Computer vision using python for environmental monitoring and measurement

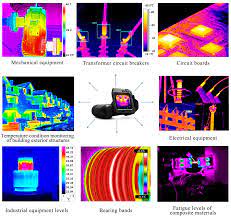
# What is computer vision technology?

It’s a computer science that focuses on enabling computers to identify and understand objects and people in image and videos.

## Opencv

OpenCV is a popular Computer Vision library to develop applications built using C++ and C. It has several uses like Object Detection and Video Processing. Computer Vision overlaps with fields like Image Processing, Photogrammetry, and Pattern Recognition. A popular wrapper Emgu CV is used to run OpenCV using C#.

### Uses of computer vision for environment monitoring





Measuring changes to our environment is an important part of understanding progress made toward a more sustainable world. Historically, measuring the world around us required time-intensive human monitoring and measurement by a small number of expert scientists. Thanks to advances in AI, we can now automate and scale understanding changes to our environment with the use of computer vision.

In this guide, you’ll learn how to build a remote sensing system, powered by computer vision, to measure distinct characteristics within aerial images. This process can be applied to understand changes in waterways, measure crop health, understand forest density, monitor deforestation, and many more environmental use cases.

#### Face recognition project:

Step 1: Setting Face Recognition Libraries

In order to install the face recognition library, we need to first install the dlib.

dlib : It is a modern C++ toolkit that contains ML-related algorithms and tools.

# installing dlib

pip install dlib

face recognition The actual face recognition library can be installed after dlib.

# installing face recognition

pip install face recognition

Opencv for some image pre-processing

# installing opencv

pip install opencv

Step 2: Loading Image

Step 3: Detecting and Locating Faces

The library face\_recognitioncan quickly locate faces on its own, we don’t need to use haar\_cascade and other techniques.

Step 4: Sample Image Recognition

The library face\_recognition is based on deep learning, it supports single-shot learning which means it needs a single picture to train itself to detect a person

import cv2

import numpy as np

def nothing(x):

pass

cap = cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

cv2.namedWindow("Frame")

cv2.createTrackbar("Neighbours", "Frame", 5, 20, nothing)

while True:

\_, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

neighbours = cv2.getTrackbarPos("Neighbours", "Frame")

faces = face\_cascade.detectMultiScale(gray, 1.3, neighbours)

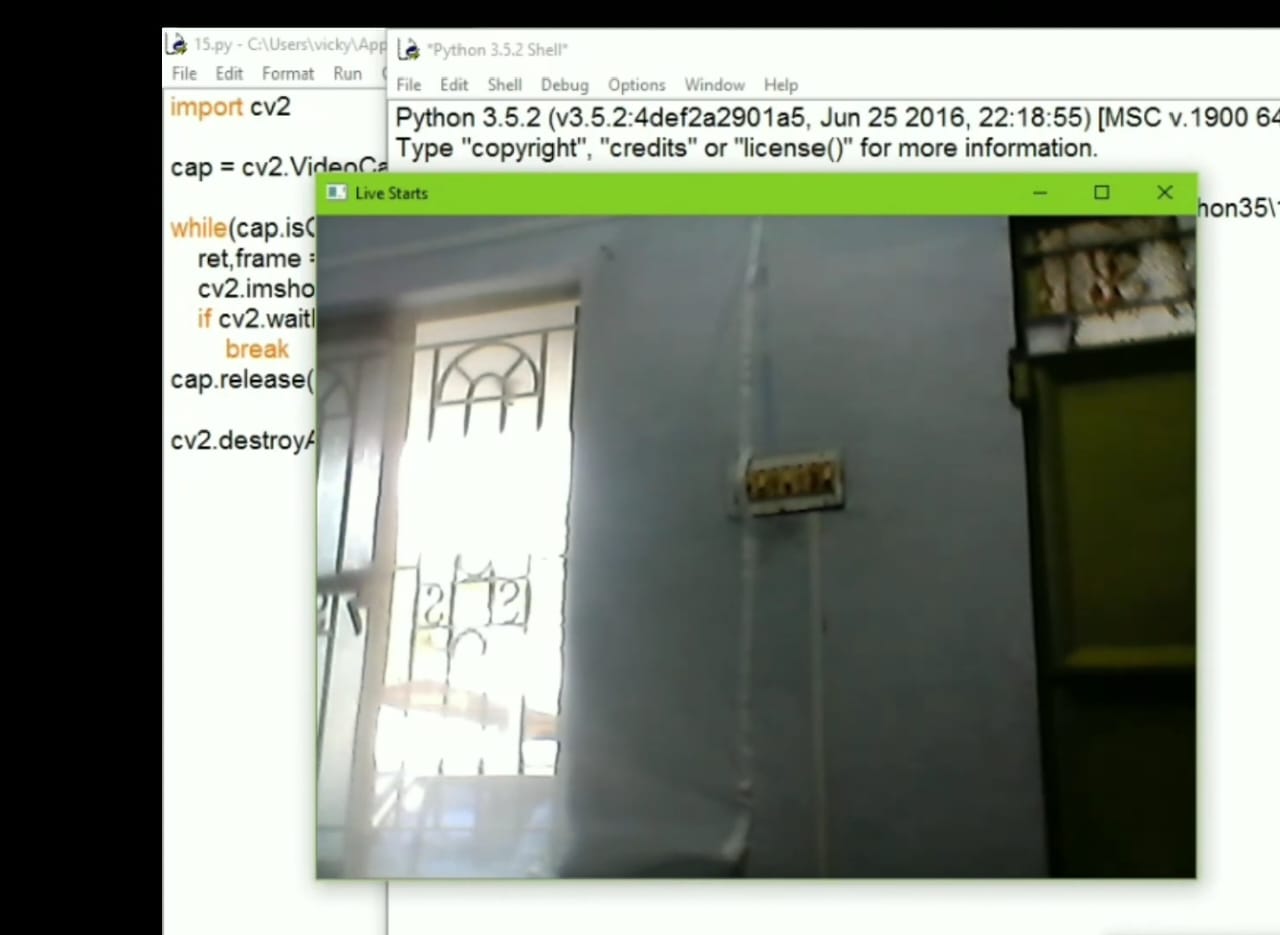
for rect in faces:

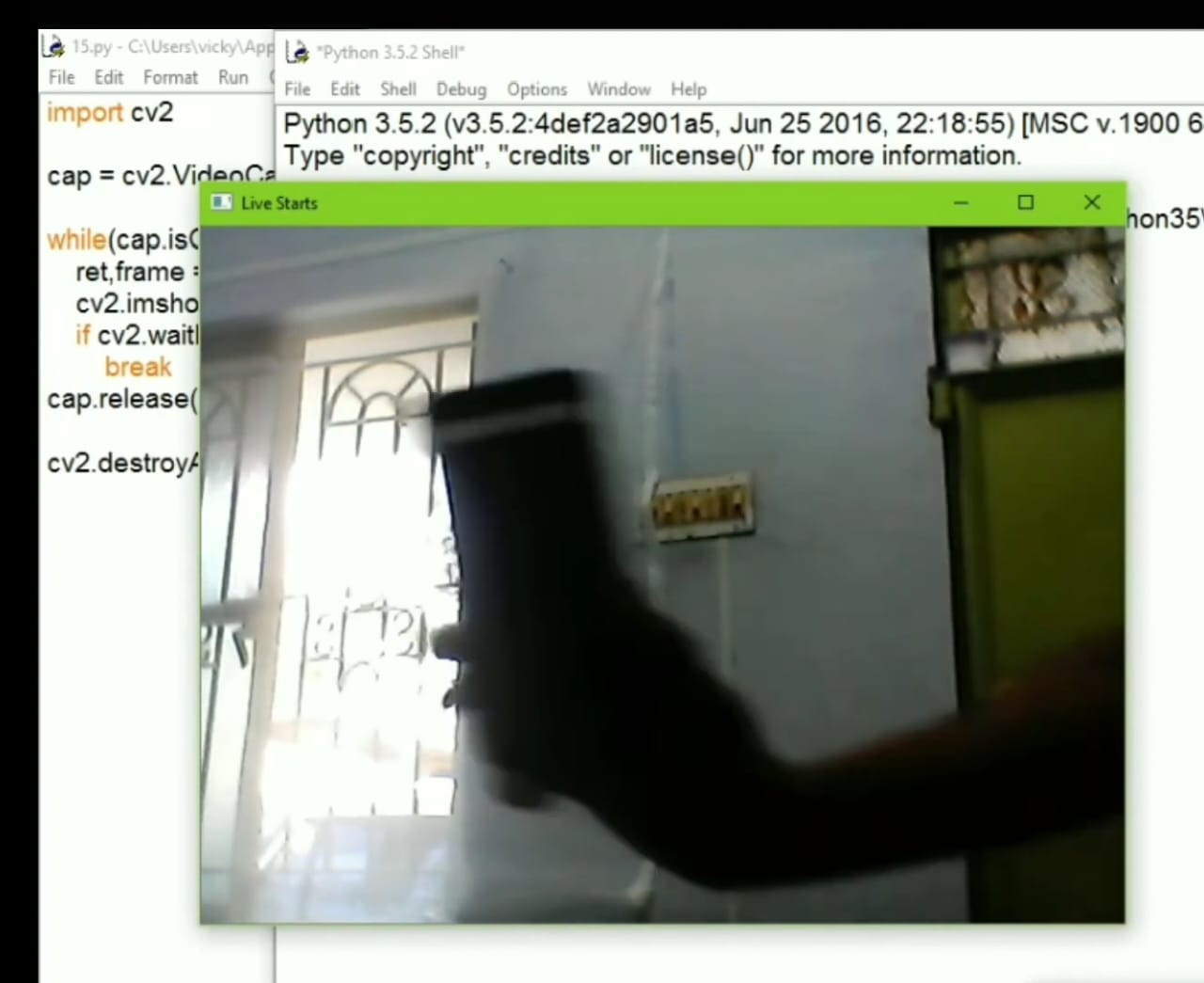
(x, y, w, h) = rect

frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

Then the result will look like this

Before the face shown





After the face shown



##### **Challenges in Recognition Systems**

Facing challenges while working on recognition systems in common, all you need to learn is how to get out of them. Here are some common challenges:

Pose: Recognition systems are susceptible to the human pose. Facial recognition systems will not be able to predict if the person’s face is not visible.

Illumination: Illumination changes the face contours drastically. Face recognition pictures should be clear with proper brightness.

Facial Expressions: Different facial expressions can result in different predictions of the same person’s Image.

Low Resolution: Low-resolution pictures contain less information, hence not good for face recognition training.

## Conclusion

This article discussed how to implement a face recognition system using python with a single-shot image training technique. You can further use GUI like python Tkinter to design a GUI-based attendance system. We saw various challenges that affect a recognition system and how to solve them. In the next article, we will create a face recognition attendance system using the same concepts which we have discussed today.