Saniya Nazeer

FIRE 198

Dr. Nitin Sanket

March 21, 2022

ASN4

Step 2:

1.

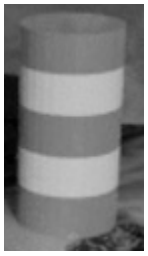
```
# Step 2

# Question 1

# reading in a template image
# then converting the template into grayscale
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

Template used:



I read in the template and saved it as “image”. Then I converted the image into grayscale and named that the template.

The following code is for questions 2, 3, and 4.

```
# Questions 2, 3, 4

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (max_loc[0] + w, max_loc[1] + h)

    cv2.rectangle(frame, max_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

2. The video was read frame by frame through a while loop and each frame was converted to a grayscale image one by one using cvtColor and the parameter cv2.COLOR_BGR2GRAY.
3. Using the OpenCV's Template Matching tutorial, I was able to create a barrel detector. cv2.matchTemplate allows the comparison of a template against an overlapped image through patches/regions.
4. Through taking the correct sizes of where the template matches the frames of the video, a boundary box was created using cv2.rectangle.

5. The first template matching method I used was TM_COEFF, as seen from the code above. The following code is all the other methods (TM_CCOEFF_NORMED, TM_CCORR, TM_CCORR_NORMED, TM_SQDIFF, TM_SQDIFF_NORMED) used.

```python
# Question 5
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (max_loc[0] + w, max_loc[1] + h)

    cv2.rectangle(frame, max_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

```python
import cv2

# Question 5
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_CCORR)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (max_loc[0] + w, max_loc[1] + h)

    cv2.rectangle(frame, max_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

```python

# Question 5
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_CCORR_NORMED)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (max_loc[0] + w, max_loc[1] + h)

    cv2.rectangle(frame, max_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

```python
# Question 5
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_SQDIFF)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (min_loc[0] + w, min_loc[1] + h)

    cv2.rectangle(frame, min_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

```python
# Question 5
image =  cv2.imread("C:\\Users\\nazee\\Desktop\\templateNew2.PNG")
template = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

w, h = template.shape[::-1]

cap = cv2.VideoCapture("C:\\Users\\nazee\\Desktop\\Vid.mp4")

while(True):
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(gray, template, cv2.TM_SQDIFF_NORMED)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    bottom_right = (min_loc[0] + w, min_loc[1] + h)

    cv2.rectangle(frame, min_loc, bottom_right, 255, 2)
    cv2.imshow('gray', frame)

    k = cv2.waitKey(30) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows
```

All of these methods seemed to work around the same. However, through a bit of research, I was able to understand that TM_CCORR, which is cross-correlation and that is the least effective, especially when using a larger image. This is because a part of an image could be more correlated to the template than another part of the image when in reality those two parts of the image should correlate about the same amount to the template. As for the most effective, I found that TM_SQDIFF_NORMED worked the best. I tried playing around with different templates and TM_SQDIFF_NORMED seemed to match more accurately and for a longer period of time than the other methods. I believe this is because the values calculated to that method that is closer to zero are more likely to be relevant. However, with some of the other methods it is based on how small or large the value is. In other words, the other methods are not as precise because there is no definitive number that the values calculated should be close to.

6. To use color in a template matching method, one can split the frames and templates into separate R, B, and G images and compare the templates that way and then put the images back together (the RGB channels) to get color.

Third-Party Code Used:

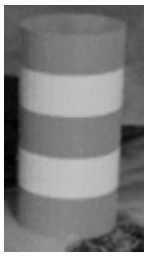
https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html

Challenges and Feedback

Overall, this was one of the easier assignments so far. I did not face many challenges besides trying to find a template that would match the video properly. To get the template, at first, I tried screenshotting the mask video however after many failed attempts in the next parts of the assignment I decided to use the other video and saved many frames of that video. From there I had to do a bit of playing around to determine which image would be best for the template.

This was one of the first templates I tried and it kind of worked, however, the bounding box also contained a large portion of the video that was not the barrel.

I also tried using this image as the template, however with this one a boundary box was not showing up. I believe this is because the image was too large so when the template does try to match up through parts of the frame of the video, the white barrel was not aligning to the real barrel.



Lastly, I ended up using this image as my template because, with the matchTemplate function in OpenCV, the template has to be more precise and exact so that it can detect the object properly. The template will not match the image if it is rotated or scaled in a different way.