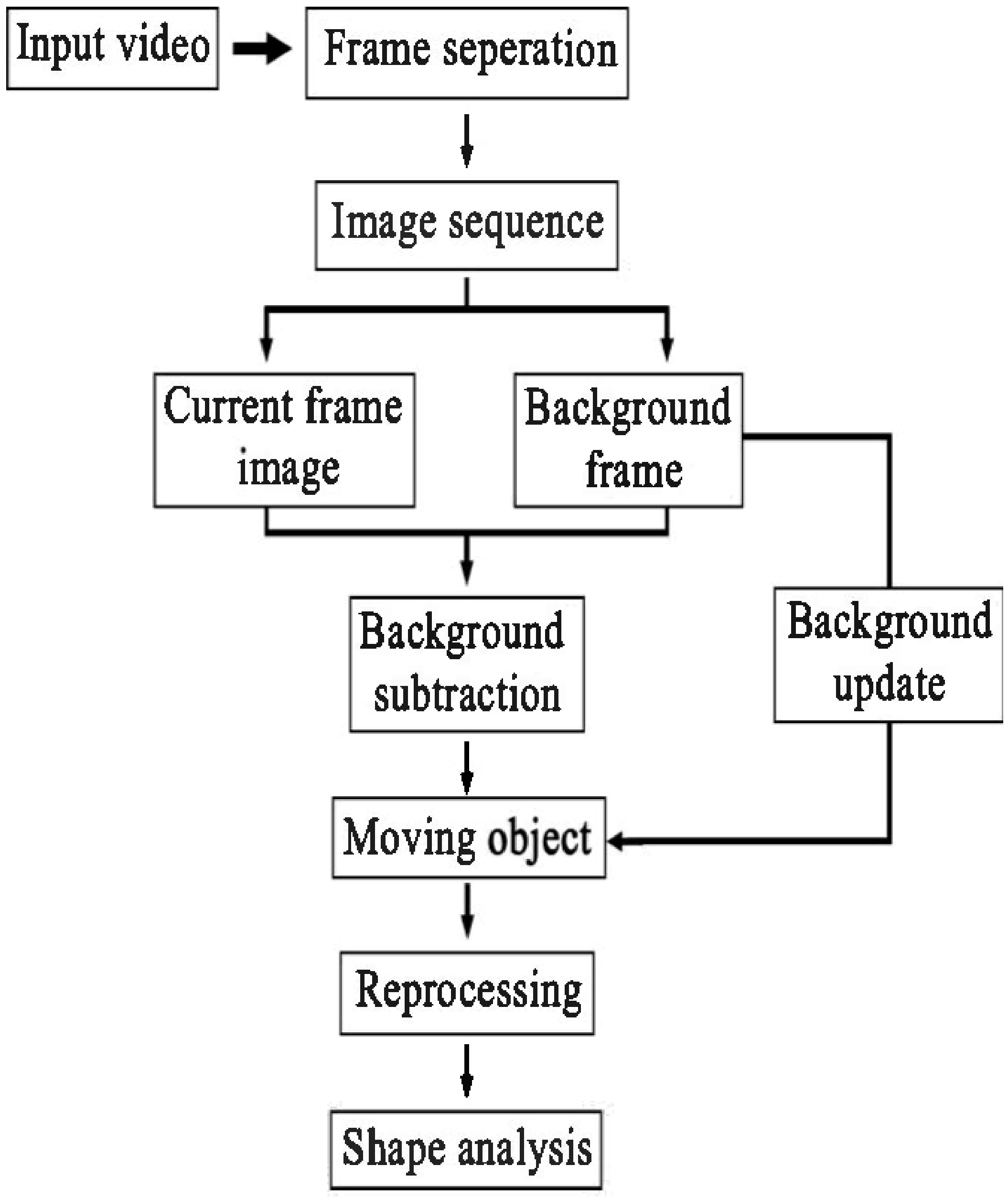


Fig: Flow chart of frames analysis

7. TESTING

7.1 ANALYSIS OF ALL WINDOWS:

After running the code there 4 new window will appear on screen. Let's analyse it one by one:

Gray Frame : In Gray frame the image is a bit blur and in grayscale we did so because, In gray pictures there is only one intensity value whereas in RGB(Red, Green and Blue) image thre are three intensity values. So it would be easy to calculate the intensity difference in grayscale.

Difference Frame : Difference frame shows the difference of intensities of first frame to the current frame.

Threshold Frame : If the intensity difference for a particular pixel is more than 30(in my case) then that pixel will be white and if the difference is less than 30 that pixel will be black.

Color Frame : In this frame you can see the color images in color frame along with green contour around the moving objects.

The Time_of_movements file will be stored in the folder where your code file is stored. This file will be in csv extension. In this file the start time of motion and the end time of motion will be recorded.

7.2 GRAY SCALE IMAGE:

Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest.

Grayscale images can be the result of measuring the intensity of light at each pixel according to a particular weighted combination of frequencies (or wavelengths), and in such cases they are monochromatic proper when only a single frequency (in practice, a narrow band of frequencies) is captured. The frequencies can in principle be from anywhere in the electromagnetic spectrum.

The intensity of a pixel is expressed within a given range between a minimum and a maximum, inclusive. This range is represented in an abstract way as a range from 0 (or 0%) (total absence, black) and 1 (or 100%) (total presence, white), with any fractional values in between.

Converting into gray scale

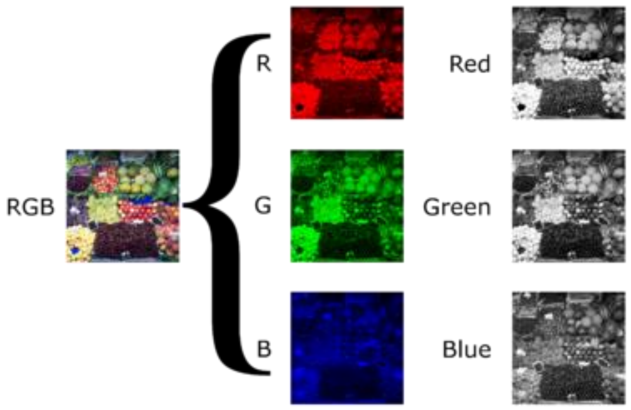
A color photo converted to grayscale.

Conversion of an arbitrary color image to grayscale is not unique in general; different weighting of the color channels effectively represent the effect of shooting black-and-white film with different-colored photographic filters on the cameras.

Grayscale as single channels of multichannel color images:

Color images are often built of several stacked color channels, each of them representing value levels of the given channel. For example, RGB images are composed of three independent channels for red, green and blue primary color components; CMYK images have four channels for cyan, magenta, yellow and black ink plates, etc.

Here is an example of color channel splitting of a full RGB color image. The column at left shows the isolated color channels in natural colors, while at right there are their grayscale equivalences:



Composition of RGB from 3 Grayscale images

The reverse is also possible: to build a full color image from their separate grayscale channels. By mangling channels, using offsets, rotating and other manipulations, artistic effects can be achieved instead of accurately reproducing the original image.

7.3 GAUSSIAN BLUR IMAGE:

Image Smoothing using OpenCV Gaussian Blur

1. Simple Thresholding

Here, the matter is straight forward. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is **cv2.threshold**. First argument is the source image, which **should be a grayscale image**. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function.

Different types are:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

2. Adaptive Thresholding:

In the previous section, we used a global value as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. In this, the algorithm calculate the threshold for a small regions of the image. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination. It has three ‘special’ input parameters and only one output argument.

Adaptive Method - It decides how thresholding value is calculated.

- cv2.ADAPTIVE_THRESH_MEAN_C : threshold value is the mean of neighborhood area.
- cv2.ADAPTIVE_THRESH_GAUSSIAN_C : threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.

Block Size - It decides the size of neighborhood area.

C - It is just a constant which is subtracted from the mean or weighted mean calculated.

Syntax of cv2 Gaussian blur function

OpenCV provides cv2.gaussianblur() function to apply Gaussian Smoothing on the input source image. Following is the syntax of GaussianBlur() function :

dst = cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType=BORDER_DEFAULT]]])

# read image

src = cv2.imread('/home/img/python.png', cv2.IMREAD_UNCHANGED)

# apply guassian blur on src image

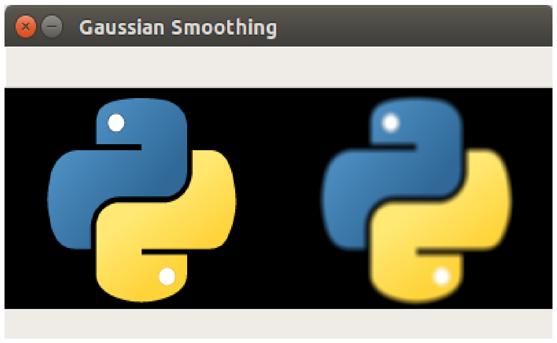
dst = cv2.GaussianBlur(src,(5,5),cv2.BORDER_DEFAULT)

# display input and output image

cv2.imshow("Gaussian Smoothing",numpy.hstack((src, dst)))

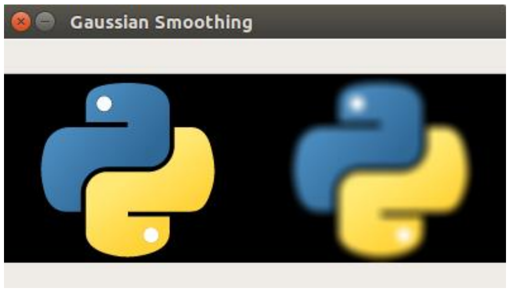
cv2.waitKey(0) # waits until a key is pressed

cv2.destroyAllWindows() # destroys the window showing image



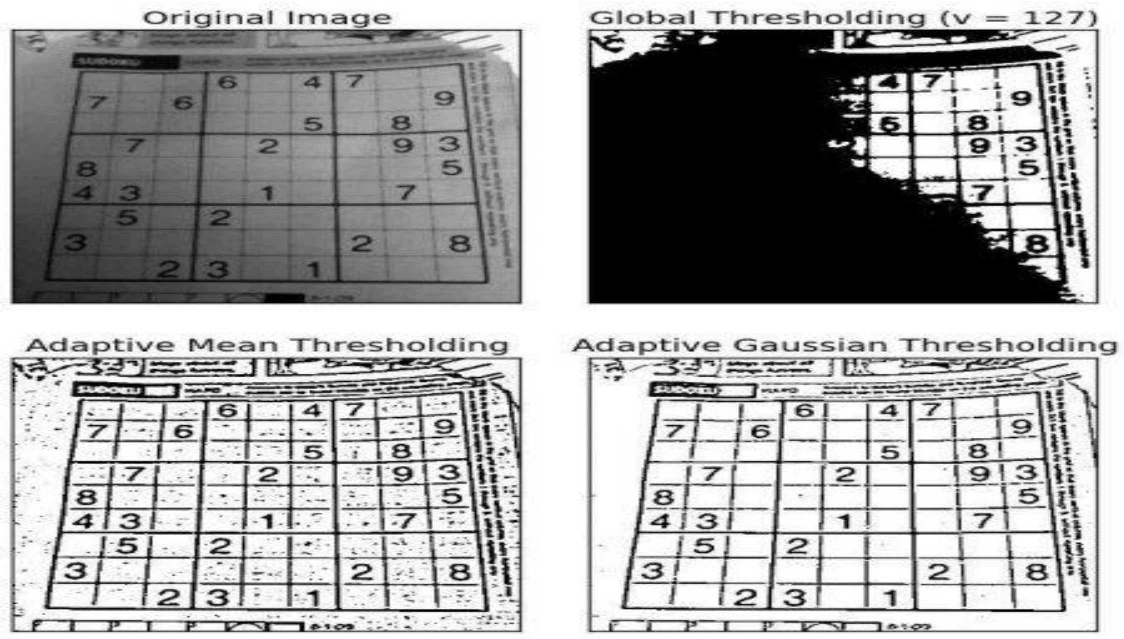
Now let us increase the Kernel size and observe the result.

```
dst = cv2.GaussianBlur(src,(10,10),cv2.BORDER_DEFAULT)
```



You may change values of other properties and observe the results.

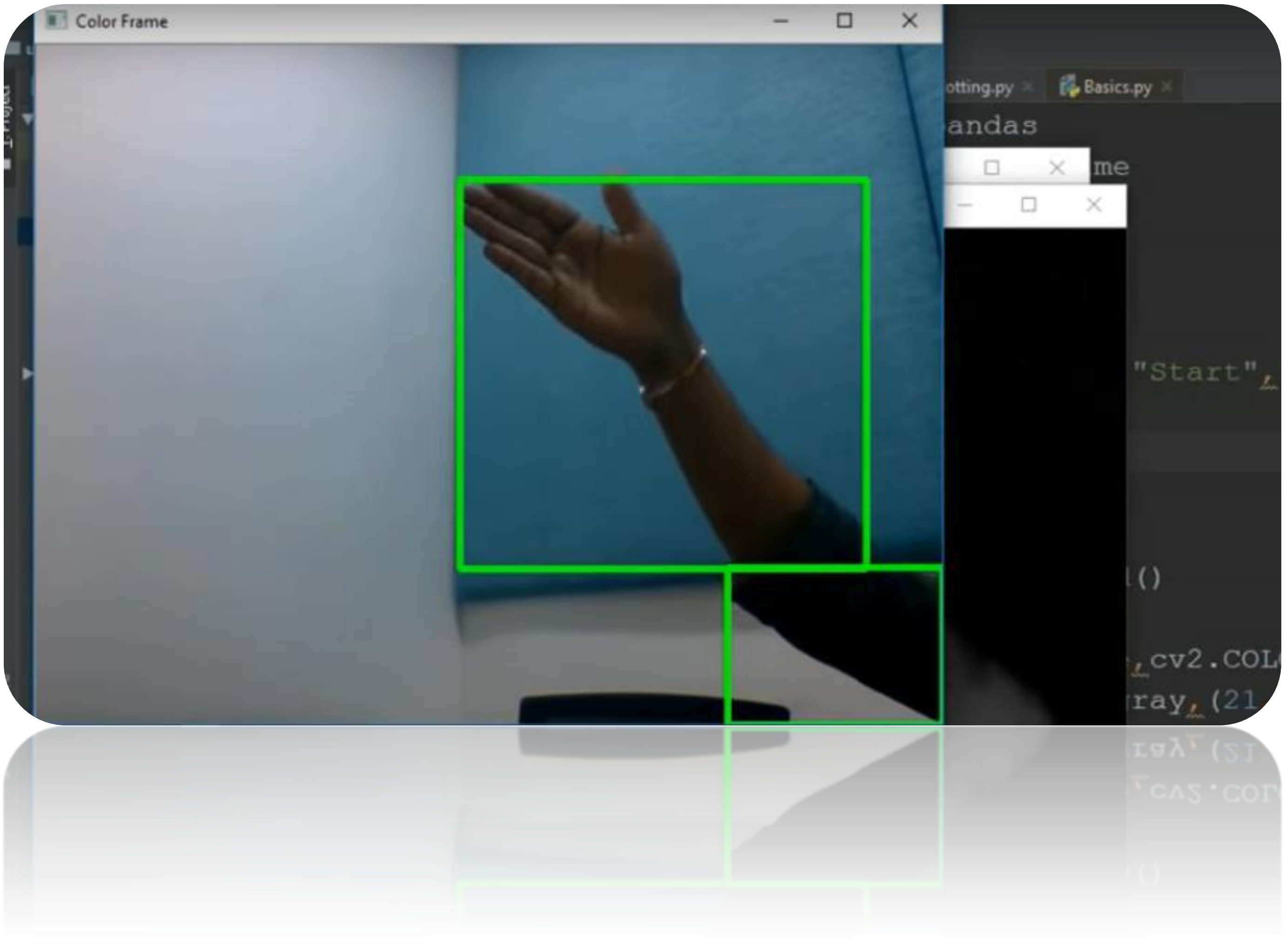
Result Fig: Different types of threshold frames


Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

8. OUTPUT SCREENS

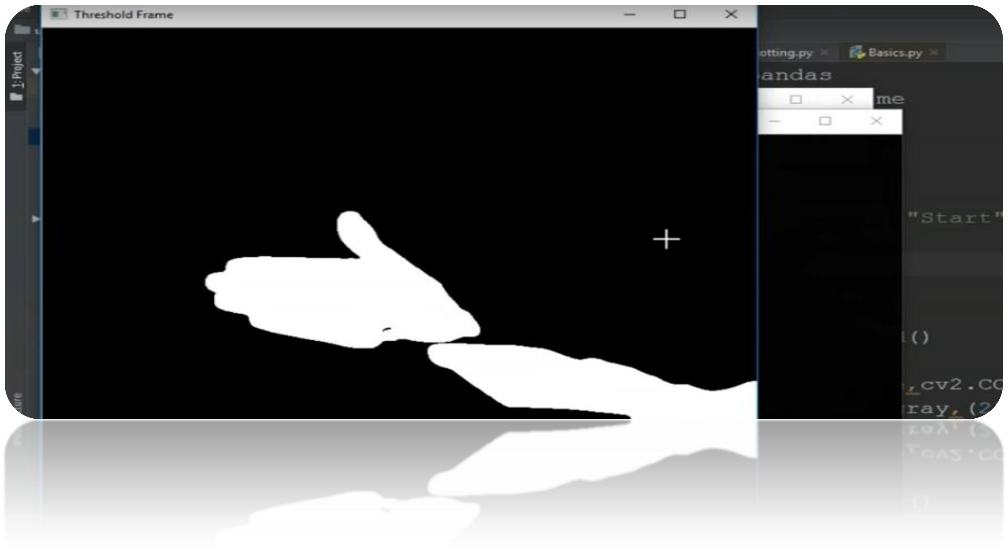
FOUR DIFFERENT FRAMES OF OUTPUT:

8.1 COLOR FRAME WITH RECTANGLE:

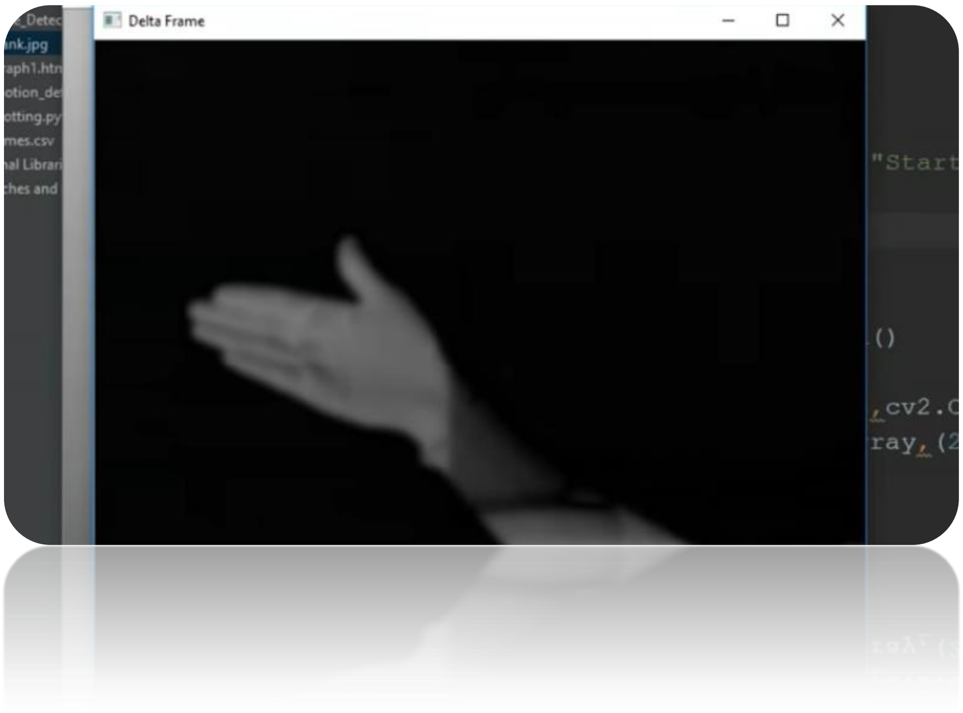


```
andas
me
"Start",
()
,cv2.COL
ray, (21
```

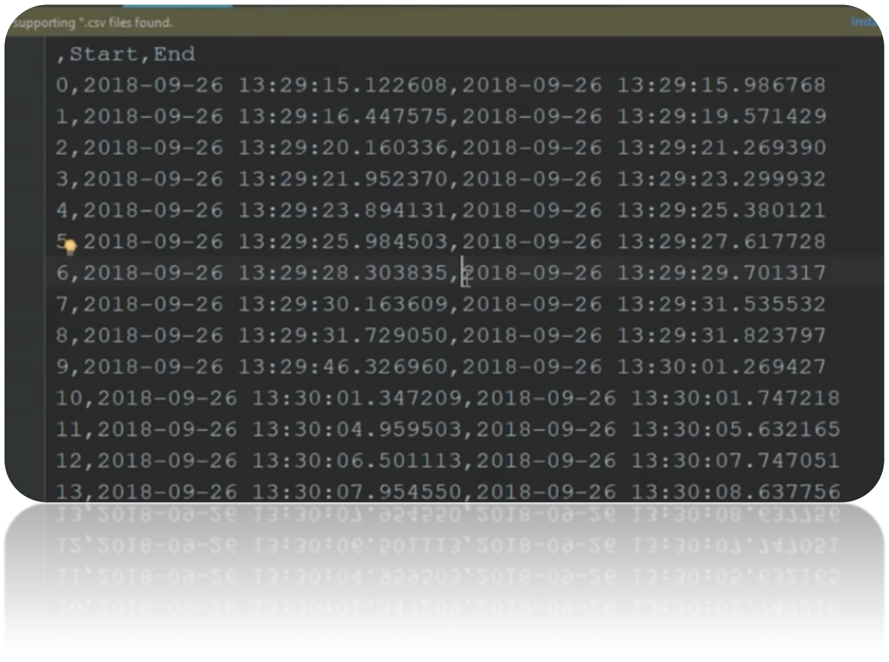
8.2 THRESHOLD FRAME WINDOW:



8.3 DELTA FRAME WINDOW



8.4 TIMES.CSV FILE OUTPUT:

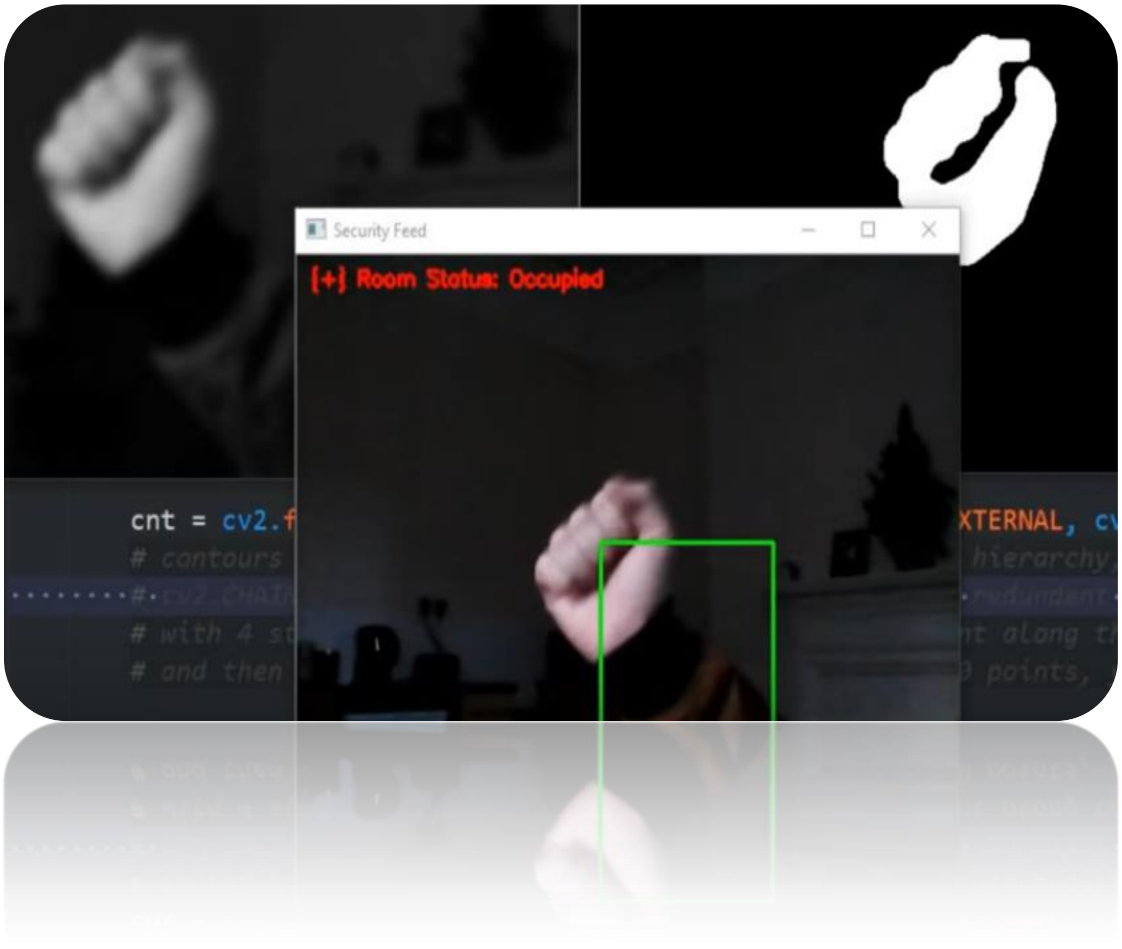


```
,Start,End
0,2018-09-26 13:29:15.122608,2018-09-26 13:29:15.986768
1,2018-09-26 13:29:16.447575,2018-09-26 13:29:19.571429
2,2018-09-26 13:29:20.160336,2018-09-26 13:29:21.269390
3,2018-09-26 13:29:21.952370,2018-09-26 13:29:23.299932
4,2018-09-26 13:29:23.894131,2018-09-26 13:29:25.380121
5,2018-09-26 13:29:25.984503,2018-09-26 13:29:27.617728
6,2018-09-26 13:29:28.303835,2018-09-26 13:29:29.701317
7,2018-09-26 13:29:30.163609,2018-09-26 13:29:31.535532
8,2018-09-26 13:29:31.729050,2018-09-26 13:29:31.823797
9,2018-09-26 13:29:46.326960,2018-09-26 13:30:01.269427
10,2018-09-26 13:30:01.347209,2018-09-26 13:30:01.747218
11,2018-09-26 13:30:04.959503,2018-09-26 13:30:05.632165
12,2018-09-26 13:30:06.501113,2018-09-26 13:30:07.747051
13,2018-09-26 13:30:07.954550,2018-09-26 13:30:08.637756
```

Output in times.csv file

	Start	End
0	31:04.2	31:04.3
1	31:04.4	31:04.4
2	31:04.8	31:04.9
3	31:04.9	31:05.1
4	31:05.1	31:05.2
5	31:06.4	31:14.2

8.5 ANALYSIS OF ALL WINDOWS:



```
cnt = cv2.f                                        XTERNAL, cv
# contours                                          hierarchy,
#.cv2.CHAI                                          redundent
# with 4 st                                         nt along th
# and then                                          9 points,
```

9. MOTION DETECTION BENEFITS & ADVANTAGES

9.1 BENEFITS:

Motion detectors detect any sort of motion. When connected to a transformer the motion gets translated into an electric signal. This device is connected to a burglar alarm that immediately alerts a homeowner of any motion that gets detected.

There are three types of sensors: passive infrared sensors, ultrasonic active sensors and microwave active sensors. Most motion detectors use a combination of different technologies. These dual-technology detectors benefit each other to reduce complications of false alarms.

Motion Detectors have pet-immune functions, allowing detectors to ignore pets that weigh up to 40 pounds or 80 pounds. Technology also ensures to maximize accuracy and reduce energy usage. With a wide assortment of uses to suit different requirements, motion detectors are flexible, useful devices to have around the house.

The most important benefit is that motion detectors offer feelings of protection and security for the average homeowner. They can be used in both residential and commercial locations. Most people find them suitable and useful in everyday situations and use them as driveway alarms, yard alarms, burglar alarms and door announcers.

Residential motion detectors have a variety of uses. Besides improving the exterior lighting to a home, they also help eliminate the fear of coming home to a dark house and tripping on the steps. They can also control lights and create well-lit paths, lighting up your driveway, sidewalk and porch.

Motion detectors make a safe path to welcome you back home, deterring would-be-intruders. When lights are triggered by motion this gives the impression that someone is home and able to see the burglar. Infrared motion detectors placed in crucial areas of the house can detect any burglars and alert the homeowner or police.

Whether you reside in a house or apartment, having an alarm system in your dwelling is a must nowadays. There are many home protection products that are effective, affordable and clever. They offer an alternative solution to the often pricey wired and wireless home alarm systems.

Motion detectors have the advantage of being portable, transportable, and easy to install with no monthly monitoring fees. Studies on crime prevention indicate that houses set up with motion detectors do help enhance security. They are a great way to get all the benefits of having a watchdog without the hassle of owning a dog.

9.2 APPLICATIONS OF MOTION DETECTION:

Some of the key applications of motion detectors include:

- Intruder alarms
- Automatic ticket gates
- Entryway lighting
- Security lighting
- Hand dryers
- Automatic doors
- Ultrasonic sensors are used for triggering the security camera at home and for wildlife photography.
- Active infrared sensors used to indicate the presence of products on conveyor belts.

10. FUTURE SCOPE

During the last years, there has been a rapid and successful expansion on computer vision research. Parts of this success have come from adopting and adapting machine learning methods, while others from the development of new representations and models for specific computer vision problems or from the development of efficient solutions. In many applications, we may be interested in detecting objects that are usually considered as background.

The detection of background objects, such as rivers, walls, mountains, has not been addressed by most of the here mentioned approaches. In general, this kind of problem has been addressed by first segmenting the image and later labelling each segment of the image. An important problem is to incrementally learn, to detect new classes, or to incrementally learn to distinguish among subclasses after the "main" class has been learned. If this can be done in an unsupervised way, we will be able to build new classifiers based on existing ones, without much additional effort, greatly reducing the effort required to learn new object classes. Note that humans are continuously inventing new objects, fashion changes, etc., and therefore detection systems will need to be continuously updated, adding new classes, or updating existing ones.

Motion detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

11. BIBLIOGRAPHY

BOOKS:

Python: The Complete Reference - Martin C. Brown

Learning OpenCV- Adrian Kaehler and Gary Bradski

HTML and CSS: Design and Build Websites - Jon Duckett

WEBSITES:

https://www.docs.pyhton.org/

https://stackoverflow.com/

https://www.w3schools.com/

https://www.geeksforgeeks.org/python-tutorial/