ASC^{3}I: ALGORITHM FOR SPATIALLY CONTIGUOUS COLOR-BASED CLUSTERING OF INFORMATION

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

M.S.

by

Chinmaya Joshi

August 2016

ABSTRACT

Characterization and identification of processing properties, including device performance metrics and defects, is a critical challenge in the semiconductor industry. Metrics of interest may include, for example, transistor V_t thresholds, channel conductance, lithographic critical dimension (CD) variations, pattern overlay error or defect densities. Modern inspection tools are capable of generating enormous data sets that require machine learning and interpretation to identify and detect critical defect and process variations. These challenges are particularly critical in maintaining the integrity of lithographic process flows. One particular challenge can be expressed in a computational framework, as the need to segment a spatially resolved 2D metric(image) into finite and specific number of contiguous clusters while minimizing the variance within each cluster. This is equivalent to segmenting physical images and thus, can be applied to a larger community of applications beyond lithography.

In this work, an efficient and fast two stage clustering algorithm has been developed. In the first stage, an image is segmented into spatially homogeneous regions based on a maximum variance threshold yielding a finite set of clusters. In the second step, these clusters are then intelligently merged to yield the specified (generally small) number of homogeneous and spatially connected regions. A number of alternative measures of the region variance were considered. Images from the Berkeley Segmentation Dataset and Intel's semiconductor dataset were used to test the algorithm. Additionally, the results and their performance were compared and shown to be visually similar or better, and computationally more efficient, than those of several state-of-the-art segmenta-

tion algorithms. The algorithm is shown to be computationally efficient even for large scale datasets, and can be extended to incorporate multi-spectral information in the clustering criteria. We have demonstrated this capability using three color channels of an RGB image.

BIOGRAPHICAL SKETCH

Chinmaya obtained his Bachelors in Technology (B Tech.) in Metallurgy and Materials Science Engineering from College of Engineering, Pune in India. His undergraduate thesis was on recovery of pure metallic iron powder from mill scale (a waste product of steel plant) using hydrogren gas as the reducing agent. During his undergraduate study at College of Engineering, Pune, Chinmaya was fortunate to have the opportunity to intern at MIT Media Labs in the Camera Culture group from May 2011 to February 2012 under Prof. Ramesh Raskar. The work done in this internship was also responsible for further arousing his interest in image processing and computer vision. In 2012, Chinmaya interned at MIT Materials Science and Engineering under Prof. Eugene Fitzgerald.

After his B Tech., Chinmaya got a rare opportunity to work under Mr. Ratan N. Tata, the former chairman of the $100+ billion Tata Group. His experience there included getting a glimpse into how very senior leadership thinks, operates and handles day-to-day business. In addition to this, he also got a rare experience to analyze several startup companies across the globe in the renewable energy, e-commerce and aerospace sectors, interacting with whom has inspired him and ignited his curiosity further in picking and solving technically challenging problems.

Chinmaya continued his Masters study at Cornell University in Materials Science and Engineering, and was fortunate to have an opportunity to work under Professor Michael Thompson for his masters research in the thermal transient lab. His research work started by working on the project of sub-millisecond laser spike annealing to improve InGaAs electrical performance for future channel material purpose. He then got an opportunity to intern at Intel Corporation in the Computational Lithography Group. Through extended in-

ternships at Intel, Chinmaya worked on his Masters thesis under Prof. Michael Thompson on efficient algorithms for spatial and color-based clustering of images. He received his Masters of Science degree in Materials Science and Engineering in May 2016 from Cornell University.

Dedicated to My Beloved Parents, Dr. Mrs. Kalyani R. Joshi and Mr. Rajiv M. Joshi who have taught me the importance of simplicity and hard work, and Boss (Mr. Ratan N. Tata), whose humbleness, simplicity, hard-work and strong ethics inspire me to follow him in every walk of my life.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and respect for Prof. Michael Thompson, for his advice and support of my research. His critique of the experimental data as well as this manuscript has greatly helped me in progressing towards a better researcher. I want to thank him for always being patient with me and for helping me out when I faced obstacles. It was because of Prof. Thompson's permission and encouragement, that I got an opportunity to intern at Intel and get a glimps of industrial research early in my career. His honesty with himself towards his work, research and the ideal teacher-student relationship, is what greatly inspires me and has inculcated a desire to emulate similar qualities in my career as well. Through this thesis project, he has given me an invaluable experience of an in-depth approach towards solving a challenging problem statement, and has also inculcated the abilities of being an independent researcher.

I am also grateful to Prof. Kavita Bala for being my thesis committee member and all the support and the suggestions she has given on my thesis project during our regular updates. I am especially thankful to her for always being available to meet personally or email, despite having a very hectic schedule.

I am also thankful to have the opportunity, provided by Prof. Thompson, to intern at Intel Corporation's Computational Imaging Techniques group. I am grateful to Dr. Vivek Singh, director - Computational Imaging Techniques group for allowing me to intern in his group and my PI at Intel, Dr. Bikram Baidya. I am grateful to Dr. Bikram Baidya for being patient with me in the initial stages of the internship when I was still trying to get hold of the learning curve and further ahead in the internship to have made sure that my work was moving in the right direction and at right pace. I am also thankful to my mentors, Dr.

Camal Bilgin Cagatay, Dr. Hale Erten and Dr. Sumedh Risbud for not only guiding me in the technical details of research project, but also teaching me how to deal with stress and uncertainities in an industrial research environment.

I sincerely thank Tata Sons Ltd. for giving me a study-leave to pursue my Masters degree at Cornell University.

It is my pleasure to thank my subgroup members Victoria Sorg, Megan Hill, Suki Zhang, Mardochee Reveil, Jingyang Wang at Cornell University. I would also like to thank my other colleagues in the Thompson Group, David Lynch, Chen-yang Chung, Alan Jacobs, Robert Bell. I also feel fortunate to have friends like Dr. Safir Merchant, Dr. Kshitij Auluck, Aibar Nurmukhanov who have guided me in difficult times and have also been present in my happy moments.

I sincerely thank my parents for their unconditional love, numerous sacrifices, moral support throughout my life. Last but not the least, I thank the almighty God for giving me an able mind to work on challenging problem statement like this one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Transister feature sizes in the 1980s and 1990s were larger than the wavelength of the light used in lithography making the task of patterning relatively straightforward. Since the early 2000s, however, the feature sizes have continued to shrink from 130 nm to 15 nm and 10 nm today, generally following Moore's law. However, the wavelength of light used to pattern these featurees has remained at 193 nm making the task of defining the desired patterns tougher and tougher. While EUV (Extreme Ultraviolet) light sources exists at 13nm, the source power continues to have issues limiting the introduction of EUV into process flow.

Resolution Enhancement Techniques (RET), along with Computational Lithography (CLT), come to aid here by introducing several tricks to achieve the desired feature size by compensating for the errors due to diffraction or process effects. Optical Proximity Correction (OPC) and Inverse Lithographic Techniques (ILT) are two computational lithography techniques that introduce suitable additions and adjustments to the mask design respectively. These adjustments ensure the nearly ideal features are defined at the 10 nm node despite continued use of 193 nm laser light sources for the lithography.

Empirically testing these 'tricks' is very expensive and hence a computational route is normally taken to develop models that will generate the desired design. These computational models must process a large amount of information under a strict time constraint with data on the order of 10^{12} (1 trillion) pixels during the manufacturing stage. As result, there is a need of intelligent merging algorithms to reduce a dense pixelated image, containing many thousands

of 'clusters', into a small number of spatially homogeneous clusters while retaining the shape information. This is necessary to help the manufacturing engineers understand the design being patterned. In addition to the general segmentation challenge, these clustering algorithms must be reliable, efficient and robust against outliers and noise.

Identifying spatially contiguous clusters, based on some assigned thresholds, can also help to identify defect features or regions of interest in the semiconductor image. This information could be used in a variety of ways from being a verification step to making a suitable rectification or change in the mask design to avoid a defect feature.

Population maps and MRI (Magnetic Resonance Imaging) datasets are some other applications where similar shape connectivity and efficient segmentation algorithms are needed. Population maps have been used to show the distribution of people in a given region, but recently they are also being used to graphically display several kinds of information, e.g. crime rate or literacy rates. This graphical information in the form of maps is used for decision making and in predictive analytics. A population map could range from a county, district, state, an entire country all the way to the entire world. Depending on the size of the data to be processed, and the need for accurate decision making, suitable parameters would be provided to a clustering algorithm that would efficiently generate the desired number of spatially connected regions with similar characteristics. Similarly, with MRI datasets, one can use a clustering algorithm to accurately identify the region of interest and show it as a spatially connected region/s for further diagnosis.

The common need of the applications is the division of an image, based

on a variety of constraints, into clusters according to their similarity and spatial connectivity such that each segmented cluster should be contiguous (spatially connected) and pixels with a common 'color'. Constraints can be of two types: a color threshold value(labeled in this work as eps) or the number of clusters(similarly labeled in this work as k). In the first case, we need to divide the image into homogeneous regions such that each cluster fits within a threshold range provided. In the second case, the division must result in a fixed number of segments while minimizing the standard deviation i.e. maintaining the color uniformity and shape connectivity within each segment. Additionally, the algorithm should be capable of handling large image datasets and thus needs to be computationally fast and efficient.

We also explore the prospect of handling multispectral information in our clustering model. With more information or attributes available per pixel or data point, the resulting clustering outcomes should be significantly better. These attributes in the real use-case scenario could be several metrics of interest such as transistor V_t thresholds, channel conductance, lithographic critical dimension (CD) variations, pattern overlay errors or defect densities. We demonstrate this capability by using three-channel information contained within RGB images.

1.1 Color Image Segmentation

Image Segmentation is the process of partitioning pixels of an image into groups of coherent properties to facilitate practical manipulation, recognition, and object-based analysis of multimedia resources[1]. It is also the first step in image analysis and pattern recognition and one of the most difficult tasks in the pro-

cess, as it determines the quality of the final results of analysis. Unfortunately, image segmentation is also a psychophysical problem and thus, it is not possible to obtain a purely analytical solution[2]. Monochrome image segmentation has been an active area of research for several decades and continues to be so today. Along with this, color perception in human beings as a combination of tristimuli R (Red), G (Green) and B (Blue) has been extensively studied. From R, G, B representation, several other kinds of color representations (color-spaces) using either linear or non-linear transformations have been derived for the purpose of understanding and analyzing color images. Several color spaces such as RGB, HSI, CIE L*c*v* are used in image segmentation, but none dominate the others for all types of color-images[3]. Several techniques in monochrome image segmentation, when applied to operating in different color spaces, led to the field of color image segmentation. Color image segmentation has since become an active area of research in the image processing community as: (1) color images provided more information than gray scale images, and (2) the power of personal computers had increased rapidly allowing PCs to be used to process color images.

Hundreds of methods for color image segmentation have been proposed in the past years. These methods can be mainly classified into two categories: one is contour paths and those based on region-analysis. Methods of the first category use discontinuities in an image to detect edges or contours, and then use them to partition the image. Methods of the second category try to divide pixels in an image into different groups corresponding to coherent properties such as color, etc., using a decision criteria to segment an image into different regions according to the similarity of the pixels.[2]

1.1.1 Region-based Color Image Segmentation

Region-based segmentation techniques mainly include region growing methods and clustering. Recently, there appears to be an increasing trend towards treating the segmentation problem as an unsupervised classification problem or clustering problem[4][5][6][7][8]. In these methods, segmentation is obtained as the global minima of the criterion functions with the associated attribute distances between the prototypes and the image pixels[8]. By partitioning pixels according to their global feature distribution, these methods achieve good global partitioning results for most of the pixels. But, in these methods, the spatial relationship of the pixels is rarely considered. This loss of spatial information of the pixels probably leads to unreasonable segmentation results as the pixels that are similar in color, but different in spatial extent, will be grouped into one region.

Region growing along with region splitting, region merging and their combination attempt to group pixels into homogeneous regions. In the region growing approach, a seed region is first selected, then expanded to include all homogeneous neighbors, and this process is repeated until all pixels in the image are classified. One problem with region growing is its inherent dependence on the selection of seed region and the order in which pixels and regions are examined. On the contrary, in a region splitting approach, the initial seed is simply the whole image. If the seed region is not homogeneous, it is usually divided into four squared sub-regions, which become new seed regions. This process is repeated until all sub-regions are homogeneous. The major disadvantage of region-splitting is that the resulting image tends to mimic the data structure used to represent the image and comes out too square[3]. Additionally, the

process of selecting seeds and performing region growing, and/or splitting on them, is computationally expensive process and hence slow.

These techniques often work best on images that are homogenous and appear to be less sensitive to noise because homogeneity is typically determined statistically. They are better than feature space thresholding or clustering techniques as they take into consideration, the feature space and the spatial relation between pixels, simultaneously. One limitation, however, with all region based approaches is that they are sequential by nature which leads to their inherent dependence on the starting point and the order in which pixels are examined.

1.2 Problem Statement

In this thesis, I consider a very specific challenge of the segmentation process. Given spatially resolved data, a heuristic threshold and a desired 'k' number of clusters, we need to develop an algorithm that

(a) Divides the image into spatially connected contiguous clusters based on an assigned attribute threshold (eps)

(b) Intelligently merges the clusters to have exactly 'k' contiguous clusters while minimizing the variance i.e. maintain shape reference

(c) Is able to scale to extremely large datasets of the order of 10^{12} datapoints (1 trillion)

1.3 Proposed Algorithm

We have approached to solve this problem using concepts of region growing followed by region merging and/or splitting without having the need to identify the seeds. In order to improve the run time complexity of the algorithm, we have used the Breadth First Search (BFS) algorithm to implement the region growing algorithm. If the number of clusters formed are more than those desired by the user (k), we rank the clusters formed according to their sizes, retain the largest k clusters and merge all the remaining pixels individually again using a BFS inspired approach. The run time complexity of this algorithm is lower than those of the global cluster methods. One of the problems with this method, as with all unsupervised methods is that the selection of the thresholds are subjective and are image dependent. Another limitation is that it is not applicable to images with shadows or shading. However, such characteristics are not common within the images that must be analyzed within the computational lithography field.

CHAPTER 2

LITERATURE REVIEW

Several image segmentation and data clustering methods that were useful at one and/or more steps in developing the core concepts of our algorithm are initially reviewed. Following this, literature describing the state-of-the-art segmentation approaches and relevant to the current problem statement we aim to solve has been reviewed.

2.1 Region Based Segmentation

Contextual Segmentation are more successful in separating individual objects because they account for closeness of pixels that belong to an individual object. Two basic approaches to contextual segmentation are based on pixel discontinuity or similarity. Discontinuity-based techniques try to locate complete boundaries enclosing relatively more uniform regions and assume that the measured quantity changes abruptly over the boundary. Similarity-based techniques, on the other hand, try to generate uniform regions by grouping spatially connected pixels together that satisfy certain similarity criteria. [9]

Region based segmentation is a type of similarity-based contextual image segmentation approach. Region-based methods assume that the neighboring pixels within one region have similar value. A pixel is compared with its neighbors according to a similarity criterion. Upon satisfying this similarity criterion, it can be set to belong to the cluster as one ore more of its neighbors. Selecting this similarity criterion is a crucial step and the results are influenced by noise in all instances. The manner in which the adjacent neighbors are chosen also

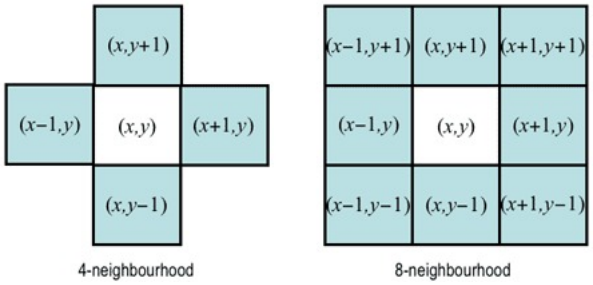


Figure 2.1: Four and Eight Neighborhood Connectivity

impacts the results. Region-based methods can be broadly divided into seeded region growing, unseeded region growing, region splitting and merging and fast scanning algorithms.[10]

2.1.1 Seeded Region Growing

The neighboring pixels of a set of points, known as seed points are examined, to determine whether they could be classified to cluster of the seed point or not. This makes seeded-region growing one of the simplest region-based segmentation methods. The steps in the procedure are as follows:

1. Start with a number of seed points which have been clustered into n clusters, called C_1, C_2, C_3, ..., C_n and the positions of seed points as p_1, p_2, p_3,..., p_n respectively.
2. Compute the difference of pixel value of initial seed point p_i and its neighboring points , if the difference is smaller than the threshold criterion, we define, the neighboring point could be classified into C_i, where i = 1,2,3,...,n.
3. Recompute the boundary of C_i and set those boundary points as new seed

points p_i (s). In addition to this, the mean pixel values of C_i have to be recomputed, respectively.

4. Repeat Step 2 and 3 until all pixels in image have been allocated to a suitable cluster

The selection criterion or the threshold criterion is given by the user and it is usually based on intensity, gray level, or color values in case of an image. The regions are chosen to be as uniform as possible. The regions formed using seeded region growing have high color similarity and no fragmentary problem. However, it still has two drawbacks: initial seed-points and time-consumption. The initial seed-points problem means the different sets of initial seed points cause different segmentation results. This problem reduces the stability of segmentation results obtained from the same input image. Additionally, initial number of seed points required are different for different images. However, the requirement of high computation is the most significant limiting factor on the use of seeded region growing.

2.1.2 Unseeded Region Growing

As the name suggests, unseeded region growing does not need explicit seed selection and the algorithm may generate seeds autonomously or may not require seeds. The steps are as follows:

1. The process initializes with cluster C_1 containing a single image pixel, and the running state of the process composes of a set of identified clusters, c_1, C_2, ..., C_n.

2. The set of all unsigned pixels which border at least one of those clusters is then defined as:

$$S = \left\{ x \notin \bigcup_{i=1}^{n} C_i \Lambda \exists k : N(x) \bigcap C_k \neq \phi \right\}, \tag{2.1}$$

where N(x) are current neighboring pixels of point x. Moreover, let δ be a difference measure

$$\delta(x, C_i) = \left| g(x) - \underset{y=c_i}{mean}[g(y)] \right|, \tag{2.2}$$

where g(x) denotes the pixel value of point x, and i is an index of the cluster such that N(x) intersect C_i.

3. To chose a point $z = S$ and cluster C_j where $j \in [1, n]$ such that

$$\delta(z, C_j) = \underset{x \in S, k \in [1,n]}{min} \{\delta(x, C_k))\} \tag{2.3}$$

If $\delta(z, C_j)$ is less than the predefined threshold t, the pixel is clustered to C_j. Else, we must select the most considerable similar cluster C such that

$$C = \underset{C_k}{argmin} \{\delta(z, C_k))\} \tag{2.4}$$

If $\delta(z, C) < t$, then we can allocate the pixel to C. If neither of two conditions conform, it is obvious that the pixel is substantially different from all the clusters found so far and so a new cluster C_{n+1} would be generated and initialized with point z.

4. After the pixel has been allocated to the cluster, the mean pixel value of the cluster is updated.

5. Iterate Step 2 and 4 until all pixels have been assigned to a cluster.

[11]. This method is very similar to part A of our algorithm. This consists of rapidly scanning the image from the top left corner to the bottom right corner, based on an assigned threshold, to form the original set of clusters. There is no upper limit on the number of clusters that can be formed during this stage.

2.1.3 Region Merging and Splitting

Distinguishing the homogeneity of the image is the primary goal of region splitting and merging. Its concept is based on quad trees, which means each node of trees has four descendant's and the root of the tree corresponds to the entire image. Besides, each node represents the subdivision of a node into four descendant nodes. The instance is shown in Figure 2.2, only R_4 was subdivided further. The basics of splitting and merging are discussed below. [12]

Let R represent the entire image regions and decide a predicate P. The purpose is that if $P(R) = False$, we divide the image R into quadrants. If P is $FALSE$ for any quadrant, we subdivide that quadrant into sub-quadrants, and so on. Until then, for any region R_i, $P(R_i) = True$. After the process of splitting , merging process is to merge two adjacent regions R_j and R_k if $P(R_j \bigcup R_k) = True$. The summarized procedure is described as follows:

- Splitting Steps: For any region R_i, which $P(R_i) = FALSE$, we split into four disjoint quadrants.
- Merging Steps: When no further splitting is possible, merge any adjacent regions R_j and R_k for which $P(R_j \bigcup R_k) = TRUE$.
- Stop only if no further merging is possible.

The advantages and disadvantages of region splitting and merging:
Advantages:

(a) The image could be split progressively according to the desired resolution, as the number of splitting levels is determined by us.

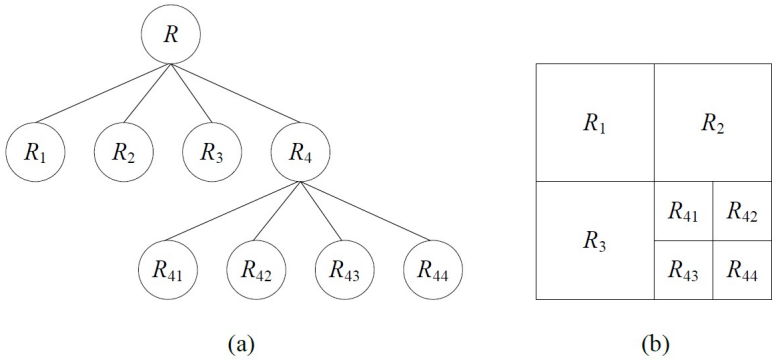


Figure 2.2: (a) The structure of quad-tree, where R represents the entire image regions. (b) Corresponding partitioned image

(b) We could split the image using the criteria we decide, such as mean or variance of segment pixel value. In addition, their merging criteria could be different to the splitting criteria.

Disadvantages:

(a) It may produce the blocky segments.

The blocky segmentation problem could be reduced by splitting in higher level, with the trade-off being higher computation time.

2.2 Data Clustering

Clustering is a process of dividing a dataset into mutually exclusive groups such that the members of each group are as ”close” as possible to one another, and different groups are as ”far” as possible from one another, where the distance is measured with respect to all available variables.

Clustering can be of different types, i.e. Hierarchical vs. Partitional, Exclusive vs. Overlapping vs. Fuzzy, and Complete vs. Partial.

(a) Hierarchical vs. Partitional: partitional clustering refers to a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset. Hierarchical clusters, on the other hand, are a set of nested clusters that are organized as a tree where each node(cluster) in the tree (except the leaf nodes) is the union of its children (subscribers), and the root of the tree is the cluster containing all the objects.

(b) Exclusive vs. Overlapping vs. Fuzzy: Each data object in exclusive clustering is assigned to a single cluster. In a case of overlapping clustering, an object can simultaneously belong to more than one cluster. In fuzzy clustering, every object belongs to every cluster with a membership weight that is between 0 and 1.

(c) Complete vs. Partial: A complete clustering assigns every object to a cluster, whereas a partial clustering does not. The motivation to do so in partial clustering is that some objects in a dataset may not belong to well-defined groups.

Depending on the type of clustering, there are different types of clusters:

(a) Well-Separated: In these clusters, the data objects are much closer to every other data object within the same cluster and than to any object not in the cluster. These idealistic clusters are possible only when the data contains natural clusters that are quite far from each other.

(b) Prototype-based: A cluster in which each object is closer to the prototype that defines the cluster than to the prototype of any other cluster.

(c) Graph-based: Graph-based clusters are when the data is represented as a graph, where the nodes are objects and the links represent the connections among the objects. These are also called as connected components.

(d) Density-based: A cluster in this case is a dense region on objects that is surrounded by a region of low density. A density-based type of clustering is used when the data is irregular and intertwined, and when noise and outliers are present.

In the current context of this work, we briefly discuss k-means and DBSCAN algorithms:

2.2.1 K-means:

K-means is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points, and is typically applied to objects in a continuous n-dimensional space. The basic k-means algorithm is simple and involves choosing the k-initial centroids where k is the user-specified parameter, namely, the number of clusters desired. Each data point is then assigned to its closest centroid, and each collection of points is assigned to a centroid is a cluster.[13] The centroid of each cluster is then updated based on the points assigned to the cluster. The assignment is repeated by updating the steps until the centroids remain the same. The basic K-means algorithm can be formally described as follows:

1. Select K points as initial centroids
2. repeat 1
3. Form k clusters by assigning each point to its closest centroid
4. Recompute the centroid of each cluster

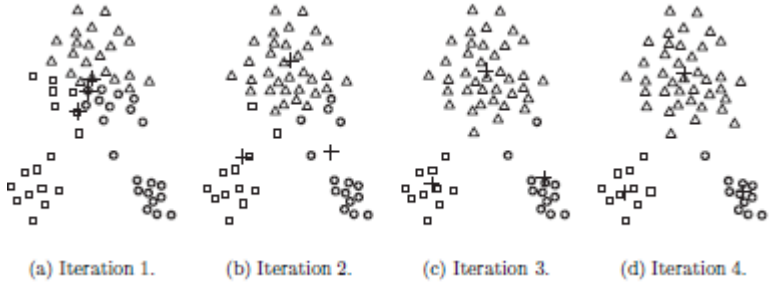


Figure 2.3: Using the k-means algorithm to find three clusters in sample data

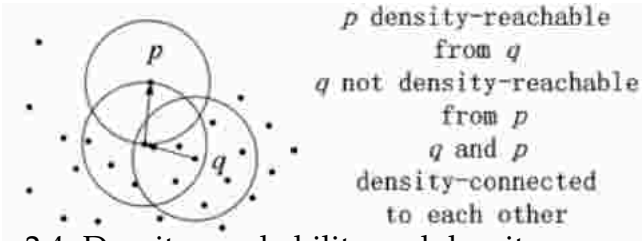


Figure 2.4: Density-reachability and density-connectivity

5. until Centroids do not change

2.2.2 DBSCAN

The basic idea of density-based clustering involves a several new definitions. The ideas are described intuitively here with the introduction of the main idea of the algorithm: Given a spatial dataset and objects in it distributed in a two-dimensional space:

- The neighborhood within a radius of a given object is called the Eps of the object
- If the Eps of the object contains at least a minimum number, MinPts, of objects with similar property, then it is called as a core object.

- For the set of objects, I, we say that object p is directly density-reachable from object q if p is within the Eps of q, and q is a core pixel.
- Density reachability is the transitive closure of direct density reachability and this relationship is asymmetric. This asymmetric density reachability is density connectivity. Refer Fig. 2.4

DBSCAN searches for clusters by checking the Eps of each point in the dataset. If the Eps of a point p contains more than $MinPts$, a new cluster with p as the core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merging of a few density-reachable clusters. The process terminates when no new point can be added to any cluster.

In the current context, the image can be considered to be a special spatial dataset in which each pixel has a spatial location and a color value. The method used to discover clusters in a spatial domain will be effective in discovering clusters in an image. Pixels which are similar in color and connective in spatial can be clustered together to form a segmented region. Thus, in pixel clustering, image pixels distribute according to the feature space such as color in addition to the spatial space. This results in pixels that are grouped in one cluster to be connective in spatial as well similar in color value. [14]

2.3 RGB images - a special case of multispectral images

The literature on color image segmentation is not as extensively present as that on monochrome image segmentation.There are two critical issues for color im-

age segmentation:

1. Choice of an appropriate segmentation method, and
2. Choice of color space

At present, color image segmentation methods are generally extended from monochrome segmentation approaches. Pal et al. [15] mention that color image segmentation can be considered to be a special case of multi-spectral images and any segmentation method for multi-spectral images can be applied to color images. According to Riseman et al. [16], when the color is projected onto three components, there is scattering of the color components, so that the color image is an example of a multispectral image. Choosing the color representation is another challenge. Each color-space has its own advantages and disadvantages. From literature, there is no single color-space that surpasses all others for better color image segmentation results.

In most of the color image segmentation approaches, the definition of a region is based on the similarity of color. This makes it difficult for algorithms to separate the objects with highlights, shadows, shadings or texture which lead to inhomogeneities of colors of the objects' surface. Color segmentation could also be viewed as an image classification problem based on color and spatial features. RGB color-space is an interesting one as all three channels in this color space are made of both, color and luminance and hence, all three channels have significant influence over the results of segmentation. Different color-spaces have been explored in the past and no one method has helped to solve the problem in a better way than the other. However, using a combination of several existing supervised and unsupervised segmentation approaches like maximum

likelihood, decision tree, K nearest neighbors, neural networks, etc. and adaptive thresholding, fuzzy c-means, split & merge, etc. respectively, improvements have been observed in the number of correctly identified regions [17]

2.4 State of the art algorithm

In this section, work relevant to the current problem statement has been reviewed.

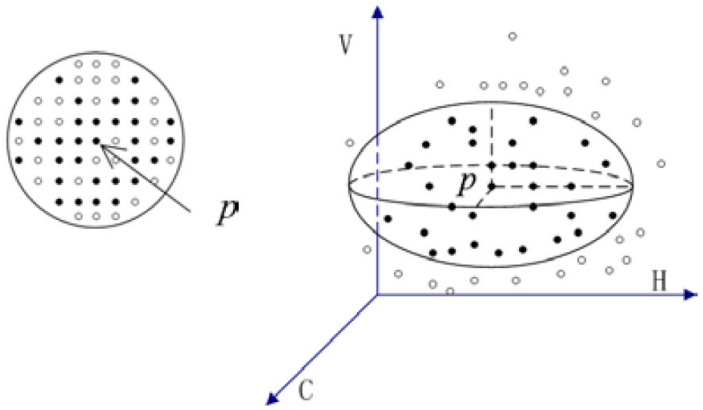
Lézoray et al.[18]: A color image segmentation method considering pairwise color projections. Each pairwise projection is analyzed according to an unsupervised morphological clustering which looks for the dominant colors of a 2D histogram. This leads to obtaining three segmentation maps combined by superposition after being simplified. Oversegmentation occurs as a result of superposition. As a result, pairwise region merging is performed according to a similarity criterion and is bound by a termination criterion. As a result, this work achieves one of the best compromises between precision and complexity. The boundaries of the objects in segmented images are visually better delineated and according to the authors, the mean squared error (MSE) between the color median image of the segmentation and the original is far better than the existing methods. This algorithm is more robust to impulse noise but does not respond well to Gaussian noise, as the energy increases assessing an increase in noise and segmentation complexity.

Hamarneh and Li[19] have utilized the prior shape and appearance knowledge to enhance the watershed algorithm. An improved version of kmeans algorithm is applied onto the watershed segments and the results of this im-

plementation is iteratively aligned to a shape histogram. This is a machine learning algorithm and consists of a training and segmentation stages. In the training stage, the shape and appearance knowledge is modeled using a shape histogram and image intensity statistics. The segmentation stage is made of four stages and is an iterative procedure. These four stages are generating the classical watershed segments, implementing an improved version of k-means algorithms on these segments, aligning the shapes and refining them. Some of the intrinsic problems associated with watershed segmentation are addressed by this algorithm. Since mean intensity for each segment is computed, the effect of noise is suppressed. However, this algorithm suffers from some limitations of the k-means algorithm like predicting the ideal 'k' for a given dataset. Also, the algorithm has been tested only on 2-dimensional data.

Dutta et al.[20] applied the fuzzy c-means algorithm to images to obtain the cluster centers. Each pixel was then evaluated by each rule of fuzzy c-means and the results were plotted on a histogram. A peak detection and valley extraction (PDVE) algorithm was applied to the histogram and thresholds were extracted from the histogram for segmentation. The fuzzy c-means is an unsupervised clustering method and hence no prior information on the number of regions was required for the segmentation. The authors claim that this algorithm could be useful for feature extraction where extraction of regions and segments is required for a large set of images.

Banerjee et. al[21] used a minimum spanning tree approach to propose an unsupervised color image segmentation method. Natural groupings of the image pixels were done to find clusters of pixels having RGB values within a certain range. Then the pixels nearest to the centers of these clusters were marked



(a) SpatialEps (b) ColorEps

Figure 2.5: Spatial Eps and Color Eps of a pixel p

as seeds and were then used for region growing based image segmentation. Following this, a region merging based segmentation was performed to eliminate the effect of oversegmentation. The elimination of possible oversegmentation if that persisted in the algorithm is clearly one of the advantages of this algorithm. Additional scanning of the source image is required to perform region merging, and this comes at a cost of computational time.

Ye et al.[22] proposes a region-based image segmentation method to divide an image into different regions based on their color similarity and spatial relation. The authors have used density based clustering over region growing to integrate spatial connectivity. HVC color space has been used for the segmentation. Two threshold criteria: one for spatial connectivity and second for color similarity were used with the DBSCAN approach. The authors have used the spatial eps and color eps values to determine the number of seeds using an unsupervised approach, but this comes at a cost of having to identify three parameters, all of which depend on the nature of the image and the type of the data and are reported to be very sensitive.

The authors also discuss their attempts in parameter estimation. Spatial eps is a heuristic based assignment and requires understanding of the nature of the image and the type of data. Through empirical observations, the minpts value was set to half of the value of the spatial eps. In order to predict the color eps value, the authors performed histogram concavity analysis in the HVC color space. The HVC color space was chosen because it is perceptually uniform. Thus, two color pixels, if equal in distance in the color space would be viewed as equal in distance by the viewer in the real case scenario as well. Therefore, this ensures that the color change in the segmented regions is uniform. This algorithm, however, does not work well with images having texture.

Chunming Li et al.[23] proposed a novel region-based image segmentation method by dealing with intensity inhomogeneities in the segmentation. The local intensity clustering property of the image intensities was derived first based on the model of images with intensity inhomogeneities, and a local clustering function was then defined for the image intensities in a neighborhood of each point. Following this, the local clustering criterion function was then integrated with respect to the neighborhood center to give a global criterion of image segmentation. In a level set formulation, this criterion defines an energy in terms of the level set functions that represent a partition of the image domain and a bias field that accounts for the intensity inhomogeneity of the image. The level set method provided an advantage that, numerical computations involving curves and surfaces could be performed on a fixed Cartesian grid without having to parameterize these objects. This method was more robust to initialization, faster and more accurate than the well known piecewise smooth model. The marked disadvantage of the level set method is its usage being limited to intensity inhomogeneities.

Fan et al.[24] used an edge detection scheme by combining an isotropic edge detector and a fast entropic thresholding technique to obtain the color edges in an image. These color edges provided the major geometric structures, which were used to calculate the centroids between the adjacent edge regions, and these were taken as the initial seeds for seeded region growing (SRG). The seeds were then replaced by the centroids of the generated homogeneous image regions by incorporating the required additional pixels step by step. Homogeneous image regions with accurate and closed boundaries were obtained by integrating the extracted color edges and SRG. Following this, a growth procedure was utilized to spatially integrate pixels in a recursive fashion, to an appropriately chosen seed from the entire set until the final segmentation was achieved.

Contrast information was used by Chen et al.[25] to develop a color image segmentation approach. The input color image was converted from RGB to Lab color space and the Lab color information was used to estimate the contrast information with four directional operators. A set of initial regions were then identified from this contrast map and were then merged using a connection and verification process to form 2-D boundaries.

Qin et al.[26] used semantic information for effective region growing, and proposed an MRF-based (Markov Random Field) multivariate image segmentation algorithm. The MRF spatial context model and adaptive edge penalty were combined and applied to regions. This helped to reduce the impact of interclass variation and computational cost. Beginning with watershed oversegmentation, semantic region growing was performed alternately with segmentation and this helped to reduce the solution space size, which improved

the segmentation effectiveness.

Shih et al.[27] developed a segmentation algorithm based on seeded region growing and merging, incorporating strategies to avoid pixel order dependencies. In this algorithm, fixed threshold values can produce reasonably good results but not for all the images and the time complexity is high.

The JSEG algorithm developed by Deng et al.[7] integrated color quantization and spatial segmentation for extraction of color-texture regions in images and video. This method was capable in deriving spatially compact regions. Initially, a 'color class map' was obtained using a color quantization step, which is then used compute a J-image corresponding to the measurements of local inhomogeneities that acquired high values at region boundaries and low values for homogeneous color-texture regions. Subsequently, the J-image is utilized as a reference to identify suitable seed point to initiate a region growing process, wherein the obtained regions are eventually refined in the merging process using a color similarity criterion. Although the JSEG method was efficient in deriving spatially compact regions, it suffered from the fact that the use of color quantization caused over segmentation in regions of varying shades due to illumination or texture disparities.

Ugarriza at al.[28] used the information obtained from detecting edges in color images and used a color gradient detection technique to propose a new unsupervised color image segmentation algorithm. The pixels without edges were clustered and labelled individually to identify some initial portion of the input image content. Elements that contained higher gradient densities were included by the dynamic generation of the clusters as the algorithm progressed. Quantization followed by local entropy computation of this quantized image

was used for texture modeling. A multiresolution merging procedure was used to blend regions with similar characteristics by using the region growth map consisting of all fully grown regions along with the obtained texture and color information. A high level of detail was achieved in the segmentation of images with higher complexity using this algorithm. The problem of varying illumination is addressed by using an initial clustering technique. The region merging procedure helps to efficiently handle the background occlusion problem. However, the algorithm has high computational complexity and hence is slow.

Ali et al.[29] proposed an improved version of the fuzzy c-means clustering (FCM) by merging the segmented results generated by fuzzy clustering based on pixel intensity and location of pixels. Initially, the image was segmented into regions using the fuzzy clustering algorithm based on individually considering the pixel intensities and pixel locations. The overlapping regions between the two merged regions were determined and if the ratio between the number of overlapping and the respective region pixels was greater than 0.5, the overlapping region was removed from the merged region which possessed the minimum ratio. Otherwise, the overlapping pixels were redistributed between the two regions using fuzzy c-means considering only the pixel locations. This algorithm called, image segmentation using fuzzy clustering incorporating spatial information (FCSI), is independent of the actual clustering algorithm. It uses the FCM algorithm to obtain the results based on pixel location or pixel intensity or the combination of the two and is thus, and enhanced version of the FCM algorithm for image segmentation. However, the number of clusters desired in the final result are to be provided as an input to the algorithm.

Ding et al. proposed the Fast-Scan algorithm[30] to achieve high speed,

good reliability and avoid fragmentary results. The concept involves scanning through the entire image from the upper left corner to the lower right corner and determine if we can merge the pixels into an existing clustering. The merging criterion is based on the assigned threshold. If the difference between the pixel value and the adjacent cluster is less than the threshold, then the pixel is merged into the cluster. Each pixel in the image was scanned only once. The unique advantages of this work are: 1. The pixels of each cluster are spatially connected and have a similar pixel intensity value, i.e. good shape connectivity; 2. The computation efficiency is much better than the region growing as well as region growing and splitting algorithms; 3. The results have good shape matching, i.e. segmentation results match exactly with the shape of real objects. The authors also propose the concept of adaptive threshold wherein, the threshold value would be adjusted for different parts of the image based on the local frequency and variation.

Nock et al.[31] have proposed a Statistical Region Merging (SRM) algorithm that establishes a statistical basis for image segmentation by region merging. They follow a particular order in the choice of regions such that the segmentation error is limited from both, qualitative and quantitative standpoint. Image segmentation is formulated as an inference problem. It is reconstruction of regions on which the true regions that are seeked are statistical regions whose borders are defined from a simple axiom. The idea is to start with 1 region per pixel and then applying a statistical test on neighboring regions (in ascending order), whether the mean intensities are sufficiently similar enough to be merged. A merging predicate and ordering of these merges is designed for pairwise groups of adjacent pixels. Based on the differences between these pairs, boundaries are determined. SRM algorithm is a fast and robust algorithm to segment an image

into desired number of regions of similar intensity such as color. However, a disadvantage of this method is overmerging, i.e. the fact that some observed regions may contain more than one true regions. However, the authors claim that this overmerging error is small.

In their paper of efficient graph-based segmentation, Felzenszwalb et al.[32] have presented an efficient graph based method to perform image segmentation using Minimum Spanning Trees (MST) and is widely used as it runs video rate in practice. The authors assume that edges between vertices in the same region should have relatively low weights than edges between vertices of different segments. Initially, each vertex is considered as a segment. Then, in a greedy way two segments are repeatedly selected to consider for merging. Based on a comparison predicate, the decision of merging two segments is made. This comparison predicate is determined based on the internal difference between a segment and the difference between two segments. The internal difference of a segment is the largest weight in the minimum spanning tree of the segment, which is given by equation 2.5.

$$int(s) = max_{e=MST(S,E)}(w_e) \tag{2.5}$$

The difference between two segments is calculated as the minimum weight edge connecting the two segments. When the difference between the two segments is less than or equal to the minimum of any of the internal difference of the two segments, then the predicate allows the two segments to be merged. The authors show that the segmentation produced by this method is neither too coarse not too fine. Also, any attempt to change the definition of the difference between the two segments (e.g. median edge weight instead of minimum edge weight) to make the method more robust, led to the solution becoming NP-hard. As the method merges, two segments based on a single low weight edge between

them, there are possibilities that the results could considerably be affected by noise if no initial filtering of the image is done.

CHAPTER 3

ALGORITHM DEVELOPMENT

Given a spatially resolved dataset, a heuristic threshold and a desired 'k' number of clusters, our goal is to:

- Divide into spatially connected 'contiguous' clusters based on the assigned attribute threshold (eps)
- Intelligently merge the clusters to have exact k contiguous clusters while minimizing the variance i.e. maintain shape relevance
- Scale to extremely large datasets: of the order of 10^{12} data points (1 trillion)

To address our problem statement, we have developed a hybrid contextual image segmentation algorithm using concepts in region growing, region merging and splitting and density-based clustering. Based on the requirements, we have divided the problem statement into two algorithms as follows:

(A) Given the criterion threshold, we divide the image into contiguous regions according to this threshold.

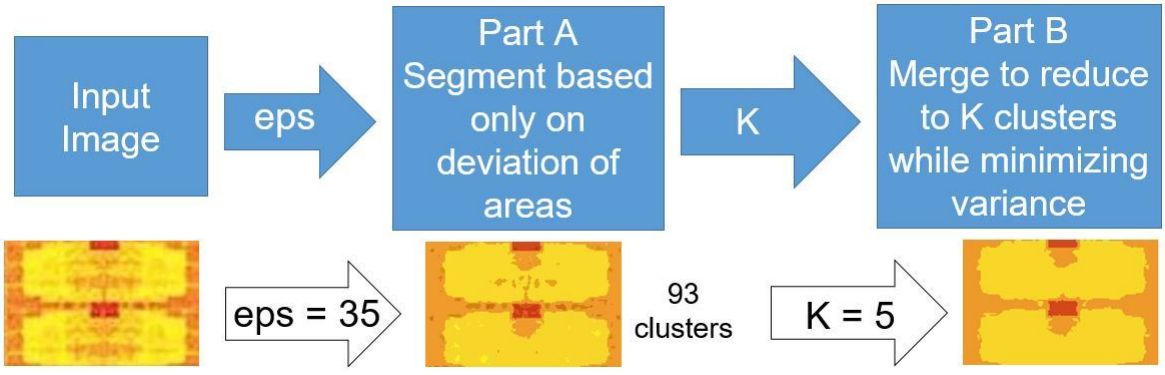


Figure 3.1: Algorithm Schematic

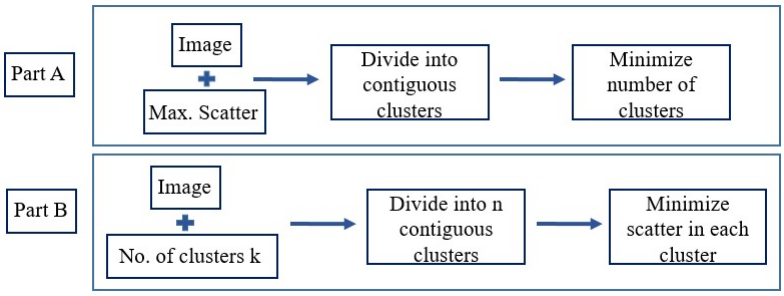


Figure 3.2: Problem Statement

(B) Given the number of regions required, we divide the image generated in part A into exact number of contiguous regions such that the standard deviation for each segment is minimized.

One of the most important constraints are the requirement of contiguity among the segments, i.e. the pixels in any given segment need to be spatially connected with each other. In addition to this, the algorithm needs to be capable of handling large images, and thus, computationally efficient. Initially we used grayscale images for the algorithm and later implemented RGB colors as a special case of multi-spectral image data. The details of the algorithm are as follows:

3.1 Initial Partitioning: Given a criterion threshold, segment the image into contiguous regions such that every segment conforms to this threshold

Initially, we started off with a a region growing approach and used the 8-corner neighborhood search for every pixel. Although 8-neighborhood search for ev-

ery pixel would be computationally more expensive than a 4-neighborhood search, we decide to use 8-neighborhood approach for a better accuracy of the final results. The Manhattan distance measure was used to measure the intensity difference between the pixels being compared. Breadth First Search approach was used to implement the region growing method. This allowed us to mark a pixel as visited once it was assigned to a cluster which avoided visiting/examining assigned pixels again. The implementation of queues, with adding unvisited pixels to the queue and removing them once assigned helps to save on memory.

This approach is similar to the unseeded region growing algorithm Fast-Scan[30]. The purpose of using this approach is the requirement to achieve high speed while ensuring good reliability and avoiding fragmentary results. This approach fulfills these requirements. The algorithm needs to be scalable to large scale data. Thus, efficiency is important while maintaining the accuracy of the results. As compared to seeded region growing, unseeded region growing is almost 20x faster and thus is, more aligned to the requirement of scalability. The color threshold, i.e. 'eps' value plays an important role in deciding the quality of segmentation. If the eps value is very small, it results in a very tight tolerance, leading to too many clusters being formed. This might pose a challenge while achieving the desired number of clusters during the merging stage. On the other hand, if the eps value is very large, it is still unfavorable as it could lead to few segments being formed and thus, not the best clustering result. The eps value, thus requires an understanding of the type or nature of the input data and is thus, a heuristic value at present. Other measures taken to improve the performance and runtime complexity are discussed as a separate section later in this chapter.

The algorithm was initially tested on a synthetic dataset that we created. This dataset contains images of increasing size from 4 pixel, 8 pixels, 16 pixels all the way upto 100 pixels. Having a synthetic dataset helped us to compare the results of the implementation to the expected outcomes and thereby get the dry run results of the algorithm and inspect the performance as the complexity of the image size and data increased.

100	100	200	200
100	111	214	200
114	101	205	200
121	105	204	203

(a) Input

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

(b) Expected Outcome

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

(c) Result

(d) Sythetic Image: Input (e) Sythetic Image: Result

Figure 3.3: Results of Part A of the algorithm on 16 pixel synthetic dataset

3.2 Merging to k clusters: Given a fixed number of desired regions k, segment the image into k contiguous regions such that the standard deviation for every segment is minimized

The clusters generated in part A are ranked according to their sizes. The first k largest clusters are retained all the other pixels that do not belong to the first k clusters are marked as unvisited. Then each of the unvisited pixels are visited with a region growing approach. In the 8-neighborhood of the target pixel, if it has at-least 3 pixels belonging to the assigned k clusters, their intensity difference with the target pixel is calculated and the target pixel is assigned to the closest target pixel. Once assigned, it is marked visited and the process is repeated until there are no unvisited pixels.

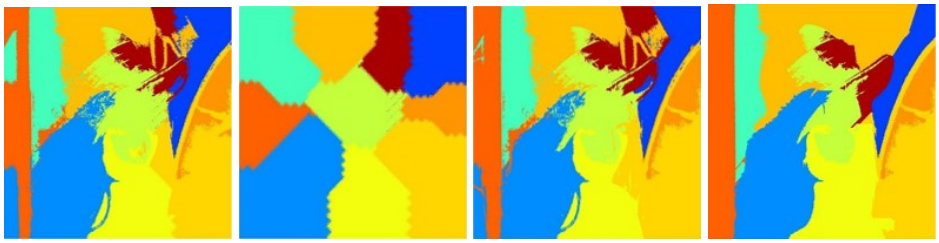
3.2.1 Different Approaches taken in arriving at solution for Part B

Several alternatives were explored to arrive at the above mentioned method for Part B of the algorithm. Part A of the algorithm was initially executed on the lena image to generate the clusters before the region merging step could be applied. As seen in figure 3.4(b), part A generated 1704 clusters with the color threshold set to 25. The experiments for region merging i.e. part B of the algorithm were implemented on this image with the required number of clusters (k) set to 10. The results with each of these methods are explained below. While going through these different approaches, we realized that using a 4/8 neigh-

borhood approach of region growing would ensure the contiguity or the spatial connectedness of a given cluster.

(a) Input (b) Part A

Figure 3.4: Results after executing part A of the algorithm on lena image with 1704 contiguous clusters and eps=25



(a) clusters = 548, time = 0.93s (b) clusters = 711, time = 1.36s (c) clusters = 356, time = 9.01s (d) clusters = 10, time = 1.16s

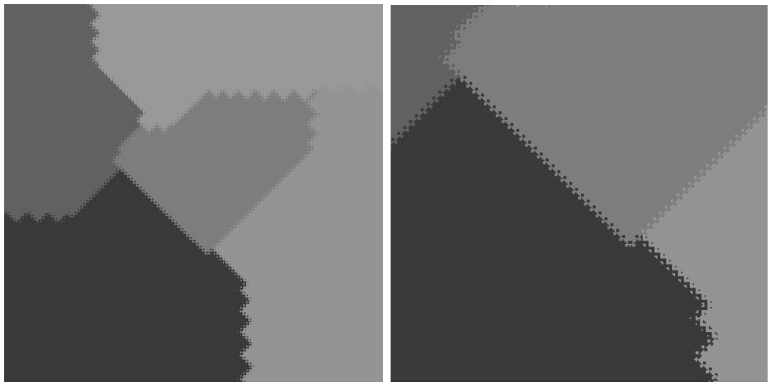
(e) clusters = 10, time = 1.49s (f) clusters = 10, time = 1.22s (g) clusters = 10, time = 1.77s

Figure 3.5: Different approaches taken while exploring part B of the algorithm

(a) Having calculated k centroids, pixels that did not belong to k clusters were marked unvisited. The distance for each pixel from the k centroids was measured. These pixels were assigned to the clusters of closest centroids: The results generated much more clusters than the required number as cal-

culating distances directly from the centroids of the first k clusters did not guarantee.

(b) Having calculated k centroids, all points were marked unvisited. The distance for each pixel from the k centroids was measured and pixels were assigned to the clusters of closest centroids (Voronoi Segmentation): The results were much worse than the previous method as only the spatial relation was used to assign clusters without, while forgetting the pixels that belonged to the clusters of the first k cluster by virtue of their intensity. The total number of clusters created was 711 which was significantly more than those in the previous approach. This was because, in the previous example, the pixels of the contiguous clusters generated by previous approach were retained with their original clusters which maintained their contiguity. The time required in this approach was almost 1.5 times that of the previous case as every pixel had to be visited to calculate the distance as against only the unvisited pixels in the previous case.



(a) Voronoi Segmentation Results

(b) Zoomed In portion showing non-contiguous clusters

Figure 3.6: Results of Voronoi segmentation on Lena image and evidence of non-contiguous clusters

(c) Centroids of the k largest clusters were identified. For every non-k centroid,

its distance from each of the k centroids were calculated and it was assigned to the cluster of the closest centroid and thereby, all pixels of this respective cluster were assigned to the cluster of the closest centroid. Results show that the calculation of centroid for each of the clusters consumed atleast 10 times more CPU power. The total number of contiguous clusters formed were still much more than the required value although this approach did show some improvement over the previous two methods. Calculating centroids for every cluster is a time consuming process and with this approach, we would be going against our objective of achieving an efficient algorithm. Once again, the metric of pixel intensity was getting compromised during the merging step which would result in bad shape connectivity or irrelevant clustering.

(d) (Majority Voting) The first k clusters were retained and all remaining pixels ere marked as unvisited. Thereafter, every unvisited pixel is accessed using the Breadth First Search Region Growing approach. For every non-visited pixel, its 8 corner neighborhood was examined to see if there were any of the pixels belonging to the first k clusters. If yes, majority voting was used for the assignment. The results generated exact k contiguous clusters, but the concept of majority voting could lead to faulty assignment. For example, consider a pixel surrounded by 5 pixels of one of the first k clusters and 3 pixels of another cluster belonging to the first k clusters. In such a case, this pixel would get assigned to the cluster index as that of the 5 neighbors even though it may be close in its color value to the cluster index of the 3 neighbors. Since, it is a region growing approach, either of the assignment would guarantee spatial connectivity, but the majority voting approach would lead to wrong assignment which in turn would increase the standard deviation

rather than minimizing it.

(e) From the learnings in the previous approach, some steps were adopted, except, at the assignment stage, the color difference of the target pixel with the cluster centroids of each of the relevant neighbors was measured. The assignment was done based on the closest color similarity. The results, once again, showed that exact k clusters were formed. The color assignment was visually similar to the previous method. One can notice that the arch behind lena capturing wring pixels in the cluster in the right top corner of the image which was not expected. In the assignment stage, the color assignment would happen even though there would have been just one relevant neighbors, which involved no comparison but just assignment. The time required was slightly more than the previous method but guaranteed better relevance in cluster assignment.

(f) We retained the first k clusters (and their pixels), marked all the remaining pixels as unvisited, and used a Breadth First Search Approach to visit every unvisited pixel. If in the 8 neighborhood of target pixel, there were atleast 1 or more assigned pixels, we compared the intensity value of target pixel with each of these relevant neighbors and assigned them to the cluster to which they closely matched: We realized that comparing the target pixel intensity with cluster centroid intensity of the relevant neighbors was not as accurate as comparing the target pixel intensity with the intensity of the local relevant neighbors themselves. The time required was reduced slightly as the centroids were not required to be accessed all the time for comparison. However, the clustering error still persisted and we believe, it was arising due to faulty assignment when only when a target pixel would have just one relevant neighbor (belonging to one of the first k clusters) and would

get assigned to it.

(g) We tried to address the issue of the possibility of wrong assignment when just one of the eight neighbors of the target pixel belonged to one of the k clusters. We realized that we needed to have some ordering or a structured approach in the way we were accessing the non-visited pixels. We decided to first address those pixels which had atleast 3 neighbors belonging to one of the k clusters. This would definitely cost us more computation time, but atleast the accuracy of the clustering would not be compromised which was more critical. The results show that the arch behind lena discussed above being clustered correctly and as expected, the CPU time required was 150% of the previous result.

3.2.2 Adopted method for merging clusters

We adopted the last approach discussed in **??**.g above as our final solution, as, it guaranteed exact k clusters, with accurate clustering maintaining good shape connectivity, although, the computation time was compromised to some extent. We tested our algorithm on a synthetic dataset that we had created. We created images of increasing size of 4, 8, 16 and 100 pixels. This synthetic dataset allowed us to examine the results and thereby test the implementation of Part B of our algorithm to the expected outcomes and thus have a dry run of the algorithm. It also allowed us to see how the algorithm performed as the image size and complexity increased.

70	1	150	160
2	5	159	165
40	9	221	219
52	7	230	225

(a) Input

70	2	150	150
2	2	150	150
40	2	223	223
40	2	223	223

(b) Expected Outcome

70	2	150	150
2	2	150	150
40	2	223	223
40	2	223	223

(c) Result

Figure 3.7: Results of Part B of the algorithm on 16 pixel synthetic dataset

100	131	70	70	70	70	50	71	74	76
121	140	70	70	71	51	69	81	71	71
151	152	40	10	8	11	12	19	2	70
154	154	42	1	2	9	8	2	1	72
156	157	51	5	2	1	6	4	5	115
159	160	53	2	3	4	7	140	130	121
151	152	54	1	6	7	5	150	131	121
170	170	211	42	42	42	42	121	120	124
200	211	212	42	42	42	42	126	128	129
222	212	216	250	250	250	250	131	127	124

(a) Input

122	122	70	70	70	70	70	70	70	70
122	122	70	70	70	70	70	70	70	70
122	122	1	9	9	9	9	9	9	70
122	122	1	9	9	9	9	9	9	9
160	160	1	9	9	9	9	9	9	132
160	160	1	9	9	9	9	132	132	132
160	160	1	9	9	9	9	132	132	132
160	160	213	1	1	1	1	132	132	132
213	213	213	1	1	1	1	132	132	132
213	213	213	213	213	213	213	1	132	132

(b) Result

Figure 3.8: Results of Part B of the algorithm on 100 pixel synthetic dataset

3.3 Use of mutlispectral data

To improve the accuracy of the results, one of the approaches taken was to improve and/or increase the quality and amount of input information. One of these efforts was to use multispectral information. We have demonstrated this capability by using RGB color images as input. RGB color space is good for display purposes but not for color image segmentation as it suffers from high cor-

relation. The RGB color-space can be treated as a special case of multi-spectral image data as it is unbiased and each of the individual channels represent the color information in their respective part of the color space. The type of applications we focus on could have several channels of information for every pixel with different relative weights assigned to each channel, and this closely aligns to the examples of using RGB color space.

3.3.1 Weghted Distance Measures

Another improvement we did in our approach was the use of Euclidean Distance measure or RGB color images by replacing Manhattan Distance Measure. Euclidean Distance measure comes out to be more precise than Manhattan Distance measure for RGB color-space and less precise than Vertical Cosine Angle Distance (VCAD) measure. However, VCAD measure is computationally very expensive in comparison to Euclidean Distance measure. Hence, the use of Euclidean Distance measure.

$$\text{Euclidean Distance} = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2) + (B_1 - B_2)^2} \tag{3.1}$$

$$\text{Manhattan Distance} = |grayvalue_1 - grayvalue_2| \tag{3.2}$$

3.4 Evaluation

3.4.1 Histograms

Histograms have been generated for each of the results to verify the number of clusters formed. Additionally, histogram data reveals which clusters have been retained, which ones got merged, and to which clusters they got merged to. This helps to reveal the effectiveness of a given result/method for the desired region growing/ merging approach.

3.4.2 Verify Contiguous Clusters

We have written a custom verification script to count the number of spatially homogeneous (contiguous) clusters in order to verify the results on the part B of the algorithm. Another purpose intended with this is to have a count of how many contiguous clusters were generated in part A of the algorithm and how many clusters were merged during the merging procedure.

3.4.3 Effect of starting point and order of accessing neighbors

Since we start scanning the image from the top left corner of the image to the bottom right corner of the image, the possibility of the starting point of the algorithm affecting the total number of clusters cannot be neglected. Hence, we implemented the algorithm from for different corners of the image and compared the results to see whether the starting point of the algorithm had any significant

impact on the clustering result or the number of clusters formed. Similarly, the order in which the eight-neighborhood pixels were accessed could also impact the clustering results. Hence, we experimented two possibilities where the pixels in the eight neighborhood were accessed in a clockwise and anti-clockwise fashion to generate their respective clustering results. One could also compare the CPU time required in each of these instances to see if there are any changes in the computation time required to create clusters. These experiments should help to determine the robustness of our algorithm.

3.4.4 Quantifying Segmentation "Quality"

Getting an SSE value for an entire image as a whole allowed us to compare the compactness of resultant segmentation - one of the measures to assess how good was the segmentation.

3.4.5 Comparison with other methods

Although this being a problem statement with unique requirements, the results were compared with other state of the art methods including graph-based segmentation, fuzzy c-means and k-means algorithms highlighting why the results with the current approach are more relevant to the desired problem statement.

3.5 Scalability to large scale data

To check the scalability to large size images, we tested the algorithm's performance against increasing the size of the input image. The image size was increased by replicating a semiconductor image in the horizontal and vertical dimensions. The size range was from 10000 pixel image to 200,000,000 pixel image.

From Figure 3.9, we can see that the algorithm processes data of the order of 200MN pixels in under 100 seconds when tested on a 2.6 Ghz Pentium 4 machine. This given an idea of the scalability of our algorithm to even larger datasets.

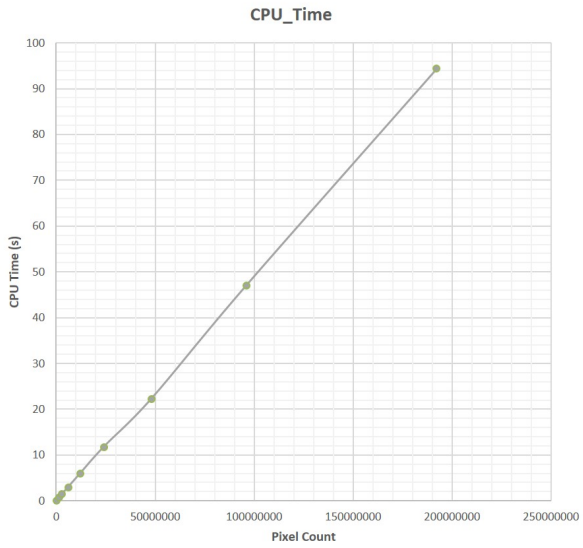


Figure 3.9: Scalability of the algorithm to large datasets

3.6 Performance Improvement and Runtime Complexity

Since we had to take the region merging approach of going through the 8-neighborhood region of a pixel, we decided to save our computation time by not going through the seed identification and generation route. This helped us greatly in improving our computation time. Given this approach, there could be a possibility of the starting point of the algorithm affecting the results. However we have shown in Section 4.3.3, that the choice of starting point does not produce any significant changes in the clustering results in our range of color threshold values.

We further decided to adopt the Breadth First Search (BFS) approach to access data and use concepts in Density Based Clustering (DBSCAN) for achieving our clustering goal. The theoretical advantages of BFS search are that, it is never trapped in exploring useless paths and if there are more than one solutions to a problem, BFS, by design finds the path that requires minimum number of steps. One of the the drawbacks of BFS is that it is a memory instensive algorithm and consumes memory proportional to the total number of nodes at each level to save this data in order o go to next level. However, we have tried to alleviate the further memory intensive operations as discussed below. Additionally, BFS, unlike DFS, is not recursive and hence ensures that we do not visit the same node and thereby he same pixels again.

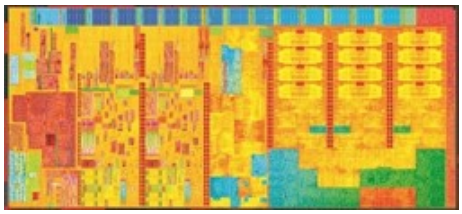
By incorporating the BFS search with DBSCAN approach and with a suitable indexing structure, we execute exactly one search query for each point. In our case we use an R* indexing structure as it allows us to store the pixel intensity and x, y co-ordinates together. A boolean visited array has been used with the

R* tree structure to avoid processing a node more than once. We avoided the use of graphs as they may contain cycles which may bring us to the same node more than once. Thus we ensure that we visit and process every pixel in our image only once in our algorithms. Thus an overall complexity of $O(n)$ was obtained, if the eps was chosen in a meaningful way.

Figure 3.10 shows the improvement in our cpu run-time over the period of development. Initially, we did not have the BFS approach as well as the DB-SCAN approach. Incorporating both of them helped us to exponentially improve the run-time results of our algorithm. Trimming the unnecessary operations and variables further helped us to make some more progress in our run-time.

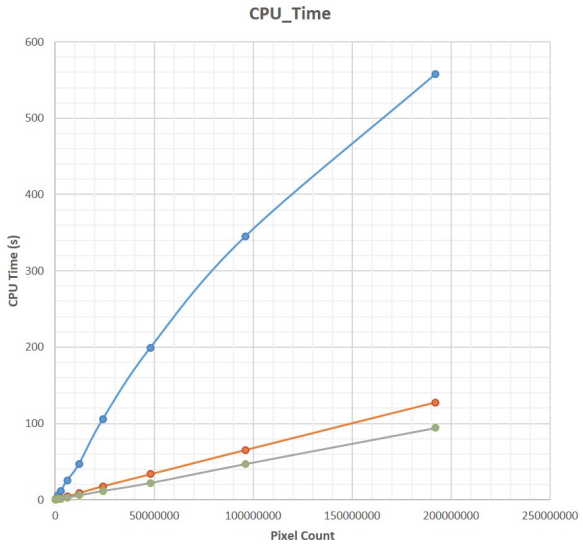
We had initially stored the data in the form of vectors. However, our usage with the data was to be sequential and hence, did not require random access. Thus we decided to use stacks/queues against vectors to hold the data for processing the clustering algorithm. We needed the data structure to be FIFO (First In First Out) for the data that was passed to it and hence adopted the usage of queues over stacks.

Initially our vectors were holding all image information at once, and whenever required the respective data was being accessed (random access). This contiguous structure of information was unnecessary as we could access linked data in a structured fashion. With queues, we were holding only the desired information of the current node as it was being processed, and it was cleared once its assignment was completed. This dynamic nature of the queues helped us to significantly improve our peak memory usage by more than 400% for a 200,000,000 pixel image from 13 GB to little over 3GB.

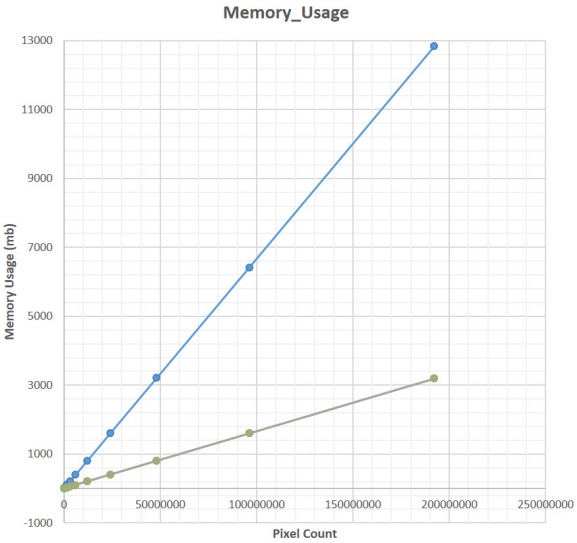


(a) Image

(b) Legend



(c) CPU Time Improvement



(d) Peak Memory Usage Improvement

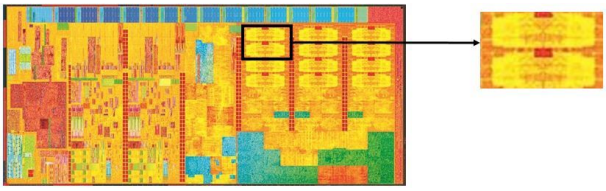
Figure 3.10: Improvement in CPU time and Peak Memory Usage

CHAPTER 4

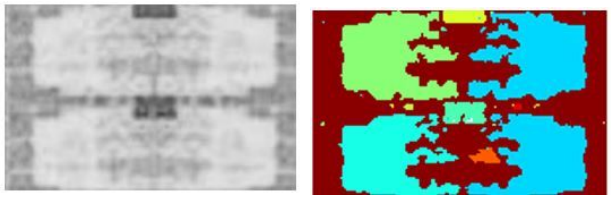
RESULTS, DISCUSSION AND COMPARISONS

4.1 Algorithm Part A

After getting expected results with the synthetic datasets (figure 3.3), the algorithm was implemented on an test-case from real dataset where the algorithm would be applied to. It was tested on one of the features of Intel's broadwell processor chip. At this point, the focus was to ensure the correct implementation and thus, not on the efficiency of the algorithm or its speed. We wanted to be sure that the shape connectivity was preserved and that the pixels within a given cluster were similar in color.



(a) Selected feature in a broadwell chip



(b) Input (c) Output

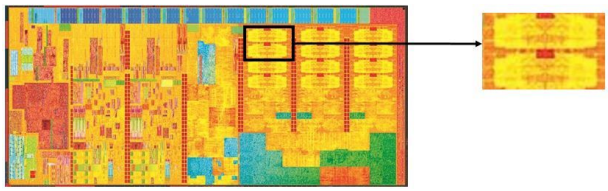
Figure 4.1: Results of Part A of the algorithm on a feature of broadwell processor (eps = 25)

One can clearly see the background being segmented as a single cluster and the four features clustered as separate clusters accurately along with the two square features segmented individually which are pretty accurate, visually to

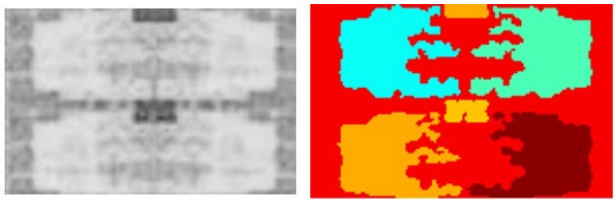
their original shape. Some small clusters are formed in the background where the color intensity changes are sharp, and the part B of the algorithm should merge them to the background when implemented.

4.2 Algorithm Part B

The implementation of the part B of our algorithm delivered the expected results and algorithm was tested on the same test case of the semiconductor image dataset that was used for testing part A of the algorithm. Once again, at this point, our focus was to examine and ensure the correct implementation and hence we chose a small test case which we could visually examine as well as compare the results to that of part A of the algorithm.



(a) Selected feature in a broadwell chip



(b) Input (c) Output

Figure 4.2: Results of Part A of the algorithm on a feature of broadwell processor (k = 7)

In figure 4.2, one can see that the 7 precise and spatially connected clusters formed (including the background) and all the main features of the semicon-

ductor image being preserved in the final segmentation result. These type of results help the device engineer to identify broadly if there are any defects in the lithography or whether any changes are needed to be made in the mask design. In addition to this, for applications such as population maps, a single spatially connected cluster can be represented by a prototype pixel and can help in the decision making process for which the population map may be generated. These decisions could range from assignment of patrol force based on crime rate, population examination based on several criteria, etc.

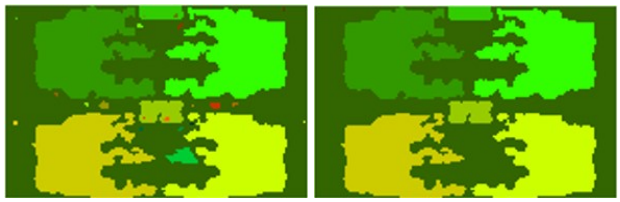
4.3 Evaluations

It was important to havecertain evaluation metric to examine the results of our algorithm mathematically as well as prove several aspects of the implementation. In addition to this, we also decided to include evaluation criteria to compare our results with other algorithms based on a quantitative metric. Following are the evaluation criteria we used and their results.

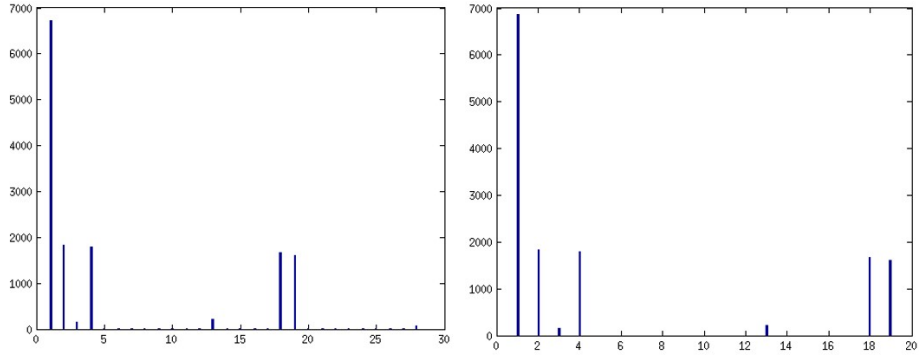
4.3.1 Histograms

As discussed in section **??**, histograms would help to verify the number of unique clusters formed as well help us observe which clusters were retained and which of the clusters were merged. Histograms are a good test of examining whether the part B of the algorithm was implemented correctly. For example, in Figure 4.3, one can verify the twenty eight clusters formed from the histogram seen in Figure 4.3(c) as well as seven unique clusters after implementing part B

as seen in Figure 4.3(d). From Figure 4.3(c), one would expect the clusters with indices 1, 2, 3, 4, 13, 18 and 19 to be retained and rest all merged according to part B's implementation. This aligns to results as they are seen in the subfigure 4.3(d).



(a) Eps = 35, # = 28 (b) Eps = 35, # = 7



(c) Histogram: Part A (d) Histogram: Part B

Figure 4.3: Histograms results with on the test cases from the semiconductor dataset

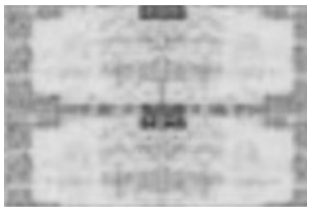
4.3.2 Span and Standard deviation per cluster

As mentioned in section **??**, standard deviation helps to examine whether the clustering of part A of the algorithm conformed to the assigned threshold. Calculating standard deviation after implementing part B of the algorithm also helps us to inspect and analyze whether the merging was relevant or not. If the standard deviation after implementing part B went over the assigned threshold, it would imply that the merged clusters were not close in color values and hence

there was an instance of irrelevant clustering. Noise in the image would not impact the standard deviation of the entire cluster as the number of pixels in the noise region would be significantly less than the number of pixels in the cluster they were merged to, thereby not significantly affecting the standard deviation of the entire cluster. In cases where the standard deviation would go above the assigned scatter threshold, it would be an indication to increase the number of desired clusters (k) for part B or increase the assigned scatter threshold for part A. In this way, plotting the standard deviation also helps in determining whether the parameter k (desired number of clusters) given to part B of the algorithm, or the assigned scatter threshold parameter given to part A of the algorithm is appropriate or needs to be changed, whichever might be permissible.

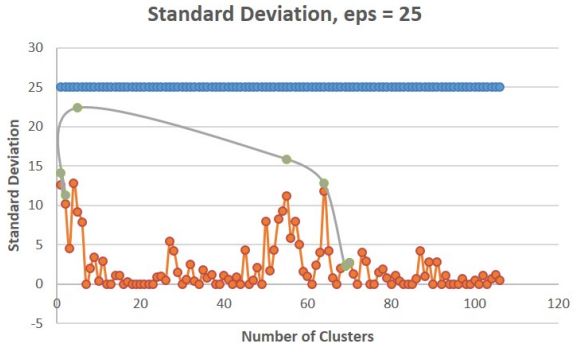
As seen in Figure 4.4, one can observer the standard deviation before merging and after merging. For some of the clusters in part A of the algorithm, the standard deviation shows zero. This is because, these are single pixel sized clusters and hence their standard deviation would come out to be zero. In both the figures, the standard deviation after merging is still lower than the criterion threshold.

Similarly, the span for every cluster gives the difference between the maximum color value and the minimum color value for a given cluster. This would give an idea of the range of color values that could be present in a given cluster which could be higher than the assigned scatter threshold. After implementing part B of the algorithm, the span for every cluster should be higher than the scatter threshold value, if pixels from other clusters were merged to it. If, for a given cluster, the span value after implementing part B of the algorithm and merging pixels of other clusters was lower than the scatter threshold, it would

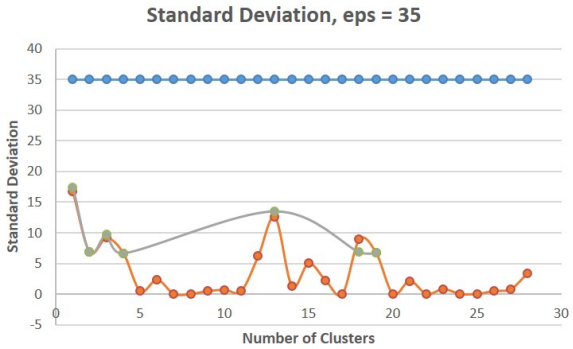


(a) Test Image

(b) Legend



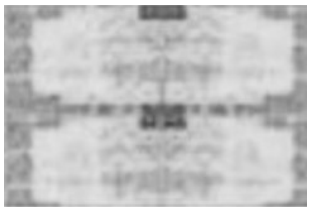
(c) Standard Deviation, eps = 25



(d) Standard Deviation, eps = 35

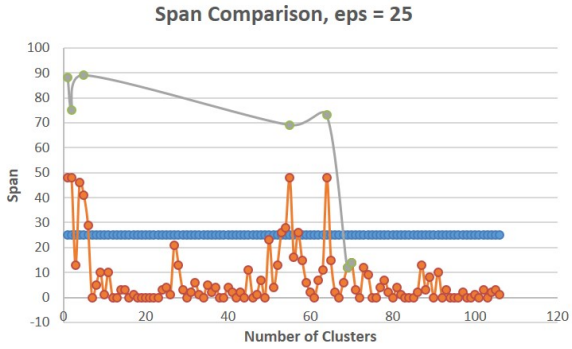
Figure 4.4: Standard deviation results

imply wrong clustering in the part A itself as the merged clusters should have anyways been a single cluster after implementation of part A of the algorithm as their standard deviation would have been lower than the scatter threshold. Span value would also help in determining where noise was incorporated as there would be a sudden rise in the span value of the cluster in which the pixels of the noisy region were merged to.

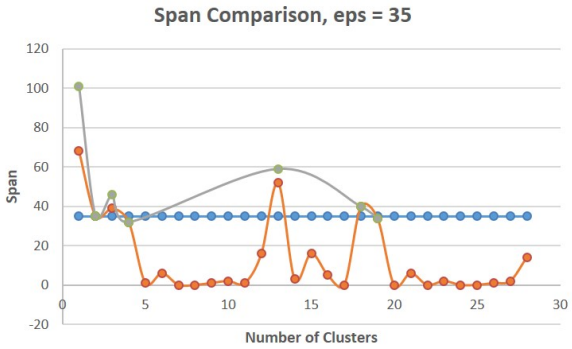


(a) Test Image

(b) Legend



(c) Span, eps = 25



(d) Span, eps = 35

Figure 4.5: Span per cluster results

As in Figure 4.5, one can see that the span per cluster is higher than the scatter threshold for a few clusters after part A of the algorithm, and significantly higher than the scatter threshold after part B of our algorithm. In a couple of instances, the span after part B is equal to the span value after part A, indicating that no pixels were merged to these respective clusters.

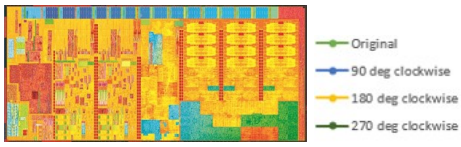
4.3.3 Effect of starting point and order of accessing neighbors on the clustering results

As discussed in section **??**, these experiments should help determine whether the clustering results are stable to the change in the starting point or the order in which the neighborhood pixels are accessed. As seen in Figure 4.6(c), the number of clusters formed is almost the same in our range of scatter thresholds i.e. (between 25 and 50). The CPU time as seen in Figure 4.6(d) in this range is also similar for each of the four starting points. These results show the robustness of our algorithm to the change in stating point in our operating range of scatter thresholds.

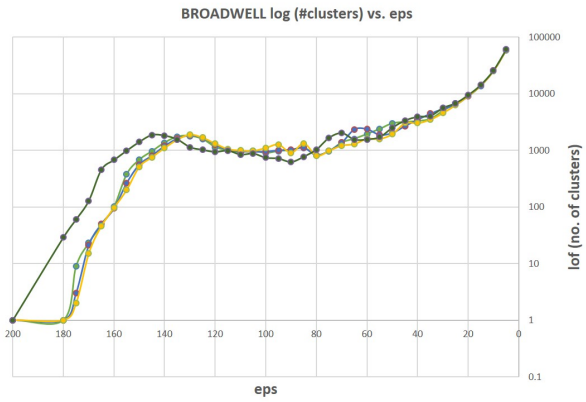
From Figure 4.7, one can see that changing the order in which the neighborhood pixels are accessed has no effect at all throughout the entire range of the scatter threshold on both, the number of clusters as well as the cpu time required.

4.3.4 Sum of Squared Errors

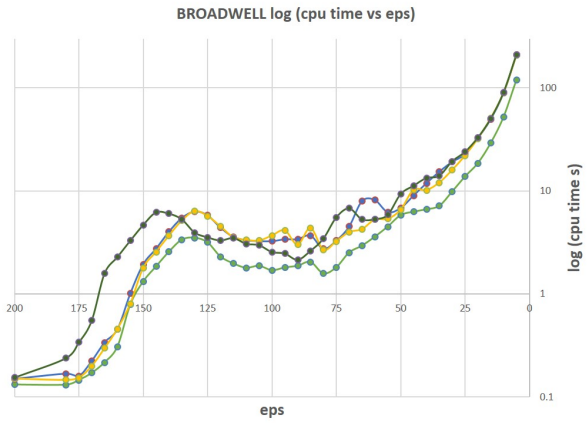
The sum of squared errors would help us determine how compact and relevant our clustering is. As seen in Table 4.1, for a high contrast image like the broadwell feature, the SSE value of the part B of our algorithm is almost three times that of the result of kmeans segmentation. For a low contrast or a smooth image like the lena image, the SSE value of our algorithm is almost five and half times more than the SSE value of the result with k-means segmentation. When compared to k-means, our algorithm shows less compact clusters as our clustering is



(a) Test Image (b) Legend



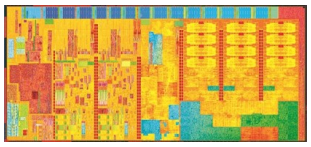
(c) No. of clusters vs. eps



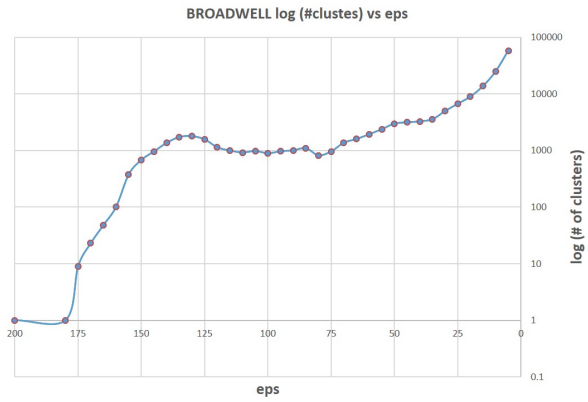
(d) CPU time vs. eps

Figure 4.6: Effect of starting point on clustering results

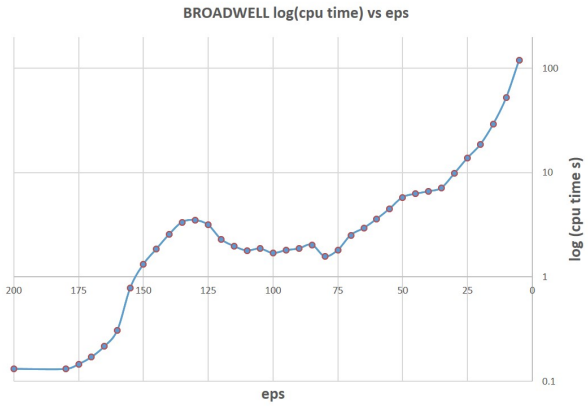
constrained by spatial connectivity as well as fixed number of clusters. Thus to achieve the desired number of clusters, pixels are merged to the relatively closer pixels based on their color, whether or not, they satisfy the variance threshold. This would significantly affect the SSE value. As against this, kmeans does generate desired number of clusters, but these are not spatially connected and thus



(a) Test Image (b) Legend



(c) No. of clusters vs. eps



(d) CPU time vs. eps

Figure 4.7: Effect of changing the order in which 8-neighborhood pixels are accessed

the resulting images show countless number of fragments even though they might have a better SSE value.

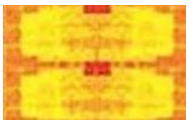
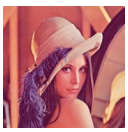
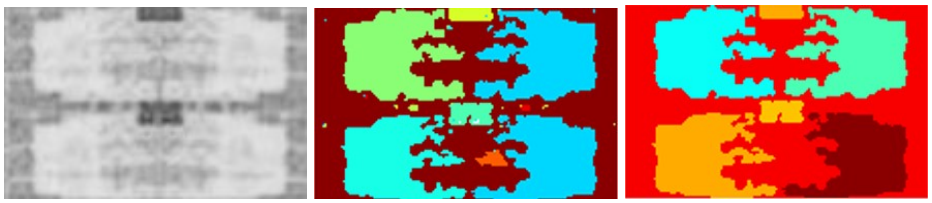
Image	k	Our Algorithm	k-means algorithm
	5	1.22e+07	4.094e+06
	5	9.2e+08	1.668e+08

Table 4.1: Sum of Squared Errors comparison

4.4 Incorporating multispectral information

The RGB channel information per pixel was used for clustering to demonstrate the capability of our algorithm to handle multispectral data. It also proved that having more input information per data point (which in this case is a pixel) would help to improve the segmentation results further.



(a) Grayscale Input (b) Result Part A (c) Result Part B

(d) RGB Input (e) Result part A (f) Result Part B

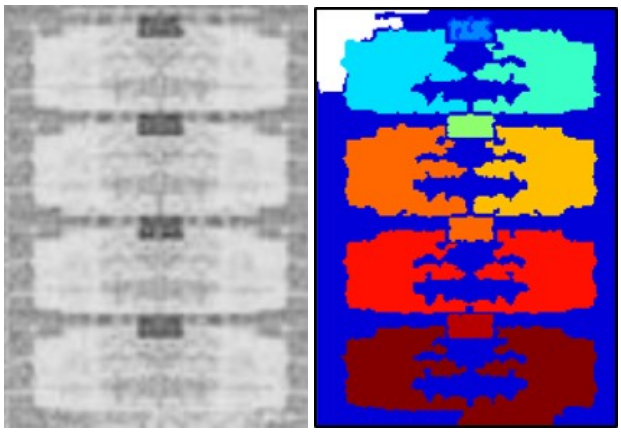
Figure 4.8: Improvement in clustering results when RGB channel information was used

As seen in Figure 4.9, one can see that the segmentation results for both part A and part B of the algorithm are more accurate to the desired clustering. Es-

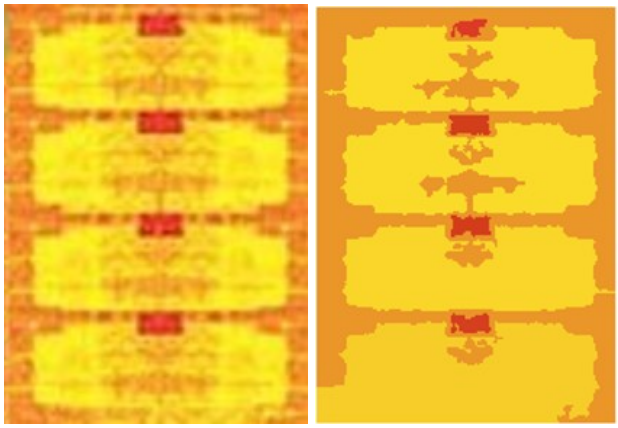
pecially for part B of the algorithm, one can see that with the RGB results, the larger features appear connected as expected and observed in the original input image. Whereas, in the part B results of grayscale input, these features appear disconnected.Visually, these features appear to be even more accurately segmented that the results with the grayscale input. In addition to this, use of RGB channels helps to avoid the leak in segmentation results which occurred in a few cases with monochrome input. As seen in Figure **??**, the result with grayscale input shows faulty segmentation due to formation of an additional cluster in the top left corner of the image, which, ideally should have been a part of the background. Once RGB channel information was used for segmentation, this error appears to have been removed and the segmentation results appear to be relatively more precise than the results with grayscale input. The use of Euclidean distance measure (Equation 3.1) with RGB channels as against Manhattan distance (Equation 3.1) measure with grayscale input, for comparing the color similarity could also be one of the contributing factors these improvement in the results.

4.5 Datasets

In addition to semiconductor images, we implemented our algorithm on other datasets like the one containing MRI images to show the nature of applications. In addition to this, we also teste our algorithm against the images in the Berkeley segmentation dataset[33], a standard dataset used for comparison by the image processing and computer vision community.



(a) Grayscale Input (b) Result Part B

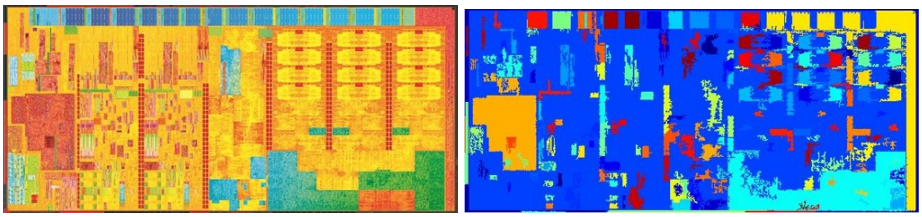


(c) RGB Input (d) Result part B

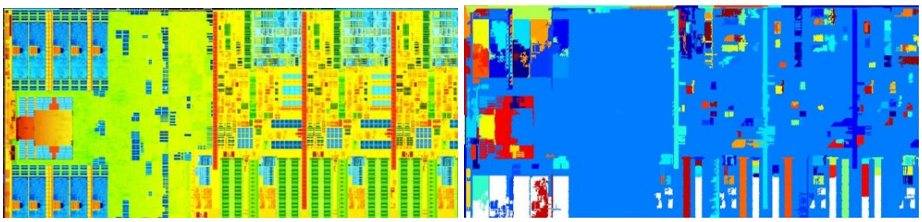
Figure 4.9: Removal of faulty segments upon upgrading to multispectral information for segmentation

4.5.1 Semiconductor Image Dataset

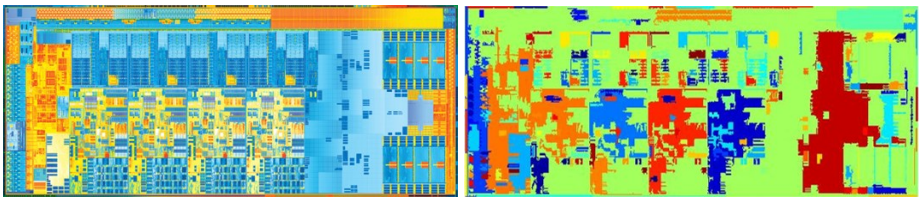
The algorithm was run on a several processor chip images color. Each of these images were of the order of several tens of thousands of pixels. In order to visualize the clustering results, better they were clustered into 200 clusters as we used a 256 color coding scheme. In practical use cases, the number of clusters in each of these images would be between 1000 and 2000, as shown in Figure 4.11. It is difficult to show 1000 unique clusters with 256 color coding due to the probability of adjacent unique clusters getting color coded to the same color.



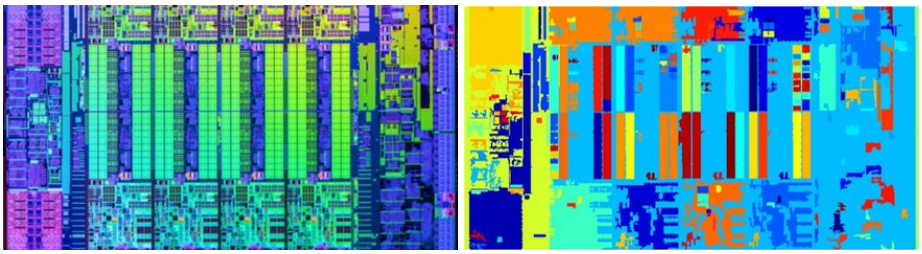
(a) Broadwell (b) Result - 200 clusters



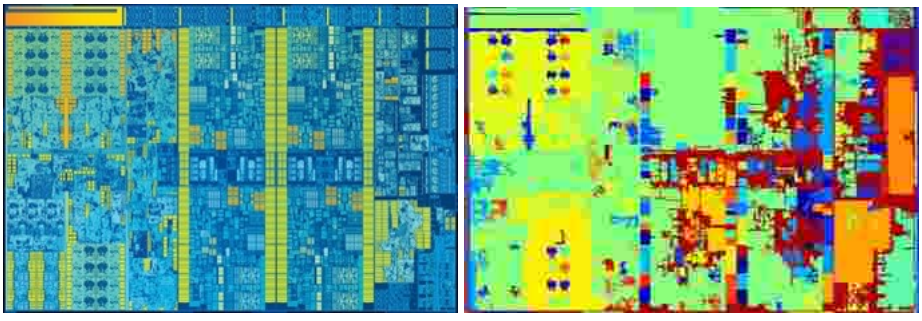
(c) Haswell (d) Result - 200 clusters



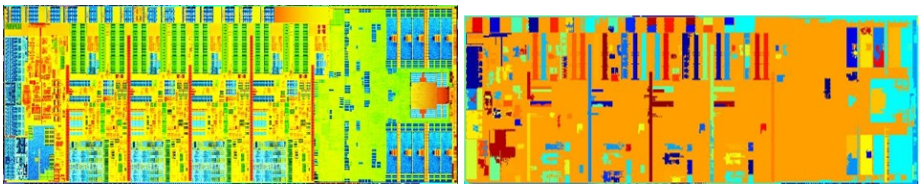
(e) Ivy Bridge (f) Result - 200 clusters



(g) Canonlake (h) Result - 200 clusters

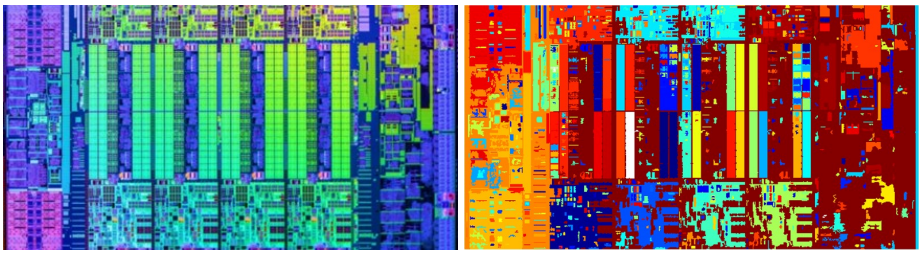


(i) Skylake (j) Result - 200 clusters



(k) Haswell High Resolution (l) Result - 200 clusters

Figure 4.10: Semiconductor Dataset



(a) Broadwell (b) Result - 1000 clusters

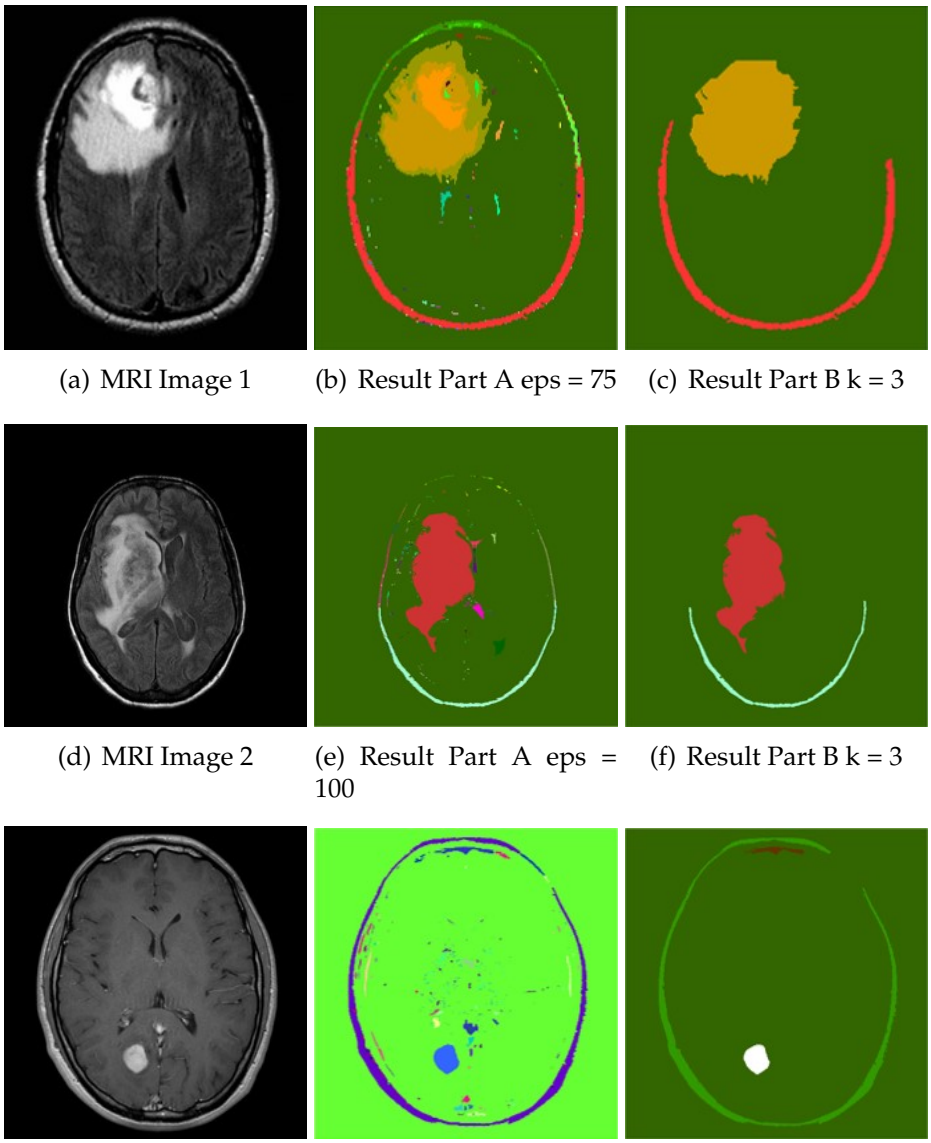
Figure 4.11: Clustering Result with 1000 clusters

4.5.2 MRI Dataset

We identified MRI datasets wherein the goal was to detect the specific regions of interest. For example, in our dataset, the goal was to identify the spatially contiguous regions of the brain tumor. From figure 4.12, one can visually see that the results of our algorithm help in identifying these regions of interest.

4.5.3 Berkeley Segmentation Dataset

The algorithm was also implemented on the set of images available in the Berkeley Segmentation Dataset[34]. This is a standard dataset used by several research groups in the image processing and computer vision community to implement their work and thereby have a common ground to compare their results.

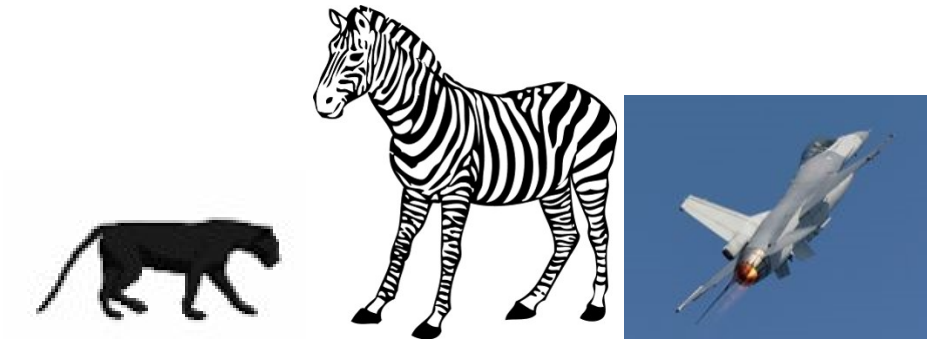


(a) MRI Image 1 (b) Result Part A eps = 75 (c) Result Part B k = 3

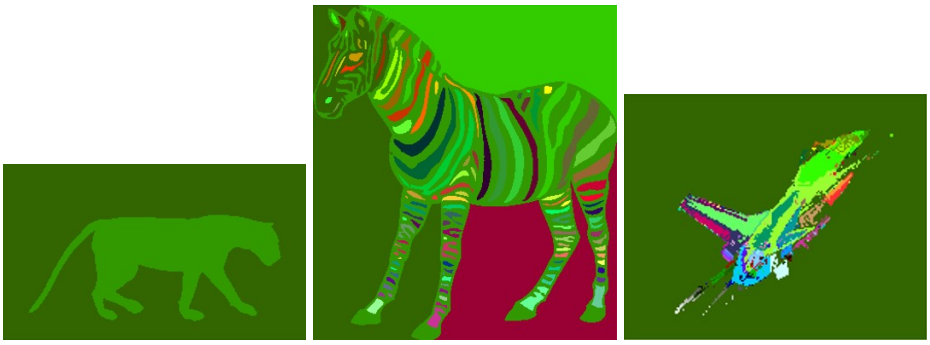
(d) MRI Image 2 (e) Result Part A eps = 100 (f) Result Part B k = 3

(g) MRI Image 3 (h) Result Part A eps = 100 (i) Result Part B k = 4

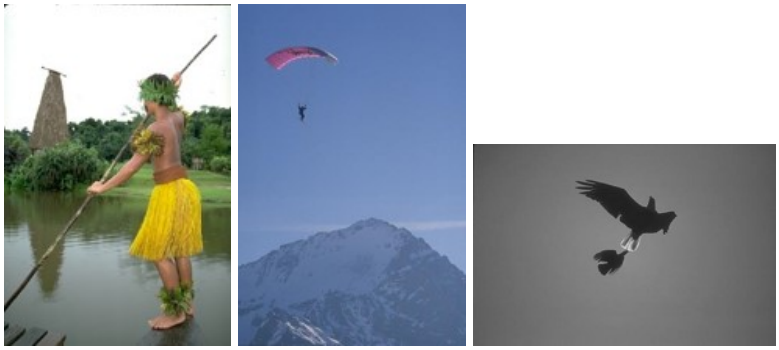
Figure 4.12: MRI Dataset



(a) Panther (b) Zebra (c) Fighter Jet



(d) Result Part A eps = 25 (e) Result Part A eps = 25 (f) Result Part A eps = 25



(g) Zulu (h) Birds (i) Parachute

(j) Result Part A eps = 45 (k) Result Part A eps = 35 (l) Result Part A eps = 25

Figure 4.13: Berkeley Segmentation Dataset

4.6 Comparison

We compared our algorithm with other state-of-the-art clustering methods. These included kmeans segmentation, Statistical Region Merging (SRM)[31], hierarchical clustering[35], and Efficient graph based segmentation[32]. These algorithms were implemented on one of the images of the Berkeley Segmentation Dataset and one of the images of the semiconductor dataset that we had used earlier.

We also compared the results of part A of our algorithm with those of region growing and k-means segmentation with respect to the CPU run-time required and number of contiguous regions formed as seen in Figure 4.3.(Here I have to explain kmeans gives scattered countless number of contiguous clusters although it divides into exact k clusters and slightly similar CPU time whereas region growing produces exactly k contiguous clusters as we desire but the process of going through seeds is very intensive and consumes a lot of time. Against our motive of having an efficient and fast algorithm scalable to large datasets. It would be practically very expensive computationally to identify seeds in a 200 Million Pixels image)

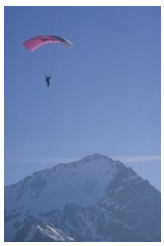
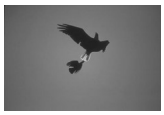
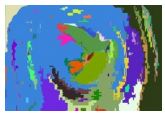
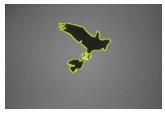
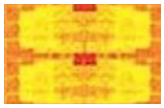
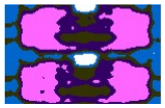
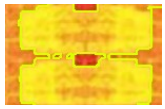
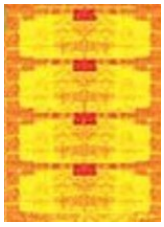
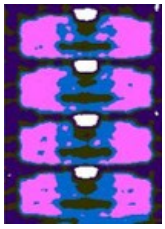
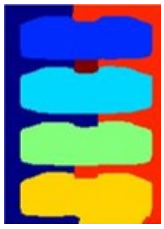
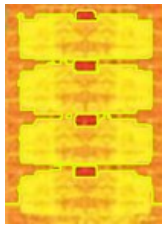
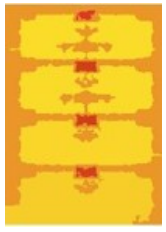
Image	kmeans	Graph based	Heirarchical	SRM	Our Algorithm
	 193 clusters t= 2.448 s	 17 clusters t = 195.4s	 128 clusters t = 0.08s	 58 clusters t = 1.367s	 58 clusters t = 0.444s
	 143 clusters t= 2.208 s	 3 clusters t = 170.676s	 25 clusters t = 0.082s	 13 clusters t = 0.278s	 25 clusters t = 0.131s
	 99 clusters t= 2.140 s	 9 clusters t = 167.722s	 99 clusters t = 0.07s	 6 clusters t = 0.0.315s	 5 clusters t = 0.136s
	 110 clusters t= 1.873 s	 7 clusters t = 7.553s	 10 clusters t = 0.0152s	 7 clusters t = 0.122s	 5 clusters t = 0.017s
	 113 clusters t= 2.698 s	 11 clusters t = 12.644s	 17 clusters t = 0.0208s	 9 clusters t = 0.141s	 9 clusters t = 0.037s

Table 4.2: Comparison with other basic approaches

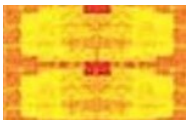
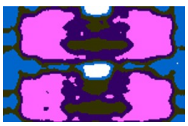
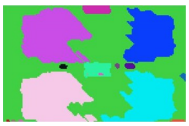
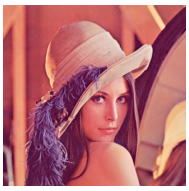
Image	k-means	Region-Growing	Our Algorithm
	 Clusters = 110 CPU time = 1.873s	 Clusters = 18 CPU time = 2.377s	 Clusters = 5 CPU time = 0.0135s
	 Clusters = 349 CPU time = 7.966s	 Clusters = 542 CPU time = 49.726s	 Clusters = 10 CPU time = 0.521s

Table 4.3: Comparison with other basic approaches

CHAPTER 5

SELECTION OF K IN KMEANS

5.1 Introduction

We have used this concept to find the color threshold or variance for our segmentation algorithm as the one corresponding to the K value of kmeans at the elbow of the graph. Since the number of desired can be decided by the user, the color threshold, 'eps' is currently a heuristic value based on the type and nature of the image. The correct choice of K of often ambiguous, often depending on the shape and scale of distribution of points in a dataset and the desired clustering resolution of the user. In addition, increasing K without penalty should reduce the amount of error in the resultant clustering, to the extreme case of zero error if each data point is considered its own cluster.We attempt to explore this direction, by calculating the color threshold value 'eps' with the help of elbow method used in determining the ideal K from kmeans.

5.2 Elbow Method

The elbow method looks at the percentage of variance explained as a function of the number of clusters. The number of clusters, K, should be chosen such that further adding more clusters does not give better modeling results of the data. Thus, if the percentage of variance against the number of clusters is plotted, the first clusters will keep add information, but at some point the marginal gain should drop, giving a noticeable change to the slope of the graph. The ideal

number of clusters is chosen at this point and hence the criterion is called as elbow criterion. However, this elbow sometimes can be difficult to identify if the nature of the data is very smooth.

5.3 Our Approach

Literature survey revealed the method developed by Pham et. al[36] as the most straightforward and well-performing approach. The goal of this algorithm is to find identify regions in which the data points are concentrated. It is also important to analyze the internal distribution of each cluster as well as its relation to other clusters in the data set. The distortion of a cluster is a measure of the distance between points in a cluster and its centroid:

$$I_j = \sum_{x_i \in C_j}^{b} \parallel x_i - \mu_i \parallel \tag{5.1}$$

The global impact of all clusters distortions is given by the quantity:

$$S_k = \sum_{j=i}^{k} I_k \tag{5.2}$$

The authors Pham et al. proceed to discuss further constrains that the sought-after function f(K) should verify for it to be informative to the problem of selection of K. They finally arrive at the following definition: Nd is the number of dimensions (attributes) of the data set and alpha is a weight factor. With this definition, f(K) is the ratio of the real distortion to the estimated distortion and decreases when there are areas of concentration in the data distribution. Values of K that yield small f(K) can be regarded as giving well defined clusters.

$$f(x) = \begin{cases} 1, & \text{if K = 1} \\ \frac{S_K}{\alpha_K S_{K-1}}, & \text{if } S_{K-1} \neq 0, \forall K > 1 \\ 1, & \text{if } S_{K-1} = 0, \forall K > 1 \end{cases} \tag{5.3}$$

$$\alpha(x) = \begin{cases} 1 - \frac{3}{4N_d}, & \text{if } K = 2 \ \& \ N_d > 1 \\ \alpha_{K-1} + \frac{1-\alpha_{K-1}}{6}, & \text{if } K > 2 \ \& \ N_d > 1 \end{cases} \tag{5.4}$$

The average distance between the centroids for a for every K value of kmeans is calculated from K = 2 to K = 10. The corresponding compactness values are also noted. These values are then sorted based on the values of $\frac{avg.dist.}{sqrt(compactness)}$ and the largest value is chosen as the desired threshold i.e. eps value.

5.4 Results

We tested this approach on the Broadwell feature images and get the calculated color threshold value of 43 for k = 5 as against our heuristic value of 50. In addition, this eps value is obtained at K = 5 i.e. elbow of the curve as seen in figure 5.1.

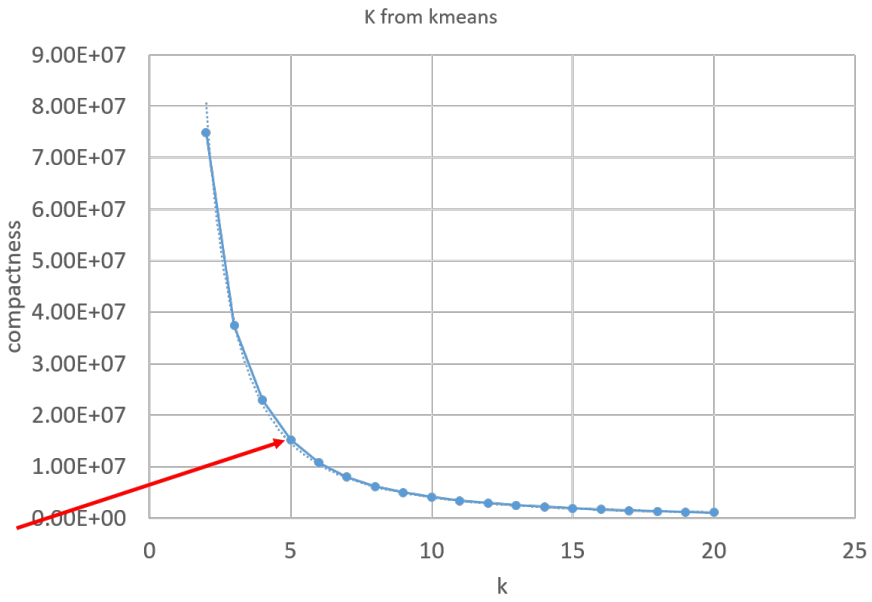


Figure 5.1: Scalability of the algorithm to large datasets

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

A two stage computational algorithm has been designed to address image segmentation requirements for semiconductor lithography. In the first stage we divide the given spatial matrix (in this case, an image) into spatially contiguous regions constrained by specified thresholds. In the second stage, based on the number of spatially contiguous regions desired, we perform region merging to produce an exact number of spatially contiguous regions, while minimizing the standard deviation within each cluster. The algorithm is fast and efficient and scalable to large sets of data as demonstrated by processing a 200 million pixels image in under 100 CPU runtime seconds. The runtime complexity has been calculated at $O(n)$ when meaningful parameters were provided.

The final output results have been verified and validated by use of several evaluation measures including histograms, standard deviation per cluster, span per cluster, effect of starting point for the algorithms, order in which neighbors are accessed, and by measuring the sum of squared error (SSE). A number of target datasets were used to test the algorithm which included: images of processor chips in the semiconductor dataset, MRI images of the brain, and several images in the Berkeley Segmentation Dataset.

The algorithm's performance was compared against several state-of-the-art clustering methods and the comparison results show similar or slightly better segmentation with control over the number of segments while still maintaining

the shape connectivity. Use of RGB channel information showed improvement in the clustering results which helps to demonstrate the advantage of having multispectral input data as input as well as the capabilities of our algorithm to handle multispectral information.

6.2 Future Work

While the algorithm has shown promise for well defined images, it is critical that its behavior with different types of noise i.e. salt and pepper, quantization and shot be evaluated. Currently, the algorithm is known to be robust to a gaussian noise profile. Anothter area of future study would be quantitative measures of the 'efficacy' of segmentation. Based on the literature survey, several methods such as the Global Clustering Estimate (GCE), Local Clustering Estimate (LCE) and Precision and Recall rates are possibilities that should be considered within our problem statement.

BIBLIOGRAPHY

[1] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[2] King-Sun Fu and JK Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.

[3] Heng-Da Cheng, XH Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[4] Raghu Krishnapuram and James M Keller. The possibilistic c-means algorithm: insights and recommendations. *Fuzzy Systems, IEEE Transactions on*, 4(3):385–393, 1996.

[5] Jie Wei. Image segmentation using situational dct descriptors. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 738–741. IEEE, 2001.

[6] Kobus Barnard, Pinar Duygulu, and David Forsyth. Clustering art. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–434. IEEE, 2001.

[7] Yining Deng and BS Manjunath. Unsupervised segmentation of color-texture regions in images and video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):800–810, 2001.

[8] Tuan D Pham. Image segmentation using probabilistic fuzzy c-means clustering. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 722–725. IEEE, 2001.

[9] Patrice Delmas. Image segmentaion. `https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm`, 2016.

[10] Y.H.Wang. Tutorial: Image segmentation. Technical report, Graduate Institute of Communication & Engineering, National Institute of Taiwan, Taipei, Taiwan.

[11] H. Talbot Z. Lin, J. Jin. Unseeded region growing for 3d image segmentation. volume 9, pages 31 – 37, 2000.

[12] T. Pavlidis S. L. Horowitz. Picture segmentation by a tree traversal algorithm. *JACM*, 23:368 – 388, April, 1976.

[13] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[14] Qixiang Ye, Wen Gao, and Wei Zeng. Color image segmentation using density-based clustering. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 2, pages II–401–4 vol.2, July 2003.

[15] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, sep 1993.

[16] Edward M. Riseman and Michael A. Arbib. Computational techniques in the visual segmentation of static scenes. *Computer Graphics and Image Processing*, 6(3):221–276, jun 1977.

[17] G.A. Hance, S.E. Umbaugh, R.H. Moss, and W.V. Stoecker. Unsupervised color image segmentation: with application to skin tumor borders. *IEEE Eng. Med. Biol. Mag.*, 15(1):104–111, 1996.

[18] O. Lézoray and C. Charrier. Color image segmentation using morphological clustering and fusion with automatic scale selection. *Pattern Recognition Letters*, 30(4):397–406, mar 2009.

[19] Ghassan Hamarneh and Xiaoxing Li. Watershed segmentation using prior shape and appearance knowledge. *Image and Vision Computing*, 27(1-2):59–68, jan 2009.

[20] Soumya Dutta and Bidyut B Chaudhuri. Homogenous region based color image segmentation. In *Proceedings of the world Congress on Engineering and computer Science*, volume 2, pages 20–22, 2009.

[21] Biplab Banerjee, Tanusree Bhattacharjee, and Nirmalya Chowdhury. Color image segmentation technique using natural grouping of pixels. *International Journal of Image Processing (IJIP)*, 4(4):320–328, 2010.

[22] Qixiang Ye, Wen Gao, and Wei Zeng. Color image segmentation using density-based clustering. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–345. IEEE, 2003.

[23] Chunming Li, Rui Huang, Zhaohua Ding, J Chris Gatenby, Dimitris N Metaxas, and John C Gore. A level set method for image segmentation in the presence of intensity inhomogeneities with application to mri. *Image Processing, IEEE Transactions on*, 20(7):2007–2016, 2011.

[24] Jianping Fan, David KY Yau, Ahmed K Elmagarmid, and Walid G Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *Image Processing, IEEE Transactions on*, 10(10):1454–1466, 2001.

[25] Hsin-Chia Chen, Wei-Jung Chien, and Sheng-Jyh Wang. Contrast-based color image segmentation. *Signal Processing Letters, IEEE*, 11(7):641–644, 2004.

[26] AK Qin and David A Clausi. Multivariate image segmentation using semantic region growing with adaptive edge penalty. *Image Processing, IEEE Transactions on*, 19(8):2157–2170, 2010.

[27] Frank Y Shih and Shouxian Cheng. Automatic seeded region growing for color image segmentation. *Image and vision computing*, 23(10):877–886, 2005.

[28] Luis Garcia Ugarriza, Eli Saber, Sreenath Rao Vantaram, Vincent Amuso, Mark Shaw, and Ranjit Bhaskar. Automatic image segmentation by dynamic region growth and multiresolution merging. *Image Processing, IEEE Transactions on*, 18(10):2275–2288, 2009.

[29] Ameer Ali, Gour C. Karmakar, and Laurence S. Dooley. Image segmentation using fuzzy clustering incorporating spatial information. In *International Conference on Artificial Intelligence and Applications (AIA '04)*, February 2004.

[30] Jian-Jiun Ding, CJ Kuo, and WC Hong. An efficient image segmentation technique by fast scanning and adaptive merging. *CVGIP, Aug*, 2009.

[31] Richard Nock and Frank Nielsen. Statistical region merging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1452–1458, 2004.

[32] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[33] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[34] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[35] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[36] Duc Truong Pham, Stefan S Dimov, and CD Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.