Tracking multiple pedestrians using boosted particle filter

May 24, 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 $MIN(1, (MAX(0, weights - 2.8)))$.

Proposal distribution: proposal distribution is an estimation of the real distribution and we sample based on this distribution.

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

Here Q_{ada} is a Gaussian distribution centered in the Adaboost detection with a fixed variance. If we set $= 0$, it reduces to the bootstrap filter, if we set $\alpha = 1$ it becomes a boosted filter. Here by increasing the α value, we place more importance or trust on our detector.

2 Process

1.**Add mixture components**
For all the detection windows given by the detector at frame t, we compare with the frame $t - 1$ tracker output. The threshold of neighbor is a settled pixel distance between the centre of detected windows. If there is no neighbor, we add a new mixture component for this newly detected pedestrian with a weight. This initial weight should based on our detector score, higher weight means more likely to be a pedestrian. If there's a neighbor founded at frame $t - 1$, for the founded window or mixture component, we record the Adaboost detection result to each particle belongs to that mixture component.

2.**Updating**
For each mixture component, we do the following updating process independently.
2.1 Transmit all the particles of current mixture component based on the dynamic model.
2.2 Update each particle's weight.

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

2.3 Resampling
Resample the particles according to their weight. We propose the systematic resampling algorithm(stochastic universal sampling.) After sampling we set the weight of each particle equally.

3.**Global weight updating**
For each mixture component, it has a weight parameter which is a compare between the already existing

mixture components. A mixture with relative higher weight means it's more likely to be a pedestrian. The weight of all the mixture components sum up to 1.

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

4.**Remove mixture component**
For all the existing mixture component, if the weight is lower than a threshold or the score of the detector is lower than a threshold, we remove the mixture component.

5.**Display tracker result**

3 Parameters setting

- **neighbor threshold**: distance threshold for finding a neighbor, currently setting is 8 pixel value.
- **weight of newly added mixture**: this parameter should reflects the likely hood of a pedestrian here. It based on the detection score. Meanwhile this weight should keep the equilibrium between the newly added mixture components and the already existing mixture components. If it's too high, it "steal" weight from the already existing mixture component, which may make us removing already existing true pedestrians. If it's too low, then it's quite difficult to add new mixture components, which means we always lose the new coming pedestrians. Higher weight setting means trust our detector more. Currently it's set as $\frac{detectorscore}{\sharp components}$
- **dynamic model moving variance**: $p(x_t|x_{t-1})$ follows a Gaussian distribution with mean 0 variance unknown both in x and y direction.
- $Q_{ada}(x_t|y_t)$ **variance**: part of the proposal distribution. here is a Gaussian distribution with mean in the detected window position and variance unknown.
- α **parameter for proposal distribution**: between $(0, 1)$. Higher value means more importance on the detector.
- **remove mixture components weight threshold**: the weight of each mixture component depends on the number of existing mixture components. Is it better if we remove mixtures by percentage?
- **remove mixture components detector score threshold**: for the newly detected result we use our detector again.

4 Opening questions

- When finding neighbors it's more reasonable to match between current detector result with the updated mixture components. But without this matching we do not know how to add new particles and could not continue the updating process for each mixture component.
- When adding mixture component, for the already existing mixture, if we could not find a neighbor in the detector, how could we set the $Q_{ada}(x_t|y_t)$ in the proposal distribution?
- Is there scenario we need a merge for the mixture components?
- Setting the above parameters rationally or if there is a way to remove some parameter?