# Traffic density estimation using OpenCV functions

Aditya Goel — 2019CS10671
Abhinav Kumar — 2019CS50415

March 31, 2021

## Metrics

- **Utility:** Utility is a metric that essentially depicts how close the queue and dynamic density estimation are to the baseline method. Utility is inversely proportional to Root Mean Square value of relative error in queue and dynamic density estimation using a method with particular parameters as opposed to the baseline method over the whole video.
- **Relative Error:** Relative error is the squared change of estimated queue and dynamic density as obtained from a particular method with certain parameters, with respect to the baseline method.
- **Run Time:** Run Time is measured as the time required for the method to process the whole video from the start of the frames till the end.

## Methods

- **Method1:** The method implemented estimation of queue and dynamic density with sub-sampling of the frames, that is, processing every x frame from the video for used fps. It process frame N and then frame N+x, and for all intermediate frames just use the value obtained for N. It takes the video name and a parameter x for it which is an int value and tell about how many intermediate frame we have to skip which takes same value as previous processed frame.
- **Method2:** The method implemented estimation of queue and dynamic density, reducing resolution for each frame. Parameter for the method is the video name and the resolution, i.e. X and Y which is the resolution XxY. Lower resolution frames might be processed faster, but have higher errors.
- **Method3:** The method implemented estimation of queue and dynamic density, by splitting work spatially across threads (application level pthreads) and giving each thread part of a frame to process. Parameters for this method is the video name and the number of splits i.e. number of threads, where each thread gets one split.
- **Method4:** The method implemented estimation of queue and dynamic density, by splitting work temporally across threads (application level pthreads), and giving consecutive frames to different threads for processing. Parameter to the method is the video name and number of threads.
- **Method_Extra:** The method made the use of sparse optical flow to estimate the dynamic density which will be compared to the dynamic density calculated using dense optical flow. Sparse might be faster, but have errors compared to dense optical flow. Parameter to the method is the video name.

# Trade-Off Analysis

- **Benchmark:** We made the trade-off analysis for the assignment on the video "trafficvideo.mp4" as our benchmark.
  - source: https://www.cse.iitd.ac.in/ rijurekha/cop290_2021/trafficvideo.mp4
- **Baseline:** The output of queue and dynamic density generated by the main function in the code for task2, wherein we found queue and dynamic density, processing every 3rd frame i.e 5fps, used Background Subtraction for calculating queue density and dense optical flow for calculating dynamic density, acts as a baseline for our trade-off analysis in this report.

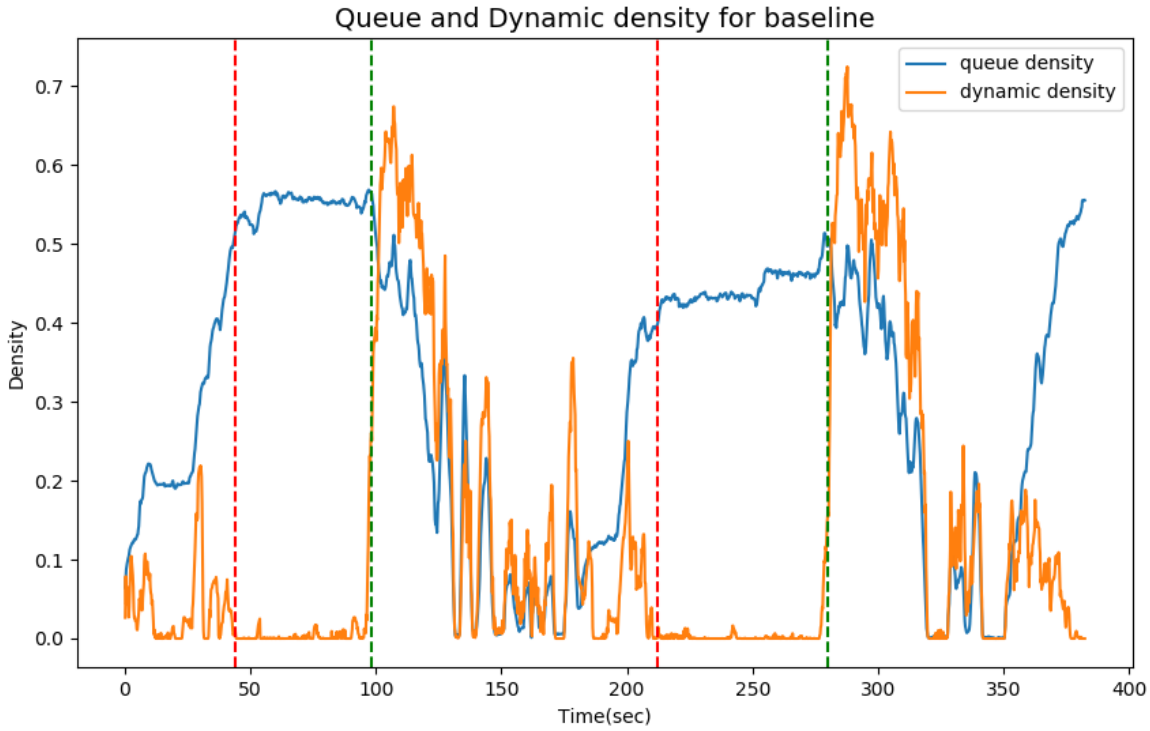



Figure 1: Queue and Dynamic Density by baseline

1. **Method1:** Here the video was processed sub-sampling the frames, i.e. processing every 2nd, 4th, 6th, 8th and 10th frame from the video, as opposed to processing every frame from the video as in the baseline execution for 5fps. It is evident from the Run-Time v/s x graph, that the run-time reduces with increase in the value of x, for the reasons that with increasing value of x, a lower number of frames are required to be processed for a given benchmark. While, with increasing value of x, there in an increase in the RMS value of relative error, as we miss out on relevant data from frames that we skip while estimating queue and dynamic density. Hence the utility decreases with increase in value of x.

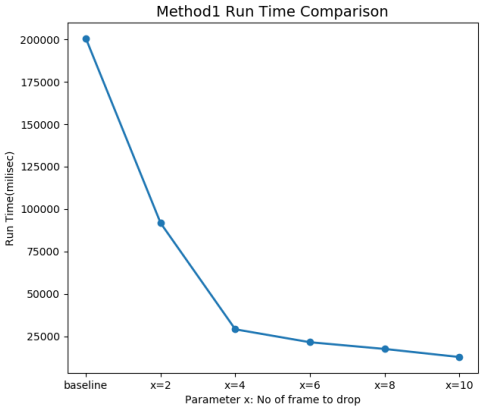
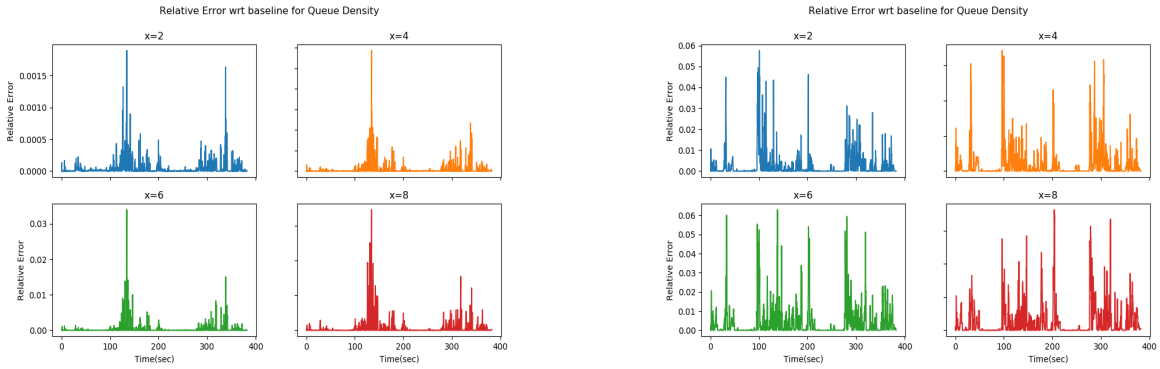


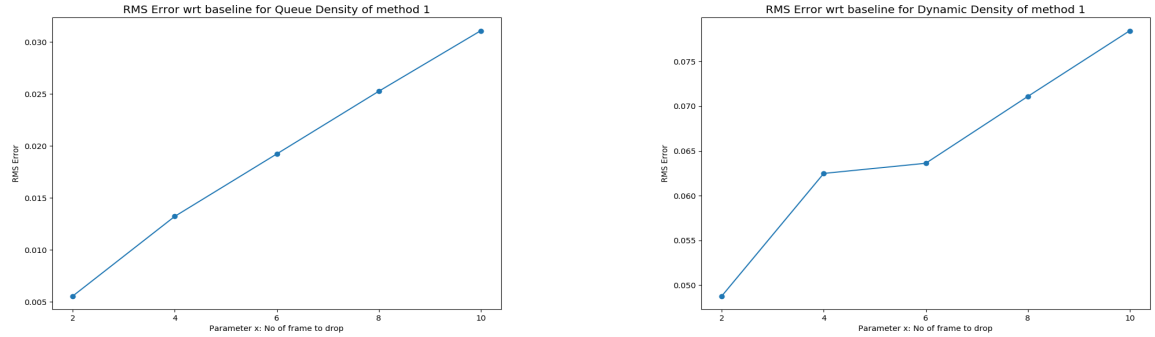

Figure 2: Run Time for method1




(a) Queue Density

(b) Dynamic Density

Figure 3: Relative error v/s time for method1




(a) Queue Density

(b) Dynamic Density

Figure 4: RMS value of Realtive Error v/s x (Number Of Frames Dropped) for method1

2. **Method2:** Here the video was processed using 250x270, 200x600, 150x450, 100x300 as the resolution of each frame as opposed to the baseline resolution of approx 290x900. It is evident from the plot for Run-Time v/s Resolution Graph, that the run-time for a benchmark reduces with decreasing resolution as there is less no of pixels. While the error for different reduced resolution is quite interesting, for just half resolution the error is minimum, so we can't say a general trend for this.

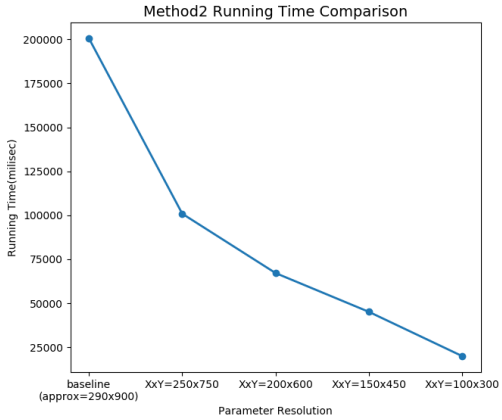
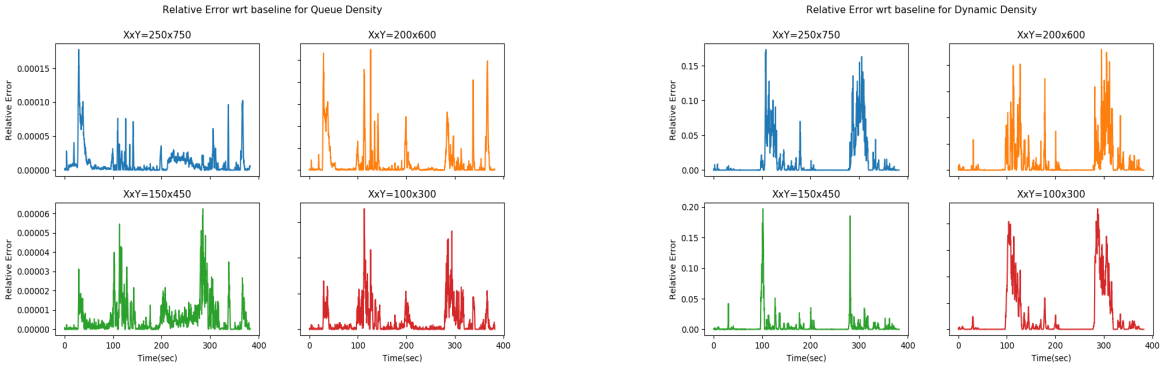


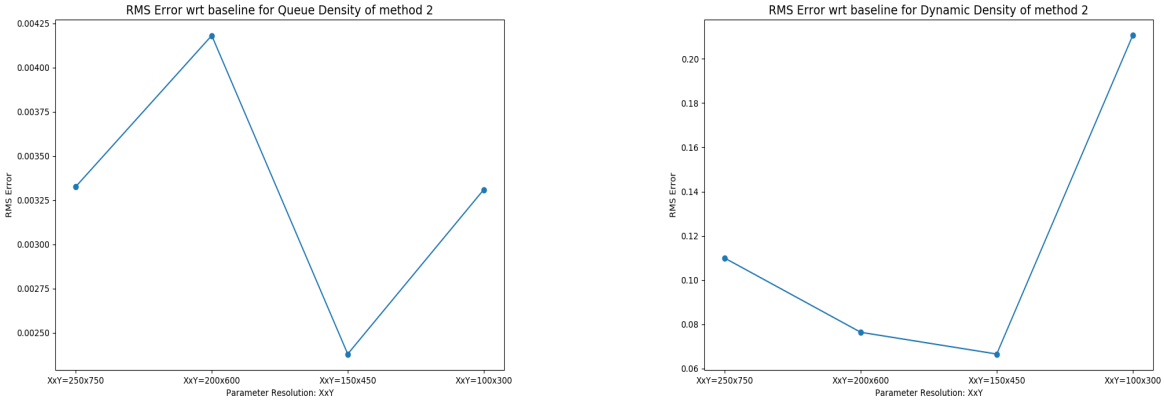

Figure 5: Run Time for method4




(a) Queue Density

(b) Dynamic Density

Figure 6: Relative error v/s time for method2




(a) Queue Density

(b) Dynamic Density

Figure 7: RMS value of Relative error v/s Resolution for method2

3. **Method3:** Here the video was processed, by splitting work spatially between 2, 3, 4 and 5 threads in each case respectively as opposed to a single thread in the baseline execution. For all number of thread the run-time is more than the baseline but among 2,3,4,5 the Run-Time v/s Number of Threads Graph shows a decrease in run-time with an increase in number of threads as we increase the threads from 1 to 4, since a larger number of threads allows for simultaneous processing of the same frame. While the run-time with 5 threads exceeded the run-time with 4 threads, it is due to the maximum usage of CPU which slow down the process.The CPU usage for 4 thread is around 370% of 400% of CPU as of there are 4 core in my pc, but using more thread does not increase CPU usage so that it slow down the process. This method had a lower values of Relative errors and also RMS error when compared to other methods, as for any number of threads, it involved no loss of information for each frame. Hence the utility stays approximately constant for any number of threads.

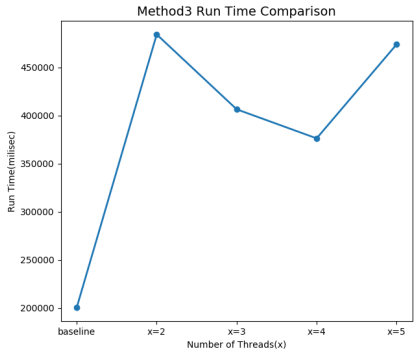
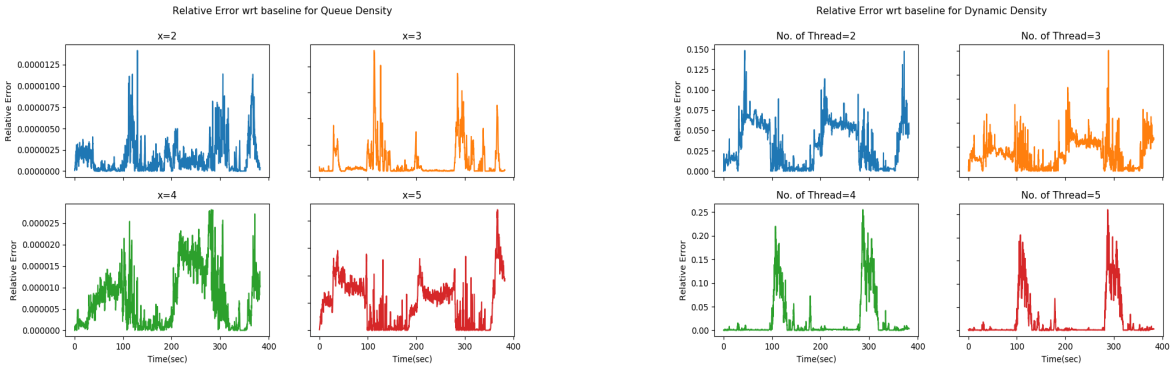


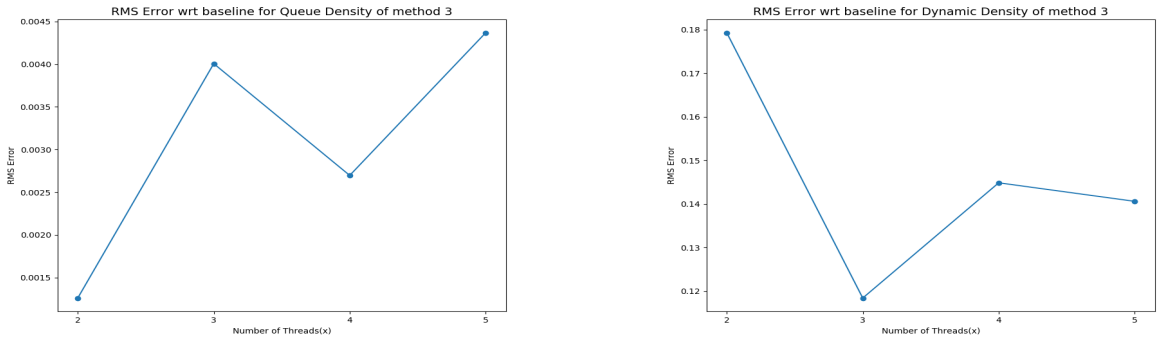

Figure 8: Run Time for method3




(a) Queue Density

(b) Dynamic Density

Figure 9: Relative error v/s time for method3




(a) Queue Density

(b) Dynamic Density

Figure 10: RMS value of Relative error v/s Number Of Threads for method3

4. **Method4:** Here the video was processed by splitting work temporally across threads and giving consecutive frames to different threads, with 2, 3 ,4 and 5 threads been used in each case, as opposed to single thread used in the baseline execution. It can be observed from the Run-Time v/s Number Of Threads Graph, that the run-time reduces with increase in number of threads until a certain number of threads, i.e. 3, due to simultaneous processing of different frames by different threads. While after a certain number of threads, i.e. 3, the run-time increases with increase in number of threads, it is due to the maximum usage of CPU which slow down the process which hence overshadows the benefits of using multiple threads. Also it is evident from the RMS values of relative error, that the RMS error increases with increase in the number of threads, as here the Background Subtraction model and Dense Optical Flow being separately used for each thread, we loose on some details from frames being analysed by different threads. Only for 5 threads the error decrease it might be just by chance because we have tested this only on one video. Hence the utility decreases, with an increase in the number of threads.

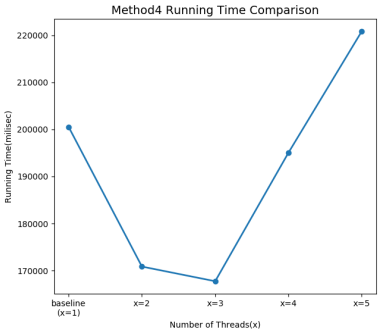
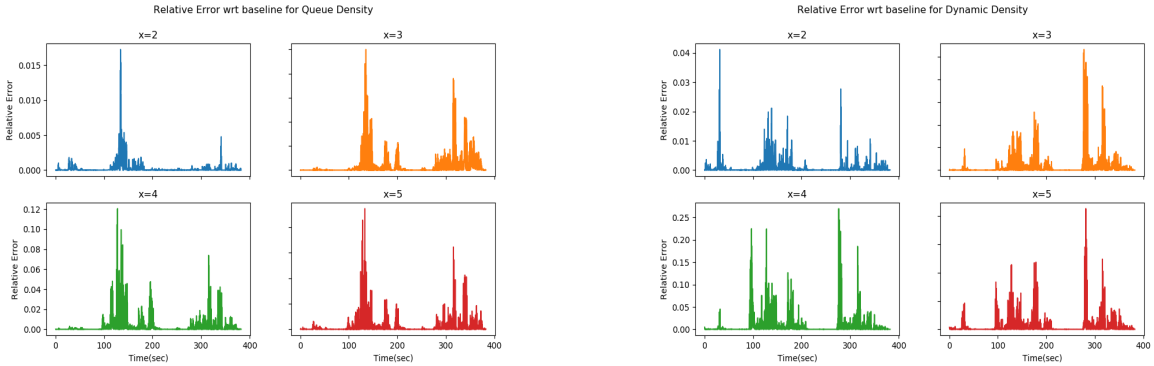


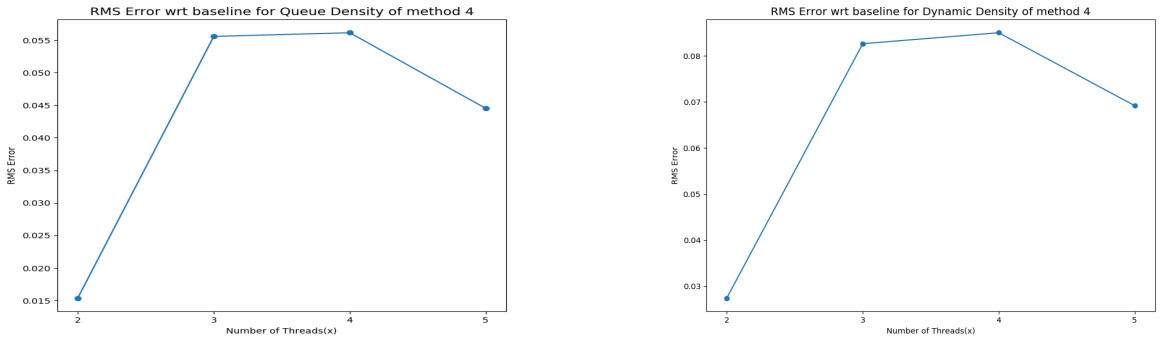

Figure 11: Run Time for method4




(a) Queue Density

(b) Dynamic Density

Figure 12: Relative error v/s time for method4




(a) Queue Denisty

(b) Dynamic Denisty

Figure 13: RMS value of Relative error v/s Number Of Threads for method4

5. **Method_Extra:** While sparse optical flow is based on motion of certain features of objects like edges, corners etc., dense optical flow is based on motion of each pixel of the frame. Hence sparse optical flow results has a very low run-time as compared to dense optical flow(see figure 12), but produces a less accurate result with an RMS value of Relative Error of 0.117. Hence the sparse optical flow for dynamic density, results in a decrease of utility.

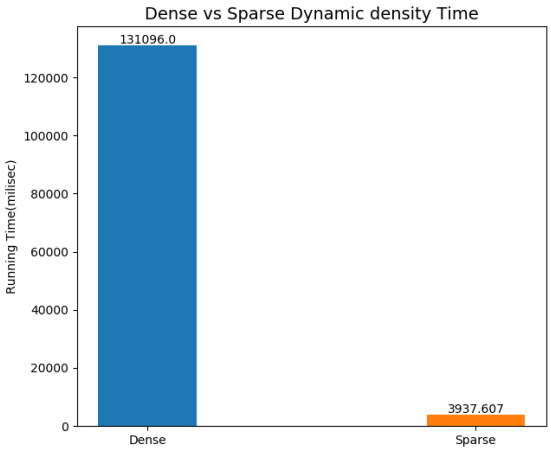
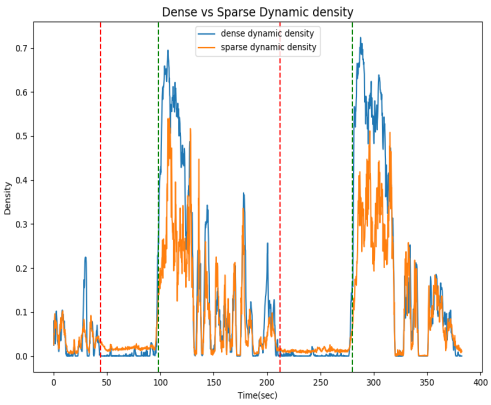
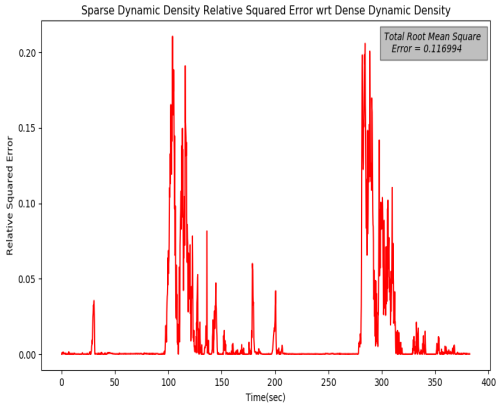



Figure 14: Run time for dense vs sparse optical flow

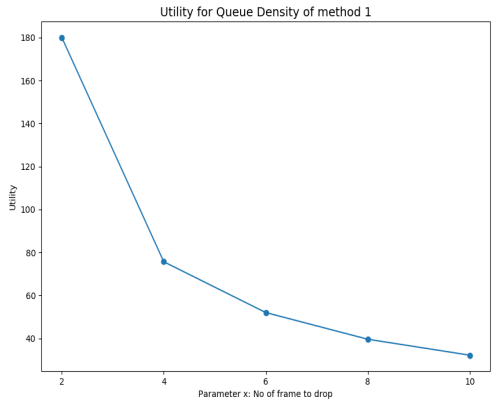



(a) Dynamic Density v/s time as calculated using sparse and dense optical flow

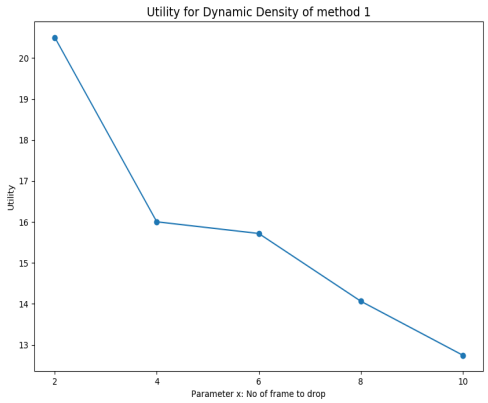



(b) Relative error v/s time

Figure 15: method_extra

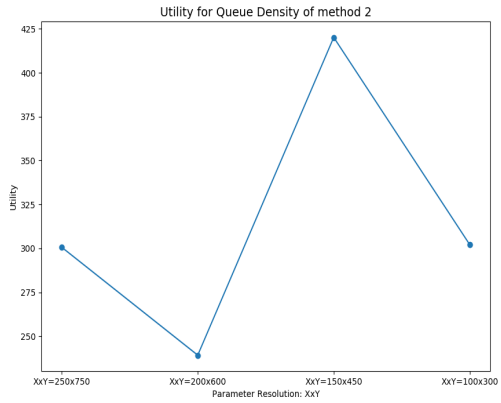



(a) Queue Density

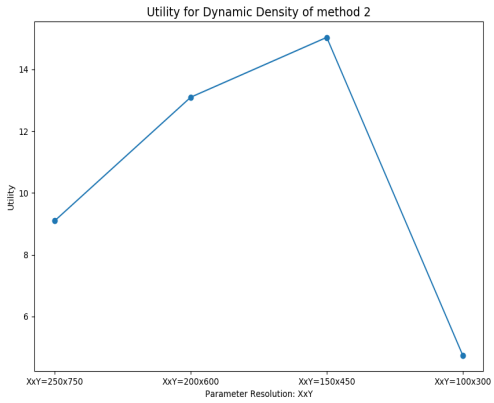



(b) Dynamic Density

Figure 16: Utility Graph for Method 1

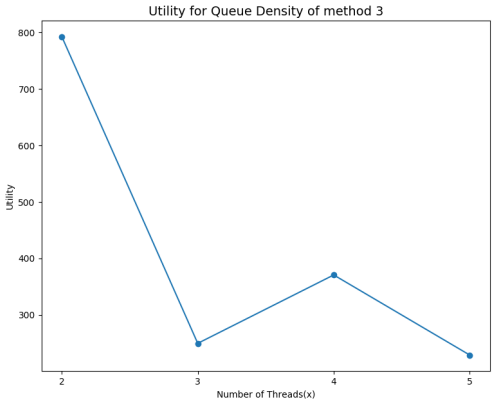



(a) Queue Density

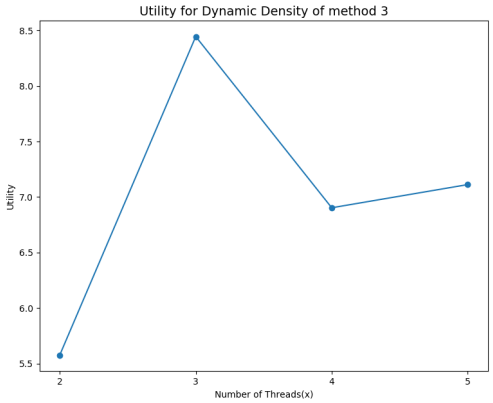



(b) Dynamic Density

Figure 17: Utility Graph for Method 2

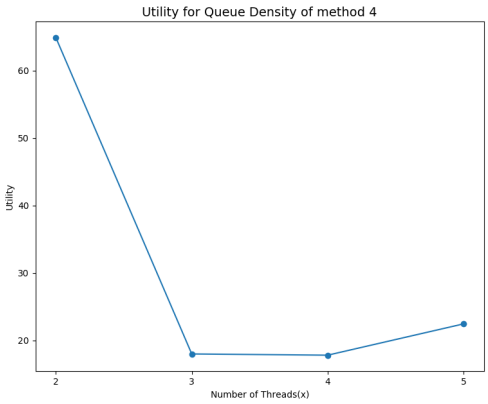



(a) Queue Density

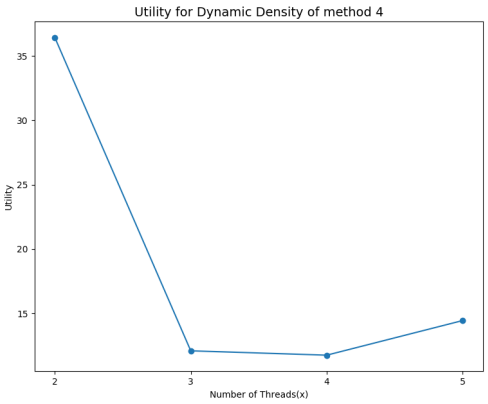



(b) Dynamic Density

Figure 18: Utility Graph for Method 3




(a) Queue Density




(b) Dynamic Density

Figure 19: Utility Graph for Method 4