COMP2043.GRP Interim Group Report

Sequential Monte Carl

Team 6

2017.12.7

Supervisor: Dr. Liang Dai

Member:

Hejia Qiu zy18720

Cong Liu zy19426

Zexi Song zy15777

Xiang Zhang zy18743

Kaiwen Zhang zy18672

Contents

1 Time line

09.28
As most of team members do not have teamwork experience of software development, we do not know how to design and develop a software in a group.

09.28
Solution: One of the team member(Hejia) have the experience during intime job in vacation, he shared how to use Git GUI and Github in the team, then we learned how to develop and version control.

10.13
This the first time we heard about Sequential Monte Carlo, so we do not have any acknowledge about it. In addition, as it is based on high level math knowledge, we are not available to understand it directly by reading books or novels with high level mathematical format.

10.13-10.18
Solution: We have former meeting with supervisor Dr.Liang Dai and he provide us some basic reading materials and give guidance on math. We keep learning with existing material and guidance, try to understanding how it works.

10.18
As we have the basic understanding about Sequential Monte Carlo, we still cannot make it in code/programming language and can not get output data to understand how parameters works and influence the whole system.

10.18-10.30
Solution: We find Matlab code and use different pairs of parameters to understand how them works in a visible way.

10.30
As the algorithm based on high level math knowledge and it is hard for everyone to understand every detail of algorithm. If we still trying to understand by everyone it would takes too much time for the group and result in the situation that no enough time for other works.

10.30-11.14
Solution: During the former meeting we decided to divide into two small groups one for algorithm(Cong,zexi) and another for software design(Hejia,Xiang,Kaiwen). Every small group can focus on their work and that is an efficient way. In detail, software design contains software positioning, requirement confirmation, programming language choosing and function UI design.

11.14-11.23
We have a meeting for confirming software detail and keep writing documentation and report.

11.23-12.01
During the report writing, as each member works for own part, there exists some parts conflict and some part call another´s data.

12.01-12.07
Solution: After finished everyoneś part, we communicate in meeting and solved this problem by teamwork.

2 Particle filter

2.1 The basis of Particle filter

It has been well known that a large amount of realistic data analysis tasks consist of estimating unknown quantities from some given observations. Indeed, if the data are modeled as a linear Gaussian state-space model, it is possible to derive a desired analytical expression to compute the evolving sequence of posterior distributions by implementing Kalman filter, besides, it's still reachable to develop an analytical solution for the data are modeled as a partially observed, finite state-space Markov chain. However, real data can be very complicated, typically involving elements of non-Gaussianity, high dimensionality and nonlinearity, which conditions usually preclude analytic solution. To deal with these sophisticated problems, many of approximation schemes such as Grid-based filters and Gaussian sum approximations have been proposed to solve these problems. One of these proposals named Sequential Monte Carlo(SMC) methods(is equivalent to Partical filter) are a set of simulation-based methods which can be used to compute the posterior distributions. Compared to other schemes, SMC is more flexible, easy to implement, parallelisable and applicable in very general settings. The first paper introducing what is now known as the bootstrap particle filter was published by Steward and McCarty (1992) but this papar has for some reason been more or less unnoticed. The derivation of the bootstrap PF as a solution to the nonlinear filtering problem by Gordon (1993) has made a significant impact. Kitagawa (1993, 1996) also introduced the PF at the same time. The adaptive particle filter was originally derived by Pitt and Shephard (1999) using auxiliary variables. Pitt and Shephard (1999) made use of two resampling steps in deriving the adaptive particle filter, whereas we only made use of one resampling step according to the derivation of

Carpenter et al.(1999).

2.2 The implementation of particle filter

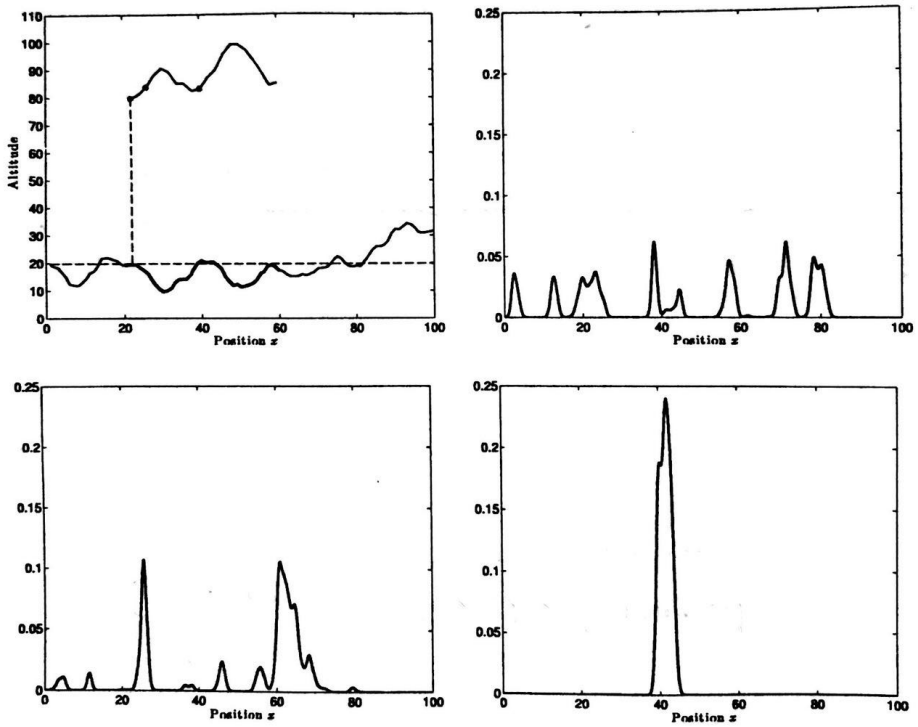
Particle filter can be implemented to solve a large number of practical application. A simple localization problem is one of the example. Localization is the process of determining the position of an object with various onboard sensors from which the aircraft can obtain the knowledge of its velocity and its altitude. A simple model of the aircraft motion is provided by a (discrete-time) integrator

$$x_{t+1} = x_t + v_t + w_t, \qquad w_t \sim \mathcal{N}(0, 5) \tag{1}$$

where x_t denotes the postion and w_t denotes process noise. The velocity can only help in establishing a relative position, which means that additional information is needed to solve the localization problem. One solution is to make use of a map of terrain elevation and downward-facing radar, measuring the distance between the aircraft and the ground. The corrsponding measurement equation is

$$y_t = h(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0, 1) \tag{2}$$

where y_t denotes the distance over ground measured by the radar, $h()$ denotes a look-up in the map encoding the terrain elevation and e_t denote the measurement noise. However, the measurement equation (2) is nonlinear and the function $h()$ is only defined in discrete points(according to the resolution of the map). But it fits the Particle filter(PF) perfectly, since it can deal with nonlinear functions and a varying number of possible hypotheses.



The result of implementing the PF with N = 200 particles is shown in the above picture. At time 1, the figute(upper left) illustrates the situation that the aircraft is flying at an altitude of roughly 80 m at position $x_1 = 22$ m,where the terrain elevation is 20 m. Now, compare these positions to the PDF $\hat{p}^N(x_1|y_1)$ provided by the PF after the first measurement has been received and processed(upper right plot). As we can see from the upper right plot, the PF provides several possible dominating modes representing the estimated position. One dominating mode is located in $x = 22$, which is corresponding with the upper right plot. When more measurement and information have been received and processed, PF can estimate the more accurate position. The principle illustrated in this example of using a PF to solve the localization problem by combining the information from sensors and maps has been successfully used to solve many different localization problems, including for example underwater vessels, ships, cars and people moving around in both indoor and outdoor environments.

2.3 The algorithm of bootstrap Particle filters

Here, we introduce the bootstrap particle filters which is the simple representation of Particle filters. The nonlinear filtering problem amounts to computing the filtering PDFs $\{p(x_t|y_{1:t})\}$ Sequentially in time. One princi-

ple solution is implemented by Forward filtering

$$p(x_t|y_{1:t}) = p(x_t|y_t, y_{1:t-1}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{3}$$

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})d_{x_{t-1}} \tag{4}$$

$$\hat{p}^N(x_{t-1}|y_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^i \delta_{x_{t-1}^i}(x_{t-1}) \tag{5}$$

$$p(x_t|y_{1:t}) \approx \frac{1}{p(y_t|y_{1:t-1})} \sum_{i=1}^{N} w_{t-1}^i p(y_t|x_t)p(x_t|x_{t-1}^i) \tag{6}$$

As you can see from the equation (4), to solve the $\{p(x_t|y_{1:t})\}$, the integral in (4) will need to be handled, which can be approximated using an importance sampler targeting the filtering distribution at time $t-1$. At time t=1, we can obtain a point-mass approximation according to (5). Inserting (5) into (4), then, inserting the result into (3). Finally, we get (6).
Now, it opens up for targeting $p(x_t|y_{1:t})$ with an importance sampler. Now we choose a similar type of mixture as proposal density. namely

$$q(x_t|y_{1:t}) = \sum_{i=1}^{N} w_{t-1}^i q(x_t|x_{t-1}^i, y_t) \tag{7}$$

There are a large amount of options when we decide to choose the proposal density $q(x_t|x_{t-1}^i, y_t)$
For example, In Bootstrap Particle Filters, the $q(x_t|x_{t-1}^i, y_t) = p(x_t|x_{1:t}^j)$ And $w_{t-1}^i = w_{t-1}^i$.
In Auxiliary Particle Filter, $q(x_t|x_{t-1}^i, y_t) = p(x_t|x_{1:t}^j, y_t)$ And $w_{t-1}^i = w_{t-1}^i p(y_t|x_{t-1}^i)$
.

$$w(x)\frac{\pi(x)}{q(x)} \tag{8}$$

Then, According to equotion (8), the weight function can be calculated by

$$w(x_t, y_t) = \frac{p(y_t|x_t)\sum_{j=1}^{N} w_{t-1}^j p(x_t|x_{t-1}^j)}{\sum_{j=1}^{N} w_{t-1}^j q(x_t|x_{t-1}^j, y_t)} \tag{9}$$

This completes the algorithm, for the reason that the weighted particles can be reused to approximate the filtering PDF next time.

2.3.1 bootstrap PF in matlab simulation

```
function [xhat] = PF(f,h,pe,Q,P0,M,y)
n = size(P0,2);
x = sqrtm(P0)*randn(n,M); % 1. Initialize particles
for t = 1:100
  e = repmat(y(t),1,M) - h(x);  % 2. Calculate weights
  q = feval(pe,e);              % The likelihood function
  q = q/sum(q);                 % Normalize importance weights
  xhat(t) = sum(repmat(q,n,1).*x,2);
  ind = resampling(q);      % 3. Resampling
  x = x(:,ind);             % The new particles
  x = feval(f,x,t)+sqrtm(Q)*randn(n,M); % 4. Time update
end
function [i] = resampling(q)
qc = cumsum(q);     M=length(q);
u = ([0:M-1]+rand(1))/M;
i = zeros(1,M);     k = 1;
for j = 1:M
  while (qc(k)<u(j))
    k = k + 1;
end
i(j) = k; end;
```

The three first input arguments to the PF function are the model equations f,h and the likelihood function pe. Q represents the covariance matrix, while initial state covariance matrix $p0$, the number of particles M, and finally, y is the measurements y.

Test file for PF

```
M = 1000;
P0 = 5;
Q = 10;
R = 1;
pe = inline('1/(2*pi*1)^(1/2)*exp(-(x.^2)/(2*1))');
f = inline('x./2+25*x./(1+x.^2)+8*cos(1.2*t)','x','t');
h = inline('(x.^2)/20');
% Number of particles
% Initial noise covariance
% Process noise covariance
% Measurement noise covariance
```

```
x(1) = sqrtm(P0)*randn(1);  % Initial state value
y(1) = feval(h,x(1)) + sqrtm(R)*randn(1);
for t = 2:100                % Simulate the system
  x(t) = feval(f,x(t-1),t-1) + sqrtm(Q)*randn(1);
  y(t) = feval(h,x(t)) + sqrtm(R)*randn(1);
end
xTrue = x;
xhat = PF(f,h,pe,Q,P0,M,y);
plot(1:100,xhat,'b--',1:100,xTrue,'r');
xlabel('Time');
```

The system that will be implemented with PF to estimate the states is

$$x_{t+1} = \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8cos(1.2t) + w_t, \tag{10}$$

$$y_t = \frac{x_t^2}{20} + e_t, \tag{11}$$

3 Software positioning

The software is positioned as a cross-platform teaching software. Targeted users are teachers and students who have a teaching and learning requirements for particle filtering and related algorithms.

For teacher users, the software needs to provide the display function. Teachers can display the algorithm they want to teach by selecting the algorithm type and data source. In order to facilitate the operation, a sample of the input should be stored in the software and the visual output can be obtained after running. And teacher can modify the parameters to re-run then get comparison for showing the impact of parameters change on the output of the algorithm.

Similarly, for student users, the software should provide the access to select the type of algorithm and input data, allowing students to choose how the algorithm is to run and, at the end of the run, have the same ability to modify parameters and make comparation, as the teacher user does. For new student users, there should be teaching and help documentation, to explain the algorithm and guide of the software.

As student users may need to submit the output, homework or other possible needs, the software should provide data and image export function.

In addition, for someone use this program to learn particle filter, help documentation and algorithm explaining are required. With this program, they can do a step-by-step self-study then make some progress.

As the software users may use different types and versions of the operating system, the software is required to support Windows 7 and later Windows operating system, mainstream Mac OS system and mainstream Linux system.

Future versions of the software may include support for mobile platform using Android or iOS, but due to the limitation of developing cost and time, this requirement is not considered in software design and language selection.

4 requirement

The software is a free education-assisting software and it is mainly used by college students and teachers. In order to meet the needs of users and provide good user experience, this software should satisfy a large number of significant requirements. In next parts, there are two types of requirements and each specific requirement will be analyzed and explained in details.

4.1 Functional requirements

1. There should be some way to illustrate how to use this software and explain the principle of algorithm.
2. The software should provide the function of selecting different algorithms.
3. For users with no other specific data, the software should provide some existing sample data to them
4. The software need the function of prompting and checking the format of the file when users import specific sample data.
5. The software should use the algorithm which is chosen by user to deal with the import sample data and generate the outcome image.
6. The software should provide modifiable parameter list and some simple explanations of each parameter.

7. The software should have the function of modifying parameters which is set as default value.
8. The software should provide the output function to export images, data, or both according to the user's choice.
9. The software should permit user to combine two images together to compare the differences between them.
10. The software should provide the function of clearing the current data and results if users want to import another sample data.
11. The software should provide a function of putting the software window into the top.
12. When the software crashes for some reason, there should be some way to restart it.
13. The software need some function of checking update and submitting bugs by users.

4.2 Non-functionalrequirements

1. The window size of the software should be suitable for demonstration
2. The user interface should be convenient for user activity.
3. The running time of using the algorithm to deal with the sample data should be as quick as possible.
4. The software needs to run on major platforms and operating systems.
5. The software should be convenient for update and improvement.
6. The software require appropriate software capacity to be convenient for users download and use in a short period of time.
7. The software is free, it is better that the development and maintenance costs should be properly reduced

5 Language choice

5.1 Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

5.1.1 Advantage

As a cross-platform language, Java is conducive to cross-platform development and application of the software. It can easily and effectively solve the problem of multi-platform compatibility. As major user, students and teachers can run the program on different operating systems, such as Windows, MacOS and Linux.

With the optimization for Java by ARM architecture, the software can even run on mobile platform, like mobile phone or pad.

Java's design focuses on objects and interfaces. As an object-oriented language, state variables and methods are encapsulated in objects, enabling modular development and information hiding. Its also convenient for work assignments and communication in the development team. Everyone only need to provide interface document but bot need to understand others all code. While debugging or updating, its also easy for programmer to change his own part without influencing others.

In Java program, programmer can simply add new methods or class without affecting other parts. As for facilitating this program and adding features, such as adding a new algorithm type or parameter, its also available.

5.1.2 Disadvantage

Lack closure, so cannot use functional programming, while in this program theres much process in functional style.

Java program depends on the Java virtual machine, not directly execute the machine code, so the speed is not very fast.

5.2 Front-end development

The use of front-end technology to develop applications is a new popular lightweight direction based on Node.js technology. Programmer can use Electron or NW.js framework to package front-end web pages and open source browser kernel, like Chromium, then make a program.

5.2.1 Advantage

In this way programmer can use H5 and WebGL to achieve awesome dynamic effects easily.

Development tools such as front-end frameworks (like ECharts, jQuery or

Angular) and vue-cli scaffolding can be called to significantly reduce development costs of the software.

Support cross-platform development, as long as this operating system support the browser kernel.

5.2.2 Disadvantage

Compilation of framework and browser kernels may increase load time and software capacity. And its not easy like Java for co-work and program updating.

5.3 Others

As for .NET and C++ , they are not suitable for cross-platform development, so we do not choose them. Similarly, as the limitation of co-work and ability, python, C, Haskell and C are not chosen as a considered one.

6 Functions

The software is a teaching software used to illustrate how the Sequential Monte Carl algorithm works by using visualization techniques. Therefore, all the functions are designed to lead students to understand the process of algorithm and the affect of arguments. Furthermore, as a teaching demonstration tool, making users comprehend it quickly is more significant than finishing tough missions through complicated operations. The following part will declare the functions designed according to user requirements and user-friendly principle.

6.1 Introduction

Here is a brief introduction of how user can use the software. All functions will be explained in details in next part.

After opening the software, the first step is "**choose an algorithm**" to select one algorithm from Sequential Monte Carl algorithms which have different approaches to deal with data.Afterwards, Users can import files in the software or using exist sample data. The result will be showed as both image and data. Otherwise, users can choose to change the value of some argument and run it again. The software will show the comparison of these two performances which can assist user to understand the meaning of arguments directly.

6.2 Functions Details

6.2.1 Common Functions

There are a few functions which softwares need commonly. Below is the common functions in the software.

- **Restart**

  When some error occurs or some setting changes, user can restart the software by clicking "Restart" button.

- **Exit**

  Exit the software.

- **Manual**

  Manual is the instruction of the software. The content contains the introduction of software ,information of version updating and operation guide. General problems can be solved by checking manual.

- **Document of File Format**

  Imported file will be extracted by uniform approach. The document provides users with correct format of file and required information that imported files need to contain.

- **About Us**

  Provide the personal and contact information of developers. If any problem occurs in use, user can contact developers to fix it.

6.2.2 Specific Functions

- **Starting page**
  - **select a specific algorithm**
    Sequential Monte Carl have amount of methods to process data in resampling procedure Thus, a specific algorithm which its brief explanation will be displayed can be selected by user in the first step. At the beginning, the user get access to choose one of two algorithms including Bootstrap Particle filter and Auxiliary particle filter. Different algorithm have different proposal distribution density.
- **Main page**
  - **Import file**
    The software allows user to import correct-format file. The contents stored in the file can be extracted as data which offer to algorithm.

– **Preset sample data**

  Several sample data are preset in the software. This function is use to satisfy the requirement that user only need a simple presentation without specific data. The arguments of sample data can be modified as well.

– **Run algorithm**

  Algorithm allows to use the data extracted from import file or sample data. After running the algorithm, the result is showed by two parts: Images and data. In addition, only two results based on the same original data can be showed in the screen simultaneously.

  * Images
    There will be two lines in the image. One is the truly line created by offered data, another is created according to the inference of Sequential Monte Carl algorithm.
  * Data
    The crucial data of Sequential Monte Carl algorithm will be showed in the scene.

– **Clear**

  Clear is to clean previous data and image showed in the scene instead of imported file. Thus, users can run the same file multiple times to observe the image and data.

– **Modify value of arguments**

  This function allows user to modify part of data which can affect the performance of result. New result will be produced at the same page. According to modify value of arguments, the influence of each argument in the algorithm can be virtualized directly. For example, the user can adjust the value of the number of particle that will be simulated. Then, the user can determine different resampling algorithm.

– **Compare with images**

  To display the different performance caused by different data, two images can be exist. These two images can be showed separately or combined in one image.

– **Export**

  All the results can be exported by users. Because of the two forms of contents, export will be divided into two parts.

  * Images
    Both single image and combined image can be saved in PNG format or xxx.

* Data
  Data can be saved in xxx format.

7 UI Design

According to Alan(2006), all of the designs are designed for consumers. For this software, The design is for the main users: teachers and students which need to understand Sequential Monte Carl algorithm. Favourable user experience is the ultimate goal. Therefore,according to the features and purpose of the software, there is a simple design of user interface to represent the using of the software. It will be divided into two parts to describe the UI design. The first part is general design which includes the window and frame design. Each individual page will be explained as the next part.

7.1 General Design

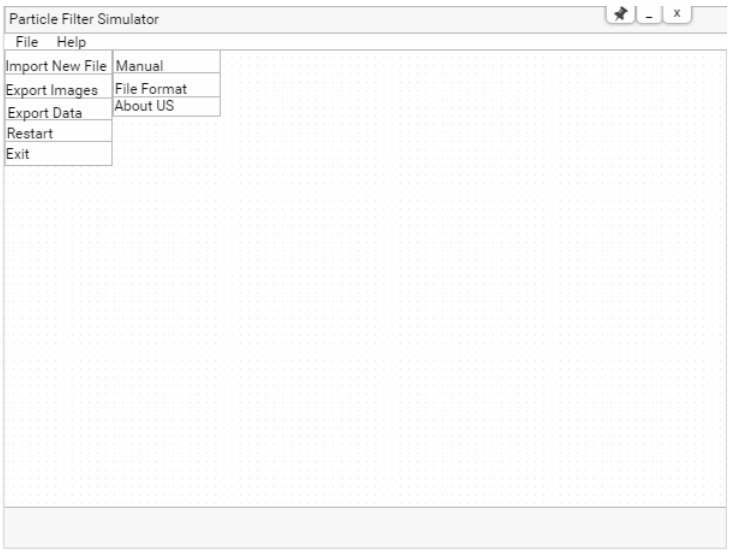


Figure 1: Window

The software will be performed in a fixed-size window which can satisfy mainstream screen size. As shown above, Users can choose to minimize the window or stick it. Stick feature can guarantee the window keep showing on the screen and not hinder the operations on other softwares.

In addition, there is a menu bar on the top of window contains two menu: File and Help. Several functions are placed in these menu as showed

for convenience of getting these functions. However, they are not available in the whole process. It will be explained in next part.

7.2 Individual pages

7.2.1 Starting page

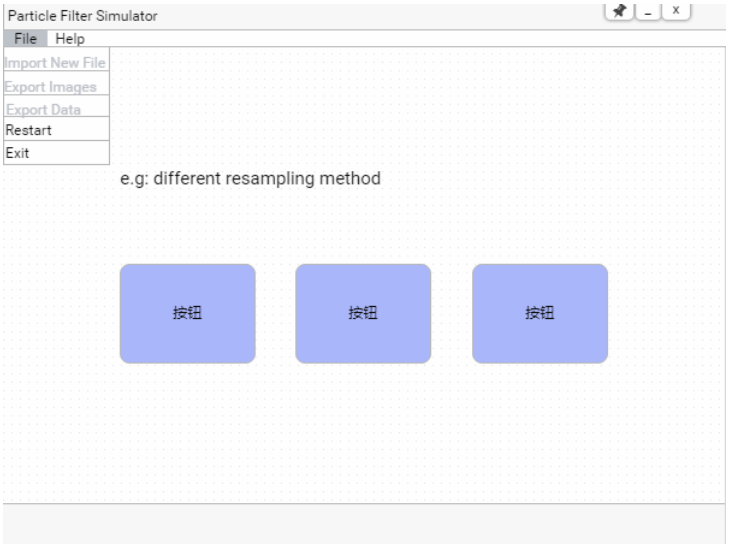


Figure 2: first page:File

This page is to choose a algorithm according to click buttons in the middle of page. Especially, because the algorithm have not been chosen and no data can be exported, Only restart and exit function can be clicked by user in the File menu. Once users have their selection of algorithm and click the button, it will be jump to the next page automatically.

7.2.2 Main page

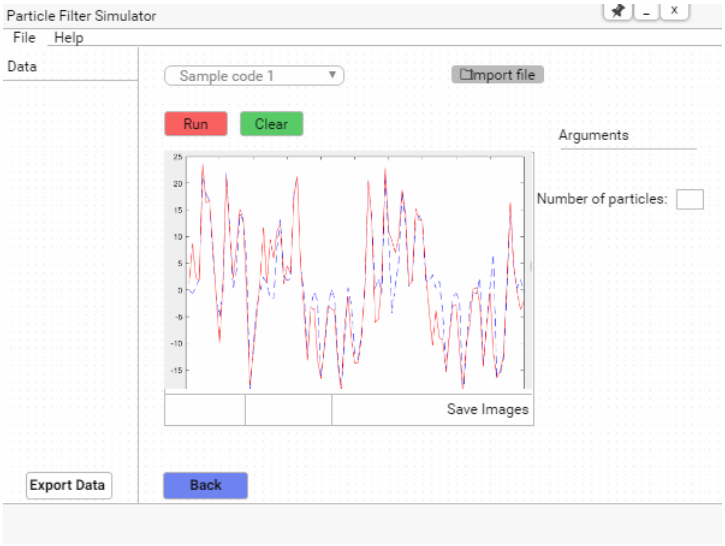


Figure 3: Main page

It is the main page of the software. All the functions used to demonstrate Sequential Monte Carl algorithm are placed to this page.

The left side is separated to show and export data which are significant to run the algorithm. It is a choicebox on the top of right side. The checkbox contains amount of sample data can be chosen by user. If user want to run the algorithm with their data, they can click "Import File" to upload. Image box is in the middle of page, so that users can observe images comfortably. Four buttons:"Show image 1","Show image 2","Compare" and "Save" are on the bottom of image box. Thus, switch different images and compare them can be easy to use. Run and Clear buttons are closed to the image with apparent colors. At right of the image box, users can set value of some arguments by inputting value in the textbox. For example, number of particles in figure 3. Besides, this page allows users back to previous page to make a different choice by "back button.

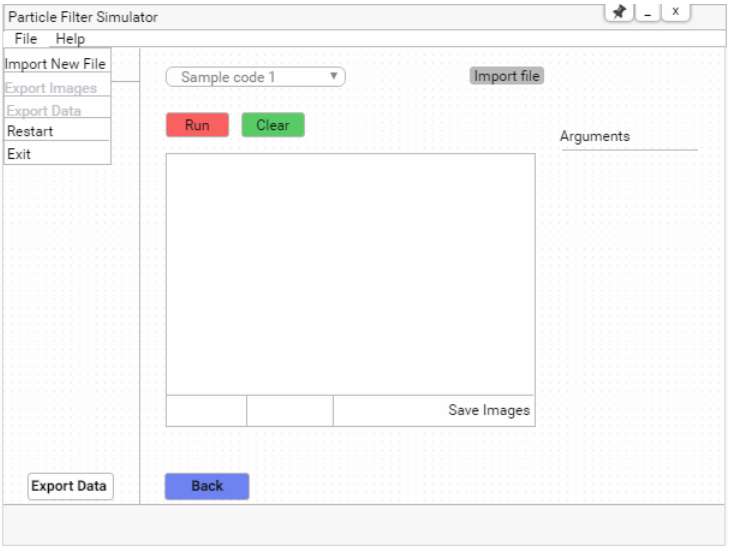


Figure 4: Before running

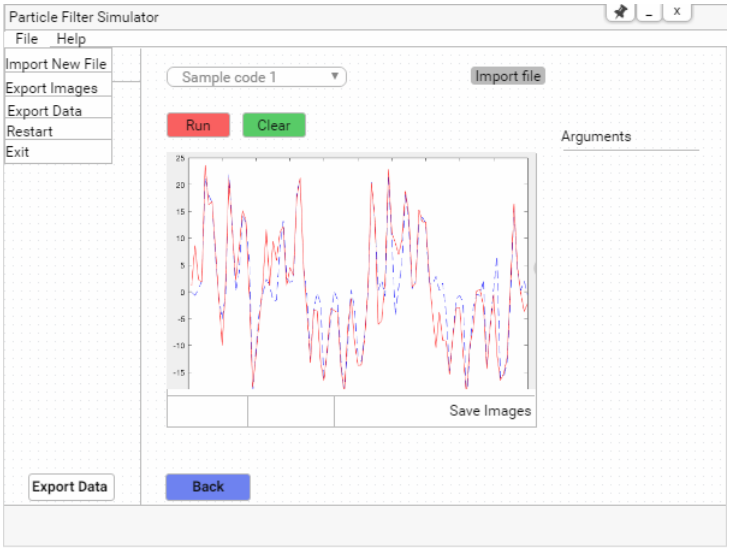


Figure 5: After running

8 Future plan

This software could be released in the next few months. The main part of the software will be written by Java. Development team could be divided

into several groups to finish the task step by step. This part will introduce the future plan of finishing this software.

First, at the beginning of January, the UI design group should build the framework and user interface by using the JavaFX and other resource. Then, before the end of the January, main function with the algorithm should be finished by the algorithm research group and the functional design group should finish the half part of the software function implementation.

During the February, the functional design group should finish the rest of software function implementation and the software testing group should finish the unit testing and debugging. At the end of the February, a demo version of the software should be released. The software testing group should start user testing at the beginning of March.

After receiving user feedback from the user testing, the software testing team should start to deal with the software performance and fine-tune implementation. the rest of time until to 19th April could be used to finish the final reports of each development team members. Final report and complete software should be finished before 19th April.

1st January - 15th January
Build the framework and user interface.

16th January - 31st January
Main function with the algorithm should be finished. Finish the half part of the software function implementation.

1st February - 15th February
Finish the rest of software function implementation.

16th February - 28th February
Start testing and debugging. A demo version of the software should be released.

1st Match - 15th Match
Finish unit testing and user testing.Finish to deal with the software performance and fine-tune implementation

16th Match - 19th April
Finish final report and complete software

....

9 Reference

Cooper, A. (2004). p.p1 margin: 0.0px 0.0px 0.0px 0.0px; line-height: 19.0px; font: 13.0px 'Helvetica Neue'

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. In: Microsoft Research Ltd.

Schon, T. B. (2006). Estimation of Nonlinear Dynamic Systems Theory and ApplicationsIn. Sweden: Linkopings Universitet.

Thomas B., S., Fredrik, L. (August 23, 2017). Learning of dynamical systems-Particle filters and Markov chain methods, 129. Retrieved from

Wekipedia(2017) Java[online], Available from$\langle https://en.wikipedia.org/wiki/Java$