CONNECTIVITY ENFORCED BAYESIAN SUPERPIXELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR EKER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

CONNECTIVITY ENFORCED BAYESIAN SUPERPIXELS

submitted by **ONUR EKER** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Eng. Dept., METU** _____

Dr. Kutalmış Gökalp İnce
Co-supervisor, **Center for Image Analysis, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Selim Aksoy
Computer Engineering Dept., Bilkent University _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering Dept., METU _____

Assoc. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering Dept., METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Onur Eker

Signature :

ABSTRACT

CONNECTIVITY ENFORCED BAYESIAN SUPERPIXELS

Eker, Onur
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan
Co-Supervisor: Dr. Kutalmış Gökalp İnce

September 2019, 132 pages

In this study a general flow for clustering-based superpixel (SP) extraction methods is presented, while each step is analyzed in detail, and improvements are proposed. Considering general SP extraction method steps, initial grid alternatives are examined. The necessity of initial grid refinement is studied and unlike current approaches, a novel Edge Based Refinement step which does not break regular grid structure is proposed. Label update constraints are also analyzed in terms of preserving regular initial tiling, and Just Connected method enforcing connectivity from the beginning is proposed. The requirement of adjusting one of hyper-parameters, iteration count, for different image resolutions and different number of SPs is eliminated by determining the number of iterations relative to SP area. Considering these proposals,extensions to the state-of-the-art SP methods, SLIC+ and LASP+, are proposed which normalize spatial term with SP spatial covariance. Novel cost-functions SLIC++ and LASP++ are also presented for a further improvement with the normalization of spectral term with SP specific dynamic parameter. Finally, a Bayesian classifier is proposed for pixels during SP label assignment. Based on improvements in various steps mentioned above, a family of superpixel extraction methods including, SLIC++/R, SLIC++/H,

LASP++/R. LASP++/H, BSP/R and BSP/H are presented. For the evaluation, a novel Boundary Achievable Segmentation Accuracy metric is proposed that replaces three frequent metrics from the literature. Compactness and area under curve approaches are also proposed as evaluation methods to minimize any performance ambiguity for the literature benchmarks. Both proposed spectral term and spatial term improvements significantly increase accuracy of generated SPs with no execution-time burden. In addition, employing Bayesian classifier leads to generate more accurate SPs in a shorter amount of run-time. Besides, with the proposed label update criteria, connectedness of SPs are ensured during generation process that preserves regular grid topology enabling them to be fed into conventional neural-networks.

Keywords: Superpixel, Oversegmentation, Bayesian, Clustering-based, Connectivity

ÖZ

BAĞLANIRLIĞA ZORLANMIŞ BAYESÇİ SÜPERPİKSELLER

Eker, Onur
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan
Ortak Tez Yöneticisi: Dr. Kutalmış Gökalp İnce

Eylül 2019 , 132 sayfa

Bu çalışmada küme tabanlı süper piksel (SP) çıkarım metodları detaylı incelenmektedir. Bu metodların genel akışı ortaya konularak, akışın her bir adımı detaylı analiz edilmekte ve iyileştirmeler önerilmektedir. Genel SP çıkarım metod adımlarından olan ilk örgü alternatifleri incelenmektedir. İlk örünün düzeltilmesi gerekliliği çalışılarak diğer yöntemlerden farklı olarak, düzenli örü yapısını bozmayan Kenar Tabanlı Düzeltim önerilmektedir. Etiket güncelleme kısıtları düzenli ilk örüyü koruması bakımından analiz edilmektedir ve bağlanırlığı baştan zorlayan Sadece Bağlı yöntem önerilmektedir. Yineleme sayısı parametresinin farklı görüntü çözünürlükleri ve farklı SP sayıları için ayarlanması gerekliliği, yineleme sayısını SP alanına göreli hale getirilerek ortadan kaldırılmaktadır. Uzamsal terimi SP uzamsal kovaryans ile normalize ederek güncel metodları iyileştiren SLIC+ ve LASP+ metodları önerilmektedir. SP dinamik parametreleri ile spektral terimi normalize ederek metodları daha da iyileştiren SLIC++ ve LASP++ maliyet fonksiyonları önerilmektedir. Son olarak piksellere SP etiketi ataması için bir Bayesian sınıflandırıcı önerilmektedir. Yukarıda bahsedilen algoritmanın farklı adımlarında yapılan iyileştirmeler sonucunda bir süper piksel çıkarım metod ailesi önerilmektedir, bu aile SLIC++/R, SLIC++/H, LASP++/R. LASP++/H,

BSP/R ve BSP/H metodlarını içermekte ve R kare ilk örü, H ise altıgen ilk örüyü temsil etmektedir. Ölçüm değerlendirmesinde ise literatürde kullanılan 3 metrik; Sınır Çağrışımı, Yetersiz Bölütleme Hatası ve Erişilebilir Bölütleme Doğruluğu metriği yerine geçen Sınır Erişilebilir Bölütleme Doğruluğu metriği önerilmektedir. Literatürdeki ölçüm belirsizliğini gidermek için değerlendirme metodu olarak kompaktlık ve eğri altındaki alan metodları önerilmektedir. Önerilen spektral ve uzamsal terimlerdeki iyileştirmeler, üretilen süper piksellerin doğruluğunu koşum zamanını etkilemeden belirgin şekilde arttırmaktadır. Ek olarak, Bayesian sınıflandırıcı kullanımı daha kısa koşum zamanında daha doğru süper pikseller üretilmesini sağlamaktadır. Bununla birlikte, önerilen etiket atama kriteri üretim aşamasında süper piksellerin bağlanırlığını garanti etmekte ve böylece düzgün örü topolojisini koruyarak onların sinirsel ağlara beslenebilmesini sağlamaktadır. Ölçümlerde yapılan iyileştirmeler ile ölçümleri parametre bağımlılığından kurtararak metod performans değerlendirmesinin kapsamlı şekilde yapılabilmesini sağlamaktadır.

Anahtar Kelimeler: Süperpiksel, Aşırı Bölütleme. Bayesian, Küme Tabanlı, Bağlanırlık

to Can

ACKNOWLEDGMENTS

I would like to thank my thesis supervisor Prof.Dr. A. Aydın Alatan of Electrical Electronics Engineering Department at Middle East Technical University and co-supervisor Dr. Kutalmış Gökalp İnce of Center for Image Analysis at Middle East Technical University for enormous guidance and support. Without their assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. It was honour for me to be their student. Thank you.

I must express my very profound gratitude to my wife and my son for providing me with unfailing support and continuous encouragement throughout my study and through the process of researching and writing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
SP	SuperPixel
EBR	Edge Based Refinement
PDR	Predefined Re-segmentation
BSP	Bayesian Superpixels Method
LASP	Local Adaptive Superpixels Method
SLIC	Simple Linear Iterative Clustering Method
RGB	Red-Green-Blue Dimensions
RGB-D	Red-Green-Blue-Depth Dimensions
CIE	International Commission on Illumination
LAB	L(luminance), A(red-green axis), B(blue-yellow axis) Dimensions
RECT	Rectangular
HEX	Hexagonal
BR	Boundary Recall
UE	Under-segmentation Error
ASA	Achievable Segmentation Accuracy
BASA	Boundary Achievable Segmentation Accuracy
EV	Explained Variation
CO	Compactness
AUC	Area Under Curve
SOTA	State of the Art
JC	Just Connected
BSD	Berkeley Segmentation Data Sets

Fr	Freemans metric
PR	Proffitt and Rosens Metric
VS	Vossepoel and Smeulders Metric
PC	Pixel Count Metric
ED	Erosion/dilation Difference Metric
ST	Stutz benchmark code Metric
FG	Foreground Pixels
BG	Background Pixels
W	Watershed Method
PF	Path Finder Method
STP	Speeded-up Turbo Pixels Method
SEEDS	Superpixels Extracted via Energy-Driven Sampling Method
SCS	Simply-connected Superpixels Method
DASP	Depth-Adaptive Superpixels Method
VC	VCells Method
VCSS	Voxel Cloud Connectivity Segmentation-Supervoxels for Point Clouds Method
LSC	Linear Spectral Clustering Superpixel Method
ERGC	Eikonal-based Region Growing Clustering Method
ETPS	Real-Time Coarse-to-fine Topologically Preserving Segmentation Method
CRS	Contour Relaxed Superpixels Method

CHAPTER 1

INTRODUCTION

Picture elements, or their well-known abbreviation *pixels* store discrete information proportional to the light radiance and spectral content from the scene points onto an image. However, this individual information is quite poor without spatial neighborhood relations of these pixels. On the other hand, processing an image in pixel-wise requires high computation power as it needs extra local deductions to correlate pixels. In order to ease those problems, pixels are grouped into clusters having similar properties in terms of color and spatial distance, namely *superpixels* [1].

Using superpixels (SPs) rather than pixels, significantly decreases the number of units to be processed and also provides more compact neighborhood information to applications, such as segmentation, optical flow, tracking, object detection, 3D-reconstruction. In other words, SP extraction is a pre-processing step that reduces computational complexity and improves performance of subsequent applications in many applications.

It should be noted that the output of any segmentation algorithm that yields relatively large number of regions that contain relatively small number of pixels with no semantic meaning is denoted as *oversegmentation*. Superpixels are oversegmented representations with further requirements.

1.1 Superpixel Definition

A SP representation should have the following requirements: [2]:

1. SPs should partition an image with a connected group of pixels having similar

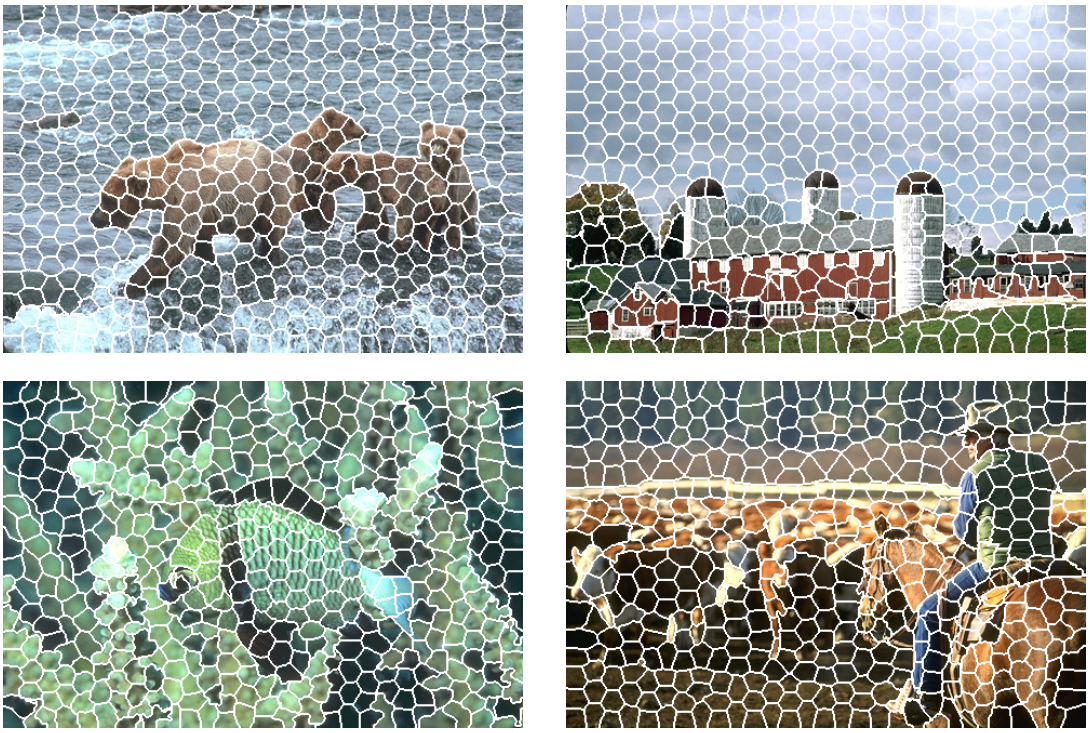


Figure 1.1: Typical Superpixel representations

spectral (i.e. color) properties.

2. Each pixel should be assigned to one SP, overlaps are not allowed.

3. SPs should cover whole image.

4. SPs should present a regular and compact structure, while following the object boundaries (This requirement makes SPs differ from other oversegmentation techniques).

5. Each SP should (preferably) contain pixels from a single semantic object.

6. Number of SPs should be controllable.

7. SP extraction methods should present a fair performance in terms of time and memory, since they are emerged against inefficient pixel-wise processing.

Figure 1.1 shows typical SP representations having 1000 SPs generated using proposed methods in Section 5.1.

1.2 Scope of Thesis

The scope of this thesis can be summarized as follows:

- This thesis focuses on SP extraction, which carry no semantic information, on N (spectral) channel still images. SP specific requirements that distinguish SP extraction from other oversegmentation methods are also mentioned.
- Different approaches for SP extraction are investigated. Their advantages and drawbacks are explained in terms of performance and efficiency.
- As the most effective ones among SP extraction approaches, clustering-based methods are analyzed in detail. Alternatives for each step of clustering-based methods are evaluated and new ones are proposed.
- SP performance evaluation is studied; drawbacks and strengths of performance evaluation methods and metrics are presented, and alternative approaches are proposed.

1.3 Contributions of Thesis

The fundamental drawbacks related to SP research can be stated as follows:

SP extraction methods in the literature are grouped into several categories. Performance of these methods are evaluated based on SP requirements given in Section 1.1. However, the related literature work does not make it clear to distinguish performance of methods. For instance, there is no metric that is independent of SP number. Even for a fixed number of SPs, the present metrics measure segmentation accuracy of individual semantic objects; however, they do not take compactness of SPs into account. In general, as the compactness decreases, the segmentation accuracy increases. Compactness is usually adjusted by a hyper-parameter, however this parameter is insufficient to quantify the compactness of SPs.

SP extraction methods consist of various steps, although each step has an effect on overall performance, effects of these steps are not examined rigorously by related

past works. Furthermore, those methods are usually configured by method specific hyper-parameters, and the effect of those parameters are completely ignored. Each method defines its own set of parameters and even if there are some common ones, they do not affect the performance in the same manner. Besides, benchmarks ignore the shapes of SPs which are usually controlled by those hyper-parameters. Moreover, the method configurations in experiments are not the optimum ones due to lack of published information. Therefore, currently state-of-the-art methodologies are evaluated regardless of inner step options and benchmark comparisons do not distinguish performances of SP extraction methods well.

In this thesis work, it is aimed to focus on above mentioned problems in literature and to propose some contributions. This effort includes examining every step of clustering-based SP extraction methods, investigating performance of each available option, and proposing alternative approaches to improve performance when possible. In order to get rid of confusion with hyper-parameters, it is aimed to have a minimum set of parameters to configure the methods. Hyper-parameter dependency problems in benchmarks will also be investigated; new approaches will be put forward to have a common base benchmark framework to ease the comparison of different methods in a fair way, and provide sufficient information to applications that to use SP extraction methods.

SPs are employed by graph-based segmentation methods [3], and shown to be quite effective. However, since SP methods are usually do not preserve a regular grid structure, they cannot be employed by neural networks. One of the priorities of this work is to propose methods that generate SPs in a regular grid.

Finally, a family of clustering-based SP methods is proposed which follows object boundaries better in a shorter time, while allowing to control compactness and number of SPs, and resulting in a regular SP grid. Since every step of clustering-based SP extraction methods are examined in terms of their effect on performance, pointed improvements can be applied to rest of the methods in the literature. In addition to new methods, a benchmark framework will be proposed to be used in further studies that more accurately investigates the performance of alternative methods.

1.4 Outline of Thesis

This work investigates the framework of clustering-based Superpixel extraction methods, extract drawbacks of state-of-the-art methods, and propose new approaches to get a better performing family of methods. To achieve this, first, the state-of-the-art SP extraction methods are investigated in Chapter 2. Second, benchmark metrics for performance comparison are presented, and alternative metrics and evaluation methods are proposed in Chapter 3. Third, each step of a general clustering-based SP extraction method is analyzed in detail, alternatives are proposed, and their affect on performance is evaluated with experiments in Chapter 4. Then, in Chapter 5 new SP extraction methods are proposed by bringing the best performing proposed alternatives of each step together, and a comprehensive set of experiments are conducted to compare performance of proposed methods to the top performing state-of-the-art methods. Finally, in Chapter 6 conclusions are presented.

CHAPTER 2

RELATED WORK

Superpixel (SP) extraction methods can be categorized based on the difference between their fundamental strategies. In such an effort, Achanta [2] divides all SP extraction methods into two categories as *graph-based* and *gradient-ascent* methods. In the graph-based approaches, the pixels are assumed to form graphs and neighboring pixel similarities set the edge weight of the graph nodes. These graphs are partitioned forming SPs that minimize a cost function. On the other hand, in gradient-ascent methods, the image is initially clustered and clusters are updated iteratively. In these iterative approaches, the algorithm starts either 1) from a non-regular clustered structure called seeds and seeds grow until meeting a specific criteria or 2) from regular distributed clusters and at each iteration clusters grow around their centers by exchanging a single pixel or a block of pixels.

In general, one can argue that graph-based methods adhere well to object boundaries, however they have a very high computational complexity increasing with the image size. Iterative methods have fair performance in terms of boundary adherence and undersegmentation error with a very low time complexity and regular distribution of SPs.

As a different taxonomy of SP extraction methods, Stutz [4] divides SP extraction methods into eight categories as depicted in 2.1, in order to provide high-level abstraction of algorithm details. In this chapter, Stutz [4] and Achanta's [2] categorization approach are combined as shown in Figure 2.2and the resultant categories with well-known state-of-the-art techniques from each technique are explained in the following sections.

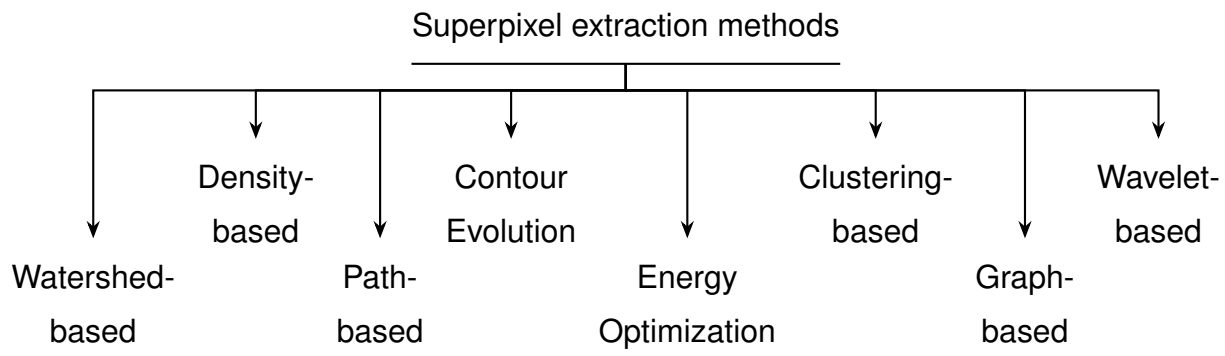


Figure 2.1: Superpixel extraction method categorization of Stutz [4]

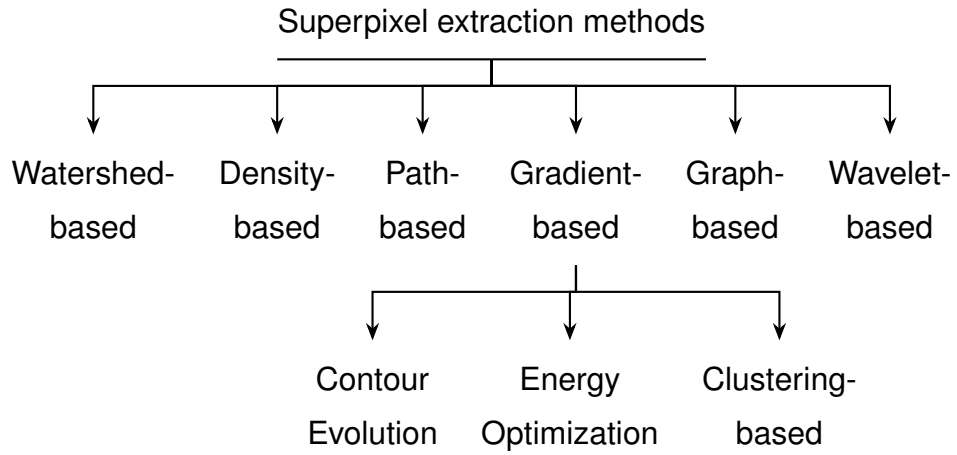


Figure 2.2: Proposed combined categorization for superpixel extraction methods of Stutz [4] and Achanta [2]

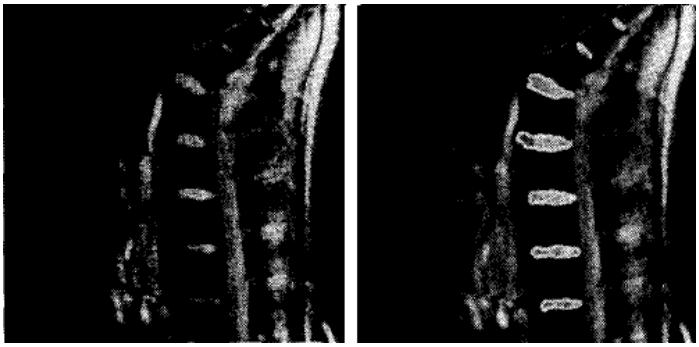


Figure 2.3: Watershed-based method SP extraction example (Vincent [5] Figure 16). Left to right represents original and segmented image respectively.

2.1 Watershed-based Methods

Watershed is a mature segmentation method from the literature, typically creating an oversegmented representation. These methods apply gradient-ascent method from an initial local minimum to compute lines that separate basins called watershed. Watershed (W) [5] algorithm is known to be a relatively fast algorithm generating very compact structures. However, this method lacks regularity in shape and size and shows poor performance in boundary recall with no control over SP count and their compactness. Figure 2.3 depicts an example extraction of this category taken from Vincent's [5] work.

2.2 Density-based Methods

The methods belonging to this category assign each pixel to a closest partition in the density image. One of the well known method of this category Edge augmented mean shift [6] (EAMS) algorithm, finds modes of image in terms of intensity or color and assigns each pixel to the closest one. The algorithm has no control over generated number of pixels and compactness. Method shows fair boundary adherence but its performance is not stable under different images, i.e. variations on performance metrics are high between images [4]. Figure 2.4 shows an example extraction of EAMS taken from Meer's [6] work.

Figure 2.4: Density-based SP method extraction example (Meer [6] Figure 7). Left to right represents original and segmented image respectively.

Figure 2.5: Path-based method SP SP extraction example (Drucker [7] Figure 5). Left to right represents original and segmented image respectively.

2.3 Path-based Methods

This type of SP extraction methods connect seed points along SP paths using discrete image gradients or edge detection. One popular state of-the-art-method, Path finder (PF) [7], employs discrete image gradients as connection criteria. As indicated in [4], that method is able to control the number of SPs, but does not provide a control over compactness and generates highly non-compact pixels. Computation time is relatively low, however method performs poorly under performance metrics, such as boundary recall and under-segmentation error, and shows unstable performance across images. Figure 2.5 presents an example extraction of PF taken from Drucker's [7] work.

Figure 2.6: Graph-based method SP extraction example (Liu [9] Figure 5). Ground truth segments labels are color-coded

2.4 Graph-based Methods

These methods form graphs whose nodes are pixels and partition the graphs to minimize a cost function. In normalized cuts [8] (NC) method, edge-weights of graphs partitioning the image is based on similarities among the pixels. Stutz [4] states that the number of SPs is controllable for this method and regular clusters are generated, however computation time and memory usage is extremely high and boundary adherence performance is low.

Entropy Rate Superpixel Segmentation (ERS) [9] applies a graph segmentation algorithm that optimizes graph topology, having entropy rate maximization term to generate homogeneous and compact SPs while a balancing term to force SPs having identical size. ERS enhances greedy algorithm in entropy rate maximization term. According to Stutz [4], ERS shows one of top boundary adherence performance with consistency over different data sets. Its drawback is having relatively higher run-time. Figure 2.6 shows an example extraction of ERS taken from Liu's [9] work.

2.5 Wavelet-based Methods

Edge-avoiding wavelets [10] (SEAW), use multi-scale image analysis approach such that for each SP a reconstruction image is formed using pre-calculated weights, and at the end all the resultant images are merged. The number of generated SPs can be

Figure 2.7: Wavelet-based method SP extraction example (Strassburg [10] Figure 8). Left to right represents original image and gray-coded segments respectively

controlled, but method provides no control over compactness. Boundary adherence performance is very low, computation time and memory usage is very high according to Stutz [4]. Figure 2.7 depicts an example extraction of SEAW taken from Strassburg's [10] work.

2.6 Gradient-based Superpixel Extraction Methods

2.6.1 Contour Evolution Methods

In this approach, the seeds are initially assigned and they grow iteratively to reach to a local minimum (gradient ascent) of an appropriate cost function. Some popular methods of this category can briefly explained as follows:

- **TurboPixels - Fast Superpixels:** TP [11] starts from an initial uniformly seeded distribution. At each iteration, seeds dilate until no expanding region is left. Velocity calculation has two arguments; 1) the proximity of SPs to other SPs and 2) local affinity of the pixels. First one is for non-overlapping SPs, the latter is for boundary recall. According to calculated velocity, boundary expands on the following iteration. Boundary velocities are calculated and skeleton is formed and maintained using an algorithm which is based on the distances between regions. At the end of the algorithm, the resultant skeleton gives SP extraction. Stutz [4] reports that method generates very compact SPs, however does

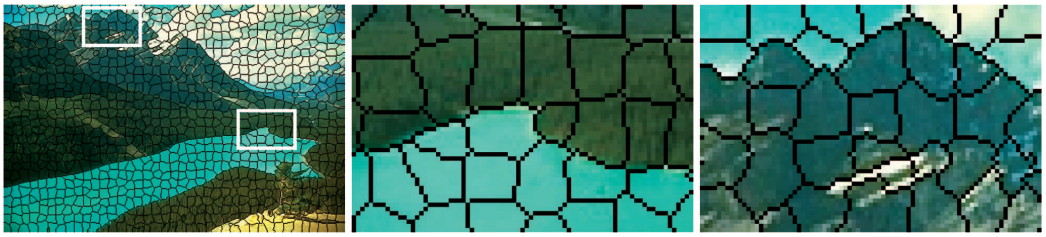


Figure 2.8: TurboPixels SP extraction example (Levinshtein [11] Figure 5). Left to right represents segmented image and its two zoomed-in version respectively

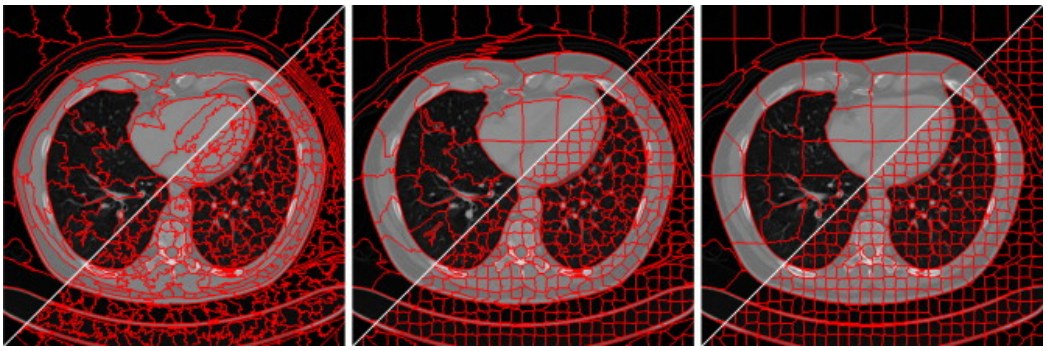


Figure 2.9: ERGC SP extraction example (Buyssens [12] Figure 5). Left to right represents segmented image in different compactness levels.

not provide a parameter to control compactness. Method shows inadequate boundary performance and generates irregular SPs with lack of smoothness. An extraction example taken from Levinshtein's [11] work is shown in Figure 2.8.

- **Eikonal-based Region Growing Clustering:** ERGC [12] starts from an initial set of seeds which are dilated iteratively. Meanwhile cuts are added or removed to refine clustering. Clustering is performed with the solution of Eikonal equation [13]. Fast-matching method is suggested to solve the equation efficiently. Method controls both number of SPs and their compactness. It demonstrates good and stable performance in terms of boundary adherence, under-segmentation error and computation time [4]. An extraction example taken from Buyssens' [12] work is shown in Figure 2.9.

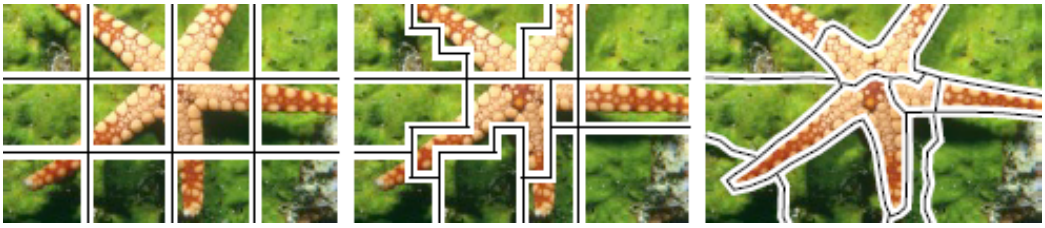


Figure 2.10: SEEDS SP extraction example (Van [14] Figure 1). Left to right represents evolution of SPs as iterations go.

2.6.2 Energy Optimization-based Methods

In these methods, image is initially clustered and clusters are iteratively reshaped to optimize an energy function. Pixel transfers between clusters are performed if energy decreases with the new form, and each transfer can contain either a single pixel or group of pixels.

State-of-the-art methods of this category are:

- **SEEDS Superpixels Extracted via Energy-Driven Sampling:** SEEDS [14] initially starts from a lattice-like clusters. Then iteratively updates regions to minimize color distribution within region while preserving the compactness. In color distribution calculation and updates, histograms are used. If movement of randomly chosen pixels or block of pixels between regions increases the energy function, new partitioning is preserved. Hill-climbing approach is employed to reduce calculation complexity. If no update in region boundaries is required, algorithm is stopped. It does not ensure connectivity across regions, post processing must be applied to connect separated regions to nearest region. An extraction example taken from Van's [14] work is shown in Figure 2.10.

- **Real-time coarse-to-fine topologically preserving segmentation:** ETPS [15] starts from an regular initial grid on which SP centers and statistics are calculated. Fine-to-grid approach is enhanced such that at first iterations, large blocks (group of pixels) are checked for label update and as iterations go on number of pixels in each transferred block decreases reaching to single pixel update at the end. Updates are performed if total energy defined by energy

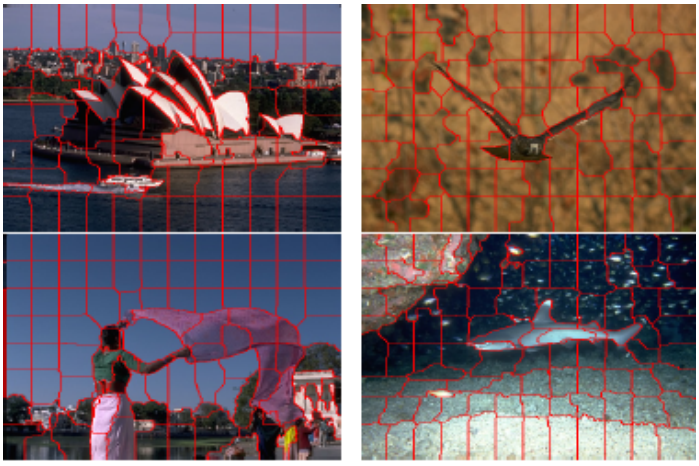


Figure 2.11: ETPS SP extraction example (Yao [15] Figure 6).

function decreases. To speed-up computations only boundary pixels/blocks are iterated. Method is the top performing one in Stutz benchmark [4], showing very good boundary adherence, low under segmentation error and consistent performance across different image sets. It provides control over number of SPs and compactness; however, generates irregular shapes. Figure 2.11 depicts an example extraction of ETPS taken from Yao's [15] work.

- **Contour-relaxed superpixel:** Contour Relaxed Superpixels [16] is a Maximum a-posteriori approach that aims to obtain maximum homogeneity of the texture inside of each cluster and as well as get the highest matching ratio of contours with image content and Gibbs-Markov random field model. In order to achieve this, an energy function is formulated with using less parameters which is maximized at each SP label update. As generated SPs are not well shaped, afterwards a shape control parameter is integrated into the formula to adjust compactness of SPs. In benchmark results of Stutz [4], CRS is one of the top performing methods in terms of boundary recall and stability across different images. However, producing highly irregular and disconnected SPs and having relatively high processing time are disadvantages of this method.Figure 2.12 depicts an example extraction of ETPS taken from Yao's [15] work.

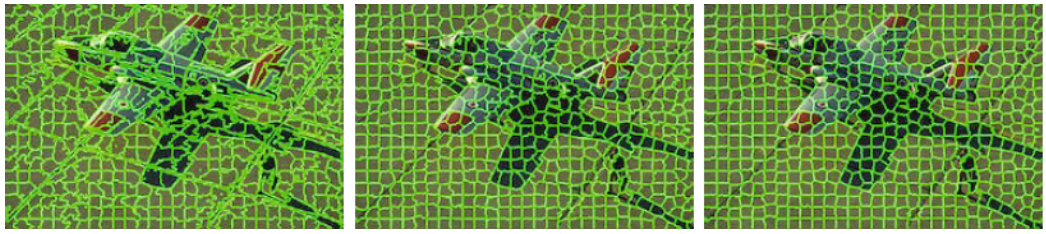


Figure 2.12: CRS SP extraction example (Conrad [16] Figure 4). Left to right represents segmented images in different compactness levels.

2.6.3 Clustering-based Methods

These methods initially cluster the image with a uniform shape, and iteratively reshapes clusters to minimize a cost function. Initial clusters can be rectangular or hexagonal (honeycomb) shaped. Centers are already assigned at this stage. Before iterations, in order to speed-up convergence a refinement operation can be applied. Refinement stage reshapes SPs to make them closer to their final shapes with a minimum amount of time. Connectivity might be broken during iterations, therefore a post-processing process might be needed to ensure connectivity.

Label updates are performed iteratively to minimize a cost function. At each iteration, pixels are assigned to the closest SPs in terms of several parameters such as color and spatial distance. Generally, the cost function to be minimized consists of spectral distance and spatial distance with adjustable weights. SP center positions and statistics are updated at the beginning of each iteration.

Color space is method dependent; however, most of them supports either RGB, LAB, or both. SP centers or SP mean statistics are used to calculate Euclidean distances for each pixel. Instead of using global definitions, some methods enhances variance of SPs to regularize spectral distances to achieve local adaptiveness.

Spatial distance calculations are used to preserve SPs compactness. Two dimensional Eulidian distances with respect to SP centers are calculated. There are also some methods specialized for images having depth information on. These methods use depth information as another dimension in the spatial domain. Compactness and spectral similarity out-weights each other, algorithms provide parameters to prioritize

Figure 2.13: SLIC SP extraction example (Achanta [2] Figure 1). Images are composed of images segmented into various number of SPs.

them called compactness parameter. High values of compactness parameter prioritize spatial distance, resulting compact SPs having less boundary adherence; on the other hand low values of this parameter, prioritize the spectral similarity resulting less compact SPs with high boundary adherence.

State-of the-art methods of this category are:

- **Simple Linear Iterative Clustering:** SLIC [2] initiates clusters using a rectangular grid, and iteratively updates clusters to achieve a set of pixels with similar spectral affinities using k-means algorithm. To speed up the k-means algorithm, grouping computations are only performed within a predefined region for each cluster. Weighted spectral and spatial distances are utilized to calculate affinities. At each iteration, pixels are assigned to the cluster that have the minimum affinity difference, and first order cluster statistics are updated. Stutz [4] states that, method generates compact SPs and provides control over compactness. Its run-time, boundary recall, under-segmentation error performances are good and stable across different images. However, method needs post-processing to ensure connectedness as the update procedure do not enforce it. Example extraction taken from Achanta's [2] is shown n Figure 2.13
- **LASP: Local Adaptive Super-Pixels:** LASP [17] segments image into regu-

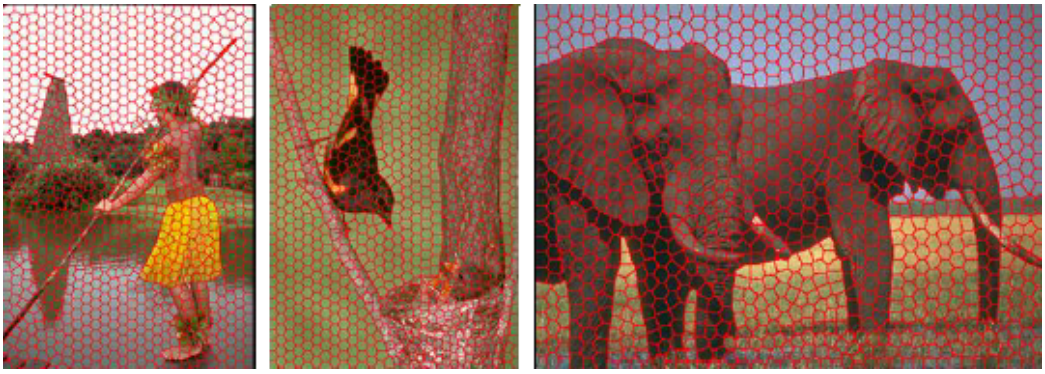


Figure 2.14: LASP SP extraction example (Ince [17] Figure 1). Images are composed of images segmented into various number of SPs.

lar shaped convex SPs having regular spectral distributions. Image is clustered initially with equally sized uniformly distributed honey-comb clusters. Before starting iterations, to preserve spectral similarity across hexagons, a predefined re-segmentation is performed. Re-segmentation chooses the best fit in terms of contrast among five different combinations of sub-clusters. To reduce complexity, iterative pixel label updates are only applied to the pixels on the boundaries of non-settled SPs. Spectral distributions (variance) of each SP is calculated and updated at each iteration. Local adaptiveness is applied by using minimum of neigbour SPs variance to regularize spectral distances. Weight factor between spectral and spatial can be adjusted to set priority. Method generates regular SPs and provides control over compactness. Due to local adaptiveness, boundary recall and under-segmentation error performance is higher than SLIC [2]. However, this method also needs post-processing to ensure connectedness. Extraction results taken from Ince's [17] work are show in 2.14

- **Speeded-up Turbo Pixels:** STP [18] is quite similar to SLIC [2]; however, as spectral distance L1 norm is employed and has a better pixel label update procedure than SLIC. Iteratively pixels are exchanged between neighboring clusters. To speed-up computation, iterations last till the count of interchanged pixels are below a threshold. Parameters of the cost function enable to adjust weight between convexity of clusters and spectral similarity of pixels within clusters. Run-time, boundary recall, under-segmentation error performances are good, generates compact, convex and connected SPs. This method also needs post-

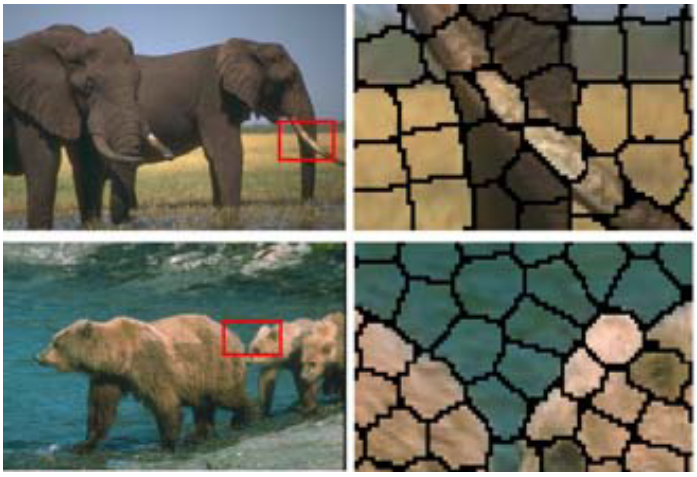


Figure 2.15: STP SP extraction example (Çla [18] Figure 2). Left to right represents original image and zoomed in version segmented image. Images are composed of images segmented into various number of SPs.

processing to ensure connectedness. STP extraction results taken from Çla [18] can be seen in Figure 2.15.

- **A Fast method for inferring high-quality simply-connected superpixels:** SCS [19] improves related methods that are using Gaussian Mixture Model by enabling them to be parallel processed and replacing fixed spatial covariances by Bayesian prior. It aims to guarantee connectivity across regions without post-processing by ensuring clusters stay simply-connected during label updates. Initialization is done clustering the image with a honey-comb. At the first step of iteration, parameters (statistics) are updated where Freifeld [19] claims that parallel processing is better for small number of SPs. At the second step labeling is performed with minimizing a cost function based on Gaussian Mixture Model. During label updates, for each pixel the neighboring 3x3 block is checked to ensure connectivity. This approach results in the dependency of label updates in 3x3 neighborhood. To speed-up the method parallel processing is enabled using 3 pixels steps on horizontal and vertical directions. Compactness and number of SPs can be controlled. Method generates connected clusters, therefore no post-processing is needed. Figure 2.16 shows results of SP extraction taken from Freifeld's [19] work.

- **Depth-Adaptive SuperPixels:** DASP [20] exploits depth information in ad-

Figure 2.16: SCS SP extraction example (Freifeld [19] Figure 1).

dition to color. Hence, it can only be applied to images having depth information such as RGB-D images. Pixel positions in 2D are transferred to 3D surface points using depth information, and then cluster centers are assigned such that density of SPs is increased with increasing distance from camera and inclination angle with image surface. During over-segmentation to decide label update, k-means algorithm is applied where Euclidean-distances are calculated in terms of 1) point normal angles, 2) color space values and 3) distance to the cluster centers. To speed-up label assignment calculations, comparison window can be limited to a predefined region as done in SLIC [2]. According to Stutz [4], method presents similar performance with SLIC [2] except stability, i.e. performance is not stable across different images. Method enables to control compactness and the number of SPs; needs post-processing to ensure connectedness but generates many SPs at post-processing. Stutz [4] concludes that regarding methods enhancing depth information, DASP performance is better than VCCS [21]. Figure 2.17 depicts results of SP extraction taken from Weikersdorfer's [20] work.

- **VCells: Simple and efficient superpixels using edge-weighted centroidal Voronoi tessellations:** VC [22] initially tiles image with hexagon-like shaped clusters using Lloyd's algorithm [23]. Then, at each iteration assigns each pixel to nearest cluster in terms of color and spatial distances to cluster centers. Cost function encourages pixels to take the label of a SP that the majority of their neighbor pixels within a predefined distance are assigned to. Method produces

Figure 2.17: DASP SP extraction example (Weikersdorfer, [20] Figure 1). Left to right represents original and segmented image respectively.

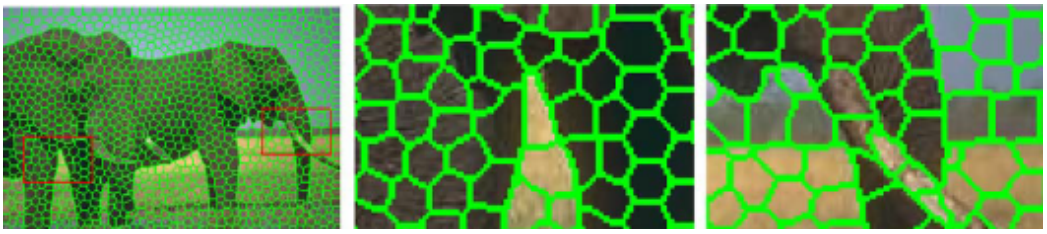


Figure 2.18: VC SP extraction example (Wang, [22] Figure 8). Left to right represents segmented image and its two zoomed-in version respectively.

compact SPs, and gives control over compactness and the number of SPs. Its performance is not stable across different images and processing time is relatively long. It generates many SPs when connectivity is enforced [4]. Example results of VC taken from [22] are depicted in Figure 2.18.

- **Voxel cloud connectivity segmentation-supervoxels for point clouds:** VCSS [21] takes depth information of RGB-D images into account for over-segmentation. Works on 3D space (x, y, depth), setups a voxel grid and selects initial cluster centers that are closest to voxel grid centers. At each iteration, pixels get label of the closest cluster in terms of weighted color (LAB), spatial (x, y, depth) and geometrical local surface similarities. Compactness is adjustable; however, number of generated SPs cannot be controlled. Weikersdorfer [20] states that, VCSS performance is worse than the other depth-based method DASP [20] and it does not follow the surface as well as DASP does. Moreover post-processing for connectivity results in many SPs. Figure 2.19 shows example VCCS extraction results take from Weikersdorfer's [21].

- **Compact watershed and preemptive SLIC:** PreSLIC[24], makes SLIC al-

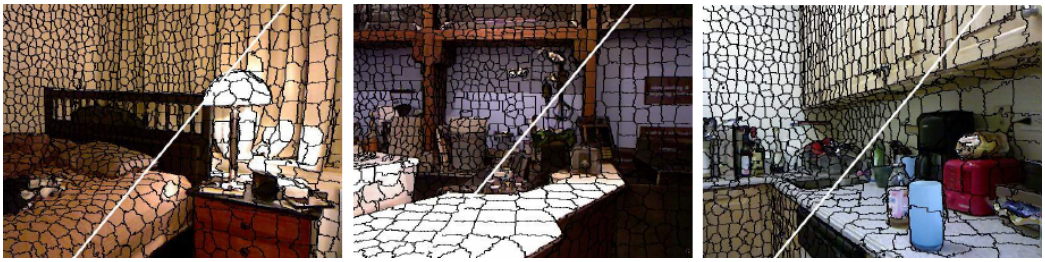


Figure 2.19: VCSS SP extraction example (Papon, [21] Figure 6)

Figure 2.20: PreSLIC SP extraction example (Neubert, [24] Figure 2).

gorithm faster by employing local termination criteria for each SP that stops evolution boundaries in homogeneous regions earlier. Its boundary recall and under-segmentation error performance is slightly worse than SLIC but outperforms it in runtime performance achieving real-time processing capability [4]. PreSLIC sample results taken from Neuberts' [24] work can be seen in Figure 2.20

- **Linear spectral clustering superpixel:** LSC [25] adopts N-Cuts algorithm and combine its objective function with the objective function of K-means. Method maps each pixel to a 10-dimensional space by transforming weighted 5-dimensional spectral (CIELAB color space l,a,b) and spatial (x, y) space. Clusters are initiated using a regular rectangular grid. Then, each pixel is assigned to the closest cluster in aforementioned 10-dimensional space. At the end of each iteration, cluster statistics are updated. To speed-up computations, method suggests limiting windows for cluster membership updates. According to Stutz [4], LSC offers control over SP number and compactness; however, generates irregular SPs and exceeds the number of required SPs when post-processing for connectedness is applied. Its performance is unstable on different images and generated SPs are irregular, i.e. generated SPs varies in

Figure 2.21: LSC SP extraction example (Chen, [25] Figure 2). Images are composed of images segmented into various number of SPs.

size much and their arrangements within image do not represent a regular grid structure. Figure 2.21 depicts sample results taken from Chen' [25] work.

2.7 Summary and Discussion

In Table 2.1, the type, number of configuration parameters, status of control over number of SPs and compactness of state-of-the-art SP extraction methods are presented for the examined SP extraction methods.

Among different SP extraction categories, clustering-based approach effectively satisfies SP extraction requirements; they present high boundary following and semantic object performance with almost real-time performance. These techniques provide control over number of SPs and compactness. Furthermore, due to their ease of implementation and wide range of usage, the clustering-based methods are selected to be analyzed in detail and improved in the upcoming chapters of this thesis. Starting from a general clustering-based SP extraction method, every step of this generic method will be examined to obtain a better approach.

Table 2.1: List of state of art Superpixel extraction methods

Method	Type	Number Of Parameters	Number of SuperPixel Control	Compactness Control
SLIC [2]	Clustering-based	2	✓	✓
LASP [17]	Clustering-based	2	✓	✓
STP [18]	Clustering-based	2	✓	✓
SCS [19]	Clustering-based	2	✓	✓
DASP [20]	Clustering-based	5	✓	✓
VC [22]	Clustering-based	6	✓	✓
VCCS [21]	Clustering-based	4	-	✓
PreSLIC [24]	Clustering-based	4	✓	✓
LSC [25]	Clustering-based	4	✓	✓
TP [11]	Contour evolution	4	✓	-
ERGC [12]	Contour evolution	3	✓	✓
SEEDS [14]	Energy optimization	6	✓	-
ETPS [15]	Energy optimization	5	✓	✓
CRS [16]	Energy-optimization	4	✓	✓
W [5]	Watershed-based	1	✓	-
EAMS [6]	Density-based	2	-	-
NC [8]	Graph-based	3	✓	-
ERS [9]	Graph-based	3	✓	-
PF [7]	Path-based	2	✓	-
SEAW [10]	Wavelet-based	3	-	-

CHAPTER 3

SUPERPIXEL PERFORMANCE METRICS AND PERFORMANCE EVALUATION

SPs are specialized subset of over-segmentation having their own requirements. Extracting SPs is a pre-processing step for user applications that decreases their number of computations while increasing accuracy. However, the methods in literature have different characteristics. In order to be able to compare their performance, precise performance metrics and a test benchmark is needed. This benchmark should measure how well methods satisfy SP requirements. Therefore, the benchmark contains several metrics that unveils method performance such as accuracy and run-time for the benefit of user applications. When these metrics are chosen from the ones commonly used in literature, results become globally acceptable. However, not all of the common metrics give results effectively. There have been proposals of improving existing ones. This study are investigating SP extraction methods and proposing new ones. During the study, in order to observe the affect of proposed improvements and compare proposed SP extraction methods with the literature ones, a benchmarking is presented. This chapter defines benchmark used in this study, by first defining metrics used and then explaining evaluation methods.

3.1 Superpixel Performance Metrics

Several metrics are used to evaluate performance of Superpixel (SP) extraction methods. Metrics like Boundary recall (BR), Under-segmentation Error (UE), Achievable Segmentation Accuracy (ASA) measure how well the generated SP represent the objects in the image. In order to measure this object-based representation, ground-

truth images that label each object with a unique id are used. Examples of commonly used data sets for benchmarking are The Berkeley Segmentation Data Sets (BSDS300, BSDS500), The Stanford Background Dataset (SBD), The NYU Depth Dataset (NYUV1, NYUV2), The SUN RGB-D Dataset (SUNRGBD) and The Fashionista Dataset (Fash), which all provide ground-truth images.

Other metrics look from the structural point of view without relying on ground-truth. Since SP generation is a pre-process that significantly decreases computational complexity of succeeding algorithms, resultant SP structure should be used effectively by those algorithms. This kind of metrics gives an idea about how well the generated SP can be utilized by the following processes by measuring explained variation (EV), compactness and regularity etc.

The metrics relying on ground-truth segments, evaluate performance based on the idea that each SP should belong to a single object in the ground-truth image. Hence, each SP is assigned to a single object and metrics penalize method for the mislabeled pixels.

Following sections describe the metrics used to evaluate performance of SP extraction methods. At the end, a new metric is proposed that cover boundary recall, under-segmentation error and achievable segmentation accuracy.

3.1.1 Boundary Recall

Boundary Recall (BR), one of the most commonly used metrics, shows SP adherence to object boundaries. It measures the proportion of ground-truth boundaries overlapping with SP boundaries. Let S be SP boundaries and G be ground-truth boundaries. Boundary Recall (Rec) is calculated as follows [26]:

$$Rec(G, S) = \frac{\sum G \cap S}{\sum G} \tag{3.1}$$

For this metric, higher values mean better boundary adherence. One of the major drawbacks of this metric is over-penalizing slight boundary following errors. The other drawback is, it also fails to distinguish SP extraction method performance well. As an example, in Figure 3.1 BR of proposed alternatives in Section 5.1 are pre-

sented, there is less than 1% difference between proposed alternatives for the same compactness parameter.

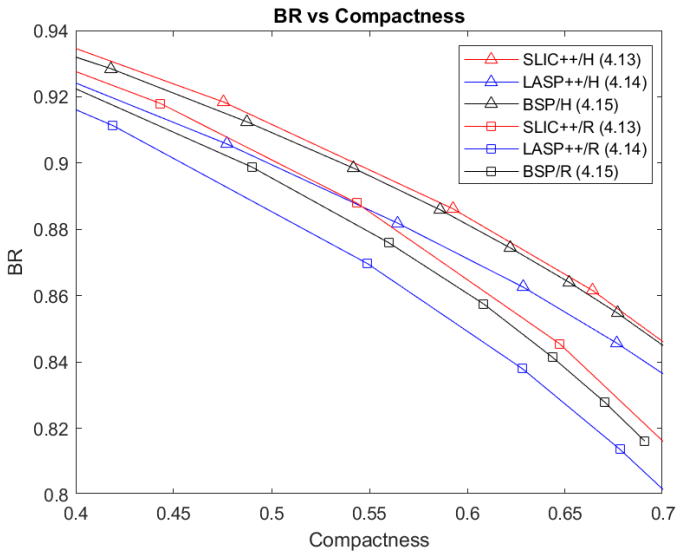


Figure 3.1: Boundary recall metric example, proposed methods generate 1000 SPs

3.1.2 Undersegmentation Error

Undersegmentation error (UE) measure the overlap of SP with multiple ground-truth segments. It is also referred as leakage or bleeding error. It also gives a notion about boundary recall. Error becomes higher when SPs overlap (leak) more with other objects. Hence, lower values are better. Levinshtein [11] was the first to use this metric as a benchmark and the related equation is as follows:

$$UE_{Levin}(G,S) = \frac{1}{|G|}\sum_{G_i}\frac{(\sum_{S_j\cap G_i\neq\emptyset}|S_j|)-|G_i|}{|G_i|} \tag{3.2}$$

For this metric, each ground-truth object is evaluated one by one. For each object, SPs intersecting with ground-truth object are assigned to that object, and the rate of mis-assigned pixels are obtained.

However, Achanta [2] argues that former equation of Levinshtein [11] unfairly penalizes clusters that leaked slightly and suggested an improvement. In order to tolerate small amount of leakages, Achanta adds an overlap threshold to former equation, such that only SP that at least 5% of pixels are mis-overlapped are considered as leakage.

In order to overcome drawbacks of the aforementioned metrics, Van den Bergh [14] proposed a new approach for measuring UE. SPs are assigned to their largest overlap ground-truth clusters and leakage with respect to other clusters is taken into account. With this approach, UE metric becomes complementary to Achievable Segmentation Accuracy (ASA) metric explained in following section (UE = 1 - ASA). Van den Bergh [14] UE equation is as follows:

$$UE_{Bergh}(G,S) = \frac{1}{N}\sum_{S_j} |S_j - \arg\max_{G_i} |S_j \cap G_i|| \tag{3.3}$$

For each SP, leakage is calculated by getting the difference between SP area and its highest overlap area with a ground-truth cluster. Calculated leakages are accumulated and divided to SP count.

Neubert and Protzel [24] proposed another metric for UE. In this approach, minimum of overlap and non-overlapped region of each SP and ground-truth segment is considered as leakage. SPs are not assigned to its largest overlapping segments as Bergh [14] proposed; instead for each SP larger part among overlapping and non-overlapping parts with each segment is considered to be assigned to that segment and remaining one is calculated as error.

$$UE_{NP}(G,S) = \frac{1}{N}\sum_{G_i}\sum_{S_j \cap G_i \neq \emptyset} \min\{|S_j \cap G_i|, |S_j - G_i|\} \tag{3.4}$$

Error is accumulated by calculating it for each combination of SP and ground-truth segment and divided to SP count.

During the experiments throughout this thesis, UE metric 3.4 proposed by Neubert and Protzel (NP) is used. One drawback of UE is, it double penalizes mislabel assignments. The other one is, like BR, it is not adequate to distinguish SP extraction method performance well. As an example, in Figure 3.2 UE of proposed alternatives in Section 5.1 are presented, there is less than 1% difference between proposed alternatives for the same compactness.

3.1.3 Achievable Segmentation Accuracy

Achievable Segmentation Accuracy (ASA) [9] shows the highest achievable accuracy when SPs are used for object segmentation. SPs are assigned to ground-truth seg-

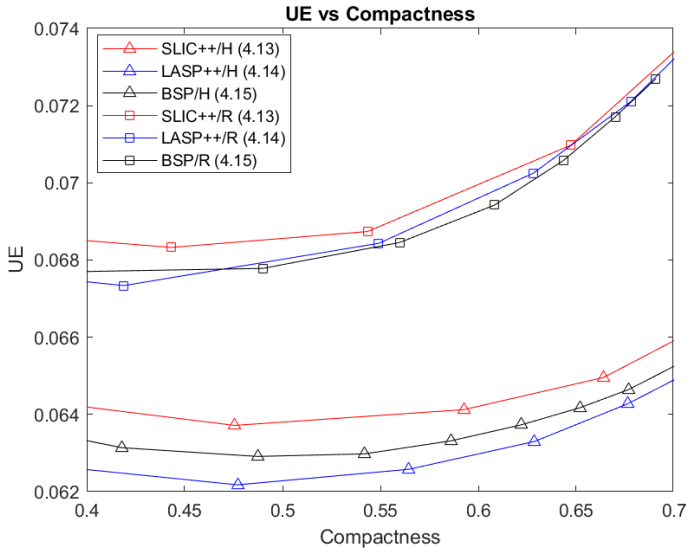


Figure 3.2: Undersegmentation error metric example, proposed methods generate 1000 SPs

ments with largest overlap. Then, fraction of mis-assigned pixels to all pixels gives ASA showing the ratio of correctly labeled pixels. For this metric, higher values are better.

$$ASA(G,S) = \frac{1}{|G|}\sum_{j}\max_{G_i}\{|S_j \cap G_i|\} \tag{3.5}$$

Drawback of this metric is being SP number dependent. As SP number increase metric gives better results, which means 100% success can be achieved when SP count is equal to pixel count. On the other hand, similar to UE, ASA is yet to distinguish SP extraction method performance well. As an example, in Figure 3.3 ASA of proposed alternatives in Section 5.1 are presented, there is less than 1% difference between proposed alternatives for the same compactness.

3.1.4 Explained Variation

As ground-truth images are labeled by human operators, the process is hard and prone to error. Explained variation (EV) proposed by Moore [27] removes this human dependency by providing a metric that does not rely on ground-truth images. Metric

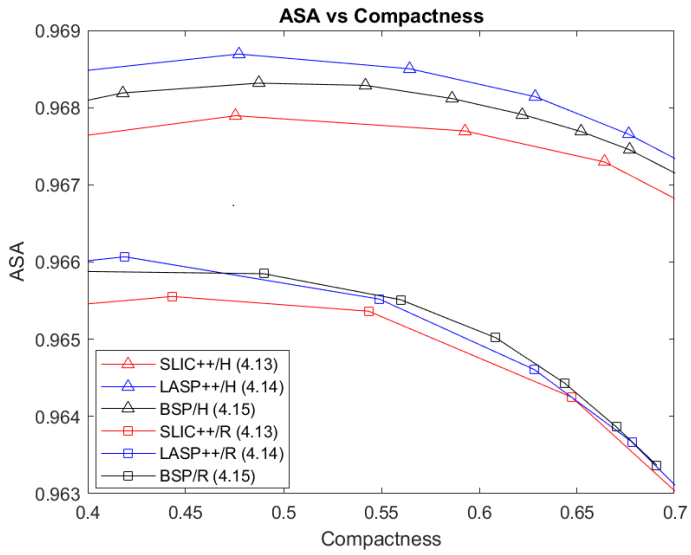


Figure 3.3: Achievable segmentation error metric example, proposed methods generate 1000 SPs

exploits high color variation on object boundaries to show boundary precision using image only data. EV is calculated as:

$$EV(S) = \frac{\sum_{S_j} |S_j|(\mu(S_j) - \mu(I))^2}{\sum_{x_n}(I(x_n) - \mu(I))^2} \tag{3.6}$$

where $\mu(S_j)$ and $\mu(I)$ are color mean of SP S_j and image I respectively. In (3.6), total color variation of SPs with respect to global mean is divided into total color variation of pixels with respect to global image mean. As a result, metric shows the fraction of image variation that is explained when the detail is removed. SPs with high spectral variances are penalized by the metric; but this effect decreases by increasing number of SPs. EV takes values between 0 and 1 where higher values indicates better performance.

This metric is quite useful when no ground truth image exists. However, data set used throughout this work contains ground truth images. Hence this metric becomes redundant, and is not used in this thesis.

3.1.5 Proposed Metric : Boundary Achievable Segmentation Accuracy

Achievable Segmentation Accuracy (ASA) gives the amount of correctly labeled pixels with respect to ground-truth segment. However, only SPs located on object boundaries can contain mislabeled pixels, SPs far from object boundaries usually fully overlap with a single ground-truth segment. Since the ratio of boundary SPs is significantly less than non-boundary ones; non-boundary SPs dominate this metric which makes the metric fail to distinguish the performance of segmentation algorithms well. Moreover as mentioned earlier, ASA is dependent to number of SPs, as the number of SPs increase, ratio of boundary SPs decreases and ASA increases.

The other metrics BR and UE are not fair either. While BR over-penalize slight boundary errors, UE doubly penalize segmentation error. All BR, UE and ASA are insufficient to evaluate performance of different SP extraction methods clearly.

In order to overcome those drawbacks, an alternative metric is proposed, namely *Boundary Achievable Segmentation Accuracy* (BASA). Unlike ASA that takes into account SPs without categorization, BASA only considers the SPs close to object boundaries, and calculates ASA on the regions close to object boundaries. By getting rid of non-boundary domination, metric allows to compare segmentation accuracy of different SP generation methods more effectively.

In order to calculate BASA, first boundaries on ground-truth image are dilated with a circle having radius proportional to average SP area, and then achievable segmentation accuracy for this dilated region is calculated. For this metric, higher values indicate better segmentation accuracy. BASA is obtained as follows:

$$BASA(G^r, S) = \frac{\sum_j \max_{G_i^r}\{|S_j \cap G_i^r|\}}{\sum_{|S_j \cap G^r|>0} |S_j|} \tag{3.7}$$

where j is SP index, i is object index, and G^r is the masked ground-truth image. The mask is the dilated ground-truth boundary image with a circular dilation element having radius r, and r is calculated as $\sqrt{SP_{area}/\pi}$.

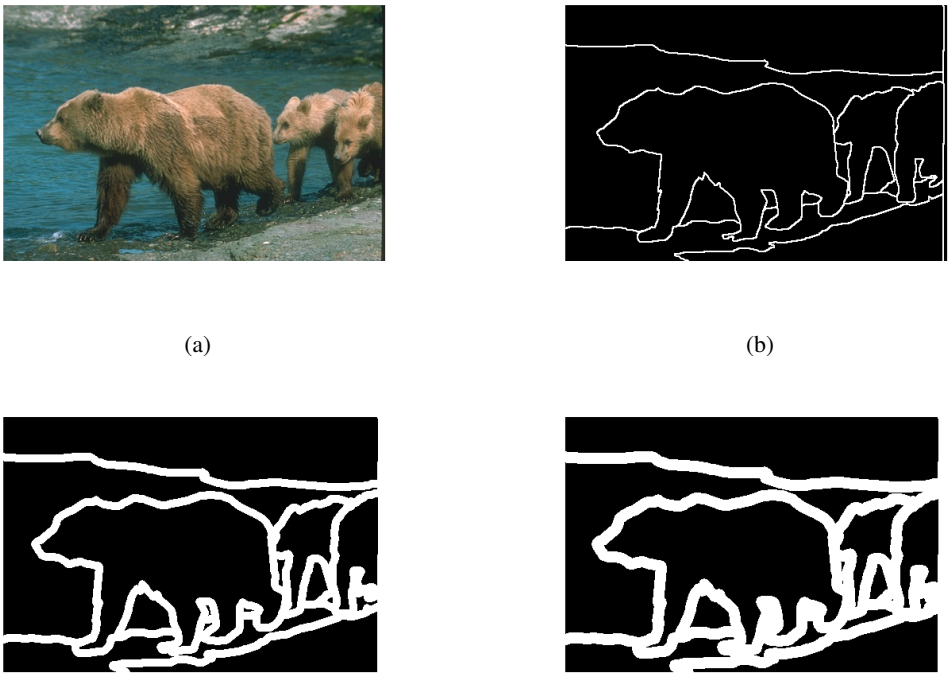
Figure 3.4 depicts BASA evaluation mask on an example for 500 SPs and 1000 SPs. As shown in figure, a decrease in SP number results an increase in mask size due to an increase in area of SPs.

In literature UE, BR and ASA performance of SP extraction methods are usually presented together. However, while comparing two different methods, one can perform better in BR and worse in UE which makes hard to make a conclusion. However, BASA makes it possible to evaluate and compare different methods with a single distinctive metric, instead of non-distinctive three metrics. Since the evaluation mask is obtained with a circular dilation proportional to average SP area, as the number of SPs increases the evaluation mask is getting smaller and even small segmentation errors become visible. This approach allows BASA to evaluate the performance independent from number of SPs.

Figure 3.5 depicts BASA for the proposed alternatives in Section 4 as an example. Besides being a fair metric, BASA distinguishes performance better than the others as this can be clearly observed from the figure. The gap between curves is larger than the ones for BR, UE and ASA which are shown in Figures 3.1, 3.2 and 3.3, respectively. Performance of alternative methods has the same order for all four metrics, meaning they all give the same decision about ranking of methods.

3.2 Superpixel Performance Evaluation

Metrics unveil performance of SP extraction methods from several point of views such as object boundary following accuracy and run-time. In order to achieve an effective comparison, evaluation of the measurement results needs to be achieved comprehensively. Otherwise, ambiguity arises for the results that makes them unusable. Although the metrics are selected from the globally accepted ones, current evaluation of the results are benchmark dependent, as they are generally based on number of generated SPs and method hyper-parameters. However, method dependent evaluation prevents achieving a fair comparison and causes misinterpreted results. This section defines evaluation approach of this study that gets SP extraction experiment results evaluated effectively and also gets rid of ambiguity in literature benchmarks. First *Compactness* to be used as metric base parameter is explained, and then Area-under-curve approach is proposed to make evaluation independent of hyper-parameters.



(a) (b)

(c) (d)

Figure 3.4: BASA mask on an example; a) original image, b) ground-truth object boundaries, c) BASA mask for 1000 SPs d) BASA mask for 500 SPs

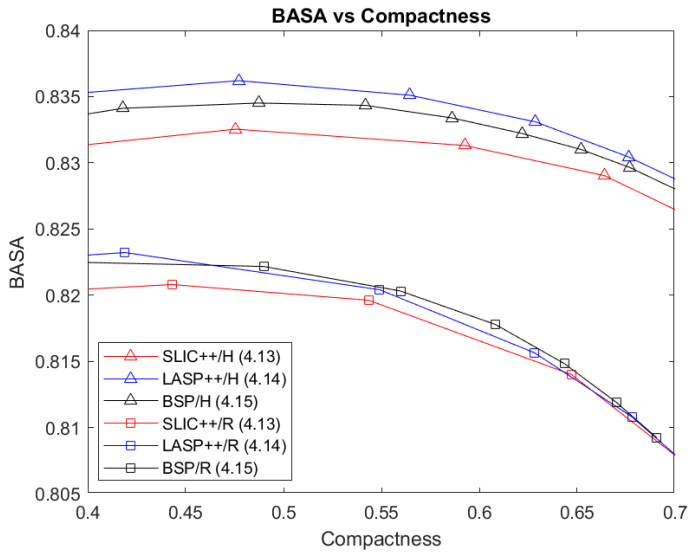


Figure 3.5: Proposed boundary achievable segmentation error for proposed alternatives

3.2.1 Compactness Parameter

Compactness (CO) metric [28] gives information about SP shapes by measuring their closeness to a circle. It is also denoted as circularity measure. As the circle is the most compact shape having a maximum achievable area for a given boundary length; higher values of compactness corresponds to shapes closer to circle, and for a perfect circle compactness score is 1.

SP extraction methods are used as a pre-processing step for image processing applications to decrease complexity of image data. Many of those applications require compact clusters to process them efficiently. Hence, the methods proving better compactness while preserving boundary adherence or achievable segmentation accuracy could be a better choice for those applications. However, shape of generated SPs strongly dependent to approach of the method. Moreover some methods provides compactness parameter to adjust compactness of clusters waiving spectral similarity and boundary performance. Therefore, different SP extraction methods cannot be fairly compared without considering compactness parameter.

Stutz [4], [16], variate compactness hyper-parameter of various methods [14], [15],

[17], [19] to take compactness into account in comparison, but not the compactness of SPs itself. Since compactness hyper-parameter is method dependent, it does not have a common definition. This parameter is used for adjusting compactness of the resulting SPs. In this thesis, compactness of SPs is the target, parameter is a tool to achieve it. Hence, instead of relying on method dependent parameter, directly measuring the compactness generated SPs is a better approach. Compactness is a common metric, independent from method and image. When compactness is used as base component of other metrics, it helps to display performance of methods within a wide range compactness levels. In this manner, applications employing SP extraction methods can easily select fitting method depending on desired compactness level. Due to the reasons mentioned above, in this work, compactness is selected as base of other performance metrics.

CO compares area of each SP with the area of circle. Hence, metric compares area of all SPs with area of circle and has a range of [0-1] where higher values indicate a better compactness. Compactness is defined as follows:

$$CO(S) = \frac{1}{A(I)} \sum_{S_j} |S_j| \frac{4\pi A(S_j)}{P(S_j)^2} \tag{3.8}$$

where $A(I)$ and $A(S_j)$ are area of image and SP S_j respectively and $P(S_j)$ is perimeter of SP S_j.

However due to discrete nature of images, perimeter measurement of SPs can differ from actual one which results misleading compactness evaluations. This error between measured and actual one increases when size of SPs get smaller due to increase in dominance of boundary pixels. Since compactness is an evaluation method it needs to be measured accurately for effective comparisons. There are several methods in literature for measuring perimeter of objects having discrete nature. In Appendix A, measurement accuracy of these methods are examined by conducting experiments and then the most accurate one is selected to be used in compactness measurements of generated SPs throughout this study.

3.2.2 Proposed Evaluation Method: Area Under Curve

Most of oversegmentation methods have hyper-parameters which enables to adjust compactness of generated SPs. The reason why SP number is kept constant during comparative evaluations in literature is same for compactness which should be kept constant as well, as with the increasing number of SPs or decreasing compactness usually increase the performance. Figure 3.6 depicts ASA vs compactness performance of two state of the art methods SLIC [2] and ERGC [12] that are calculated during experiments in Chapter 5. It cannot be stated easily which method perform better by just analyzing the graph. Additionally, without measuring the performance without compactness, depending on chosen hyper-parameters one method may easily be presented as better than other, which is not the case as shown in Figure 3.6.

Moreover, the researchers usually select hyper-parameters for a specific data set which might give completely different results in other data sets. As a result, performance of methods should be measured for entire range of hyper-parameters.

In order to completely evaluate performance of SP extraction methods, in this work measuring Area Under Curve (AUC) method is proposed. To enhance AUC, experiments are conducted for various set of values of hyper-parameters, and area under the curve of desired metrics are calculated for a range of compactness to achieve a fair comparison. In machine learning, AUC is used together with Receiver Operating Characteristics (ROC) [29] for the evaluation of signal (or object) detection performance. ROC represents performance of a detection result at all thresholds while AUC of ROC represents overall performance across all possible thresholds measuring degree of separability.

In this work, AUC of UE, BR, ASA, BASA and run-time with respect to compactness are evaluated. As shown in Figure 3.6, it is hard to make a decision here by analyzing the graph, since for specific compactness values one method can outperform the other and vice versa. However, calculating AUC in the required range of compactness clearly indicates the better performing method. For the example shown in Figure 3.6, BASA AUC values of ERGC [12] and SLIC [2] is 80.7% and SLIC is 81.3%, respectively. Hence, it can be concluded that SLIC has a better BASA performance

than ERGC. For the rest of this work, both raw graphics of various performance metrics and the area under curve for the selected compactness range will be presented for a fair comparison.

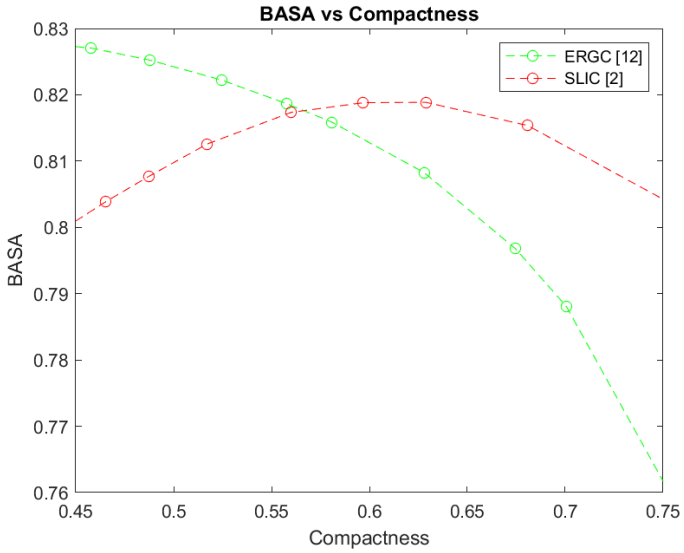


Figure 3.6: Area under curve evaluation example. BASA AUC values of ERGC [12] and SLIC [2] is 80.7% and SLIC is 81.3%, respectively

CHAPTER 4

ANALYSIS OF A GENERAL CLUSTERING-BASED SUPERPIXEL EXTRACTION METHOD

Clustering-based SP extraction methods are widely used due to their high performance with a low response time. They solve a cost-function iteratively where each iteration pixels are assigned to closest cluster. In this chapter, clustering-based method framework will be extracted, alternatives to steps of this framework will be investigated in detail, alternative approaches will be proposed and performance of the alternatives will be analyzed.

4.1 A General Clustering-based Superpixel Extraction Method

A general clustering-based SP extraction method initially clusters image into regularly distributed and compact SPs. Then, before advancing to iterations as an optional pre-processing step, this grid may be refined further to ease convergence. After that, iterations takes place. SP statistics are calculated and based on these statistics, cost-function to be minimized and other constraints like connectivity, pixels are assigned to pixel to SP assignments are updated iteratively. Iterations may last till all SPs converge or number of iterations may be a pre-defined hyper-parameter. After iterations phase, an optional connectivity post-processing phase takes place for methods produce disconnected clusters.

Algorithm 1 shows a general flow of a clustering-based SP extraction method. In following sections, all steps are analyzed in detail.

Algorithm 1 A General Clustering-based Superpixel Extraction Method

1. Initial tiling: Initial tiling is performed and cluster centers are assigned. Rectangle or hexagon is chosen as cluster shape.

2. Refinement of initial tiling (optional): To start with better cluster statistics, initial grid may be refined.

3. Iterations:

for i in $MaxIteration$ **do**

for all S in $SPList$ **do**

4. Statistics update: First and second order SP statistics are calculated.

end for

for all (x, y) in $Image$ **do**

5. Label update: Pixel at (x, y) is assigned to its closest neighbor cluster in terms of spectral and spatial distance to cluster estimated statistics. A cost-function is minimized while updates. There may be additional criteria such as connectivity enforcement, to enable label update.

end for

end for

6. Post-processing (optional): If a method produce disconnected clusters, a post-processing step is required to obtain connected clusters. Depending on the post-processing algorithm, either new clusters are generated for disconnected clusters or they can be assigned to a neighbor cluster. This method may break regular grid structure when new SPs are generated.

4.2 Initial Tiling

Due to the nature of clustering-based extraction methods, an initial grid is formed at the beginning and clusters are reshaped based on a cost function iteratively. At the initial tiling step, each pixel is assigned to a cluster which is shaped according to the selected tiling method. In order to form initial tiling, clusters can be selected either rectangular or hexagonal shaped. Tiling type has effect on SP extraction performance which is analyzed in the following sections. Advantages of using rectangular grid is its low computational cost over hexagonal one. On the other hand, since the performance is evaluated with respect to compactness, starting with a shape close to circle which is hexagon, leads to early convergence and more compact SPs with a better accuracy.

Figure 4.1 depicts rectangular and hexagonal tiling example. Due to its nature, hexagonal tiling has advantage over rectangular tiling by starting with a higher compactness value. Therefore hexagonal tiling gives significantly better boundary adherence than rectangular one for the same compactness. However, both tiling alternatives have different usage areas that makes it no sense to compare their performance. In other words, if the generated SPs are going to be fed into graph based processing applications, hexagonal tiling should be used; on the other hand, rectangular tiling has an advantage such that SPs can form a regular rectangular grid, which can be treated as an image on which all computer vision methods can be applied easily especially the neural network applications. Hence, in order to track performance of the two independently, throughout this work, experiments are conducted by both starting with hexagonal and rectangular tiling.

4.3 Cluster Connectivity

Both initial tiling options results in initially connected clusters. However preserving cluster connectivity is a major issue for SP extraction algorithms. Due to definition of SPs, every pixel in an SP (or cluster) should be connected to other pixels in the SP. For methods generating disconnected clusters, an additional post-processing step should be applied to form connected clusters and obtain SPs. Mentioned post-processing

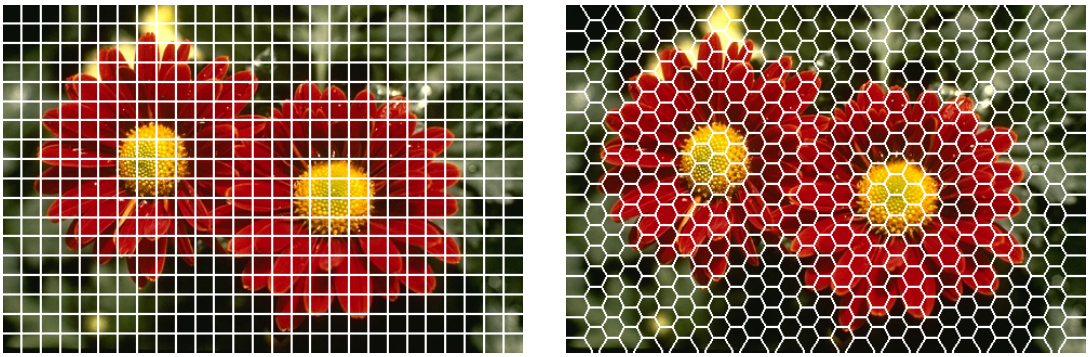


Figure 4.1: Initial tiling example; rectangular tiling on the left side, hexagonal tiling on the right

step should either connect disconnected pixels of a cluster to closest and connected neighbor clusters or create new SPs from them.

A cluster is connected and therefore can be called as an SP, if for each member pixel of that cluster there exists at least one within cluster path to rest of the member pixels. In other words, a cluster can form an SP if there is no disconnectivity within a cluster, which makes necessary every pixel of that cluster can be accessible by staying inside that cluster. Figure 4.2 a and b shows connected and disconnected clusters respectively.

0	0	0	0	0	0
0	1	1	1	0	0
0	0	1	1	0	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	0	1	1	0

a)

0	0	0	0	0	0
0	1	1	1	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	0	1	1	0

b)

Figure 4.2: Connected cluster example; a) a connected cluster, b) a disconnected cluster

For a 2 dimensional image, a pixel represented by x and y has 8 neighbors located at $(x \pm 1, y \pm 1), (x, y \pm 1), (x \pm 1, y)$ positions. Depending on the relation with neighbors, within the context of this work 2 types of connectivity are observed; 4-connected

and 8-connected.

4.3.1 4 or 8-Connected Update

4-connected pixels are the ones that touch to the neighbor pixels' edge. 4 out of 8 neighbors are 4-connected and located at $(x, y+1), (x+, y), (x-1, y), (x, y-1)$ positions. If at least one of 4-connected neighbors belong to same cluster, that pixel is 4-connected to that cluster. If every pixel of a cluster is 4-connected to that cluster and there is no discontinuity within the cluster, the cluster becomes 4-connected. Figure 4.3 shows examples of 4-connected pixel and region. Foreground pixels are labeled 1, backgrounds are labeled 0.

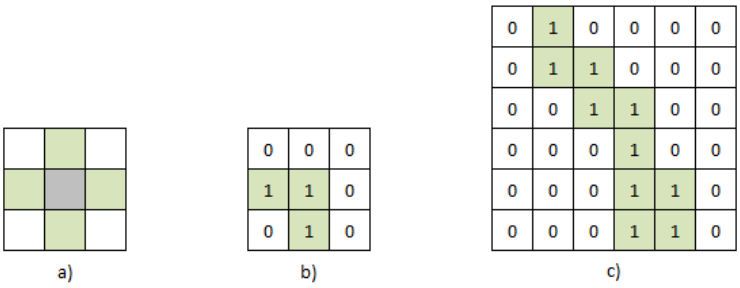


Figure 4.3: 4-connected examples; a) 4-connected neighbors, b) 4-connected pixel to a region, c) 4-connected region

8-connected pixels are the ones that touch to the neighbor pixels' edge or corner. So all possible neighbors of every pixel are 8-connected to them for an 2-D image. If at least one of those neighbors belong to same cluster, that pixel is called 8-connected to that cluster. In addition, if every pixel of a cluster is 8-connected to that cluster and there is no discontinuity within that cluster, the cluster becomes 8-connected. 4 connectivity is additional restricted subset of 8-connectivity in a way that, if pixels and clusters are 4-connected, they are also 8-connected. Figure 4.4 shows examples of 4-connected pixel and region. Foreground pixels are labeled 1, backgrounds are labeled 0.

At the label update step of SP extraction methods, pixels take SP labels minimizing a cost function iteratively. Pixels are only able to take values of their neighboring

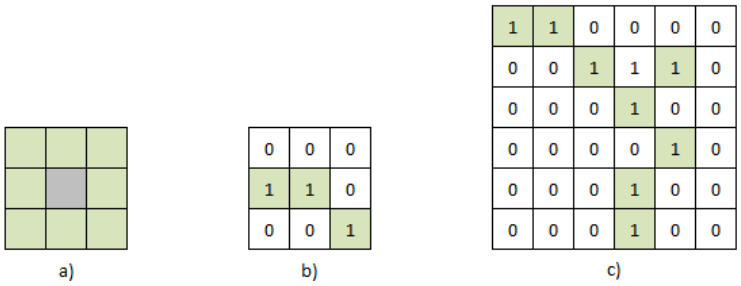


Figure 4.4: 8-connected examples; a) 8-connected neighbors, b) 8-connected pixel to a region, c) 8-connected region

clusters. So that at each iteration for each pixel, distance to their neighbor clusters are calculated and the minimum one is selected. At this point, selection of neighbor pixels determines the connectivity of the clusters. SP extraction methods in literature choose either 4-connected or 8-connected update, that means pixel can get value of their either 4-connected or 8-connected neighbors respectively. However this type of update may break connectivity within a cluster.

Figure 4.5 shows two examples of label update scenario of initially 4-connected pixel to region. Pixels belonging to the SP that the center pixels belongs are shown with ones and the other labels are shown with zeros. In a), changing the label of the center pixel, causes foreground pixels disconnect from each other in terms of 4-connectedness. This update will break connectivity of the foreground cluster. In b), update is safe since foreground pixels remains 4-connected and background is either. In c), break of connectedness due to label update of a pixel is shown in a wide view. Initially foreground pixels are 4-connected, however when the marked pixel gets a label from background clusters, foreground pixels become disconnected and as a result two foreground regions are formed in terms of 4-connectivity.

Figure 4.6 shows two examples of label update scenario of initially 8-connected pixel to region. In a), update may result a disconnected regions. In b), safe update as foreground and background pixels remain 8-connected. In c), example of breaking connectivity with in cluster after label update. After update of marked foreground pixel, foreground cluster split into three clusters and becomes a disconnected cluster.

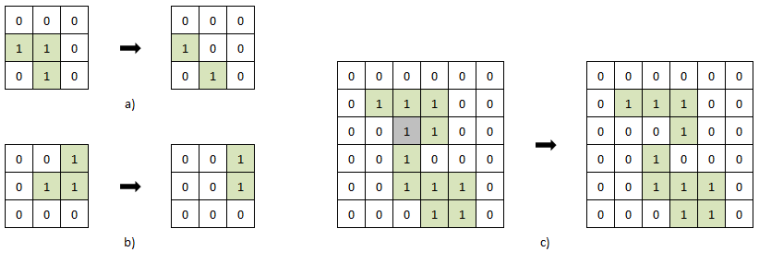


Figure 4.5: 4-connected update examples; a) label update breaking connectivity, b) label update preserving connectivity, c) disconnected region

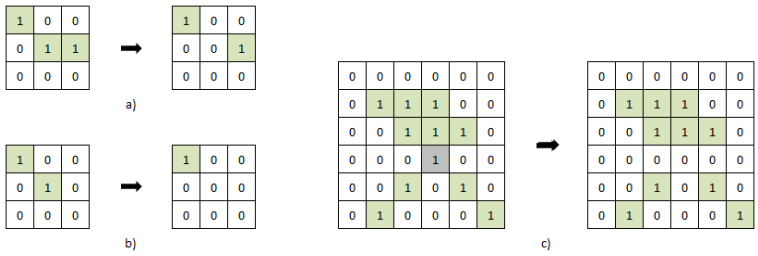


Figure 4.6: 8-connected update examples; a) label update breaking connectivity, b) label update preserving connectivity, c) disconnected region

4.3.2 Simply-Connected Update

As explained in previous section, 4 and 8-connected updates may result disconnected regions, and those regions must be handled by a post-processing step. This step would result in either creating new SPs for disconnected regions or connecting those disconnected pixels to their closest clusters. In order to get rid of this post-processing step and to ensure connectivity during iterations, Freifeld [19] proposed a simply-connected update method. This method is initiated from the concept of simple point defined by Bertrand [30]; if removing of a point does not change the topology, the point is called simple point. This can be rephrased in image domain such as if changing label of a pixel does not change topology, which is the number of connected components, the pixel is called as a simple pixel. Freifeld [19] let only the simple pixels to update their label during iterations.

The update mechanism of Freifeld [19], enables pixel label update if number of fore-

ground and background clusters do not change within a 3X3 neighbour block. Foreground pixels are defined as all the pixels belong to same cluster with center pixel and the background ones are the rest. The count of 4-connected foreground regions gives Foreground Number, FG, and the count of 8-connected background regions gives Background Number, BG. Before update FG and BG values are calculated, then center pixel is replaced with background value and new FG and BG values are calculated. If FG and BG do not change through label update, the center pixel is defined as a simple pixel and it can get label of one of its 4-connected neighbors. Figure 4.7.a shows a denied label update. Initially there are 1 FG and 2 BG regions and after label update there are 2 FG and 2 BG regions, where count of FG changes indicating that pixel is not a simple pixel. However for the case 4.7.b, label update is allowed. At the beginning there are one FG and one BG region, and after the update number of FG and BG remains the same indicating pixel is a simple point.

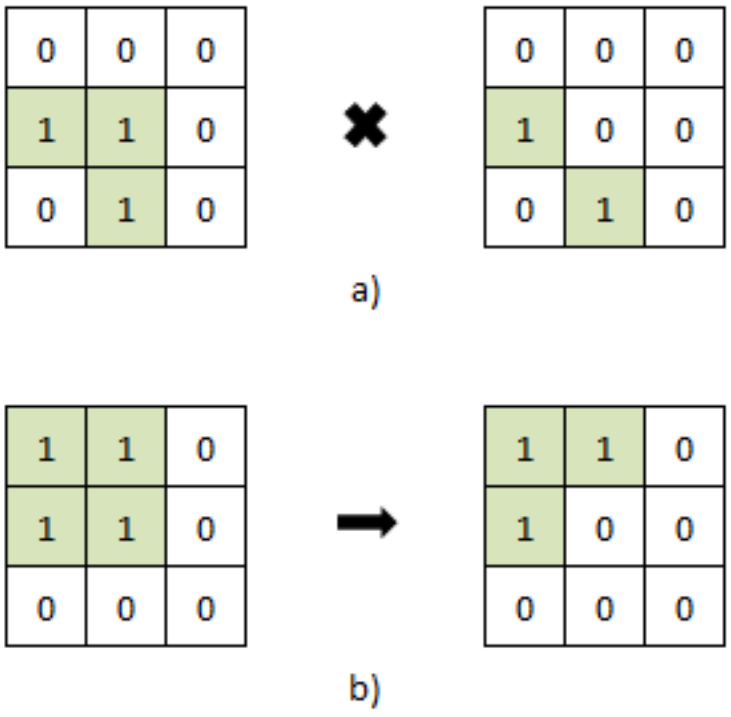


Figure 4.7: Simply-connected update examples; a) not allowed update, b) allowed update

This update procedure preserves 4-connectivity within a cluster and hence ensures 4-connectivity within cluster. There is an update pattern for simply-connected update;

image is divided by 3x3 blocks and 9 sub-iterations performed where at each iteration, only pixels at the same location of the 3x3 block are processed for label update as show in Figure 4.9. However, there is a different approach between paper work and its open source implementation. Although it is not stated as in paper work, in its implementation part, every 4-connected neighbor value is tested against Simply Connectedness and when the tests for all background labels are passed label update for that pixel becomes available. This approach requires checking simply-connectedness for each neighbor which brings a significant computational burden. In addition, this approach makes label update harder leading to late convergence of SPs which also requires extra processing time.

4.3.3 Proposed Method: Just-Connected Update

In order to find an efficient label update constraint that satisfies the need of generating connected regions and enabling fast convergence of SP methods, in this work a neighbor selection method called Just Connected (JC) is introduced. When the initial clusters are (4 or 8) connected regions, the proposed method ensures the connectivity to be preserved during the label update.

The idea behind JC is; regarding 3x3 neighborhood region of each pixel, if the region is 4-connected initially, region should remain 4 connected after a label update and if it is 8-connected before label update it should remain 8-connected after label update. No 4 or 8 discontinuity is allowed within 3x3 block. The detailed algorithm is given in Algorithm 2. A sample update mechanism is presented in Figure 4.8. In a) FG is 4-connected and its count is 1 but after it is 2 so that the update is denied. In b) FG is 4-connected and its count 1 which is not changed across update so update is allowed. In c) FG is 8-connected and count is 1, however after update FG count is to 2 which means update is not allowed. In d) FG is 8-connected and count is 1, FG count remains 1 after label update meaning that update is enabled for this case. This update procedure preserves the initial connectedness both for 4-connected and 8-connected cases. Processing pattern given in Figure 4.9 is applied for this method preventing interference between neighbors during label update.

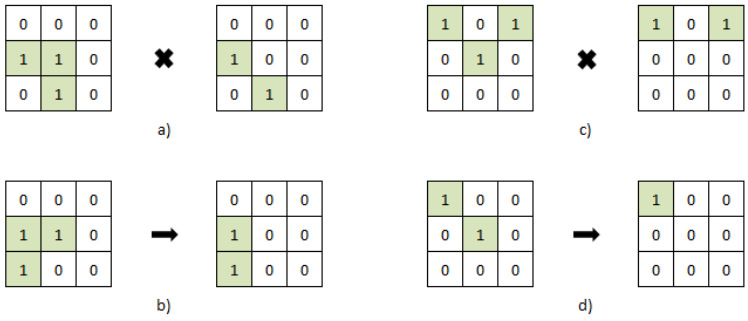


Figure 4.8: Just-connected update examples; a) 4-connected foreground not allowed update, b) 4-connected foreground allowed update, c) 8-connected foreground not allowed update, d) 8-connected foreground allowed update

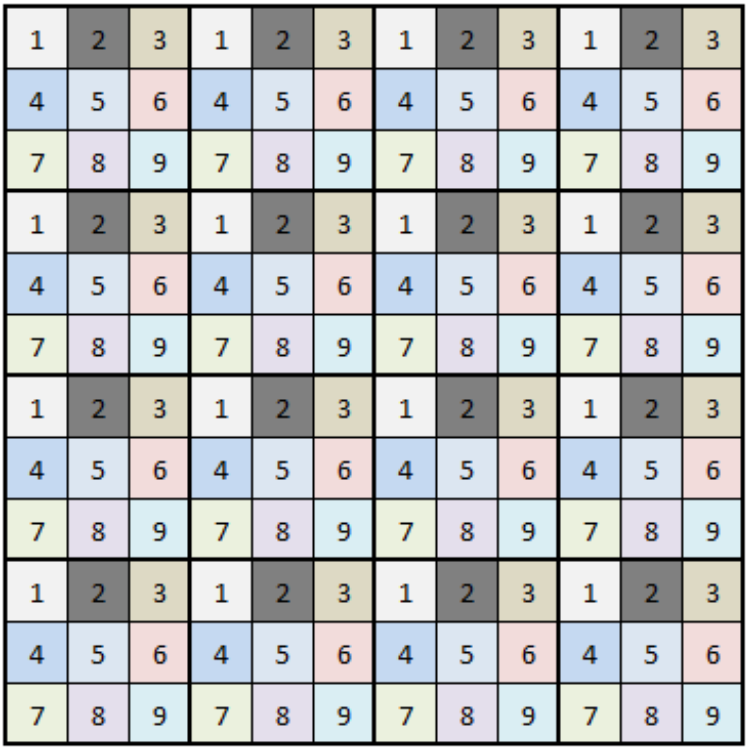


Figure 4.9: Update pattern of simply and just connected label updates

4.4 Refinement of Initial Tiling

After initial tiling step, an optional grid refinement may be applied. Initial statistics of SPs are calculated after initial tiling. This statistic has an important role on

Algorithm 2 Just Connected Label Update Decision

```
if Before label update, foreground is 4-connected then
    if After label update, number of 4-connected foreground region is 1 then
        Pixel update is allowed
    else
        Pixel update is not allowed
    end if
else if Before label update, foreground is 8-connected then
    if After label update, number of 8-connected foreground region is 1 then
        Pixel update is allowed
    else
        Pixel update is not allowed
    end if
else
    Pixel update is not allowed
end if
```

convergence of SPs during iterations. However, if statistics are not well initialized, performance of SP extraction method might degrade. To get rid of this degradation, either number if iterations should be increased that significantly increases overall processing time or grid refinement should be applied. Grid refinement corrects initial clusters, therefore the initial statistics of SPs before starting iterations.

4.4.1 Predefined Re-segmentation

Ince [17] proposed predefined re-segmentation (PDR), an initial tiling refinement method, that splits some SPs into either two or three clusters according to their inner spectral distribution. SPs whose contrast variations are high are the candidates to get divided into sub-SPs. Method is proposed to be applied after hexagonal initial tiling but can be extended to initially square tiled ones. The major drawback of PDR is, it breaks regular grid structure by creating new SPs which is out of scope of this work. This work focuses methods that only produces regular grid structure. However the idea refining the initial grid before iterations proposed by Ince [17] is investigated in

the next section in detail and a new refinement method is proposed that keeps regular grid structure. Then it is evaluated that, whether such a grid refinement is necessary or not.

4.4.2 Proposed Method : Edge-based Refinement

A good performing SP extraction method generates SPs which follow object boundaries well. Therefore, during iterations, SPs tend to converge at the edges of the objects. In other words, edge of objects become boundaries of SPs at the end. Depending on the method, the convergence may take too long. Before starting iterations, a pre-processing step that extracts these convergence points and shifts SP boundaries to those points speed-ups the overall process. In this in this work, Edge-based Refinement (EBR) method is proposed as a new grid refinement method. With EBR both boundaries and initial statistics of SPs become close to final ones that leads to an early convergence with only a few updates during iterations.

EBR can be applied with or without generating new SPs and can be used for both hexagon and rectangular initial tiling. After EBR, a post-processing step is needed to ensure cluster connectedness. Pseudo code of EBR is given in Algorithm 3.

Algorithm 3 Edge-based Grid Refinement

for all $S\ in\ SP\ List$ **do**
  for all $N\ in\ S\ Neighbor\ List$ **do**
    B is boundary point between S and N
    P and M is local extremum point on brightness derivative image and value respectively on the line between center of S, C_c, and center of N, C_n
    Th is threshold hyper-parameter
    if $M \leq Th$ **then**
      if using P as boundary does not change connectedness of S and N **then**
        $B \leftarrow P$, use P as new boundary
      end if
    end if
  end for
end for

Figure 4.10 a, b, c depicts SPs after initial tiling, EBR and iteration steps respectively for rectangular initial tiling. As seen in (b) EBR modifies initial SPs (a) by shifting boundaries to a point close to object edges which are later converged in (c). Similarly, (d), (e) and (f) depicts output of SP extraction steps: initial tiling, EBR and iterations respectively for hexagonal initial tiling. Here EBR modifies hexagonal grid given in (d), by locating boundaries through the object boundaries shown in (e) so that SPs converge faster during iterations (f).

In order to analyze the affect of initial tiling refinement, an experiment is conducted starting with rectangular and hexagonal initial tiling. Since this work focuses only on regular grid structure, connectivity must be enforced from the beginning. So proposed just connected 2 label update criteria is employed. JC, keeps connectivity as it is. If clusters are 4-connected at the beginning, they remain 4-connected after iterations, and same for 8-connected. Since both HEX and RECT initial tiling is 4-connected, generated SPs are going to be 4-connected as well with JC criteria.

Predefined re-segmentation is out of context of this experiment since it breaks regular grid structure. This experiment will clarify whether refinement of initial tiling is needed or not, therefore options edge-based refinement and no refinement are analyzed in this experiment. However, the results of PDR is also presented to make a comparison. Experiments are conducted for 500 and 1000 SPs and two different cost functions; SLIC and LASP given in Equations 4.9 and 4.10 respectively. Details of those cost-functions are described in the related sections. For each configuration, several lambda (compactness parameter) values are used to get results within a wide range of compactness and then BR, UE, ASA, BASA and Run-time values are calculated from the generated SPs. In order to make different configurations comparable, iteration length of experiments chosen such that total run-time of the algorithm with different tiling refinement methods is kept constant. To achieve a constant run-time, experiments employing EBR has two iterations less than the ones with no refinement, as EBR implementation approximately takes 2 iterations long. Table 4.1 shows summary of related configuration options of this experiment.

To begin with analyzing the refinement of initial tiling. There are 3 options, no refinement (OFF), pre-defined segmentation (PDR) and edge based refinement (EBR).

Figure 4.10: Edge-based refinement example; a, b, c are refinement of rectangular tiling, d, e, f are refinement of grid tiling

Table 4.1: Initial tiling refinement experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Hexagonal	OFF	500	SLIC (4.9)	BR
Rectangular	EBR	1000	LASP++ (4.10)	UE
	PDR			ASA
				BASA
				Run-time
				BASA AUC
				Run-time AUC

Figure 4.11 depicts performance of chosen cost functions in metrics BR, UE, ASA, BASA and run-time for 500 SP and hexagonal initial tiling. Here, PDR ones which marks with triangle shows top boundary performance. Then there is no significant difference between EBR and OFF marked with square and circle respectively. The results for 1000 SPs and hexagonal tiling case shown in Figure 4.12, results are similar to 500 SP case. PDR performs best and EBR and OFF comes after.

Besides, Table 4.2 and 4.3 shows numerical averages of BASA and run-time respectively which are area under curve (AUC) for the figures mentioned above. Numerical values are consistent with the figures. For 500 SPs and SLIC cost function, PDR has best BASA score with 0.6% above EBR and OFF score. Regarding run-time, they are close to each other as mentioned above this is what is intended. For LASP cost function, BASA score of PDR 0.5% is better than EBR and OFF where EBR and OFF has same score. For the 1000 SP case and SLIC cost function, PDR is 0.5% and 0.6% higher than OFF and EBR respectively. OFF has 0.1% higher score than EBR. For LASP, PDR is 0.5% and 0.4% shows better performance than EBR and OFF respectively. OFF beats EBR by 0.1%. Run-time difference is negligible for this experiment.

Figure 4.13 depicts performance of different tiling refinement methods for 500 SP. Here, similar to hexagonal tiling case, PDR outperforms the others in all metrics except run-time. The main difference from hexagonal tiling case is EBR performs

better than OFF. For the 1000 SPs case shown in Figure 4.14, PDR remains at the top, EBR continues to show better performance than OFF.

As shown in Tables 4.2 and 4.3, for 500 SPs and SLIC cost function, PDR has the best BASA score about 0.8% and 1.3% above of EBR and OFF score respectively, and EBR score is 0.5% higher than OFF. Regarding run-time, they are close to each other similar to hexagonal tiling case. Regarding LASP cost function performance, PDR is 0.6% and 1.0% better than EBR and OFF respectively, EBR outperforms OFF by 0.4%. For the 1000 SP case and SLIC cost function, PDR is 1.0% and 1.7% higher than EBR and OFF respectively. EBR has 0.7% higher score than OFF. For LASP cost function, PDR outperforms EBR and OFF by 0.8% and 1.4% respectively, EBR shows 0.6% better performance than OFF. Run-time difference is again negligible for this experiment.

To sum up, after a series of experiments to see effect of initial tiling refinement on SP extraction method performance, PDR is the one having best performance for all cases. Nevertheless, as mentioned earlier the drawback of PDR is, it generates new SPs breaking regular grid which prevents them to be employed by certain approaches like neural networks. In this work, only the methods preserving regular SP grid structure is considered, therefore PDR is out of scope of the work. It is used in the experiments to give idea about its performance for the users who do not care about preserving neighborhood topology. Then regarding the remaining methods OFF and EBR, EBR does not outperform OFF as expected for the hexagonal tiling case, however achieves better results when rectangle is selected as initial tiling type. EBR is designed to set final SP boundaries as early as possible, even before iterations start, however its computational burden makes its performance comparable with OFF. OFF performs slightly better than EBR in hexagonal tiling case but worse in rectangular tiling experiments.

Ince [17] claims that after initial tiling, applying a pre-processing step that refines initial tiling increases SP extraction performance and proposes a refinement method called predefined re-segmentation. Despite proposed method breaks regular grid structure, the idea initial tiling refinement is investigated in in this section. As a result, Edge-based refinement is proposed that keeps regular grid structure as it is

Table 4.2: Initial tiling refinement experiment BASA area under curve performance

BASA		**RECT**			**HEX**		
#SPs	**Method**	**OFF**	**EBR**	**PDR**	**OFF**	**EBR**	**PDR**
500	SLIC (4.9)	80,9%	81,4%	82,2%	81,3%	81,3%	81,9%
	LASP (4.10)	82,3%	82,7%	83,3%	82,7%	82,7%	83,2%
1000	SLIC (4.9)	80,6%	81,3%	82,3%	81,7%	81,6%	82,2%
	LASP (4.10)	81,8%	82,4%	83,2%	82,8%	82,7%	83,2%

Table 4.3: Initial tiling refinement experiment average run-time (ms) area under the curve performance

RT		**RECT**			**HEX**		
#SPs	**Method**	**OFF**	**EBR**	**PDR**	**OFF**	**EBR**	**PDR**
500	SLIC (4.9)	59,6	58,9	65,2	58,6	56,6	54,6
	LASP (4.10)	69,7	66,9	71,9	67,1	64,0	63,7
1000	SLIC (4.9)	53,0	51,1	50,7	50,4	49,1	47,7
	LASP (4.10)	63,0	59,0	59,0	59,1	56,2	55,8

and experiments are conducted to evaluate if there is a need of such a refinement. However, after experiments, it is observed that there is no increase in performance concluding refinement of initial tiling is not needed. Instead, performing two more iterations without refinement of initial tiling gives similar results. Since EBR calculations takes two iterations long, this time can be utilized at iterations step such that first two iteration already modifies the initial tiling in the same way.

4.5 Cost Function

Cost function is evaluated at the label update step of method flow given in Algorithm 1. Label updates are performed iteratively and applied for each pixel to minimize a

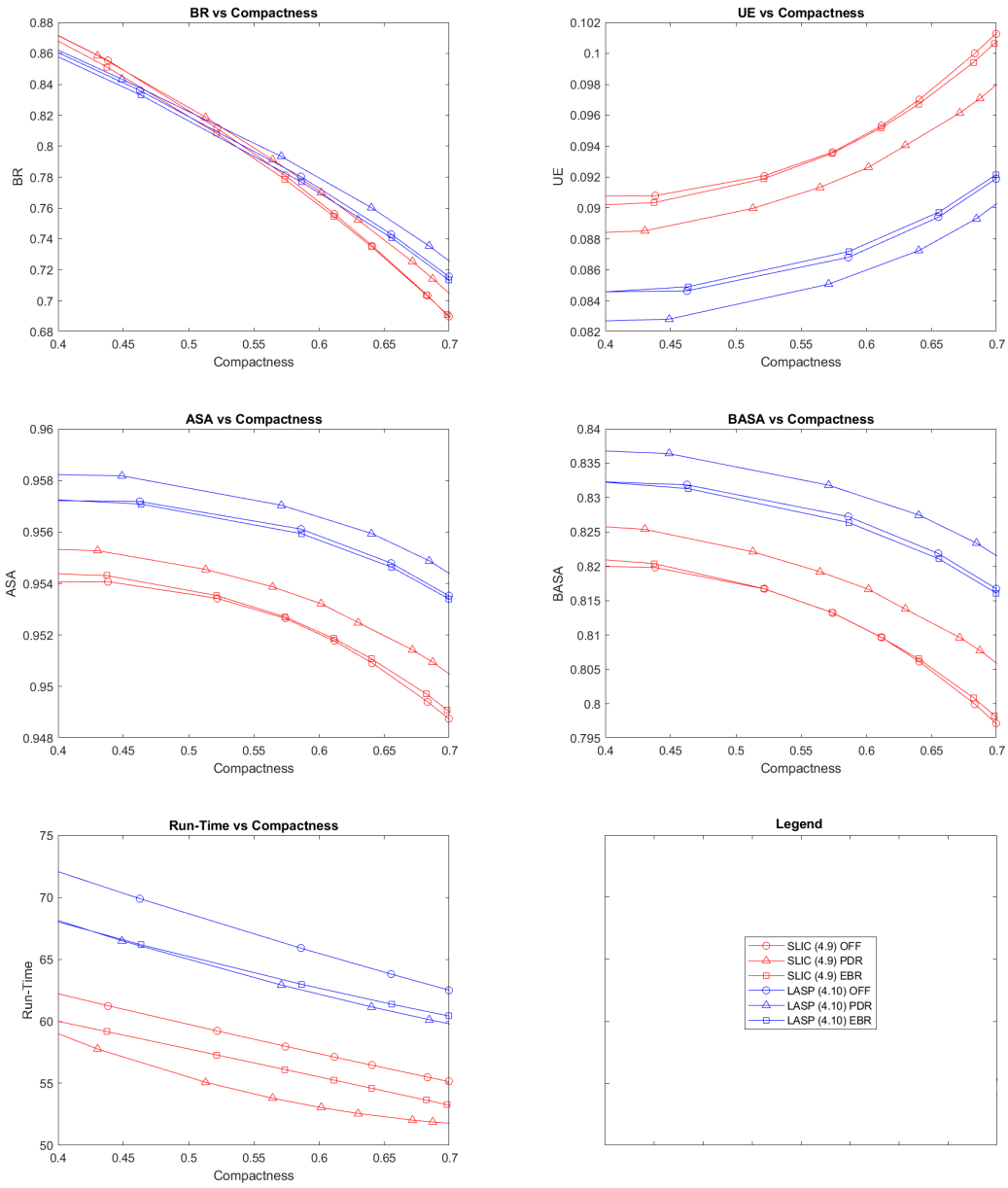


Figure 4.11: Performance of different tiling refinement methods for hexagonal initial tiling and 500 SPs

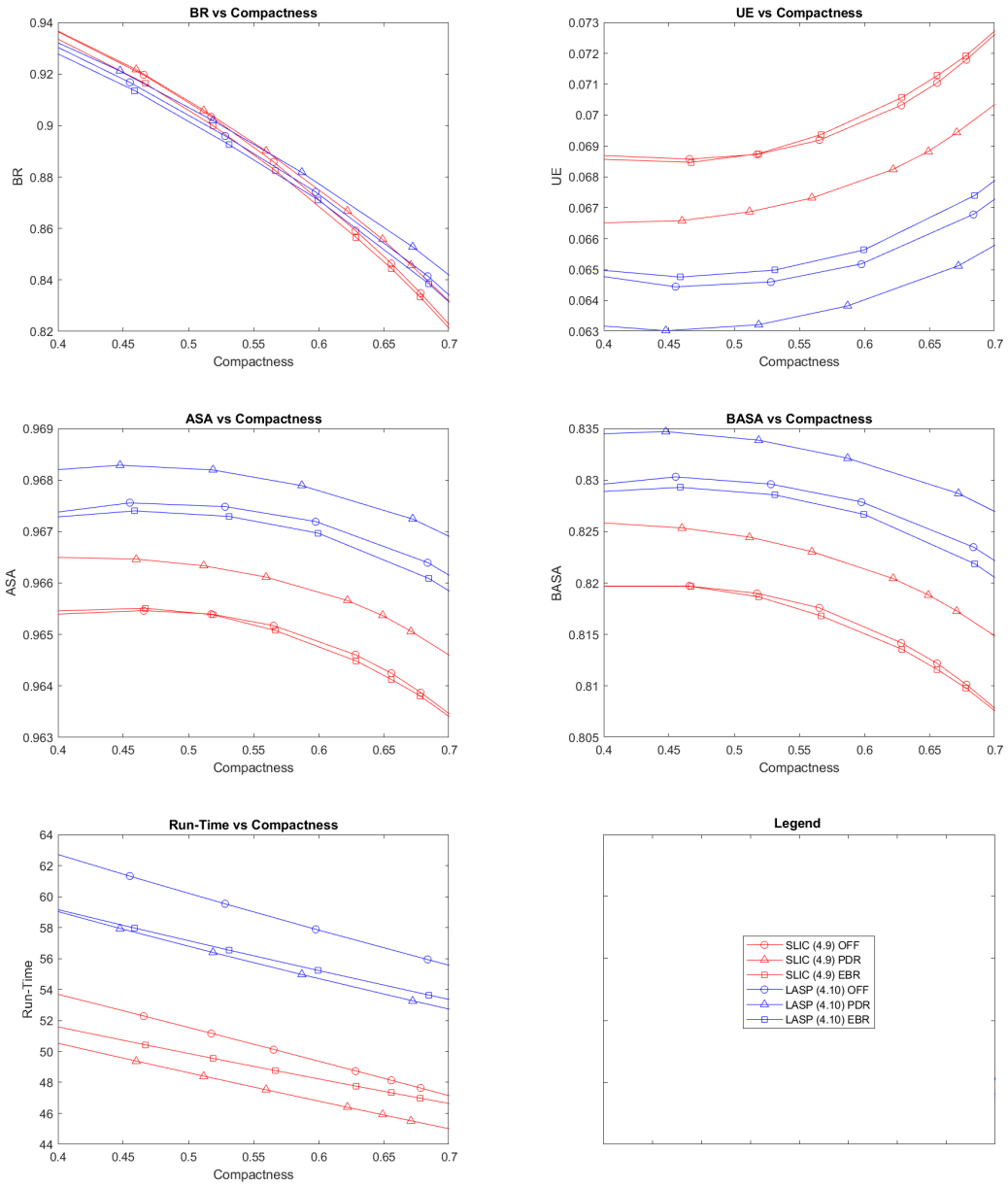


Figure 4.12: Performance of different tiling refinement methods for hexagonal initial tiling and 1000 SPs

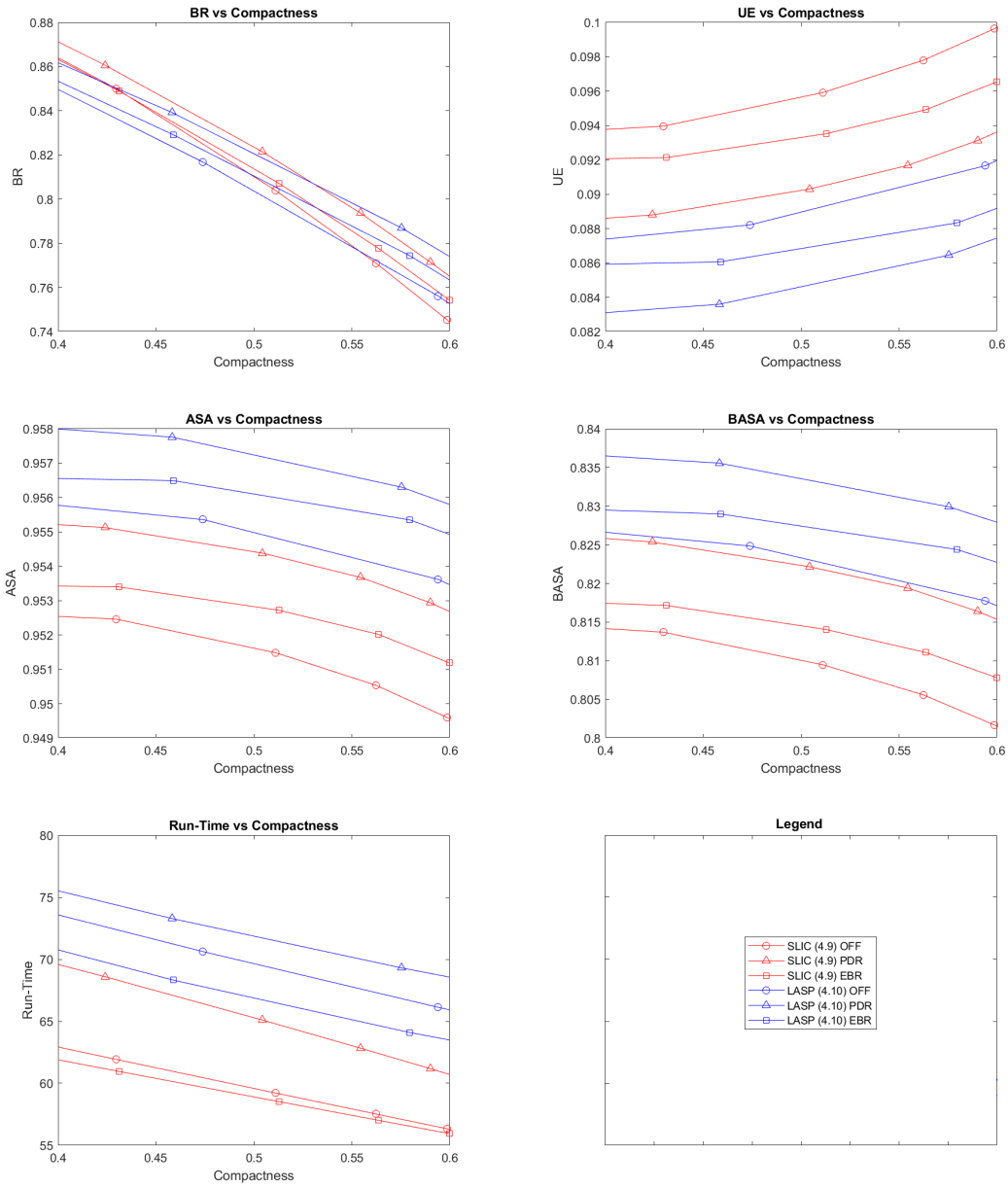


Figure 4.13: Performance of different tiling refinement methods for rectangular initial tiling and 500 SPs

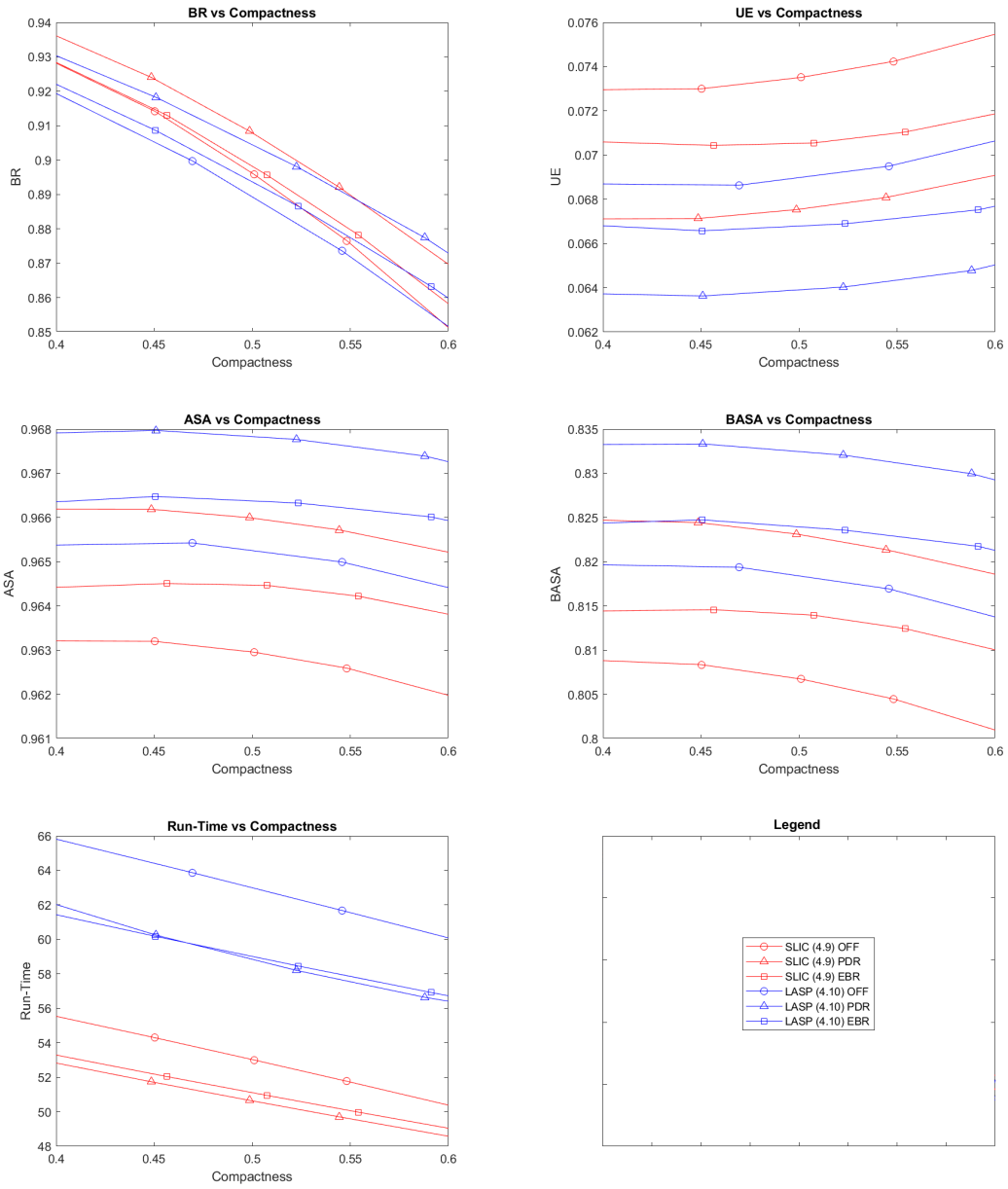


Figure 4.14: Performance of different tiling refinement methods for rectangular initial tiling and 1000 SPs

cost function. SP extraction methods differ in their proposed cost functions. As it affects the label update, the cost function the backbone of methods, that mainly determines the performance of the method. Following paragraphs describe cost function basics, presents state-of-the art ones, and then explain the proposed ones in detail.

Every pixel can be represented by a feature vector composed of visual (spectral) and spatial components:

$$X = [I_1(x,y) \quad ... \quad I_s(x,y) \quad x \quad y] \tag{4.1}$$

where $X \in R^n$ is n dimensional feature vector, $[I_1(x,y) \quad ... \quad I_s(x,y)]$ are visual components over S spectral bands and $[x,y]$ is the location on the image plane. Depth-based algorithms enhance extra depth information as the third dimension of spectral component besides $[x,y]$.

Achanta proposed SLIC method [2] which performs pixel to SP label assignment with K-means algorithm. SLIC assigns pixels to the closest SP based on Euclidean distance of feature vector X. Additionally, SLIC employs a compactness parameter to prioritize spatial distance over spectral distance or vice versa. SLIC [2] performs label updates by solving the following minimization problem iteratively:

$$\arg\min_{c \in C_{x,y}} \sum_{s=1}^{S} (I_s - \bar{I}_{s,c})^2 + \lambda((x - \bar{x}_c)^2 + (y - \bar{y}_c)^2) \tag{4.2}$$

where s represents index of S spectral bands, c is a cluster belongs to cluster set $C_{x,y}$ and λ is compactness parameter. SLIC utilize first order spectral and spatial statistics which are represented with $\bar{I}_{s,c}$ and $(\bar{x}_c, \bar{y}_c)$ in (4.2). Higher values of compactness parameter results in lower boundary adherence but more compact SPs.

The successor clustering-based cost functions use same approach such that function consists of a spectral distance part and a weighted spatial distance part.

However, Ince [17] claims that clustering-based methods based on SLIC approach solves a maximum likelihood problem. For the given conditional distribution of clus-

ters, pixel label can be calculated by maximizing the likelihood given in equation (4.3) by observing the feature vector X.

$$\arg\max_{c \in C_X} p(X|c) \tag{4.3}$$

where c is SP and C_X is the list of possible neighbor SPs that the pixel can be assigned to. Probabilistic distributions of clusters (SPs) are estimated while label updates.

If the distribution of a cluster is an n-dimensional multivariate Gaussian, pixel to SP membership function in (4.3) becomes:

$$p(X|c) = \frac{1}{(2\pi)^{n/2}|\Sigma_c|^{1/2}} e^{-\frac{1}{2}(X-\bar{X_c}^T)\Sigma_c^{-1}(X-\bar{X}_c)} \tag{4.4}$$

where $\Sigma_c \in R^{nxn}$ is covariance matrix and $\bar{X}_c$ is mean vector of cluster c in n dimensional feature space. The maximization problem in (4.3), with Gaussian distribution (4.3) assumption, can be expressed as the following minimization problem:

$$\arg\min_{c \in C_X} (X - \bar{X}_c)^T \Sigma_c^{-1} (X - \bar{X}_c) + \ln|\Sigma_c| \tag{4.5}$$

If the spectral bands are assumed to be independent and orthogonal (i.e. LAB color space), minimization equation (4.5) can be expressed as:

$$\begin{aligned} \arg\min_{c \in C_{x,y}} & \sum_{s=1}^{S} \frac{(I_s - \bar{I}_{s,c})^2}{\sigma_{s,c}^2} + \sum_{s=1}^{S} \ln \sigma_{s,c}^2 + \ln(\sigma_{x,c}^2 \sigma_{y,c}^2 - \sigma_{xy,c}^2) \\ & + \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix}^T \begin{bmatrix} \sigma_{x,c}^2 & \sigma_{xy,c} \\ \sigma_{xy,c} & \sigma_{y,c}^2 \end{bmatrix}^{-1} \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix} \end{aligned} \tag{4.6}$$

where $\sigma_{s,c}^2$ is variance of spectral band s, $\sigma_{x,c}^2$ and $\sigma_{y,c}^2$ are spatial variance over x and y axes respectively, and $\sigma_{xy,c}^2$ is spatial covariance of x and y axes of cluster c. This is the general cost function when cluster distributions are assumed to be Gaussian and spectral axes are assumed independent and orthogonal.

If it is assumed that spatial axes are independent, equation (4.6) can be expressed as:

$$\arg\min_{c\in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s-\bar{I}_{s,c})^2}{\sigma_{s,c}^2} + \frac{(x-\bar{x}_c)^2}{\sigma_{x,c}^2} + \frac{(y-\bar{y}_c)^2}{\sigma_{y,c}^2} + \sum_{s=1}^{S} \ln \sigma_{s,c}^2 + \ln \sigma_{x,c}^2 \sigma_{y,c}^2 \tag{4.7}$$

Based upon (4.7), when the variance terms are assumed to be constant they can be omitted, and (4.7) can be rewritten as follows:

$$\arg\min_{c\in C_{x,y}} \sum_{s=1}^{S} (I_s-\bar{I}_{s,c})^2 + \lambda((x-\bar{x}_c)^2 + (y-\bar{y}_c)^2) \tag{4.8}$$

where λ is a compactness parameter that prioritizes spatial distance over spectral distance. Higher values of lambda (λ) results more compact SPs with less boundary adherence.

Starting with the claim of clustering-based methods solve a maximum likelihood problem, and making several assumptions such as Gaussian distributed independent and orthogonal axes and constant variance, Ince [17] obtains SLIC cost function (4.2) as in (4.8).

Equation 4.8 can be rewritten as:

$$\arg\min_{c\in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s-\bar{I}_{s,c})^2}{\sigma_{spec}^2} + \frac{(x-\bar{x}_c)^2 + (y-\bar{y}_c)^2}{\sigma_{spa}^2} \tag{4.9}$$

where σ_{spec}^2 and σ_{spa}^2 are default spectral and spatial variances respectively. If $\sigma_{spa}^2 = \sigma_{spec}^2 A_{sp}/\lambda'$, where A_{sp} is the average SP area and λ' is compactness parameter, modified SLIC cost function [2] which can handle varying average SP area can be obtained.

Revised equation (4.9) of SLIC, indicates that cost function of SLIC uses constant global spectral variance and spatial covariance. Obviously, it is derived from equation (4.7) by assuming constant variance and covariance.

LASP [17] discards SLIC global spectral variance assumption and utilize second order statistics of spectral bands. As the variance of SPs are not converged initially, LASP estimates a robust a variance for each pixel by getting the minimum variance of neighboring SPs. The resultant equation becomes:

$$\arg\min_{c \in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s - \bar{I}_{s,c})^2}{\sigma_s^2(x,y)} + \frac{(x - \bar{x}_c)^2 + (y - \bar{y}_c)^2}{\sigma_{spa}^2} \tag{4.10}$$

where I_s is color channel s component of pixel, $\bar{I}_{s,c}$ is average color channel s component of SP c, x and y describe spatial location of pixel, $\bar{x}_c$ and $\bar{y}_c$ describe location of SP center, $\sigma_{spa}^2 = A_{sp}/\lambda$ is the spatial variance, A_{sp} is the average SP area, λ is compactness parameter and $\sigma_s^2(x,y) = \min\left\{\sigma_{s,c}^2\right\}_{c \in C_{x,y}}$ as variance.

With this approach, LASP [17] utilizes spectral distributions of clusters to take the advantage of local adaptation. Since SLIC [2] does not distinguish if processed region is texture-rich or not, it either fails to follow the object boundaries or generates less compact shapes. LASP [17], uses a region dependent normalization term for spectral distance which increases boundary adherence while preserving compactness. However, instead of directly using candidate cluster variances, method selects minimum variance among all candidate neighbors for each pixel as cluster variations are wanted to be small. Additionally, since initial statics are not reliable, even if it is not explicitly declared but just mentioned in [17], LASP limits spectral variances with a constant global hyper-parameter.

Both SLIC and LASP assumes a constant spatial covariance which forces every SP to have identical shape. This assumption significantly decreases boundary adherence as SPs should have different shapes and sizes to adapt to object boundaries well. Another disadvantage of those methods are that they rely on global spectral variance or variance limit which makes performance of methods image dependent. Although LASP cost-function introduces adaptiveness to local structure, global variance limits cause method to be image and data set dependent. Additionally, those globally constant parameters are hyper-parameters of the method making it harder to be employed by other applications.

Both methods make several assumptions while forming the cost-function. Gaussian distribution for spectral and spatial axes is quite acceptable. Orthogonal spectral axes is valid if LAB color space is used. However, independent spatial axes assumption is far away from being valid. Omitting spectral variances and employing a global spectral variance term or a limit for variance make SPs hard to adapt to local image structure which results in either low boundary adherence or irregular SP shapes. Moreover, discarding logarithmic parts of the Equation (4.7) with constant variance assumption decreases the performance further.

Based upon these facts, in the following sections, spectral and spatial part of SLIC and LASP cost functions will be improved and effect of improvements will be analyzed with a comprehensive set of experiments. New cost function alternatives will be proposed based on (4.6) which assumes Gaussian distributions, independent and orthogonal spectral axes.

4.5.1 Spatial Adaptiveness

As explained in previous section cost functions of SLIC (4.9) and LASP (4.10) are derived from (4.7) by employing several assumptions. One of them is identical spatial covariance matrix assumption. This assumption forces every SP to have same size and shape which is close to circular. However, it is quite hard to represent objects accurately with SPs having an identical shape. Shape can differ depending on the image structure and initial tiling. Therefore, enabling SPs to become ellipsoid in any alignment gives methods some flexibility to follow boundaries effectively.

A new cost function SLIC+ is proposed by employing spectral covariance onto SLIC cost function (4.9). With this update SLIC cost function, which will be called as SLIC+ for the rest of the text, is expressed as follows:

$$\begin{aligned}\arg\min_{c\in C_{x,y}} &\sum_{s=1}^{S}\frac{(I_s-\bar{I}_{s,c})^2}{\sigma^2_{spec}}+\ln(\sigma^2_{x,c}\sigma^2_{y,c}-\sigma^2_{xy,c})\\ &+\begin{bmatrix}x-\bar{x}_c\\ y-\bar{y}_c\end{bmatrix}^T\begin{bmatrix}\sigma^2_{x,c} & \sigma_{xy,c}\\ \sigma_{xy,c} & \sigma^2_{y,c}\end{bmatrix}^{-1}\begin{bmatrix}x-\bar{x}_c & y-\bar{y}_c\end{bmatrix}\end{aligned} \tag{4.11}$$

Table 4.4: SLIC vs SLIC+ cost function experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Hexagonal	None	1000	SLIC (4.9)	BR
			SLIC+ (4.11)	UE
				ASA
				BASA
				Run-time
				BR AUC
				UE AUC
				ASA AUC
				BASA AUC
				Run-time AUC

Table 4.5: Comparison summary of SLIC and SLIC+ cost functions

Cost Function	**BR**	**UE**	**ASA**	**BASA**	**Run-time**
SLIC (4.9)	88.7%	6.95%	96.4%	81.6%	51.7
SLIC+ (4.11)	89.0%	6.90%	96.5%	81.7%	51.7

In this equation, the hyper-parameter σ_{spec} acts as the compactness parameter, and can be utilized to prioritize the spatial term over spectral term or vice versa.

Proposed cost function SLIC+ (4.11), takes spatial covariance into account which allows SPs to have different sizes and orientations. This modification is expected to increase boundary adherence which is analyzed in the following experiment. SLIC and SLIC+ cost functions are compared by an experiment with the configuration summarized in Table 4.4. In this experiment initial tiling is hexagonal, initial tiling refinement is not applied, JC label update constraint is applied, 1000 SPs are extracted, and LAB color space is used.

Figure 4.15 shows the results of the experiment. It is clearly seen that utilizing spatial

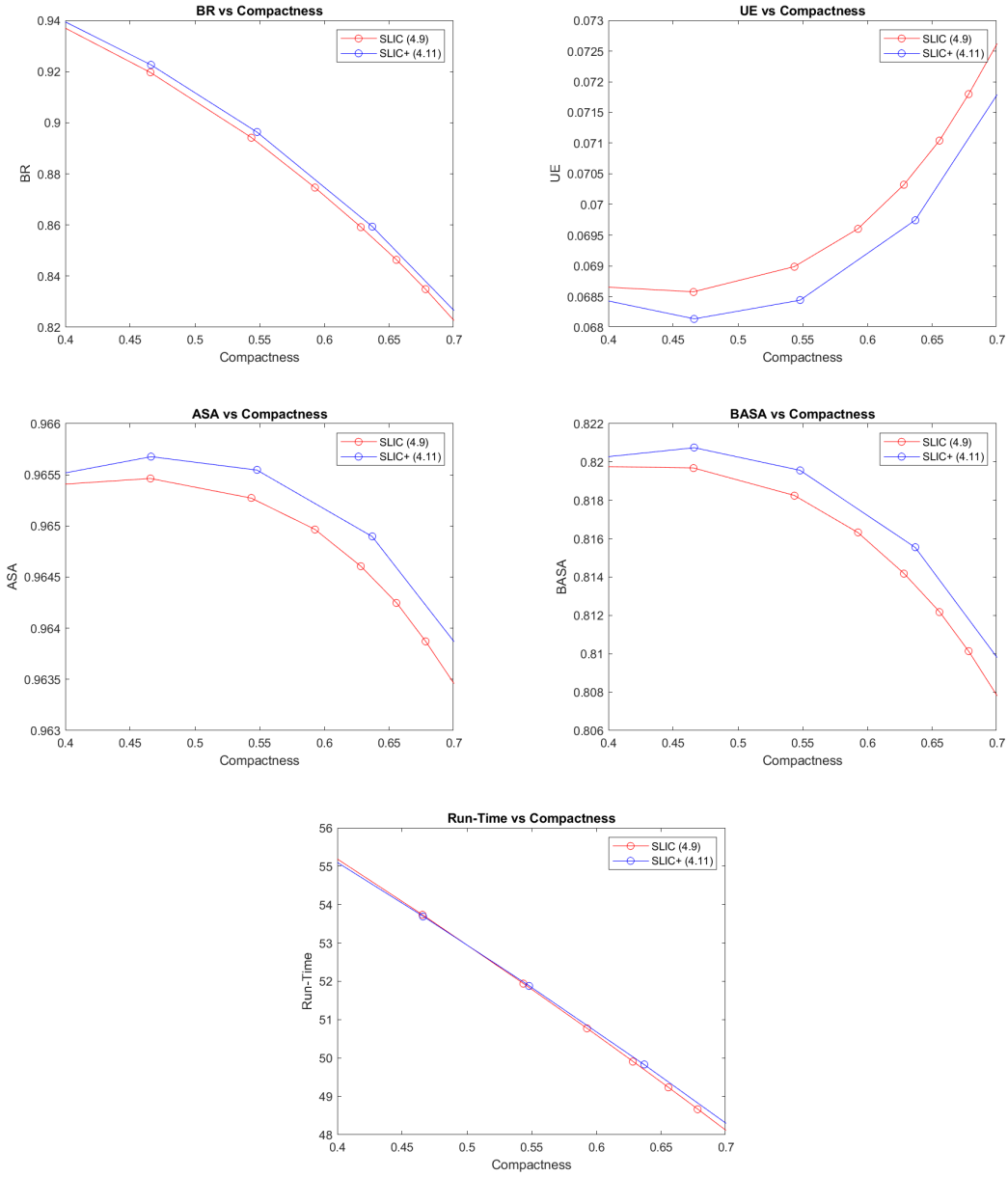


Figure 4.15: Comparison of SLIC and SLIC+ cost functions

covariances improves the performance for all measured metrics. Additionally, Table 4.5 quantifies performance of two cost-functions by calculating area under curve of each metric. As presented in Table 4.5, SLIC+ improves SLIC by 0.3% in BR, 0.05% in UE, 0.1% in ASA and 0.1% in BASA metric. Difference in run-time is negligible. It can be concluded that proposed spatial adaptiveness improves the performance.

Spatial adaptation is applied to LASP cost function (4.10) as well, to propose a new cost function called LASP+. Equation of LASP+ is as follows:

$$\begin{aligned} \arg\min_{c \in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s - \bar{I}_{s,c})^2}{\sigma_s^2(x,y)} + \ln(\sigma_{x,c}^2 \sigma_{y,c}^2 - \sigma_{xy,c}^2) \\ + \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix}^T \begin{bmatrix} \sigma_{x,c}^2 & \sigma_{xy,c} \\ \sigma_{xy,c} & \sigma_{y,c}^2 \end{bmatrix}^{-1} \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix} \end{aligned} \tag{4.12}$$

As proposed, LASP+ (4.12) takes spatial covariance into account allowing SPs to have different sizes and orientations. Like SLIC+ (4.11), this modification is expected to increase boundary adherence and it's effect is analyzed with an experiment. LASP and LASP+ cost functions are compared in the experiment whose configuration is summarized in Table (4.6); initial tiling is hexagonal, initial tiling refinement is not applied, JC label update constraint is applied, 1000 SPs are extracted, and LAB color space is used.

Figure 4.16 depicts the results of the experiment. Similar to SLIC (4.9) - SLIC+ (4.11) cost function comparison, the performance improvement is clearly seen in all measured metrics. Additionally, Table 4.7 quantifies performance of two cost functions by calculating area under curve of each metric. LASP+ improves LASP by 0.5% in BR, 0.06% in UE, and 0.1% in BASA metric. Results are similar for ASA metric. In this experiment difference in run-time is negligible meaning improvements does not bring a significant computational burden. As in SLIC cost function (4.9), proposed spatial adaptiveness improves the performance for LASP 4.10.

Regarding performance metrics for LASP+ and SLIC+, it can easily be concluded that employing spatial covariance in cost function significantly increases performance by enabling to generate different sized and shaped SPs that fit to local image structure

Table 4.6: LASP vs LASP+ cost function experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Hexagonal	None	1000	LASP (4.10)	BR
			LASP+ (4.12)	UE
				ASA
				BASA
				Run-time
				BR AUC
				UE AUC
				ASA AUC
				BASA AUC
				Run-time AUC

Table 4.7: LASP vs LASP+ cost function experiment area under curve comparison

Cost Function	**BR**	**UE**	**ASA**	**BASA**	**Run-time**
LASP (4.10)	88.6%	6.51%	96.7%	82.8%	64.2
LASP+ (4.12)	89.1%	6.45%	96.7%	82.9%	65.0

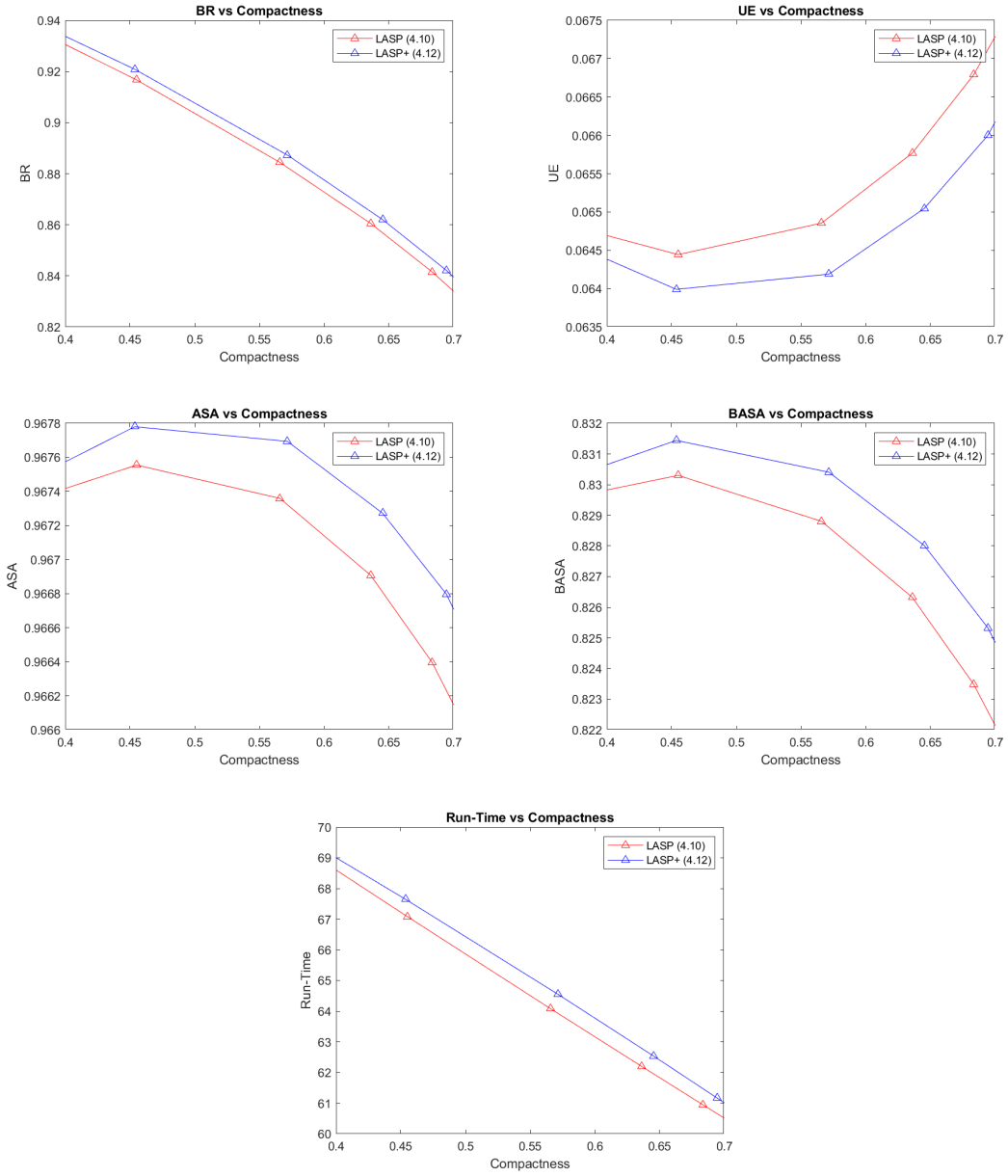


Figure 4.16: LASP vs LASP+ cost function comparison experiment results

well with insignificant computational cost.

4.5.2 Spectral Adaptiveness

The cost function utilized by SLIC (4.9) uses a global fixed variance to normalize spectral term. LASP performs spectral normalization with a pixel specific robust variance estimate (4.10), however it also employs fixed global upper and lower bounds for this variance estimate. All are method specific hyper-parameters. However, these fixed global definitions makes it harder to adapt different images. These parameters are mostly data set optimized and when algorithm is tested on a different set of images, performance of the methods might significantly change.

In order to get rid of those hyper-parameters and make them adaptive to image, various improvements are proposed in this section. Unlike SLIC (4.9) and SLIC+ (4.11), that use a fixed global variance for normalization of spectral term, variance of spectral channels can be extracted from image. In other words, making variance change dynamically by setting it to SPs average variance at each iteration may help to achieve a more adaptive method and increase performance.

A new cost function SLIC++ is proposed by incorporating dynamic variance calculation to SLIC+ (4.11) cost function:

$$\begin{aligned} \arg\min_{c \in C_{x,y}} & \sum_{s=1}^{S} \frac{(I_s - \bar{I}_{s,c})^2}{\bar{\sigma}_s^2} + \ln(\sigma_{x,c}^2 \sigma_{y,c}^2 - \sigma_{xy,c}^2) \\ & + \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix}^T \begin{bmatrix} \sigma_{x,c}^2 & \sigma_{xy,c} \\ \sigma_{xy,c} & \sigma_{y,c}^2 \end{bmatrix}^{-1} \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix} \end{aligned} \tag{4.13}$$

where $\bar{\sigma}_s^2$ is the average variance of SPs over spectral channel s.

Proposed cost function SLIC++ (4.13), employs image adaptive variance and it is expected to improve performance of cost function SLIC+ (4.11) which is superior of SLIC. In order to compare the performance of these cost functions, an experiment whose configuration is summarized in Table (4.8) is conducted. In this experiment initial tiling is hexagonal, tiling refinement is not applied, JC label update constraint

Table 4.8: SLIC, SLIC+ vs SLIC++ cost function experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Hexagonal	None	1000	SLIC (4.9)	BR
			SLIC+ (4.11)	UE
			SLIC++ (4.13)	ASA
				BASA
				Run-time
				BR AUC
				UE AUC
				ASA AUC
				BASA AUC
				Run-time AUC

is applied, 1000 SPs are extracted, and LAB color space is used. Performance evaluation is performed on Berkeley Segmentation Dataset BSD300 [31].

Figure 4.17 shows the results of experiment for various metrics. It is clearly seen that SLIC++ improves SLIC+ and hence SLIC performance in all measured metrics. Table 4.9 gives quantified performance of three cost functions by calculating area under curve of each metric. SLIC++ improves SLIC+ by 0.5% in BR, 0.48% in UE, 0.2% in ASA and 1.4% in BASA metric, on the other hand improves SLIC by 0.8% in BR, 0.53% in UE, 0.3% in ASA and 1.5% in BASA metric. There is no significant

Table 4.9: SLIC, SLIC+ vs SLIC++ cost function experiment area under curve comparison

Cost Function	**BR**	**UE**	**ASA**	**BASA**	**Run-time**
SLIC (4.9)	88.7%	6.95%	96.4%	81.6%	51.7
SLIC+ (4.11)	89.0%	6.90%	96.5%	81.7%	51.7
SLIC++ (4.13)	89.5%	6.42%	96.7%	83.1%	51.7

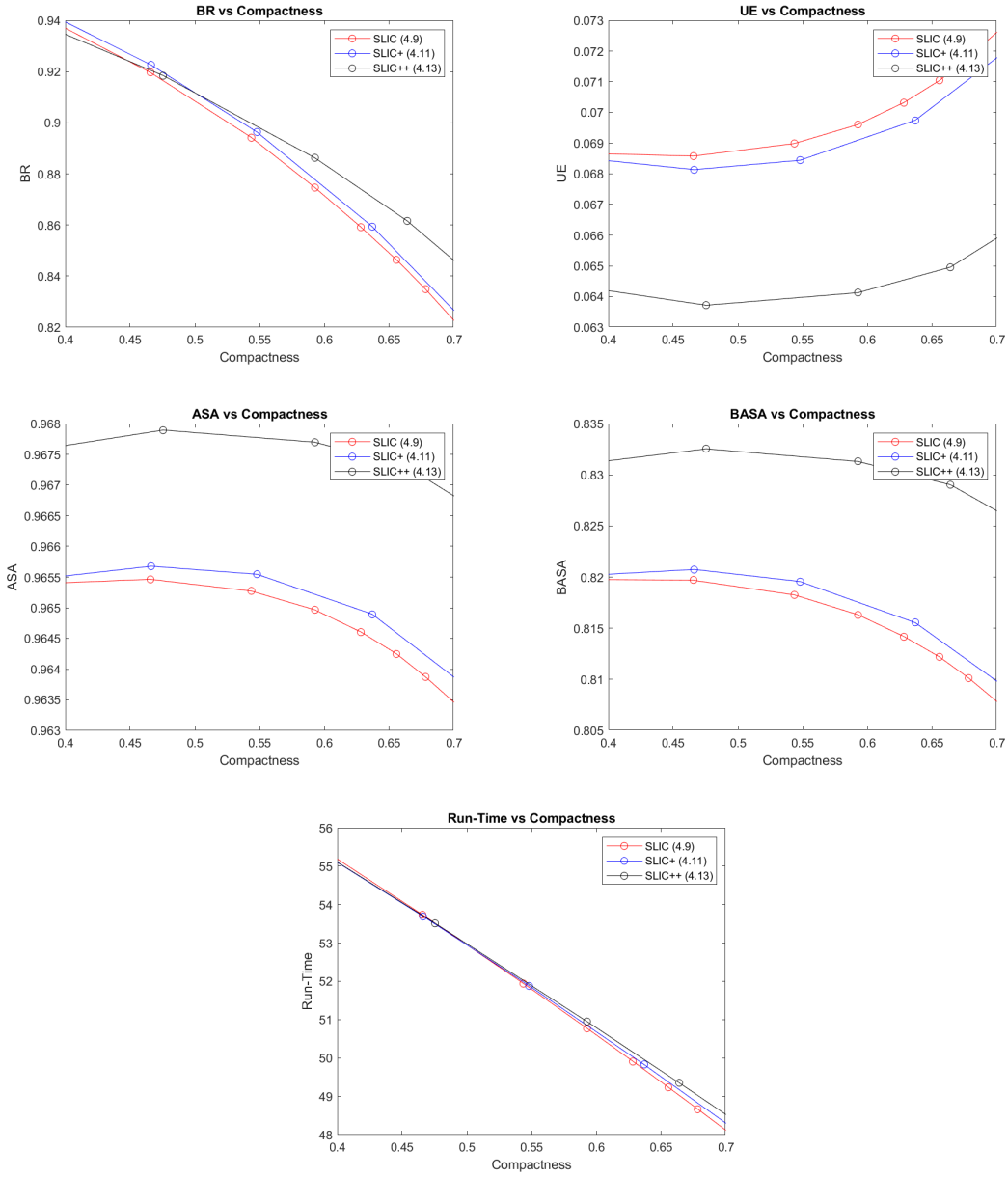


Figure 4.17: SLIC, SLIC+ vs SLIC++ cost function experiment results

difference in run-time. Based on experiment results, it can be stated that spectral enhancement proposed in this section significantly increases accuracy of SLIC 4.9 and SLIC 4.11.

The idea using image adaptive global variables instead of constant hyper-parameters can be applied to LASP (4.10) and LASP+ (4.12) cost functions as well. LASP and LASP+ limit variance of SPs with pre-defined upper and lower limits. As SLIC and SLIC+, these fixed hyper-parameters make methods incapable of adapting to different images or data sets. Instead of using fixed global limits, extracting limits from image itself may help to adapt to different images and data sets better. In proposed cost function LASP++ 4.14, upper and lower bounds are defined relative to average SP variance for each spectral band:

$$\begin{aligned} \arg\min_{c \in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s - \bar{I}_{s,c})^2}{\sigma_s^2(x,y,k)} + \ln(\sigma_{x,c}^2 \sigma_{y,c}^2 - \sigma_{xy,c}^2) \\ + \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix}^T \begin{bmatrix} \sigma_{x,c}^2 & \sigma_{xy,c} \\ \sigma_{xy,c} & \sigma_{y,c}^2 \end{bmatrix}^{-1} \begin{bmatrix} x - \bar{x}_c \\ y - \bar{y}_c \end{bmatrix} \end{aligned} \tag{4.14}$$

where $\sigma_s^2(x,y,k) = min(max(\sigma_s^2(x,y), \bar{\sigma}_s^2/k), k\bar{\sigma}_s^2)$,$k$ is upper and lower bound with $\sigma_s^2(x,y)$ is the robust variance estimate of pixel (x,y) for channel s and $\bar{\sigma}_s^2$ is average SP variance over channel s.

Proposed method LASP++ (4.14), employs image adaptive variance limits and it is expected to improve th performance of cost function LASP+ (4.12) which is superior of LASP (4.10). In order to evaluate improvement, an experiment with the configuration summarized in Table 4.10 is conducted. In this experiment initial tiling is hexagonal, tiling refinement is not applied, JC label update constraint is applied, 1000 SPs are extracted, and LAB color space is used. Performance evaluation is performed on Berkeley Segmentation Dataset BSD300 [31].

Figure 4.18 shows results of the experiment. It is clearly seen that LASP++ improves LASP+ and hence LASP performance in all measured metrics except BR. In addition, regarding Table 4.11 that shows AUC of metrics, LASP++ improves LASP+ by 0.17% in UE, 0.1% in ASA and 0.6% in BASA metric and improves LASP by 0.23%

Table 4.10: LASP, LASP+ vs LASP++ cost function experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Hexagonal	None	1000	LASP (4.10)	BR
			LASP+ (4.12)	UE
			LASP++ (4.14)	ASA
				BASA
				Run-time
				BR AUC
				UE AUC
				ASA AUC
				BASA AUC
				Run-time AUC

Table 4.11: LASP, LASP+ vs LASP++ cost function experiment area under curve comparison

Cost Function	**BR**	**UE**	**ASA**	**BASA**	**Run-time**
LASP	88.6%	6.51%	96.7%	82.8%	64.2
LASP+	89.1%	6.45%	96.7%	82.8%	65.0
LASP++	88.4%	6.28%	96.8%	83.4%	63.3

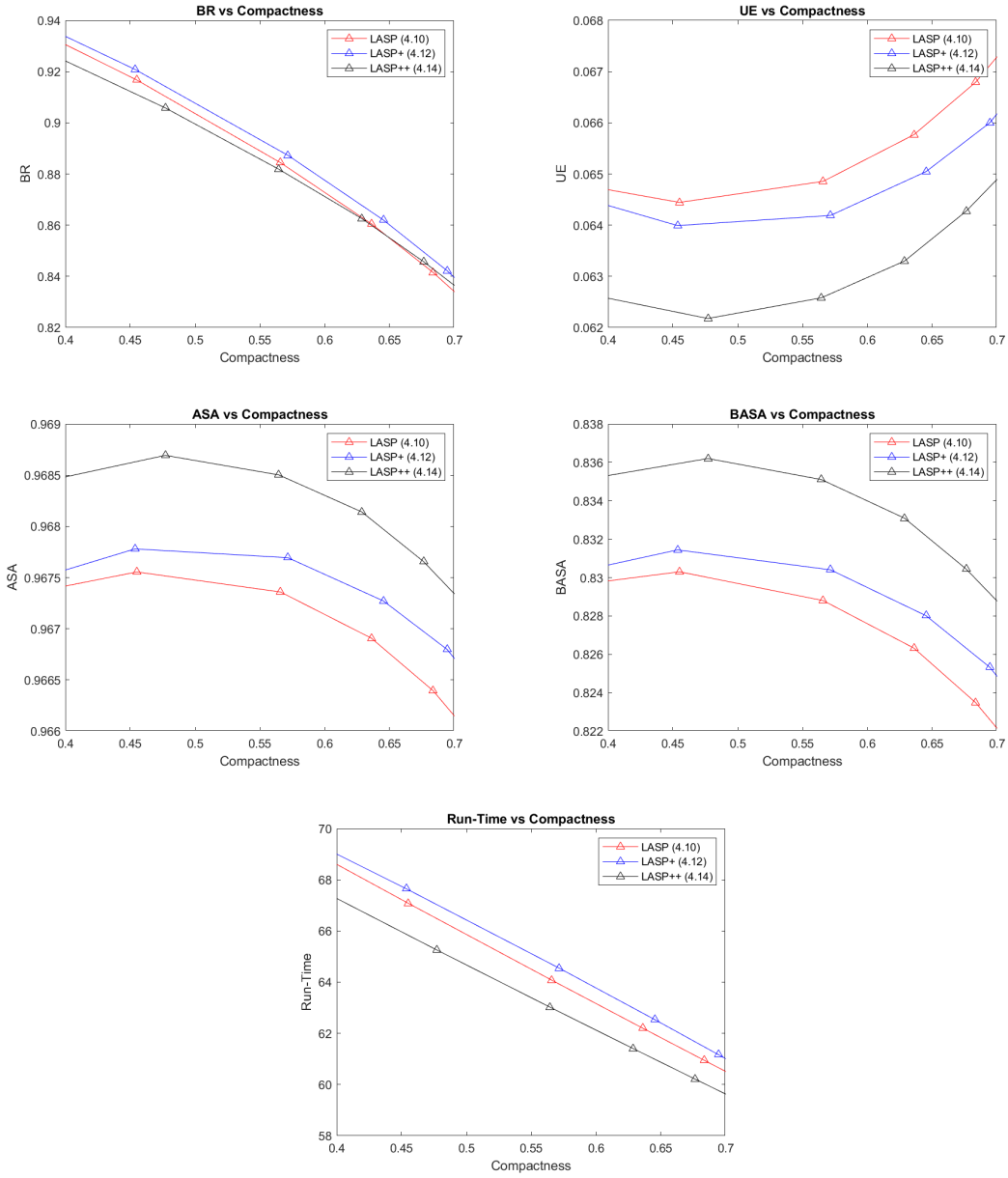


Figure 4.18: LASP, LASP+ vs LASP++ cost function experiment results

in UE, 0.1% in ASA and 0.6% in BASA metric. LASP++ under-perform in BR metric; however, this metric is not a fair metric over-penalizing errors as described in Chapter (3). In this experiment difference in run-time is negligible meaning improvements does not bring any significant computational burden to existing cost functions.

To conclude, instead of using fixed global hyper-parameters for spectral terms, extracting them from image significantly improves performance and makes cost functions stable across different images and data sets without a significant computational burden.

4.5.3 Proposed Cost Function

Throughout this chapter, SP are assumed to be Gaussian distributed. If this assumption is valid, right distribution can be obtained during iterations, and a Bayesian classifier can be achieved for pixel to SP assignments by using the most general form of the cost function (4.6).

In this section, a new cost function is proposed to achieve Bayesian classification for pixel to SP assignment. Proposed cost function assumes independent and orthogonal spectral bands and employs spatial covariance. Additionally, proposed cost function adapts variance bounds dynamically by extracting them from image. For the rest of the text the proposed cost function will be called as Bayesian cost function and the SP extraction with this cost function will be called as Bayesian Superpixels (BSP). The proposed Bayesian cost function is as follows:

$$\begin{aligned}\arg\min_{c\in C_{x,y}} \sum_{s=1}^{S} \frac{(I_s-\bar{I}_{s,c})^2}{\sigma_{s,c}^2(k)} + \sum_{s=1}^{S} \ln\sigma_{s,c}^2 + \ln(\sigma_{x,c}^2\sigma_{y,c}^2 - \sigma_{xy,c}^2) \\ + \begin{bmatrix} x-\bar{x}_c \\ y-\bar{y}_c \end{bmatrix}^T \begin{bmatrix} \sigma_{x,c}^2 & \sigma_{xy,c} \\ \sigma_{xy,c} & \sigma_{y,c}^2 \end{bmatrix}^{-1} \begin{bmatrix} x-\bar{x}_c \\ y-\bar{y}_c \end{bmatrix}\end{aligned} \tag{4.15}$$

where $\sigma_{s,c}^2(k) = min(max(\sigma_{s,c}^2, \bar{\sigma}_s^2/k), k\bar{\sigma}_s^2)$, k is upper and lower bound ratio with respect to average SP variance, $\sigma_{s,c}^2$ is the variance estimate of SP c for channel s and $\bar{\sigma}_s^2$ is average SP variance over channel s.

Method improves LASP family in terms of run-time by using SP level variance calculations. This removes per-pixel variance calculation burden comes with LASP-based cost functions. Method also employs spectral covariance to follow image structure effectively that improves LASP and SLIC state of the art cost functions. Additionally, proposed cost function (4.15) employs dynamic variance bounds as defined in LASP++ 4.14 to increase adaptiveness both within image and across different datasets.

LASP++ has high boundary adherence but suffers from high run-time, on the other hand SLIC++ has low run-time but has relatively lower performance than LASP++. It is expected that, the proposed cost function enables to achieve LASP++ boundary performance with SLIC++ run-time.

In order to evaluate Bayesian cost function, an experiment is conducted where BSP, LASP++ and SLIC++ are compared in terms of SP extraction performance. Table 4.12 shows related configuration; hexagonal and rectangular initial tiling alternatives evaluated separately, tiling refinement is not applied, JC label update constraint is applied, 1000 SPs are extracted, and LAB color space is used. Performance evaluation is performed on Berkeley Segmentation Dataset BSD300 [31].

Figure 4.19 shows the results of the experiment for hexagonal initial tiling for metrics defined in Chapter 3. BSP performance is in between SLIC++ and LASP++. Performance is close to LASP++ in terms of boundary adherence, and on the other hand close to SLIC++ in terms of run-time. Table 4.13 shows area under curve for each metric to make a qualitative comparison.

For hexagonal initial tiling, BSP is better than SLIC++ by 0.08% in UE, 0.1% in ASA and 0.2% in BASA metric; worse than SLIC++ 0.2% in BR and $4.8ms$ in run-time. Regarding BASA performance that is the major metric in this work, BSP performance is higher than SLIC++, with about 10% worse run-time. Hence it is hard to compare both while one provides high accuracy in a relatively longer time and the other provides lower accuracy in a relatively shorter time.

With respect to LASP++, BSP performance is lower by 0.06% in UE and 0.1% in BASA metric. Their ASA performance is similar, and BSP performs better by 0.9%

Table 4.12: SLIC++, LASP++ vs BSP proposed cost function experiment configuration

Initial Tiling	**Refinement**	**#SPs**	**Cost Function**	**Metrics**
Rectangular	None	1000	SLIC++ (4.13)	BR
Hexagonal			LASP++ (4.14)	UE
			BSP (4.15)	ASA
				BASA
				Run-time
				BR AUC
				UE AUC
				ASA AUC
				BASA AUC
				Run-time AUC

Table 4.13: SLIC++, LASP++ vs BSP proposed cost function experiment area under curve comparison

Tiling	**Cost Function**	**BR**	**UE**	**ASA**	**BASA**	**Run-time**
HEX	SLIC++ (4.13)	89.5%	6.42%	96.7%	83.1%	51.7
	LASP++ (4.14)	88.4%	6.28%	96.8%	83.4%	63.3
	BSP (4.15)	89.3%	6.34%	96.8%	83.3%	56.5
RECT	SLIC++ (4.13)	89.9%	6.87%	96.5%	81.9%	55.2
	LASP++ (4.14)	88.4%	6.81%	96.5%	82.1%	67.5
	BSP (4.15)	89.4%	6.80%	96.5%	82.1%	59.3

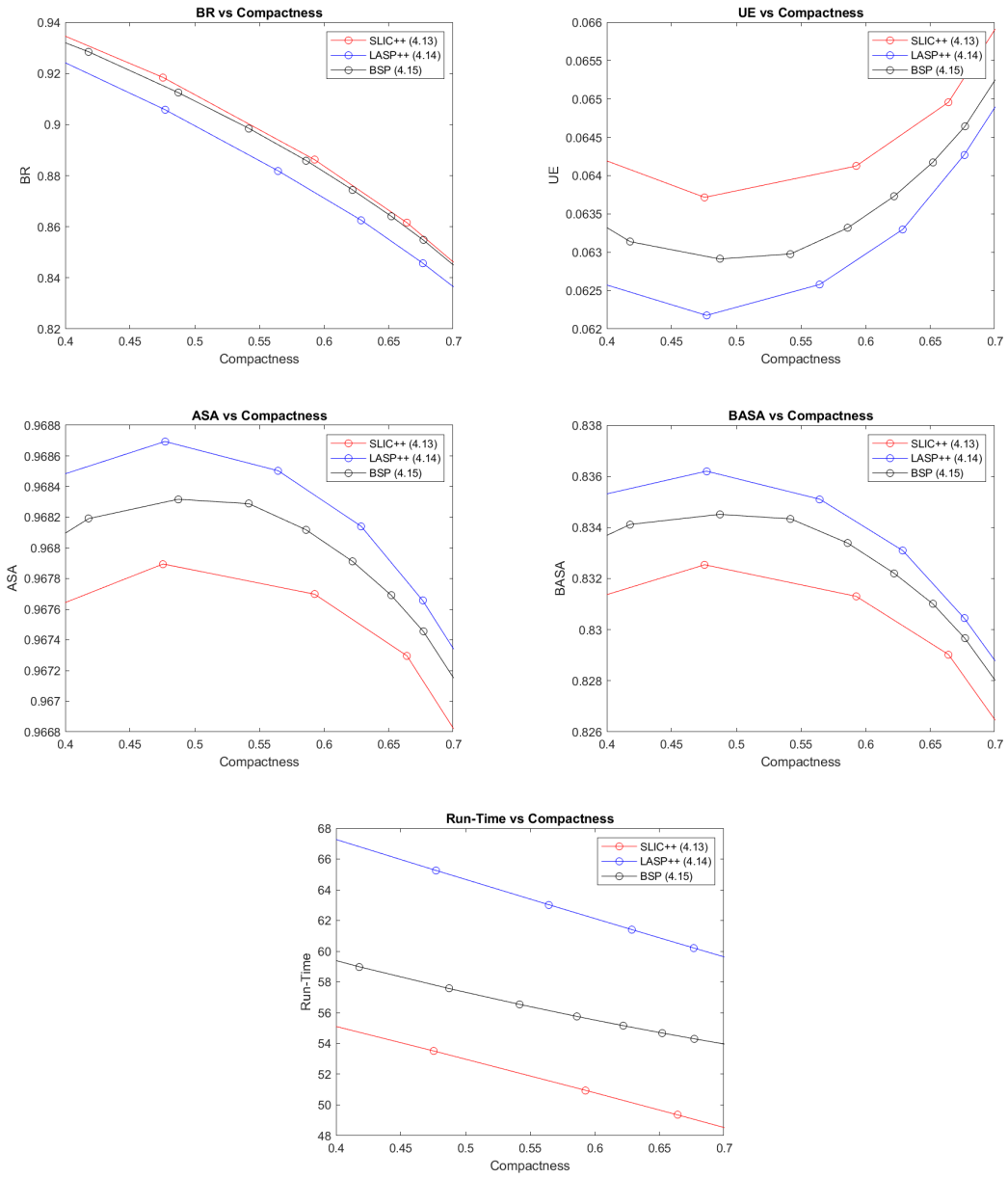


Figure 4.19: SLIC++, LASP++ vs BSP cost function experiment results for hexagonal initial tiling

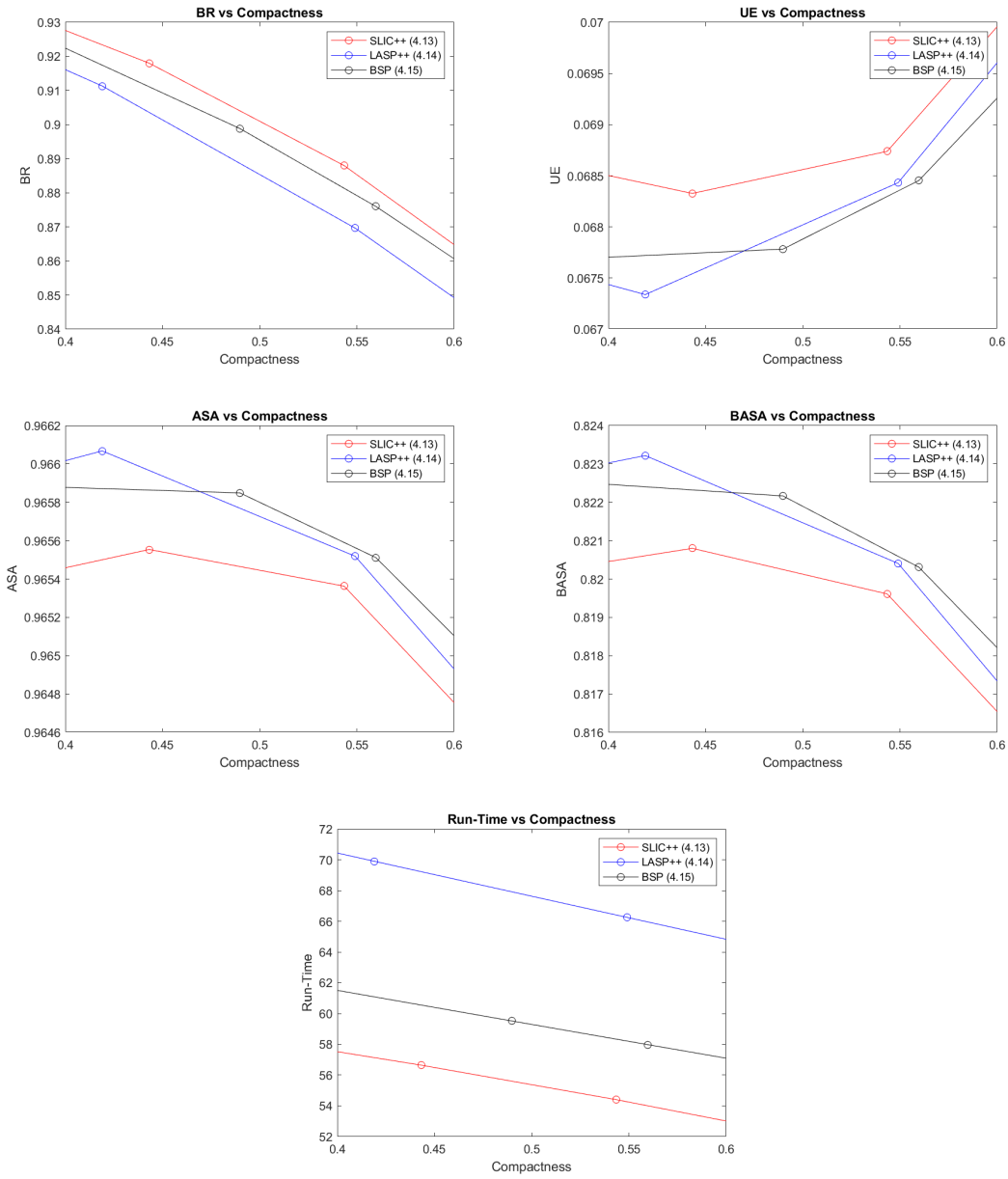


Figure 4.20: SLIC++, LASP++ vs BSP cost function experiment results for rectangular initial tiling

in BASA and $7.8ms$ in run-time metric. Hence regarding BASA and run-time, BSP has 0.1% worse accuracy than LASP++ achieved in about 12.5% shorter time. Same conclusion can be stated here, as accuracy increases run-time decreases that makes it difficult to choose one to another.

To compare SLIC++ and LASP++, LASP++ is better in UE by 0.14%, in ASA by 0.1% and in BASA by 0.3%. Nevertheless, SLIC++ has higher performance in BR by 1.1%, $11.6ms$ in run-time. So, LASP++ achieves 0.3% better accuracy in a 22.5% more time.

There is no certain winner for hexagonal initial tiling. But BSP seems on step ahead of rest by having accuracy as high as LASP++ with run-time as short as SLIC++. Then SLIC++ comes, with relatively lower accuracy with the shortest run-time.

Figure 4.19 depicts the results for rectangular initial tiling. Unlike hexagonal tiling, BSP shows top performance almost for all metrics. In order to compare results numerically, AUC values of all metrics are presented in Table 4.13. BSP has shows similar performance with LASP++ in BR, ASA and BASA metric and performs better than it by 0.01% in UE and $8.2ms$ in run-time metric. As a result BSP has similar accuracy with LASP++ obtained in a 13.5% shorter time.

On the other hand, BSP has performance better than SLIC++ in UE by 0.07%, in BASA by 0.2%, same with it in ASA, and worse than it in BR by 0.5% and by $4.1ms$ in run-time metric. Regarding BASA and run-time performance, BSP shows 0.2% better performance than SLIC++ in a $7,5\%$ longer time. Therefore, it can be stated that both perform close to each other.

When SLIC++ and LASP++ are compared for rectangular initial tiling, LASP++ is better in UE by 0.06% and in BASA by 0.2%. They show same accuracy in ASA metric but SLIC++ has higher performance in BR by 0.5%, $12.3ms$ in run-time. To sum up, LASP++ achieves 0.2% better accuracy in a 22% longer run-time.

Unlike hexagonal initial tiling, the experiments for rectangular initial tiling has more distinguishable results. BSP shows better accuracy than LASP++ with a lower run-time. SLIC++ has close performance to LASP++ achieved in a significantly lower run-time. BSP is a step further than SLIC++ with top accuracy with insignificant

run-time difference.

In this section, a new cost function BSP is proposed that enhances Gaussian distribution assumed general cost function (4.6). BSP improves state of the art methods with no assumptions on general cost function (4.6), and like other proposed cost functions in this chapter, further improves by employing spatial covariance and adaptive spectral variance bounds. Based upon experiments, second order SP-level statistics significantly improves run-time with no significant loss in boundary adherence. As a result, BSP provides image structure following performance as high as LASP++ in a run-time as low as SLIC++.

To sum up, clustering-based methods iteratively assign each pixel the the closest cluster with respect to spectral and spatial distances. A compactness parameter is enhanced to prioritize compactness over spectral similarity. Methods differ in their cost functions; they generally utilize Euclidean distance with different normalization terms. State of the art clustering-based algorithms has made several assumptions in order to simplify the cost function (4.5). Among these assumptions, orthogonal spatial axis assumption is certainly not valid, which decreases the performance by forcing all SP to have the same shape and size.

In this chapter, new cost-functions are proposed for clustering-based methods to improve existing ones. SLIC and LASP cost functions are investigated, and improved first by employing spatial covariance in SLIC+ and LASP+ respectively. With this update, for spatial term Mahalanobis distance employed instead of Euclidian distance. As another improvement, using adaptive spectral global parameters in SLIC++ and LASP++ is proposed. And finally, BSP cost function is proposed, which utilizes Mahalanobis distance both for spectral and spatial term and makes less assumptions compared to preceding ones. BSP decreases computational burden by allowing SP level statistic calculations while preserving high boundary adherence.

As summarized in Table (4.14), SLIC [2] uses a global spectral variance and a spatial variance calculated from expected SP area; LASP [17] uses a spectral variance for each cluster which is minimum of neighbor SPs and a global spatial variance. The proposed cost functions SLIC+ and LASP+ improve those state-of-the-art methods by employing spatial covariance in spatial distance term. SLIC++ and LASP++ further

improve their base methods SLIC+ and LASP+ respectively by using dynamic spectral variance and spectral variance bounds which are updated at each iteration instead of static global ones. Bayesian Superpixels (BSP) proposed in this work, enhances SP-level spectral variances and spatial covariance to employ Mahalanobis distance. As the spectral variance terms might be erroneous at the beginning, spectral variances are bounded with a limit relative to average SP variance for each spectral band. By using per SP Bayesian prior, method gives similar boundary adherence with LASP and LASP+ in a much shorter time comparable to SLIC.

Regarding performance results of BSP, SLIC++ and LASP++ cost functions, it can be concluded that as the accuracy increases run-time performance decreases. Therefore it can not be concluded that one outperforms the another. It depends on the preference of applications employing the SP extraction method. However, in general, BSP is top with lower run-time and higher accuracy and SLIC++ comes second with top run-time and with worst accuracy and LASP++ comes third having high accuracy with worst run-time.

4.6 Number of Iterations

Clustering-based SP extraction methods solves their cost function iteratively. The number of iterations is usually defined as a method specific hyper-parameter. As the number of iterations increases, the run-time also increases, which might make the method useless for some applications due to run-time limit. On the other hand, finishing iterations earlier than it should be, decreases performance since SPs are yet to converge. Therefore, an optimum point should be selected to stop iterations, which is enough for convergence and keeps run-time as short as possible. As clustering-based methods are members of gradient descent-based approach, SPs converge after a certain number of iterations and more iterations beyond convergence do not increase performance. Therefore, the point where SPs are about to converge is defined as the optimum iteration length. To determine this optimum point a set of experiments are conducted, different values of iteration count is tested and metrics are analyzed to find the convergence point.

Table 4.14: Spectral and spatial normalization approaches of analyzed cost functions

Cost Function	**Spectral**	**Spatial**	**Reference**
SLIC	Global constant variance	Expected area	4.9
SLIC+	Global constant variance	Spatial covariance	4.11 (Proposed)
SLIC++	Dynamic average variance	Spatial covariance	4.13 (Proposed)
LASP	Neighbor minimum, global limits	Expected area	4.10
LASP+	Neighbor minimum, global limits	Spatial covariance	4.12 (Proposed)
LASP++	Neighbor minimum, dynamic limits	Spatial covariance	4.14 (Proposed)
BSP	Neighbor regulation, dynamic limits	Spatial covariance	4.15 (Proposed)

As mentioned before, iteration count is a method specific hyper-parameter, however it might be possible to get rid of this parameter. The maximum number of iterations should allow an SP to lose all of its pixels. In a single iteration, at worst SP may transfer all of its boundary pixels to neighboring SPs. For rectangular initial tiling, it takes $edgeLength/2$ iterations for an SP to transfer all of its pixels to other SPs, where $edgeLength$ is SP initial edge length. In other words, iteration count can be said to depend on $edgeLength$, and can be configured independent from number of SPs by setting it relative to edge length, i.e. $\alpha * edgeLength$ where $alpha$ is a constant. Experiments in this section are conducted to find a reasonable value for parameter $alpha$ which removes iteration length from hyper-parameter list.

Table 4.15 shows the configuration of the related experiment. Hexagonal and rectangular initial tiling is used separately, no grid refinement is employed, JC label update criteria is used, SP count is taken as 500, 1000, 1500 and 2000 and $alpha$ search values are range from 0.25 to 2.50 in 0.25 steps. Proposed cost-functions SLIC++ 4.11, LASP++ 4.12, BSP 4.15 are tested in this experiment. Common metrics through out this work are used as performance measures, and proposed BASA AUC with run-time AUC are also presented to reveal the numerical differences.

Figure 4.21, 4.22, 4.23 and 4.24 depict the performance with respect to iteration count for hexagonal and rectangular tiling and 500, 1000, 1500 and 2000 SPs respectively. As claimed above, it is clearly seen on figures that curves start to converge at some point even for different number of SPs. Based on figures, this point can be decided as $alpha = 1.25$. Regarding run-time performance, as expected, run-time increases proportional to iteration count, as more iterations take more time. Tables 4.16 and 4.18 show related AUC values for BASA and run-time experiments respectively for HEX ad RECT tiling for 500 and 1500 SPs. Tables 4.17 and 4.19 show related AUC values for 1500 and 2000 SPs. Consistently, as alpha increases for both HEX and RECT, BASA values converge, and this point can be extracted as 1.25 for all cost functions. Beyond this point there is no significant improvement on any metric. It is inefficient to iterate beyond this point since SPs has already converged. As run-time increases proportional to selected α value, selecting higher values decreases run-time performance.

Table 4.15: Number of iteration experiment configuration

Initial Tiling	**Grid Ref.**	**#SPs**	**Cost Function**	**Alpha**	**Metrics**
Hex	Off	500	SLIC++ (4.11)	0,25	BR
Rect		1000	LASP++ (4.14)	0,50	UE
		1500	BSP (4.15)	0,75	ASA
		2000		1,00	BASA
				1,25	Run-time
				1,50	BASA AUC
				1,75	Run-time AUC
				2,00	
				2,25	
				2,50	

To conclude, during the experiments in this section, different iteration counts are tested and SP performance metrics are observed with respect to changing number of iterations. It is seen that independent from experiment configuration, metrics converge after a certain iteration length which is proportional to average SP area. Minimum iteration count which leads to convergence is observed as $1.25xedgeLength$. Run-time results shows that computation time is linearly dependent to the iteration count. Setting the iteration count proportional to average SP area makes it possible to get rid of iteration count hyper-parameter.

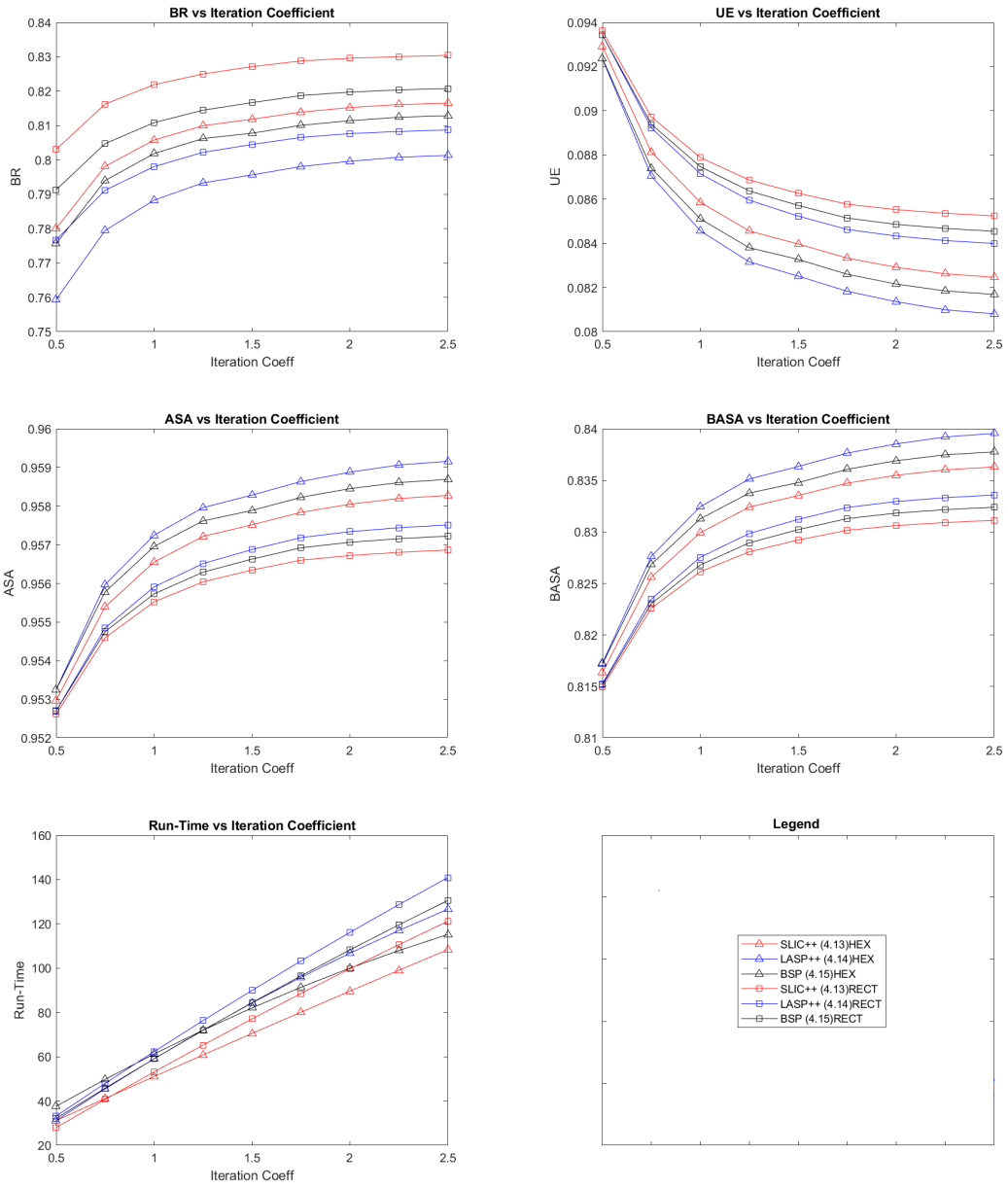


Figure 4.21: Metric convergence wrt. iteration count for 500 SPs, for hexagonal and rectangular initial tiling where iteration coefficient is $alpha$

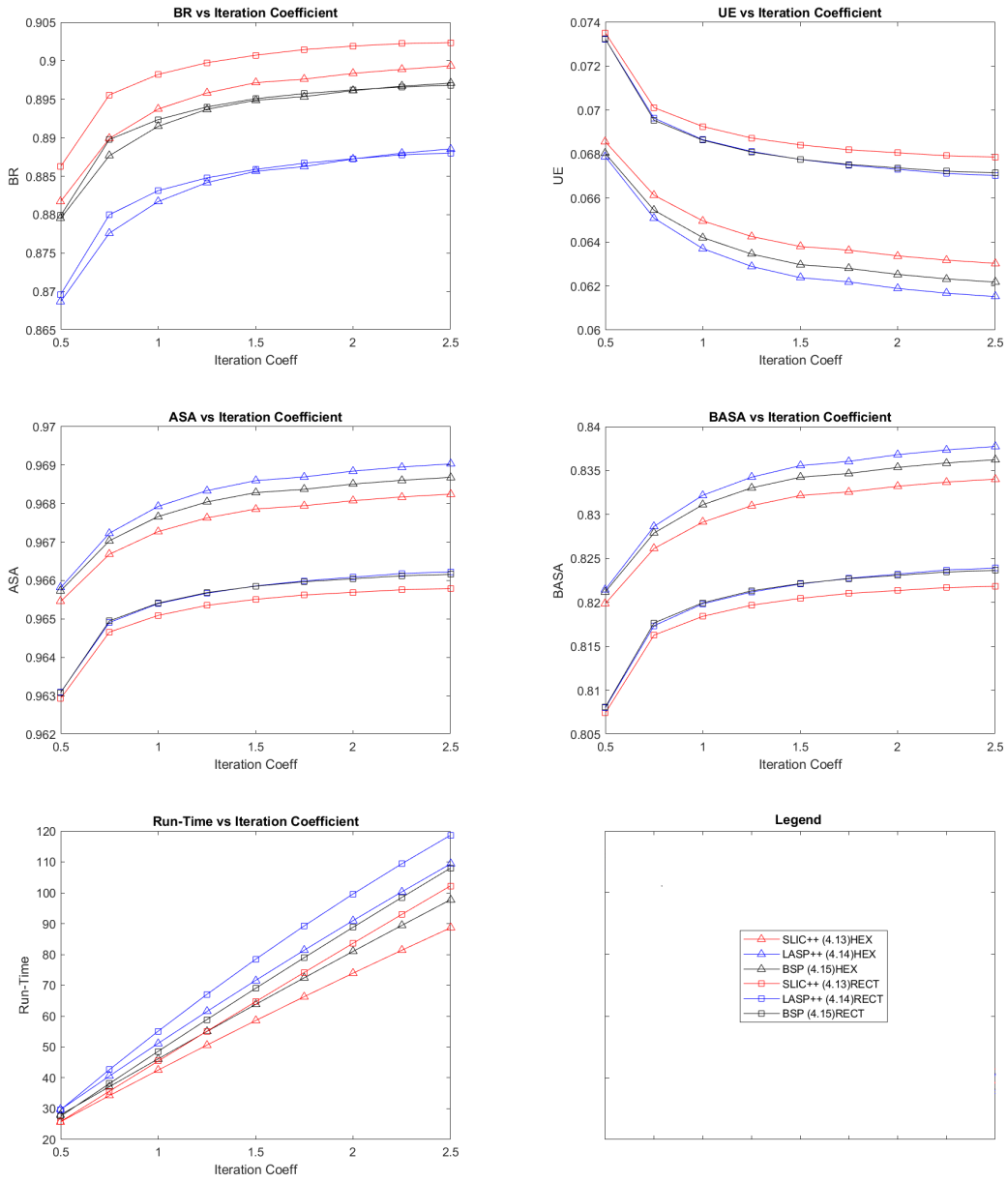


Figure 4.22: Metric convergence wrt. iteration count for 1000 SPs, for hexagonal and rectangular initial tiling

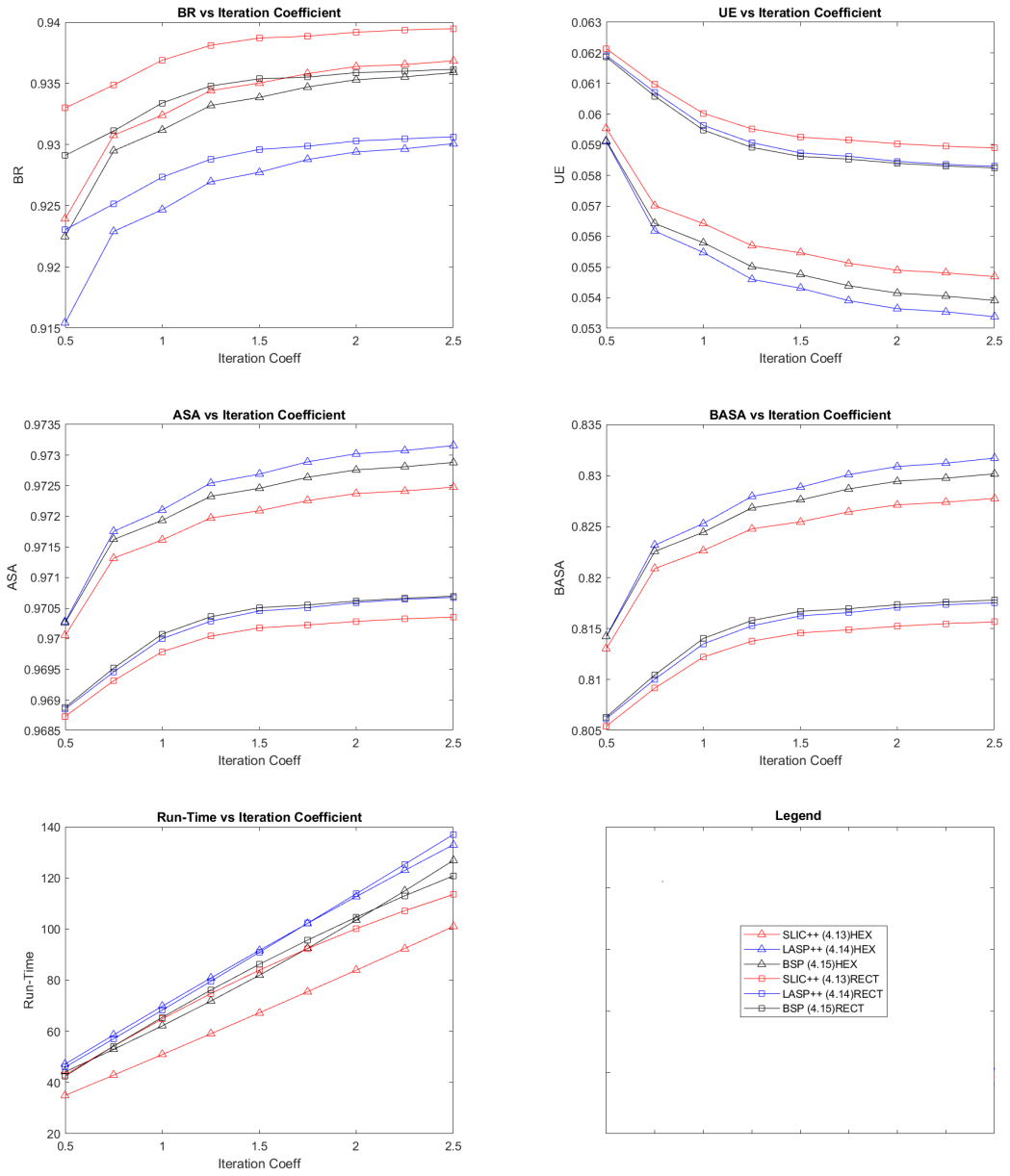


Figure 4.23: Metric convergence wrt. iteration count for 1500 SPs, for hexagonal and rectangular initial tiling

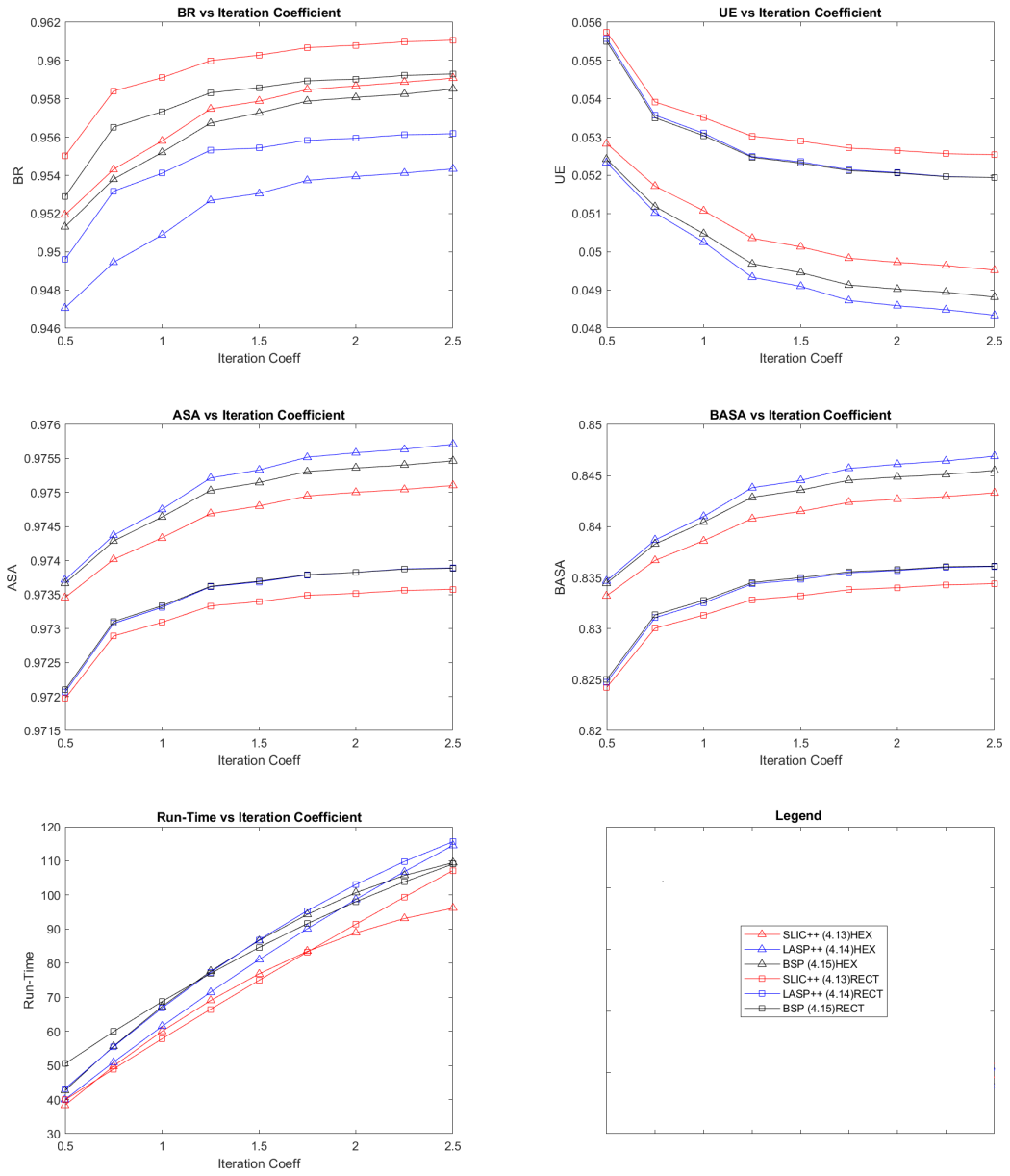


Figure 4.24: Metric convergence wrt. iteration count for 2000 SPs, for hexagonal and rectangular initial tiling

Table 4.16: BASA results of iteration count experiment for 500 and 1000 SPs

BASA	**Tiling**	**Hexagonal**			**Rectangular**		
#SPs	**alpha**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**
500	**0,25**	79,8%	79,8%	79,9%	79,3%	79,3%	79,3%
	0,50	81,6%	81,7%	81,7%	81,5%	81,5%	81,5%
	0,75	82,6%	82,8%	82,7%	82,3%	82,3%	82,3%
	1,00	83,0%	83,2%	83,1%	82,6%	82,8%	82,7%
	1,25	83,2%	83,5%	83,4%	82,8%	83,0%	82,9%
	1,5	83,4%	83,6%	83,5%	82,9%	83,1%	83,0%
	1,75	83,5%	83,8%	83,6%	83,0%	83,2%	83,1%
	2,00	83,5%	83,9%	83,7%	83,1%	83,3%	83,2%
	2,25	83,6%	83,9%	83,7%	83,1%	83,3%	83,2%
	2,50	83,6%	84,0%	83,8%	83,1%	83,4%	83,2%
1000	**0,25**	79,6%	79,6%	79,7%	78,6%	78,5%	78,6%
	0,50	82,0%	82,1%	82,1%	80,7%	80,8%	80,8%
	0,75	82,6%	82,9%	82,8%	81,6%	81,7%	81,8%
	1,00	82,9%	83,2%	83,1%	81,8%	82,0%	82,0%
	1,25	83,1%	83,4%	83,3%	82,0%	82,1%	82,1%
	1,50	83,2%	83,6%	83,4%	82,0%	82,2%	82,2%
	1,75	83,3%	83,6%	83,5%	82,1%	82,3%	82,3%
	2,00	83,3%	83,7%	83,5%	82,1%	82,3%	82,3%
	2,25	83,4%	83,7%	83,6%	82,2%	82,4%	82,3%
	2,50	83,4%	83,8%	83,6%	82,2%	82,4%	82,4%

Table 4.17: BASA results of iteration count experiment for 1500 and 2000 SPs

BASA	**Tiling**	**Hexagonal**			**Rectangular**		
#SPs	**alpha**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**
1500	**0,25**	80,0%	80,0%	80,1%	78,8%	78,8%	78,8%
	0,50	81,3%	81,4%	81,4%	80,5%	80,6%	80,6%
	0,75	82,1%	82,3%	82,3%	80,9%	81,0%	81,0%
	1,00	82,3%	82,5%	82,4%	81,2%	81,4%	81,4%
	1,25	82,5%	82,8%	82,7%	81,4%	81,5%	81,6%
	1,50	82,5%	82,9%	82,8%	81,5%	81,6%	81,7%
	1,75	82,6%	83,0%	82,9%	81,5%	81,7%	81,7%
	2,00	82,7%	83,1%	82,9%	81,5%	81,7%	81,7%
	2,25	82,7%	83,1%	83,0%	81,5%	81,7%	81,8%
	2,50	82,8%	83,2%	83,0%	81,6%	81,8%	81,8%
2000	**0,25**	80,1%	80,1%	80,1%	81,3%	81,4%	81,4%
	0,50	83,3%	83,5%	83,4%	82,4%	82,5%	82,5%
	0,75	83,7%	83,9%	83,8%	83,0%	83,1%	83,1%
	1,00	83,9%	84,1%	84,0%	83,1%	83,3%	83,3%
	1,25	84,1%	84,4%	84,3%	83,3%	83,4%	83,5%
	1,50	84,1%	84,5%	84,4%	83,3%	83,5%	83,5%
	1,75	84,2%	84,6%	84,5%	83,4%	83,5%	83,6%
	2,00	84,3%	84,6%	84,5%	83,4%	83,6%	83,6%
	2,25	84,3%	84,6%	84,5%	83,4%	83,6%	83,6%
	2,50	84,3%	84,7%	84,5%	83,4%	83,6%	83,6%

Table 4.18: Average run-time (area under curve) of iteration count experiment for 500 and 1000 SPs

RT	**Tiling**	**Hexagonal**			**Rectangular**		
#SPs	**alpha**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**
500	**0,25**	20,7	19,6	23,7	16,4	21,1	19,5
	0,50	30,9	33,2	37,0	28,9	35,1	33,0
	0,75	41,0	46,5	49,5	41,2	48,9	46,2
	1,00	51,0	59,3	61,2	53,2	62,5	59,1
	1,25	60,8	71,7	72,1	65,1	76,0	71,8
	1,50	70,6	83,7	82,2	76,8	89,4	84,2
	1,75	80,2	95,2	91,6	88,3	102,6	96,2
	2,00	89,7	106,3	100,1	99,5	115,7	108,1
	2,25	99,0	117,0	108,0	110,6	128,7	119,6
	2,50	108,3	127,3	115,0	121,5	141,5	130,8
1000	**0,25**	17,3	18,5	17,8	15,2	16,5	16,2
	0,50	25,9	29,6	27,4	25,4	29,9	27,2
	0,75	34,3	40,5	36,8	35,5	42,8	38,0
	1,00	42,5	51,2	46,1	45,4	55,1	48,5
	1,25	50,6	61,5	55,1	55,2	67,0	58,9
	1,50	58,5	71,6	64,0	64,8	78,4	69,1
	1,75	66,3	81,4	72,6	74,3	89,2	79,1
	2,00	73,9	91,0	81,1	83,7	99,6	88,9
	2,25	81,4	100,3	89,4	93,0	109,4	98,5
	2,50	88,7	109,3	97,5	102,1	118,7	107,9

Table 4.19: Average run-time (area under curve) of iteration count experiment for 1500 and 2000 SPs

BASA	**Tiling**	**Hexagonal**			**Rectangular**		
#SPs	**alpha**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**	**SLIC (4.9)**	**LASP (4.10)**	**BSP (4.15)**
500	**0,25**	28,4	37,3	37,7	31,4	34,3	31,0
	0,50	35,8	48,2	45,3	43,3	45,6	43,1
	0,75	43,3	59,1	53,5	54,4	56,9	54,6
	1,00	51,0	70,0	62,3	64,9	68,3	65,6
	1,25	59,0	80,7	71,7	74,7	79,7	76,1
	1,50	67,1	91,4	81,6	83,9	91,1	86,1
	1,75	75,3	102,0	92,2	92,3	102,5	95,6
	2,00	83,8	112,5	103,3	100,1	113,9	104,5
	2,25	92,4	123,0	114,9	107,2	125,4	113,0
	2,50	101,3	133,3	127,2	113,6	136,8	121,0
1000	**0,25**	24,7	26,9	27,1	30,8	32,8	40,0
	0,50	37,7	39,0	41,9	40,0	45,0	50,2
	0,75	49,5	50,5	55,3	49,0	56,4	59,8
	1,00	60,0	61,4	67,2	57,8	67,1	68,7
	1,25	69,2	71,7	77,8	66,5	77,1	77,0
	1,50	77,1	81,4	86,9	75,0	86,3	84,7
	1,75	83,7	90,5	94,6	83,3	94,9	91,7
	2,00	89,1	99,0	100,9	91,4	102,7	98,1
	2,25	93,1	106,9	105,7	99,4	109,8	103,9
	2,50	95,9	114,2	109,2	107,2	116,2	109,0

CHAPTER 5

COMPARISON OF SUPERPIXEL EXTRACTION METHODS

In Chapter 4, steps of a general clustering-based SP extraction method are analyzed in detail. For each step alternative approaches are investigated and some alternatives are proposed to enhance the performance. Then, performance of those alternatives together with proposed ones are evaluated with a set of experiments. At the end it is concluded that, 1) employing spatial and spectral adaptiveness increases performance of all investigated cost function alternatives, 2) refinement of initial grid does not improve the performance significantly, 3) number of iterations can be removed from hyper-parameters by setting it relative to average SP area.

Based on these conclusions, following sections will propose SP extraction methods and compare their performance with top performing state-of-the-art methods.

5.1 Proposed Alternative Superpixel Extraction Methods

This section proposes alternative SP extraction methods due to outcome of analysis in Chapter 4. Hexagonal and rectangular tiling should remain as initial tiling options as they provide different SP neighborhood topology which should be selected by the application employing the SP extraction. Proposed JC label update constraint should be employed since connectivity is required to be enforced from the beginning. No initial grid refinement is employed and cost functions BSP (4.15), SLIC++ (4.13) and LASP++ (4.14) are selected due to their top performance.

Proposed SP extraction methods are as follows:

1. SLIC++/R: employs rectangular initial tiling and SLIC++ (4.13) cost-function.

Table 5.1: Summary of proposed SP extraction methods

Proposed Method	**Initial Tiling**	**Grid Ref**	**Label Update**	**Cost Function**
SLIC++/R	Rectangular	None	JC	SLIC++ (4.13)
SLIC++/H	Hexagonal	None	JC	SLIC++ (4.13)
LASP++/R	Rectangular	None	JC	LASP++ (4.14)
LASP++/H	Hexagonal	None	JC	LASP++ (4.14)
BSP/R	Rectangular	None	JC	BSP (4.15)
BSP/H	Hexagonal	None	JC	BSP (4.15)

2. SLIC++/H: employs hexagonal initial tiling and SLIC++ (4.13) cost-function.
3. LASP++/R: employs rectangular initial tiling and LASP++ (4.14) cost-function.
4. LASP++/H: employs hexagonal initial tiling and LASP++ (4.14) cost-function.
5. BSP/R: employs rectangular initial tiling and BSP (4.15) cost-function.
6. BSP/H: employs hexagonal initial tiling and BSP (4.15) cost-function.

Rather than proposing a single method, a family of methods are proposed such that depending on desired Sp neighborhood topology and selection of accuracy vs. runtime performance there are alternative solutions. In the next section, performance of these 6 methods is compared to the top performing state-of-the-art methods.

5.2 Experiments

This section is dedicated to compare performance of proposed methods against top performing state-of-the-art methods. In the experiments, The Berkeley Segmentation Dataset and Benchmark BSD300 data set ([31]) is used. Dataset contains 300 481x321 RGB images and their respective ground truth segmentation images. Methods are compared in terms of Boundary Recall 3.1, Under-segmentation Error 3.4,

Table 5.2: State of the art comparison experiment configuration

SP Extraction Method	**#SPs**	**Metrics**
CRS [16]	500	BR
ERGC [12]	1000	UE
ERS [9]	2000	ASA
ETPS [15]		BASA
SEEDS [14]		Run-time
SLIC [2]		BR AUC
SLIC++/H (4.13)		UE AUC
LASP++/H (4.14)		ASA AUC
BSP/H (4.15)		BASA AUC
SLIC++/R (4.13)		Run-time AUC
LASP++/R (4.14)		
BSP/R (4.15)		

Achievable Segmentation Accuracy 3.5, Boundary Achievable Segmentation Accuracy 3.7 with respect to Compactness 3.8.

For comparison, best performing algorithms are selected from the work of Stuart [4] where methods are evaluated from various perspectives and ranked according to their overall performance. The selected methods to be tested against proposed methods are; CRS [16], ERGC [12], ERS [9], ETPS [15], SEEDS [14] and SLIC [2].

For comparison, a series of experiments are executed and resultant performance metrics are analyzed. Table 5.2 summarizes the configuration of experiments. In order to cover all compactness region of interest, hyper-parameters of state of the art methods are adjusted so, nevertheless some methods either has no control over compactness or they only produce SPs within a limited compactness region. Among selected ones, only SEEDS [14] does not provide control over compactness of generated SPs. So in order to observe its performance in compactness region of interests existing hyper-parameters are adjusted.

In addition, as described in Chapter 2, one of the drawbacks of some state-of-the-art methods is producing disconnected clusters. In order to get-rid-of these disconnected regions, new SPs are created from those disconnected ones with a post-processing step which results higher SP count than desired. Producing higher number of SPs than required is an indication for corresponding methods having no-control on number of SPs although it is not stated explicitly in literature. Proposed methods, uses proposed JC label update criteria to enforce cluster connectivity from the beginning and provides full-control over compactness and number of generated pixels.

Regarding the metric figures of this section, to make it easier to read, legend is aligned such that dashed curves represent state-of-the-art methods and continuous ones represent the proposed methods. Additionally, some state-of-the-art methods may reside out of figure window for some metrics because of their poor performance. In order to analyze good performing ones clearly, only region of importance of each figure is just drawn. As stated above, not every state-of-the-art method provide control over compactness of SPs so such methods like ETPS may only be seen within a very limited compactness region.

Figure 5.1 depicts boundary recall metric for 500, 1000 and 2000 respectively. It is clearly seen that proposed methods significantly outperform state-of-the-art ones except ETPS [15]. Especially when compared to SLIC, SLIC++/H and SLIC++/R significantly improve the performance, which confirms employing SP dependent global and local terms for normalization of spectral and spatial distances respectively is useful. Only for the 1000 SPs case ETPS performs slightly better than the proposed ones, nevertheless its performance is worse for 500 and 2000 SPs case. Besides within a very limited region, CRS shows close performance to proposed ones for 2000 SPs case, but worse for the rest.

Under-segmentation error for the compared alternatives is depicted in Figure 5.2. For 500 SPs case, proposed methods over-perform state-of-the art ones. On the other hand for 1000 and 2000 SPs, proposed methods employing HEX initial tiling still over-perform rest of methods but ones employing RECT initial tiling comes behind ETPS, CRS and ERGC. Performance of the proposed alternatives is more robust against compactness than state-of-the-art methods. Similar to metric BR results, ETPS has a

lower UE in some limited region for 1000 SPs but performance is worse for 500 and 2000 SPs. Finally, SLIC++ cost function significantly outperforms SLIC.

Figure 5.3 shows achievable segmentation accuracy performance of methods. The results are similar to BR and UE experiments, the proposed alternatives with hexagonal initial tiling clearly outperform state-of-the-art. For 500 SPs, proposed methods starting with RECT tiling has better performance than state-of-the-art ones for high compactness levels. As SP number increases ETPS, CRS and ERGC shows better performance than proposed alternatives employing rectangular initial tiling. Similar to UE and BR, ETPS has some good performance within a limited region for only 1000 SP and SLIC++ cost function significantly outperforms SLIC.

For boundary achievable segmentation accuracy which is depicted in Figure 5.3, results are similar to other metrics. Proposed alternatives employing hexagonal initial tiling outperforms the rest for all SP counts. Additionally, as compactness level increases the difference between state-of-the-art and proposed methods increases, which makes proposed methods a good alternative especially for the applications requiring high compact SPs. Proposed methods employing rectangular initial tiling performs better than state-of-the-art for 500 SPs, but worse than ETPS, CRS and ERGC when SP number gets higher.

Metrics are more meaningful when evaluated together with run-time performance. For instance, if a well performing method achieves this in a rather longer time, it becomes unfair to rank it above other methods as poorly performing ones may perform better when such a time is entitled by applications employing these methods. Therefore, Figure 5.5 shows related run-time performance of methods. Here, SEEDS has the lowest run-time for 500 and 1000 SP counts. But proposed ones come after having performance close to SEEDS and staying in acceptable limits. Especially for 2000 SPs, proposed SLIC++/R and SLIC++/H have the minimum run-time performance outperforming the rest, including SEEDS. Regarding accuracy metrics, ETPS which has best scores among state-of-the-art methods, shows worse run-time performance than the proposed ones and SEEDS and moreover gap increases as the number of SPs is increased. Although, ETPS has slightly better performance in a limited region for 1000 SPs metrics, when compared with the proposed alternatives in terms of run-time,

it is not possible to conclude that ETPS outperforms the proposed alternatives.

As a result, after analyzing performance graphics of methods, proposed methods SLIC++/H, LASP++/H and BSP/H take top place in all metrics for all SP numbers with a significant difference. The other proposed methods SLIC++/R, LASP++/R and BSP/R show better performance than state-of-the-art methods as number of SPs is decreased on the other cases 3 of the state of the art methods ETPS, CRS and ERGC have higher performance than RECT initial tiling proposed methods. But CRS and ERS achieves those scores with far worse run-time performance. Additionally, it is clearly seen that SLIC++/R and SLIC++/H over-performs SLIC with proposed improvements.

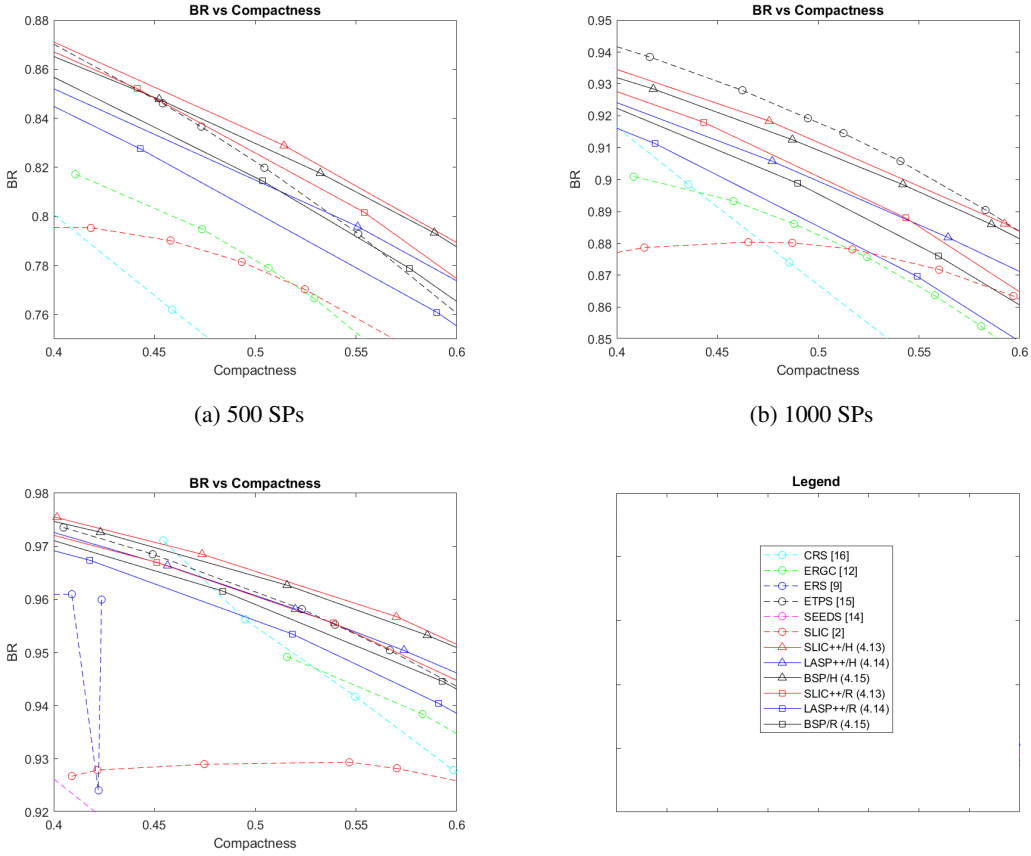
Tables 5.3 and 5.4 show area under curve (AUC) values of BR, UA, ASA and BASA, RT respectively which are calculated from the figures presented in this section. Although figures help to see performance in all compactness levels, AUC values enables numerical comparison by providing average values. As expected, AUC values are compatible with figures, and as seen in tables performance of methods increase as number of SPs are increased but for the run-time state-of-the-art methods performance are decreased significantly. CRS and ERS reaches up to 1 second, ERGC and ETPS runs in 3 times and 1.5 times slower than proposed ones respectively. Although SEEDS run-time is top here but it is the worst in other boundary metrics. SLIC++/H and SLIC++/R has a significantly better run-time than published version of SLIC. Also for other metrics, SLIC++/R outperforms SLIC, which means the proposed enhancement over SLIC is succeeded. Tables 5.3 and 5.4 also clarify the best performing methods on the average; proposed alternatives with hexagonal initial tiling outperforms state-of-the-art.

As presented in Tables 5.3 and 5.3, despite having shortest run-time, SEEDS has the worst BR, UE, ASA and BASA score 19%, 5%, 2.5% and 10% behind the top respectively for 500 SP. ERS has good score in BR but poorly performed in UE, ASA, BASA and run-time, moreover it has the worst run-time score almost 20 times behind the fastest method, depending on SP count. ETPS has the best scores among state-of-the-art methods in BR, UE, ASA and BASA but behind the top especially 1-1.5% in BASA and 50-150% in run-time. CRS has poorly performed in all the metrics, it

is the second poorly performed in terms of run-time, being 10-20 times slower than the fastest one. ERGC is better than CRS but its performance is poor compared to top ones, 1.5% and 400% behind in BASA and run-time metrics. SLIC did not perform well with scores 3-5%, 1-2%, 0.5%, 3-4% and 75% behind the top performing in BR, UE, ASA, BASA and run-time.

For 500 SPs, SLIC++/H and SLIC++/R improves SLIC by 5.9% and 5,1% in BR, 1.25% and 0,9% in UE, 0.7% and 0,5% in ASA, 2.6% and 1,9% in BASA and 22.0% and 20,0% in run-time respectively. For 1000 SPs, SLIC++/H and SLIC++/R improve SLIC performance by 3.5% and 2,4% in BR, 0.8% and 0,4% in UE, 0.5% and 0,2% in ASA, 2.4% and 1,2% in BASA and 36.0% and 32,0% in run-time respectively. And finally for 2000 SPs, improvements are 3.7% and 3,2% in BR, 1.12% and 0,85% in UE, 0.6% and 0,4% in ASA, 3.6% and 2,8% in BASA and 19.0% and 20,0% in run-time respectively.

SLIC++/H, LASP++/H and BSP/H outperform rest of the methods in all boundary performance metrics AUC performance for all SP counts. Moreover they have the lowest run-time after SEEDS. On the other had, SLIC++/R, LASP++/R and BSP/R has better performance than state-of-the art methods when run-time is concerned. Only ERGC has a similar high boundary performance achieved in a longer time. As a result it can be concluded that, enhancements proposed in this work significantly improve performance and at the end top boundary adherence is achieved with the lowest run-time. Besides, it is clearly seen from the metric figures, one of the advantage of the proposed algorithms is compactness is controllable for the entire compactness range. Moreover, rate of change of performance with respect to compactness is lower when compared to state-of-the-art methods, especially along the interval of importance, their performance is almost flat meaning that at every level of compactness same high performance can be achieved.



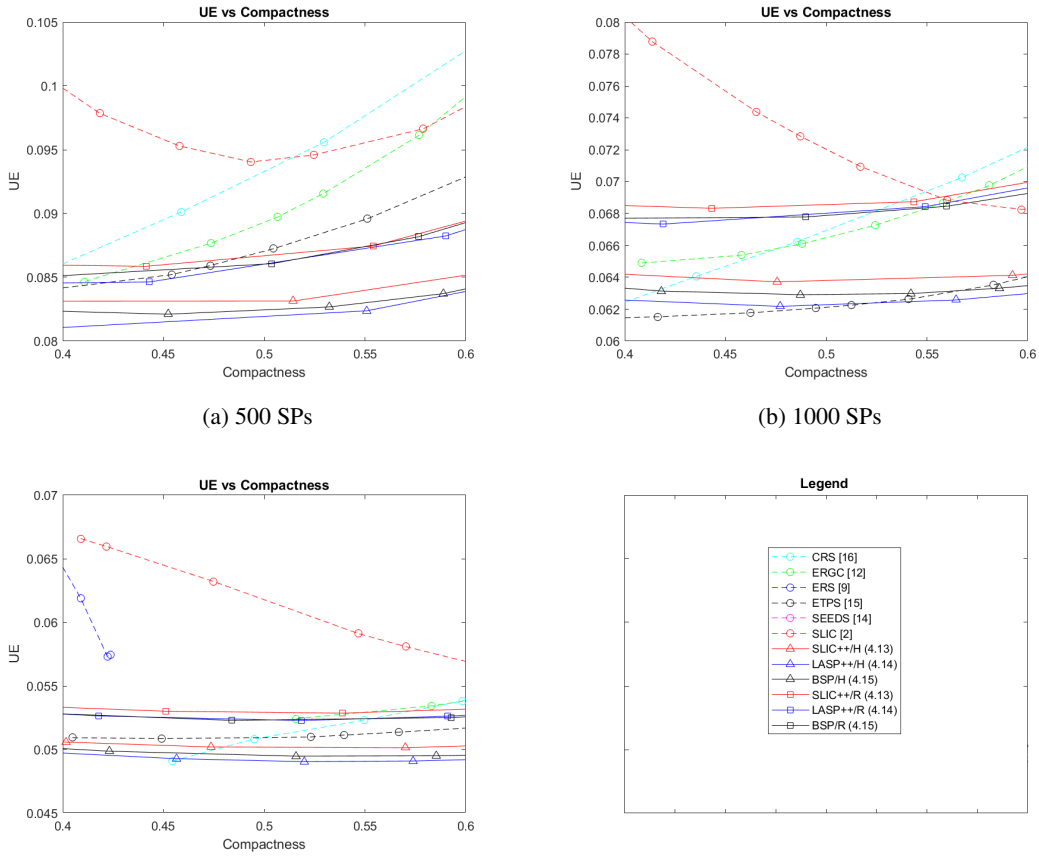
(a) 500 SPs

(b) 1000 SPs

(c) 2000 SPs

(d) Legend

Figure 5.1: Boundary recall performance of state-of-the-art and proposed methods



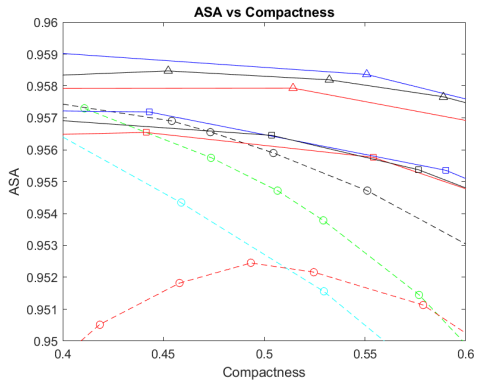
(a) 500 SPs

(b) 1000 SPs

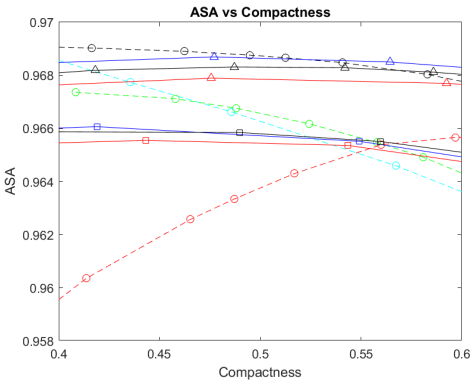
(c) 2000 SPs

(d) Legend

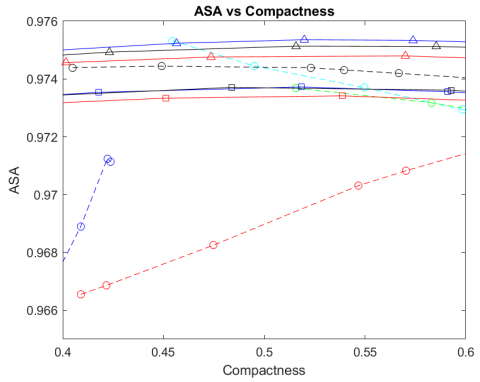
Figure 5.2: Under-segmentation error performance of state-of-the-art and proposed methods



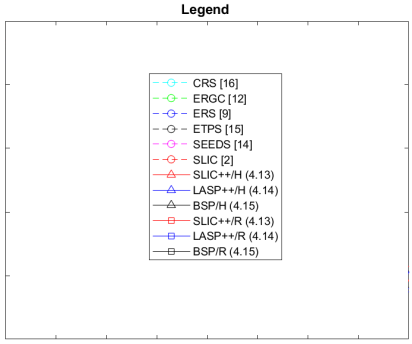
(a) 500 SPs



(b) 1000 SPs

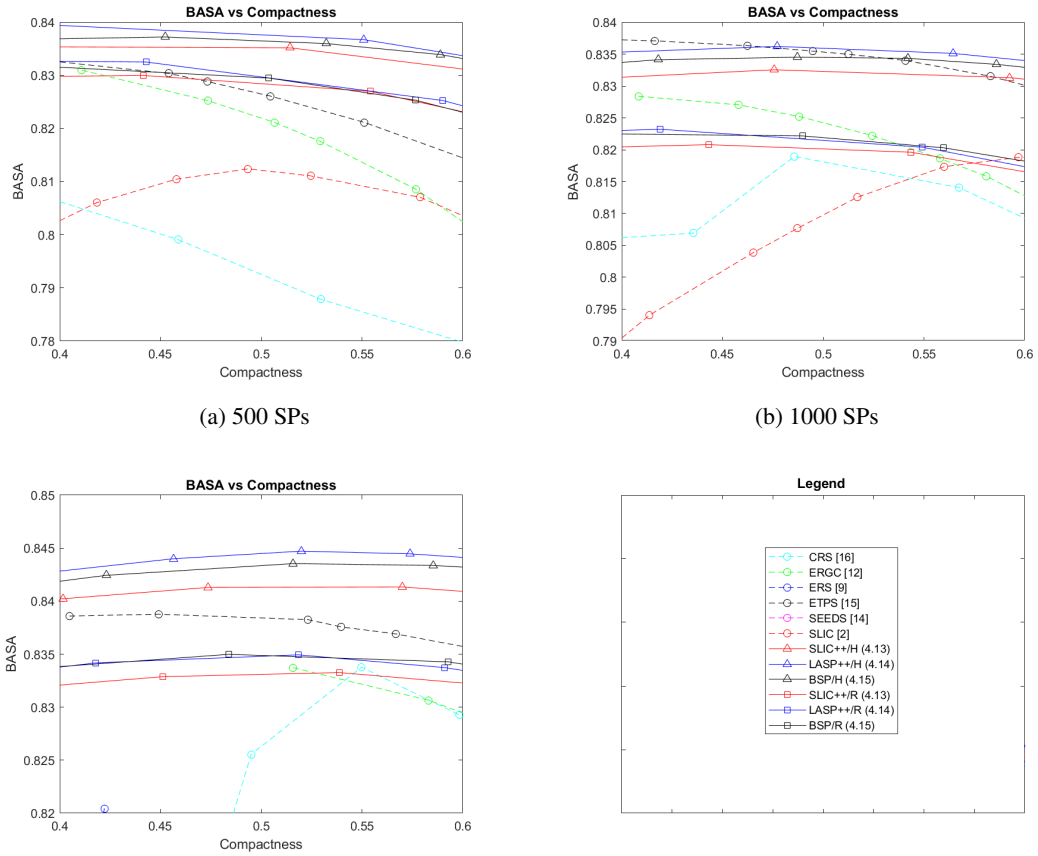


(c) 2000 SPs



(d) Legend

Figure 5.3: Achievable segmentation accuracy performance of state-of-the-art and proposed methods



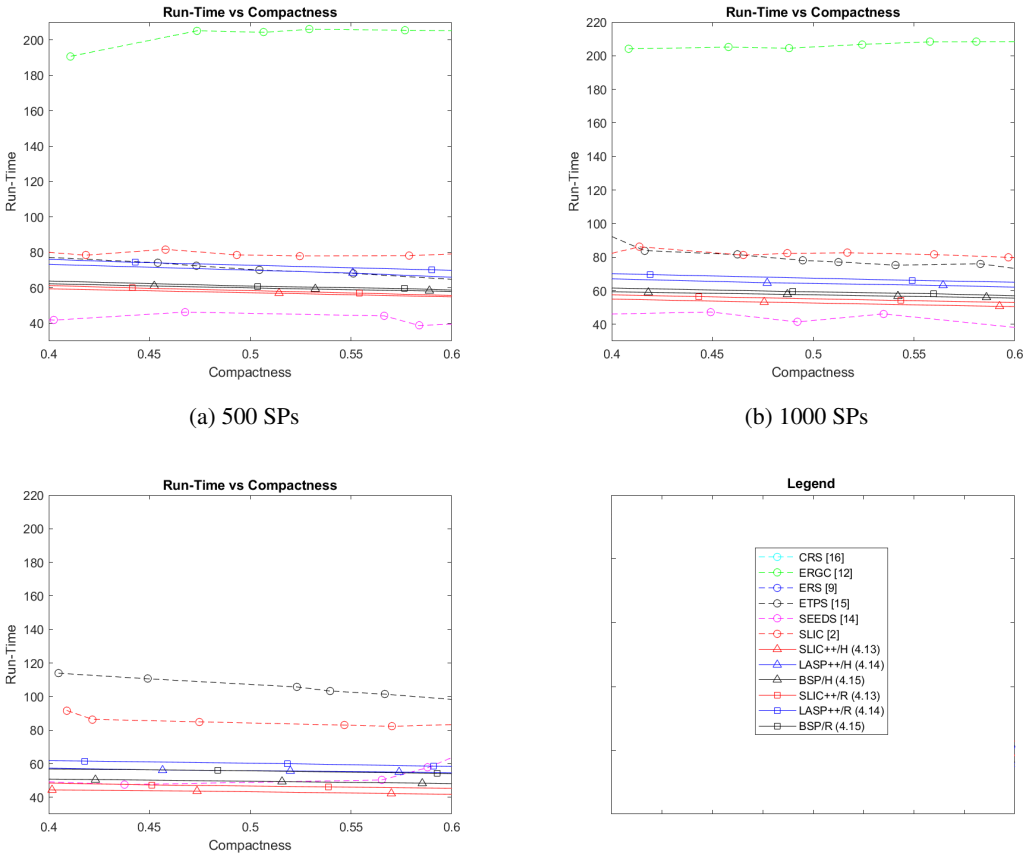
(a) 500 SPs

(b) 1000 SPs

(c) 2000 SPs

(d) Legend

Figure 5.4: Boundary achievable segmentation accuracy performance of state-of-the-art and proposed methods



(a) 500 SPs

(b) 1000 SPs

(c) 2000 SPs

(d) Legend

Figure 5.5: Run-time performance of state-of-the-art and proposed methods

Table 5.3: BR, UE and ASA AUC performance of state-of-the-art and proposed methods for 500, 1000 and 2000 SPs. All values are in percentage

	BR			**UE**			**ASA**		
	500	**1000**	**2000**	**500**	**1000**	**2000**	**500**	**1000**	**2000**
CRS [16]	73,5	86,7	94,7	9,37	6,71	5,18	95,3	96,6	97,4
ERGC [12]	77,3	87,9	94,2	9,08	6,70	5,31	95,4	96,6	97,3
ERS [9]	82,3	89,9	96,0	12,14	9,33	7,22	93,9	95,3	96,4
ETPS [15]	82,0	**91,6**	96,0	8,75	**6,23**	5,11	95,6	**96,9**	97,4
SEEDS [14]	64,5	78,2	86,8	13,30	10,70	9,37	93,3	94,6	95,3
SLIC [2]	77,4	87,6	92,8	9,59	7,27	6,15	95,1	96,3	96,9
SLIC++/H (4.13)	**83,3**	91,1	**96,5**	8,34	6,39	5,03	95,8	96,8	**97,5**
LASP++/H (4.14)	81,5	89,9	96,0	**8,19**	6,24	**4,92**	**95,9**	**96,9**	**97,5**
BSP/H (4.15)	82,9	90,8	96,4	8,26	6,31	4,96	95,8	96,8	**97,5**
SLIC++/R (4.13)	82,5	90,0	96,0	8,69	6,87	5,30	95,6	96,5	97,3
LASP++/R (4.14)	80,2	88,5	95,5	8,60	6,81	5,25	95,7	96,6	97,4
BSP/R (4.15)	81,4	89,4	95,8	8,64	6,81	5,25	95,6	96,6	97,4

Table 5.4: BASA and Run-time AUC performance of state-of-the-art and proposed methods for 500, 1000 and 2000 SPs. BASA values are in percentage

	BASA			**Run-time**		
	500	**1000**	**2000**	**500**	**1000**	**2000**
CRS [16]	79,3	81,3	82,7	546,0	833,8	1049,5
ERGC [12]	81,9	82,3	83,2	203,4	206,2	224,7
ERS [9]	76,0	75,7	77,4	813,2	769,8	1004
ETPS [15]	82,5	**83,5**	83,8	71,0	79,1	106,4
SEEDS [14]	73,5	72,4	71,7	**44,0**	**44,1**	50,1
SLIC [2]	80,9	80,8	80,5	79,1	82,4	84,4
SLIC++/H (4.13)	83,5	83,2	84,1	57,3	52,7	43,2
LASP++/H (4.14)	**83,8**	**83,5**	**84,4**	70,0	66,7	55,9
BSP/H (4.15)	83,6	83,4	84,3	60,1	57,9	49,6
SLIC++/R (4.13)	82,8	82,0	83,3	58,5	55,2	**46,8**
LASP++/R (4.14)	83,0	82,1	83,4	72,9	67,5	60,2
BSP/R (4.15)	82,9	82,1	83,5	61,3	60,6	55,7

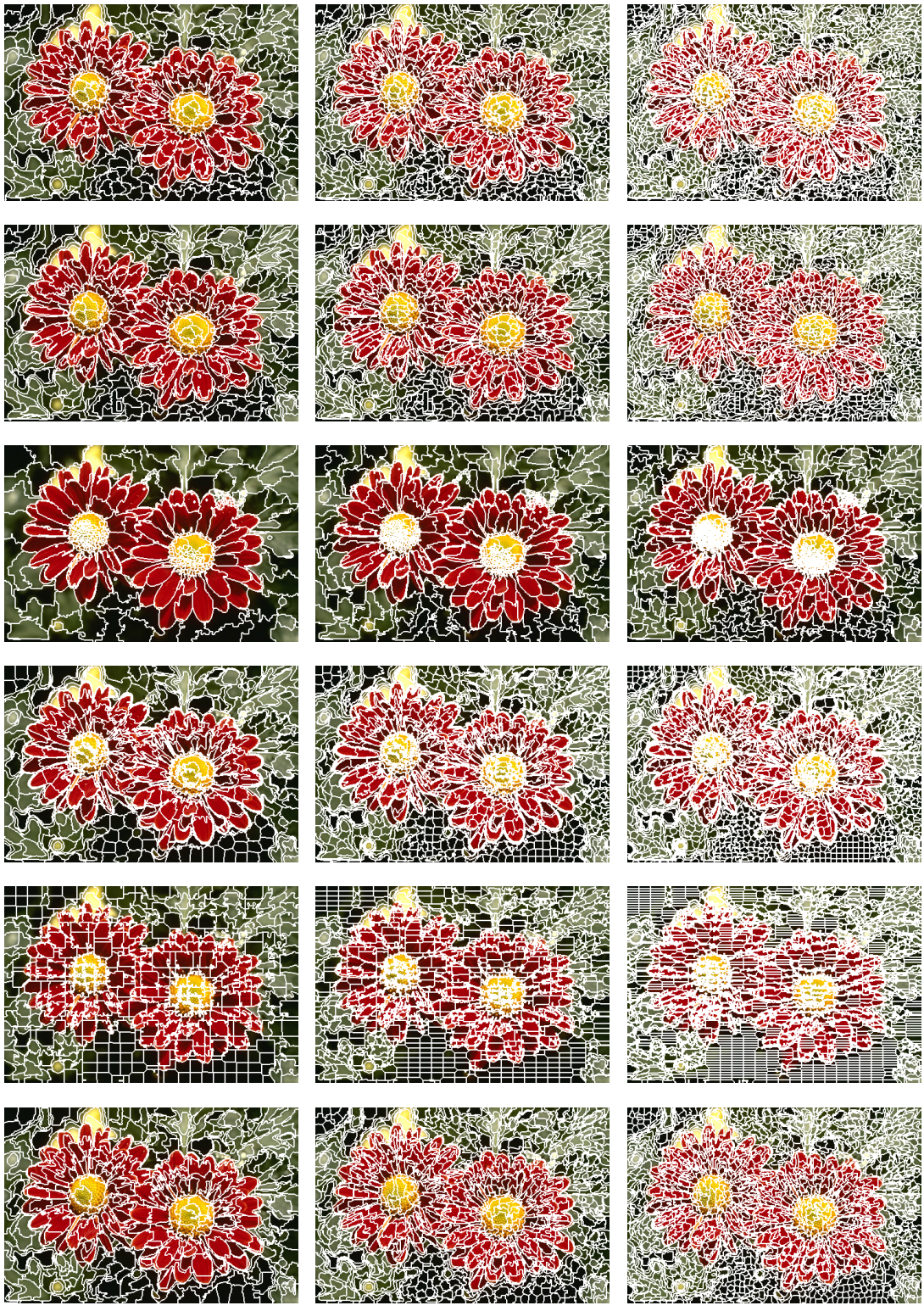


Figure 5.6: Sample outputs of state-of-the-art SP extraction methods: CRS [16], ERGC [12], ERS [9], ETPS [15], SEEDS [14] and SLIC [2] are presented from top to bottom and 500, 1000 and 2000 SPs from left to right respectively.

Figure 5.7: Sample outputs of proposed SP extraction methods: SLIC++/H, SLIC++/R, LASP++/H, LASP++/R, BSP/H and BSP/R are presented from top to bottom and 500, 1000 and 2000 SPs from left to right respectively.

CHAPTER 6

CONCLUSION

In this work, the framework of clustering-based SP extraction methods are investigated in detail and alternative approaches are proposed to improve the performance of existing methods. In addition to requirements of SPs, preserving regular grid structure is given importance in order to be able to feed generated SPs to neural network applications. Grid structure can be preserved with connectivity enforced label updates like proposed Just Connected (JC) constraint.

Regular 4 or 8 connected label updates in literature produce disconnected clusters that need to be connected via a post-processing step, which either breaks regular grid structure and disables the control over generated number of SPs or results in poor performance. Therefore JC label update criteria is proposed in this work to enforce connectivity from the beginning.

Hexagonal and rectangular initial grid alternatives are analyzed separately due to their different application areas. These alternatives results in different SP neighborhood topology, while hexagonal tiling is applicable for graph-based methods, rectangular grid fits better to neural network applications.

Based on literature work claiming refinement of initial tiling increases performance, tiling refinement concept is investigated. However, refinement proposed by Ince [17] breaks regular grid structure that is out of scope of this work. In order to analyze the effect of refinement, Edge-based Refinement method is proposed which preserves initial neighborhood topology. Experiments reveal that such a refinement does not improve the performance as expected. Instead, spending the time consumed in this step to the iteration phase results in similar performance.

Methods in literature are configured through method specific hyper-parameters. Each method has a different number of hyper-parameters to be adjusted which can cause ambiguity. Moreover, as the number of parameters is increased, configuring methods effectively becomes harder. One of the aim of this work is to get rid of as many parameter as possible by extracting them from image or SPs. Number of iterations is one of these parameters. As shown through experiments, number of iterations can be determined with respect to average SP area, and for different number of SPs or for different image resolutions there is no need to adjust a hyper-parameter.

Performance of SP extraction methods significantly depend on employed cost function. Therefore, cost functions of existing methods are investigated. Cost function is composed of spectral and spatial distances. A compactness hyper-parameter is utilized to prioritize spatial one to spectral. Higher values of this parameter results more compact but less accurate SPs.

It is observed that cost functions in literature are formed by making several assumptions, some of which causes a decrease in performance. Regarding spatial distance term, discarding spatial covariance by assuming orthogonal axes results in poor performance in term of following local image structure, as without spatial covarince all SPs are forced to have same shape and size. This problem is solved by the proposed cost function alternatives SLIC+ (4.11) and LASP+ (4.12) which employs spatial covariance for the normalization of spatial distance. Proposed alternatives effectively result in using Mahalanobis distance rather than Euclidian distance.

SLIC [2] use a static and global term for the normalization of spectral distance, which forces each SP to behave same within an image and across different images. On the other hand, LASP [17] utilize local SP variance for normalization; however, upper and lower bounds of variance is set by two hyper-parameters. Hence, these static and global definitions make cost functions incapable of adapting to different images and data sets. Moreover, using global definitions [2] prevent spectral term to adapt different textured regions within image. This drawback is got rid of by proposing alternative cost functions that extract those parameters from the image itself. With this approach, SP extraction performance aimed to be high and stable across different image and data sets and even for different textured regions within image. In this

manner, spectral term in cost function (4.9) proposed by SLIC [2], is normalized with average SP variance for each spectral band, resulting in a new cost function SLIC++ (4.13). Upper and lower bounds of SP variance are defined as hyper-parameters in LASP [17]. In this study, it is proposed to set these two relative to average SP variance with a single hyper-parameter, and the cost function LASP++ (4.14) is proposed. Finally, by performing spectral normalization of SP variance over spectral channels and spatial normalization with spatial covariance of SPs a Bayesian classifier for pixel to SP assignment is proposed which is called as Bayesian Superpixels (BSP).

To observe the performance of proposed alternatives, a set of experiments is conducted. It is observed that proposed cost functions outperform existing ones. Each proposed alternative cost function SLIC+, SLIC++, LASP+, LASP++ improves the preceding one. Bayesian Superpixels (BSP) cost function which only sticks to Gaussian distribution assumption extracts SPs with a low run-time as SLIC++ does and high accuracy as LASP++ can provide. Unlike LASP++, BSP works with SP level statistics during pixel label updates which increases run-time performance close to SLIC++ and keeping accuracy close to LASP++ which are top observed performances. It can be concluded that, employing spatial covariance, extracting variance definitions from image and using Bayesian prior statistics increases performance.

Besides improving existing SP methods or proposing alternatives, in order to evaluate the methods effectively a benchmark is defined. Benchmark consist of metrics used in the literature, BR, UE, ASA, run-time. Since each of BR, UE and ASA have drawbacks and should be evaluated together for an effective analysis, it is proposed to evaluate SP extraction accuracy with a single parameter, and BASA metric is proposed. BASA proposes a fair measurement method that removes this need by enabling measuring of accuracy using a single metric.

There are many options and hyper-parameters available to be used in SP extractions methods but the information in literature is insufficient for hyper-parameter values and comparison. Moreover, there is no information about configuration of methods in performance benchmarks. Therefore results highly depend on chosen parameters for individual experiments and seem to be unfair. In order to get the results independent of hyper-parameters, it is proposed to evaluate performance metrics with respect

to average SP compactness. Compactness is calculated directly from the generated SPs which makes it independent of hyper-parameter selection and provides an effective comparison between different methods. Besides, since compactness can be controllable, metrics based on compactness gives sufficient information to SPs user applications about the performance of method at desired compactness level.

Compactness is one of the major considerations throughout this work, it needs to be measured accurately. Nevertheless, due to discrete nature of image, measured compactness significantly depends on accuracy of perimeter measurement. Perimeter measurements become erroneous as SP area decreases. Therefore, alternatives for perimeter measurement are investigated and the most accurate one is determined by comprehensive experiments.

Methods shows different performance at different compactness levels. A method generating more accurate SPs at lower compactness levels than another, may generate less accurate at higher levels. Such a behaviour makes it difficult to compare performance of two methods and choose one another. Moreover, benchmarks in literature may configure methods such that a better performing method may fail to outperform the other as those methods may operate at different compactness levels. In order to get rid of this ambiguity in performance measurement, area under the curve approach is proposed. This approach gets the average of performance metrics within a range of compactness. With this approach, metrics become comparable independent of compactness. Such an approach also eliminates the dependency on hyper-parameters. In order to enhance this evaluation method, average results are taken along the region of interest.

Performance of the proposed methods are evaluated against top performing state-of-the-art methods. Throughout experiments, performance is evaluated with various accuracy metrics and area under curve approach. Even if it was hard to decide which method performs better by investigating performance curves with respect to compactness, area under curve approach reveals the best performing methods. With AUC approach, it is observed that proposed methods outperform the state-of-the-art methods in both accuracy and run-time.

To sum up, in this work clustering-based SP extraction methods are investigated in

detail. Their general algorithm flow is extracted and for each step performance of alternatives are evaluated and improvements are proposed whenever applicable. At the end, a family of SP extraction methods is proposed which outperforms state-of-the-art. Besides, improvements are proposed for performance metrics and evaluation benchmarks that enable comparison of methods effectively and in a fair way.

REFERENCES

[1] X. Ren and J. Malik, "Learning a classification model for segmentation," in *null*, p. 10, IEEE, 2003.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, *et al.*, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[3] C. Çığla and A. A. Alatan, "Efficient graph-based image segmentation via speeded-up turbo pixels," in *2010 IEEE International Conference on Image Processing*, pp. 3013–3016, IEEE, 2010.

[4] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: an evaluation of the state-of-the-art," *Computer Vision and Image Understanding*, vol. 166, pp. 1–27, 2018.

[5] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 583–598, 1991.

[6] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, May 2002.

[7] F. Drucker and J. MacCormick, "Fast superpixels for video analysis," in *Workshop on Motion and Video Computing*, pp. 1–8, 2009.

[8] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[9] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2097–2104, IEEE, 2011.

[10] J. Strassburg, R. Grzeszick, L. Rothacker, and G. A. Fink, "On the influence of superpixel methods for image parsing.," in *VISAPP (2)*, pp. 518–527, 2015.

[11] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2290–2297, 2009.

[12] P. Buyssens, I. Gardin, S. Ruan, and A. Elmoataz, "Eikonal-based region growing for efficient clustering," *Image and Vision Computing*, vol. 32, no. 12, pp. 1045–1054, 2014.

[13] L. C. Evans, *Partial Differential Equations, AMS Graduate Texts in Mathematics*. American Mathematical Society, 1997.

[14] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *European conference on computer vision*, pp. 13–26, Springer, 2012.

[15] J. Yao, M. Boben, S. Fidler, and R. Urtasun, "Real-time coarse-to-fine topologically preserving segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2947–2955, 2015.

[16] C. Conrad, M. Mertz, and R. Mester, "Contour-relaxed superpixels," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 280–293, Springer, 2013.

[17] K. G. İnce, C. Çığla, and A. A. Alatan, "Lasp: Local adaptive super-pixels," in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 4092–4096, IEEE, 2015.

[18] C. Çığla and A. A. Alatan, "Efficient graph-based image segmentation via speeded-up turbo pixels," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 3013–3016, IEEE, 2010.

[19] O. Freifeld, Y. Li, and J. W. Fisher, "A fast method for inferring high-quality simply-connected superpixels," in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 2184–2188, IEEE, 2015.

[20] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-adaptive superpixels," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 2087–2090, IEEE, 2012.

[21] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2027–2034, 2013.

[22] J. Wang and X. Wang, "Vcells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1241–1247, 2012.

[23] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[24] P. Neubert and P. Protzel, "Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 996–1001, IEEE, 2014.

[25] J. Chen, Z. Li, and B. Huang, "Linear spectral clustering superpixel," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3317–3330, 2017.

[26] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 530–549, 2004.

[27] A. P. Moore, S. J. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8, Citeseer, 2008.

[28] A. Schick, M. Fischer, and R. Stiefelhagen, "Measuring and evaluating the compactness of superpixels," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 930–934, IEEE, 2012.

[29] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve.," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[30] G. Bertrand, "Simple points, topological numbers and geodesic neighborhoods in cubic grids," *Pattern recognition letters*, vol. 15, no. 10, pp. 1003–1011, 1994.

[31] "The berkeley segmentation dataset and benchmark bsd300 image data set." `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`. Accessed: 2019-02-25.

[32] R. Giraud, V.-T. Ta, and N. Papadakis, "Evaluation framework of superpixel methods with a global regularity measure," *Journal of Electronic Imaging*, vol. 26, no. 6, p. 061603, 2017.

[33] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, no. 2, pp. 260–268, 1961.

[34] H. Freeman, "Computer processing of line-drawing images," *ACM Computing Surveys (CSUR)*, vol. 6, no. 1, pp. 57–97, 1974.

[35] "Measuring boundary length." `https://www.crisluengo.net/archives/310`. Accessed: 2019-02-15.

[36] D. Proffitt and D. Rosen, "Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges," *Computer Graphics and Image Processing*, vol. 10, no. 4, pp. 318 – 332, 1979.

[37] A. Vossepoel and A. Smeulders, "Vector code probability and metrication error in the representation of straight lines of finite length," *Computer Graphics and Image Processing*, vol. 20, no. 4, pp. 347 – 364, 1982.

APPENDIX A

PERIMETER MEASUREMENT

A.1 Accuracy Problem during Measurements

In theory, compactness of a continuous ideal circle, a square and a hexagon is 1, 0.785 and 0.907 respectively. However during the implementation, due to discrete nature of pixels, this values may deviate. Especially as the area of SP decreases, perimeter approximation method start to dominate the metric, i.e. errors in those calculations has more significant effect on results. If the perimeter of SP is not computed precisely, low compactness values could be assigned to convex shapes. For instance, Stutz [4] uses Schick's method [28] to measure a perimeter. In the released code of the work, perimeter is measured such that, for each boundary pixel the number of 4-connected neighbors pixels that differ in cluster label are encountered. This approach calculates 0.78 as compactness score for square which is close to theoretical one however scores 0.58 for hex and 0.62 for circle are far away of their theoretical ones.

Giraud [32] addresses this problem and suggests that all convex shapes are regular and their compactness should be same. The ones that is not regular shaped with respect to them get lower compactness values. The method calculates overlap of cluster with its convex hull and weights it with spatial distribution of pixels within cluster. As a result regularity measure of circle, square, hexagon becomes 1, 0.830 and 0.940, respectively.

A.2 Accuracy of Perimeter Measurement

In the upcoming experimental section of this work, the metrics are evaluated with respect to varying compactness values, since it provides comparison of SP extraction algorithms independent of their input parameters. Compactness measure is crucial in evaluating performance of algorithms, and significantly affected by accuracy of perimeter measurement. Therefore different perimeter measurement approaches in literature are investigated to select the most accurate one. Hence to select the best fit, perimeter measurement approaches listed in Table A.1 are examined in detail.

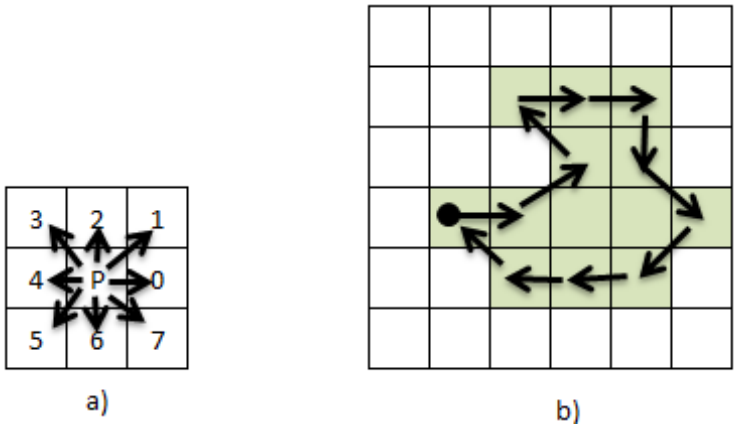


Figure A.1: Freeman code chain; a) directions and their assigned label, b) example of a chain code calculation

- **Freeman chain-code based methods:** Freeman [33, 34] introduces a chain-code formed by stepping along object boundary. While stepping, it assigns a code to each transition from one pixel to another consecutive pixel on the boundary according to transition direction. Figure A.1.a shows the directions and related codes. Even number are given to horizontal or vertical steps and odd numbers are given to diagonal ones. Figure A.1.b shows an example of a chain code by stepping through boundary. The chain code of this boundary is "0, 1, 3, 0, 0, 6, 7, 5, 4" starting from the pixel marked with dot. Following methods evaluates this codes differently to measure the boundary perimeter.

  Cris [35] investigated performance of Freeman chain code-based perimeter metrics which are as follows:

- **Freeman's metric (Fr):** Since diagonal steps (odd ones) are longer Freeman suggests to multiple odd ones by $\sqrt{2}$. As a result, metric adds 1 for each even, and $\sqrt{2}$ for each odd (diagonal transition). Metric formula is given as:

$$P = \sum_{i=0, C_i \epsilon 2a}^{8} 1 + \sqrt{2} \sum_{i=0, C_i \epsilon 2a+1}^{8} 1 \tag{A.1}$$

  where P is perimeter, C is Freeman chain code array calculated of that boundary and has size N, i is the index of array C, and a is a sequence of numbers $0, 1, 2, 3$.

- **Proffitt and Rosen's metric (PR):** Proffitt and Rosen's [36] proposes using weights for even ad odd codes and introduces corner count calculation. Metric applies weight correction to Fr metric. Measures 0.948 for each even, and 1.340 for each odd coded. Related formula is given as:

$$P = 0.948 \sum_{i=0, C_i \epsilon 2a}^{N} 1 + 1.340 \sum_{i=0, C_i \epsilon 2a+1}^{N} 1 \tag{A.2}$$

- **Vossepoel and Smeulders' metric (VS):** Vossepoel and Smeulders [37] adds corner correction to PR metric and uses optimal values for odds, evens and corner count such as measures 0.980 for even, and 1.406 for each odd coded and subtracts 0,091 for each corner. Metric formula is given as:

$$P = 0.980 \sum_{i=0, C_i \epsilon 2a}^{N} 1 + 1.406 \sum_{i=0, C_i \epsilon 2a+1}^{N} 1 - 0.091 \sum_{i=0, C_i \neq C_{i+1}}^{N} 1 \tag{A.3}$$

- **Pixel Count (PC):** PC is most commonly used metric, than just counts pixels without evaluating transitions. Hence, it measures 1 for both even and odd codes. Related formula is given as:

$$P = N \tag{A.4}$$

Apart from Freeman-based methods, two other alternatives are investigated:

Table A.1: Perimeter measurement method experiment configuration

Shape	**Perimeter**	**Metric Description**	**Metric**
RECT	80	Freeman's metric (Fr) (diagonal pixels counted as $\sqrt{2}$))	Perimeter
	800	Proffitt and Rosen's (PR) (Fr + weight correction, such that estimate is unbiased)	Compactness
HEX	72	Vossepoel and Smeulders' metric (VS) (PR + corner correction)	Root-mean-square (edge length range $[2 - 200]$)
	750	Pixel count (PC)	
		Erosion/dilation difference (ED)	
		Stutz [4] benchmark code (ST)	

- **Erosion/dilation difference (ED):** Erosion and dilation method, gets the difference of a region between dilated and eroded versions with a 3x3 structuring element composed of ones. Then boundaries are detected and the number of the boundary pixels are counted.
- **Stutz's metric (ST):** Stutz also propose an erosion/dilation based method [4] to measure perimeter of SPs. Among 4-connected neighbors, method measures 1 for each distinct labeled ones.

In order to to evaluate the accuracy of different perimeter measurement methods, perimeter and compactness of square and hexagon in different orientations are calculated. Accuracy is measured for typical values of SP area as well as larger values to understand the effect of discretization better. Table A.1 shows respective experiment configuration.

Figure A.2 shows perimeter measurement results of each approach for different ori-

entations of a square. Typical value of 20 px is chosen as the edge length of square which results in 400 px^2 area and 80 px perimeter. TR shows the true perimeter length. As shown in the figure, Fr, PR and VS are stable around the true perimeter for all orientations. Among them, Fr covers the ground-truth line better than PR and VS. On the other hand, methods ST, ED and PD produce unstable and erroneous values. For the methods ST and ED, perimeter increases through $45°$ orientation and then decreases, and vice versa for PD.

Figure A.3 shows the results for edge length is equal to 200 pixel of a square having corresponding 40000 $pixel^2$ area and 800 px perimeter. Similar to results above, Fr, PR and VS outperform the rest in perimeter measurement accuracy for large clusters. Among them, PR and VS perform better. The other 3 methods, ST, ED and PD, behave same as they do in analysis of edge length of 20 px. Their output deviates from the true perimeter length and error increases around $45°$.

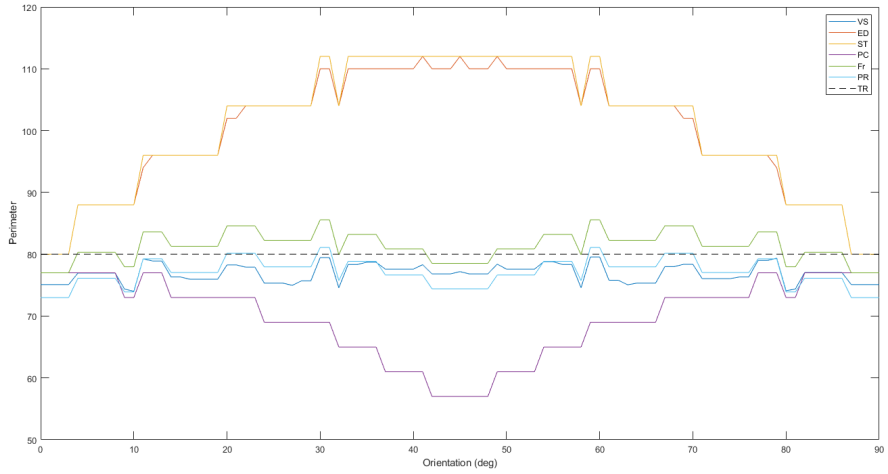


Figure A.2: Perimeter calculation results of various algorithms for a square having perimeter 80 px in different orientations

Figure A.4 and A.5 depict compactness result calculated for each method for edge length 20 px and 200 px respectively. TR shows the actual compactness value of the square as reference. Since compactness is inversely proportional to perimeter, as expected Fr, PR and VS shows better performance than ST, ED and PD.

Figure A.6 shows results of perimeter calculations of related approaches measured for

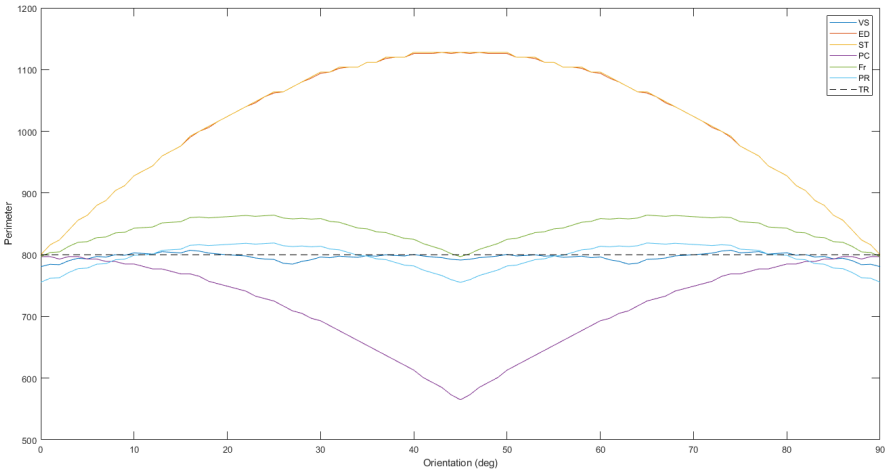


Figure A.3: Perimeter calculation results of various algorithms for a square having perimeter 800 px in different orientations

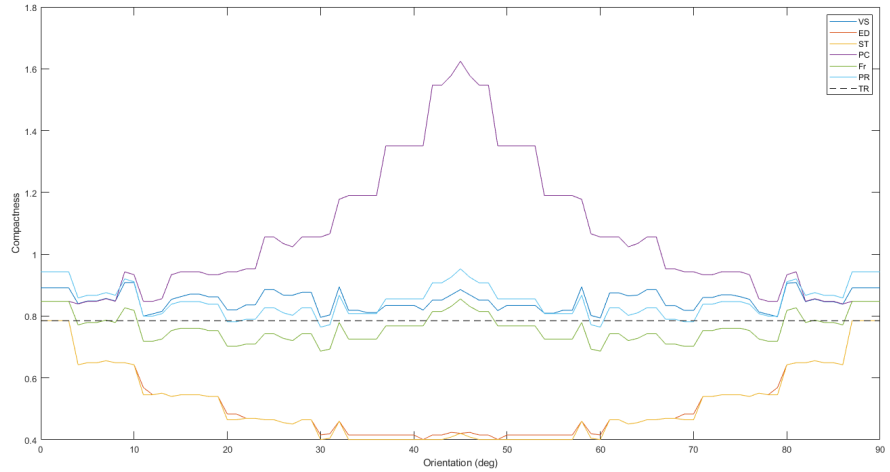


Figure A.4: Compactness calculation results of various algorithms for a square having perimeter 80 px in different orientations

a hexagon in different orientations. Typical value 12 px is chosen for edge length of hexagon having corresponding area about 374 px^2 and perimeter 72 px. TR shows actual perimeter value of square as reference with 72 px edge length. Similar to square calculations, Fr, PR and VS are stable around the true perimeter length for all orientations. Among them, Fr covers ground-truth line better PR and VS. On the other hand, results of methods ST, ED and PD are unstable and erroneous like

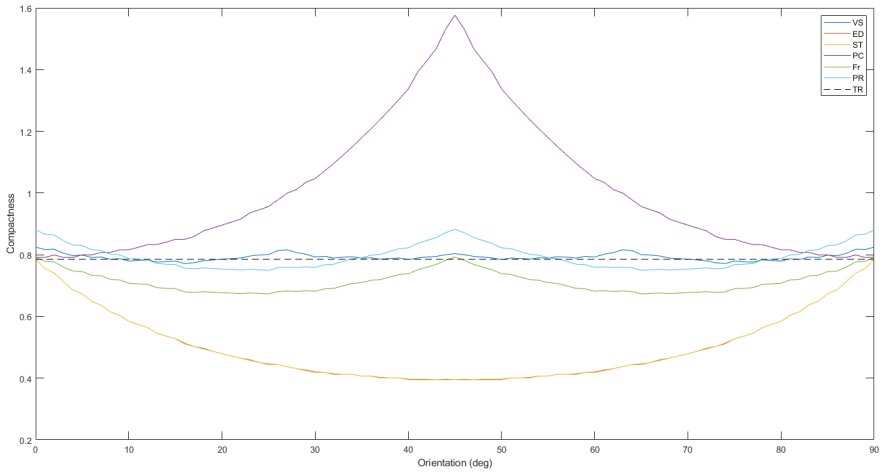


Figure A.5: Compactness calculation results of various algorithms for a square having perimeter 800 px in different orientations

in square case. Perimeter increases through $45°$ orientation and then decreases in ST and ED calculations, and vice versa in PD ones.

Figure A.7 shows the perimeter results of different methods for a hexagon with edge length equal to 125 pixel which is about 40594 $pixel^2$ in area of hexagon. Similar to small hexagon case, Fr, PR and VS outperforms the rest in large cluster measurements. Between them, PR and VS covers the ground-truth line better than Fr. The other 3 methods behave same as they do in analysis for edge length of 12 px. Their outputs are not erroneous and deviate for different orientations.

Figure A.8 and A.9 depict compactness calculated with perimeters obtained with different methods for a hexagon with edge length 12 px and 125 px respectively. TR shows the actual compactness value of the hexagon as reference. Since compactness is inversely proportional to perimeter, as expected Fr, PR and VS shows better performance than ST, ED and PD.

Above, for large and small cluster size behaviour of perimeter calculation methods in different orientations of square and hexagon are analyzed. In order to analyze their performance for different cluster sizes, root mean square error (RMS) with respect to true compactness value of each shape is depicted in Figure A.10 and A.11. Edge length is range from 5 to 200 for both shapes.

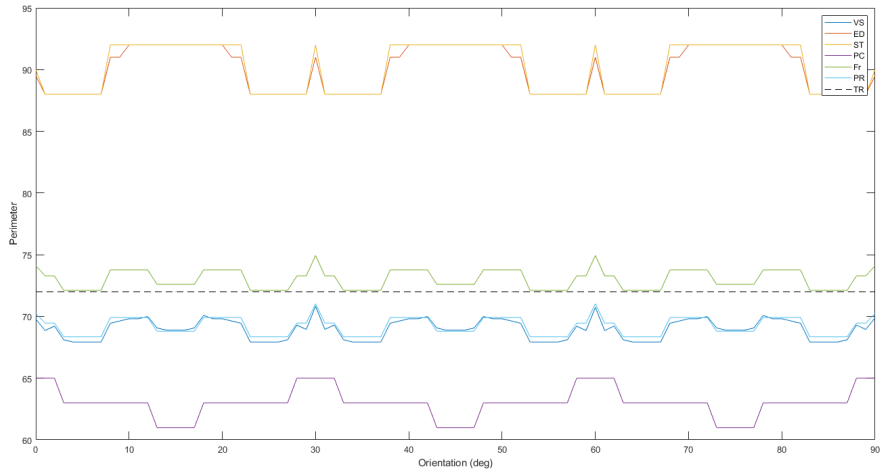


Figure A.6: Perimeter measurement results of various algorithms for a hexagon having perimeter 72 px in different orientations

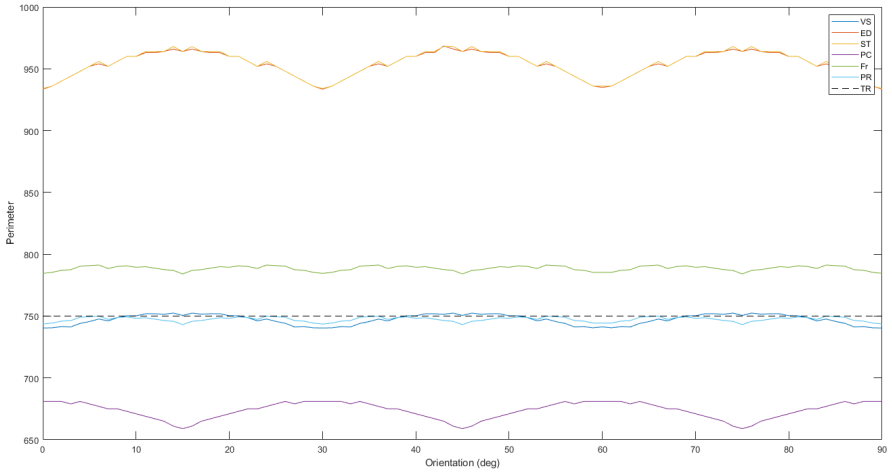


Figure A.7: Perimeter measurement results of various algorithms for a hexagon having perimeter 750 px in different orientations

Regarding square, as it is seen from the Figure A.10, for edge length less than 26 pixels Fr, for higher values VS has the minimum error for compactness. For hexagon, when edge length is less than 17 pixels Fr, for higher values PR has the minimum error as shown in Figure A.11. In both cases Fr perform better, corresponding area is approximately 700 px^2, which means Fr is better for clusters having size up to 700 px^2. For larger clusters, although VS and Pr has close performance for hexagon, VS

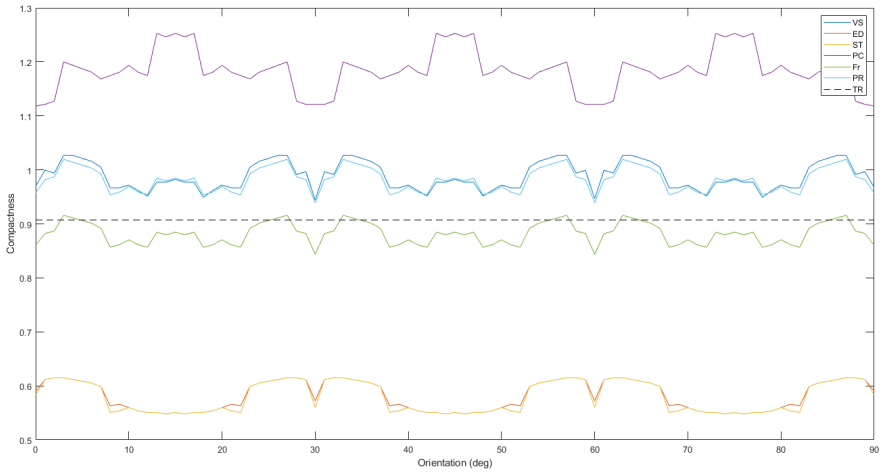


Figure A.8: Compactness calculation results of various algorithms for a hexagon having perimeter 72 px in different orientations

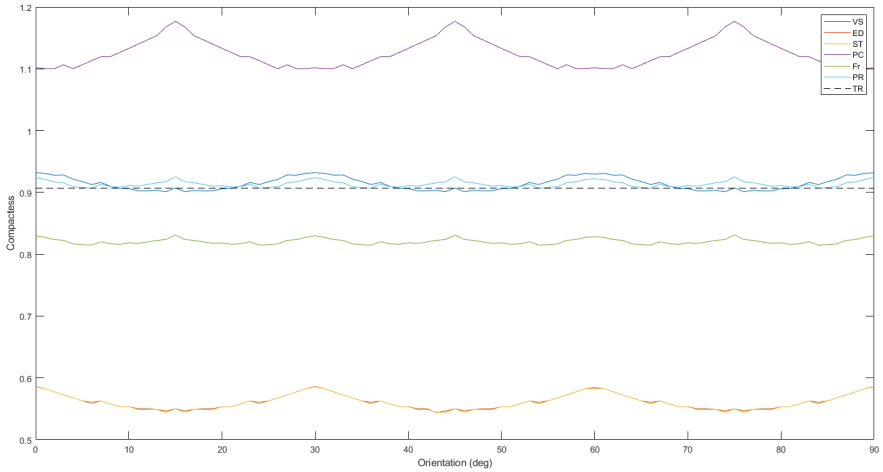


Figure A.9: Compactness calculation results of various algorithms for a hexagon having perimeter 750 px in different orientations

is significantly more precise than Pr in calculating square perimeter. Table A.2 and A.3 show RMS errors of approaches with respect to typical edge length of square and hexagon respectively. This values are also align with the inference above; Fr has the minimum error for shapes having typical edge length.

Table A.2: RMS error of measured compactness of square at different orientations by different perimeter metrics

Edge length	**VS**	**ED**	**ST**	**PC**	**Fr**	**PR**
20	0,07	0,29	0,29	0,35	0,05	0,08
15	0,11	0,28	0,28	0,38	0,05	0,11
10	0,16	0,27	0,29	0,41	0,08	0,15

Table A.3: RMS error of measured compactness of hexagon by different orientations for different perimeter metrics

Edge length	**VS**	**ED**	**ST**	**PC**	**Fr**	**PR**
12	0,08	0,33	0,33	0,28	0,03	0,08
9	0,10	0,33	0,33	0,28	0,03	0,09
6	0,19	0,30	0,30	0,37	0,07	0,18

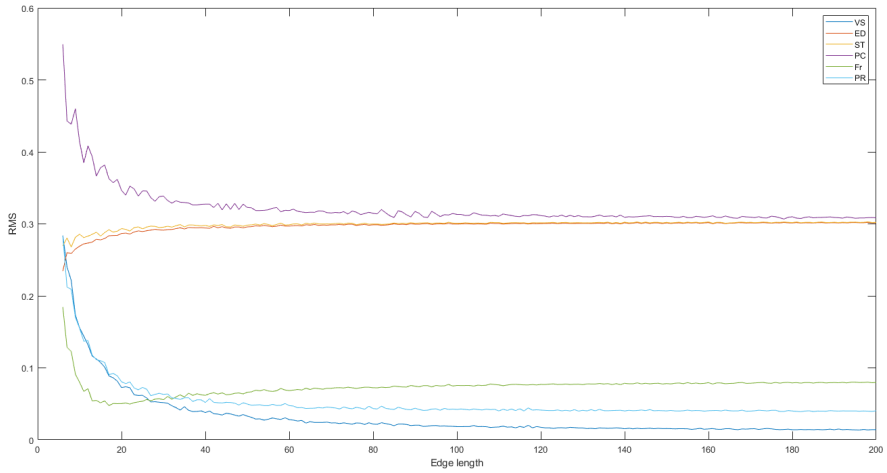


Figure A.10: Root mean square error of measured compactness for different perimeter metrics for squares of various edge lengths

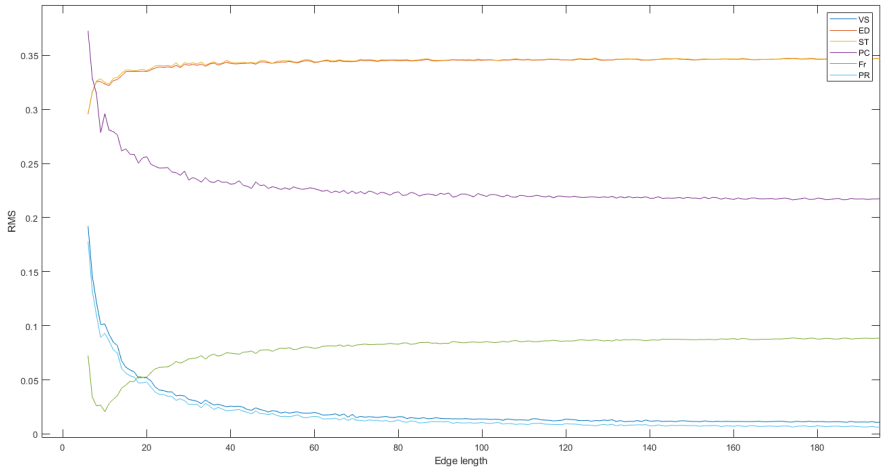


Figure A.11: Root mean square of measured compactness for different perimeter metrics for hexagons of various edge lengths

A.3 Conclusions on Perimeter Measurement

In the current work, images up to 150k pixels are processed, and the related typical area of an SP for 500, 1000 and 2000 SPs is 300, 150 and 75 px^2 respectively. Therefore, typical area is at most 300 px^2 when 500 SPs are generated. SPs can be extend during iterations by gaining pixels from neighbor SPs, but at the same time make

others reduce their volume keeping the average area constant. Considering typical area for 1000 SP over 150k image, Fr seems a better choice to measure the perimeter and hence the compactness as well. On the other hand, for higher resolutions than 700k and for at most 1000 SPs, VS gives more reliable results in terms of perimeter calculations.

To sum up, compactness is one of the major evaluation parameter of SP extraction methods. As compactness depends on SP perimeter, perimeter has to be measured accurately for a fair evaluation. Due to discrete nature of images, perimeter measurement of different methods may vary from actual one resulting misleading compactness evaluations. The error in measurements increases when SP size decreases as ratio of boundary pixels increases within SP. There are several perimeter measurement methods, and in order to determine the most accurate one, experiments are conducted in this section. Perimeter calculation methods are compared in terms of stability against orientation change and aligning with the theoretical values. Fr is chosen due to its stable behaviour with minimum deviation from actual value for both square and hexagon shapes at different orientation. Fr shows best performance for clusters having area up to 700 px^2 which is much higher than typical values used throughout this work.