Writeup for Template-matching based Target Tracking

Overview

The objective of this machine problem was to implement template based tracking on a video. This process is done here frame by frame where it matches the template image to the image provided and draws a box around the expected location of where the object being tracked went.

Packages Used

- NumPy: Utilized for numerical operations and array manipulation.
- OpenCV: Utilized to load in the images and convert the stream of images to a video output.

Algorithm

Template Tracking Algorithm

The main algorithm here is performed by one function called template tracking. For a normal tracking, A template is chosen from the first picture where I created a bounding box and then a template from it after cropping the image. Here the inputs are frame image, the template and a method to do the tracking.

First image and the template is turn to grayscale as it helps better with image tracking. For each method, the algorithm goes over the full image and measures error metric for the selected method. This is very similar to a convolution.

There are three tracking methods :

1. 'ssd' - Sum of Squared Difference :
    - For each instance, the difference between two image matrix is taken. This difference is then squared and each element is added up to get the error metric. The position of x and y which leads to the minimum value of ssd is chosen and then a bounding box is created around it.
2. 'cc' - Cross-Correlation :
    - For each instance, the cross-correlation between the template and a portion of the image is calculated. This correlation is calculated by multiplying two images and taking the sum of it. The maximum correlation is chosen and the indices for that correlation are chosen and then bounding box is created around it.
3. 'ncc' - Normalized Cross-Correlation:
    - For each instance, the normalized cross-correlation between the template and a portion of the image is calculated. This correlation is calculated by first subtracting the mean of the respective image is subtracted from them. Now they are multiplied and the sum of all the elements is taken. This sum is then normalized by dividing the sum by the the square root of the sum of the square of each normalized image.

Helper Functions

Bounding Box

The opencv rectangle function is used to create a bounding box on the image provided at the x and y location and the with and height of the rectangle provided. This is used to create the bounding box with the x and y position provided.

Crop Image This returns a cropped rectangular image of the image provided with the position x and y as the vertex and the height and width mentioned to create the rectangle. This function is used to extract the template from the picture.

Video From List This function uses cv2 where it takes in a list of images and then converts it to a mp4 video.

Result

The resulting videos will be attached with the pdf:

1. ssd
2. cc
3. ncc

Result Analysis

1. The ssd is a super simple implementation and this works very well. The implementation with a template chosen from the first picture works well and it is good with tracking the lady in the video when she is facing the camera. It is not good when she turns around as the black doesn't match the template and messes up the bounding box. It catches up as soon as the lady turns around. This was good for more than 250 frames and then corrects itself after it loses some tracking due to no image found when the lady turns. The overall frame number where this method works was ~300.
2. The cc video is the most noisy proces to do object tracking. As it is only the sum of the template with the image, It is very much depended on the video frame as some intensities will be very high and addition of them will maximize and find the position which might not match with the position of the object. This was good till ~20 frames.
3. The ncc method seems to work the best for object tracking. This subtraction of mean and normalizing the sum using the sqrt of sum of square of both images leads to a very good matching of the image. Ths subtraction of the mean of the image is a very vital step as I missed it the first time I ran it and could see similar inconsistency like the cc method. The normalizing is very good with maximizing the correlation. This method was my best and I can say that this worked well for abound ~350 frames and corrects itself very quickly.

Bonus

For bonus I thought about finding a way to change the template with accordance to the vide.

1. For my first try I tried to change the template every iteration to the boxed image found it last step. This was implemented with ssd.

This worked well till the lady turns around the x,y location then finds the black as the object detected which leads to having a weight of black in the template which moves the template to her hair. The video is called video_changing_temp_every_iteration

2. For my second try I implemented averaging out the template with the new frame found for every iteration. This increase the robustness as the average with the last frame allows the algorithm to get a better estimate in terms of where the object is. This has the similar issue as last implementation as this became as issue when the lady turns around in the video. This is a little robust than before. This was implemented with 'ncc'.This video is called video_change_ncc

3. For my final implementation I decided to do something similar and take a weighted average with the base template I created. As the base template will always involve the lady, It could be see that the algorithm is very robust to changes. Even when the lady moves around, it is able to detect the box specially while using 'ncc' and 'ssd' as the template which was created last frame has that hint of black which increases the chances of error for ssd being low there or correlation for ncc being high there. This method significantly improves the recognition, specially with ncc and ssd, it can be seen that object tracking is significantly improved by 100 frames for each method. The videos video_Test_ssd, video_Test_ncc and video_Test_cc are implementation of these method with all three methods. The cc method seems to be still very error prone but it can be seen that it tries to follow the person. The function is called weighted temp and some lines commented out in the code where I implemented it.

a. ssd

b. ncc

c. cc

This method can be made more robust by being able to calculate the weights of the template and the image at each step considering what will come next. This seems like a good implementation of learning algorithms and usually will be achieved by that.