COP290
Task1-Subtask3
Report
Prof: Rijurekha Sen

Abhisek Maji
2018CS10323

Lokesh Acharya
2018CS10352

Complex Systems, **Volume** (year) 1–1+; year Complex Systems Publications, Inc.

Keywords: metrics, parameters, error, methods, plots, trade-off, analysis

1. Metrics

In the subtask 3 we have defined our utility metrics and runtime metrics as follows:

1. **utility metrics:**

   We have taken the error of the queue density of each frame from the respective methods compared to our base method which we did in subtask2. Now to define one utility for each parameter of a method we took the average of the absolute values of the error. This error is inversely related to the utility, i.e lower the error higher the utiltiy.

2. **runtime metrics:**

   In this we have simply taken the time taken of execution of our code to evaluate the queue density. The runtime is calculated in seconds having 3 rounds of decimal places.

2. Methods

We implemented methods 1,2,3,4 but did not implement the extra credit part.

In method 1 we have taken 16 parameters from 1 to 16 i.e we kept on increasing the gap between two successive frames by one units.

In method 2 we implemented 15 parameters from 2 to 16 i.e kept on increasing the the resizing factor by 1 unit till 16.

In method 3 we run for number of threads = 1, 2, 4, 8, 16, 32 as parameter. Here, we split work among threads as sections of a single frame.

In method 4 we run for number of threads = 1, 2, 4, 8, 16, 32 as parameter. Here, we split work among threads as different frames.

3. Trade of Analysis

We divided our Analysis onto the two broader parts method1 vs method2 and the other being method3 vs method4.

Method 1 and Method 2

We did this because the method 1,2 were involved in reducing the processing of number of frames in the video while the later two involved in processing each frame by dividing the work among threads.

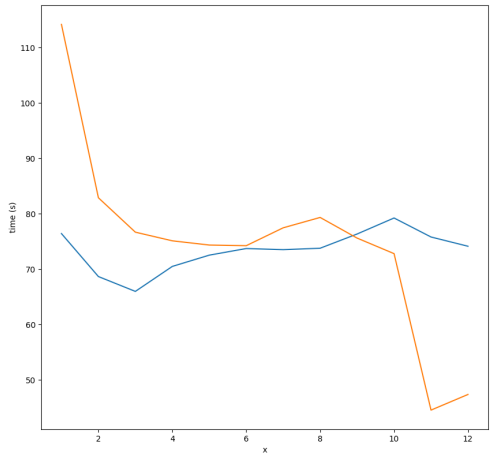

fig1: run-time vs parameter

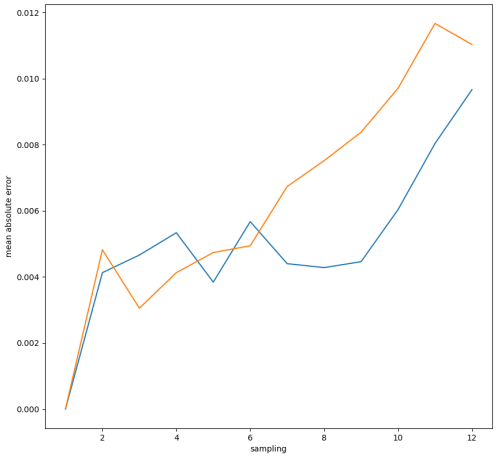

fig2: error vs parameter

The above two graphs are the plots for method1 and method2. The orange line corresponds to the method1 and the blue line corresponds to the method2. We have attached the data used for plotting the above graphs.

First we start analyzing Fig2. In both methods the error increase which seems to be logical. This is because in method 1, method 2 we processed the frames partially, in one case we skipped intermediate frames in another we decreased the resolution of the frame i.e we decreased the number of pixels to be processed. So in either case we are bound to get an error from the baseline and the error should increase as we tend to skip more of the work while processing the frames.

From the graph we can also see that the the error for method2 is less than the error for method1. This suggests that if we want to reduce our work for processing the video, we can chose method2 to do so, since its utility will be more than that of method1.

Now we look into Fig1. From the grap The run-time for method1 decreases which is logical because we keep on decreasing the frames to be processed by increasing the gaps. But the same does not happen so for method2, it kind of remains constant in runtime. This can be argued from the fact that in method2 we have to process every frame and because of this we have to take find the queue density of each frame although we decreased the work, which takes more time to do so, compared to method2 since in this we totally skip the intermediate frames by copying the previous result.

We now look into the trade off analysis between method1, method2 So in method1 the runtime is lesser than compared to method2 but the error becomes more than the method2. So considering that if we need to execute the processing faster we should be using method1 with high parameter value. Also if we need more utility we should be using method2. Moreover if we had to chose both, then method1 should be used since the runtime decreases much faster than method1, and the error though is more, but the differnce between them is almost constant throughout.

Method 3 and Method 4

In method 3 and 4 we are using pthreads to carry the utility-runtime analysis. The figures below show the relation between runtime vs threads of threads and error vs number of threads.

Runtime analysis

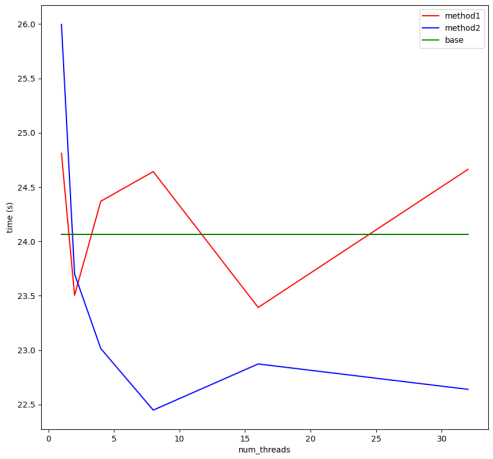

fig3: run-time vs number of threads

This figure shows relation between runtime and number of threads. Here we can see that for both method1 and method2 runtime is higher than base line, which is due to additional overhead over a single thread.
Method3: Here threads are not independent of frames. Here we can see that time then increases continuously from parameter value of 2 to 32 (except 16) because as number of threads increases for a single frame, the overhead incurred contribute more to the runtime hence time increases.
Method4: Here we observe that runtime continuously decreases for higher value of number of threads because here we have more workers to process a dedicated frame and independent of frames being processed. Therefore, the runtime decreases.

Utilty analysis

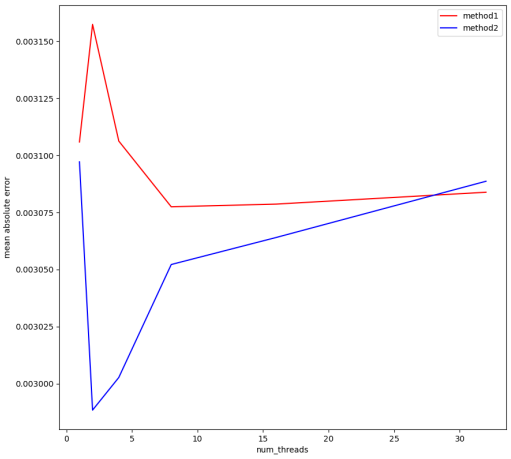

fig4: error vs number of threads

This figure shows relation between mean absolute error and number if threads. Lower the error higher the utility. Initially for both method3 and method4 the value of error is close because a single thread execute the whole work. Method3: In this method the value of error continuously decreases from thread 2 to 8, then as more threads get engaged in processing same frame the error value goes slightly up. For number of threads > 2 we can observe that the value of error is less than the initial error, therefore utility improves.
Method4: For this method we can observe that error increases continuously except the initial error. This may be due to the fact that since the threads works independently the locality of operation is not utilize in a single core instead distributed across multiple cores. Therefore the utility continuously decrease as we increase the number of threads.

Conclusive result

Method3: Here we come across the trade-off that as the runtime increases, utility improves.
Method4: Here we come across the trade-off that as the runtime decreases, utility degrades.