PARTICLE FILTERING METHOD FOR OBJECT TRACKING AND SENSOR MANAGEMENT USING MUTUAL INFORMATION

by

Luyu Yang

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2012

CONTENTS

LIST OF FIGURES

ABSTRACT

Generally, there is no analytic solution to object tracking problems in non-linear non-Gaussian scenarios, which is a common type of problem nowadays. A particle filter is a numerical method that can be applied to any class of model regardless of linear and Gaussian assumptions as in the Kalman filter, and has the same benefits of constant memory requirement and real-time recursive estimation. In this report, a hidden Markov model is set up for state and observation evolution, and both the particle filter and the Kalman filter are developed and applied to generate tracking results. Our results show that in linear and Gaussian case, the performance of particle filtering is very close to the classic Kalman filter, which achieves the Cramer–Rao lower bound, while the particle filtering method can be applied much more extensively when linear and Gaussian assumptions are not justified in real problems.

In both object tracking and other problems such as detection, sensor management is an issue, as there has to be a trade-off between performance and cost. As sensors are commonly utilized for multiple purposes, a generic performance measure based on mutual information is developed. An overall sensor cost is computed by summing up the one-time cost of installation and the life-time cost of operation. To make the information gain and the cost comparable, a bit-dollar exchange rate is defined to compute the monetary value of the information gain. By combining the monetary gain and the cost into a single objective, a sensor configuration strategy can be chosen among multiple options.

1.1 Object Tracking and Particle Filter

In object tracking problems, there is a time series of the state of interest which we do not have direct access to and want to estimate, based on another series of measurements generated from the state. Usually the measurements are modeled as random variables, since there always exist some unpredictable inaccuracies in the observations regarding to the state of interest, often as noise. The goal is to estimate the unknown underlying state as closely as possible using not only the observations, but also our prior knowledge of the statistical characteristics of the inaccuracies of measurements and the evolution of the state. The performance is often evaluated statistically via the mean squared error.

For example, in guidance and navigation, we are interested in real-time positions of a target object, and we only have indirect observations contaminated with noise with regards to the object location, through remote technologies of radar, sonar, etc. Also we have prior knowledge of how the object moves around based on the laws of motion, environmental constraints, and the mechanical structure and motor type of the object. In most cases, we are aware of an area the object most probably resides in before we start tracking.

Special cases with linearity and Gaussian noise admit analytic solutions. Particularly, the Kalman filter is widely applied when linear and Gaussian assumptions are close to the background model, and achieves statistical optimality when the two assumptions are satisfied. However, more complicated object tracking problems are not possible to analyze analytically, and the naive linear and Gaussian assumptions can deteriorate the tracking performance in many real applications if they do not fit with the circumstance. As such, numerical solutions are used instead.

Particle filtering idea originated from Gordon et al. in 1993 [4]. After its introduction, the particle filter became a popular numerical method for estimation and inference in more general classes of problems where analytic methods like the Kalman filter are inadmissible, especially in non-linear non-Gaussian problems.

Compared with classical numerical methods that are extensions of the Kalman filter, the particle filter does not reply on a linear approximation scheme as in the extended Kalman filter proposed by McElhoe [6], and is not restricted to deterministic sampling and Gaussian approximation as in the unscented Kalman filter first introduced by Julier and Uhlmann [5]. Furthermore, with sufficient samples, the particle filtering method approaches statistical optimality, and can outperform either the extended Kalman filter or the unscented Kalman filter. Although particle filter is computationally more expensive, nowadays ever-increasing computing power has already been enabled particle filtering in real-time applications such as econometrics [3] and computer vision [1].

1.2 Sensor Management for Generic Purposes

The use of sensors has become popular nowadays, from navigation in robotics to object detection in radar. Theoretically, the more sensors we use, the more reliability we can get, since additional information can never hurt, as information theory claims, although sometimes in applications extra information can confuse useful content and therefore make the objective harder to achieve and becomes harmful. On the other hand, in practice, sensors can become very expensive depending on the situation. For example, a high-precision radar can cost millions of dollars. Therefore, there is an issue in how to manage sensors in order to meet our objective and also reduce cost.

For sensor applications, multiple criteria can be used for evaluation purposes. In object tracking problems, mean squared error is often adopted as the criterion to evaluate the performance of a system. In detection problems, the false-alarm and miss probabilities are computed to gauge performance. Just in these two common problems, as the objective varies, the optimal configuration of the sensors is different in general, i.e., the best sensor system for object tracking usually is not the best system in detection, and vice versa. Without knowing in advance what the sensors will be used for, it is hard to find an optimal configuration of the sensors, and often sensors are utilized for multiple purposes. For instance, radar can be

used for both object tracking and detection, and as it is usually very expensive. It is quite common to use radar in dual systems, with information extracted to analyze object location in one system and determine if there is any intruder in the other system.

Therefore, there needs to be a way to evaluate the generic performance of a sensor system, and to find the optimal configuration of the sensor system with a balance between generic performance and cost. The problem is often illustrated in the choice among multiple options. Thus, the goal becomes choosing the best option among all candidates with the most desired balance between performance and cost.

1.3 Organization

The report is organized as follows. Section 2 includes the model setup for our object tracking problem, and the framework of particle filters in general. Then it describes our designed particle filter and illustrates the performance with experimental results. Section 3 moves to our mutual information-based sensor management method and shows the result in experiments. Section 4 concludes the report by giving a summary of our work and pointing out potential applications.

2.1 Model Setup

In object tracking problems, a hidden Markov model [7] is often used to describe the time series of state evolution and observed measurements, as the current state is only dependent on the most recent state variables given the physical laws of motion and independent of the state variables in the past, and the measurement only reflects the observation with respect to the current state variable, containing noise reasonably believed to be independent of the time indices.

In a hidden Markov model, there is a discrete Markov process $\{X_1, X_2, \ldots\}$ as the state variables, with $X_n \in \mathcal{X}, n \geqslant 1$. For discrete $\mathcal{X}$,

$$p(X_1 = x_1) = \mu(x_1) \tag{2.1}$$

$$p(X_n = x_n | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = p(X_n = x_n | X_{n-1} = x_{n-1}) \tag{2.2}$$

$$= f(x_n | x_{n-1}), \quad n > 1. \tag{2.3}$$

And there are corresponding observations modeled by another time series $\{Y_1, Y_2, \ldots\}$, with each Y_n dependent and only dependent on X_n, i.e., for $Y_n \in \mathcal{Y}$,

$$p(Y_n = y_n | X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) = p(Y_n = y_n | X_n = x_n) \tag{2.4}$$

$$= g(y_n | x_n), \quad n \geqslant 1. \tag{2.5}$$

In general, the transition and observation functions f and g can be different with respect to different time indices. In our case, the homogenous model is considered, which has the same transition and observation functions for all time indices.

In the Bayesian context of hidden Markov models, the prior distribution of the hidden process is defined by (2.1) to (2.3), and the likelihood function is defined by

(2.4) and (2.5), i.e.,

$$p(x_{1:n}) = \mu(x_1) \prod_{k=2}^{n} f(x_k|x_{k-1}) \tag{2.6}$$

$$p(y_{1:n}|x_{1:n}) = \prod_{k=1}^{n} g(y_k|x_k). \tag{2.7}$$

The posterior distribution is given by

$$p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})}, \tag{2.8}$$

where

$$p(x_{1:n}, y_{1:n}) = p(x_{1:n})p(y_{1:n}|x_{1:n}), \tag{2.9}$$

$$p(y_{1:n}) = \begin{cases} \int p(x_{1:n}, y_{1:n}) dx_{1:n}, & \text{for continuous } \mathcal{X} \\ \sum_{x_{1:n}} p(x_{1:n}, y_{1:n}), & \text{for discrete } \mathcal{X}. \end{cases} \tag{2.10}$$

In our object tracking problem, the state variables are the positions of the target object, and the observations are our measurements through indirect methods, containing inaccuracies. The goal is to estimate the hidden invisible state variables $\{X_1, X_2, \ldots\}$ solely based on observations $\{Y_1, Y_2, \ldots\}$ and the joint distribution of $\{X_n\}$ and $\{Y_n\}$.

2.2 Particle Filter

The particle filter [2] uses Monte–Carlo algorithms to provide a numerical solution for estimation problems, using particles as samples to approximate the posterior distribution $p(x_{1:n}|y_{1:n})$ and the marginal distribution $p(y_{1:n})$. In the Kalman filter, the mean and covariance are kept to represent the distributions as they are sufficient to define any Gaussian distribution exactly and the linearity guarantees the all conditional and marginal distributions are Gaussian given a prior Gaussian

distribution. While in non-linear non-Gaussian cases, only mean and covariance are not enough to represent the distributions. The particle filter uses a finite number of particle samples to approximate such distributions, and can be applied to any class of model regardless of the linear and Gaussian constraints. It approaches statistical optimality as the number of particles grows and guarantees good performs if there is a sufficient number of samples. Also the constant particle number feature is well-suited to high dimensional spaces, where traditional numerical schemes have storage requirement increasing at least linearly with the rate of space size growth.

For the object tracking problems, the particle filter aims at the approximation of $p(x_n|y_{1:n})$, which represents the characteristics of the current underlying state of interest given the information of all the observations in the history up to the present time. Then by taking advantage of the information regarding $p(x_n|y_{1:n})$ it can provide a statistically sound solution to estimate x_n at the present time, which is equivalent to finding the object location with all the noisy observations in the past and the present. In the following description of the particle filter, continuous $\mathcal{X}$ and $\mathcal{Y}$ are assumed, with the discrete cases handled by taking the sum in place of the integral when appropriate.

Given the properties of the hidden Markov model, the posterior distribution satisfies

$$p(x_n|y_{1:n}) = \frac{p(x_n, y_n|y_{1:n-1})}{p(y_n|y_{1:n-1})} \tag{2.11}$$

$$= \frac{p(x_n|y_{1:n-1})g(y_n|x_n)}{p(y_n|y_{1:n-1})}, \tag{2.12}$$

where

$$p(x_n|y_{1:n-1}) = \int f(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}. \tag{2.13}$$

Commonly (2.11) and (2.12) are called the prediction step, and (2.13) is called the update step.

In particle filtering, we want to approximate the distribution $p(x_n|y_{1:n})$ numerically in a sequential manner over time, which can be achieved by the integration of

$P(x_{1:n}|y_{1:n})$. We can equivalently write

$$p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})}. \tag{2.14}$$

Using Monte–Carlo methods, we can approximate any probability density function with a finite number of samples, i.e., if we sample N times from $p(x_{1:n}|y_{1:n})$ with $X^i \sim p(x_{1:n}|y_{1:n}), 1 \leqslant i \leqslant N$, then the approximation becomes

$$\hat{p}(x_{1:n}|y_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i}(x_{1:n}). \tag{2.15}$$

Since $p(x_{1:n}|y_{1:n})$ can be very complex and high-dimensional which is hard to generate samples, and the computational complexity of such scheme increases as n grows, we instead generate samples from a simpler distribution, named the importance distribution, and then associate the samples with different weights as an indirect way to approximate the original distribution. To make the importance distribution $q_n(x_n)$ able to work, it must satisfy

$$q_n(x_{1:n}) > 0, \text{ for } p(x_{1:n}|y_{1:n}) > 0. \tag{2.16}$$

Then (2.14) becomes

$$p(x_{1:n}|y_{1:n}) = \frac{w_n(x_{1:n})q_n(x_{1:n})}{p(y_{1:n})}, \tag{2.17}$$

where $w_n(x_{1:n})$ is the weight

$$w_n(x_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{q_n(x_{1:n})}. \tag{2.18}$$

And (2.15) becomes

$$\hat{p}(x_{1:n}|y_{1:n}) = \frac{\sum_{i=1}^{N} w_n(X^i)\delta_{X^i}(x_{1:n})}{\sum_{j=1}^{N} w_n(X^j)}. \tag{2.19}$$

Typically we want the importance distribution to be easy to generate samples, and also be close to the original distribution, as it reduces the variance of the weights, which allows many samples to contribute to the overall approximation rather than have only a few samples dominate the performance, losing both efficiency and accuracy with fewer effective samples.

To have fixed memory requirement rather than keep it growing as the time evolves, the importance distribution is selected to have Markov property. Typically,

$$q_n(x_n|x_{1:n-1}) = q_n(x_n|x_{n-1}, y_n). \tag{2.20}$$

So at each time step it depends only on the previous state and the current observation, which has constant memory usage.

With the Markov property of the importance distribution, the weights can be decomposed into

$$\begin{aligned} w_n(x_{1:n}) &= \frac{p(x_{1:n-1}, y_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{p(x_{1:n}, y_{1:n})}{p(x_{1:n-1}, y_{1:n-1}) q_n(x_n|x_{1:n-1})} && (2.21)\\ &= w_{n-1}(x_{1:n-1}) \alpha_n(x_{1:n}) && (2.22)\\ &= w_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}), && (2.23) \end{aligned}$$

where the weight update coefficient is

$$\begin{aligned} \alpha_n(x_{1:n}) &= \frac{p(x_{1:n}, y_{1:n})}{p(x_{1:n-1}, y_{1:n-1}) q_n(x_n|x_{1:n-1})} && (2.24)\\ &= \frac{p(x_n|x_{n-1}) p(y_n|x_n)}{q_n(x_n|x_{n-1}, y_n)}. && (2.25) \end{aligned}$$

At each time step, an effective sample size (ESS) value is computed, which measures the variance of weights.

$$\text{ESS} = \frac{1}{\sum_{i=1}^{N} (W_n^i)^2}, \tag{2.26}$$

where

$$W_n^i = \frac{w_n(X^i)}{\sum_{j=1}^{N} w_n(X^j)}, \tag{2.27}$$

which can be achieved by first setting W_n^i to be $w_n(X^i)$ and then normalizing it to sum to one, denoted by $W_n^i \propto w_n(X^i)$.

If the ESS is below a threshold, particles are resampled, to generate new particles with equal weights of $\frac{1}{N}$, and keep the resampled particles instead. Resampling allows the removal of particles with low weights and multiplies particles with high weights in high probability, so to make sure that particles all contribute to the overall approximation, instead of letting a few particles control the major part of the approximation, since it is a waste of the limited particle resource and results in poor performance as the number of effective particles is reduced. The ESS is evaluated and then used as a criterion to avoid unnecessary resampling, which introduces noise, and preserve the advantages of concentration on area with high probability mass to prevent degeneracy when necessary.

In summary, particle filtering works as follows, with N denoting the number of particles.

At time step $n = 1$

- Sample $X_1^i \sim q(x_1|y_1)$
- Compute weights $w_1(X_1^i) = \frac{\mu(X_1^i)g(y_1|X_1^i)}{q(X_1^i|y_1)}$ and $W_1^i \propto w_1(X_1^i)$
- If ESS is below a threshold, resample $\{W_1^i, X_1^i\}$ to obtain N equally weighted particles $\{\frac{1}{N}, \bar{X}_1^i\}$ and replace $\{W_1^i, X_1^i\}$

At time step $n \geqslant 2$

- Sample $X_n^i \sim q(x_n|y_n, X_{n-1}^i)$
- Compute weight update coefficient $\alpha_n(X_{n-1:n}^i) = \frac{g(y_n|X_n^i)f(X_n^i|X_{n-1}^i)}{q(X_n^i|y_n, X_{n-1}^i)}$ and set $W_n^i \propto \alpha_n(X_{n-1:n}^i)W_{n-1}^i$
- If ESS is below a threshold, resample $\{W_n^i, X_n^i\}$ to obtain N equally weighted particles $\{\frac{1}{N}, \bar{X}_n^i\}$ and replace $\{W_n^i, X_n^i\}$

At each time step, the approximation is given by

$$\hat{p}(x_n|y_{1:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_n^i}(x_n). \tag{2.28}$$

The first time step initilizes the particles and weights using the prior distribution. At following time steps, samples are first generated under the importance distribution only dependent on the previous state and the current observation, which forms the particles and acts as the prediction step. Then weights associated with particles are computed and normalized. Finally, the ESS is computed to determine whether resampling is needed to prevent degeneracy, which concludes the update step.

2.3 Experiments

With experiments, we illustrate the performance of particle filter compared with the classic Kalman filter.

First we set up system model following the typical framework of hidden Markov model with linear and Gaussian properties, i.e., the progression functions are linear and the distributions are Gaussian, where the Kalman filter can be applied directly. Specifically, for $X_n \in \mathcal{X} = \mathfrak{R}^3$ and $Y_n \in \mathcal{Y} = \mathfrak{R}^3$,

$$X_n = AX_{n-1} + BV_n, \text{ with } X_1 \sim N(0, I_{3\times 3}),\ V_n \overset{iid}{\sim} N(0, I_{3\times 3}) \tag{2.29}$$

$$Y_n = CX_n + DW_n, \text{ with } W_n \overset{iid}{\sim} N(0, I_{3\times 3}), \tag{2.30}$$

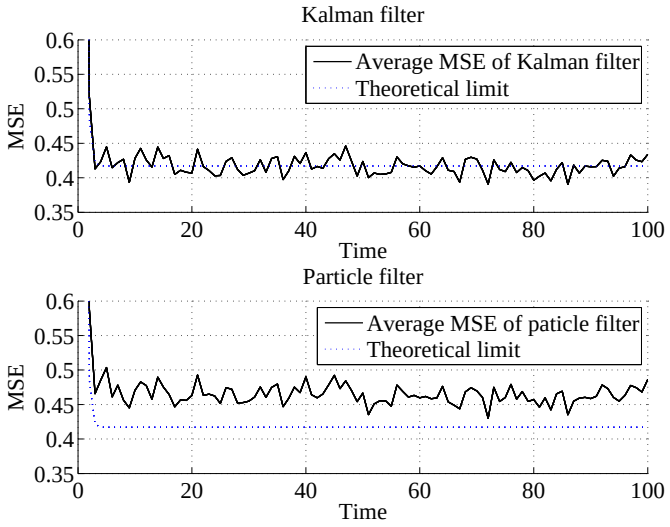

Figure 2.1: Mean squared error using the Kalman filter (top) and the particle filter (bottom).

with parameters chosen to be

$$A = \begin{bmatrix} 0.4 & 0.1 & 0.2 \\ 0.2 & 0.3 & 0.2 \\ 0.1 & 0.3 & 0.3 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.3 & 0.01 & 0.02 \\ 0.01 & 0.5 & 0.03 \\ 0.01 & 0.02 & 0.4 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.2 & 0.3 & 0.5 \end{bmatrix}$$

$$D = \begin{bmatrix} 0.3 \end{bmatrix}.$$

In this case, $\mu(x) \sim N(0, I_{3\times 3})$, $f(x'|x) \sim N(Ax, BB^T)$, and $g(y|x) \sim N(Cx, DD^T)$.

The importance distribution is defined with an overall idea to combine the information from particles in the last time step and also the current observation. As the variance indicates the precision level, the two individual predictions from the last

state and the current observation are combined to generate the overall prediction, with weights inversely proportional to the variance. For matrices, covariance is used instead of their variances. The mean of $q(x_1|y_1)$ is set to C^+y_1 and the mean of $q(x_n|x_{n-1}, y_n)$ is normalized version of $BB^TC^+y_1 + var_Y Ax_{n-1}$, where observation prediction variance is $var_Y = (C^+DI_{3\times3})(C^+DI_{3\times3})^T$.

The ESS threshold is selected to be $N/2$, and systematic resampling is used in the resampling process., which first generates $U_1 \sim U[0, \frac{1}{N})$ and sets $U_i = U_1 + \frac{i-1}{N}$ for $2 \leqslant i \leqslant N$, and select $\bar{X}_n^i = X_n^j$ in an iterative process to increase i and j alternatively up to N and choose the pair satisfying new particles $\sum_{k=1}^{j-1} W_n^k \leqslant U_i < \sum_{k=1}^{j} W_n^k$.

Theoretically, the Kalman filter achieves statistical optimality, with the mean squared error reaching Cramer–Rao lower bound. The result of the Kalman filter is in accordance with such expectation. The performance of the particle filter is not as good as the Kalman filter since it is a numerical method, but very close to the theoretical lower bound, which shows it can have satisfactory performance to object tracking problems.

3 MUTUAL INFORMATION-BASED SENSOR MANAGEMENT

3.1 Problem of Interest

We are interested in object tracking and detection types of problems, i.e., we have a state of interest, but we only have access to an indirect observation contaminated by noise. The state of interest can be either discrete or continuous, and the observations are related to the state of interest probabilistically. We want to get the best knowledge of the state of interest from the observations, regardless of whether the sensor system is used for object tracking or detection. And we are comparing options between sensor Y_1 alone, sensor Y_2 alone or both, and takes the cost of installation and operation into consideration. We aim to set up a combined objective to encourage better performance and lower cost.

3.2 Performance and Cost

Two criteria to evaluate sensor configurations are performance and cost. For generic purposes, mutual information is not directed at any specific application type, but only considers the information gain between two random variables. Since we aim at sensor management for generic purposes, which means the same set of sensors can be used for multiple tasks, mutual information is adopted as the performance metric. The higher the mutual information between the sensor observations and the state is, the better the performance is. Our goal is to get good performance, which is equivalent to high mutual information.

On the other hand, installation and operation of sensors incur cost, which is another consideration in sensor management. Intuitively, the more higher-precision sensors we use, the better the performance is in terms of mutual information. However, the additional cost may be an obstacle. In order to put the cost of installation and operation into convenient metric, we sum up the overall cost per life time into

our defined overall sensor cost, i.e.,

$$\text{cost}_{\text{sensor}} = \text{cost}_{\text{installation}} + \text{cost}_{\text{per year operation}} \times \text{expected life time.} \quad (3.1)$$

As mutual information has units of bits and the cost has units of dollars, in order to make them comparable, a per-bit monetary exchange rate is defined to transform information into money. By multiplying the mutual information with the pre-defined per-bit monetary exchange rate, we get the monetary gain of the information. And by subtracting the cost, we have an overall objective of net value that we want to optimize, which is our target function. The option with the highest value of the target function is the optimal solution in this context.

3.3 Experiments

We experiment with both the discrete and continuous state cases. In each case, for each option, the mutual information gain between sensor measurements and target state is first computed. Then by multiplying with the monetary exchange rate per bit of information and then subtracting the cost, we get the overall net value. Then comparison between three options is conducted. The option with the highest overall net value is selected to be the best strategy, with the optimal balance between information gain and cost saving in our context.

We divide our experiments into two categories. In each category, we have ten cases. In one category, the conditional variance of Y_1 itself is always 1, and Y_1 and Y_2 are always conditionally independent. In the ith case, the conditional variance of Y_2 becomes $1/i$. In the other category, the individual conditional variances of Y_1 and Y_2 are set to 1 and $1/3$ respectively. In the ith case, the conditional correlation coefficient between Y_1 and Y_2 is $(i-1) \times 0.1$.

In the discrete case, we set the state of interest $X \sim \text{Bernoulli}(0.7)$. The sensor

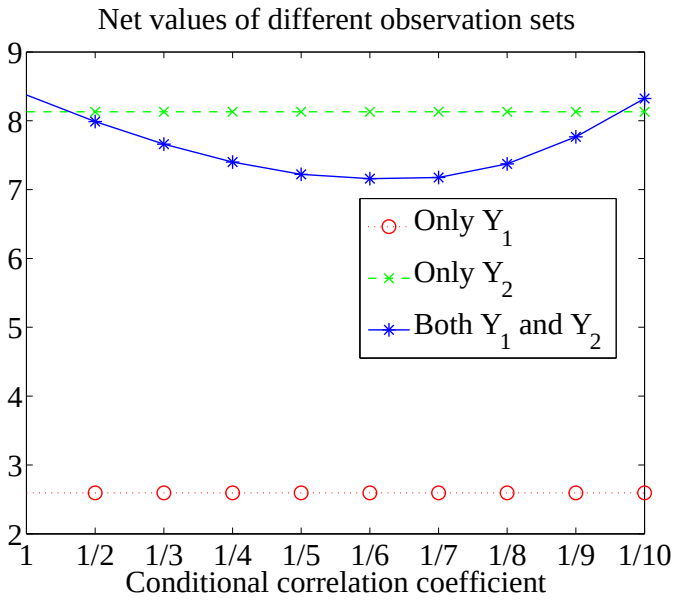

Figure 3.1: Net values for discrete case with varying conditional correlation coefficient.

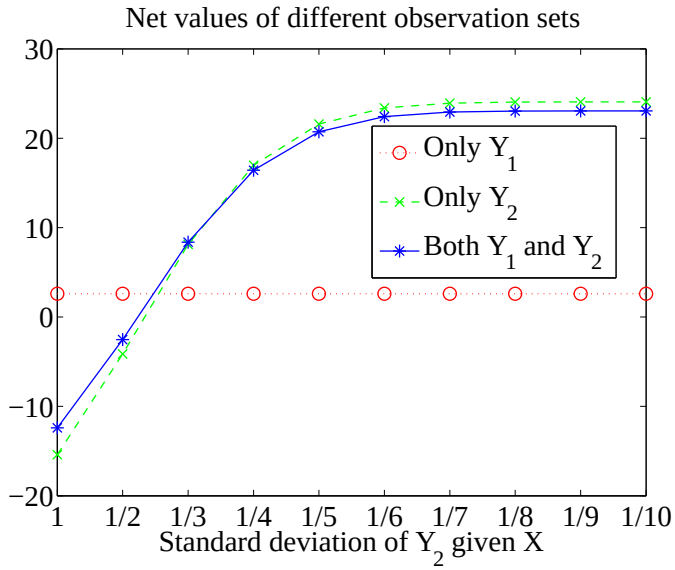

Figure 3.2: Net values for discrete case with varying conditional variance.

observations Y_1 and Y_2 are related to X in different ways, with

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} |X = 0 \sim N(\text{mean}_0, \text{cov}_0) \tag{3.2}$$

and

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} |X = 1 \sim N(\text{mean}_1, \text{cov}_1). \tag{3.3}$$

We set mean_0 to be $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and mean_1 to be $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. We set the per life-time cost of Y_1, Y_2 and both to be $\$2,000$, $\$20,000$, and $\$21,000$. The monetary exchange rate per bit of information is set to be $\$50,000$ per bit.

In the continuous case, we set the state of interest $X \sim N(0,1)$. The sensor observations are related to X in the following manner,

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} |X \sim N(\begin{bmatrix} X \\ X \end{bmatrix}, \text{cov}). \tag{3.4}$$

We set the per life-time cost of Y_1, Y_2 and both to be $\$1,500$, $\$2,500$, and $\$3,500$. The monetary exchange rate per bit of information is set to be $\$50,000$ per bit.

The results in both discrete and continuous scenarios show that in the fixed conditional variance case, using both sensors has better overall net value when the conditional correlation coefficient is close to 0 or 1. When the two sensors are not related to each other, the combined information gain is bigger as there is not much redundancy in their individual information regarding the state of interest. On the other hand, if the conditional correlation coefficient is close to 1, the two sensors confirm each other, and therefore it helps by having an additional sensor. In the conditionally independent observation case, using both sensors is a better choice when sensor Y_2 has a high precision but not too high compared with Y_1. When Y_2 loses precision, Y_1 is preferred because it is less expensive, and as Y_2 gets better, the information gain gets bigger, with the overall net value increasing. As the conditional variance of Y_2 decreases, using both is first preferred because if Y_2

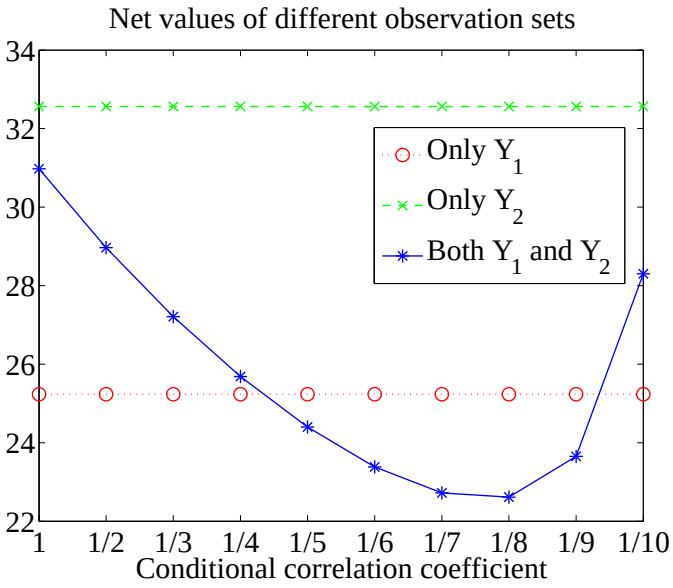

Figure 3.3: Net values for continuous case with varying conditional correlation coefficient.

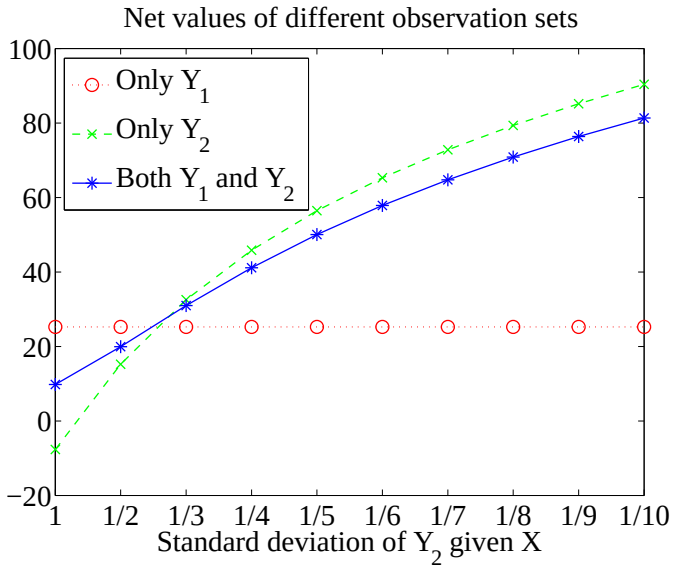

Figure 3.4: Net values for continuous case with varying conditional variance.

is not too much better than Y_1 then the information of Y_1 is also helpful. And if the variance of Y_2 gets too small, then using only Y_2 is preferred because Y_1 can no longer help much.

4 CONCLUSION

In the linear and Gaussian case, the particle filtering method has been shown to perform comparable to the Kalman filter, which achieves the statistical optimality of Cramer–Rao lower bound. The Kalman filter can only be applied in the strictly linear Gaussian case, and can have acceptable performance only if the background model is very close the linear and Gaussian assumptions, which generally limits its range of application.

Although some extensions to the Kalman filter exist, they rely heavily on either linear or Gaussian approximation. The most common extension, the extended Kalman filter, replies on the first-order approximation of the state evolution, and can perform poorly if it is not close to the true evolution process. Another common extension, unscented Kalman filter, is based on deterministic sampling and Gaussian approximation, which is not suitable for general non-linear, non-Gaussian situations and high-dimensional scenarios. Particle filter can be applied to any class of model, and the core Monte–Carlo algorithm guarantees the performance given a sufficient number of particles is used. Also it has fixed memory requirement and real-time recursive operation processing. Along with ever-increasing computing power, the particle filter appears alluring for modern applications.

Our sensor management method is based on mutual information for generic purposes. Rather than relying on the mean squared error in object tracking or false-alarm and miss probabilities in detection, mutual information is a generic measurement of the gain from the observation regarding the state of interest, regardless of the application purpose, as different purposes contradict with each other even in the simple case of object tracking and detection. By transforming the information gain into a monetary gain through a pre-defined bit-dollar exchange rate, it allows the comparison of the information gain with the cost. The experiments show that the method follows with common objectives and therefore can be a valuable and convenient tool to select a sensor configuration option, with parameters set to meet our requirements depending on the situation.

REFERENCES

[1] Canton-Ferrer, Cristian, Josep R. Casas, and Montse Pardàs. 2011. Human motion capture using scalable body models. *Computer Vision and Image Understanding* 115(10):1363–1374.

[2] Doucet, Arnaud, Email Arnaudismacjp, and Adam M Johansen. 2008. A tutorial on particle filtering and smoothing : Fifteen years later. *Handbook of Nonlinear Filtering* l(December):4–6.

[3] Flury, Thomas, and Neil Shephard. 2008. Bayesian inference based only on simulated likelihood : particle filter analysis of dynamic economic models. *Econometric Theory* 27(413):1–28.

[4] Gordon, N.J., D.J. Salmond, and A.F.M. Smith. 1993. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F* 140(2):107–113.

[5] Julier, S., and J. Uhlmann. 1997. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*.

[6] McElhoe, Bruce A. 1966. An assessment of the navigation and course corrections for a manned flyby of mars or venus. *Aerospace and Electronic Systems, IEEE Transactions on* AES-2(4):613–623.

[7] Stratonovich, R. L. 1960. Conditional markov processes. *Theory of Probability and its Applications* 5(2):156–178.