MultiModalityTracking

May 12, 2016

1 Models

Dynamic model: simple gaussian distribution model. $P((x_t, y_t)|(x_{t-1}, y_{t-1}))$. x, y is the left up corner coordinate of each detected window.

Appearance model: store the SVM descriptor vector physically, use hog.detect() OPENCV API to retrieve the detection score, which is the distance from the hyperplane. For the detector, we use the same API, and detected as a pedestrian if the distance is larger than a threshold(2.8 in our experience).So here for the appearance model, the score is $\sqrt{(MAX(0, weights - 2.8))}$.

Proposal distribution: Here we do not consider the newly detected pedestrian in current t frame, we only consider the pedestrians we initialized in the first frame. the proposal distribution is

$$q(x_t|x_{t-1}, y_t) = \alpha L(x_t|y_t) + (1 - \alpha)D(x_t|x_{t-1})$$

L is the appearance model, D is the dynamic model.

2 Process

1. Initialization: for the first frame, we use the detector to give the m pedestrians(removed the false positive). We build a mixture particle filter with 10 components, each of which has 50 particles. we assign this 500 particles around the m detected pedestrians.

2. Apply the transition model, to mimic pedestrian moving.

3.Weight Updating

- Update the weight of each particle:

$$W_t = \frac{W_{t-1} * L(x_t|y_t) * D(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)} \tag{1}$$

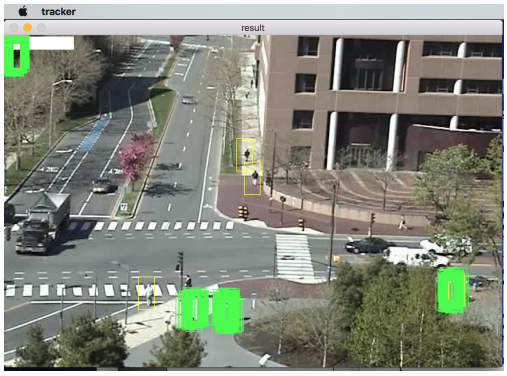
- Groupby the particles by their ID(mixture component id) and normalize the weight inside each mixture component.

- Update the weight of the mixture components:

$$\pi_{m,t} = \frac{\pi_{m,t-1} * w_{m,t}}{\sum_{n=1}^{M} * \pi_{n,t-1} * w_{n,t-1}} \tag{2}$$

!ht

(a) 1 st frame



(b) 10th frame

Figure 1

4.Resample: Each particle has a field in proportion to the multiply value between weight of the particle and the weight of the corresponding mixture component. randomly generating numbers between $(0, 1)$ and resample the particles. We try to maintain an array of particles with variables like x,y coordinate, weight and corresponding mixture component id. Here for each mixture component, if the number of particles belong to this component is bigger than a threshold(50), we first sort these particles by weight and then remove the particles with lower weight(these particles points to null and belongs to no group). After removing, we reassign the weight of all the "alive" particles with the same weight.

Hopefully, by removing particles of high weighted mixture components, we could decrease the speed that all particles converge to one object. Meanwhile, we could assign these "removed(died)" particles to new coming objects.

3 Algorithm

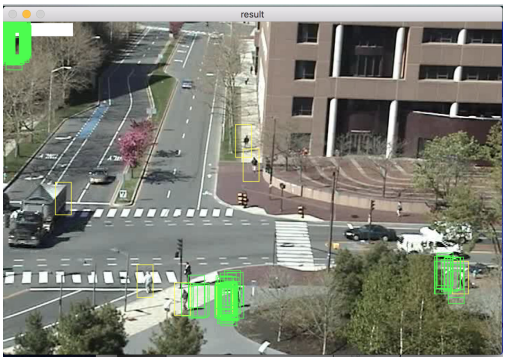
Algorithm 1 `multi modality tracking`

Require: adaboost detector's m detected rectangle for 1st frame.

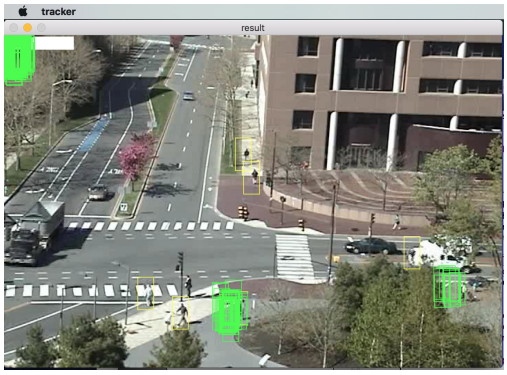
1: assign the $totalParticles$ around the m objects.
2: **for** VideoCapture **do**
3: move the particles: $x_t = x_{t-1} + 2 * gslrangaussian(rng, 1)$;
4: retrive the weight of each particle:$W_t = \frac{W_{t-1} * L(x_t|y_t) * D(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}$
5: update the weight of each mixture component
6: resample the particles
7: **if** mixture i 's particles number $\geq threshold$.
8: sort the particles and remove the ones with lower weight.
9: **end if**
10: assign equal weight to all alive particles. assign 0 weight to removed particles.
11: display particles distribution.
12: **end for**

4 Rough result

As shown in the above sequences, as time going on, we remove more and more particles, and after 20 frames, 1 target object disappeared totally. Here it's quite difficult to keep the weight balance for the mixture components. Since each component(detetcted window)'s ground truth score is different.(Some of the targets



(a) 1 st frame



(b) 10th frame

Figure 2

have higher score to be detected as a pedestrian, which means they are easy to distinguish. For these obvious terget, it's normal that the mixture components represent this target will have higher weight). If there is no balance, there's no surprise that all the particles finally converge to the most obvious(or easily detected) target after some time.

5 Problems

- we should make use of the adaboost detector's result. We could roughly believe the newly added targets given by the detector are all new coming targets, we could assign the "removed" particles(particles at the left corner) in a rational way.But the problem appears since there are false positives in each frame. Like in frame t there are m_1 true positive and m_2 false positive, then by tracking, in frame $t+1$, we have m_1 true positive, m_2 false positives are removed. But meanwhile, by using the adaboost detector, we add newcoming objects as well as false positives generated by the detector at time $t+1$.

- For the multiple objects converge problem, the reason is explained in the result part. I am not sure if we add "removed particles" back to the frame based on the result of the detector should help or not. Like at time t, true positive object m has only 1 particle, at time $t+1$, there's no particle from the tracking side, but if the adaboost detector detect object m, we could assign the "removed particle" back and continue the tracking of object m. Currently, I do not catch up an idea of assigning the "removed" particles back in a right way.