Combined Kalman Filter and Siamese Network for Real-Time Visual Tracking

Liming Gao, Chengyu Hu, Zekai Liu, Mingzhao Yu

The core problem we try to solve is that, when we're using Siamese Network for real-time visual tracking, the tracker lost the object easily if the object is temporarily blocked by other things.

To solve the problem, we decided to use a combination of Kalman Filter and Siamese to track the object, and Kalman Filter significantly improves the performance while the object is blocked. However, in our practice training of the program, we found out that we can also let our program distinguish two similar objects from each other. The dataset we use is ILSVRC 2015, and the typical example is the video of two similar planes flying horizontally.

We assume three situations which may strongly disrupt Siamese's prediction:

1. Simple occlusion. When the object is simply blocked by something else, the Kalman filter can easily solve the problem. As we showed on our last project PowerPoint, Kalman filter can predict the movement by previous frames. We embedded a Kalman filter into the original Siamese Network tracking code, checking whether the

maximal score in the correlation map is lower than 0.3. The maximal score represents the similarity between the tracking target and the template. If score<=0.3, we can assume that the object is occluded, which means that the measurement values are not available; Under this situation, Kalman filter assign all the weight to the model prediction value, i.e. the target position is only the output of prediction.

For the Kalman filter, we choose target position and velocity (px, py, vx, vy) as the state variables of the transition model, and position (px, py) are the measurement variables. We adapted a constant velocity motion model in the Kalman filter. We choose a constant velocity model and measurement noise. Although we tried more variations of Kalman filter, their performances are not better than the ones with a constant velocity model and constant noise. We first tried to take the acceleration as the model transition noise. However, the tracking accuracy decay. The reason may be that the target we tracked moves with very small acceleration. We also tried the constant turn rate and velocity model, which is a nonlinear motion model widely used in vehicle tracking. The nonlinearity requires the extended Kalman filter, resulting in more complicated parameter tuning and calculation. But unfortunately, its prediction performances are still not as good as the simple constant velocity

motion model.

2. Interference. Since our tracking is following the part of the image which is similar to the template, it may be distracted by other objects which are also similar to the template.

   Since the object is moving and the film is taken from one direction, sometimes the false object may be more similar to the template. For example, both of the object and the template are all heading left at the beginning, but the tracking object changes the angle a little bit then on the score map the maximal score of the tracking object may be lower than the false one, making the original Siamese program to change the object. In this situation, the location of the "tracking object" will have a relatively great difference to the previous ones. Since our object isn't an alien, it can't just suddenly flash to another place within two successive frames. As a result, we set a rule that, comparing the location of successive frames, whenever the difference is greater than 100, then the tracked target is a false one, we need to block the false target position. And choose the other as the true target.

3. The third condition can be recognized as a combination of problem 1 and problem 2.

   If our tracking object is occluded, the tracker is still going to find a template-similar object in the frame, and it may go for the false

object instead. Once the measured position difference between two successive frames is greater than 100, then block the jump target, and got a score map <=0.3. Then use the Kalman filter to predict the occlusion target position.

Basically, all the problems we popped up above are solved in the code we submitted, and the result is shown on the video in the PowerPoint. However, we still have several directions that we hope to improve in the future. First of all, the performance isn't as fast as we think, it might be improved after more training datasets. Secondly, the prediction is almost perfect for constant speed objects, but we want to adjust it so that it can well predict an accelerated object. Last but not least, it's still a simple target prediction; if we can make a multi-object tracker that will be satisfying.