Finding Objects of Interest in Images using Saliency and Superpixels

THÈSE N^{O} 4908 (2011)

PRÉSENTÉE LE 13 JANVIER 2011

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE COMMUNICATIONS AUDIOVISUELLES 2
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Radhakrishna ACHANTA

acceptée sur proposition du jury:

Prof. R. Hersch, président du jury
Prof. S. Süsstrunk, directrice de thèse
Prof. P. Fua, rapporteur
Prof. B. Macq, rapporteur
Prof. M. Salzmann, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2011

Contents

Abstract

The ability to automatically find objects of interest in images is useful in the areas of compression, indexing and retrieval, re-targeting, and so on. There are two classes of such algorithms - those that find any object of interest with no prior knowledge, independent of the task, and those that find specific objects of interest known a priori. The former class of algorithms tries to detect objects in images that stand-out, i.e are salient, by virtue of being different from the rest of the image and consequently capture our attention. The detection is generic in this case as there is no specific object we are trying to locate. The latter class of algorithms detects specific known objects of interest and often requires training using features extracted from known examples. In this thesis we address various aspects of finding objects of interest under the topics of saliency detection and object detection.

We present two saliency detection algorithms that rely on the principle of center-surround contrast. These two algorithms are shown to be superior to several state-of-the-art techniques in terms of precision and recall measures with respect to a ground truth. They output full-resolution saliency maps, are simpler to implement, and are computationally more efficient than most existing algorithms. We further establish the relevance of our saliency detection algorithms by using them for the known applications of object segmentation and image re-targeting. We first present three different techniques for salient object segmentation using our saliency maps that are based on clustering, graph-cuts, and geodesic distance based labeling. We then demonstrate the use of our saliency maps for a popular technique of content-aware image resizing and compare the result with that of existing methods. Our saliency maps prove to be a much more effective replacement for conventional gradient maps for providing automatic content-awareness.

Just as it is important to find regions of interest in images, it is also important to find interesting images within a large collection of images. We therefore extend the notion of saliency detection in images to image databases. We propose an algorithm for finding salient images in a database. Apart from finding such images we also present two novel techniques for creating visually appealing summaries in the form of collages and mosaics.

Finally, we address the problem of finding specific known objects of interest in images. Specifically, we deal with the feature extraction step that is a pre-requisite for any technique in this domain. In this context, we first present a superpixel segmentation algorithm that outperforms previous algorithms in terms quantitative measures of under-segmentation error and boundary recall. Our superpixel segmentation algorithm also offers several other advantages over existing algorithms like compactness, uniform size, control on the number of superpixels, and computational efficiency. We prove the effectiveness of our superpixels by deploying them in existing algorithms, specifically, an object class detection technique and a graph based algorithm, and improving their performance. We also present the result of using our superpixels in a technique for detecting mitochondria in noisy medical images.

Keywords: saliency, segmentation, object detection, superpixels, medical image segmentation.

Résumé

La capacité à trouver automatiquement les objets intéressants dans une image est très utile dans les domaines de la compression, de l'indexation et de la recherche, du redimensionnement etc.. Il existe deux types d'algorithmes, ceux qui trouvent les objets d'intérêt sans connaissances préalables, indépendamment de la tâche et ceux qui trouvent des objets d'intérêt spécifiques déjà connus à priori. La première catégorie d'algorithmes essaie de détecter les objets qui ressortent de l'image, c'est-à-dire qu'ils sont saillants, car différents du reste de l'image et donc attirent notre attention. La détection dans ce cas est générique car il n'y a pas d'objet spécifique que nous essayons de localiser. La deuxième catégorie d'algorithmes détecte des objets d'intérêts connus spécifiques et nécessite souvent un entraînement utilisant les caractéristiques extraites d'exemples connus. Dans cette thèse nous abordons différents aspects de recherche des objets d'intérêts avec comme sujets la détection de saillance et la détection d'objet.

Nous présentons deux algorithmes de détection d'objets saillants qui reposent sur le principe de contraste centre périphérie. Nous montrons que ces deux algorithmes sont supérieurs à de nombreux algorithmes de pointe en terme de précision et de rappel mesurés par rapport à une référence. Ils donnent des cartes de saillance en pleine résolution, sont plus simples à implémenter et sont plus efficaces à calculer que la plupart des algorithmes existants. Nous établissons ensuite la pertinence de notre algorithme de détection de saillance pour des applications connues. Nous montrons l'usage de notre algorithme de détection de saillance en segmentant des objets saillants. Pour cela nous présentons trois techniques différentes basées sur les groupements, les graph-cuts et les distances géodésiques. Nous utilisons aussi nos cartes de saillance pour l'application populaire de redimensionnement d'images en fonction du contenu et comparons le résultat avec ceux des méthodes existantes. La carte de saillance d'une image est l'alternative la plus efficace par rapport à la carte de gradients conventionnelle pour la reconnaissance automatique de contenu.

Il est tout aussi important de trouver les régions d'intérêt dans des images que de trouver les images intéressantes a l'intérieur d'une large collection d'entre elles. Nous étendons à cet effet la notion de détection de saillance indépendamment de la tâche d'une image aux bases de données d'images. En plus de trouver ces images nous présenterons aussi deux nouvelles techniques pour créer des résumés d'images visuellement attractifs sous la forme de collages et de mosaïques.

Pour finir, nous abordons le problème de la détection d'objets d'intérêt spécifiques et connus. Plus précisément, nous traitons de l'étape d'extraction de caractéristiques qui est un pré-requis pour n'importe quelle technique dans ce domaine. Dans ce contexte, nous présentons d'abord un algorithme de segmentation par superpixels qui dépasse les performances de tous les algorithmes précédents en terme de mesures quantitatives des erreurs de sous-segmentation et de rappel des bords. Notre algorithme de segmentation par superpixels offre aussi plusieurs avantages sur les algorithmes existants comme la compacité, la taille uniforme, le contrôle sur le nombre des superpixels et l'efficacité de calcul. Nous prouvons l'efficacité de nos superpixels en les déployant dans des algorithmes existants de détection de classes d'objets et d'algorithmes basés sur la théorie des graphs tout en augmentant leurs performances. Nous présentons aussi une technique utilisant nos superpixels pour détecter les mitochondries dans des images médicales bruitées.

Mots clés: saillance, segmentation, détection d'objets, superpixels, segmentation d'images médicales.

Acknowledgments

I have had a wonderful time at EPFL and in Switzerland. I am indebted to several people who contributed directly or indirectly during the last few years on my way to realizing this thesis. I take this opportunity to thank as many of them as I can.

I would like to thank my supervisor Prof. Sabine Süsstrunk for providing me the opportunity to work in EPFL and for her guidance and support. In September 2005 I had a meeting with Sabine that perhaps lasted ten minutes and at the end of which she accepted me into her lab. I later realized I was quite lucky to be selected so easily. She gave me the freedom to choose the projects that interested me and made sure I delivered on time. Working with her has helped me significantly improve the quality of my work.

It was an honor to have as thesis committee members Prof. Roger Hersch, Prof. Pascal Fua, Prof. Benoit Macq, and Prof. Matthew Salzmann. I am grateful to them for taking time off their busy schedules to evaluate my work and for providing me with useful feedback.

I would like to thank the students I worked with on various projects: Patrick Zbinden, Patricia Wils, Olivier Küng, Patrick Shoenmann, Patrick Hammer, and Andrea Guidi. They gave me the opportunity to learn new things and experiment with things I did not have time for.

My thanks also go to Jocelyne Plantefol and Jacqueline Aeberhard for their less noticed but crucial role in making our labs run like well-oiled machines. Thanks also to Simon Hiscox for taking care of the computer infrastructure and accompanying me for a tea almost every morning of the last couple of years.

I would like to thank some extremely smart people I collaborated with and co-authored papers with. I thank Paco who has been both a jolly friend and an advisor. I thank Prof. Sheila Hemami whom I had the opportunity to collaborate with. During her short stint of three months in our lab I learnt and lot and produced some interesting results that are part of my thesis now. I also collaborated with Appu Shaji who is one of the most knowledgeable and helpful people I have come across. Every idea flies with him. I am thankful to him for being a friend and an advisor. Soon after joined EPFL I also got the opportunity to collaborate with Pascal Fua, Aurelien Lucchi, and Kevin Smith, all of whom I am very thankful to.

A special thanks Stefan Winkler whom I met in Singapore and who put me in touch with Sabine, which then unfolded the rest of my life as PhD student. Thanks also to Thomas Lochmatter and his parents whom I stayed with when I came to visit Sabine the first time. I met Thomas in Singapore and it was he who first gave me the idea of doing a PhD at EPFL. It was Jacques-Albert and his parents who took care of me when I came to start my Phd program. He also helped me find a temporary place to stay with two charming ladies Madeleine and Marianne, who call themselves my 'mammans suisses', and later some friends like Sverre and Isabelle who me feel at home as soon as my life in Switzerland started.

I am grateful to my friends and colleagues who enriched my life during the PhD program. A few I can mention here are Francisco Pinto, Luciano Sbaiz, Christof Faller, Patrick Vandewalle, Laurence Meylan, Daniel Tamburrino, Joanna Marguier, Ali Hormati, Ivana Jovanovic, Olivier Roy, Engin Tola, Mustafa Özuysal, Andrea Fosatti, German Gonzales, Vincent Lepetit, Michael Colander, Masoud Alipour, Cristian Carmeli, Francois Roy, and Jim Pugh. Among my friends, Roberto Costantini has been a mainstay support during my thesis. He voiced his opinion when I most needed it, and has always been there when I needed him. I thank him from the bottom of my heart.

Celine, my wife, the best thing that ever happened to me, came into my life when I was still a PhD student, occasionally lacking confidence, morbidly preoccupied with my research, living in a 27 square meter apartment. Her unconditional love and unflinching confidence in me made me a stronger person and added so much richness to my life in ways I could not imagine. Magically, my first paper got accepted after I met her, and my research just got better and better. It is only after I met her, I looked up suddenly and realized that my life is much bigger than my thesis. I will be thanking her and my lucky stars for the rest of my life. Thanks to her I have a whole new family in Europe. Her parents, siblings, grandparents, and relatives have all been very supportive.

My sister Lavanya, my brother Prakasam, his wife Neha, have been a constant source of support and love to me in every walk of my life. They often look up to me and make me think better of myself when I am in doubt. It is always my brother that I turn to whenever I need advice in my life. Though younger to me, it never ceases to amaze me how he quickly he grasps what I have to say and how he tells me exactly what I need to hear. I can never sufficiently express my gratitude for them.

The ones I will be forever indebted to are my parents. They are the very reason I am here doing whatever I am. They have taken pride in everything I have ever done, provided me with help, support, confidence, and love. I would strive to pass on what they gave me to our children, beginning with our daughter Sanjana. I dedicate this thesis to my parents.

Chapter 1

Introduction

Modern day life is rife with digital image and video capturing devices. The low cost of these devices and their compactness have led to their use in almost any sufficiently sophisticated electronic gadget. According to a comprehensive study by the Consumer Electronics Association [31], 77% of American households now own at least one digital camera. Most digital camera owners snapped an average of 72 digital photographs at the most recent event they attended and shared 51% of them. For sharing the pictures, 55% use email, 48% use personal computers, and 40% rely on sites such as MySpace, Facebook, Photobucket, and Snapfish. Cameras have also become the default add-on feature for mobile phones and their resolution is steadily rising. Consumer cameras are but one source of image data. There is also plenty of image data generated by security and surveillance cameras, geo-stationery satellites, reconnaissance aircrafts, medical imagery devices, infrared and ultraviolet cameras.

This growth in image data has led to new challenges in compression, transmission, storage, browsing, retrieval, organizing, and image data mining. Apart from this, we also need tools and techniques to deal with problems arising from digital image data and the technology associated with it. Trademark and copyright violation of images needs to be checked on the Internet. Email often contains image based messages that are used to circumvent text-based spam filters and need efficient blocking. Internet users desire tools to help them find the images they want, or conversely, avoid offensive images and advertising. Image and video archives containing years of news media recording that is languishing on magnetic tapes need to be digitized and indexed for preservation and ease of future use.

Compression and transmission needs are usually being met by current standards and technologies, thanks to lowering cost of memory devices and increasing network bandwidths. But effective storage, intelligent retrieval, and extracting information about image content are problems that do not have reliable solutions yet. This is because addressing these tasks requires a human-like comprehension of such data. This is proving quite difficult and the technology that can provide such capabilities is lagging far behind what is desired. The work of Enser [41, 42, 12] reveals the yawning gap between user needs and the capabilities of technology. To cite an example, a real world query to look for an image may look like: “Pretty girl doing something active, sporty in a summery setting, beach - not wearing lycra, exercise clothes - more relaxed in tee-shirt. Feature is about deodorant so girl should look active - not sweaty but happy, healthy, carefree - nothing too posed or set up - nice and natural looking”. A human being might retrieve Fig. 1.1[1] from some photograpic archives in response to such query, but it is far beyond the prowess of any technology to automatically extract information from images well enough to associate such a query with one.

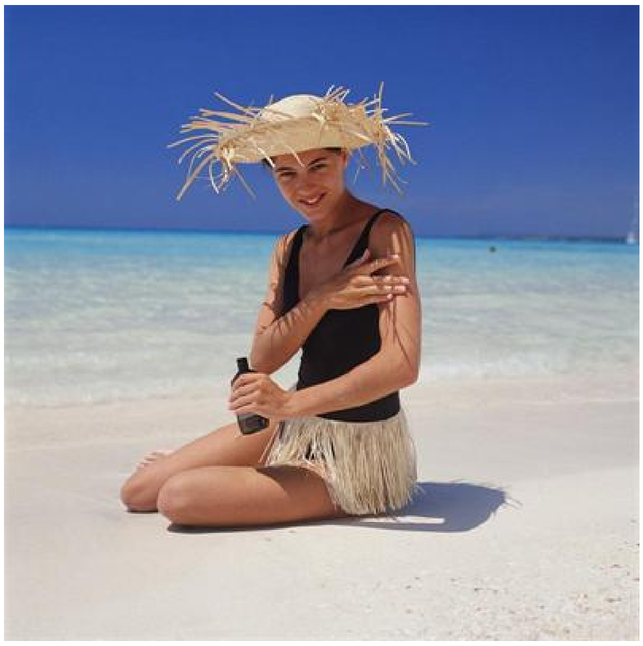


Figure 1.1: *An image a human being may retrieve from stock footage in response to a complex query that is beyond a machine's capability to understand.*

Humans have an unsurpassed capacity to locate objects in an image, recognize them instantly, understand the context, match the image with another one, categorize the image, give it titles and tags, have an emotional response to an image, like or dislike it, and so on. What we need is for machines to be able to understand, like our vision allows us to. Computer vision is the science and technology of making machines understand visual information. It deals with the theory and practice behind artificial systems that extract information from images that is meaningful to us.

The intriguing aspect of our ability to see and comprehend the images of the world is that we have limited understanding of how we do so. It is the continuing

[1]Unless explicitly stated, all images in this thesis are from two publicly available databases: the Berkeley Segmentation Database [132] and MSRA Salient Object Database [103]

endeavor of computer vision algorithms to match, at least in part if not entirely, the incredible ability of humans to perceive images. These algorithms focus on problems such as object recognition, motion analysis, scene reconstruction, image restoration, and so on, all with the aim of describing the content of images as well as unraveling the workings of the human visual system. Looking at the situation, where on one hand there is a relentless deluge of digital visual content and on the other hand computer vision algorithms are far from adequate to meet our needs, it seems there has never been a better time to be a computer vision scientist!

1.1 Objects of interest

Quoting Marr [101], "vision is the process of discovering from images what is present in the world, and where it is." One specific problem of computer vision algorithms in extracting information from images is to find objects of interest. Animate or inanimate objects present in our environment capture our attention for various reasons like, being new or unusual, being or having something we want, and causing us to fight or flee. Fig. 1.2 shows some examples of objects of interest.

Humans have an uncanny knack of spotting objects of interest almost instantaneously in a scene. Such objects capture our attention whether we are aimlessly gazing at scene or whether we are searching for something specific. Sometimes, objects in the visual scene 'pop-out' because of their distinctiveness with respect to the rest of the environment in terms of their shape, symmetry, color, brightness, etc.. Such objects are noticed involuntarily by virtue of their relative contrast. This perceptual quality of an object, person, or pixel, which makes it stand out relative to its neighbors and thus direct our attention is called *saliency*. On the other hand, there are specific objects in a scene that we might be looking for. This could be a key we have dropped in a field, or a friend in a crowd. Such objects grab our attention because we are specifically searching for them. They are noticed by us even when there are other distracting visual inputs in the vicinity.

Human visual attention results both from bottom-up visual saliency of the retinal input that is task-independent and fast, as well as from slower, top-down, memory and volition-based processing that is task-specific [108]. Treisman and Gelade [128] demonstrated that visual attention in still images is comprised of a pre-attentive step, which is fast and unconscious, and an attentive, conscious, saccade-based image analysis, which is slower. The faster part of the attention grabbing mechanism responds to saliency while the slower one responds to finding a match with a known object in memory.

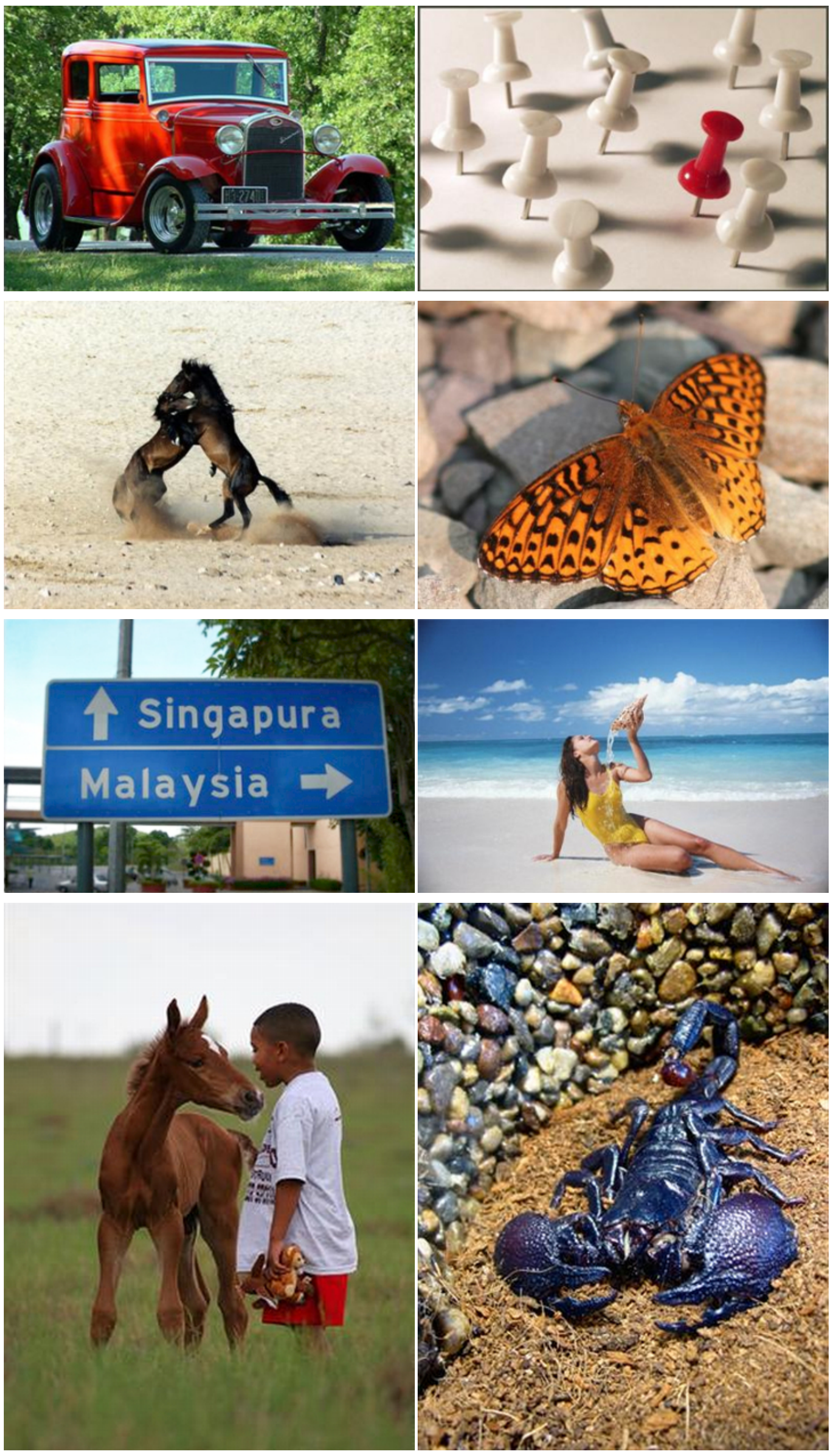


Figure 1.2: *Examples of objects that grab our attention in the image because we like them, hate them, fear them, need them, or just find them odd and interesting.*

1.2 Applications

Automatically finding objects of interest has direct benefits in numerous applications, especially those that require *content awareness*. Video compression is one such area. In scenarios like web-chats, network bandwidth savings can be achieved by adaptively assigning more bits to faces, which are more important as opposed to background. Domestic robots and autonomous vehicles can identify objects in their environment for navigation and collision avoidance. Another area is that of automatic image re-targeting, which is the task of changing aspect ratios of images to suit different display devices without distorting the important content of the image. Surveillance and security applications can benefit from automatic intruder detection. Automatic number plate detection in vehicle image databases can safeguard the privacy of owners or help automated parking systems. Medicine and biology can vastly benefit from automatic detection of tumors or cellular structures. Image and video database and indexing systems can profit immensely from knowledge of image content. Evidently, almost all aspects of our modern lives that involve digital images can take advantage of automatically finding objects of interest.

1.3 Goals of this thesis

The focus of this thesis is on one of our visual capabilities - finding objects of interest in images. Some examples of finding objects of interest are presented in Fig. 1.3. There are two main facets of this problem - finding an unknown salient object and finding a specific known object. In this thesis, we name the first problem of finding an *unknown* object of interest *saliency detection* and the second one of detecting a *known* object of interest task-specific object detection or simply *object detection*. For the former, we develop saliency detection algorithms. For the latter, which is a vast topic of research on its own, we focus on the step of feature extraction and show how it may be used for object detection.

1.4 Overview of the thesis

We first address the problem of saliency detection. In literature, saliency detection is done using a wide range of techniques and most of them suffer from several drawbacks. We take a look at several state-of-the-art algorithms from a frequency domain perspective to get an insight into their workings and shortcomings. We then devise an algorithm, which despite being simple, outperforms the state-of-the-art algorithms in terms of precision and recall as well as computational complexity. The algorithm overcomes many of the drawbacks of existing algorithms. It relies on the hypothesis that when the scale of an object in an image is unknown, as is usually the case, it is best to assume the largest scale. We then present an improved algorithm that takes a cue from the image borders to guess the scale. This leads to better

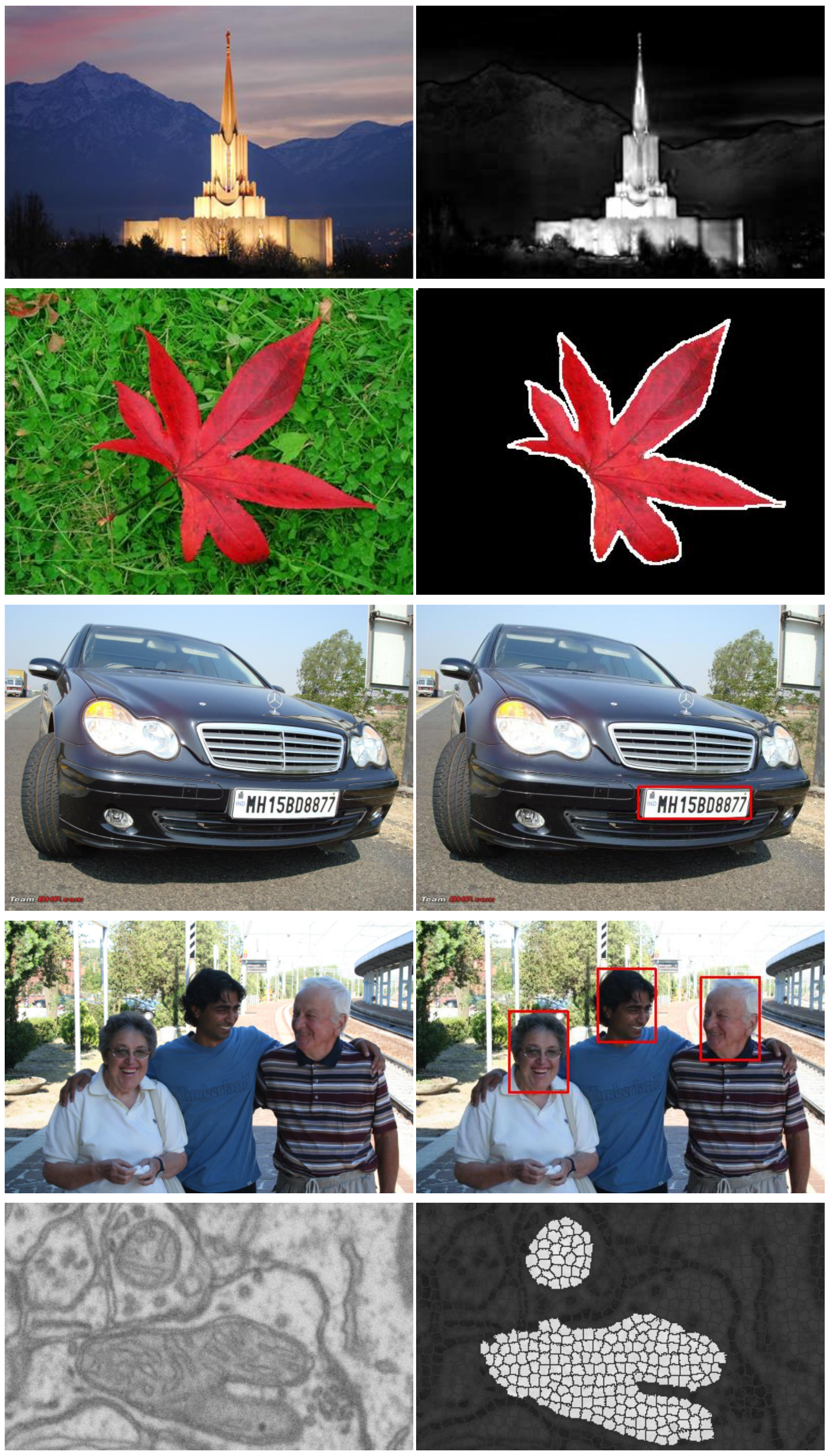


Figure 1.3: *Examples of automatic detection of objects of interest. The images on the left contain object(s) of interest. The ones on the right show the corresponding detection using computer vision algorithms.*

saliency detection as shown by quantitative comparison. It additionally overcomes the drawback of the previous algorithm of incorrectly detecting the background as salient in certain images where the object is too large or the background is complex. We present applications of these two algorithms in salient object segmentation and image re-targeting.

We then extend the concept of task-independent saliency detection to image databases. We present an algorithm for finding salient images from a database. Such an algorithm is useful in applications that summarize the database. We present two novel ways of summarizing image databases in the form of image mosaics and collages.

We then address the problem of detecting specific objects of interest. In this connection we present a superpixel segmentation algorithm that is used as the basis for extracting features. The superpixel algorithm is extensively compared with state-of-the-art algorithms. Our algorithm is shown to be superior to the existing algorithms according to the measures of under-segmentation error and boundary recall with respect to a publicly available ground truth. Our superpixel algorithm is also computationally the most efficient and offers much better control over the number of superpixels and their compactness using fewer parameters. Applications of our superpixels are shown in improving a couple of existing algorithms and in detecting mitochondria in noisy medical images.

A visual overview of the thesis is provided in Fig. 1.4. In the sub-sections that follow, we provide an overview of the content of the thesis and its contributions with the aim of aiding the reader to understand the flow of the dissertation. Each chapter is briefly introduced in a sub-section, some example images are presented, and the contributions associated with the chapter are listed.

1.4.1 Chapter 2: State-of-the-art

This chapter presents a review of existing literature on task-independent *saliency detection* and *superpixel segmentation*. For the former, a broad classification is provided for the techniques in existing literature. This is followed by listing their main drawbacks. Then we take a frequency domain point of view and analyze some of these techniques. This provides an insight into the workings of these techniques and how to address the requirements of a saliency map. For the latter, we present the literature review relevant to superpixel segmentation. We classify the state-of-the-art techniques into two main categories - graph based and gradient based. Following this, we present a review of the clustering techniques that are used in later chapters of this thesis.

1.4.2 Chapter 3: Saliency detection

In this chapter we present two saliency detection algorithms. The frequency-domain analysis of the previous chapter permits us to identify the cause of some of the

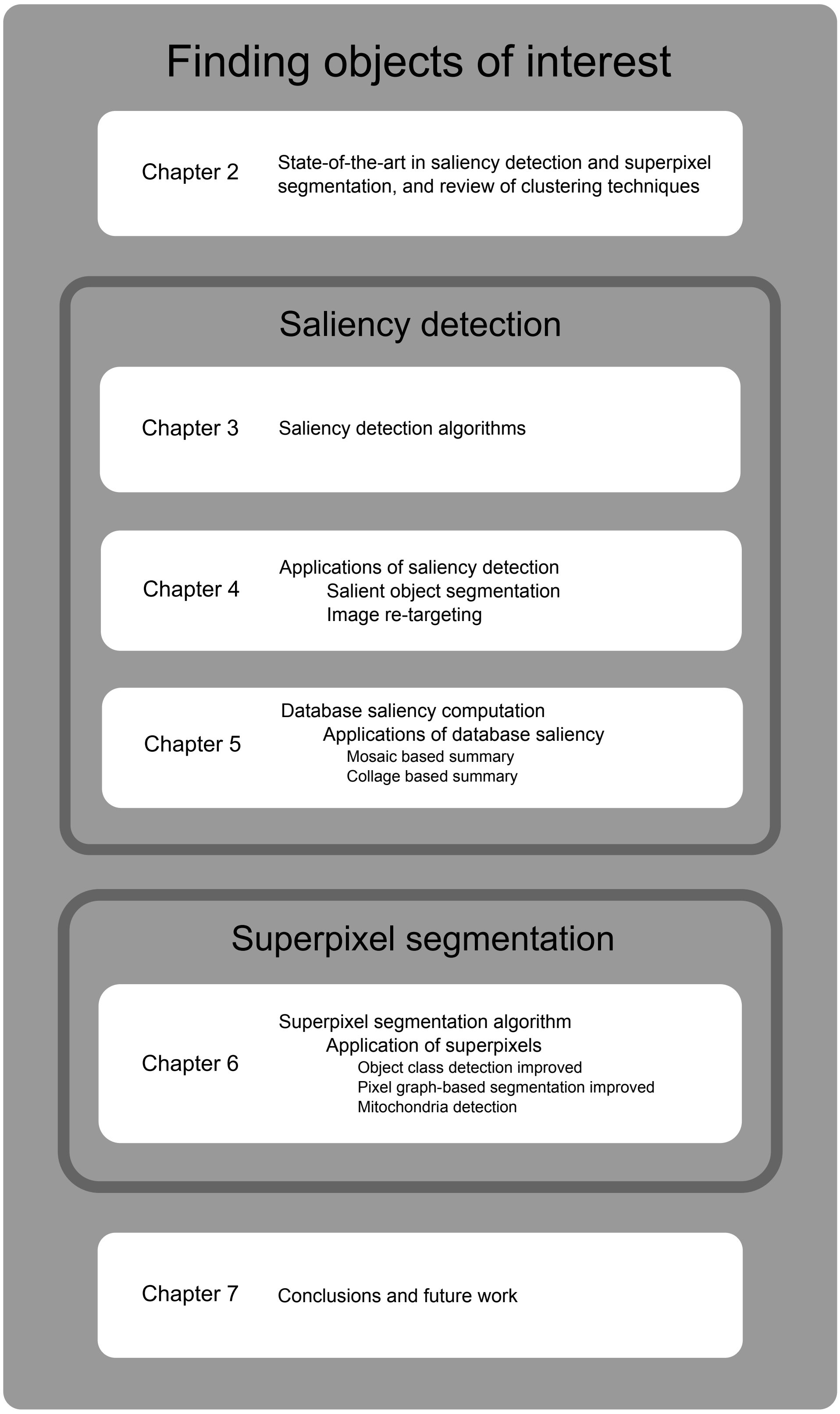

Finding objects of interest
Chapter 2
State-of-the-art in saliency detection and superpixel segmentation, and review of clustering techniques
Saliency detection
Chapter 3
Saliency detection algorithms
Chapter 4
Applications of saliency detection
Salient object segmentation
Image re-targeting
Chapter 5
Database saliency computation
Applications of database saliency
Mosaic based summary
Collage based summary
Superpixel segmentation
Chapter 6
Superpixel segmentation algorithm
Application of superpixels
Object class detection improved
Pixel graph-based segmentation improved
Mitochondria detection
Chapter 7
Conclusions and future work

Figure 1.4: *A visual overview of the contents and structure of thesis.*

drawbacks of existing saliency detection algorithms. Based on what we learn, we develop two saliency detection algorithms that overcome these drawbacks.

The main assumption of the first of these algorithms is that in the absence of a priori knowledge of the salient object's scale, the largest scale should be assumed. The second algorithm challenges this premise and varies the scale assumption with respect to the position of the pixel relative to the image borders.

For both saliency detection algorithms we perform a quantitative comparison with existing methods in literature. Our saliency maps demonstrate better precision and recall as compared to existing methods with respect to ground truth data. They are also computationally more efficient than most methods. Fig. 1.5 provides a visual comparison of our method as compared to the state-of-the-art.

Contributions

- Two saliency detection algorithms, which are quantitatively shown to significantly outperform several existing algorithms.
- Publicly available code (C and MATLAB) as well as ground truth database of 1000 images for comparison.

1.4.3 Chapter 4: Applications of Saliency detection

In this chapter we present some of the applications of saliency detection. We first present the use of saliency maps in segmenting salient objects. We present three techniques for this purpose, one using clustering and adaptive thresholding, another using graph-cuts, and a third using geodesic distances. Fig. 1.6 shows examples of objects segmented automatically using their saliency maps. In all three cases we compare segmentation results obtained using our saliency maps with those obtained using state-of-the-art saliency detection techniques. This further provides evidence of the effectiveness of our saliency detection techniques in object segmentation.

We then present the application of our saliency maps in the task of image re-targeting (see Fig. 1.6). Specifically, we focus on the re-targeting technique of *seam carving*, where we replace conventional gradient based energy maps with our saliency maps. The re-targeting results are qualitatively compared with other image re-targeting techniques to prove the effectiveness of our approach.

Contributions

- Three algorithms for segmenting salient objects using saliency maps.
- Improved seam carving using our saliency maps.

1.4.4 Chapter 5: Database saliency

In this chapter the idea of image saliency is extended to image databases. We claim that just as certain regions of the image are more salient than others, in databases,

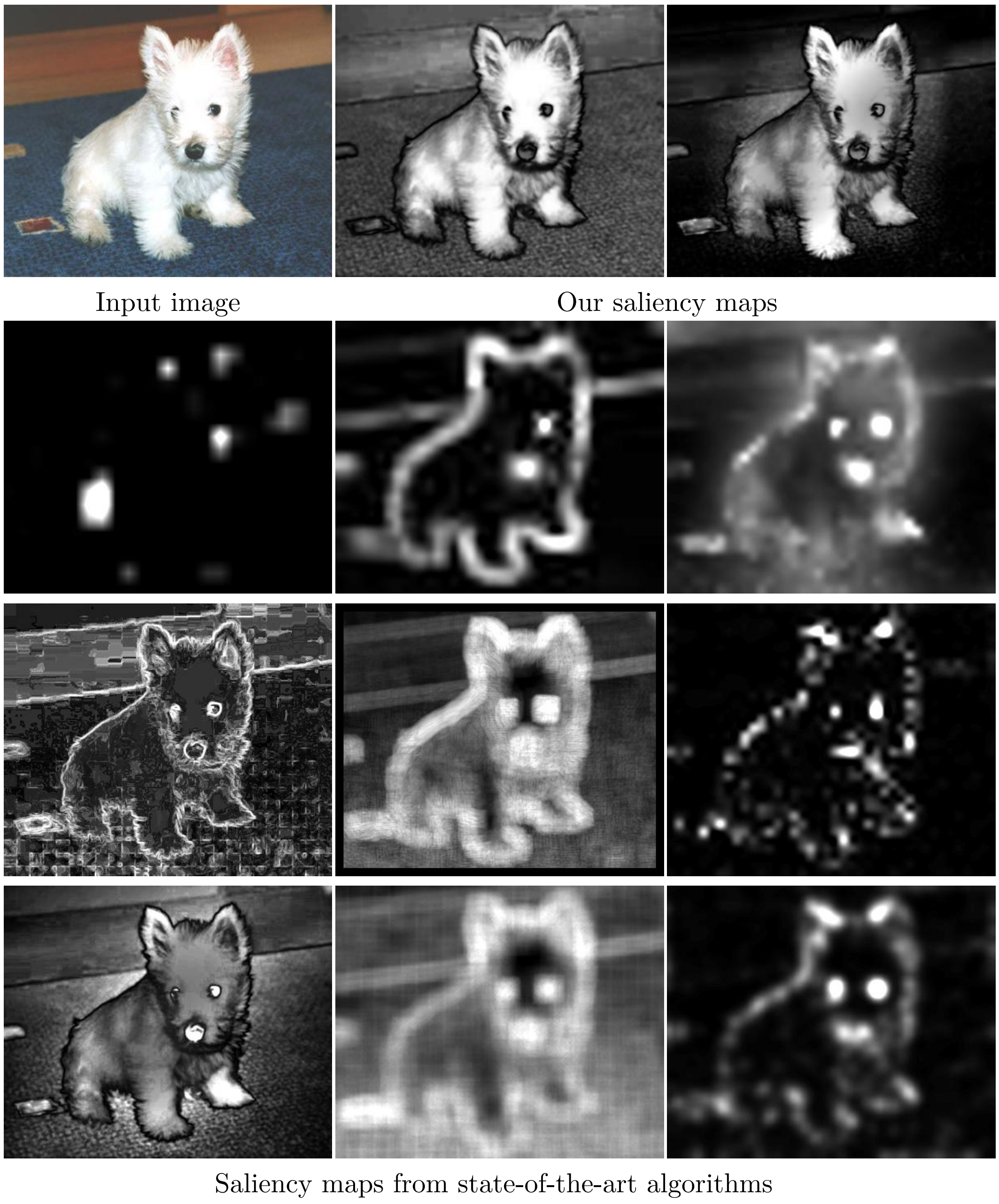
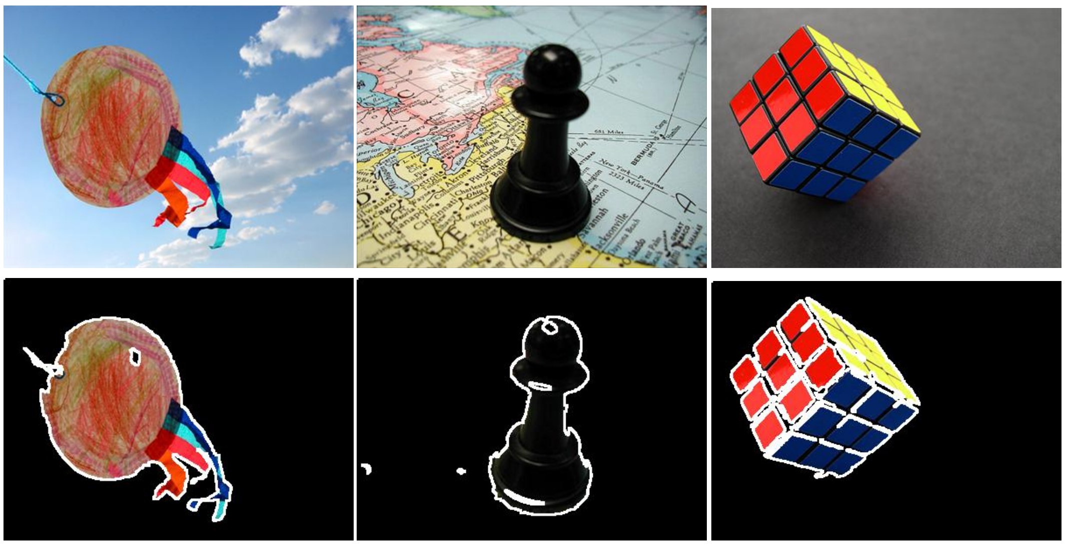


Figure 1.5: *Visual comparison of saliency maps of several state-of-the-art saliency detection techniques. Our methods produce saliency maps that have well-defined borders, highlight whole object regions, and suppress the background better than most methods even in the presence of complex backgrounds or when the salient object is very large.*



Top row shows input images. Bottom row shows objects segmented automatically using their saliency maps.

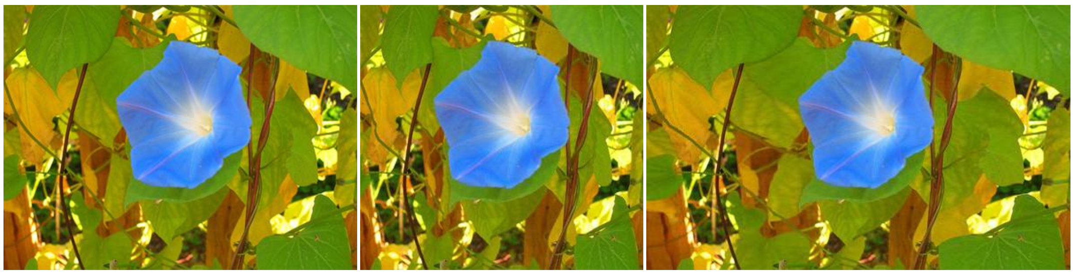
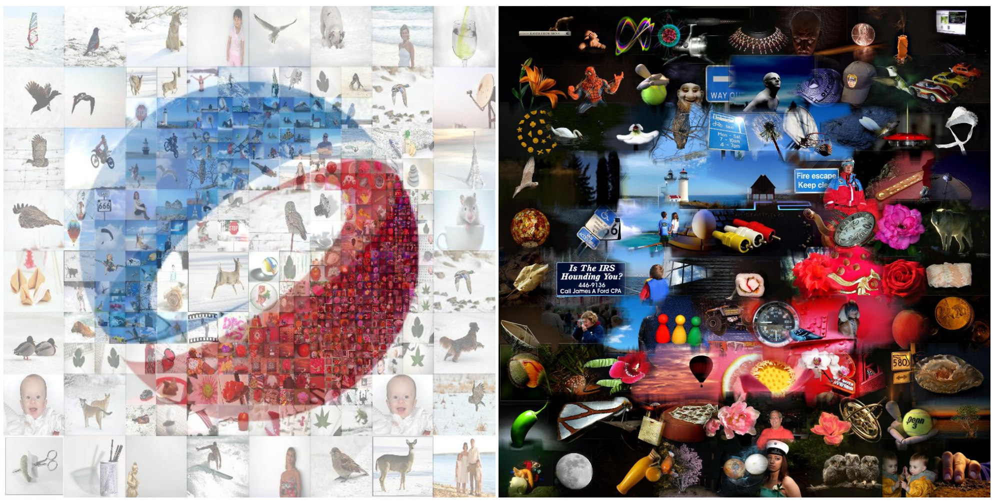


Figure 1.6: *Applications of saliency maps. The first set of images above show how segmentation can be done automatically. The second set of images below show how the aspect ratio of an image can be altered without changing the important content of the image.*

certain images are more conspicuous than others. To find them, we group the images into a desired (as required by the application) number of clusters and pick those images from each cluster that are the farthest from all the cluster centers. For clustering images we introduce a new image saliency based feature. Having found salient images of the database we use them to create summaries in the form of image mosaics and collages. An example of each of these summaries is shown in Fig. 1.7.



(a) (b)

Figure 1.7: *Finding salient images in a database and creating summaries. (a) A summary in the form of a mosaic. (c) A collage based image summary.*

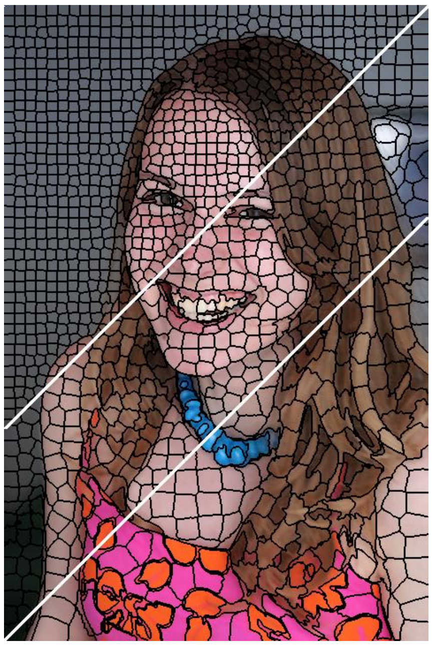
Contributions:

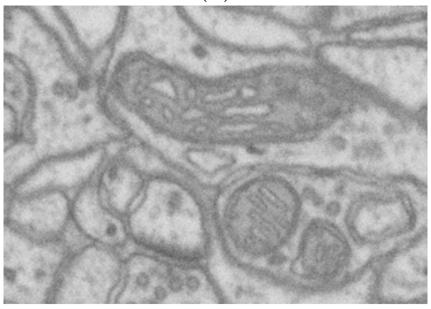
- Algorithm for summarizing databases by clustering images using a novel saliency based image descriptor.
- Application of creating image summaries using a novel image mosaics.
- Application of creating collage based image summaries.

1.4.5 Chapter 6: Superpixel segmentation

The focus of the chapter is a superpixel segmentation algorithm (see Fig. 1.8) that can be applied in feature extraction, which can in turn be used for object detection. Superpixels are clusters of spatially connected pixels, i.e segments of an image that share similar properties. We first present our superpixel segmentation algorithm. We then compare our superpixel segmentation algorithm to the existing ones in terms of under-segmentation error and boundary recall. We show how our superpixel segmentation can improve the accuracy and speed of existing algorithms. One such example is shown in performing object category recognition. Finally, we show the use our superpixels to detect and segment mitochondria from noisy medical images. An example of the detection is presented in Fig. 1.8(d).

(a)


(b)


(c)

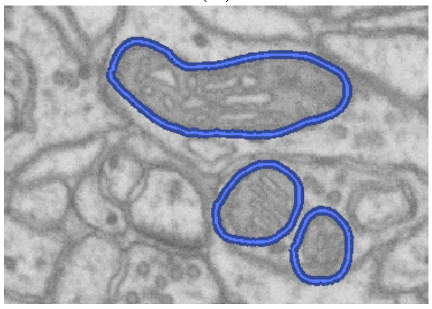

(d)

Figure 1.8: *(a) Input image. (b) Segmentation into superpixels of three different sizes. (c) Mitochondria in noisy medical images. (d) Detection of mitochondria.*

Contributions:

- Superpixel segmentation algorithm that outperforms state-of-the-art algorithms.
- Application in detection of mitochondria in medical images.

1.4.6 Chapter 7: Conclusions

In this chapter we present a summary of the thesis and discuss its contributions. We point towards the possibilities for improvement and the directions for future work.

Chapter 2

State of the art

The focus of this chapter is to give an overview of existing research relevant to the contributions made in this thesis. The literature survey consists of three main parts. The first part presents a broad classification of saliency detection techniques and their general limitations. The second part presents an overview of superpixel segmentation techniques, which are often used as a precursor to feature extraction for applications like object detection. On more than one occasion we use certain clustering algorithms in this thesis. The third part of this chapter therefore introduces the clustering techniques we use.

2.1 Saliency Detection Techniques

The term *saliency* was used by Tsotsos et al. [131] and Olshausen et al. [110] in their work on visual attention, by Sha'ashua and Ullman [122], and by Itti et al. [72] in their work on rapid scene analysis. It is widely accepted that unconscious human visual attention is guided by saliency. Saliency is known to be the consequence of *contrast*, *unpredictability*, *rarity*, or *surprise* [131, 98, 74, 70, 100].

Saliency estimation methods can broadly be classified as biologically based, purely computational, or a combination. The goal of any method, in general, is to detect properties of contrast, rarity, or unpredictability in images, of a central region with its surroundings, either locally or globally, using one or more low-level features like color, intensity, and orientation. Biological model based methods [72, 140, 50] attempt to mimic known models of the visual system for detecting saliency. Purely computational methods predominantly rely on principles of information theory, spectral domain processing, or signal processing to achieve this objective.

Some of these algorithms detect saliency over multiple scales [72, 4], while others operate on a single scale [98, 68]. In some cases individual feature maps are created separately and then combined to obtain the final saliency map [71, 98, 68, 50], while in some others, a combined-feature saliency map is computed [98, 4].

2.1.1 Biological model based approaches

Itti et al. [72] base their technique on the biologically plausible architecture proposed by Koch and Ullman [77]. They determine center-surround contrast using a Difference of Gaussians (DoG) approach. Walther and Koch [140] base their work on the model of Itti et al. [72] and extend it to infer the extent of *proto-objects* [115, 114] at the attended locations. This is similar to the later work of Han et al. [61] and Ko and Nam [76], who use their saliency maps to segment salient objects. Frintrop et al. [50] present a method inspired by Itti's method, but they compute center-surround differences using Difference of Boxes (DoB) filters, and use integral images to speed up the calculations.

2.1.2 Purely computational approaches

Most other methods are either purely computational, or partly rely on biological principles. Zhang et al. [149] present a Bayesian saliency model that combines target-independent bottom-up saliency and target-dependent top-down saliency. They focus on the bottom-up saliency part, which is independent of any knowledge of the target class and is computed as the self-information at a given point with respect to its features. Their model implies that the rarer a feature is, the more it will attract our attention. Self-information is also the basis of the bottom-up saliency work of Bruce and Tsotsos [24], Oliva et al. [109], Torralba and Oliva [126], and Mancas et al. [100], although the features used by these methods differ. Bruce and Tsotsos [24] employ features that were learned from natural images using independent component analysis (ICA), while Oliva and colleagues [126] use biologically inspired linear filters of different orientations and scales (steerable filters). Oliva et al. [109], and Torralba and Oliva [126] rely on the scene *gist* to compute saliency. This is equivalent to the method of maximizing self-information of Bruce and Tsotsos [24] if the neighborhood considered is the entire image. On similar lines, Mancas et al. [100] treat global rarity as the primary criterion for saliency and compute it as self-information using the histogram of local pixel-neighborhood means and variances. They show an application of their saliency detection in anisotropic filtering of images.

Zhang et al. [149] show two different methods of computing bottom-up saliency each of which uses a different set of features. The first method uses DoG outputs of color and luminance channels over four scales. The second method uses ICA features obtained from training image patches, similarly as Bruce and Tsotsos [24]. The two methods compute saliency as a linear sum of 12 DoG and 362 ICA features responses, respectively.

Gao et al. [54, 56, 55] propose a unified framework for top-down and bottom-up saliency as a classification problem with the objective of minimizing the classification error. They first apply this framework to object detection [56] in which a set of features is selected such that a class of interest is best discriminated from all other classes, and saliency is then computed as the weighted sum of features that are salient for that class. Gao et al. [54] define bottom-up saliency using the idea that pixel locations are salient if they are distinct from their surroundings. They use DoG and Gabor filters to compute features, and measure the saliency of a point as the Kullback Leibler (KL) divergence between the histogram of the filter responses at the point and the histogram of the filter responses in the surrounding region. Mahadevan and Vasconcelos [99] applied this bottom-up saliency detection to background subtraction in highly dynamic scenes.

Ma and Zhang [98] compute contrast at a location in an image with its surrounding region as a cumulative sum of color distances of the center region with surround regions. They use fuzzy growing on their saliency maps to confine segment rectangles containing the salient regions. Achanta et al. [4] estimate saliency using center-surround feature distances. Instead of using the more popular Difference of Gaussians (DoG) filters, they use a DoB approach using integral images (like Frintrop et al. [50]), which results in lower computational costs despite filtering full resolution images. Hu et al. [68] estimate saliency by applying heuristic measures on initial saliency estimates obtained by histogram thresholding of feature maps. Seo and Milanfar [120] present a method of saliency detection in images and video that measures similarity of feature descriptors at a location in an image (or video) with the surrounding descriptors.

Rosin [116] uses an edge-based scheme to compute image saliency for grayscale images. Simple gradient detection is performed on the input image using an operator like Sobel. The resulting gray level gradient map is thresholded at several gray level to produce a set of binary edge images. A distance transform is applied to each of the binary edge images to propagate the edge information. The resulting outputs are averaged to obtain the saliency map. A salient region is then extracted from the saliency map using a binary thresholding algorithm [130].

The first spectral domain approach for detecting saliency is due to Hou and Zhang [67], who compute saliency as the residual difference between the perceived spectrum and characteristic spectrum (termed spectral residual) of greyscale images. In their algorithm the phase component is not taken into account. Guo et al. [60] found that simply taking the inverse Fourier transform of the phase spectrum alone and discarding the amplitude produced equivalent results. This method was further simplified by Bian and Zhang [21] so that phase separation and multiplication with an exponential term before inverse transform are not needed. Both Guo et al. [60] and Bian and Zhang [21] extend their saliency detection models to color using the quaternion fourier transform [40].

2.1.3 Hybrid approaches

The third category of methods are those that incorporate ideas that are partly based on biological models and partly on computational ones. For instance, Harel et al. [62] create feature maps using Itti's method but perform their normalization using a graph-based approach. Bian and Zhang [21] explain the biological plausibility of their spectral domain saliency detection method. Similarly, for the techniques of Torralba et al. [126] and Zhang et al. [149], which rely on global scene context and natural image statistics, respectively, the authors allude towards the biological plausibility of their models.

2.1.4 Limitations and common criticisms

The saliency maps generated by several methods have low resolution [72, 98, 62, 50, 67]. Itti's method produces saliency maps that are just $1/256^{th}$ the original image size in pixels, while Hou and Zhang [67] output maps of size 64×64 pixels for any input image size. There are few saliency detection schemes [100, 4] that output saliency maps with original input resolution.

Some methods tend to highlight textured regions as more salient regardless of their context or surround [73, 55, 24]. Zhang et al. [149] explain that this is often the consequence of using distributions of local oriented filters responses.

Depending on the salient region detector, some maps additionally have ill-defined object boundaries [72, 62, 50], limiting their usefulness in certain applications. This arises from severe downsizing of the input image, which reduces the high frequency content in the original image considered in the creation of the saliency maps. Other methods highlight the salient object boundaries, but fail to uniformly highlight the entire salient region [98, 67, 100], or alternatively, better highlight smaller salient regions than larger ones [4, 67, 21]. These shortcomings result from the limited range of spatial frequencies retained from the original image in computing the final saliency map as well as the specific algorithmic properties.

As pointed out by Rosin [116], in certain methods such as those of Itti et al. [72] and Walther and Koch [140], finding and justifying appropriate parameters for the number, types, and sizes of the filters, normalization schemes etc, can be problematic. This is also the case with the approach of Aziz and Mertsching [18] that combines properties of orientation, color, size, symmetry, and eccentricity of regions obtained by their segmentation algorithm into a single measure of saliency.

Since most of the saliency detection algorithms compute *pre-attentive saliency*, i.e. fast bottom-up saliency, they should preferably be computationally inexpensive. The algorithms by Itti et al. [72] and Harel et al. [62], for instance, are computationally quite expensive, which can restrict the usefulness of such saliency detection algorithms.

2.2 Frequency domain analysis of saliency detection techniques

While the features used for saliency detection, and the linear and non-linear operations differ from model to model, interestingly, the appearance of most of the output saliency maps can be explained to a large extent from a unified frequency-domain perspective. In this section we examine the spatial frequency content retained from the original image in the creation of the saliency maps of nine state-of-the-art methods. We "designed" the frequency-analysis scheme previously [5] to assess the state-of-the art from a unified point of view. These methods are as follows: the classic multi-scale DoG based scheme of Itti et al. [72], the cumulative neighborhood contrast scheme by Ma and Zhang [98], the graph-based saliency scheme by Harel et al. [62], the scheme of attention using information maximization by Bruce and Tsotsos [25], the method of Mancas et al. [100] of self-information of local means and variances, the spectral residual scheme by Hou and Zhang [67], the scheme of saliency using natural statistics by Zhang et al. [149], the color and intensity based difference of boxes (DoB) scheme by Achanta et al. [4], and the spectral whitening using quaternion transform by Bian and Zhang [21]. These methods hereby are referred to as IT98, MA03, GB06, GR07, AIM07, SR07, SUN08, AC08, and SWQ09 respectively.

The choice of these algorithms is motivated by the following reasons: citation in literature (the classic approach of IT98 is widely cited), recency (GB06, GR07, AIM07, SR07, SUN08, AC08, SWQ09, are recent), and variety (IT98 and STB06 are biologically motivated, MA03 and GR07 are purely computational, GB06 is a graph-based hybrid approach, AIM07 uses an information maximization approach, SUN08 relies on natural image statistics, SR07 and SWQ09 estimate saliency in the frequency domain, and GR07 and AC08 output full-resolution maps).

2.2.1 Spatial frequency content of saliency maps

To analyze the properties of the nine saliency algorithms, we examine the spatial frequency content from the original image that is retained in computing the final saliency map. We provide the following analysis in one dimension and specify extensions to two dimensions when necessary. A summary of the frequency ranges as well as some other attributes of these methods is presented in Table 2.1.

IT98

In method IT98, a Gaussian pyramid of 9 levels (level 0 is the original image) is built with successive Gaussian blurring and downsampling by a factor of 2 in each dimension. In the case of the luminance image, this results in a successive reduction of the spatial frequencies retained from the input image. Each smoothing operation approximately halves the normalized frequency spectrum of the image. At the end of 8 such smoothing operations, the frequencies retained from the spectrum of the original image at level 8 range within $[0, \pi/256]$. The technique computes differences of Gaussian-smoothed images from this pyramid, resizing them to size of level 4, which results in using frequency content from the original image in the range $[\pi/256, \pi/16]$. In this frequency range the DC (mean) component is removed along with approximately 99% ($(1 - \frac{1}{16^2}) \times 100$) of the high frequencies for a 2-D image. As such, the net information retained from the original image contains very few details and represents a very blurry version of the original image (see the bandpass filtered image of Fig. 2.1). This is similarly the case with the model by Walther and Koch [140] for finding proto-objects that extends the model of Itti et al. [72].

MA03

In method MA03, a low-resolution image is created by averaging blocks of pixels and then downsampling the filtered image such that each block is represented by a single pixel having its average value. The averaging operation performs low-pass filtering. While the authors do not provide a block size for this operation, we obtained good results with a block size of 10×10 pixels, and as such the frequencies retained from the original image are in the range $[0, \pi/10]$.

GB06

In method GB06, the initial steps for creating feature maps are similar to IT98, with the exception that fewer levels of the pyramid are used to find center-surround differences. The spatial frequencies retained are within the range $[\pi/128, \pi/8]$. Approximately 98% ($(1 - \frac{1}{8^2}) \times 100$) of the high frequencies are discarded for a 2D image. As illustrated in Fig. 2.1(d), there is slightly more high frequency content than in 2.1(b).

AIM07

The saliency model of AIM07 relies on estimating the distribution of each of bases of ICA (Independent Component Analysis) projection of random image patches across an image. Several thousand patches of size 7×7 pixels are used to compute the ICA bases. In the default mode of operation, the input image is not resized. However, the use of patches (or averaged blocks as in MA03) implicitly limits the high frequency content that makes it into the final saliency maps. The larger the patch size, the more high frequency content gets discarded. So the frequency content of AIM07 is roughly in the range of $[0, \pi/7]$. This explains why the salient objects have thick borders and they appear larger than in the input images.

GR07

The technique of GR07 bases saliency on the criterion of global rarity of a given feature. As features, they use the mean and variance values in 3×3 local neighborhoods. According to the authors, the size of this neighborhood has little impact on the final result. In the default mode of operation the image is not resized. As such, the method uses the entire bandwidth of $[0, \pi]$. However, the averaging of the two saliency maps computed using mean and variance values results in suppressing low frequencies in the output saliency map.

SR07 and SWQ09

In method SR07 (as well as PFT and PQFT [60], and SWQ09 [21]), the input image is resized to 64×64 pixels (via low-pass filtering and downsampling) based on the argument that the spatial resolution of pre-attentive vision is very limited. The resulting frequency content of the resized image therefore varies according to the original size of the image. For example, with input images of size 320×320 pixels (which is the approximate average dimension of the images of our test database), the retained frequencies are limited to the range $[0, \pi/5]$. As seen in Fig. 2.1(g and j), higher frequencies are smoothed out. Obviously, if the input images are larger, even more high frequency content will get discarded when resizing them to a size of 64×64 pixels.

SUN08

SUN08 [149] uses four scales of DoG (with surround $\sigma = 4$, 8, 16, or 32 pixels) on each of the three channels, leading to 12 feature response maps. In the default mode input images are downsampled by 2 along both dimensions. In this case the largest possible range of spatial frequencies retained is $[\pi/100, \pi/2]$.

AC08

In method AC08 [4], a difference-of-boxes filter is used to estimate center-surround contrast. The lowest frequencies retained depend on the size of the largest surround filter (which is half of the image's smaller dimension) and the highest frequencies depend on the size of the smallest center filter (which is one pixel). As such, method AC08 effectively retains the entire range of frequencies $(0, \pi]$ with a notch at DC. All the high frequencies from the original image are retained in the saliency map but but the low frequencies get suppressed in the final saliency map as a result of averaging the filter output (see Fig. 2.1(h)).

Method	Freq. range	Resolution	Complexity	Computational approach	Category	
	Feautures					
IT98	$[\pi/256, \pi/16]$	$S/256$	$O(k_{IT98}N)$	Center-surround, DoG	Biological	Color, Intensity, and orientation
MA03	$[0, \pi/10]$	$S/100$	$O(k_{MA03}N)$	Center-surround	Computational	Color and intensity
GB06	$[\pi/128, \pi/8]$	$S/64$	$O(k_{GB06}N^4K)$	Graph-based	Hybrid	Color, Intensity, and orientation
GR07	$[0, \pi]$	S	$O(k_{GR07}N)$	Self-information	Computational	Intensity
AIM07	$[0, \pi/7]$	S	$O(k_{AIM07}N)$	Information maximization	Hybrid	ICA bases of intensity patches
SR07	$[0, \pi/5]$	64×64	$O(k_{SR07}N)$	Spectral domain	Computational	Intensity
AC08	$(0, \pi]$	S	$O(k_{AC08}N)$	Center-surround, DoB	Computational	Color and intensity
SUN08	$[\pi/100, \pi/2]$	$S/4$	$O(k_{SUN08}N)$	Self-information	Hybrid	ICA bases of intensity patches
SWQ09	$[0, \pi/5]$	64×64	$O(k_{SWQ09}N)$	Spectral domain	Hybrid	Color and intensity

Table 2.1: *A comparison of 1-D frequency range, saliency map resolution, and computational efficiency. S is the input image size in pixels. Although the complexity of all methods except GB06 is proportional to N, the operations per pixel in these methods vary ($k_{MA03} < k_{SR07} < k_{AC08} < k_{GR07} < k_{SWQ09} < k_{AIM07} < k_{SUN08} < k_{IT98} < k_{GB06}$). GB06 has an overall complexity of $O(k_{GB06}N^4K)$, depending on the number of iterations K.*

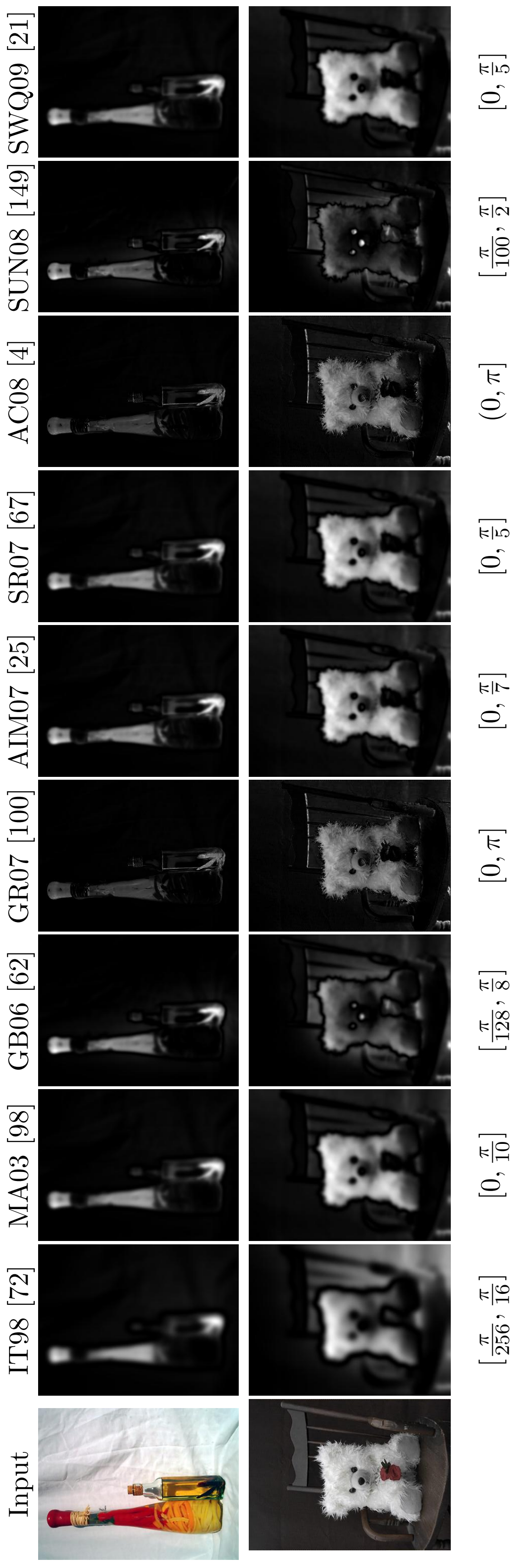


Figure 2.1: *Original image bandpass filtered with cut-off frequencies given in Table 2.1. The spatial frequency content retained explains the final appearance of the saliency map in most of the cases. For instance, IT98 can only show blobby salient regions since the method discards a lot of high frequency information. MA03 and GB06 usually highlight objects, especially large ones, near the object boundaries. GR07 and AC08 retain most of the high frequency content and therefore have well-defined object boundaries.*

2.2.2 Other algorithmic properties of the methods

This section touches upon other aspects of some of the techniques presented in the previous section. This provides further insight into the properties of the saliency maps generated by these algorithms. The reader may want to consult the visual comparison of the saliency maps given in Figures 3.8 to 3.12.

MA03

In MA03, the saliency value at each pixel position (i, j) is given by:

$$S(x,y) = \sum_{(m,n)\in\mathcal{N}} d[\mathbf{p}(x,y), \mathbf{q}(m,n)] \tag{2.1}$$

where $\mathcal{N}$ is a small neighborhood of a pixel (in the resized image obtained by 10×10 box filtered and downsampled image) at position (x, y) and d is a Euclidean distance between CIELUV pixel vectors $\mathbf{p}$ and $\mathbf{q}$. In our experiments explained in Section 3.5, we choose $\mathcal{N}$ to be a 3×3 neighborhood. The method is fast but has the drawback that the saliency values at either side of an edge of a salient object are high, i.e the saliency maps show the salient object to be bigger than it is, which gets more pronounced if block sizes are bigger than 10×10. In addition, for large salient objects, the salient regions are not likely to be uniformly highlighted.

SR07

In SR07, the *spectral residual* R is found by subtracting a smoothed version of the FFT (Fast Fourier Transform) log-magnitude spectrum from the original log-magnitude spectrum. The saliency map is the inverse transform of the spectral residual. The FFT is smoothed using a separable 3×3 mean filter. Examining this operation in one dimension, this is equivalent to forming the residue $R(k)$ as:

$$R(k) = ln|X(k)| - f * ln|X(k)| \tag{2.2}$$

with $f = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$, and $*$ denoting convolution. A simple manipulation of this equation demonstrates that the (1-D) spectral residue $R(k)$ can be written as:

$$R(k) = \frac{1}{3} ln\left[\frac{|X(k)|^2}{|X(k-1)||X(k+1)|}\right] \tag{2.3}$$

When this is performed in two dimensions on the 2-D FFT coefficients, only 3 low-frequency AC coefficients are divided by the DC (mean) value (if the FFT coefficients are circularly extended for the filtering, then the 3 highest-frequency FFT AC coefficients are also divided by the mean). On the other hand, in SW [21] each FFT coefficient is divided by its own magnitude. In comparison to these two methods, contrast measures typically normalize all FFT AC coefficients by the mean value [64].

AC08

In method AC08, center-surround contrast is computed as a Euclidean distance between average Lab vectors of the center and surround regions. The saliency value at pixel position (x, y) is given as:

$$\begin{aligned} S(x,y) &= \tfrac{1}{3}[F_{\frac{H}{2}}(x,y) + F_{\frac{H}{4}}(x,y) + F_{\frac{H}{8}}(x,y)] \\ F_t(x,y) &= d(\mathbf{c}(x,y), \mathbf{s}_t(x,y)) \end{aligned} \tag{2.4}$$

where feature map value $F_t(x, y)$ is found as a Euclidean distance d between the CIELAB pixel vector $\mathbf{c}(x, y)$ (center) and the average CIELAB pixel vector $\mathbf{s}_t(x, y)$ in window t (surround). The square surround region is varied as $t = \{\frac{H}{2}, \frac{H}{4}, \frac{H}{8}\}$, assuming H, the height of the image, to be smaller than the width W of the image.

Objects that are smaller than a filter size are detected (i.e. highlighted in the saliency map) completely, while objects larger than a filter size are only partially detected (closer to edges). Smaller objects that are well detected by the smallest filter are detected by all three filters, while larger objects are only detected by the larger filters. Since the final saliency map is an average of the three feature maps (corresponding to detections of the three filters), small objects will almost always be better highlighted.

Such averaging of band pass filtered images is also done in the DoG version of SUN08 [149] saliency detection method and consequently the low frequencies that are used in the making of the saliency map get suppressed.

2.3 Object scale and frequency cut-offs

To summarize, low frequency content is essential if the salient regions need to be uniformly highlighted (and not just at the object borders). The more high frequency content is retained, the sharper are the boundaries, and the better-defined is the extent of the salient object in the saliency map. The low frequency content from the input image that should be exploited depends on how large the salient objects are, i.e. what their scale is.

Object scale and the low-frequency cut-off of bandpass filtering are closely related. The larger the scale of the object, the lower should be the low-frequency cut-off of the filter in order to completely detect, i.e., highlight it. For the purpose of edge detection, which is finer scale information, the low-frequency cut-off value is high [101]. This obviously narrows down the bandwidth since the difference between the low and high cut-offs is small. Similarly, for corner and interest point detection a narrow bandwidth is used [91, 95].

The scale of a symmetric convex object can be detected using local scale space maxima [91, 89]. However, for large objects of elongated and irregular shapes it is harder to detect the characteristic scale. A way to circumvent this is to segment objects of interest a priori but this itself is an ill-posed problem with no reliable solution yet.

If the scale of the object of interest in an image is not known a priori then preferably all of the low frequency content should be used. Similarly, the amount of high frequency that should be used should depend on how well-defined the salient object boundaries should be, and how much high frequency detail (i.e. noise, or image substructures like texture, small objects, etc.) is to be discarded.

2.4 Superpixel Segmentation Techniques

In computer vision, *image segmentation* refers to the process of partitioning an image into groups or clusters of spatially connected pixels, which share similar properties like intensity, color, and texture. Each of the clusters is called a segment of the image. It is for this reason that for images, clustering and segmentation are often used synonymously. The goal of segmentation is usually to simplify the representation of an image such that it is more meaningful and easier to analyze.

Most computer vision problems like depth estimation, object recognition etc. often require image abstraction in the form of segmentation since pixel level information is too fine grained and often unwieldy. Popular image segmentation algorithms [45, 30] try to segment images in a semantically meaningful way. But this is an ill-posed problem, and such segmentation algorithms generally output too few or too many segments. The former problem is called *under-segmentation*, and the latter, *over-segmentation*.

A practical way of circumventing semantic limitations of segmentation for certain applications is to allow over-segmentation and build upon the output. Over-segmented images have been used for depth estimation [65, 66], user-guided image segmentation [86, 63], and so on. The segments of an over-segmented image are termed *superpixels* [113]. *Superpixel segmentation* is a specific case of the general problem of image segmentation.

In this section, we specifically review the state-of-the-art in superpixel segmentation techniques. Not all of the techniques were designed initially for the specific purpose of generating superpixels and therefore often lack the ability to control the size, number, and compactness of the superpixels. We include such techniques in our discussion nonetheless as they were used in literature for the purpose of generating superpixels. We broadly classify superpixel algorithms into graph-based and gradient-ascent-based algorithms. Our survey, summarized in Table 2.2, considers the quality of segmentation, and the ability of these algorithms to control the size and number of superpixels.

	Graph-based			Gradient-ascent-based			
Properties	GS04	NC05	SL08	WS91	MS02	TP09	QS09
Superpixel No. Ctrl.	No	Yes	Yes	No	No	Yes	No
Compactness Ctrl.	No	Yes	Yes	No	No	Yes	No
Complexity $O(.)$	$N \log N$	$N^{3/2}$	$N^2 \log N$	$N \log N$	N^2	N	dN^2
Parameters	2	1	3	1	3	1	2

Table 2.2: *Comparison of state-of-the-art superpixel segmentation algorithms. N is the number of pixels in the image. GS04 and QS09 do not offer explicit control of the number of superpixels. SL08 complexity given in this table does not take into account the complexity of the boundary map computation. GS04 is $O(N \log N)$ complex and is quite fast in practice while TP09, which is $O(N)$ complex, is about 10 times slower than GS04 for 481×321 pixel images. In the case of QS09, d is a small constant (refer to [134] for details). The number of parameters listed in the table is the minimum required for typical usage.*

2.4.1 Graph-based algorithms

In graph-based algorithms, each pixel is treated as a node in a graph and the edge weight between two nodes is set proportional to the similarity between the pixels. Superpixel segments are extracted by effectively minimizing a cost function defined on the graph.

The Normalized cuts algorithm [121], recursively partitions a given graph using contour and texture cues, thereby globally minimizing a cost function defined on the edges at the partition boundaries. It is the basis of the superpixel segmentation scheme of [113] and [107] (NC05). NC05 has a complexity of $O(N^{\frac{3}{2}})$ [83], where N is the number of pixels. There have been attempts to speed up the algorithm [32], but it remains computationally expensive for large images. The superpixels from NC05 have been used in body model estimation [107] and skeletonization [82].

Felzenszwalb and Huttenlocher [45] (GS04) present a generic graph-based image segmentation scheme that has been used to generate superpixels. This algorithm performs an agglomerative clustering of pixel nodes on a graph, such that each segment, or superpixel, is the shortest spanning tree of the constituent pixels. GS04 has been used for depth estimation [65]. It is $O(N \log N)$ complex and is quite fast in practice as compared to NC05. However, unlike NC05, it does not offer an explicit control on the number of superpixels or their compactness.

A superpixel lattice is generated by Moore et al. [106] (SL08) by finding optimal vertical (horizontal) seams/paths that cut the image, within vertical (horizontal) strips of pixels, using graph cuts on strips of the image. While SL08 allows control of the size, number, and compactness of the superpixels, the quality and speed of the output strongly depend on pre-computed boundary maps.

2.4.2 Gradient-ascent-based algorithms

Usually starting from an initial rough clustering, during each iteration gradient ascent methods refine the clusters from the previous iterations until some convergence criterion is reached.

Mean shift [30] (MS02) is a mode-seeking algorithm that generates image segments by recursively moving to the kernel smoothed centroid for every data point in the pixel feature space, effectively performing a gradient ascent. The generated segments/superpixels can be large or small based on the input kernel parameters, but there is no direct control over the number, size, or compactness of the resulting superpixels.

Quick-shift [134] (QS08) is also a mode-seeking segmentation scheme like mean shift, but is faster in practice. It moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. The algorithm is non-iterative, and like mean shift, does not allow to explicitly control the size or number of superpixels. Superpixels from quick-shift have been used in applications like object localization [52] and motion segmentation [17].

We include two other segmentation methods in the category of gradient ascent algorithms: Watersheds [136] (WS91) and Turbopixels [83] (TP09). General watershed algorithms perform gradient ascent from local minima in the image plane in order to obtain watersheds, i.e. lines that separate catchment basins. Vincent and Soille [136] propose a fast version based on queuing of pixels. *Lazy Snapping* [86] applies graph cuts to the graph built on the superpixels output by this algorithm.

TP09 generates superpixels by progressively dilating a given number of seeds in the image plane, using computationally efficient level-set based geometric flow. The geometric flow relies on local image gradients, and aims to distribute superpixels evenly on the image plane. Unlike WS91, superpixels from TP09 are constrained to have uniform size, compactness, and adherence to object boundaries.

2.4.3 Limitations of superpixel algorithms

A strong requirement of superpixels is to adhere well to object boundaries. Some techniques, like NC05 [121] do not do this very effectively. GS04 [45] for instance shows better adherence to object boundaries but this comes with other drawbacks like non-uniform superpixel sizes.

In certain applications, local features are extracted superpixels of the image. In particular, if such features are region based, like SIFT [95] or SURF [20], the features are more meaningful and discriminative if the superpixels are compact (i.e. low perimeter to area ratio) and of roughly similar sizes. Algorithms like NC05 [121] and TP09 [83] provide compact and roughly equally sized superpixels. However, both these algorithms exhibit poorer adherence to object boundaries. QS09 creates compact superpixels but the sizes are not uniform, which makes it difficult to compute local features that strongly depend on this feature of superpixels.

Algorithms like NC05 [121] and TP09 [83] are computationally far more expensive than competing algorithms. These algorithms are therefore less suitable when a large number of images need to be processed or when the image resolution is greater than a million pixels. GS04 [45] is computationally very efficient but it provides unevenly sized superpixels and an unpredictable number of them.

It is worth mentioning that there appears to be a trade-off between how visually appealing the superpixel output is and how well the superpixels adhere to object boundaries. Usually, visually pleasing superpixels, arguably those generated by NC05 [121] and TP09 [83], show poorer performance in this respect as apposed to their 'uglier' counterparts like GS04 [45] and QS09 [134].

2.5 Clustering

Clustering is an operation we extensively use in this thesis. It appears as a key step in salient object segmentation application and is at the heart of the SLIC superpixel segmentation algorithm (Section 6.2). Clustering is a vast area of research and in this section we only discuss those algorithms that are employed in subsequent chapters. A comparative summary of the clustering algorithms we discuss in this section is given in Table 2.3.

2.5.1 k-means clustering

Given a set of data vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ in a d-dimensional space, the k-means clustering algorithm aims to partition the N data vectors into k clusters $(k < N)$ $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ so as to minimize the intra-cluster variance:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|\mathbf{x}_j - \mu_i\|_2^2 \tag{2.5}$$

where μ_i is the mean of the points in C_i.

The k-means algorithm, also referred to as the Lloyd's algorithm [94] proceeds by alternating between the steps of *assignment* and *update*. Initially, a set of seed cluster centers is chosen, either randomly or using specific algorithms (like the k-means++ algorithm [13]). During the assignment step, each data vector is assigned to the nearest cluster center. During the update step, the cluster centers are updated to be equal the mean of all the data vectors belonging to each cluster. These two steps are repeated in succession until the cluster centers no longer change or change minimally. These two steps can be stated formally as follows:

Assignment: Assign each data vector to the cluster with the closest mean at iteration t.

$$C_i^{(t)} = \left\{ \mathbf{x}_j : \|\mathbf{x}_j - \mu_i^{(t)}\|_2 \leq \|\mathbf{x}_j - \mu_{i^*}^{(t)}\|_2 \ \forall i^* = 1, ..., k \right\} \tag{2.6}$$

Update: Calculate the new mean vectors to be the centroid of the data vectors in the cluster for the $t+1$th iteration:

$$\mu_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j \tag{2.7}$$

where $|.|$ gives the number of data points in the cluster. The update can easily be derived from Eq. 2.5 if the distance measure used is Euclidean. The algorithm is considered to have converged when the assignments or the intra-cluster variances no longer change. While there is no guarantee the algorithm will converge to the global optimum, and that the result may depend on the choice of the initial cluster seeds [13], in practical applications the algorithm converges to a minimum in a small number of iterations that is much less than the number of data points [37].

2.5.2 Mean shift clustering

Mean shift is a non-parametric clustering technique. Unlike the k-means clustering approach it makes no assumptions about the shape of the distribution or the number of clusters. It is used for image segmentation and feature tracking in videos. The mean shift procedure was originally presented by Fukunaga and Hostetler [51]. The algorithm was extended by Cheng [28] for image analysis and was popularized for image segmentation by Comaniciu and Meer [30].

The idea behind mean shift clustering is to treat the data points in a given d-dimensional feature space as a probability density function where dense regions correspond to the local maxima or modes of the underlying distribution. For each data point in the feature space, a gradient ascent procedure is applied on a locally estimated density until convergence. The stationary points of this process represent the modes of the distribution. All the data points associated with a given stationary point are considered members of the same cluster. The iterative clustering process is as follows:

- Given n data points $\mathbf{x}_i \in \mathbb{R}^d$ and a radially symmetric kernel K (usually Epanechnikov or Gaussian kernel) of radius h, compute the mean shift vector

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i K\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} K\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x} \tag{2.8}$$

  at a data point $\mathbf{x}$.

- Translate the density estimation kernel by $\mathbf{m}(\mathbf{x})$.

- Iterate the first two steps until convergence.

The most computationally expensive part of the mean shift procedure is to identify the neighbors of a point in space. This is cumbersome for high dimensional feature spaces. Another limitation is that the value of the kernel radius parameter h needs to be set experimentally.

2.5.3 Graph-based image segmentation

An image based graph is generally represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $p_i \in \mathcal{V}$ corresponds to a pixel in the image and the edges $\mathcal{E}$ connect certain pairs of neighboring pixels (with 4 or 8 connectivity). The weight of each edge joining any two nodes p_i and p_j is given by $w(p_i, p_j)$ and is often computed as a similarity measure between the two pixels.

Felzenszwalb and Huttenlocher [45] proposed a technique that segments an image by agglomerative clustering of pixels on such an image based graph. The method is very similar to the Kruskal's shortest spanning tree algorithm where the tree is formed by choosing edges in increasing order of their weight. The main difference is that goal is not to create a single spanning tree cluster but several subtree clusters by introducing stopping criteria during tree formation. This is done with the help of the strongest edge weight, or the *maximum internal weight* $Int(C)$, which is kept track of as each subtree component C grows by adding more edges. The steps of the segmentation technique are as follows:

- Compute all edge weights as the Euclidean distance between the RGB vectors of pixels connected by the edges.
- Sort the edge weights in increasing order.
- In the beginning, each pixel is a component on its own. The maximum internal weight of each component $Int(C)$ set to a constant K_d, input by the user. As soon as a component is bigger than one pixel, this internal weight value is updated to $Int(C) + \frac{T}{|C|}$, where $|.|$ gives the cardinality of the component, and T is a constant that is either set by the user or made a function of K_d (usually $T = K_d/10$).
- For every next edge weight in the sorted list, join any two components C_a and C_b along the edge that joins them if its weight is *less* than $\min(Int(C_a), Int(C_b))$.

Prior to performing the segmentation, the image is Gaussian filtered to reduce the effects of noise and blocking artifacts. The algorithm is computationally efficient, with a complexity of $N \log(N)$, due to the sorting operation. One of the problems that arises is that there are several small components that remain at the end of the component growing process. Thus as a post processing step, all components that have a size smaller than a constant *minSize* input by the user, are merged into larger components along the weakest connecting edge. As input, the algorithm takes the value of the standard deviation for the Gaussian blur, K_d, $minSize$, and T.

2.5.4 Source-sink graph-cuts

Many current graph-partitioning approaches to segmentation, such as the one by Boykov and Jolly [23, 118], rely on minimizing a global objective function defined on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. It is often expressed as:

$$E(c|p,\lambda) = \sum_i \underbrace{\psi(c_i|p_i)}_{\text{unary term}} + \lambda \sum_{(i,j)\in\mathcal{E}} \underbrace{\phi(c_i, c_j|p_i, p_j)}_{\text{pairwise term}} \ , \tag{2.9}$$

where $c_i \in \{source, sink\}$ and the weight λ controls the relative importance of the so-called *unary* and *pairwise* terms. The *unary* term ψ assigns to each pixel its potential to belong to the *source* or to the *sink*. The *pairwise* term ϕ promotes coherence among neighboring nodes of the graph. A global optimum of such an objective function can usually be found using a standard mincut-maxflow algorithm. In optimization theory, the mincut-maxflow theorem states that in a flow network, the maximum amount of flow passing from the *source* to the *sink* is equal to the minimum capacity that needs to be removed from the network so that no flow can pass from the source to the sink. The Ford-Fulkerson algorithm [47] or its specialization, the Edmonds-Karp algorithm [38] is often used to minimize the objective function.

2.5.5 Geodesic distance computation

Geodesic distance computation is not a clustering technique by itself. But it can be used for clustering to obtain similar results as certain graph-based algorithms. Geodesic distances can also replace the more common Euclidean distances in certain clustering algorithms such as the k-means. So we devote this sub-section to the discrete computation of geodesic distances.

In the image plane, for two pixel $\mathbf{I}(p_i)$ and $\mathbf{I}(p_j)$, the unsigned geodesic distance from one to another is defined as:

$$\mathcal{G}(\mathbf{I}(p_i), \mathbf{I}(p_j)) = \min_{P\in\Gamma} d(P) \tag{2.10}$$

where Γ is the set of all paths between $\mathbf{I}(p_i)$ and $\mathbf{I}(p_j)$. A path P is defined as a sequence of spatially neighboring pixels $\{p_1, p_2, ...p_n\}$ in 4 or 8-connectivity. The cost $d(P)$ of the path P is computed as

$$d(P) = \sum_{i=2}^{n} \|\mathbf{I}(p_i) - \mathbf{I}(p_{i-1})\| \tag{2.11}$$

Eq. 2.10 and Eq. 2.11 hold for the discrete computation of a geodesic path. In such a case it is equivalent to the Dijkstra's algorithm [36] for computation of the shortest path on a graph.

Algorithm	Type	Uses
k-means	Iterative	Database saliency (Section 5.1), SLIC superpixels (Section 6.2)
Mean shift	Iterative	Salient object segmentation (Section 4.1.1)
Graph-based segmentation	Agglomerative	Superpixel-graph-based segmentation (Section 6.4.2)
Graph-cuts	Partitional	Salient object segmentation (Section 4.1.2), Mitochondria segmentation (Sections 6.4.3 and C.2)
Geodesic distances	Iterative	Salient object segmentation (Section 4.1.3), Pretty superpixels (Section 6.6)

Table 2.3: *Comparison of clustering algorithms used in this thesis.*

2.6 Summary of the chapter

In this chapter we reviewed the literature for saliency detection, superpixel segmentation, and selected clustering techniques. For saliency detection, we reviewed the state-of-the-art techniques and analyzed some of the algorithms from a frequency-domain perspective. We concluded that there is a need for algorithms that provide saliency maps with sharp object boundaries, well-highlighted salient objects, and have a low computational cost. In the literature review for superpixel segmentation we noted that most of the algorithms suffer from some drawbacks. There is room for algorithms that can provide compact and uniformly-sized superpixels, which provide good adherence to object boundaries, need few input parameters, and are computationally inexpensive to generate. Finally, we also discussed the clustering algorithms that are put to use in later chapters.

Chapter 3

Saliency Detection

This chapter deals with techniques for task-independent object detection, i.e saliency detection. The observations we made in the previous chapter using a frequency-domain analysis form the basis of the two novel saliency detection algorithms we present in this chapter[1]. These two algorithms are compared with several other state-of-the-art techniques in terms of precision and recall measures with respect to a publicly available ground truth database.

3.1 Requirements for a saliency map

In Section 2.1.4 the shortcomings of existing saliency detection methods were mentioned. We set the following requirements for a saliency detector:

1. Emphasize salient objects of all sizes.
2. Uniformly highlight whole salient regions.
3. Establish well-defined boundaries of salient objects.
4. Disregard high frequencies arising from texture, noise, and blocking artifacts, if any.
5. Compute saliency efficiently.
6. Output full resolution saliency maps.
7. Use minimal or no free parameters.

To highlight large salient objects, we need to consider very low frequencies from the original image. This also helps highlight salient objects uniformly. In order to have well defined boundaries, we need to retain high frequencies from the original image. However, to avoid noise, coding artifacts, and texture patterns, the highest frequencies need to be disregarded.

[1]The content presented in this chapter can be found in references [5] and [9]

3.2 Saliency detection algorithm - I

In this section we present the first saliency detection algorithm based on the insight gained from Section 2.2.

3.2.1 Saliency using bandpass filtering

We propose to compute saliency using center-surround bandpass filtering as we can easily control the low and high frequency cut-offs to suit our needs of saliency detection. It is also one of the most frequently used means of saliency detection [72, 98, 62]. For this purpose, we choose the Difference of Gaussians (DoG) filter (Eq. 3.1). The DoG filter is widely used in edge detection since it closely and efficiently approximates the Laplacian of Gaussian (LoG) filter, cited as the most satisfactory operator for detecting intensity changes when the standard deviations of the Gaussians are in the ratio 1:1.6 [101]. A DoG filter is a simple bandpass filter whose cut-off values are controlled by the standard deviations σ_1 and σ_2 and whose bandwidth is controlled by the ratio $\sigma_1 : \sigma_2$. The relation between the σ and the cut-off value is given in Appendix B.2. The DoG filter has also been used for interest point detection [95] and saliency detection [72, 62]. For tasks like edge detection, interest point detection, or saliency detection, only the magnitude of the DoG filter output is of interest. The DoG filter is given by:

$$\begin{aligned} DoG(x,y,\sigma_1,\sigma_2) &= \frac{1}{2\pi}\left[\frac{1}{{\sigma_1}^2}e^{-\frac{(x^2+y^2)}{2{\sigma_1}^2}} - \frac{1}{{\sigma_2}^2}e^{-\frac{(x^2+y^2)}{2{\sigma_2}^2}}\right] \\ &= G(x,y,\sigma_1) - G(x,y,\sigma_2), \end{aligned} \tag{3.1}$$

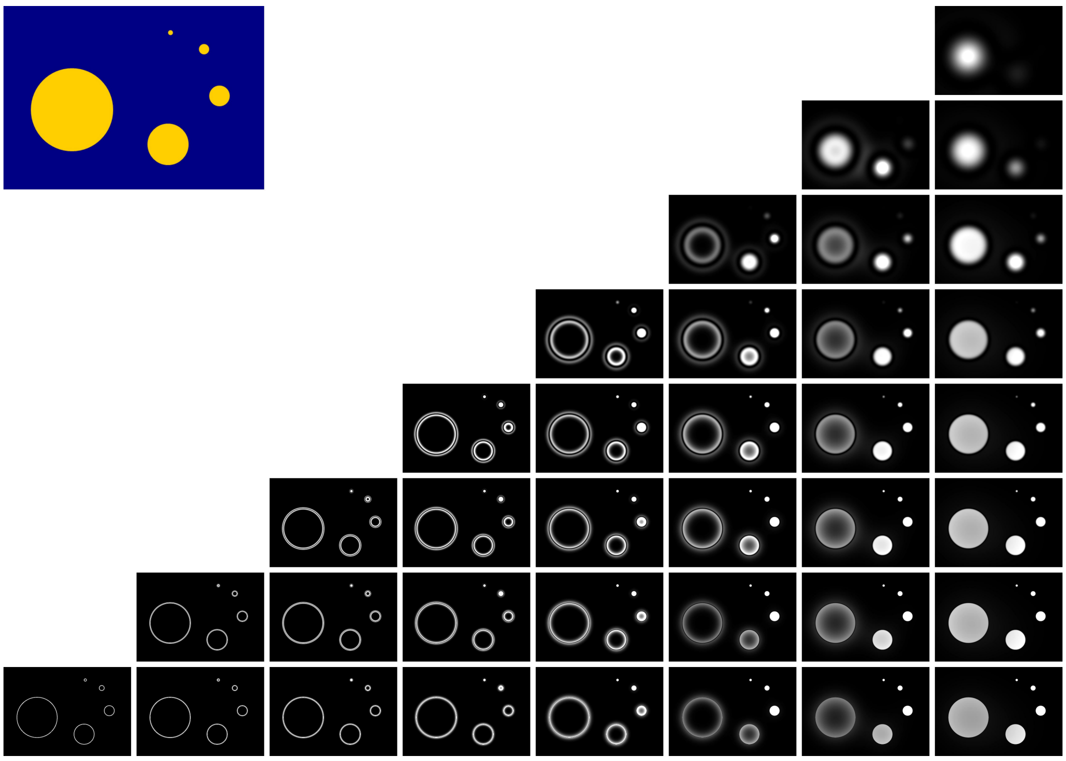
where σ_1 and σ_2 are the standard deviations of the Gaussian ($\sigma_1 > \sigma_2$).

3.2.2 Parameter selection

We use color and intensity features for our saliency detection. This allows us to use a single feature vector in the CIELAB color space and simplifies feature combination. The CIELAB color space is used since Euclidean distances in this color space are approximately perceptually uniform. This approach is similar to the one used by Achanta et al. [4] in AC08 and different from that of Itti et al. [72] where color and intensity features are decoupled and separate feature maps are created for them. Since we are only interested in the magnitude of the DoG filter output, we can rewrite Eq. 3.1 to use color and intensity information for computing saliency as

$$S(x,y) = \|\mathbf{I}_{\sigma_1}(x,y) - \mathbf{I}_{\sigma_2}(x,y)\| \tag{3.2}$$

where $S(x,y)$ is the pixel saliency value at position (x,y), $\mathbf{I}_{\sigma_1}$ and $\mathbf{I}_{\sigma_2}$ are obtained by Gaussian filtering each channel of the CIELAB image using surround and center standard deviations σ_1 and σ_2, respectively, and $\|.\|$ is the L_2 norm (i.e. Euclidean



							$[\frac{\pi}{320}, \frac{\pi}{160}]$	$\sigma_2 = 0$
						$[\frac{\pi}{160}, \frac{\pi}{80}]$	$[\frac{\pi}{320}, \frac{\pi}{80}]$	$\sigma_2 = 1$
					$[\frac{\pi}{80}, \frac{\pi}{40}]$	$[\frac{\pi}{160}, \frac{\pi}{40}]$	$[\frac{\pi}{320}, \frac{\pi}{40}]$	$\sigma_2 = 2$
				$[\frac{\pi}{40}, \frac{\pi}{20}]$	$[\frac{\pi}{80}, \frac{\pi}{20}]$	$[\frac{\pi}{160}, \frac{\pi}{20}]$	$[\frac{\pi}{320}, \frac{\pi}{20}]$	$\sigma_2 = 4$
			$[\frac{\pi}{20}, \frac{\pi}{10}]$	$[\frac{\pi}{40}, \frac{\pi}{10}]$	$[\frac{\pi}{80}, \frac{\pi}{10}]$	$[\frac{\pi}{160}, \frac{\pi}{10}]$	$[\frac{\pi}{320}, \frac{\pi}{10}]$	$\sigma_2 = 8$
		$[\frac{\pi}{10}, \frac{\pi}{5}]$	$[\frac{\pi}{20}, \frac{\pi}{5}]$	$[\frac{\pi}{40}, \frac{\pi}{5}]$	$[\frac{\pi}{80}, \frac{\pi}{5}]$	$[\frac{\pi}{160}, \frac{\pi}{5}]$	$[\frac{\pi}{320}, \frac{\pi}{5}]$	$\sigma_2 = 16$
	$[\frac{\pi}{5}, \frac{\pi}{2.5}]$	$[\frac{\pi}{10}, \frac{\pi}{2.5}]$	$[\frac{\pi}{20}, \frac{\pi}{2.5}]$	$[\frac{\pi}{40}, \frac{\pi}{2.5}]$	$[\frac{\pi}{80}, \frac{\pi}{2.5}]$	$[\frac{\pi}{160}, \frac{\pi}{2.5}]$	$[\frac{\pi}{320}, \frac{\pi}{2.5}]$	$\sigma_2 = 32$
$[\frac{\pi}{2.5}, \pi]$	$[\frac{\pi}{5}, \pi]$	$[\frac{\pi}{10}, \pi]$	$[\frac{\pi}{20} \pi]$	$[\frac{\pi}{40}, \pi]$	$[\frac{\pi}{80}, \pi]$	$[\frac{\pi}{160}, \pi]$	$[\frac{\pi}{320}, \pi]$	$\sigma_2 = 64$
$\sigma_1 = 1$	$\sigma_1 = 2$	$\sigma_1 = 4$	$\sigma_1 = 8$	$\sigma_1 = 16$	$\sigma_1 = 32$	$\sigma_1 = 64$	$\sigma_1 = 128$	

Figure 3.1: *Center-surround DoG filtering of an artificial image containing five salient objects of different sizes using Eq. 3.2. Surround* σ_1 *increases from left to right (x-axis) and center* σ_2 *increases from bottom to top (y-axis). As* σ_1*, of the surround gaussian, is increased, objects of larger scales get highlighted progressively. As* σ_2*, of the center gaussian, is increased, more of the high frequency content is lost, so that the smaller scale objects are progressively lost. The table below shows frequency ranges for each saliency map. Note that for each filtered image/saliency map the values are normalized and re-scaled to lie in the range* $[0, 255]$

distance in CIELAB color space), which outputs the magnitude of the vector differences.

An appropriate selection of σ_1 and σ_2 will provide the right bandpass filter to retain the desired spatial frequencies from the original image for computing the saliency map. If the scale of the objects to be detected and the scale of those to be discarded from the input are known a priori, then the appropriate set of σ values can be chosen. This is often not the case. The two σ values, and therefore frequency parameters, are hence selected as follows. To implement a large ratio in standard deviations (so as to be able to detect the largest salient regions), we drive σ_1 to infinity (hence the name of our algorithm: Infinite Gaussian Saliency or IGS). This results in a notch in frequency at DC while retaining all other frequencies. To remove high frequency noise and textures (fourth criterion), we use a small Gaussian kernel keeping in mind the need for computational simplicity. For small kernels, the binomial filter approximates the Gaussian very well in the discrete case [33]. We use $\frac{1}{16}[1, 4, 6, 4, 1]$ giving a high-frequency cut-off value of $\pi/2.75$. We therefore retain more than twice as much high-frequency content from the original image as GB06 and at least 40% more than SR07. This choice of σ values also eliminates the need for setting or fine-tuning of any parameters (last criterion).

3.2.3 Intuitive understanding of the choice of parameters

Let us consider combining several narrow bandpass DoG filters. If we define $\sigma_1 = \rho\sigma$ and $\sigma_2 = \sigma$ so that $\rho = \sigma_1/\sigma_2$, we find that a summation over DoG with standard deviations in the ratio ρ results in

$$\begin{aligned}\sum_{n=0}^{N-1} G(x, y, \rho^{n+1}\sigma) - G(x, y, \rho^{n}\sigma) \\ = G(x, y, \sigma\rho^{N}) - G(x, y, \sigma)\end{aligned} \tag{3.3}$$

for an integer $N \geq 0$, which is simply the difference of two Gaussians (since all the terms except the first and last add up to zero), whose standard deviations can have any ratio $K = \rho^N$. If we assume that σ_1 and σ_2 are varied in such a way as to keep ρ constant at 1.6 (as needed for an ideal edge detector), then we essentially add up the output of several edge detectors (or selective band pass filters) at various scales. This corresponds to adding up the images along the diagonal of Fig. 3.1. This gives an intuitive understanding of why the salient regions will be fully covered and not just highlighted on object borders or as blobs in the center of the salient regions.

3.2.4 Computing saliency

Our method of finding the saliency map S for an image I of width W and height H pixels can thus be formulated as:

$$S(x, y) = \|\mathbf{I}_{\mu} - \mathbf{I}_{f}(x, y)\| \tag{3.4}$$

where $\mathbf{I}_{\mu}$ is the mean image feature vector, $\mathbf{I}_{f}(x, y)$ is the image pixel vector value in the Gaussian filtered version (using a 5×5 separable binomial kernel) of the original image (to eliminate fine texture details as well as noise and coding artifacts). We use a color and intensity feature vector by transforming the given sRGB image to the CIELAB color space. Each pixel location is an $[l\ a\ b]^T$ vector. This is computationally quite efficient (fifth criterion). In practice, we avoid the computationally expensive square root operation in computing the Euclidean distance as this does not affect the saliency detection significantly. Also, as we operate on the original image without any downsampling, we obtain a full resolution saliency map (sixth criterion). Thus, our method, summarized in Eq. 3.4 allows us to fulfill all of the requirements for salient region detection listed earlier in this chapter.

3.2.5 Various interpretations

We have already explained our model from the frequency domain perspective. In this section we explain our model from other perspectives to help relate other models to ours and to understand the relative advantages and disadvantages.

Global rarity

The rarer pixels of a certain color are, the more salient they are likely to be. Rarity is treated to be saliency by Mancas et al. [100] where it is computed as $I(m_i) = -log(p(m_i))$, where $p(m_i)$ is the normalized frequency of occurrence, i.e histogram bin size, of the *message* i (grayscale value), at a given pixel position i. In Eq. 3.4, the contribution to the global mean vector $\mathbf{I}_{\mu}$ from the rarer pixels is smaller. So the mean vector is further from the rarer pixels in terms of the color distance, making them more salient. In this sense, Eq. 3.4 captures global rarity.

Self-information

The expression $-log(p(m_i))$ is essentially the self information of pixel i. Self-information is the basis of several saliency detection techniques [24, 109, 126, 149] except that the method of evaluation and the features used differ. Our algorithm is therefore similar in philosophy to computing the self-information of a pixel with respect to the whole image.

Full-wave rectification

Our method can also be considered to be performing a full-wave rectification of a mean-subtracted signal. This is evident from Eq. 3.4 where we perform mean subtraction followed by the norm operation, which rectifies the signal.

3.3 Saliency detection algorithm - II

The saliency detection algorithm presented in Section 3.2 treats the entire image as the common surround (abstracted as the average image CIELAB color vector) for any given pixel. The premise is that there is no knowledge of the scale of the salient object and therefore it is best to pass all the low-frequency content. We base our new saliency detection algorithm on the premise that we can make assumptions about the scale of the salient object to be detected based on its position relative to the image borders. This can lead to smarter filtering bandwidth choices. This also helps overcome shortcomings of our previous algorithm.

3.3.1 The surround assumption of IGS

In Fig. 3.2 we observe that the larger the scale of the object is, the smaller has to be the low-frequency cut-off for detecting it (i.e., highlighting it fully in the saliency map), i.e larger the surround of the center-surround filter. Since we do not usually know the scale of an object beforehand, it is prudent to assume the worst case scenario and choose a very small low-frequency cut-off, i.e., a large surround. This allows detecting both large and small objects. This is why the method presented in Section 3.2 works well usually.

However, the method suffers from two drawbacks. In cases where the salient object is quite small, the surround is unusually large (whole image) and asymmetric, thereby potentially including a lot of noise. This affects the quality of detection. The second drawback is that if the salient object or region occupies more than half of the image pixels, or if the background is of a highly varied nature such that the largest contribution to the mean vector $\mathbf{I}_\mu$ in Eq. 3.4 comes from the salient object, the salient object becomes closer to the mean and is less salient. Instead, it is the background that appears more salient.

3.3.2 New surround assumption

To detect a large object fully, as in Fig. 3.2, we need a small value of the low-frequency cut-off, which is achieved using a larger surround for center-surround filter. Ideally, we would like to choose a surround just sufficient to fully highlight the object. The question we want to address is if it is possible to choose a compact surround sufficient to highlight the salient object without knowing its scale a priori.

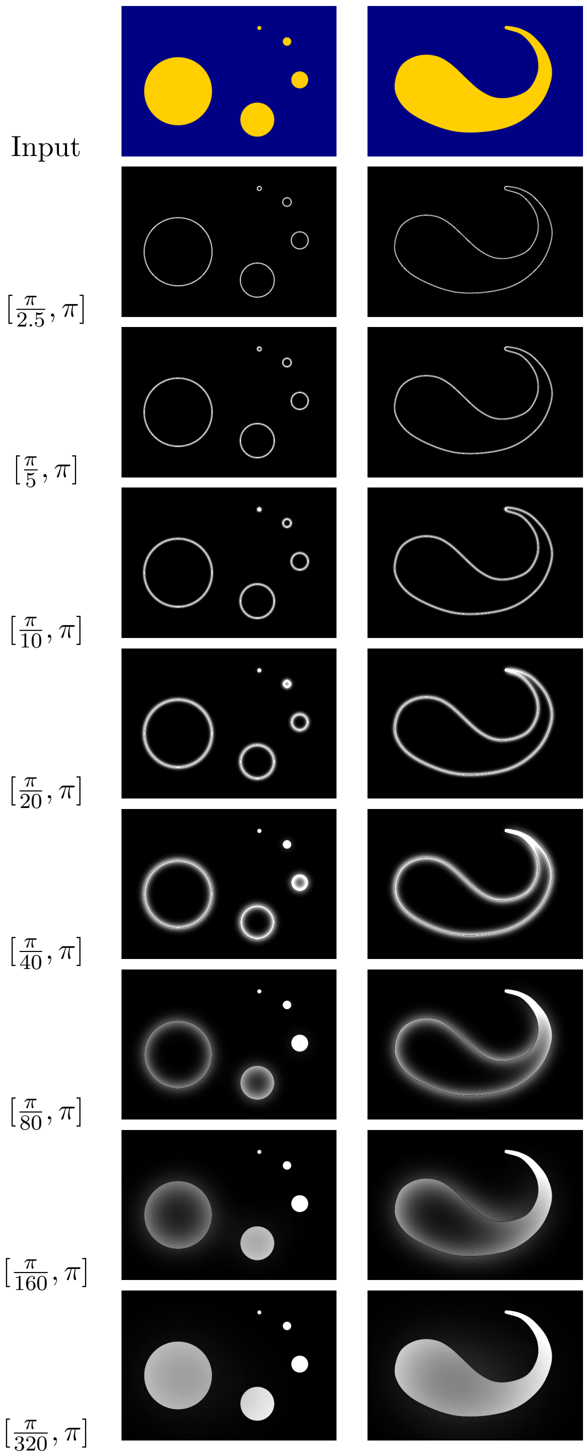


Figure 3.2: *Bandpass filtering of the input image with progressively increasing bandwidth from top to down (values in brackets show spatial frequency range). The first column of images is the same as the last row of Fig. 3.1. The high-frequency cut-off is kept the same while the low-frequency cut-off is reduced. We make three related observations here. One, the larger the scale, the smaller should be the low-frequency cut-off. Two, this also means that the more interior pixels of a salient object need a smaller low-frequency cut-off than the ones closer to edges of the object. Three, if we succeed in detecting a large object, the cut-off chosen usually also allows detecting smaller objects. These three observations form the basis of our improved method of saliency detection.*

Taking a cue from the image borders

Fig. 3.2 shows that a large value of low-frequency cut-off only lets us detect the pixels at the borders of a large object. As we lower the value of the low-frequency cut-off, we progressively detect more interior pixels of the object. If we know that we are performing center-surround filtering at the object border, we can use a large value of the low-frequency cut-off. Alternately, if we are performing filtering at the center of the large object, we need to use a small value of the low-frequency cut-off. We do not possess this knowledge a priori about the object size and location. However, we observe that if the pixel belonging to a salient object is close to the image borders, then it is likely to be close to the object borders (see Fig. 3.3). This means that we can use the position of the pixel relative to the image borders as a cue to limit the low-frequency cut-off.

In other words, we can vary the surround of the center-surround filter with respect to each pixel position. We choose to vary the surround symmetrically with respect to the center pixel position. To justify the use of a symmetric surround, let us imagine that at each pixel position in the image we are at the center of a large symmetric object (shown as a dotted blue ellipse) not touching the image borders (Fig. 3.4). We need to choose a low-frequency cut-off for the center-surround filtering that is enough to detect the innermost-pixel of this assumed large object. If we succeed in detecting this fictitious object then we can as well detect any object or part of object smaller than this.

To exploit the boundary based cue, we need to assume that salient objects are not touching image borders, i.e they are fully inside the image. This is a reasonable assumption as Fig. 3.5 shows. The figure is obtained by averaging the 1000 images of our ground truth (Section 3.5) where white indicates object and black indicates the background.

3.3.3 Saliency computation

With the new assumption about the surround we can compute saliency by applying a position-varying bandpass filter at each pixel. The filter bandwidth should vary in such a way that the low-frequency cut-off value of each filter at each pixel reduces progressively as we move towards the center pixel from the image borders.

Position variant difference of boxes filtering

We use box filters for performing bandpass filtering. It allows us to use integral images made popular by Viola and Jones [139] to perform the desired position-dependent center-surround filtering at each pixel. This is both computation and memory efficient, albeit with the tradeoff that the bandpass filtering is not ideal. Thus, for an input image of width W and height H, the symmetric surround saliency

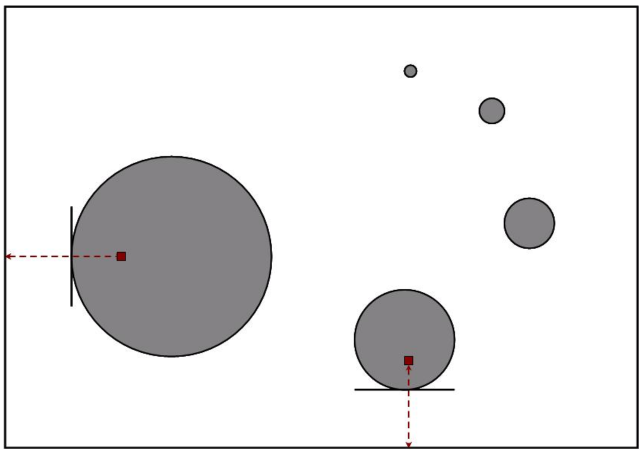


Figure 3.3: *If a pixel belonging to a salient object lies close to the image border, then it can not be far from the object borders (assuming the object is lying inside the image). Such a pixel does not need a very low value of the low-frequency cut-off, suggesting the the low-frequency cut-off for the center-surround filter can be varied according to the position of the pixel relative to the image borders.*

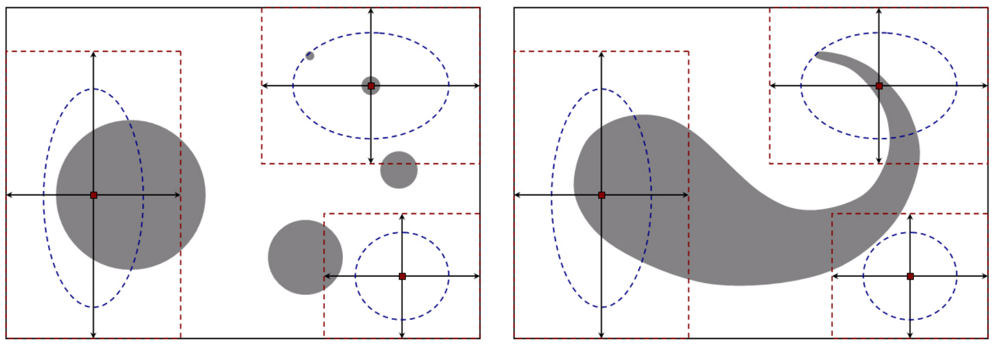


Figure 3.4: *Images explaining the premise that we can guess the scale of the salient object based on how far from the image borders we are when we are performing center-surround filtering. At each position in the image the low-frequency cut-off value i.e the surround of the center-surround filter should be such that we are able to detect the center pixel of a fictitious large elliptical object (in blue dots). If we can detect this then we can also detect any object smaller than the elliptical object.*

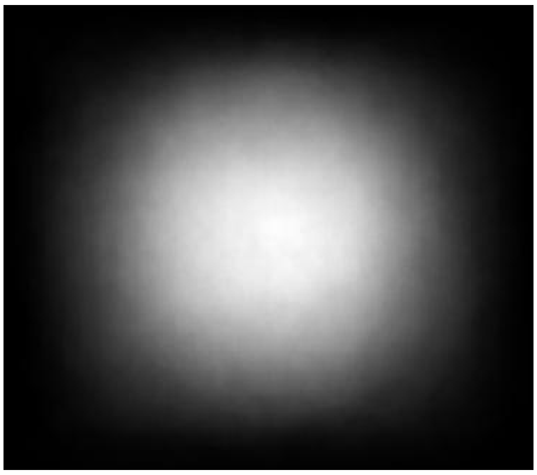


Figure 3.5: *Average of a thousand ground truth images in which white represents the salient object and black represents the non-salient background. Despite the fact that objects come in all sizes, shapes and locations, most of them lie away from the image borders, and have roughly half the extent of the image dimensions.*

value at a given pixel $S_{ss}(x, y)$ is obtained as:

$$S_{ss}(x, y) = \|\mathbf{I}_{\mu}(x, y) - \mathbf{I}_{f}(x, y)\| \tag{3.5}$$

where $\mathbf{I}_f$ is the Gaussian blurred image as in Eq. 3.4 and $\mathbf{I}_{\mu}(x, y)$ is the average CIELAB vector of the sub-image whose center pixel is at position (x, y) as given by:

$$\mathbf{I}_{\mu}(x, y) = \frac{1}{A} \sum_{i=x-x_o}^{x+x_o} \sum_{j=y-y_o}^{y+y_o} \mathbf{I}(i, j) \tag{3.6}$$

with offsets x_o, y_o, and area A of the sub-image computed as:

$$\begin{aligned} x_o &= \min(x, W - x) \\ y_o &= \min(y, H - y) \\ A &= (2x_o + 1)(2y_o + 1) \end{aligned} \tag{3.7}$$

The sub-image regions obtained in Eq. 3.6 using Eq. 3.7 are the maximum possible symmetric surround regions for a given pixel at the center. This is the reason we call our second saliency algorithm MSSS, for maximum symmetric surround saliency.

The closer a pixel is to the edges, the narrower will be its surround. Notice that in Fig. 3.4 we also show a pixel that falls on the background and the assumed extent of the plausible salient object as blue dotted ellipse. The filtering bandwidth we use here is suited for detecting an object with an extent shown by the blue ellipse and is not sufficient for detecting the background as salient. This is because the

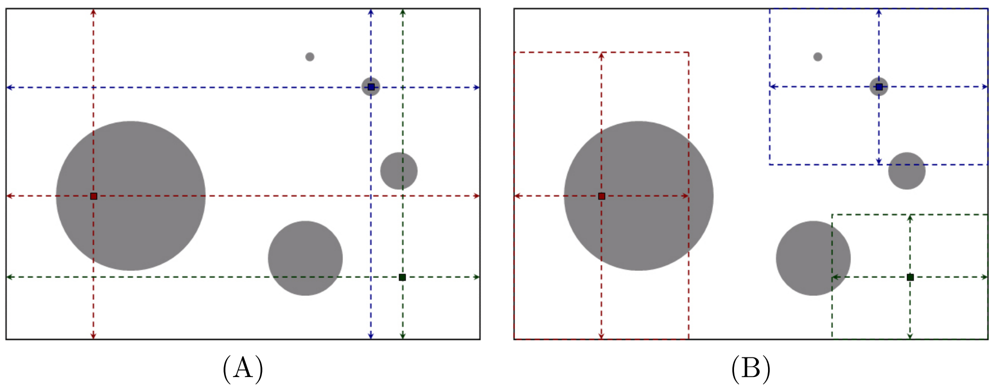


Figure 3.6: *(A) In the method of Eq. 3.4, for a pixel at the center (red) or elsewhere (blue), the surround region used for computing saliency remains the same, namely the whole image area. (B) Our improved algorithm uses surround regions (sub-images) as in Eq. 3.5 that are symmetric w.r.t the pixel whose saliency needs to be computed.*

background is much bigger than the largest assumed object size and is not within the image borders (it meets the borders).

Since the surround is usually more compact than the method of Section 3.2 it results in more local treatment and better detection (see Fig. 3.6). More importantly, the algorithm performs better in the case of varied backgrounds or when the salient object is large, and generally suppresses the background better.

Using other bandpass filters

We can also use other bandpass filters like DoG for computing such position-variant center-surround saliency. If we assume separable filters, then for any pixel position (x, y), the low-frequency cut-offs of the bandpass filters applied to a row and column should be roughly $\frac{\pi}{2x+1}$ and $\frac{\pi}{2y+1}$, respectively. If we use time-invariant filters, then we will have as many bandpass outputs as filters used. The final saliency map in this case will be the result of sampling from the appropriate bandpass outputs. Since only a few of the pixels are sampled from each bandpass filtered output, a lot of computation is redundant. In addition a lot more memory is required. This is avoided by the use of DoB filters.

3.4 Spatial frequency content of IGS and MSSS

From the frequency domain perspective we presented in Section 2.2, both for IGS and MSSS the spatial frequency content is in the range $(0, \pi/2.75]$, though in the latter case the low-frequency cut-off is not a constant. A comparison of bandpass filtered images with cut-off frequencies given in Table 2.1 with those of IGS and MSSS is shown in Fig. 3.7. Examples of our saliency maps using our algorithm IGS

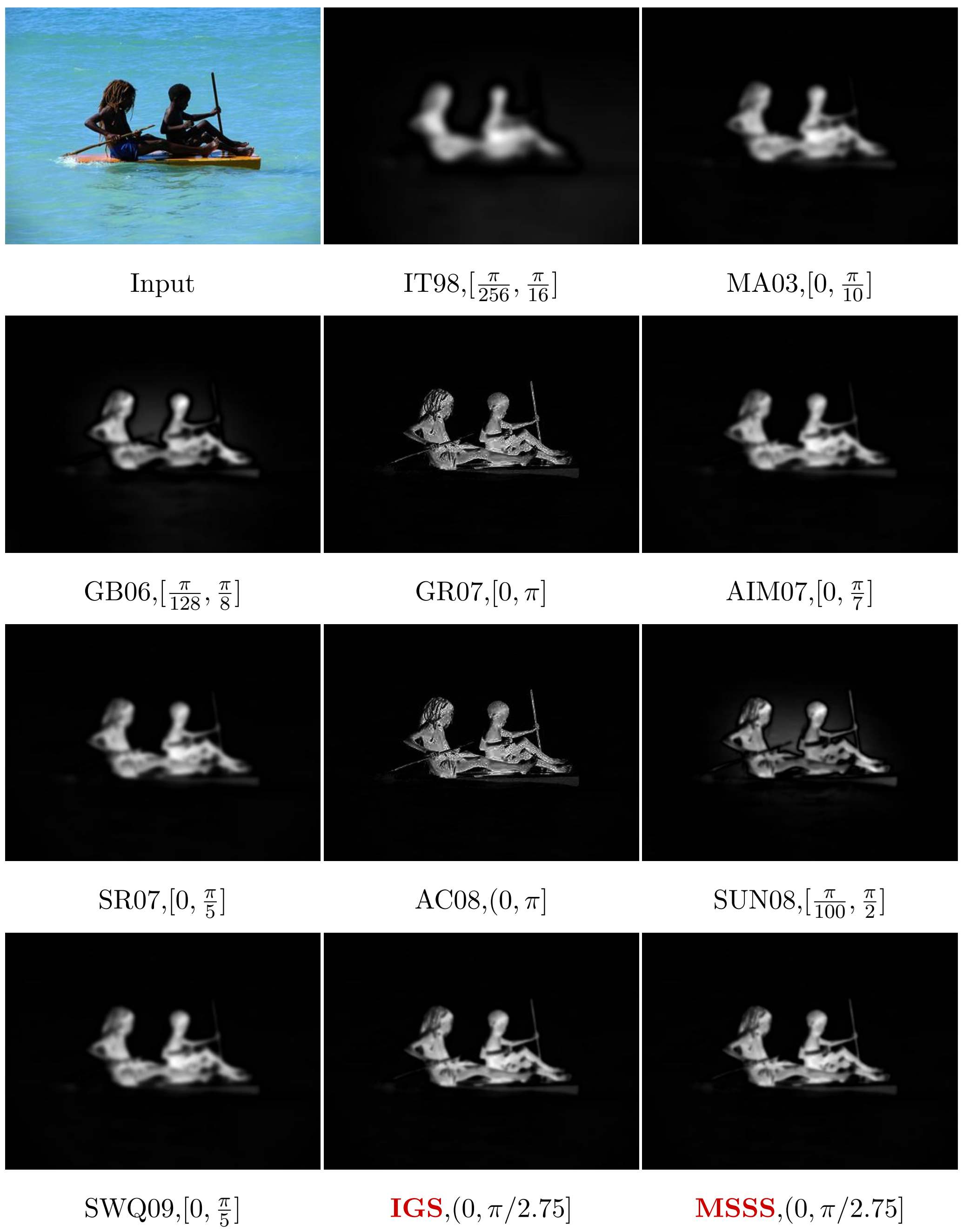

Input
IT98,$[\frac{\pi}{256}, \frac{\pi}{16}]$
MA03,$[0, \frac{\pi}{10}]$
GB06,$[\frac{\pi}{128}, \frac{\pi}{8}]$
GR07,$[0, \pi]$
AIM07,$[0, \frac{\pi}{7}]$
SR07,$[0, \frac{\pi}{5}]$
AC08,$(0, \pi]$
SUN08,$[\frac{\pi}{100}, \frac{\pi}{2}]$
SWQ09,$[0, \frac{\pi}{5}]$
IGS,$(0, \pi/2.75]$
MSSS,$(0, \pi/2.75]$

Figure 3.7: *Original image bandpass filtered with cut-off frequencies given in Table 2.1. The spatial frequency content retained after performing linear operations explains to some extent the appearance of the saliency maps shown in Fig 3.8 to Fig. 3.12.*

and the improved algorithm MSSS are shown in Fig. 3.8 to Fig. 3.12. An objective comparison of the saliency maps using our methods and several state-of-the art algorithms is presented in the next section.

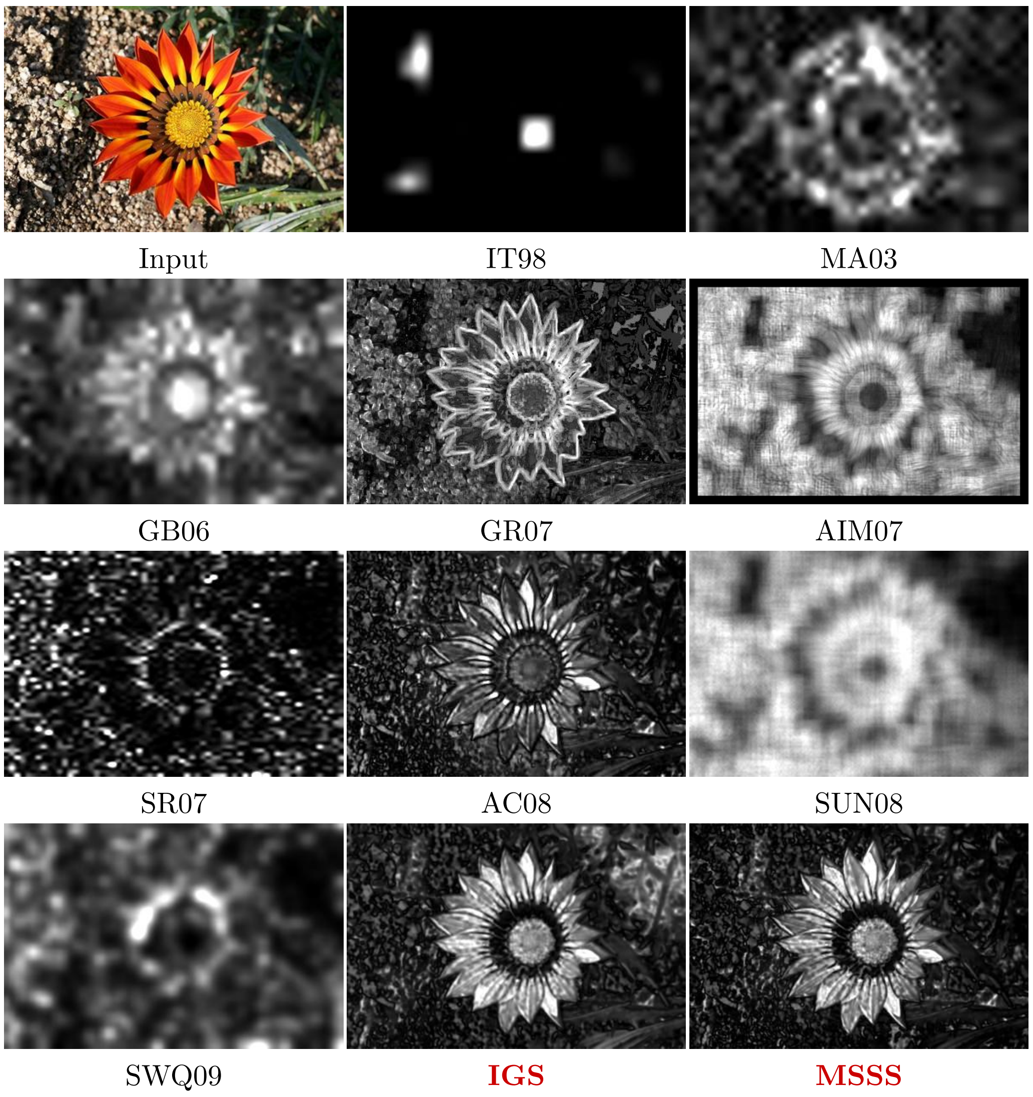


Figure 3.8: *Visual comparison of saliency maps.*

3.5 Comparison with state-of-the art

To compare the quality of saliency maps for the task of segmenting salient objects we rely on a ground truth database. We derived the database from the publicly available database used by Liu et al. [93]. This database provides bounding boxes drawn around salient regions by nine users. However, a bounding box-based ground

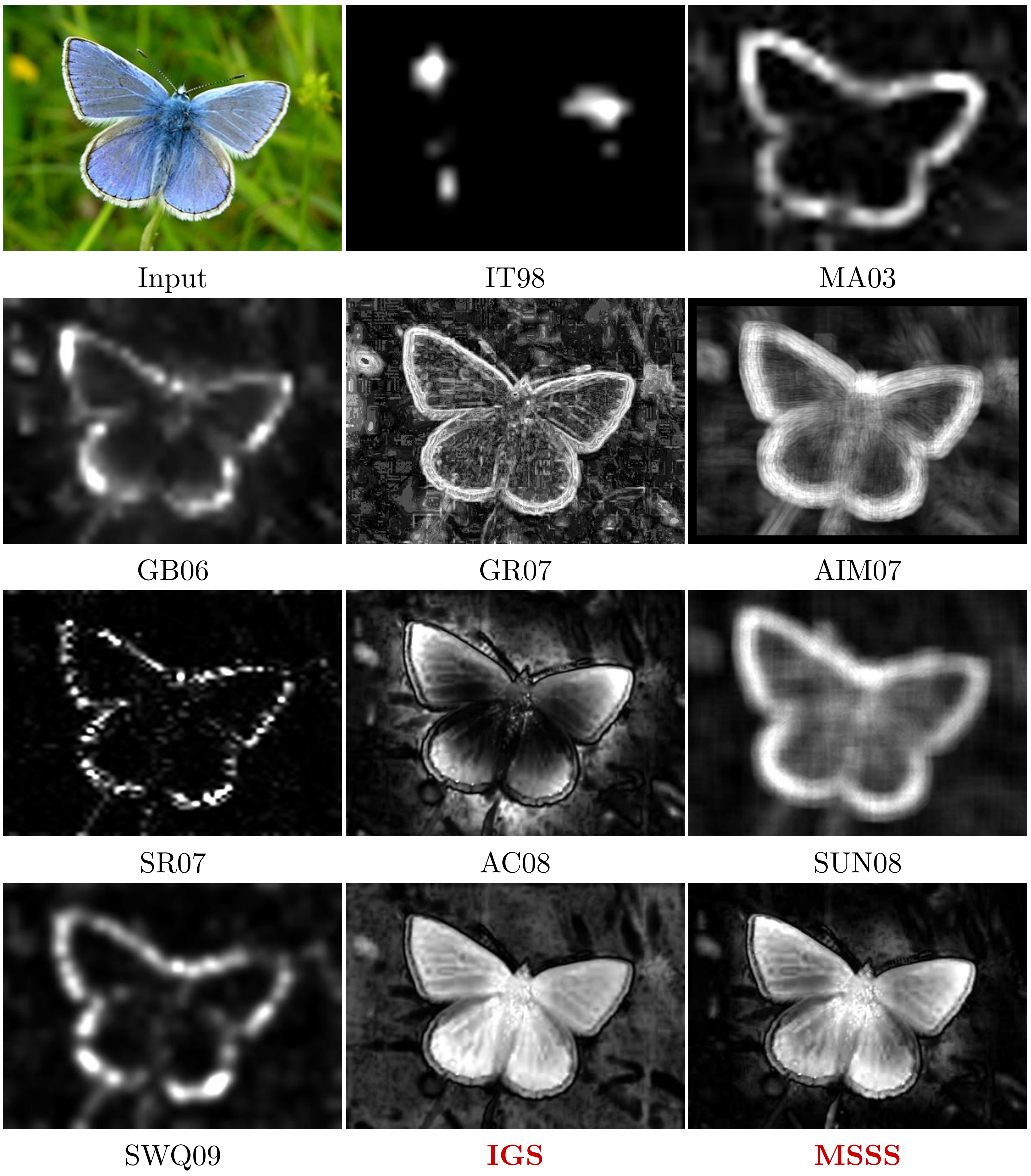


Figure 3.9: *Visual comparison of saliency maps.*

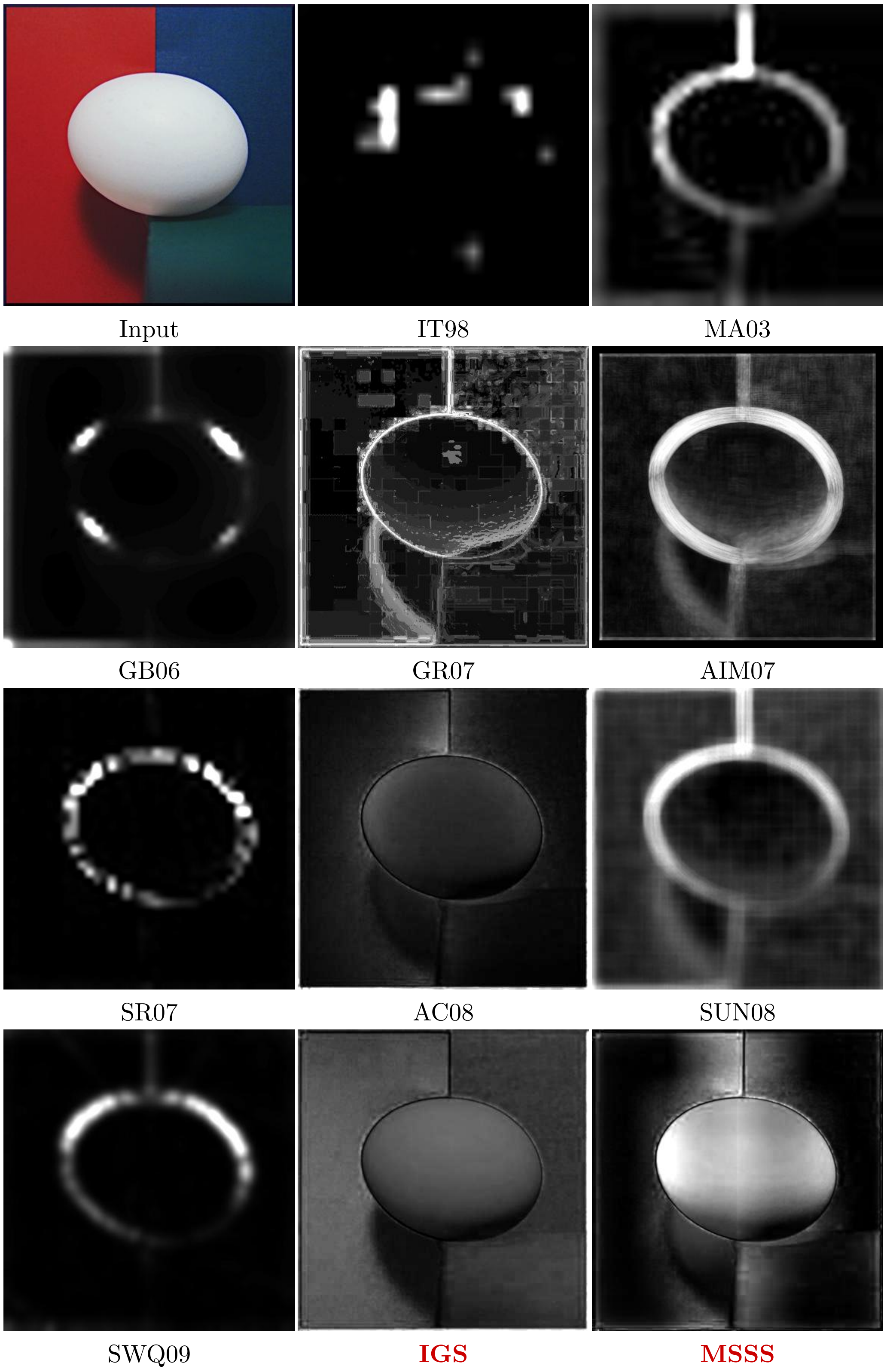


Figure 3.10: *Visual comparison of saliency maps.*

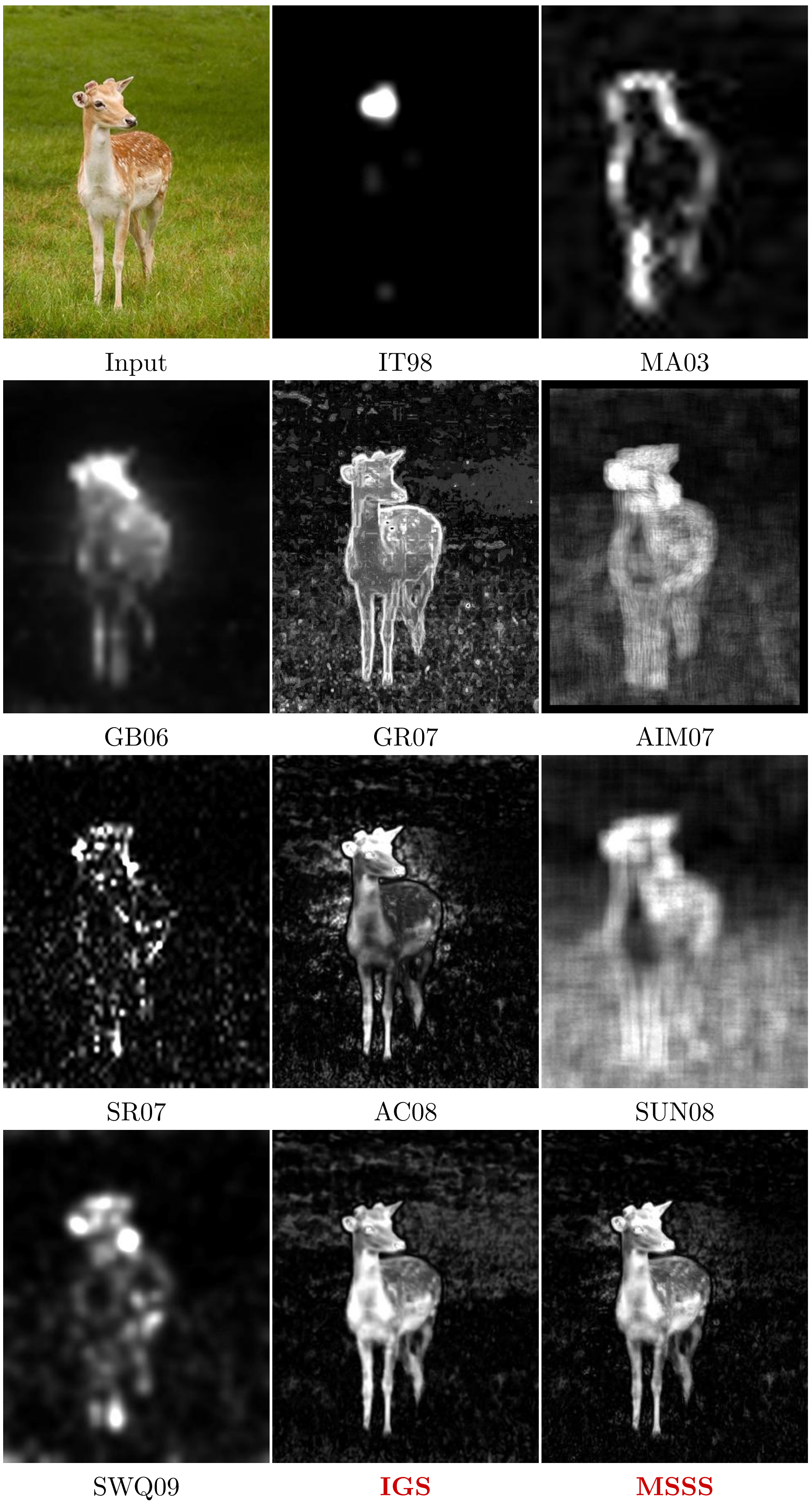


Figure 3.11: *Visual comparison of saliency maps.*

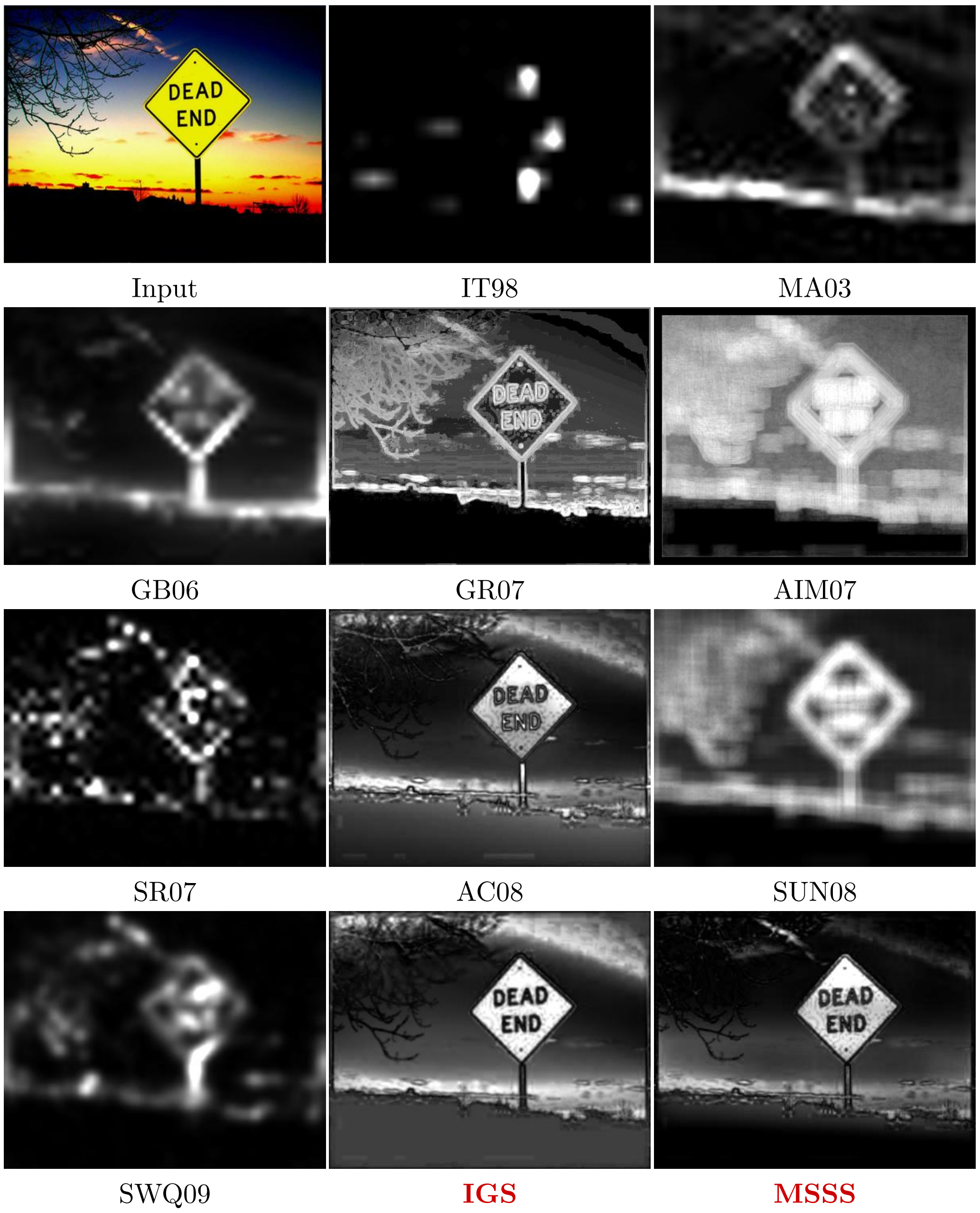


Figure 3.12: *Visual comparison of saliency maps. Our methods MSSS and IGS produce saliency maps that have well-defined borders, highlight whole object regions, and suppress the background better than most methods. MSSS achieves better background suppression than IGS even in the presence of complex backgrounds or when the salient object is large.*

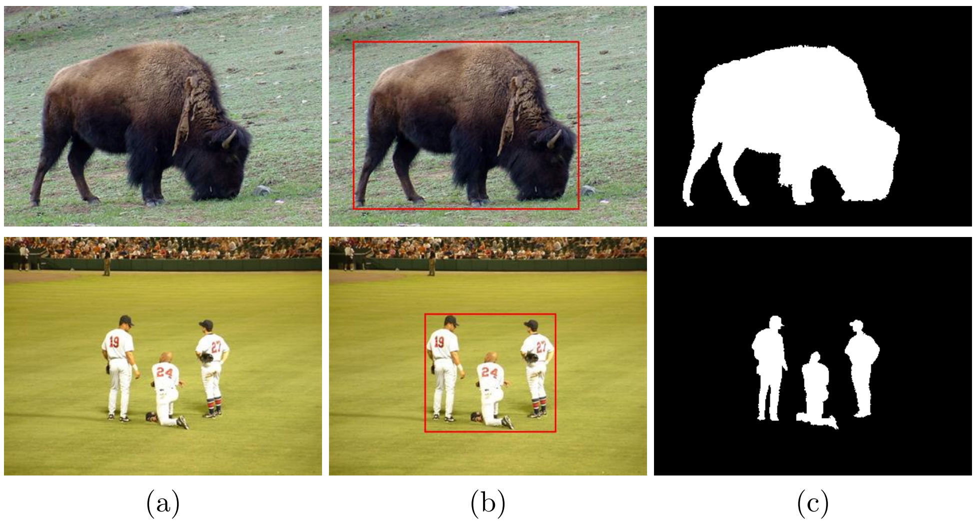


Figure 3.13: *Ground truth examples. (a) Original image. (b) Bounding box based ground truth by Wang and Li [142]. (c) Our ground truth is more accurate since it is object contour based and it treats multiple objects separately.*

truth is far from accurate, as also stated by Wang and Li [142]. Thus, we created an accurate object-contour based ground truth database[2] of 1000 images (examples in Fig. 3.13). We compute precision and recall measures (Section 3.5.1) of the saliency maps with respect to our ground truth images.

3.5.1 Precision and recall

Precision and recall are two widely used statistical measures. The former is a measure of accuracy while the latter is a measure of completeness. In a statistical classification task, the precision for a class is the number of true positives, i.e. the number of elements correctly identified as belonging to the positive class, divided by the total number of elements identified as belonging to the positive class. Recall is computed as the number of true positives divided by the total number of elements that actually belong to the positive class. The opposite of true positives is true negatives, which are the items correctly identified as belonging to the negative class. On the same lines, false positives are the items wrongly identified as positive while false negatives are items wrongly identified as negative.

$$Precision = \frac{tp}{tp + fp} \tag{3.8}$$

$$Recall = \frac{tp}{tp + fn} \tag{3.9}$$

[2]http://ivrg.epfl.ch/supplementary_material/RK_CVPR09/index.html

where tp is true positives, fp is false positives, and fn is false negatives. Fig. 3.14 illustrates how tp, fp, and fn are found.

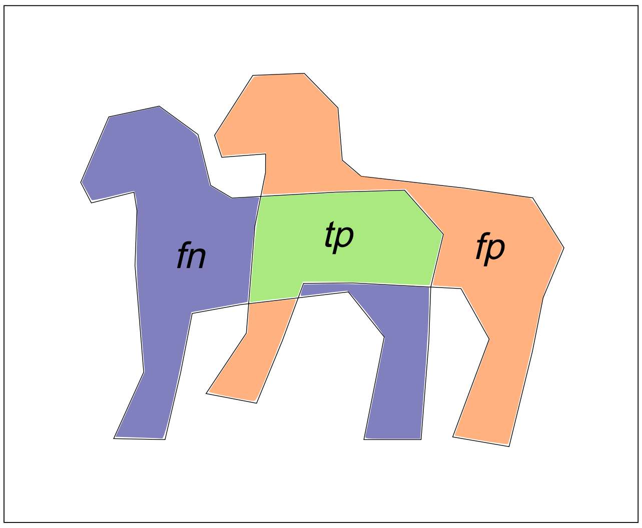


Figure 3.14: *In this image, the blue pattern represents ground truth while the orange one represents an attempt to match the true shape. The degree to which this match is done well can be quantified in terms of the precision and recall measures of Eq. 3.9. These measures are computed using the values of true positives (tp) in green, where the match takes place correctly, false positives (fp) in orange, and false negatives (fn), in blue.*

3.5.2 Comparison by thresholding

Our aim is to obtain a quantitative measure of how well a saliency map detects or highlights a known salient object. We obtain this by performing naïve thresholding of a given saliency map at several thresholds and comparing each output with the ground truth.

For a given saliency map, with saliency values normalized in the range $[0, 255]$, we obtain a binary mask at all 256 integral thresholds T_f in the interval $[0, 255]$, and compute the precision and recall at each value of the threshold with respect to our ground truth. We repeat this for all the 1000 images of the ground truth database and average the values of precision and recall giving us 256 pairs of precision-recall values. We plot a precision versus recall curve to obtain Fig. 3.15. This curve provides a reliable comparison of how well various saliency maps highlight salient regions in images. A higher curve indicates a better capability of segmenting a saliency map by simple thresholding.

It is interesting to note that Itti's method shows high accuracy for a very low recall and then the accuracy drops steeply. This is because the salient pixels from this method fall well within salient regions, and have near uniform values, but do

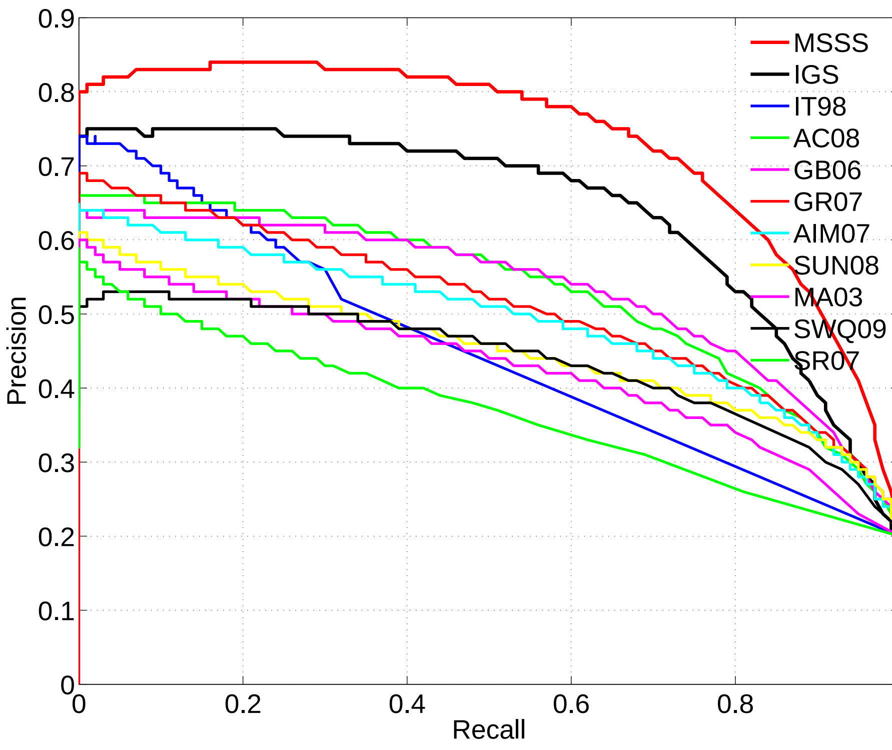


Figure 3.15: *Precision-recall curve for naïve thresholding of saliency maps. Our methods IGS and MSSS are compared against the nine methods of IT98 [72], MA03 [98], GB06 [62], GR07 [100], AIM07 [25], SR07 [67], AC08 [4], SUN08 [149], and SWQ09 [21] on 1000 images with segmentation ground truth. The legend is ordered from top to down in decreasing order of quality. IGS and MSSS significantly outperform all the other state-of-the-art algorithms. Among our methods, MSSS outperforms IGS.*

not cover the entire salient object. Methods GB06 and AC08 have similar performance despite the fact that the latter generates full resolution maps as output. At maximum recall, all methods have the same low precision value. This happens at threshold zero, where all pixels from the saliency maps of each method are retained as positives, leading to an equal value for true and false positives for all methods. The curve for MSSS curiously rises in precision before falling (it also happens for SWQ09 and slightly for IGS). This is attributed to relatively greater number of false positives at high thresholds (around $T_f = 245$) as compared to lower thresholds. As T_f gets lower, the number of true positives rises in relation to the false positives leading to a small rise in the curve. As T_f lowers even further, the curve descends like the other curves.

3.6 Discussion

While our methods IGS and MSSS are simple and efficient, there are instances of images where they fail to highlight the salient objects well. We show one such example (from the database of 1000 images) in Fig. 3.16. In images where there is poor contrast between the salient object and the surround, our methods perform poorly. Poor contrast may be the consequence of uneven lighting conditions, the object of interest having similar colors as the background, or the background being quite cluttered with several regions of different colors. In addition, our methods, particularly IGS, take only global contrast into account so objects/regions that are locally salient with respect to their immediate neighborhood, but not globally salient may not be well-highlighted. The goal is to not miss large salient objects since larger salient objects are assumed to be more significant than smaller ones.

MSSS improves on IGS in terms of background suppression and dealing with large salient objects (Fig. 3.17). However, it shows filtering artifacts, because of the use of box-filters, which are not ideal bandpass filters. As we will see in the next chapter, for applications like object segmentation this is not a problem.

Notably, our algorithms use only color and intensity information while the human visual system relies on several contrasts including orientation, texture, shape and symmetry for spotting objects in a bottom-up fashion. In our experiments we tried using the features of orientation and texture but found it difficult to choose the right scales and combining these features with the color and intensity features. Despite using only color and intensity our algorithms perform better than most other algorithms that take into account other features [72, 25, 149].

3.7 Summary of the chapter

In this chapter we presented two saliency detection algorithms. The first algorithm uses the whole image to compute the surround. This is based on the observation that when the scale of the salient object is not known a priori all the low-frequency content should be retained. The second algorithm uses a variable surround based on the observation that near the image borders we do not need a large surround because we can limit the low-frequency content required for good saliency detection. The saliency maps of both our algorithms prove to be superior to those of several state-of-the-art algorithms when compared in terms of precision and recall measures with reference to a publicly available ground truth database. In addition, they are computationally inexpensive and do not require any input parameters.

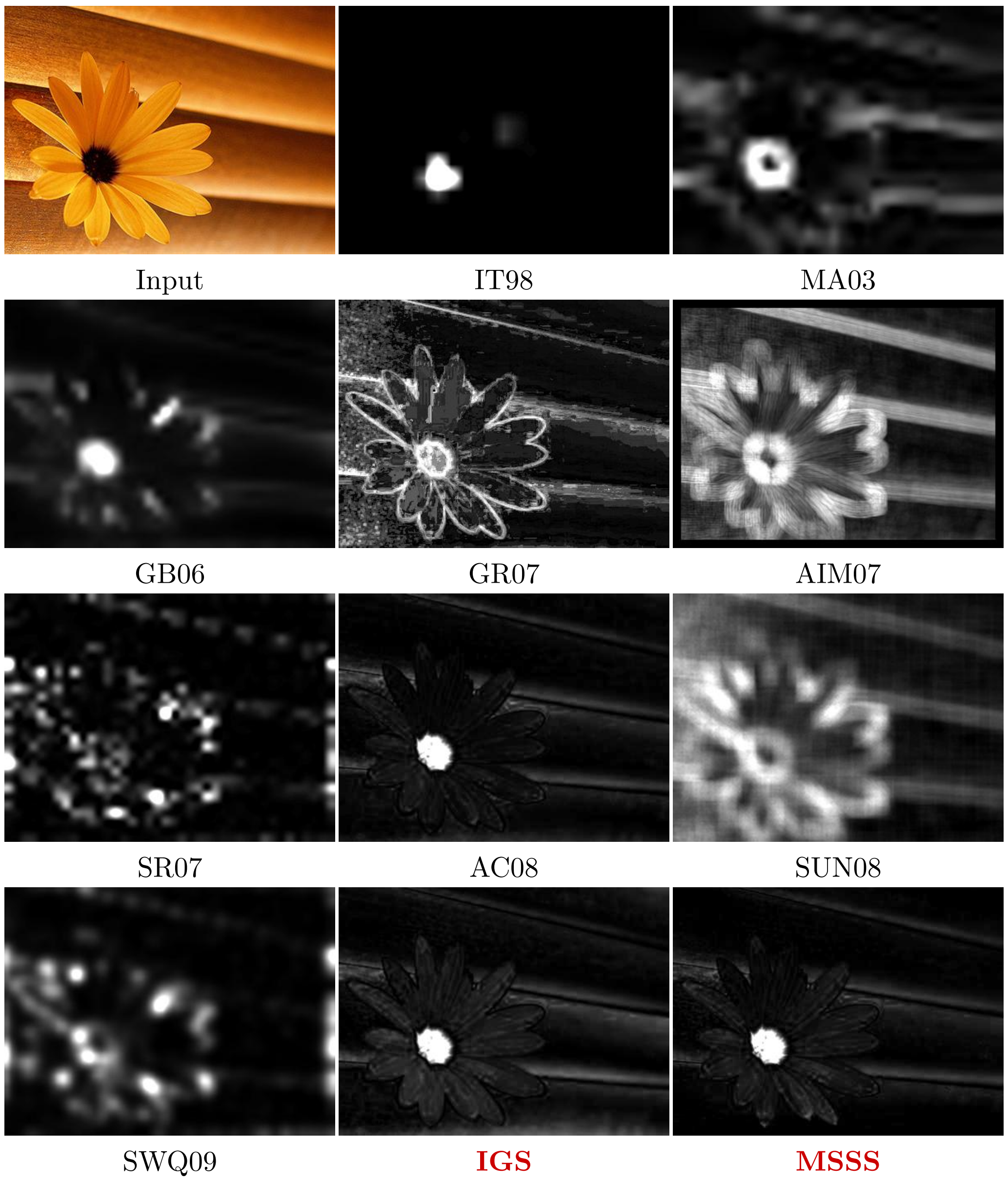


Figure 3.16: *Visual comparison of saliency maps where our methods IGS and MSSS perform poorly. In this example the use of orientation information could have helped detecting the entire flower as salient. However, there are other schemes, notably IT98 and GB06, which use orientation information, but their method does not prove general enough to detect the salient object in this case.*

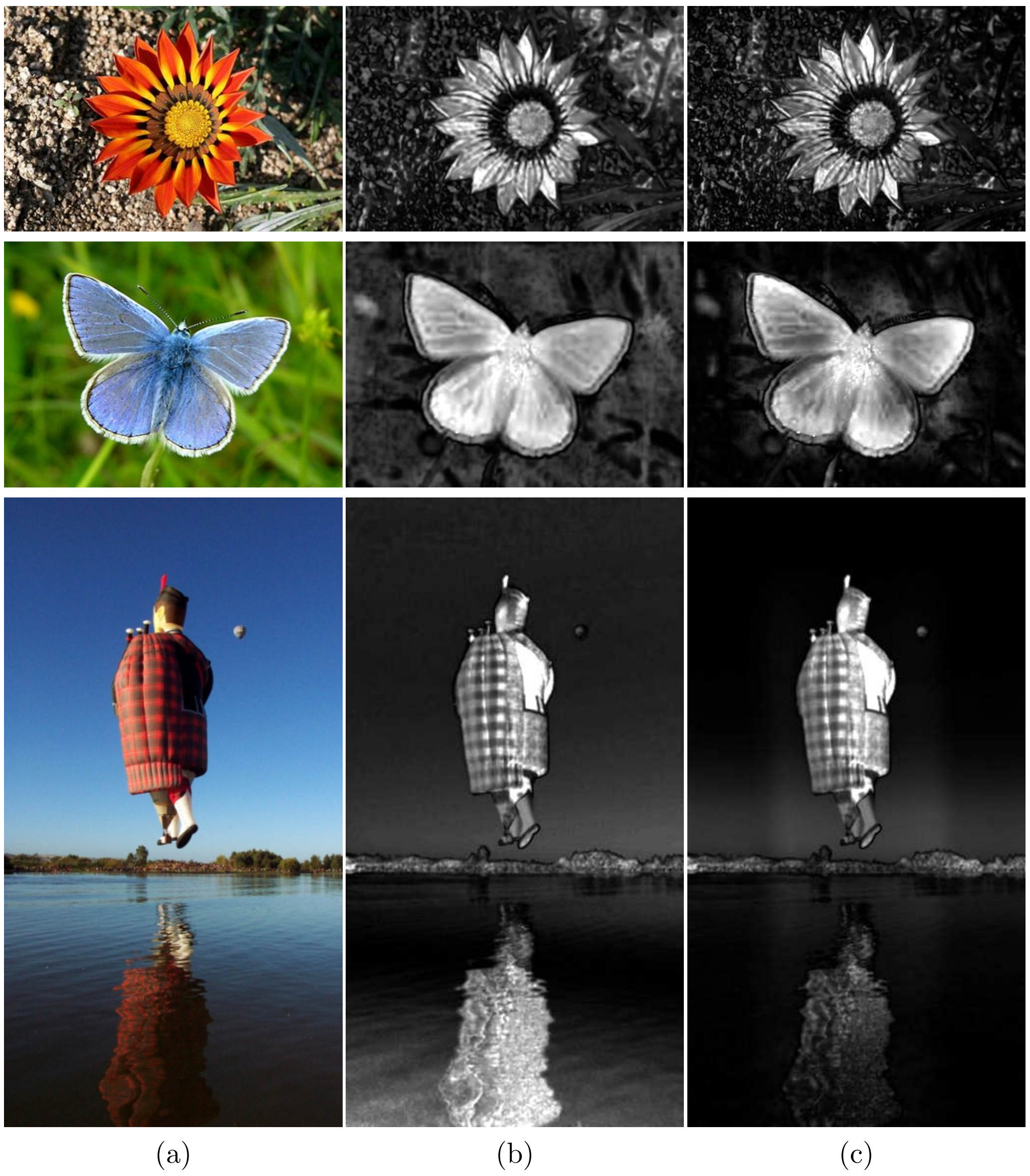


Figure 3.17: *Comparison of saliency output of IGS and MSSS. (a) Input image. (b) Saliency maps of IGS. (c) Saliency maps using MSSS. Usually, MSSS suppresses the background better than IGS. Notice how the reflection of the balloon in the water is suppressed well by MSSS because it touches the image border. But sometimes this advantage comes with the price of uneven highlighting or a mild halo around the objects as in the case of the butterfly and the balloon.*

Chapter 4

Applications of Saliency Detection

In this chapter we present techniques for two applications of saliency detection: salient region segmentation and content-aware image re-targeting[1]. These applications also serve as a way of comparing our saliency detection techniques presented in Chapter 3 with those of the state-of-the-art. For the former application, we present three different techniques. For each, we compare the quality of segmentation using our saliency maps with those of other saliency detection techniques, using measures of precision, recall, and F_β-score (Eq. 4.2). For the latter task of content-aware image resizing, we focus on the technique of *seam carving* [16]. We show by comparison with state-of-the-art techniques of image re-targeting that our saliency maps perform better than currently used energy maps.

4.1 Salient object segmentation

The maps generated by saliency detectors can be employed in salient object segmentation using more sophisticated methods than simple thresholding. Saliency maps produced by Itti's approach [72] have been used in unsupervised object segmentation. Han et al. [61] use a Markov random field to integrate the seed values from Itti's saliency map along with low-level features of color, texture, and edges to grow the salient object regions. Ko and Nam [76] utilize a Support Vector Machine trained on image segment features to select the salient regions of interest using Itti's maps, which are then clustered to extract the salient objects. Ma and Zhang [98] use fuzzy growing on their saliency maps to confine salient regions within a rectangular region. We present three methods of segmenting salient objects using saliency maps. The first one uses saliency-adaptive thresholding on pre-segmented images. The second method relies on graph-cuts to segment regions. The third method uses geodesic distances for this purpose.

[1]Most of the content of this chapter is also available in the references [5] and [8]

4.1.1 Segmentation by adaptive thresholding

We use a method for segmenting salient objects that is a modified version of the one we published earlier [4]. In this technique we over-segment the input image using k-means clustering and retain only those segments whose average saliency is greater than a constant threshold. This threshold is set at 10% of the maximum normalized saliency value. The binary maps representing the salient object are thus obtained by assigning ones to pixels of chosen segments and zeroes to the rest of the pixels.

We present here two improvements to this method. First, we replace the hill-climbing based k-means segmentation algorithm by the mean shift segmentation algorithm [29], presented in Section 2.5.2, which provides better segmentation boundaries. We perform the segmentation in CIELAB color space (instead of CIELUV space as done by the authors) on the input image. We use fixed parameters of 7, 10, 20 for $sigmaS$, $sigmaR$, and $minRegion$, respectively, for all the images (see [29]). Second, we introduce an adaptive threshold that is image saliency dependent, instead of using a constant threshold for each image. This is similar to the adaptive threshold proposed by Hou and Zhang [67] to detect proto-objects [140]. The adaptive threshold is computed as $T_a = 2 \times S_{avg}$ where the average saliency value S_{avg} is obtained as:

$$S_{avg} = \frac{1}{W \times H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} S(x,y) \tag{4.1}$$

Here W and H are the width and height of the saliency map in pixels, respectively, and $S(x,y)$ is the saliency value of the pixel at position (x,y). A few results of salient object segmentation using our improvements are shown in Fig. 4.1. We choose all those segments of the image whose average saliency exceeds this threshold. In addition, we ignore all those segments that touch any of the image borders since salient objects usually do not touch image borders (see Fig. 3.5). We obtain binarized maps of salient object from each of the saliency algorithms where all pixels of chosen segments are white (ones) and all others are black (zeroes). This process is adopted for saliency maps of all the methods we compared in Chapter 3. Once we have the binarized maps we can compute precision and recall scores (Eq. 3.9) with respect to a ground truth.

Generally, the precision is high if the saliency map correctly identifies the salient object pixels and assigns low values to non salient regions and recall is high when the whole object is highlighted.

But high precision may occur even if the object boundaries are not well defined. Similarly, high recall may occur when a portion of the saliency map larger than the salient object is highlighted without respecting its boundaries. Precision and recall considered separately may lead to incorrect conclusions about the performance of an algorithm. So, it is useful to have a combined score of the precision and recall measures to judge the performance of an algorithm. This is computed in the form

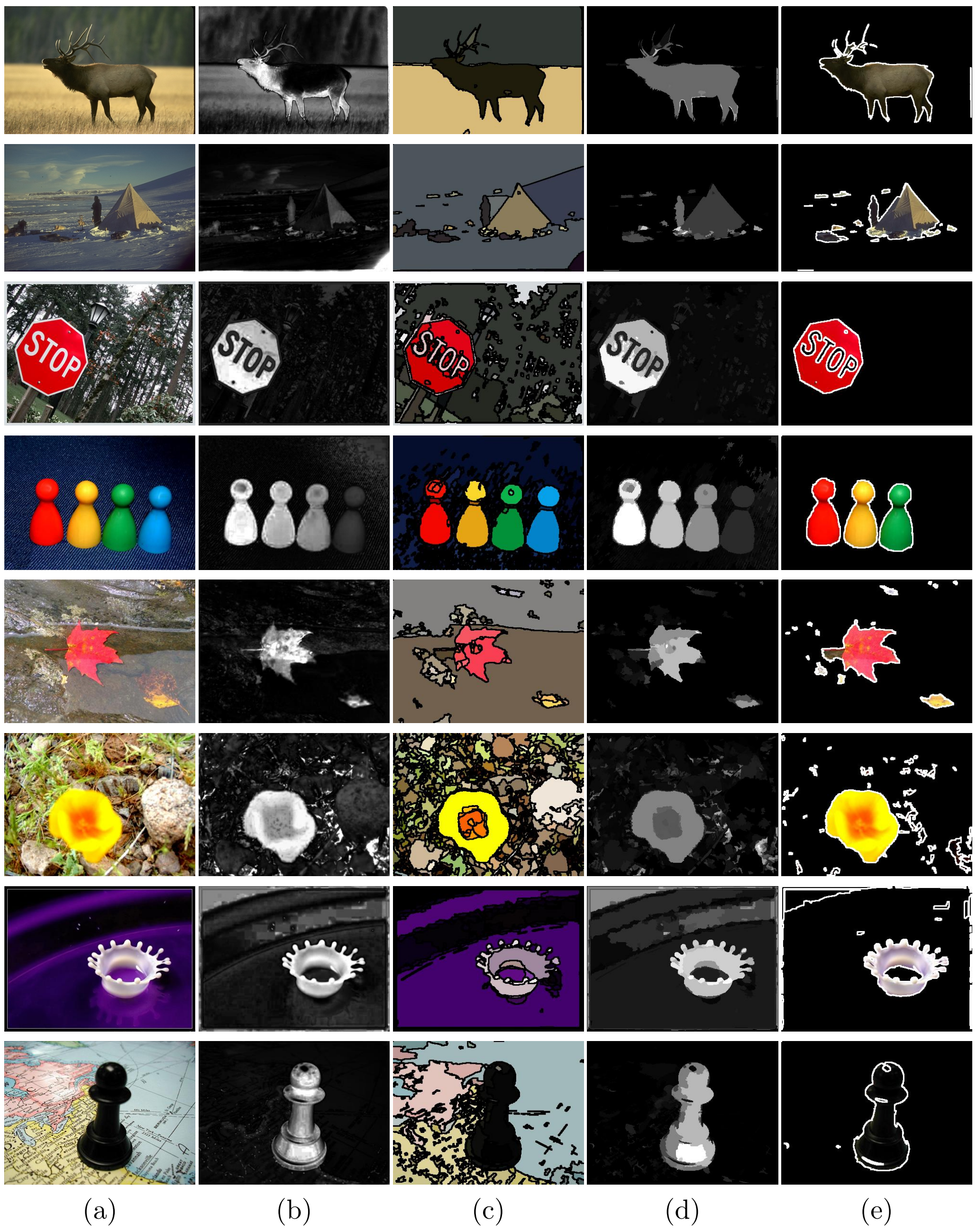


Figure 4.1: *Salient object segmentation using saliency-adaptive thresholding. (a) Original image. (b) Saliency map using MSSS (Section 3.3). (c) Mean-shift segmented image. (d) Average saliency per segment assigned to each pixel of the segment. (e) Those segments that have a saliency value greater than the adaptive threshold T_a are chosen while others discarded. If the saliency map fails to highlight an object well, it is not segmented successfully (fourth row). Similarly, if the saliency map highlights objects in the background, they may be get segmented (sixth row).*

of F_β-score as follows.

$$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{4.2}$$

where β is usually chosen to be 0.5, 2.0, or 1.0 to give more weight to precision, recall, or give equal weight to precision and recall, respectively. We use $\beta = 0.5$ consistently as we weigh precision higher than recall in our segmentation applications. The average values of precision, recall, and F_β-score (Eq. 4.2) are obtained over the same ground-truth database used for evaluation of saliency maps in Section 3.5. A plot of these values is shown in Fig. 4.2.

Itti's method (IT98 [72]) shows a high precision but very poor recall, indicating that it is better suited for gaze-tracking experiments, but perhaps not well suited for salient object segmentation. Among all the methods, our techniques (IGS and MSSS) usually show the highest precision, recall, and F_β-score.

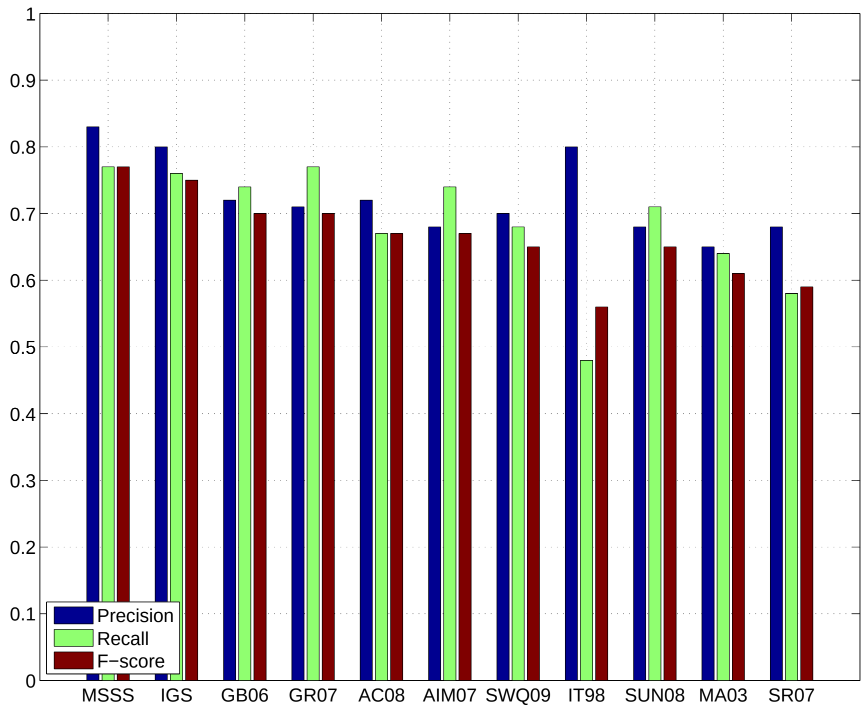


Figure 4.2: *Precision-Recall bars for adaptive thresholding based salient object segmentation in decreasing order of F_β-score ($\beta = 0.5$). Our methods IGS and MSSS shows high precision, recall, and F_β values computed on the 1000 image database.*

4.1.2 Salient region segmentation using graph-cuts

We use a graph-cuts based approach for salient object segmentation from saliency maps. Graph-cuts based methods are popular for image segmentation applications. Boykov and Jolly [23] perform interactive segmentation where a graph-cuts algorithm [22] segments foreground from background. As input, their method requires a user to draw scribbles to indicate foreground and background regions. Similarly, in GrabCut [118], the user draws a rectangle around the object of interest in an image. The background is defined by the pixels outside the rectangle. The foreground is then segmented using a graph-cuts based optimization that iteratively assigns pixels to background or foreground based on their proximity to the respective multi-modal color distributions. In our scheme of segmenting objects using graph-cuts, instead of relying on user input, we use the saliency map to assign these pixels automatically.

We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ over the pixels of the image. Each node in $\mathcal{V}$ corresponds to a pixel p_i. Edges $\mathcal{E}$ connect neighboring pixels (using 8-connectivity). We seek an optimal cut between pixels belonging to salient and non-salient regions by minimizing the objective function of the form given by Eq. 2.9. The *unary* term ψ assigns to each pixel its potential to belong to the foreground object or background. The unary potential terms are assigned by Boykov and Jolly [23] using intensity histograms created separately for background and foreground pixels obtained as scribbles from the user. In the case of *GrabCuts*, Rother et al. [118] assign these values using a Gaussian Mixture Model (GMM) each for the background and foreground pixels.

The *pairwise* term ϕ promotes coherence among similar pixel neighbors. It penalizes the assignment of different labels to neighboring pixels of similar color. It is computed as:

$$\phi(c_i, c_j | p_i, p_j) = \begin{cases} \exp\left(-\frac{\|\mathbf{I}(p_i)-\mathbf{I}(p_j)\|^2}{2\sigma^2}\right) & \text{if } c_i \neq c_j \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

where $\mathbf{I}$ is the CIELAB color vector of a pixel and σ is assigned the value of the average value of all the edge weights as done in several graph-cuts applications [23, 118], i.e.

$$\sigma = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}, i\neq j} \|\mathbf{I}(p_i) - \mathbf{I}(p_j)\| \tag{4.4}$$

In our graph-cuts setup we link the source to the salient object and the sink to non-salient regions. As our initialization step, we assign a *hard link*, i.e a value greater than the largest binary edge weight computed in Eq. 4.3, to the source and sink. For this we use two thresholds that depend on the average saliency value S_{avg} computed as in Eq. 4.1. For all pixels whose saliency value is greater than the threshold $T_s = 3 \times S_{avg}$ we assign hard links to the source and to all pixels whose saliency value is less than the threshold $T_t = \frac{1}{3} S_{avg}$ we assign hard links to the sink.

These two thresholds were chosen experimentally to ensure that the hard links are assigned only to pixels that have high probability of belonging to the salient object or background.

In our case, the unary term assignment (using histograms or clustering) has minimal effect on the segmentation quality. This is because unlike user-drawn scribbles, using our thresholding we assign a lot more pixels to the source and sink with hard links. So, in our segmentation scheme we ignore the unary term. A few results of salient object segmentation using our method are shown in Fig. 4.3.

We perform the graph-cuts based segmentation for the saliency maps of all methods on our database and compare the average values of precision, recall, and F_β-score (Eq. 4.2) in Fig. 4.4. Our method MSSS has an advantage over other saliency detection techniques for such an application since it achieves both high precision and high recall in detecting salient objects.

4.1.3 Salient region segmentation using geodesic paths

Geodesic paths have been used for segmentation and matting based on user-drawn scribbles by Bai and Sapiro [19]. They use the fast marching algorithm proposed by Yatziv et al. [147] for this. We use a geodesic paths approach to segment salient object from images. However, unlike Bai and Sapiro, we rely on a thresholded saliency map instead of user-drawn scribbles as input.

As in the previous section, we set saliency-dependent thresholds for labeling some of the pixels. If the saliency value is greater than $3 \times S_{avg}$ it is labeled as a salient object pixel, if the saliency value is less than $\frac{1}{3}S_{avg}$ it is considered a background pixel. Each of these labeled pixels acts as a *seed* from which we reach out to other pixels along the shortest geodesic paths. Each pixel p at given position in the xy-plane of the image is assigned the *label* of its nearest seed if the geodesic path to it from the seed is shorter than that from any other seed.

Geodesic path computation is presented in Section 2.5.5. In the computation of the cost $d(P)$ of the path P, $\|\mathbf{I}(p_i) - \mathbf{I}(p_{i-1})\|$ represents the Euclidean distance between the CIELAB color vectors of pixels p_i and p_{i-1}. We present examples of segmentation using geodesic distances in Fig.4.5. We compare the segmentation output using our saliency maps with that using others with respect to our ground truth database in Fig. 4.6.

4.1.4 Discussion

The simplest of the three salient object segmentation methods is the adaptive thresholding method. It usually provides the highest F-scores for all methods, even when the saliency map is of poor quality. The graph-cuts and geodesic based methods are less forgiving in this respect - when the quality of the saliency map is poor, it results in poor segmentation. The graph-cuts based approach and the geodesic path based approach usually provide similar segmentations. The graph-cuts approach though is

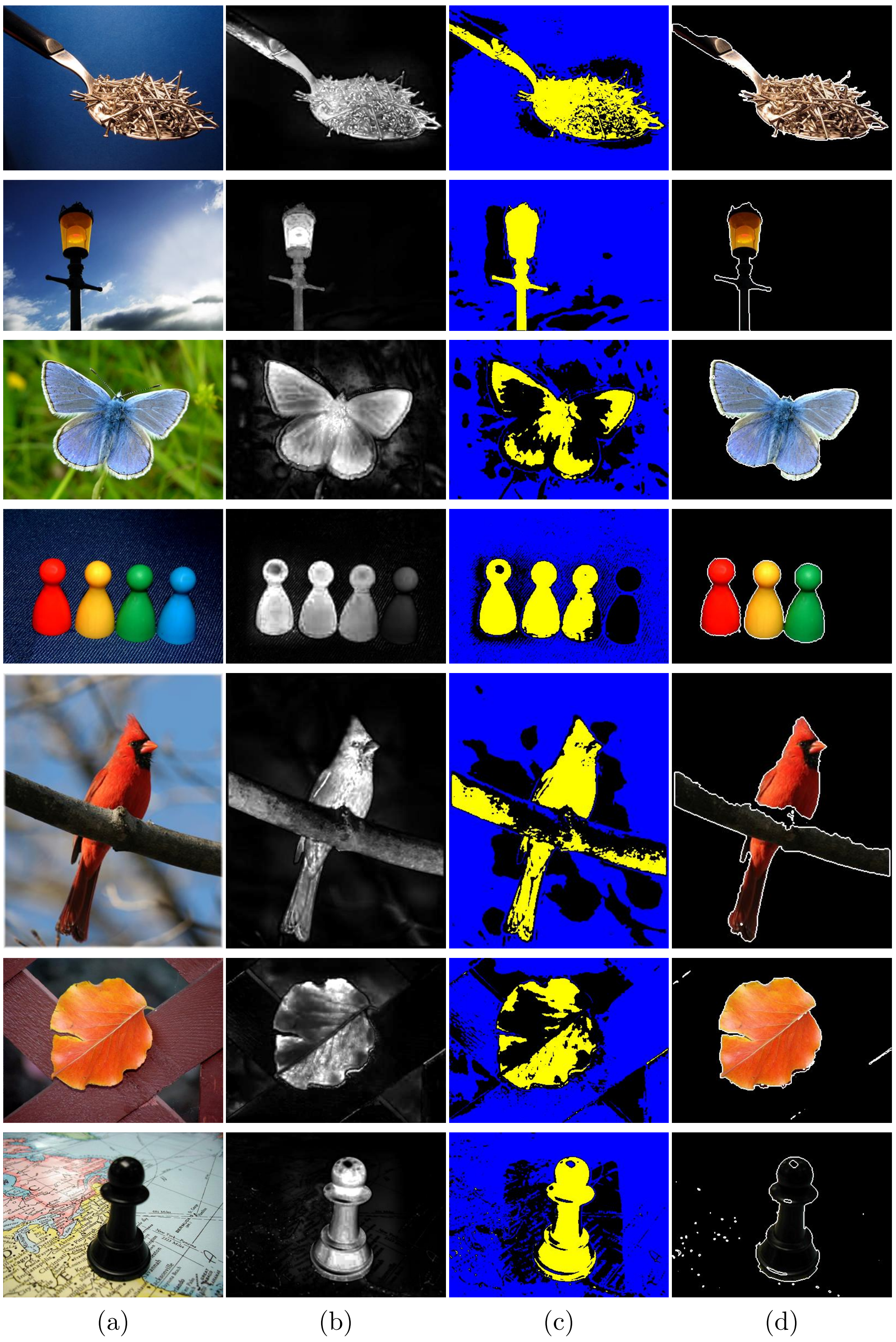


Figure 4.3: *Salient object segmentation using graph-cuts. (a) Original image. (b) Saliency map using our method MSSS (Section 3.3). (c) Labeling by thresholding explained in Section 4.1.2 - blue pixels belong to the background and yellow ones to the salient object. The black pixels have unknown initial labels. Yellow pixels are assigned hard links to the source (object) while blue pixels are assigned hard links to the sink (background). (d) The segmentation obtained using graph-cuts.*

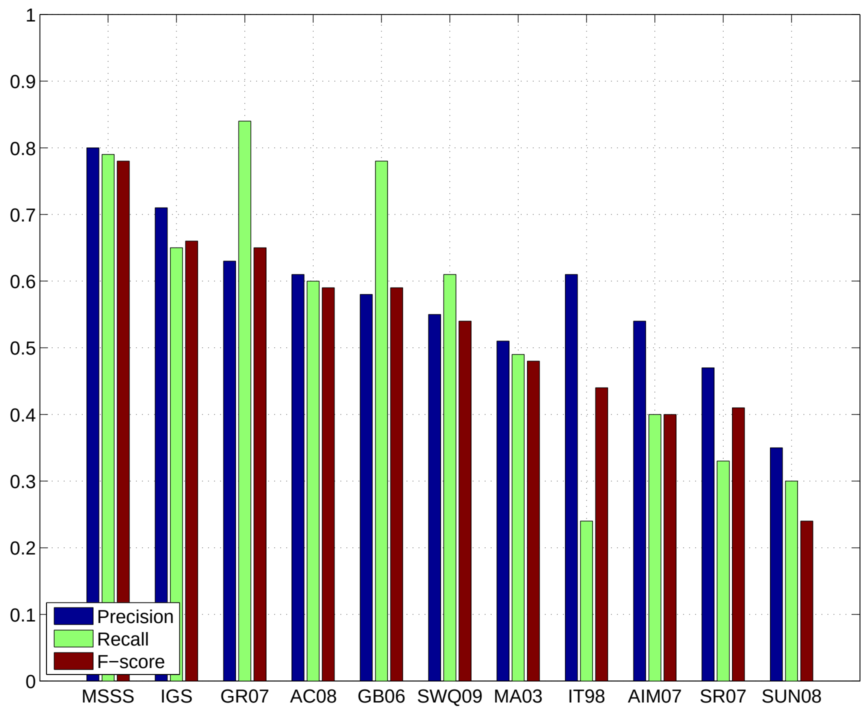


Figure 4.4: *Precision-Recall bars for graph-cuts based salient object segmentation, in decreasing order of F_β-score ($\beta = 0.5$). Our methods IGS and MSSS show high precision, recall and F_β scores computed on the 1000 image database.*

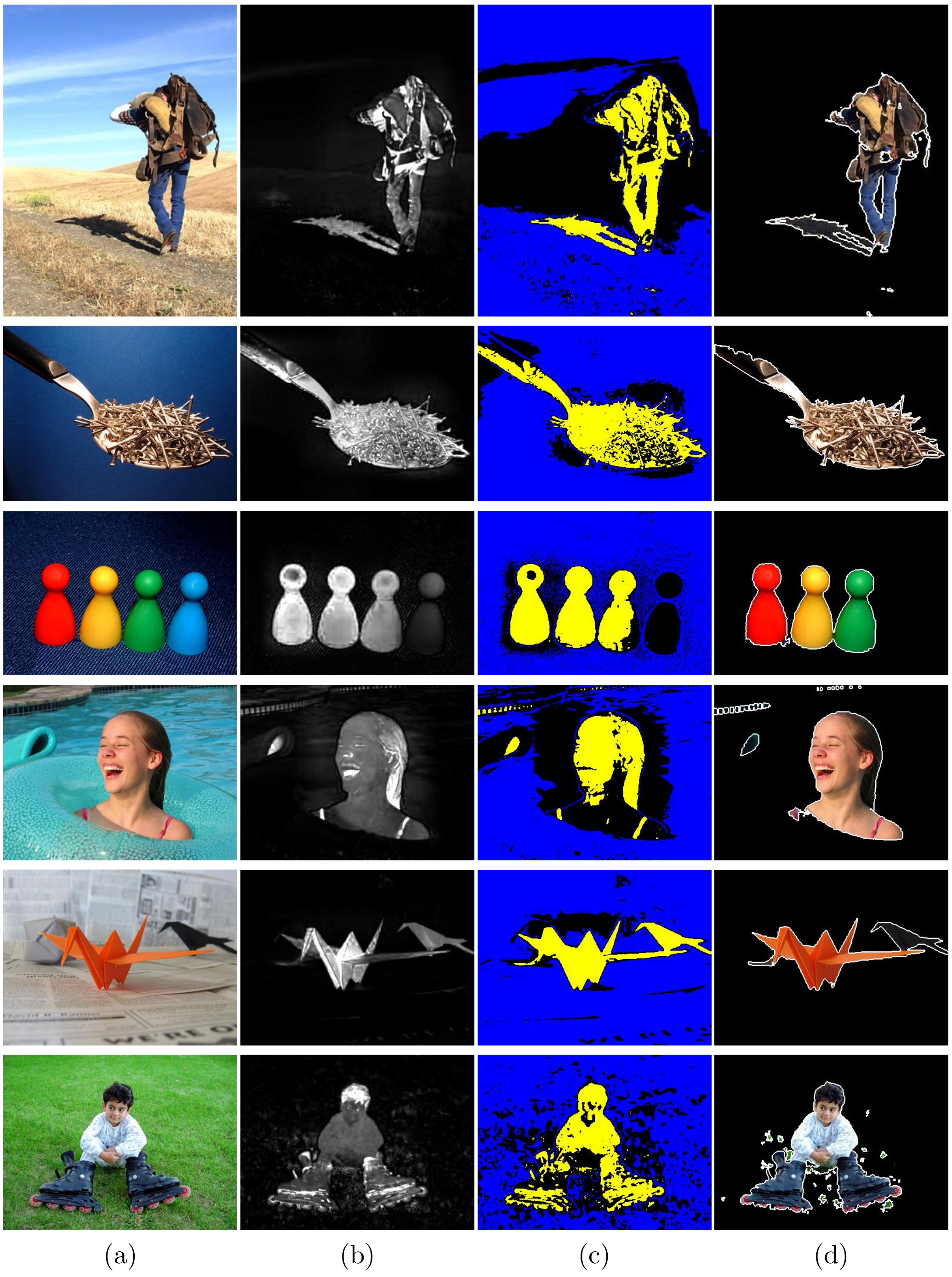


Figure 4.5: *(a) Original image. (b) Saliency map obtained using MSSS (Section 3.3). (c) Labeling by thresholding - blue pixels belong to the background and yellow ones to the salient object. The black pixels have unknown starting labels. (d) Result after geodesic path based segmentation. Each labeled (blue or yellow) pixel serves as a starting seed from which geodesic distances are computed to all the other pixels. Each pixel is assigned the label of the seed it is nearest to in terms of the geodesic distance. In a way, each seed competes with the other seeds for assigning its label to a pixel.*

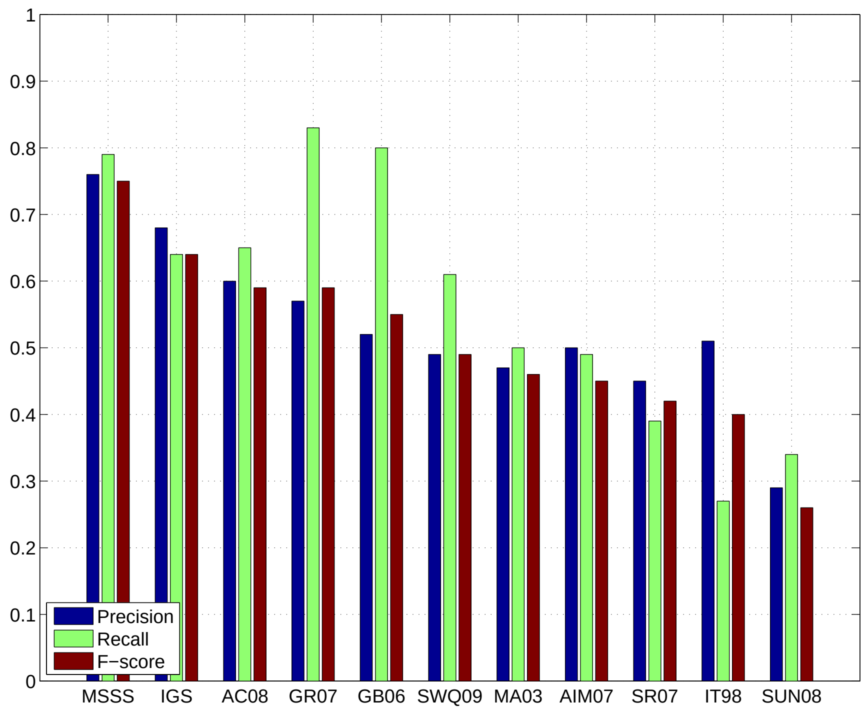


Figure 4.6: *Precision-Recall bars for geodesic path based salient object segmentation, in decreasing order of F_β-score ($\beta = 0.5$). Our methods IGS and MSSS show high precision, recall and F_β scores on 300 images with ground truth.*

fastest of the three approaches while the geodesic path based method is the slowest. The greater the number of seeds, i.e pixels labeled as belonging to salient object and background by initial thresholding, the more time is taken by geodesic path based segmentation.

4.2 Automatic Image Re-targeting

The diversity of today's display device sizes and aspect ratios demands smarter ways of re-targeting images than simple resizing, or adding black bars to fill empty space, to better deliver visually important or salient content for the given display dimensions. While cropping [26] is one option, image content adaptive warping [141] and seam carving [16] are other options that accentuate visually important content with minimal loss of original intent. These two re-targeting approaches have also been extended to videos [119, 144].

Gal et al. [53] were the first to propose a solution to the general problem of re-targeting an image while preserving regions of interest. In their method, the user has to manually specify the regions of interest based on which the image is adaptively warped.

Automatic *content awareness*, i.e the choice of visually important regions in re-targeting schemes, was introduced by [16, 119, 144, 141]. All such automatic re-targeting methods rely on finding visual importance values for each pixel. Avidan and Shamir [16] proposed the popular content aware re-targeting scheme of *seam carving*. They iteratively remove a seam, i.e. a connected set of vertical (horizontal) pixels, to reduce the width (height) of an image.

Rubinstein et al. [119] extend the original seam carving idea of [16] to videos by removing pixel manifolds in 3D volumes of video frames. Video re-targeting is also done by Wolf et al. [144] who locally warp the frames of the video. This is done by optimizing the best mapping between a source image and the re-targeted image.

Wang et al. [141] present another way of re-targeting images to arbitrary aspect ratios while preserving visually prominent features given by a saliency map using mesh based adaptive warping.

Assigning visual importance values (Section 4.2.1) is fundamental to all these automatic re-targeting methods. In this section, we demonstrate that our saliency maps provide better seam carving results than the commonly used gradient maps (see Fig. 4.7).

4.2.1 Visual Importance Maps

The key to content awareness, needed by all automatic re-targeting schemes, is a map of values that quantifies the relative visual importance of each pixel. The main methods of assigning importance values to pixels are measures of L_1-norm [16, 119]

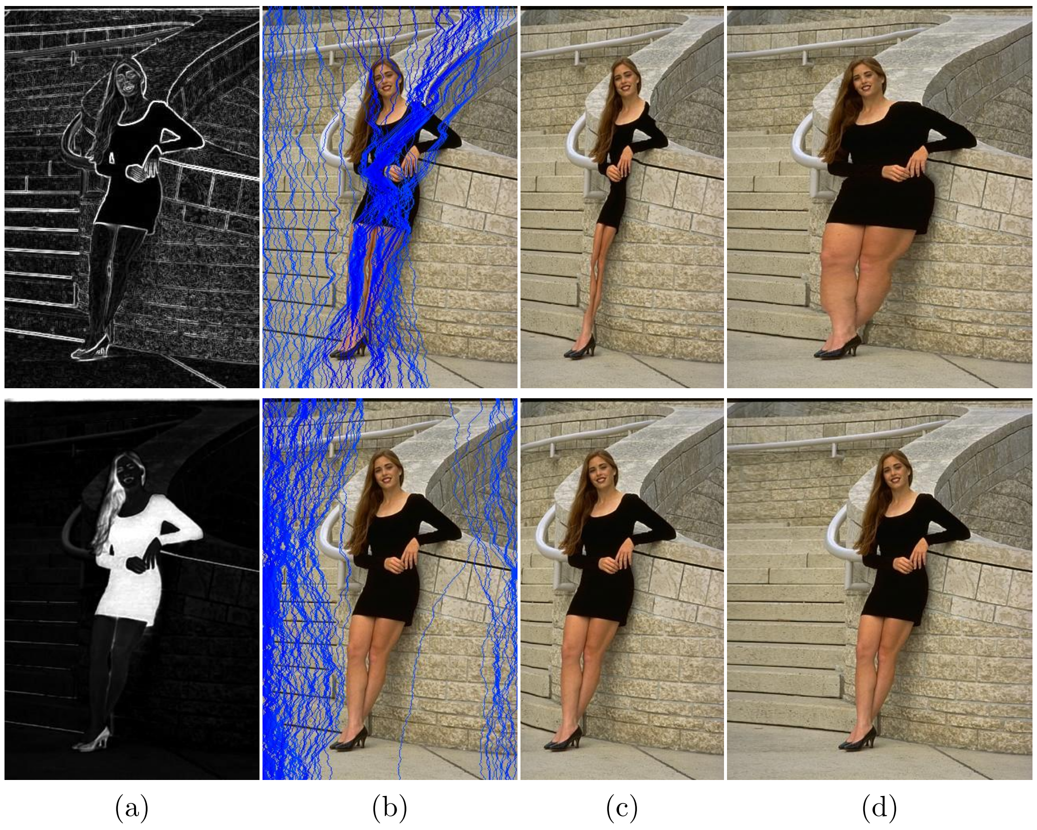


Figure 4.7: *Our seam carving compared to state-of-the-art [119]: top row results show the use of intensity gradient based energy map [119]; bottom row results show the use of our global contrast based saliency map. Column (a) is the visual importance map (the darker the pixel the lower the importance). Column (b) shows the seams chosen for re-targeting, superimposed on the original image. Column (c) shows image with 80% width. Column (d) shows image with 120% width.*

or L_2 norm [144] of the grayscale intensity gradient, face or other object detectors, saliency maps, or a combination of these [141, 144].

Avidan and Shamir [16] use the L_1-norm of the grayscale intensity gradient to compute their energy map. The energy map lets them successively remove seams of minimal energy as determined using a dynamic programming algorithm. They compare several ways of computing the energy maps, including Itti's saliency maps [72], and conclude that sums of magnitudes of gradients along the x and y axes (Eq. 4.5) and the same normalized by the maximum of the histogram of oriented gradients [34] give good results in general. To extend the spatial energy computation of Avidan and Shamir [16] to a spatiotemporal one for their video re-targeting case, Rubinstein et al. [119] introduce an inter-frame L_1-norm gradient term.

Wolf et al. [144] use a saliency map that combines the results of a face detector and a motion detector with the L_2-norm of the intensity gradient. The importance map of Wang et al. [141] is generated by multiplying the L_2-norm of the intensity gradient of the image with Itti's saliency maps [72]. Itti's maps do not highlight salient regions uniformly and are highly downsized as compared to the input image [4]. Thus, the resulting energy map has lower values for gradients that are not in the vicinity of a saliency blob of Itti's map.

Pixel energy computed from simple L_1-norm [16, 119] or L_2-norm [144] of the grayscale intensity gradient suffers from certain drawbacks. First, the values peak at edges rather than whole salient regions. Thus, energy is assigned to visually important image content only at edges and not whole regions (see Fig. 4.7, top-left image). Second, color information is ignored. The third disadvantage w.r.t iterative re-targeting schemes like seam carving [16] is the need to recompute the energy after seams are removed, since the local gradients may change after a seam is removed. Finally, gradient based maps can be noise sensitive.

Our saliency maps as computed using Eq. 3.4 uniformly assigns saliency values to entire salient regions, rather than just edges or texture regions. This is achieved by relying on the global contrast of a pixel rather than local contrast, measured in terms of both color and intensity features rather than just intensity as done previously [16, 119]. The saliency map is computed only once irrespective of the number of seam carving operations performed and is robust in the presence of noise. We show the effectiveness of our method in avoiding the usual artifacts of seam carving in normal and noisy images.

4.2.2 Seam carving

Avidan and Shamir [16] introduced the idea of seam carving for arbitrarily changing aspect ratios of images automatically. This is done by removing seams of low importance pixels from the image. They define a vertical (horizontal) seam to be a connected path of low energy pixels in the image from top to bottom (left to right) containing one, and only one, pixel in each row (column) of the image. Thus, removing a vertical (horizontal) seam reduces the width (height) by one pixel. Finding the globally minimal energy seam, which removes the least salient content, is posed as a dynamic programming optimization problem. The energy maps are computed using the L_1-norm of the intensity gradient as:

$$E_g(x, y) = \left|\frac{\partial}{\partial x} I(x, y)\right| + \left|\frac{\partial}{\partial y} I(x, y)\right| \tag{4.5}$$

where $E_g(x, y)$ is the resulting importance value of a pixel at column x and row y, and I is the grayscale intensity image. For a vertical seam removal, the dynamic programming memoization table entry $M(x, y)$ is given as:

$$M(x, y) = E_g(x, y) + min \begin{cases} M(x-1, y-1) \\ M(x, y-1) \\ M(x+1, y-1) \end{cases} \tag{4.6}$$

The globally minimal energy seam is found by backtracking from the minimum value of the last row in M to the first row.

Rubinstein et al. [119] note that despite seam carving being an energy removal operation, a removed seam may actually introduce more energy than it takes away because of previously non-adjacent pixels becoming neighbors. They therefore introduce *forward energy* criteria (equally applicable for both image and video cases; illustrated in Fig.4.8), such that the optimal seam found is one whose removal reintroduces minimum amount of energy. This changes Eq. 4.6 to:

$$M(x, y) = E_g(x, y) + min \begin{cases} C_L(x, y) + M(x-1, y-1) \\ C_U(x, y) + M(x, y-1) \\ C_R(x, y) + M(x+1, y-1) \end{cases} \tag{4.7}$$

where C_L, C_U, and C_R are image gradients resulting from non-adjacent pixels becoming neighbors when a seam pixel separating them is removed, and are computed as:

$$\begin{aligned} C_U(x, y) &= |I(x+1, y) - I(x-1, y)| \\ C_L(x, y) &= |I(x, y-1) - I(x-1, y)| + C_U(x, y) \\ C_R(x, y) &= |I(x, y-1) - I(x+1, y)| + C_U(x, y) \end{aligned} \tag{4.8}$$

4.2.3 Improved seam carving

As mentioned above, the importance maps used by [141, 16, 119, 144] determine local grayscale contrast using gradients that result in higher importance values for

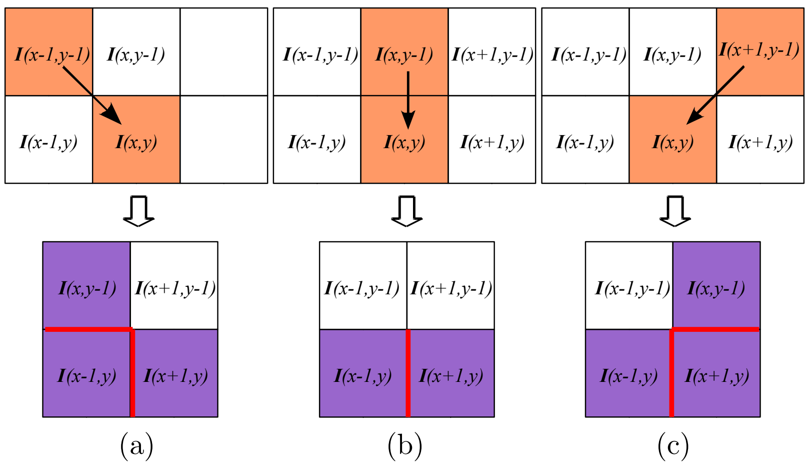


Figure 4.8: *Calculating the three possible vertical seam step costs for the pixel at position (x, y) using forward energy. After removing the seam (shown in orange color), new neighbors (shown in magenta) and new pixel edges (red line segments) are created. In each of the three cases, the cost is defined by the forward difference in the newly created pixel edges. Note that the new edges created in row $y-1$ were accounted for in the cost of the previous row pixel.*

textured areas and edges, but lower values for smooth salient regions. Wang et al. [141] attempt to address this problem by multiplying the L_2 norm of the gradient with Itti's saliency maps [72], while Wolf et al. [144] combine the results of a face detector and a motion detector [92] with the L_2-norm of the intensity gradient. However, these do not significantly alleviate the drawbacks of intensity gradient maps.

On the other hand, our saliency maps as computed using the method of Section 3.2 have uniformly highlighted salient regions with well-defined boundaries. We also introduce the use of color information in the forward energy terms of Eq. 4.8. This is done by replacing the scalar gray scale differences with the corresponding vector distances in CIELAB color space to obtain:

$$\begin{aligned} C_U(x,y) &= \|\mathbf{I}(x+1,y) - \mathbf{I}(x-1,y)\| \\ C_L(x,y) &= \|\mathbf{I}(x,y-1) - \mathbf{I}(x-1,y)\| + C_U(x,y) \\ C_R(x,y) &= \|\mathbf{I}(x,y-1) - \mathbf{I}(x+1,y)\| + C_U(x,y) \end{aligned} \tag{4.9}$$

This computes forward energy better as both color and intensity information is taken into account. Although, the use of color in forward energy terms gives an advantage over intensity (see Fig. 4.9), the more significant advantage is provided by the use of our saliency maps. Our saliency maps (see Fig. 4.7, and Fig. 4.14) generated using Eq. 3.4 coupled with the modified forward energy terms of Eq. 4.9 overcome the limitations of previously used importance maps [141, 16, 119, 144].

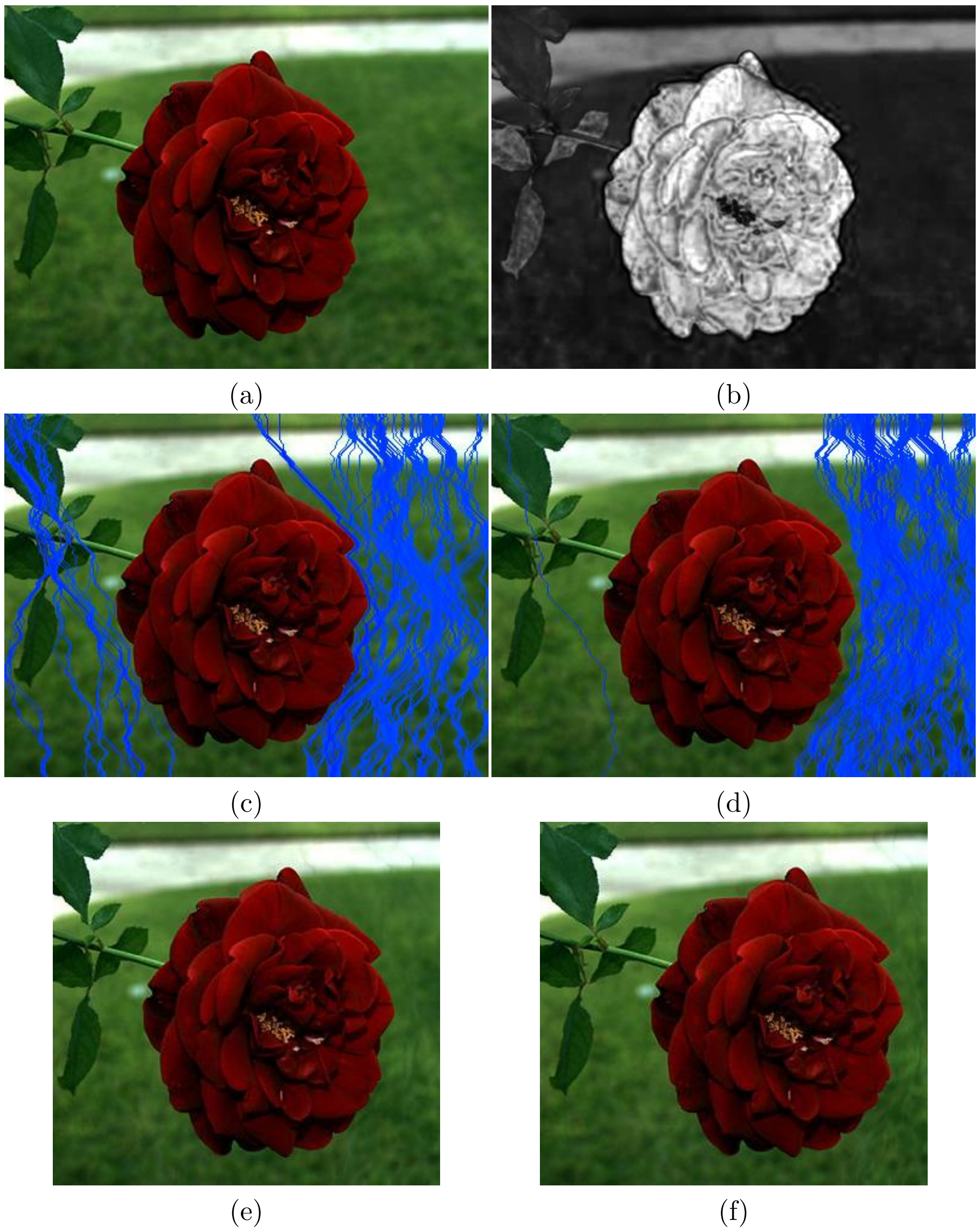


Figure 4.9: *Figure showing the difference in the choice of seams when taking into account color information instead of only intensity in forward energy terms. (a) Input image. (b) Saliency map for seam carving as computed using Eq. 3.4. (c) Seams chosen for removal when forward energy terms use intensity information only. (d) Seams chosen when forward energy terms use color and intensity information. (e) 80% width using intensity based forward terms (f) 80% width using color and intensity based forward terms.*

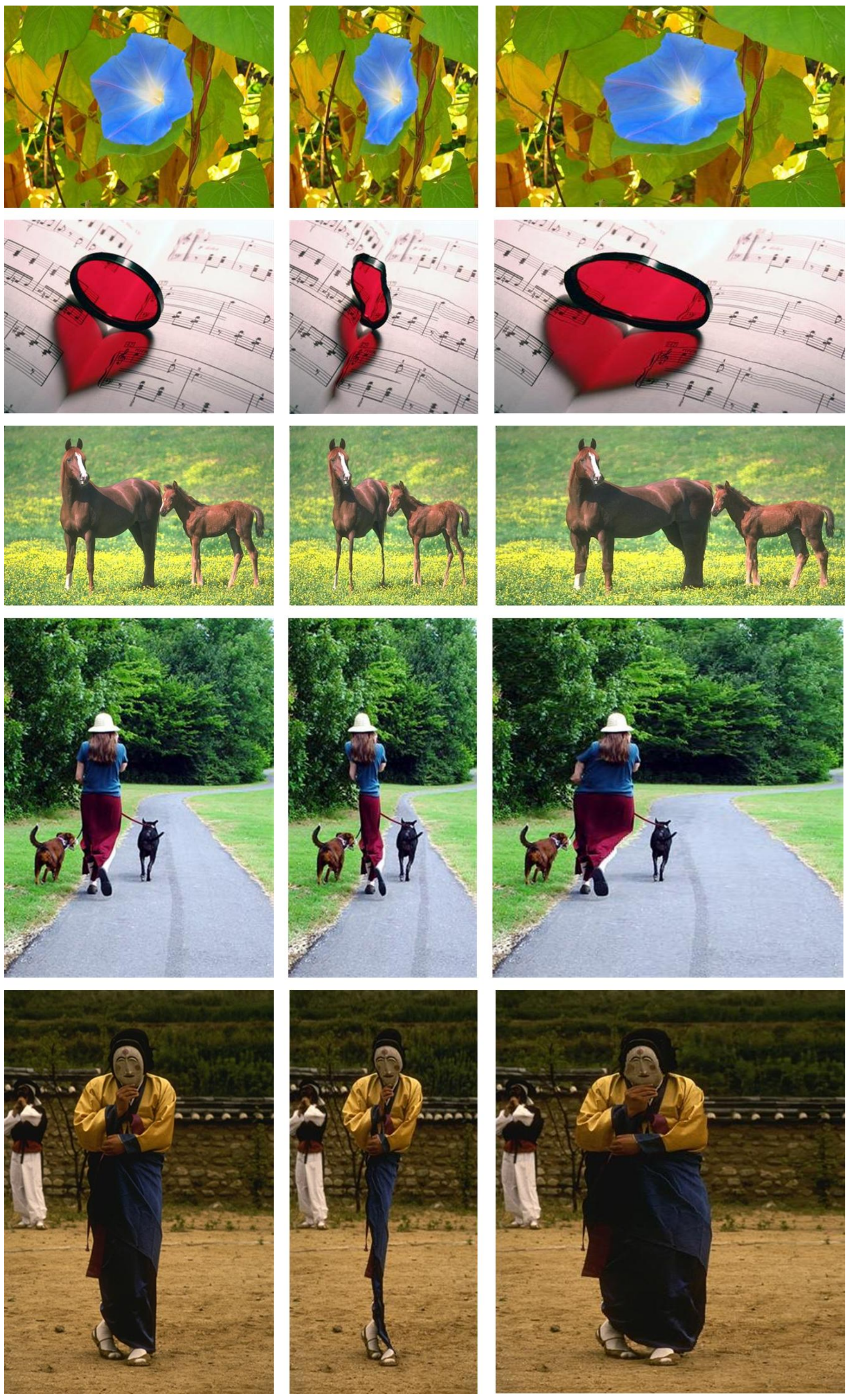

(a) Input (b) 70% width (c) 130% width

Figure 4.10: *Image re-targeting comparison for 70% and 130% of original width. Column(a) shows the original images.(b) and (d) are the outputs of the method of Rubinstein et al. [119] implemented by us.*

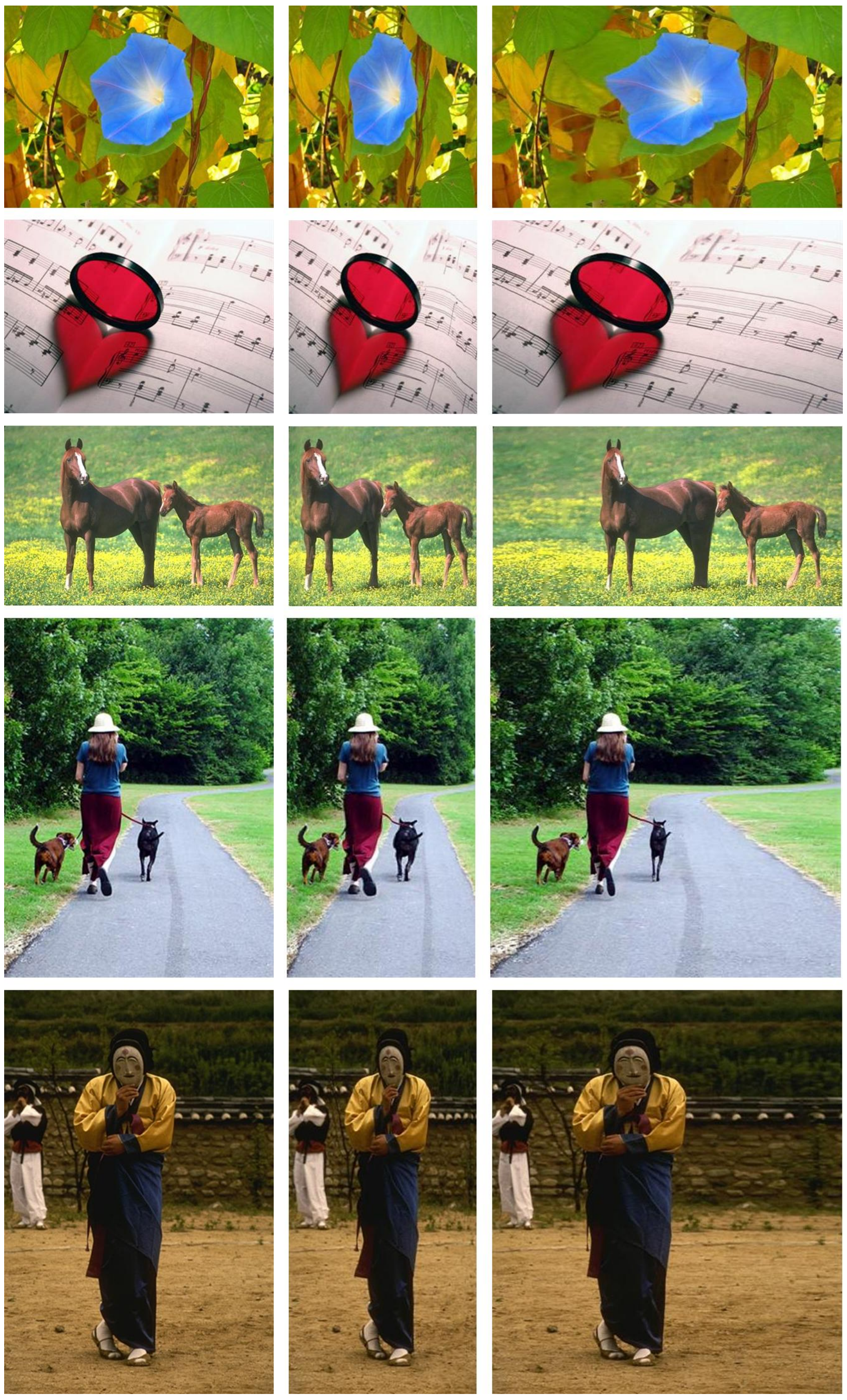


Figure 4.11: *Image re-targeting comparison for 70% and 130% of original width: Column(a) shows the original images. (b) and (c) are the outputs of Wolf et al. [144] as provided by the authors. Apart from grayscale gradient information they also use face detection results.*

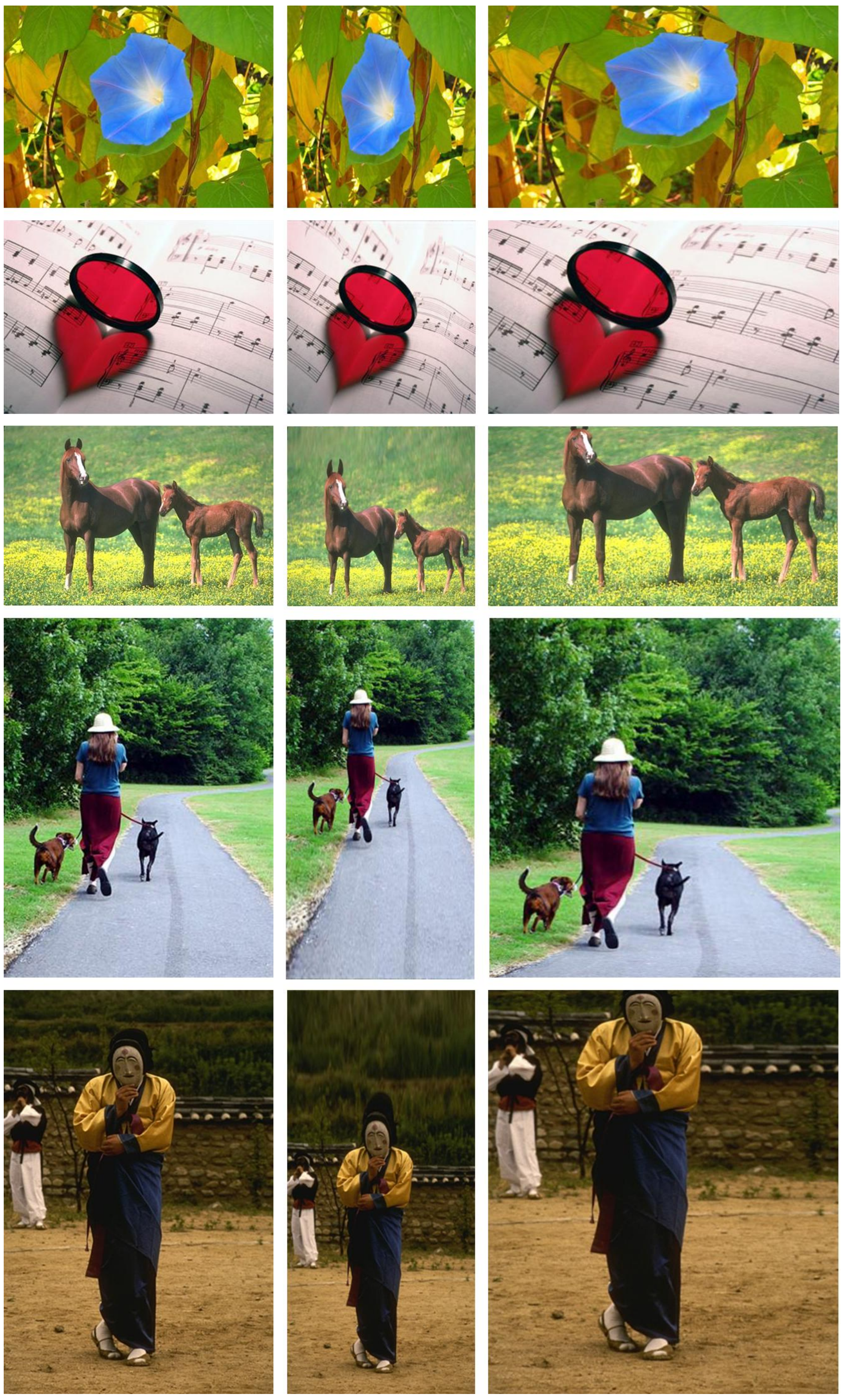


Figure 4.12: *Image re-targeting comparison for 70% and 130% of original width: Column(a) shows the original images. (b) and (c) are the outputs of Wang et al. [141] as provided by the authors. They combine grayscale gradient maps with Itti's saliency maps [72] to obtain their importance maps.*

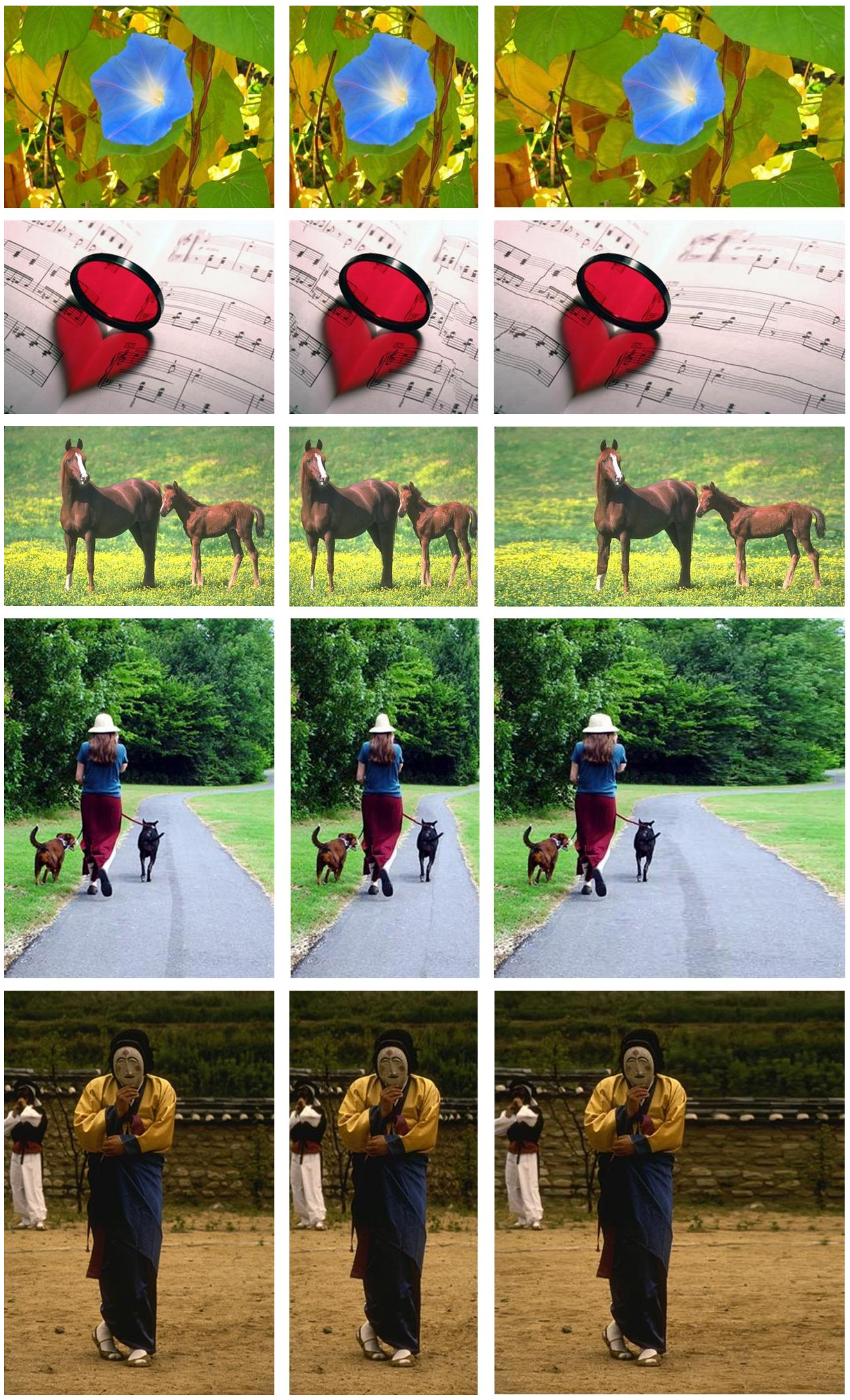

(a) Input (b) 70% width (c) 130% width

Figure 4.13: *Image re-targeting comparison for 70% and 130% of original width: Column(a) shows the original image. (b) and (c) are our seam carving results.*

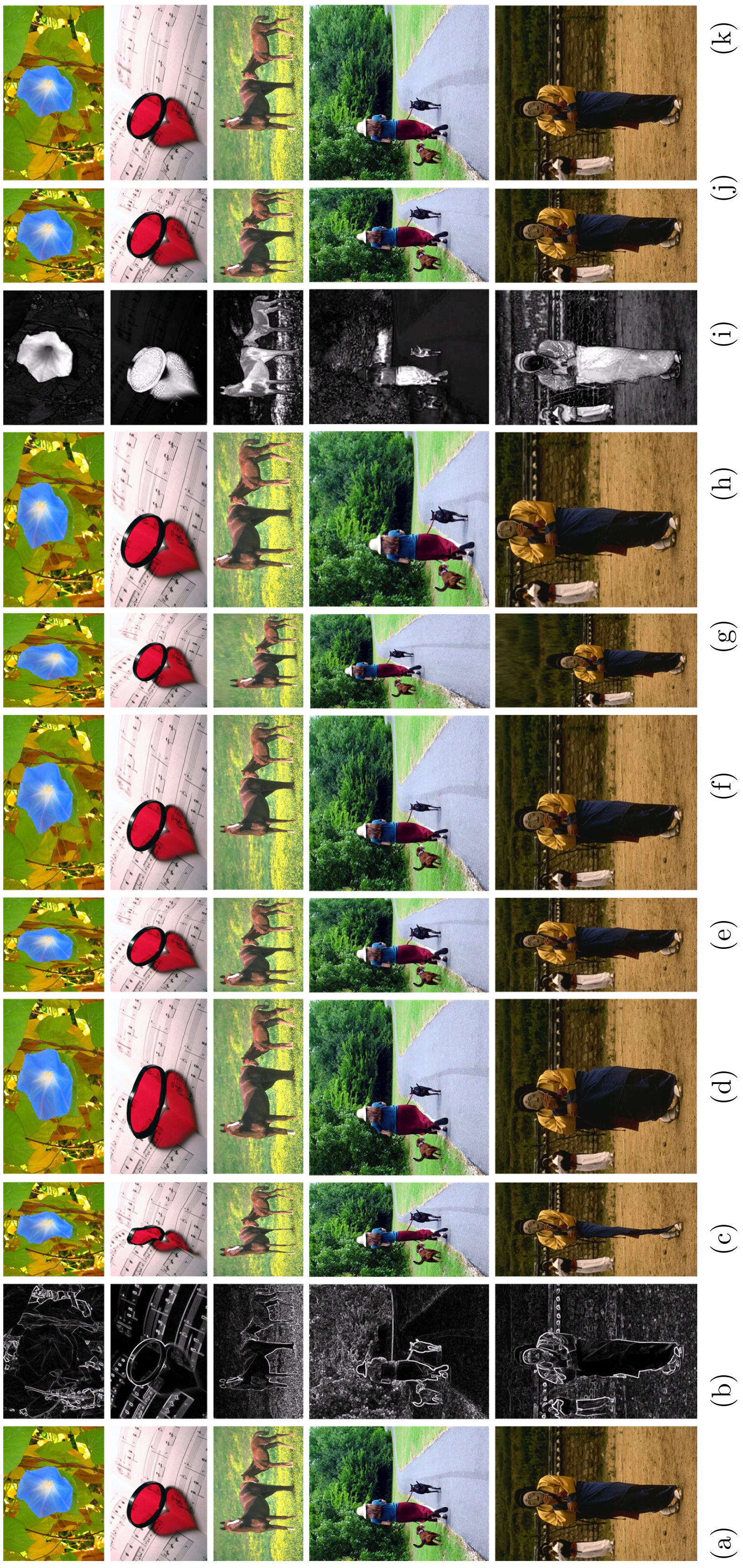


Figure 4.14: *A side-by-side comparison of image re-targeting comparison for 70% and 130% of original width: Column (a) shows the original images. Column (b) shows the gradient map obtained using Eq. 4.5. Columns (c) and (d) are the outputs of Rubinstein et al. [119], (e) and (f) are the outputs of Wolf et al. [144], (g) and (h) are the outputs of Wang et al. [141]. Column (i) shows our saliency maps obtained using the method presented in Section 3.2, and (j) and (k) are our seam carving results. Columns (c) to (h) use gradient maps for content awareness while we use saliency maps.*

The seam carving results (Fig. 4.13 and Fig. 4.14) obtained by using our saliency map and modified forward energy terms are visually compared against those of Rubinstein et al. [119] obtained using gradient maps (Fig. 4.10 and Fig. 4.14), as well as those of Wang et al. [141] (Fig. 4.11 and Fig. 4.14)and Wolf et al. [144] (Fig. 4.12 and Fig. 4.14). The results from the methods of Wang et al. [141] and Wolf et al. [144] were provided by the respective authors. In Fig. 4.15 to Fig. 4.18 we also show a comparison of seam carving performed using gradient maps with that obtained using the saliency maps of various state-of-the-art methods we covered in Chapter 3. In general, saliency maps perform better than simple gradient maps for images with salient content.

We show results for changing aspect ratios by both removing and adding seems as needed. As proposed by Avidan and Shamir [16], to enlarge an image by p seams, we first find p seams for removal and duplicate them. To affect a change in both dimensions of an input image, we choose between a vertical or a horizontal seam at each step depending on which has lower energy.

Our saliency maps highlight visually important regions of the image uniformly and not just at their edges. Thus, the seams chosen do not pass through high energy regions, such as salient objects (see Fig. 4.7b). This permits us to obtain seam carving results without artifacts in salient regions. It must be added that in cases where the salient object is not highlighted correctly by our maps, gradient based energy maps from Eq. 4.5 may provide better re-targeting results.

In our saliency maps, the importance value associated with a pixel is computed with respect to the entire image (and not the immediate four or eight neighbors). High saliency values are not assigned just at the edge of a region, but the entire region. Once we know which pixels are less salient with respect to the original image, we can remove them without having to recompute their importance after each removal. This is unlike local gradients, whose values depend on local pixel neighborhood, which may change when a seam of pixels is removed. Thus, unlike the gradient maps, there is no need to recompute the saliency maps independent of the number of seam carving operations performed.

4.2.4 Noise Robustness

Our saliency maps are more robust to noise than local intensity gradient based maps. There are two reasons for this. First, our global approach is independent of local noise patterns that strongly affect gradient based energy maps. Second, Eq. 3.4 allows using Gaussian blurring that can be increased according to the requirements of the application. Although a 3×3 binomial kernel suffices, if very low bit-rate coding is used or if Exif (Exchangeable Image File Format [2]) data indicates the use of high ISO values (indicating probability for higher noise), one can increase the binomial kernel size.

We experimented with Gaussian noise up to variance 0.1 (Fig. 4.19), and salt

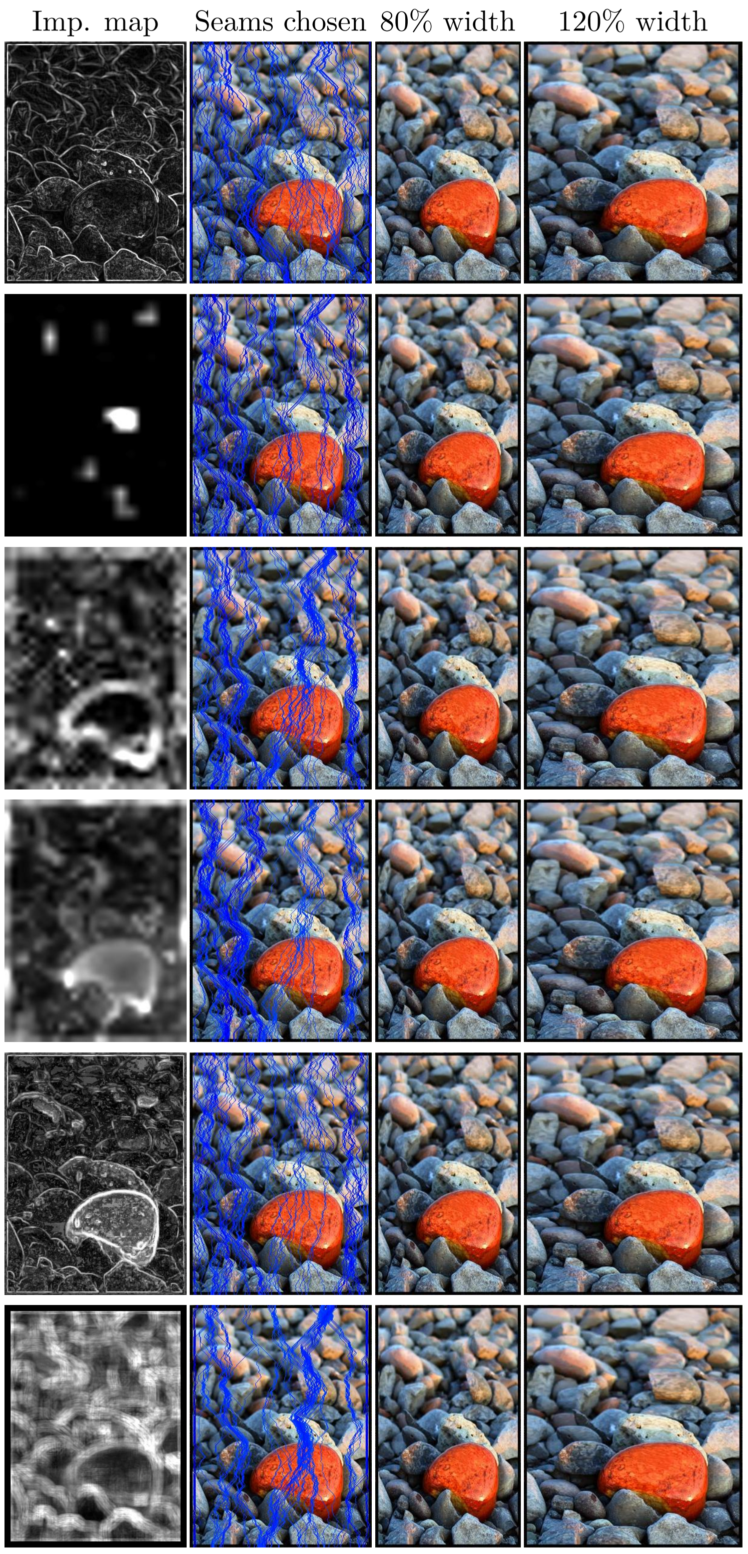


Figure 4.15: *Comparison of seam carving using saliency maps of state-of-the-art techniques. The far left column shows the importance maps used for seam carving. The first of these on the top is the gradient edge map. The ones that follows are the saliency maps obtained using the techniques IT98 [72], MA03 [98], GB06 [62], GR07 [100], and AIM07 [25], respectively.*

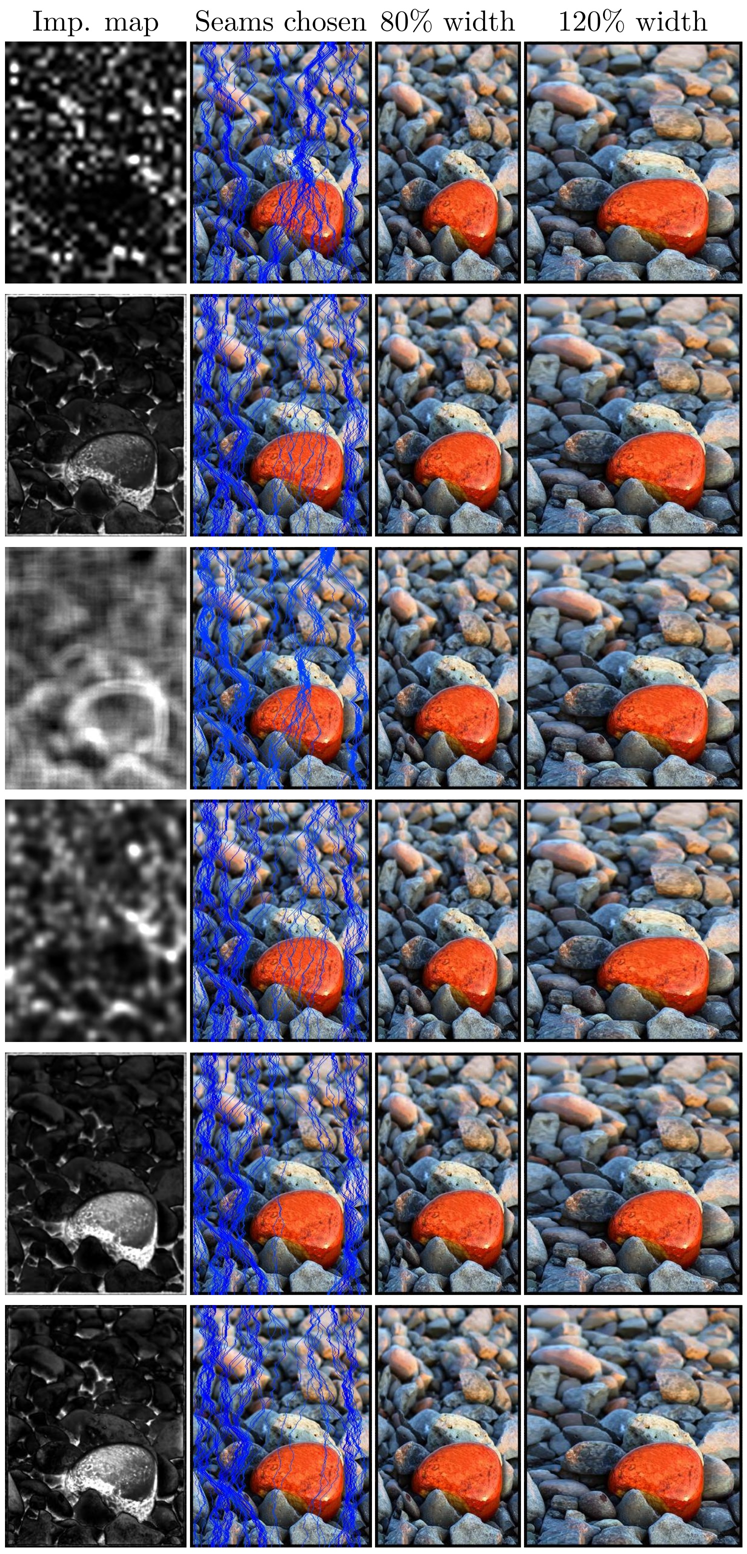


Figure 4.16: *Comparison of seam carving using saliency maps of state-of-the-art techniques. The far left column shows the importance maps used for seam carving. This column shows saliency maps obtained from the techniques SR07 [67], AC08 [4], SUN08 [149], SWQ09 [21], and our techniques IGS and MSSS, respectively.*

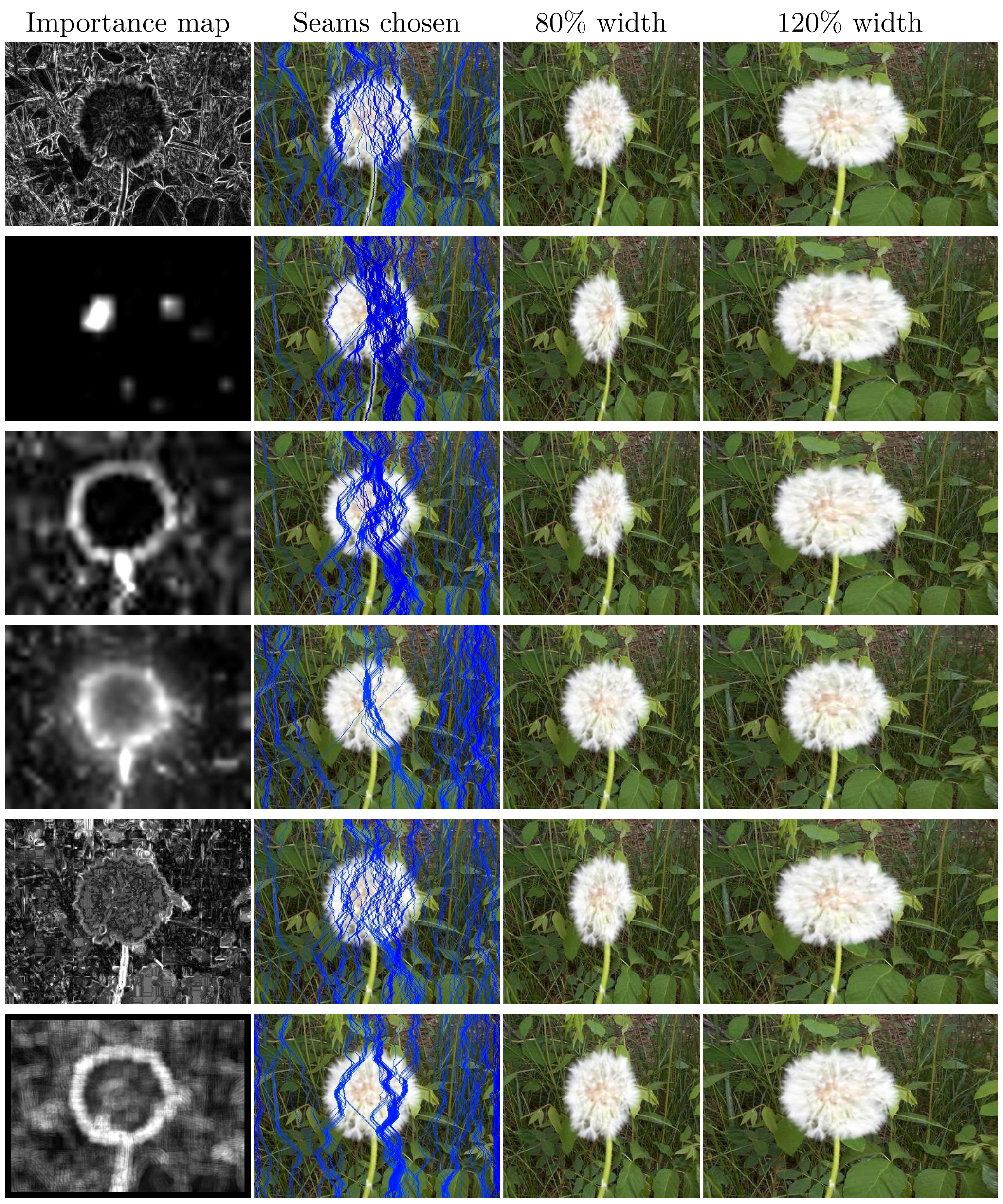


Figure 4.17: *Comparison of seam carving using saliency maps of state-of-the-art techniques. The far left column shows the importance maps used for seam carving. The first of these on the top is the gradient edge map. The ones that follows are the saliency maps obtained using the techniques IT98 [72], MA03 [98], GB06 [62], GR07 [100], and AIM07 [25], respectively.*

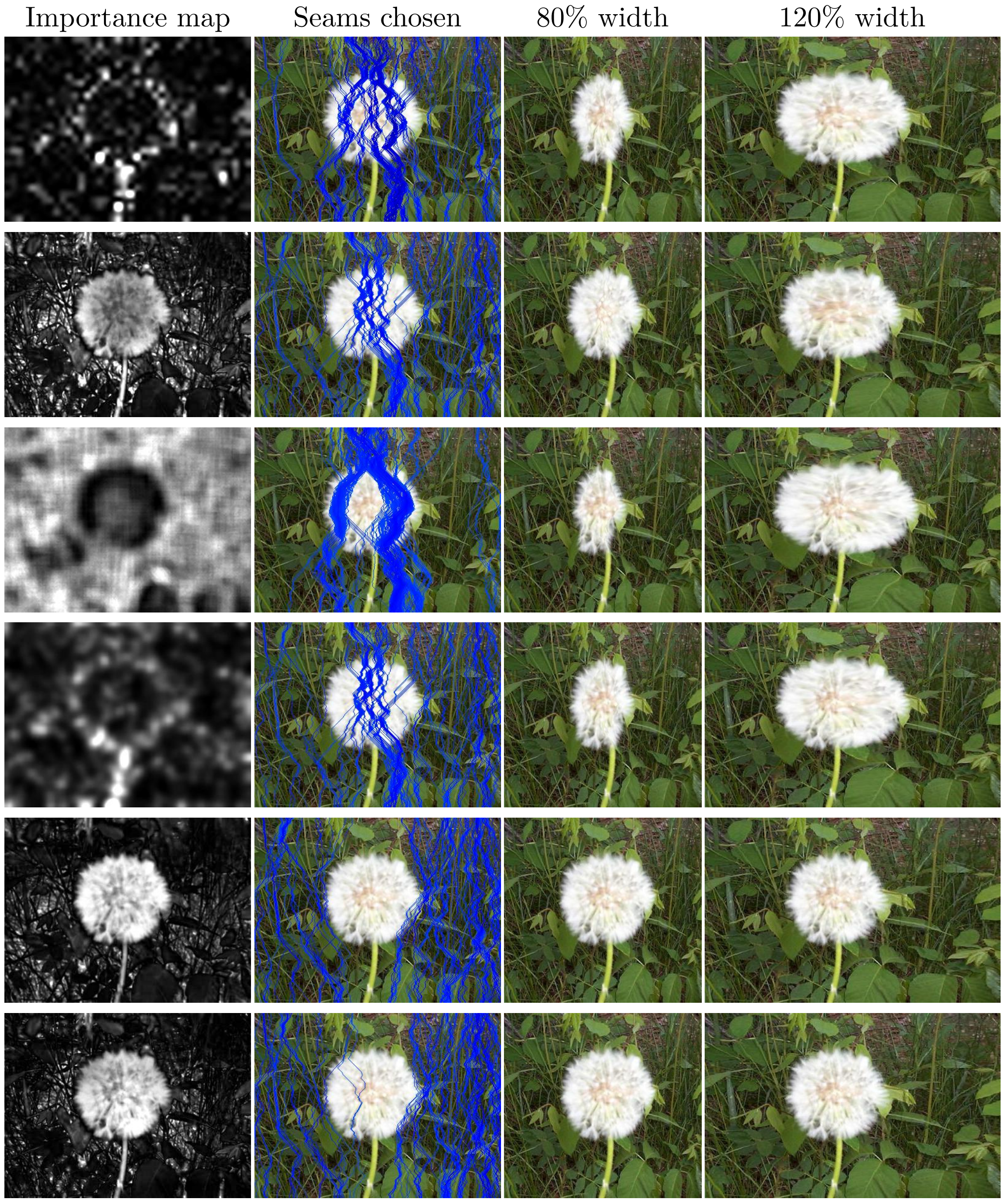


Figure 4.18: *Comparison of seam carving using saliency maps of state-of-the-art techniques. The far left column shows the importance maps used for seam carving. This column shows saliency maps obtained from the techniques SR07 [67], AC08 [4], SUN08 [149], SWQ09 [21], and our techniques IGS and MSSS, respectively.*

and pepper noise with noise density up to 0.1. We retain the binomial kernel at 3×3 for all experiments (with and without noise). Our saliency maps provide good seam carving results even in the presence of noise, as illustrated in Fig. 4.19.

4.3 Summary of the chapter

The first application of saliency detection we presented in this chapter is salient object segmentation. We presented three techniques for doing this: using mean-shift clustering and saliency adaptive thresholding, using graph-cuts, and using geodesic distances. We applied these three techniques to saliency maps obtained from other saliency detection techniques also and compared the results. We then presented the application of saliency detection in content aware image resizing using seam carving. The performance of our saliency maps in these two applications establishes their effectiveness as compared to state-of-the-art saliency detection techniques. It is also possible to combine generic saliency detection like ours with task-specific detection like text detection. This is shown in Appendix A.6.

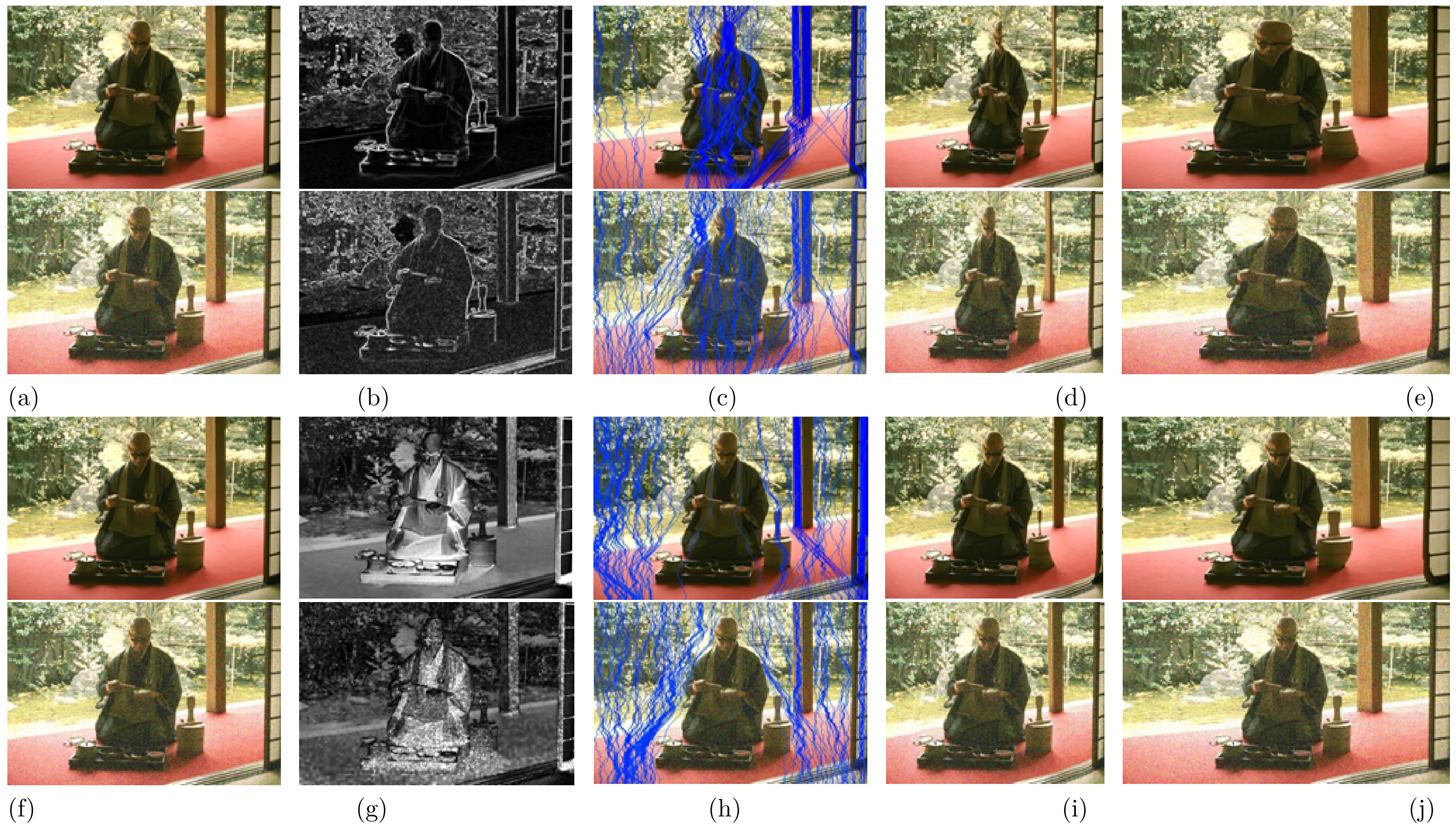


Figure 4.19: *Column (a) shows original (above) and noisy (below) versions of the image. Column (b) shows the corresponding gradient energy maps. Column (c) shows the seams chosen based on the gradient maps using the method of Rubinstein et al. [119]. Columns (d) and (e) show width re-targeted to 80% and 120% of the original. Columns (f) to (j) show the result of using our maps instead. Note how much the seam selection is affected by noise in gradient maps as compared to our maps. Despite the noise, the seams continue to be chosen from the same regions of the image (Column (h)).*

Chapter 5

Database Saliency and Image Summaries

Most digital camera owners accumulate databases in the order of a few thousand images. It is useful to have a small set of representative images from a database of thousands of images to summarize its content. There are two related aspects of such image summaries: how to generate them and how to present them. We address both issues[1].

In this chapter, we extend the idea of image saliency to databases and introduce the notion of *database saliency*. We argue that in image databases, there are certain images that are more uncommon or *salient* than others and therefore are likely to be more interesting (see Fig. 5.1). These images should be given more priority in the image summaries as opposed to less interesting ones.

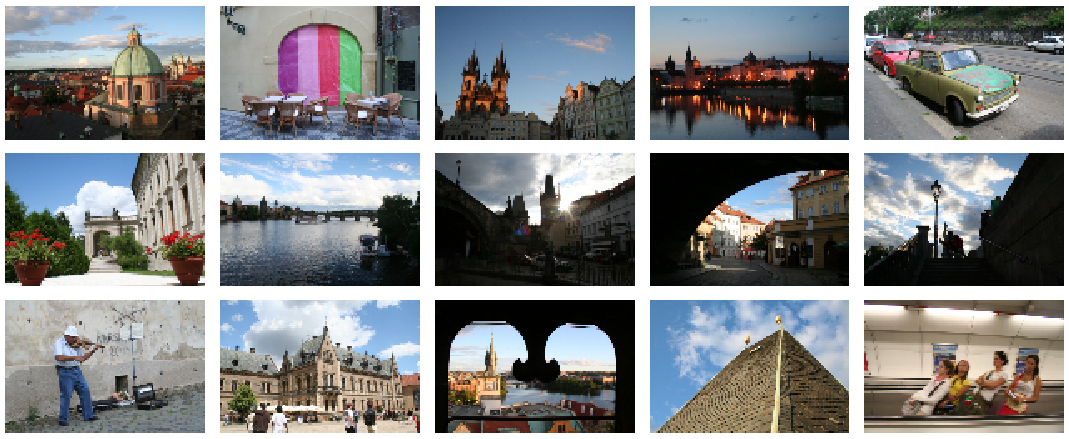
5.1 Ranking images

We rank the images in decreasing order of their *database saliency values*. The database saliency value of an image corresponds to its degree of 'interstingness'. We compute the database saliency value of an image as its total distance from all the cluster centers of the database. This is illustrated in Fig. 5.4 and explained in the following sections.

5.1.1 Image clusters and database saliency

We group the images of the database into a given number of clusters. The number of clusters formed depends on how many images are required by the user to create a summary of the database. Clustering is performed using the k-means algorithm (Section 2.5.1). For this we need an image signature, i.e. a compact feature representation of the image, and a distance measure to compute the similarity between

[1]The material presented in this chapter is also available in the reference [6]


(a) An example database of images

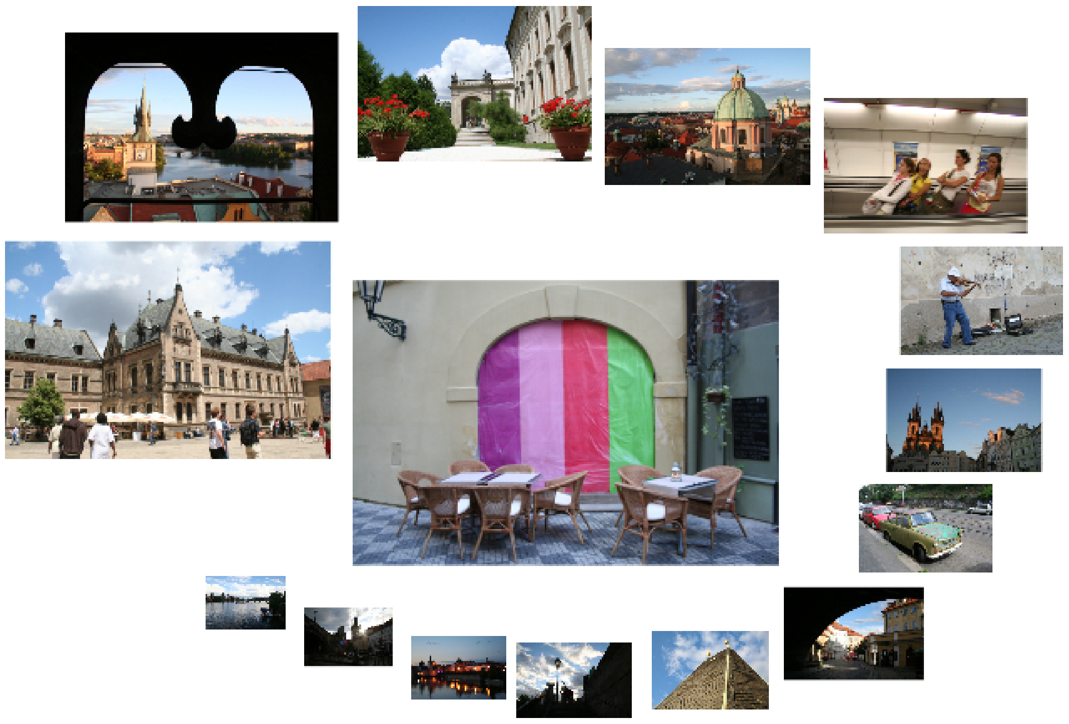

(b) Some images are more interesting than others. Here the size of the image is relative to the degree of 'interestingness'.

Figure 5.1: *The goal of database saliency detection is to automatically rank images in the order of their 'interestingness'.*

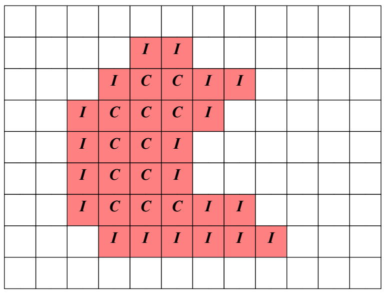


Figure 5.2: *An artificial input image with coherent pixels labeled C and incoherent pixels labeled I. In the case shown here, coherent pixels have more than six similar neighbors while incoherent pixels have six or less.*

two images. We propose the use of *image saliency weighted color coherence vectors* (ISWCCV) as the image signature and the χ^2 distance as a similarity measure [80].

Image saliency weighted color coherence vectors (ISWCCV)

Simple color histograms are poor abstractions of images as two visually dissimilar images can have the same histogram (see Fig. 5.3(a)). Color coherence vectors (CCV) were proposed by Zabih and Pass [111] to alleviate this problem. They label pixels as belonging to one of the two classes: *coherent* and *incoherent*, as illustrated in Fig. 5.2, and create a separate histogram for each. A coherent pixel is one that has at least a threshold T_n number of similar pixels as its neighbors, while an incoherent pixel has fewer.

To obtain an ISWCCV, we use image saliency computed using the method presented in Section 3.2. Instead of simply counting a pixel belonging to a bin, we take into account the saliency value $S(x, y)$, normalized in the interval $[0, 1]$, to compute the height of the bin. Thus, more weight is given to the salient pixels of an image rather than considering the entire image uniformly. The height $h(n)$ of the nth bin is computed as:

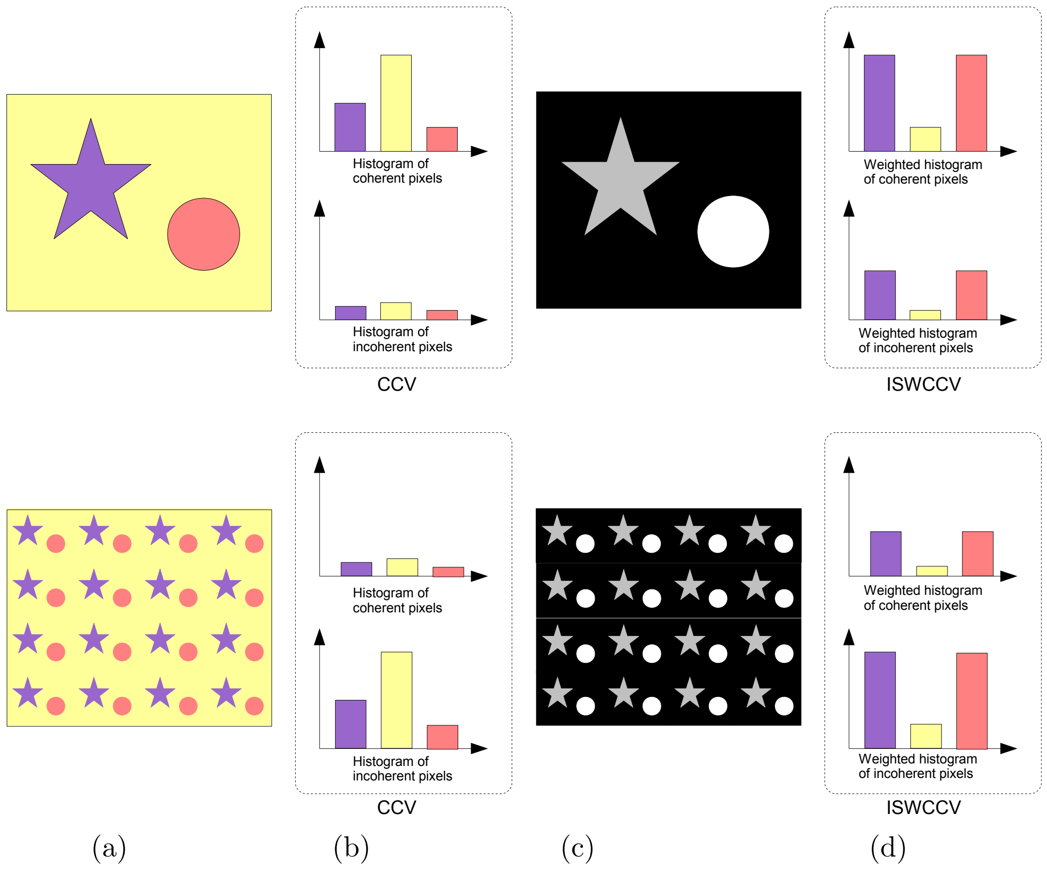
$$h(n) = \frac{\sum_{(x,y)\in bin(n)} S(x, y)}{\sum_{n=1}^{B} \sum_{(x,y)\in bin(n)} S(x, y)} \tag{5.1}$$

where B is the number of bins of the histogram. The expression in the denominator is the normalizing constant. Fig. 5.3 illustrates the creation of ISWCCV using saliency maps.

The χ^2 significance test [80] provides a measure of similarity of two histograms h_1 and h_2 as:

$$\chi^2(h_1, h_2) = \frac{1}{2} \sum_{n=1}^{B} \frac{(h_1(n) - h_2(n))^2}{h_1(n) + h_2(n)} \tag{5.2}$$

The degree of similarity D between two images is computed as a weighted sum of



Histogram of coherent pixels
Histogram of incoherent pixels
CCV
Weighted histogram of coherent pixels
Weighted histogram of incoherent pixels
ISWCCV
Histogram of coherent pixels
Histogram of incoherent pixels
CCV
Weighted histogram of coherent pixels
Weighted histogram of incoherent pixels
ISWCCV
(a)
(b)
(c)
(d)

Figure 5.3: *(a) The two images on the left have the same global color histogram but different CCV's.(b) Coherent and incoherent pixel histograms forming the CCV's for the two images. (c) Saliency maps corresponding to the input image. (d) Image saliency weighted color coherence vectors obtained by taking into account image saliency for creating the coherent and incoherent color histograms.*

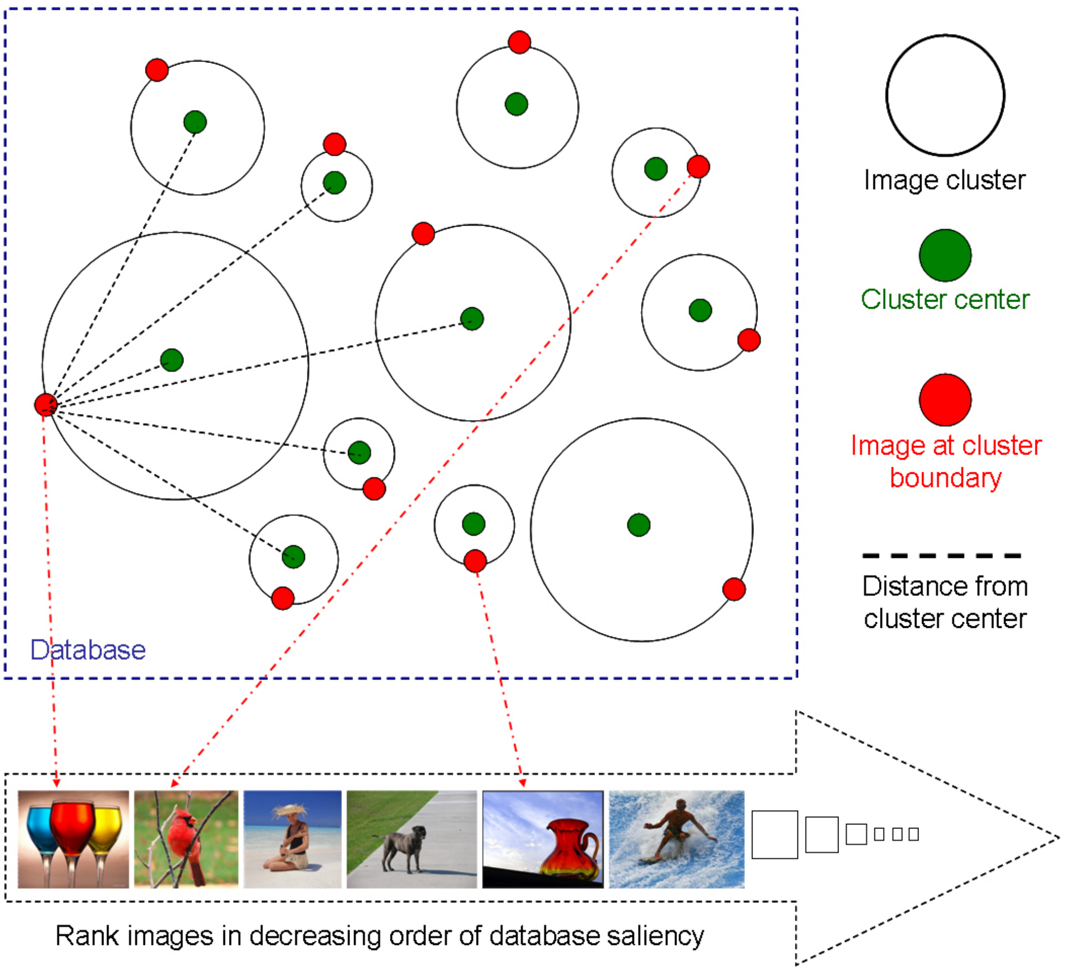


Image cluster
Cluster center
Image at cluster boundary
Distance from cluster center
Database
Rank images in decreasing order of database saliency

Figure 5.4: *Computing database saliency as a sum of distances from all cluster centers.*

the χ^2 similarity measures of the coherent and incoherent pixel histograms:

$$D({}^ch_1, {}^ih_1, {}^ch_2, {}^ih_2) = \alpha\chi^2({}^ch_1, {}^ch_2) + (1-\alpha)\chi^2({}^ih_1, {}^ih_2) \tag{5.3}$$

where the superscripts c and i represent coherent and incoherent pixel histograms, respectively, and $\alpha \in [0, 1]$ decides their relative weight. This similarity measure D is used by us for performing the k-means clustering of the images.

We create the histograms on images resized to 40×30 pixels. We quantize the values in each channel in the interval $[0, 3]$ i.e. in to four bins. In our implementation, we choose T_n to be 6, i.e, a pixel is considered coherent if it has at least 6 similar neighbors. A neighboring pixel is considered similar if the quantized value of each color channel value is the same as the quantized value of each channel of the pixel in consideration. As as result of the quantization, each histogram has 64 $(4\times4\times4)$ bins.

Finally, we use $\alpha = 0.3$ to relatively weigh the coherent and incoherent histograms.

Computing database saliency

The 'interestingness' value i.e. the database saliency value of the nth image in the database is computed as:

$$S(n) = \sum_{m=1}^{k} D({}^{c}h_m, {}^{i}h_m, {}^{c}h_n, {}^{i}h_n) \tag{5.4}$$

where k is the number of clusters created, and ${}^{c}h_m$ and ${}^{i}h_m$ correspond to the cluster centers. This pair of values (i.e. ISWCCV) is computed as the average of all the coherent and incoherent histograms belonging to a cluster.

The bigger $S(n)$ is, the more interesting we consider the image to be. An image is further away from its cluster center when it is less similar to the rest of the images in that cluster. By considering the sum of distances from all the cluster centers we choose images that are most dissimilar to the rest of the images in the database. In other words, we consider those images to be more interesting that have fewer images similar to them. This is visually explained in Fig. 5.4. A possible drawback of this measure of database saliency is that some neighboring images of a cluster can have similar values. To ensure that they father images chosen are dissimilar to each other, we could also consider in computing the database saliency value mutual distances between the images ranked as most salient.

Our method can also be used for an application like video summarization using keyframe extraction. In this case however, we would choose images that are *closest* to the cluster centers rather than farther away.

5.2 Choosing images for creating a summary

In this chapter we present two types of image summaries: mosaics and collages. In both cases, we create a composite image that consists of several individual images. To begin with, the user specifies the number of images k needed to create a summary. This is the number of clusters formed from the images in the database. Instead of just choosing the top k images from the ranked list to create a summary, we can also create such summaries based on a template image provided by the user (e.g. Fig. 5.11 and Fig. 5.12). In this case, we also need to make sure that the image picked from the list of ranked images matches the region of the template image that should be occupied by the chosen image. To find such a match, we compute the Euclidean distance between the average CIELAB vectors of all the pixels in the region and that of the image. This leads to another list of images in decreasing order of similarity with the template image region. We introduce an affinity function C_w that takes into account two rank based scores - the one resulting from the database saliency rank, and the other resulting from similarity to the template image region.

A rank based score is computed as:

$$Rw = 1 - \frac{r}{r_{max}} \tag{5.5}$$

where r is the rank and r_{max} is the largest value of such a rank. The affinity function C_w assigns relative importance to database saliency ranking and matching an image to the CIELAB average pixel value of the image region. It is computed as follows.

$$C_w = \beta(Rw_{dbs}) + (1 - \beta)(Rw_{is}) \tag{5.6}$$

where Rw_{dbs} is the database saliency rank based weight, Rw_{is} is the image similarity rank based weight (corresponding to proximity of average CIELAB values of an image in the database and the region of the image to be matched), and β (chosen to be 0.5) is a constant that takes a value in the interval $[0, 1]$.

5.3 Image summaries

Creating image compositions like mosaics and collages from images is an interesting consumer application [14, 117]. The image mosaics we present use variable tile sizes, unlike conventional mosaics. We also present a way to create image collages that are created by joining images at their best seam cut.

5.3.1 Variable tile size image mosaics

Conventional image mosaics have same sized tiles that are occupied by the best matching image from a database. If the image tiles are big, the composite image appears blocky, while if the tiles are small, their content is hardly discernible. We create an image mosaic that assigns variable size image tiles based on a novel energy map decomposition technique.

We use the image saliency maps proposed in Section 3.2 as our energy maps (alternatively, an edge gradient map can also be used). We recursively subdivide the input image in four equal quadrants, subject to the energy contained within each quadrant of the corresponding energy map exceeding a threshold T_e. Each quadrant is then overlaid by the database saliency ranked image whose average CIELAB vector has the lowest Euclidean distance from the average CIELAB vector of the quadrant, according to the affinity function of Eq. 5.6. The original image and the chosen tile images are averaged to get the final mosaic. Some examples of how the final mosaic is created from the input image are presented in Fig. 5.5 to Fig. 5.8. In the mosaics shown, the base image is blended with the tiles. It is not necessary to perform this blending. This choice is made to produce visually pleasing results. In a real-world application such attributes can also be user-defined or assigned in accordance with user studies.

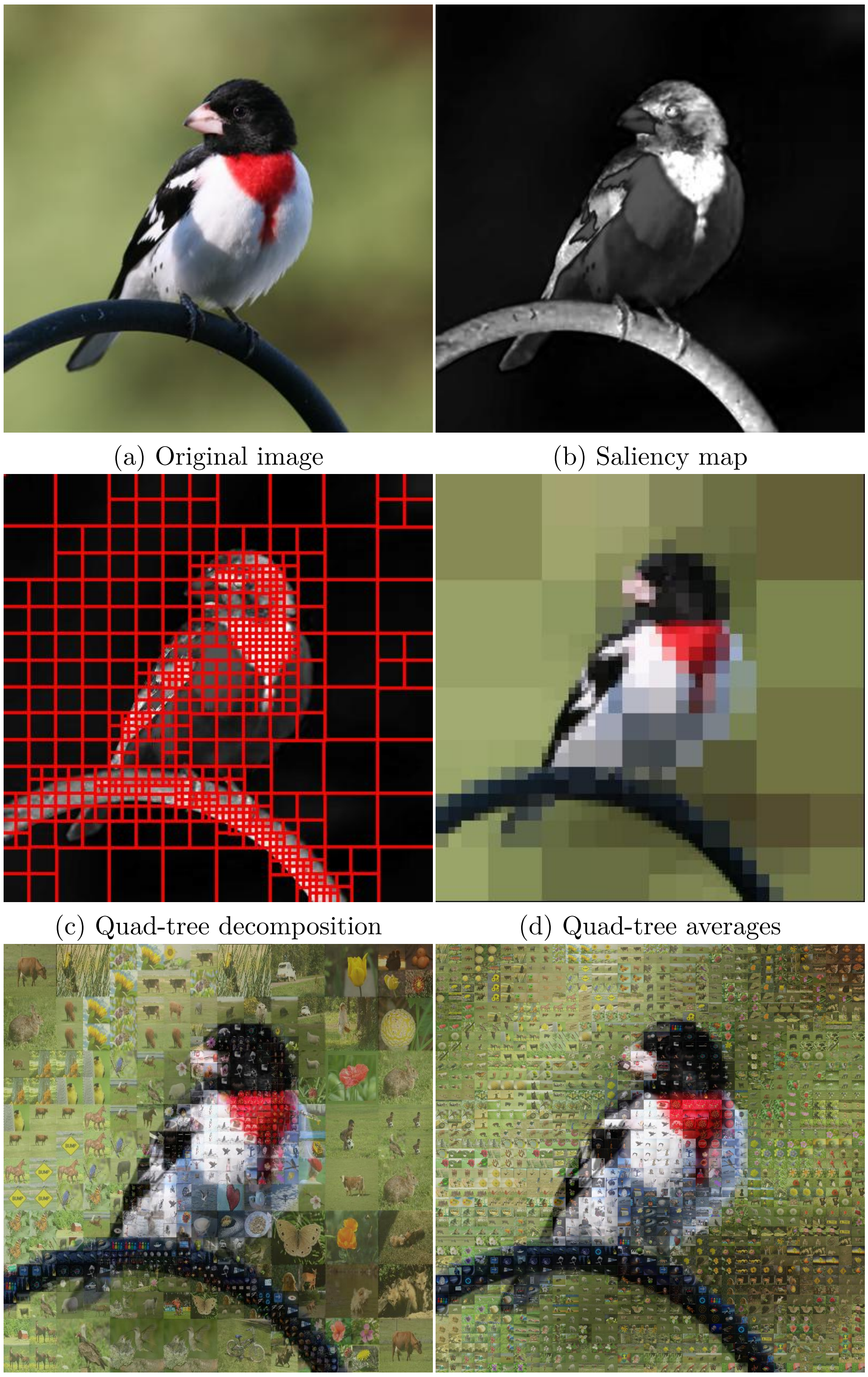
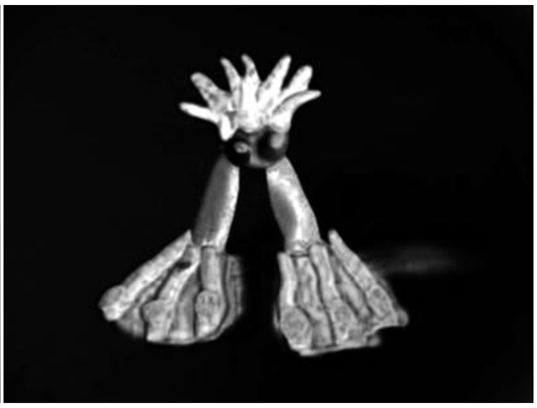
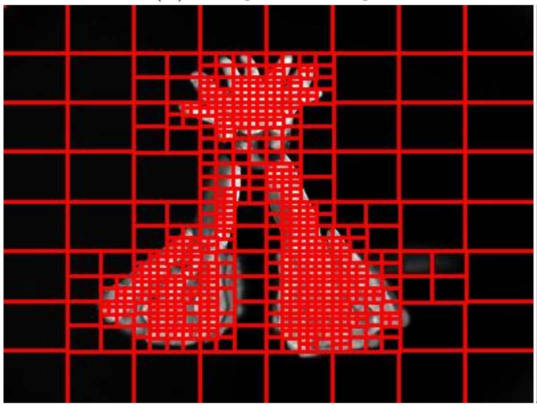
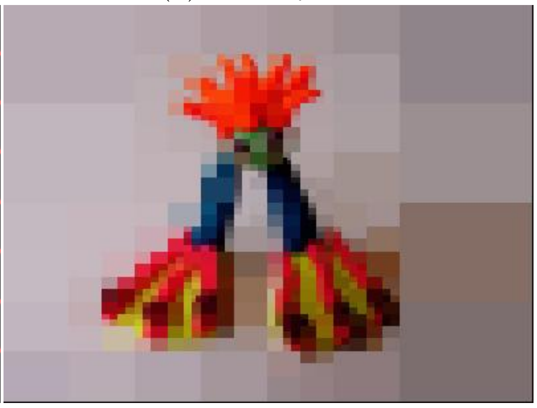

(a) Original image (b) Saliency map
(c) Quad-tree decomposition (d) Quad-tree averages
(e) Our mosaiced image (f) Conventional mosaic

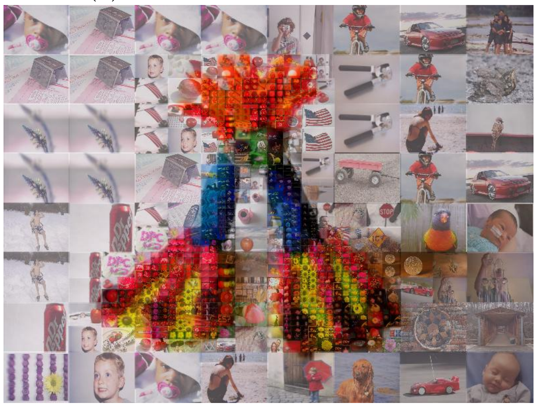
Figure 5.5: *Steps for creating our mosaiced image and comparison with a conventional mosaic.*

(a) Original image


(b) Saliency map


(c) Quad-tree decomposition


(d) Quad-tree averages


(e) Our mosaiced image

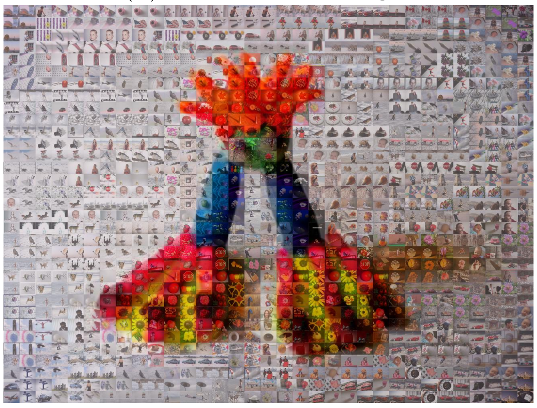
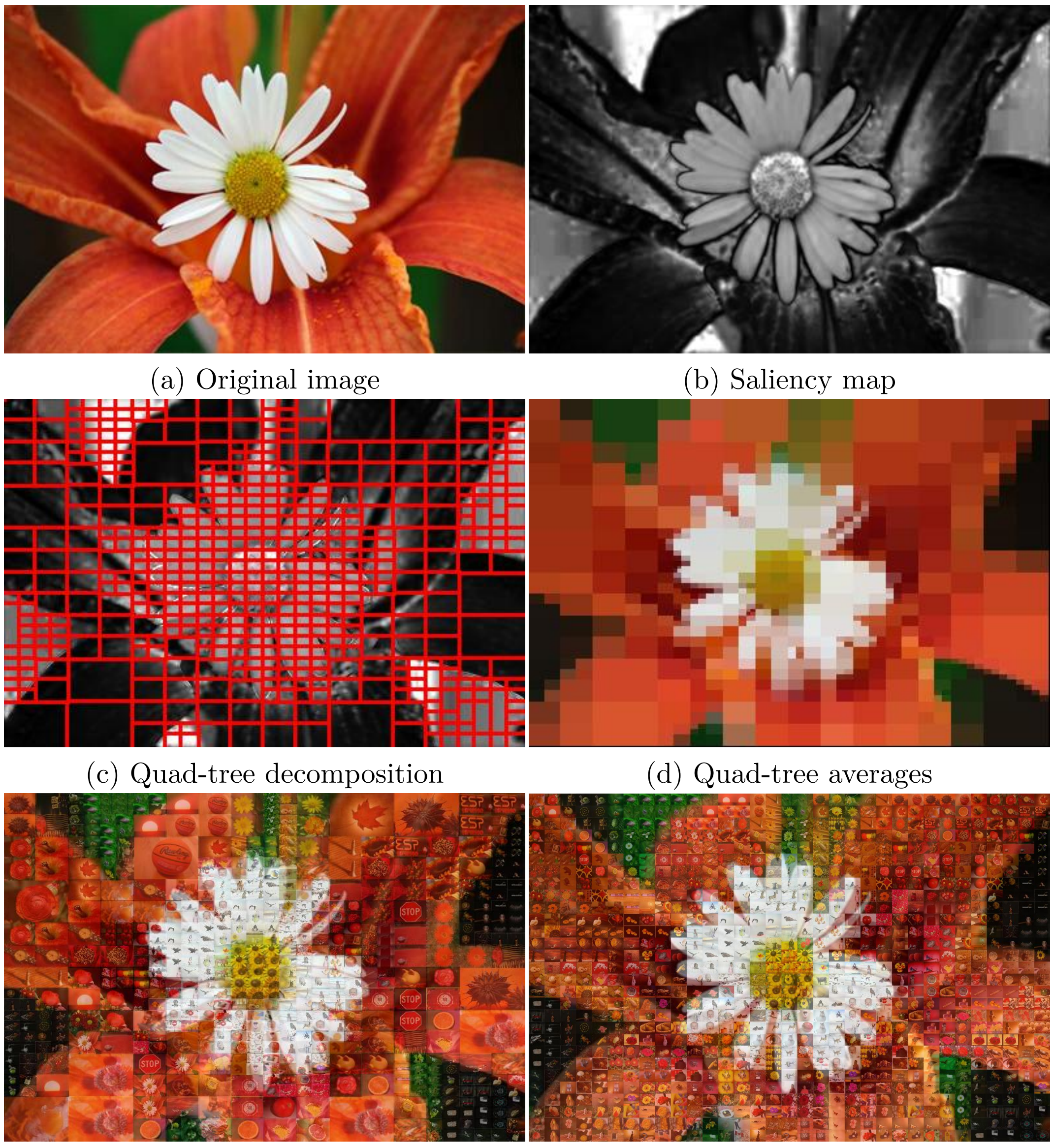

(f) Conventional mosaic

Figure 5.6: *Steps for creating our mosaiced image and comparison with a conventional mosaic.*



(a) Original image

(b) Saliency map

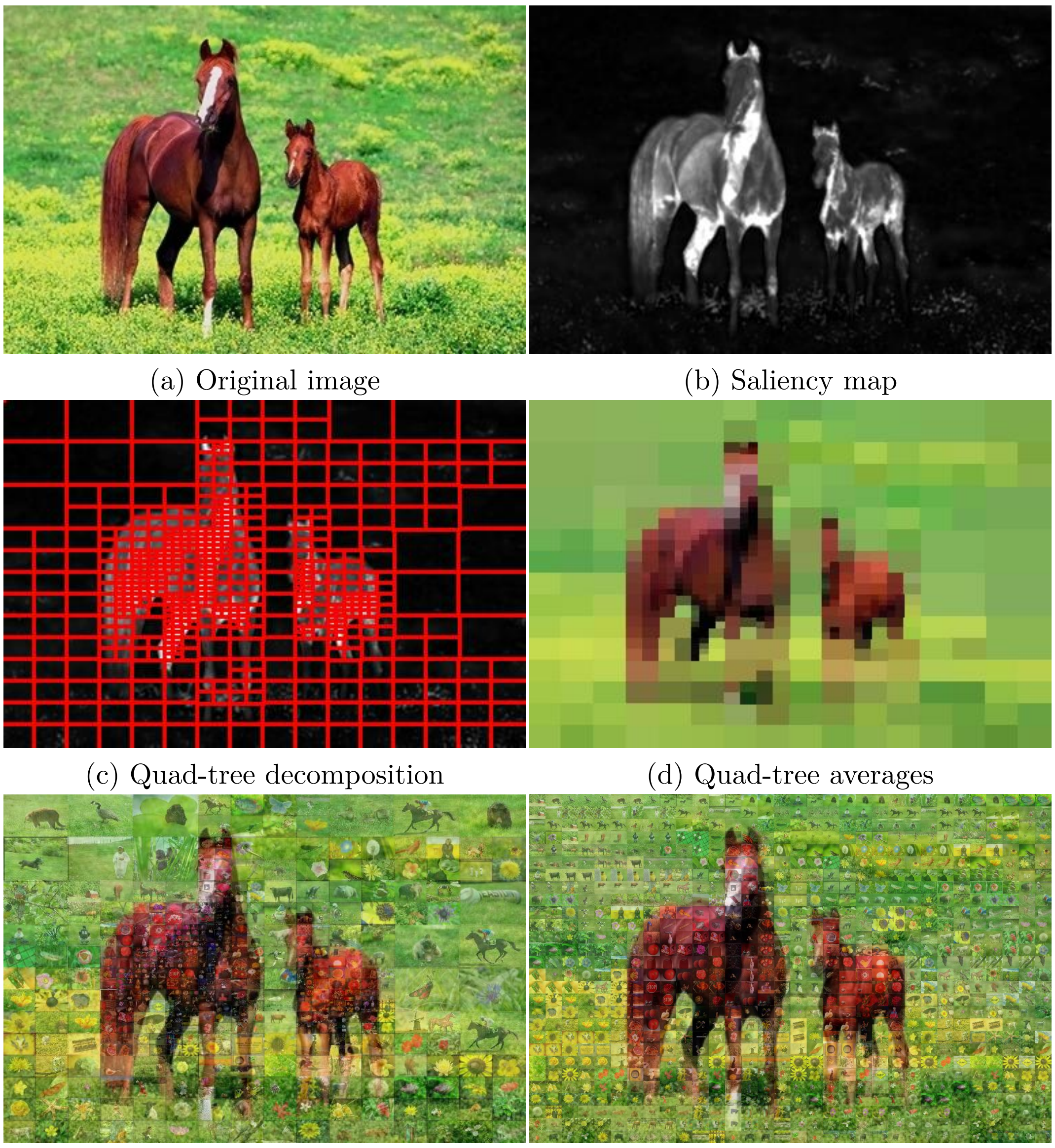
(c) Quad-tree decomposition

(d) Quad-tree averages

(e) Our mosaiced image

(f) Conventional mosaic

Figure 5.7: *Steps for creating our mosaiced image and comparison with a conventional mosaic.*



(a) Original image

(b) Saliency map

(c) Quad-tree decomposition

(d) Quad-tree averages

(e) Our mosaiced image

(f) Conventional mosaic

Figure 5.8: *Steps for creating our mosaiced image and comparison with a conventional mosaic.*

5.3.2 Automatic image collages

In order to create a collage as in Figs. 5.11(b) and 5.11(d) we first stitch together rows of salient images chosen by our database saliency technique, and then vertically stitch the row compositions. To stitch any two adjacent images, we choose the optimal cut between the two images in their overlap region. This is illustrated in Fig. 5.9. The method of finding the best cut to stitch the two images at their overlapping

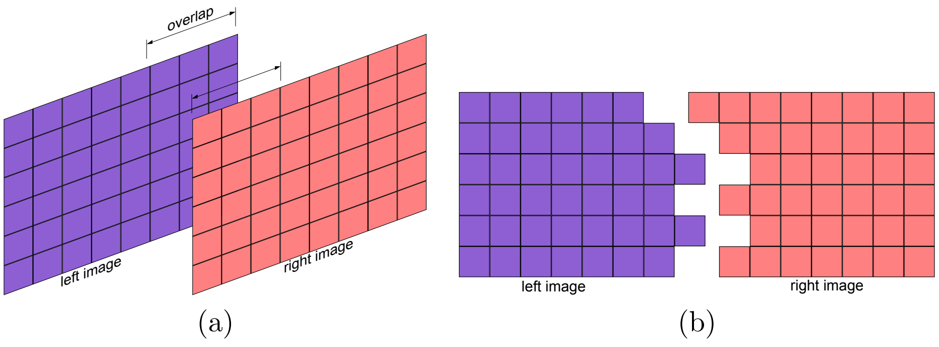


Figure 5.9: *Joining two adjacent overlapping images. (a) The overlapping region. (b) The optimal joint found in the overlapping region.*

region is a modified form of the dynamic programming based method presented in Section 4.2.2. To join two images, we take into account the saliency maps of both the images as well as the energy of the edge arising out of the seam-joint of the two images. For vertically joining two images on the left and the right, the dynamic programming memoization table entry $M(x, y)$ is given as:

$$M(x,y) = S_l(x,y) + S_r(x,y) + \min \begin{cases} \|\mathbf{I}_l(x-1,y-1) - \mathbf{I}_r(x,y-1)\| + M(x-1,y-1) \\ \|\mathbf{I}_l(x,y-1) - \mathbf{I}_r(x+1,y-1)\| + M(x,y-1) \\ \|\mathbf{I}_l(x+1,y-1) - \mathbf{I}_r(x+2,y-1)\| + M(x+1,y-1) \end{cases} \tag{5.7}$$

where E_l is the saliency of the left image, E_r is the saliency of the right image, and $\|.\|$ is the L_2 norm that takes into account the color and intensity edge energy of the left and right pixels at the joint. The (x, y) coordinates in Eq. 5.7 refer to the pixel positions relative to the top-left corner of the overlapping region of the two images.

Fig. 5.10 shows an example of a row composition. After stitching two images at the optimal joint, slight smoothing is performed to remove any edge artifacts. In the resulting collage *image saliency* helps stitch images without altering salient content while *database saliency* helps in choosing the most interesting images. In our mosaics, we first stitch several images along a row. Then several such row compositions are stitched vertically along their overlap region (using the same technique as used for individual images) to obtain the final composition.

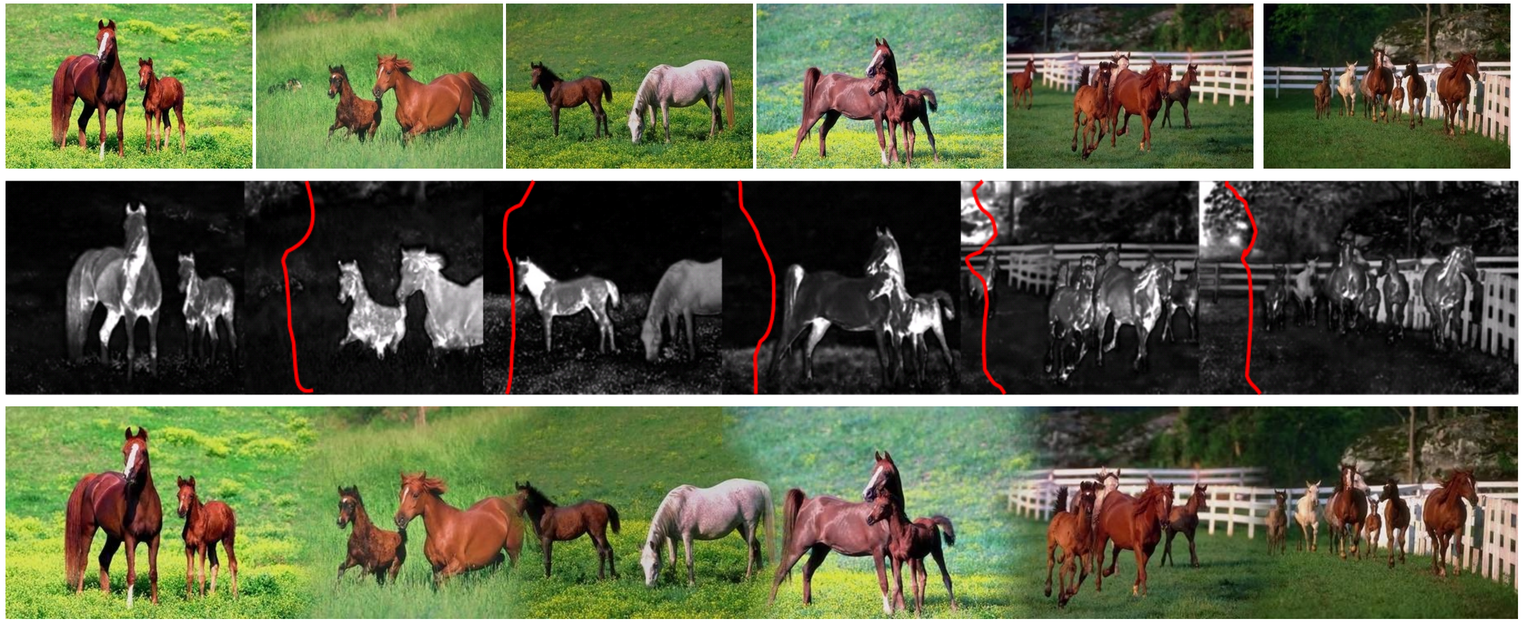


Figure 5.10: *The first row shows Original images. The second row shows their corresponding saliency maps. The third row shows the stitched and blended image that preserves salient regions and at the same time joins images at their lowest energy seam to minimize artifacts.*

5.4 Summary of the chapter

We presented a computationally feasible approach for summarizing a database and two novel image presentation schemes. While we take into account only the color content of the images into account using ISWCCV's, there are a lot of other aspects of the image's content (e.g. the objects, the aesthetic aspects, etc.), which can be taken into account using more sophisticated features, for assigning an 'interestingness' value to it. This may require additional computer vision techniques as well as extensive user studies.

The number of clusters created is dependent on the number of images required to create a summary in the form of a mosaic or a collage. So, the way database saliency is computed is dependent on the application. Although the method is fast enough to take into account a few thousand images, the computational efficiency depends on the size of the user's database and the number of images needed for summarization.

In the case of collages the goal is not necessarily adhering to a template image structure faithfully (see Fig. 5.11 and Fig. 5.12). A template image is used to influence the choice placement of images since visually it produces a more pleasing collage as opposed to placing images randomly or in order according to the database saliency rank.

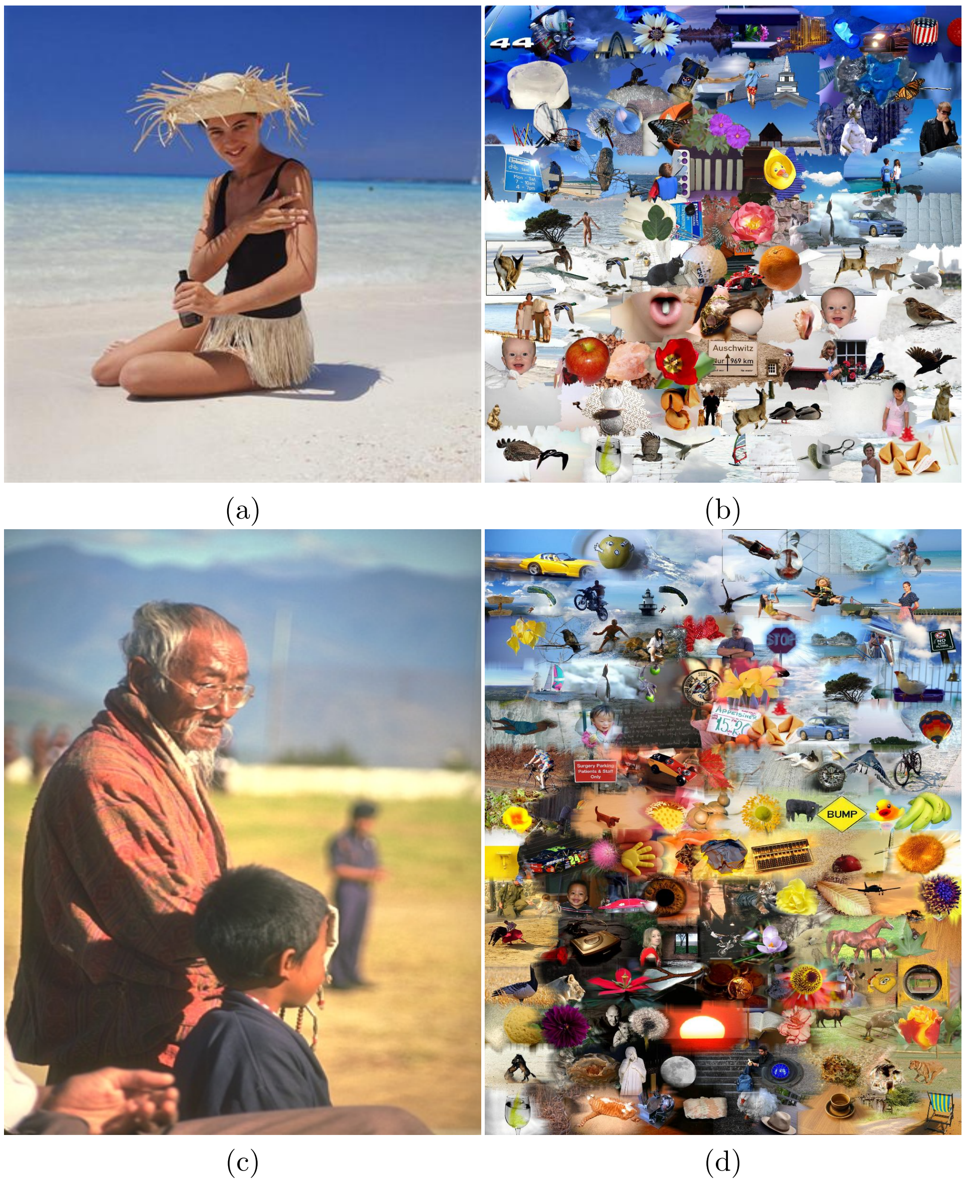


Figure 5.11: *Examples of collages of database saliency ranked images that are based on a template image. The collage (b) does not use any smoothing across the image stitch while collage (d) does.*

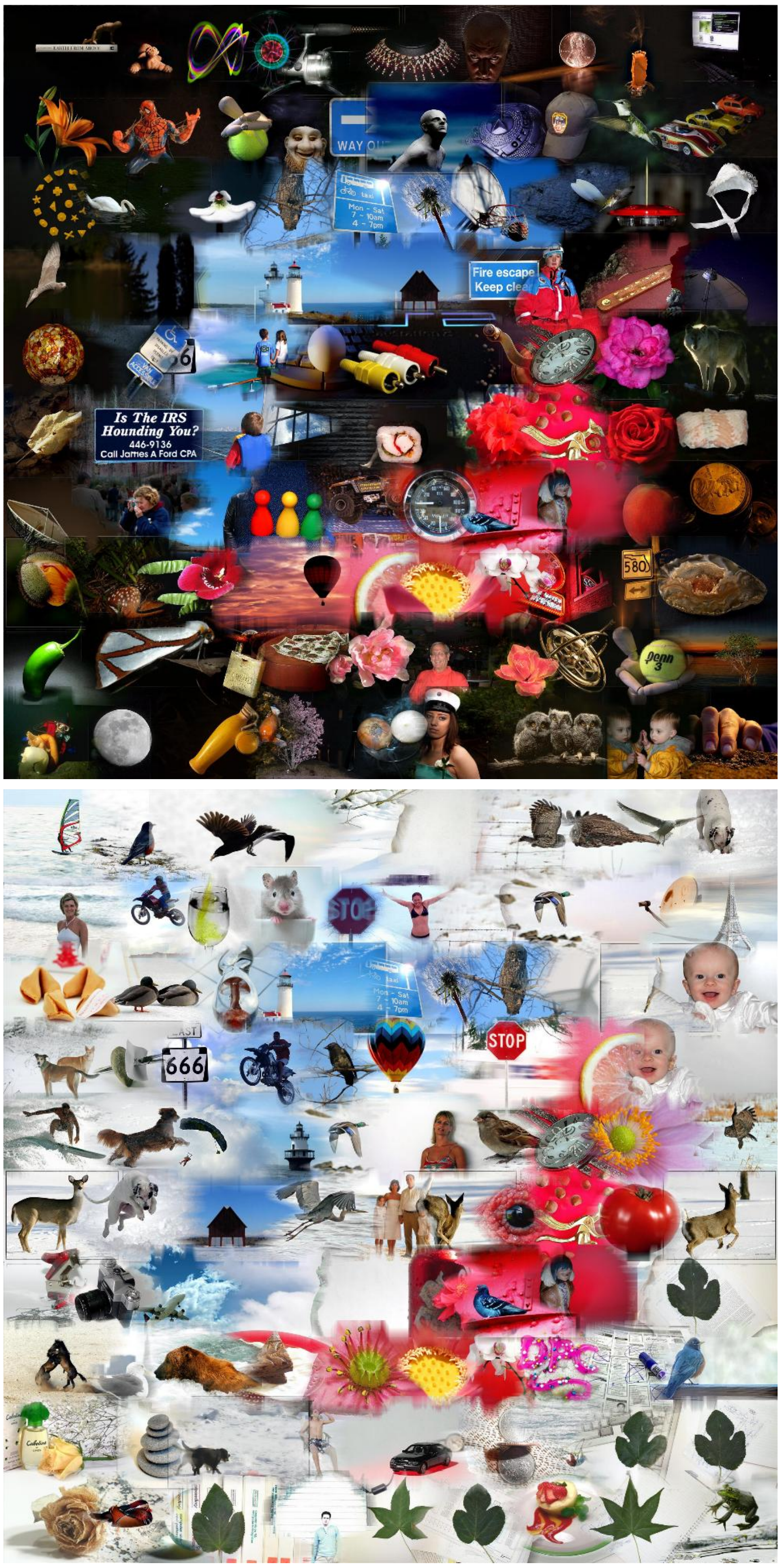


Figure 5.12: *More examples of collages of database saliency ranked images that are based on a user provided template image (not shown).*

Chapter 6

Superpixel Segmentation

In this chapter we present a technique for superpixel segmentation. We compare our algorithm with the state-of-the-art in terms of under-segmentation error and boundary recall. Then we show applications of superpixel segmentation in object class identification and mitochondria detection[1]. We show how our algorithm can be extended to supervoxel segmentation. Finally, we present an extension of our method to generate visually pleasing superpixels.

6.1 Introduction

Superpixels are clusters of spatially connected pixels in an image that share similar properties. Superpixels are generated when an image is oversegmented, i.e when more segments are generated than what may define whole object regions. Aggregating image data into superpixels preserves natural image boundaries while capturing redundancy in the data [113]. Furthermore, superpixels provide a convenient primitive from which to compute local image features. They capture redundancy in the image [113] and greatly reduce the complexity of subsequent image processing tasks. They have proved increasingly useful for applications such as depth estimation [65], image segmentation [86, 63], skeletonization [82], body model estimation [107], and object localization [52].

For superpixels to be useful they must strictly adhere to object boundaries, provide control over the number of superpixels, output regularly sized compact superpixels, be computationally efficient, and easy to deploy. Unfortunately, most state-of-the-art superpixel methods do not meet all these requirements. As we will demonstrate, they often suffer from a high computational cost, poor quality segmentation, inconsistent size and shape, or contain multiple, difficult-to-tune parameters.

The approach we advocate in this work, while strikingly simple, addresses these issues and produces high quality, compact, nearly uniform superpixels more efficiently than state-of-the-art methods [45, 121, 82, 134]. The algorithm we propose,

[1]The material presented in this chapter is also accessible in the papers [7] and [97]

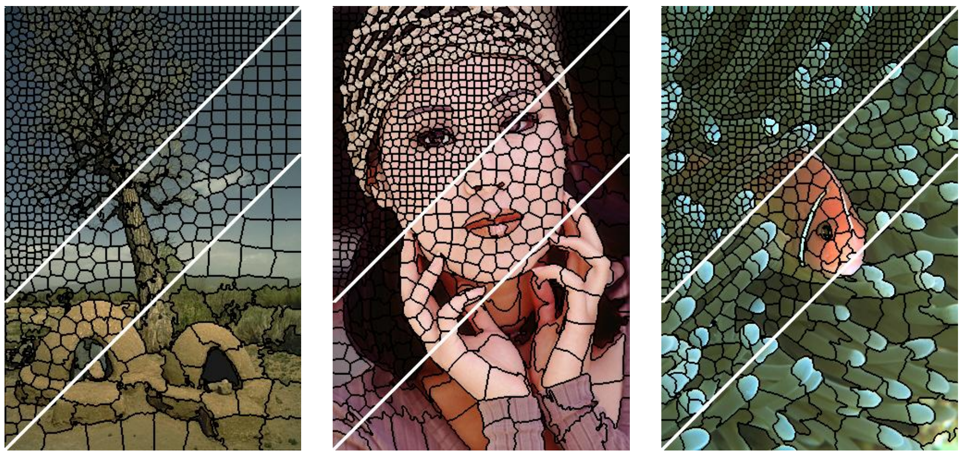


Figure 6.1: *Images segmented using our algorithm into superpixels of (approximate) size 64, 256, and 1024 pixels. The superpixels are compact, uniform in size, and adhere well to region boundaries.*

simple linear iterative clustering (SLIC) performs a local clustering of pixels in the five-dimensional *labxy* space where $[l\ a\ b]^{\mathrm{T}}$ are vectors of the CIELAB color space and the $[x\ y]^{\mathrm{T}}$ are pixel coordinates vectors in the image plane. A novel distance measure enforces compactness and regularity in the superpixel shapes, and seamlessly accomodates grayscale as well as color images. SLIC is simple to implement and easily applied in practice – the only parameter specifies the desired number of superpixels. Experiments on the Berkeley benchmark dataset [102] show that SLIC is significantly more efficient than competing methods, while producing segmentations of similar or better quality as measured by standard boundary recall and under-segmentation error measures.

For many vision tasks, compact and highly uniform superpixels that respect image boundaries, such as those generated by SLIC in Fig. 6.1, are desirable. For instance, graph-based models such as Conditional Random Fields (CRF) can see dramatic speed increases when switching from pixel-based graphs to superpixels [86, 52], but loose or irregular superpixels can degrade the performance. This is because local features such as SIFT extracted from the image at superpixel locations become less meaningful and discriminative if the superpixels are loose or irregular, and learning statistics over cliques of two or more superpixels can be unreliable. This effect can be seen when we compare the performance of SLIC superpixels to competing methods for two vision tasks: object class recognition and medical image segmentation. In both cases, our approach results in similar or greater performance at a lower computational cost in comparison to existing methods.

6.2 SLIC superpixel algorithm

Our approach generates superpixels by clustering pixels based on their color similarity and proximity in the image plane. This is done in the five-dimensional *labxy* space, where $[l\ a\ b]^{\mathrm{T}}$ is the pixel color vector in CIELAB color space, which is widely considered as perceptually uniform for small color distances, and $[x\ y]^{\mathrm{T}}$ represents the pixel position in the image plane. While the maximum possible distance between two colors in the CIELAB space (assuming sRGB input images) is limited, the spatial distance in the xy plane depends on the image size. It is not possible to simply use the Euclidean distance in this five-dimensional space without normalizing the spatial distances. In order to cluster pixels in this five-dimensional space, we therefore introduce a new distance measure that considers superpixel size. Using it, we enforce color similarity as well as pixel proximity in this five-dimensional space such that the expected cluster sizes and their spatial extents are approximately equal.

6.2.1 Distance measure

The distance measure we propose combines spatial and color similarity distances. If spatial pixel distances outweigh pixel color similarities, it will result in superpixels that do not respect region boundaries, only proximity in the image plane. Therefore, Instead of using a simple Euclidean norm in the five-dimensional space, we used a normalized and weighted sum of color and spatial distances.

$$\begin{aligned} d_{lab} &= (l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2 \\ d_{xy} &= (x_j - x_i)^2 + (y_j - y_i)^2 \\ D' &= \sqrt{\left(\frac{w_{lab}}{N_{lab}}\right)^2 d_{lab} + \left(\frac{w_{xy}}{N_{xy}}\right)^2 d_{xy}}, \end{aligned} \tag{6.1}$$

where w_{lab} and w_{xy} are the weights given to the color similarity and spatial proximity if one wishes to emphasize one distance over another, respectively, and N_{lab} and N_{xy} are the color similarity and spatial proximity normalization factors, respectively. The normalization factors are the maximal color and spatial distances within a cluster.

Our algorithm SLIC takes as input a desired number of approximately equally-sized superpixels k. For an image with N pixels, the approximate size of each superpixel is therefore N/k pixels. For roughly equally sized superpixels there would be a superpixel center at every grid interval $S = \sqrt{N/k}$. At the onset of our algorithm, we choose k superpixel cluster centers $C_i = [l_i\ a_i\ b_i\ x_i\ y_i]^{\mathrm{T}}$ with $i = [1, k]$ at regular grid intervals S.

The spatial normalization factor is in fact the grid step distance S since the algorithm tries to group pixels into a circle of diameter S in a superpixel. That is, the spatial maximum distance between two pixels is expected to be no bigger than S. The color normalization factor, however, depends on the pixels constituting the superpixel and could vary from one superpixel to another. This is fixed to a constant heuristically based on the color space used and the needs of the application. By replacing $D'\frac{N_{lab}}{w_{lab}}$ by D we can rewrite Eq. 6.1 as:

$$D = \sqrt{d_{lab} + \left(\frac{N_{lab} \times w_{xy}}{N_{xy} \times w_{lab}}\right)^2 d_{xy}}, \tag{6.2}$$

which simplifies to

$$D = \sqrt{d_{lab} + \frac{m^2}{S^2} d_{xy}}, \tag{6.3}$$

by replacing N_{xy} by S, and $\frac{N_{lab} \times w_{xy}}{w_{lab}}$ by a constant m, which serves to vary the relative weight between color similarity and spatial proximity. Eq. 6.3 is the distance measure we use in practice. The greater the value of m, the more spatial proximity is emphasized and the more compact the cluster. This value can be in the range $[1, 20]$ for CIELAB color space.

6.2.2 Algorithm

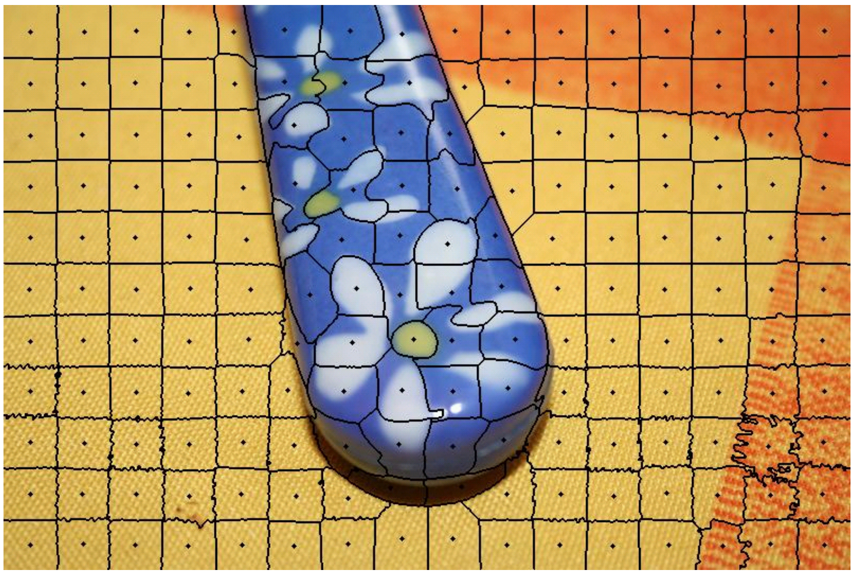
The *Simple Linear Iterative Clustering* (SLIC) algorithm is summarized in Algorithm 6.1. We begin by sampling k regularly spaced cluster centers and moving them to seed locations corresponding to the lowest gradient position in a 3×3 neighborhood. This is done to avoid placing them at an edge and to reduce the chances of choosing a noisy pixel. Image gradients are computed as:

$$G(x, y) = \|\mathbf{I}(x+1, y) - \mathbf{I}(x-1, y)\|_2^2 + \|\mathbf{I}(x, y+1) - \mathbf{I}(x, y-1)\|_2^2 \tag{6.4}$$

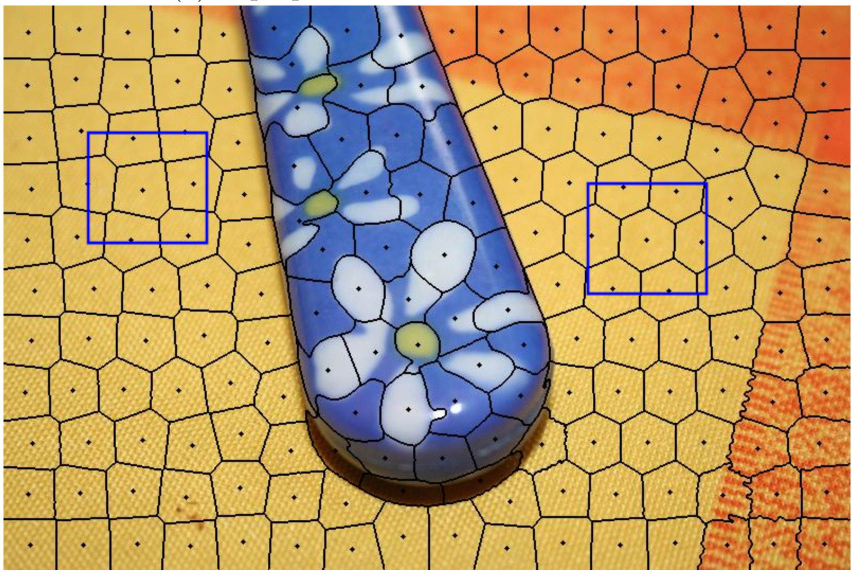
where $\mathbf{I}(x, y)$ is the *lab* vector corresponding to the pixel at position (x, y), and $\|.\|$ is the Euclidean L_2 norm as before. This takes into account both color and intensity information.

Each pixel in the image is associated with the nearest cluster center whose search area overlaps this pixel. The key advantage of SLIC over regular k-means clustering appears here. We note that the spatial extent of any superpixel is approximately S^2 (the approximate area of a superpixel). Since we enforce spatial proximity in Eq. 6.3, we can safely assume that pixels that are associated with a given cluster center lie within a $2S \times 2S$ area around the superpixel center on the xy plane. This then becomes the search area for the pixels nearest to each cluster center, drastically reducing the number of distances computed to determine cluster membership. This is unlike regular k-means clustering where the entire image plane would be considered for the search. This is explained in Fig. 6.2.

After all the pixels are associated with the nearest cluster center, a new center is computed as the average $[l\ a\ b\ x\ y]^{\mathrm{T}}$ vector of all the pixels belonging to the cluster. We then iteratively repeat the process of associating pixels with the nearest cluster center and recomputing the cluster center until convergence (i.e when residual



(a) Superpixels after 1 iteration of SLIC



(b) Superpixels after 10 iterations of SLIC

Figure 6.2: *Explanation for the speed-up achieved by SLIC over conventional k-means algorithm. (a) The image shows the superpixel clusters at the end of one iteration with centers (black dots) chosen at regular grid steps S. (b) The superpixels at the end of ten iterations. In this case the centers have moved and the clusters have taken their compact form such that the maximum expected spatial distance between two pixels is the grid step size S. The blue squares overlaid on the image show the search regions used by SLIC to determine cluster membership. SLIC computes distances from each cluster center to pixels within the search region of area $2S \times 2S$. This is unlike general k-means algorithm that computes the distances from each cluster center to all the pixels in the image. Since the search area, and hence the number of distance computations, is inversely related to the desired number of superpixels, SLIC not only reduces the number of distances computed but also makes the algorithmic complexity independent of the number of superpixels.*

difference between previous centers and recomputed centers is less than a threshold).

6.2.3 Post processing

At the end of superpixel clustering, a few stray labels may remain. These stray labels belong to pixels in the vicinity of a larger segment that have the same label as the larger segment but are not connected to it (in terms of 4 or 8-connectivity). While this occurs rarely, this may arise despite using xy-plane distances computing D (Eq. 6.3) since the k-means algorithm clusters pixels in the five-dimensional $labxy$-space without explicitly enforcing connectivity in the two-dimensional xy-plane. Our clustering process does not explicitly enforce connectivity. Hence, we enforce connectivity in the last step of our algorithm by relabeling disjoint segments with the labels of the nearest neighboring cluster using a connected components algorithm.

This step can be accomplished using a two-pass algorithm. In the first pass, a usual connected components algorithm is run, all components are assigned new component labels, and an adjacency graph is built. In the next pass, except the largest component, any component that shares the same original SLIC-assigned label is merged with the nearest adjacent component.

We however use a simpler version of this algorithm that uses only one pass and does not need the adjacency graph. We use a queue based flood-fill algorithm to connect SLIC-labeled pixels. During the component building step, as soon as we encounter a component smaller than one-fourth the required superpixel size, we merge it with the nearest previous component.

The complexity of the post-processing step depends on the number of unconnected 'stray' components and can vary from image to image. In practice, this process is linear in the number of image pixels and takes less than 10% of the total time required for segmenting an image.

Algorithm 6.1 Simple linear iterative clustering (SLIC)

```
1: Initialize cluster centers $C_i = [l_i\ a_i\ b_i\ x_i\ y_i]^{\mathrm{T}}$ by sampling pixels at regular grid
   steps $S$.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: repeat
4:    for each cluster center $C_i$ do
5:       Assign the best matching pixels from a $2S \times 2S$ square neighborhood around
         the cluster center according to the distance measure (Eq. 6.3).
6:    end for
7:    Compute new cluster centers and residual error $E$ {$L1$ or $L2$ distance between
      previous centers and recomputed centers}.
8: until $E \leq$ threshold
9: Enforce connectivity.
```

6.2.4 Complexity

It is easy to notice that the idea of iteratively evolving local clusters and cluster centers used in our algorithm is a special case of k-means adapted to the task of generating superpixels. Interestingly, by virtue of using our distance measure of Eq. 6.3, we are able to *localize* our pixel search to an area $(2S \times 2S)$ on the image plane that is inversely proportional to the number of superpixels k (see also Fif. 6.2). A pixel falls in the local neighborhood of no more than eight cluster centers. We also note that the convergence error of our algorithm drops sharply in a few iterations as shown in Fig. 6.3. Our experiments show that it suffices to run the algorithm for 4 to 10 iterations. We use 10 iterations for all the results in this chapter. The trivial

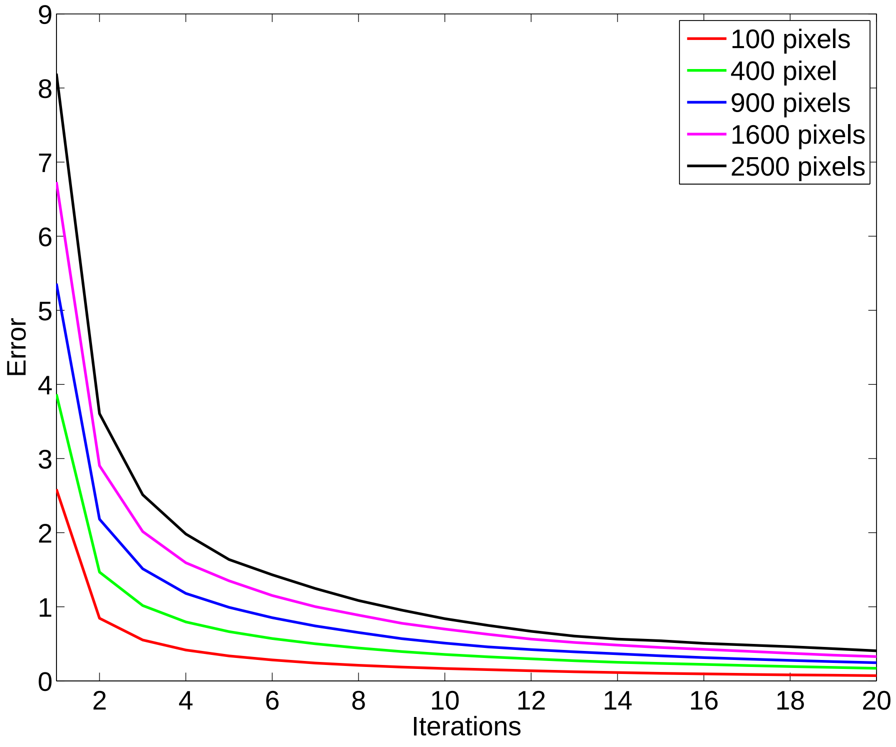


Figure 6.3: *Plot of the residual error E in step 7 of Algorithm 6.1 of clustering versus number of iterations of SLIC. The plot shows five different curves each for a different size of superpixels. As can be seen the error drops rapidly after the first few iterations.*

upper bound for the classical k-means algorithm [94] is $O(k^N)$, which was slightly improved by Inaba et al. [69] to $O(N^{kd})$ for d-dimensional space. The practical time complexity for the classical k-means algorithm is $O(NkI)$ [39] based on the number of distance computations, where N is the number of data points (pixels in the image), k is the number of clusters (or seeds), and I is the number of iterations required for convergence. Our method achieves $O(N)$ complexity (see Fig. 6.12), since we need to compute distances from any point to no more than eight cluster centers and the number of iterations is constant. Thus, SLIC is specific to the problem of superpixel segmentation, and unlike conventional k-means clustering, avoids several redundant distance calculations.

Speed-up schemes for the k-means algorithm have been proposed using prime number length sampling [135], random sampling [79], by local cluster swapping [75], and by setting lower and upper bounds [39]. However except for [39], these methods do no achieve linear complexity for a given k. SLIC, on the other hand, is linear in the number of pixels irrespective of k. This is shown in Fig. 6.4 where the comparison is made with the regular k-means clustering. Note that, like [39], SLIC implicitly sets bounds on distance calculations, but does not need to compute or carry forward these bounds for the subsequent iterations. Unlike most of these segmentation schemes, which are very general in nature, SLIC is specifically tailored to perform superpixel clustering using the distance measure of Eq. 6.3 and is much simpler.

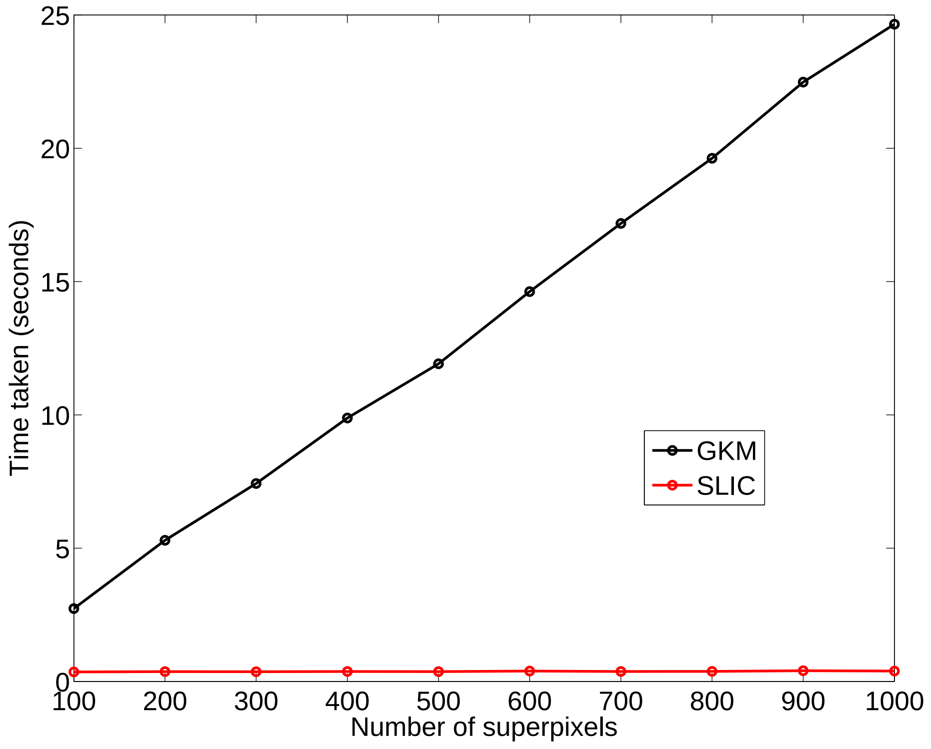


Figure 6.4: *Plot of the time taken for 10 iterations of general* k-means *(GKM) versus our algorithm for different number of superpixels on input images of size* 481×321*. Our algorithm takes less than 0.5 second for any superpixel size.*

6.3 Comparison

In this section we compare our superpixel segmentation method with four state-of-the-art algorithms, namely, GS04[2] [45], NC05[3] [107], TP09[4] [83], QS09[5] [52] using publicly available source codes. GS04 and NC05 are graph based methods while TP09 and QS09 are gradient based approaches. NC05 and TP09 are designed to output a desired number of superpixels while GS04 and QS09 require parameter tuning to obtain a desired number of superpixels. This choice of algorithms provides a good representation of the state-of-the-art.

Fig. 6.5 to Fig. 6.10 provides a visual comparison of our output with these algorithms. Fig.6.11 provides a quantitative comparison in terms of under-segmentation error and boundary recall measures, similar to the ones used by Levinshtein et al. [83] for this purpose, computed with respect to the Berkeley segmentation ground truth [102].

6.3.1 Algorithm Parameters

As mentioned in the introduction, it is important for superpixel algorithms to be easy to use. Parameters needed as input should be few in number and easy to set. Table 2.1 summarizes the number of parameters that must be tuned or set for each method. SLIC, like NC05 and TP09 requires a single parameter. It is also important to note that GS04 and QS09 do not allow the user to control the number of superpixels. We had to perform a search on the parameter space to be able to control the number of superpixels in order to make a fair comparison to the other methods.

6.3.2 Under-segmentation error

Under-segmentation error essentially measures the error an algorithm makes in segmenting an image when compared to a known ground truth (human segmented images in this case). This error is computed in terms of the ‘bleeding’ of the segments output by the algorithm when superposed over ground truth segments. This measure thus penalizes superpixels that do not tightly fit the limits of a ground truth segment.

Given ground truth segments $g_1, g_2, ..., g_M$ and a superpixel output $s_1, s_2, ..., s_L$, the under-segmentation error for a ground truth segment g_i is quantified as:

$$U = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{[s_j | s_j \bigcap g_i > B]} |s_j|\right) - N\right] \tag{6.5}$$

[2]`http://people.cs.uchicago.edu/~pff/segment/`

[3]`http://www.cs.sfu.ca/~mori/research/superpixels/`

[4]`http://www.cs.toronto.edu/~babalex/turbopixels_supplementary.tar.gz`

[5]`http://www.vlfeat.org/download.html`

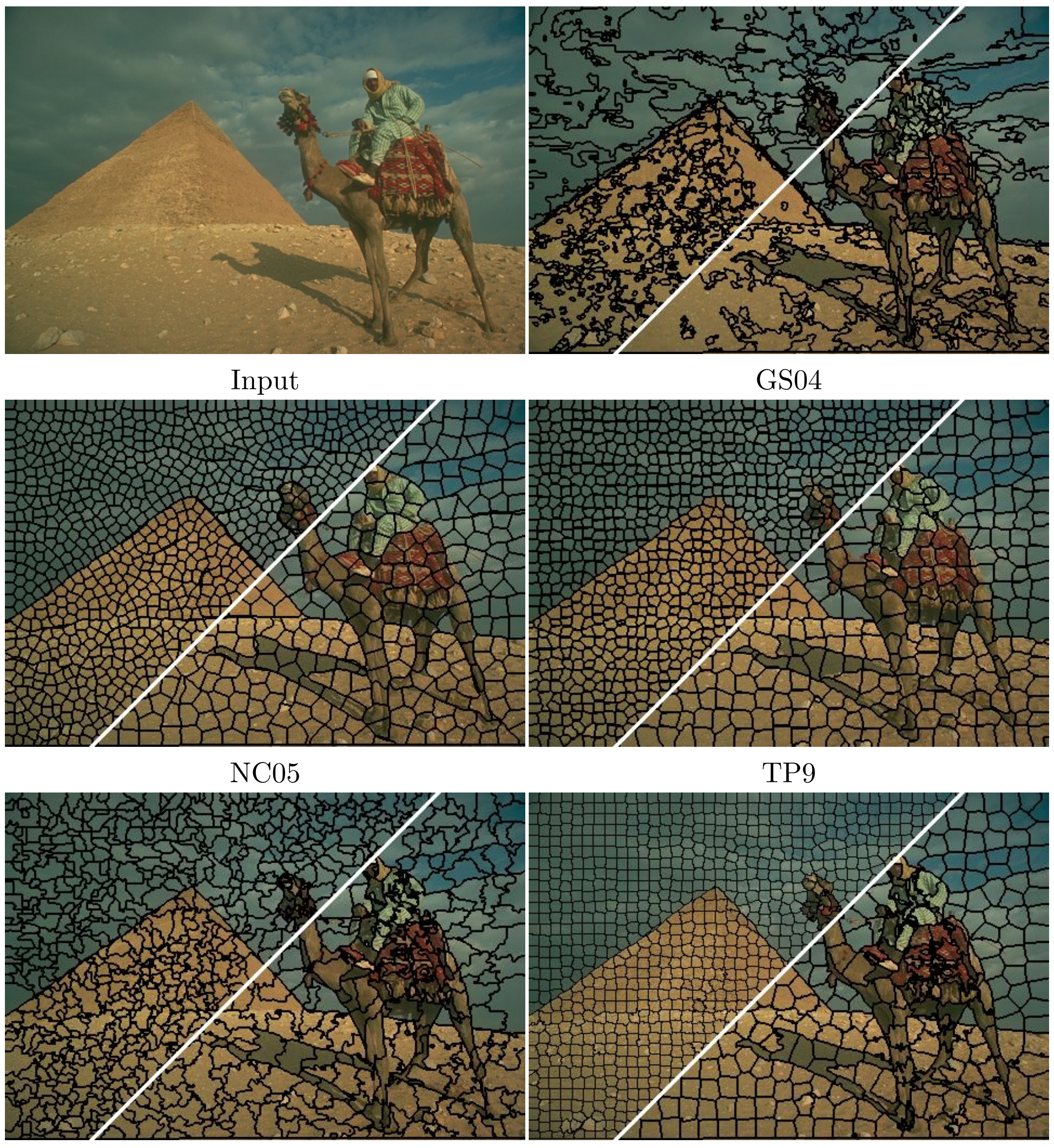


Figure 6.5: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

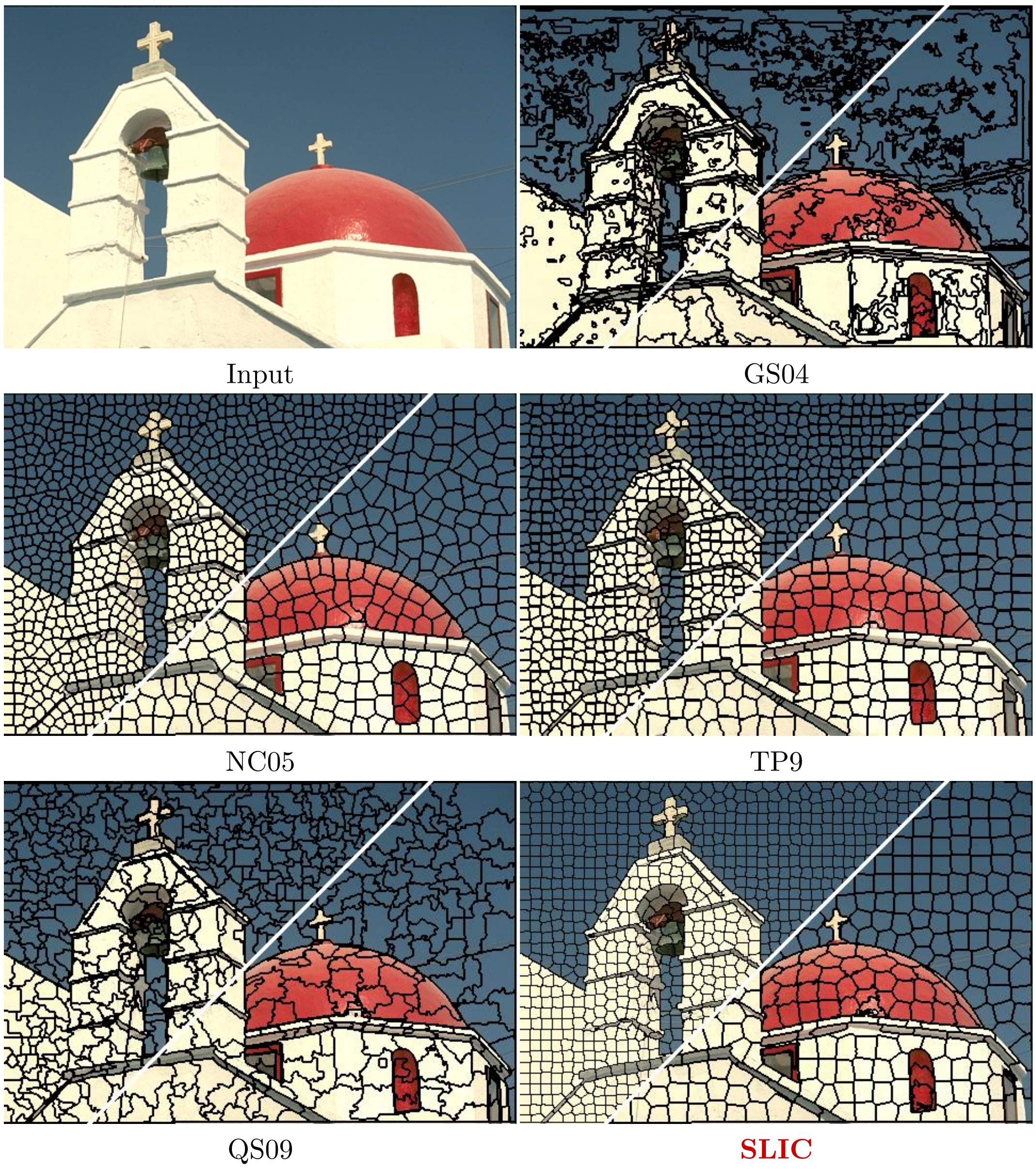


Figure 6.6: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

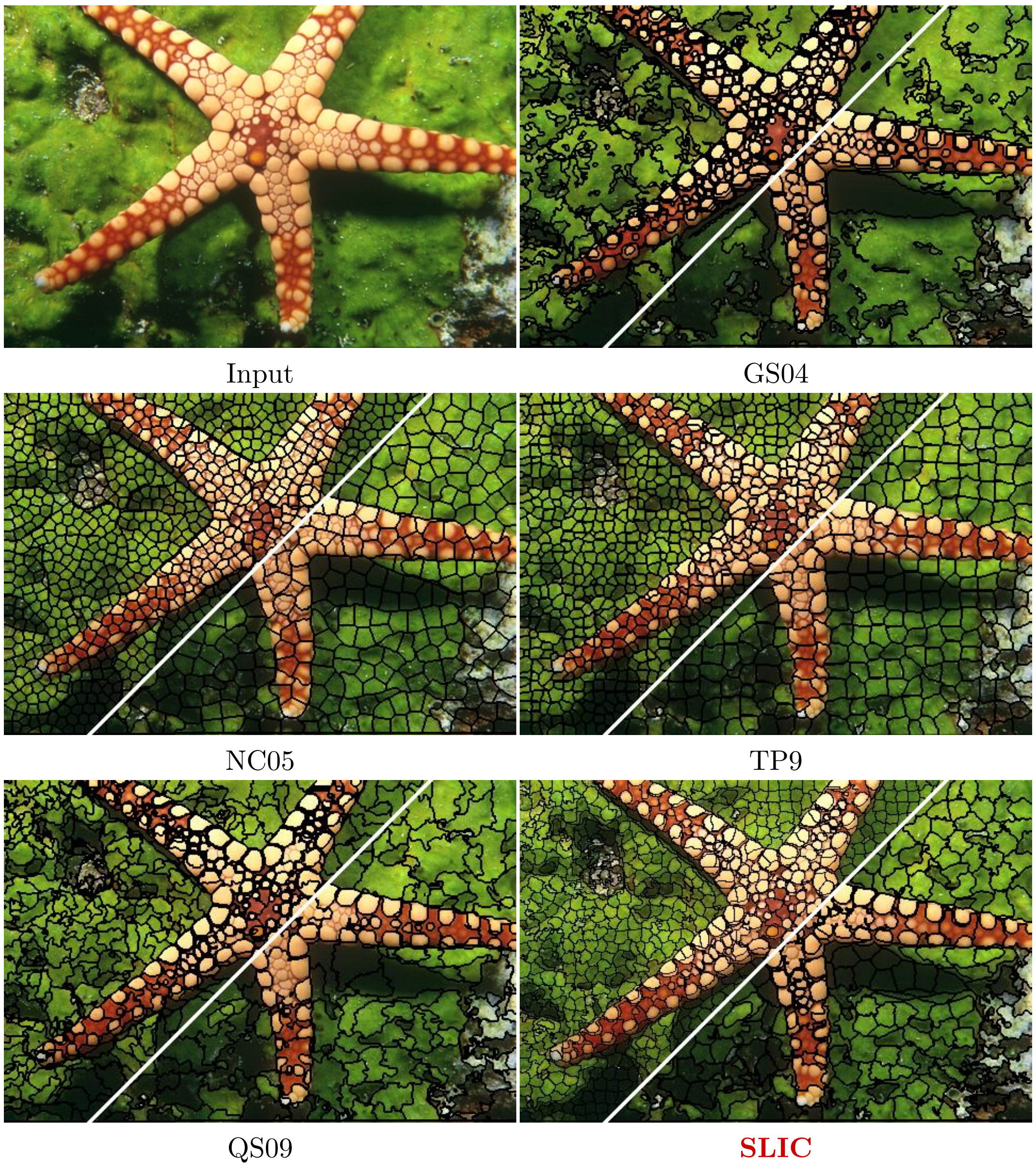


Figure 6.7: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

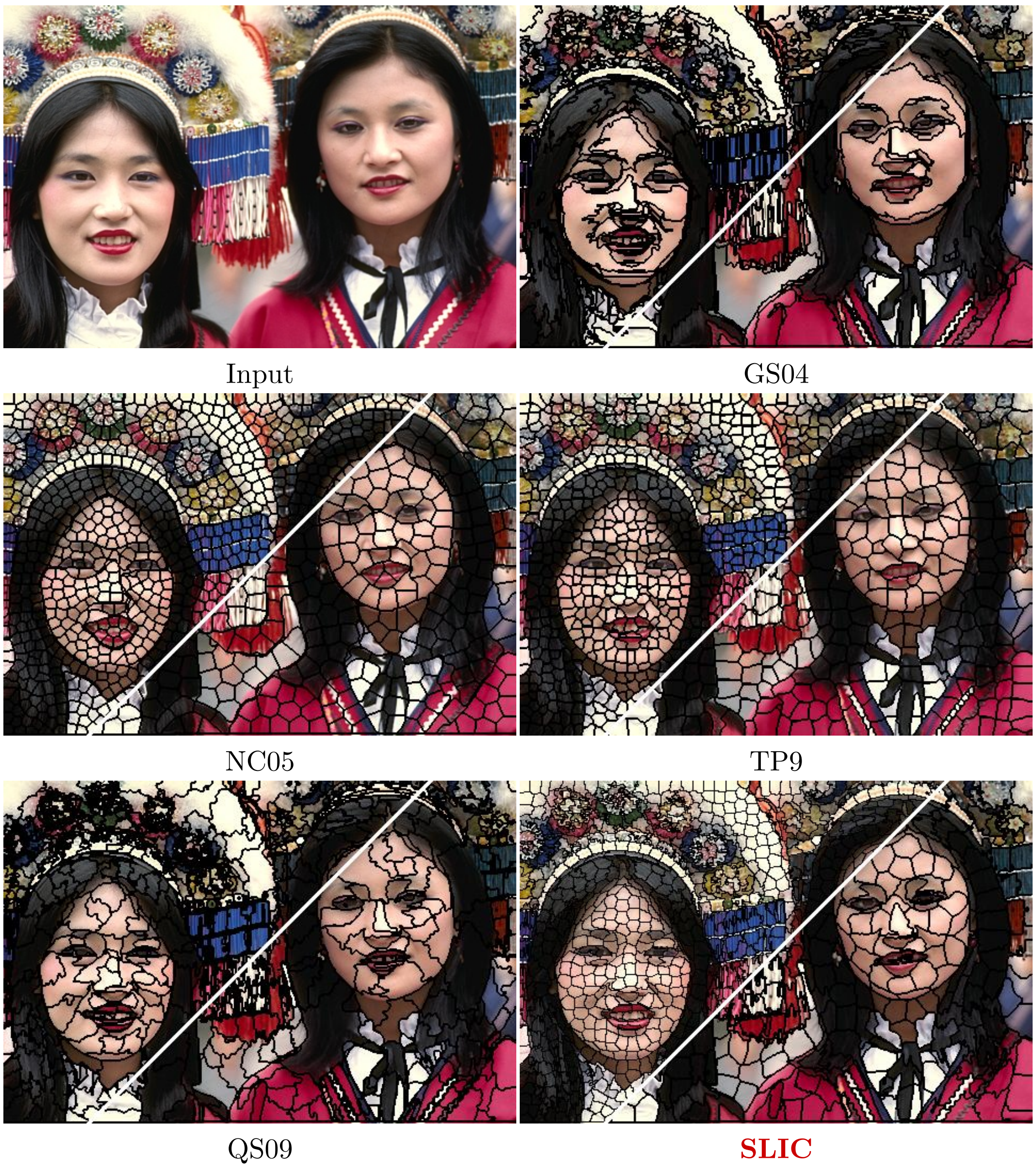


Figure 6.8: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

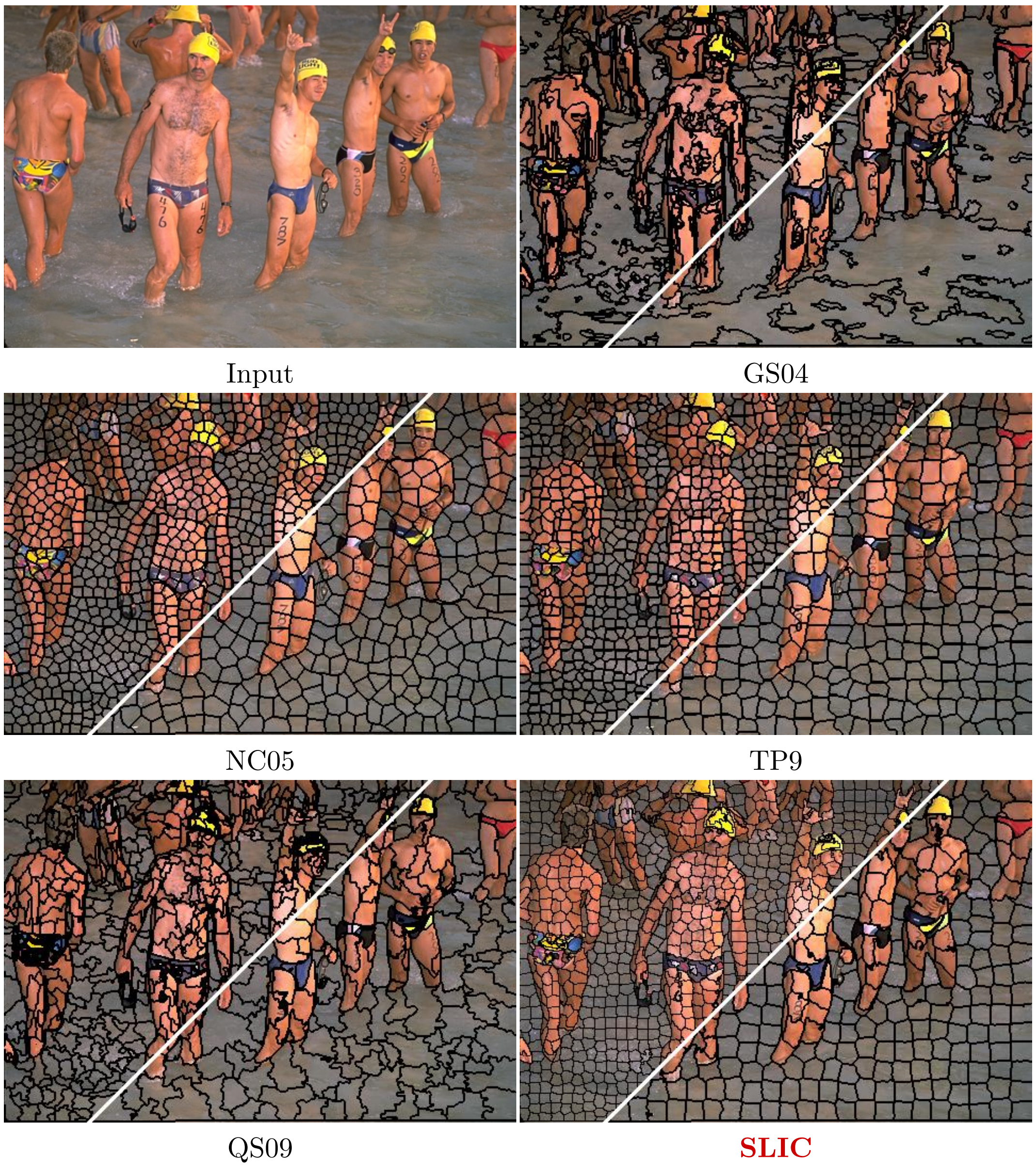


Figure 6.9: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

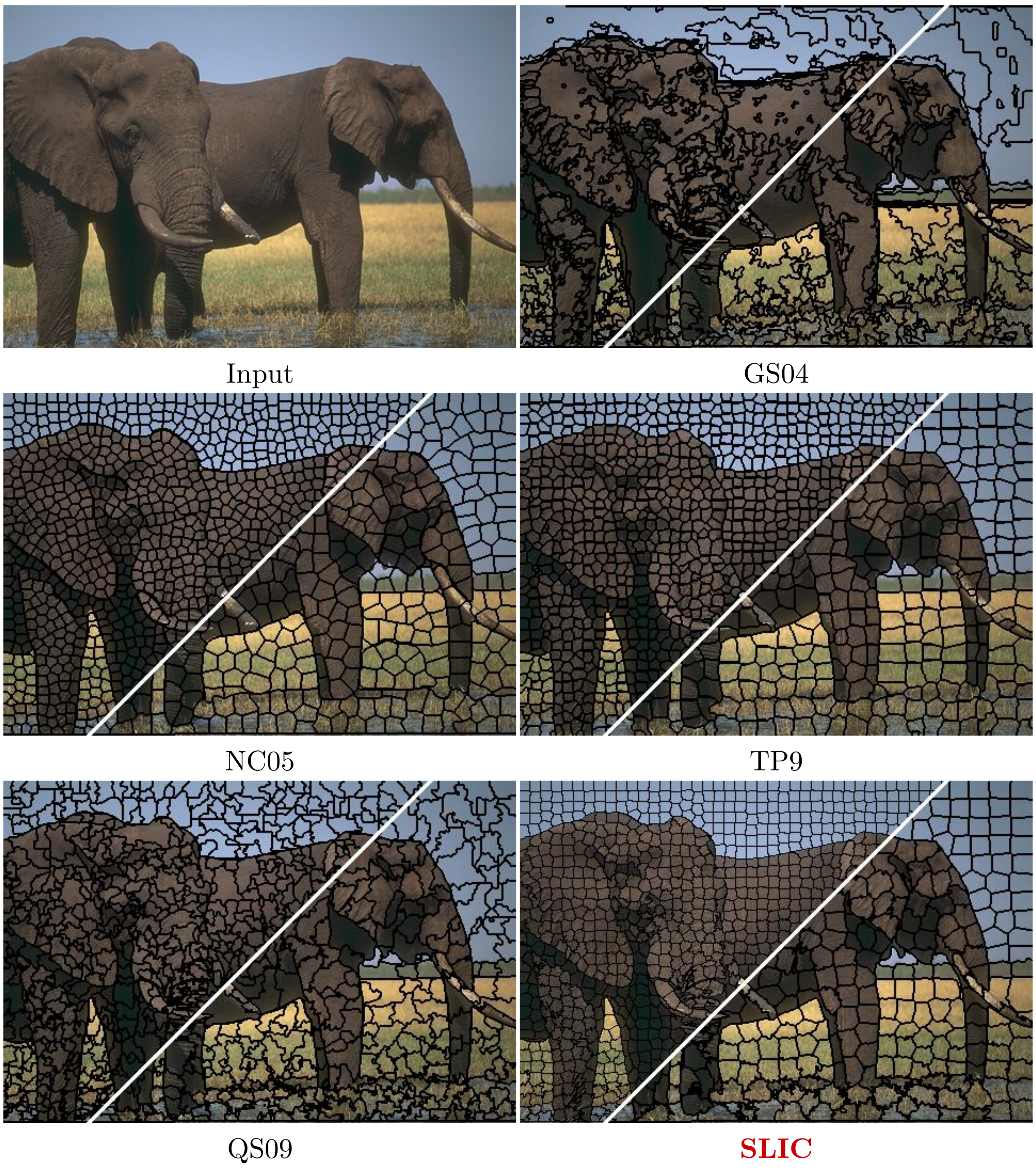


Figure 6.10: *Visual comparison of the superpixels from state-of-the-art algorithms. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each.*

where $|.|$ gives the size of the segment in pixels, N is the size of the image in pixels, and B is the minimum number of pixels that need to be overlapping. The expression $s_j \bigcap g_i$ is the intersection or overlap error of a superpixel s_j with respect to a ground truth segment g_i. B is set to be 5% of $|s_j|$ to account for small errors in ground truth segmentation data. The value of U is computed for each image of the ground truth and then averaged to obtain the graph in Fig. 6.11(a). As can be seen, SLIC shows the lowest under-segmentation error for all superpixel sizes.

6.3.3 Boundary recall

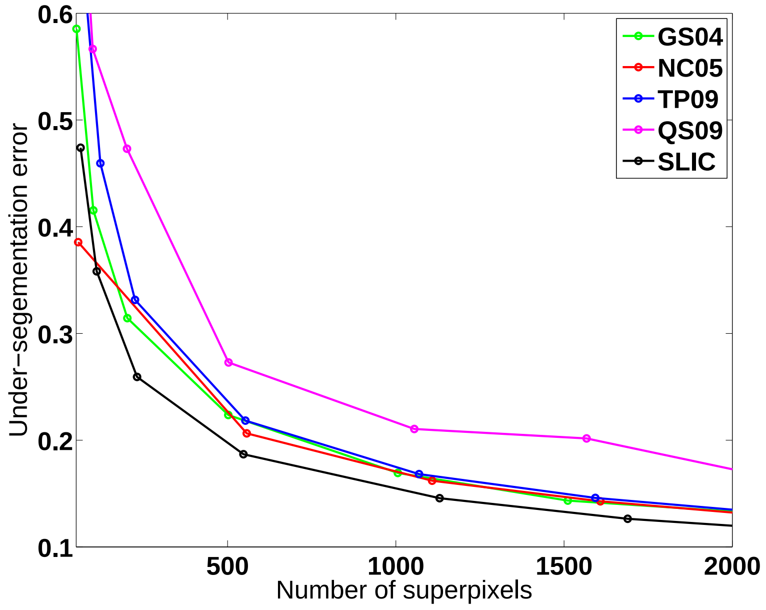
We adopt the standard boundary recall measure, which computes what fraction of ground truth edges fall within one pixel of a least *one* superpixel boundary. We use the internal boundaries of each superpixel. So in effect we search for a ground truth boundary in the margin of two pixels of the algorithm-generated boundaries. The boundary recall of each of the considered methods is plotted against the number of superpixels in Fig. 6.11(b). The boundary recall is second only to GS04 and is quite similar to QS09. It must be noted that GS04 places too many boundary pixels near segment boundaries resulting in a higher value of boundary recall as compared to other methods.

6.3.4 Computational and memory efficiency

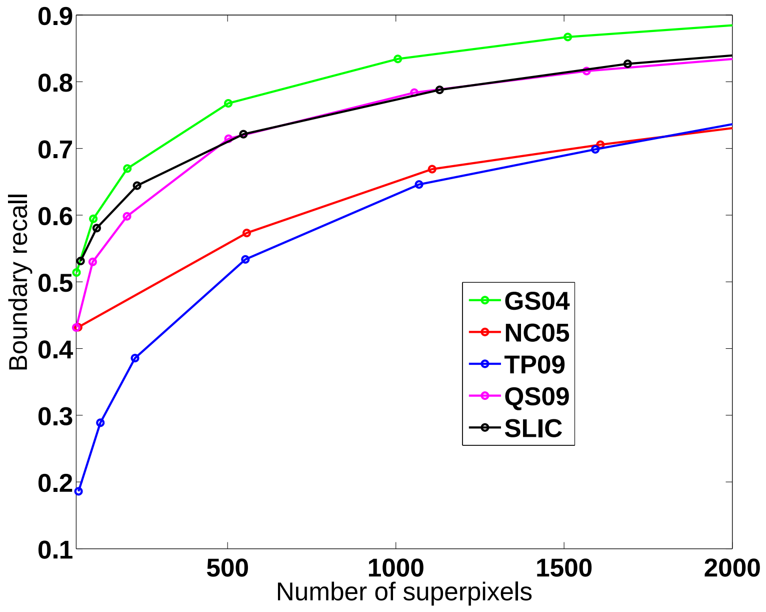
For images of size 480×320, SLIC is more than 10 times faster than TP09 and more than 500 times faster than NC05. What is encouraging is that it is even faster than GS04 for images greater than half a million pixels (see Fig. 6.12). This is because our algorithm always operates at $O(N)$ complexity while GS04 has $O(N \log N)$ complexity. This is of interest because even low end consumer digital cameras produce images exceeding 3 million pixels. GS04 requires $5 \times N$ memory to store edge weights and thresholds, as opposed to SLIC, which needs $1 \times N$ memory (to store the distance of each pixel from its nearest cluster center). Also, as mentioned before, if a search needs to be performed to obtain a given number of superpixels then GS04 and QS09 take more time in practice since they need to be run several times to get the desired output.

6.3.5 Discussion

A good superpixel segmentation algorithm should have low under-segmentation error as well as high boundary recall. To be useful as a pre-processing algorithm, such a segmentation should result in equally sized compact superpixels with control on its number. For the same reason, the algorithm should preferably also have low computational cost and require few input parameters. Fig. 6.11(b) shows that the highest boundary recall is achieved by GS04. This is because it produces several segments close to object boundaries as seen in Fig. 6.5 to Fig. 6.10. However, GS04



(a)



(b)

Figure 6.11: *(a) Plot of the under-segmentation error w.r.t. number of superpixels. (b) Plot of the boundary recall w.r.t. number of superpixels. We use 10 iterations and* $m = 5.0$ *for SLIC. The output of NC05 is visually the most appealing but its boundary recall is quite poor. GS04 has a higher boundary recall than all algorithms, including ours, but this is because of the fact that it generates a lot of segments in the vicinity of object boundaries. SLIC has the lowest under-segmentation error and a high boundary recall for all superpixel sizes.*

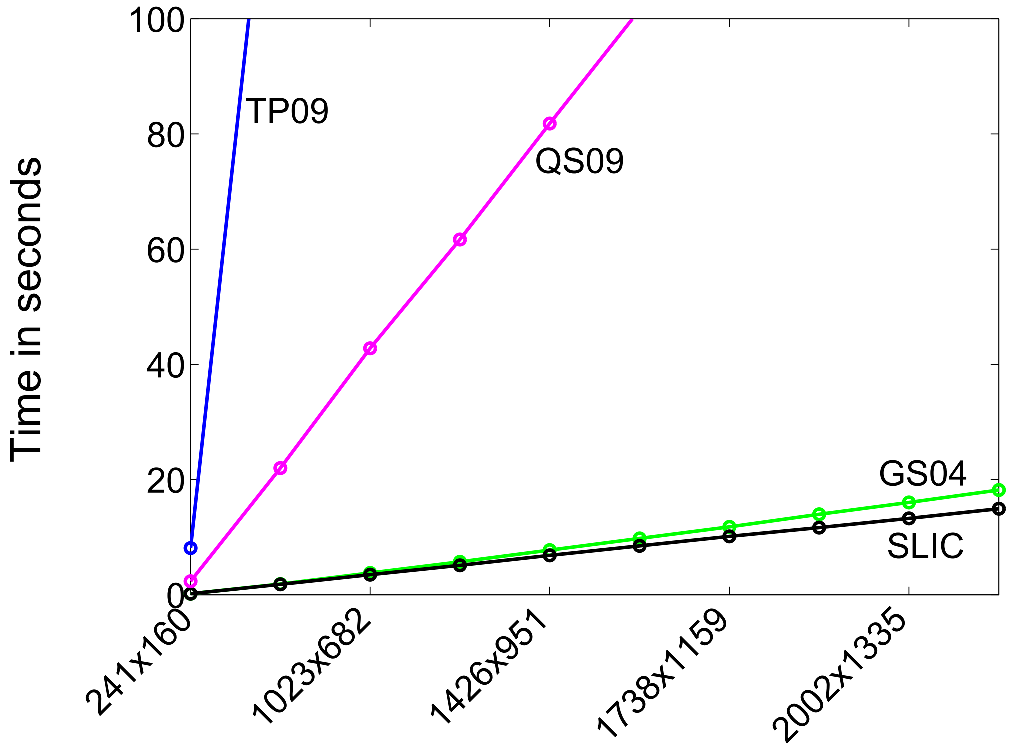


Figure 6.12: *Plot of the segmentation time taken in seconds versus image size in pixels.*

also exhibits higher under-segmentation error than our algorithm, as can be seen in Fig. 6.11(a). Our algorithm actually shows the lowest under-segmentation error in Fig. 6.11(a) as well as high boundary recall in Fig. 6.11(b) next only to GS04.

Note that GS04 as well as QS09, however, are not meant to output a desired number of superpixels of a given size unless a parameter search is performed, which requires several runs of the algorithm. Even then the superpixel sizes are unequal, making the algorithm far less suitable for superpixel-based applications [82, 107, 52]. In addition, as seen in Fig. 6.12(b) our algorithm is faster than the compared state-of-the-art algorithms for any image size (including GS04 for a single run) and it outputs the desired number of equally-sized compact superpixels.

Referring back to Table 2.2 SLIC offers a control on the number of superpixels (and consequently their size) and their compactness. It is $O(N)$ complex and faster than all the competing algorithms. It takes only one parameter - the desired number of superpixels (and optionally, the compactness factor, the default value for which is set to 10). When compared to NC05 and TP09, the only other algorithms to offer a compact superpixels of the desired number, SLIC proves superior in terms of under-segmentation error, boundary recall, and computational efficiency.

6.4 Superpixel Applications

Operating on superpixels instead of pixels can speed up existing pixel-based algorithms, and even improve results in some cases [52]. For instance, certain graph-based algorithms can see a 2 to 3-fold speed increase using superpixels [86]. Of course, the superpixel generation itself should be fast for this to be practical.

Below, we consider two typical vision tasks that benefit from using superpixels: object class recognition and medical image segmentation. In each case, superpixels have been shown to increase the performance of an existing algorithm while reducing computational cost. We show that SLIC superpixels outperform state-of-the-art superpixel methods on these tasks, but with a lower computational cost.

6.4.1 Object class recognition

Our first task is to perform object class recognition for 21 object classes (some examples in Fig. 6.13) from the STAIR vision library[6] based on the work of Gould et al [59]. Color, texture, geometry, and location features are computed for each superpixel region. Then boosted classifiers are learned using these features for each region class. Finally, a Conditional Random Field (CRF) model is learned using the output of the boosted classifiers as features. In the original work [59], NC05 is used to segment each image (of size 320 × 240 pixels) into about 200 superpixels. By applying SLIC superpixels instead of NC05, the classification accuracy increases, as shown in Table 6.1, while the computational cost is reduced by a factor of 400.

[6]`http://ai.stanford.edu/~sgould/svl`

Figure 6.13: *Example images for some of the 21 classes of the STAIR vision library.*

6.4.2 Superpixel-graph-based segementation

In this section we demonstrate the application of our superpixel in improving the quality of a graph based segmentation algorithm. Felzenszwalb and Huttenlocher [45] present an agglomerative algorithm (GS04) for segmenting images that relies on a graph built over pixels, as explained in Section 2.5.3 previously. We replace the graph on pixels by a graph on our superpixels. We use euclidean distances of average CIELAB values superpixels as the edge weights (as opposed to RGB distance based weights as in the original algorithm). GS04 mainly takes three parameters as input: a distance threshold K_d, a standard deviation value for the initial Gaussian blur, and a threshold on the smallest acceptable size of a segment, $minSize$. Since we use superpixels, which obviate Gaussian blurring as well as the need to fix $minSize$, we retain only the distance threshold K_d. The segmentation quality improves in most cases. Some results are shown in Fig. 6.14. In the case of larger images this also results in a speed-up in segmentation since the graph has to built on superpixels, which are far fewer in number than the pixels in the image. For the same reason it also results in lower memory usage.

6.4.3 Mitochondria segmentation

The superpixels generated by SLIC have been used for segmenting Mitochondria [97] in Electron Microscope (EM) imagery. In this section we present some results from early experimentation in the use of superpixels for this purpose. The technique presented in this section uses SIFT [95] features computed over each superpixel, much like Fulkerson et al. [52]. These results serve to compare SLIC with various superpixel segmentation algorithms for the task of mitochondria segmentation. The

	GS04	NC05	TP09	QS09	SLIC
Pixelwise accuracy	74.6%	75.9%	75.1%	62.0%	**76.9%**

Table 6.1: *Object class recognition for various superpixel methods.*

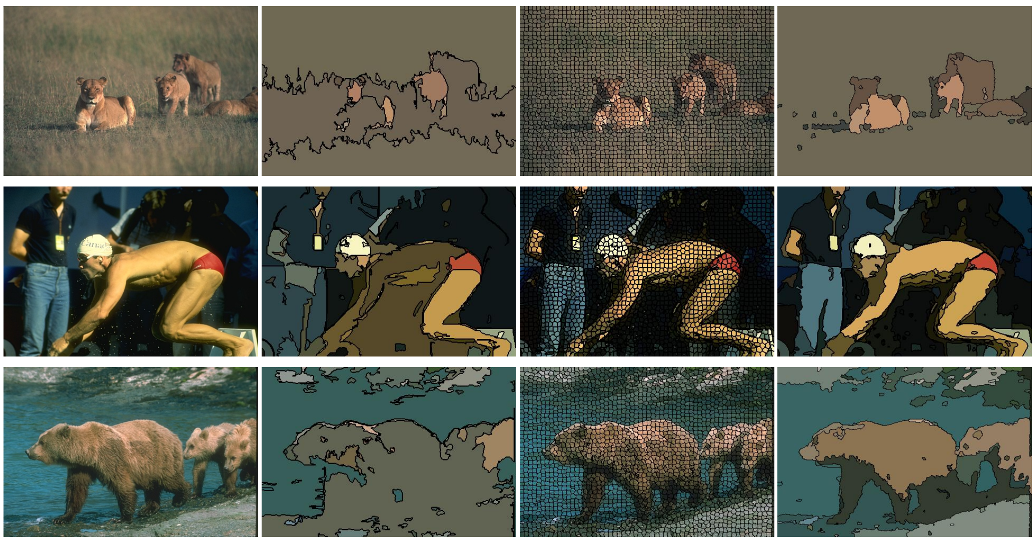


Figure 6.14: *The use of SLIC superpixels for graph based algorithms can result in higher speed, lower memory usage, and better quality output. In column (d) we show the improvement in segmentation quality of GS04 using our superpixels.*

results obtained using a more sophisticated version of this algorithm that exploits local texture, shape, and boundary cues, are presented in Appendix C. The interested reader may also refer to citation [97] for details. To assist recent efforts towards a 'bottom up' understanding of brain function (e.g. the Diadem challenge [1]) by segmenting mitochondria from neural electron microscopy (EM) images. Neuroscientists attempting to reconstruct neural structures at an extremely fine level of detail must typically perform a painstaking manual analysis on such data. Modern segmentation algorithms such as the one proposed by Fulkerson et al. [52] can efficiently automate this process by taking advantage of superpixels. Fulkerson et al. [52] use QS09 generate superpixels. Below, we compare mitochondrial segmentations using an approach based on [52] for various types of superpixels including GS04, TP09, QS09, and SLIC. NC05 is omitted because its computational cost is impractical for the high resolution images we experiment with.

The first step of the approach is to perform a superpixel over-segmentation of the image, and define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to the superpixels. Each node in $\mathcal{V}$ corresponds to a superpixel p_i. Edges $\mathcal{E}$ connect neighboring superpixels. Then, SIFT descriptors are extracted at center of each superpixel at various scales and a fixed orientation.

The segmentation is performed using graph-cuts, which partitions the graph into disjoint partitions by minimizing an objective function of the form given by Eq. 2.9. The *unary* term ψ assigns to each superpixel its potential to be mitochondria or background based on a probability $P(c_i|\mathbf{f}(p_i))$ computed from the output of a

Table 6.2: *Mitochondria segmentation results for various superpixel methods.*

	GS04	TP09	QS09	SLIC
VOC score	65.3%	58.9%	66.6%	**67.3%**

support vector machine (SVM) classifier trained using the SIFT descriptors. The *pairwise* term ϕ assigns to each pair of superpixels a potential to have similar or differing labels based on the difference of intensities between the two superpixels,

$$\psi(c_i|p_i) = \frac{1}{1 + P(c_i|\mathbf{f}(p_i))}, \tag{6.6}$$

and

$$\phi(c_i, c_j|p_i, p_j) = \begin{cases} \exp\left(-\frac{||I(p_i)-I(p_j)||^2}{2\sigma^2}\right) & \text{if } c_i \neq c_j \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

Segmentations obtained for each superpixel method were compared over a set of 23 annotated EM images of 2048×1536 resolution, containing 1023 mitochondria. We used $k = 5$ k-folds cross validation for testing and training. Results are given in Table 6.2, and example images appear in Fig. 6.15. The VOC score = $\frac{tp}{tp+fp+fn}$ is used to evaluate segmentation performance, as it is more informative than pixelwise accuracy for sparse objects such as mitochondria[7].

The advantages of the SLIC superpixel are demonstrated in the examples appearing in column 5 of Fig. 6.15. Features extracted over the regular, compact SLIC superpixels tend to be more discriminative, helping the graph-cut to produce better segmentations. The good adherence to image boundaries exhibited by SLIC superpixels result in smoother and more accurate segmentations. We can also see the drawbacks of other superpixel methods by considering the examples in columns 2 through 4 of Fig. 6.15. The irregularity of GS04 superpixels in column 2 makes the extracted features less discriminative, often causing the segmentation to fail. TP09, in column 3, performs the worst of the four methods. Because the intensity gradients in the EM images are not particularly strong, TP09 tends to merge smaller superpixels, causing issues in the segmentation. In column 4, the superpixels of QS09 appear most similar to SLIC, but still result in numerous segmentation failures where they do not respect mitochondrial boundaries as well as SLIC.

6.5 Supervoxel segmentation

Our superpixel segmentation algorithm SLIC easily extends to generating supervoxels for use in segmenting image volumes and videos. In this case the algorithm

[7]tp=true positives, fp=false positives, fn=false negatives

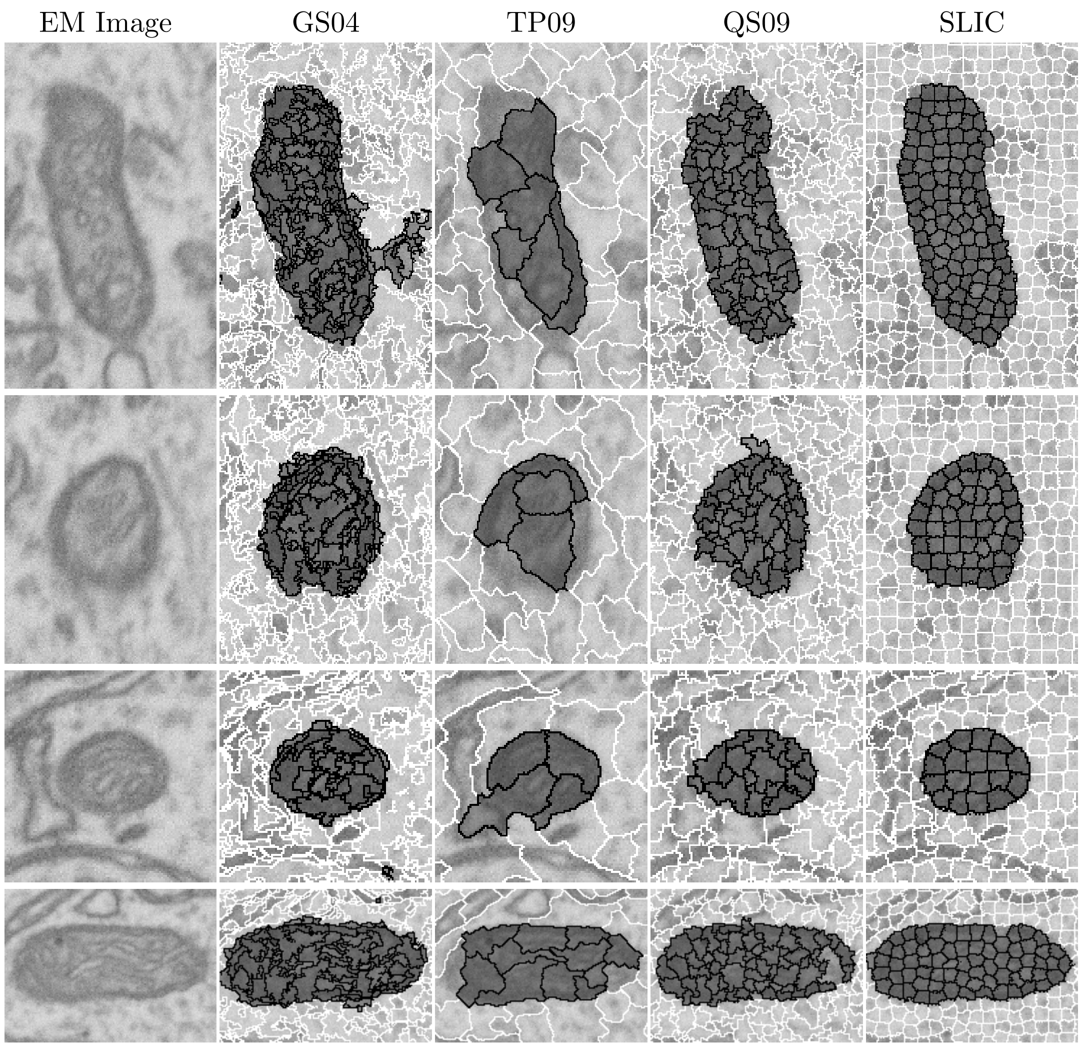


Figure 6.15: *Segmentation results on EM imagery (courtesy of Graham Knott: graham.knott@epfl.ch). Examples segmentations for GS04, TP09, QS09, and SLIC are cropped from* $2048{\times}1536$ *micrographs. Regular, compact superpixels generated by SLIC exhibit good boundary adherence and produce the best segmentation results. The large superpixels produced by TS09 are a result of the algorithm merging small superpixels over relatively weak intensity gradients in the EM image despite parameters being set for smaller superpixels. See text for further discussion.*

clusters voxels in the six-dimensional *labxyz* space. Eq. 6.3 modifies to:

$$D = \sqrt{d_{lab} + \frac{m^2}{S^2} d_{xyz}}, \tag{6.8}$$

where $d_{xyz} = (x_i - x_j)^2 + (y_i - y_j) + (z_i - z_j)^2$ for cluster center i and a pixel j in the volume. In addition, the search area for finding voxels nearest to a cluster center is performed in a three-dimensional volume of $(2S)^3$ voxels. The rest of the algorithm is similar to the superpixel case as presented in Algorithm 6.1. The post-processing step is the same too, except that the flood-fill operation is now performed in the image volume. The supervoxels generated by our technique have also for mitochondria detection. Some examples of the results are presented in the Appendix.

6.6 Pretty superpixels

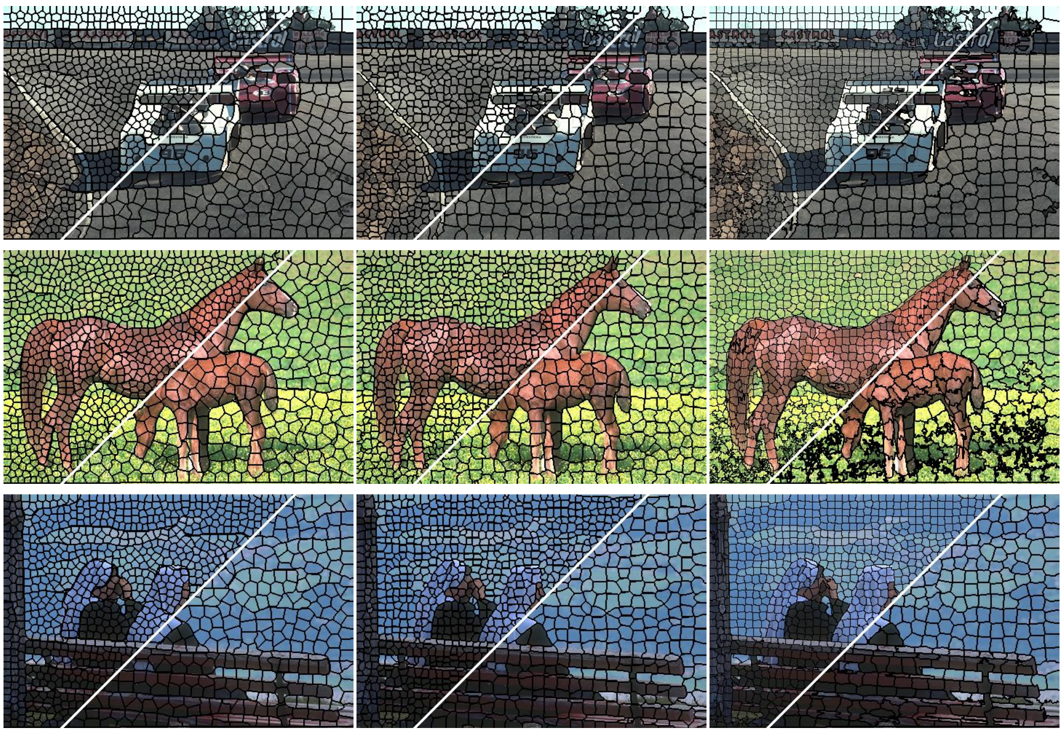
As we visually compare superpixels, we note that certain superpixels are visually more pleasing than others (Fig. 6.16). This does not necessarily mean they perform better segmentation since the boundary recall may be poorer and under-segmentation error higher than the less visually appealing superpixel algorithms. In this section we present a way to obtain visually more pleasing superpixels. This is done by changing the post-processing step. As the post-processing step, SLIC uses a connected components based algorithm to merge disconnected pixels with the nearest superpixels. Instead of this, we use geodesic distances to connect all pixels in the image to the nearest cluster seed. The resulting superpixel algorithm is termed Geodesic Simple Linear Iterative Clustering or GSLIC.

6.6.1 Euclidean versus geodesic distance

In Chapter 6 we presented the SLIC superpixel segmentation algorithm that uses a novel k-means based algorithm. It needed a post-processing step for merging stray superpixels. This was because the Euclidean distance based clustering in the five-dimensional *labxy* space did not automatically enforce connectivity in the two-dimensional *xy*-plane. Instead of the Euclidean distance it is also possible to use geodesic distances, as presented in Chapter 4, for clustering in the *labxy* space. The advantage is that the connectivity in the *xy* plane is inherent in such clustering. However, such clustering is quite sensitive to the starting seed value. Also, the under-segmentation error and the boundary recall are worse than those of SLIC.

6.6.2 Geodesic distance based superpixels

In Chapter 4 we used discrete geodesic distances to label pixels as foreground or background using seeds obtained by thresholding the saliency map of an image. So



(a) NC05 (b) TP09 (c) **SLIC**

Figure 6.16: *Superpixel segmentation of different algorithms is not equally pretty. Segmentations from NC05 for instance is considered more visually appealing than others, but its boundary recall is poorer and under-segmentation error greater than other algorithms.*

the labeling problem was binary in this case. Performing superpixel segmentation using geodesic distances is the multi-label version of the same problem, with the labels being the number of superpixels. The segmentation is done the same way as in Section 4.1.3 of Chapter 4 except that now we have as many label seeds as the number of superpixels (instead of binary foreground and background labeled seeds). Each pixel at a position (x, y) is assigned the label of its nearest labeled seed if the geodesic path to the pixel is shorter than that from any other seed. In a sense, every labeled seed in the vicinity of a pixel competes the ownership of the pixel.

6.6.3 Post-processing using geodesic distances

We perform several iterations of the usual SLIC algorithm and replace the connected components based post-processing step with one iteration of geodesic distance based labeling (or equivalently clustering). The choice of the starting seed is quite crucial for geodesic distance based clustering. As seeds for the geodesic clustering we the lab values of pixels whose position is nearest to the xy values of the k-means seeds of SLIC.

6.6.4 GSLIC comparison with state-of-the-art

NC05 arguably outputs the most visually pleasing superpixels followed by TP09 but these two algorithms have the worst boundary recall. The SLIC plus geodesic post-processing scheme we present provides prettier superpixels the SLIC and yet has better boundary recall and under-segmentation error as compared to NC05 and TP09 as shown in Fig. 6.18. There is a compromise between how visually appealing the superpixels are and how good their boundary recall is.

6.7 Summary of the chapter

In this chapter we presented a superpixel segmentation algorithm that generates a desired number of roughly uniformly sized superpixels. The algorithm is computationally efficient, takes as an input parameter only the desired number of superpixels, and produces superpixels that demonstrate lower under-segmentation error and higher boundary recall than most state-of-the-art algorithms. In addition, the algorithm is easily extended for generating supervoxels. We presented applications of our superpixels in object class identification, graph based image segmentation, and mitochondria detection.

Our algorithm SLIC generates superpixels efficiently but they are visually less appealing than certain competing methods. SLIC offers speedy convergence and is quite robust to the starting seed position as long as it uses Euclidean distances. But when the k-means clustering is performed using geodesic distances, the superpixels are prettier and inherently connected but the convergence is sensitive to the starting seed value. The advantages of the two approaches can be obtained if we first use

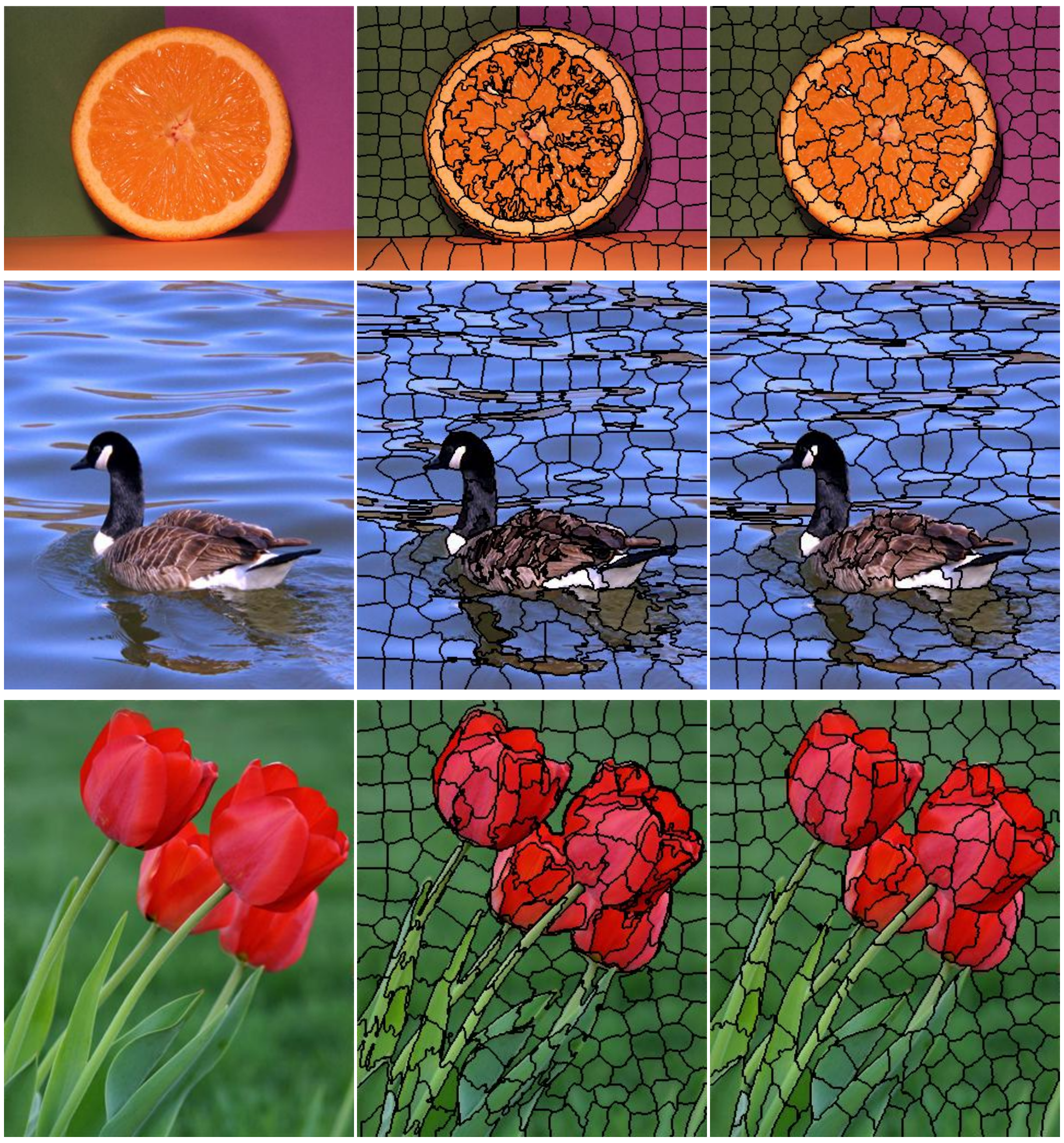
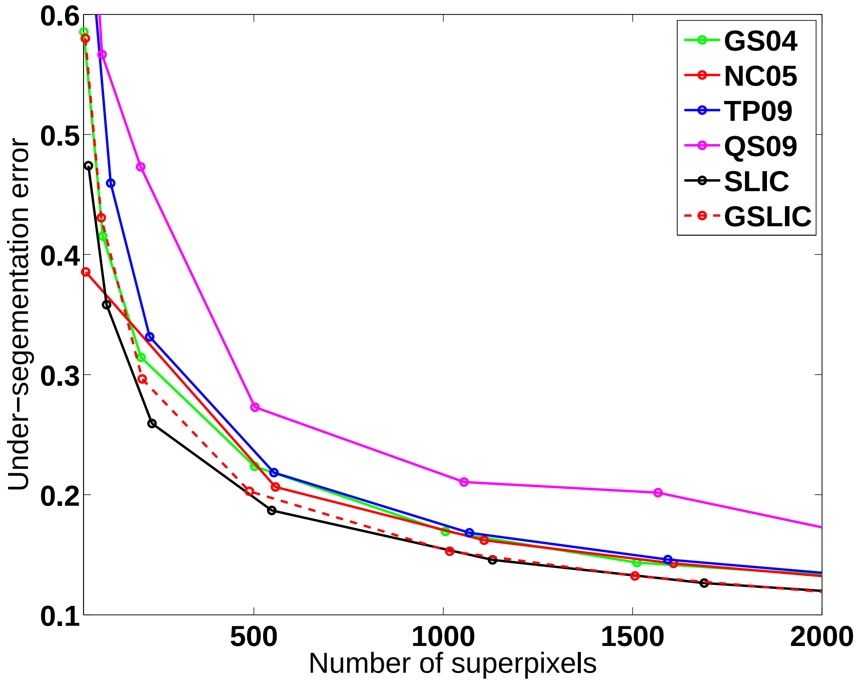
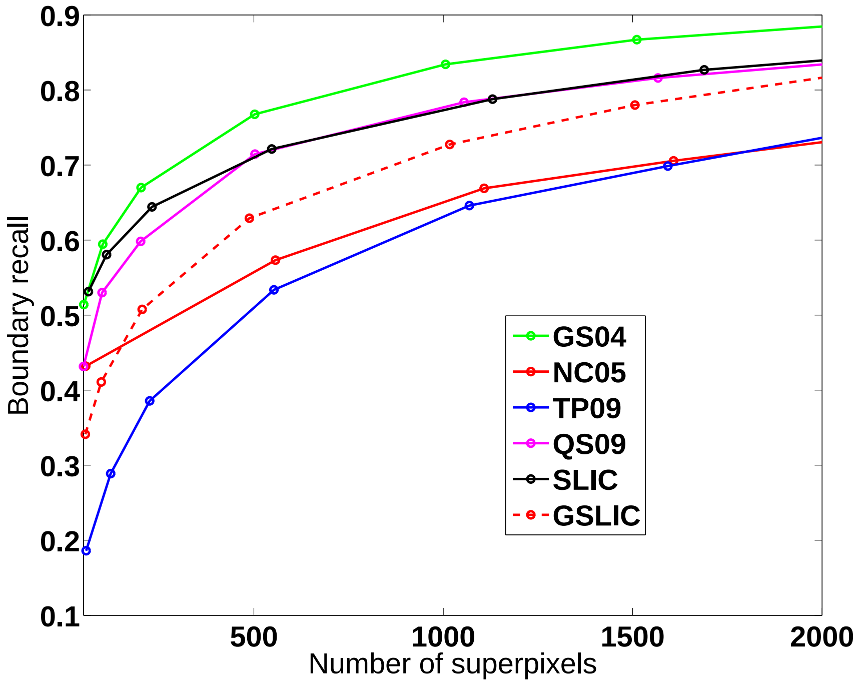


Figure 6.17: *Output of one iteration of geodesic distance based clustering using pixels at the SLIC centroidal positions as seeds. The visual appeal of GSLIC is better than that of SLIC.*



(a)



(b)

Figure 6.18: *(a) Under-segmentation error and (b) boundary recall of segmentations for GSLIC. GSLIC shows almost the same undersegmentation error as SLIC. The boundary recall is lower than SLIC, but it is still much better than NC05 and TP09.*

SLIC with Euclidean distance based k-means clustering followed by one iteration of geodesic path based clustering using the converged SLIC centers as initial seeds. This gives us superpixels that are visually appealing, well-connected, and more importantly, have low under-segmentation error and high boundary recall than the case when only geodesic distance based clustering is performed.

Chapter 7

Conclusions

Automatically finding objects of interest in images is one of the most basic computer vision problems, which has both motivated researchers because of the widespread applications, and challenged them because of its difficulty. Knowledge of the objects of interest can provide content awareness that has applications in security, entertainment, and biological image processing, to name a few. The goal of this thesis was to advance the research in two aspects of this problem: task-independent saliency detection and task-specific object detection. We presented algorithms for saliency detection and demonstrated their applications. We presented the concept of database saliency and applied it for novel forms of image summarization. We presented a superpixel segmentation and showed its applications in image segmentation, object class detection, and mitochondria detection.

7.1 Thesis summary

We presented a review of the state-of the art for *saliency detection* and *superpixel segmentation* in Chapter 2. We studied some of these saliency detection algorithms from a frequency-domain perspective to get more insights into their performance. For superpixel segmentation, which aids the feature extraction step necessary for certain techniques of *object detection*, we reviewed existing techniques by defining them based on the two categories of graph-based methods or gradient based methods. In addition, we covered certain clustering techniques that were deployed in later chapters.

The conclusion we drew from the frequency-domain perspective of saliency detection techniques was that in the event of not knowing what object we are looking for in an image, or its scale, it is better to use a simple approach and rely on center-surround filtering with a very low-frequency cut-off. This led to our first algorithm for saliency detection presented in Chapter 3. For the second saliency detection algorithm presented in Chapter 3, we showed that despite not knowing the object scale a priori, we can make a practical assumption about the scale based on the pixel position in the image and as a result improve our first algorithm. Both our algorithms for saliency detection significantly outperform state-of-the art algorithms.

In Chapter 4, we showed applications of saliency detection for the task of salient object segmentation and image re-targeting. In the task of object segmentation, unlike some previous techniques that rely on user-input indicating object and background regions, we relied on our saliency maps to automatically label these regions. We then segmented salient objects using three different methods - adaptive thresholding, graph-cuts, and geodesic distance based labeling. In all three cases, we compared the efficacy of using our saliency maps as compared to those of existing algorithms and demonstrated that saliency maps from our algorithms are better suited for salient object segmentation. This is because our saliency maps have better object boundaries and usually highlight salient objects better.

For image re-targeting we used the method of *seam carving*. We improved this technique, firstly, by using our saliency maps instead of the conventional gradient energy maps, and secondly, by introducing the use of color information in computing the best seam. We compared our improved seam carving technique with several existing algorithms to demonstrate the improvement in output quality. The additional advantage of using our saliency maps is that they do not need to be recomputed after each seam removal and they are more robust to image noise as compared to conventional energy maps.

While saliency detection is limited to images, in Chapter 5, we extended the notion of task-independent saliency to image databases and proposed a scheme to detect "interesting images" from a collection of images. We further showed how attractive summaries can be created from such images. The first such summary is an image mosaic, which improves upon the visual appeal of conventional mosaics by using varied sizes of images rather than tiny images of the same size. The second technique for creating image summaries we presented is a collage that stitches together interesting images chosen using our database saliency detection technique.

In Chapter 6, we presented a superpixel and supervoxel segmentation algorithm that was used to develop techniques of detecting known objects of interest, specifically mitochondria, in noisy medical images and image volumes. Superpixel segmentation serves to simplify and reduce computational load by breaking down the image into roughly equally sized segments with similar properties, whose number, size, and compactness can preferably be controlled. Our algorithm is particularly effective for this task since it shows higher boundary recall and lower under-segmentation error as compared to competing algorithms. It offers roughly equally sized superpixels based on the sole user input parameter, namely the number of superpixels, and boasts of lower computational and memory overhead as compared to existing algorithms. It is also one of the few algorithms that can be used in practice for very large images. In addition, we showed how our method is easily extended for computing supervoxels on image or video volumes. The technique of object detection developed using our superpixels was used to detect mitochondria in noisy medical images. In this task, our superpixel algorithm performed better than the other algorithms.

7.2 Some reflections and future research

We took a first principles approach in terms of spatial frequency content of saliency maps and it led us to develop a couple of simple but very effective algorithms for salient object detection using only the features of color and intensity. Our algorithms consistently proved to be more effective that more complicated algorithms that use additional features or more sophisticated models. We found that it is better to use simple features for saliency detection since very little can be guessed about the form and features of an unknown object. An evident course of future work is saliency detection in videos. It would require taking into account object motion, which is a very significant cue for visual attention

Human visual attention is also guided by other aspects of saliency apart from the one arising from color and intensity contrast, namely, shape, orientation, closure, symmetry, as well as direction and speed of motion, in case of moving scenes. Some researchers have tried taking into account some of these aspects but without resounding success unless dealing with very specific images. This is both because of the limitations of the features used by them as well as the lack of a robust mechanism to combine the use of various features. This therefore remains a future opportunity to improve saliency detection.

A lot of what we know about vision is from psycho-visual studies and neuroscience research. Usually the proposed models explain only some of the observed visual capabilities but there are newer discoveries being made in the areas of psychophysics and neuroscience. We should update our models with these discoveries and improve saliency detection.

Saliency detection can be used for several simple but useful applications. Some of the works in the offing are non-photo-realistic rendering, saliency-based image filtering, and saliency-based contrast stretching and color correction. Saliency detection may also aid in clever use of printing ink - more dots per inch for salient regions as opposed to non-salient ones. This warrants further investigation.

Not everything that is salient is important to us. Saliency may direct visual attention, but our visual system quickly ignores it for what is more important, even if it is less salient. While we assume that saliency alone can direct visual attention, it would be true only in a completely alien environment. But humans have a very strong familiarity with most of the environments they encounter in their daily lives. So, in most cases we are able to ignore ‘salient noise’, i.e salient regions or objects that grab our involuntarily attention for a very brief moment, to be able to find the things that matter to us. To widen the range of real world applications for everyday use, the next step to task-independent saliency detection is task-dependent object detection, particulary multi-category object detection. As a research area, this is a vast and difficult topic. We can make a small headway, at least in domain specific object detection, using novel feature extraction techniques like superpixel segmentation, and using clever machine learning algorithms. However, the evident fact that the human visual system is capable of discriminating between thousands of objects in a fraction of a second does suggest emphatically that there is a highly simplified and efficient underlying abstraction scheme that is not explained by the algorithmic models we use presently. This remains an open challenge to overcome in the future.

Image database saliency is an interesting area to delve into given the deluge of images that is getting increasingly difficult to manage. While we take into account the content of the images using simple features, there are a lot of other aspects of the image content (e.g. the objects, the aesthetics, etc.) that can be considered for assigning an interestingness value to it. Further research can be done in abstracting the images better, computing image similarity better, and experimenting other schemes of defining and finding interesting images. This can surely be aided by user studies since the notion of interesting images is highly subjective. It is also possible to extend database saliency techniques to web-mined images from popular online image databases like Flickr and Picasa.

Another aspect that needs more research in this connection is image clustering techniques. One of the main limitations of several clustering techniques is that they are not easily scalable. Ideally, it should be possible to update clusters on the fly with images added or removed from the database. With ease of scalability in mind, approaches like locality sensitive hashing could be used. The approaches should be non-iterative in nature and should be able to support clustering in high-dimensional spaces.

The number of pixels in images is growing at a rapid rate. It puts a heavy computational overhead on most image algorithms. Superpixels offer an excellent option of lowering the complexity by a few orders of magnitude. The abstraction performed by superpixels is preferable over sub-sampling since it is anisotropic and takes into account the local statistics. So, there is room for more applications of superpixels, including in image compression, optical flow computation, and so on. We can also look into parallelization of superpixel segmentation techniques and deployment on mobile devices.

Finally, we would like to make a small case for simplicity and result oriented top-down research. For saliency detection, using center-surround filtering by keeping the appearance of the saliency map as the objective was surprisingly ignored. The same can be said about using color and spatial information together for superpixel segmentation using a known clustering technique instead of using more complex models. In both cases, our algorithms were developed using a goal-oriented approach. Instead of choosing a model or tool to find an application for, we tried to find an effective approach for dealing with the task at hand. The result was simple algorithms that outperformed more complex state-of-the-art approaches.

Appendix A

Text Detection

A.1 Introduction

The availability of information about the content of an image has always helped in image classification problems. Text of different languages frequently appears in various forms in images, and conveys useful information about names of people, titles, locations, dates of events, etc.. This information is potentially very useful for annotating images automatically, thereby aiding image retrieval. There can be several other applications of such a text reader, like automatic video annotation, number plate recognition, robot navigation etc..

Detecting and reading text automatically is more difficult if it appears in a natural image (as opposed to a scanned and thresholded binary image of a scanned document).

The task of text reading requires both *localization* (or *detection*) of text regions in the image, as well as *recognition* of this text. While previous works have mainly addressed the issue of text detection/localization in images, relying on third party Optical Character Recognition (OCR) software for text recognition, we address both issues in this chapter.

In this chapter, as a first step we first detect text regions in images using an AdaBoost [49, 137] based detector and then we binarize the text region so that it is suitable for being fed to an OCR engine.

A.2 Approaches in text detection

In this section we present research on text region detection and post-processing that aims to use an OCR engine for reading the text. The approaches presented make use of the appearance and gradient properties of text [105] in images. In this chapter we choose to classify these approaches into two categories. To the first category belong all the approaches that detect text regions using edges [146, 57, 127], texture properties [84, 85, 145] or connected component analysis [44]. Most works often use more than one of these properties in combination.

In the work of Lienhart and Effelsberg [87] the character features of monochromacity and contrast with the local neighbourhood are used to classify a pixel as a part of a connected component of a character. After this, some geometrical constraints of specific ranges of width and height as well as texture features are used to filter out character regions. Connected component based character detection techniques in combination with other text properties are also presented in [44] where character extraction is done in steps of Sobel edge detection, RGB image binarization, connected component analysis, followed by filtering of false detection based on certain heuristics.

Text detection schemes that use machine learning schemes fall in the second category. In the technique of Li et al. [84, 85], a three layer neural network is trained to detect text based on wavelet features of the multi-resolution wavelet pyramid of video frames. An AdaBoost cascade is used as the text region detector by Vanhoucke and Gokturk [133] inspired from the fast detection results of Chen and Yuille [27] in ICDAR'05 text detection competition [96]. They introduce some additional features apart from the basic haar-like features presented in [139].

This has partly to do with OCR being considered a closed area of research, with existing techniques being near perfect. However, OCR systems do not perform well on non-binarized images with complex backgrounds [145]. This section presents research done so far in text detection in natural images. We first cite general research on text detection using any method, followed by work that specifically uses AdaBoost for this purpose.

A.2.1 Adaboost

Boosting is a general method for improving the performance of learning algorithms. The AdaBoost, which stands for adaptive boosting, was introduced in 1995 by Freund and Schapire [49]. AdaBoost was deployed by Viola and Jones [139] in 2001 to get some impressive results in face detection. The key improvements were the use of extremely simple haar-like features that were easy to compute and the use of cascades of AdaBoost learners to gain speed. Improvements on this were presented by Lienhart et al. [88] and Shapire [?]. Since then AdaBoost has been used for other applications as well [138, 104].

A.2.2 Text Detection and AdaBoost

Of particular interest to us in this chapter is the use of the AdaBoost cascade in text detection. After detecting text regions using the AdaBoost cascade approach similar to Chen and Yuille [27], Vanhoucke and Gokturk [133] use three different OCR to read text from the extracted text regions and the results are fused using a Bayesian approach. Although the approach of Chen and Yuille [27] was the second best in accuracy in ICDAR'05 [96] competition, it was about forty times faster than the best approach.

A.3 Text recognition using AdaBoost

The goal of our work is to detect text in natural scenes and binarize it. A typical example would be the name of a place on a sign post or the words on a shop's facade. This is not a simple problem given the variations in size, font type, perspective, color of text and its background. So the learning algorithm used for text recognition needs to be very robust. Given the previous success of AdaBoost in object detection [139], it seemed like a good choice. In addition it provides speed advantages too if haar-like features and cascading is used.

The first step consists of detecting text regions using an AdaBoost text region detector. This helps text region localization. The text region given by the detector then is then segmented using a k-means algorithm on CIELAB color space. We then perform connected component analysis on the segmented image to produce a binarized image that only contains characters and background. This binary image may be used to perform OCR using a third party OCR engine.

A.3.1 AdaBoost approach

First we will review the AdaBoost algorithm A.1 and then we will see more in details the two main contributions of Viola and Jones: the haar-like features computed using integral images and the cascade A.3.3. AdaBoost is an algorithm to create a strong classifier as a weighted sum of several weak classifiers. The *weak* learners or classifiers classify data in a rough manner with an accuracy rate better than random. Formally, let the training set be $(x_1, y_1), ..., (x_m, y_m)$ where each x_i is a sample taken from the training set and each label $y_i \in \{-1, +1\}$, i.e. belongs to the negative or positive set. Furthermore let $h_1, h_2, ..., h_T$ be a set of hypotheses (features or weak learners), then the weighted sum of weak classifiers $f(x)$ is given by

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{A.1}$$

$$H(x) = sign(f(x)) \tag{A.2}$$

Here α_t denotes the weight with which the hypothesis h_t is combined and $H(x)$ the final classifier.

AdaBoost aggressively selects one weak classifier at each step $t = 1, ..., T$. One of the main ideas of the algorithm is to maintain a distribution of a set of weights over the training set. The weight of this distribution on training example i a iteration t is denoted $d_t^{(i)}$. Initially, all weights are set equally, but in each round, the weights of incorrectly classified examples are increased so that the base learner is forced to

Algorithm A.1 Adaboost algorithm

```
1.Input: $S = \{(x_1, y_1), ..., (x_N, y_N)\}$, Number of iterations T.
2.Initialize: $d_1^{(i)} = 1/N$ for all $i = 1, ..., N$
3.Do for $t = 1, ..., T$,
  (a) Train classifier with respect to the weighted sample set $\{S, d^{(i)}\}$ and obtain
  hypothesis $h_t \longmapsto \{-1, +1\}$,i.e. $h_t = L(S, d^{(i)})$.
  (b) Calculate the weighted training error $\varepsilon_r$ of $h_t$ :
      $$\varepsilon_t = \sum_{i=1}^{N} d_i^{(t)} I(y_i \neq h_t(x)_i)$$
  (c) Set:
      $$\alpha_t = \frac{1}{2} log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
  (d) Update the weights:
      $$d_{t+1}^{(i)} = d_i^{(t)} exp\{-\alpha_t y_i h_t(x_i)\}/Z_t$$
  where $Z_t$ is a normalization constant, such that $\sum_{i=1}^{N} d_i^{(t+1)} = 1$
4.Break if $\varepsilon_t = 0$ or $\varepsilon_t \geq \frac{1}{2}$ and set $T = t - 1$
5.Output: $f_T(x) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(x)$
```

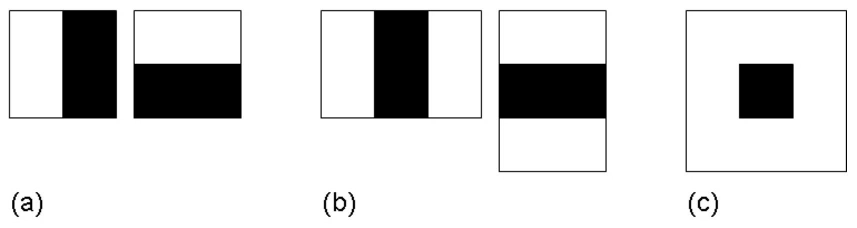


Figure A.1: Feature set used

focus on the hard examples in the training set. The weak learner's job is to find a weak classifier (also called rules-of-thumbs or weak hypothesis) $h_t : x \rightarrow \{-1, +1\}$ appropriate for the distribution d_t. In other words, its job is to minimize the error of misclassified examples

$$\varepsilon_t = \sum_{i=1}^{N} d_t^{(i)} I(y_i \neq h_t(x_i)) = \sum_{i:h_t(x_i)\neq y_i} d_t^{(i)}$$

Once the weak hypothesis h_t has been chosen, AdaBoost chooses a parameter α_t. Intuitively, α_t measures the importance that is assigned to h_t. Note that $\alpha_t \geq 0$ if $\varepsilon \leq 1/2$ and that α_t gets larger as ε_t gets smaller. The distribution d_t is next updated using the rule 3.(c) shown in Algorithm A.1. The effect of this rule is to increase the weight of examples misclassified by h_t, and to decrease the weight of correctly classified examples. Thus, the weights tend to "concentrate" on difficult examples. The final hypothesis H is a weighted majority vote of the T weak hypotheses where α_t is the weight assigned to h_t.

A.3.2 Features

We use the same basic haar-like features as Viola and Jones [139]. The experiments showed that even the basic set shown in figure A.1 results in very good detection of letters. So we could capitalize on the same simplicity and speed of feature calculation using integral image approach as for face detection in [139].

A.3.3 Cascade

This section describes an algorithm for constructing a cascade of classifiers, which achieves increased detection performance while radically reducing computation time. The key insight is that smaller, and therefore more efficient, boosted classifiers can be constructed, which reject many of the negative sub-windows while detecting almost all positive instances (i.e. the threshold of a boosted classifier can be adjusted so that the false negative rate is close to zero). Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates. A cascade of classifiers is a degenerated decision tree where at each stage a classifier is trained to detect almost all objects of interest while rejecting a certain fraction of the non-object patterns.

The cascade design process is driven from a set of detection and performance goals. For our detection, we need good detection rate (about 95%) and extremely low false positive rate (about 10^{-6}). The number of cascade stages and size of each stage must be sufficient to achieve similar detection performance while minimizing computation. Given a trained cascade of classifiers, the false positive rate of the cascade is

$$F = \prod_{i=1}^{K} f_i$$

where F is the false positive rate of the cascaded classifier, K is the number of classifiers and f_i is the false positive rate of the ith classifier on the examples that get through to it. The detection rate is

$$D = \prod_{i=1}^{K} d_i$$

where D is the detection rate of the cascaded classifier, K is the number of classifiers, and d_i is the detection rate of the ith classifier on the examples that get through to it. Given concrete goals for overall false positive and detection rates, target rates can be determined for each stage in the cascade process. For example a detection rate of 0.9 can be achieved by a 10 stage classifier if each stage has a detection rate of 0.99 (since $0.9 \approx 0.99^{10}$). While achieving this detection rate may sound like a daunting task, it is made significantly easier by the fact that each stage needs only achieve a false positive rate of about 30% ($0.30^{10} \approx 6 \times 10^{-6}$).

The original AdaBoost algorithm has to be modified to ensure the minimization of the false negative rate instead of the training error. One simple way to impose that is to adjust the final threshold. Increasing this threshold will badly affect the detection rate and improve the false positive rate. The AdaBoost classifier with a threshold b is now

$$f_T(x) = \sum_{t=1}^{T} \Big(\frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(x) \Big) + b$$

.

The cascade structure has three main parameters that need to be determined:

- Total number of classifiers: K
- Number of features n_i of each stage i
- Threshold b_i of each stage i

Finding the three optimal parameters is quite complicated considering that the goal is to minimize the computation time of the total classification. The principle is to increase the number of features and stages until the given detection objective are reached. The algorithm is given in Algorithm A.1.

Algorithm A.2 Cascade algorithm

```
 1: User selects values for $f$, the maximum acceptable false positive rate per layer
    and $d$, the minimum acceptable detection rate per layer.
 2: User selects target overall false positive rate $F_{target}$
 3: $P$= set of positive examples
 4: $N$ = set of negative examples
 5: $F_0 = 1.0; D_0 = 1.0$
 6: $i = 0$
 7: while $F_i > F_{target}$ do
 8:   $i \leftarrow i + 1$
 9:   $n_i = 0; F_i = F_{i-1}$
10:   while $F_i > f \times F_{i-1}$ do
11:     $n_i \leftarrow n_i + 1$
12:     Use P and N to train a classifier with $n_i$ features using AdaBoost
13:     Evaluate current cascaded classifier on validation set to determine $F_i$ and
        $D_i$
14:     Decrease the threshold for the $i$th classifier until the current cascaded clas-
        sifier has a detection rate of at least $d \times D_{i-1}$. Revaluate current cascaded
        classifier and update $F_i$ and $D_i$.
15:   end while
16:   empty the negative set
17:   if $F_i > F_{target}$ then
18:     Evaluate the current cascaded detector on the set of non-face images and
        put any false detections onto the set $N$
19:   end if
20: end while
```

A.3.4 Training set

The training set for AdaBoost consists of positive and negative examples. We require are two kinds of training sets - one for text region detection and one for character recognition.

The positive examples training set for text regions detector is text regions containing roughly three letters each. The negative examples are random regions from natural images that do not contain any text. The size of the examples is 30×15 pixels (width times height). We use 100 positive examples and 200 negative examples.

A.4 Text region detection and localization

The AdaBoost cascade trained to detect text regions gives several overlapping rectangular regions as output as shown in images (a) of Fig. A.2(a). In the figure, the blue rectangles show text detected on the original image, while the yellow rectangles are rectangles detected on the inverted (or negative) image. The following algorithm is used to find the final rectangle encompassing the text region, as shown in images (b) of Fig. A.2.

1. The average area of rectangles is determined. All rectangles having an area smaller than half this area are rejected as a first step.

2. The average height of the remaining rectangles is found. Choosing this to be the bin size, a histogram of the y coordinates of the centers of the rectangles is created.

3. The bins are sorted in terms of the number of y coordinates of the centers. Those bins that contain eighty percent of the rectangles are chosen.

4. The largest bounding rectangle for the all the rectangles chosen this way is found. If at the end of this process there are any rectangles that overlap beyond a certain threshold, then they are merged by finding the largest bounding rectangle.

A.5 Text region binarization

Once the text region is localized, we perform a k-means segmentation of it using $k = 2$. This is because usually, the text and the background are uniformly colored. We perform connected components analysis and binarize the text region assuming that the largest connected component(s) belong(s) to the background, and smallest ones are noise, and the intermediate ones to the text region. The steps of the binarization are illustrated using some examples in Fig. A.2.

A.6 Combining text detection and generic saliency detection

The output of our detector is several overlapping *detection rectangles* around the text region, as shown in Fig. A.3(b). We obtain *text saliency maps* by counting at each pixel the number of times it is detected by the detection rectangles. After normalizing this count to lie between $[0, 255]$ we obtain an image as shown in Fig. A.3(c). The combined saliency map is obtained by a pixel-wise multiplication of the text saliency map and the generic saliency map. This is shown in Fig. A.3(d). The combined saliency maps are more amenable to binarizing and isolating text in order to be fed to an OCR engine.

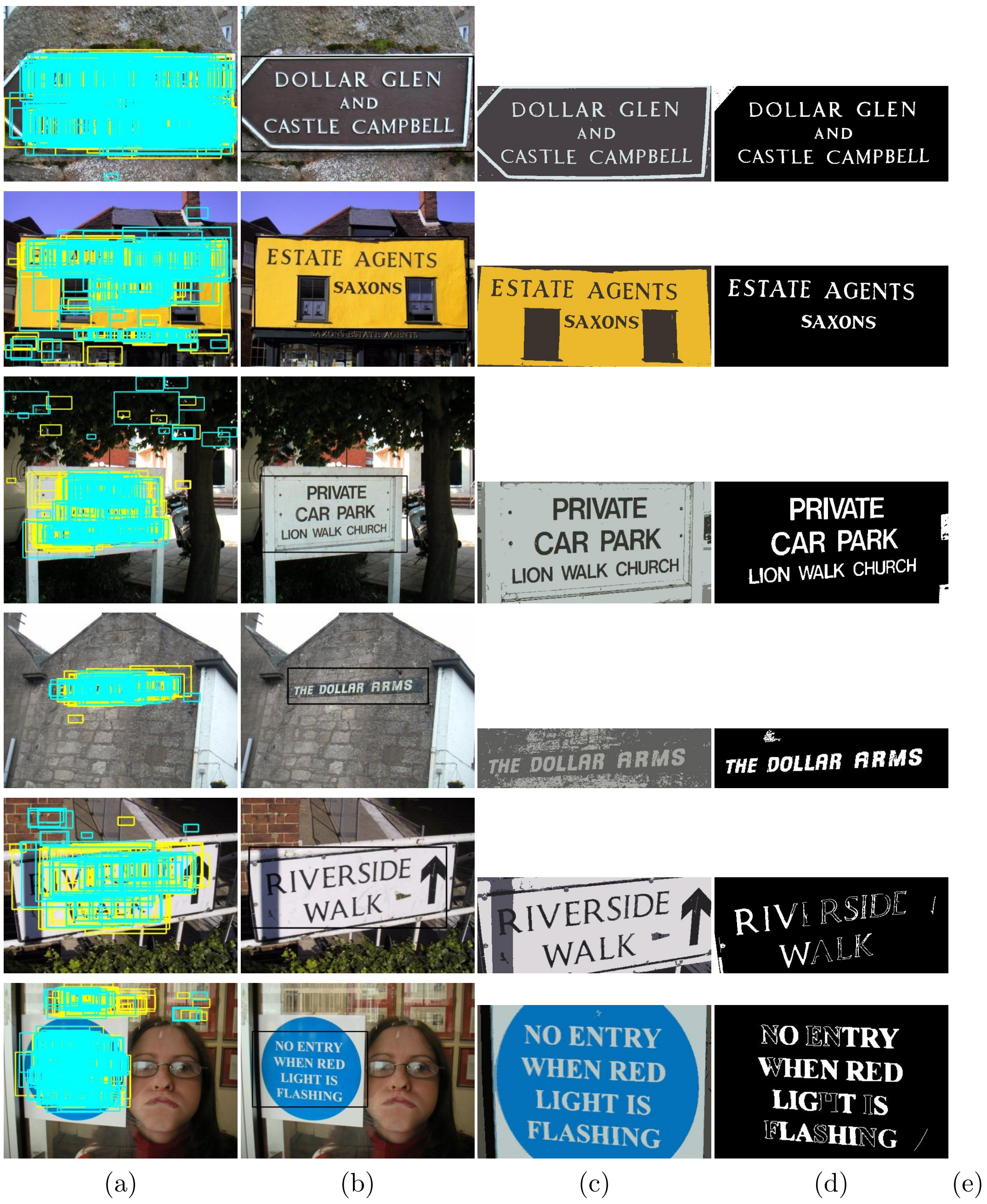


Figure A.2: *Figure showing text detection and binarization examples. (a) The initial detections on the input image using the AdaBoost cascade. (b) The combined detection (black bounding box) using the algorithm presented in Section A.4. (c) The k-means segmented result. (d) The binarized output.*

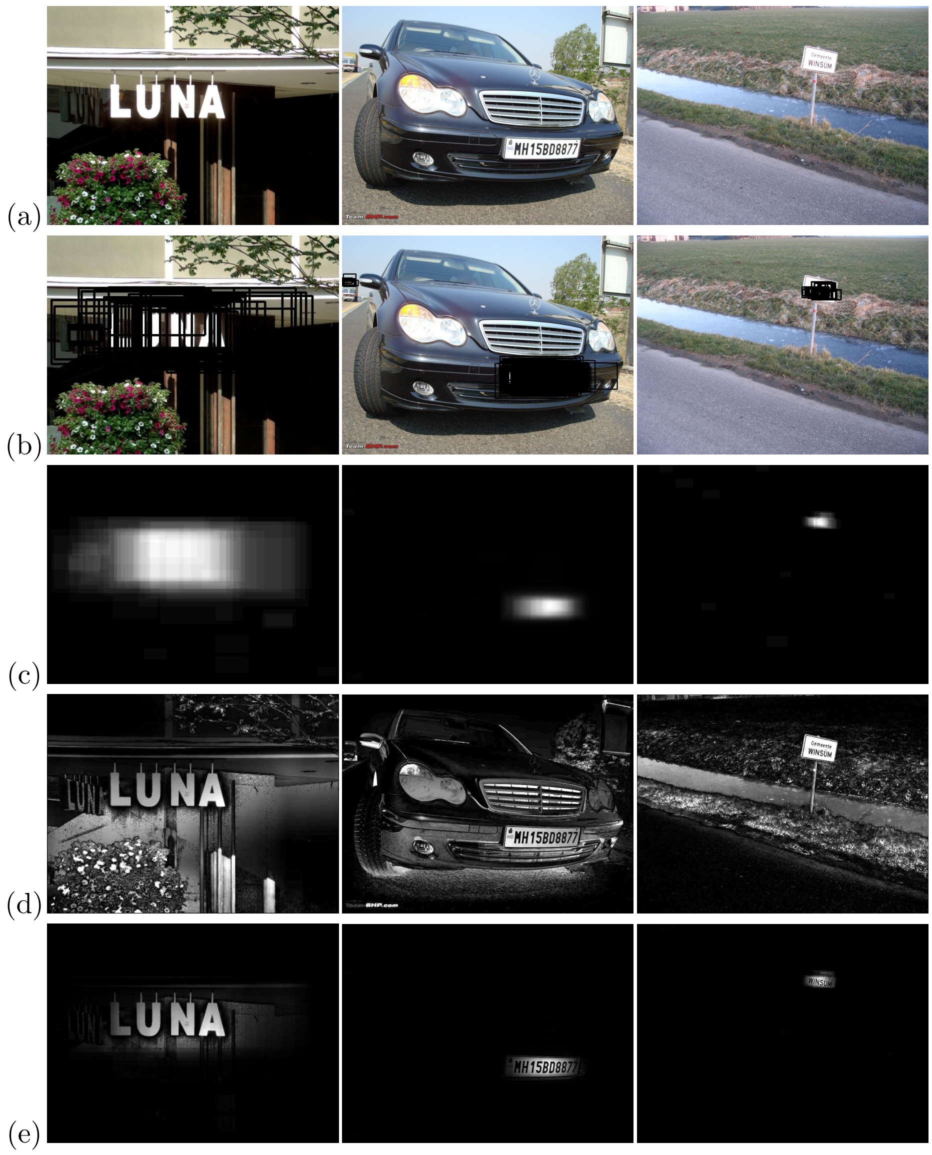


Figure A.3: *Combining text saliency map and generic saliency detection. (a) Original natural image with a text region. (b) Raw AdaBoost text detection rectangles (in black). (c) Text saliency map by counting the number of detections at each pixel. (d) Generic saliency detection using techniques from Chapter 3(e) Multiplying text saliency and generic saliency detection. The combined saliency map can be more easily used for binarizing text regions and feeding to an OCR engine.*

A.7 Summary of the chapter

In this chapter we presented a task-specific detection technique for text occurring in natural images. We first presented the state of the art in text detection. We then elaborated upon the text detection technique that is based on a well-known face detection approach. Our method takes few examples to train, uses simple features, is robust, and computationally efficient. We showed how the multiple detections that result can be combined to obtain the final detection. We then presented a method of binarizing the detected text so that it can be passed on to an OCR engine. Finally, we showed how text saliency maps can be created and combined with generic saliency maps to obtain task-specific saliency maps.

Appendix B

Object scale and Gaussian filtering

What appears to be a yellow patch of land from an aeroplane turns out to be a field of sunflowers closer on land. An even closer look can show the constituent molecules, atoms, and subatomic particles. We say the yellow patch is observed at a coarse scale while the atoms are visible at the much finer scale. Real-world objects appear differently depending on the *scale* of observation. Just as objects in the world, details in an image exist only over a limited range of resolution. For a computer vision system analyzing an unknown scene, there is no way to know a priori what scales are appropriate for describing the structures of interest in the image. Hence, a reasonable approach is to consider descriptions of the image at multiple scales. The formal theory for handling image structures at different scales, by representing an image as a one-parameter family of smoothed images, is the *scale-space theory* [143, 78, 46, 90]. The notion of scale-space applies to signals of arbitrary numbers of variables. Here we restrict ourselves to two-dimensional images. For a given image $I(x, y)$, its linear (Gaussian) scale-space representation is a family of derived signals $L(x, y; \sigma)$ defined by the convolution of $I(x, y)$ with the Gaussian kernel

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{B.1}$$

with σ being the standard deviation of the Gaussian kernel such that

$$L(x, y; \sigma) = (g(., .; \sigma) * I)(x, y) \tag{B.2}$$

Typically only a finite discrete set of levels of L for $\sigma \geq 0$ are considered in the scale-space representation.

For $\sigma = 0$, g becomes an impulse function such that $L(x, y; 0) = I(x, y)$, so that the scale-space representation at the finest scale level $\sigma = 0$ is the image I itself. As σ increases, L is the result of smoothing I with a larger Gaussian filter, thereby removing more fine structures, i.e. high frequency detail. Specifically, high frequency details, which are significantly smaller than σ in extent are removed from the image as we move towards coarser scales, leaving us with low-pass versions of the image. This illustrates the relation between scale and spatial frequency content of images.

It would seem that any low-pass filter g could be used to generate a scale-space. This is, however, not the case. It is of crucial importance that no new structures (i.e, that do not correspond to simplifications of corresponding structures at a finer scale) are introduced at the coarse scales. The Gaussian filter is unique for generating a linear scale-space based on this essential requirement [78, 46, 90].

B.1 Gaussian filtering in practice

Gaussian filtering in practice is done using separable filters to reduce the computational overhead. For good results, in the discrete signal case binomial filters [15] are used as they approximate the Gaussian filters well for small values of σ. In addition, they using shift and additions instead of computationally expensive division operation (used for normalization). However, for large values of σ, the use of the binomial kernel or a discrete approximation of the Gaussian kernel (Eq. B.1) becomes computationally expensive. In such cases, recursive filtering approaches are far more advantageous as they use a small constant number of operations for filtering for any σ value. The most popular approaches for this are by Deriche [35] and Young et al. [148]. In this thesis we use the latter method along with the correct boundary conditions proposed by Triggs and Sdika [129].

B.2 The relation between sigma and cut-off frequency

The Fourier transform of a one-dimensional normalized Gaussian function

$$g(x,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{B.3}$$

is given by

$$F(\omega) = e^{-\frac{1}{2}\omega^2\sigma^2}. \tag{B.4}$$

Knowing that power is $20log_{10}(F(\omega))$, the relationship between cut-off frequency ω_c and standard deviation σ of the Gaussian at t_c (we use 6db) is:

$$\omega_c = \sqrt{\frac{t_c log_e 10}{10\sigma^2}} \tag{B.5}$$

Appendix C

Mitochondria Segmentation using SLIC Superpixels

In this chapter we present a more sophisticated mitochondria[1] segmentation algorithm than the one presented in Section 6.4.3. For more details the interested reader is suggested to refer to the work of Lucchi et al. [97].

C.1 Segmenting objects in Electron Microscope imagery

Mitochondria segmentation belongs to the specialized category of segmenting objects in EM imagery. State-of-the-art algorithms that perform well on natural image segmentation benchmarks such as the Berkeley dataset [102] or Pascal VOC dataset [43] do not perform as well when applied to Electron Microscope (EM) imagery. This is because the image features they rely on are not sufficiently discriminative for segmenting cellular structures such as mitochondria.

Mitochondria and similar objects in EM imagery exhibit shapes that are not easily captured using standard techniques of shape abstraction. Their texture can easily be confused with that of other cellular structures like endoplasmic reticula. Mitochondrial boundaries, for instance, are difficult to distinguish from other membranes that share a similar appearance. For such a task of segmenting objects in EM imagery, all available visible cues should be taken into account simultaneously.

An example of generic object detection, TextonBoost [123], uses sophisticated texture and boundary cues although only simple haar-like rectangular features are used to capture shape [123]. Fulkerson et al. [52] compute SIFT descriptors [95] on superpixels of the image to capture local texture and gradient information, but shape information is not taken into account.

For segmenting neural EM imagery, Frangakis and R. Hegerl [48] use a normalized cuts based approach. More recently, Vazquez-Reina et al. [3] use a level set approach for this task. However, this approach is sensitive to initialization and

[1]Images courtesy of Graham Knott: graham.knott@epfl.ch

allows dealing with only a single object at a time. Thévenaz et al. [125] use an active contour approach that is aimed to detect elliptical blobs but it fails to segment mitochondria, which can assume various non-ellipsoidal shapes. Andres et al. [11] rely upon a convolutional neural network in their approach. However, it only exploits local information obtained from using a watershed supervoxel segmentation. In yet another approach to detect mitochondria, Narashimha et al. [112] use texton features to learn mitochondrial texture but ignore shape information.

C.2 Mitochondria detection and segmentation

Like the method presented in Section 6.4.3, this method also relies on graph-cuts to minimize a cost function as in Eq. 2.9. The difference is that more sophisticated features are used for training the SVM and such training is performed for the pairwise terms ϕ also in Eq. 2.9. The pairwise term used in this case is a more general one:

$$\phi(c_i, c_j|x_i, x_j) = \begin{cases} \frac{1}{1+P(c_i,c_j|x_i,x_j)} & \text{, if } c_i \neq c_j \text{ ,} \\ 0 & \text{, otherwise.} \end{cases} \tag{C.1}$$

that uses the predictions $P(c_i, c_j|x_i, x_j)$ of the pairwise SVM training.

The feature vector $\mathbf{f}$ used for training combines three types of features:

$$\mathbf{f} = [\mathbf{f}^{\mathrm{Ray}^{\mathrm{T}}}, \mathbf{f}^{\mathrm{Rot}^{\mathrm{T}}}, \mathbf{f}^{\mathrm{Hist}^{\mathrm{T}}}]^{\mathrm{T}} \text{ ,} \tag{C.2}$$

where $\mathbf{f}^{\mathrm{Ray}}$ represents Ray features that capture semi-local object shape [124], $\mathbf{f}^{\mathrm{Rot}}$ stands for rotational features describing texture and boundaries [58], and $\mathbf{f}^{\mathrm{Hist}}$ represents histograms describing the local intensity. These features, as shown in Fig. C.1, are explained below.

Ray Descriptors describe the shape of local objects for each point in the image in a way that standard shape modeling techniques can not. Typically, other methods represent object shape using contour templates [10] or fragment codebooks [81]. While these approaches can successfully segment a single object with known shape, they tend to fail when the shape is highly variable or when many objects appear in the image.

For a given point x_i in the image, four types of Ray features are extracted by projecting rays from x_i at regular angles $\boldsymbol{\Theta} = \{\theta_1, \ldots, \theta_N\}$ and stopping when they intersect a detected edge (r) [124]. The distance from x_i to r form the first type of feature f_{dist}. The other three types of features compare the relative distance from x_i to r for rays in two different directions (f_{diff}), measure the gradient strength at r (f_{norm}), and measure the gradient orientation at r relative to the ray (f_{ori}). While [124] uses individual Ray features as AdaBoost learners, we aggregate all features extracted for a single point into a *Ray descriptor* $\mathbf{f}^{\mathrm{Ray}} = [f_{\mathrm{dist}}\ f_{\mathrm{diff}}\ f_{\mathrm{norm}}\ f_{\mathrm{ori}}]^T$. We make it rotation invariant by shifting the descriptor elements so that the first

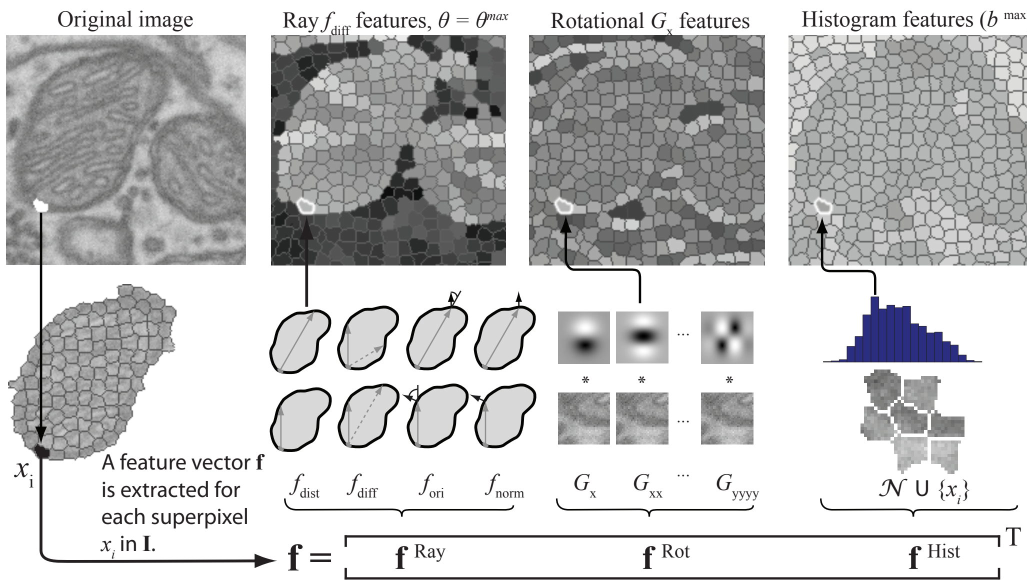

Original image
Ray f_{diff} features, $\theta = \theta^{max}$
Rotational G_x features
Histogram features (b^{max})
x_i
A feature vector **f** is extracted for each superpixel x_i in **I**.
f_{dist} f_{diff} f_{ori} f_{norm}
G_x G_{xx} ... G_{yyyy}
$\mathcal{N} \cup \{x_i\}$
$\mathbf{f} = [\mathbf{f}^{\text{Ray}} \quad \mathbf{f}^{\text{Rot}} \quad \mathbf{f}^{\text{Hist}}]^{\text{T}}$

Figure C.1: *For each superpixel, the SVM classifiers in Eqs. 6.6 and C.1 predict the presence of mitochondria based on a feature vector* $\mathbf{f}$ *we extract.* $\mathbf{f}$ *captures shape cues with a Ray descriptor* $\mathbf{f}^{Ray}$*, texture and boundary cues with rotational features* $\mathbf{f}^{Rot}$*, and intensity cues in* $\mathbf{f}^{Hist}$*.*

element corresponds to the longest ray.

Rotational Features capture texture and image cues indicating boundaries such as edges, ridges, crossings and junctions [58]. They are projections of image patches around a superpixel center x_i into the space of Gaussian derivatives at various scales, rotated to a local orientation estimation for rotational invariance.

Histograms complement $\mathbf{f}^{\text{Ray}}$ and $\mathbf{f}^{\text{Rot}}$ with simple intensity cues from superpixel x_i's neighborhood $\mathcal{N}$.

Unlike the approach presented in Section 6.4.3, SVM training is performed for boundary prediction also. This is done using features extracted from pairs of superpixels containing true object boundaries. The pairwise feature vector $\mathbf{f}_{i,j}$ is a concatenation of $\mathbf{f}_i$ and $\mathbf{f}_j$ extracted from each superpixel $\mathbf{f}_{i,j} = [\mathbf{f}_i^{\text{T}}, \ \mathbf{f}_j^{\text{T}}]^{\text{T}}$. The scheme of the detection and segmentation is presented in Fig. C.2.

C.3 Mitochondria segmentation using SLIC supervoxels

The supervoxels generated by our technique (Section 6.5) have also been used for detection and segmentation of mitochondria in EM image volumes. The methodology remains the same as in Section C.2. The difference is that a three-dimensional version of the Ray features [124] is used. Fig. C.3 illustrates supervoxel segmentation and mitochondria detection in a volume image.

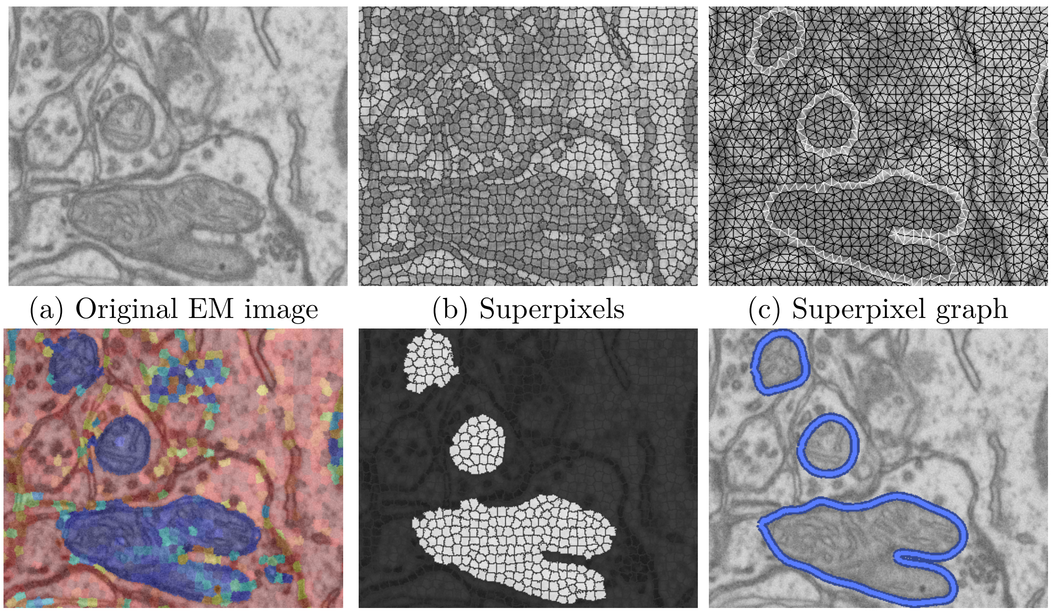
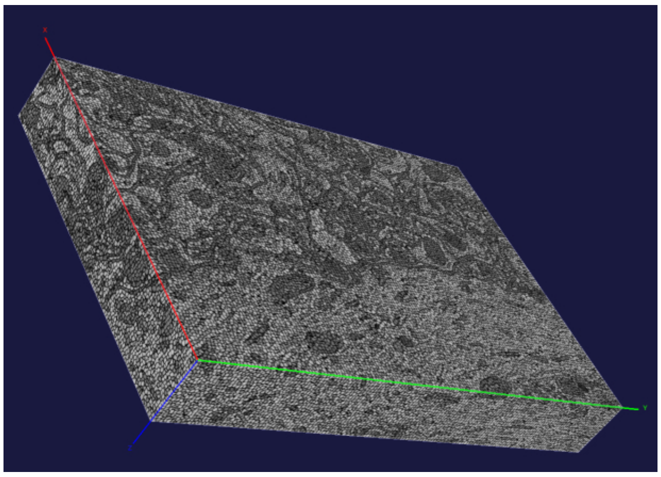
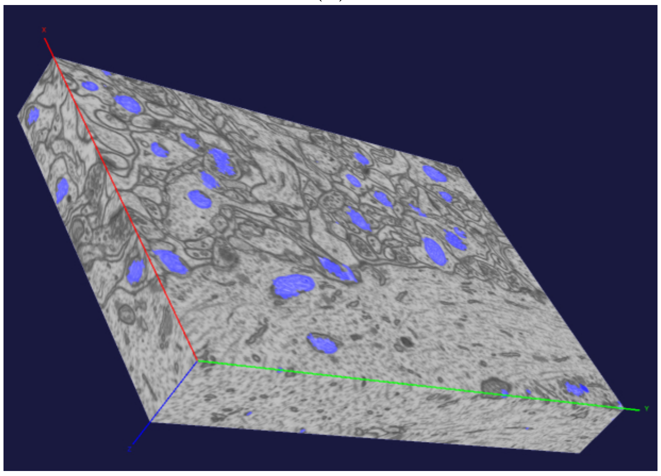


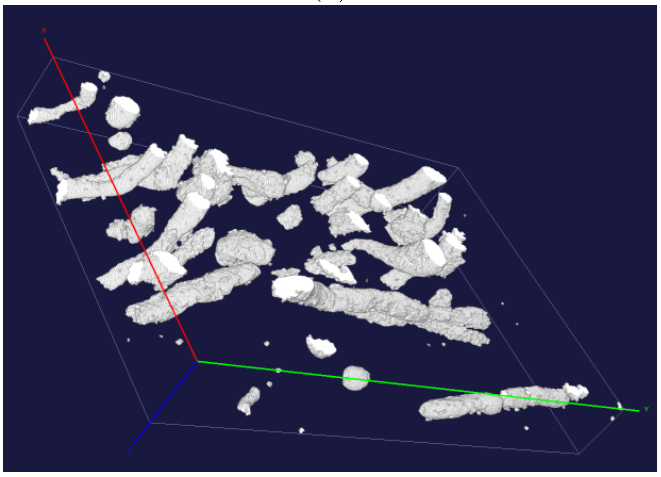
Figure C.2: *The scheme of object detection presented in Section C.2.(a) A portion of an original EM image. (b) Superpixel over-segmentation. (c) Graph defined over superpixels. White edges indicate pairs of superpixels used to train an SVM that predicts mitochondrial boundaries. (d) SVM prediction where blue indicates a probable mitochondrion. (e) Graph cut segmentation. (f) Final results after automated post-processing. Note: the same image is used in this figure for clarity; images in the training and testing sets are disjoint. The method described in Section 6.4.3 follows the same steps except that the features used are different and there is no training performed for the pairwise term.*



(a)



(b)



(c)

Figure C.3: *(a) Supervoxel segmentation of an EM volume. (b) Blue regions show detected mitochondria on three of the surfaces. (c) The detected mitochondria binarized and shown with maximum intensity projection.*

Bibliography

[1] "Diadem challenge," http://www.diademchallenge.org/.

[2] "Digital still camera image file format standard (exchangeable image file format for digital still cameras: Exif) Version 2.1, Specification by JEITA," June 1998.

[3] A. Vazquez-Reina, E. Miller, and H. Pfister, "Multiphase Geometric Couplings for the Segmentation of Neural Processes," in Proc. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[4] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Salient region detection and segmentation," *International Conference on Computer Vision Systems*, vol. 5008, pp. 66–75, 2008.

[5] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk, "Frequency-tuned salient region detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1597–1604, June 2009.

[6] R. Achanta, A. Shaji, P. Fua, and S. Süsstrunk, "Image summaries using database saliency," *ACM Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia*, 2009.

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels," École Polytechnique Fédérale de Lausanne, Tech. Rep. EPFL-REPORT-149300, 2010.

[8] R. Achanta and S. Süsstrunk, "Saliency Detection for Content-aware Image Resizing," in Proc. *IEEE International Conference on Image Processing*, 2009.

[9] ——, "Saliency Detection using Maximum Symmetric Surround," in Proc. *IEEE International Conference on Image Processing*, 2010.

[10] A. Ali, A. Farag, and A. El-Baz, "Graph cuts framework for kidney segmentation with prior shape constraints," *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2007.

[11] B. Andres, U. Koethe, M. Helmstaedter, W. Denk, and F. Hamprecht, “Segmentation of Sbfsem Volume Data of Neural Tissue by Hierarchical Classification,” in Proc. *Symposium of the German Association for Pattern Recognition (DAGM)*, 2008.

[12] L. H. Armitage and P. G. B. Enser, “Analysis of user need in image archives,” *Journal of Information Science*, vol. 23, pp. 287–299, 1997.

[13] D. Arthur and S. Vassilvitskii, “k-means++: the advantages of careful seeding,” in Proc. *ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007, pp. 1027–1035.

[14] C. B. Atkins, “Blocked recursive image composition,” in Proc. *ACM international conference on Multimedia*, 2008, pp. 821–824.

[15] M. Aubury, “Binomial filters,” *Journal of VLSI Signal Processing*, vol. 12, pp. 35–50, 1996.

[16] S. Avidan and A. Shamir, “Seam carving for content-aware image resizing,” *ACM Transactions on Graphics*, vol. 26, no. 3, p. 10, July 2007.

[17] A. Ayvaci and S. Soatto, “Motion segmentation with occlusions on the superpixel graph,” in Proc. *Workshop on Dynamical Vision, Kyoto, Japan*, October 2009.

[18] M. Aziz and B. Mertsching, “Fast and robust generation of feature maps for region-based visual attention,” *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 633–644, May 2008.

[19] X. Bai and G. Sapiro, “A geodesic framework for fast interactive image and video segmentation and matting,” *IEEE International Conference on Computer Vision*, vol. 0, pp. 1–8, 2007.

[20] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[21] P. Bian and L. Zhang, “Biological plausibility of spectral domain approach for spatiotemporal visual saliency,” *Advances in Neuro-Information Processing*, vol. 5506/2009, pp. 251–258, 2009.

[22] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[23] Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images,” *IEEE International Conference on Computer Vision*, vol. 1, pp. 105–112, July 2001.

[24] N. Bruce and J. Tsotsos, "Saliency based on information maximization," *Advances in Neural Information Processing Systems*, vol. 18, pp. 155–162, 2006.

[25] ——, "Attention based on information maximization," *Journal of Vision*, vol. 7, no. 9, pp. 950–950, 6 2007.

[26] L. Chen, X. Xie, X. Fan, W.-Y. Ma, H.-J. Zhang, and H. Zhou, "A visual attention model for adapting images on small displays," *ACM Transactions on Multimedia Systems*, vol. 9, pp. 353–364, November 2003.

[27] X. Chen and A. Yuille, "Detecting and reading text in natural scenes," in Proc. *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, june 2004, pp. 366–373.

[28] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, p. 790Ű799, 1995.

[29] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in low level vision," *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pp. 150–155 vol.4, 2002.

[30] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[31] Consumer Electronics Association, "Digital imaging: A focus on sharing," http://www.ce.org.

[32] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in Proc. *IEEE Computer Vision and Pattern Recognition*, vol. 2, June 2005, pp. 1124–1131 vol. 2.

[33] J. L. Crowley, O. Riff, and J. Piater, "Fast computation of characteristic scale using a half octave pyramid," *International Workshop on Cognitive Computing*, October 2002.

[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893 vol. 1, June 2005.

[35] R. Deriche, "Recursively implementing the gaussian and its derivatives," *International Conference on Image Processing*, pp. 263–267, 1992.

[36] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, p. 269Ű271, 1959.

[37] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* Wiley-Interscience, November 2000.

[38] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, p. 248Ű264, 1972.

[39] C. Elkan, "Using the triangle inequality to accelerate k-means," *International Conference on Machine Learning*, 2003.

[40] T. Ell and S. Sangwine, "Hypercomplex fourier transforms of color images," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 22–35, January 2007.

[41] P. Enser, "Query analysis in a visual information retrieval context," *Journal of Document and Text Management*, vol. 1, pp. 25–39, 1993.

[42] ——, "Progress in documentation: Pictorial information retrieval," *Journal of Documentation*, vol. 51, no. 2, pp. 126–170, 1995.

[43] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results."

[44] N. Ezaki, M. Bulacu, and L. Schomaker, "Text detection from natural scene images:towards a system for visually impaired persons," in Proc. *17th International Conference on Pattern Recognition (ICPR'04)*, vol. 2, 2004, pp. 683 – 686.

[45] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. 167–181, September 2004.

[46] L. Florack, B. Romeny, J. Koenderink, and M. Viergever, "Scale and the differential structure of images," *Image and Vision Computing*, vol. 10, pp. 376–388, 1992.

[47] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399Ű404, 1956.

[48] A. Frangakis and R. Hegerl, "Segmentation of two- and three-dimensional data from electron microscopy using eigenvector analysis," in Proc. *Journal of Structural Biology*, vol. 138, 2002, pp. 105–113.

[49] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," in Proc. *European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[50] S. Frintrop, M. Klodt, and E. Rome, "A real-time visual attention system using integral images," in Proc. *International Conference on Computer Vision Systems (ICVS)*, March 2007.

[51] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, p. 32Ű40, 1975.

[52] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in Proc. *IEEE International Conference on Computer Vision)*, 2009.

[53] R. Gal, O. Sorkine, and D. Cohen-Or, "Feature-aware texturing," *Eurographics Symposium on Rendering*, pp. 297–303, 2006.

[54] D. Gao, V. Mahadevan, and N. Vasconcelos, "On the plausibility of the discriminant center-surround hypothesis for visual saliency," *Journal of Vision*, vol. 8, no. 7, pp. 1–18, 6 2008.

[55] D. Gao and N. Vasconcelos, "Discriminant saliency for visual recognition from cluttered scenes," *Advances in Neural Information Processing Systems*, pp. 481–488, 2004.

[56] ——, "Integrated learning of saliency, complex features, and object detectors from cluttered scenes," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 282–287, 2005.

[57] J. Gllavata, R. Ewerth, and B. Freisleben, "Finding text in images via local thresholding," in Proc. *IEEE International Symposium on Signal Processing and Information Technology(ISSPIT 2003)*, December 2003, pp. 539–542.

[58] G. Gonzalez, F. Fleuret, and P. Fua, "Learning rotational features for filament detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1582–1589, 2009.

[59] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision (IJCV)*, vol. 80, no. 3, pp. 300–316, 2008.

[60] C. Guo, Q. Ma, and L. Zhang, "Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform," in Proc. *IEEE Computer Vision and Pattern Recognition*, 2008.

[61] J. Han, K. N. Ngan, M. Li, and H.-J. Zhang, "Unsupervised extraction of visual attention objects in color images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 1, pp. 141–145, January 2006.

[62] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," *Advances in Neural Information Processing Systems*, pp. 545–552, 2007.

[63] X. He, R. Zemel, and D. Ray, "Learning and incorporating top-down cues in image segmentation," in Proc. *European Conference on Computer Vision*, 2006, pp. 338–351.

[64] S. S. Hemami and T. N. Pappas, "Perceptual metrics for image quality evaluation," Tutorial presented at Human Vision and Electronic Imaging, 2007.

[65] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," in Proc. *ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, August 2005.

[66] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in Proc. *IEEE International Conference on Computer Vision*, vol. 1. IEEE, October 2005, pp. 654 – 661.

[67] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," *IEEE Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[68] Y. Hu, X. Xie, W.-Y. Ma, L.-T. Chia, and D. Rajan, "Salient region detection using weighted feature maps based on the human visual attention model," *Springer Lecture Notes in Computer Science*, vol. 3332, no. 2, pp. 993–1000, October 2004.

[69] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted voronoi diagrams and randomization to variance-based k-clustering," *Proceedings of the tenth annual symposium on Computational geometry*, pp. 332–339, 1994.

[70] L. Itti and P. F. Baldi, "Bayesian surprise attracts human attention," in Proc. *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA: MIT Press, 2005, pp. 547–554.

[71] L. Itti and C. Koch, "Comparison of feature combination strategies for saliency-based visual attention systems," *SPIE Human Vision and Electronic Imaging IV (HVEI)*, pp. 473–482, May 1999.

[72] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, November 1998.

[73] T. Kadir and M. Brady, "Saliency, scale and image description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[74] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in Proc. *European Conference on Computer Vision*, 2004.

[75] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," *Annual symposium on Computational geometry*, pp. 10–18, 2002.

[76] B. C. Ko and J.-Y. Nam, "Object-of-interest image segmentation based on human attention and semantic region clustering," *Journal of Optical Society of America A*, vol. 23, no. 10, pp. 2462–2470, October 2006.

[77] C. Koch and S. Ullman, "Shifts in selective visual attention: Towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 219–227, 1985.

[78] J. Koenderink, "The structure of images." *Biological Cybernetics*, vol. 50, pp. 363–370, 1984.

[79] A. Kumar, Y. Sabharwal, and S. Sen, "A simple linear time (1+e)-approximation algorithm for k-means clustering in any dimensions," *IEEE Symposium on Foundations of Computer Science*, vol. 0, pp. 454–462, 2004.

[80] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[81] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," *European Conference on Computer Vision*, 2006.

[82] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," in Proc. *IEEE International Conference on Computer Vision*, September 2009.

[83] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[84] H. Li and D. Doermann, "Automatic text tracking in digital videos," *IEEE Workshop on Multimedia Signal Processing*, pp. 21–26, 1998.

[85] H. Li, D. Doermann, and O. Kia, "Automatic text detection and tracking in digital videos," *IEEE Transactions on Image Processing*, vol. 9, pp. 147–156, January 2000.

[86] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 303–308, 2004.

[87] R. Lienhart and W. Effelsberg, "Automatic text segmentation and text recognition for video indexing," *Multimedia Systems*, vol. 8, pp. 69 – 81, 2000.

[88] R. Lienhart, L. Liang, and A. Kuranov, "A detector tree of boosted classifiers for real-time object detection and tracking," in Proc. *IEEE Conference on Multimedia and Expo*, vol. 2. IEEE Computer Society, 2003, pp. 277–280.

[89] T. Lindeberg, “Scale space for discrete images,” in Proc. *Scandinavian Conference on Image Analysis (SCIA)*, 1989, pp. 1098–1107.

[90] ——, “Scale-space theory: A basic tool for analysing structures at different scales,” *Journal of Applied Statistics*, vol. 21(2), pp. 224–270, 1994.

[91] ——, “Feature detection with automatic scale selection,” *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, November 1998.

[92] S.-C. Liu, C.-W. Fu, and S. Chang, “Statistical change detection with moments under time-varying illumination,” *IEEE Transactions on Image Processing*, vol. 7, no. 9, pp. 1258–1268, Sep 1998.

[93] T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum, “Learning to detect a salient object,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[94] S. P. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, vol. IT-28, no. 2, pp. 129–137, March 1982.

[95] D. G. Lowe, “Distinctive image features from scale-invariant feature points,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[96] S. M. Lucas, “Text locating competition results,” *International Conference on Document Analysis and Recognition (ICDAR)*, vol. 0, pp. 80–85, 2005.

[97] A. Lucchi, K.Smith, R. Achanta, V. Lepetit, and P. Fua, “A fully automated approach to segmentation of irregularly shaped cellular structures in em images,” *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2010.

[98] Y.-F. Ma and H.-J. Zhang, “Contrast-based image attention analysis by using fuzzy growing,” *ACM International Conference on Multimedia*, pp. 374–381, November 2003.

[99] V. Mahadevan and N. Vasconcelos, “Background subtraction in highly dynamic scenes,” *IEEE Conference on Computer Vision and Pattern Recognition*, June 2008.

[100] M. Mancas, B. Gosselin, and B. Macq, “Perceptual image representation,” *Journal of Image and Video Processing*, vol. 2007, no. 2, pp. 3–3, 2007.

[101] D. Marr, *Vision: a computational investigation into the human representation and processing of visual information.* W. H. Freeman, San Francisco, 1982.

[102] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and

measuring ecological statistics," *IEEE International Conference on Computer Vision*, vol. 2, pp. 416–423, July 2001.

[103] Microsoft Research Visual Computing Group, "MSRA Salient Object Database," http://research.microsoft.com/en-us/um/people/jiansun/SalientObject/salient_object.htm.

[104] S. Mitri, S. Frintrop, K. Pervölz, H. Surmann, and A. Nüchter, "Robust object detection at regions of interest with an application in ball recognition," in Proc. *IEEE International Conference on Robotics and Automation*, 2005.

[105] M.Léon and A.Gasull, "Text detection in images and video sequences," in Proc. *IADAT International Conference on Multimedia, Image Processing and Computer Vision*, Madrid, Spain, march 2005.

[106] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel Lattices," *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[107] G. Mori, "Guiding model search using segmentation," in Proc. *IEEE International Conference on Computer Vision*, 2005, pp. 1417–1423.

[108] E. Niebur and C. Koch, *The Attentive Brain.* Cambridge MA:MIT Press, October 1995, ch. Computational architectures for attention, pp. 163–186.

[109] A. Oliva, A. Torralba, M. Castelhano, and J. Henderson, "Top-down control of visual attention in object detection," *International Conference on Image Processing*, pp. 253–256, 2003.

[110] B. Olshausen, C. Anderson, and D. Van Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *Journal of Neuroscience*, vol. 13, pp. 4700–4719, 1993.

[111] G. Pass and R. Zabih, "Histogram refinement for content-based image retrieval," *IEEE Workshop on Applications of Computer Vision*, p. 96, 1996.

[112] R. Narashimha, H. Ouyang, A. Gray, S. McLaughlin, and S. Subramaniam, "Automatic Joint Classification and Segmentation of Whole Cell 3D Images," *Journal of Pattern Recognition*, vol. 42, no. 2009, pp. 1067–1079, 2007.

[113] X. Ren and J. Malik, "Learning a classification model for segmentation," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 10–17, 2003.

[114] R. A. Rensink, "The dynamic representation of scenes," *Visual Cognition*, 2000.

[115] ——, "Seeing, sensing, and scrutinizing," *Vision Research*, pp. 469–1487, 2000.

[116] P. L. Rosin, "A simple method for detecting salient regions," *Pattern Recognition*, vol. 42, no. 11, pp. 2363 – 2371, 2009.

[117] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake, "Autocollage," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 847–852, 2006.

[118] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[119] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–9, August 2008.

[120] H. J. Seo and P. Milanfar, "Static and space-time visual saliency detection by self-resemblance," *Jounal of Vision*, vol. 9, no. 12, pp. 1–27, November 2009.

[121] 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, Aug 2000.

[122] A. S. Shlmon and S. Ullman, "Structural saliency: The detection of globally salient structures using a locally connected network," *IEEE International Conference on Computer Vision*, 1998.

[123] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation," in Proc. *European Conference on Computer Vision*, 2006.

[124] K. Smith, A. Carleton, and V. Lepetit, "Fast ray features for learning irregular shapes," *IEEE International Conference on Computer Vision*, 2009.

[125] P. Thévenaz, R. Delgado-Gonzalo, and M. Unser, "The Ovuscule," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[126] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search," *Psychological Review*, vol. 113, pp. 766–786, 2006.

[127] H. Tran, A. Lux, H. L. N. T., and A. Boucher, "A novel approach for text detection in images using structural features." in Proc. *International conference on advances in pattern recognition*, August 2005, pp. 627–635.

[128] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, no. 1, pp. 97–136, January 1980.

[129] B. Triggs and M. Sdika, “Boundary conditions for young-van vliet recursive filtering,” *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2365–2367, June 2006.

[130] W. Tsai, “Moment-preserving thresholding: A new approach,” *Computer Vision Graphics and Image Processing*, vol. 29, pp. 377–393, 1985.

[131] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo, “Modeling visual attention via selective tuning,” *Artificial Intelligence*, vol. 78, no. 1-2, pp. 507–545, 1995.

[132] U.C. Berkeley Computer Vision Group, “The Berkeley Segmentation Dataset and Benchmark,” http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[133] V. Vanhoucke and S. B. Gokturk, “Reading text in consumer digital photographs,” in Proc. *SPIE Document Recognition and Retrieval XIV*, vol. 6500, 2007.

[134] A. Vedaldi and S. Soatto, “Quick shift and kernel methods for mode seeking,” in Proc. *European Conference on Computer Vision*, 2008.

[135] O. Verevka and J. Buchanan, “Local k-means algorithm for color image quantization,” *Graphics Interface*, pp. 128–135, 1995.

[136] L. Vincent and P. Soille, “Watersheds in digital spaces: An efficient algorithm based on immersion simulations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[137] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, December 2001.

[138] P. Viola, M. Jones, and D. Snow, “Detecting pedestrians using patterns of motion and appearance,” Mitsubishi Electric Research Lab Technical Report., Tech. Rep. TR-2003-90, 2003.

[139] P. Viola and M. Jones, “Robust real-time object detection,” Cambridge Research Laboratory, Tech. Rep. CRL 2001/01, February 2001.

[140] D. Walther and C. Koch, “Modeling attention to salient proto-objects,” *Neural Networks*, vol. 19, no. 9, pp. 1395–1407, August 2006.

[141] Y.-S. Wang, C.-L. Tai, O. Sorkin, and T.-Y. Lee, “Optimized scale-and-stretch for image resizing,” *ACM Transactions on Graphics*, vol. 27, no. 5, December 2008.

[142] Z. Wang and B. Li, "A two-stage approach to saliency detection in images," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 965–968, 31 2008-April 4 2008.

[143] A. P. Witkin, "Scale-space filtering," in Proc. *International Joint Conference on Artificial Intelligence*, 1983, pp. 1019–1022.

[144] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," *IEEE International Conference on Computer Vision*, pp. 1–6, Oct. 2007.

[145] V. Wu, R. Manmatha, and E. M. Riseman, "Finding text in images," in Proc. *ACM international conference on Digital libraries*, 1997, pp. 3–12.

[146] J. Xi, X.-S. Hua, X.-R. Chen, L. Wenyin, and H.-J. Zhang, "A video text detection and recognition system," in Proc. *IEEE International Conference on Multimedia and Expo*, 2001, p. 222.

[147] L. Yatziv, A. Bartesaghi, and G. Sapiro, "O(N) implementation of the fast marching algorithm," *Journal of Computational Physics*, vol. 212, pp. 393–399, March 2006.

[148] I. T. Young and L. J. van Vliet, "Recursive implementation of the gaussian filter," *Signal Processing*, vol. 44, no. 2, pp. 139–151, 1995.

[149] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: A Bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, pp. 1–20, 12 2008.

Curriculum Vitae

Radhakrishna Achanta

Images and Visual Representation Group (IVRG/LCAV)
School of Computer and Communication Sciences (I&C)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

Email: `radhakrishna.achanta@a3.epfl.ch`
Web: `http://ivrg.epfl.ch/~achanta`

Education

2006 - 2010 **Ph.D.** in Computer and Communication Sciences, IVRG/LCAV, I&C, EPFL, Switzerland.

2000 - 2002 **M.Sc.** in Computer Science, National University of Singapore, Singapore.

1995 - 1999 **B.E.** in Electrical Engineering, Government Engineering College Jabalpur, India.

Professional Experience

2006 - 2010 **Teaching Assistant**, EPFL, Switzerland.
Assisted students in undergraduate and graduate courses.
Supervised semester and master projects.

2004 - 2006 **Software Engineer**, Muvee Technologies Private Limited, Singapore.
Worked on the core engine of the automatic video editing software that extracts features and optimizes the choice of audio, image, and video input by the user.

2002 - 2004 **Research Assistant**, National University of Singapore, Singapore.
Worked on the Digital Image and Video Album project that was aimed to automatically annotate image and video content.

Skills

1. Programming skills: C, C++, x86 Assembler, Java, Jasmin, HTML, JavaScript, Verilog, VHDL.
2. Tools and Libraries: MATLAB, MFC, DirectX, MPEGDC, MSSG Encoder, Intel OpenCV.
3. Natural Languages: English, Hindi, Telugu, Tamil and French.

Awards and Honors

1. Cleared first round of VentureLab 2007 Start-up competition in Lausanne.
2. Best Executive Summary award in StartUp@Singapore business plan contest. Led team of 6.
3. Qualified in first phase of Junior and Senior Level National Mathematics Olympiad.
4. Qualified in first stage of National Talent Search Exam. in Physics.
5. Merit Certificate for 99% marks in Mathematics in All India Secondary School Exam(AISSE).
6. Black Belt 1st dan in (Shitoryu) Karate (2006).

Acivities

1. Contributed text detection code to the open source project of ImageCerbus, a plug-in for the spam filter SpamAssassin.
2. Member of National University of Singapore Karate club, Singapore.
3. Cadet in the National Cadet Corps (NCC), India.
4. Member of Electrical Society (GEC Jabalpur), India.
5. Current Interests: Tango, Yoga, Traveling, Motorbiking.

Publications

1. R. Achanta and S. Süsstrunk, "Saliency Detection using Maximum Symmetric Surround," in Proc. *IEEE International Conference on Image Processing*, Sep. 2010.
2. A. Lucchi, K. Smith, R. Achanta, V. Lepetit, and P. Fua, "A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Structures in EM Images," in Proc. *International Conference on Medical Image Computing and Computer Assisted Intervention*, Sep. 2010.

3. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “SLIC Superpixels,” EPFL Technical Report no. 149300, Jun. 2010.

4. A. Guidi, R. Achanta, C. Fredembach, and S. Süsstrunk, “GUI-Aided NIR and Color Image Blending,” in Proc. *IEEE Mediterranean Electrotechnical Conference*, Apr. 2010.

5. R. Achanta, A. Shaji, P. Fua, and S. Süsstrunk, “Image Summaries using Database Saliency,” in Proc. *ACM SIGGRAPH ASIA*, Dec. 2009.

6. R. Achanta and S. Süsstrunk, “Saliency Detection for Content-aware Image Resizing,” newblock in Proc. *IEEE International Conference on Image Processing*, Nov. 2009.

7. R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk, “Frequency-tuned Salient Region Detection,” in Proc. *IEEE International Conference on Computer Vision and Pattern Recognition*, Jun. 2009.

8. R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, “Salient Region Detection and Segmentation,” in Proc. *International Conference on Computer Vision Systems*, May 2008.

9. R. Achanta, Y. Weiqi, J. Yi, M. S. Kankanhalli, “Modeling Intents for Home Video Repurposing,” in Proc. *IEEE Multimedia Magazine*, 2005.

10. J. Wang, R. Achanta, M.S. Kankanhalli, and M.J.T. Reinders, “A sensor fusion approach for tracking faces in compressed video,”, in Proc. *Eleventh annual conference of the Advanced School for Computing and Imaging*, 2005.

11. C. Madhwacharyula, W. Jun, Y. Weiqi, J. Yi, A.S.V Radhakrishna, S. Bissol, J. Charlson Yu, Z.Qiuying, S.H Srinivasan, H. Hassan Abdulredha, P. Mulhem, M. S. Kankanhalli, “An Information-Integration Approach to Designing Digital Video Albums,” in Proc. *IEEE Pacific-Rim Conference on Multimedia*, December 2003.

12. J. Wang, R. Achanta, M. Kankanhalli, “A Hierarchical Framework For Face Tracking Using State Vector Fusion For Compressed Video,” in Proc. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2003.

13. R. Achanta, J. Wang, M. Kankanhalli, “A Sensor Fusion Based Object Tracker for Compressed Video,” in Proc. *International Workshop on Advanced Image Technology*, January 2003.

14. R. Achanta, M. Kankanhalli, P. Mulhem, “Object Tracking in Compressed Domain MPEG Video for Automated Content-Based Indexing,” in Proc. *IEEE International Conference on Multimedia and Expo*, Aug. 2002.