COP-290 Assignment-1

Mohit Thakur 2019CS10373
Sidharth Agarwal 2019CS50661

March 31, 2021

Aim

To analyse the trade off between time and utility while implementing different methods to compute same functions.

Metrics

We took average absolute difference as metric for error and time to run after initialization as the metric for runtime. We can define any utility function which is monotonic inverse related to error. We took all our readings on "Intel i5-8265U (8) @ 3.900GHz" cpu, with minimum extra load possible.

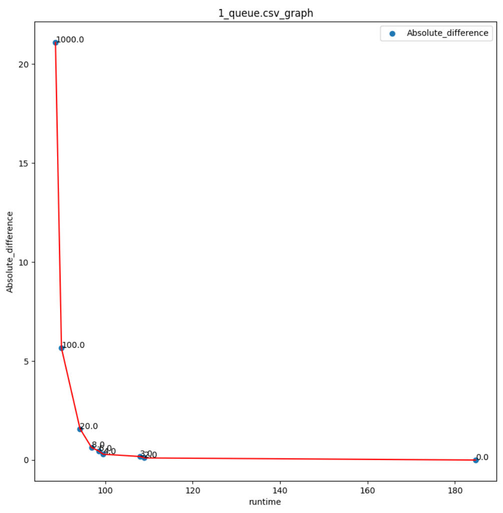
Methods

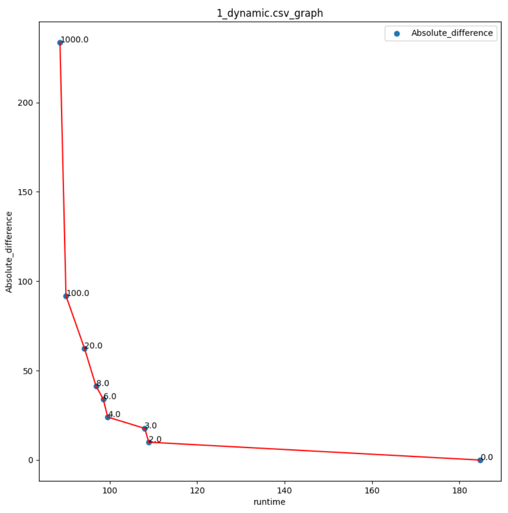
Method-1 We did sub-sampling of the frames, i.e. took one frame from every consecutive x frames and assigned the the calculated value of queue density to rest of them. For dynamic density part we skipped the rest x-1 frames from consideration. x is the parameter here
Method-2 It takes 2 parameters as input, for number of pixel per inch in horizontal and vertical direction. It loads image, lowers/highers its resolution, changes selected and final points accordingly. As this has 2 parameters we followed the standard video qualities for analysis.
Method-3 We treated each thread as a user viewing different section of the road. For each frame we first cropped it to fit this area_of_interest and followed the usual procedure in thread. We used x+1 threads for the same, main thread for opening loading next frame when all threads are done. And rest of the threads for viewing. It is kind of a map-reduce model where main thread check when all threads are done with processing it brings another frame to be processed. As matrix multiplication was the major task, cropping the frames initially reduced the size of matrix to multiply and thus the time of computation. This initial cropping causes some distortion in frame under observation
Method-4 In this case all threads are equivalent. We maintained a running variable,framenum. Each thread reads a frame when the framenumber is acceptable, and does the computation while rest of the threads are reading their frames. Algorithm stops when framenumber crosses the hard-coded frames value.

For uniformity in the analysis, we fixed the points clicked by user.After writing codes, we manually ran them by changing parameters. We also created the csv's for plotting this with parameter(error-runtime relationship) manually. We wrote basic scripts for calculating error and

Trade-off analysis

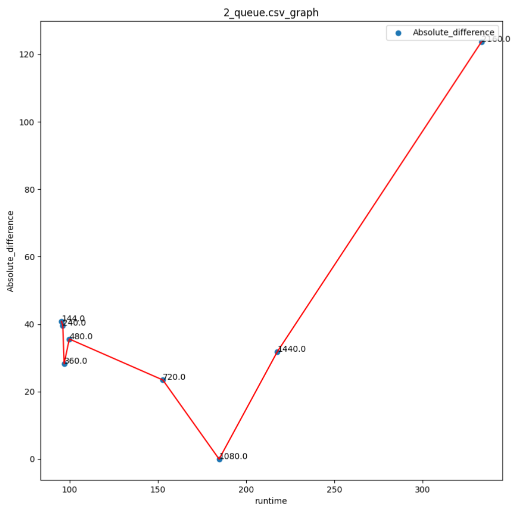
Method-1

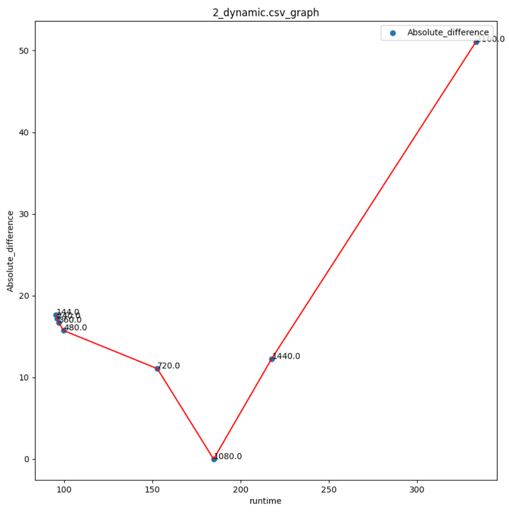




Vertical asymptotic behaviour follows from the fact that it will be loading every coming frame irrespective of the value of x. Error is decreasing with increasing x, and runtime is decreasing with increase in x was expected as we have lesser computation.

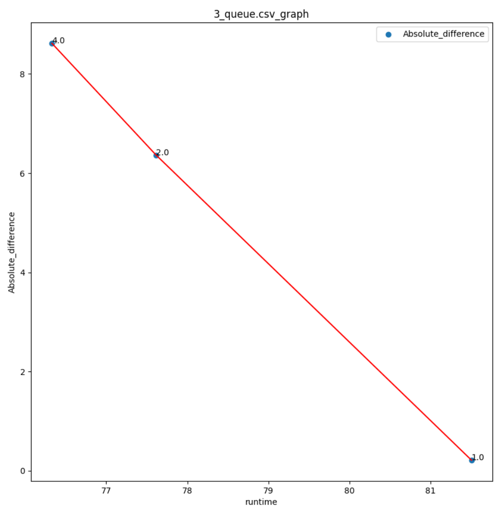
Method-2

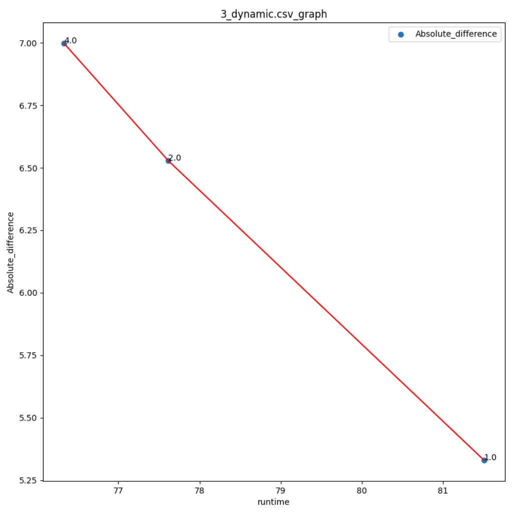




By increasing resolution, we expected increase in runtime and increase in error. We cannot create clear image from already blurred image, error observed is in increased resolution and not in original. By decreasing resolution, we expected for runtime to decrease as their is lesser computation. We expected the errors to increase with decreasing resolution, as lesser information is now stored, and observed the same.

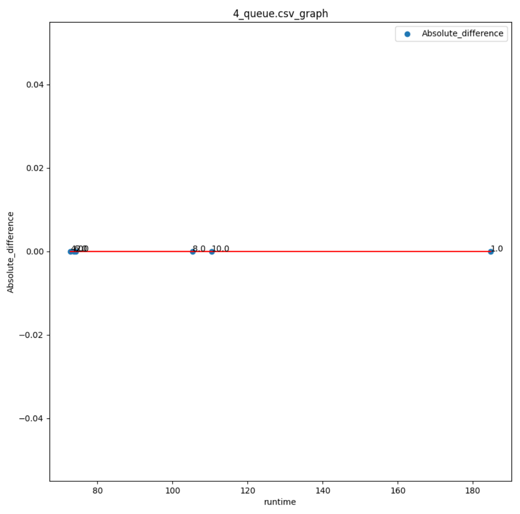
Method-3

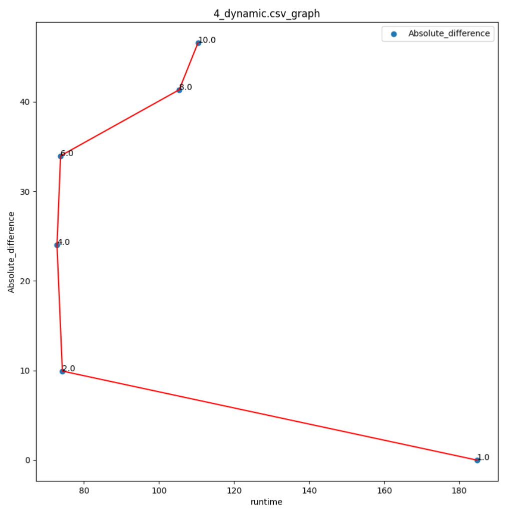




We were able to run this method for 1,2 and 4 only for unknown reason. Decreasing runtime with number of splits was expected, but the lesser decrease from 2 to 4, might be due to the fact that main thread now will have to check for more threads that are completed, which will cause delay. Error increases by increasing splits, follows from the distortion introduced by splitting.

Method-4





Error in queue_density part is always zero due to the fact that it exactly mimics the reference algorithm for its calculation. A thread created occupies it completely, as it keeps checking if framenum is favorable or either doing its computation. Thus, it slows down when using 8 or 10 threads.