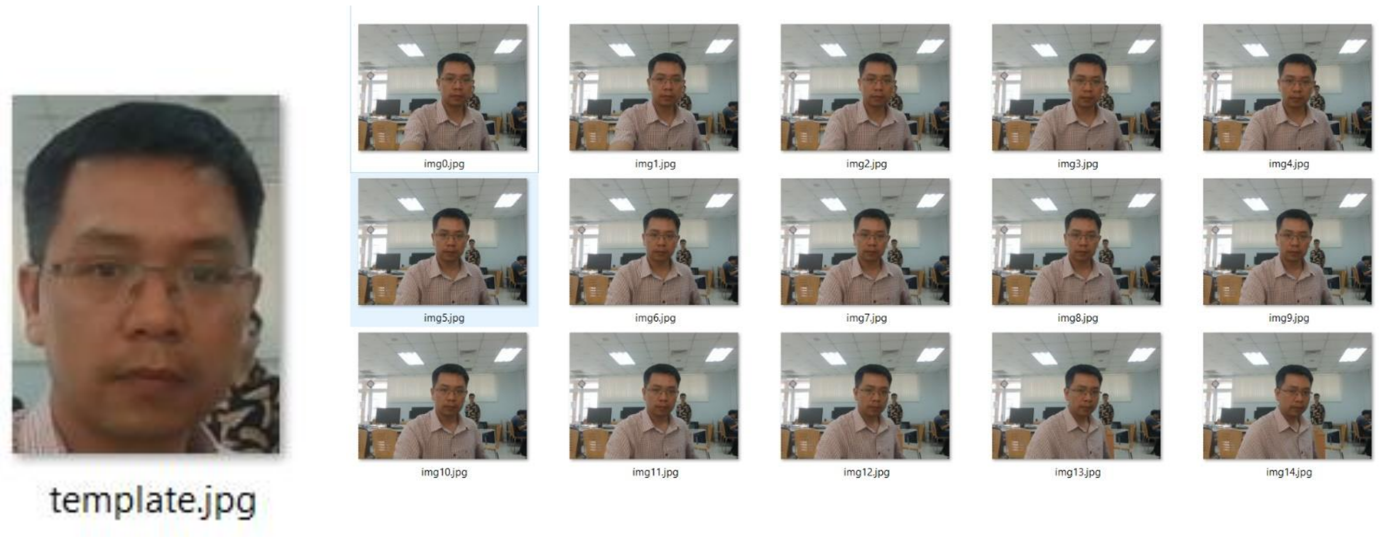
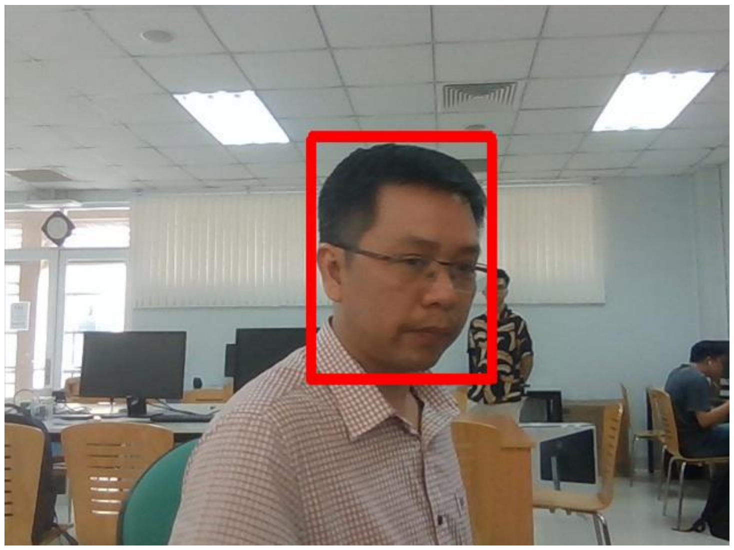
Name: Do Minh Quang
ID: 10421051

Programming 1 – Project 2

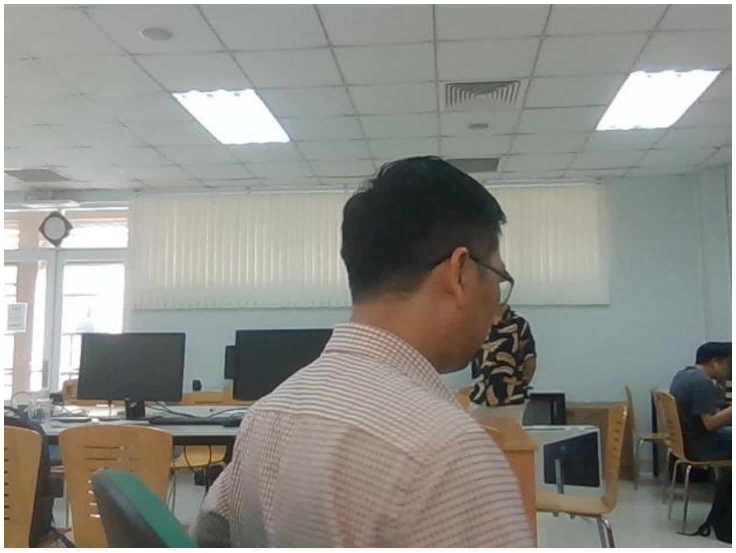
Project target: Given a template image containing an object and a set of background images, use the template to detect that object in a background images. (Object: front face of Mr.Vinh)



Input



Result1: object exists -> detectable



Result2: object not exist -> can't detect

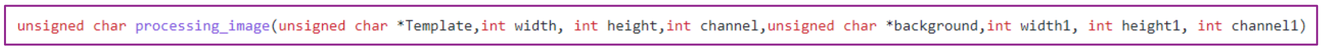
Process of solution:

Idea: slide the template over the image, calculate the likeliness between the area of the background and the template, if the likeliness is over a certain threshhold (over = object, under = not object) and draw the border of that area to red to mark the object.

Code:

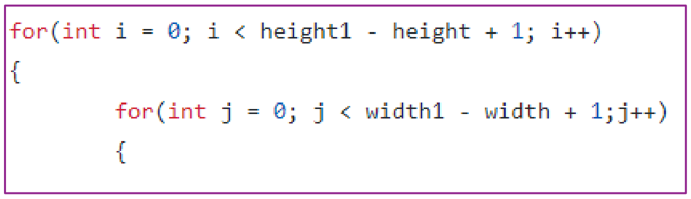
https://github.com/minWang916/object-detection-C-/blob/main/project%202/main.cpp

Function processing image is the function will detect the object:



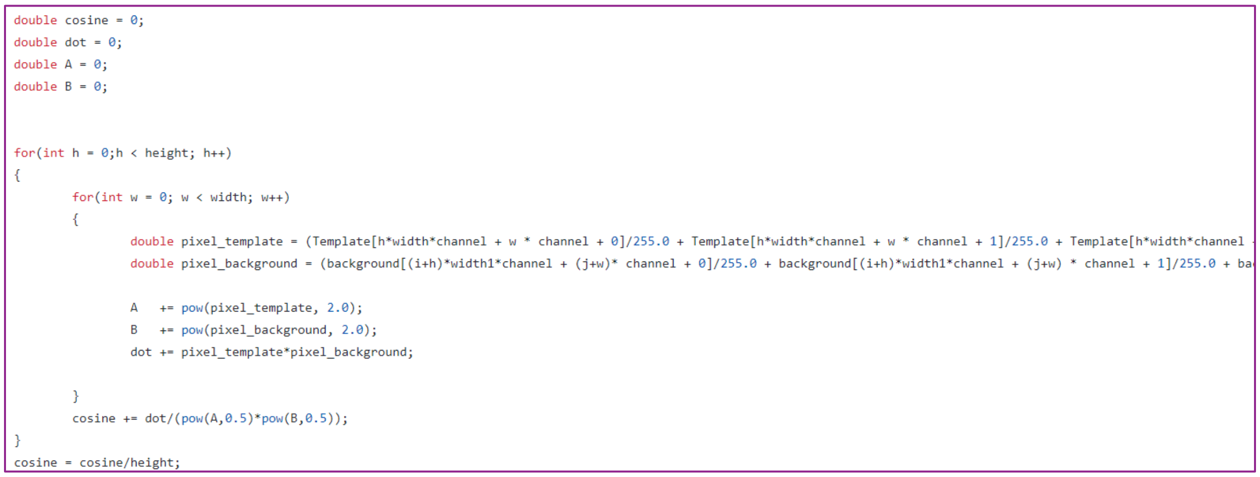
```cpp
unsigned char processing_image(unsigned char *Template,int width, int height,int channel,unsigned char *background,int width1, int height1, int channel1)
```

Its parameters are the template image, its width, height, channel and the background image with its width, height, channel.



```cpp
for(int i = 0; i < height1 - height + 1; i++)
{
        for(int j = 0; j < width1 - width + 1;j++)
        {
```

The function slides through the background image from left to right and from top to bottom. The number of times that it can slide in each row and column is the length of row/column (background) – length of row/column (tempalte) + 1. This is how the template is slide throught each eare in the background.



```
double cosine = 0;
double dot = 0;
double A = 0;
double B = 0;


for(int h = 0;h < height; h++)
{
        for(int w = 0; w < width; w++)
        {
                double pixel_template = (Template[h*width*channel + w * channel + 0]/255.0 + Template[h*width*channel + w * channel + 1]/255.0 + Template[h*width*channel
                double pixel_background = (background[(i+h)*width1*channel + (j+w)* channel + 0]/255.0 + background[(i+h)*width1*channel + (j+w) * channel + 1]/255.0 + ba

                A   += pow(pixel_template, 2.0);
                B   += pow(pixel_background, 2.0);
                dot += pixel_template*pixel_background;

        }
        cosine += dot/(pow(A,0.5)*pow(B,0.5));
}
cosine = cosine/height;
```

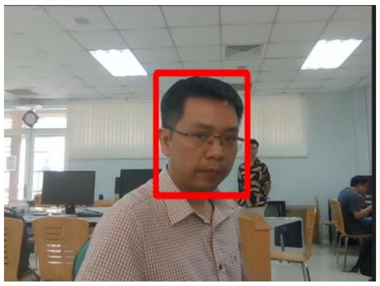
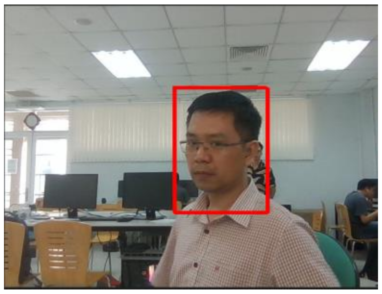
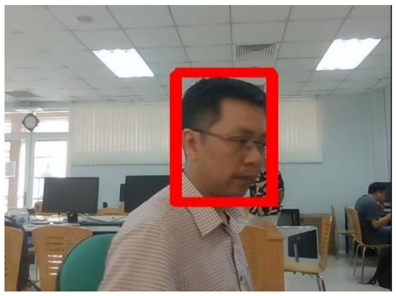
In each area, the template is compared with the area for the likeliness. The metric for this likeliness is ***cosine similarity.*** Each pixel of the template and the background is represented as the mean of the total value of its RGB value normalised from range ***0 – 255*** to ***0 – 1*** (so that the calculate is much faster)**.** Since cosine similarity is only used for 1 vector by 1 vector, we can calculate the cosine similarity in each row of the template with each row of the area and take the mean of the sum of the cosine similarity of the rows.

```
if (cosine > 0.93) {



                for(int w = 0; w < width;w++){
                        background[(i)*width1*channel + (j+w)* channel + 0] = 255;
                        background[(i)*width1*channel + (j+w)* channel + 1] = 0;
                        background[(i)*width1*channel + (j+w)* channel + 2] = 0;

                        background[(i+height - 1)*width1*channel + (j+w)* channel + 0] = 255;
                        background[(i+height - 1)*width1*channel + (j+w)* channel + 1] = 0;
                        background[(i+height - 1)*width1*channel + (j+w)* channel + 2] = 0;
                }

                for(int h = 0; h < height;h++){

                        background[(i+h)*width1*channel + (j)*channel + 0] = 255;
                        background[(i+h)*width1*channel + (j)*channel + 1] = 0;
                        background[(i+h)*width1*channel + (j)*channel + 2] = 0;

                        background[(i+h)*width1*channel + (j+width-1)*channel + 0] = 255;
                        background[(i+h)*width1*channel + (j+width-1)*channel + 1] = 0;
                        background[(i+h)*width1*channel + (j+width-1)*channel + 2] = 0;
                }



}
```

Then we compare it to a threshold (this threshold determined by firstly setting a beautiful value like 0.5 and keep increasing or decreasing it until it can distinguish what is the object or not the object). If the consine similarity rate is over ***0.93*** (this number is the result of me experimenting over and over), then that area is contains the object.

Then that area is colored **red** around its border to mark the location. (255,0,0) stands for **red**

Some of the results of the *processing_image* function:

The thick red border means it detects many location around that area matches with mr.Vinh's face. However, the capture results are still correct and able to locate his face.