**APSIT SKILLS INTERNSHIP – PROJECT ABSTRACT**

**BE COMPUTER: Batch 2**

**Project Batch: 2**

**Team Members Names :**

Pradipt Kalamkar (Leader)

**Technology Selected:** Python3, OpenCV

**Project Topic Name :**

Basic Human Movement Detection using Python3

**Domain:** Digital Image Processing

**Topic Description :**

* In today’s modern world of CCTVs and surveillance, many systems are being developed to track human motion in order to keep an eye on any suspicious person or his activity.
* For this, our first step will be to detect every person in the frame of the camera properly.
* Hence, our aim in this project will be developing a system that should detect all moving objects in our camera frame with a minimal number of errors.
* The objective of this project will be to develop a system that should detect moving objects using Computer Vision (OpenCV library) in Python3.

**Technologies Used:**

* Python3
* Libraries used :

-OpenCV – For our processing our frames.

-NumPy – For use of arrays.

* IDE: Jupyter Lab

**Detailed Workflow :**

1. **Importing Libraries that we will need.**

OpenCV as cv2 and NumPy as np (NumPy library provides a high-performance multidimensional array and basic tools to compute with and manipulate these arrays. SciPy builds on this, and provides a large number of functions that operate on NumPy arrays and are useful for different types of scientific and engineering applications.)

1. **Capturing the Video (In our case we will use a test video)**

We will use the *VideoCapture* method for this.

1. **Convert the image into its grayscale.**

We will use the *cvtColo*r method for this

1. **Blur and carry out the filtering process on the video.**

We will use the Gaussian Blur and Simple thresholding. We have also used the dilate method to fill the holes.

**GaussianBlur**

|  |  |  |  |
| --- | --- | --- | --- |
| **dst = cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType=BORDER\_DEFAULT]]] )** | | | |
| Parameter | Description |
| src | input image |
| dst | output image |
| ksize | Gaussian Kernel Size. [height width]. height and width should be odd and can have different values. If kernel size is set to [0 0], then ksize is computed from sigma values. |
| sigmaX | Kernel standard deviation along X-axis (horizontal direction). |
| sigmaY | Kernel standard deviation along Y-axis (vertical direction). If sigmaY=0, then sigmaX value is taken for sigmaY |
| borderType | Specifies image boundaries while kernel is applied on image borders. Possible values are : cv.BORDER\_CONSTANT cv.BORDER\_REPLICATE cv.BORDER\_REFLECT cv.BORDER\_WRAP cv.BORDER\_REFLECT\_101 cv.BORDER\_TRANSPARENT cv.BORDER\_REFLECT101 cv.BORDER\_DEFAULT cv.BORDER\_ISOLATED |

**Image thresholding** is one of the simplest methods to separate regions which are higher than the set threshold.

Two broad types are

1. Simple or Global thresholding: Where one provides the threshold value as an input constant. This threshold is applied for all pixels of the image.
2. Adaptive thresholding: where the threshold is not a constant scalar - rather a distribution that is applied over a small window of pixels.

**The first argument is the source image, which should be grayscale.**

**The second argument is the threshold value which is used to classify the pixel values.**

**The third argument is the maximum value which is assigned to pixel values exceeding the threshold.**

OpenCV provides different types of thresholding which is given by the fourth parameter of the function. Basic thresholding as described above is done by using the type **[cv.THRESH\_BINARY](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html" \l "ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59" \o " )**. All simple thresholding types are:

* [**cv.THRESH\_BINARY**](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59)
* [**cv.THRESH\_BINARY\_INV**](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a19120b1a11d8067576cc24f4d2f03754)
* [**cv.THRESH\_TRUNC**](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576ac7e89a5e95490116e7d2082b3096b2b8)
* [**cv.THRESH\_TOZERO**](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a0e50a338a4b711a8c48f06a6b105dd98)
* [**cv.THRESH\_TOZERO\_INV**](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a47518a30aae90d799035bdcf0bb39a50)

The method returns two outputs. The first is the threshold that was used and the second output is the **thresholded image**

**Dilate**

cv.**Dilate**(src, dst, element=None, iterations=1) → None

|  |  |
| --- | --- |
| **Parameters:** | * **src** – input image; the number of channels can be arbitrary, but the depth should be one of CV\_8U, CV\_16U, CV\_16S, CV\_32F` or ``CV\_64F. * **dst** – output image of the same size and type as src. * **element** – structuring element used for dilation; if element=Mat() , a 3 x 3 rectangular structuring element is used. * **anchor** – position of the anchor within the element; default value (-1, -1) means that the anchor is at the element center. * **iterations** – number of times dilation is applied. * **borderType** – pixel extrapolation method (see **[borderInterpolate()](https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate" \l "int%20borderInterpolate(int%20p,%20int%20len,%20int%20borderType)" \o "int borderInterpolate(int p, int len, int borderType))** for details). * **borderValue** – border value in case of a constant border (see **[createMorphologyFilter()](https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate" \l "Ptr%3CFilterEngine%3E%20createMorphologyFilter(int%20op,%20int%20type,%20InputArray%20kernel,%20Point%20anchor,%20int%20rowBorderType,%20int%20columnBorderType,%20const%20Scalar&%20borderValue)" \o "Ptr<FilterEngine> createMorphologyFilter(int op, int type, InputArray kernel, Point anchor, int rowBorderType, int columnBorderType, const Scalar& borderValue))** for details). |

1. **Find Contours.**

What are contours?

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

Contours is a python list of all contours in the image.

Each contour is a NumPy array of (x,y) coordinates of object boundaries.

**Syntax:**

contours, hierarchy = [cv.findContours](https://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html#gae4156f04053c44f886e387cff0ef6e08)(thresh, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)

See, there are three arguments in **[cv.findContours()](https://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html" \l "gadf1ad6a0b82947fa1fe3c3d497f260e0" \o "Finds contours in a binary image. )** function,

**The first one is the source image**

**second is contour retrieval mode,**

**third is the contour approximation method**

Outputs given are the contours and hierarchy (optional o/p vector containing information about image topology)

1. **Once contours are located, highlight the movement using boxes.**

This is a relatively easy step where use the boundingRect, rectangle(for the shape), and puttext methods.

**putText**

 cv2.putText() method is used to draw a text string on any image.

**Syntax:** cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]])

**Parameters:**  
**image:** It is the image on which text is to be drawn.  
**text:** Text string to be drawn.  
**org:** It is the coordinates of the bottom-left corner of the text string in the image. The coordinates are represented as tuples of two values i.e. (**X** coordinate value, **Y** coordinate value).  
**font:** It denotes the font type. Some of font types are **FONT\_HERSHEY\_SIMPLEX, FONT\_HERSHEY\_PLAIN,**, etc.  
**fontScale:** Font scale factor that is multiplied by the font-specific base size.  
**color:** It is the color of text string to be drawn. For **BGR**, we pass a tuple. eg: (255, 0, 0) for blue color.  
**thickness:** It is the thickness of the line in **px**.  
**lineType:** This is an optional parameter.It gives the type of the line to be used.  
**bottomLeftOrigin:** This is an optional parameter. When it is true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

**Return Value:** It returns an image.

1. **Give the output video and save it.**

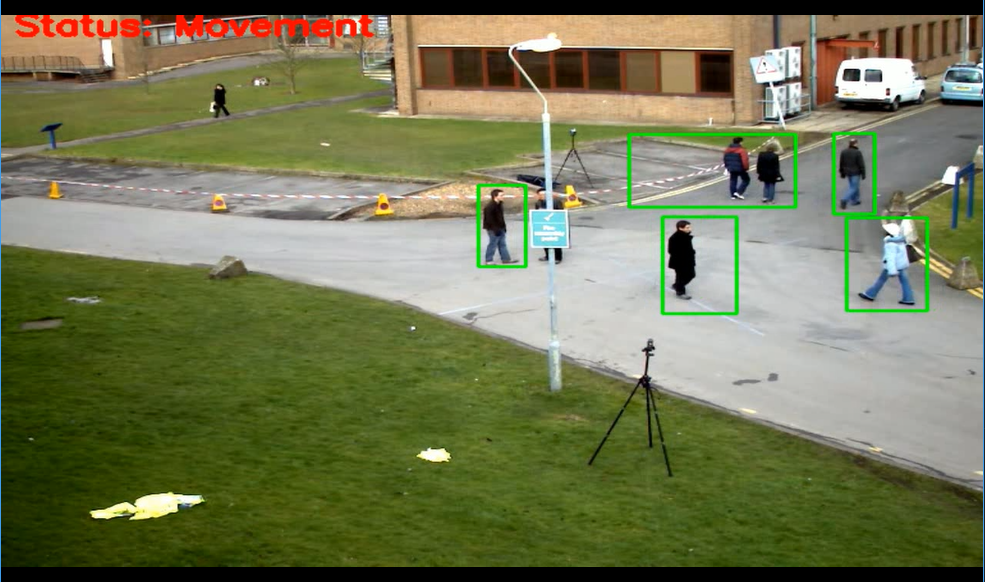
We will use the write and iamshow methods in this final step.

**GitHub / Drive link of project :**

<https://drive.google.com/drive/folders/1kwKXZgV9xjxpmsB7ifQFWXesxFPiZuI7?usp=sharing>

**Output Screenshots :**

****

****

****

**References :**

* D. Lee, H. Suk, S. Park and S. Lee, "Motion Influence Map for Unusual Human Activity Detection and Localization in Crowded Scenes", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 10, pp. 1612-1623, Oct. 2015.
* [Arun Kumar Jhapate](https://ieeexplore.ieee.org/author/37398268400); [Sunil Malviya](https://ieeexplore.ieee.org/author/37086393728); [Monika Jhapate](https://ieeexplore.ieee.org/author/37088468902) Unusual Crowd Activity Detection using OpenCV and Motion Influence Map https://ieeexplore.ieee.org/document/9170704/
* Varsha Shrirang Nanaware, Mohan Harihar Nerkar and C.M. Patil, "A Review of the Detection Methodologies of Multiple Human Tracking & Action Recognition in a Real Time Video Surveillance", IEEE International Conference on Power Control Signals and Instrumentation Engineering (ICPCSI-2017).

**Acknowledgment :**

I would like to thank my Project guide Prof. Archana Kotangale for giving me guidance and motivation. I would also like to thank Prof. Charles Severance whose wonderful PY4E course gave me an in-depth knowledge of python3 and the Coursera guided project coursework which helped me in learning the OpenCV library which is a big world in itself. Lastly but not the least, I would sincerely like to thank APSIT SKILLS for giving me this platform and access to wonderful courses on Coursera which have helped me in this project and the skills that I have learned, I hope they keep helping me in the future !!!!