Analysis of Trade-offs in System Design

Shreya Arora ; Shreya Gupta

July 2021

Abstract

Active monitoring and analysis of road traffic are becoming increasingly important day by day as the number of daily commuters and consequently, the traffic on the roads is increasing. The current traffic monitoring systems are also quite ineffective because of frequent malfunctions and limited suitable mechanisms to enforce traffic rules properly. This leads to a lot of congestion of vehicular traffic on the roads leading to very high chances of road accidents.

Thus, having proper traffic monitoring and analysis systems becomes all the more important to ensure the safety of the commuters on roads. The congestion on roads caused by the difference in the accumulation of vehicles and the dispersal of vehicles at traffic signals and intersection crossings can lead to severe accidents. The impacts due to such road accidents can be highly detrimental. Additionally, when vehicles on the roads have to wait in queues for long durations due to traffic jams or traffic signals, they continue to use vehicle fuel at quite high rates and this ultimately makes the process extremely costly for the commuters, particularly the ones who travel on a daily basis. Thus, there is a need for proper traffic monitoring and analysis systems in order to ensure enhanced road safety and convenience for the commuters on the roads.

1 Introduction

1.1 Motivation

Traffic monitoring and analysis is an extremely complex process that requires certain sophisticated algorithms along with suitable access to data relating to the traffic density at different points of road stretches recorded over a period of time. This is an essential component in reducing the waiting durations of vehicles as well as their corresponding fuel consumptions and emissions during the commute time. In addition to this, it also allows for enhanced road safety, and better enforcement of traffic rules through more effective traffic management and control techniques.

1.2 Problem Statement

We have to estimate the queue density for the vehicles on a certain stretch of a road waiting at the traffic signal junction through various methods and conduct a utility-latency tradeoff analysis by comparing the results from each method with the baseline results.

1.3 Objectives

1. Estimating the queue density and the dynamic density of vehicles at a traffic junction over a period of time for a certain stretch of the road.

2. Conducting a runtime-utility tradeoff analysis to optimize the parameters by applying various methods and comparing them with the baseline results. This would lead to optimization of our software design for estimation of queue density and dynamic density in order to achieve more accurate results within shorter execution times

1.4 Contribution

1. Our primary focus is on designing an effective traffic control and management system that provides for enhanced road safety and enforcement of traffic rules.
2. This analysis would also prove to be useful for developing suitable traffic control and monitoring systems in order to achieve enhanced navigation and traffic management systems

1.5 Report Outline

Through this assignment, we are estimating the queue density for vehicles on a certain stretch of a road over a period of time through various methods and approaches in order to optimize the traffic monitoring and analysis system to ensure better accuracy of results in shorter execution time periods. Towards the end of this report, we also move on to analyzing the various tradeoffs between conflicting optimizing parameters like utility and latency.

2 Metrics

In order to optimize the traffic monitoring and analysis system through the estimation of queue densities at various points in time, we need to consider the following metrics in order to understand how the optimization process should proceed.

1. Accuracy: This is the primary parameter that needs to be optimized. Our program should output the accurate values of queue density at any given point in time.

2. Latency: This metric requires accurate results to be produced within a short span of time i.e. the execution time of the program should not be too large and the program should be efficient.

3. Throughput: This implies that the program will generate a large number of outputs per unit of time.

4. Temperature: This consideration requires the maintenance of temperatures of the processor under control and this becomes particularly important in the cases where the processor has to be deployed in places that have high temperatures. In these situations, since, cooling down through other external ways would be very difficult and wherever possible would also lead to high costs, thus optimization of the temperature becomes very important.

5. Energy: The system should be efficient in terms of its usage of energy and it should be such that it operates properly through limited energy supply as well.

6. Security: This metric requires the protection of the software from hackers in order to ensure the continued proper and safe functioning of our traffic monitoring and analysis system.

In this assignment, we will be prioritizing broadly two parameters for optimizing our traffic monitoring and analysis system. These would be accuracy and latency determined by utility and runtime measles as explained below.

We will be considering the Baseline for our analysis as the baseline code that processes each frame of the video at the traffic junction one by one and evaluates the queue density for each of these frames in a sequential manner.

Now, since accuracy and latency i.e. accuracy of the output and the runtime are conflicting in nature and thus we need to perform certain tradeoff analyses in order to optimize our system. Through this report, we would be discussing each of the different optimizing methods in detail along with an in-depth analysis of the tradeoffs performed in the optimization process.

We are defining the following metric as the utility in our program for all 4 methods. We have used the following formula for evaluating the utility metric for all the video frames being processed to estimate the queue density and compare it with the baseline output values:

$$Error(inpercentage) = \sum_{n=0}^{numframes} \frac{QDn - BDn}{BDn} \ x100$$

$$Utility(inpercentage) = 100 - Error(inpercentage)$$

where QDn and BDn are the queue density and baseline queue density for the nth frame.

3 Methods

3.1 Method 1: Sub-Sampling Frames

In this method, we skip through a fixed number of frames, say x, as we progress through each frame of the traffic video. At each step, we would be skipping the same number of frames i.e. x and this will play the role of our optimizing parameter in this case. Once, the frame number N has been processed, the next frame to be processed will be N + x and the process will continue similarly for the rest of the frames of the video. For all the frames having the numbers between N and N+x, we will be assigning the same queue density to them as the one calculated for the frame N. A natural consequence of this method will be the reduction in utility or accuracy along with the reduction in runtimes as the value of x increases.

Now, this method processes fewer frames than the baseline code which princesses each and every frame of the video. Thus, this method will lead to a decrease in the total execution time or the runtime of the program as compared to the runtime of the baseline code. However, this method would also lead to a loss of accuracy of the queue density values evaluated for each frame as the frames lying between N and N+x will be prone to some error due to the loss of the exact information at this frame. Thus, this method also leads to a decrease in the utility value of the output.

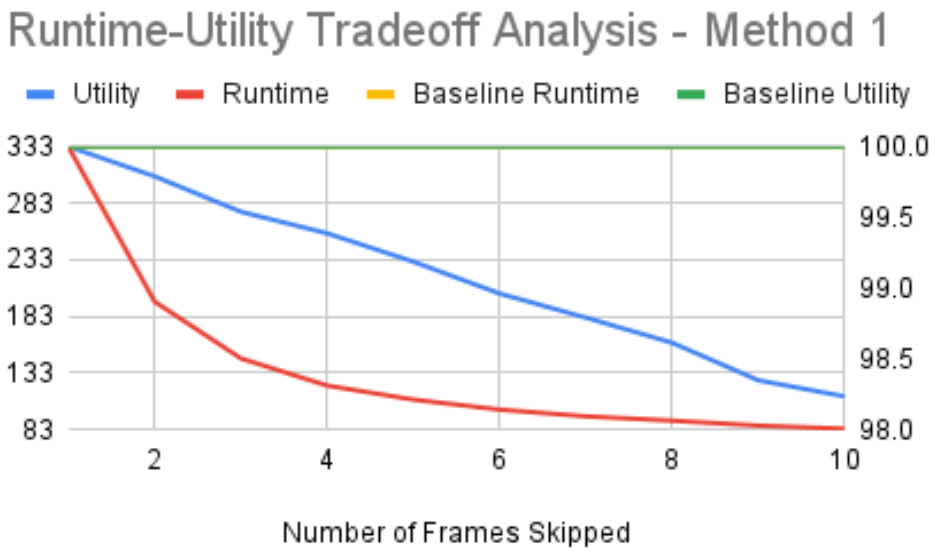

Figure 1: Runtime-Utility Tradeoff Analysis - Method 1

Through the above graph, we have observed that our initial assumptions of the runtime as well as the utility being reduced with the increase in the number of skipped frames as been validated. Thus, the above plot is in accordance with our initial expectations. Lastly, the optimum value of the parameter x (number of frames to be skipped) in order to have a tradeoff between the optimization for the runtime as well as the utility value of the output can be considered to be 9. Thus skipping 9 frames would optimize the values of both the runtime as well as the utility for the given traffic video.

3.2 Method 2: Reducing Resolution of Frames

Through this method, we process each frame of the video after reducing its resolution. We ensure that the aspect ratio of the frames' lengths and widths remains the same, hence, we take only one parameter by which we would be reducing the length and the width of the frame being processed. This float value of the resolution factor would be the optimizing parameter for this method.

If the resolution factor being considered is f, then, the frame being processed would be of f times the initial resolution of the frame. Thus, if the original resolution of the frame of the video was (x,y), then the new frame being considered for processing will have the resolution (fx, fy).

Frames having lower resolution may not require a lot of time and hence would lead to a lower runtime value for the program (or faster implementation). However, the reduction in the resolution of the frame being processed would also lead to some information loss for the frame being processed and hence the utility value for the queue density would reduce as well. Thus, the accuracy of this method would come out to be lower than that of the baseline program. Additionally, since we are also maintaining the aspect ratio of the frame while lowering the resolution of the frames being processed, their corresponding queue density, as well as the utility measure, are expected to be constant.

We observe that the runtime decreases with decreasing resolution factor. The utility comes out to be 100 percent at resolution factor = 1 and less than 100 percent for the rest of the values. Hence, the curves show the expected behavior.

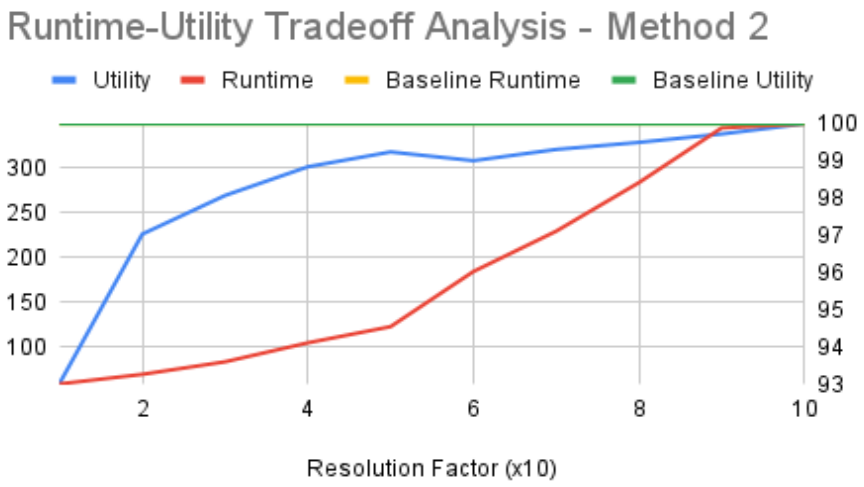

Figure 2: Runtime-Utility Tradeoff Analysis - Method 2

Through the above graph, we have observed that our initial assumptions of the runtime, as well as the utility being reduced with the reduction in the resolution factor, prove to be correct. Thus, the above plot is in accordance with our initial expectations. Lastly, the optimum value of the parameter XxY in order to have a tradeoff between the optimization for the runtime as well as the utility value of the output can be considered to be 0.5*X x 0.5*Y. Thus lowering the resolution of the video frames by a factor of 0.5 would optimize the values of both the runtime as well as the utility for the given traffic video.

3.3 Method 3: Frame Splitting across Multiple Threads

In this method, we utilize the concept of multithreading for estimating the value of queue density for the different frames of the video. Thus, in this process, we split each frame into a certain number of divisions, say x, and each such subpart is allocated to a different thread for processing. This process is similarly followed for all the frames of the video. This variable x i.e. the number of divisions of each frame of the video forms the optimizing parameter in this case. The processing happening in each of the individual threads goes on parallelly with the processing in the other threads to ultimately calculate the queue density of the entire frame of the video. Finally, after all the threads are done with the processing for their respective parts of the frame, then the values of the queue density obtained from each subpart of the frame are added up to obtain the value of queue density for the entire frame. We are utilizing application pthreads here.

As, we increase the number of threads involved in the processing, in this case, the process becomes more and more parallel and consequently, we can expect a reduction in the runtime of the program because multiple operations are happening in parallel to each other. However, after the multithreading process for each of the frames gets completed, there are very limited operations left to do and hence we realize that there may not be a lot of efficiency in the process of the reduction of the runtime. Additionally, splitting up of the frame into its corresponding subparts to be given to the different threads may also cause an increase in the runtime. Also, once the processing of any particular frame is completed, that thread has to wait all the way till the very last thread completes its processing for any particular frame of the video in order to join with the other threads and generate the value of the queue density for the entire frame together. Thus, these factors along with additional considerations required for the creation and the joining of threads can ultimately lead to an overall increase in the runtime of the program as well. Lastly, the method for processing all the frames is very similar to the one deployed in the baseline program.

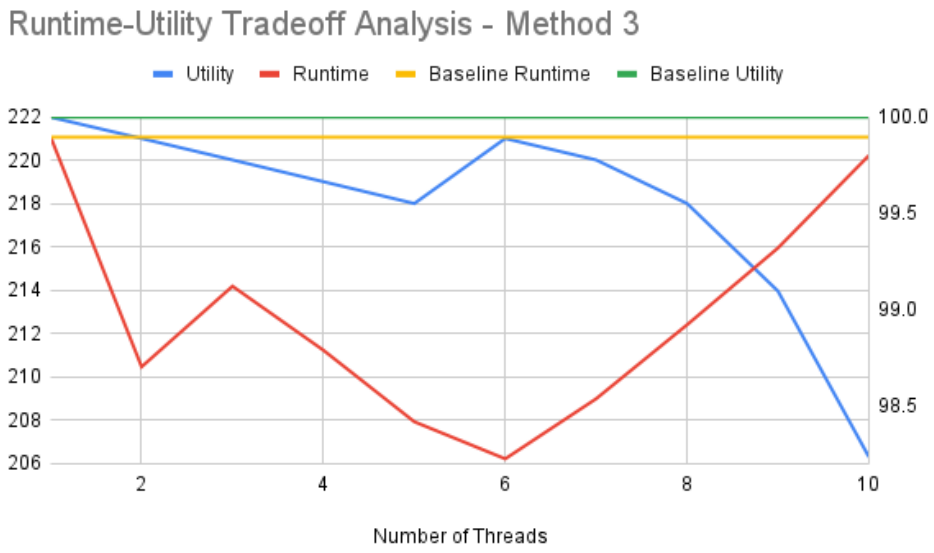

Figure 3: Runtime-Utility Tradeoff Analysis - Method 3

Through the above graph, we have observed that the runtime for the program has reduced because of multithreading and thus the overall time efficiency of the program has enhanced. One prominent observation from this method is that there is no regular trend that is being followed with the increase in the number of threads, unlike the previous 2 methods. In our case, we evaluate the queue density for each of the threads individually and then combine those values. In cases where the frame length is not exactly divisible by the number of threads being specified in the program, there may be a loss of a few pixels at the boundaries during the division process of the frame and this loss of information from the boundaries leads to some amount of reduction in utility in some cases. Lastly, the optimum value of the parameter, i.e. the number of splits (number of threads) in order to have a tradeoff between the optimization for the runtime as well as the utility value of the output can be considered to be 7 threads or 7 splits. Thus splitting the processing of each frame into 7 threads would optimize the values of both the runtime as well as the utility for the given traffic video.

3.4 Method 4: Temporal Splitting across multiple threads

In this method, we utilize the concept of multithreading for estimating the value of queue density for the different frames of the video. In this case, we can have consecutive frames being processed by multiple threads. Here, the number of threads, x forms the optimizing parameter. We take up x consecutive frames of the video and give each of these frames to different threads for processing them parallelly with each other. Thus, since, multiple frames are being processed in a parallel manner rather than a sequential manner, the total runtime of the program is expected to decrease. Additionally, each of the frames is processed in such a way that the process for the evaluation of the queue density is very similar to that utilized in the baseline program. The output values are expected to be the same as the baseline program as the only difference in this method from the baseline program is that some frames are processed in parallel rather than in a sequential manner. The value of the utility or accuracy, in this case, is expected to stay at 100 percent for all possible number of threads.

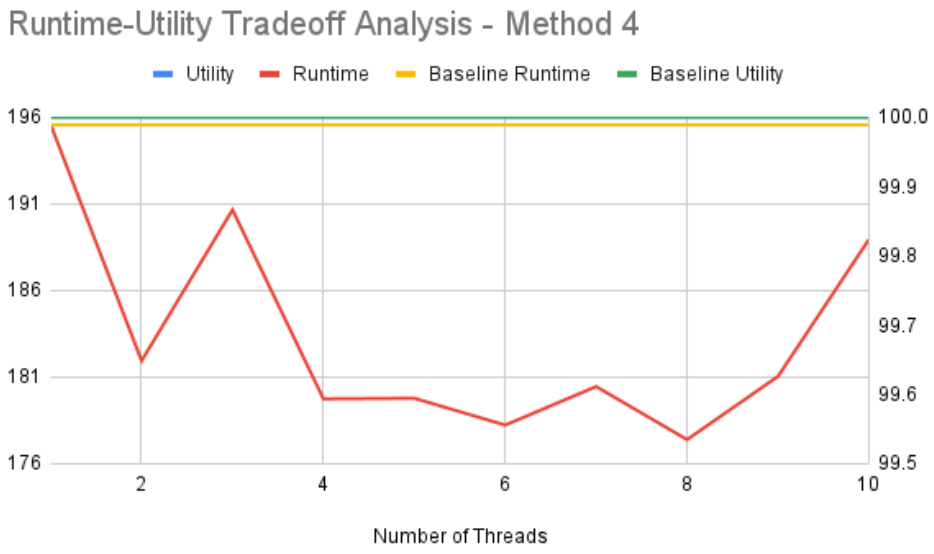

Figure 4: Runtime-Utility Tradeoff Analysis - Method 4

Through the above graph, we have observed that the runtime for the program has reduced because of multithreading and thus the overall time efficiency of the program has enhanced. In fact, the runtime for the program keeps on decreasing with the increase in the number of threads.

Lastly, the optimum value of the parameter, in order to have a tradeoff between the optimization for the runtime as well as the utility value of the output can be considered to be 8 threads. Thus splitting the processing into 8 threads would optimize the values of both the runtime as well as the utility for the given traffic video. The utility for this method comes out to be 100 percent as was expected. Thus, the above plot is in accordance with our initial expectations.

4 Results and Analysis

In section 3.1, through the method of sub-sampling of frames, we observed that the runtime decreases with an increase in the parameter, i.e. the number of frames skipped. However, here, there is a tradeoff involved with the reduction in the accuracy of the output values and hence a consequent reduction in the utility metric. Thus, an optimum value of the parameter, in this case, would be when 5 frames are skipped and in this case, the values of both the runtime as well as the utility are suitably optimized well and we are not losing out on one due to the other.

In section 3.2, through the method of resolution reduction of frames, we observed that the runtime decreases with an increase in the parameter, i.e. the resolution factor (fX x fY, where f is the resolution factor). However, here, there is a tradeoff involved with the reduction in the accuracy of the output values due to loss of information during the processing and hence a consequent reduction in the utility metric. Thus, an optimum value of the parameter, in this case, would be when the frames are resolved by a factor of 0.5 and in this case, the values of both the runtime as well as the utility are suitably optimized well and we are not losing out on one due to the other.

In section 3.3, through the method of frame splitting across multiple threads, we observed that for all values of the parameter, the runtime was less than the baseline program runtime and it fluctuated as we progressed from one value of the parameter to another. This was due to a number of factors as described in section 3.3. In cases where the frame length is not exactly divisible by the number of threads being specified in the program, there may be a loss of a few pixels at the boundaries during the division process of the frame and this loss of information from the boundaries leads to some amount of reduction in utility in some cases.

Thus, there is a tradeoff involved here with a reduction in the accuracy of the output values due to loss of information during the processing and hence a consequent reduction in the utility metric. Thus, an optimum value of the parameter, in this case, would be when the frames would be split up into 7 threads and in this case, the values of both the runtime as well as the utility are suitably optimized well and we are not losing out on one due to the other.

In section 3.4, through the method of frame splitting across multiple threads, we observed that for all values of the parameter, the runtime was less than the baseline program runtime and it reduced further considerably with an increase in the number of threads. Also, since all the frames are processed in a similar manner to the baseline program, we would have 100 percent utility from this method. Thus, an optimum value of the parameter, in this case, would be when the frames would be split up into 8 threads and in this case, the values of both the runtime as well as the utility are suitably optimized well and we are not losing out on one due to the other. This method is particularly better than the 3rd method as it does not cause a reduction in utility with an increase in the number of threads. Additionally, it also has a considerably less runtime as compared to the 3rd method. However, while the first method showed an even greater reduction in the runtimes, it also leads to a reduction in the utility, unlike the 4th method. Thus, out of the 4 methods utilized above, the 4th method is an optimum one as it reduces the runtime considerably while not compromising on the utility as well.

5 Conclusion

In conclusion, through the above analysis, we observed that while methods 1 and 2 reduce the runtime, they also lead to a significant reduction in the utility values of the output. We also observed that both methods 3 and 4, reduce the runtime for the program and make it more time-efficient, but only the 4th method does not compromise on the utility values of the output. Thus, multithreading has given better results as compared to the first 2 methods.