



# **Improving Accuracy of Supapixel Segmentation using Colour Histogram based Seed Initialization**

By

Kulkarni Abhinav Anand

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A DISSERTATION SUBMITTED IN PARTIAL  
FULFILMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF

MASTER OF SCIENCE IN EMBEDDED SYSTEMS

May 2017

# Table of Contents

Abstract.....	3
Acknowledgement .....	4
List of figures .....	5
List of Tables.....	6
Abbreviations .....	7
Chapter 1: Introduction .....	8
1.1 Overview.....	8
1.2 Motivation.....	9
1.3 Objectives and Contribution .....	10
1.4 Organisation of report .....	10
Chapter 2 Background and Related Work .....	11
2.1 Superpixels.....	11
2.2 Existing Superpixel Segmentation Algorithms.....	12
2.3 Summary .....	13
Chapter 3: SLIC Algorithm.....	14
3.1 Preliminaries .....	14
3.1.1 Cluster formation for K-Means .....	15
3.1.2 Centroid update for K-Means .....	17
3.2 SLIC Algorithm .....	17
3.3 Limitation of SLIC.....	21
3.4 Summary.....	23
Chapter 4: Improved SLIC with colour histograms .....	24
4.1 Preliminaries .....	24
4.2 Initial Seeding .....	24
4.3 Label Enforcement.....	33
4.4 Summary.....	34
Chapter 5: Observation and Results .....	35
5.1 Evaluation Metrics .....	35
5.1.1 Boundary recall.....	35
5.1.2 Undersegmentation Error .....	35
5.1.3 Boundary Precision .....	35
5.1.4 Runtime .....	36

5.2	Experimental Setup.....	36
5.3	Results.....	38
5.3.2	Calibration of label enforcement parameter .....	40
5.3.3	Comparison of SLIC and Histogram SLIC (Figure 19) .....	42
Chapter 6:	Conclusion and future work .....	46
6.1	Conclusion .....	46
6.2	Recommendations for future work .....	47
References	.....	48
Appendix A.	Additional Results.....	52
Appendix B.	OpenCV .....	68
Appendix C.	Real-time environment.....	69

# Abstract

Superpixels are perceptually meaningful segments of an image formed by grouping several image pixels together based on a common property (e.g. colour). They provide a high-level abstraction of the raw image which can be exploited for accelerating computer vision applications. Applications that have been shown to benefit from superpixels include object recognition, 3D image reconstruction, medical analysis etc.

The well-known SLIC (Super Linear Iterative Clustering) algorithm for superpixel segmentation does not take into account content sensitivity i.e. it does not form initial clusters (seeds) based on colour or spatial information available in the raw image. This is a limitation as the accuracy of superpixel segmentation is dependent on the distribution of initial cluster centres on the raw image.

In this work, we propose the Histogram SLIC algorithm to improve the accuracy of SLIC algorithm by using colour histogram to induce colour sensitivity during the seed initialization. The quantized colour histogram captures the localized colour profiles within uniform grids in the raw image. The quality of the initial seeding can be controlled by user defined parameters such as histogram suppression window and the maximum seeds per grid box. As a post-processing step, label enforcement of erroneous pixels is implemented over the superpixel segmented image. This processing is essential since superpixels need to be continuous and should not form fragmented minute superpixels. Enforcement can be achieved by asserting a pixel with a particular cluster label based on the majority labels surrounding it.

The proposed Histogram SLIC was tested on 20 images from BSDS500 dataset for boundary recall, undersegmentation error, boundary precision and runtime. The results are promising, for example the Histogram SLIC achieves an average improvement in superpixel segmentation accuracy of 5% in boundary recall, 22% reduction in undersegmentation error and 20% increase in boundary precision over SLIC for a sample image. The results were obtained as an average of 50 to 500 superpixel segmentations. Histogram SLIC also incurs an average increase in 50% runtime for the same evaluation. While for certain images, the proposed Histogram SLIC shows worst performance, the accuracy-runtime trade-off of the proposed method can be tuned based on the available computing resources and the application requirements by adjusting the parameters of histogram window and maximum seeds per grid.

# Acknowledgement

I would like to express my sincere gratitude to Assistant Professor Lam Siew Kei for guiding and mentoring me throughout the project. Also, I would like to thank Dr Wu Meiqing for providing me with right approach and benchmarking tools required for the project. I would also like to thank my friends and classmates for helping me with technical issues regarding my practical work. My heartfelt appreciation goes to my beloved parents, Mr. And Mrs Kulkarni, for their support and love throughout my studies at the University. Finally, I would like to thank School of Computer Engineering, Nanyang Technological University, Singapore for their support.

# List of figures

Figure 1 : Overview of a computer vision framework with superpixel segmentation .....	8
Figure 2 : Superpixel segmentation of raw image .....	11
Figure 3 : CIE Lab representation .....	14
Figure 4 : Flowchart, K-Means .....	16
Figure 8 : SLIC, Random seed initialization with uniform step size. 52 super pixels. ....	22
Figure 9 : Image divided uniformly for histogram analysis into 9 segments. ....	26
Figure 10 : Localized histograms of grid boxes. ....	31
Figure 14 : Label enforcing for error pixel 3x3 kernel (8 –connected pixels): The red pixel label 1 will be forced to 2. ....	34
Figure 15 : Label enforcing for unlabelled corner pixel 3 x 3 kernel (8 –connected pixels): The red pixel forced to 1. ....	34
Figure 16 : OpenCV GUI. ....	37
Figure 17 : Effect of histogram window on images: 15(upper) -31(lower) histogram window. .....	39
Figure 18 : Label Connectivity Enforcement: 50(upper)-350(lower) super pixels. ....	41
Figure 19 : SLIC and Histogram SLIC; 50(upper)-450(lower) superpixels .....	42
Figure 20 : Boundary recall comparison for Figure 19. ....	43
Figure 21 : Boundary precision comparison for Figure 19. ....	43
Figure 22 : Undersegmentation error comparison for Figure 19. ....	44
Figure 23 : Runtime comparison for Figure 19. ....	44

# List of Tables

Table 1 : Evaluating maximum seeds per grid with histogram maxima suppression window size..... 38

Table 2 : Effect of different label enforcement kernel .....40

Table 3 : Performance evaluation of SLIC and Histogram SLIC.....45

## Abbreviations

SLIC	Super Linear Iterative Clustering
OpenCV	Open Computer Vision
STL	Standard Template Library
PCA	Principal Component Analysis
ADAS	Advanced Driver Assisting System
EAMS	Edge Mean Shift
SEEDS	Superpixels Extracted via Energy Driven Sampling
ERS	Entropy Rate Superpixel
RGB	Red Green Blue
HSV	Hue Saturation Value
CMY	Cyan Magenta Yellow
CIELAB	Commission Internationale de L'éclairage lab
OpenCL	Open Computing Language
TP	True Positive
FN	False Negatives
GUI	Graphical User Interface
API	Application programming interface
BSDS500	Berkeley Segmentation Data Set and Benchmarks 500
MMX	Intel SIMD instruction set
SSE	Streaming SIMD technology

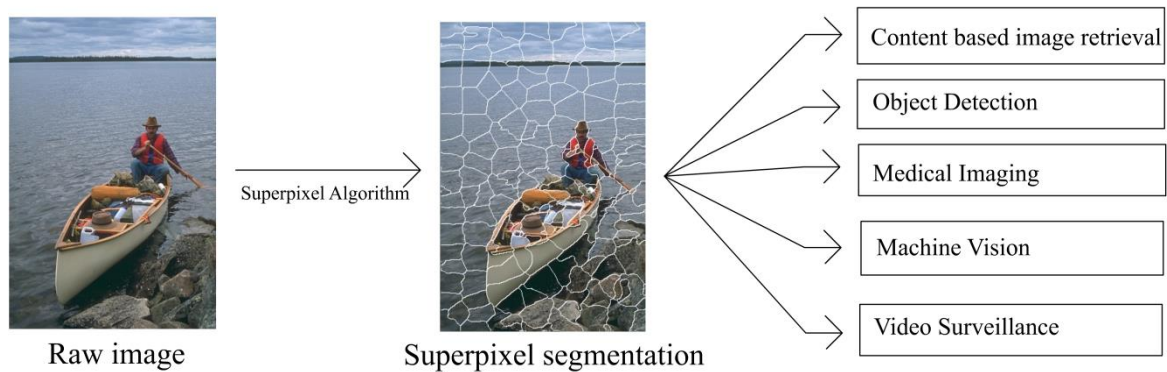


# Chapter 1: Introduction

## 1.1 Overview

Computer vision is a domain which encompasses methodical image acquisition and image processing to deliver visual sensory data to computing systems. The main objective for computing systems to incorporate computer vision in their environment is to automate and simulate the task of human visual perception. This domain has matured to the extent where its applications now span a broad range of areas like industrial automation, biotechnology, medical diagnosis to automated vehicle driving. In many of these applications, the accuracy and speed of the computer vision algorithms is of major concern. As such, a great deal of research has been undertaken to improve the speed and accuracy of computer vision algorithms that can take advantage of the available computation resources. However, the complexity of the image representations poses a bottleneck to the performance of computer vision systems.

The pixel grid is a standardized format for representing the image information on digital medium. However, this format does not provide a natural and intuitive representation of the visual scenes. Clustering pixels to form perceptually meaningful entities, such as superpixels, could provide a high-level of abstraction of the raw image. Superpixels can be rapidly generated by oversegmenting raw images to incorporate a high-level of abstraction of the raw image. A general computer vision framework for superpixel segmentation has been provided in Figure 1.



**Figure 1 : Overview of a computer vision framework with superpixel segmentation**

Figure 1 shows an overview of the superpixel segmentation. A raw image is processed using a superpixel segmentation algorithm to generate intelligible superpixel segments. The superpixel segmented image is then used to accelerate typical computer vision applications like content based image retrieval, object detection, medical imaging, machine vision and video surveillance.

## 1.2 Motivation

Superpixels are perceptually meaningful oversegmentations which consists of labelled clusters of image pixels with similar cues (e.g. colour, texture etc.). There are several advantages of working with such superpixel based segmentation. Transitioning the problem to superpixel domain enables feature statistics to be encapsulated in the image representation in an adaptive manner such that they carry more intelligible information than rigid and fine grained pixel representation [24]. Superpixels also provide a better susceptibility to noise [27]. Compared to fine-grained pixel representations, superpixels contribute towards reduction in the dimension of data, and hence can result in significantly lower complexity when used in computer vision algorithms. In addition, superpixels preserve the natural topology of the image [31], which could lead to higher robustness and result quality.

Superpixel segmentation is increasingly being used in a number of computer vision applications (see Figure 1). For example, in automated medical analysis, superpixels have been shown to improve detection of cancer tissues in parts of the human body. The work in [25][27] has demonstrated the effectiveness of superpixels in segmenting lung parenchyma for diagnosing lung cancer. In the study of biomolecular structures at micro level, superpixels can improve image segmentation of irregular cellular structures [26]. Superpixel segmentation has also been widely adopted in many ADAS (Advanced Driver Assisting System) [28] applications. For example, they have been used to accelerate automatic detection of vehicular traffic and road signs. Recent work in superpixel segmentation involves attaching semantic information to the segments [29] in order to improve the robustness of the algorithms

Although the current superpixel segmentation techniques can provide a consistently deterministic performance, there is still much more room for improvement. Errors in superpixel segmentation can to the later stages of the computer vision pipeline. Felzenszlerwalb and Huttenlocher [14] has shown that existing superpixel segmentation algorithms performs poorly in the presence of salt and pepper noise. As such, superpixel algorithms are generally not used in computer vision applications where raw images are transmitted over a long distance. In order to achieve high accuracy in superpixel segmentation, the superpixel algorithm needs to be robust to noise or undulating illumination. The well-known SLIC superpixel segmentation algorithm [3] does not utilize the finer colour information of the raw image to set its initial cluster centres. Hence, its accuracy can be improved further by utilizing content sensitivity of the raw image. Manifold SLIC [2] introduces this feature by using topological manifolds. However, it is not suited for real-time application as its algorithm includes complex trigonometric operations. Also, the number of extra seeds resulting from image content sensitivity is fixed which makes it less modular. In the proposed work, we intend to introduce a modular approach to induce image content sensitivity in the SLIC algorithm which can also be implemented for real-time systems.

### **1.3 Objectives and Contribution**

The objective of this dissertation is to improve the accuracy of superpixel segmentation. In particular, we aim to:

- Perform extensive literature survey on superpixel segmentation algorithms.
- Study the state-of-the-art SLIC algorithm for superpixel segmentation and identify its limitation.
- Propose a method called Histogram SLIC that is based on using colour histogram for initial seeding to improve the original SLIC algorithm by leveraging on image content sensitivity.
- Evaluate the impact of various parameter settings of Histogram SLIC on the superpixel results.

### **1.4 Organisation of report**

This report is organized as follows. Chapter 1 motivates the problem of superpixel segmentation. In Chapter 2, the concept of superpixel and several state-of-art algorithms are discussed. In Chapter 3, SLIC algorithm is described and its limitations are discussed. Chapter 4 provides the preliminaries for the discussion of the improved algorithm. The proposed Histogram SLIC method is introduced in Chapter 5. Experimental results that compares the performance of SLIC and Histogram SLIC, as well as the impact of different parameter setting in Histogram SLIC is provided in Chapter 6. Chapter 7 concludes the dissertation and provides a discussion of future work.

## Chapter 2 Background and Related Work

In the previous chapter, we explained how Superpixel segmentation can be used to accelerate computer vision algorithms. In this chapter, we will first discuss some background on superpixel segmentation before reviewing some of the prominent state-of-art superpixel techniques.

### 2.1 Superpixels

Superpixels are oversegmentations of an image that form cluster of pixels based on low-level cues like colour, texture etc. Superpixels need to be a subset of object region i.e. a superpixel should not be split by object boundary. This enables the complete object boundaries to be easily recovered using computer vision algorithms if required by the application. The number of superpixel segments is usually more than the object boundaries present in the image. Figure 2 illustrates an example of superpixel segmentation using the SLIC algorithm with 150 superpixel segmentations.



Figure 2 : Superpixel segmentation of raw image

In a superpixel segmentation algorithm, the definition criterion for pixels depends on the application requirements [32]. There are primarily two categories of superpixel segmentation algorithms, i.e. graph based and gradient based. Graph based methods mark each of the graphical nodes as superpixels. Gradient based methods use region growing from a set of initialized cluster points. The segments in both methods are based on image cues that can be exploited to compute similarity and discontinuity. Typical choices are grey level intensity, colour, texture cues and the spatial image location. In particular, for colour and texture cues, there exist a wide range of possible alternatives. A brief review of these methods is provided in the next section.

## 2.2 Existing Superpixel Segmentation Algorithms

The following section provides a brief overview of state-of-art superpixel segmentation techniques. Graph cuts is a graph based technique [33] which segments the image recursively using colour and texture cues. It achieves this by optimally minimizing a global cost function defined on superpixel boundaries. It produces visually pleasing superpixels but has low adherence to image boundaries and is slower compared to other similar algorithms [34]. Felzenszwalb and Huttenlocher [14] proposed another algorithm with complexity  $O(\log N)$  that is based on pixels represented as nodes in a spanning tree. It does not provide control over the number of superpixels generated.

Investigations were also carried on clustering based methods. Some of the prominent work includes Turbopixels [15], which propagates segments from seeds using a level-set method. However, this method is found to be slower on real datasets even though it has  $O(N)$  runtime complexity [15]. SLIC (Super Linear Iterative Clustering) [3] is a superpixel segmentation method based on region growing which uses K-Means as its core processing. SLIC uses a limited scan range unlike K-Means [3], which enables it to run fast with linear time complexity. The number and compactness of the superpixels can be controlled explicitly in SLIC [3]. Details of the SLIC algorithm will be provided in Chapter 3. Another clustering method called SSS (structurally sensitive superpixels) makes use of the non-homogenous image features like content-sensitivity to form non-uniform superpixels [16]. It uses geodesic distances for clustering formation thereby making it computationally expensive.

Using superpixels to form groups of pixels replaces rigid pixel distribution boundaries [3]. These groups can be formed on the basis of colour, brightness or texture variation within the image surface. However, calculation of superpixels on the basis of colour is easily achievable. Although texture based methods are more challenging, there are some approaches undertaken in this direction (e.g. textons) [1]. The general requirements for any superpixel algorithm is that it needs to adhere well to image boundaries, should have a low computational complexity, and should not compromise on the speed and reliability of overall recognition algorithm [3].

The Edge Mean Shift (EAMS) algorithm [35] is based on the model of edge detection and mean shift operation in images. Unlike K-Means, mean shift operation does not require K-centres beforehand to initiate the algorithm. The iterative mean shift process calculates the weighted kernels to reform the feature space of input image. Watershed algorithm [45] is a gradient based algorithm and its initial seeds need to be supplied to it externally. Markers are set at the boundaries and pixel regions are grown to fill these regions. This algorithm is also available as a part of OpenCV distribution.

In the Superpixels Extracted via Energy-Driven Sampling (SEEDS) algorithm, segments are initially formed as square regions all over the image [36]. The ownership of pixels at the near boundaries of these regions is exchanged between superpixels based on squared histograms computation. This algorithm is particularly suited for applications with fast changing segment boundaries (i.e. video streams). Manifold SLIC [2] transforms the pixel grid into 5 dimensional manifolds and computes the associated areas using triangular approximation.

These areas are considered as an approximation of the colour variation, thereby inducing content sensitivity when it is transformed back again into pixel domain. It has a variable scan length unlike normal SLIC and also decides the number of seeds dynamically [2]. Introduction of trigonometric and root values computations in an intermediate step within the algorithm outperform normal SLIC in accuracy, but increase the use of arithmetic computations.

In Entropy Rate Superpixel (ERS) [37], clustering is achieved by optimizing an objective function consisting of a random walk on the image graph representation. Edges are iteratively added to form a superpixel cluster to maximize the gain of their objective function until desired number of clusters (superpixels) is achieved. A graph based technique by Veksler [38] processes superpixel segmentation as an energy minimization problem. In general, these techniques can be broadly divided into graph based (Graph Cuts, Felzenszwalb and Huttenlocher, ERS) or gradient based (SLIC, Turbopixels, EAMS). SEEDS and EAMS have been integrated into real-time framework (OpenCV), while others are mostly developed analytically.

## **2.3 Summary**

In this chapter, a brief introduction to the concept of superpixel was presented and some of the state-of-art superpixel segmentation techniques were discussed. Super Linear Iterative Clustering (SLIC) is a simple superpixel segmentation method, which employs modified K-Means algorithm for superpixel segmentation. SLIC has linear time complexity and its runtime is independent of the number of superpixel segmentation of the image. It adheres well to the image boundaries and shows less undersegmentation error as compared to the other algorithms[3]. Hence, with SLIC as a baseline, we intend to further improve upon its accuracy by utilizing the colour information of the raw image for initial distribution of superpixel cluster centres.

## Chapter 3: SLIC Algorithm

In this chapter, we will first provide some preliminaries for the SLIC algorithm and then explain the SLIC algorithm in detail. We will also explore the limitation of the SLIC algorithm which does not rely on content sensitivity for placing the initial seeds over the raw image for superpixel segmentation.

### 3.1 Preliminaries

Choice of colour scheme for image representation significantly affects any computer vision algorithm. RGB representation is the most common representation used for colour image captured by digital camera. A particular colour pixel is represented by 24 bit colour value (8 bit for each of the RGB channels). Alpha component is also added to RGB to have explicit control over the intensity of colour, if required for the application. Another colour scheme, CIELAB encompasses the entire spectrum of available colours, including colours outside the capability of human vision. It means that same amount of numerical change in colour brings the change in colour perception. Hence it is well suited for applications like SLIC, which utilizes distance metrics like Euclidean distance to distinguish between different colour pixels.

Colour scheme is described by a gamut which encloses all the possible colour values of that scheme. Colour gamut of CIELAB scheme is device independent which makes the algorithms built on it more robust.

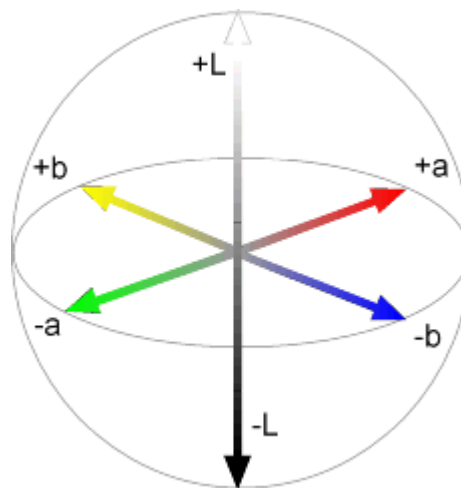


Figure 3 : CIE Lab representation

The L axis represents colour lightness parameter and ranges from 0-100. Axis  $a^*$  and  $b^*$  represent the specific colour parameters. The entire three colour axes are orthogonal to each other and the intersection is considered as neutral colour. This whole model is based on the principal that colour cannot be both green and red; or yellow and blue simultaneously. The  $a^*$  axis covers red at one extremity and green at the other, while the  $b^*$  axis covers blue at one end ( $-b$ ), and yellow ( $+b$ ) at the other. A zero value or very low numbers on either of the axis will describe a neutral or near neutral colour. Axis values  $a^*$  and  $b^*$  do not possess maximum

values, but in practice they are usually ranged from -128 to +127 (256 levels). Figure 3 shows the orthogonal representation of L, a and b axis.

Determination of total colour difference between all three coordinates is shown by the following formula:

$$\Delta E = (\Delta L^2 + \Delta a^2 + \Delta b^2)^{\frac{1}{2}} \quad (1)$$

$\Delta L^*$  ( $L^*$  sample -  $L^*$  standard) = difference in intensity (+ = lighter, - = darker).

$\Delta a^*$  ( $a^*$  sample -  $a^*$  standard) = difference in red and green (+ = redder, - = greener).

$\Delta b^*$  ( $b^*$  sample -  $b^*$  standard) = difference in yellow and blue (+ = yellower, - = bluer).

$\Delta E^*$  = total colour difference.

K-means [43] clustering makes use of an iterative refinement procedure to find the optimal clustering solution of a given data set. The algorithm initializes with K number of clusters and an unlabelled data set.. The initial K centroids can be selected randomly from the data set or can be generated specifically through a random number generation algorithm. The general flow of the K-Means algorithm is presented in the Figure 4.

The K-Means algorithm then iterates between the following two steps: 1) Cluster formation, and 2) Centroid update.

### 3.1.1 Cluster formation for K-Means

A cluster of data samples is defined by a centroid. In this step, each data sample is assigned to its nearest centroid based on the numerical metric. The typical metric used is Euclidean distance but other metric like Manhattan distance can also be used for evaluating the distance between two data samples. A data sample ownership to a certain centroid is set in accordance with the following:

$$d = \arg \min_{c_i \in C} \text{distance}(c_i, x) \quad (2)$$

$d$  = minimum distance between centroid  $c_i$  and data sample  $x$ .

$c_i = i^{th}$  centroid in collection of centroids set  $C$ .

$\text{distance}(a, b)$  = Euclidean distance between point  $a$  and  $b$ .

$x$  = single data sample among overall set of data sample.



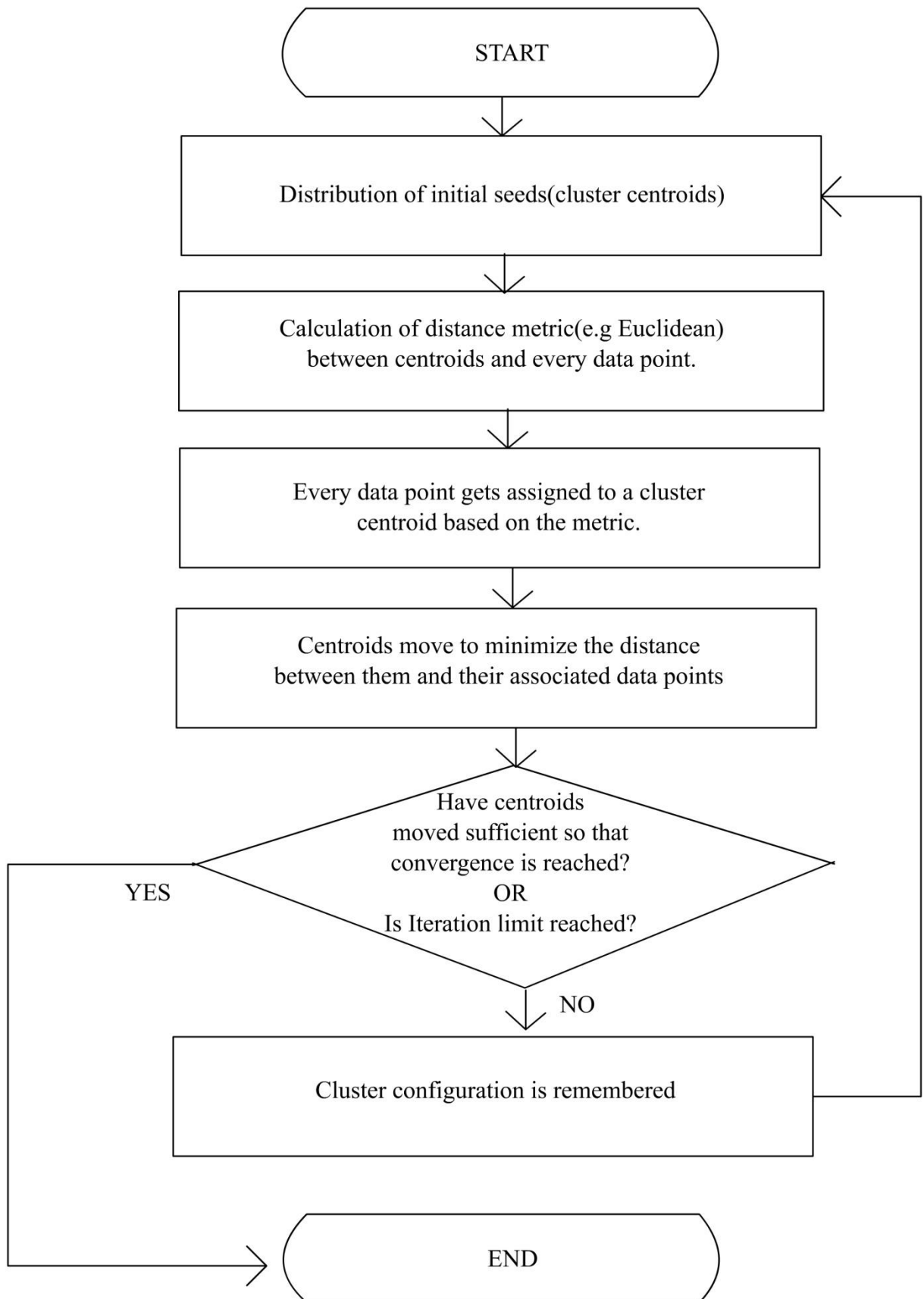


Figure 4 : Flowchart, K-Means

### 3.1.2 Centroid update for K-Means

Centroids are re-evaluated with new data sample - centroid configuration procured as a result of the previous step. The arithmetic mean of the data samples in a cluster forms the new centroid of the cluster. It is updated in the centroid collection for the next iteration. The centroid update is done in accordance with:

$$c_i = \frac{1}{|S_i|} \sum_{x_k \in S_i} x_k \quad (3)$$

$c_i = i^{th}$  centroid in collection of centroids set  $C$ .

$S_i =$  Set of data samples belonging to cluster defined by  $c_i$ .

$x_k = k^{th}$  data sample in the set defined by  $S_i$ .

The K-Means algorithm iterates between these two steps until a stopping criterion is met. This criterion is the maximum number of iterations set at the start of algorithm or can be minimum value of cost function. The algorithm eventually converges to a clustering configuration, which might be a local optimum. Re-adjusting the algorithm with different randomized initial centroids provide varied outcomes of the K-Means algorithm. Figure 4 shows K-Means algorithm flow for single dimensional data samples.

## 3.2 SLIC Algorithm

Super Linear Iterative Clustering (SLIC) [3] is an efficient abstraction of the K-Means algorithm for superpixel segmentation. A seed value is the initial value of centroid provided either by random generator algorithm or can be chosen arbitrarily. In subsequent iterations, the data samples go through a series of centroid ownership changes before converging to a final value of error. This error is calculated by a marked cost function in each iteration.

$$\Delta = \sum_{k=1}^K \sum_{i \in S_k} ||x_i - c_k|| \quad (4)$$

$\Delta =$  Cost function of K-Means

$K =$  Total number of seeds

$X_i = i^{th}$  data sample belonging to set  $S_k$

$C_k =$  Centroid of data sets within set  $S_k$

In standard K-Means algorithm, each data sample is compared with every cluster centroid in the data set. The convergence of the algorithm is highly influenced by the choice of initial seed values (i.e. cluster centres). The complete flow of SLIC is divided into initial seed selection and K-Means iteration as shown in Figure 5. For the current application, a data sample is a 5-dimensional vector; 3 colour channels and 2 spatial co-ordinates:

$$C = [L \ a \ b \ x \ y] \quad (5)$$

$C$  = Sample vector for pixel  $P$ .

$L$  =  $L^*$  component of pixel  $P$ .

$a$  =  $a^*$  component of pixel  $P$ .

$b$  =  $b^*$  component of pixel  $P$ .

$x$  = Horizontal spatial component of pixel  $P$ .

$y$  = Vertical spatial component of pixel  $P$ .

The comparison between two-pixel sample vectors is based on Euclidean distance. However, colour distance and spatial distances are evaluated separately:

$$d_c = \sqrt{(p_L - q_L)^2 + (p_a - q_a)^2 + (p_b - q_b)^2} \quad (6)$$

$d_c$  = Colour distance between pixel  $p$  and  $q$ .

$p_L$  =  $L^*$  component of pixel  $p$ .

$p_a$  =  $a^*$  component of pixel  $p$ .

$p_b$  =  $b^*$  component of pixel  $p$ .

$q_L$  =  $L^*$  component of pixel  $q$ .

$q_a$  =  $a^*$  component of pixel  $q$ .

$q_b$  =  $b^*$  component of pixel  $q$ .

$$d_s = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (7)$$

$d_s$  = Spatial distance between pixel  $p$  and  $q$

$p_x$  = Horizontal spatial component of pixel  $p$

$p_y$  = Vertical spatial component of pixel  $p$

$q_x$  = Horizontal spatial component of pixel  $q$

$q_y$  = Vertical spatial component of pixel  $q$

The colour distance  $d_c$  and the spatial distance  $d_s$  do not change equally in proportion to each other in any given image. If the spatial distance  $d_s$  is more pronounced than colour distance  $d_c$ , it results in compact super pixels that do not adhere well to superpixel boundaries as shown in Figure 6. On the contrary, if the colour distance  $d_c$  is more pronounced than the spatial distance  $d_s$ , the superpixel clusters will be fragmented and disjoint as shown in Figure 7. Hence, to obtain an accurate measure of the total distance between colour pixels,  $d_c$  and  $d_s$  should be weighted unequally. The total Euclidean distance between pixel  $p$  and  $q$  is given by:

$$D = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} \quad (8)$$

$d_c$  = color distance between pixels  $p$  and  $q$ .

$d_s$  = spatial distance between pixels  $p$  and  $q$ .

$N_c$  = color distance weight parameter.

$N_s$  = spatial distance weight parameter.

$D$  = total distance between pixels  $p$  and  $q$ .

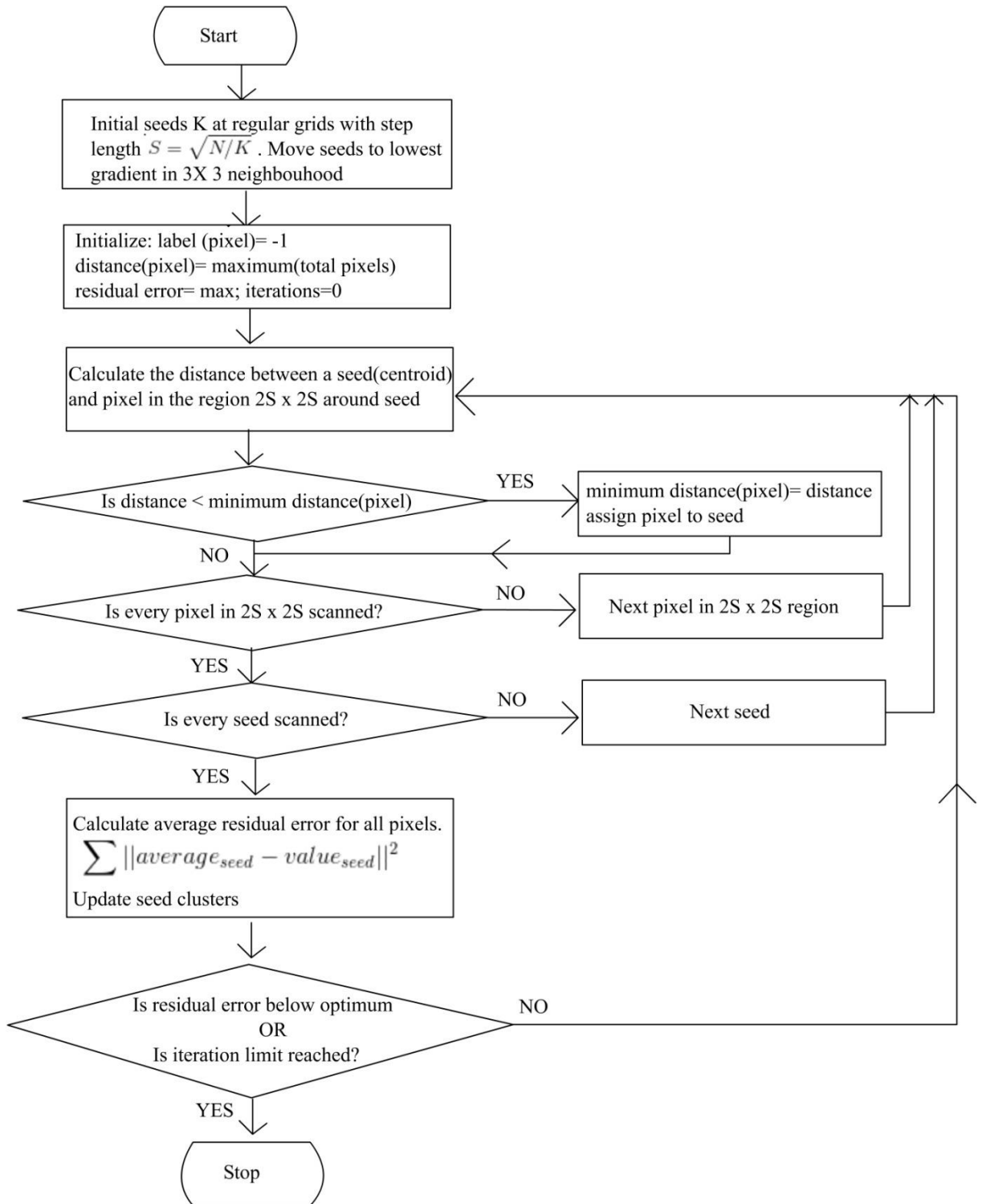
Two parameters  $N_C$  and  $N_S$  are introduced to weigh the proximity of  $d_c$  and  $d_s$  unequally. Unlike standard K-Means algorithm where each data sample is compared with every cluster centre, SLIC has a limited search range of comparison. Search range for each cluster centre is  $2S \times 2S$  in SLIC, where  $S$  is the step size in the initial grid:

$$S = \sqrt{\frac{N}{K}} \quad (9)$$

$N$  = Number of image pixels (width  $\times$  height).

$K$  = Number of super pixels (initial cluster centres).

$S$  = Step size.



**Figure 5 : Flowchart of SLIC Algorithm**



Figure 6 : Superpixel segmentation, 50-450 superpixels with  $N_c=2$ ,  $N_s=0.5$



Figure 7 : Superpixel segmentation, 50-450 superpixels with  $N_c=0.5$ ,  $N_s=2$

### 3.3 Limitation of SLIC

As K-Means forms the crux of SLIC algorithm, the drawback of seed initialization of K-Means is inherited by SLIC. The seeds are chosen at regular intervals over a grid, as shown in Figure 8, which results in uniform pixel oversegmentation. On the contrary, it is possible to utilize the prior knowledge (colour, texture etc.) of the image to set initial seeds (cluster centres) which has not been exploited in SLIC. The shortcomings of SLIC with regard to initial seeding can be directly related to its underlying K-Means algorithm. This optimization problem of K-Means is NP-complete; meaning that there's no algorithm that can find the optimal solution of the resulting clustering configuration in polynomial time. K-Means provides an approximate solution because its objective function approaches local minima on algorithm iteration. One of the local minima can be numerically close to the global minimum, but both do not possess the same value. Clusters can get trapped in local minima's and won't be able to shift to optimal positions [40]. This increases the repetition of redundant clusters.

K-Means in general does not provide unique clustering solution and hence, neither does SLIC.



Figure 5 : SLIC, Random seed initialization with uniform step size. 52 super pixels.

To overcome the seed initialization drawback of K-Means, several techniques were proposed. The technique should be of linear complexity, similar to K-Means. Another important requirement is that the level of randomness in such techniques should be as low as possible. Inducing any randomness in the initial seeding technique itself is equivalent to choosing random seeds and would prove the technique futile. Also, if the initial seeds are close to final cluster centres, it is observed that convergence of the K-Means algorithm is faster with good clustering performance [40].

Several algorithms are proposed to set the initial locations of the seeds and even the optimal number of seeds using prior information available from the input data set. One such algorithm is K-Means++ [17]. It uses weighted probability distribution to find out the initial cluster positions and number of clusters required, and further performs normal K-Means algorithm. A method using genetic algorithm [18] sets initial cluster centres of the K-means algorithm based on the fitness criteria. Accordingly, initial clusters are placed non-uniformly over the image with iterative crossover and mutation operation. In general, for K-Means clustering, increasing initial cluster centres with same penalty provides less error in clustering. In extreme case if each data sample is considered as a cluster, it would provide zero error in clustering. However, an optimal choice has to be made between assigning maximum data to a cluster and assigning each pixel to its rightful cluster. Statistical method of elbowing [19] provides optimal number of seeds (cluster centres) based on the percentage variance in the given data. The X-means [20] estimates clusters by the process of binary splitting until criterion like Bayesian information criterion (BIC) or Akaike information criterion (AIC)[21] is met. The silhouettes method explores the possibility of setting initial clustering by querying how closely the pixel matches with the current cluster by datum metric [22].

In essence, clustering performance of K-Means algorithm can be improved by setting initial cluster centres intelligibly using the priori information of the data set. In the proposed method of Histogram SLIC discussed in the next chapter, we will use the colour information of the raw image as a priori information to set the initial cluster centres of superpixels. This priori information is extracted using localised colour histograms obtained from the raw image.

### 3.4 Summary

A brief overview of the SLIC algorithm has been presented in this chapter. It is a popular algorithm for superpixel segmentation owing to its low complexity. Colour distance and spatial distance for any image do not vary in the same proportion based on the intended application. Hence, it is difficult to capture the total distance between two colour pixels in a rigid mathematical abstraction where  $N_s$  and  $N_c$  have universal constant values. The search range of SLIC is not definitive and can be adapted to suit specific computer vision application and available computation resources. However, it is possible to improve upon the performance of SLIC by setting the initial superpixel clusters more intelligently.



## Chapter 4: Improved SLIC with colour histograms

In the previous chapter, we discussed the limitations of SLIC due to initial seeding of the superpixel segments. The improved SLIC method proposes the use of localized colour histograms to induce content sensitivity in the SLIC algorithm for improving its accuracy.

### 4.1 Preliminaries

A histogram measures the frequency of a value in a set of given values. In a similar manner, a colour histogram represents the count of occurrence of a colour in a given image. For example, if each of the RGB channel is represented in an 8-bit format, the total colours available in the scheme would be about  $256 \times 256 \times 256 = 16777216$ . A histogram of such large number of bins would require intense computation power to process in real-time environment and would certainly account for significant runtime overhead. Quantised colour histogram addresses this specific problem by reducing the bin size by partitioning the colour channel into groups. Accordingly, each of the colour channels are divided into groups and are combined to output a single integer value for a colour. Each of this single integer value is known as a bin. It can be regarded as a lossy compression of colour data and every colour does not possess a unique value. However, this representation suffices to provide a high-level of abstraction to facilitate colour sensitivity for the proposed algorithm.

The colour histogram of an image shows relative invariance with translation and rotation about the viewing axis [41]. Generally, it is possible to measure the histogram signatures of two images even when the position and rotation of the object is not known within a particular environment [43]. Also, varying light levels have significantly lower effect on the histograms even in device dependent models like RGB.

### 4.2 Initial Seeding

SLIC algorithm distributes initial seeds uniformly by placing them at the centre of grid boxes which are determined using a given step size  $S$ . This uniform distribution is shown in Figure 8. To ensure that these seeds do not overlap with a high gradient object boundary, they are localized to a spatial location of low gradient in  $3 \times 3$  kernels [3]. Using these seeds, K-Means clustering is performed. The number of grid boxes is equal to the number of superpixels desired. The proposed algorithm uses colour histogram for distribution of initial seeds. We will henceforth refer to this proposed algorithm as Histogram SLIC.

When image is acquired for segmentation, a colour histogram is calculated for each of the grid box. This histogram is based on quantized colour values obtained from each of the RGB channel of the colour pixel in the following manner:

$$r_q = \text{floor}(R/64) \quad (10)$$

*R = Red channel of pixel representation.*

*r<sub>q</sub> = Quantized value of red channel.*

*floor(.) = function calculating the greatest or equal integer.*

$$g_q = \text{floor}(G/64) \quad (11)$$

*G = Green channel of pixel representation.*

*g<sub>q</sub> = Quantized value of green channel.*

*floor(.) = function calculating the greatest or equal integer.*

$$b_q = \text{floor}(B/64) \quad (12)$$

*B = Blue channel of pixel representation.*

*b<sub>q</sub> = Quantized value of blue channel.*

*floor(.) = function calculating the greatest or equal integer.*

The individual quantised values are mapped to form a single colour value:

$$C = 16 * r_q + 4 * g_q + b_q \quad (13)$$

*C = Quantised value of a particular colour*

*r<sub>q</sub> = Quantised value of Red channel*

*g<sub>q</sub> = Quantised value of Green channel*

*b<sub>q</sub> = Quantised value of Blue channel*

The obtained values for a particular pixel are used as an input to form the colour histogram representing 64 bins. Spatial locations of a particular value of  $C$  bin are stored in an averaged format:

$$X_{loc} = \text{floor}\left(\frac{\sum_{x \in C} x}{NP_C}\right) \quad (14)$$

$X_{loc}$  = Horizontal co-ordinate of  $C$  valued bin of colour histogram

$NP_C$  = Total number of pixels with bin value  $C$

$x$  = Horizontal co-ordinate of a particular pixel with value  $C$

$$Y_{loc} = \text{floor}\left(\frac{\sum_{y \in C} y}{NP_C}\right) \quad (15)$$

$Y_{loc}$  = Vertical co-ordinate of  $C$  valued bin of colour histogram

$NP_C$  = Total number of pixels with bin value  $C$

$y$  = Vertical co-ordinate of a particular pixel with value  $C$

Localized histograms of the sample image (Figure 9) are presented in Figure 10. The sample image is divided into 9 grid squares. The 64-bin colour histogram of each grid box is shown in Figure 10. For example, the colour histogram of grid square B1 of Figure 9 is shown in row B1 of Figure 10. Each colour histogram is characterized by local maxima peaks which are the possible dominant colours of the corresponding grid box.

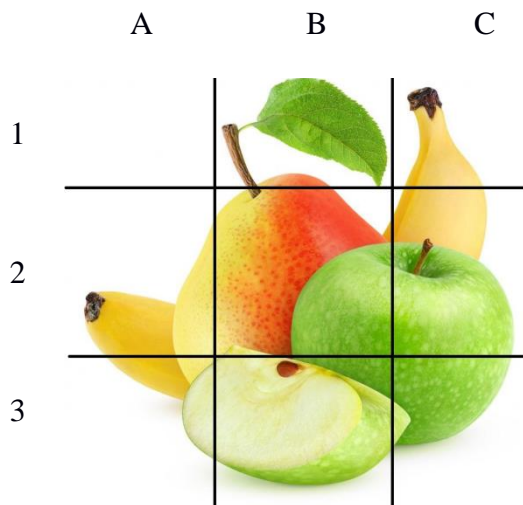
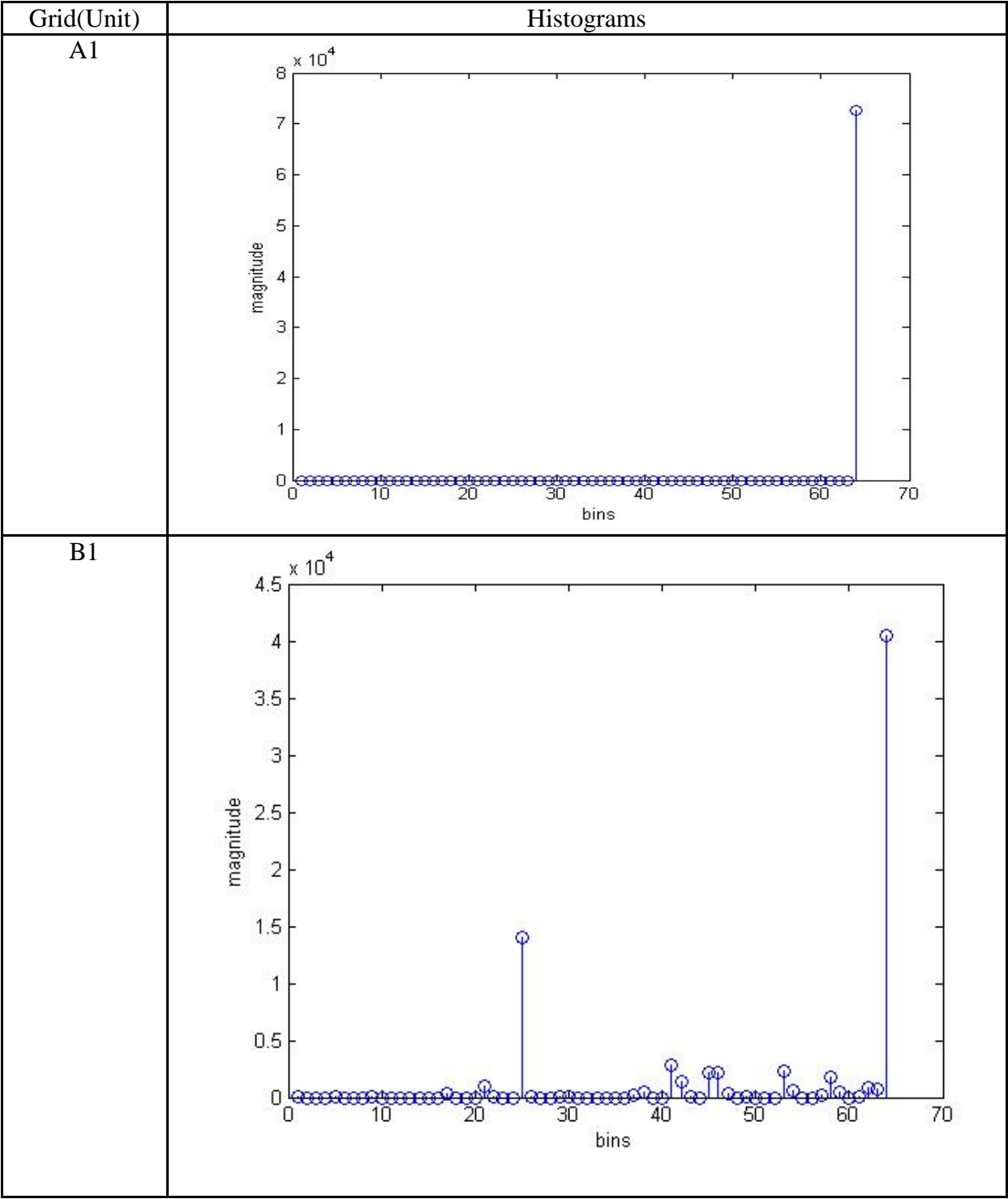
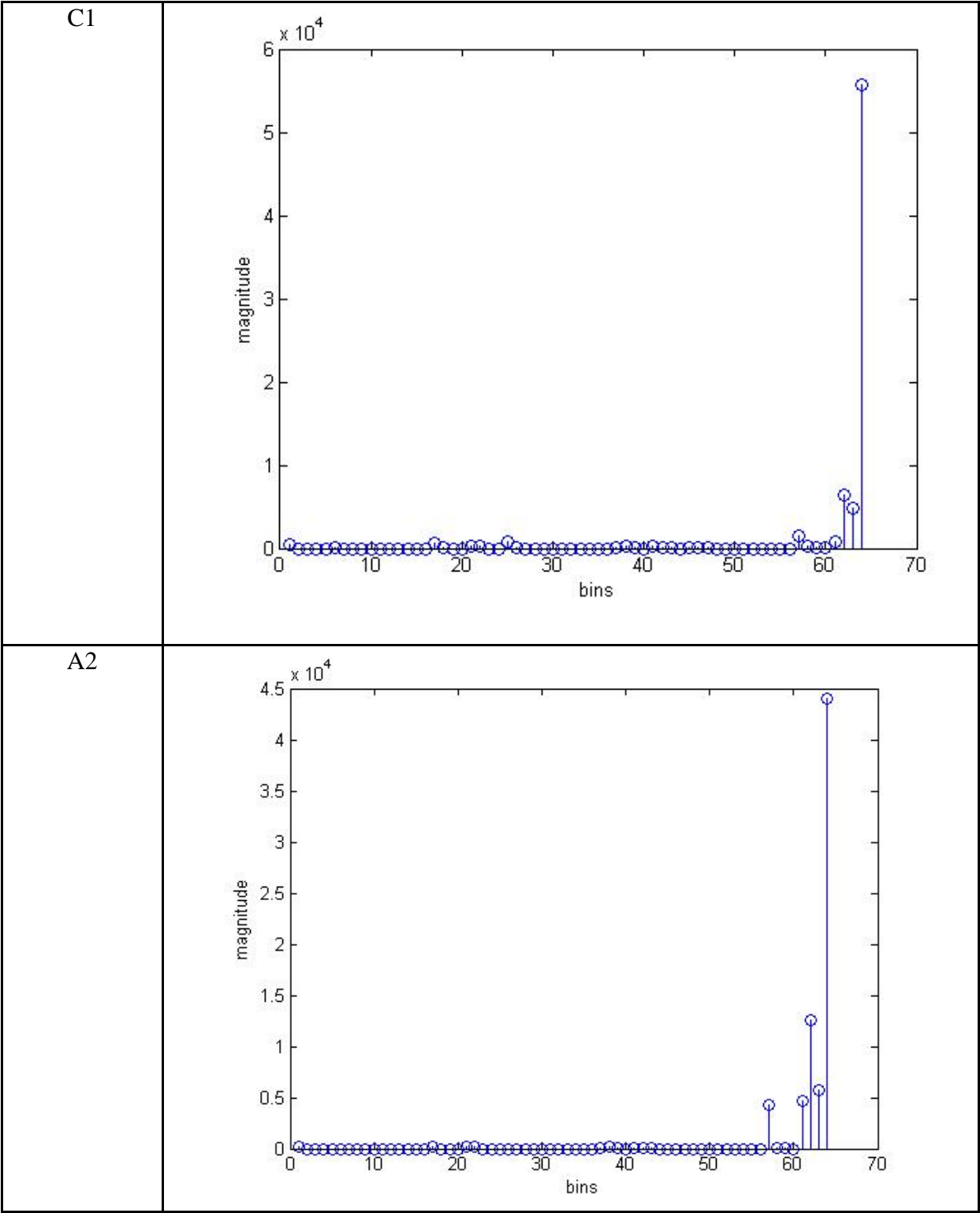
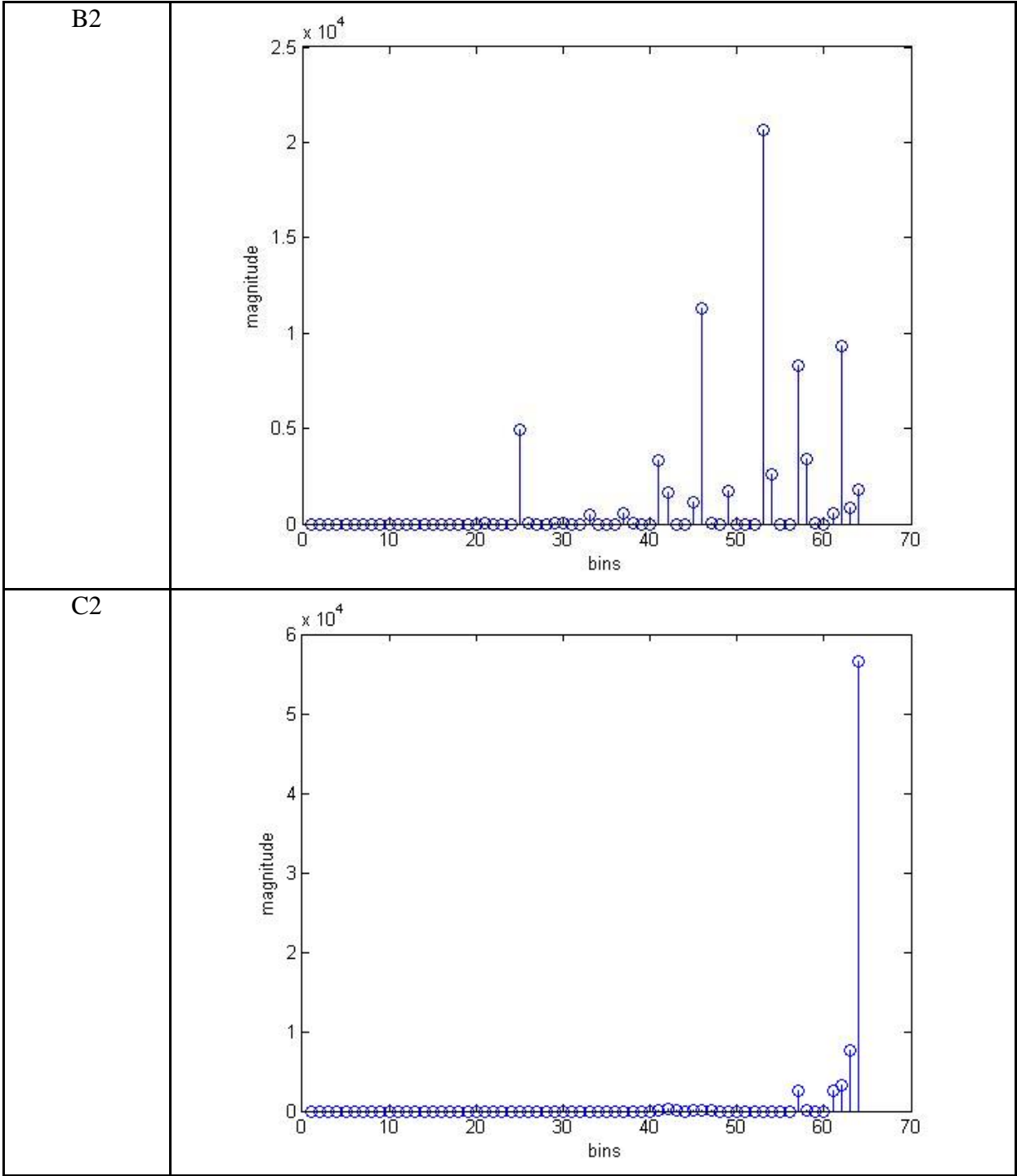
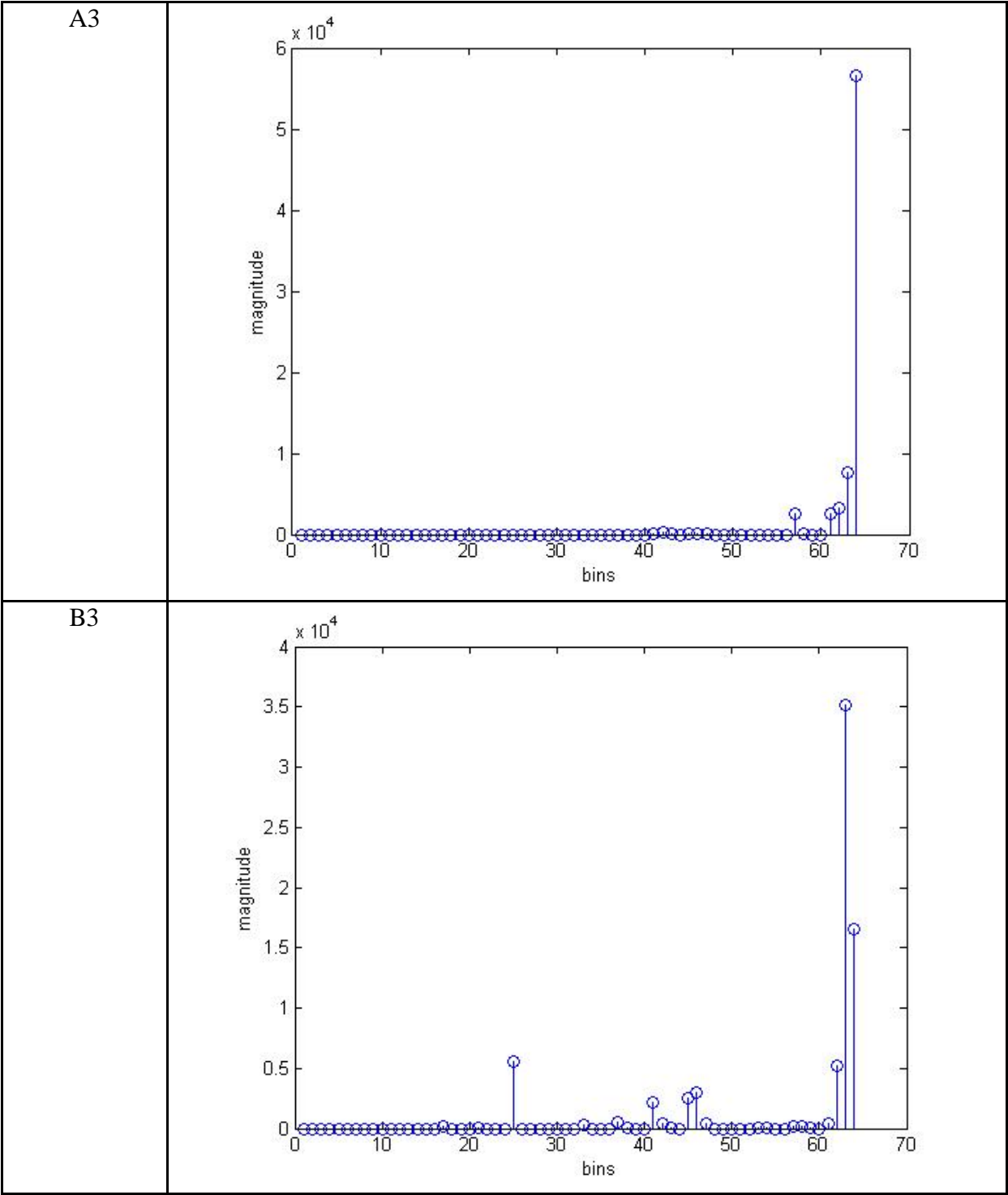


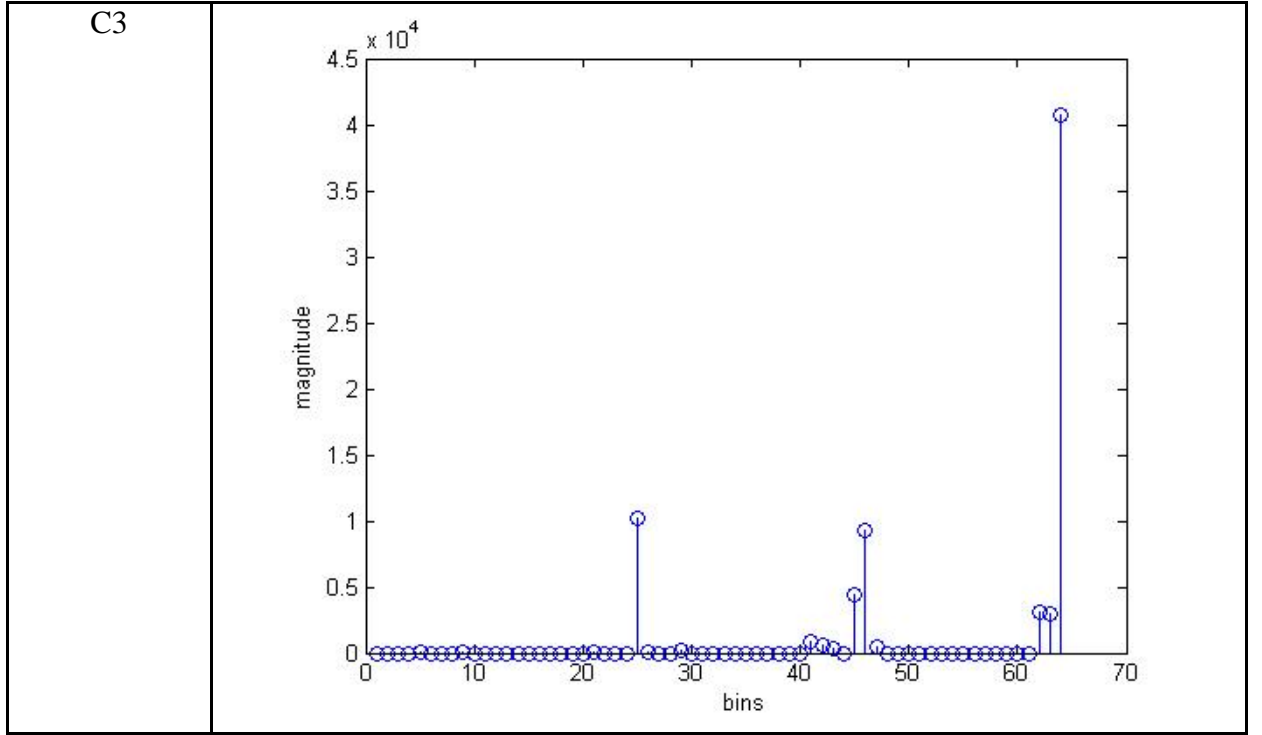
Figure 6 : Image divided uniformly for histogram analysis into 9 segments.











**Figure 7 : Localized histograms of grid boxes.**

In the SLIC algorithm [3], one seed per grid is considered for clustering. However, parameter MAX\_SEEDS\_PER\_GRID dictates the number of maximum seeds required for a particular grid box in the proposed algorithm. It serves as an upper limit for the number of seeds in a particular grid box based on the maxima obtained from the corresponding colour histogram. In such a manner, it is able to exploit the content sensitivity of the image. This model makes use of the colour information within the grid box to decide on the number, colour and spatial location of seeds in the grid box. The complete flow Histogram SLIC algorithm is provided in Figure 11.



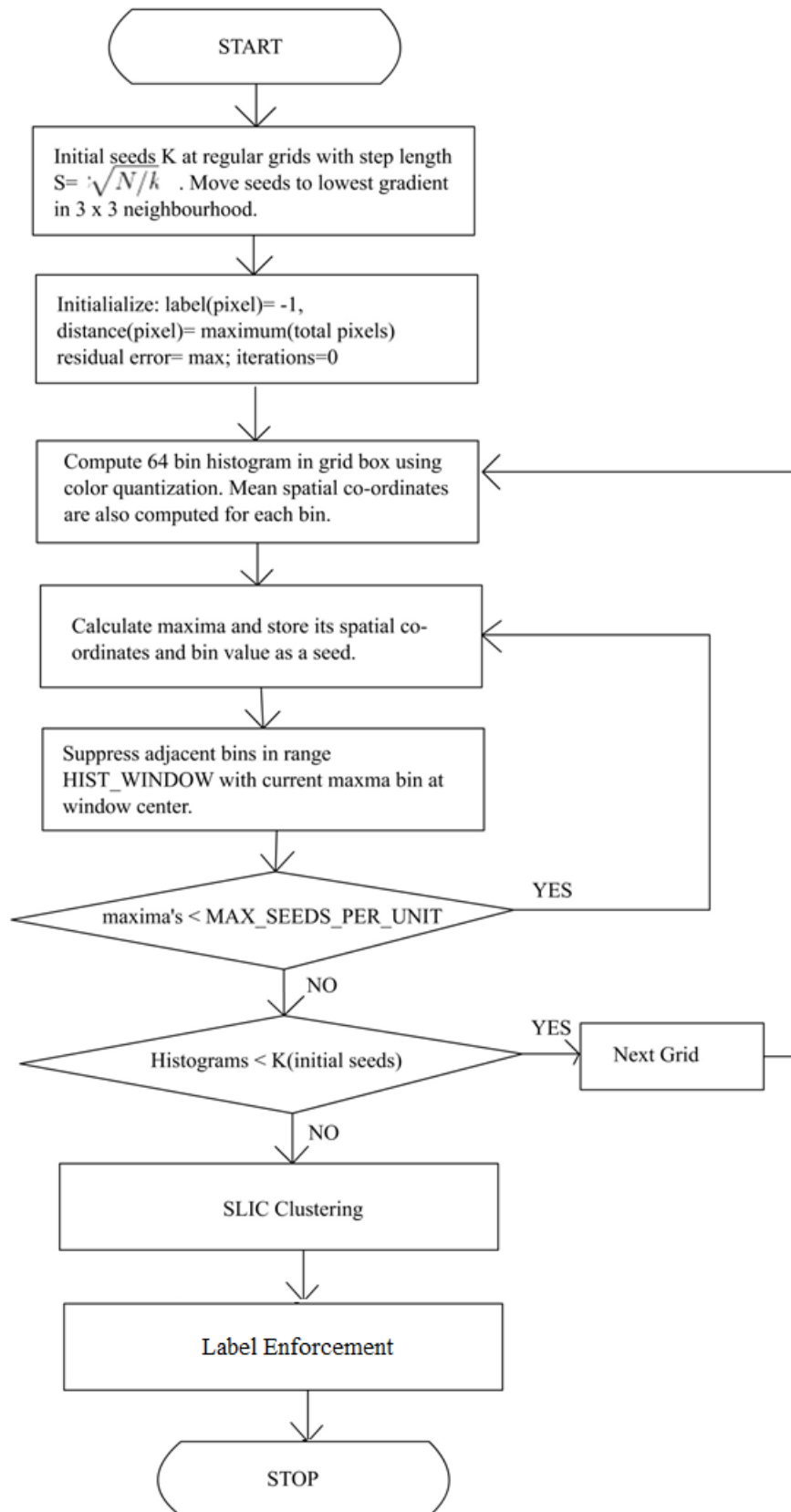


Figure 11 : Flowchart, Histogram SLIC



**Figure 12 : Random seed distribution, non-suppression of adjacent maxima. >100 superpixels.**

However, there can be cluttering of seeds around a single position of the image as shown in Figure 12. This is because adjacent colour bins in histogram form ascending maxima and quota of MAX\_SEEDS\_PER\_GRID gets exhausted to accommodate these extraneous seeds. To avoid this phenomenon, adjacent significant maxima along the first maximum are suppressed to zero within a window governed by the parameter HIST\_WINDOW. In essence, only a single maximum represents for adjacent group of maxima. Subsequently, all the new maxima are spread out more intelligently through whole histogram. Modified seed distribution after application of histogram suppression window of 11 is shown in Figure 13. It can also be observed from Figure 13 that seeds are less cluttered. The overall effect of these two parameters on image benchmarks is shown in the Table 1.



**Figure 13 : Random seed distribution by applying suppression window of 11. ~100 superpixels.**

Consequently after initial seeding, the SLIC algorithm is performed. It was observed that in the original SLIC algorithm, 7 iterations are sufficient for the results to significantly converge till optimal clustering [3]. The error from the objective cost function of algorithm is accumulated at each iteration step. In case the optimum allowable error occurs before iteration limit, the iterations are stopped. Final superpixel segmentation result is obtained in form of a matrix of labelled image pixels, at the end of the last iteration.

### 4.3 Label Enforcement

After performing Histogram SLIC, some of the corner pixels might remain unlabelled. Superpixels need to be connected to each other explicitly. As such, disjoint fragmented pixels

which forms minute superpixel clusters, need to be corrected to improve the visual quality of segmentation. To enforce this requirement, the unlabelled and disjoint pixels are force labelled using a majority pixel label from neighbouring pixels. Centre pixel as shown in Figure 14 could potentially form a fragmented superpixel. However, label enforcement process forces the centre pixel to label 2 since majority of the pixels in its neighbourhood belong to label 2. Unlabelled centre pixel in Figure 15 is force labelled as 1 as the majority of the pixels within the area restricted by kernel, belong to the label 1. The bigger kernel (kernel size of 5 x 5 or 7 x 7) gives visually pleasing super pixels but the overall image segmentation quality for boundary adherence decreases. This phenomenon has been evaluated for benchmark performance in Table 2.

1	2	2
1	1	2
2	2	2

Figure 8 : Label enforcing for error pixel 3x3 kernel (8 –connected pixels): The red pixel label 1 will be forced to 2.

1	2	
1		

Figure 9 : Label enforcing for unlabelled corner pixel 3 x 3 kernel (8 –connected pixels): The red pixel forced to 1.

## 4.4 Summary

Colour histogram is a robust method of capturing distribution of pixel colour over the entire image. Quantized colour histograms are used for setting the initial seeds for the proposed Histogram SLIC algorithm. Cluttering of seeds is eliminated by intelligently selecting colour histogram maxima with their spatial locations on the image. Initial seeds are set near the actual cluster centres of the superpixel segments, which increases the superpixel segmentation accuracy. Label enforcement enhances the visual quality of segmentation after clustering of superpixels and also provides an explicit control over the visual quality.

## Chapter 5: Observation and Results

The algorithm was tested on 20 images from BSDS500 dataset, which provides hand annotated ground truth for benchmarking. Benchmarking environment and the OpenCV experimental setup is discussed in the chapter.

### 5.1 Evaluation Metrics

#### 5.1.1 Boundary recall

This metric [12] measures the fraction of ground truth edges that are within a fixed range of at least one superpixel boundary. If the ground truth boundary image is  $G$  and the algorithms boundary image is  $B$ , boundary recall is calculated as follows:

- True Positives (TP) are the number of pixels in  $G$  for which there exists a boundary pixel in  $B$  within the given range.
- False Negatives (FN) are the number of boundary pixels in  $G$  for which there does not exist a boundary pixel in  $B$  within a given range.
- Boundary Recall =

$$R = \frac{TP}{TP+FP} \quad (16)$$

$TP = \text{True Positives}$

$FP = \text{False Negatives}$

#### 5.1.2 Undersegmentation Error

This evaluates the extent that the segment areas of superpixels cover the ground truth segment borders. A ground truth segment forms division of superpixel area as in-part and out-part. So for each segment, the metric is summed over all the out-parts of all superpixels that overlap the segment of the ground truth [12].

Undersegmentation Error =

$$UE_G(S) = \frac{\sum_i \sum_{k: S_k \cap G_i \neq \emptyset} |S_k - G_i|}{\sum_i G_i} \quad (17)$$

$G_i = i^{th}$  ground truth segment.

$S_k = k^{th}$  superpixel segment.

$S_k$  segments captures all the non-overlapping parts of the  $G_i$  segment.

#### 5.1.3 Boundary Precision

Boundary recall [12] is a one of the state-of-art evaluation algorithm for super pixel segmentation. However, it is biased towards long boundaries i.e. an anfractuous boundary that labels each image pixel as a boundary would achieve perfect boundary recall. Boundary

precision [11] is evaluated for TP (True Positives) and FN (False negatives) separately. Unlike boundary recall, TP and FN are not assumed to be a duality of each other and hence are evaluated separately. Definition of TP and FN changes for boundary precision; however the evaluation formula is the same as the boundary recall.

#### **5.1.4 Runtime**

The algorithm was implemented on Linux (Ubuntu 14.04) platform. Since precise calculation of runtime is not possible, approximate runtime was obtained by setting time probes on the algorithm breakpoints with the use of system clock. The estimated value of runtime might change with every run as the computational software resources (i.e. memory) accessed by the program are dynamic in nature on general purpose microprocessors. The experiments were carried out on Intel(R) Core(TM) i3-2370 M CPU @ 2.40 GHz with 4 GB RAM support on 64-bit architectural platform.

## **5.2 Experimental Setup**

All the experimentation was carried out on Linux-Ubuntu platform. OpenCV (C++) API's were used with support from STL framework. HighGUI package from OpenCV supported the parameter GUI application (Figure 16) to test and compare the Histogram SLIC algorithm with SLIC.

Evaluation parameters (Figure 16):

- superpixel: Number of super pixels desired (0-500).
- mode: Mode0: SLIC , Mode1: Histogram SLIC
- hwindow: the size of the suppression window used for histogram (0-64).
- munit: maximum number of superpixels per grid box
- label: Kernel size of label enforcement, 0-None, 1- 3x 3 kernel, 2- 5x 5 kernel, 3- 7x 7 kernel

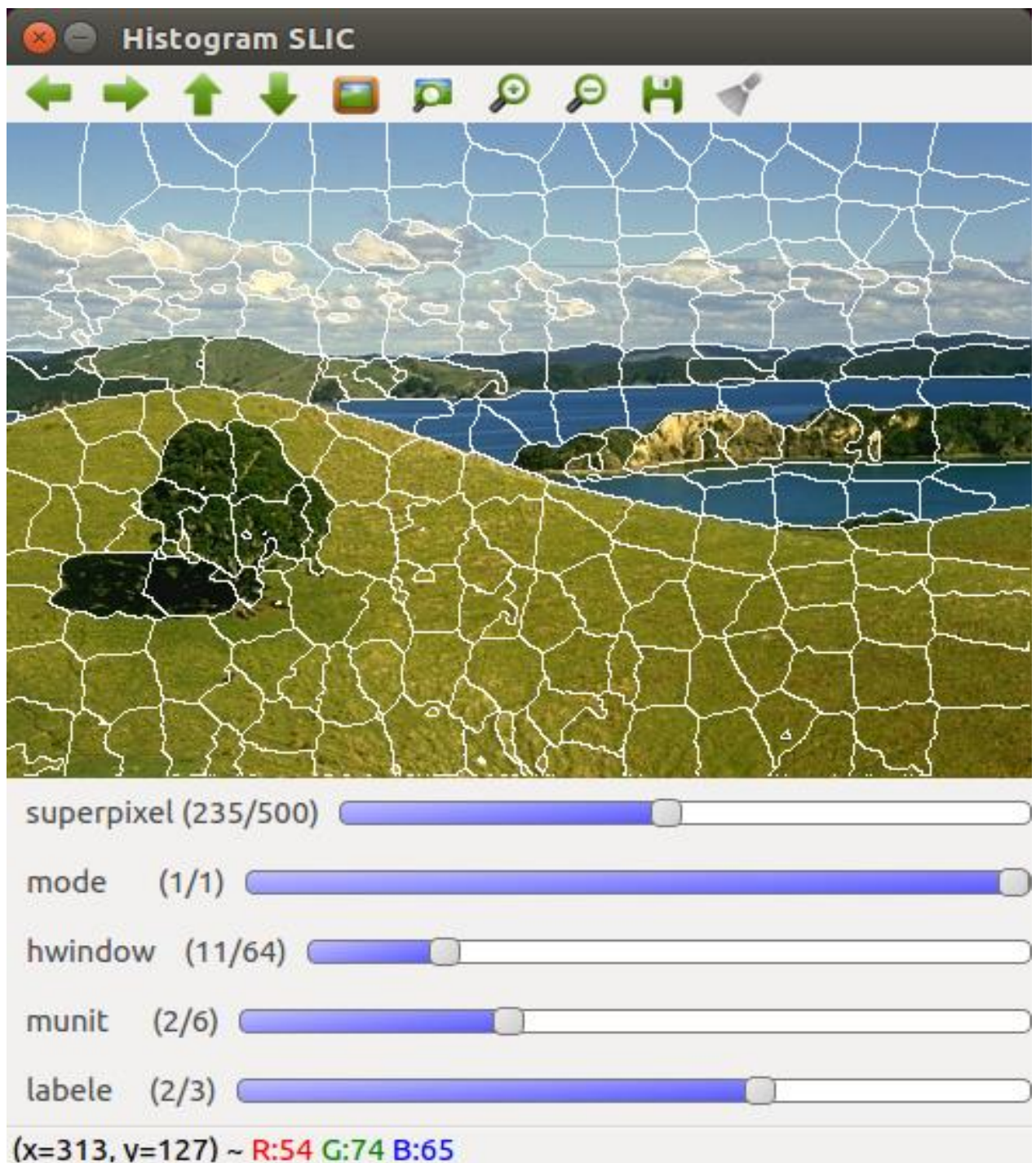


Figure 10 : OpenCV GUI.

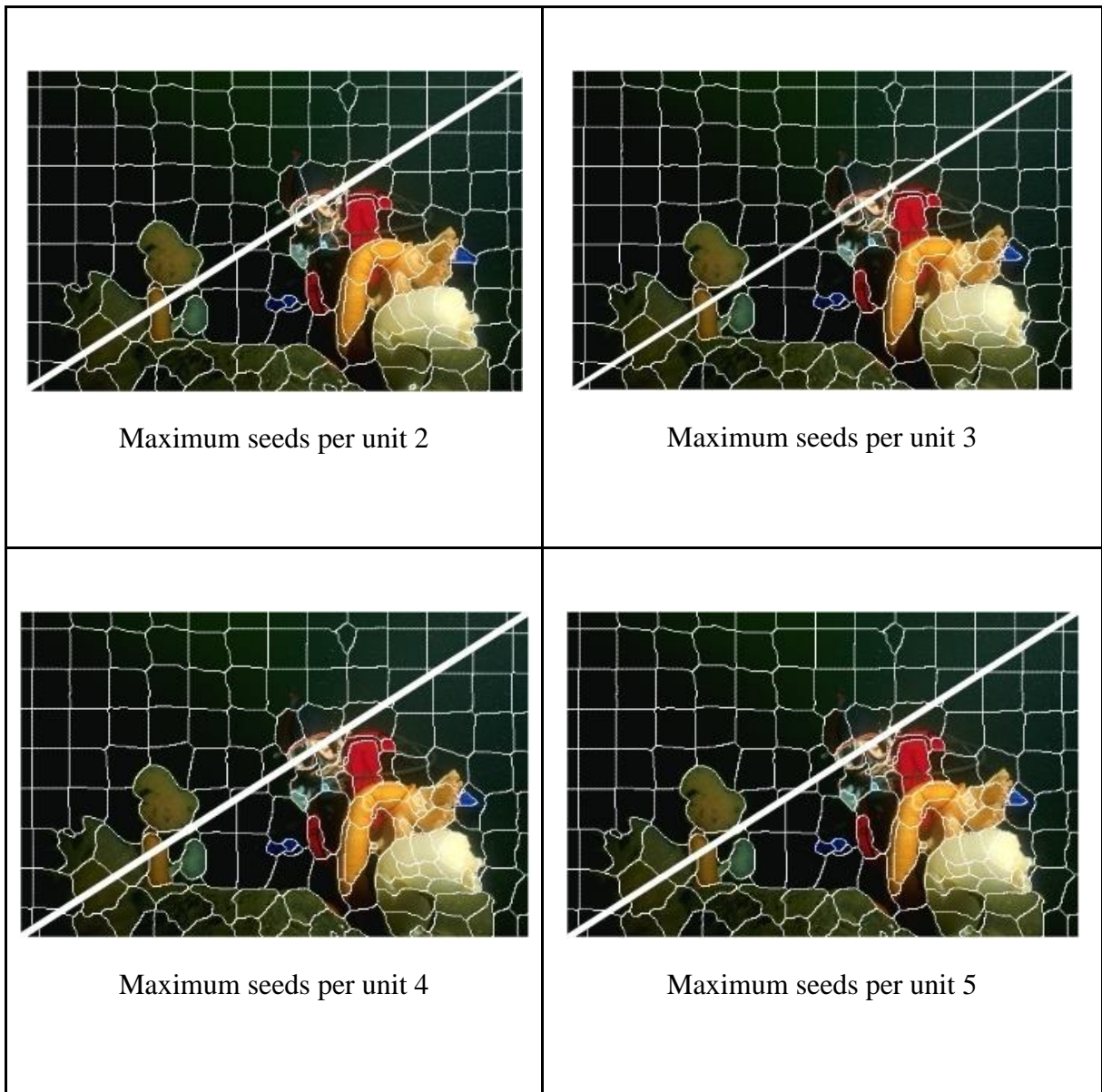


## 5.3 Results

### 5.3.1 Calibration of MAX\_SEEDS\_PER\_GRID and HIST\_WINDOW parameter

Image: 45096.jpg								
Histogram SLIC (250 superpixels)								
	Max seeds per grid box: 2				Max seeds per grid box:3			
Histogram Window	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
3	0.783916	0.0239701	0.104313	0.858577	0.782517	0.0229208	0.0981364	0.973985
7	0.762937	0.0242356	0.10213	0.840055	0.781818	0.0239571	0.0984545	0.948997
11	0.747552	0.028018	0.106342	0.757516	0.776573	0.0249351	0.105691	0.836199
15	0.762937	0.0275387	0.108014	0.76151	0.767832	0.0269687	0.104547	0.867375
19	0.763287	0.0272213	0.108343	0.747038	0.770979	0.0268522	0.105195	0.852807
23	0.741259	0.0293262	0.105305	0.76789	0.76014	0.0271436	0.105268	0.823415
27	0.741259	0.0293262	0.105305	0.754796	0.76014	0.0271436	0.105268	0.798859
31	0.756643	0.026891	0.107421	0.743821	0.767832	0.0261656	0.106246	0.845818
	Max seeds per grid box: 4				Max seeds per grid box:5			
3	0.838462	0.0218781	0.101597	1.07276	0.838462	0.0215607	0.100129	1.15585
7	0.835315	0.0230309	0.101803	1.04756	0.836364	0.0232511	0.100975	1.10714
11	0.782517	0.0249092	0.105229	0.891031	0.786014	0.0247019	0.105199	0.920784
15	0.767483	0.0268068	0.103915	0.853838	0.766434	0.0267291	0.103906	0.872045
19	0.76993	0.0267615	0.104529	0.857827	0.766434	0.0267874	0.104301	0.897564
23	0.767832	0.0269169	0.10599	0.854792	0.767832	0.0269169	0.10599	0.805848
27	0.767832	0.0269169	0.10599	0.862129	0.767832	0.0269169	0.10599	0.815851
31	0.772028	0.0263276	0.10644	0.866294	0.772028	0.0263276	0.10644	0.8696

**Table 1 : Evaluating maximum seeds per grid with histogram maxima suppression window size.**



**Figure 11 : Effect of histogram window on images: 15(upper) -31(lower) histogram window.**

It can be inferred from Table 1, more number of seeds per grid result in improved value of boundary recall, however at the cost of increase in runtime. On the contrary, increasing the suppression window loses colour content resulting in degradation of boundary recall value, but it also decreases the required runtime. Figure 17 shows the improvement in visual quality and segmentation detail in superpixel segmentation with increase in number of seeds per grid box. However, the suitable values of these parameters can be selected based on the available computing resources and intended application. Suppression window and max seeds per unit are set to a value of 11 and 2 respectively for experimentation of the proposed algorithm on image data set.



### 5.3.2 Calibration of label enforcement parameter

Image: 45096.jpg			Histogram window: 11			Max seeds per grid box:2		
Histogram SLIC								
	No Label Enforcement				Label enforcement 1:(3 x 3 kernel)			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.683217	0.0310425	0.176242	0.791174	0.619231	0.0307446	0.190185	0.781295
100	0.656643	0.0385814	0.108149	0.817479	0.615734	0.0379596	0.125867	0.825475
150	0.738462	0.0295723	0.103906	0.743862	0.700699	0.0292679	0.120065	0.758886
200	0.882168	0.0236138	0.114598	0.740322	0.828671	0.0238276	0.129303	0.736289
250	0.821678	0.0277718	0.0959419	0.750941	0.747552	0.028018	0.106342	0.762163
300	0.857692	0.0236721	0.0925346	0.714709	0.804545	0.023711	0.105165	0.706675
350	0.842657	0.0247472	0.085814	0.735843	0.773776	0.0241967	0.0952156	0.715351
400	0.826923	0.0252265	0.0793464	0.710088	0.748601	0.0254791	0.088336	0.701702
450	0.901049	0.0194817	0.0833441	0.707665	0.808392	0.019812	0.0898911	0.713766
500	0.862587	0.0212499	0.0791822	0.651073	0.79021	0.0206864	0.0864047	0.660063
	Label enforcement 2 : (5 x 5 kernel)				Label enforcement 3 : (7 x 7 kernel)			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.582168	0.0307317	0.193492	0.773837	0.567483	0.0308612	0.198387	0.76387
100	0.582518	0.0383806	0.131078	0.82415	0.525874	0.0381021	0.124524	0.820535
150	0.636713	0.0285749	0.119976	0.748991	0.583566	0.0302071	0.117901	0.768792
200	0.71958	0.0255568	0.125541	0.706849	0.645105	0.0257447	0.119263	0.732254
250	0.659441	0.0286073	0.101387	0.745876	0.593007	0.0291837	0.0964568	0.76298
300	0.743706	0.0228949	0.105937	0.737127	0.682517	0.0229208	0.102742	0.650719
350	0.682168	0.0247149	0.0916221	0.722979	0.600699	0.0257965	0.0849318	0.731685
400	0.670979	0.0267161	0.0869545	0.71045	0.625524	0.0275128	0.0861463	0.684463
450	0.721329	0.0218587	0.08694	0.705171	0.642657	0.0243587	0.0808161	0.704187
500	0.65	0.0226553	0.0768436	0.667881	0.613986	0.0238794	0.0751905	0.661462

**Table 2 : Effect of different label enforcement kernel**



No label enforcement



Enforcement: 3 x3 kernel



Enforcement: 5 x 5 kernel



Enforcement: 7 x 7 kernel

Figure 12 : Label Connectivity Enforcement: 50(upper)-350(lower) super pixels.

Enforcing label to the pixel with a given kernel size is evaluated (Table 2). The central pixel is forced with label of majority pixels in the area, restricted by the kernel size. The enforcement procedure is explained in detail in the section 5.2. Figure 18 shows visual quality improvement in Histogram SLIC with increase in the kernel size.

### 5.3.3 Comparison of SLIC and Histogram SLIC (Figure 19)

Settings: Histogram window: 11

Maximum seeds per unit: 2



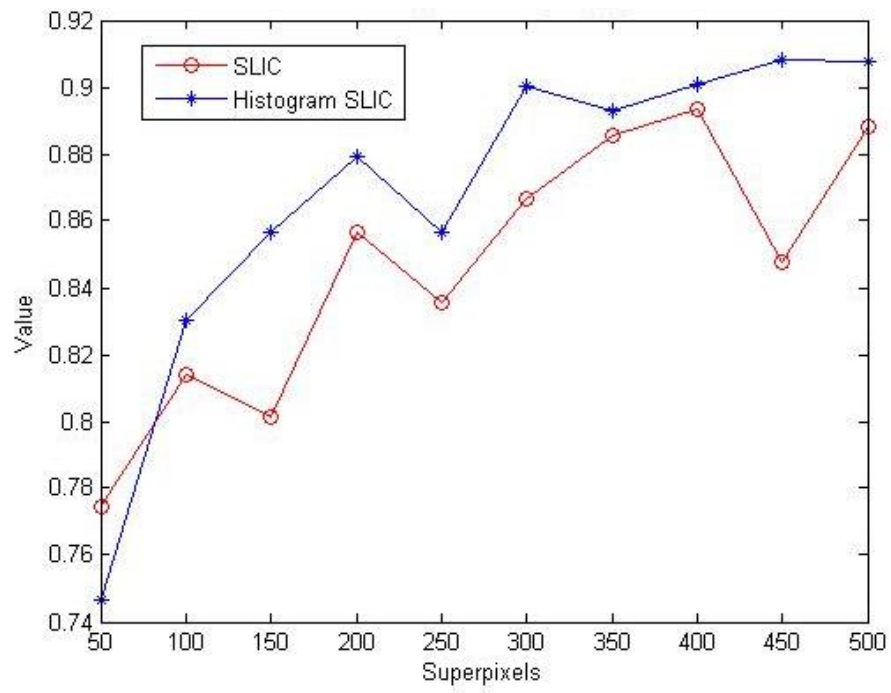
SLIC (Im3.jpg)



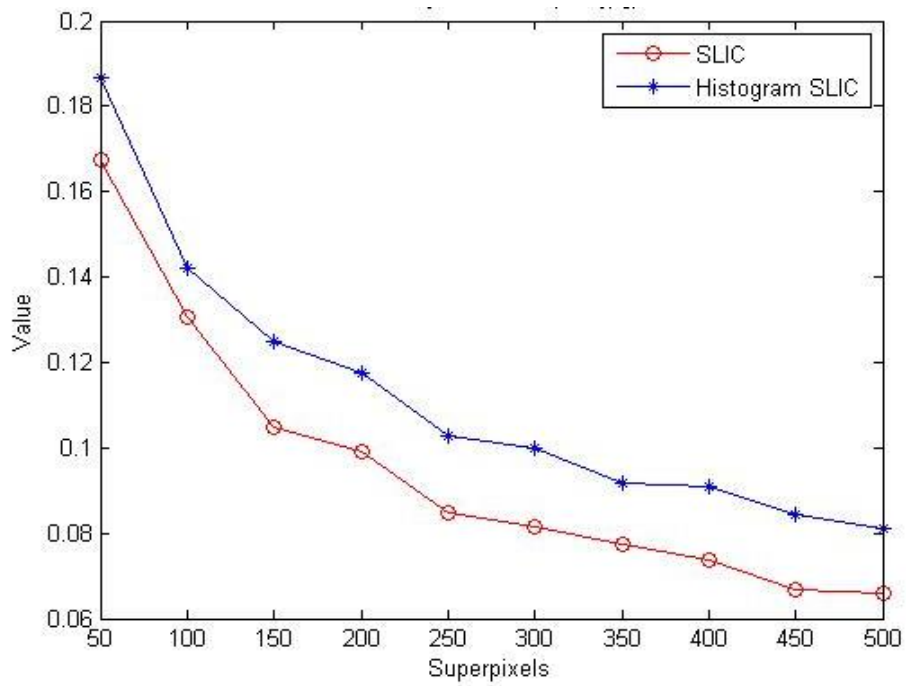
Histogram SLIC (im3.jpg)

Figure 13 : SLIC and Histogram SLIC; 50(upper)-450(lower) superpixels





**Figure 14 : Boundary recall comparison for Figure 19.**



**Figure 15 : Boundary precision comparison for Figure 19.**

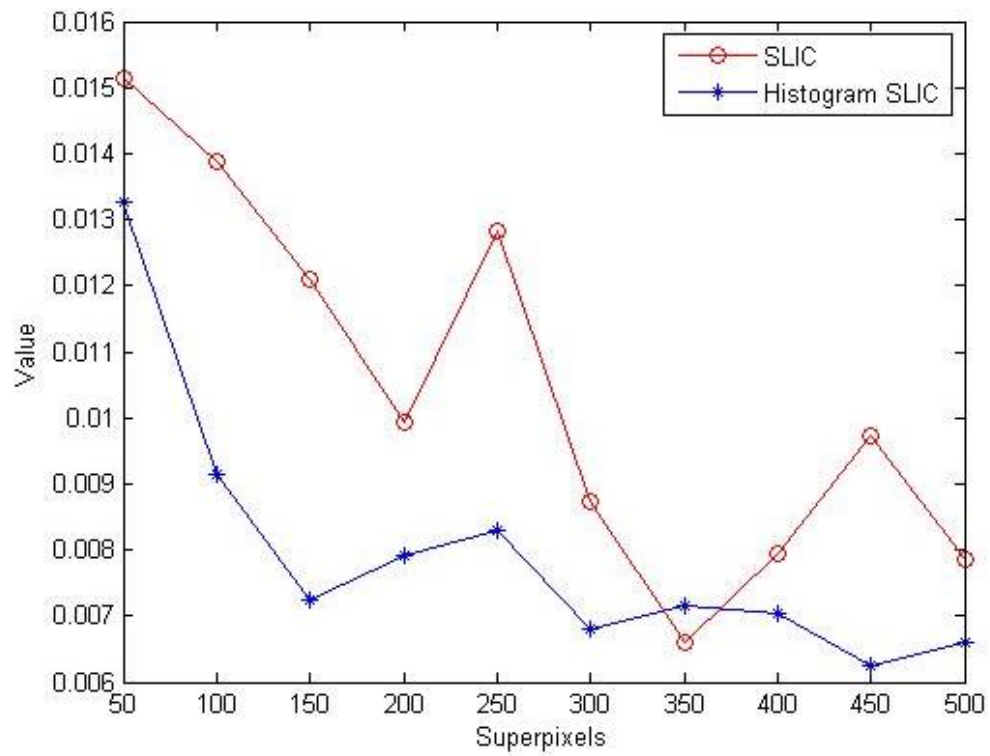


Figure 16 : Undersegmentation error comparison for Figure 19.

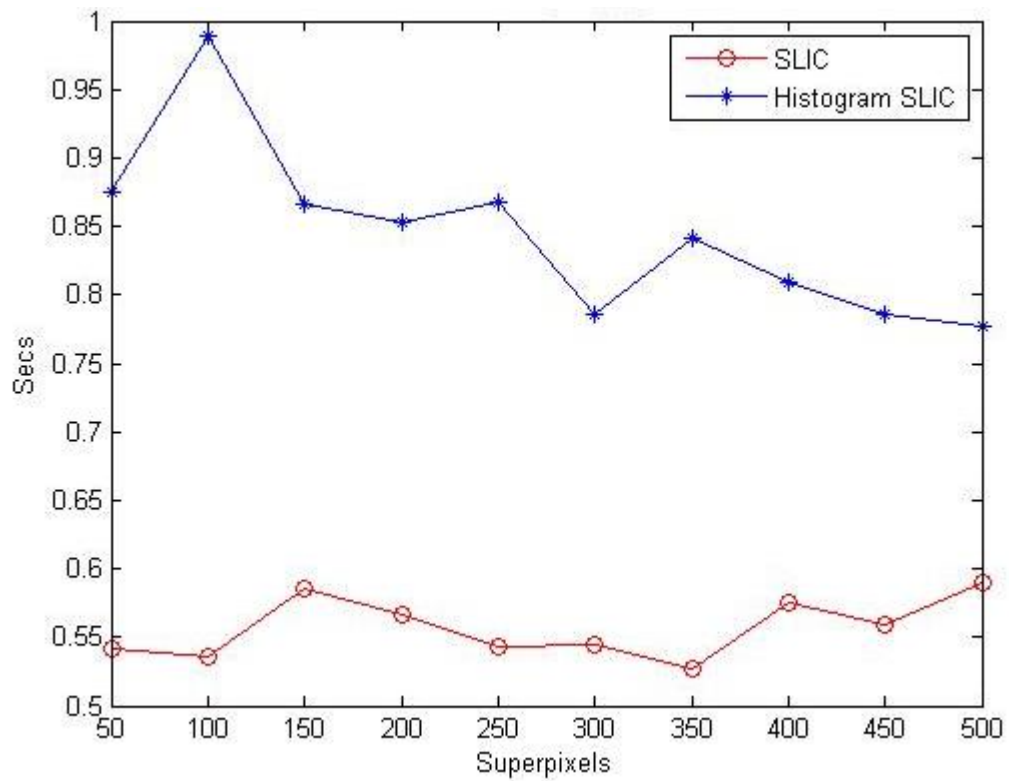


Figure 17 : Runtime comparison for Figure 19.

Image: im3.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Undersegmentation Error	Boundary Precision	Runtime*
50	0.77442	0.0151359	0.167408	0.541067	0.746386	0.0132577	0.186678	0.87536
100	0.813841	0.013873	0.130716	0.536328	0.830048	0.00913854	0.142033	0.989564
150	0.801577	0.0121048	0.104919	0.584919	0.856767	0.00724089	0.12508	0.865905
200	0.856767	0.00992222	0.0990179	0.565987	0.879106	0.00792093	0.117575	0.85253
250	0.835742	0.0128238	0.0848113	0.543709	0.856329	0.00829658	0.102792	0.867139
300	0.866404	0.00873051	0.0814595	0.544641	0.900131	0.00681343	0.0997089	0.785209
350	0.885677	0.00661265	0.0773084	0.526686	0.892685	0.00716317	0.0916532	0.841977
400	0.893561	0.00794036	0.0736223	0.574992	0.901007	0.00704659	0.0907927	0.808588
450	0.847569	0.00971496	0.0667356	0.558553	0.908016	0.00624996	0.0842683	0.786461
500	0.888305	0.00784969	0.0657865	0.590573	0.907578	0.00661265	0.0810515	0.776574

**Table 3 : Performance evaluation of SLIC and Histogram SLIC**

The proposed algorithm was evaluated on 20 images from BSDS500 dataset and a baseline of 50 to 500 superpixels was used. Evaluation of Figure 19 is provided in Table 3. Results on additional images are provided in the Appendix A along with visual representation of superpixel segmentation. The histogram SLIC provided more precise result for higher number of superpixel segmentation as it makes use of finer colour detail of the raw image. However, SLIC has been observed to be more accurate in boundary recall for low superpixel segmentation. In both cases of the above mentioned cases, undersegmentation error of Histogram SLIC is lower. Histogram SLIC shows improvement in boundary precision for all the cases of experimentation with different number of superpixel segmentations. However, achieving improvement in all the three benchmarks simultaneously is a challenging task for varying number of superpixel segmentations. For all the images considered in the experiments, Histogram SLIC performed better than SLIC in 95% of the image samples in terms of boundary value recall, 70% of the image samples in terms of boundary precision and 45% of the image samples in terms of undersegmentation error. Average runtime increase of 60% was observed when the algorithm was implemented and tested on the Intel(R) Core(TM) i3-2370 M CPU @ 2.40 GHz 64-bit architectural platform with 4 GB RAM support .

Figure 20 and Figure 21 show comparison between SLIC and Histogram SLIC over boundary recall and boundary precision respectively for sample image (Figure 19). Figure 22 provides numerical analysis of undersegmentation error for both methods. Figure 23 evaluates runtime required for running SLIC and Histogram SLIC in real-time environment (section 4.4).

## Chapter 6: Conclusion and future work

### 6.1 Conclusion

In contemporary research on computer vision, superpixel segmentation has become a crucial component. Typical computer vision tasks like pattern recognition, video surveillance etc. are growing in functional complexity. It has now become possible to collect and store vast amount of image data for processing, due to revolution in solid state electronics. However, utilizing this image data for computer vision applications is increasingly becoming a challenging task due to limitation in current computing capabilities. To accelerate typical computer vision tasks, superpixel segmentation techniques were introduced. These techniques group similar image pixel based on a particular property like colour, texture etc. at a very negligible computation cost. In the process, redundant information is eliminated from the image to reduce data dimensionality of a particular image. Standard algorithms for computer vision applications are dependent on images segmented into superpixels to accelerate the overall computation. The performance of a computer vision algorithm is directly influenced by the accuracy and precision of the underlying superpixel segmentation algorithm. Hence, search for superpixel algorithm which provides highest segmentation accuracy at lowest computational overhead, is an area of active research today. SLIC [3] is one of the state-of-art superpixel segmentation algorithms which is based on modified form of K-Means clustering algorithm. It has been shown to be robust and computationally efficient on standardized benchmarks [3] However, as it does not utilize content sensitivity of the image during its initial seeding, there is scope for further development to improve upon the accuracy of SLIC algorithm.

A qualitative study of the state-of-art algorithms has been conducted in this work. In the proposed work, improvement over the existing SLIC algorithm has been proposed. Accordingly, initial seeds are set for algorithm by exploring the content sensitivity of the raw image. The proposed algorithm uses quantised colour histogram on grid boxes to obtain finer colour detail for initial seed setting. Control over the algorithm flow is provided by two new parameters. The quality of the superpixel can be configured by these parameters as suited for a particular computing requirement. Improvement in boundary recall, undersegmentation error and boundary precision has been observed for varied number of superpixel segmentations as demonstrated in Table 3. Label enforcement has been carried out at the final stage of the algorithm to correct the unlabelled or disjoint minute superpixel. Setting big kernel size for label enforcement provides visually pleasing superpixels at the cost of segmentation accuracy and increased runtime. Smaller kernel sizes degrade visual performance, but improve the segmentation accuracy with minimal time overhead. Histogram window suppression with HIST\_WINDOW parameter eliminates cluttering of initial seeds over a single position on the image. MAX\_SEEDS\_PER\_GRID parameter sets the upper limit of the number of initial seeds within each grid box. By making use of the image content sensitivity, the proposed method of Histogram SLIC is able improve upon the segmentation accuracy of SLIC at the cost of acceptable increase in time overhead. The algorithm has been

simulated on Linux based system with OpenCV/C++ environment, with 20 images from BSDS500 data set.

## **6.2 Recommendations for future work**

The following lists the future work of this dissertation.

- The proposed work uses OpenCV and STL API's as a part of Histogram SLIC algorithm code. Standard computing data structures with logarithmic time complexities could be employed to store the image data to achieve more acceleration for Histogram SLIC. This approach could possibly reduce the time overhead of the Histogram SLIC.
- Linux is a popular operating system used in most of the embedded platforms as it is compact and has a small foot print. Combination of Linux and OpenCV facilitates porting of the Histogram SLIC over a wide range of real-time platforms. The algorithm can be implemented on various platforms and to explore the potential for achieving higher speed.
- In the proposed work, a 64-bin colour histogram has been used to extract priori information for initial seeding. Detailed colour histogram of larger bin size could be used to improve upon the current accuracy if high capacity number crunching resources are available. This can provide improved the segmentation performance without runtime penalty.
- Histogram SLIC uses localised colour information and high-level spatial information of the image pixels for seed initialization. If computation resources permit, other cue information like texture or gradient orientation can also be incorporated to provide more detailed priori information, which would further improve the segmentation accuracy. However, it should also be noted that the clustering process would increase in data dimension in this case.
- Histogram SLIC makes use of  $2S \times 2S$  scan size, same as SLIC. However, variable scan size can be explored to induce more content sensitivity, making the algorithm more adaptive in nature.
- Histogram SLIC can be evaluated more precisely by integrating and simulating it with the intended computer vision algorithms. Algorithmic redundancy, if any, could be resolved with a hybrid algorithm that would make the whole application faster.



## References

- [1] C. C. Fowlkes and J. Malik D. R. Martin, "Learning to detect natural image boundaries using local brightness, color, and texture cues," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, May 2004, pp. pp. 530-549.
- [2] C. C. Yu, M. J. Yu and Y. He Y. J. Liu, "Manifold SLIC: A Fast Method to Compute Content-Sensitive Superpixels," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. pp. 651-659.
- [3] A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süssstrunk R. Achanta, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, Nov. 2012, pp. pp. 2274-2282.
- [4] Alexander Hermans, Bastian Leibe David Stutz, "Superpixels: An Evaluation of the State-of-the-Art," in *Computer Vision and Image Understanding*, April 2017.
- [5] Qingxiong Yang, Yihong Gong, Ming-Hsuan Yang, Narendra Ahuja Xing Wei, "Superpixel Hierarchy," in *arXiv:1605.06325*.
- [6] M. A. Ruzon and C. Tomasi, "Color edge detection with the compass operator," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, Fort Collins, CO, 1999, p. pp. 166 Vol. 2.
- [7] J., Belongie, S., Leung, T. et al Malik, "Contour and Texture Analysis for Image Segmentation," in *International Journal of Computer Vision (2001)*, pp. 43: 7.
- [8] NIKLAS ROSÉN PHILIP ELIASSON, "Efficient K-means clustering and the importance of seeding," Bachelor Thesis.
- [9] Oscar Deniz Suarez, José Luis Espinosa Aranda, Jesus Salido Tercero, Ismael Serrano Gracia, Noelia Vállez Enano Gloria Bueno García, *Learning Image Processing with OpenCV*.: Packt Publishing.
- [10] Gary Bradski Adrian Kaehler, *Computer Vision in C++ with the OpenCV Library*.: O'REILLY.
- [11] Peer, and Peter Protzel. Neubert, "Superpixel benchmark and comparison.," in *Proc. Forum Bildverarbeitung*, 2012, pp. pp. 1-12.
- [12] Peer Neubert, "Superpixels and their Application for Visual Place Recognition in Changing Environments," , 2015.
- [13] M. Recky and F. Leberl, "Windows Detection Using K-means in CIE-Lab Color Space," in *International Conference on Pattern Recognition*, Istanbul, 2010, pp. pp. 356-359.

- [14] P. Felzenszwalb and D. Huttenlocher, "Efficient Graph-Based Image Segmentation," in *Int'l J. Computer Vision*, vol. 59, no. 2, September 2004, pp. 167-181.
- [15] A. Stere, K. N. Kutulakos, D. J. Fleet, A. Levinshtein, "Turbopixels: Fast superpixels using geometric flows," in *IEEE Trans. Pattern Analysis and Machine Intelligence* 31(12), 2009, pp. 2290-2297.
- [16] G. Zeng, R. Gan, J. Wang, and H. Zha P. Wang, "Structure-sensitive superpixels via geodesic distance.," in *International Journal of Computer Vision*, 103(1), pp. 1-21.
- [17] David Arthur and Sergei Vassilvitskii, "k-means++: the advantages of careful seeding," in *eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1027-1035.
- [18] Vidya T., Shama, Jayaram M.A., Manjunath A.S. Karegowda A.G., "Improving Performance of K-Means Clustering by Initializing Cluster Centers Using Genetic Algorithm and Entropy Based Fuzzy Clustering for Categorization of Diabetic Patients.," in *Proceedings of International Conference on Advances in Computing. Advances in Intelligent Systems and Computing*, vol 174. Springer, New Delhi, 2013, pp. 899-904.
- [19] Robert, Guenther Walther, and Trevor Hastie Tibshirani, "Estimating the number of clusters in a data set via the gap statistic.," in *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2001, pp. 411-423.
- [20] Andrew Moore Dau Pelleg, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *Proceedings of the 17th International Conf. on Machine Learning*, 2000.
- [21] C., Hansen, L. K., Liptrot, M. G. and Rostrup, E Goutte, "Feature-space clustering for fMRI meta-analysis," in *Hum. Brain Mapp.*, 13:, 2001, pp. 165-183.
- [22] Christian Hennig, Renato Cordeiro de Amorim, "Recovering the number of clusters in data sets with noise features using feature rescaling factors," in *Information Sciences*, Volume 324, 2015, pp. Pages 126-145.
- [23] Y.X., Qin, K., Liu, Y., Gan, S.Z., Zhan, Y Chen, "Feature modelling of high resolution remote sensing images," in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2012, pp. 467-472.
- [24] B., Vedaldi, A., Soatto, S. Fulkerson, "Class segmentation and object localization with superpixel neighborhoods," in *ICCV*, Vol. 9, 2009, pp. 670-677.
- [25] Juanjuan Zhao, Cheng Jiao, Lei Lei, Yan Qiang, Qiang Cui Xiaolei Liao, "A Segmentation Method for Lung Parenchyma Image Sequences Based on Superpixels and a Self-Generating Neural Forest," in *Comput Math Methods Med.*, 2014.
- [26] Smith K., Achanta R., Lepetit V., Fua P. Lucchi A., "A Fully Automated Approach to

Segmentation of Irregularly Shaped Cellular Structures in EM Images.," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2010*, 2010, pp. pp 463-471.

- [27] Fuhai Li, Liang Gao, Zhiyong Wang, Michael J. Thrall, Yehia Massoud, Stephen T. C. Wong Ahmad A. Hammoudi, "Automated Nuclear Segmentation of Coherent Anti-Stokes Raman Scattering Microscopy Images by Coupling Superpixel Context Information with Artificial Neural Networks," in *Machine Learning in Medical Imaging. MLMI 2011.*, pp. pp 317-325.
- [28] Oihana Otaegui, Gorka Vélez Marcos Nieto, "On creating vision-based advanced driver assistance systems.," in *IET Intelligent Transport Systems, Volume: 9, Issue: 1*, January 2015, pp. 59 - 66.
- [29] Liwei, Junliang Xing, Haizhou Ai, and Shihong Lao Liu, "Semantic superpixel based vehicle tracking.," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 2222-2225.
- [30] Yingen Xiong, Kari Pulli, Linda Shapiro Dingding Liu, "Estimating Image Segmentation Difficulty," in *Machine learning and data mining in pattern recognition*, 2011, pp. 484-495.
- [31] Xiaofeng, and Jitendra Malik Ren, "Learning a classification model for segmentation," in *IEEE*, 2003, p. p. 10.
- [32] R. Stolkin., *Scene Reconstruction, Pose Estimation and Tracking.*: I-Tech Education and Publ ISBN 9783902613066, 2007.
- [33] J. Shi and J. Malik., "Normalized cuts and image segmentation.," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8).
- [34] Y. Boykov, and P. Mehrani O. Veksler, "Superpixels and supervoxels in an energy optimization framework," in *European Conference on Computer Vision (ECCV '10)*, 2010, pp. pages 211–224.
- [35] B. Georgescu, and P. Meer M. Christoudias, "Synergism in low level vision," in *In Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2002, pp. pages 150–155.
- [36] X. Boix, G. Roig, B. de Capitani, and L. Van Gool M. Van den Bergh, "Seeds: Superpixels extracted via energy-driven sampling.," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2012, pp. pages 13–26.
- [37] O. Tuzell, S. Ramalingam, and R. Chellappa M.-Y. Liu, "Entropy rate superpixel segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern (CVPR)*, 2011.
- [38] Y. Boykov, and P. Mehrani. O. Veksler, "Superpixels and supervoxels in an energy

- optimization framework," in *Proceedings of the the 11th European Conference on Computer Vision: Part V (ECCV)*, Berlin, Heidelberg, 2010., pp. pages 211–224.
- [39] O. Veksler, and R. Zabih. Y. Boykov, "Fast approximate energy minimization via graph cuts.," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2001, pp. 1222–1239.
- [40] Shehroz S., and Amir Ahmad. Khan, "Cluster center initialization algorithm for K-means clustering.," in *Pattern recognition letters* 25, no. 11 , 2004, pp. 1293-1302.
- [41] C.L. Shafer, S.A Novak, "Anatomy of a color histogram," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Proceedings*, 1992, pp. 599 - 605.
- [42] Wang, Jun-Feng Wu,Hong-Ying Yang Xiang-Yang, "Robust image retrieval based on color histogram of local feature regions," in *Springer Netherlands, 2009 ISSN 1573-7721*.
- [43] E.A, Kostyukova, N.S Bashkov, "Effectiveness estimation of image retrieval by 2D color histogram," in *Journal of Automation and Information Sciences*, 2006, 2006, pp. 84-89.
- [44] James MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967*.
- [45] J.B. and Meijster, A. Roerdink, "The watershed transform: Definitions, algorithms and parallelization strategies.," in *Fundamenta informaticae*, 41(1, 2), 2000, pp. pp.187-228.

## Appendix A. Additional Results

Image: im4.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.6243	0.0688661	0.248457	0.558459	0.674278	0.0724542	0.258059	0.90383
100	0.697544	0.0534258	0.20292	0.56644	0.749246	0.0525644	0.191826	0.971621
150	0.73072	0.0510424	0.173762	0.593849	0.783068	0.0450191	0.180138	1.02241
200	0.795993	0.0434129	0.16472	0.564991	0.798794	0.043957	0.168968	0.921058
250	0.781344	0.0394557	0.144232	0.557528	0.796639	0.0409453	0.151688	0.9275
300	0.795562	0.0378884	0.136208	0.561044	0.838863	0.036185	0.148728	0.912754
350	0.821844	0.0348702	0.129348	0.525168	0.830676	0.0372795	0.135374	0.916529
400	0.820982	0.032869	0.123999	0.597422	0.847264	0.037707	0.138	0.892567
450	0.827014	0.0321436	0.118054	0.542016	0.839293	0.0372536	0.124992	0.8781
500	0.813873	0.03272	0.110623	0.587334	0.841663	0.0290024	0.122549	0.814983

Image: 12084.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.430004	0.0969942	0.0824691	0.65855	0.556694	0.101424	0.0951689	0.985279
100	0.600087	0.0735164	0.0948835	0.611135	0.666376	0.0874994	0.0816763	1.10523
150	0.654383	0.0697793	0.0883531	0.626436	0.706934	0.0673376	0.0803549	1.07281
200	0.670737	0.0573053	0.0829602	0.613634	0.735935	0.0669879	0.0789104	1.04636
250	0.741169	0.0534906	0.0856365	0.620681	0.757523	0.0580372	0.0755414	1.14584
300	0.703009	0.0509129	0.0781746	0.591552	0.746838	0.0550191	0.0715136	1.11112
350	0.750545	0.0463209	0.0803511	0.604447	0.772787	0.0547924	0.0694984	1.08146
400	0.779546	0.0467678	0.0787064	0.586706	0.823812	0.0526227	0.0734505	1.05171
450	0.724815	0.0462821	0.0751441	0.560413	0.790013	0.0512108	0.0668969	1.09111
500	0.773877	0.0454013	0.074334	0.635401	0.817924	0.0459453	0.0690029	1.01483

Image: 21077.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.563812	0.144565	0.208997	0.588866	0.593602	0.159131	0.221318	0.881868
100	0.63981	0.113957	0.189132	0.600301	0.63981	0.113957	0.189132	0.600301
150	0.747461	0.0774542	0.182208	0.598795	0.76608	0.0761264	0.170799	1.05285
200	0.767434	0.063076	0.172651	0.587556	0.780806	0.0732962	0.163495	0.989187
250	0.797393	0.0666965	0.157453	0.567177	0.797224	0.0583805	0.15337	1.04641
300	0.823798	0.0558546	0.158886	0.576869	0.800271	0.0569491	0.144028	0.997938
350	0.803318	0.0531603	0.143923	0.555527	0.838185	0.0474803	0.143159	0.997499
400	0.822783	0.0509841	0.140601	0.584503	0.837508	0.0526616	0.140166	0.959941
450	0.86019	0.0359324	0.145271	0.558355	0.838693	0.0398378	0.130138	0.979707
500	0.858666	0.0454984	0.134445	0.615907	0.864252	0.0415282	0.131202	0.923422

Image: 24077.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.595613	0.264519	0.330988	0.590562	0.61181	0.282621	0.329801	0.886221
100	0.634455	0.236728	0.304686	0.597036	0.704985	0.234429	0.311395	0.989683
150	0.674241	0.203716	0.291786	0.605126	0.752477	0.188159	0.297834	0.999193
200	0.704592	0.175983	0.281996	0.589496	0.729989	0.197143	0.272258	0.9596
250	0.730461	0.161702	0.276431	0.562763	0.814672	0.168645	0.278836	1.00044
300	0.75515	0.147357	0.266578	0.567403	0.801148	0.153237	0.265615	0.96619
350	0.780154	0.144028	0.262981	0.552708	0.815616	0.150964	0.260066	0.989846
400	0.775751	0.132266	0.255034	0.623828	0.819311	0.140019	0.259114	0.969224
450	0.787388	0.128646	0.251318	0.591304	0.828039	0.126074	0.248144	0.982868
500	0.811999	0.117422	0.24701	0.60945	0.85037	0.12963	0.250661	0.915032

Image: 37073.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.403051	0.254849	0.295486	0.546833	0.484855	0.246916	0.336634	0.884811
100	0.507517	0.17417	0.277737	0.570048	0.582578	0.178023	0.282195	1.06421
150	0.604466	0.133497	0.2753	0.589111	0.605793	0.144002	0.258942	1.03726
200	0.638514	0.120867	0.271136	0.576073	0.68218	0.123814	0.265305	1.0177
250	0.670241	0.111379	0.248097	0.571366	0.707384	0.108043	0.247237	1.06252
300	0.706942	0.0961587	0.242363	0.625436	0.697435	0.107732	0.224703	1.02697
350	0.71457	0.0925124	0.228055	0.584168	0.760889	0.0987235	0.234084	1.0453
400	0.733142	0.0881082	0.224312	0.602277	0.760226	0.100116	0.22586	1.00635
450	0.752708	0.0831342	0.219433	0.539898	0.794716	0.0897663	0.218504	1.04352
500	0.760115	0.0771692	0.208389	0.606702	0.80367	0.0826484	0.215133	0.984098

Image: 38092.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.630952	0.131722	0.282184	0.656942	0.703602	0.120401	0.294493	0.919782
100	0.708333	0.0912753	0.242883	0.624398	0.783425	0.0836134	0.219725	1.06672
150	0.754884	0.072778	0.206998	0.623496	0.786783	0.0808738	0.187611	0.997926
200	0.767552	0.0719425	0.185223	0.60358	0.796093	0.0660877	0.180885	0.977209
250	0.786477	0.061146	0.176117	0.598881	0.820971	0.0632185	0.167403	1.07944
300	0.794567	0.0574478	0.16494	0.612212	0.833791	0.0609841	0.158564	1.02145
350	0.836996	0.0528883	0.161232	0.595556	0.829212	0.0542613	0.149546	0.994126
400	0.820513	0.0517548	0.152636	0.621236	0.848596	0.0563144	0.154384	0.981296
450	0.845085	0.0551162	0.150103	0.583654	0.85699	0.0474673	0.146652	0.995019
500	0.843407	0.0457057	0.14261	0.660575	0.870116	0.0477069	0.143429	0.904501

Image: 45096.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.56993	0.0337692	0.140967	0.597932	0.619231	0.0307446	0.190185	0.781295
100	0.644056	0.0351487	0.116068	0.546393	0.615734	0.0379596	0.125867	0.825475
150	0.678671	0.0323638	0.101607	0.556497	0.700699	0.0292679	0.120065	0.758886
200	0.745455	0.0269946	0.0987906	0.561512	0.828671	0.0238276	0.129303	0.736289
250	0.705944	0.0233742	0.6678810	0.541612	0.747552	0.028018	0.106342	0.762163
300	0.710839	0.0244234	0.0774683	0.542407	0.804545	0.023711	0.105165	0.706675
350	0.745105	0.0220918	0.0758363	0.566497	0.773776	0.0241967	0.0952156	0.715351
400	0.775874	0.0219234	0.0750448	0.535619	0.748601	0.0254791	0.088336	0.701702
450	0.826573	0.0194105	0.0772372	0.561022	0.808392	0.019812	0.0898911	0.713766
500	0.788462	0.0208354	0.0682775	0.601831	0.79021	0.0206864	0.0864047	0.660063

Image: 41069.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.554351	0.11077	0.142024	0.577762	0.551098	0.102687	0.100963	0.891555
100	0.626728	0.0777521	0.105812	0.622582	0.689618	0.0713143	0.085723	1.10182
150	0.657902	0.0568714	0.0905969	0.631377	0.793711	0.0448313	0.088754	1.05447
200	0.729195	0.0515152	0.0865564	0.599308	0.742749	0.0471823	0.073458	1.06492
250	0.795066	0.0401293	0.0843616	0.585434	0.762808	0.0395658	0.0689858	1.09534
300	0.779615	0.0366643	0.0779066	0.568839	0.722689	0.0444039	0.0610586	1.08084
350	0.787476	0.0356928	0.073292	0.597505	0.785579	0.0377264	0.0680969	1.1381
400	0.776633	0.0361915	0.0718567	0.616719	0.796151	0.0356669	0.0668016	1.02338
450	0.805909	0.0332187	0.0707015	0.575444	0.809976	0.0348055	0.0645719	0.9787
500	0.799946	0.0337627	0.0648301	0.663991	0.808349	0.0330309	0.0618775	0.975056



Image: 42012.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.494038	0.179843	0.194383	0.591289	0.548179	0.172667	0.217061	0.924126
100	0.593458	0.144934	0.180442	0.587925	0.697067	0.115764	0.157515	1.01904
150	0.643248	0.122441	0.162972	0.60554	0.697067	0.115764	0.157515	1.02609
200	0.652111	0.102169	0.15551	0.590131	0.673542	0.105038	0.138493	0.998013
250	0.712375	0.0914437	0.14784	0.529443	0.742991	0.0948828	0.143538	1.0453
300	0.732678	0.0785811	0.147152	0.576921	0.772478	0.0825578	0.137985	0.986648
350	0.72865	0.0782443	0.130445	0.570531	0.76426	0.0749736	0.130066	1.01795
400	0.783435	0.0626097	0.1368	0.605319	0.793748	0.0705954	0.13168	0.958239
450	0.785852	0.0646628	0.132161	0.561991	0.803577	0.0661395	0.125658	1.00728
500	0.804061	0.058931	0.129989	0.638138	0.839832	0.0563144	0.128172	0.948717

Image: 183066.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.565517	0.0628429	0.173531	0.511014	0.627321	0.0503688	0.188431	0.778953
100	0.628382	0.0527782	0.147776	0.522511	0.720424	0.0411591	0.160939	0.891152
150	0.679045	0.0534841	0.128411	0.524088	0.658621	0.0522017	0.124642	0.865755
200	0.703714	0.0415088	0.121971	0.518409	0.692838	0.0399933	0.121066	0.849603
250	0.752255	0.0323249	0.119855	0.500349	0.754642	0.0337044	0.119392	0.868694
300	0.774801	0.0276294	0.11454	0.502278	0.806897	0.0255568	0.121895	0.861645
350	0.776127	0.0247084	0.106718	0.493769	0.835809	0.0236138	0.118812	0.886115
400	0.783289	0.0291708	0.102831	0.535787	0.821485	0.0226359	0.113033	0.841846
450	0.788859	0.0281151	0.0994516	0.493833	0.82679	0.022325	0.104398	0.870095
500	0.791512	0.02454	0.0963824	0.540298	0.858886	0.020803	0.107204	0.818207

Image: 185092.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.526994	0.0949994	0.234682	0.518332	0.563054	0.126741	0.262318	0.726753
100	0.580983	0.0761524	0.193987	0.508797	0.674658	0.0650125	0.230902	0.781341
150	0.665995	0.0603623	0.180587	0.509934	0.668815	0.0538079	0.197937	0.767686
200	0.649678	0.0555502	0.15798	0.51018	0.69722	0.0515282	0.184557	0.746041
250	0.678284	0.0448831	0.147747	0.495113	0.681104	0.0477588	0.162065	0.769228
300	0.740532	0.0405179	0.148357	0.496355	0.730862	0.0389829	0.160297	0.739381
350	0.750403	0.0391772	0.137769	0.486359	0.764907	0.0327653	0.163277	0.731904
400	0.764504	0.032571	0.136349	0.505579	0.73469	0.0420982	0.151353	0.700257
450	0.769541	0.0300451	0.130871	0.487682	0.745367	0.0376099	0.141081	0.725323
500	0.782635	0.0283612	0.124871	0.533642	0.760878	0.0321306	0.139501	0.683351

Image: 187058.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.379429	0.137253	0.0974808	0.544965	0.3584	0.154572	0.0871595	0.787133
100	0.441371	0.103542	0.0937652	0.539706	0.517257	0.0918971	0.0948569	0.897749
150	0.513371	0.0828298	0.0914086	0.538284	0.529829	0.0906147	0.0922881	0.896651
200	0.497829	0.0763013	0.082475	0.537245	0.541257	0.0791446	0.0858562	0.868293
250	0.532343	0.0632379	0.0826737	0.51935	0.536914	0.0742094	0.082369	0.900592
300	0.552914	0.0591253	0.0815219	0.521878	0.576	0.060725	0.0849343	0.852493
350	0.574857	0.0577004	0.0796541	0.513222	0.621257	0.0532574	0.0886873	0.876998
400	0.5696	0.0564957	0.0761055	0.549993	0.573257	0.0595786	0.078853	0.851702
450	0.5776	0.0554012	0.0770474	0.525218	0.6672	0.053931	0.0880331	0.872506
500	0.623086	0.0497082	0.0780239	0.559866	0.5824	0.0587108	0.0751489	0.825507

Image: 187099.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.350493	0.16406	0.191727	0.547866	0.369581	0.193308	0.173198	0.791988
100	0.357759	0.156942	0.155397	0.536369	0.34298	0.151035	0.13238	0.926591
150	0.465517	0.119792	0.168427	0.539489	0.386576	0.144002	0.133785	0.92264
200	0.448768	0.115796	0.15084	0.529843	0.441133	0.118179	0.147748	0.896534
250	0.489163	0.102797	0.155338	0.519605	0.46601	0.103471	0.14429	0.938735
300	0.49298	0.0909709	0.146727	0.520948	0.478941	0.0971108	0.136898	0.899864
350	0.534606	0.0916251	0.148172	0.512951	0.495443	0.0945525	0.138485	0.922642
400	0.517611	0.0868906	0.138924	0.545587	0.509852	0.102214	0.13601	0.882235
450	0.539778	0.0837883	0.140499	0.511131	0.515764	0.090103	0.131798	0.90041
500	0.564655	0.0813596	0.135852	0.561624	0.533005	0.0893453	0.133878	0.857598

Image: 188025.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.46671	0.161022	0.321746	0.541909	0.557084	0.14112	0.396427	0.78712
100	0.552063	0.129449	0.302747	0.526983	0.615695	0.105032	0.305232	0.876269
150	0.629666	0.0913077	0.286915	0.529089	0.634578	0.0907313	0.281946	0.8837
200	0.638616	0.0857831	0.267023	0.518445	0.663065	0.0771174	0.266331	0.866688
250	0.64189	0.0826614	0.241262	0.504833	0.687732	0.0756407	0.252991	0.90514
300	0.691334	0.0696239	0.238219	0.512907	0.741214	0.0602522	0.246533	0.858355
350	0.742414	0.0631278	0.241497	0.506444	0.748308	0.0624672	0.238918	0.887821
400	0.723641	0.0664827	0.224183	0.532541	0.729644	0.0606667	0.233798	0.843017
450	0.758895	0.0583869	0.224907	0.506092	0.752347	0.0580566	0.221029	0.86397
500	0.75704	0.0583027	0.210354	0.556046	0.725824	0.0594685	0.208968	0.822039






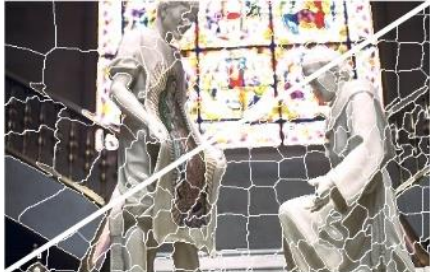


Image: 189006.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.469286	0.154312	0.296945	0.535661	0.513143	0.170601	0.325379	0.782481
100	0.590205	0.108665	0.279719	0.523858	0.598783	0.118659	0.269624	0.863768
150	0.634339	0.0882896	0.242605	0.526902	0.678749	0.0918906	0.2511	0.845294
200	0.692446	0.0805241	0.239485	0.515726	0.713199	0.0794878	0.238404	0.812321
250	0.683453	0.0770138	0.211708	0.505987	0.739485	0.0757702	0.226425	0.869821
300	0.759408	0.0599608	0.217507	0.507116	0.765634	0.0691576	0.222795	0.815555
350	0.751799	0.0614633	0.19964	0.497748	0.768124	0.0661136	0.20833	0.829463
400	0.769092	0.0603429	0.193633	0.535279	0.788046	0.06169	0.209736	0.790872
450	0.779745	0.0515411	0.1897	0.496069	0.77518	0.059747	0.190436	0.813755
500	0.777947	0.0544815	0.177757	0.541412	0.801328	0.0546952	0.191364	0.740826

Image: 189029.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.548948	0.116696	0.269093	0.525788	0.531212	0.169682	0.273207	0.757992
100	0.555903	0.11544	0.215431	0.511956	0.657277	0.0851225	0.241164	0.827348
150	0.698313	0.0743065	0.213096	0.511097	0.725091	0.0648247	0.22731	0.813481
200	0.704225	0.0736394	0.197071	0.508901	0.736046	0.0690345	0.211798	0.784132
250	0.727873	0.057409	0.179664	0.488869	0.745957	0.0613144	0.191091	0.814734
300	0.747174	0.053795	0.16944	0.491597	0.758825	0.0573312	0.177745	0.767417
350	0.763867	0.051431	0.164132	0.486939	0.76265	0.0522276	0.170821	0.79176
400	0.779691	0.0455114	0.157665	0.518604	0.764389	0.0523636	0.169769	0.755481
450	0.776213	0.0469362	0.149929	0.481845	0.800383	0.0447342	0.163163	0.777532
500	0.80073	0.0404078	0.146642	0.531067	0.795166	0.0412173	0.155873	0.714905

Image: 189096.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.592376	0.136838	0.203949	0.552574	0.578468	0.16303	0.175872	0.798689
100	0.608173	0.113963	0.178044	0.528277	0.718407	0.0782573	0.185141	0.90958
150	0.651957	0.0869813	0.161389	0.545184	0.724416	0.0808997	0.162432	0.889658
200	0.650069	0.089423	0.149261	0.523302	0.722871	0.0696692	0.154263	0.886455
250	0.697115	0.0776096	0.147043	0.514631	0.748455	0.0558805	0.149041	0.912706
300	0.73592	0.0579659	0.146095	0.518859	0.781422	0.0574413	0.145964	0.893219
350	0.712569	0.058633	0.135718	0.509569	0.774382	0.0463987	0.14045	0.900159
400	0.743647	0.0533028	0.135479	0.545576	0.747253	0.0534258	0.131604	0.854113
450	0.729911	0.0559388	0.130395	0.509168	0.777988	0.0480632	0.131681	0.885903
500	0.751374	0.0470981	0.12705	0.54913	0.79035	0.0439246	0.133964	0.811318

Image: 196027.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.3689	0.133199	0.231763	0.53223	0.442962	0.135472	0.284433	0.746207
100	0.399498	0.115258	0.187	0.513182	0.517496	0.100271	0.224171	0.843152
150	0.525498	0.0869036	0.191634	0.502037	0.552173	0.0858544	0.217182	0.806831
200	0.568806	0.0911328	0.188881	0.501335	0.650871	0.0767612	0.224204	0.753959
250	0.591087	0.0772145	0.169311	0.489619	0.666562	0.0647081	0.208685	0.774388
300	0.672368	0.0594815	0.175385	0.484636	0.708771	0.0518196	0.203459	0.736656
350	0.704849	0.0544038	0.171156	0.478604	0.73121	0.0482963	0.200793	0.713899
400	0.692609	0.0559258	0.161007	0.502831	0.739369	0.0587302	0.195179	0.720076
450	0.71034	0.0477069	0.156758	0.478235	0.766201	0.0445528	0.186524	0.726559
500	0.760082	0.0405243	0.156642	0.525202	0.777342	0.0393974	0.18338	0.683434

Image: 196040.jpg								
	SLIC				Histogram SLIC			
Superpixels	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*	Boundary Recall	Under segmentation Error	Boundary Precision	Runtime*
50	0.313194	0.132104	0.181075	0.547529	0.339238	0.117558	0.16525	0.787481
100	0.357218	0.106988	0.144639	0.531824	0.462322	0.111178	0.147721	0.923178
150	0.456637	0.0870137	0.147575	0.535447	0.513882	0.0803492	0.144978	0.919528
200	0.448176	0.0851031	0.131283	0.534147	0.500793	0.0878686	0.131177	0.901156
250	0.531333	0.0744231	0.143178	0.522583	0.529746	0.0733804	0.132973	0.942603
300	0.573374	0.0702457	0.144417	0.525142	0.574432	0.0677392	0.131854	0.907374
350	0.535563	0.070712	0.128273	0.521858	0.55817	0.06669	0.123591	0.929873
400	0.559889	0.0684452	0.134376	0.548155	0.604971	0.0676032	0.130672	0.888313
450	0.579852	0.0641576	0.134309	0.495091	0.580645	0.0678104	0.11878	0.920446
500	0.630222	0.0631214	0.136258	0.565467	0.634849	0.0623312	0.129777	0.869064

Comparison between normal SLIC and Histogram SLIC( Upper half: 50 super pixels; Lower half 350 super pixels	
Normal SLIC	Histogram SLIC
 <p>12084.jpg</p>	 <p>12084.jpg</p>
 <p>21077.jpg</p>	 <p>21077.jpg</p>
 <p>24077.jpg</p>	 <p>24077.jpg</p>
 <p>37073.jpg</p>	 <p>37073.jpg</p>





38092.jpg



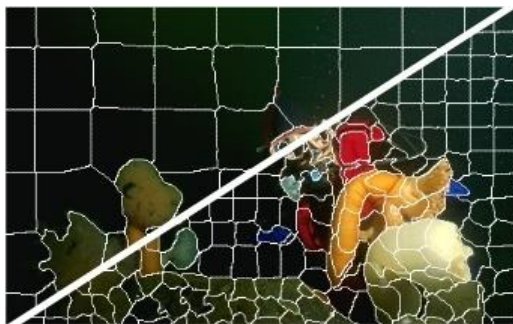
38092.jpg



41069.jpg



41069.jpg



45096.jpg



45096.jpg



Im4.jpg



Im4.jpg

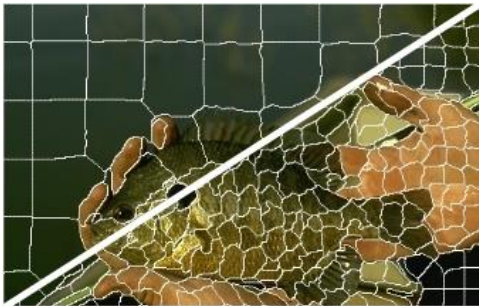




42012.jpg



42012.jpg



180592.jpg



180592.jpg



183066.jpg



183066.jpg



187058



187058





187099.jpg



187099.jpg



188025.jpg



188025.jpg



189006.jpg

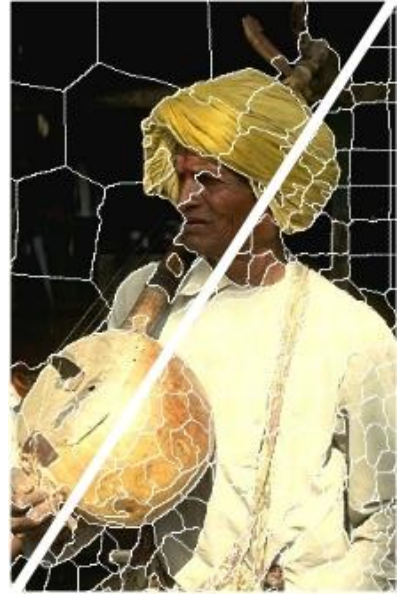


189006.jpg





189029.jpg



189029.jpg



189096.jpg



189096.jpg



196027.jpg



196027.jpg



196040.jpg



196040.jpg

## Appendix B. OpenCV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning software library. It was specially built to provide a common framework to develop machine vision applications on varied hardware platforms. The library has optimized algorithms for specialized image processing functionalities. It includes a comprehensive collection of state-of-the-art computer vision and machine learning algorithms. Such algorithms are used for applications like detection and classification of human actions or visual reconstruction of human faces in 3-D from 2-D image representation. It is also used extensively for identification and tracking of objects in motion over a camera. It supports 3D image processing also and has been widely used for extraction of 3D models from objects and production of 3D point clouds from stereo cameras.

OpenCV can be also employed for non-typical tasks like stitching of image segments together for production of a high resolution images from the entire scene. One of its prime uses is seen in image retrieval from vast amount of image database in an automated manner. It is used in professional photographic platforms for red eye correction occurring due to poor lightening conditions on images taken using flash lightening. In highly evolving technologies, OpenCV is been used for recognition of image scenery and annotation with markers for overlay with augmented reality.

OpenCV currently supports C++, C, Java, Python and MATLAB based interfaces and can operate on Windows, Linux, Android and Mac OS platforms. However, it emphasizes more on real-time vision applications and takes advantage of MMX and SSE based instructions when platform support is available. In future releases, it is planning to extend support for Cuda and OpenCL interfaces, which are being actively developed in the next phase. As OpenCV is natively written in C++, it has a template interface that adapts seamlessly with STL containers in C++.

Although MATLAB is popular for image processing applications, OpenCV has peculiar advantages over MATLAB when it comes to real-time computing environment. Since OpenCV is optimized specifically for image processing application it runs much faster than MATLAB. It can be easily ported to new hardware platform supporting C. It is freely available as an open source bundle unlike MATLAB and hence it is widely becoming a crucial tool in computer vision community.



## Appendix C. Real-time environment

Most of the superpixel algorithms provide excellent results for evaluation purpose. However, while implementing these algorithms on embedded platforms (e.g. Raspberry Pi, Beagle bone etc.), several other considerations have to be made. These systems are highly application specific and resource constrained. As super pixel segmentation is a pre-processing step, its complexity should be kept to a minimum level. Intensive arithmetic computations should be optimized. It is always advisable to avoid floating point calculations where ever possible on embedded systems as they may result into critical quantization errors. The algorithm should also avoid dynamic memory allocation especially on Linux based systems.

OpenCV comes with numerous dependencies to be installed on the Linux platform [10]. It has specialized high level API for basic image processing functionalities. In case of image data storage in internal computing memory, the Mat data type can be accessed in two ways. If high reliability of the system is of concern, then the API `Mat.at<datatype>(rows,cols)` can be used to access individual pixels in a 2-D representation. If the application demands more speed, pointer based approach can be followed to access pixel values as it does not employ value range check. OpenCV also provides a basic GUI functionality through HighGUI module, which was utilized to test results over various parameters in the project.