Cairo University
Faculty of Engineering
Department of Computer Engineering

Tracking Module

Mosse Tracker

Report to explain the approach we took to make a tracking module using Mosse Tracker

Introduction

After detecting cars we need a way to assign to each car an id so we can see how the changes of the frames in a specific car so we can later enter these frames into a vif descriptor to detect accident.

We tried before another tracking algorithm but its main problem is the speed of tracking using normal computer hardware, so as we searched for another method for tracking and found Mosse a tracking method depends on a correlation filter and get real time performance.

Why “Minimum Output Sum of Squared Error (MOSSE)” filter?

Because Mosse runs using a correlation filter which depends on the frequency domain and using fast fourier transform you could get the frequency to the car image fast,

And because Mosse filter is a correlation filter so it can track complex objects through rotations, occlusions and other distractions,

And Mosse filter produces stable correlation filters when initialized using a single frame and it’s robust to variations in lighting, scale, pose and non-rigid deformations.

Mosse Tracker Algorithm

The target car is initially selected based on a small tracking window centered on the car in the first frame,

And from here, we start the first stage which is the preprocessing stage, first the pixel values are transformed using a log function which helps with the low contrast lighting situations then the pixel values are normalized then the image is multiplied by a cosine window which gradually reduces the pixel vaues near the edge to zero and this will put a more focus on the center of the cut image,

Then going to stage two which is making a filter trained and this will need training images so we will use current one and make some roations for the image then we will have trained filter,

from this point the tracking and filter training work together,

The tracked car is tracked by correlating the filter over a search window in next frame, the location corresponding to the maximum value on the correlation output indicates the new position of the target,

An online update is then performed based on the new location,
The correlation is computed in the fourier domain using fast fourier transform,

First compute Fourier transform to input car image : car_in_fourier = F(car_in spatial)
Second compute Fourier transform to the filter : filter_in_fourier = F(filter_in_spatial)
Third using the Convolution Theorem which states that correlation becomes an elementwise multiplication in the Fourier domain. Using the $\odot$ symbol to explicitly denote element-wise multiplication and $*$ to indicate the complex conjugate, correlation takes the form:-

$$G = F \odot H^*$$

The correlation output is transformed back into the spatial domain using the inverse fast fourier transform,

Then we get the max value around the center which will indicate the new position for te car, and using psr (peak to sidelobe ratio) to indicate the goodness of the tracking filter in the moment,

Peak to sidelobe ratio can be used to detect occlusions or tracking failier, to stop the online update and to reacquire the track if the object reappears with a similar appearance.

Implementation

Here are the steps that we implemented in the program for Tracking Module “Mosse Tracker”:-

Input: gray_frame,dimensions of the car in the frame, frame_width, frame height and
tracker_id: Sequence of frames.
Output : tacked car.

1. Extract the car image from whole frame.
2. Make a window and set its center to one.
3. Run gaussianblur on it with segma = 3 //according to paper segma = 2 but we found this is better in our problem.
4. Normalize the window to become [0,1].
5. Get the fast fourier transform to the window.

6. Make another two windows of the same size of first and assign all its values to zero as we will use it later to calculate the numenator and denominator in fig(1).
7. Make gaussian blur to cut image //not mentioned in paper but found it get better results in noisy images.
8. Start loop on num_of_training_imgs //advised in the paper to make 125 training numbers but found that is bad for our problem as our problem doesn't have lots of rotations so tried different numbers and found best is 25.
9. Make random rotations.
10. Make preprocess to image as paper stated and illustrated above.
11. Then get fourier transform to it.
12. Multiply G with F* and F with F*

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*}$$ fig(1)

13. Then train all the num_of_training images so later update the filter and calulate H*.
14. Now we start to update the tracking this is the step we will use each time we want to update tracking.
15. Extract the car frame , then make gaussian blur then make preprocessing stage then start to correlate this image with the filter.
16. Convert car image into fourier transform F.
17. Multiply F with H*

$$G = F \odot H^*$$ fig(2)

18. Convert the output G to spatial domain using inverse fast fourier transform as we stated that above.
19. Calculate the max value in the matrix and get its position // the max value means this is the highest correlation point between the new image and the filter.
20. The peak position is the new position for the tracked car.
21. Then calculate the psr according to the paper , get the mean and standard deviation.
22. After we calculate the psr we compare its result if it's lower than 7 then the tracker has lost the car.
23. If it's higher, then make an online training update to the new image using what we illustrated above from step 12 to 17 and update the old values with new by learning rate of 0.225 .
24. And so on every new frame goes from step 14 to 23 for 30 frames.
25. Then save all these 30 frames to disk or enter it directly to the vif descriptor we will make later.

Processing Time Takes

As we don’t take the whole frame to process then the processing time differes but on averge it can track in speed of 50 to 70 fps which is really good for tracking and we used multi-threading technique to save the tracked car.