**GestureSense**

**Aim -** The aim of this project is to develop a user-friendly navigation system for increased accessibility. By utilising the webcam, the system will enable app navigation and computer interaction using hand gestures.

**Description -**

**In Hand gesture recognition problem, to correctly identify hand gestures, the very first thing is to obtain a precise segmentation of the hand and finger regions from an image or a video stream.**

**Without such accurate segmentation, it becomes *extremely challenging*. Thus, when using 2D cameras as an input source of data, it’s common to apply background segmentation using a combination of:**

1. **Background subtraction**
   * **Traditional background subtraction relies on a static model of the background region.**
2. **Motion detection**
3. **Skin segmentation**
   * **The most preferable method, allowing us to *only* segment out regions of the image that contain “skin”. The challenge is to defining appropriate *color thresholds* for *skin regions*.**
     + **Example: Africans will have substantially darker skin tones than Europeans. And Asians will have dramatically different tones than both Africans and Europeans, like that.**

**By combining background subtraction and motion detection, we are able to adjust to our environment and more easily segment the hand region from the image.**

### **How we attack the problem??**

**Depending on my environment and lighting conditions, the problem could become challenging. So, what we do?**

1. **Pre-process by applying *Background subtraction* and *motion detection*, thus we are able to adjust to our environment and more easily segment the hand region from the image.**
2. **Segment the hand and finger regions from an image.**

### **Solution Approach:**

**Define a motion detection ROI(Region of Interest) of frames read from our video stream. We’ll then take this ROI, process it, and segment out the region that contains “motion” — this region will correspond to our hand and fingers.**

**Advertisements**

**REPORT THIS AD**

**For performing motion detection, we’ll be using the** [***running average model***](https://cvexplained.wordpress.com/tag/background-subtraction/)**. We’ll compute the weighted average over a set of *N* frames, where older frames contribute less to the average than newer ones.**

**We then take the difference between the *average background model* and the current frame to determine if motion is taking place.**

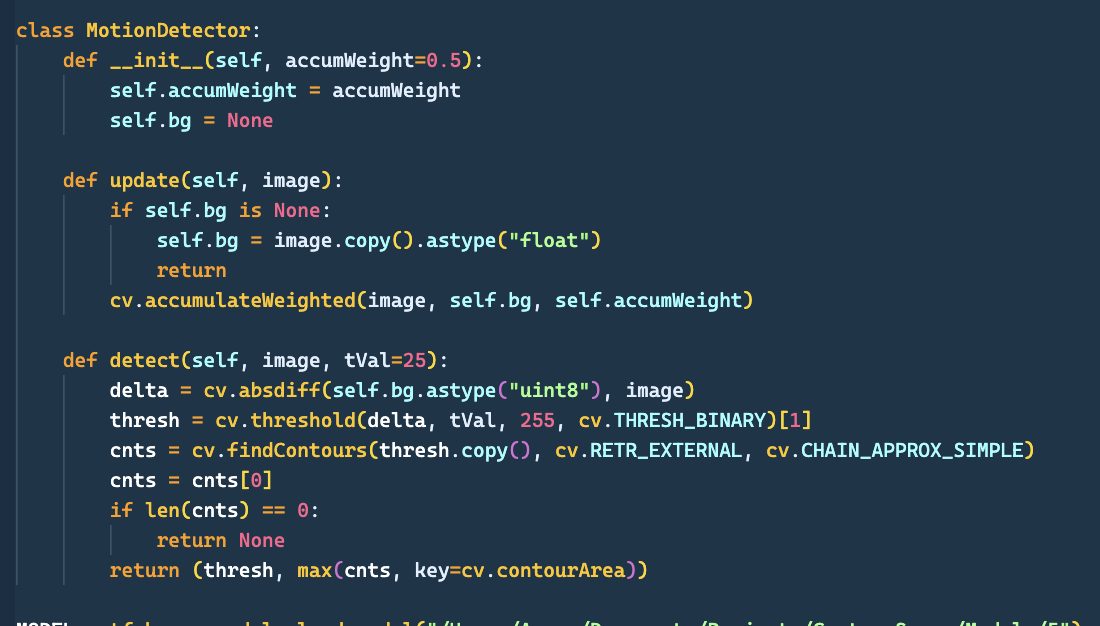
**The function we use to calculate average background model (using Running Average concept) is cv2.accumulateWeighted().**

**cv2.accumulateWeighted(src, dst, alpha)**

**The important parameter — alpha/accumWeight .**

* **The *larger* accumWeight is, the *less* older frames are weighted when computing the running average of the background. So, the average of the background updates faster.**
* **The *smaller* the accumWeight value, the *more* older frames contribute to the running average.**

### **1 Basic Segmentation – MotionDetector class**

**To start, let’s define the MotionDetector class.If our background model bg is None, we initialize it. Otherwise, we compute the weighted average over the background and supplied image using the cv2.accumulateWeighted function.**

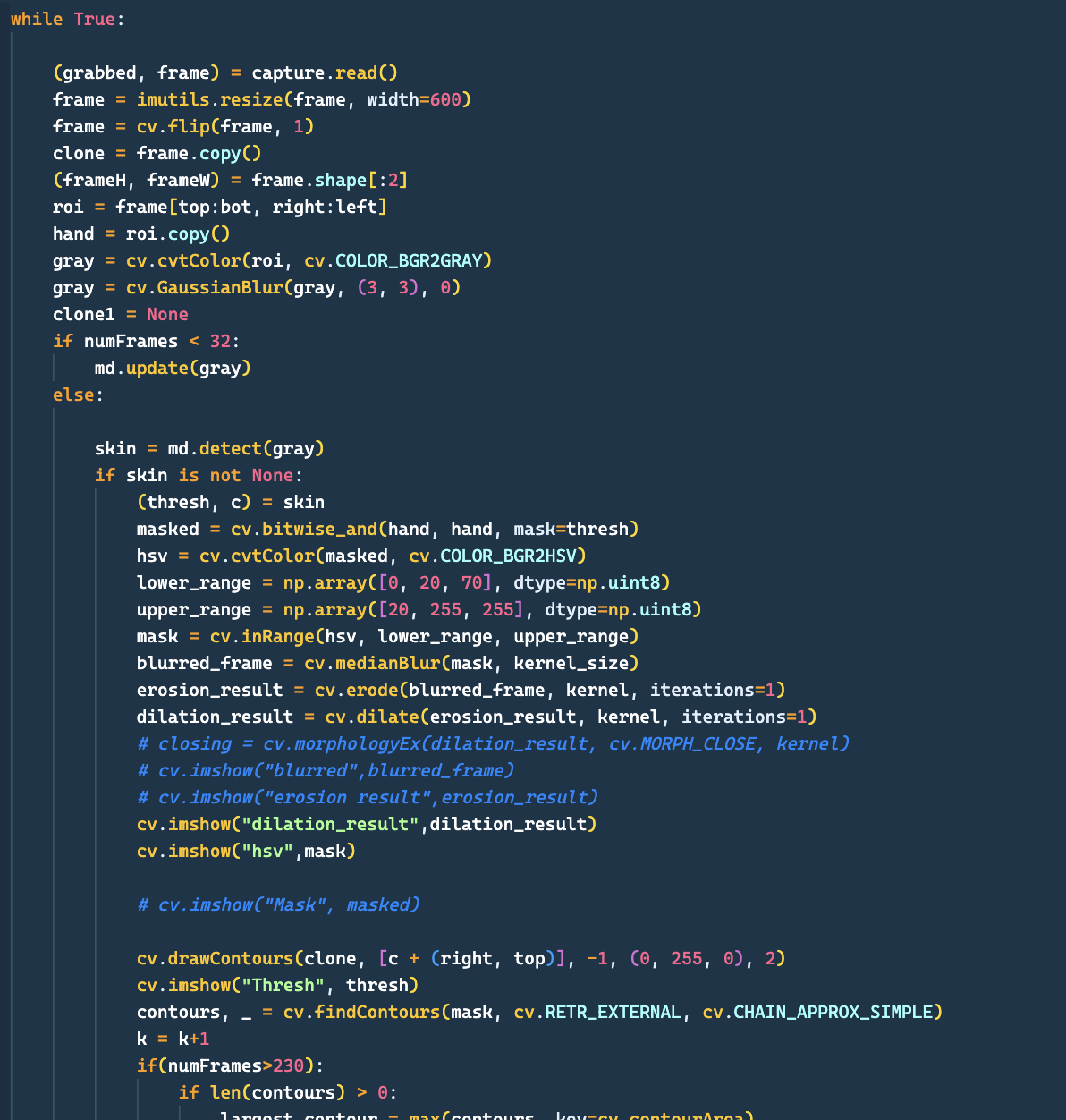
**Inside our detect() method, the cv2.*absdiff*  method computes the *absolute difference* between the background model and the image, leaving us with the delta (i.e., the difference image). We then threshold the delta image to find regions that contain substantial variation in pixel value — these regions correspond to *movement*.**

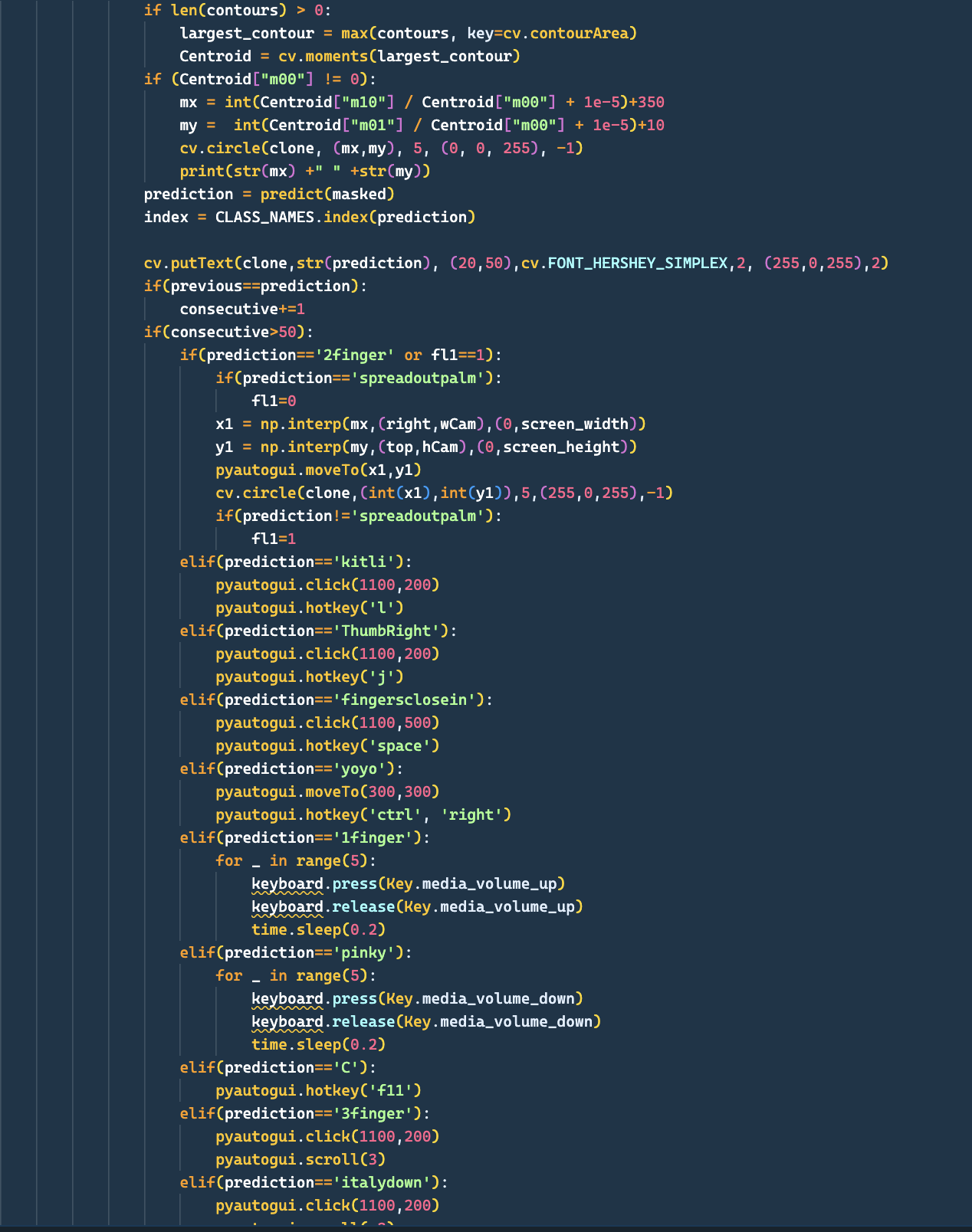
**We then find contours in the thresholded image.**

### 

### **2 Applying Motion Detector**

**Now that we have defined our MotionDetector class, let’s apply it to a specific ROI. We require coordinates of the ROI we are going to track for motion. We will *ignore* all regions of our frames *except* for this ROI region, which is defined using four points: *top,* *right*, *bottom*, *left*.**

**We will *only* be tracking motion (and thus be performing hand gesture recognition) inside the red bounding box ROI. Of course, when using this code in your own projects, you can define your own custom bounding box.**

****

****