A Project Report

on

Motion Detection Using Background Subtraction Method for Home Surveillance

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
In Computer Science and Engineering

Under The Supervision
of Ms. Swati Sharma
Assistant Professor

Submitted By :
Sahil Gupta
(18SCSE1010205)
Vineeta Chaudhary
(18SCSE1010405)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA OCTOBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **Motion Detection Using Background Subtraction Method For Home Surveillance** in partial fulfillment of the requirements for the award of the B.Tech. CSE submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of August, 2021 to December 2021, under the supervision of Swati Sharma, Assistant Professor, School of Computing Science and Engineering, Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

- Sahil Gupta (18SCSE1010205)
- Vineeta Chaudhary (18SCSE1010405)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Swati Sharma
Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of Sahil Gupta (18SCSE1010205) & Vineeta Chaudhary (18SCSE1010405) has been held on _____ and his/her work is recommended for the award of B.Tech. C.S.E.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021
Place: Greater Noida

Acknowledgement

We would like to express our special thanks of gratitude to our dean Dr Munish Sabharwal who gave us the golden opportunity to do this wonderful project on the topic **"Motion Detection Using Background Subtraction Method For Home Surveillance"** which also helped us in doing a lot of research and we came to know about so many new things.
We are really thankful to him.
Secondly, I would like to thank our guide Ms. Swati Sharma who guided us throughout the project and helped us in finishing this project within a limited time.
It helped us increase our knowledge and skills.

Date:
17-12-2021

Sahil Gupta (18SCSE1010205)
Vineeta Chaudhary (18SCSE1010405)

Table of Contents

Abstract

Motion detection is the procrress of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. Artificial intelligence for video surveillance will utilize this program that analyse the images from the video surveillance camera in order to recognize any change in motion. Motion detection is one of the key techniques for automatic video analysis to extract crucial information from scenes in video surveillance systems. We will employ our system in home surveillance, which will alarm us of any theft or undesired motion in our house. We will use background subtraction algorithm. Background subtraction is a mainstream algorithm for moving object detection in video surveillance systems. We will focus on identifying and tracking any moving object in the video feed from our webcam using open-cv and playing an audio message using pyttsx3 library. It will work in real time which will help us to detect motion and raise alerting alarm. The result of this study is expected to be beneficial and able to assist users on effective motion detection and Tracking.

Introduction

1.1 Introduction

The rapid development in the field of digital image processing, motion detection and tracking are attractive research topics. In recent years, real-time video applications were inapplicable due to the expense computational time. Where an intelligent method to analyze the motion in a video stream line using the methods of background subtraction, frame differencing, and optical flow, methods are proposed. This system is designed to detect and track any moving event in a frame automatically. Organizations, commercial places and residential areas need to secure their facilities; this can be achieved by using security monitoring system with latest technology. An intelligent video sensor (Motion detector) was developed to support the monitoring security systems to detect unexpected movement without human intervention. The conventional systems are mostly human based, and it has its drawbacks. In order to eliminate this issues a technically enhanced security system need to be incorporated. Therefore this gave raise to the need of the security system, where new techniques were used in these security systems which are based on event movement (Motion). Detection of the movement, location, speed and any desired information of the event from the captured frames can be taken from the camera and can be transferred to the analysis part of the system. Motion detection is one of these intelligent systems which detect and track moving events. Where cameras capture the images of the securing area workspace; these images are processed to detect the event .Many algorithms and techniques have been used to perform this process and improve its outcome. Motion detection system is one among the latest technologies used for security purpose. This is broadly used in many computer vision tasks like pose estimation, human tracking and face recognition,

these are all the basic part of computer vision tasks. By using this technology, it is possible to monitor and capture every motion accurately/precisely in the area of interest. Motion detection is a process of confirming a change in position of an object relative to its surroundings or the change in the surroundings relative to an object. It is applied to various domestic and commercial applications starting from simple motion detectors to high speed video surveillance systems. The main task of a motion detection system is to detect an motion present in an “area of environment being monitored”.

Foreground detection based on video streams is the first step in computer vision applications, including real-time tracking and event analysis. Many researchers in the field of image and video semantics analysis pay attention to intelligent video surveillance in residential areas, junctions, shopping malls, subways, and airports which are closely associated with foreground detection.

Background modeling is an efficient way to obtain foreground objects. Though background modeling methods for foreground detection have been studied for several decades, each method has its own strength and weakness in detecting objects of interest from video streams.

Some of them are very significant for BS, and not usual in the other benchmarks. I have proposed a comparison of BGS methods namely (Adaptive BG Learning, ZivkovicGMM, Fuzzy Integral), with various methodologies.

1.2 Features:

- Can eliminate noise in the sequence of frames 9ffectively using suitable BGS methods.

- Can efficiently detect foreground provided alpha and threshold is fixed.

- Motions in different challenges can be detected by subtracting issues like dynamic background etc.

1.3 BG Modeling Steps:

- **Background initialization**: The first aim to build a background model is to fix number of frames. This model can be designed by various ways (Guassian, fuzzy etc.).

- **Foreground detection**: In the next frames, a comparison is processed between the current frame and the background model. This subtraction leads to the computation of the foreground of the scene.

- **Background maintenance:** During this detection process, images are also analyzed in order to update the background model learned at the initialization step, with respect to a learning rate. An object not moving during long time should be integrated in the background for example.

1.4 BG Subtraction Methods step by step:

1. Adaptive BG Learning: In a simple way, this can be done by setting manually a static image that represents the background, and having no moving object

- For each video frame, compute the absolute difference between the current frame and the static image.

- If absolute difference exceeds threshold, frame is regarded as background, otherwise foreground.

2. Gaussian mixture model (GMM): In order to model a background which is dynamic texture(such as waves on the water or trees shaken by the wind), each pixel with a mixture of K Gaussians distributions is modeled.

- For each video frames, find the probability of input pixel value x from current frame at time t being a background pixel is represented by the following mixture of Gaussians

$$P(x_i) = \sum_{i=1}^{k} \omega_{i,t}.\eta\left(x_t, U_{i,t}\sum_{i,t}\right)$$

- A new pixel is checked against the exiting K Gaussian distributions, until a match is found.

- If none of K distributions match the current pixel value, the least probable distribution is replaced

- with a distribution with the current value as its mean value.

- If pixel values cannot match the background model distributions, they will be labeled “in motion”, otherwise background pixel.

3. Fuzzy Integral:: The background initialization is made by using the average of the N first video frames where objects are present. An update rule of the

background image is necessary to adapt well the system over time to some environmental changes. For this, a selective maintenance scheme is adopted as follows:

$$B_{t+1}(x,y) = (1-\alpha)B_t(x,y) + \alpha T_{t+1}(x,y)$$
$$,if(x,y)isBackground$$
$$B_{t+1}(x,y) = (1-\beta)B_t(x,y) + \beta T_{t+1}(x,y)$$
$$,if(x,y)isBackground$$

The fuzzy integrals aggregates nonlinearly the outcomes of all criteria.

The pixel at position(x, y)is considered as foreground if its Choquet integral value is less than a certain constant threshold which means that pixels at the same position in the background and the current images are not similar.
This a constant value depending on each video data set.

Literature Survey

One of the common approaches for a motion detector is to compare the current frame of a streaming video with the previous frame. It is very useful in video compression especially in estimation of changes, writing only the changes and not the whole frame. Firstly, Difference and Threshold filters are used to distinguish the difference regions between an original gray scaled frame and the previous video gray scaled frame. An image with white pixel on the difference regions is obtained on the specified threshold value. A motion event can be signaled if the value is greater than a predefined alarm level [1]. Then, erosion filter is used to remove random noisy pixels since mostly of cameras produce a noisy image. The Erosion filter is a morphological filter that changes the shape of objects in an image by eroding (reducing) the boundaries of bright objects and enlarging the boundaries of dark ones. It is often used to reduce or eliminate small bright objects [2]. This filter assigns minimum value of surrounding pixels to each pixel of the result image. Surrounding pixels, which should be processed, are specified by structuring element: 1 to process the neighbor or -1 to skip it. It is very useful for binary image processing, where it removes pixels, which are not surrounded by specified amount of neighbors. It gives ability to remove noisy pixels or shrink objects [3]. At this stage, an actual motion is obtained since mostly only the interest regions are being detected. From the below picture, the disadvantages of the approach had been discovered. If the object is moving smoothly, small changes from frame to frame was received. Hence, it is a problem and difficulty to get the whole moving object. Things become worse when the object is moving so slowly and the algorithms will not give any result at all.

The importance and popularity of human motion analysis has led to several previous surveys.

Neeti A. Ogale[4] discussed a agent sample of techniques for finding people using visual input. These techniques are classified with respect to the need for pre-processing ,features

used to describe human appearance,use of explicit body models.

Prithviraj Banerjee and Somnath Sengupta[5] proposed Automated Video Surveillance System .The system employs a novel combination of an Adaptive Background Modeling Algorithm ,based on the Gaussian Mixture Model and a Human Detection for Surveillance (HDS) System. The HDS system incorporates a Histogram of Oriented Gradients based human detector which is well known for its performance in detecting humans in still images.

Xiaofei Ji, Honghai Liu[6] provides a total survey of human motion detection with the variation on view-invariant expression, and detection of special facial expressions and proceedings. In order to help readers understand the incorporated development of visual analysis of human motion detection, this paper presents recent growth in human detection, view-invariant pose demonstration and estimation, and human performance understanding.Public available standard datasets are recommended. The last replace assesses the development so far, and outlines some observed issues and future guidelines, and solution to what is necessary to get the goals of total human motion examination.

Murat Ekinci, Eyup Gedikli[7] presented a real-time background modeling and maintenance based human motion detection and analysis in an indoor and an outdoor environments for visual surveillance system is described. The system operates on monocular gray scale video imagery from a static CCD camera. In order to detect foreground objects, background scene model is statistically learned using the redundancy of the pixel intensities in a preparation stage, even the background is not completely stationary. This redundancy information of the each pixel is separately stored in an history map shows how the pixel intensity values changes till now. Then the highest ratio of the redundancy on the pixel intensity values in the narration map in the training sequence is determined to have initial background model of the scene. A background maintenance model is also proposed for preventing some kind of falsies, such as, illumination changes, or physical changes. At the background modeling and maintenance,

the consistency and computational costs of the algorithm presented are comparatively discussed with several algorithms. Based on the background modeling, candidate foreground regions are detected using thresholding, noise cleaning and their boundaries extracted using morphological filters.

Hanzi Wang and David Suter[8] presented an effective and adaptive background modeling method for detecting foreground objects in both static and dynamic scenes. The proposed method computes sample consensus (SACON) of the background samples and estimates a statistical model per pixel.

Sumer Jabri, Zoran Duric, Harry Wechsler, Azriel Rosenfeld [9] proposed a new method of finding people in video images is presented. Detection is based on a novel background modeling and subtraction approach which uses both color and edge information. We introduce confidence maps, gray-scale images whose intensity is a function of our confidence that a pixel has changed to fuse intermediate results and to represent the results of background subtraction. The latter is used to define a person's body by guiding contour collection to segment the person from the background. The method is understanding to scene clutter, slow illumination changes, and camera noise, and runs in near real time on a standard platform.

Jing Li and Zhaofa Zeng, Jiguang Sun, and Fengshan Liu [10] presented Ultra wideband (UWB) radar technology which has emerged as one of the chosen choices for through-wall detection due to its high range resolution and good dispersion. The motion is a result of high bandwidth of Ultra wideband radar and helpful for better separation of multiple targets in complex environment. One important attribute of human is the periodic motion, such as lungful of air and limb movement. In this paper, the human life is detected and identified by the methods based on fast Fourier transform and S transform; they apply the UWB radar system in through-wall human detection. In particular, they can extract the center frequencies of life signals and locate the position of human targets from experimental data with high accuracy. Compared with other examine studies in through-

wall detection, this ultra wideband radar technology is well-built in the particular deliberation and identifying of the continued existence signal under strong insecurity.

Project Design

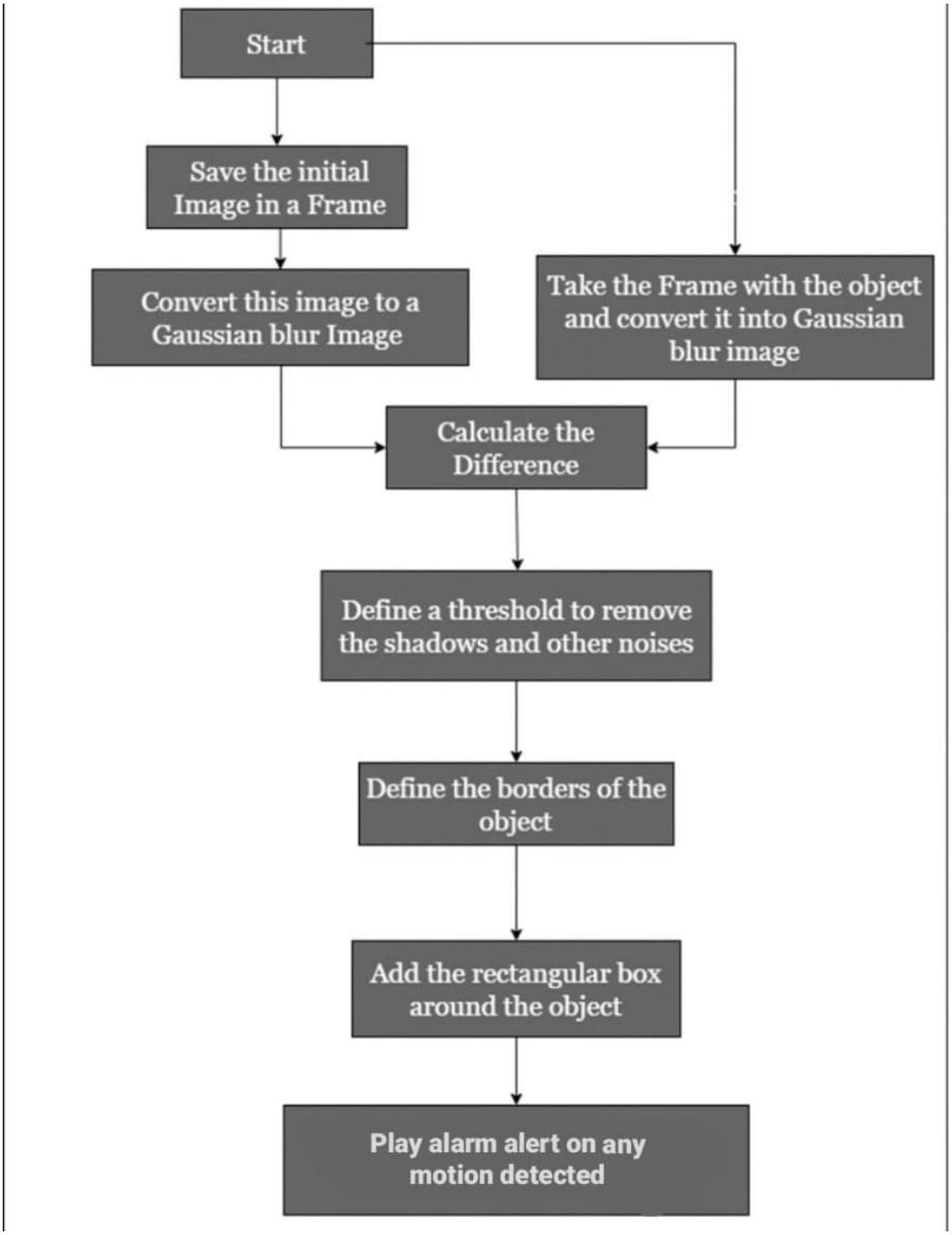


Fig 1: Flow diagram of proposed model.

Videos can be treated as stack of pictures called frames. Here we comparing different frames(pictures) to the first frame which should be static(No movements initially). We compare two images by comparing the intensity value of each pixels.

Analysis of all windows

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

1. 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 there are three intensity values. So it would be easy to calculate the intensity difference in grayscale.

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

3. 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.

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

Module Description

There are two separate modules in this program that you can use on their own, depending on the idea you are trying to implement:

1. Detecting Motion(using OpenCV)

2. Playing the Audio/Text to Speech (Using pyttsx3)

We will begin by installing the following Python libraries using pip.

```
pip install pyttsx3
pip install pywin32
pip install imutils
pip install numpy
pip install opencv-python
```

4.1 OpenCV:

4.1.1 Installation and Usage of OpenCV

1. If you have previous/other manually installed (= not installed via pip) version of OpenCV installed (e.g. cv2 module in the root of Python's site-packages), remove it before installation to avoid conflicts.
2. Make sure that your pip version is up-to-date (19.3 is the minimum supported version): pip install --upgrade pip. Check version with pip -V. For example Linux distributions ship usually with very old pip versions which cause a lot of unexpected problems especially with the manylinux format.
3. Select the correct package for your environment:

   There are four different packages (see options 1, 2, 3 and 4 below) and you should **SELECT ONLY ONE OF THEM**. Do not install multiple different packages in the same environment. There is no plugin architecture: all the packages use the same namespace (cv2). If you installed multiple different

packages in the same environment, uninstall them all with pip uninstall and reinstall only one package.

a. Packages for standard desktop environments (Windows, macOS, almost any GNU/Linux distribution)

- Option 1 - Main modules package: pip install opencv-python
- Option 2 - Full package (contains both main modules and contrib/extra modules): pip install opencv-contrib-python (check contrib/extra modules listing from OpenCV documentation)

b. Packages for server (headless) environments (such as Docker, cloud environments etc.), no GUI library dependencies

These packages are smaller than the two other packages above because they do not contain any GUI functionality (not compiled with Qt / other GUI components). This means that the packages avoid a heavy dependency chain to X11 libraries and you will have for example smaller Docker images as a result. You should always use these packages if you do not use cv2.imshow et al. or you are using some other package (such as PyQt) than OpenCV to create your GUI.

- Option 3 - Headless main modules package: pip install opencv-python-headless
- Option 4 - Headless full package (contains both main modules and contrib/extra modules): pip install opencv-contrib-python-headless (check contrib/extra modules listing from OpenCV documentation)

4. Import the package:

```
import cv2
```

All packages contain Haar cascade files. cv2.data.haarcascades can be used as a shortcut to the data folder. For example:

```
cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
```

4.1.2 CI build process

The project is structured like a normal Python package with a standard setup.py file. The build process for a single entry in the build matrices is as follows (see for example .github/workflows/build_wheels_linux.yml file):

1. In Linux and MacOS build: get OpenCV's optional C dependencies that we compile against
2. Checkout repository and submodules
    - OpenCV is included as submodule and the version is updated manually by maintainers when a new OpenCV release has been made
    - Contrib modules are also included as a submodule
3. Find OpenCV version from the sources
4. Build OpenCV
    - tests are disabled, otherwise build time increases too much
    - there are 4 build matrix entries for each build combination: with and without contrib modules, with and without GUI (headless)
    - Linux builds run in manylinux Docker containers (CentOS 5)
    - source distributions are separate entries in the build matrix
5. Rearrange OpenCV's build result, add our custom files and generate wheel
6. Linux and macOS wheels are transformed with auditwheel and delocate, correspondingly
7. Install the generated wheel
8. Test that Python can import the library and run some sanity checks
9. Use twine to upload the generated wheel to PyPI (only in release builds)

Steps 1--4 are handled by pip wheel.

The build can be customized with environment variables. In addition to any variables that OpenCV's build accepts, we recognize:

- CI_BUILD. Set to 1 to emulate the CI environment build behaviour. Used only in CI builds to force certain build flags on in setup.py. Do not use this unless you know what you are doing.
- ENABLE_CONTRIB and ENABLE_HEADLESS. Set to 1 to build the contrib and/or headless version
- ENABLE_JAVA, Set to 1 to enable the Java client build. This is disabled by default.
- CMAKE_ARGS. Additional arguments for OpenCV's CMake invocation. You can use this to make a custom build.

See the next section for more info about manual builds outside the CI environment.

4.1.3 Manual builds

If some dependency is not enabled in the pre-built wheels, you can also run the build locally to create a custom wheel.

1. Clone this repository: git clone --recursive https://github.com/opencv/opencv-python.git
2. cd opencv-python
    - you can use git to checkout some other version of OpenCV in the opencv and opencv_contrib submodules if needed
3. Add custom Cmake flags if needed, for example: export CMAKE_ARGS="-DSOME_FLAG=ON -DSOME_OTHER_FLAG=OFF" (in Windows you need to set environment variables differently depending on Command Line or PowerShell)
4. Select the package flavor which you wish to build with ENABLE_CONTRIB and ENABLE_HEADLESS: i.e. export ENABLE_CONTRIB=1 if you wish to build opencv-contrib-python
5. Run pip wheel . --verbose. NOTE: make sure you have the latest pip version, the pip wheel command replaces the old python setup.py bdist_wheel command which does not support pyproject.toml.
    - this might take anything from 5 minutes to over 2 hours depending on your hardware
6. You'll have the wheel file in the dist folder and you can do with that whatever you wish
    - Optional: on Linux use some of the manylinux images as a build hosts if maximum portability is needed and run auditwheel for the wheel after build
    - Optional: on macOS use delocate (same as auditwheel but for macOS) for better portability

4.1.4 Manual debug builds

In order to build opencv-python in an unoptimized debug build, you need to side-step the normal process a bit.

1. Install the packages scikit-build and numpy via pip.

2. Run the command python setup.py bdist_wheel --build-type=Debug.
3. Install the generated wheel file in the dist/ folder with pip install dist/wheelname.whl.

If you would like the build produce all compiler commands, then the following combination of flags and environment variables has been tested to work on Linux:

```
export CMAKE_ARGS='-DCMAKE_VERBOSE_MAKEFILE=ON'

export VERBOSE=1

python3 setup.py bdist_wheel --build-type=Debug
```

4.1.5 Source distributions

Since OpenCV version 4.3.0, also source distributions are provided in PyPI. This means that if your system is not compatible with any of the wheels in PyPI, pip will attempt to build OpenCV from sources. If you need a OpenCV version which is not available in PyPI as a source distribution, please follow the manual build guidance above instead of this one.

You can also force pip to build the wheels from the source distribution.

If you need contrib modules or headless version, just change the package name (step 4 in the previous section is not needed). However, any additional CMake flags can be provided via environment variables as described in step 3 of the manual build section. If none are provided, OpenCV's CMake scripts will attempt to find and enable any suitable dependencies. Headless distributions have hard coded CMake flags which disable all possible GUI dependencies.

4.2 pyttsx3 (Text-to-speech x-platform):

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.

4.2.1 Installation:

```
pip install pyttsx3
```

If you recieve errors such as No module named win32com.client, No module named win32, or No module named win32api, you will need to additionally install pypiwin32.

4.2.2 Usage :

```
import pyttsx3

engine = pyttsx3.init()

engine.say("I will speak this text")

engine.runAndWait()
```

4.2.3 Changing Voice , Rate and Volume :

```
import pyttsx3

engine = pyttsx3.init() # object creation



""" RATE"""

rate = engine.getProperty('rate')   # getting details of current speaking rate

print (rate)                        #printing current voice rate

engine.setProperty('rate', 125)     # setting up new voice rate
```

```
"""VOLUME"""

volume = engine.getProperty('volume')   #getting to know current volume level (min=0
and max=1)

print (volume)                          #printing current volume level

engine.setProperty('volume',1.0)    # setting up volume level  between 0 and 1



"""VOICE"""

voices = engine.getProperty('voices')       #getting details of current voice

#engine.setProperty('voice', voices[0].id)  #changing index, changes voices. o for
male

engine.setProperty('voice', voices[1].id)   #changing index, changes voices. 1 for
female



engine.say("Hello World!")

engine.say('My current speaking rate is ' + str(rate))

engine.runAndWait()

engine.stop()

"""Saving Voice to a file"""

# On linux make sure that 'espeak' and 'ffmpeg' are installed

engine.save_to_file('Hello World', 'test.mp3')

engine.runAndWait()
```

4.3 Imutils:

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

4.3.1 Installation:

Provided you already have NumPy, SciPy, Matplotlib, and OpenCV already installed, the imutils package is completely pip-installable:

```
$ pip install imutils
```

Finding function OpenCV functions by name:

OpenCV can be a big, hard to navigate library, especially if you are just getting started learning computer vision and image processing. The find_function method allows you to quickly search function names across modules (and optionally sub-modules) to find the function you are looking for.

4.3.2 Funcations:

Translation:

Translation is the shifting of an image in either the x or y direction. To translate an image in OpenCV you would need to supply the (x, y)-shift, denoted as (tx, ty) to construct the translation matrix M:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

And from there, you would need to apply the cv2.warpAffine function.

Instead of manually constructing the translation matrix M and calling cv2.warpAffine, you can simply make a call to the translate function of imutils.

Rotation**:**

Rotating an image in OpenCV is accomplished by making a call to cv2.getRotationMatrix2D and cv2.warpAffine. Further care has to be taken to supply the (x, y)-coordinate of the point the image is to be rotated about. These calculation calls can quickly add up and make your code bulky and less readable. The rotate function in imutils helps resolve this problem.

Resizing:

Resizing an image in OpenCV is accomplished by calling the cv2.resize function. However, special care needs to be taken to ensure that the aspect ratio is maintained. This resize function of imutils maintains the aspect ratio and provides the keyword arguments width and height so the image can be resized to the intended width/height while (1) maintaining aspect ratio and (2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.

Another optional keyword argument, inter, can be used to specify interpolation method as well.

Skeletonization:

Skeletonization is the process of constructing the "topological skeleton" of an object in an image, where the object is presumed to be white on a black background. OpenCV does not provide a function to explicitly construct the skeleton, but does provide the morphological and binary functions to do so.

For convenience, the skeletonize function of imutils can be used to construct the topological skeleton of the image.

The first argument, size is the size of the structuring element kernel. An optional argument, structuring, can be used to control the structuring element -- it defaults to cv2.MORPH_RECT , but can be any valid structuring element.

Displaying with Matplotlib:

In the Python bindings of OpenCV, images are represented as NumPy arrays in BGR order. This works fine when using the cv2.imshow function. However, if you intend on using Matplotlib, the plt.imshow function assumes the image is in RGB order. A simple call to cv2.cvtColor will resolve this problem, or you can use the opencv2matplotlib convenience function.

URL to Image:

This the url_to_image function accepts a single parameter: the url of the image we want to download and convert to a NumPy array in OpenCV format. This function performs the download in-memory. The url_to_image function has been detailed here on the PyImageSearch blog.

Checking OpenCV Versions

OpenCV 3 has finally been released! But with the major release becomes backward compatibility issues (such as with the cv2.findContours and cv2.normalize functions). If you want your OpenCV 3 code to be backwards compatible with OpenCV 2.4.X, you'll need to take special care to check which version of OpenCV is currently being used and then take appropriate action. The is_cv2() and is_cv3() are simple functions that can be used to automatically determine the OpenCV version of the current environment.

Automatic Canny Edge Detection:

The Canny edge detector requires two parameters when performing hysteresis. However, tuning these two parameters to obtain an optimal edge map is non-trivial, especially when working with a dataset of images. Instead, we can use the auto_canny function which uses the median of the grayscale pixel intensities to derive the upper and lower thresholds.

4-point Perspective Transform:

A common task in computer vision and image processing is to perform a 4-point perspective transform of a ROI in an image and obtain a top-down, "birds eye view" of the ROI.

Sorting Contours:

The contours returned from cv2.findContours are unsorted. By using the contours module the the sort_contours function we can sort a list of contours from left-to-right, right-to-left, top-to-bottom, and bottom-to-top, respectively.

(Recursively) Listing Paths to Images:

The paths sub-module of imutils includes a function to recursively find images based on a root directory.

4.4 Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

• NumPy arrays have a fixed size at creation, unlike Python lists (which can grow

dynamically). Changing the size of an ndarray will create a new array and delete the original.

• The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

• NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

• A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

4.4.1 Why is NumPy Fast?

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just "behind the scenes" in optimized, pre-compiled C code. Vectorized code has many advantages, among which are:

• vectorized code is more concise and easier to read

• fewer lines of code generally mean fewer bugs

• the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)

• vectorization results in more "Pythonic" code. Without vectorization, our code would be littered with inefficient and difficult to read for loops.

Broadcasting is the term used to describe the implicit element-by-element behavior of operations; generally speaking, in NumPy all operations, not just arithmetic operations,

but logical, bit-wise, functional, etc., behave in this implicit element-by-element fashion, i.e., they broadcast. Moreover, in the example above, a and b could be multidimensional arrays of the same shape, or a scalar and an array, or even two arrays of with different shapes, provided that the smaller array is “expandable” to the shape of the larger in such a way that the resulting broadcast is unambiguous. For detailed “rules” of broadcasting see basics broadcasting.

4.4.2 Installing NumPy:

To install NumPy, we strongly recommend using a scientific Python distribution. If you’re looking for the full instructions for installing NumPy on your operating system

If you already have Python, you can install NumPy with:

```
conda install numpy
```

or

```
pip install numpy
```

4.4.3 How to import NumPy:

To access NumPy and its functions import it in your Python code like this:

```
import numpy as np
```

We shorten the imported name to np for better readability of code using NumPy. This is a widely adopted convention that you should follow so that anyone working with your code can easily understand it.

4.5 What Is a Thread?

A thread is a separate flow of execution. This means that your program will have two things happening at once. But for most Python 3 implementations the different threads do not actually execute at the same time: they merely appear to.

It’s tempting to think of threading as having two (or more) different processors running on your program, each one doing an independent task at the same time. That’s almost

right. The threads may be running on different processors, but they will only be running one at a time.

Getting multiple tasks running simultaneously requires a non-standard implementation of Python, writing some of your code in a different language, or using multiprocessing which comes with some extra overhead.

Because of the way CPython implementation of Python works, threading may not speed up all tasks. This is due to interactions with the GIL that essentially limit one Python thread to run at a time.

Tasks that spend much of their time waiting for external events are generally good candidates for threading. Problems that require heavy CPU computation and spend little time waiting for external events might not run faster at all.

This is true for code written in Python and running on the standard CPython implementation. If your threads are written in C they have the ability to release the GIL and run concurrently. If you are running on a different Python implementation, check with the documentation too see how it handles threads.

If you are running a standard Python implementation, writing in only Python, and have a CPU-bound problem, you should check out the multiprocessing module instead.

threading.**active_count**()
Return the number of Thread objects currently alive. The returned count is equal to the length of the list returned by enumerate ().

The function activeCount is a deprecated alias for this function.
threading.**current_thread**()
Return the current Thread object, corresponding to the caller's thread of control. If the caller's thread of control was not created through the threading module, a dummy thread object with limited functionality is returned.

The function currentThread is a deprecated alias for this function.
threading.**excepthook**(*args, /*)
Handle uncaught exception raised by Thread.run().

The *args* argument has the following attributes:

- *exc_type*: Exception type.
- *exc_value*: Exception value, can be None.

- *exc_traceback*: Exception traceback, can be None.
- *thread*: Thread which raised the exception, can be None.

If *exc_type* is SystemExit, the exception is silently ignored. Otherwise, the exception is printed out on sys.stderr.

If this function raises an exception, sys.excepthook() is called to handle it.

threading.excepthook() can be overridden to control how uncaught exceptions raised by Thread.run() are handled.

Storing *exc_value* using a custom hook can create a reference cycle. It should be cleared explicitly to break the reference cycle when the exception is no longer needed.

Storing *thread* using a custom hook can resurrect it if it is set to an object which is being finalized. Avoid storing *thread* after the custom hook completes to avoid resurrecting objects.

threading.**__excepthook__**
Holds the original value of threading.excepthook(). It is saved so that the original value can be restored in case they happen to get replaced with broken or alternative objects.

threading.**get_ident**()
Return the 'thread identifier' of the current thread. This is a nonzero integer. Its value has no direct meaning; it is intended as a magic cookie to be used e.g. to index a dictionary of thread-specific data. Thread identifiers may be recycled when a thread exits and another thread is created.

threading.**get_native_id**()
Return the native integral Thread ID of the current thread assigned by the kernel. This is a non-negative integer. Its value may be used to uniquely identify this particular thread system-wide (until the thread terminates, after which the value may be recycled by the OS).

threading.**enumerate**()
Return a list of all Thread objects currently active. The list includes daemonic threads and dummy thread objects created by current_thread(). It excludes terminated threads and threads that have not yet been started. However, the main thread is always part of the result, even when terminated.
threading.**main_thread**()

Return the main Thread object. In normal conditions, the main thread is the thread from which the Python interpreter was started.

threading.**settrace**(*func*)
Set a trace function for all threads started from the threading module. The *func* will be passed to sys.settrace() for each thread, before its run() method is called.
threading.**gettrace**()
Get the trace function as set by settrace().

threading.**setprofile**(*func*)
Set a profile function for all threads started from the threading module. The *func* will be passed to sys.setprofile() for each thread, before its run() method is called.
threading.**getprofile**()
Get the profiler function as set by setprofile().

threading.**stack_size**([*size*])
Return the thread stack size used when creating new threads. The optional *size* argument specifies the stack size to be used for subsequently created threads, and must be 0 (use platform or configured default) or a positive integer value of at least 32,768 (32 KiB). If *size* is not specified, 0 is used. If changing the thread stack size is unsupported, a RuntimeError is raised. If the specified stack size is invalid, a ValueError is raised and the stack size is unmodified. 32 KiB is currently the minimum supported stack size value to guarantee sufficient stack space for the interpreter itself. Note that some platforms may have particular restrictions on values for the stack size, such as requiring a minimum stack size > 32 KiB or requiring allocation in multiples of the system memory page size - platform documentation should be referred to for more information (4 KiB pages are common; using multiples of 4096 for the stack size is the suggested approach in the absence of more specific information).

This module also defines the following constant:

threading.**TIMEOUT_MAX**
The maximum value allowed for the *timeout* parameter of blocking functions (Lock.acquire(), RLock.acquire(), Condition.wait(), etc.). Specifying a timeout greater than this value will raise an OverflowError.

This module defines a number of classes, which are detailed in the sections below.

The design of this module is loosely based on Java's threading model. However, where Java makes locks and condition variables basic behavior of every object, they are separate objects in Python. Python's Thread class supports a subset of the behavior of Java's Thread class; currently, there are no priorities, no thread groups, and threads cannot be destroyed, stopped, suspended, resumed, or interrupted. The static methods of Java's Thread class, when implemented, are mapped to module-level functions.

All of the methods described below are executed atomically.

Thread-Local Data:

Thread-local data is data whose values are thread specific. To manage thread-local data, just create an instance of local (or a subclass) and store attributes on it:

```
mydata = threading.local()
mydata.x = 1
```

The instance's values will be different for separate threads.

class threading.**local**
A class that represents thread-local data.

For more details and extensive examples, see the documentation string of the _threading_local module.

Thread Objects:

The Thread class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: by passing a callable object to the constructor, or by overriding the run() method in a subclass. No other methods (except for the constructor) should be overridden in a subclass. In other words, *only* override the __init__() and run() methods of this class.

Once a thread object is created, its activity must be started by calling the thread's start() method. This invokes the run() method in a separate thread of control.

Once the thread's activity is started, the thread is considered 'alive'. It stops being alive when its run() method terminates – either normally, or by raising an unhandled exception. The is_alive() method tests whether the thread is alive.

Other threads can call a thread's join() method. This blocks the calling thread until the thread whose join() method is called is terminated.

A thread has a name. The name can be passed to the constructor, and read or changed through the name attribute.

If the run() method raises an exception, threading.excepthook() is called to handle it. By default, threading.excepthook() ignores silently SystemExit.

A thread can be flagged as a "daemon thread". The significance of this flag is that the entire Python program exits when only daemon threads are left. The initial value is inherited from the creating thread. The flag can be set through the daemon property or the *daemon* constructor argument.

There is a "main thread" object; this corresponds to the initial thread of control in the Python program. It is not a daemon thread.

There is the possibility that "dummy thread objects" are created. These are thread objects corresponding to "alien threads", which are threads of control started outside the threading module, such as directly from C code. Dummy thread objects have limited functionality; they are always considered alive and daemonic, and cannot be join()ed. They are never deleted, since it is impossible to detect the termination of alien threads.

class threading.**Thread**(*group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None*)
This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-*N*" where *N* is a small decimal number, or "Thread-*N* (target)" where "target" is target.__name__ if the *target* argument is specified.

args is the argument tuple for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If not None, *daemon* explicitly sets whether the thread is daemonic. If None (the default), the daemonic property is inherited from the current thread.

If the subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Changed in version 3.10: Use the *target* name if *name* argument is omitted.

Changed in version 3.3: Added the *daemon* argument.

start()
Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.
run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the *target* argument, if any, with positional and keyword arguments taken from the *args* and *kwargs* arguments, respectively.
join(*timeout=None*)
Wait until the thread terminates. This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception – or until the optional timeout occurs.

When the *timeout* argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the *timeout* argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raise the same exception.

name

A string used for identification purposes only. It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

getName()

setName()

Deprecated getter/setter API for name; use it directly as a property instead.

ident

The 'thread identifier' of this thread or None if the thread has not been started. This is a nonzero integer. See the get_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

native_id

The Thread ID (TID) of this thread, as assigned by the OS (kernel). This is a non-negative integer, or None if the thread has not been started. See the get_native_id() function. This value may be used to uniquely identify this particular thread system-wide (until the thread terminates, after which the value may be recycled by the OS).

Availability: Requires get_native_id() function.

is_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. The module function enumerate() returns a list of all alive threads.

daemon

A boolean value indicating whether this thread is a daemon thread (True) or not (False). This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when no alive non-daemon threads are left.

isDaemon()

setDaemon()

Deprecated getter/setter API for daemon; use it directly as a property instead.

4.6 Approach:

The approach is very simple. When the program starts, we will capture a picture called *baseline_image.* This is the image without any object/intruder. Our program will keep comparing the new frame with this baseline_image. If nobody enters or exits the frame, there will be no difference. However, when somebody enters the frame, the contents of the image will be different and if this difference is beyond a certain threshold, our program will treat it as an intruder and play an audio.

For detecting motion, we will use the Open-CV module. We start with a baseline_image, which is the frame captured without any moving object inside it. As soon as the camera fires, the first image is set to our baseline_image, which means that we expect no moving object when our program first starts. Next, when somebody enters the frame, certain pixels in that frame will be different. We deduce this difference using the "cv2.absdiff" method.

There are a few steps we will take to achieve this.

1. *Capture the baseline_frame (with no object)*

1.1 Convert the frame to Gray

1.2 Smoothen the frame to remove noise

2. *Capture the new_frame (with object)*

2.1 Convert the frame to Gray

2.2 Smoothen the frame to remove noise

3. *Calculate the difference between the two frames*

3.1 ***If*** *difference is greater than the threshold, assume motion is detected*

3.2 ***Else*** *assume no motion detected*

First we will convert the image to gray scale and soften(blur) the image using a Low Pass Filter (LPF). A LPF is generally used in image processing to smoothen the image (eg:- used in skin smoothening, background blurring) and a High Pass Filter(HPF) is used to sharpen the image (eg:- sharpening of eyes and other details). This helps to increase the accuracy by removing the noise. We are using the GaussianBlur for this purpose.

The amount of Blur is up to us to play around. Here we have Gaussian kernel size width & height to be (25,25) and a standard deviation 0. The width and height should be a positive odd number. Think of it as the slider that changes the amount of blur in an image editing tool. The code to perform this is shown below.

```
#Gray conversion and noise reduction (smoothening)
gray_frame=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray_frame=cv2.GaussianBlur(gray_frame,(25,25),0)
```

The next step is to deduce the difference between the baseline and the new frame. We pass the two images to the cv2.absdiff() function. This is converted into a binary image using a method called Image Thresholding, meaning, if a particular pixel value is greater than a certain threshold (specified by us here as 35), it will be assigned the value for White (255) else Black(0). Now we have an image which has only 2 types of pixels (pure black or pure white, nothing in between). This is done so that we can identify the contour region around our detected object. We will use this to draw a green box around the object in the frame.

Now we will find all the contours in our binary image. ***Contour*** is simply a curve drawn along the perimeter or boundary having same colour or intensity. In essence, it will draw a curve around the white area on the black background. It expects the background to be black and the foreground object to be white. Using the cv2.findContours() method we will identify all the contours in our image. This method expects 3 parameters, **(a)** image, **(b)** contour retrieval mode and **(c)** contour approximation method.

This method returns the list of contours identified. We are using the cv2.contourArea() method to filter out any small contours that may not be of any interest to us. The cv2.boundingRect() returns the (x, y) coordinates of the top left corner along with the width and height of the rectangle containing the particular contour. We are then drawing a rectangle to show it in the screen.

```
#Calculating the difference and image thresholding
delta=cv2.absdiff(baseline_image,gray_frame)
threshold=cv2.threshold(delta,35,255, cv2.THRESH_BINARY)[1]# Finding all the contours
(contours,_)=cv2.findContours(threshold,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)#
Drawing rectangles bounding the contours (whose area is > 5000)
for contour in contours:
    if cv2.contourArea(contour) < 5000:
        continue
    (x, y, w, h)=cv2.boundingRect(contour)
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 1)
```

For playing the audio, we will be using “pyttsx3” python library to convert text to speech. You can choose your voice (male/female), the speed of the speech delivery and the volume. A sample piece of code which we used here is shared below.

```
import pyttsx3engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
engine.setProperty('rate', 150)
engine.say("Object Detected")
engine.runAndWait()
```

BACKGROUND SUBTRACTION APPROACH

This approach builds up on the foundation set by the frame subtraction approach. The principle of this method is to build a model of the static scene (i.e. without moving objects) called background, and then compare every frame of the sequence to this background in order to discriminate the regions of motion, called foreground (the moving objects).

This approach requires image manipulation to differentiate the foreground from the background. In general, the following manipulations are required. Assuming we have 2 images X and Y, we are manipulating these images to obtain image Z.

1. Difference

   The difference of two images of the same size and pixel format, produces an image, where each pixel equals to absolute difference between corresponding pixels from provided images.

   For each pixel (x) in Image Z:

   $red = \left| X.getPixel(x).R - Y.getPixel(x).R \right|$

   $green = \left| X.getPixel(x).G - Y.getPixel(x).G \right|$

   $blue = \left| X.getPixel(x).B - Y.getPixel(x).B \right|$

   $Z.setPixel(x) = Color(red, green, blue)$

   The reason why 32bpp images are not used is because if images with alpha channel are used, visualization of the result image may seem a bit unexpected – perhaps nothing will be seen (in the case if image is displayed according to its alpha channel)., the reason being the fact that after differencing the entire alpha channel will be zeroed (zero difference between alpha channels), what means that the resulting image will be 100% transparent.

2. Threshold It does image binarization using specified threshold value. All pixels with intensities equal or higher than threshold value are converted to white pixels. All other pixels with intensities below threshold value are converted to black pixels. For each pixel (x) in Image Z:

   If X.getPixel(x).Intensity > threshold

   Z.setPixel(x) = White

Else

Z.setPixel(x) = Black

3. Algorithm:

backgroundFrame – A grayscale image of the first image of the scene\video.
currentFrame – A grayscale image of the current frame of the scene.
threshold – The threshold that determine whether the movement is motion or not.

1. *We Calculate the Difference between the backgroundFrame and the currentFrame.*

For each pixel (x)

I(x) -> | backgroundFrame(x) – currentFrame(x) |

The image that we obtain is the one where all the pixels having same values (i.e. pixels that don't change) are zeroed out, and all the pixels that change (i.e. regions of motion) are highlighted. The work doesn't ends here, the resulting image will contain both relevant and irrelevant areas of motion. Now we have to filter those out.

2. *Using the threshold value as a Threshold for the image calculated in (1),*

For each pixel(x)

If I(x) > threshold

I(x) -> White

Else I(x) -> Black

By using an appropriate threshold value, we can filter out and neglect irrelevant areas i.e. movement of tree leaves in wind etc.

3. *Resulting image from (2) is then highlighted in the currentFrame to indicate areas of motion.*

4. *The last step is updating the background. This is done by moving the background to the current frame by a specified amount. If we replace our*

background with the currentFrame, this method becomes frame subtraction.

Updating the background is usually achieved by morphing the background slightly toward the currentFrame. The easiest form of morphing can be achieved by combining the two images by taking specified percent of pixels' intensities from the first image and the rest from second image. The value background percent value is set to 0.75. For each *pixel(x) in Image Z*

*Z.setPixel(x) = 0.75*background.getPixel(x) + (1 - .75)* currentFrame.getPixel(x)*

Use of OpenCV:

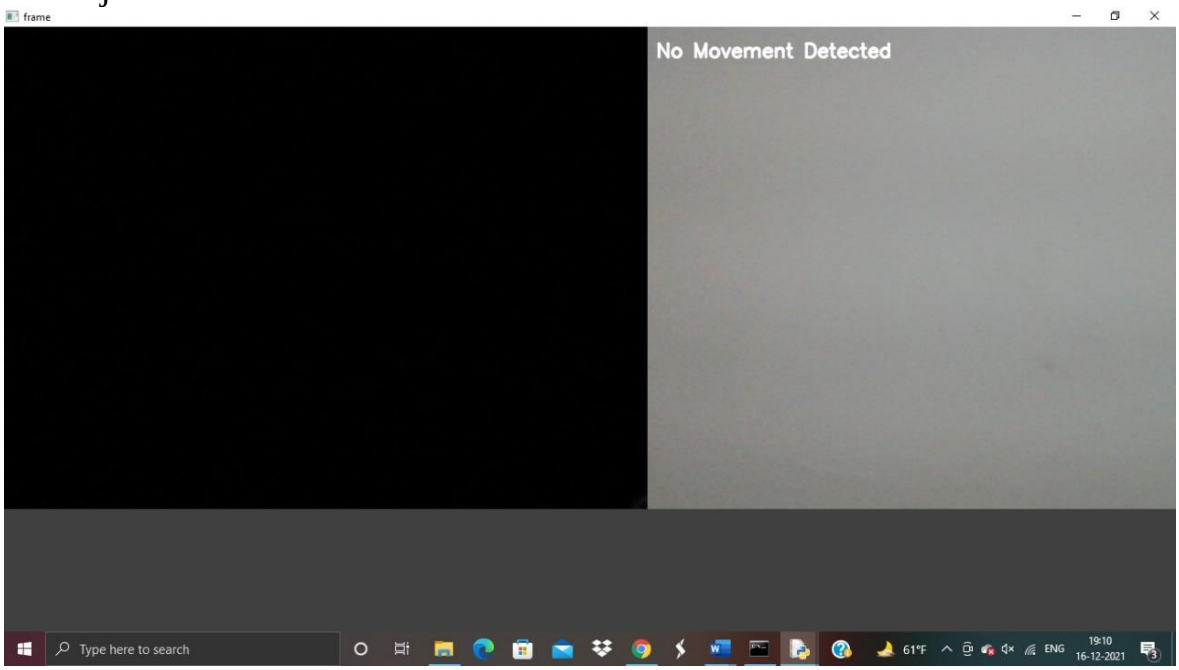
- OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

For playing the audio, we will be using “pyttx3” python library to convert text to speech. You can choose your voice (male/female), the speed of the speech delivery and the volume.This audio message will be used to alert user or authorized person of ongoing theft or any other kind of activity going on.

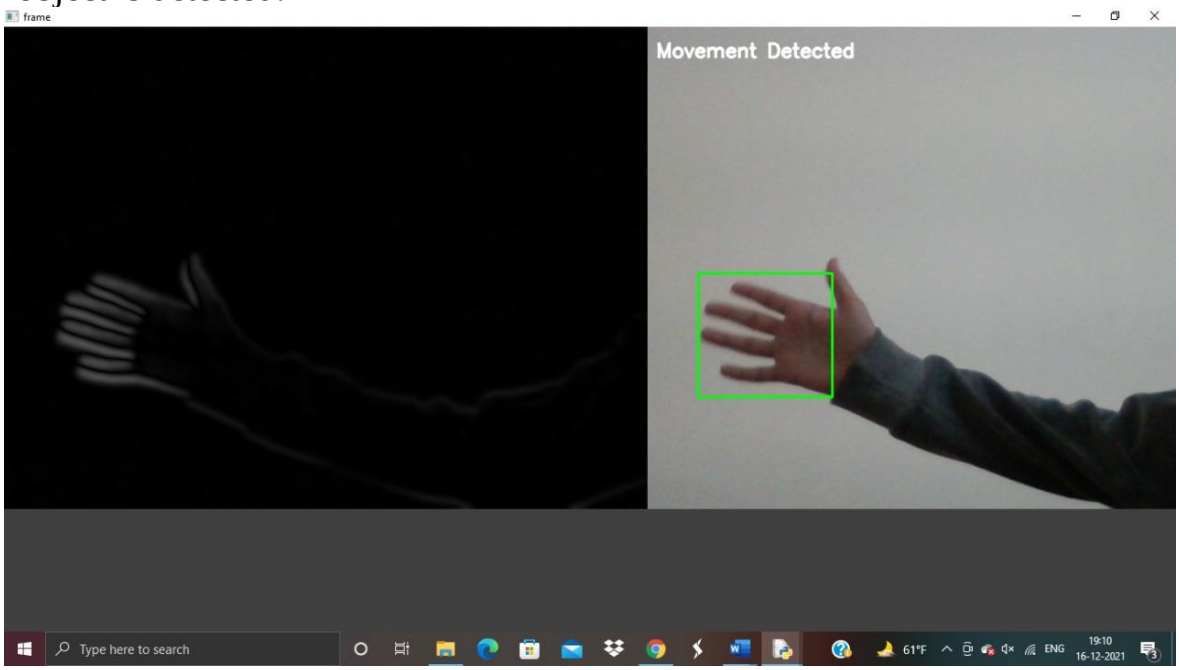
Results

The code for background subtraction for motion detection was executed and examined Successfully and an alarming alert is activated whenever a motion is detected.

When no object is detected:



When object is detected:



Conclusion

In conclusion, an effective monitoring system for motion detection and assessment tool has been developed. The level of motion is used as the input for the monitoring system to generate assessment to the motion detection to the particular person specifically. Results and findings show that the monitoring system is effective and consistent in producing relevant results to the detected motion. This monitoring system can be developed in the algorithm for speech recognition system in order to obtain more accurate and reliable voice input. In future, this system will be upgraded to mobilize resources to provide the necessary infrastructure, supplies and materials needed to ensure every assessment is achieving the motion analysis potential. This is important to increase the reliability and effectiveness of this monitoring system. A more detailed concept of motion detection will be more useful in later processing stages. As in image flow algorithm, all the information is need to be incorporated on the direction of motion. Optimization in realization is very important for a optimize solution from the beginning.

References

[1] Heijmans, H. J. and Ronse, C. (1990). The algebraic basis of mathematical morphological-I: Dilations and erosions, J. Comput Vision, Graphic, Image Process, Vol. 50.
[2] Ramprasad, P. and Randal, C. N. (1994). “Recognition of Activities”, Proc. International Conference on Pattern Recognition, Jerusulem, Israel, A815-820.
[3] Tetsuya, M., Makoto, N., Tomohiro, Y. and Shinji, T. (2010). “Comparison of Color Space in Extraction of a Hand Region for Computer Human Interface Using Color Image Processing”, Technical report of IEICE. PRMU 98(528).
[4] Neeti A. Ogale, “A Survey of Techniques for Human Detection from Video,”, unpublished.
[5] Prithviraj Banerjee and Somnath Sengupta, “Human Motion Detection and Tracking for video Surveillance,”, unpublished.
[6] Xiaofei Ji, Honghai Liu, “Advances in View-Invariant Human Motion Analysis:A Review,”in IEEE Trans.on Systems,Man, Cybernetics,vol.40,no.1,2010.
[7] Murat Ekinci, “Silhouette Based Human Motion Detection and Analysis For Real Time Automated Video Surveillance,”in Turk J Elec Engin,vol.13,no.2,2005.
[8] Hazi Wang and David suter, “Background Subtraction Based on a Robust Consensus Method ,” Monash University, Clayton Vic. 3800, Australia.
[9] Sumer Jabri, Zoran Duric, Harry Wechsler, Azriel Rosenfeld “Detection and Location of People in Video Images Using Adaptive fusion of color and edge information,”.
[10] Jing Li, Zhaofa Zeng, Jiguang Sun and Fengshan Liu, "Through Wall Detection of Human Being’s Movement by UWB Radar,"in IEEE Geoscience and Remote Sensing Letters,Vol.9, no.6, Nov 2012.