Utility-Runtime Trade-Off Analysis in Queue-Density Estimation

---

BY

KURISETI RAVI SRI TEJA - 2019CS10369
GATTU KARTHIK - 2019CS10348

Contents

1 Introduction

1.1 Traffic Densities and their Calculations

1.1.1 Queue Density

Queue Density is defined as the fraction of the area of the road that is occupied by the vehicles.It's value lies in the range [0,1].For calculating the queue density of a particular road, we first transform the frame i.e., perform perspective correction using homography to make the road parallel to the plane of paper and then crop the unnecessary portion. The obtained matrix is associated a name frame f. The same operation is performed to a reference image which contains no traffic(i.e., empty roads) and the obtained frame is given a name frame ref. We associate a quantity **required cells** for each frame.It is defined as the number of cells in frame f such that absolute difference in pixels between frame f and frame ref is less than 30.Then queue density is calculated by the formula.

$$\text{Queue Density} = \frac{\text{Required cells}}{\text{Total no.of cells in frame f}}$$

1.1.2 Dynamic Density

Dynamic Density is defined as the fraction of the area of the road that is occupied by the moving vehicles.It's value lies in the range [0,1].For calculating dynamic densities usually optical flow algorithms are used which are very complex and gives accurate results.For calculating dynamic density,we used highly approximate techniques i.e., we found difference between every two successive frames and counted the no.of cells whose pixel values are more than 5 after applying homography and cropping the image.

$$\text{Dynamic Density} = \frac{\text{Number of cells with pixel value} > 10}{\text{Total no.of cells in frame f}}$$

2 Metrics

2.1 Baseline for Metrics

We will consider the outputs produced when we are processing the whole video frame-by frame as the baseline.

2.2 Runtime Metrics

For measuring the runtime of program, we used the built in clock provided by the std::chrono library which can be used to calculate the time taken for execution of the program in milliseconds which depends on the way video will be processed.

2.3 Utility Metrics

We used the mean of squares of difference of queue-density value obtained in each method used and corresponding queue-density in baseline as absolute error value and took utility as (1-relative error)*100 for plotting various graphs where relative error was taken as

$$\text{Relative Error} = \frac{\text{Absolute Error}}{\text{Average Queue Density of all frames in the video}}$$

.

3 Various Methods for Trade-Off Analysis

3.1 Sub-Sampling Frames

In this process we would process every kth frame starting from 1st frame. The baseline for this method is to process every frame. We tested this process with the values of k as 2,5,10,50,100,500.

3.2 Changing Resolutions

In this process we would use the standard resolutions which are lower than the resolution of given video i.e., 960*540,480*270,240*135,120*67,60*33 and compare them with 1920*1080 resolution(original image).

3.3 Using pthreads

For improving the speed of execution we used the pthreads provided by the OS. We tried to speed up the execution in two possible ways.

3.3.1 Processing Consecutive frames using different threads

In this process we created multiple threads and used them to process consecutive frames in such a way that each frame is processed by only one thread.

4 Utility-Runtime Plots

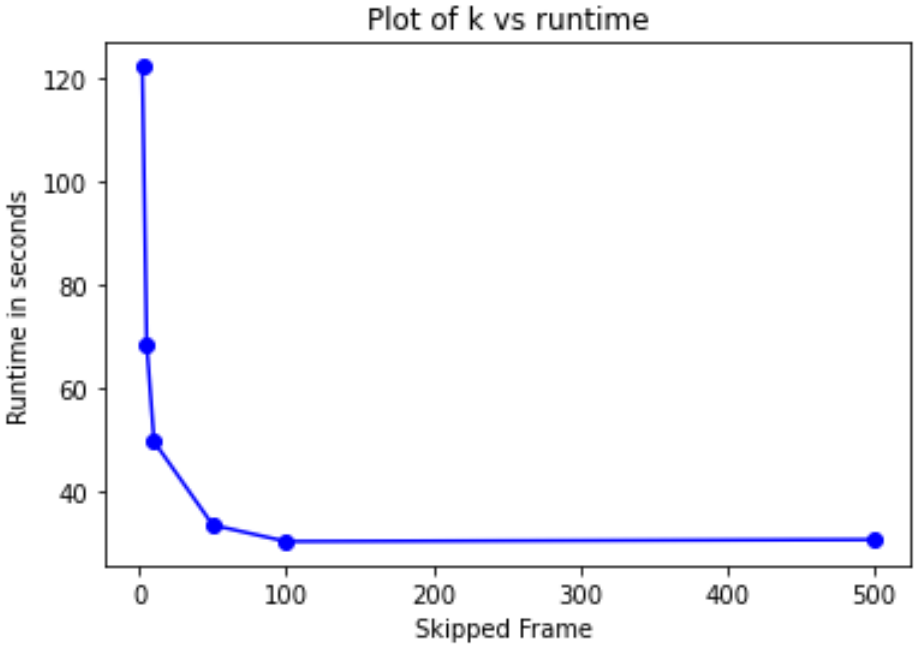
All the data used here was produced using Mac-Book Air laptop with 1.6GHz dual-core Intel Core i5, Turbo Boost up to 3.6GHz, with 4MB L3 cache.

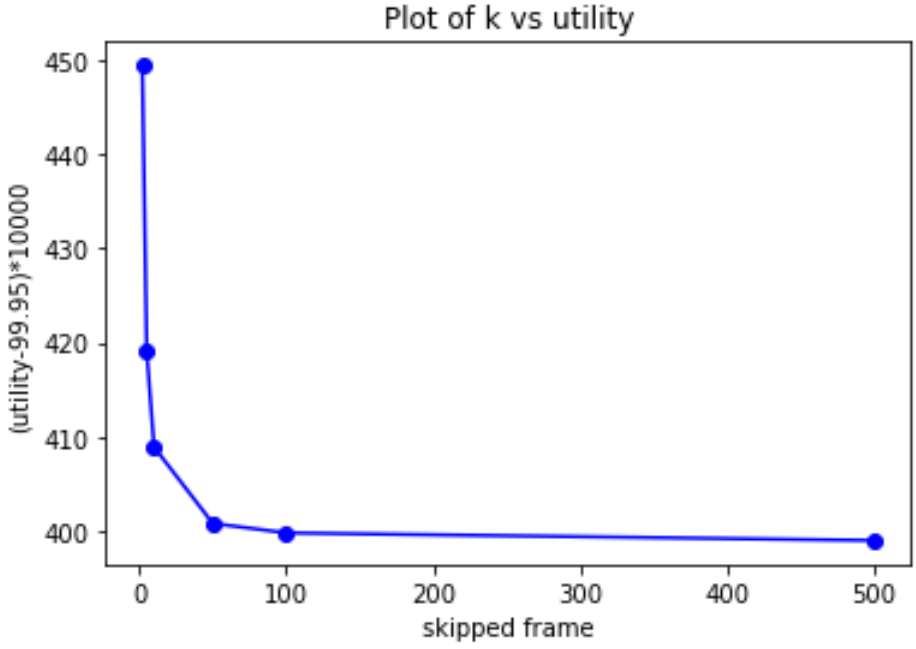
4.1 Sub-Sampling Frames

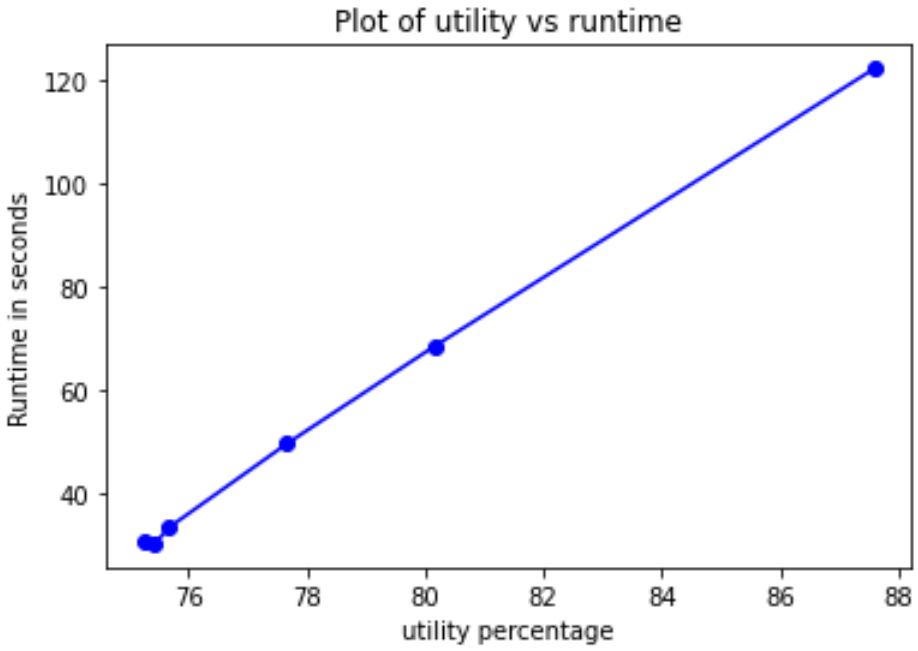
k is the parameter by which we skipped frames.

Value of k	**Runtime in seconds**	**Utility Value** *(utility-99.95)*10000*
2	122.337	449.408
5	68.520	419.045
10	49.715	408.909
50	33.548	400.806
100	30.383	399.799
500	30.740	398.997

Corresponding Plots:







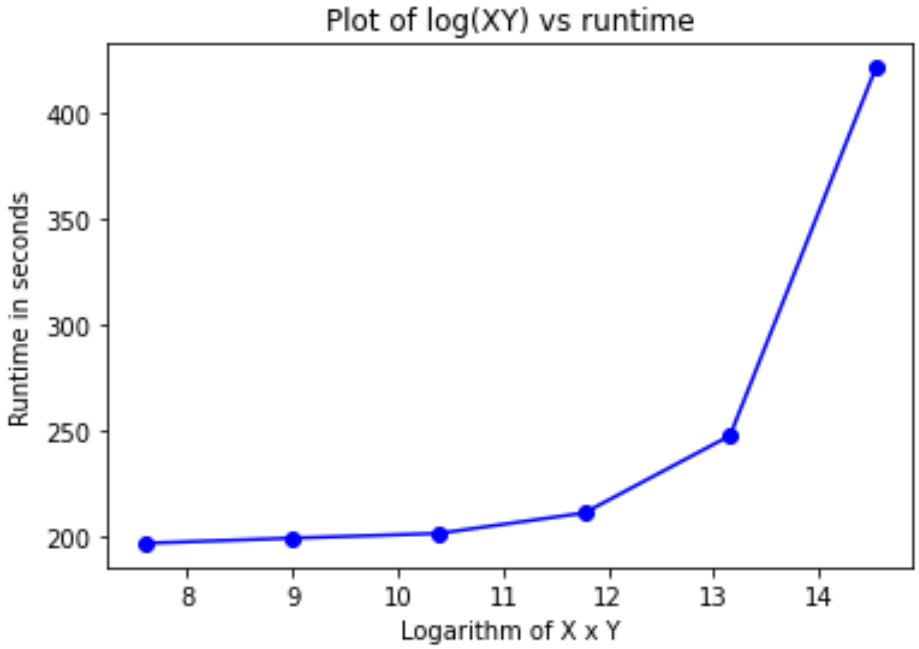
4.2 Resolution Change

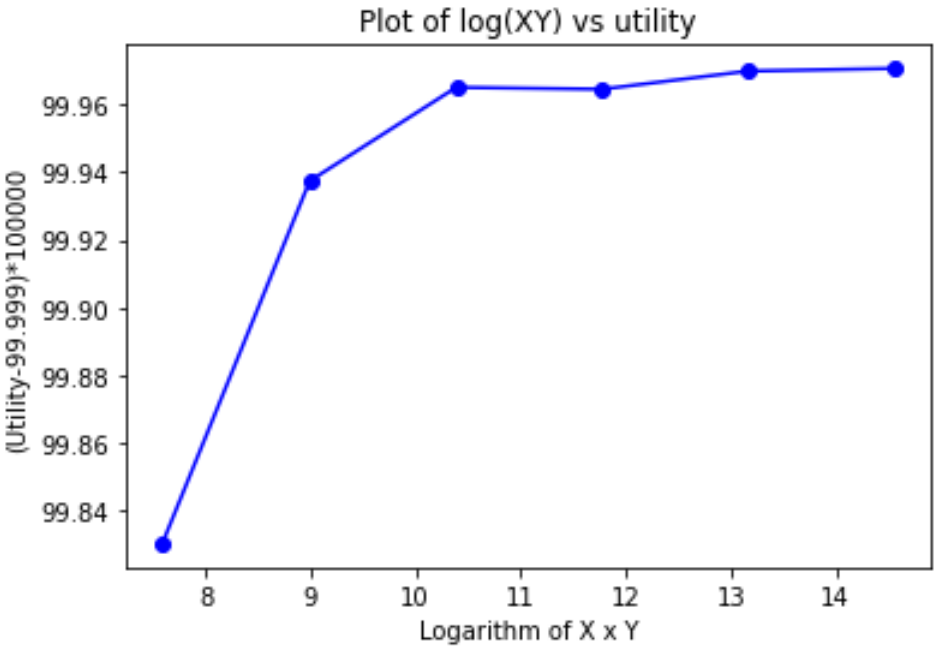
X,Y are dimensions of image to which we are compressing or expanding by changing resolution is the parameter by which we skipped frames.

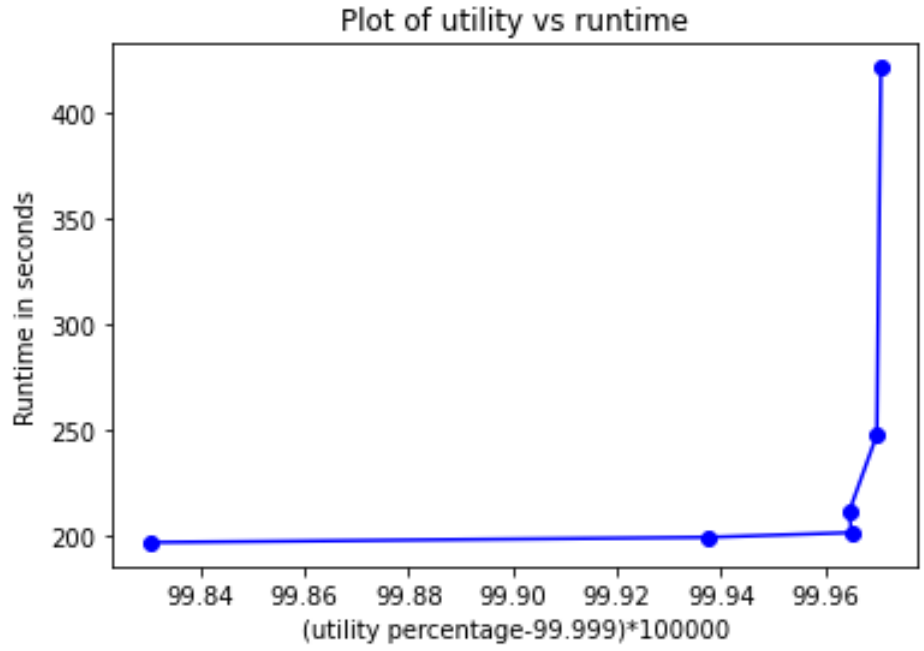
Value of X	**Value of Y**	**Value of XY**	**Value of log(XY)**
60	33	1980	7.591
120	67	8040	8.992
240	135	32400	10.386
480	270	129600	11.772
960	540	518400	13.159
1920	1080	2073600	14.545

Value of log(XY)	**Runtime in seconds**	**Utility Value** *(utility-99.999)*100000*
7.591	196.950	99.830
8.992	199.389	99.937
10.386	201.616	99.965
11.772	211.288	99.964
13.159	247.884	99.970
14.545	421.867	99.971

Corresponding Plots:




Plot of log(XY) vs utility
(Utility-99.999)*100000
99.96
99.94
99.92
99.90
99.88
99.86
99.84
8
9
10
11
12
13
14
Logarithm of X x Y


Plot of utility vs runtime
Runtime in seconds
400
350
300
250
200
99.84
99.86
99.88
99.90
99.92
99.94
99.96
(utility percentage-99.999)*100000

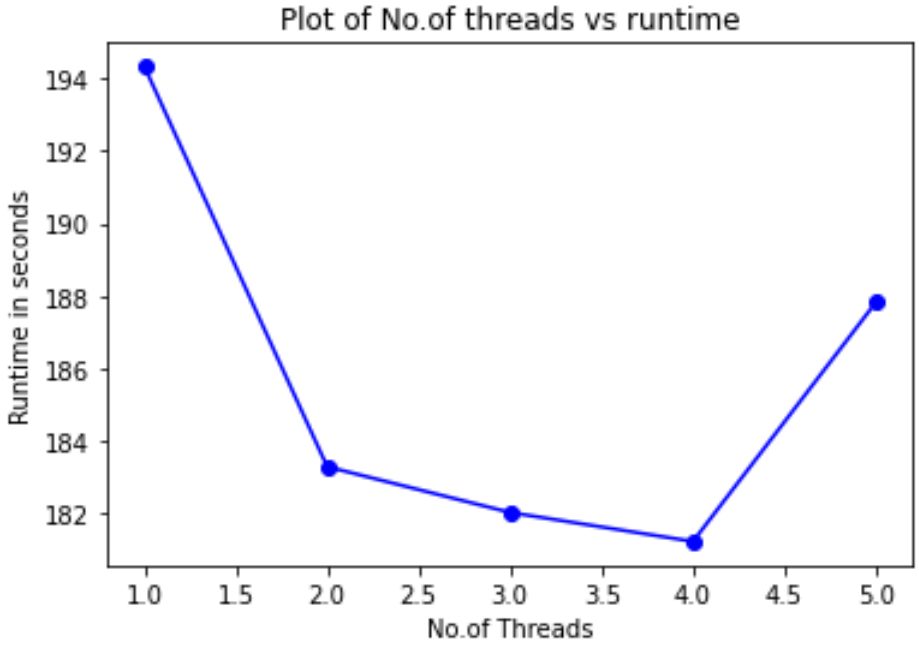
4.3 Using pthreads

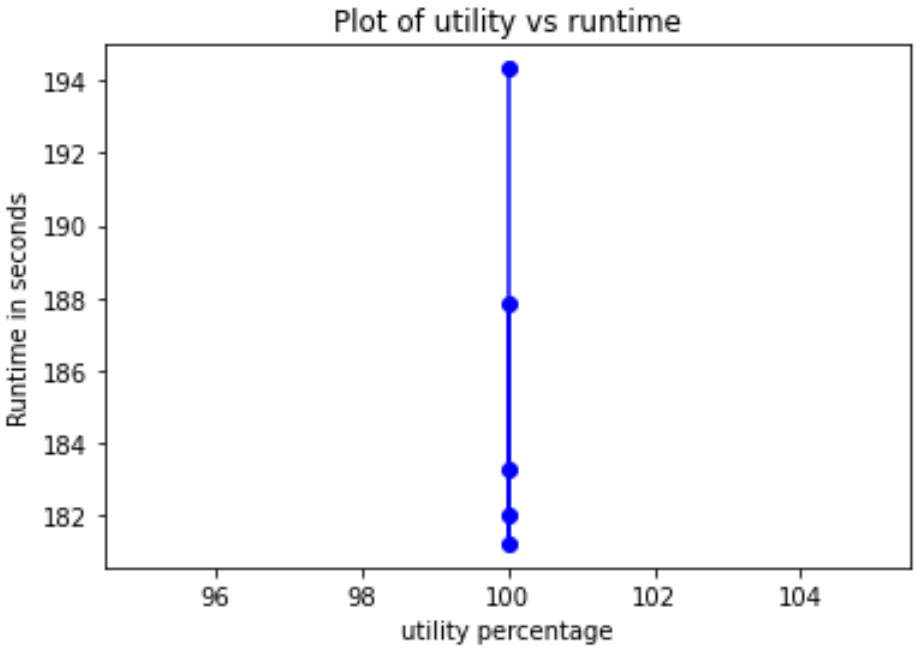
4.3.1 Using distinct threads for consecutive frames

n is the number of threads used.In this case utility is redundant as its value will be 100.

Value of n	Runtime in seconds	Utility
1(Base-Case)	194.334	100.0
2	183.289	100.0
3	182.028	100.0
4	181.228	100.0
5	187.827	100.0

Corresponding Plots:




Plot of utility vs runtime
Runtime in seconds
194
192
190
188
186
184
182
96
98
100
102
104
utility percentage

5 Conclusions

- When we are using pthreads to process consecutive frames,we see that the runtime doesn't change much with the number of threads used.This might be due to the fact that the time taken for creating the thread could be almost same as the time taken by that thread to process the frames.
- We see that decreasing resolution has negligible effect on utility i.e., the error value is less than 0.01% but we see that the runtime is very less with smaller resolutions frames although it takes some time for creating new frames from older frames it is very small(3-4 seconds).
- We see that utility increases with runtime in all cases(except in threads case where error is obviously **zero**) which is evident as for minimizing the error value we need to make more number of computations thereby increasing the runtime.
- We can use mixture of above methods i.e., we can reduce resolution of images and use multiple threads for reducing runtime and with utility of around 99.999% as multiple threads have no effect on utility values.
- When we use multiple threads,it increases the CPU Temperature and so using multiple pthreads is not preferred for embedded systems used in hot conditions.

6 References:

1. https://docs.opencv.org/master/
2. https://learnopencv.com/