A Comprehensive Study on Neural Decoding in Motor Cortex
Midterm Project
STA5107 Computational Methods in Statistics II

Yuhang Liu

March 5, 2015

Abstract

In this project, we focus on different algorithms in neural decoding. Specifically, we want to explore the performance of Sequential Monte Carlo (SMC) method in a Kalman filter model and an Inhomogeneous Poisson process model. Also, we want to compare SMC with Kalman filter algorithm and point process filter algorithm in terms of accuracy and time cost. Our results show that, as an overall trend, the R^2 accuracy rate increases as the sample size of SMC n increases and gets very close to closed-form algorithms with large n. Sometimes SMC works poorly with small n. For x-position and x-velocity, R^2 changes evidently with the sample size, but for y-position and y-velocity, R^2 values are relatively stable and good, even with small sample sizes. This may be due to the nature of the data. The time cost of SMC increases rapidly as the sample size increases. This should always be a concern when we apply SMC. When the model allows for straightforward derivation of closed-form formulae, they should be the first to consider. When SMC has to be used, there is always a trade-off between estimation accuracy and time cost.

Project Introduction

Motor Cortex

Motor cortex is the region of the cerebral cortex involved in the planning, control, and execution of voluntary movements. Classically motor cortex is an area of the frontal lobe located in the dorsal precentral gyrus immediately anterior to the central sulcus.[4] The motor cortex can be divided into three subareas. The primary motor cortex is the main part of generating neural impulses and the control center of physical movements. The premotor cortex is responsible for some aspects of motor control. The supplementary motor area has many proposed functions including the internally generated planning movement sequences, and the coordination of left and right side of the body. Figure 1. visualizes the location of these three subareas in human brain. For more information on motor cortex, please see [4].

Neural Decoding

Neural decoding is a hot topic in the field of neuroscience. Basically researchers are concerned with the reconstruction of physical movements and positions of the body from information that has already been encoded and represented in the brain by networks of neurons. The reconstruction involves creating appropriate mathematical models to explain for the relationship between neural information and physical information. Neural decoding allows the researcher to explicitly interpret the neuron actions. Therefore, the main goal of neural decoding is to characterize how the electrical activity of neurons elicit activity and responses in the brain.[5]
With the rapid growth in large-scale bioinformatics and computational biological techniques, researchers have been able to handle large-sample neural decoding problems and explain for the neural behavior more deeply. Also, neuroscientists have initiated large-scale brain activity mapping or brain decoding projects. Figure 2. is an illustration of human neural decoding.

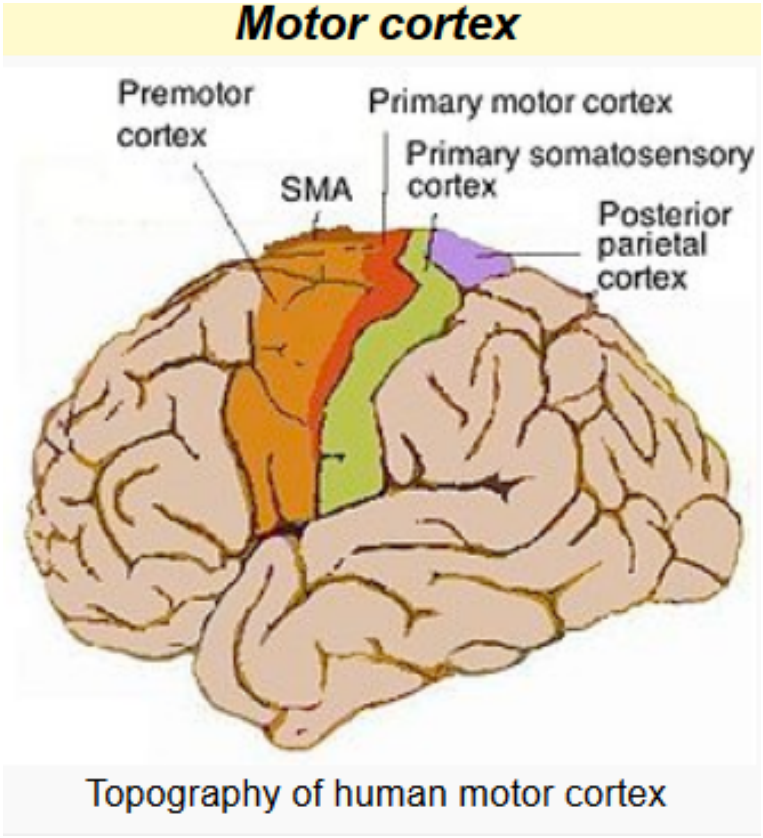

Figure 1: A Picture of Human Brain and Motor Cortex

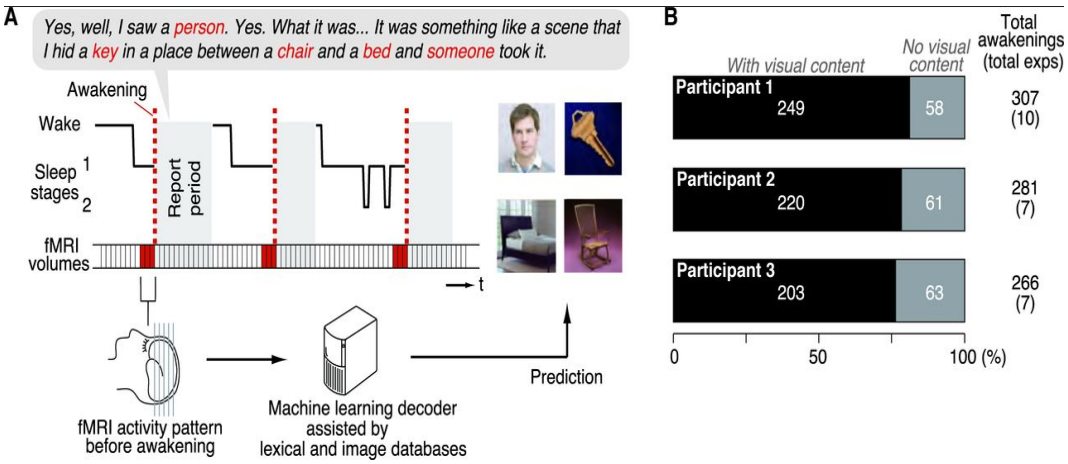

Figure 2: An Illustration of Human Neural Decoding

Data Description

In this project, we have two data sets (*midterm_train.mat* and *midterm_test.mat*). The first one is the training set and the second is the test set. Each of them has two variables: *kin* and *rate*. The variable *kin* is 4-dimensional and contains the kinematic state of the hand which includes x-position, y-position, x-velocity, and y-velocity. The variable *rate* is 42-dimensional and contains the spiking rates of 42 neurons, where the rate at each time is the number of spikes within 70ms. This data structure is the true case we have in practice. The behavior of neurons is always much more complicated than the external behavior of the body (movements). The training set has 3100 observations and the test set has 910 observations. We will use the training data to identify the statistical models. Then in the test data, we "pretend" not to know *kin*. Instead, we use the identified model and *rate* to reconstruct and make inferences on *kin*, and compare the estimates with the true values. In this project, we mainly focus on the first two components of *kin*, which are the hand positions.

Computational Methodologies

Kalman Filter Algorithm

A Kalman filter model views the data as a time series indexed by t ($t = 1, 2, ..., M$). Let x_t in $\mathbb{R}^4$ denote [x-position, y-position, x-velocity, and y-velocity] of a 2-dimensional hand movement at time t, and y_t in $\mathbb{R}^{42}$ denote the spiking rate of 42 neurons in the primary motor cortex at the same time. A classical Kalman filter model assumes linear relationship between x_{t+1} and x_t, also, between y_t and x_t. Specifically, we have the following model:

$$x_{t+1} = Ax_t + w_t$$

$$y_t = Hx_t + q_t$$

where $w_t \sim N(0, W)$, $q_t \sim N(0, Q)$. A, H, W, Q are matrices.
From matrix calculus, we can derive the closed-form formulae for estimating A, H, W, Q as following:

$$\hat{A} = (\sum_{k=2}^{M} x_k x_{k-1}^T)(\sum_{k=2}^{M} x_{k-1} x_{k-1}^T)^{-1}$$

$$\hat{W} = \frac{1}{M-1}(\sum_{k=2}^{M} x_k x_k^T - \hat{A}\sum_{k=2}^{M} x_{k-1}x_k^T)$$

$$\hat{H} = (\sum_{k=1}^{M} y_k x_k^T)(\sum_{k=1}^{M} x_k x_k^T)^{-1}$$

$$\hat{Q} = \frac{1}{M}(\sum_{k=1}^{M} y_k y_k^T - \hat{H}\sum_{k=1}^{M} x_k y_k^T)$$

To estimate (reconstruct) x_t from y_t, we have the following recursive formulae:
Prior estimate

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

Error covariance

$$P_k^- = AP_{k-1}A^T + W$$

The above two formulae are also called "time update" (or "prediction estimation").
Posterior estimate

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-)$$

Error covariance

$$P_k = (I - K_k H)P_k^-$$

Kalman gain

$$K_k = P_k^- H^T(HP_k^- H^T + Q)^{-1}$$

These three formulae are also called "measurement update" (or "filtering estimation").
The matrices A, H, W, and Q are constant in this project and are replaced by their estimates $\hat{A}$, $\hat{H}$, $\hat{W}$, and $\hat{Q}$. To measure how accurate the estimates are, we commonly use R^2 error as a criterion given below:

$$R^2 = 1 - \frac{\sum_k \parallel x_k - \hat{x}_k \parallel^2}{\sum_k \parallel x_k - \bar{x} \parallel^2}$$

R^2 error can also be computed component-wise. That is, for the i-th component, we have:

$$R_i^2 = 1 - \frac{\sum_k (x_{k,i} - \hat{x}_{k,i})^2}{\sum_k (x_{k,i} - \bar{x}_{.,i})^2}$$

Inhomogeneous Poisson Process (IPP)

In this model let $x_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^T$ in $\mathbb{R}^4$ denote (x-position, y-position, x-velocity, and y-velocity) of a 2-dimensional hand movement at time t_k, and $y_k = \{y_{k,c}\}$ in $\mathbb{R}^{42}$ be the spiking rate of 42 neurons in the primary motor cortex at the same time. Also, we still assume linear transition of x_k over time. That is,

$$x_{k+1} = Ax_k + w_k,$$

where $w_k \sim N(0, W)$. But there is no longer linear relationship between x_k and y_k. One of the consequences is that the posterior density of x given y is no longer Gaussian. Instead, we assume a Poisson regression model. That is,

$$y_{k,c} \sim Poisson(\lambda_{k,c})$$

where $\lambda_{k,c} = exp(\mu_c + \alpha_{1,c}p_{x,k} + \alpha_{2,c}p_{y,k} + \alpha_{3,c}v_{x,k} + \alpha_{4,c}v_{y,k})$. The logarithm of mean of the Poisson model at each time is linearly regressed on the x variable at the same time.

Since the mean changes over time, the series make up a sequence of inhomogeneous Poisson process. To estimate the parameters, it is hard to derive closed-form formulae, but we can apply Newton-Raphson method. Given $\{x_k\}$ and $\{y_k\}$, the joint log-likelihood is

$$LL = \sum_{k=1}^{M} y_{k,c}\log(\lambda_{k,c}) - \lambda_{k,c} + constant = \sum_{k=1}^{M} y_{k,c}(\theta_c^T X_k) - \exp(\theta_c^T X_k) + constant,$$

where $\theta_c = (\mu_c, \alpha_c^T)^T, X_k = (1, x_k^T)^T$.
Find the first and second order derivatives with respect to θ_c, we have

$$\frac{\partial LL}{\partial \theta_c} = \sum_{k=1}^{M}(y_{k,c}X_k - \exp(\theta_c^T X_k)X_k)$$

$$\frac{\partial^2 LL}{\partial \theta_c \partial \theta_c^T} = -\sum_{k=1}^{M}\exp(\theta_c^T X_k)X_k X_k^T$$

The recursive update is:

$$(\theta_c)_{i+1} = (\theta_c)_i - (\frac{\partial^2 LL}{\partial \theta_c \partial \theta_c^T})^{-1}(\frac{\partial LL}{\partial \theta_c})$$

Sequential Monte Carlo (SMC) Method

Sequential Monte Carlo (SMC) method, also called "particle filters", is a set of on-line posterior density estimation algorithms that estimate the posterior density of the state-space by directly implementing the Bayesian recursion equations.[6] SMC method uses a sampling (for estimation) and resampling (for next iteration) approach, with a set of particles to represent the posterior density. Each particle is assigned a weight that represents the probability of that particle being sampled from the probability density function. The resampling is also based on these weights. SMC is powerful since it can handle any state-space models and the initial state and noise distributions can take any form. SMC does not rely on typical assumptions such as linearity, so it can be widely used. One problem is that SMC may not work well in high-dimensional data. However, "dimensional curse" is a common drawback of many statistical procedures, not only SMC. SMC has become very popular since it was invented. In our project, the SMC algorithm can be summarized as following:
(1) Generate n samples $x_0^i \sim f(x_0)$. f can be any kind of initial distribution. Set $t = 0$.
(2) **Prediction:** Generate the prediction set using:

$$\tilde{x}_{t+1}^i \sim f(x_{t+1}|x_t^i), i = 1, 2, ..., n.$$

(3) **Update:** Compute the weights $w_{t+1}^i = f(y_{t+1}|\tilde{x}_{t+1}^i)$, and normalize them using $\tilde{w}_{t+1}^i = w_{t+1}^i / \sum_{j=1}^n w_{t+1}^j$.
(a) Estimate θ_{t+1} using

$$\hat{\theta}_{t+1} = \sum_{i=1}^{n} g(\tilde{x}_{t+1}^i)\tilde{w}_{t+1}^i.$$

(b) Resample from the set $\{\tilde{x}_{t+1}^1, \tilde{x}_{t+1}^2, ..., \tilde{x}_{t+1}^n\}$ with probabilities $\{\tilde{w}_{t+1}^1, \tilde{w}_{t+1}^2, ..., \tilde{w}_{t+1}^n\}$ n times to obtain the samples $x_{t+1}^i, i = 1, 2, ..., n$.
(4) Set $t = t + 1$, and return to Step 2.

Results Summarization

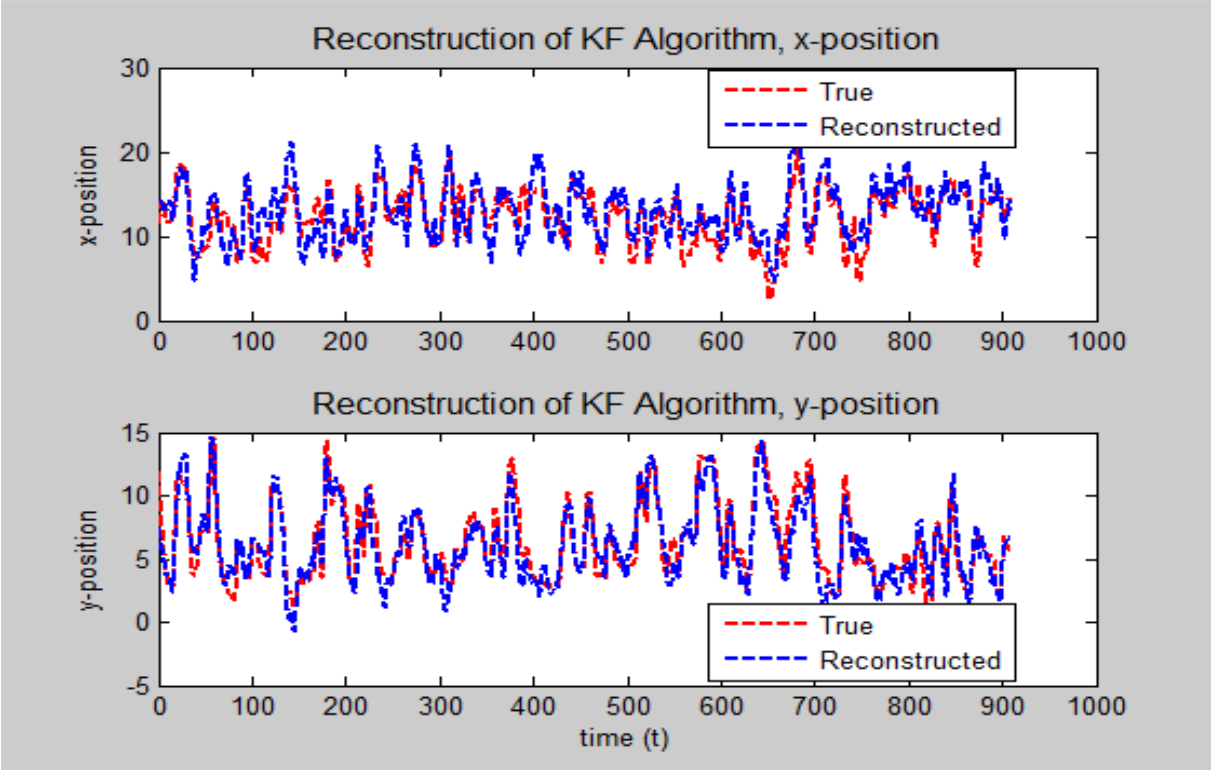

Figure 3: Reconstruction of Kalman Filter Algorithm, x and y positions

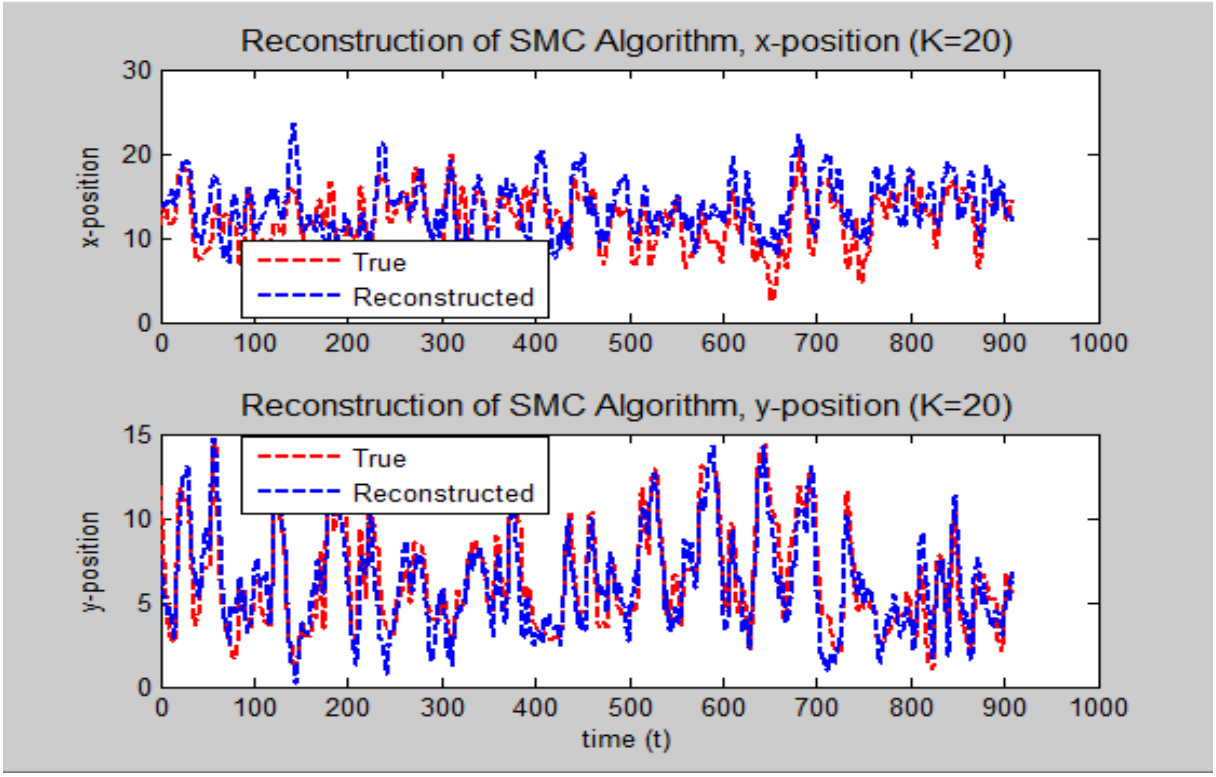

Figure 4: Reconstruction of Sequential Monte Carlo Method, x and y positions, K=20

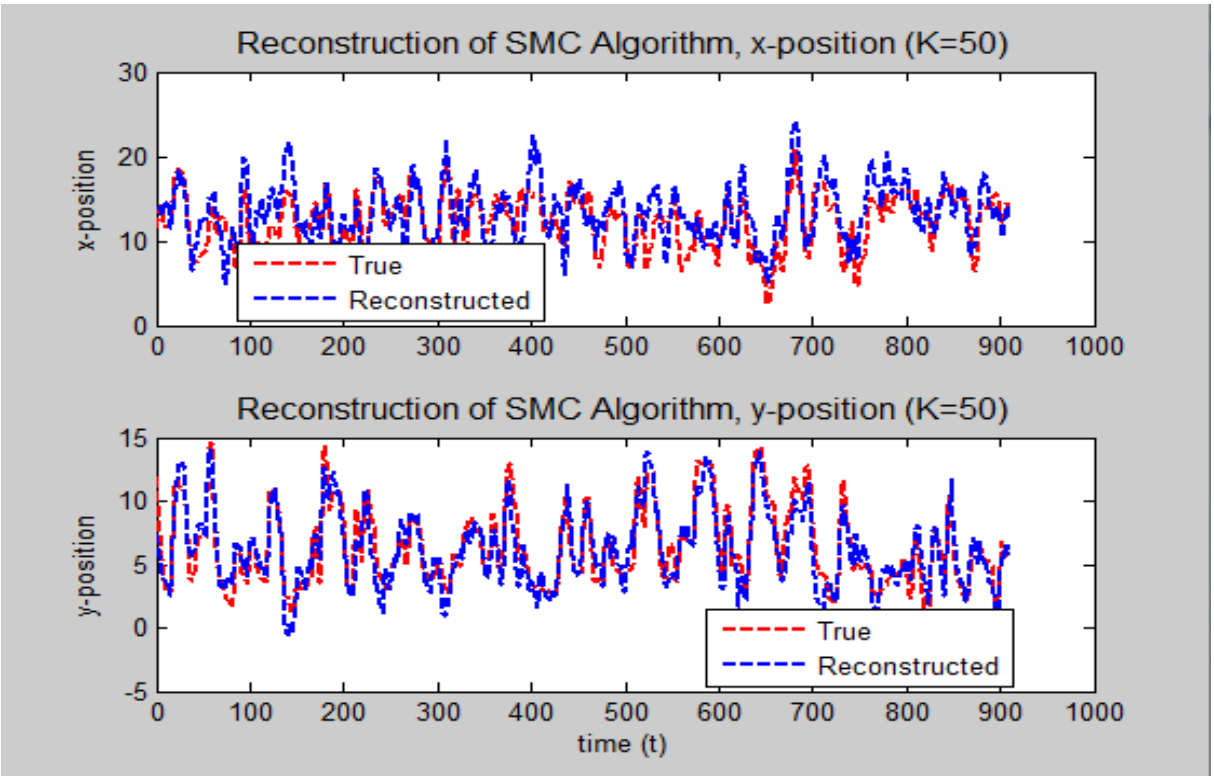

Figure 5: Reconstruction of Sequential Monte Carlo Method, x and y positions, K=50

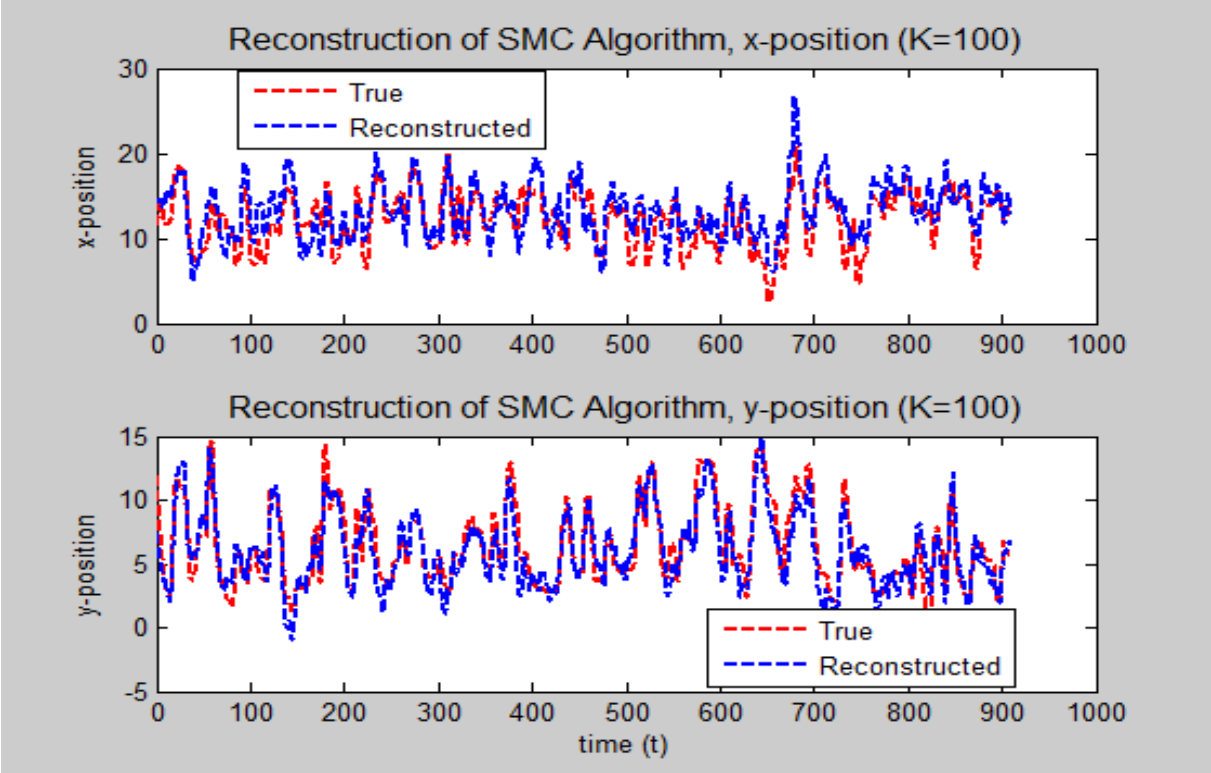

Figure 6: Reconstruction of Sequential Monte Carlo Method, x and y positions, K=100

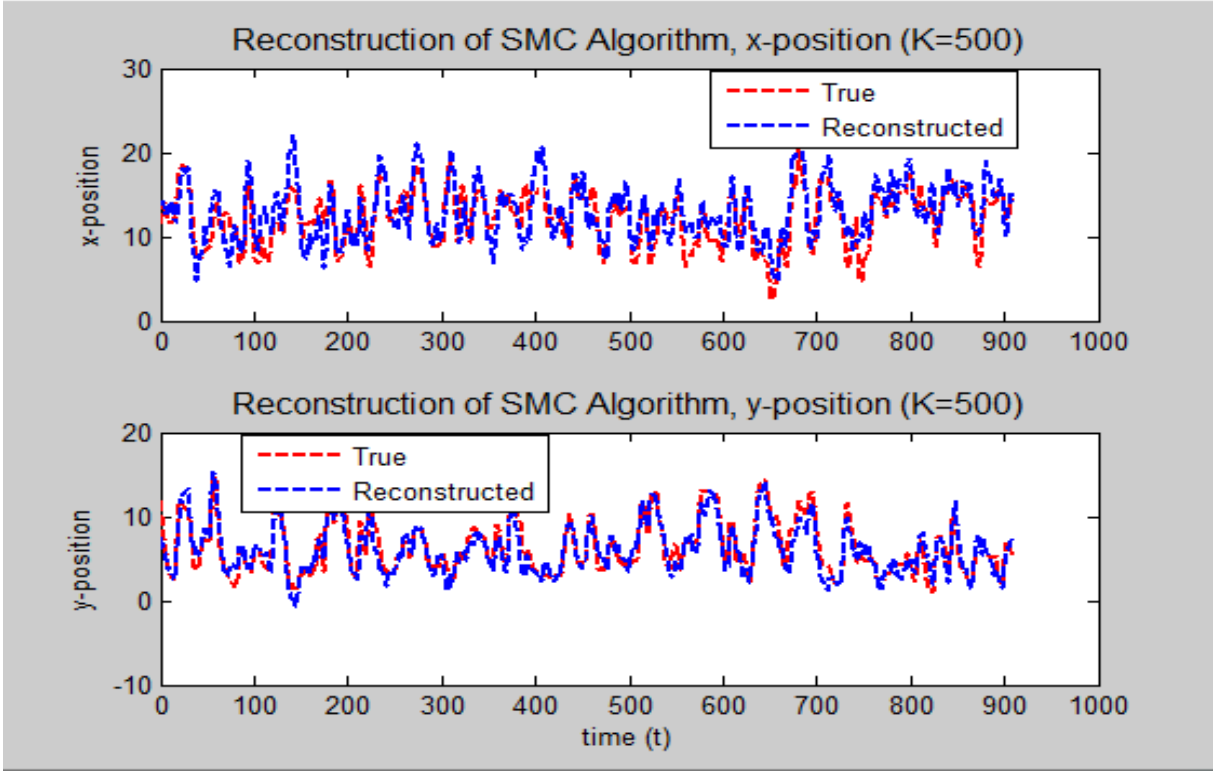

Figure 7: Reconstruction of Sequential Monte Carlo Method, x and y positions, K=500

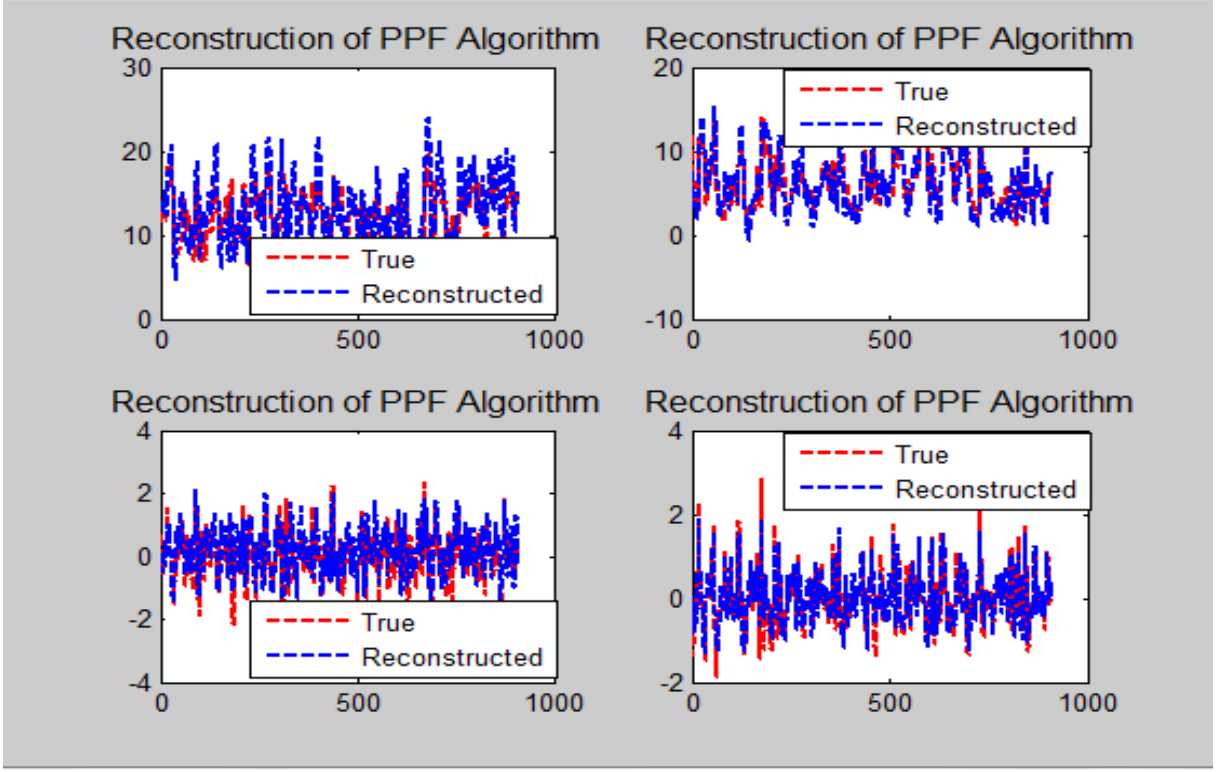

Figure 8: Reconstruction of Point Process Filter Algorithm, all components of x

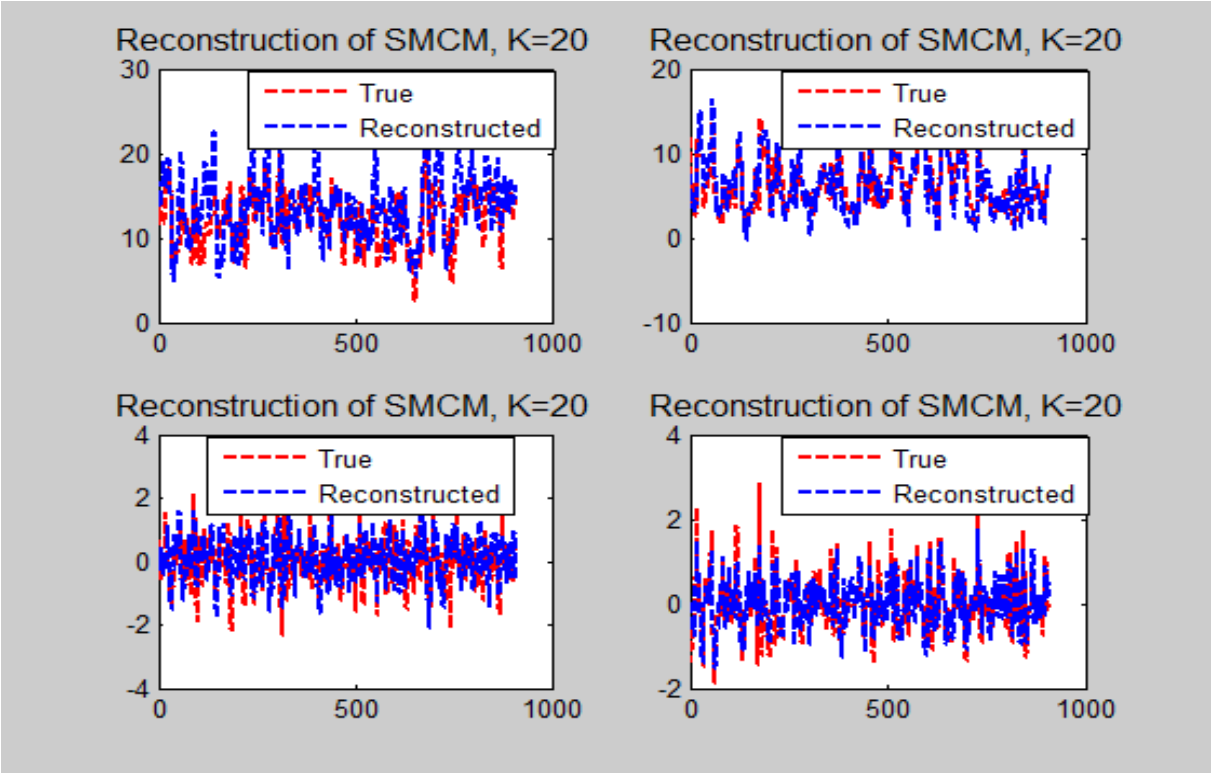

Figure 9: Reconstruction of Sequential Monte Carlo Method, all components of x, K=20

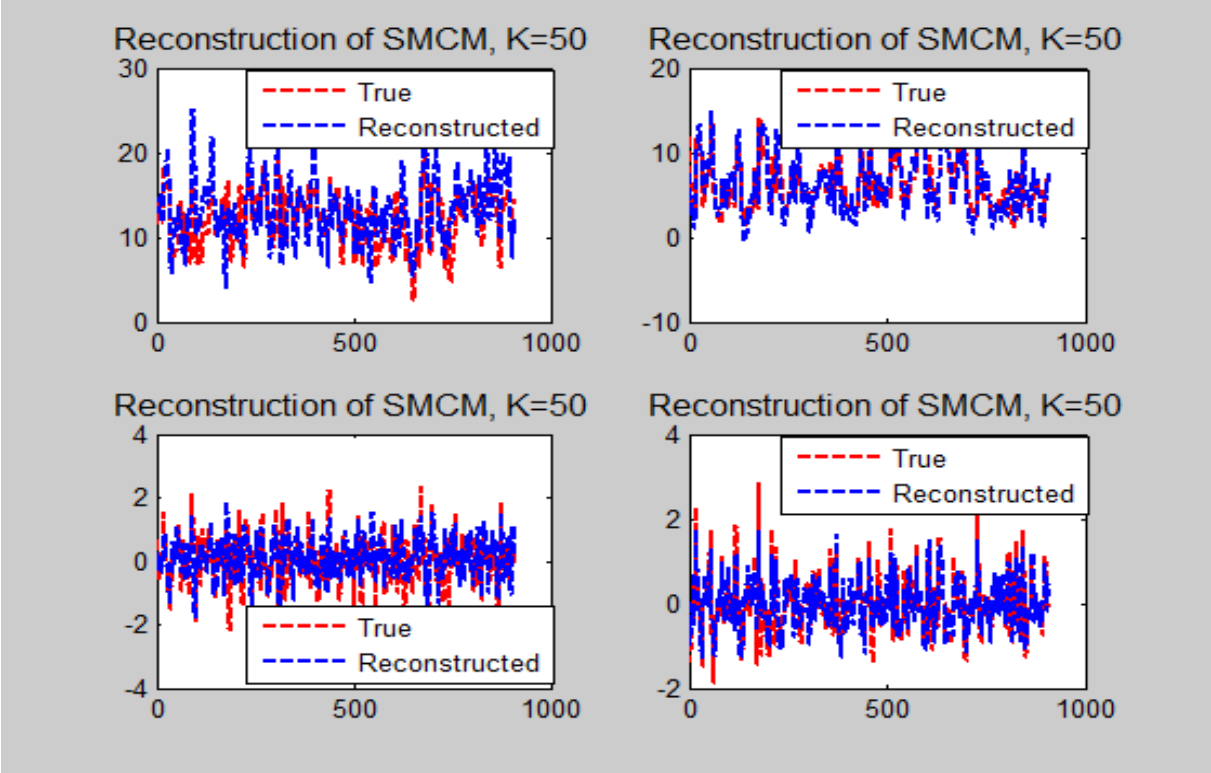

Figure 10: Reconstruction of Sequential Monte Carlo Method, all components of x, K=50

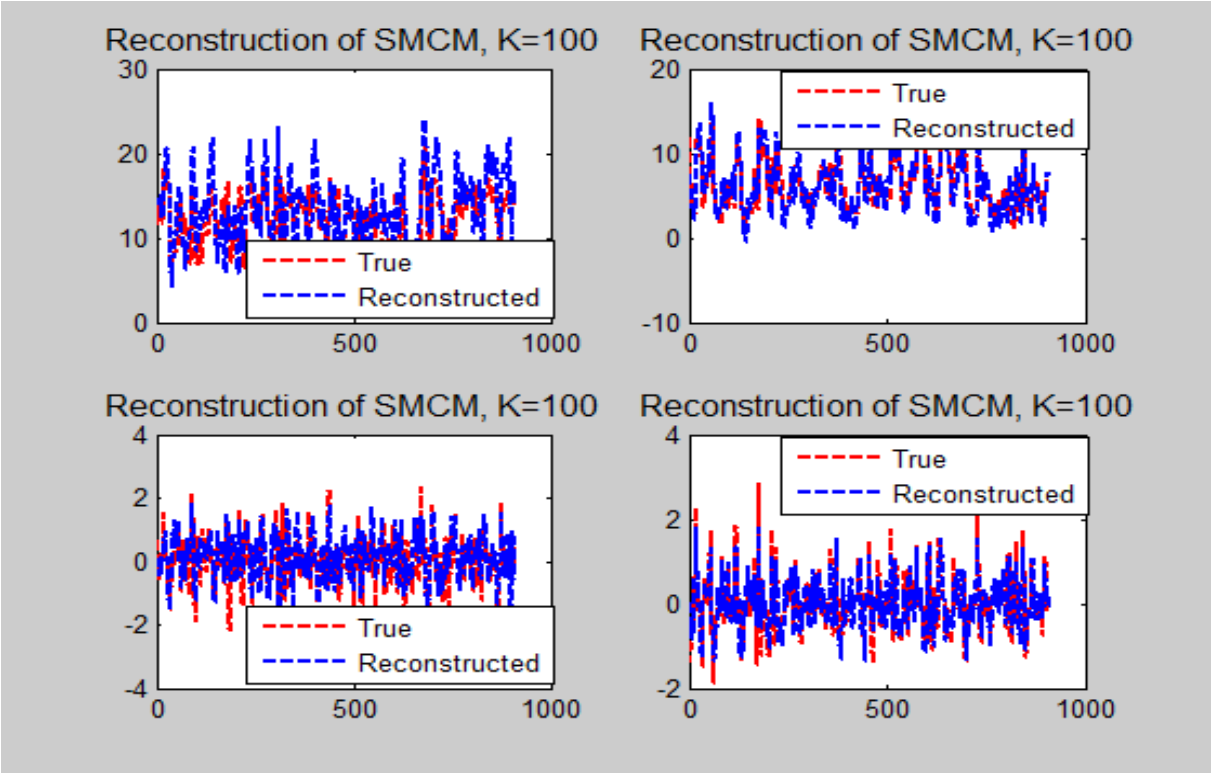

Figure 11: Reconstruction of Sequential Monte Carlo Method, all components of x, K=100

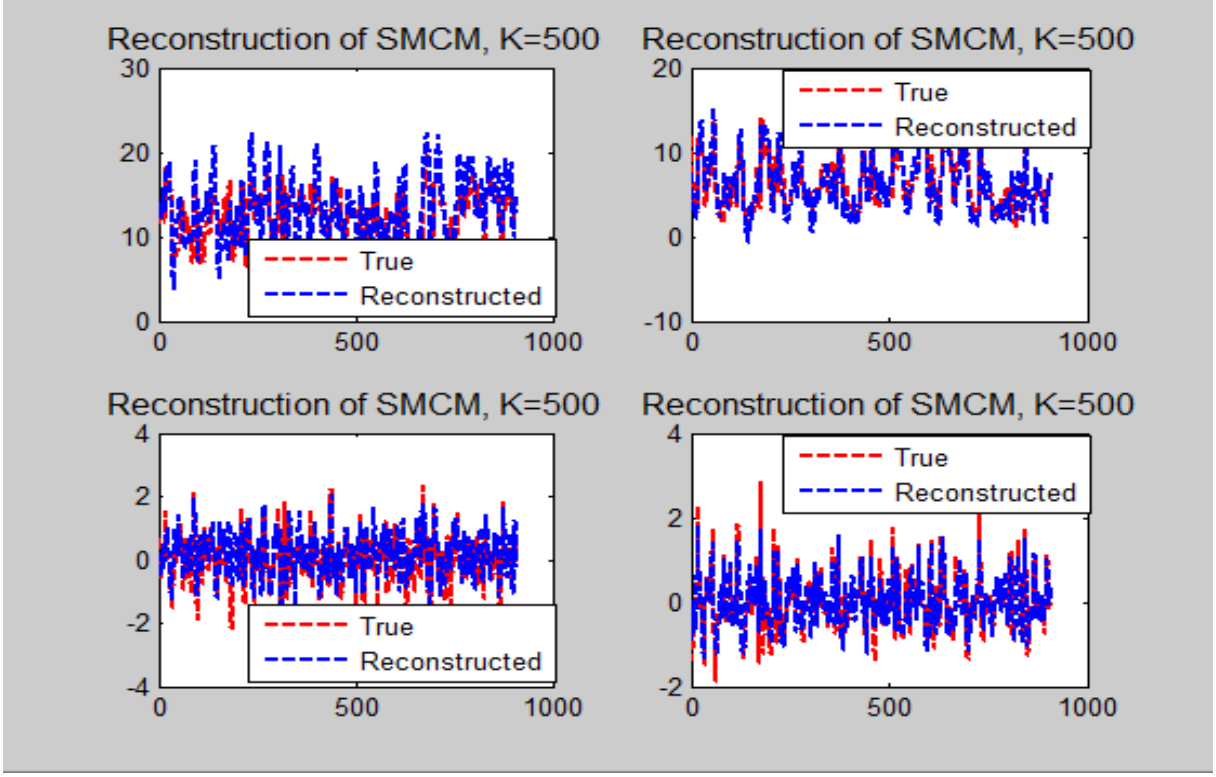

Figure 12: Reconstruction of Sequential Monte Carlo Method, all components of x, K=500

	Kalman Filter	SMC(K=20)	SMC(K=50)	SMC(K=100)	SMC(K=500)
x-position	0.6081	0.3314	0.4443	0.5537	0.5942
y-position	0.8534	0.8165	0.8471	0.8467	0.8546

Table 1: R^2 Error Comparison for Kalman Filter and SMC

	Point Process Filter	SMC(K=20)	SMC(K=50)	SMC(K=100)	SMC(K=500)
x-position	0.5598	0.1914	0.2775	0.3896	0.5404
y-position	0.8133	0.7894	0.7938	0.8079	0.8122
x-velocity	0.4751	0.2797	0.4181	0.4547	0.4823
y-velocity	0.7530	0.6841	0.7015	0.7309	0.7427

Table 2: R^2 Error Comparison for Point Process Filter and SMC

Comments and Discussions

The previous section summarizes all the table and plot outputs. As directed, in the first model (Kalman filter model), we reconstruct only the x-position and y-position. In the second model (Inhomogeneous Poisson process model), we try to reconstruct all the four components of x. Since the dimension of y is much larger than that of x, we are never concerned about the problem of lack of dimension. In both Table 1 and Table 2, the first column can be seen as the reference (we can call it "pseudo theoretical value") and we are interested in comparing the other columns with it. It is clear that, as an overall trend, the R^2 accuracy rate increases as the sample size of SMC n increases and gets very close to their "pseudo theoretical values" when $n = 500$. When $n = 20$ and $n = 50$, SMC still works relatively poorly. For x-position and x-velocity, R^2 changes evidently with the sample size, but for y-position and y-velocity, R^2 values look good even with small sample sizes. This may be due to the nature of the data (covariance structure, different effects of noise components, etc.). Another noticeable aspect is that the time cost of SMC increases in a higher order than linear as the sample size increases. For example, in the second model, $n = 500$ takes 2 minutes and 36 seconds to run but $n = 20$ takes less than 3 seconds. Depending on the power of computer, the time spent on computation can be different, but we always have to pay attention to the time consuming nature of SMC.

As a conclusion, when the model assumes linearity transition of the state-space, normality holds for the posterior densities, it is not hard to directly derive closed-form estimation formulae. In this case, closed-form solution should be the first priority. Even when the model does not have many good properties, we can still try recursive (iterative) algorithms other than SMC. When SMC has to be used, there is always a trade-off between estimation accuracy and time cost. So far SMC has become a very popular algorithm in computational statistics, but still more work can be done to strictly prove some asymptotic properties of SMC. Once its theoretical foundation becomes more solid, SMC can be a great finding that opens a new era in statistics.

References

1. Alan Agresti. 2013. *Categorical Data Analysis*, 3rd edition. Hoboken, New Jersey: John Wiley & Sons, Inc.

2. Amos Gilat. 2004. *MATLAB: An Introduction with Applications*. Hoboken, New Jersey:

John Wiley & Sons, Inc.
3. Geof H. Givens, and Jennifer A. Hoeting. 2005. *Computational Statistics*. Hoboken, New Jersey: John Wiley & Sons, Inc.
4. Webpage: http://en.wikipedia.org/wiki/Motor_cortex
5. Webpage: http://en.wikipedia.org/wiki/Neural_decoding
6. Webpage: http://en.wikipedia.org/wiki/Particle_filter
7. Webpage: http://www.sciencemag.org/content/340/6132/639/F1.large.jpg