COP290 Run-time Analysis

Aniket Gupta,
2019CS10327

Aayush Goyal
2019CS10452

March 2021

1 Metrics

1.1 Utility

The error is defined as the average of percentage error over all the frames analysed. For a frame, let d1 be the density obtained by baseline method and d2 be the density obtained by the new method, then Percentage Error = 100*|d1-d2|/d1. The average of these percentage errors gives the error for **Reducing Resolution, Spatial Threading and Temporal Threading methods**.

For **Sub-Sampling method**, let n-1 be no. of contiguous frames we are skipping. For each kth frame analysed in sub-sampling method, we calculate the average of densities obtained by baseline method for frames n*k to n*(k+1)-1. Let this average be $d1_k$ and density for kth frame analysed by sub-sampling method is d2, then

$$PercentError = \sum_{k=0}^{n-1} \frac{(|d1_k - d2|)}{d1_k} * \frac{100}{n}$$

where n is the no. of pages analysed.

1.2 Runtime

Runtime is the time taken in seconds to complete the execution for a particular method and parameter. We have used std::chrono to measure the execution time. Note: all the run-times in this section were done on system with Intel® Core™ i7-8565U CPU @ 1.80GHz × 8 Processor while the system was on airplane mode and no other application was working, and system was under charging. These specs affect the absolute values of run-time, but the trend remains independent of them.

2 Methods

We have implemented following methods for finding Queue density and Dynamic density and have analysed run-time and utility trade-offs between them. The methods are listed below:

2.1 Sub-Sampling

After reading frames from the video, we process each frame to extract some sort of information. Now processing a frame can sometimes be more heavy operation then just reading the frame and most of the times this is the case. So what we do in this method is we skip processing some frames and this saves us the time of processing some frames. The parameter we chose here was no. of frames we are dropping. For the frames dropped we just assume some average value based on the surrounding calculated values. This causes some loss of information but significantly improves the run-time.

2.2 Reducing Resolution

Reducing the resolution of a image means reducing the size of the image. By reducing the size of the image, the number of pixels that we need to process in the frame significantly reduce and thus improving the run-time. Again this leads to loss of information and thus there might be errors.

2.3 Spatial threading

Spatial threading is a parallelization method in which each thread is given different part of the frame to process. Suppose we have 4 threads, then we divided one frame into 4 vertical strips of equal area. Each thread is then given one vertical stripe to process, in this way the processing time of each frame reduced to quarter. The parameter that we choose here was no. of threads we are dividing the current frame into. Number of vertical stripes made are equal to the number of threads. There is no overall information loss in this case, but there still might be some errors because while processing a pixel of the frame, we sometimes need the surrounding values to that pixel, in these cases there might be some errors because of splitting the frame into different parts and unavailability of some pixel values.

2.4 Temporal threading

Temporal threading is a kind of paralleization in which we split the work temporally. What we have done in temporal threading is, we pass consecutive frames into different threads for processing. We read the video sequentially but process the frames parallely. For example let's say we have 4 threads. First frame is passed into thread_1 then we read next three frames and pass them into the corresponding threads. After reading 4 frames, now we will have to use the thread_1 again, so first we read the frame, then change it's warp perspective and crop it. After this we wait for the thread_1 to join. Similarly for rest of the threads. Now there is no overall loss of information in this and the results are pretty much accurate. Run-time reduced significantly and there was absolutely no error for Queue and Dynamic density calculation.

2.5 Sparse Optical flow

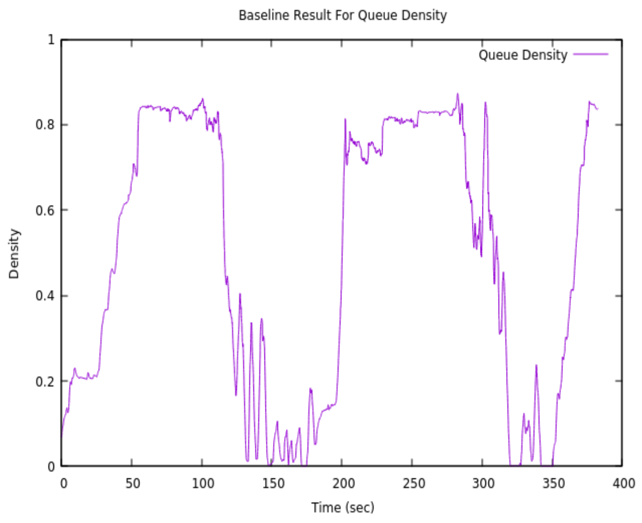
For calculating Dynamic density we used Dense optical flow. In dense optical flow we track the movement of every pixel of the image. Tracking every pixel of the image is quite time consuming for the algorithm, but this gives accurate result since we are tracking every pixel of the image. Now in sparse method, we first choose some feature points to analyse. The feature points are mainly the points where there is rapid change in pixel gradient, like corners in different objects. Then only motion of these points are observed, which requires much less time than dense flow method. The feature points are updated in each frame for better result.

3 Trade-off Analysis

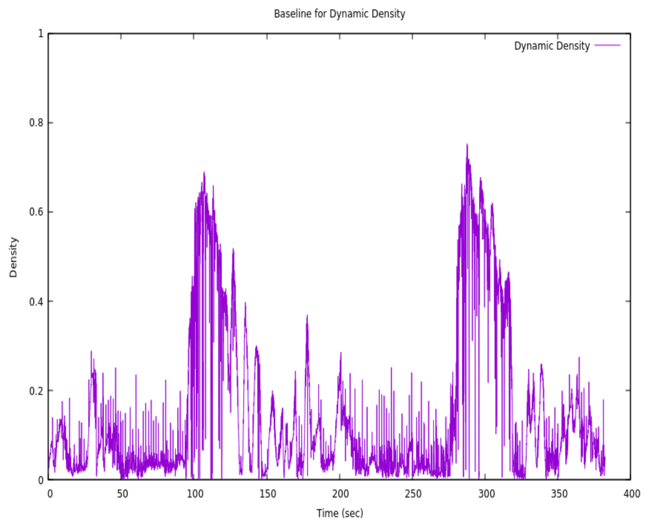
After execution of each method we collected the values of Queue Density and Dynamic density. Then we compared these values to the baseline values in which there was no loss of information and no kind of parallelization. We have analysed the effects of changing parameters for different methods on the run-time of programme and the ulitility of the method. Here we will analyse the Percentage error values v/s parameter. The lesser the error, more is the utility of that method.

The result obtained using baseline method is summarised below:

- **Queue Density:**
  Runtime- 57 sec



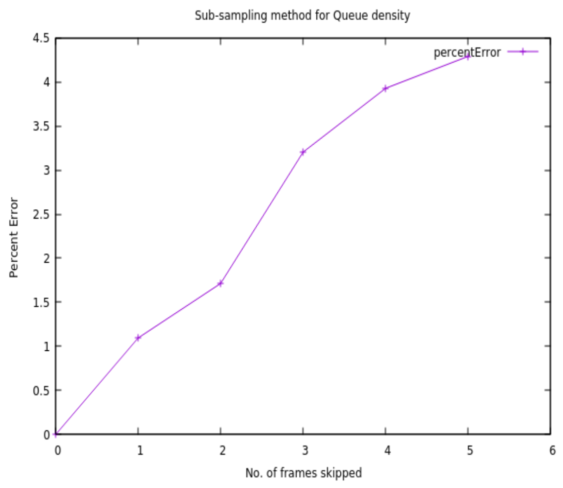
- **Dynamic Density:**
  Runtime- 397 sec

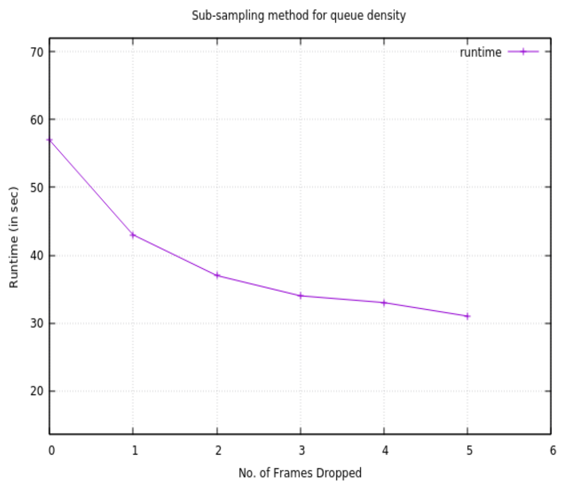


3.1 Sub-Sampling

3.1.1 Queue Density

Frames skipped	Run-Time	Percentage error
0	57	0
1	43	1.0952
2	37	1.71351
3	34	3.20353
4	33	3.92634
5	31	4.28856

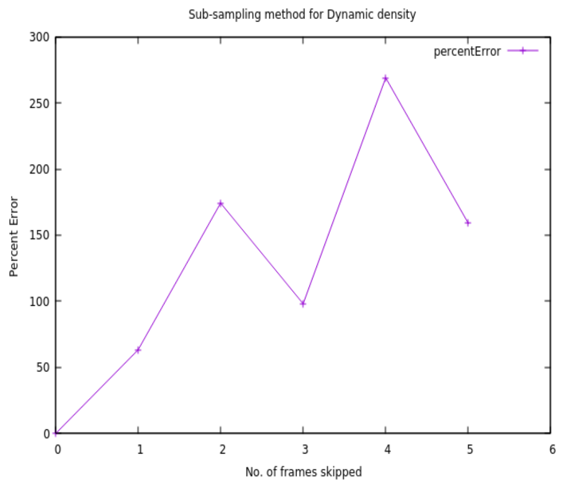


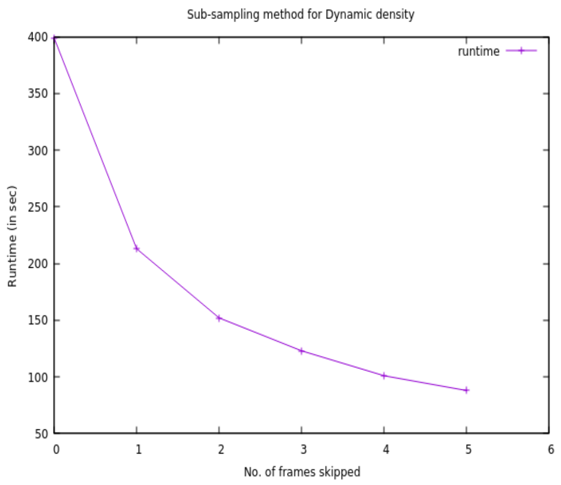


Explanation:
For this method, the parameter is no. of frames skipped for each analysed frame. As the parameter increases, the amount of data loss also increase which leads to increase in error. Skipping greater no. of frames means that we need to analyse lesser no. of frames. Thus, the runtime decreases with increasing parameter, as expected.

3.1.2 Dynamic Density

Frames skipped	Run-Time	Percentage error
0	399	0
1	213	63.3783
2	152	174.249
3	123	98.1554
4	101	268.924
5	88	159.076



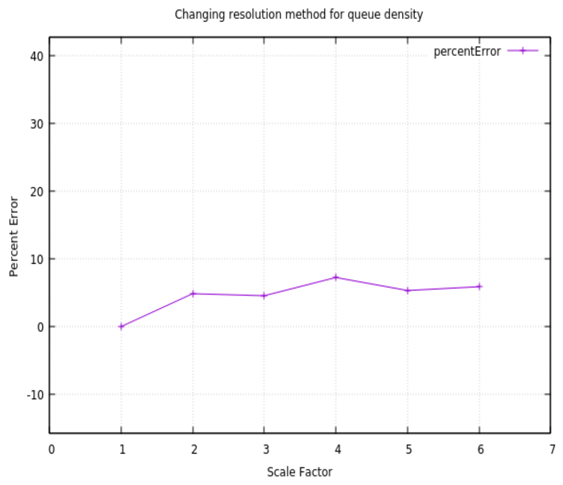


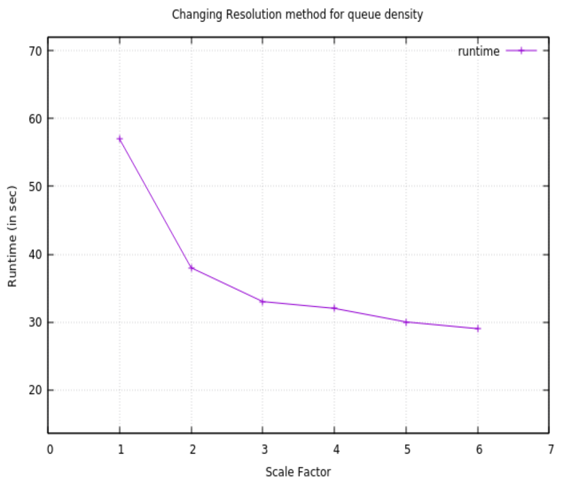
Explanation:
For this method, the parameter is no. of frames skipped for each analysed frame. The error is increasing with some valleys as the parameter increases. These valleys occur because the error will also depend on type of frames being skipped along with the no. of frames being skipped. If frames with higher dynamic density are skipped, then error will be larger. But it can be seen that the decrease in error for unit increment in parameter is less than increase in error for similar increment in parameter at almost same parameter value and thus the error is increasing due to increase in parameter. The runtime decreases on increasing parameter as lesser no. of frames need to be analysed, and processing frame is the most heavy operation since we are using Dense Optical flow.

3.2 Reducing Resolution

3.2.1 Queue Density

Scale Factor	Run-Time	Percentage error
1	57	0
2	38	4.86794
3	33	4.56993
4	32	7.25928
5	30	5.34017
6	29	5.89267



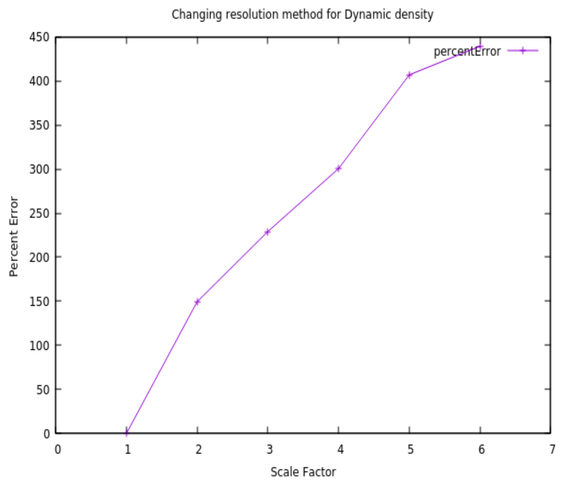


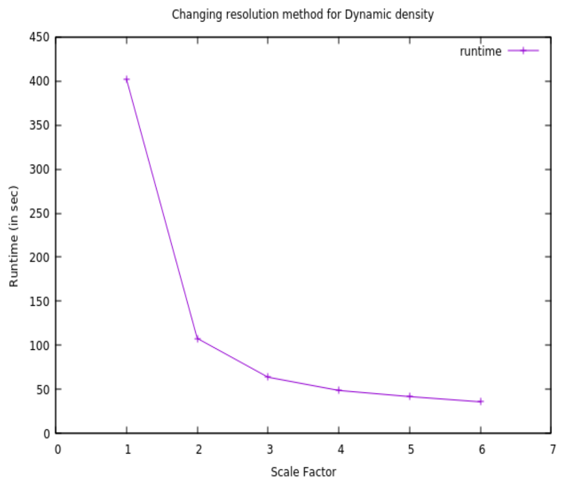
Explanation:

For this method, parameter is Scale factor. A scale factor of 2 means that we reduce the height and width of the image by $\frac{1}{2}$. The no. of pixels that we will have to process in the frame will thus reduce by $\frac{1}{4}^{th}$. Thus increasing this scale factor significantly reduces the time taken to process each frame. Since there is some loss of information, error mostly increases with increasing scale factor. One can use a scale factor of 5 as error is tolerable and there is significant improvement in the run-time.

3.2.2 Dynamic Density

Scale Factor	Run-Time	Percentage error
1	402	0
2	108	149.294
3	64	229.026
4	49	300.435
5	42	406.719
6	36	439.209



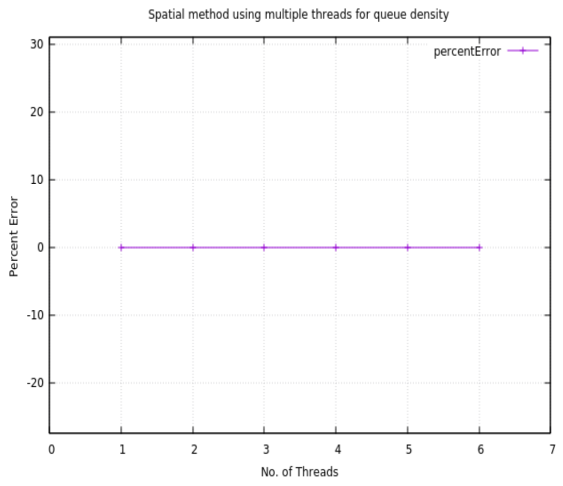


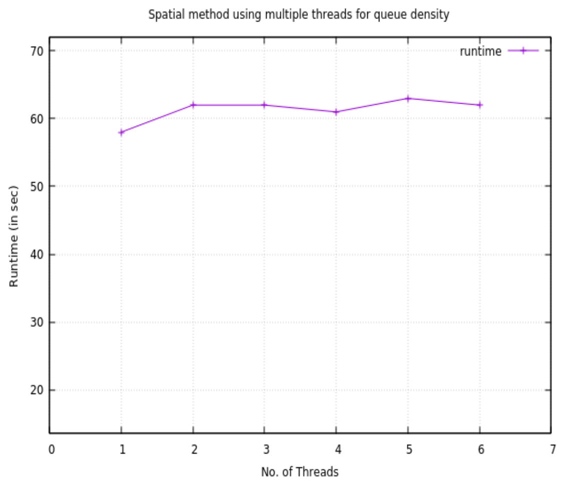
Explanation:
In case of Dense optical flow also we can see that reducing area has caused a significant decrease in the run-time. In the start we can see that as we have decreased the area by $\frac{1}{4}$ the run-time goes from 402 to 108. After this the slope of decrease in run-time is quite low but the error value is shooting up. Thus an ideal case for Reducing resolution would be to Reduce by atmost a scale of 2.

3.3 Spatial threading

3.3.1 Queue Density

No. of threads	Run-Time	Percentage error
1	58	0
2	62	0
3	62	0
4	61	0
5	63	0
6	62	0

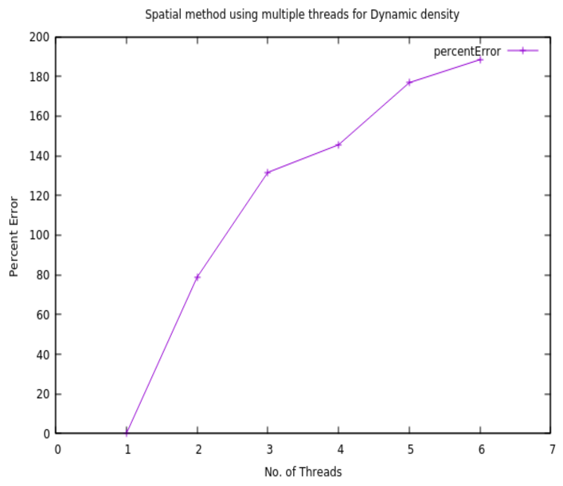


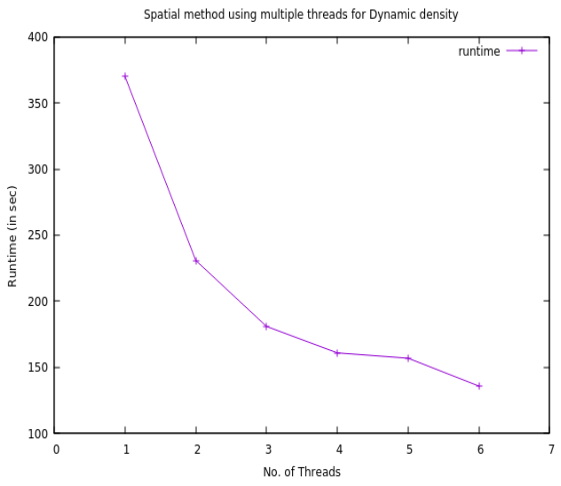


Explanation:
In this method, parameter is No. of threads. Based on number of threads we divide the work spatially. If the number of threads are 4, then we divide the frame into 4 equal parts. Now each thread has to process only quarter of the frame. This decreases the processing time. In case of Finding Queue Density we are doing a very light operation of "absdiff" between 2 frames. But still we are reading the frame sequentially and only after changing the warp perspective we are processing the frame in different threads. In our implementation, the processing of frame runs parallel to reading and warping the next frame. And since Reading and warping is the main time consuming process, the overall run-time depends on Reading and warping the frame, which is done sequentially. So the total processing time remains almost same for any no. of threads and is equal to the time taken for reading and warping each frame. Also in queue density calculation, all pixels are independent and there is no overall loss of information. Hence the total error in this case is ZERO.

3.3.2 Dynamic Density

No. of threads	Run-Time	Percentage error
1	370	0
2	231	78.706
3	181	131.605
4	161 14	5.497
5	157 176	.832
6	136	188.427



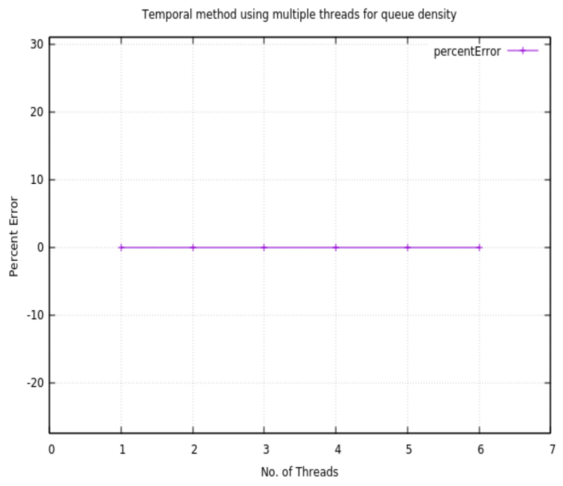


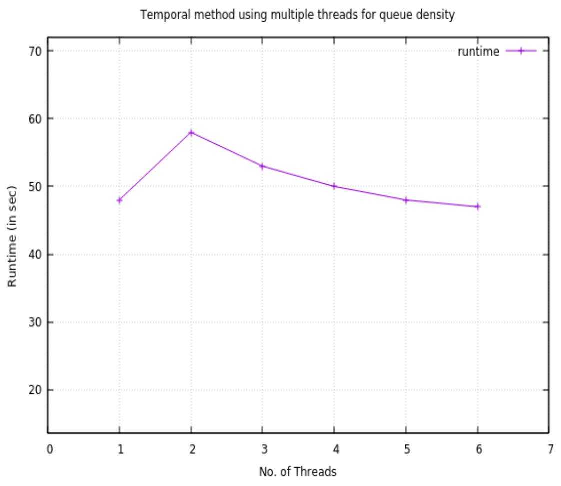
Explanation:
In contrast to Queue Density, processing a frame using dense optical flow for Dynamic Density is quite a heavy operation. So by dividing the work of processing a frame among different threads, there is a significant decrease in the run-time. As we can see that using 2 threads, the run-time almost decreased to half. But since dense optical flow requires the value of neighbouring pixels also, there is a significant increase in the error. Thus it would be optimal to only divide the frame into atmost 2 parts.

3.4 Temporal threading

3.4.1 Queue Density

No. of threads	Run-Time	Percentage error
1	48	0
2	58	0
3	53	0
4	50	0
5	48	0
6	47	0

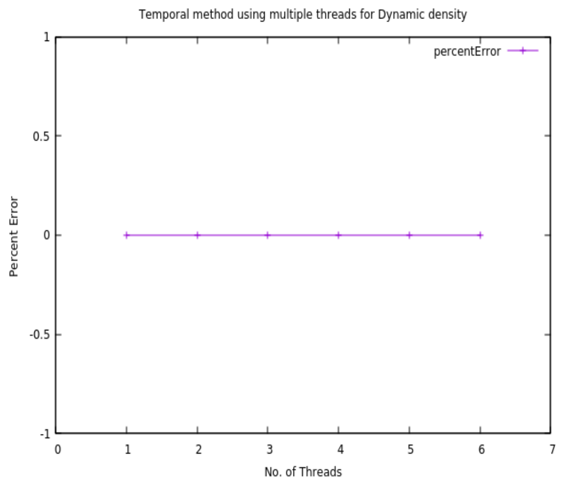


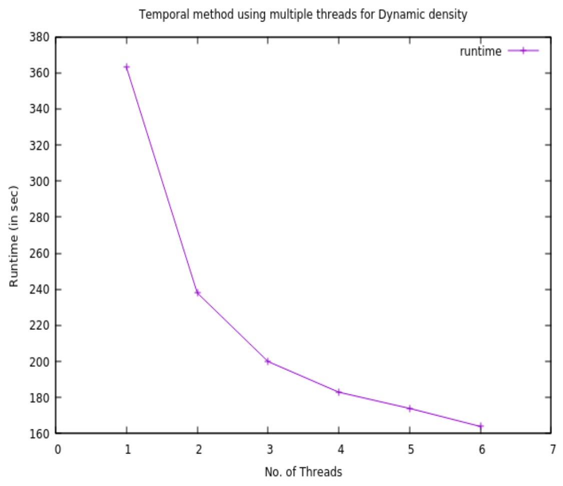


Explanation:
The parameter for this method is no. of threads being used. For queue density, only current frame and background image is required and thus even on using multiple threads, it is easily ensured that the frame-wise operation remains unchanged and thus their is no error even on using multiple threads. On using multiple threads, multiple frames can be processed parallely at the same time. This tends to decrease time of execution. But creating and destroying multiple threads also consume some time and it tends to increase time of execution. On using 2 threads, the gain due to processing 2 frames parallely is less than loss due to time taken to create and destroy threads and thus the execution time increases. On further inceasing no. of threads, the gain due to processing multiple frames parallely overcomes the loss due to time taken to create and destroy threads and thus the execution time decreases on further increase in parameter.

3.4.2 Dynamic Density

No. of threads	Run-Time	Percentage error
1	363	0
2	238	0
3	200	0
4	183	0
5	174	0
6	164	0

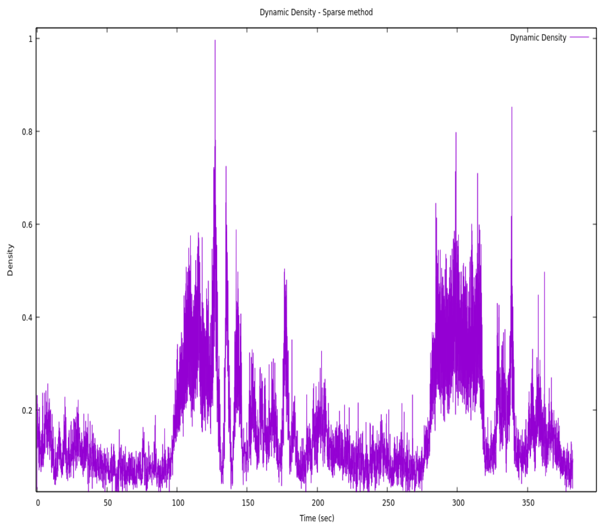




Explanation:
The parameter for this method is no. of threads being used. For calculating dynamic density at a particular frame, we need current and the previous frame. The previous frame can be stored while reading the video file in main() function and can be easily passed to different threads along with the current frame. This again ensures that the frame-wise operation remains unchanged and thus the error remains zero on increasing parameter. Using multiple threads ensure that the processing of next frame does not need to wait till the process on current frame is complete and thus multiple frames can be processes parallely at the same time. Thus, the runtime decreases on increasing parameter.

3.5 Sparse Optical flow

1. **Runtime:** 151sec
2. **Percent Error with respect to baseline (Dense Optical Flow):** 102.217



The time taken to run sparse optical flow is much less than that taken by dense optical flow. In dense optical flow, motion of all the points in a frame is analysed whereas in sparse optical flow, only selected feature points in a frame are analysed. Thus, analysing lesser no. of points tends to lower the runtime. Selecting the feature points is an additional step in sparse optical flow method. This tends to increase the runtime but this increase is small as compared to the former decrease and thus, the runtime decreases.
Analysing lesser no. of points means that much information is lost per frames. For example, vehicles with smaller variation in colour gradient and with less no. of feature points will contribute less to the density. Thus, the error is large as compared to baseline. The pattern observed in the plot is similar to dense optical flow but the actual values varies a lot and thus the error value observed is high.

4 Inference

From the above Run-time v/s utility trade-offs, we conclude that Temporal threading is the best method one can use to decrease the run-time and still there was no loss of information. The error obtained using Temporal threading was absolutely zero from the baseline, and still the run-time decreased significantly for both Queue Density and Dynamic Density Calculation. In some methods we saw that it was only optimal to vary the parameter values upto some extent only.