COP290 Assignment 1

Kuldeep Meena(2019CS10490) and Aryan Chandra(2019CS10333)

April 1, 2021

1 Introduction

When we build software, accuracy of the output or utility might not be the only metric to optimize.For example while estimating traffic density correct results only cannot be a parameter for the software, other parameters like latency,throughput,energy utilised,temperature rise etc are also important parameter of consideration. Latency is important as we often would like to get the result in live which requires the software to run considerably fast.Energy requirements is also an important consideration as the software might run on a low power battery in practice for which a low power consuming software is required. Temperature rise during runtime is important paramerter as often in a city like Delhi temperature is high during summers and it is required the software does not overheat too much.
Here we have used two such parameters:

- Runtime of function
- Utility of function(root mean square error against ideal values obtained in subtask-2)

We have done the above analysis for the following methods:

- Sub-sampling frames(Parameter: Number of frames skipped)
- Resolution(Parameter: Aspect ratio)
- Split work spatially across threads(Parameter: Number of parts of frame)
- Split work temporally across threads(Parameter: Video parts / number of threads)

2 Some design decisions

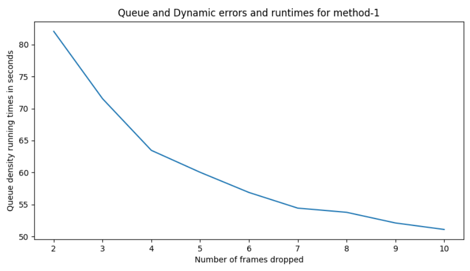
- The graphs shown are based on the runtimes and utility obtained on an ubuntu virtual machine with 4 cores assigned

3 Sub-sampling frames(Parameter: Number of frames skipped)

The method takes number of frames skipped as the parameter. Increasing number of frames makes the computation of dynamic and queue density faster as now the data to compute is less.

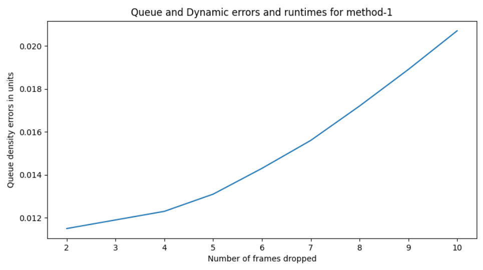
3.1 Queue density runtime

Graph shown below depicts the runtime for Queue density calculation against number of frames dropped. From the run time it is clear that the function is monotonically decreasing which is because of less frames to be computed when number of frames dropped is increased.



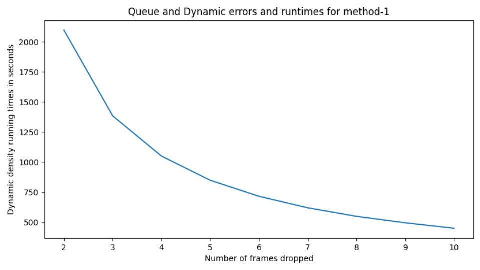
3.2 Queue density utility

Graph shown below depicts the utility(root mean square error against ideal values from subtask-2) for Queue density calculation against number of frames dropped. From the graph, it is clear that the function is monotonically increasing which is because of the fact that although we are able to reduce the amount of computation by increasing number of frames skipped but also we lose a lot of information regarding the result and so utility decreases with increase in number of frames skipped



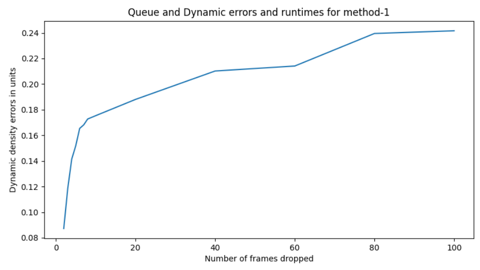
3.3 Dynamic density runtime

Graph shown below depicts the runtime in seconds for Dynamic density calculation against number of frames dropped. From the run time it is clear that the function is monotonically decreasing which is because of the fact that we are able to reduce the amount of computation by increasing number of frames dropped. Also the runtime reduced is much more sharper as compared to queue density which might be because of comparatively slower function Dense optical flow used for dynamic density estimation and so skipping frames make the overall computation much faster as compared to simple background subtraction in queue density estimation.



3.4 Dynamic density utility

Graph shown below depicts the utility(root mean square error against ideal values from subtask-2) for Dynamic density estimation against number of frames dropped. From the graph, it is clear that the function is monotonically increasing which is because of the fact that although we are able to reduce the amount of computation by increasing number of frames but also we lose a lot of information regarding the result and so utility decreases with increase in number of frames skipped. The utility is not exponentially increasing as in the case of queue density estimation which might be because of narrower peak time intervals obtained in ideal values in subtask-2 whereas there were wider peaks in queue density estimation in subtask-2.

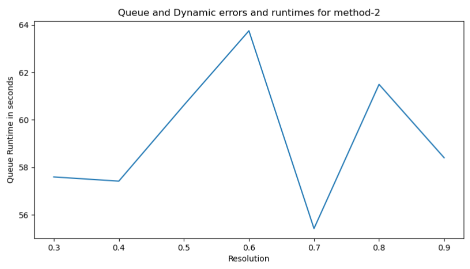


4 Resolution(Parameter: Aspect ratio)

The method takes aspect ratio for resolution as the parameter. Images with lower resolutions can be computed faster as compared to images that have higher number of pixels per inch.

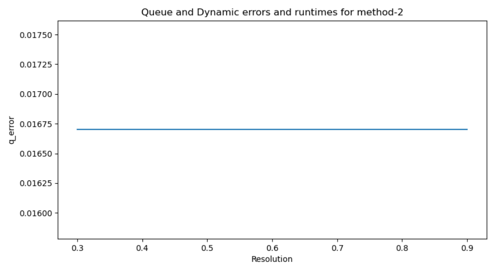
4.1 Queue density runtime

The general trend is zig zag which might be because of trade off between improved runtime of background subtraction to find queue density with an extra task of image resizing for every frame. But during the later half, the graph shows a decreasing behaviour of runtime.



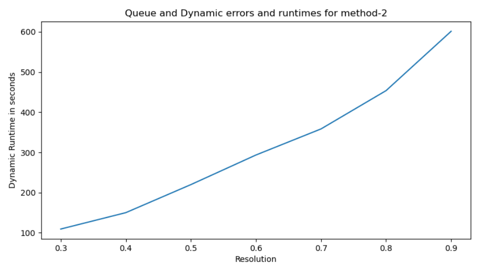
4.2 Queue density utility

The error is almost constant, it is probably because we are scaling both the empty image as well as the frame and the objects in queue are rectangles finitely large and so pixels are lost in both empty as well the frame image and density is the ratio of areas so the error is less the 0.0001 and hence the graph is almost constant for all resolutions. The finite error is because of ideal values are obtained for every frame where as in part c we have skipped 5 frames.



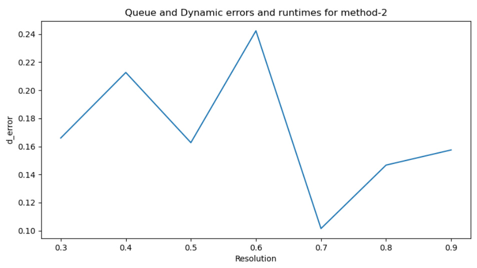
4.3 Dynamic density runtime

Graph shown below depicts the runtime in seconds for Dynamic density calculation against aspect ratio or resolution. From the graph, it is clear that more is the resolution slower is Queue density estimation as pixels are more in a higher resolution image and dense optical flow takes into count every pixel and is costlier function as compared to resizing so runtime is better for lower resolutions.



4.4 Dynamic density utility

Graph shown below depicts the utility(root mean square error against ideal values from subtask-2) for Dynamic density estimation against resolution of image. The general trend from the graph is that lower the resolution, higher is the error which can be explained by the fact that by reducing resolution we reduce information available for dense optical flow to estimate moving pixels.Some abnormal highs and lows might be because of ideal Dynamic density having narrower peaks so the error does not often indicate the real estimate of possible error.

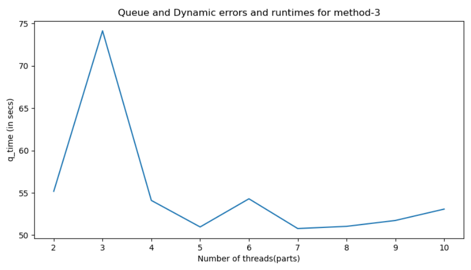


5 Split work spatially across threads(Parameter: Number of parts of frame)

The method takes number of threads or the number of parts in which the frame is to be divided. More threads often mean more availability or opportunity to do several tasks in parallel so it is often associated with improved runtime.

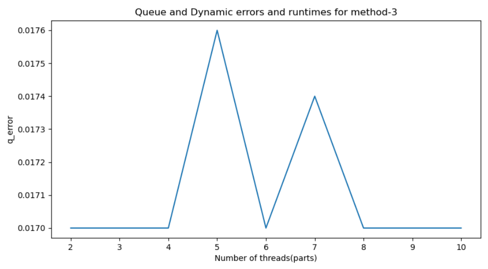
5.1 Queue density runtime

The general trend is improving runtime with increase in number of threads. Discrepancies at some points might be because of trade off between time for creating new threads and its deployment vs improvement in runtime. Also with large number of threads because of extra threads being sequential and not parallel there is slight increasing trend later on in the curve.



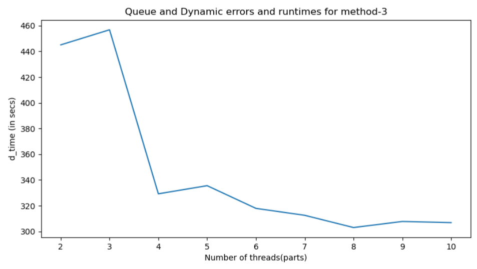
5.2 Queue density utility

The error is almost constant, it is probably because the implementation divides both images into parts and finds the difference of all the parts individually which is quite same as finding difference of two complete images.So the utility is only the error because of skipping 5 frames instead of taking every frame.



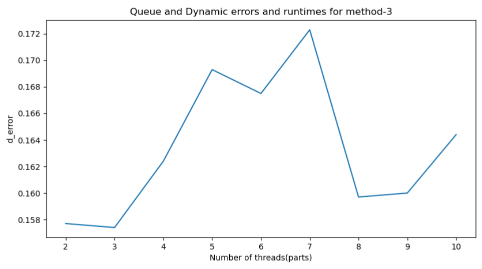
5.3 Dynamic density runtime

Graph shown below depicts the runtime in seconds for Dynamic density calculation against number of threads available.From the graph it is clear that runtime or performance is better when more number of threads are available as now multiple parts of frames can be computed simultaneously on multiple threads thus improving performance. Slight anomalies might be because of trade off between improved performance and time utilised in deployment of new threads.



5.4 Dynamic density utility

Graph shown below depicts the utility(root mean square error against ideal values from subtask-2) for Dynamic density estimation against number of threads available. The general trend is increasing error or decreasing utility as we deploy more threads as we reduce the information available for Dense optical flow method by dividing image into parts. The abnormal behaviour at some points might be because of the fact that dynamic density has narrower peaks as compared to queue density and hence it is possible that the error calculated is not the real representation of the error.

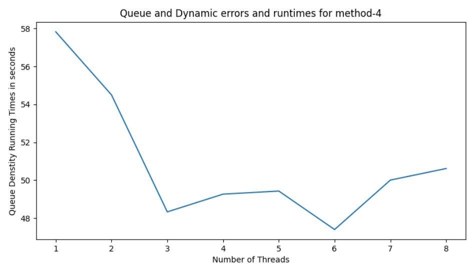


6 Split work temporally across threads(Parameter: Video parts / number of threads)

The method takes number of threads or the number of parts in which the video is to be divided. More threads often mean more availability or opportunity to do several tasks in parallel so it is often associated with improved runtime but because of less data available often has low utility.

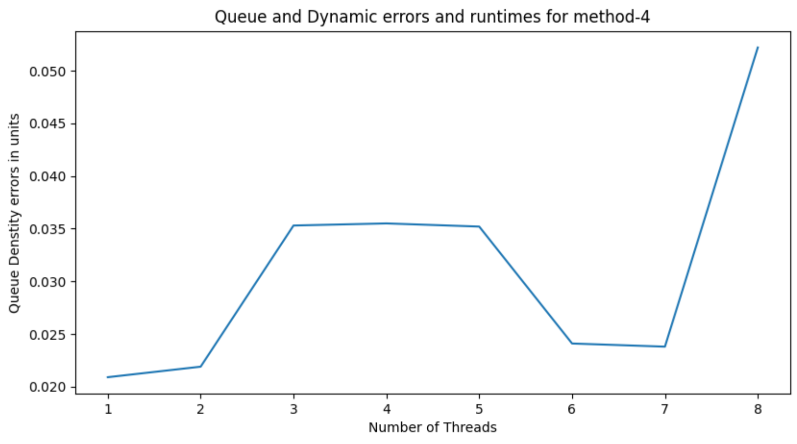
6.1 Queue density runtime

The general trend is improving runtime with increase in number of threads available.Slight discrepancies at some points might because of trade off between time required for creating new threads and its deployment vs improvement in runtime of a simple background subtraction function. Also with large number of threads because of extra threads being sequential and not parallel there is slight increasing trend but in general increasing threads often improves runtime.



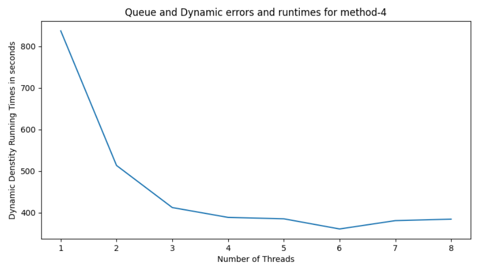
6.2 Queue density utility

The general trend from the graph is clear that with increase in number of threads or the parts in which video is divided, the error gets increased and hence utility of queue density estimation gets decreased . The increasing trend is because of more number of boundary parts of video when threads are increased or because of multiple threads utilising same empty Image.



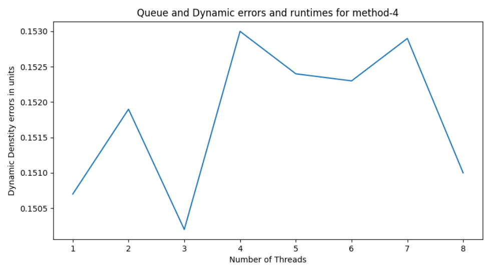
6.3 Dynamic density runtime

Graph shown below depicts the runtime in seconds for Dynamic density calculation against number of threads available.From the graph it is clear that runtime or performance is better when more number of threads are available as now multiple parts of videos can be computed simultaneously on multiple threads thus improving performance. The graph does not have anomalies as there is significant improvement in performance as compared to simple background subtraction function in queue density estimation function.



6.4 Dynamic density utility

Graph shown below depicts the utility(root mean square error against ideal values from subtask-2) for Dynamic density estimation against number of threads available. The general trend is increasing error or decreasing utility as we deploy more threads as we reduce the information available for Dense optical flow method by dividing video into several small parts. The abnormal behaviour at some points might be because of the fact that dynamic density has narrower peaks as compared to queue density and hence it is possible that the error calculated is not the real representation of the error.Also same video file is used by several threads and might too contribute to anomalies.



7 Conclusion

. We have discussed all the methods and have analysed runtime and utility for different parameters of the method. From the discussion above in general we can conclude that in general runtime and utility are complement of each other and always have trade off one when we try to improve the other. Further possibilities might be to make a method that is combination of two methods described above for example a combination of reducing resolution with threading is a possible idea.