Object Tracking

using Cosine Similarity

Nguyễn Phú Khang- 10421112
Instructor's name: Đinh Quang Vinh
9 November 2022

1 Introduction

To better understand where we are going, it is necessary to know where we are coming from; where the method has been made. It is needed to have knowledge about the basis of an image, what is the height, width and channel of an image, how it can be flatten into a vector and the way to calculate the similarity between two images.

To begin, given an image of size 640x480 with RGB channel, it is acknowledged that the actual size of an image is 640x480x3 as it has three color channels. Each pixel will be a combination of color to form a final color.

Figure 1: 3 channels of an image

When processing image in C language, it is compulsory to flatten the image into 1-dimension array, also known as vector. The flattening process of an image with three color channels will require a dynamic array of size[width*height*channel]. Each pixel will require 3 positions in the array for each color channel. The index or so-called coordinate of the pixel will be determined by the formula [i * width * channel + j * channel + k] where i, j, k are height, width, channel respectively.

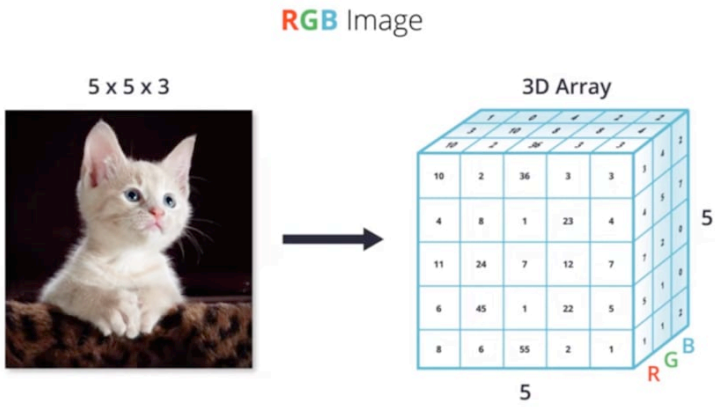
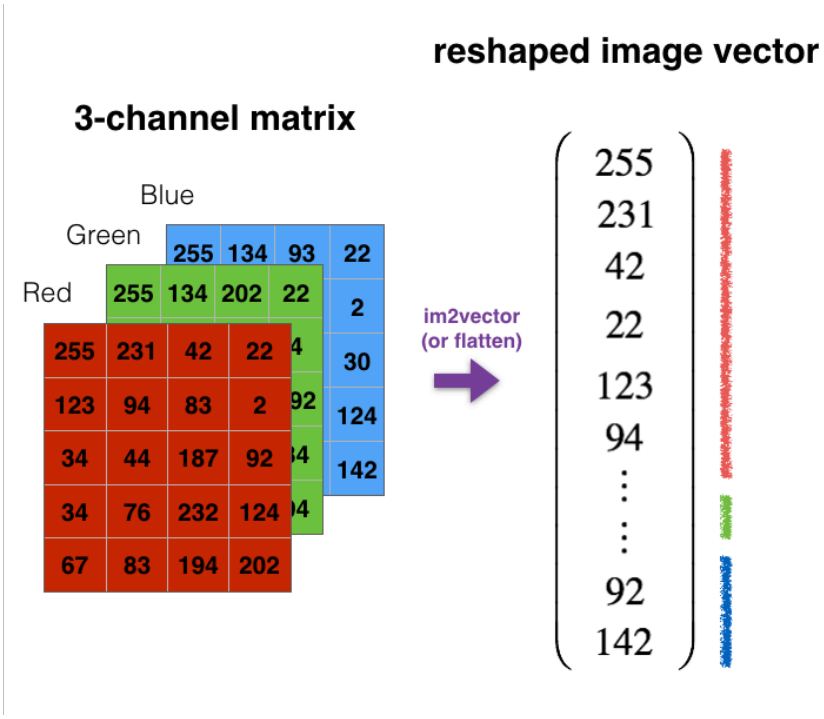


Figure 2: Matrix representation of colored image.

Each pixel of the image will always contain three layers that are called depth. When being flatten, each layer will be assigned next to each other as shown in Figure 3.



2 Cosine Similarity

2.1 Method:

The core idea of this algorithm is that using cosine similarity on flattened image, we can calculate the similarity between two images. Cosine similarity can easily compute the cos(θ) between two vectors. Since images are converted into 1D array, the task is much easier.

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 4: cosine similarity formula

The angle θ oscillates based on the features of the two vectors A and B. If there is a significant number of alike features, the angle θ would likely to be small and hence, the cos(θ) would be large.

How can we implement cosine similarity into image domain? That is when the flattening process comes in place. The template and target image would be converted into vectors and cosine similarity is used to calculate the differences between the template and each segments of image.

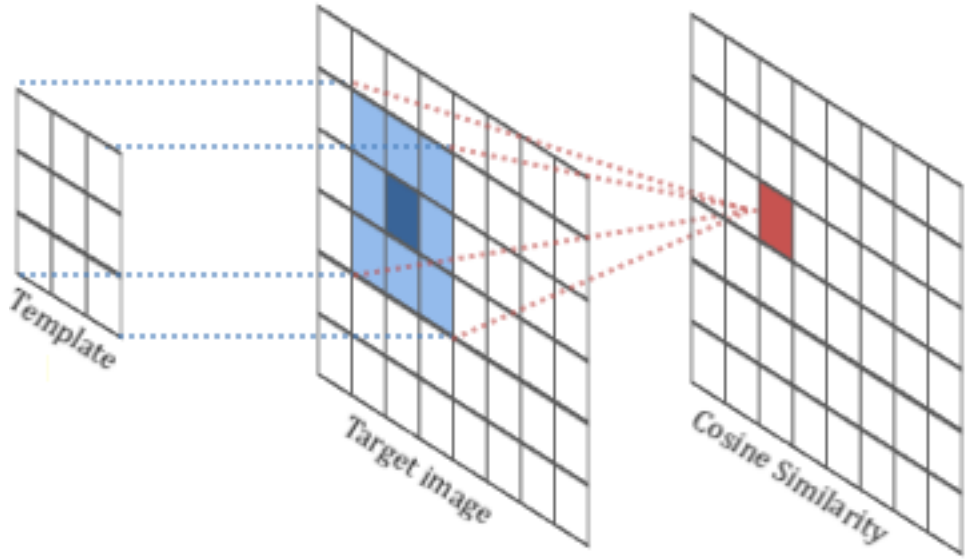
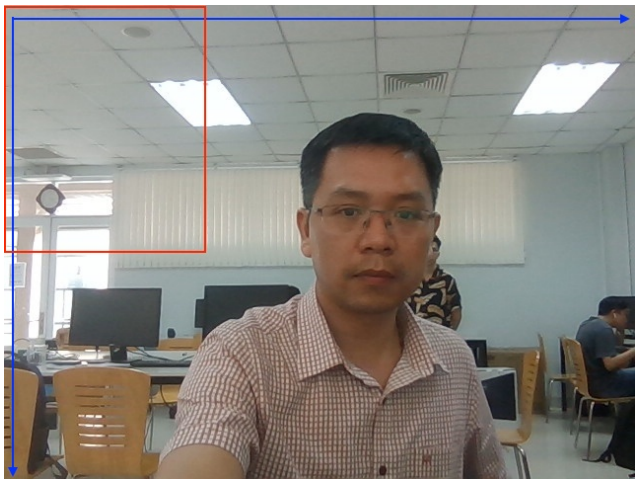


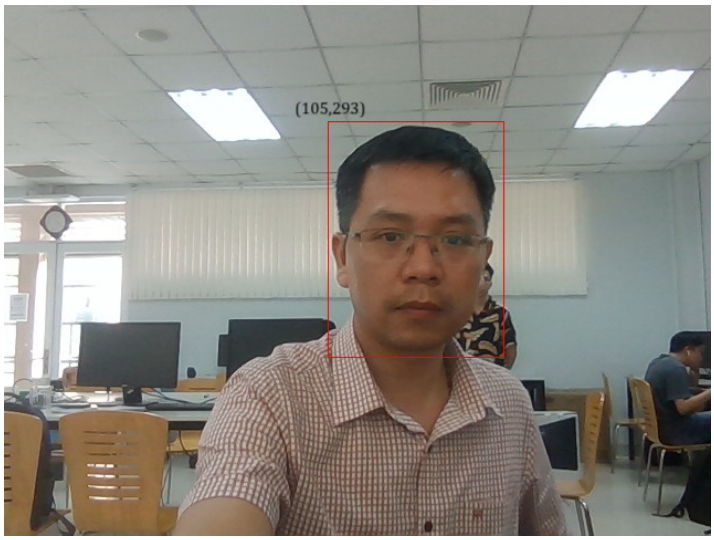
Figure 5: Template and target image cosine similarity

2.2 Implementation

The first step is to find the coordinate, or so-called index of the template inside the first target image. The first-time run would scan and calculate all the cosine similarity of each pixel and the pixel with largest similarity would be determined using the max function.



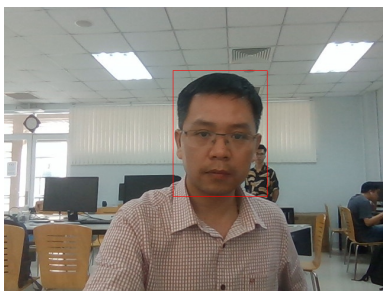
The resulting coordinate I have found is *[105, 293]*.



Base on this coordinate, we can limit the scanning space to 100 < i < 120 and 280 < j < 300 to reduce the executing time.

In the next steps, the process is quite repetitive since it finds the index of the max cosine similarity value and draw a bounding box around the target object.





The next 3 steps of the process

But the box tend to when the target object move overtime, thus I have to expand the scanning space to a larger area, which is 250 < j < 300.

3 Conclusion:

The objective of this project is to use cosine similarity to calculate the differences between two flattened images. We have known how to compute and track the target object in the image by scanning and applying template.

One disadvantage of this algorithm is a long processing time, although the result bounding box is quite accurate to the target object. Reducing the scanning area may be the solution, though the result would be negatively affected.