Probabilistic Robotics Hw3

Tahsincan Köse

12 December 2018

1

a

Markov assumption suggests that the present state depends only on the past state and control input. In Temporal-Difference algorithms, this projects onto having $\lambda = 1$. In other terms, a proper Markov assumption has unmodeled state variables that are close-to-random effects.

In this case we can specify $u_t = \ddot{x}_t$ and $x_t = [x_t \ \dot{x}_t]$, which is semantically equivalent to feeding control input as acceleration in discrete time steps that affects system states -*position and velocity*- in return.

b

True motion simply refers to a deterministic setting. It is a linear function of the past state and control input as in the form:

$$\begin{bmatrix} x' \\ \dot{x}' \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 1/2 * (\Delta t)^2 \\ \Delta t \end{bmatrix} \ddot{x}$$

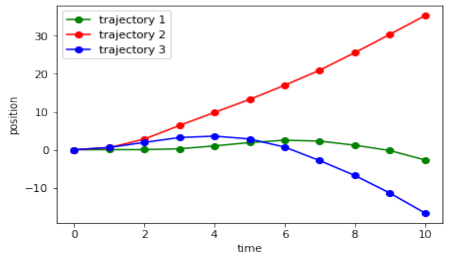

Figure 1: True Motion Simulation - 3 Run

As suggested in the problem definition, the random control input results in 3 different true trajectories at each run in the Figure 1.

c

In the previous section A and B matrices are already expressed:

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1/2 * (\Delta t)^2 \\ \Delta t \end{bmatrix}$$

And the equation for covariance matrix is $R = B * B^T$,which is:

$$R = \begin{bmatrix} 1/4\Delta t)^4 & 1/2 * (\Delta t)^3 \\ 1/2 * (\Delta t)^3 & (\Delta t)^2 \end{bmatrix}$$

After this point, these 3 matrices can be plugged into Equation (3.4) of the book to compute $p(x_t|u_t, x_{t-1})$.

e

I did specify $P(0)$ -initial state covariance- as a zero-matrix in the beginning. In each prediction iteration, the covariance changes with accumulated uncertainty. With that, the predicted state covariance has 2 eigenvalues, which corresponds to the square of radiuses of the ellipses. Also the rotation in the plots is due to eigenvectors which represents the correlation of the states. The resulting plots for $t = 1, 2, 3, 4, 5$ can be observed below.

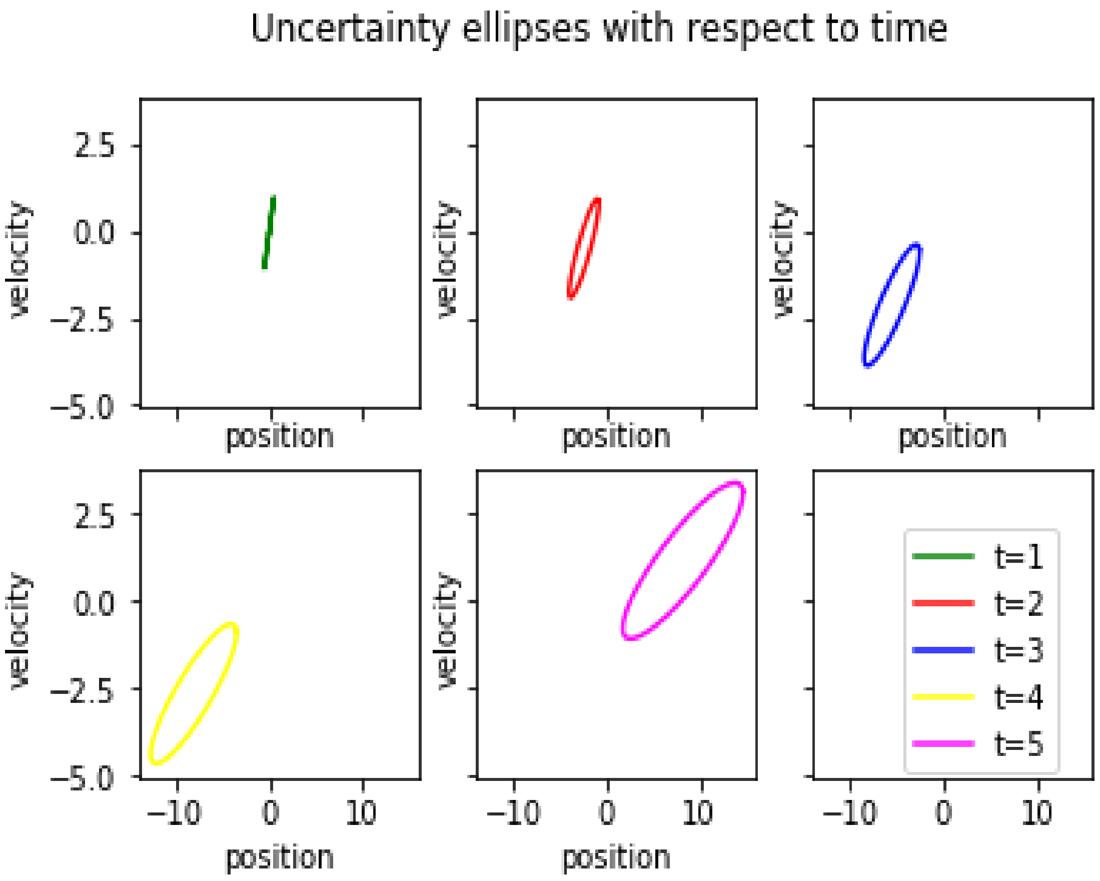

Figure 2: Uncertainty Ellipses

It might be difficult to notice the change in the correlation with these distinct plots. Below is a combined view that clearly represents the change in correlation.

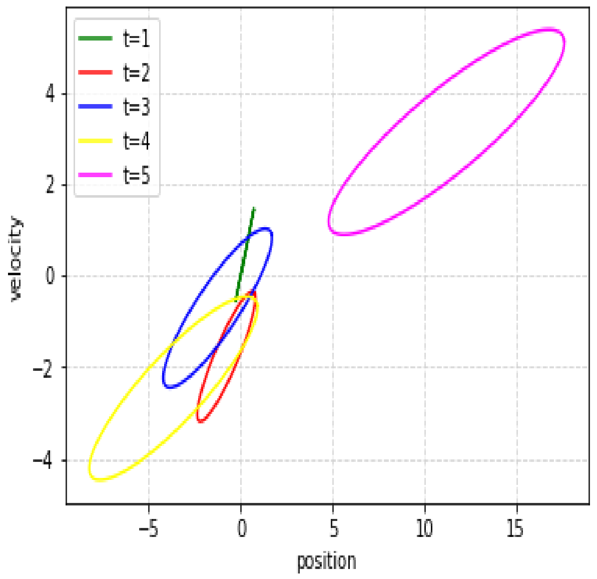

Figure 3: Uncertainty Ellipses - Combined

Note that the plots are produced in different sessions with different control inputs - accelerations, hence plots vary quantitatively.

f

In previous section, I also computed the correlation for each uncertainty ellipse.

t	**Correlation**
1	1.0
2	0.8944271909999159
3	0.8783100656536799
4	0.8728715609439696
5	0.8703882797784891

Table 1: Time-Correlation

There is an obvious tendency to 0 in the correlation as t increases. To prove this hypothesis relatively larger t values can be given as in the Figure 4.

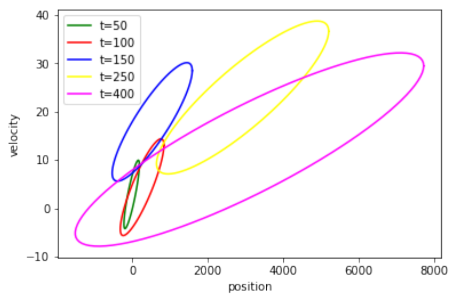

Figure 4: Uncertainty Ellipses - Large t

t	Correlation
50	0.8660687083024937
100	0.8660362293049649
150	0.866030215076776
250	0.8660271358404424
400	0.8660260803675782

Table 2: Large time-Correlation

And finally, the Table 2 further supports the hypothesis. As t goes to ∞, the correlation continues to decrease to 0.

2

a

In the problem definition, it is denoted that the measurement is only made for the position. Therefore $K = 1$. From Kalman Filter, we know that C is a matrix of size $k \times n$. In this case it is 1×2. Moreover, the problem suggests that *in expectation, our sensor measures the true location.* That is why, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$, which means that the model have a very [initial] optimistic view of the world such that it can measure the correct position precisely. Finally, since $k = 1$, the measurement noise covariance matrix is of dimension 1×1 such that $Q = \begin{bmatrix} 10 \end{bmatrix}$.

b

The initial Gaussian estimate parameters is as follows:

$\overline{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\overline{\Sigma} = \begin{bmatrix} 20 & 6 \\ 6 & 3 \end{bmatrix}$, Note that the parameters are chosen as random with a level of intuition. Since the measurement update is a relative process, absolute values don't matter. In the end, we would get a better belief with a mean value that is closer to the measurement and with a narrower variance having a covariance matrix with lower-valued entries.

After one-measurement update step with measurement $z = 5$ the Gaussian parameters is as follows:

$$\mu = \begin{bmatrix} 3.3333333 \\ 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 6.6666666733333 & 2 \\ 2 & 1.8 \end{bmatrix}$$

Naturally, the position is now closer to the measurement. Also, the updated covariance matrix have much lower-valued entries.

c

Figure 5 illustrates the uncertainty ellipse corresponding to the measurement update above.

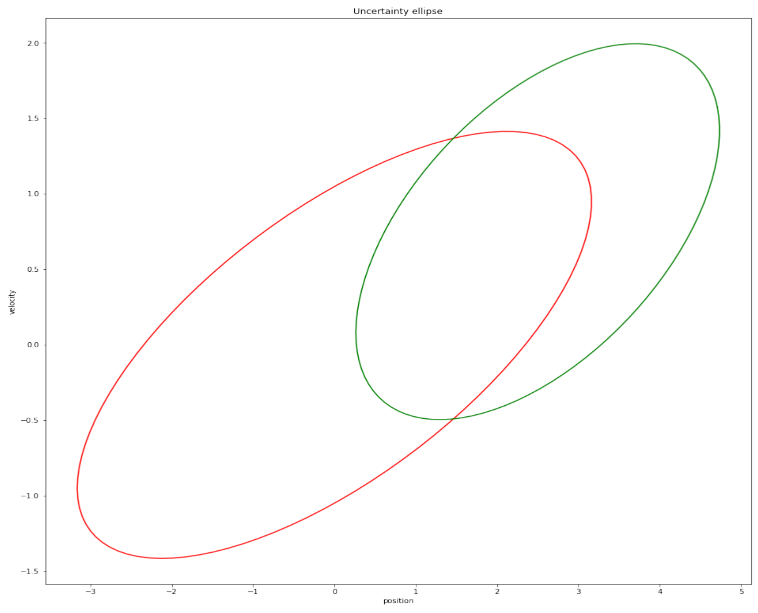

Figure 5: Uncertainty Ellipse after Measurement Update

d

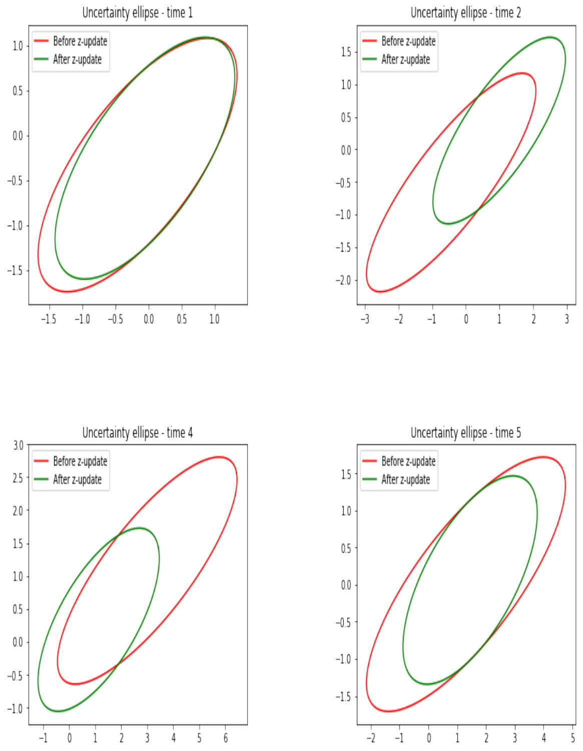

Figure 6: Uncertainty Ellipses on Full Kalman Iteration

In Figure 6, all the measurements are sampled with the scale and around the location of position. Hence, the ellipses after measurement update have slight gains. Nonetheless, the Kalman Filter proves its usefulness even in this extreme case. At each time step, the belief is improved by moving closer to *true* states. Besides the variances are reduced in all cases.

3

c

For this subproblem, only the prediction part of particle filter is implemented. The implementation can be analyzed in *q1_sol.py* for further details. In brief, it calls *sample_motion_model* function recursively, until given time. The accumulated uncertainty can be observed in the layout of particles in Figure 7.

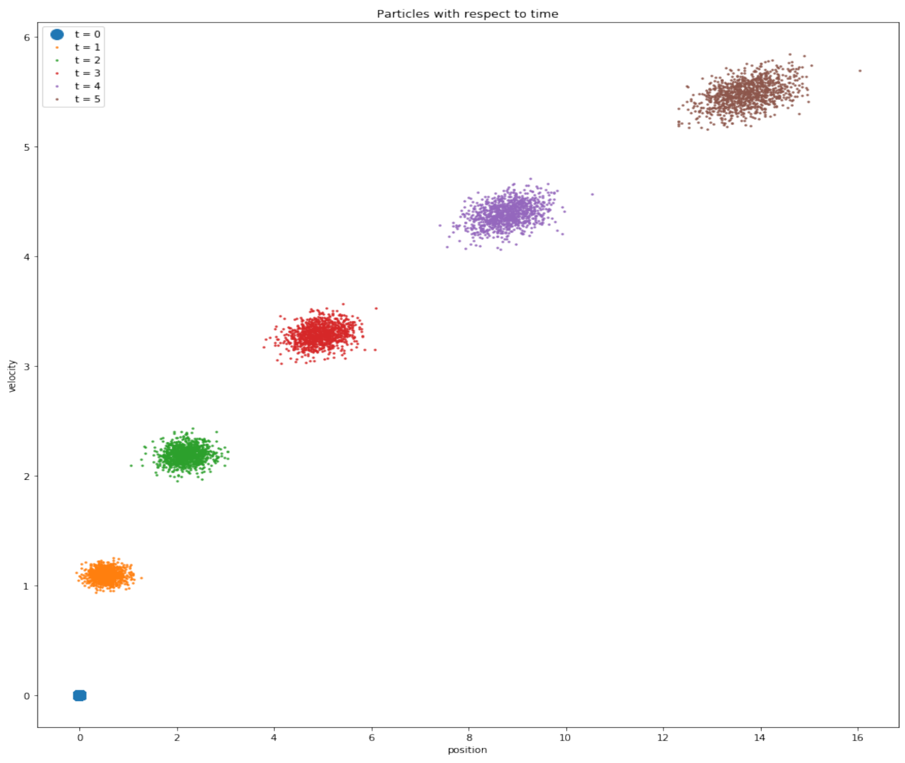

Figure 7: Particle Filter with only Prediction

d

Firstly, the one-step measurement update sub-problem is solved. It hugely reduces the number of distinctly located particles, i.e. in replacement process, certain particles dominated the state space with their relatively higher probability masses. In Figure 8 , it can be seen there exists only 3 distinct particles after measurement update. Also note that, the particles at $t = 5$ are taken from previous section as they are.

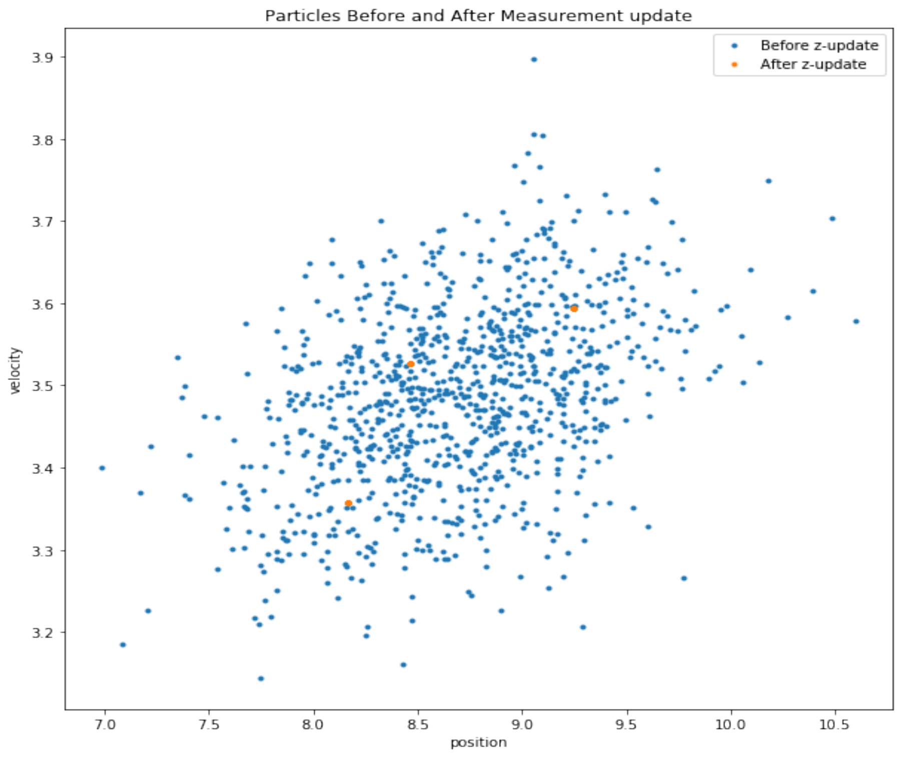

Figure 8: Particle Filter with Measurement Update

Subsequently, a full iteration of particle filter for $t = 1, 2, 3, 4$ and 5 follows. In below figures, one can observe the behavior of particle filter in full detail. Before each measurement update, the predicted particle filter set has many more distinctly located particles, whereas after measurement is integrated their count reduces significantly.

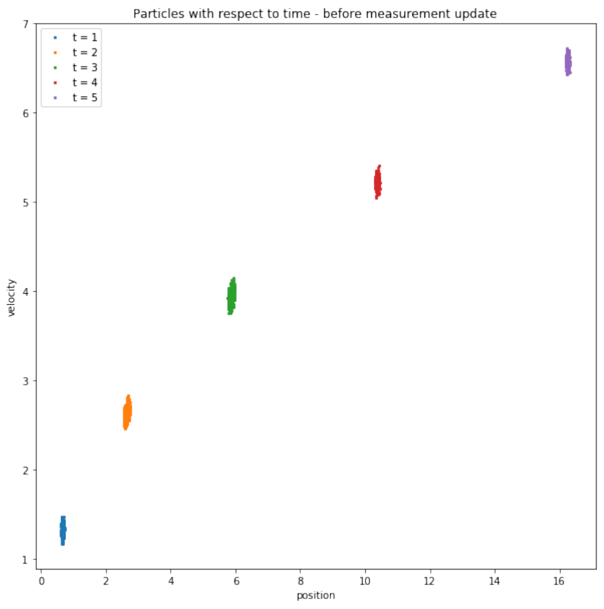

Figure 9: Particles before Measurement Update

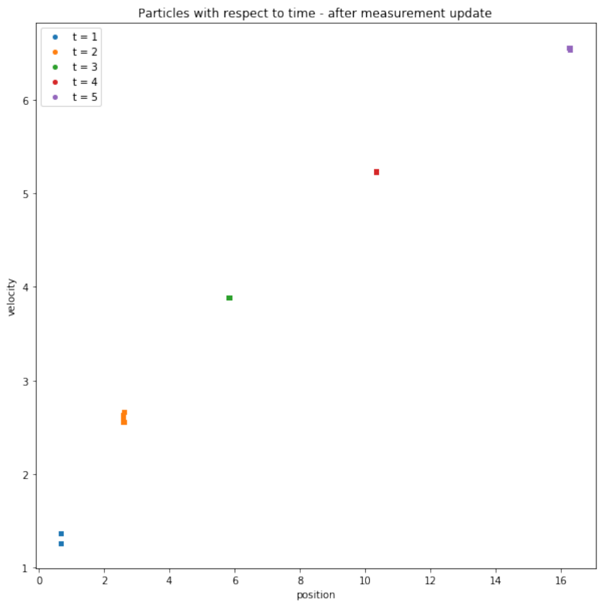

Figure 10: Particles after Measurement Update

e

Finally, a comparison involving KF and PF($M = 1000$) has surprising results. Normally, I would expect PF to outperform KF. However, it is not the case in here. In below, the timings can be inspected:

Algorithm	Time (in seconds)
Particle Filter	0.041847944
Kalman Filter	0.00029730

Table 3: Benchmark

I didn't try an optimized implementation of particle filter that exploits vectorization and SIMD architecture. In those cases, it would definitely outperform Kalman Filter.