**OPEN CV PROJECTS IN WINDOWS**

**PROJECT 2 : Real-Time Edge Detection using Webcam**

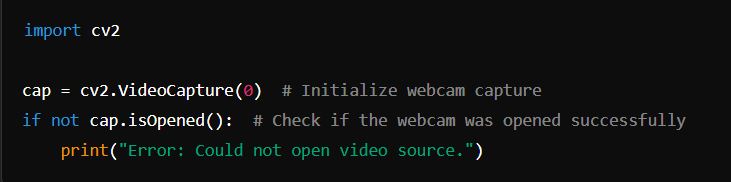
Real-time edge detection using a webcam is a simple yet effective computer vision project that captures video from a webcam, processes each frame to detect edges using the Canny edge detection algorithm, and displays the processed video in real-time. Here's a detailed explanation of how to implement this project using OpenCV in Windows.

### ****Objective:****

* Capture video from the webcam.
* Apply the Canny edge detection algorithm to each frame.
* Display the original and edge-detected video streams in real-time.

### ****Capturing Video from the Webcam :****

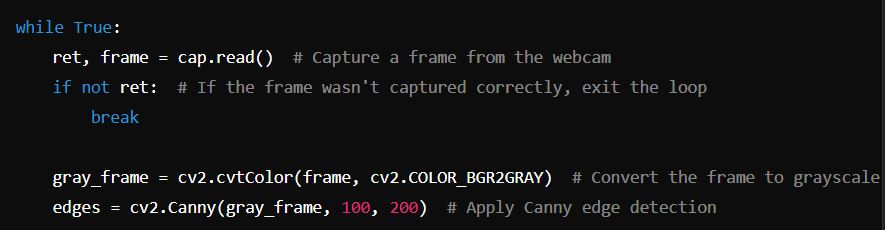
* To capture video from the webcam, you use OpenCV's cv2.VideoCapture() function.
* The VideoCapture object (cap in this case) is responsible for accessing the camera and streaming video frames from it.
* You specify 0 as the argument to cv2.VideoCapture() to use the default camera. If you have more than one camera, you can use 1, 2, etc., to access different cameras.



* cap.isOpened() checks if the webcam has been successfully opened. If not, it prints an error message.

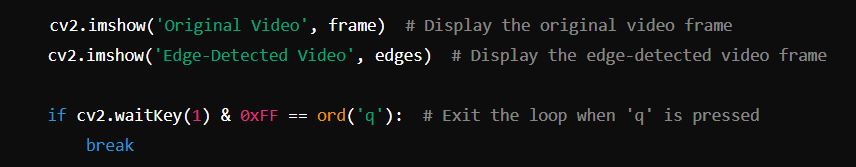
### ****Applying the Canny Edge Detection Algorithm****

* The Canny edge detection algorithm is a popular technique in computer vision to detect edges in an image. It works by identifying areas of the image where there is a rapid change in intensity, which typically corresponds to the edges of objects.
* Each video frame captured from the webcam is processed individually. The frame is first converted to grayscale (since Canny works on single-channel images) and then the Canny function is applied to detect edges.



* cap.read() captures the current frame from the webcam.
* cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) converts the captured frame from color (BGR) to grayscale.
* cv2.Canny(gray\_frame, 100, 200) applies the Canny edge detection algorithm to the grayscale frame. The two arguments, 100 and 200, represent the lower and upper thresholds for edge detection.

### ****Displaying the Original and Edge-Detected Video Streams in Real-Time****

* To visualize the real-time video processing, two separate windows are created: one for the original video feed and one for the edge-detected video feed.
* The frames are continuously captured, processed, and displayed until the user presses a key (e.g., 'q') to exit.
* cv2.imshow('Original Video', frame) displays the original video frame in a window titled "Original Video."
* cv2.imshow('Edge-Detected Video', edges) displays the edge-detected video frame in a window titled "Edge-Detected Video."
* cv2.waitKey(1) waits for 1 millisecond for a key press. If the key 'q' is pressed, the loop breaks, stopping the video capture and closing the windows.