**Object tracking**

Võ Công Minh

Vietnamese – German University

Module: Programming 1

Drive:https://drive.google.com/file/d/1OLI9cs1chj8TBd\_rtU7NKcxUOmNZZnN3/view?usp=sharing

OBJECT TRACKING: The problem is about tracking and detecting the object as they move around frames in a video, or a list of images which are nearly the same. In this case, it is to track Professor Vinh’s face.

# Components to solve the problem

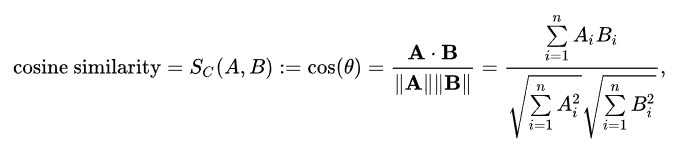
## Explaning:

* The basic idea is to follow the slight change of the object in each image, and keep detecting the object in the image so it does not slide away.
* More detailed, with the given template image of Mr.Vinh’s face, we have enough information to detect the position of the face. And after each frame, the change will be noticed and updated on the template image.

## Noticing the change:

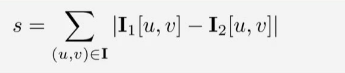
* To notice the change or to compare the current image to the previous one, we need to have some algorimths to deal with it. One of its are:

Cosine similarity: is a method to calculate the similar rate of two vectors. The value ranges is from -1 to 1 which means the exactly opposite and the exactly similar and the other between indicate the rate is how similar 2 vector is .

Here is how it is calculated:

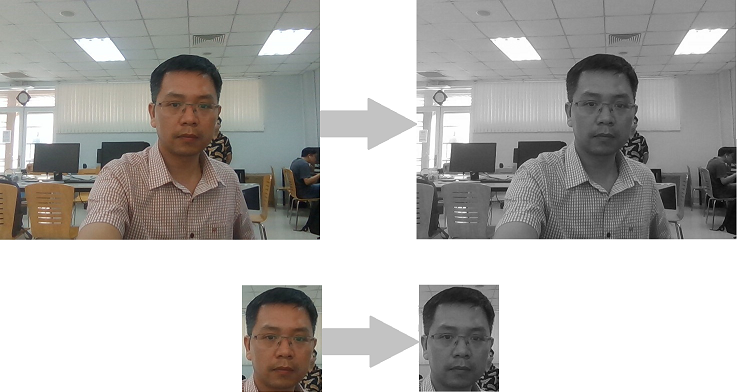
In the case, an image is expressed as an 1D array, in other words, we can say an image is expressed as an vector and we can use consine similarity to evaluate the similarity of the 2 image (or 2 vectors).

SAD(sum of absolute differences): a method to measure the similarity between image blocks. It is calculated by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison.

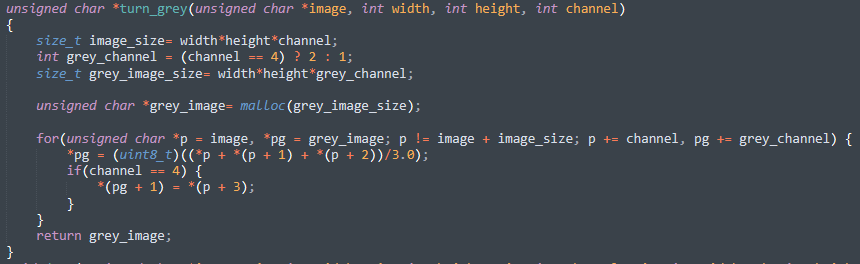


## Simplify the image:

* But to apply those methods above to the code, we need to simplify the image so that the algorithm can fit easily and at the same time, reduce the complexity without changing the result of the solution.



The code for that procedure:



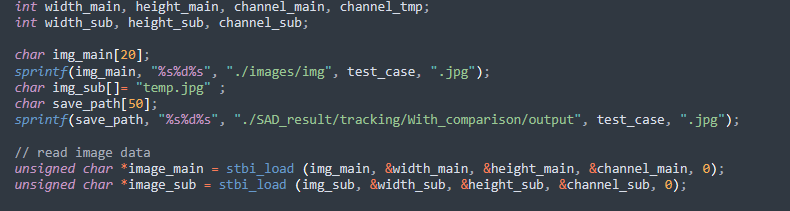
* That are all the tools we need to solve the problem.

# Steps to solve the problem:

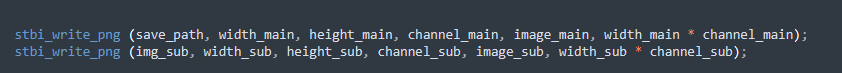
## I/O of the image:

* To get the information of the image as well as the image itself for image processing, we have to deal with massive things and problems. Instead of that, we can use STB\_IMAGE\_IMPLEMENTATION and STB\_IMAGE\_WRITE\_IMPLEMENTATION to get the data that we need.

For example, in the code:



I use stbi\_load in STB\_IMAGE\_IMPLEMENTATION to get the data of width, height and channel from it

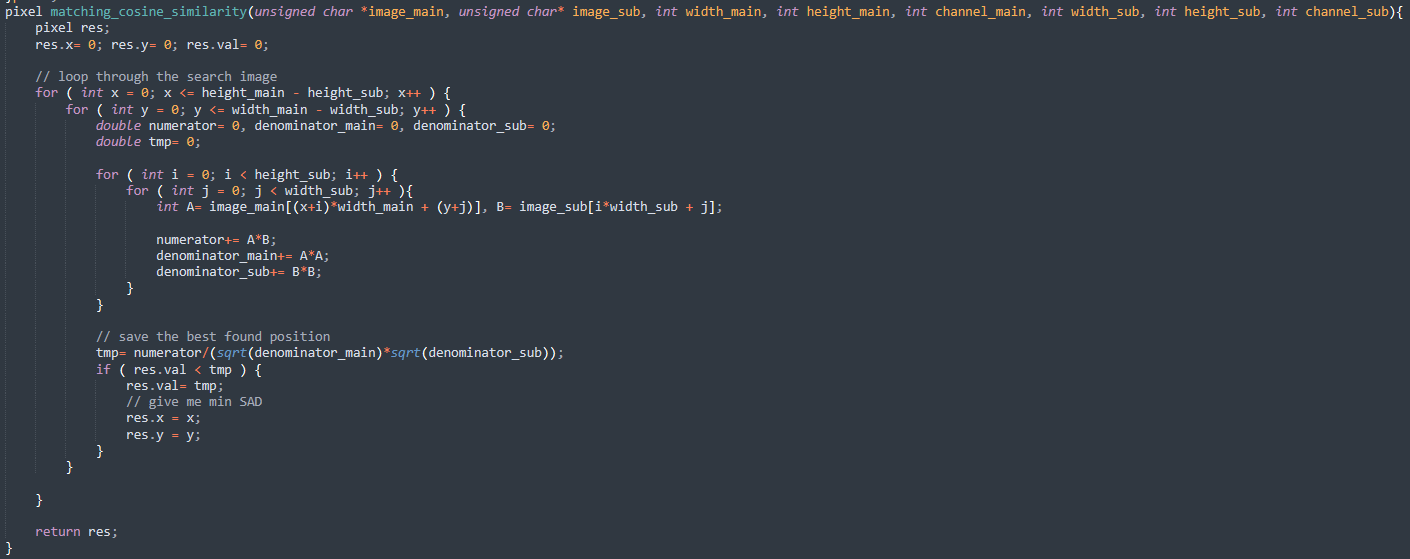


And, stbi\_write\_png from 2 library above to get the images out.

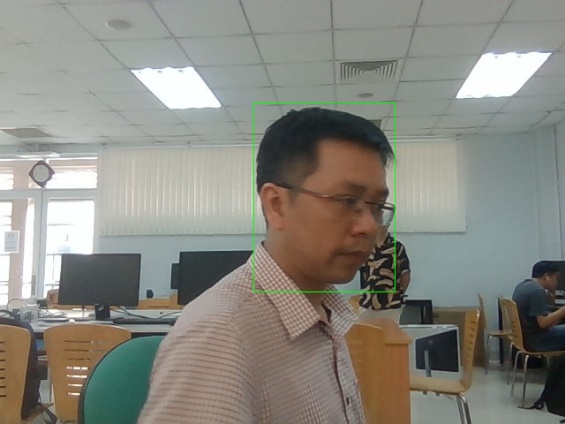
## Template matching:

After that, we need to detect the position of the template image in the given images so that we can get information to track the object. And as represented above, we have two method to detect the position:

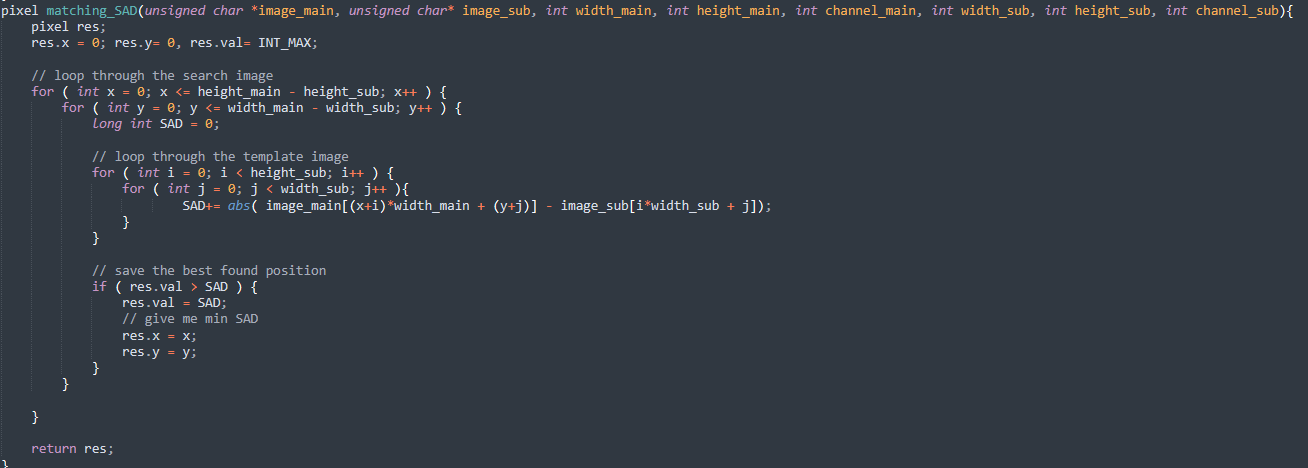
### Cosine similarity:



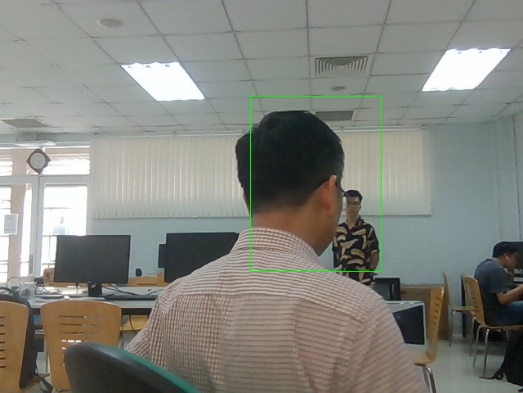
In this code, we divide the main image to small images that have the same size as the sub image(which here is the template image) and compare each of its to the sub image by using consine similarity. At the end of the procedure, the result will be the location where the highest similarity value can be found.



SAD:



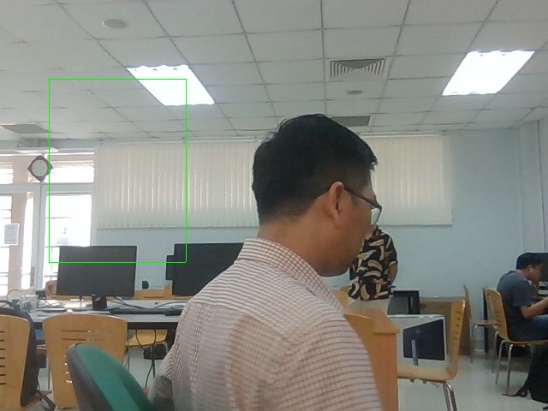
As, the same as above, we also need to divide the big main image to small image that have the same size as the sub image and we determine the similarity between 2 images by using SAD(have mentioned earlier).



## Updating after detecting the object:

Without updating the object each time having detected, the result would be really awkward as some of the frame is missing information to detect to object, so we would have both wrong result in both two algorithm:

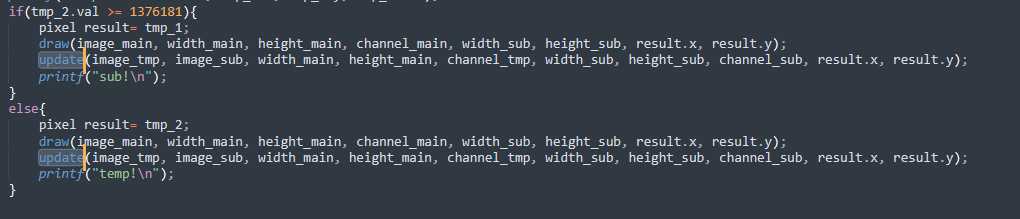
Cosine\_similarity:

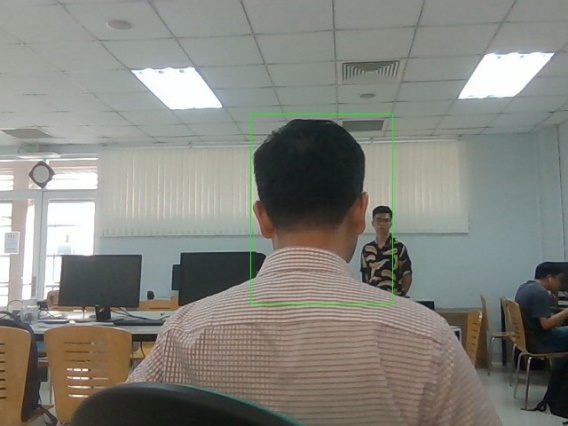
 

SAD:

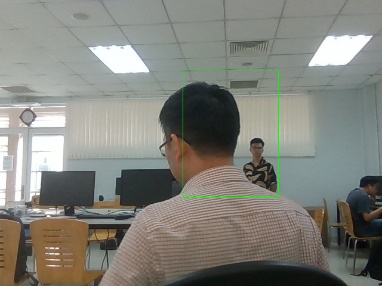
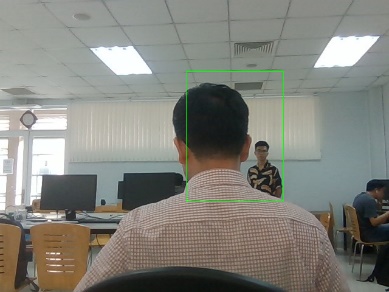
So, to fix that kind of issue, we need to update the change of each image time by time and I create a function to update the sub image each time it determine the position of the object



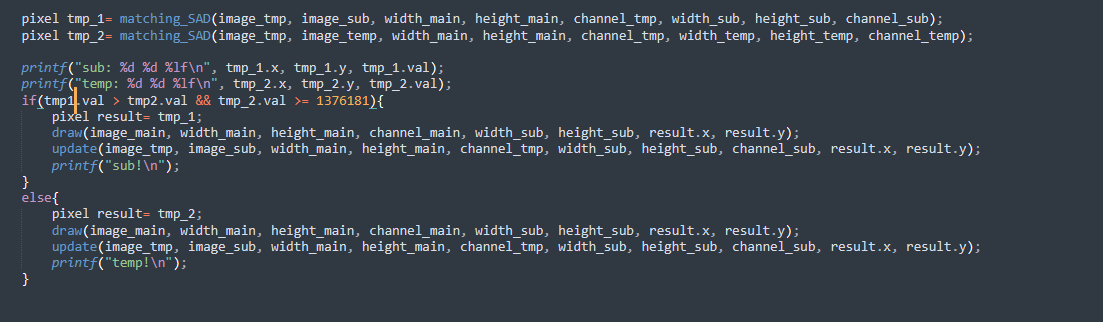


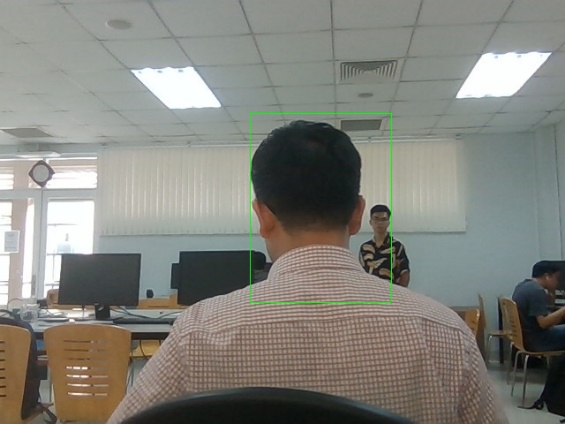
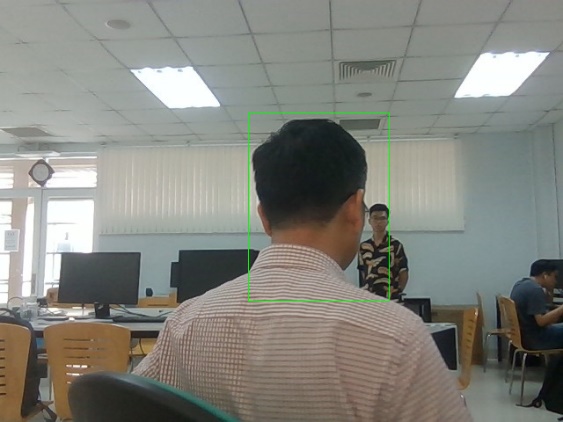
## Greedy upgrade method

However, the Mr.Vinh’s face always moves from picture not really smoothly enough to the computer the recognize the change and it causes errors. Here’s an example:

So I try to compare between the result after take the similarity between the original template image and the main image AND the similarity between the updated image and the main image:





And the result is surprisngly better. The number of wrong detection images reduced in both 2 ways SAD and similarity. You can check it by seeing the images in my google drive.

Although, I’m not really satisfied with the method as it’s only true with the current situation if you change the images or the template image, it may not be true. So it’s just a greedy method way to deal so it’s not optimized.

Drive: https://drive.google.com/file/d/1OLI9cs1chj8TBd\_rtU7NKcxUOmNZZnN3/view?usp=sharing

**That’s the end of my writing.**

**Thank you for reading.**