Assignment 3: Particle Filtering

Jia Ming Liang
November 19, 2014

1 LOG ODDS

After going through steps 1 and 2 in README (training positive and negative data with mouse click carefully) we obtain the log likelihood ratio

$$L(z) = log\left(\frac{P(z|target\ present, \vec{a})}{P(z|target\ absent, \vec{a})}\right) \tag{1.1}$$

To fit the plot by eyeballing, we need to choose some important characteristics that our logistic model taken account for. Taking a look at the plot log(P-on/P-off) (i.e. Figure 1.1 right). First we must choose to fit our model with the same zero crossing value (in the code we use the variable 'mu'). This value is important to seperate the good and bad scores. The right side of the zero-crossing indicates the positive scores and the left side is the negative scores. We also want to choose the correct inclinations to fit the graph where it is inclining upward to the right (variable 'lambda' in our code). Also we use the variable 'scl' in the code to set the scale so that the top of the fitted curve is around the same height as the peak in the logistic distribution.

Basically we want to fit the the portion of the graph so that it covers both the P-on and P-off region (i.e. Figure 1.1 Right before 0.9 in the horizontal axis). We do that because this region describes the difference between positive and negative particles. This will allow us to track our target. Based on the graphs P-on vs P-off, we can tell the target is easier to track if the two distributions have smaller overlap. Generally if target texture has more unique features, the two

curves would have smaller overlap. In Figure 1.4 left, we see the P-on vs P-off distribution for the red pigeon head has a large overlap in these two distributions. This is because the head of the red pigeon has very little features which make the head harder to track.

Below are the fitLogOdds.m plots:

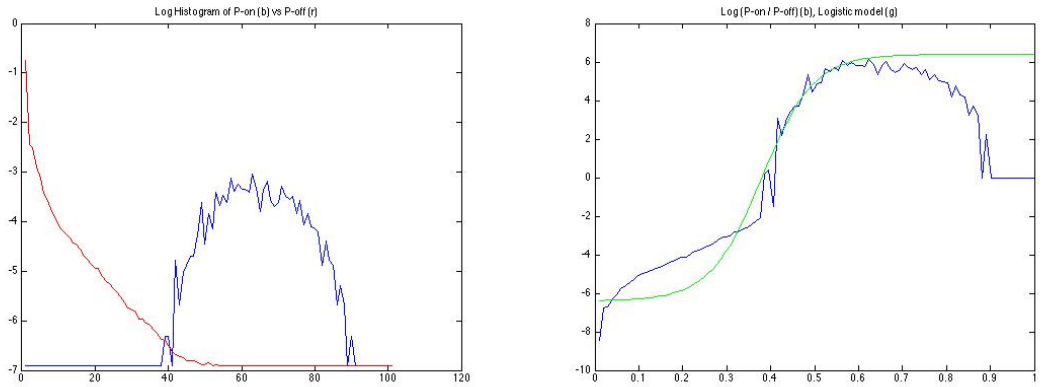

Figure 1.1: Left: P-on Vs P-off plot, Right: Log(P-on/Poff) plot for grey pigeon back

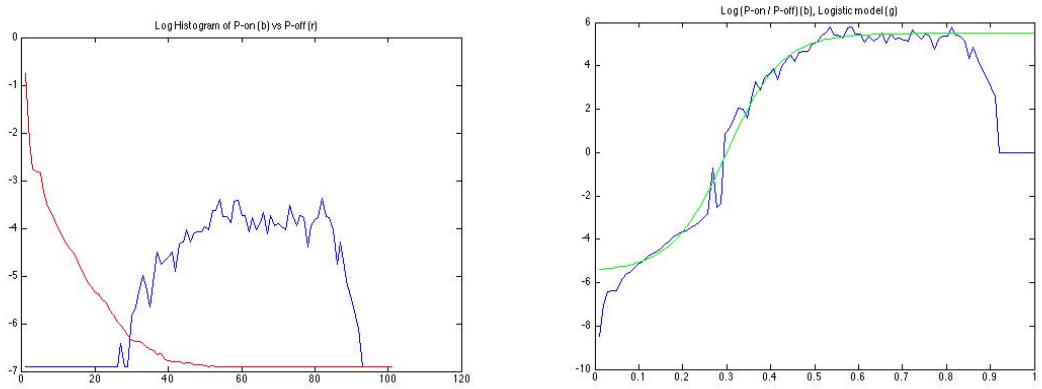

Figure 1.2: Left: P-on Vs P-off plot, Right: Log(P-on/Poff) plot for grey pigeon head

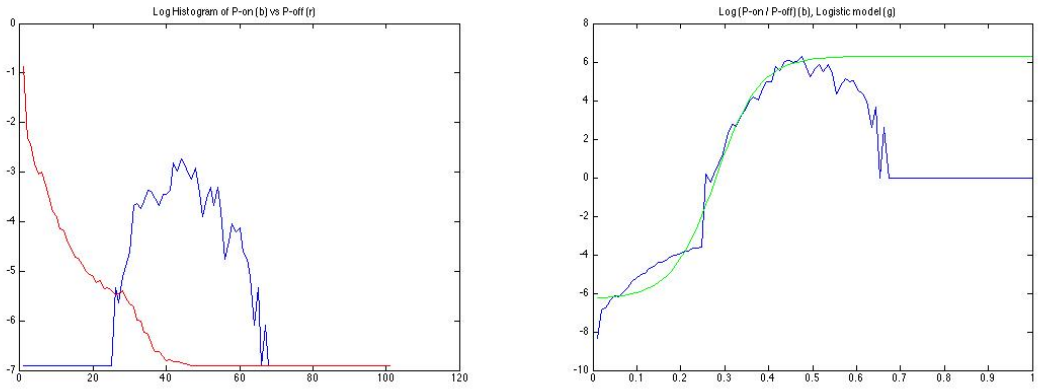

Figure 1.3: Left: P-on Vs P-off plot, Right: Log(P-on/Poff) plot for red pigeon back

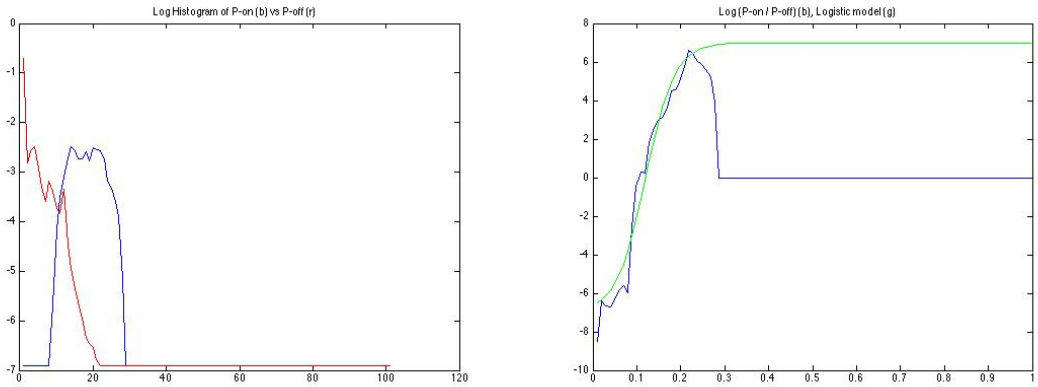

Figure 1.4: Left: P-on Vs P-off plot, Right: Log(P-on/Poff) plot for red pigeon head

Here is one example running tryHOG.m we get the following results:

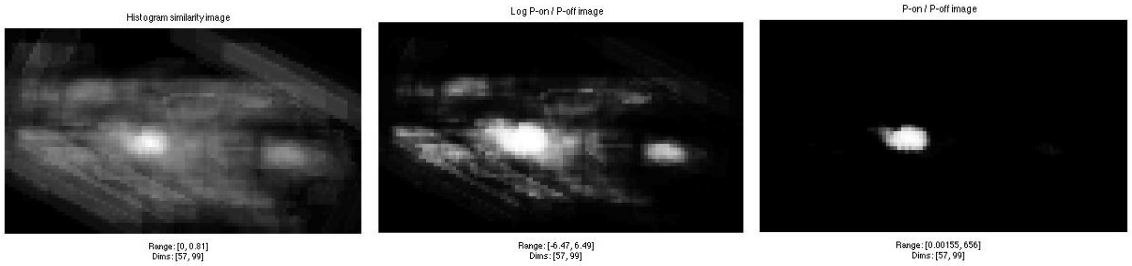

Figure 1.5: tryHOG.m plot for grey pigeon back

2 PARTICLE FILTER

(a) Model of dynamics

Our dynamics model is a zero velocity random walk model given by: $\vec{a}_t = \vec{a}_{t-1} + \vec{n}_t$. We know $\vec{n}_t$ is a mean zero normally distributed random variable with covariance C_D, then $\vec{a}_t$ must be must be a normally distributed random variable with same covariance and the mean $\vec{a}_{t-1}$. Hence, in our implementation for propagateSamples.m we simply do something like the following for every single samples:

newSample = normrnd(previousSample, variance)

Where normrnd is a function that generate a new sample point given the mean and variance. Note that this is only example pseudo code, please see propagateSamples.m for my actual implementaiton.

(b) Reweight particle set using likelihoods

Following step 3 in the lecture notes we use Bayes rule: $\frac{1}{c} = \sum p(z_t|a_t)$ and the new weight is simply $newWeights = c \cdot p(z_t|a_t)$. To get the number of effective samples, we use the formula in the lecture note p.30: $N = \frac{1}{\sum newWeights^2}$. Please see updateWeights.m for my actual implementation.

(c) Go!

To set the covariances for the dynamics, we want to keep the covariances as small as possible to get most effective samples, but if the covariance is too small it will lose track of the target when the target move too fast. In general if the target is moving faster we will need a larger covariance so that it covers the change in the position and orientation from frame to frame. However, if the covariance is too large we might end up tracking other points which are not our target because of noise. Note that the pigeon head will require a larger covariance for σ_x, σ_y and θ than the back because the head has a larger motion (moving back and forth and turning). After some tests my covariances for the 4 targets are:

Grey pigeon back:

dynamics.dynCovar = diag([5∧2 5∧2 (15/180*pi)∧2])

Grey pigeon head:

dynamics.dynCovar = diag([7∧2 7∧2 (40/180*pi)∧2])

Red pigeon back:

dynamics.dynCovar = diag([5∧2 5∧2 (35/180*pi)∧2])

Red pigeon head:

dynamics.dynCovar = diag([7∧2 7∧2 (40/180*pi)∧2])

Note that I can set larger covariance for the grey pigeon and still get good tracking but not for the red pigeon. This is because the red pigeon has less features to track which is very volunerable to noise, so larger covariance for red pigeon has higher chance in tracking noise instead of the real target. However smaller covariance will be volunerable to fast moving target. Fortunately both grey and red pigeons are not moving very fast.

The following figures are obtained by runing showResults.m:

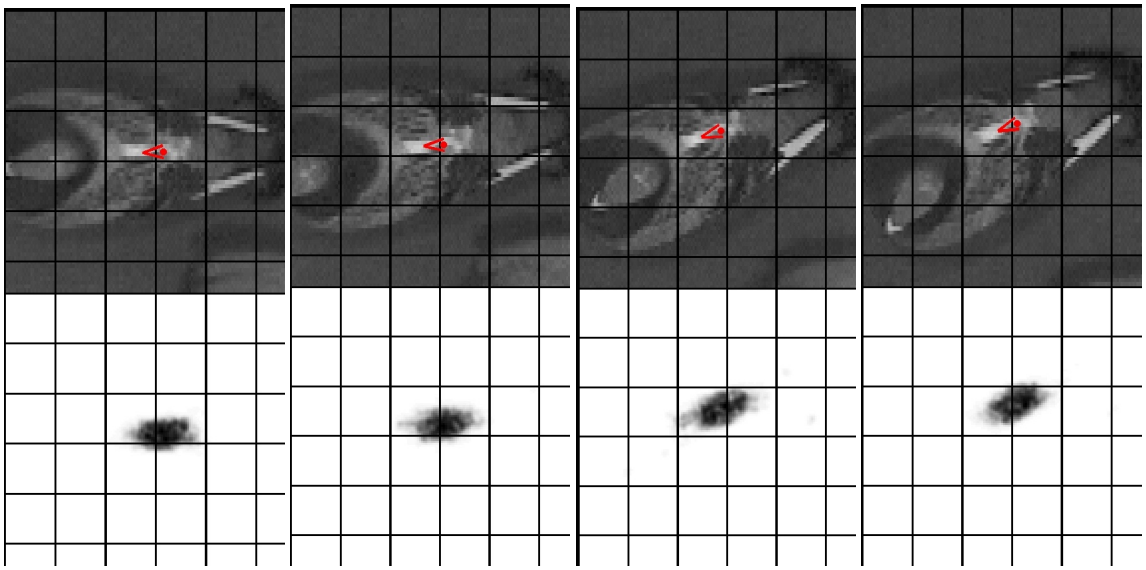

Figure 2.1: Frames 150, 158, 172, 180 (from left to right) for grey pigeon back tracking

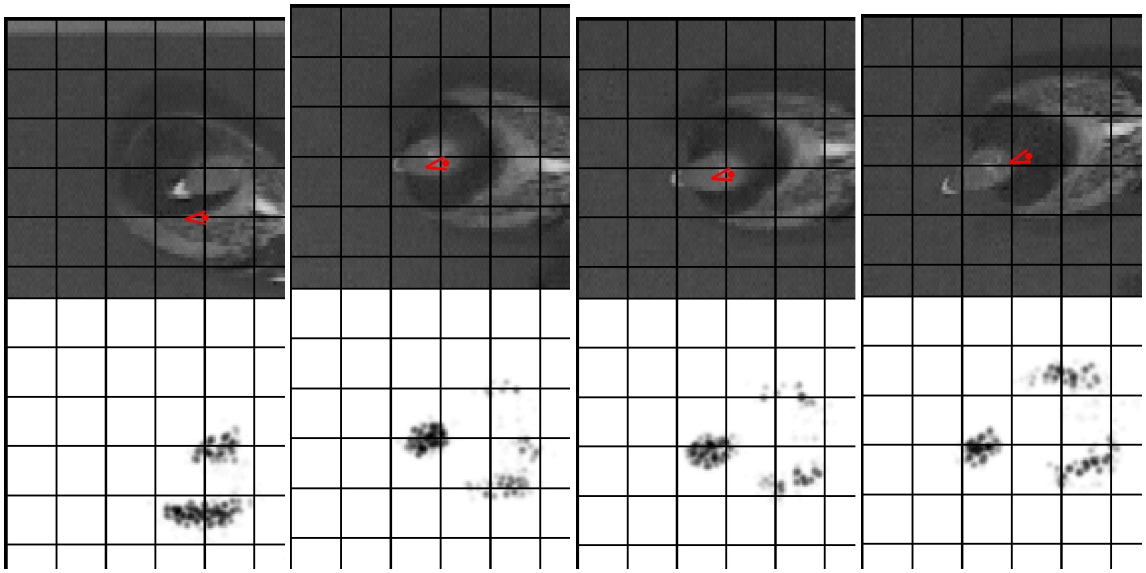

Figure 2.2: Frames 42, 126, 144, 160 (from left to right) for grey pigeon head tracking

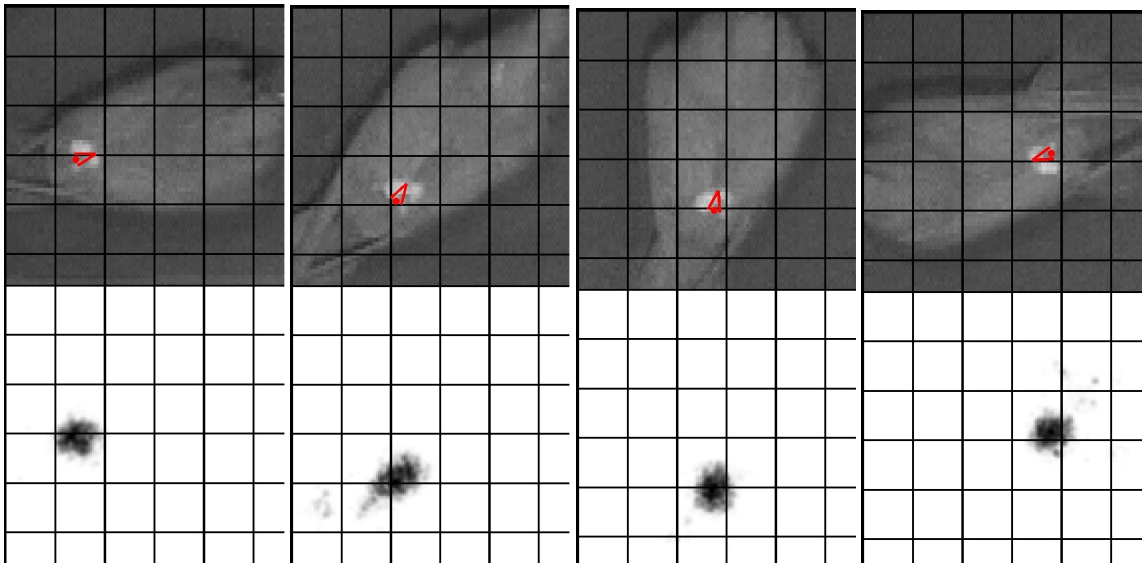

Figure 2.3: Frames 60, 84, 106, 176 (from left to right) for red pigeon back tracking

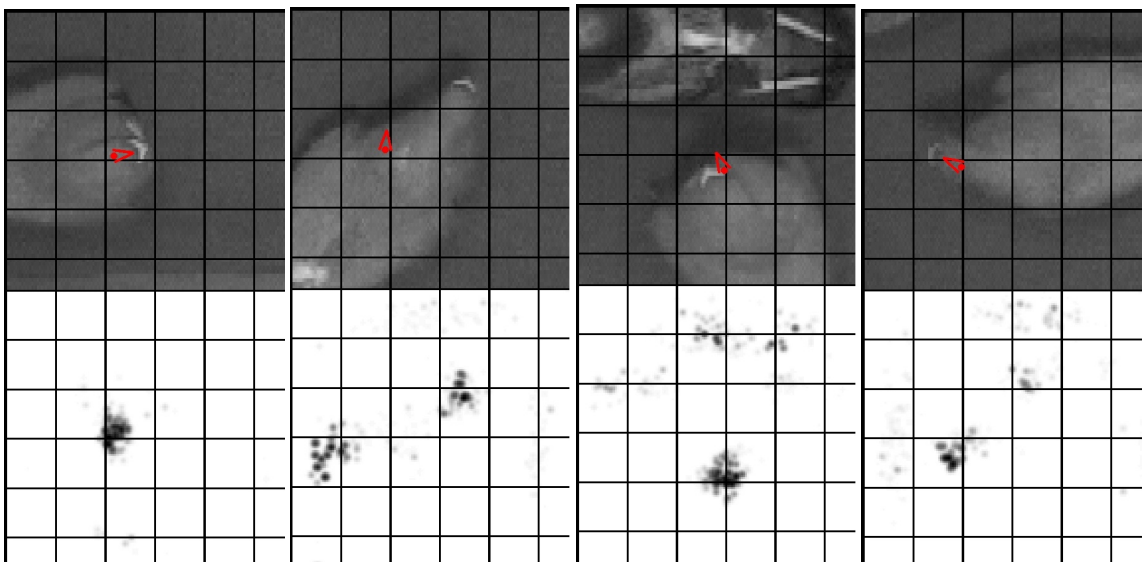

Figure 2.4: Frames 50, 84, 126, 178 (from left to right) for red pigeon head tracking

(d) Evaluation of results

Performance: The results looks decent for single target tracking. We can see from the results in the previous section that the back of the pigeon is on track most of the time and the particles form a single cluster. However, both grey and red pigeon head tracking fail sometimes because the head has a stronger movement and we can see from the clusters that the particles are more spread out and even form multiple clusters (due similarity in the scores of the nearby texture).

Expectation the algorithm will work: To get particle filter to work well we need the nearby texture of the target to be distinctive to prevent neighbour getting similar score. To do that we need our trainning data to have P-on vs P-off plot to have as small overlap as possible. We would also need the target to have a relatively small movement speed because small covariance would lose track of the target if it moves to fast but large covariance will be volunerable to noise.

Strengths: Particle filter is a good tracking method for probability distribution of some high dimensional and highly non-linear models.

Weaknesses: One problem of particle filter is that it takes up a lot of computing power. Reducing the number of particles will result in reducing accuracy. Also, our model depend on the previous state; hence, if the target move too fast and we lose track in one of the states we will end up losing the target forever. Also, if image have blur due to fast movement speed, we will end up having a lot of noise and lose track of the target.

Failure modes: If the target surrounding texture don't have enough features to distinct itself from the neighbour, or it is blurred out due to movement we will end up tracking the wrong target. If the target move too fast and our covariance is not large enough to cover the movement we will lose track of our target forever. If our covariance is too large we will end up tracking wrong target sometimes.

3 TRACKING TWO MODES

Here is the tracking two modes results for both back and head:

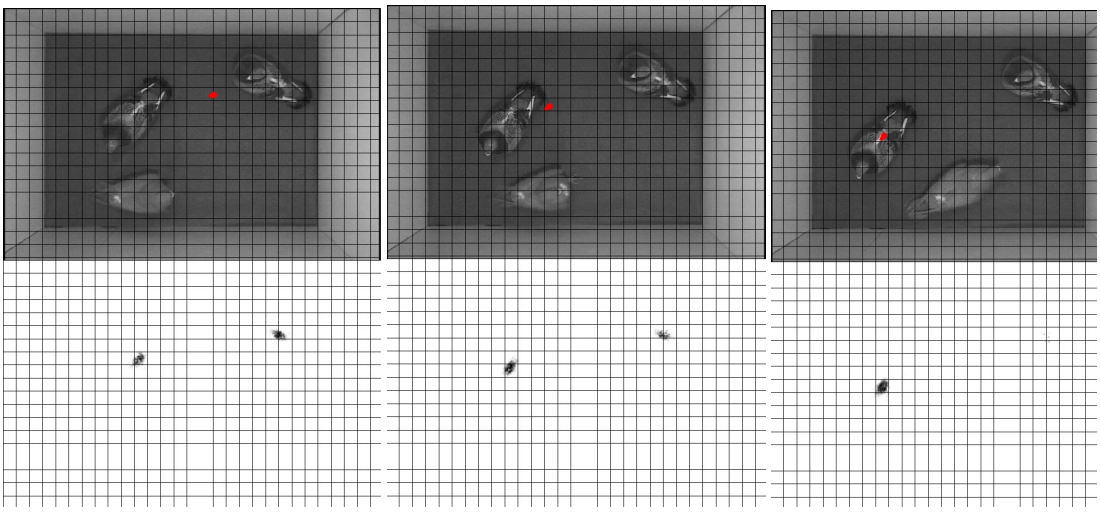

Figure 3.1: Frames 46, 56, 70 (from left to right) for two modes tracking (pigeon back)

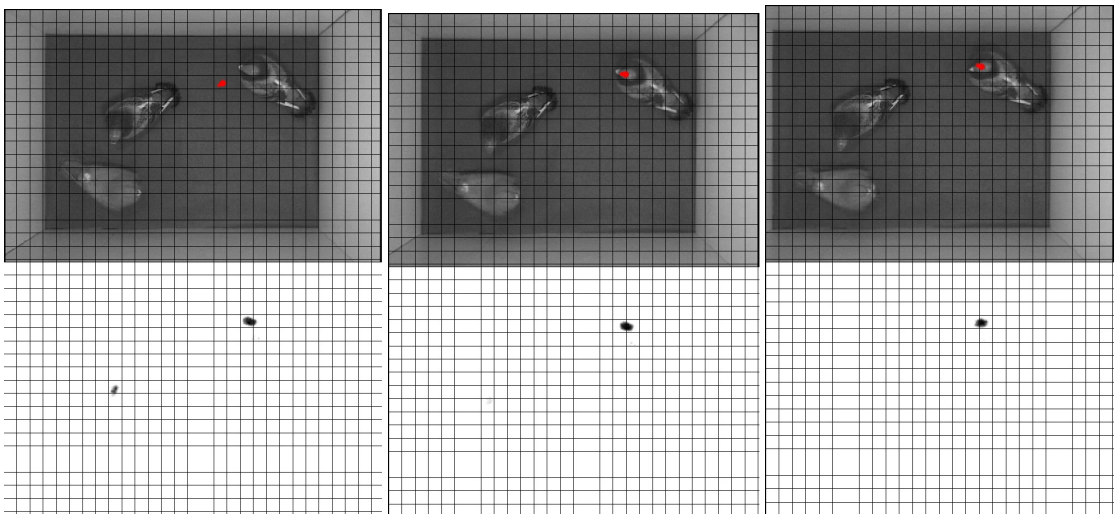

Figure 3.2: Frames 30, 32, 34 (from left to right) for two modes tracking (pigeon head)

The algorithm works in the beginning when both pigeons aren't moving. However, we can see from the results above that once one of the pigeons starts moving the two clusters quickly

converge to one. If one pigeon is doing different movement we introduce unbalance in the similarity score which our weight function will bias toward one; as a results, the two solution converge into one. I also notice that the number of effective sample is decrease dramatically when we track two targets. In the single tracking mode, the number of effective sample for the back of the grey pigeon is ~1500 but when we track two it becomes ~40. Another failure mode is due to our initial guess for the mean and covariance. If we were to track two target we would need two different initial guesses, otherwise the weight will bias toward one of the cloned pigeons and quickly converge to one solution. We can see this from tracking the grey pigeon heads. The clusters quickly converged after the first few frames.

To fix the above issue, I have come up with a solution to divide the particles into two clusters before we weight the samples. In matlab, I used the kmean function (k-mean clustering algorithm) to divide the samples into two data sets then reweight the two data sets individually so that the two clusters have equal sum of weight. This will prevent the two solutions to converge. Please see updateWeights2.m for my implementations.

Finally, with this change, the particle filter worked just like single mode but tracking two targets now. Please see the results below:

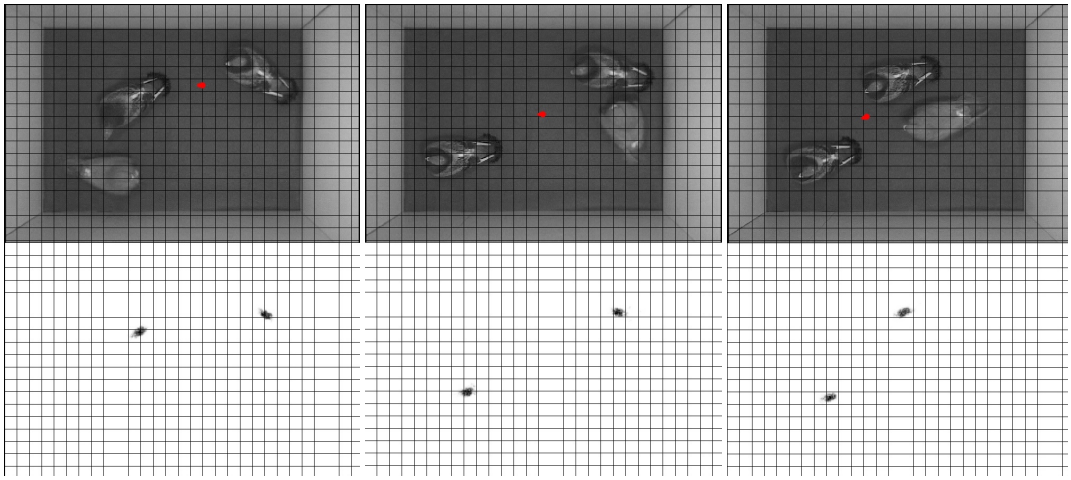

Figure 3.3: Frames 34, 124, 168 (from left to right) for two modes tracking using clustering (pigeon back)

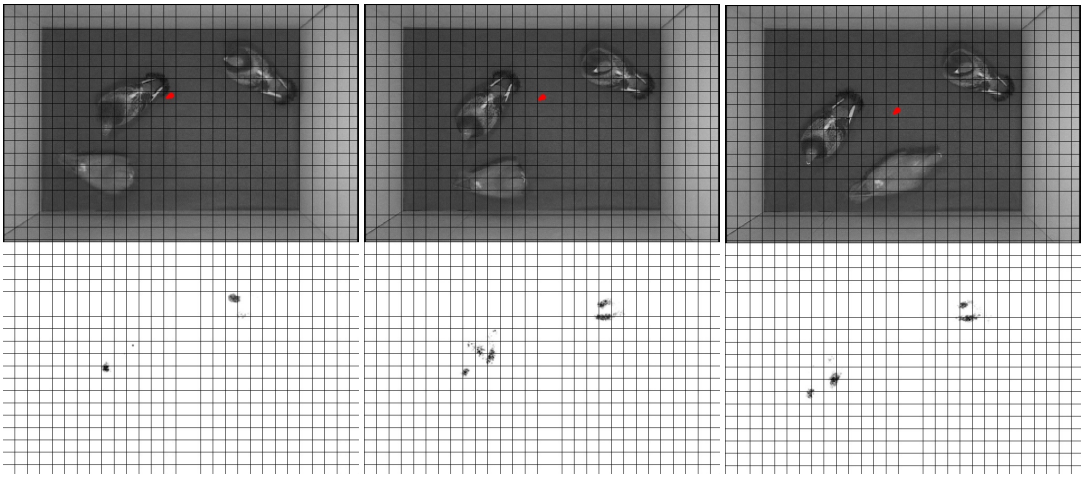

Figure 3.4: Frames 32, 50, 68 (from left to right) for two modes tracking using clustering (pigeon head)

NOTE: pigeon head two mode tracking seems more noisy I believe this is because our trainning data comes from the right pigeon not the cloned one. And the right pigeon had the same behaviour when we run single mode tracking. If the head has more distinctive features this issue will be elliminated.