**1. Importing libraries**

import cv2 # OpenCV library for computer vision tasks

import time # Used for adding delays

import imutils # Helps in image resizing and convenience functions

* **cv2**: Main library for capturing video, image processing, contour detection, etc.
* **time**: Used here to pause execution for 1 second before starting the camera.
* **imutils**: Handy library to simplify OpenCV operations (e.g., resizing, grabbing contours).

**2. Initialize camera**

cam = cv2.VideoCapture(0) # 0 means default webcam

time.sleep(1) # wait 1 second for camera to warm up

* VideoCapture(0) opens your computer’s default webcam.
* Some webcams take time to initialize, so sleep(1) ensures the first frame is stable.

**3. Setup variables**

firstFrame = None # store the very first frame (reference background)

area = 500 # minimum area for motion detection (to ignore small noise)

* firstFrame is used as the background reference image.
* area ensures only significant motion (larger than 500 pixels) is considered.

**4. Start infinite loop**

while True:

\_, img = cam.read() # capture a frame from webcam

text = "Normal" # default state (no motion)

* cam.read() grabs a frame. \_ stores success flag, img stores the image.
* Initially assume nothing is moving → text = "Normal".

**5. Preprocess image**

img = imutils.resize(img, width=1000) # resize frame for consistency

grayImg = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) # convert to grayscale

gaussianImg = cv2.GaussianBlur(grayImg, (21, 21), 0) # apply Gaussian blur

* Resize to fixed width (1000 pixels). Helps reduce processing load.
* Convert to **grayscale** because color isn’t needed for motion detection.
* Apply **Gaussian Blur** (21×21 kernel) to smooth out noise and avoid false detections.

**6. Capture first frame as reference**

if firstFrame is None:

firstFrame = gaussianImg # store first blurred grayscale frame

continue

* If this is the very first loop, store the blurred frame as the **background reference**.
* continue skips rest of loop and moves to next frame.

**7. Compare with reference**

imgDiff = cv2.absdiff(firstFrame, gaussianImg) # difference between current and background

* Finds absolute difference → areas with change become **bright (white)**.

**8. Thresholding & Morphological Processing**

threshImg = cv2.threshold(imgDiff, 25, 255, cv2.THRESH\_BINARY)[1]

threshImg = cv2.dilate(threshImg, None, iterations=2)

* threshold → converts diff image into **binary** (0 = black, 255 = white).
  + Only pixels with intensity difference >25 are set to white.
* dilate → fills small holes to get stronger white regions (removes small noise).

**9. Find moving regions (contours)**

cnts = cv2.findContours(threshImg.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

* Finds **contours** (boundaries of white regions = moving objects).
* RETR\_EXTERNAL → only outermost contours.
* CHAIN\_APPROX\_SIMPLE → compress contour data to save memory.
* grab\_contours fixes compatibility issues with different OpenCV versions.

**10. Loop through detected contours**

for c in cnts:

if cv2.contourArea(c) < area: # ignore small objects

continue

(x, y, w, h) = cv2.boundingRect(c) # bounding box for motion area

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2) # draw green box

text = "Moving Object detected"

* Each contour c is checked → if area < 500 pixels, it’s ignored (noise).
* boundingRect finds the rectangle enclosing the motion.
* Draws a **green rectangle** around detected object.
* Updates text to "Moving Object detected".

**11. Show result & display status**

print(text)

cv2.putText(img, text, (10, 20),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 2)

cv2.imshow("cameraFeed", img)

* Prints whether object detected.
* Puts text ("Normal" or "Moving Object detected") on video.
* Displays video feed in a window.

**12. Exit condition**

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

* waitKey(1) waits for key press (1 ms).
* If **"q"** is pressed → exit loop.

**13. Cleanup**

cam.release()

cv2.destroyAllWindows()

* release() → frees the webcam.
* destroyAllWindows() → closes all OpenCV windows.

✅ **In summary**:  
This code captures video from your webcam, sets the **first frame as background**, and then compares every new frame with that background. If a significant difference (motion) is detected, it highlights the moving object with a rectangle and shows "Moving Object detected". Otherwise, it shows "Normal"