Cairo University
Faculty of Engineering
Department of Computer Engineering

Crash Estimation Module

Report to explain the approach we took to make a crash estimation Module

Introduction

We found that to get better results from a ViF descriptor you need to filter out most of the vehicles as if they enter the ViF descriptor module the accuracy will be very bad because it gives lots of false alarms.

Why Crash Estimation module ?

We found that the paper[1] says that it's accuracy per vehicle is 80%, that tells us if you have one second and 10 vehicles, you will get two false alarms and that's too bad!

So, the idea is to limit the trackers entering the ViF descriptor according to the following algorithm.

We here consider a **new approach** by using gaming techniques to solve the ViF descriptor bad accuracy because of the variety of camera angles, resolutions, lighting and situations.

Crash Estimation Algorithm

First, we need to estimate vehicles speed in the video, we have already a tracker on every vehicle, so we can get the average speed of the vehicle by pixel unit, but camera angle is different from cctv camera to another and camera position height is different and its resolution too,

so, we need another unit instead of pixel unit to estimate the average speed of a tracker taking into consideration (resolution, height/scale and angle) of the camera,

To solve the camera resolution problem, we need to resize every input feed to fixed width and height (480,360),
To solve the height/scale problem, we multiply the average speed of a vehicle to a speed coefficient parameter. This parameter has an inverse relation to the area which means when the vehicle has a bigger area it has lower speed.

And the angle of the camera here plays a big role in our crash estimation module the best angle of the camera is the top view like figure(1), because we only care about x, y plane which the vehicle moves in it and we don't care about the z plane

Figure(1)

But the figure(1) is impossible unless you use a drone as a cctv camera and the possible perfect cctv angle is in figure(2), this will give us the depth we want.

Figure(2)

We found that the best way to calculate the average speed is by taking the average of every 10 frames and getting its angle and speed and you can expect the frame after 10 frames in the future.

So if the tracker is in its 11's frame then you can average 1 to 11 frames and get the 21's frame for the tracker in the future, and to expect the 15's frame in the future we need to average only the first 5 frames to the tracker.

Another thing to put in mind this will make a gap which is the first 15 frames of the tracker, and to solve this gap we need to make another detection for the tracker at the 15's frame and the next 45's frame and so on, so it will get the gap we are talking about.

Second, now we can determine if the vehicle speed is moving above the limit speed or not, so we put a threshold of 50 if above then the vehicle moves fast.

Now, you have the speed and the expectation about the future position of the vehicle

Third, compare every two vehicles with each other in different frames, Subsample the estimated future centers to 3 so you will use 15, 18, 21 frame and so on, if the two average speed are below the limit speed from its frame number to the last 10 frames then ignore them if one of them is above the limit then you need to check them.

Fourth, get the estimated centers in future frames for both vehicles then get the distance between them and we call this “distance between two estimated centers vehicles”, If the two estimated centers above some threshold we put it 40 then they will not crash because they are away from each other but if it’s below or equal the 40 then go to the next step.

Fifth, get the distance between the estimated center and the real center for each vehicle and get the max distance, that is because if the distance is so large then the vehicle center didn’t reach this position as expected and it may tell us an accident has occurred, so we make this equation (the max distance between actual and estimated center for each vehicle / the distance between the two estimated center in the future) if it’s above 0.5 then it may be a crash go to next step.

Sixth, here we filter out too small tracker size so the ViF descriptor could recognize it correctly.

Implementation

Here are the steps that we implemented in the program for Crash Estimation Module:-

Input: 30 frames, trackers
Output : frames to ViF descriptor module

1. Record history for every tracker center.
2. Calculate the average speed for each vehicle for the last 10 frames.
3. Multiply the average speed to speed coefficient as mentioned above.
4. Estimate the future position to the tracker center after the next 10 frames.
5. Average only the first 5 frames to get the estimated future center at 15 frames.
6. Now you have an estimation for 15 to 30.
7. To fill the gap from 1 to 15 you need to make another detection for vehicles after 15 of each detection so you can do the same from 1 to 6 and fill the first 1 to 15 frame gap.
8. Subsample the estimated future centers to 3 so you will use 15, 18, 21 and so on and do the next steps.
9. Compare every two trackers with each other if both are below the limit for the last 10 frames then ignore them.
10. For above the limit you will get the distance between the two estimated future centers for both vehicles if they are above 40 distance then they are away from each other and will not make an accident.
11. If below 40 then they are close to each other you need to get the maximum distance between estimated center and real center at the same frame for both vehicles.
12. If (the max distance between actual and estimated center for each vehicle / the distance between the two estimated centers in the future) > 0.5 then it may be an accident.
13. You need to filter all the small sizes of trackers if the width below 50 and height below 30 and the ratio between height/width below 0.3 .
14. Send to the ViF descriptor to say if the tracker is an accident or not.

References

[1] T. Hassner, Y. Itcher, and O. Kliper-Gross, “Violent flows: Real-time detection of violent crowd behavior,” in Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, June 2012, pp. 1–6.