The Impact of Varying Sample Size of Particle Filtering on Estimation the Pose of 2D Objects: An Experimental Report

IFN680 | Assignment One

Particle filter search

group 9: Linni Qin n9632981 Jiuzheng Rong n9814434

2017

**Table of Contents**

1. **Introduction1**

1.1 Background Statement1

1.2 Brief description of Particle Filter Algorithm1

1.3 Purpose of Report2

1. **Experimental Methodology3**

2.1 Experimental Environment3

2.2 Experimental Approach3

2.2.1 Finding particle budget3

2.2.2 Balancing the number of generation and the size of the population 4

1. **Experimental Results4**
2. **Conclusion6**

Reference

1. **Introduction**

**1.1 Background Statement**

While applying the machine learning algorithm to detect the pose of targeted objects, edge-based method requires cheaper computation than other pose estimation approaches does, such as keypoints-based approaches (Frederic, 2017, p. 1). In the context of this report, a variant of particle filter algorithm is implemented efficiently as the searching method to estimate the pose vector of a 2-dimensional geometry shape. This particular particle filtering algorithm will be explained in the next paragraph. In the meantime, the targeted object is the larger triangle displayed in the left image **imf** of *Figure 1* below. Just like other patterns, it has 4 degree of freedoms of pose vector, which are x cords, y cords, theta and scale. The other known factor about this triangle is the distance and location of the pixels for its three edges as indicated in the right float image **imd** of *Figure 1*.

|  |  |
| --- | --- |
|  |  |
| *Figure 1: imf* | *imd* |

**1.2 Brief Description of Particle Filter Algorithm**

*“# Particle filter search   
Initialize population* ***W*** *with random guesses of pose vectors****Loop*** *until computational budget exhausted* ***evaluate*** *the cost* ***C[i]*** *of each* ***W[i,:]******re-sample*** *the population according to* ***exp(C[i])******mutate*** *each new individual* ***W[i,:]*** *by adding some noise”  
 (Frederic, 2017, p. 3)*

Considering the particle filter algorithm applied in this assignment, it generates initially a collection of hypotheses (population ***W***) that are called particles as well from a random distribution. Each of them (***W[i,:]***) has four state variables as discussed above, including x cords, y cords, theta and scale. The initial particles with random poses are displayed all over the image in green colour as shown as the *Figure 2*.

|  |
| --- |
|  |
| *Figure 2: A random collection of W* |

Then, the program will iterate until the number of generation runs out. During each loop, the distance of certain particle to the target vector (100, 30, pi/3, 40) of the larger triangle is recorded as ***C[i]*** thatdenotes the cost of the particle ***W[i,:]***. Following obtaining the cost, the program generates new particles as the stated the re-sample procedure. However, they are not randomly distributed over the entire map starting from this step. They are re-generated based on previous particles with respect to their cost. After the re-sample, mutation is proceeded on each individual with equal probability with moving 1 degree in different direction to avoid the convergence being stuck at a local minimum. The best solution (*self*.best\_w), the improvement of the best weight (*self*.best\_cost) and the convergence effect at the last step of iteration will be the essential factors to evaluate the particle filtering. Those three factors look similar as the patterns shown in *Figure 3*.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Figure 3: Best solution* | *Best cost* | *Convergence* |

**1.3 Purpose of Report**

This report is going to explore the performance of the above three outcomes with varying finite sample sizes (*self*.**W**) by balancing the number of iteration and the size of the individual in each generation. The approaches of testing will be discussed in the following section two (Experimental Method) and section three (Experimental Results) will demonstrate the related effects. Finally, the conclusion with a rational suggestion will be offered about the optimal balance of the total population size and the number of generation to maximize the effectiveness and efficiency of the program.

1. **Experimental Method**

**2.1 Experiment Environment**

The following computing environment is for deploying the particle filtering and the relevant testing experiment.

*Operation Systems: Windows 7 Enterprise System x64-based  
CPU: Intel® Xeon® CPU E5-2687W v4 @ 3.00GHz   
Memory(RAM): 4.00 GB   
Programming and diagnosing Tool: Python 3.6.0 and Spyder 3.1.2*

**2.2 Experiment Approach**

Due to the random sampling of individuals in selection, the population size is set by repeated trial without any academic theories to guide the choice of the size. Manually setting the number of iteration and the size of population needed for each individual step is used until the best solution, the best cost pattern and the best convergence could be found. The relevant codes can be found from the defined function ***test\_particle\_filter\_search()*** in the enclosed Python file ***my\_submission***.

Code for setting the size of individuals for each generation: ***pop\_size = ##***Code for setting the number of the iteration*:* ***Lw, Lc = pop.particle\_filter\_search(##, log=True)***

(Meanwhile, the computational time is evaluated as well so that the extra efficiency value can contribute to the overall performance evaluation. The relating ***date.time*** code is added within the function ***test\_particle\_filter\_search()*** as well, which can be referred to the enclosed Python file ***my\_submission***.)

**2.2.1 Phase one: Finding the particle budget that is suitable for this problem**

Following the policy of varying the size of population from small to large, we started from the 5 individuals and it was incremental by (int i \* 5 ; 1 <= i). The generation was increased with interval of 10 until reaching the maximum 200 times of generation. The reason we did not start from less than 5 individuals is that the cost comparison requires at least two individuals. Furthermore, the minimum cost found before the execution of the program might be the best cost within the small size of population. There is high probability to get a blank cost image.

Considering that this step is for detecting the particle budget, to enhance the test efficiency of detecting the particle budget, we added a loop function with parameter 5 to automatically execute the test of evaluating the cost element solely. Then the suitable particle budget can be analysed from the visible cost images that will be discussed in the third section (Experimental Results). Other performance factors will be discussed in phase two.

**2.2.2 Phase two: Balancing the number of generation and the size of the population**

After finding the particle budget, the test scope is narrowed down. We enlarged the testing times to 30

1. **Experimental Results**

Results for finding the particle budget.

If a small W is chosen, there is a high risk of poor performance that causing high loss and non-convergence when reach the maximum of the iteration. The problem climbing is that it is easy to get stuck with local maximum. One idea to overcome this problem is to set a bigger size of population and the larger number of generation.

(inset some figure and table about the small size of population)

Due to the increase in filter size, the cost drops with increasing filter efficiency (XXX, XXx, respectively). Rates ranged from XXX to XXX decreased with the rated efficiency of the filter performance. Effective particle loss rates generally increased as both particle size and rated filter efficiency increased.

(inset some figure and table about the medium size of population)

(inset some figure and table about the large size of population)

Loss rates are divided into six particle size as indicated by the dashed vertical lines. Boxes represent the center distribution of the loass, whiskers represent the outlier values. (cost distribution for each experiment can be plot with using boxplot)

1. **Conclusion**

In general, larger particles (> 16000 particles) resulted in both lower losing rates (cost) and higher computational efficiency. Small size of population……

This report recommends. ….

Manually setting the number of iteration and the size of population needed for each individual step is used to explore the performance of the overall accuracy for pose detection.

Consequently, the most efficient parameters to explore the best solution will be a compromise among the size of the population, the number of the iteration and the computation time. The visible outcomes at the least iteration about the best solution and the final position of the sample convergence will be compared in the section of Experimental Results.

1. Generation值过于小，会出现图中的情况
2. 寻找最好的population和generation的比例，先设定其中一个值（population）为定值，去改变generation的值去寻找最佳比例。
3. 当比例找到之后，通过改变整体的size去找最有效的（最小的）值。 40/50 的效果和400/500的效果相差不大，前者的效率要高于后者。
4. Noise对particle的影响
5. The data framework is used for the testing are listed as the table below. For each of the total particles, xxx experiments were performed. Each combination of the size of the population and the number of the iteration was replicated xxx times to obtain multiple key values for later averaging.

Reference:

Frederic, M., (2017). 2017\_IFN680\_assignment\_1. Retrieved from https://blackboard.qut.edu.au/bbcswebdav/pid-6983471-dt-content-rid-9254318\_1/courses/IFN680\_17se2/2017\_IFN680\_assignment\_1%281%29.pdf