Traffic density estimation using OpenCV functions

Rahul Parmar (2019CS10387) , Satwik Jain (2019CS10398)

March 31, 2021

1 Metrics

We have to define metrics for our analysis. We have defined Runtime and Error as the metric against our parameters. Error is defined as the Root Mean Squared value of the difference of queue density from the baseline(temp.cpp). The baseline is defined similarly to the 4 methods with original frames and values calculated at every frame. The value of error for Methods 3 and 4 will be zero as the way they are implemented, we iterate over the frame and count the value so dividing the frame spatially or temporally doesn't change the value of density. Runtime for baseline turns out to be nearly 300 sec.

2 Methods

2.1 Method-1 (processing every x frame)

In this subsection, we perform sub-sampling of frames by processing every x frame i.e. process frame N and then frame N+x, and for all intermediate frames just use the value obtained for N. Parameter for this method is x, number of frames skipped.

2.2 Method-2 (reduce resolution for each frame)

In this subsection, we reduce resolution for each frame. So in this case running time can be reduced but error increases. The parameter is the scale by which frame resolution is reduced.

2.3 Method-3 (Split Frame according to threads)

In this part we have to split work spatially across threads (application level pthreads) by giving each thread part of a frame to process. So In this section we divided frame into n numbers of column and send each column to process in different threads which reduces our net time taken by factor of n. Parameter for this case is number of threads used.

2.4 Method4 (Give consecutive frames to different threads)

In this subsection split work temporally across threads (application level pthreads), by giving consecutive frames to different threads for processing. Parameter for this method is number of threads used. In this method we used different threads to go over every xth frame according to number of threads . It will ultimately reduce our time by $1/(\text{Num}_o f_t hreads) for process in which we are calculating Queue Density$.

3 Analysis

3.1 Method1

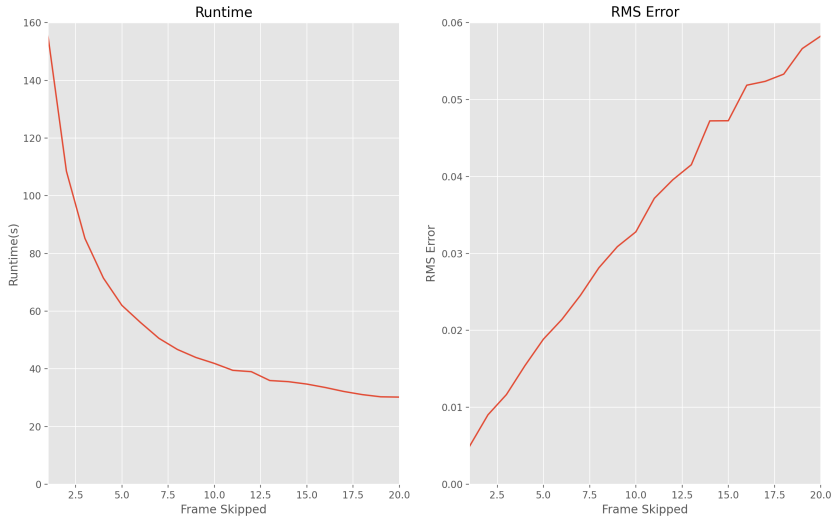

Figure 1: Method 1

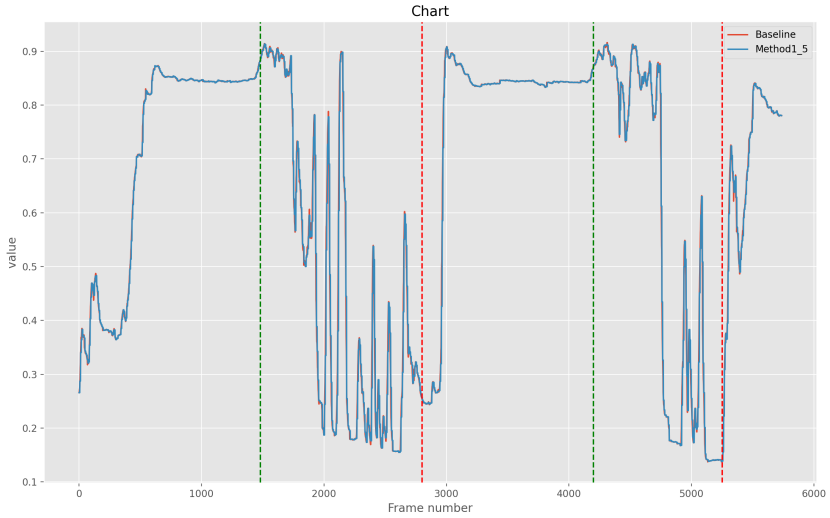

Figure 2: Baseline vs Method 1

In Method 1, we can see that runtime decreases inversely to the number of frames skipped, i.e., if baseline time is t, for initial values of frame skip, time t/(x+1), x being number of frames skipped. The value of error increases linearly to the number of frames skipped.The change in runtime can be inferred easily as now there are less frames to be cropped and runtime decreases. It is not an exact inverse relation as time is taken in traversing the video which is not decreased.

In Fig-3 we showed graph of Queue Density in which we are skipping every 5th frame and our baseline . So through this difference we calculated RMS (utility for method 1) . Same type of graph shown in Fig-4 for Method 2

3.2 Method2

In Method 2, There is a decrease in runtime if frame is scaled down by a large extent compared to the original. The value of error also increases on big scaling. A difference is seen at scale = 0.5 which can be said that at this value, there are exact value of pixel calculated(4 pixels in 1) as the frame is directly halved in both dimension. In other cases, some estimation is done for the frame values.

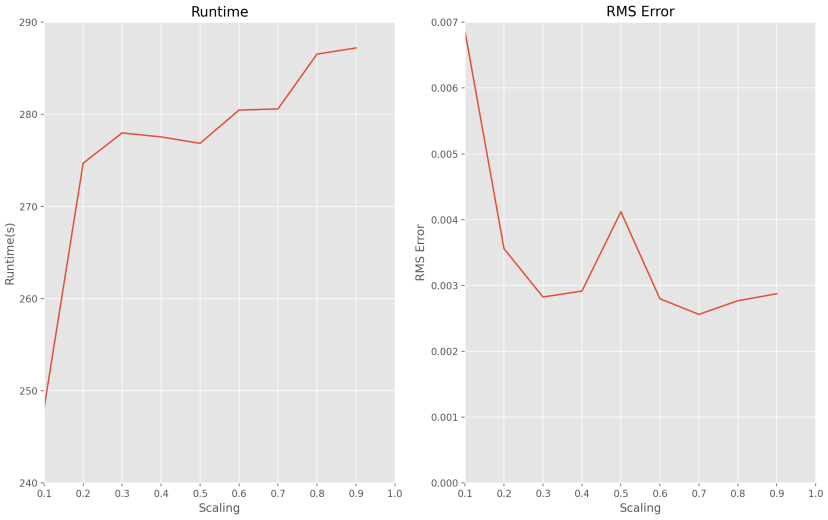

Figure 3: Method 2

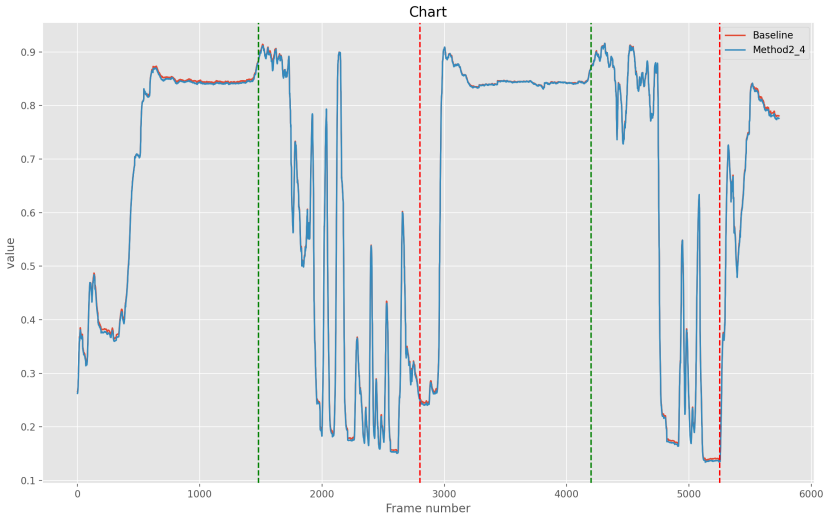

Figure 4: Baseline vs Method2

3.3 Method3

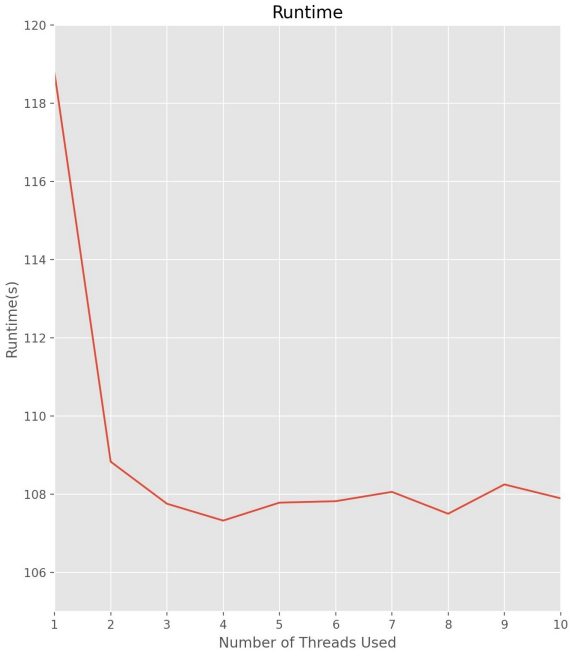

Figure 5: Method 3

In Method3. Result of Queue density is same as baseline of Assignment Hence we got no error in values of density at respective frame. Here in Fig-5 , we calculated runtime(in sec) of function to perform image subtraction using multiple threads. Here on moving from 1 thread to 2 threads there is drastic change which is due to time consumed by thread function (i.e. to iterate over whole video to calculate queue density) become almost half. because each single frame work is distributed over 2 threads . On adding another thread we see again some reduction in total time due to extra thread.

After thread 5 runtime become almost constant because our local machine doesn't have enough threads/cores to run multiple process parallelly. Here we made an number of thread vs runtime graph as (parameter vs utility)

3.4 Method4

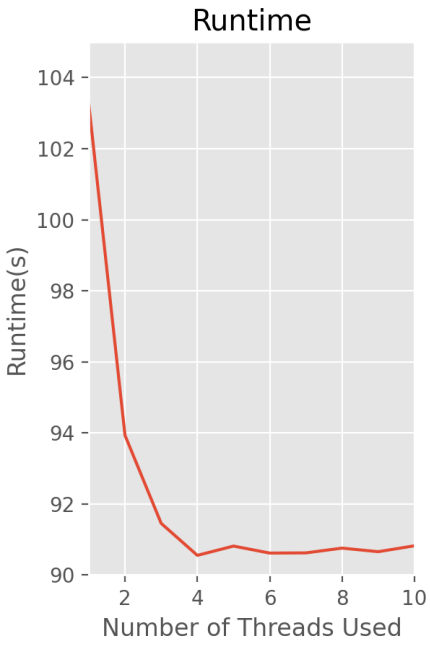

Figure 6: Method 4

Same as Method3 , Method 4 has no error w.r.t to baseline because of we calculated queue density of every frame in same way as baseline. In Figure 6 we see . Drastic decreament in runtime vs Number of threads Used for first half. Because each time on increasing a thread , Number of frame processed at per thread is reduced by a factor. After Thread 5 runtime remains almost constant because of local machine power restriction as described in method 3 analysis.