**Project2 Due by May 16th, 2017**

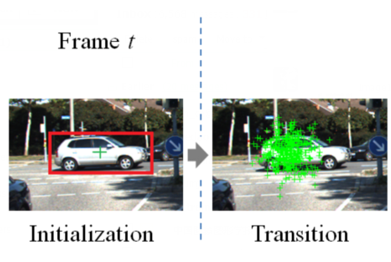
1. **A concrete introduction to particle filter for tracking**

You should understand the following key points when working on this project:

1. Particle filter uses “particles” to represent the probability on what the current state should be, or precisely, . That is, given all observed values from time 1 to t, what’s your state should be. In tracking, a particle is a rectangle indicating the location of object in image. If there are many particles in a region, then it means the object you want to track is in that region with high probability.
2. In this project, each particle has 4-dimentional state. They are (cx,cy,sx,sy). (cx,cy) is the center of rectangle, (sx, sy) are scales of x axis and y axis compared with a specific size. The reason that you need to predict (sx, sy) is that the object in image may become larger or smaller, so your rectangle should also be scaled up or down.

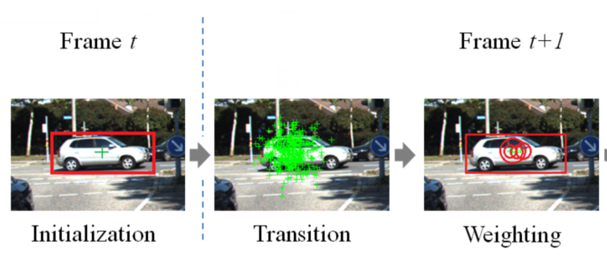


1. There are mainly two steps in particle filter, i.e., the “predict” step and the “update” step. In “predict” step, or “Transition” step, you only **blindly** predict where each particles will be in next frame. Here, we need a transition model  and prediction results are sampled from this distribution. We choose a Gaussian transition model, so . Note that there are 4 values in X (cx,cy,sx,sy), so there are actually 4 standard deviations.



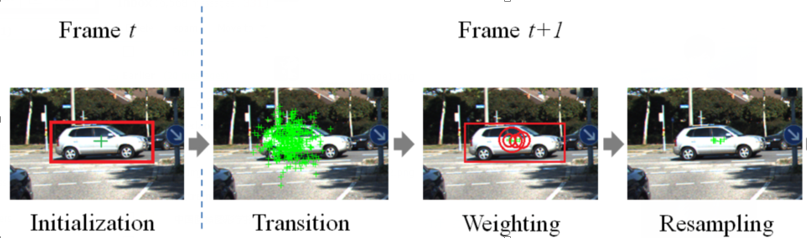
In “Initialization” figure, all particles are the same rectangle because t=0. Then, with a Gaussian transition model, particles spread out around the initial rectangle. Note, in “Transition” figure, we only show the centers of particles (which are rectangles actually) for clearness.

1. In “Update” step, we get a new observation, an image. So for each particle, we get the image content in that rectangle. Now particles are not blind any more. For each particle, we compute a confidence that it indicates the tracking object, which is called “Weighting” step. There are **different** ways to compute the confidence. In this project, we simply use the similarity between the image content in particle and the previous tracking location. For example, the similarity between the image content in particle in “Transition” figure and the red rectangle in “Initialization” figure. The reason is that, if a particle is not consistent with the previous tracking result, then it is not trusty.



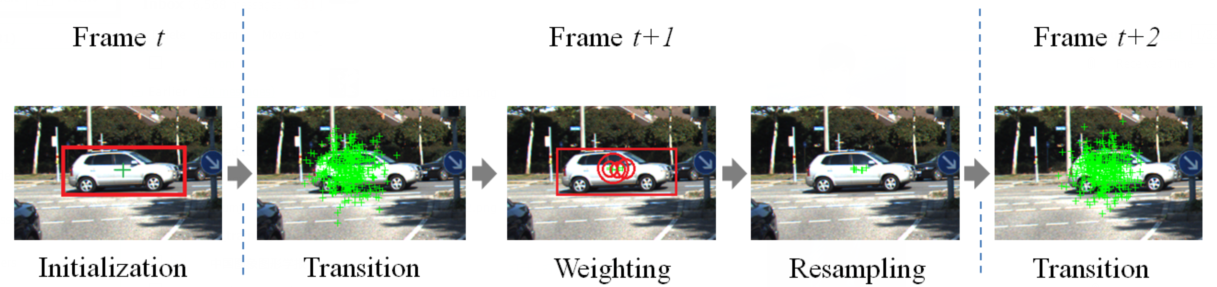
You see that, in “Weighting” figure, only a small number of particles has large weight (larger circles mean larger weights and small circles are passed). Weights may also normalized to sum to 1.

1. In the “Weighting” step, you output the particle with highest weight to be the tracking rectangle.
2. Now we get particles and weights for each of them. Recall that we use the number of particles in a region to represent the probability that the object is in that region, which means that each particle should have the same importance. (If you are confused, think about that if particles are weighted, it should be the sum of weights, not the number of particles to represent ). So we perform a “Resampling” step. In this step, we sample new particles according to the weights of old particles, i.e., around an old particle with high weight, you put many new particles. As a result, it converts the weights of old particles to the number of new particles. Technically, we can fix the number of particles to be N, for an old particle  with weight ，we simply sample particles around  by a Gaussian distribution with small standard deviations, note weight  is normalized.



You see, after Resampling, the messy particles in “Transition” figure converge to a small region.

1. Now, go to “Transition” step and move to the next frame.



1. **Implementation**

Unzip the file and you get file structure like this:

----------src/

----------data/

----------------car/

----------------David2/

----------project1-instructions.doc

In src folder, runTracker.m is the main file. You need to implement all “empty” functions in this folder. Read all comments to make sure you understand variables, functions and code structure. After you finish them, run runTracker.m and you will see a figure plot the tracking result.

In the 7-th line of runTracker.m, you can choose two datasets. ‘car’ dataset is easier, you should successfully track the car for at least 50 frames. ‘David’ dataset is more challenging, you may miss the target.

The result is stored in the directory indicated by “save\_dir” variable in runTracker.m.

1. **Improvement**

The tracker you implement is just a very basic version of particle filter. Several improvements could be made. First, in the weighting step, we compute weights by a comparison with the last tracked result. If the last tracked result is not perfect, then the tracker will break. A more robust method is to maintain a set of templates of target, and weights are computed from the comparison with templates. While our weak strategy is to use the last tracked result as template.

Second, you may try other features, like HOG, LBP, which are widely used in tracking. But this only works if the weighting step is optimized already.

1. **Submit**

TBD.