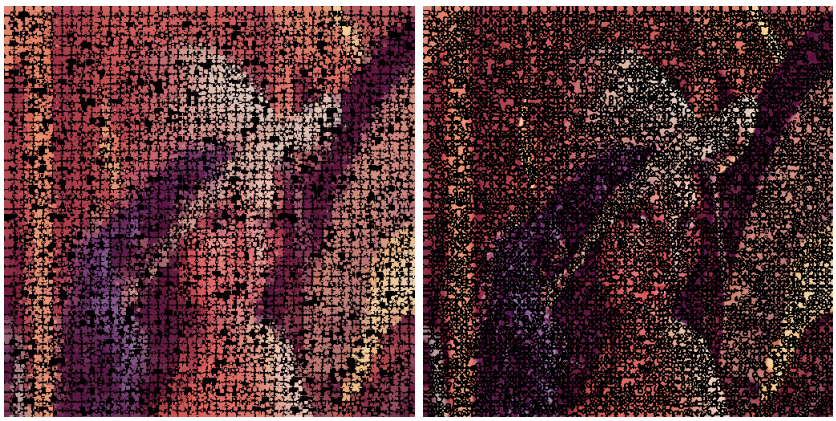
Fuzzy SLIC

Sumanth Tangirala - 201601105
Jalansh Munshi - 201601042
Avinash Arekatla - 201601100
Prayag Reddy - 201601007

Dhirubhai Ambani Institute of Information and Communication Technology

1 Description About the Problem

The main objective of our project is image segmentation. We perform the segmentation task by grouping the pixels which are homogeneous. This method is called the superpixels method, which is usually used as a pre-processing step for computer vision applications. The superpixel method generates an over-segmented image which preserves the overall structure of the image. One of the main advantages of this method is that the computational cost can be reduced by substituting thousands of pixels with fewer (hundreds) of superpixels. The segmentation we aim to optimise in our work is fuzzy segmentation where not all pixels belong to a superpixel and if it is found that a pixel is is disimilar to its surrounding superpixels, then it is called an undetermined pixel. This helps in ensuring that the superpixels are noise free and homogenous. Below are our results with the following image:



The above images show the result of fuzzy segmentation. As can be seen in the rightmost image, some pixels have been turned black since they have been determined as fuzzy or noisy pixels The serial algorithm involves four major steps: Cluster Initialisation, Finding Overlapping Search Regions, Adjustment of Clusters and computation of Degree of Membership and finally assignment of pixels to clusters. Cluster Initialisation involves placing cluster centers at equally spaced grids with a distance of S, which is determined on the basis of number of superpixels, intially and then perturbing them to a nearby position with lowest gradient. Each cluster is then assigned a search region of 2Sx2S and pixels falling in these regions are potential members of these clusters. Regions where 2 or more clusters overlap will therefore be formed and these regions are to be found. Then we iterate over these regions while computing each pixel's degree of membership, which determines the probability of belonging to a cluster, and also moving the clusters according to its members. The final degrees of membership is then utilised to determine if a pixel is undetermined or determined and if detemined then to which cluster.

Since the problem involves determining the superpixel group for each pixel which is basically a group of neighbouring pixels, spatial locality comes into play in the implementation. Temporal locality is also present here because a pixel is accessed for its neighbouring pixels as well.

Problem	Number of Computations	Number of global accesses	CGMA
Cluster Init	12 x 9 x $No.of Superpixels$	12 x 9 x $No.of Superpixels$	1
Remaining 3 parts	$ImageSize$ x $No.of Superpixels$ x 10	$ImageSize$ x $No.of Superpixels$ x 3	3.33

2 Complexity of serial algorithm:

Let us consider an image with $N * N$ size. The following are the complexities for the two major regions:

$$Complexity\ of\ serial\ code in Cluster Initialisation = O(RUNS * No.Of Superpixels) \tag{1}$$

$$Complexity\ of\ serial\ code in Remainingparts = O(RUNS{*}N{*}N{*}No.of Superpixels) \quad (2)$$

3 Complexity of Parallel Code

$$Complexity\ of\ parallel\ code in ClusterInitialisation = O(RUNS) \quad (3)$$

$$Complexity\ of\ parallel\ code in Remainingparts = O(RUNS{*}No.of Superpixels) \quad (4)$$

4 Theoretical Speedup:

Speedup is the ratio of time taken by serial code and the time taken by the parallel code. According to the equations 4 and **??**, the theoretical speedup can be given as -

$$Theoretical\ Speedup\ \alpha\ \frac{O(RUNS * No.Of Superpixels) + O(RUNS * N * N * No.of Superpixels)}{O(RUNS) + O(RUNS * No.of Superpixels)}$$

$\alpha\ O(N * N)(5)$
Where $N * N$ depicts the size of the image.

5 Optimisation strategy

The parallel implementation has been carried out in 2 major stages. The first stage consists of only Cluster Initialisation where each superpixel is assigned a thread and it perturbs itself to its lowest gradient position. In the second stage, each thread is assigned a pixel and it iterates over all the superpixels to either find out its overlapping clusters or to compute its degree of membership. There will code divergence in the case of number of superpixels and number of pixels in the image not being a multiple of 32.

To compute degree of membership we assign each overlapping region to a block and each thread of the block to a pixel of the region. However, it might only be effective in the case of high number of pixels per overlapping region.

The part of the code involving itself with forming regions of overlapping clusters has not been parallelised since it involves the usage of a hash-map whose implementation in the CUDA kernel results in race conditions. However, privitisation techniques can be used to achieve the result successfully.

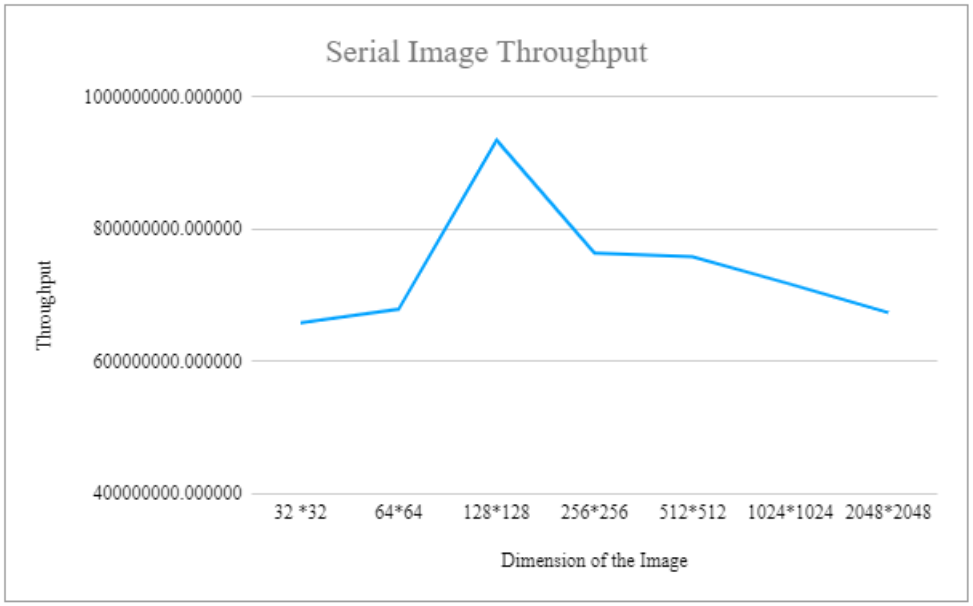
6 Problems faced during parallelization and possible solutions

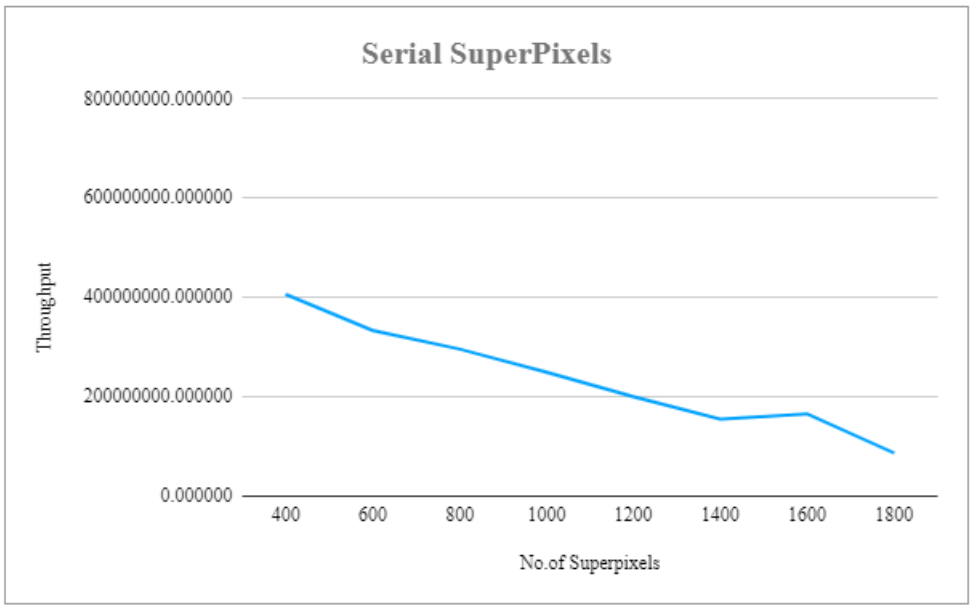
The lack of STL libraries that provide crucial data structures for handling data caused a lot of problems during the implemenations. This resulted in a lot of

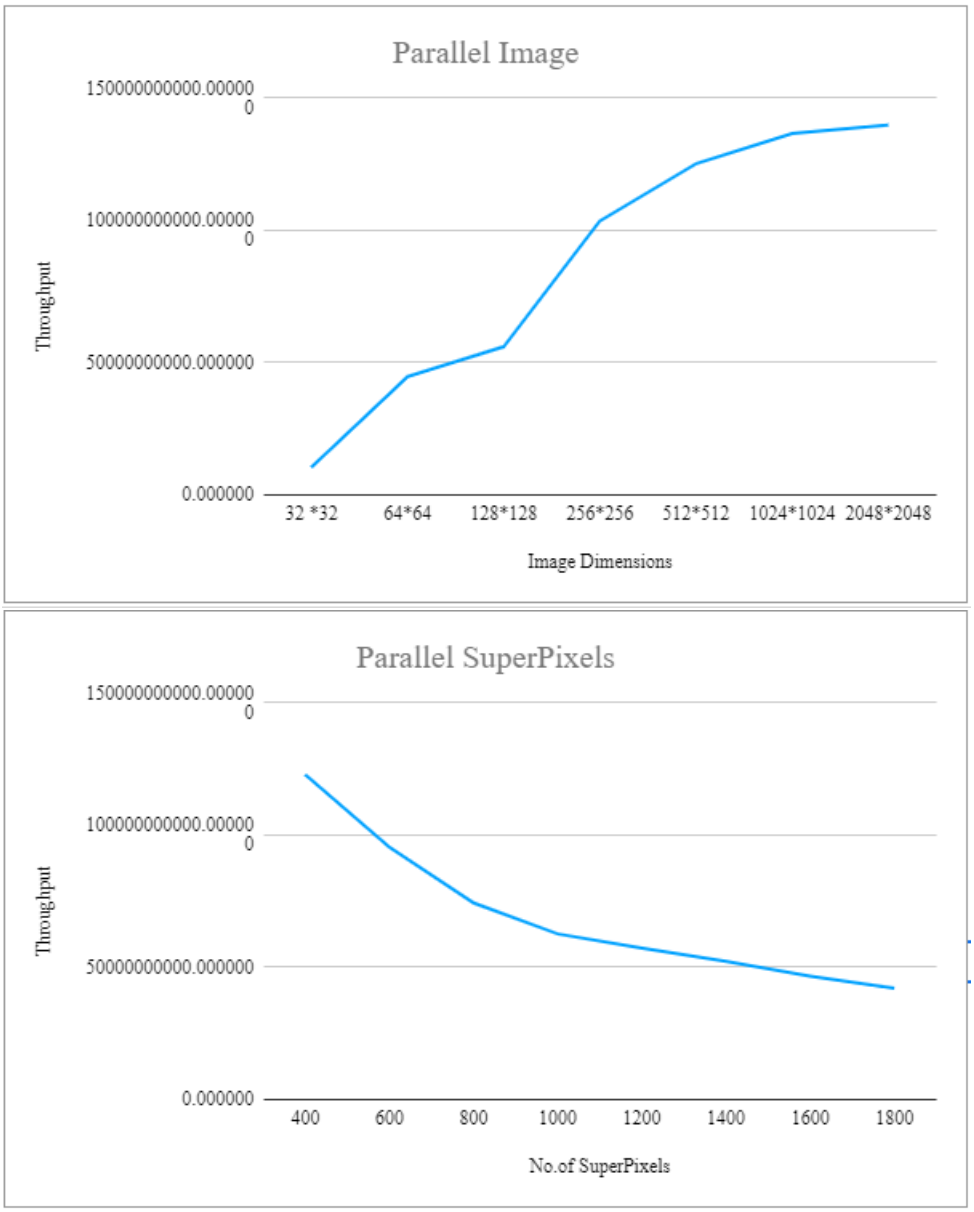
time being wasted on implementing functionalities of STL libraries using basic data structures provided by C.

The strategy of using privitisation mentioned in the previous section will require combining the private arrays and that will again result in additional complexity and this can be resolved very simply by using a hash-map.

7 Curve Based Analysis







The curves for the throughput for the serial and parallel implementation of the algorithm with varying image sizes and number of superpixels are as above.

It can be observed that the throughput curves of the CPU follow the pattern expected out of the a CPU due to the capacities of the caches. In the case of the GPU, it can be seen that the throughput increases as image size increases, however there it saturates at high sizes.

It can also be observed that there is only one step that is parallelised with each thread being assigned to one superpixel whereas there are three other steps where each thread is assigned to a pixel and this thread iterates over all the superpixels and therefore it is expected that the throughput reduces as the number of superpixels increase in the case of both CPU and GPU.

The speedup curve shows the behaviour of speedup as image size varies but increases at a lower rate with increasing number of superpixels

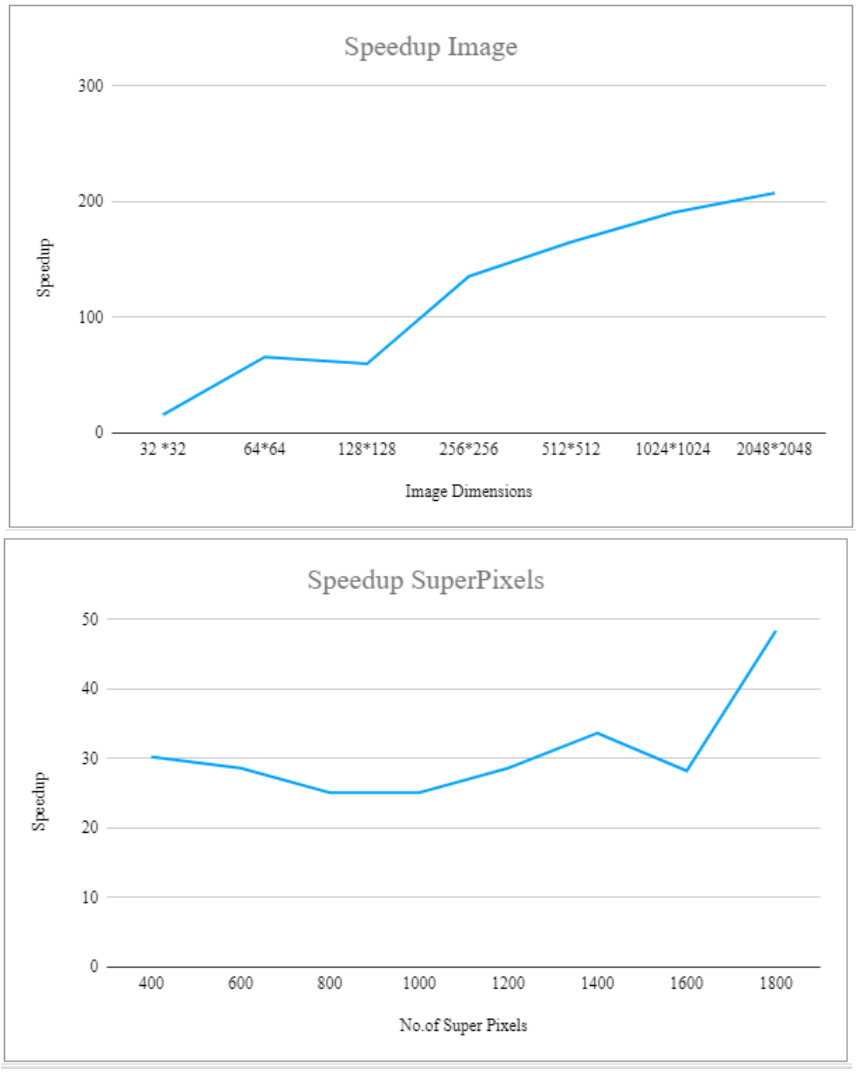


Fig. 1. The above curves indicate the speedup of parallel codes in accordance to varying image sizes for different block sizes.

It was also found that the performance remained nearly the same with multiples of 32 and the highest in such a case. However, in the case of the number of threads in the block not being a multiple of 32, the speedup varied proportionally with the block size.