IFN 680 Assignment 1

25 September 2017

Students:

Awal Singh n9645667

Ray Bastien Mayol n9588566

Phuoc Pham n9633561

Introduction

Automated systems, including robots, are quickly becoming integrated into our everyday lives. What started out as mechanical arms set on non-moving platforms in production lines, became moving units in closed locations such as Amazon warehouses. Machines are now fully mobile, working in the open unstructured world, like Boston Dynamics robots and driverless cars.

Advancements in technology have made it so that the robots can now see where they are going. So just like their organic counterparts, the path robots choose to use is highly dependent on the input they get from their visual components. It is not enough to identify an object they have already seen before, they should also consider if the object in front of them right now has been previously identified but just happens to be seen from a different angle or pose. This is where pose estimation comes in.

Problem Context

In this project, we are implementing a particle filter search algorithm to simulate real life visual pattern matching problems. This is a simplification of the 3D scenario that visual systems encounter in the real world. Shapes are worked in 2 dimensions, with only 4 degrees of freedom as opposed to 6 for real world scenarios. Real world objects can freely change their pose in 3 dimensions, but this project will only deal with objects rotating around a set plane. While this situation is significantly simpler, the methods applied to this project should generalize well back into 3D.

To simulate the problem, we are analysing two different images to test the effectiveness of the particle filter search algorithm. The first image contains two squares and two triangles while the second image only contains a single triangle. We aim for our implementation of the algorithm to be able to accurately identify the pose vector of the two-dimensional triangle we are searching for. To determine if the solution is a good fit, we will be looking at how close the estimation is to the actual pose of the triangle.

In this project, we were given two tasks to be completed. The first was to finish implementing the evaluate and mutate functions in a particle filter search algorithm for 2D pose evaluation. The second was to determine the best choice for initial population and number of generations when given a fixed computational budget for the particle filter search algorithm.

Methodology

Task 1

To complete the first task, we needed to utilise the evaluate function in the pattern_utils file provided. This evaluate function determines the cost for a given pose vector by calculating the average distance between each pixel on the pattern created by the pose vector (in this case a triangle) and an edge pixel on the edge image. To do this, it simply finds all the pixels on the pattern and evaluates them on the distance image the calculates the average value for each pixel. Once we can evaluate the cost, we need to do so for every pose vector randomly created in the initial population and store these in a cost vector.

For the next generation, we would then have to resample the population but this section of the code was already completed for us. So, then we had to determine which pose vector had the lowest cost

in all the generations and what that cost was. To do this we simply found these values for the most recent generation and updated them if we found a pose vector with a lower cost than all the pose vectors previously found. Hence, once the algorithm is finished we will have found the lowest cost pose vector overall.

The mutate function mutates a pose vector by a small amount between each generation. In the 2D example we have four degrees of freedom for a given pose vector which all needed to mutated. The x and y coordinates and the scale needed to be mutated by -1, 0 or +1 with equal probability. The angle need to be mutated by -1,0 or +1 degree. The only difficulty with completing this task was ensuring that we add one degree and not one radian when mutating the individual poses.

Task 2

For the second task in the project we needed to create experimental tests in order to determine what combination of population size vs. number of generations yields the best results with a fixed computational budget. So, first we needed to determine a reasonable computational budget which would provide a good solution in most tests. We wanted to choose a relatively large computational budget, but with the restriction that it could still be thoroughly tested in a reasonable amount of time. After various non-rigorous tests, we found that any combination of population size and generations that multiplied to 10000 would give a best cost lower than 1 which we arbitrarily determined to be a good cost result. A computational budget of 10000 was also not too large to prevent us from completing very rigorous tests on the dataset given the hardware available.

Using a computational budget of 10000 there are 25 possible combinations of initial population size and number of generations which multiply to 10000. We decided to test all of the possible combinations and run the particle filter search 100 times for each combination. We found the lowest cost individual pose for each test and plotted the costs in a box plot (Appendix. Figure 1 and 2). We used a box plot so that we could see the variance in the results which was a factor in determining the best possible combination. We wrote the code in python, but we used MATLAB to display the results in a nicer format in Figure 1 and 2. However, we were not able to complete some of the tests when there was a very small initial population size and a very large number of generations. This was because it would cause an error in the code preventing it from completing all the tests. Fortunately, for reasons discussed in the Analysis section we learned that these combinations do not provide good pose results.

We stored the median cost values over the 100 tests for each combination in Table 1 to explicitly show what the best cost of a typical result looks like. All of the tests were conducted on both images which were provided. By analysing these results, we were able to create future recommendations on how to choose initial population size and number of generations in a particle filter search algorithm.

Analysis

From analysing median values in Table 1, Figure 1 and Figure 2 we can see that the best cost initially starts high when there is a high initial population and a low number of generations. The costs then continues to decrease until it eventually flattens out when the population size is only slightly higher than the number of generations.

Notably the median costs tend to decrease again as the number of generations increases even further at the end of the graph. However, this result is actually very misleading. We found that as the number of generations became very large the best cost pose found was actually a degenerate

point. This is when the scale of the pose was continually decreased until it became a very small triangle with a very low cost. Obviously, this means that the search was not finding a useful result even though it had a very low cost. We found this started to happen very often once you had more than 500 generations. Hence, we will conclude that no combination with more the 500 generations will guarantee a good pose estimation. This could also explain both why there are so many outliers in Figure 1 and maybe why our tests stopped working once the number of generations was large enough.

We also found that in some tests of image 1 the best pose found was actually for the wrong triangle. Unfortunately, we had no way of testing how often this was happening so it could have an effect on how accurate our final recommendations are. This very clearly suggests that simply using the best cost individuals to determine the effectiveness of the particle filter search is very flawed and so more measures need to be considered in the future.

We can also make a hypothesis that having more possible patterns as interference increases the variance of the results when you have a large number of generations. It is very clear that for Figure 1 there are many outliers when there is a low population and many generations, but this is not true for Figure 2. The only significant difference between these two images is that image 1 has many more patterns which could interfere with the results. So, there might be a correlation between number of generations, variance in the pose estimation, and the number of interference patterns for an image. Generally, we can say having a large number of generations does seem to produce suspicious results.

In Figure 1 and 2 we can see that many of the combinations between 1000x10 and 80x125 (Population size x Generations) have a relatively low cost and a reasonably low variance. It is difficult to determine exactly which of these combinations would be the best choice. From this we can say that choosing any of these combinations when executing this algorithm would be a good choice. In this instance if we had to make a choice, choosing somewhere in the middle seems reasonable since you are less likely to encounter the flaws of either extreme. Hence, using this idea we can say that the safest choice seems to be around 250x40 or 200x50 if you want to ensure a good pose estimation.

In other scenarios where the computational budget is different we can say very generally that you want to have a higher initial population than number of generations, but not too large otherwise you find less well fitting poses. In general, we have seen that increasing the number of generations can have severe consequences and so it should always be done with caution. We also do not currently know if these issues are caused by the number of generations being much larger than the population size of if simply having a large number of generations will cause these issues. This would need to be discovered through further tests in the future.

Conclusion

In this project, we were able to use the particle filter search algorithm for estimating the pose of a two-dimensional triangle. We found that by using a high enough computational budget the algorithm was able to successfully find a reasonable approximation on most tests. We then aimed to determine the best possible combination of population size and number of generations when given a fixed computational budget. Through the experimental tests, we found that generally, it was better to have a slightly higher population size than generations to most often find a good estimation. However, there were a reasonably large range of values where good estimations could

be found. We also identified many issues with the algorithm especially when running tests using a large number of generations. In the future, we could study how the variance of the results are related to the number of interfering patterns or how to prevent the patterns from mutating into degenerate points. Overall, the project was successful in gaining a further understanding of the particle filter search algorithm.

Appendix

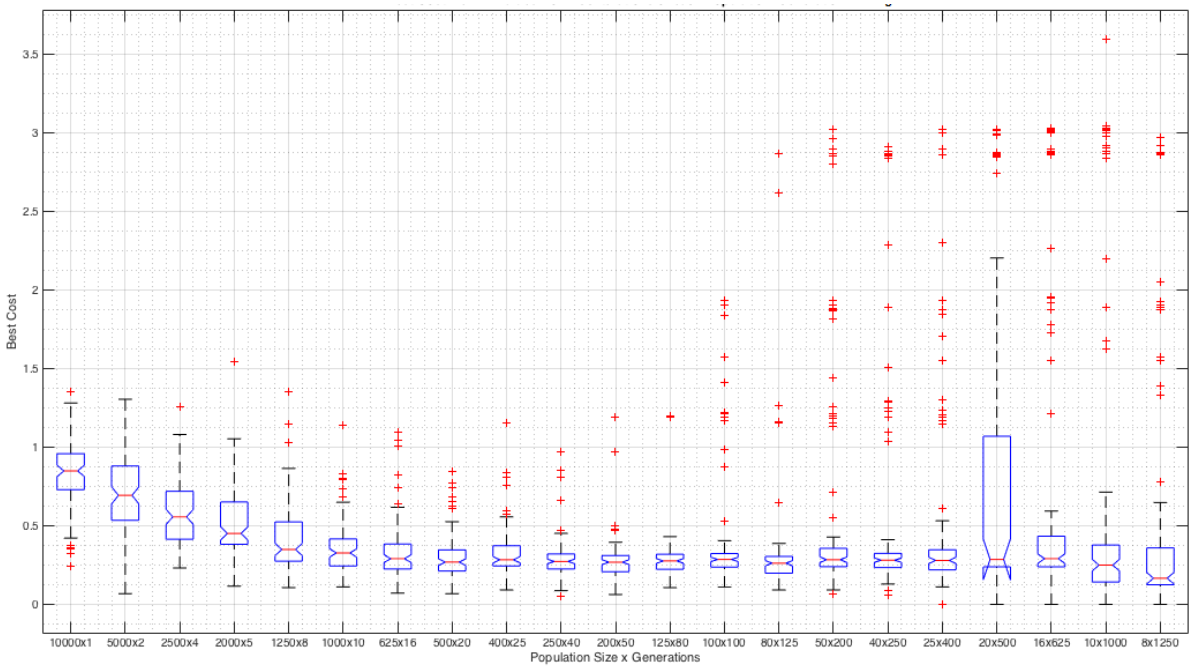

Figure 1: Best Cost Values from 100 iterations of each Population x Generation for Image 1

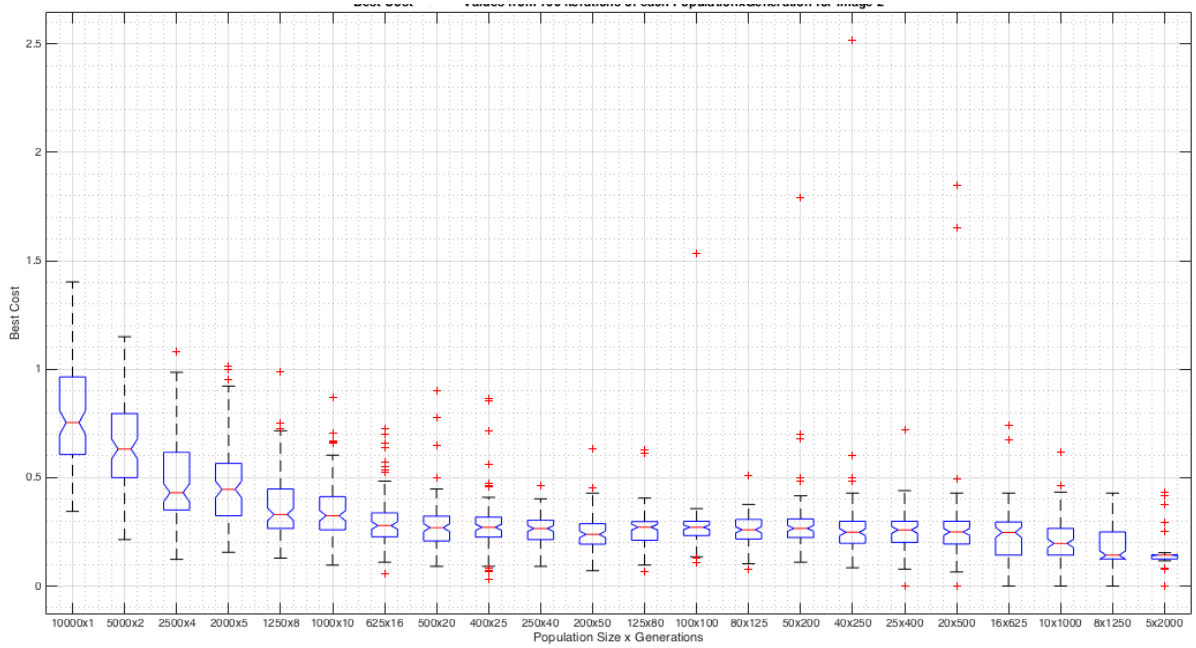

Figure 2: Best Cost Values from 100 iterations of each Population x Generation for Image 2

Population Size x Generation	Median Costs for Image 1	Median Costs for Image 2
10000 x 1	0.8495	0.7540
5000 x 2	0.6941	0.6322
2500 x 4	0.5569	0.4308
2000 x 5	0.4513	0.4458
1250 x 8	0.3495	0.3301
1000 x 10	0.3276	0.3246
625 x 16	0.2912	0.2792
500 x 20	0.2694	0.2694
400 x 25	0.2843	0.2709
250 x 40	0.2736	0.2653
200 x 50	0.2687	0.2387
125 x 80	0.2766	0.2718
100 x 100	0.2864	0.2709
80 x 125	0.2621	0.2597
50 x 200	0.2844	0.2662
40 x 250	0.2815	0.2483
25 x 400	0.2801	0.2589
20 x 500	0.2864	0.2500
16 x 625	0.2912	0.2475
10 x 1000	0.2500	0.1967
8 x 1250	0.1667	0.1428

Table 1: Median of Best Cost Values from 100 iterations of each Population x Generation for Image 1 and 2