POLITECNICO DI TORINO

Master's Degree in ICT FOR SMART SOCIETIES

Master's Degree Thesis

Comparison of min-cut models for image segmentation

Supervisors	Candidate
Prof. EDOARDO FADDA	**GUOXIN WANG**

March 2024

Summary

Image segmentation is a core task in computer vision, playing a crucial role in object recognition, image analysis, and various practical applications, such as medical imaging, autonomous driving, and surveillance. It involves dividing an image into meaningful segments, often called objects or foregrounds, based on specific properties. Image segmentation based on the s/t graph cut is a powerful approach that combines region and boundary information. It transforming the image segmentation problem into an energy function minimization problem and applying the Min-Cut algorithm to find a globally optimal solution. S/t graph cut methods are highly efficient and accurate in various segmentation scenarios, making them widely applicable to N-dimensional (N-D) problems.

This thesis introduces the basic concepts of s/t graphs and their applications in combinatorial optimization. We explore several typical graph construction methods and their parameter settings, demonstrating how to build effective Min-Cut models using these parameters. We then detail two mathematical models that define the minimum s/t cut problem from different perspectives. Following this, we cover the fundamentals and usage of the Gurobi optimization solver, explaining how to convert these mathematical models into a format compatible with Gurobi. By leveraging Gurobi's powerful algorithms, we find the global minimum of the energy function, ensuring optimal segmentation results. In the experimental section, we assess the performance of these models through real-world image segmentation tasks, evaluating their effectiveness with various metrics.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Image segmentation is a crucial aspect of computer vision and a fundamental topic in image analysis. It involves dividing an image into meaningful segments that possess unique characteristics. The primary goal of segmentation is to isolate these objects for more detailed analysis and processing. This process allows for a deeper understanding and manipulation of specific parts of an image, such as recognizing objects or editing images.

Image segmentation splits an image into several disjoint regions based on features such as grayscale, color, spatial texture, and geometric shape. The goal is to ensure consistency or similarity within the same region while ensuring significant differences between different regions. More precisely, image segmentation assigns labels to each pixel in an image, where pixels with the same label share certain common features. The purpose of image segmentation is to simplify the complexity of an image or alter its presentation to make it easier to understand and analyze.

Image segmentation is not only valuable in itself but also provides the basis for other image processing tasks. For example, accurate image segmentation is essential for object detection and tracking, helping in identification and localization. According to Nair et al. [1] and Dumouchel et al. [2], in the field of medical imaging, image segmentation helps locate lesions, extract anatomical structures, and support disease diagnosis and treatment. In industry, applications include mineral deposit analysis, contactless inspection, and product accuracy and purity analysis [3]. In transportation [4], applications include autonomous driving [5], vehicle detection [6], vehicle type recognition [7], and vehicle tracking. Additionally, image segmentation has a wide range of applications in image editing [8], virtual reality, and many other fields.

Over the years, numerous methods for performing image segmentation have been

developed, utilizing domain-specific knowledge to effectively solve segmentation problems within specific application areas, reflecting the diversity of approaches.

Image segmentation techniques can be broadly categorized into several groups. Traditional methods include threshold-based segmentation [9], edge-based segmentation [10], and region-growing algorithms [11]. Graph-based methods such as the minimum cut [12] and the normalized cut [13] were introduced by Shi and Malik in 2000. Pixel clustering based on specific theories includes methods such as K-means [14], spectral clustering [15], mean shift [16], and SLIC [17]. There is also deep learning-based image segmentation. For example, Convolutional Neural Networks (CNNs) [18] have led advancements in semantic segmentation, instance segmentation, and full-view segmentation. The Fully Convolutional Networks (FCN) [19] approach and Generative Adversarial Networks (GANs) [20], proposed by Long et al. in 2014, have been used for image segmentation and data enhancement.

Although image segmentation technologies advance rapidly and show great potential in various fields, they come with inherent challenges that need to be addressed. Threshold-based segmentation, for instance, is very sensitive to changes in lighting and noise, which can reduce the accuracy of segmentation results. Additionally, manually selecting the threshold can be challenging, as it is difficult to find the right thresholds for different images and situations. Edge-based segmentation techniques are also not resistant to noise, leading to inconsistent results. Furthermore, edge segmentation struggles with complex image structures due to its sensitivity to image details and textures. Region-growing algorithms find it hard to accurately segment objects that vary greatly in size or have irregular shapes. Deep learning approaches to segmentation require large labeled datasets for training and significant computational resources, especially for complex network architectures. Moreover, understanding the predictions of these models remains a challenge, as it is hard to translate these predictions into clear rules or principles.

The Min-Cut based graph cut segmentation method provides an efficient and accurate solution to the image segmentation problem by transforming it into a graph partitioning problem. This method leverages global optimization techniques to find the minimum cut, thus avoiding the need for manual threshold setting. The Min-Cut based graph cut segmentation method effectively integrates both region and boundary information, which helps handle variations in lighting and mitigate noise interference within images to some extent. Furthermore, by introducing the energy function, the graph cut method can integrate various a priori knowledge and image characteristics. It also allows users to introduce constraints flexibly according to specific application scenarios, enhancing segmentation performance in complex environments. Compared with deep learning methods that rely on large amounts of labeled data and computational resources, one of the primary benefits of graph cut is its minimal resource demands. This advantage reduces the reliance on extensive labeled datasets and ensures efficient operation even with limited

computational resources.

Compared to traditional graph cut methods, the combination of s-t graph cuts and minimum cuts provides more precise and efficient segmentation solutions.By combining s-t graph cut techniques and defining optimal Min-Cut models through adjusting various parameters and constructing different graph structures, more accurate image segmentation can be achieved while improving computational efficiency.

Although efficient minimum cut algorithms, such as the Ford-Fulkerson and Push-Relabel algorithms [21][22], are available for specific types of graphs, the computational complexity of finding the minimum cut remains high for complex graph structures. The introduction of the Gurobi optimizer significantly enhances the ability to achieve the global optimal solution and greatly improves the efficiency of finding the global minimum of the target energy function.

1.2 Thesis Overview

This thesis presents an in-depth exploration of an approach for object/background segmentation of images using combinatorial optimization of s/t graph cuts. This approach focuses on the application of the minimum cut technique. It constructs several different minimum cut image segmentation models through diverse graph construction and a series of parameter adjustments during in-depth study and image and graph construction analysis. This thesis comprehensively compares and analyzes the differing effects these minimum cut models have on image segmentation. We then detail two mathematical models that define the Min-Cut problem. A contribution of this thesis is utilizing the Gurobi optimization solver, which significantly enhances the achievement of the global optimal solution for the constructed Min-Cut models and greatly improves the efficiency of finding the global minimum of the target energy function. The thesis is organized as follows:

- In Chapter 2, a comprehensive literature review on image segmentation, discussing various segmentation methods including graph-based methods, pixel-clustering, deep learning-based segmentation, and other specialized approaches. This chapter serves as a foundation for understanding the current state of the field and the motivation of the proposed method.

- Chapter 3 provides an overview of the preliminary knowledge and theoretical foundations necessary for image segmentation methods. In the first part of this chapter, concepts related to color spaces, the definition of color spaces, and their significance in processing image data are elaborated, laying the groundwork for understanding how image data can be described and processed through different color models. The second part of this chapter delves into the basic

theories of graphs and the application of graph theory in image segmentation problems, thoroughly explaining the fundamentals of graph cuts theory and its application relationship in image segmentation issues. Additionally, this chapter discusses the basics of Min-Cut problems and introduces common algorithms applied to standard Min-Cut problems in image segmentation. Two mathematical models that define the minimum s/t cut problem from different perspectives are elaborated.

- In Chapter 4, the focus shifts to the key aspects of the thesis, detailing the process of constructing a graph for image segmentation. This includes establishing region terms and boundary terms, selecting edge weights, and constructing the energy function. Additionally, methods for selecting graph seeds, including both manual and automatic selection, are discussed.

- Chapter 5 starts with a comprehensive introduction, analysis of the dataset characteristics, and preparation steps before the experiments. It then provides a brief overview of the basic principles and usage of the Gurobi optimizer. Following this, five sets of experiments are conducted to progressively test various segmentation models based on different parameters. The chapter concludes by detailing the metrics and standards used to evaluate the accuracy and efficiency of the segmentation results, along with an analysis and evaluation of the experimental outcomes.

- Chapter 6 Summarizes the research findings and experiments on image segmentation, discussing the effectiveness of the Min-Cut algorithm in this context.

Chapter 2

State of the Art

Image segmentation is a classical problem in computer vision research and has become a hot spot in the field of image understanding. It is the first step of image analysis, the foundation of computer vision, and one of the most challenging problems in image processing. Since the 1970s, image segmentation has attracted significant research efforts. Although there is no universal perfect image segmentation method, the general principles have reached a consensus, resulting in many research outcomes and methods. Image segmentation techniques can be roughly categorized into graph-based methods, pixel clustering-based methods, and deep learning-based methods based on their algorithmic evolution.

In this chapter, we analyze the current state of the art in various image segmentation methods. The first part introduces graph-based image segmentation techniques, particularly emphasizing the Felzenszwalb method. Next, we explore image segmentation methods based on pixel clustering, including a detailed description of the k-means clustering method and an analysis of how its parameters affect the results. In the third section, we present methods and state-of-the-art techniques for deep learning-based segmentation. Finally, this chapter provides a general overview and review of current research on other image segmentation methods.

2.1 Graph-Based Image Segmentation

Graph-based methods map an image as a weighted undirected graph, considering pixels as nodes. They convert the image segmentation problem into a graph vertex partitioning problem, obtaining optimal segmentation by finding the maximum flow/Min-Cut of the graph.

Graph-based image segmentation offers notable advantages:

- **Established Mathematical Foundation**: Graphics is a well-developed research field with a solid mathematical foundation, providing multiple solutions to image segmentation problems.
- **Similarity to Images**: The similarity between images and graphs allows the conversion of images into graphical structures, enabling the application of numerous theoretical and mathematical tools for segmentation purposes.

Despite their computational complexity, a key advantage of these methods is that they do not require training, which makes them applicable to various segmentation tasks without preprocessing or labeled datasets.

Graph-based image segmentation techniques typically represent the problem in terms of a graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a pixel in the image, E is an edge connecting a pair of neighboring pixels, and the weights of the edges, $w(v_i, v_j)$, represent the non-negative similarity between neighboring pixels in terms of grayscale, color, or texture. The essence of segmentation methods based on graphs is to remove specific edges and divide the graph into several subgraphs to achieve segmentation. The optimality principle of segmentation is to maximize the similarity within the subgraphs and minimize the similarity between the subgraphs. Well-known graph-based methods include Normalized Cut [13], Graph Cut [23], GrabCut [24], and Felzenszwalb methods [25].

Graph cut methods can be fully automated. Combining graph cut with other techniques, such as Phase Unwrapping via Graph Cuts proposed by Bioucas-Dias [26], which uses phase expansion to obtain the seeds and then applies graph cut for automatic segmentation; Suga et al. [27], which uses the SIFT operator to get the seeds and then performs automatic segmentation; Tang et al. [28] used a saliency detection algorithm to automatically localize the object roughly and then optimized the segmentation with weighted kernel density estimation and graph cut algorithm. However, the complexity of segmented images and the presence or absence of apparent features of interest critically affect the practicality of automatic segmentation. Therefore, interactive segmentation methods based on graph cuts remain practical.

2.1.1 Felzenszwalb Method

Felzenszwalb's algorithm employs a graph-based approach to image segmentation by converting the image into an undirected weighted graph, where each pixel is a node and the edge weights represent the differences between adjacent pixels. The algorithm orders the edges based on their weights and iteratively merges pixel components with differences below a certain threshold, following the order of the weights until a specified stopping condition is reached. A minimum spanning tree algorithm is applied to maintain these merges' connectivity.

In Felzenszwalb's algorithm, k is a key factor that controls the granularity of image segmentation. Specifically, k determines the Minimum Internal Difference (MID) between the components that make up the image. Two components are considered separate only if their difference is greater than their MID. Adjusting k allows for manipulating the segmentation's fineness:

- If k is large, the algorithm tends to generate larger components, making the resulting image regions coarser.
- If k is small, the algorithm generates smaller components, resulting in a more detailed segmentation.

The Felzenszwalb algorithm is valued for its efficiency and capability of processing diverse images, consistently delivering uniform and well-connected segmentation outcomes. However, it often fails to precisely delineate object boundaries, yielding overly smoothed segmentation contours. Fine-tuning the k factor is necessary to achieve the desired level of boundary precision. Figures 2.1 and 2.2 illustrate the application of Felzenszwalb's algorithm with different k.

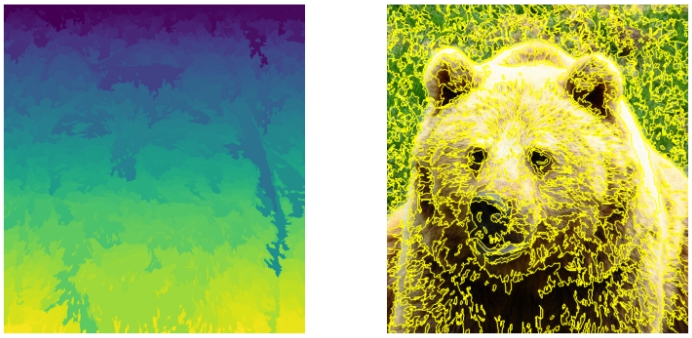


Figure 2.1: Results produced by Felzenszwalb algorithm (k = 100).

Figure 2.2: Results produced by Felzenszwalb algorithm (k = 500).

As seen in the results, smaller k lead to finer segmentation, while larger k result in coarser segmentation. In Figure 2.2, the bear eyes are segmented correctly, while in Figure 2.1, they are not.

2.2 Pixel-Clustering Segmentation

Clustering is a statistical method used for classification problems that aims to divide a collection of objects into several groups based on similarities among data points. As an unsupervised learning approach, clustering algorithms operate without prior knowledge, grouping data based on similarities among data points.

In image segmentation, clustering algorithms treat image pixels as a dataset to be classified, segmenting the image by analyzing pixel features such as grayscale, color, position, and texture. This approach enables segmentation into distinct, meaningful regions by identifying and leveraging inherent pixel similarities without relying on predetermined classification standards.

Traditional clustering methods such as FCM (Fuzzy C-Means) [29], k-means algorithms [14], spectral clustering [30], the Mean Shift algorithm [16], and SLIC (Simple Linear Iterative Clustering) [17] are prominent examples. The Mean Shift algorithm, introduced by Comaniciu and Meer, is a nonparametric clustering algorithm widely used in tasks like image segmentation and tracking. It does not require a predetermined number of clusters, handling various visual data well. In contrast, SLIC focuses on creating compact, uniform hyperpixels through iterative clustering based on color and spatial proximity, simplifying the segmentation process [17]. R. Urquhart introduced graph-theoretic clustering based on finite neighborhood sets in 1982, constructing a graph where data points are nodes connected by edges representing similarities. This improves computational efficiency, especially on large datasets, by focusing on local similarities [31].

Each clustering method has distinct characteristics suited to specific applications, providing a robust toolkit for image processing.

2.2.1 K-Means Method

Since McQueen proposed the k-means algorithm in 1967, it has been widely used for clustering. K-means is an unsupervised learning method that evaluates data point similarity by minimizing the sum of squared errors and automatically dividing the dataset into clusters.

Initially, the algorithm selects k objects randomly from the dataset to serve as the initial cluster centers. It assigns each data point to the nearest cluster center based on the Euclidean distance, computes the mean position of data points in each cluster, and updates the cluster centers. This iterative process continues until

the cluster centers stabilize. Figure 2.3 demonstrates the application of k-means to image segmentation.

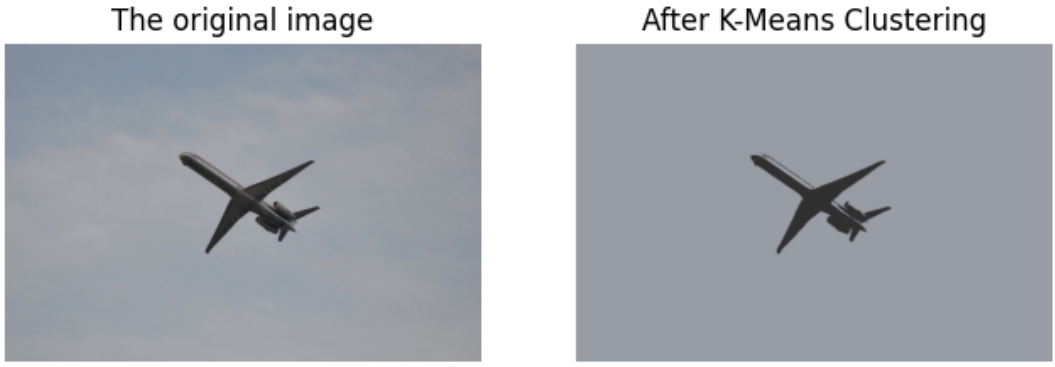


Figure 2.3: k-means clustering algorithm for image segmentation with k=2

The k-means algorithm is known for its speed, simplicity, and scalability, which makes it effective and adaptable for large datasets. Its linear time complexity facilitates efficient handling of large data volumes.

However, k-means has limitations. It is sensitive to outliers, which can distort cluster centroids and result in inaccurate outcomes. It heavily depends on the initial placement of cluster centroids, and poor initialization can trap the algorithm in a suboptimal solution. As a distance-based method, k-means is suitable only for convex datasets, not for non-convex data [32].

Despite these challenges, the simplicity and efficiency of k-means often make it a starting point for data clustering tasks, with potential for exploring more complex models like Gaussian Mixture Models (GMM) for datasets with greater heterogeneity [33].

2.3 Deep Learning-based Segmentation

Traditional image segmentation techniques, such as threshold segmentation, edge detection, and region growth, depend on manual parameter tuning. Using only lower-level content information, such as color, brightness, and texture, these methods often struggle with complex structures objects and high internal variability, leading to inaccurate segmentations in intricate scenarios.

Deep learning has revolutionized image processing tasks, including segmentation, by providing superior efficiency, accuracy, and the capacity to handle complex environments. Unlike traditional approaches requiring handcrafted feature design, deep

learning algorithms excel in autonomously deriving potent feature representations directly from the data. These models are adaptable, enabling customization for various segmentation tasks, such as instance and semantic segmentation, through model fine-tuning or additional network layers. This adaptability and learning capacity make deep learning invaluable for tackling image segmentation challenges.

In 2013, Farabet et al. trained a deep convolutional classification network using a supervised approach [18]. This technique involved multi-scale sampling around a target pixel and processing the resulting local image patches through a CNN classifier, iteratively assigning a semantic label to every pixel. In 2014, Long et al. introduced the Fully Convolutional Network (FCN) [19], transforming fully connected layers into convolutional layers for end-to-end learning and segmentation of images. FCN handles input images of any size, producing corresponding segmentation maps, and laying the groundwork for future algorithms. In 2015, Olaf Ronneberger and his team introduced U-Net, specifically addressing medical image segmentation with a distinctive U-shaped architecture that captures contextual details while maintaining edge precision [34]. SegNet, another critical contribution, focuses on scene understanding through a deep encoder-decoder architecture, effective for segmenting complex outdoor scenes [35].

Recently, Shervin Minaee and Yuri Boykov et al. published a comprehensive review paper titled "Image Segmentation Using Deep Learning: An Overview." This paper examines advances in the application of deep learning to image segmentation and constructs a development timeline for deep learning-based 2D image segmentation algorithms (Figure 2.4). Important advancements in semantic segmentation are represented by orange blocks, while key milestones in instance segmentation are indicated by green blocks.

Deep learning-based image segmentation methods bring breakthroughs, but they also face challenges. They require large labeled datasets and computational resources for training. These models often suffer from poor interpretability and risk overfitting, especially with limited or non-diverse data, reducing generalization ability. Ongoing research is gradually addressing these issues, showing a positive trend toward solving these challenges.

2.4 Other Specialized Approaches

In addition to the previously mentioned methods, several other techniques have demonstrated notable results in various application scenarios.

2.4.1 Threshold-based methods

Threshold-based methods are among the first image segmentation techniques to segment an image into distinct regions by selecting one or more threshold

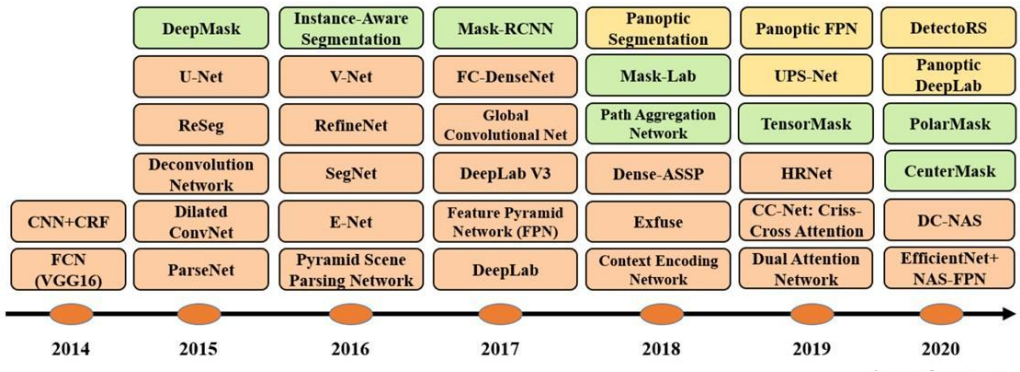


Figure 2.4: Development timeline for 2D image segmentation algorithms based on deep learning. [36]

values. Using at least one color or grayscale value to define image boundaries, they generate binary images, reducing complexity and simplifying recognition and classification processes [37]. Standard methods include global thresholding [9], local thresholding [38], and adaptive thresholding [39]. However, threshold-based methods are susceptible to lighting changes and noise, making it challenging to select a suitable threshold for complex scenes, resulting in unsatisfactory segmentation.

2.4.2 Edge detection-based methods

Edge detection-based methods utilize significant changes in an image's gray values to identify pixels that define region boundaries. Common techniques include gradient and grayscale histogram methods. The gradient method detects edges by identifying abrupt gray value changes between regions, while the grayscale histogram method separates the foreground from the background by selecting a specific threshold [40]. Notable edge detection operators include Roberts, Sobel, Prewitt, Canny, and Laplace [41]. The Sobel operator detects bidirectional edges (horizontal and vertical) [42].

Although edge detection methods effectively extract boundaries, they are prone to generating pseudo edges and struggle to enclose the complete target region, especially in textured and noisy images.

2.4.3 Model-based methods

Model-based segmentation methods guide the process using predefined templates or models of specific shapes (e.g., circles, ellipses) [43]. These methods accurately segment specific target regions by matching models with image content, suitable for scenes with regular patterns, such as medical image processing, industrial

inspection, and target recognition [43, 44, 45, 46, 47]. However, they are less effective for targets with complex or irregular shapes, and model construction and matching involve high computational complexity.

These segmentation techniques exhibit unique advantages and effectiveness in different applications, demonstrating their practicality and flexibility. However, each method has limitations. With increasing demand and complexity, existing technologies may struggle to achieve ideal results in certain situations. Therefore, research for new methods to optimize image segmentation remains crucial.

Chapter 3

Preliminary Know-How

3.1 Color Spaces and Similarity Measures

In image recognition and segmentation, choosing suitable feature is paramount for accurate object detection. These features include various dimensions such as shape, texture, grayscale and color. Color, as a key visual attribute, has found extensive application and recognition [48].

Color images provide much more information than grayscale images. They capture the properties of object surfaces across different wavelengths of light, offering a more precise representation of surface characteristics. Furthermore, color features tend to be more stable than shape or texture features when images undergo rotation and scaling. These benefits make color features particularly advantageous in image segmentation tasks, improving both accuracy and dependability. Consequently, choosing color as a segmentation feature is an effective strategy in image processing.

3.1.1 Color Space

In the practice of digital imaging, computer graphics, and display technology, the accurate identification of color is indispensable. To achieve this, various color models are utilized. A color space serves as a framework for arranging and representing colors, mathematically facilitating their description and reproduction [49]. In image processing, different color spaces can be adapted to diverse applications, and choosing an appropriate color space not only enhances the expression of color information but also optimizes the performance of image segmentation based on the requirements of individual applications.

Typically, color spaces can be categorized into two primary types: hardware-oriented and user-oriented. Hardware-oriented color spaces include RGB (Red, Green, Blue), CMY (Cyan, Magenta, Yellow), and YIQ [50]. In RGB color space, colors are defined by the intensity of three components: red (R), green (G), and

blue (B). Each component usually ranges from 0 to 255, where 0 indicates the minimum intensity (black) and 255 signifies the maximum intensity (white) [51]. Different combinations of these values produce a wide range of colors.

The RGB color space is widely used in computer graphics, digital image processing, display technology, and other fields due to its prevalence on computer monitors and television screens. It serves as the primary method for representing colors in these devices. In addition, RGB is extensively used in image processing software for tasks such as editing, compositing, and image manipulation. This widespread use underscores its significance in various applications related to visual technologies.

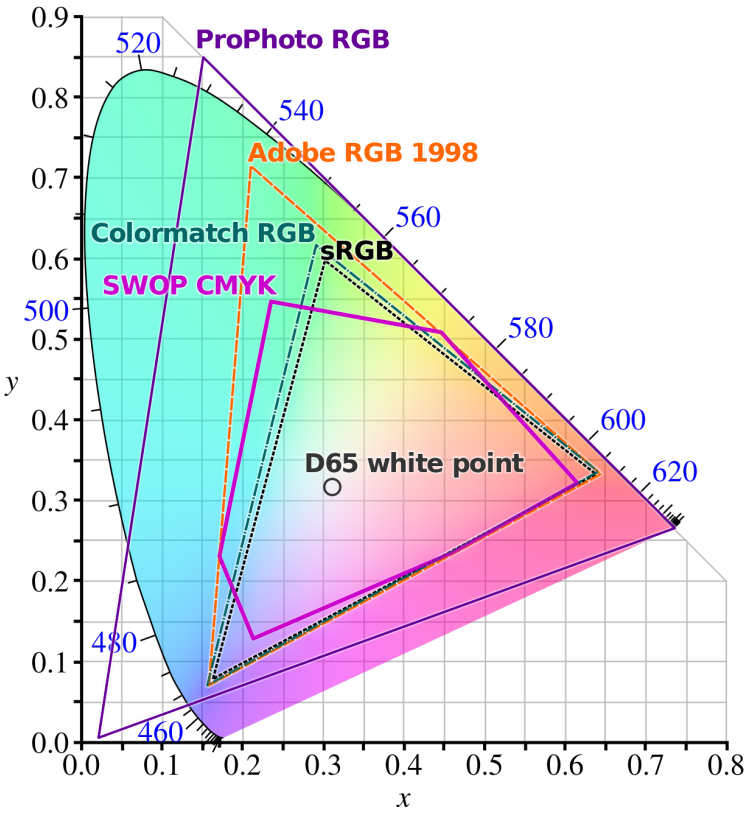


Figure 3.1: 1931 CIE chromaticity diagram showing some RGB color spaces as defined by their chromaticity triangles [52]

User-oriented color spaces, such as HLS (Hue, Lightness, Saturation), HCV, HSV (Hue, Saturation, Value), HSB, MTM, CIE-LAB, and CIELUV, are designed to simulate human visual perception. Among these, the CIE-LAB color space is particularly prevalent [50]. Unlike the RGB color space, CIE-LAB is constructed to align with the human eye's perception of colors, making it more consistent with human vision characteristics.

In the CIE-LAB color space, colors are denoted by three coordinates: L (luminance), a (green-red component), and b (blue-yellow component). The L coordinate represents the lightness of a color, ranging from 0 (black) to 100 (white). The coordinates a and b represent the position of the color in the red-green and blue-yellow directions, respectively, with values ranging from -128 to +127 [53]. These three coordinates enable the CIE-LAB color space to accurately describe nearly all visible colors, ensuring it effectively mirrors human color perception.

One of the main advantages of the CIE-LAB color space is its homogeneity. This means that colors close to each other in terms of distance within this space tend to appear visually similar to the human eye. This property makes the CIE-LAB color space highly useful for quantifying color differences and performing color comparisons. In image segmentation tasks, researchers leverage this attribute to measure the variance in color across distinct areas or objects within an image, thereby achieving enhanced precision in segmentation outcomes.

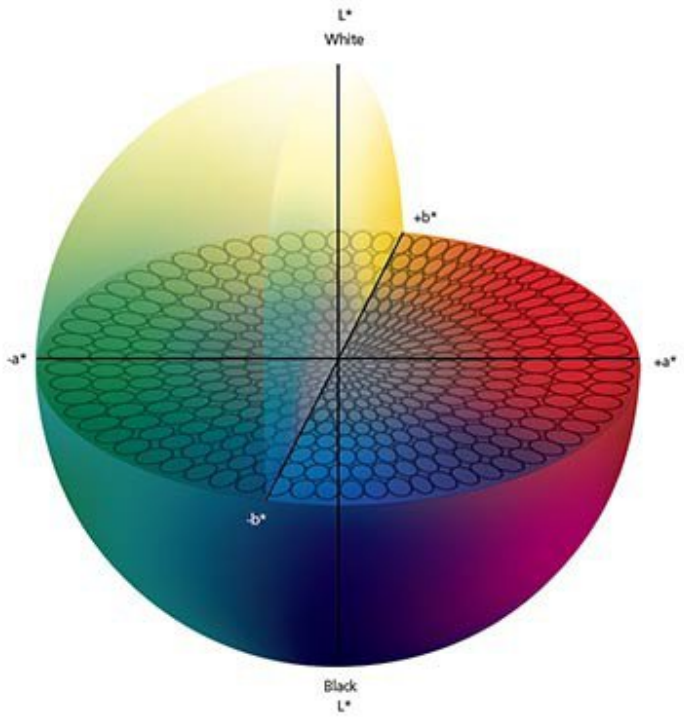


Figure 3.2: CIELAB is a 3D space, the space is perceptually uniform [54]

3.1.2 Similarity Measures

In this thesis, we use the color distance based on two different color spaces: RGB color space and LAB color space to quantify the color similarity of pixels. By examining the variances and similarities between colors in these two spaces, a more comprehensive understanding of color's impact on the segmentation of color images can be achieved.

1. **Color distance in the RGB space**

   RGB is an additive color model where colors are represented by combining different intensities of 3 channels.

   To determine the color distance between two pixels in RGB space, the Euclidean distance formula as follows:

$$\Delta RGB_{ij} = \sqrt{(R_j - R_i)^2 + (G_j - G_i)^2 + (B_j - B_i)^2} \tag{3.1}$$

   - R_i, G_i, B_i: Red, green, and blue channel values of pixel i.
   - R_j, G_j, B_j: Red, green, and blue channel values of pixel j.
   - ΔRGB_{ij}: The color distance between pixel i and pixel j in RGB space.

2. **Color distance in the LAB space**

   LAB color space is designed to mimic human perception of color and is often used in applications that require accurate color representation. To calculate the color distance between two points in the LAB space, the Euclidean distance formula as follows:

$$\Delta E^*_{ab_{ij}} = \sqrt{(L^*_j - L^*_i)^2 + (a^*_j - a^*_i)^2 + (b^*_j - b^*_i)^2} \tag{3.2}$$

   - L^*_i, a^*_i, b^*_i: The lightness (L*), red/green axis (a*), and yellow/blue axis (b*) values of pixel i in LAB space.
   - L^*_j, a^*_j, b^*_j: The lightness (L*), red/green axis (a*), and yellow/blue axis (b*) values of pixel j in LAB space.
   - $\Delta E^*_{ab_{ij}}$: The color distance between pixel i and pixel j in LAB space (according to the CIE76 standard).

3.2 Theory of Graph and Graph Cuts

3.2.1 Graph theory

Graph theory is a branch of mathematics focused on the study of mathematical structures known as graphs. These structures consist of points, called vertices or nodes, and lines connect the vertices, called edges. Graph theory serves as a powerful tool for modeling pairwise relationships among objects, allowing for the exploration and analysis of concepts such as connectivity and network structure. In a graph, vertices represent individual entities or objects, while edges symbolize links or interactions between these entities. This framework is instrumental in understanding and managing various abstract and practical problems in numerous fields, including computer science, biology, and social sciences.

A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E that connect these vertices. The vertices, denoted by symbols such as u and v, represent individual entities. The edges, denoted by symbols such as w, represent the connections between these entities. Each edge in E corresponds to a pair of vertices in V.

In an undirected graph shown in Figure 3.3, all edges have no specific direction, so the pair (u, v) is the same as (v, u). The edges in a graph can also be ordered, with $\langle u, v\rangle$ indicating an edge directed from u to v. In this case, $\langle u, v\rangle$ and $\langle v, u\rangle$ represent two different edges. If the edges of the graph are directional, such a graph is called a directed graph, as shown in Figure 3.4.

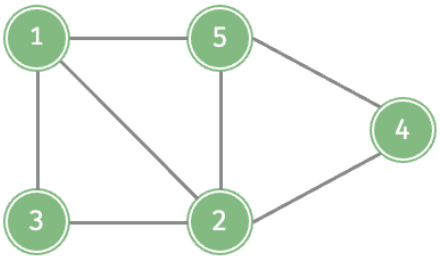


Figure 3.3: Illustration of Undirected Graph

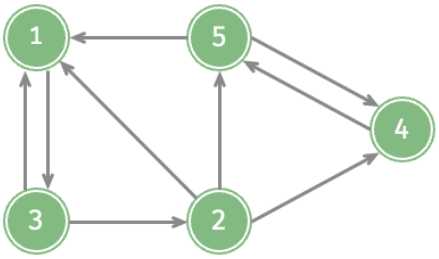


Figure 3.4: Illustration of Directed Graph

3.2.2 Graph Cut

In graph theory, connectivity refers to the property that nodes or sub-graphs within a graph can be connected through edges. In an undirected graph, if a path composed of one or more edges allows two nodes or sub-graphs to reach each other,

these nodes or sub-graphs are considered connected. If any pair of vertices in an undirected graph G are connected, the graph is called a connected graph.

For a directed graph, G is called a strongly connected directed graph if, for any two vertices u and v in G, there exists both a path from u to v and a path from v to u. If G is a directed graph and, for any two vertices u and v, there exists either a path from u to v or a path from v to u, then G is called a unilaterally connected directed graph.

The opposite of connectivity is separation (or cut). Suppose E' is a subset of the edge set of a connected graph G. If removing E' from G results in a disconnected graph, then E' is called a cut set of G. When edges are cut, the nodes or regions associated with these edges are divided into disconnected sets. An example is shown in Figure 3.5.

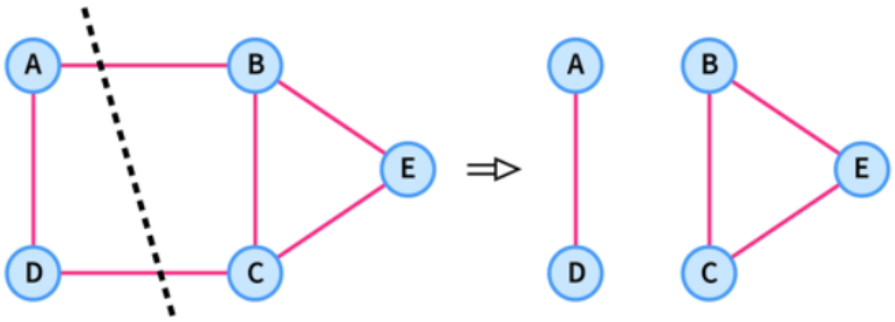


Figure 3.5: Removing the two edges $A \leftrightarrow B$ and $D \leftrightarrow C$ partitions the graph into two disconnected components, and the set containing the edges $A \leftrightarrow B$ and $D \leftrightarrow C$ forms a cut.

The definition of *cut* is as follows:

A graph $G = (V, E)$ can be partitioned into two disjoint sets, S, T, $S \cup T = V$, $S \cap T = \emptyset$, simply by removing the edges connecting the two parts. The degree of dissimilarity between these two pieces can be computed as total weight of the edges that have been removed. In graph theoretic, it is called the *cut* [55].The formula 3.3 represents the cut.

$$\text{cut}(S, T) = \sum_{u \in S, v \in T} w(u, v) \tag{3.3}$$

3.3 The Min-Cut Problem

A cut in a graph is defined as a collection of edges whose removal disconnects a graph into two distinct sub-graphs. Among all possible cuts, a *minimum cut* or a *Min-Cut*is distinguished by having the smallest number of edges or the lowest total edge weights in a weighted graph. The task of finding such minimum cuts is known as *minimum cut problem* or *Min-Cut problem*. The minimum-cut problem is a quintessential combinatorial optimization problem in the field of graph theory.

The minimum-cut problem in graph theory has two primary formulations: with and without terminal nodes. The Min-Cut problem with terminal nodes is known as the *minimum s-t cut problem*. For a given graph, $G = (V, E)$ with a particular source node s and sink node t, find the smallest set of edges such that all paths from the source node s to the sink node t must pass through this set while also minimizing the number of edges traversed (or the sum of their weights). In contexts where a source and a sink are defined, this minimum cut problem is closely related to the maximum flow problem. The duality between the max-flow and Min-Cut allows the determination of the minimum cut by maximizing the flow from the source s to the sink t.

The Min-Cut problem takes a more global perspective when no specific terminal nodes are designated. This problem does not focus on particular nodes for disconnection but seeks a cut that globally minimizes connectivity across the graph.

Although the minimum cut problem is not theoretically computationally difficult, its wide applications and elegant mathematical form make it a classic topic in graph theory research. This section will use two different mathematical models to define the minimum s-t cut problem. In Chapter 5, the performance of these two models in actual image segmentation will be compared through experiments.

1. **Model 1**

   The objective of the first model is to minimize the sum of the weights of the cut edges based on the Max-Flow Min-Cut Theorem. Constraints ensure that the total flow through the cut edges is equal to the maximum flow.

   Given $G := (V, A)$ a flow network, source node s and termination node t, and capacity u_{ij} in each arc (i, j):

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} u_{ij} y_{ij} & \\
\text{s.t.} \quad & z_i - z_j \leq y_{ij} & \forall (i,j) \in E \\
& z_s - z_t = 1 & (z_s = 1, z_t = 0) \\
& y_{ij} \in \{0,1\} & \forall\ (i,j) \in E \\
& z_i \in \{0,1\} & \forall\ i \in V
\end{aligned}
$$

- y_{ij}: binary variables taking value 1 if arc (i,j) belongs to the optimal cut (forward), 0 otherwise
- z_i: binary variables taking value 1 if node i belongs to the subset of nodes containing s, 0 otherwise

Its main features include:

- Direct binary variable definitions for nodes and arcs.
- Emphasizing flow conservation and non-negative constraints on flow.

2. **Model 2**

   The second model aims to minimize the sum of the weights of the cut while also introducing additional variables and constraints. In addition to z_i and y_{ij}, it introduces two binary variables: V_i^S and V_i^T, which signify whether a node i is a source or a terminating node.

   Given $G := (V, A)$ a flow network, source node s and termination node t, and capacity u_{ij} in each arc (i, j):

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} u_{ij} y_{ij} & \\
\text{s.t.} \quad & z_i - z_j \leq y_{ij} & \forall (i,j) \in A \\
& z_i \geq V_i^S & \forall i \in V \\
& 1 - V_i^T \geq z_i & \forall i \in V \\
& \sum_{i\in V} V_i^S = 1 & \\
& \sum_{i\in V} V_i^T = 1 & \\
& V_i^S + V_i^T \leq 1 & \forall\ i \in V \\
& y_{ij} \in \{0,1\} & \forall\ (i,j) \in A \\
& z_i \in \{0,1\} & \forall\ i \in V \\
& V_i^S \in \{0,1\} & \forall\ i \in V \\
& V_i^T \in \{0,1\} & \forall\ i \in V
\end{aligned}
$$

   - y_{ij}: binary variables taking value 1 if arc (i,j) belongs to the optimal cut (forward), 0 otherwise
   - z_i: binary variables taking value 1 if node i belongs to the subset of nodes containing s, 0 otherwise
   - V_i^S: binary variables taking value 1 if node i is the source s, 0 otherwise

- V_i^T: binary variables taking value 1 if node i is the sink t, 0 otherwise

The second model becomes more specific by introducing additional variables and complex constraints, enabling a more straightforward definition and solution to the minimum cut problem from the source node to the sink node.

3.3.1 Algorithms for Min-Cut

The minimum cut is not only an abstract concept in theoretical research, but also a practical tool in image segmentation. By redefining the image segmentation challenge as a minimum cut problem, the complex task is simplified to identify the optimal graph cut. This approach utilizes a variety of efficient algorithms and strategies from graph theory, providing a comprehensive solution. The similarities between image segmentation and finding the minimum cut in a graph allow for the development of effective and reliable solutions using insights from graph theory.

Various algorithms can solve the minimum cut problem, each offering unique advantages in different application scenarios. One of the most commonly used algorithms is based on the maximum flow-minimum cut theorem for capacitated networks. The minimum cut problem is closely related to the maximum flow problem. According to Theorem 2, the optimal solution to the maximum flow problem equals the optimal solution to the corresponding minimum cut problem. Therefore, solving the minimum cut problem can be transformed into solving the maximum flow problem [56]

Theorem 1 (Augmenting Path Theorem) *A flow is a maximum flow if and only if the residual network G contains no augmenting path.*

Theorem 2 (Max-Flow Min-Cut Theorem) *The maximum flow value from s to t equals the minimum capacity of all s-t cuts.*

The maximum flow problem involves finding the maximum possible flow from a specified source node to a specified sink node in a directed graph called a capacitated network. The objective of the maximum flow problem is to maximize the total flow from the source to the sink, which is a central issue in network flow theory.

According to Theorem 1, an essential concept in solving the maximum flow problem is the augmenting path. An augmenting path is a direct path from a source node s to a sink node t in the residual network. The residual capacity of an augmenting path is the minimum residual capacity of any arc in the path. [57]

1. **Ford-Fulkerson Algorithm**

   The Ford-Fulkerson algorithm is a maximum flow algorithm based on augmenting paths. It iteratively finds augmenting paths and updates the flow

to increase the total flow in the network gradually [21]. The steps of the Ford-Fulkerson algorithm are as follows:

(a) **Initialization**: Initialize all flows in the network to zero.

(b) **Finding Augmenting Paths**: Use depth-first search (DFS) or breadth-first search (BFS) in the residual network to find an augmenting path from the source to the sink.

(c) **Updating Flow**: Update the flow in the network based on the minimum residual capacity along the augmenting path.

(d) **Repeat**: Repeat steps 2 and 3 until no augmenting path can be found in the residual network.

The algorithm 1 provides the pseudocode of the Ford-Fulkerson algorithm.

Algorithm 1 Ford-Fulkerson Algorithm

function FORDFULKERSON(G, s, t)
  for all edge (u, v) in G **do**
    set $flow(u, v) = 0$
  end for
  while there exists an augmenting path p from s to t in the residual graph G_f **do**
    find the residual capacity $c_f(p)$
    for all edge (u, v) in p **do**
      $flow(u, v) = flow(u, v) + c_f(p)$
      $flow(v, u) = flow(v, u) - c_f(p)$
    end for
  end while
  return the total flow from s to t
end function

2. **Push-Relabel Algorithm**

The Push-Relabel algorithm, also known as the Preflow-Push algorithm, differs from augmenting path algorithms. Instead of directly finding augmenting paths, the Push-Relabel algorithm increases flow through push and relabel operations until the maximum flow is achieved. The core idea is to maintain a "preflow," allowing some nodes to have an inflow greater than their outflow, and gradually adjusting the flow through push and relabel operations until it becomes a valid maximum flow.

The steps of the Push-Relabel algorithm are as follows.

(a) **Initialization**: Initialize all edge flows to 0. Set the height of the source node to the total number of nodes n, and set the height of all other nodes to 0. Set the flow of all edges from the source to their capacity and push these flows to the corresponding nodes. The initial preflow equals the sum of the capacities of all outgoing edges from the source.

(b) **Push Operation**: For each node with excess flow, if there is an adjacent edge (u, v) such that $h(u) = h(v) + 1$ and the residual capacity is greater than 0, push as much flow as possible to v. Specifically, $\Delta f = \min(\text{excess}(u), c_f(u, v))$. Update the flow and excess flow accordingly.

(c) **Relabel Operation**: If a node with excess flow cannot push flow (i.e., there are no suitable adjacent edges), increase the node's height to enable the push operation. Specifically, set the height of node u to $h(u) = \min\{h(v) \mid c_f(u, v) > 0\} + 1$.

(d) **Repeat**: Continuously perform push and relabel operations until no nodes, except the source and the sink, have excess flow.

By iteratively performing these operations, the Push-Relabel algorithm adjusts the flow in the network until it reaches the maximum flow. The algorithm 2 provides the pseudocode of the Push-Relabel algorithm.

In addition to the Ford-Fulkerson and Push-Relabel algorithms, Dinic's algorithm (or Dinitz's algorithm) accelerates max-flow computation by constructing a hierarchical network and using layered paths to achieve a minimum cut. Edmonds-Karp's algorithm improves efficiency by using Breadth-First Search (BFS) to find augmenting paths. The Stoer-Wagner algorithm specializes in undirected graphs and finds the minimum cut by gradually shrinking the graph until only two nodes remain. Karger's algorithm is a randomized method that forms the cut set by randomly selecting edges and merging nodes. Improves correctness by running multiple iterations. [58]

These algorithms collectively enhance the solutions to the minimum cut problem, particularly in image segmentation. They provide diverse approaches for different scenarios, improving the accuracy and efficiency of segmentation and making image processing more precise and effective.

Algorithm 2 Push-Relabel Algorithm

```
    function PUSH-RELABEL(G, s, t)
 2:     initialize height and excess flow
        for all edge (s, v) in G do
 4:         flow(s, v) = capacity(s, v)
            flow(v, s) = −flow(s, v)
 6:         excess(v) = flow(s, v)
            excess(s)− = flow(s, v)
 8:     end for
        while there exists an active node u (u ≠ s and u ≠ t) do
10:         if there exists a neighbor v of u such that capacity(u, v)−flow(u, v) > 0
    and height(u) = height(v) + 1 then
                PUSH(u, v)
12:         else
                RELABEL(u)
14:         end if
        end while
16: end function
    function PUSH(u, v)
18:     send = min(excess(u), capacity(u, v) − flow(u, v))
        flow(u, v)+ = send
20:     flow(v, u)− = send
        excess(u)− = send
22:     excess(v)+ = send
    end function
24: function RELABEL(u)
        min_height = ∞
26:     for all neighbor v of u do
            if capacity(u, v) − flow(u, v) > 0 then
28:             min_height = min(min_height, height(v))
            end if
30:     end for
        height(u) = min_height + 1
32: end function
```

Chapter 4

Methodology and Implementation

This chapter explores methods for converting images into graph structures and efficiently modeling regions and boundary information within images using a graph cut segmentation framework. By defining and minimizing a cost function that integrates hard and soft constraints, efficient and accurate image segmentation can be achieved. This approach essentially transforms the challenge of image segmentation into the problem of finding the minimum cut in a graph, thereby enabling efficient and precise segmentation results.

4.1 The Image Seen as a Graph

An image can be considered as a graph $G = (V, E)$, where each pixel (in a 2D image) or voxel (in a 3D image) corresponds to a node in the graph. Neighboring pixels (nodes) are usually connected by edges in a regular grid-like structure, as illustrated in Figure 4.1 and Figure 4.2.

Specifically, an image can be defined as a set of pixels $V = \{v_{ij}\}$, where (i, j) denotes the position of pixel v_{ij} in the image domain $\Omega \subseteq \mathbb{R}^2$. Edges connected between neighboring pixels (i, j) are referred to as n-links and the set of these connected edges is denoted as N. Note that the definition of neighboring pixels can be arbitrary, and the neighborhood system N can contain all unordered pairs of neighboring pixels under the standard 8-neighborhood system (for 2D images) or 26-neighborhood system (for 3D images). In this context, we focus solely on the segmentation of 2D images.

To simplify the modeling and express the relationship between pixels and "object and background" more intuitively and accurately, a typical method to extend the regular graph is to introduce two additional special nodes called terminals. In the

Figure 4.1: Original Image

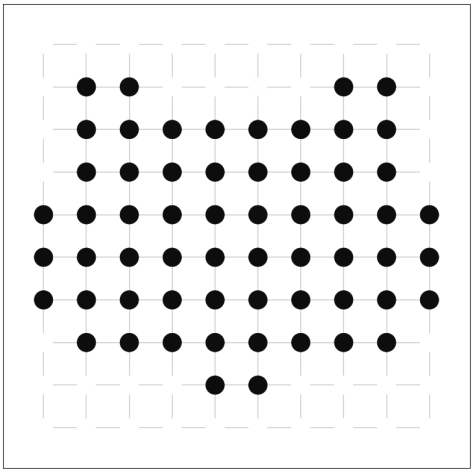


Figure 4.2: Illustration of grid-like arrangement of pixels

context of vision, the terminals correspond to the set of labels that can be assigned to pixels, where the source is associated with the foreground and the sink with the background and are denoted by the symbols s and t, respectively, which can be considered virtual nodes. The edges formed by all other vertices connected to these two terminal vertices are called t-links. Thus, we updated the graph G to $G' = (V', E')$,A simple 2D example of an updated undirected graph is shown in Figure 4.3.Therefore,

$$V' = P \cup \{s, t\}$$
$$E' = N \cup \bigcup_{p \in P} \{\{p, s\}, \{p, t\}\}$$

To partition an image into foreground and background, we use the technique of s-t cut. An s-t cut C is a subset $C \subseteq E$ of the set of edges E, which makes it possible to completely separate the two terminal points S and T in the graph after the cut and to split the nodes into two distinct subsets according to the source points. The cut can generate binary partitions with arbitrary topological properties. Let $\mathbf{A} = (A_1, \ldots, A_p, \ldots, A_{|P|})$ be a binary vector whose components A_p specify assignments to pixels p in P. The goal of segmentation is to assign a label f_p to each pixel A_p in the image, where each f_p can be either 1 or 0, representing "object" and "background," respectively.

Taking Figure 4.4 as an example, the left figure shows the construction of the graph, where S and T represent two different labels. The right figure shows the division of the graph into two parts by cutting. Pixels connected to s after a cut

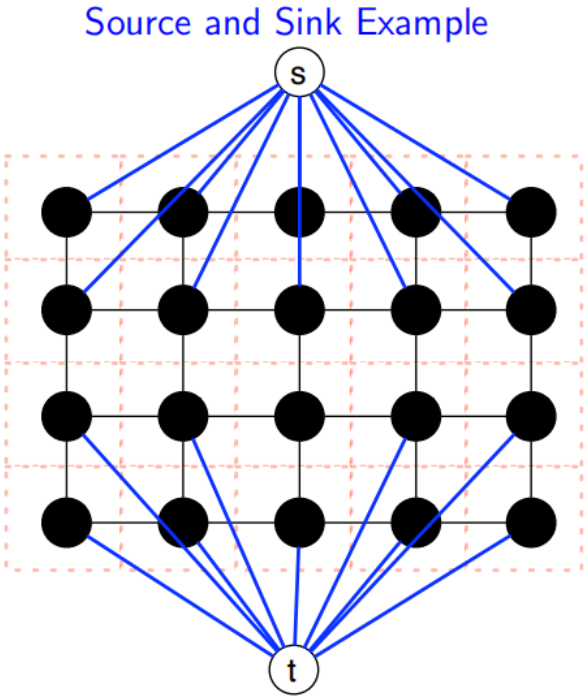


Figure 4.3: Undirected graph with source and sink nodes

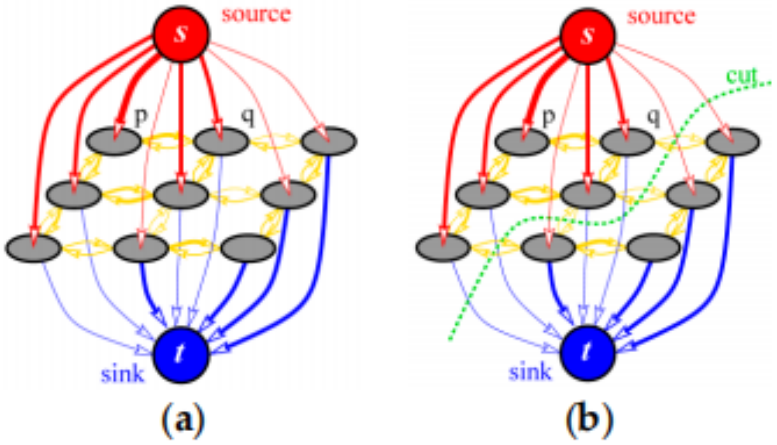


Figure 4.4: An example of a cut

are assigned to the label S, and pixels connected to t are assigned to the label T.

In practice, any cut will split the image into "object" and "background" parts. However, our goal is to find the optimal cut that not only efficiently splits the image

into foreground and background but also minimizes the cost of the cut, thereby ensuring the highest quality of the segmentation.

In combinatorial optimization, the cost of a cut is defined as the sum of the costs of the edges it cuts:

$$|C| = \sum_{e \in C} w_e.$$

4.2 Graph Edges

A graph cut operates on the edges of a graph. When converting an image into a graph, defining the edges is crucial because it directly influences the structure of the graph cut and the outcome of image segmentation. Critical aspects of defining edges include the connectivity pattern, whether the edges are directed or undirected, and the definition of edge weights.

1. **Edge Connectivity**

   Edge connectivity determines which pixels are connected by edges and is typically based on their adjacency. In 2D images, common patterns include 4-neighborhood and 8-neighborhood. The 4-neighborhood connects each pixel to its top, bottom, left, and right neighbors, while the 8-neighborhood also includes the diagonal neighbors.

   Common grid structures include square grids, triangular grids, and quadrilateral grids. As shown in the Figure 4.5, 4.6 and 4.7 they are composed of square, triangular, and irregular quadrilateral cells, respectively.

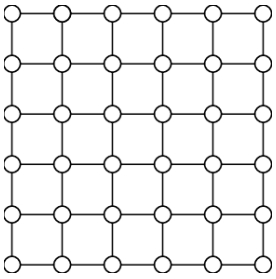


Figure 4.5: Square Grid

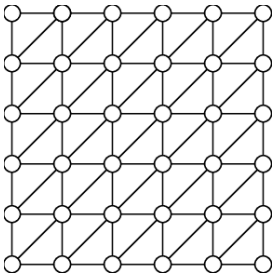


Figure 4.6: Triangular Grid

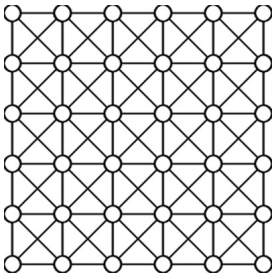


Figure 4.7: Quadrilateral Grid

2. **Edge Directionality**

   Edges can be directed or undirected. Undirected edges indicate symmetric interactions between pixels, while directed edges represent relationships in a

specific direction, such as the probability of moving from one pixel to another. In image segmentation algorithms, undirected edges are typically used because the similarity between pixels is symmetric.

In this thesis, n-links are considered undirected edges that connect neighboring pixels, whereas t-links are treated as directed edges with direction $s \to (u, v) \to t$. For the subsequent image segmentation process, particularly using graph cut methods, we regard n-links as 2 anti-parallel edges with equal weights. In addition, we conduct control experiments using n-links as unidirectional edges in the direction from s to t to verify the effect of edge directionality on the segmentation results.

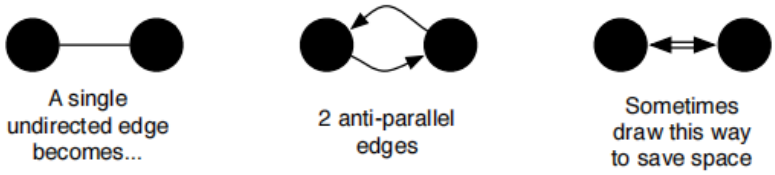


Figure 4.8: Treat undirected edges as 2 anti-parallel edges with the same weight.

3. **Edge Weights**

   In the graph construction for image segmentation tasks, each edge $\{v_i, v_j\} \in E$ is assigned a corresponding non-negative weight w_{ij}, which represents the cost of removing this edge from the graph and can also be viewed as a penalty. These weights reflect the similarity between neighboring pixels in the image, where higher weights indicate stronger similarities or connections between pixels, while lower weights imply weaker similarities or connections. In this thesis, weights are determined based on the color similarity of pixel values. As discussed in Section 3.1.1, the analysis primarily focuses on examining color differences within the RGB and LAB color spaces.

In the next section, we will detail how n-links (connections between nodes) and t-links (connections between nodes and sources or sinks) can be used to construct an exact partition energy function. This energy function quantifies the "cost" of any given segmentation, allowing us to find the optimal image segmentation by minimizing this function. The minimum cost cut C on the graph G can be calculated precisely in polynomial time.

4.3 Segmentation Energy

The graph-cut model proposed by Boykov and Kolmogorov in 2004 cleverly integrates region and boundary information and coordinates them through a regularization parameter usually determined experimentally. This achieves an ideal balance between region and boundary properties. Thus, the segmentation energy function applied to the boundary and region properties of object A can be described as follows:

$$E(A) = \lambda \cdot R(A) + B(A) \tag{4.1}$$

where

$$R(A) = \sum_{i \in V} R_i(A_j) \quad \text{(regional term)} \tag{4.2}$$

$$B(A) = \sum_{\{i,j\} \in N} B_{i,j} \cdot \delta_{A_i + A_j} \quad \text{(boundary term)} \tag{4.3}$$

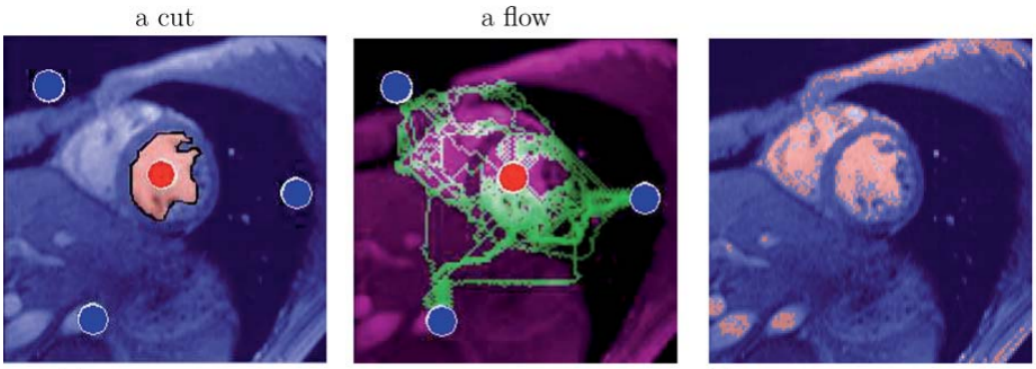
and

$$\delta_{A_i + A_j} = \begin{cases} 1 & \text{if } A_i \neq A_j, \\ 0 & \text{if } A_i = A_j \end{cases}$$

The energy function $E(A)$ quantifies the cost of a graph cut, and minimizing this energy leads to optimal segmentation. The function consists of the region term $R(A)$ and the boundary term $B(A)$. The integration of these two components constitutes the essence of the graph cut approach. Here, the regularization parameter λ acts as the relative importance factor balancing the regional and boundary terms on the energy function. Setting λ to zero causes the approach to focus only on boundary details and ignore region information. As shown in Figure 4.9 (a), segmentation using mainly boundary cues tends to be performed along places with significant differences in edge intensity, demonstrating the ability to separate regions based on visual discontinuities. While (b) segmentation using only regional cues ignores the direct relationship between neighboring pixels by assuming that the capacity of all connections (n-links) is zero [23].

4.3.1 Construction of Boundary Information

To quantify the similarity between neighboring pixels in an image, $B_{i,j}$, known as the boundary term $B(A)$, is used to represent the weights of the n-links between pixels i and j. This term evaluates whether the two pixels belong to the same region by calculating their similarity. If the difference between two neighboring pixels is small, they are likely to belong to the same object or background. In contrast, a significant difference indicates that these pixels may be located at the edges of the object and background, resulting in a higher likelihood of segmentation.



(a) Boundary cues with topological constraints (b) Region-based cues (only)

Figure 4.9: Figure 4.9 presents two basic examples of object extraction using s/t graph cuts. In (a), image pixels form a 2D grid graph where neighboring pixels are connected by n-links, with capacities based on intensity differences $|I_p - I_q|$. Seeds are connected to terminals (source and sink) by t-links. High-cost t-links hardwire seeds to terminals, providing topological constraints for segmentation. Max-flow algorithms increase flow from source to sink until edges saturate, forming a boundary (cut). Example (a) relies on boundary-based cues from n-links, whereas (b) demonstrates graph cut segmentation using only t-links, encoding region-based cues. Here, all n-links have zero capacity, and each pixel p is connected to terminals with t-link capacities $|I_p - A|$ and $|I_p - B|$, resulting in segmentation equivalent to thresholding around intensity level $\frac{A+B}{2}$. This graph cut approach combines boundary cues, regional cues, and topological constraints in a unified global optimization framework.

In addition, $B_{i,j}$ can also decrease as the distance between i and j changes. The total cost of the cut-off n-links reflects the total cost of forming these segmentation boundaries. For the term $B_{i,j}$ is defined in the following:

$$B_{i,j} \propto \exp\left(-\frac{\Delta d_{ij}^2}{2\sigma^2}\right) \cdot \frac{1}{\text{dist}(i,j)}. \tag{4.4}$$

- B_{ij} is the weight of the edge between pixels i and j.
- Δd_{ij} represents the color distance between pixels i and j.
- σ is a parameter that controls the strength of the weight.

The weight function $w(i,j)$ of n-link can be expressed as follows:

$$w_{ij} = e^{-\frac{\|\Delta d_{ij}\|^2}{\sigma_I^2}} \cdot \begin{cases} e^{-\frac{\|X(i)-X(j)\|^2}{\sigma_x^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

- $X(i)$ is the spatial location of node i, and Δd_{ij} is color distance between pixels i and j. Note that the weight $w_{ij} = 0$ for any pair of nodes i and j that are more than r pixels apart.

Figure 4.10 shows a 2D grid where each node represents a pixel, and the lines between nodes represent connections between pixels (n-links). The values on these lines indicate the weights between the pixels.

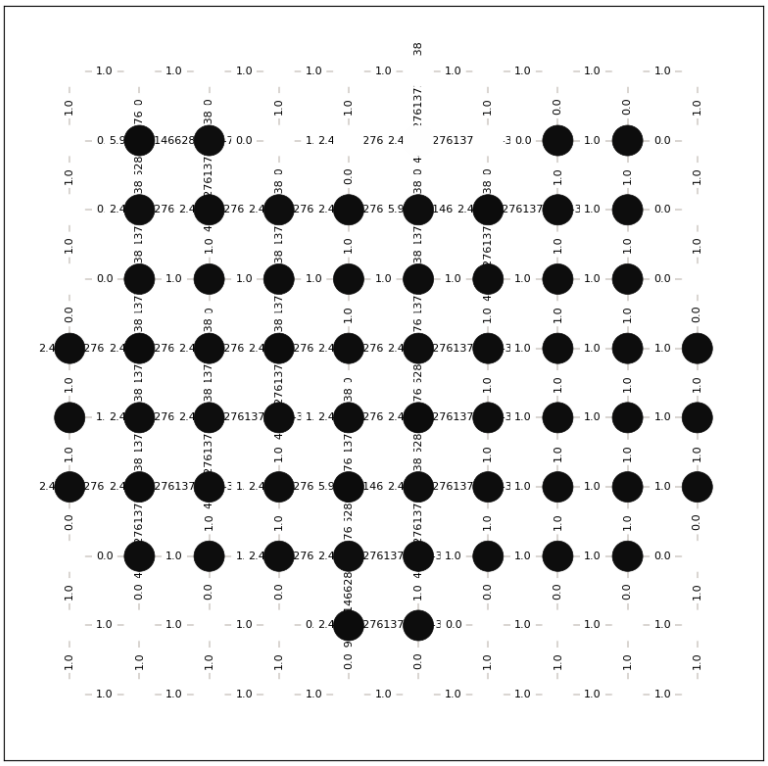


Figure 4.10: A 2D grid illustration, the numbers on the connecting lines indicate the weights between neighboring pixels, reflecting the similarity between the pixels.

4.3.2 Construction of Region Information

The region term $R(A)$ describes the likelihood that a pixel belongs to a particular region (foreground or background). This metric is usually based on features such

as intensity, color, texture, etc. In this thesis, we also consider color as a key feature for analysis. The region term evaluates the cost of assigning each pixel to a particular segmentation region, thus facilitating the division of pixels with similar properties into the same region. In the previous section 4.1, we added two terminals, s and t, associated with the foreground and background and created t-links for each pixel to these two terminals, the weights of these two edges $R_p(\text{“obj”})$ and $R_p(\text{“bkg”})$ denote the cost of pixel p belonging to the foreground and background, respectively. For example, a pixel with a high feature match to the foreground model will have a low-weight t-link to the source node s and a high-weight t-link to the sink node t, indicating that this pixel has a lower cost of assignment to the foreground and a higher cost of assignment to the background.

In practice, the exact computation of $R_p(\text{“obj”})$ and $R_p(\text{“bkg”})$ is based on the requirements of the particular application and the a priori knowledge available. Some automatic segmentation is based on a priori knowledge, such as image edges, color histogram peaks, or specific texture patterns. For example, as demonstrated by Greig et al. and Boykov et al., image segmentation algorithms use Negative Log-Likelihoods and the Maximum A Posteriori Probability (MAP) - MRF formulas to compute the region term $R_p(\cdot)$ [12]. When color is used as the feature value, I_p represents the color of the pixel p, and the probabilities $\Pr(C_p \mid \text{“obj”})$ and $\Pr(C_p \mid \text{“bkg”})$ denote the likelihood that the pixel p has a certain color given that it belongs to the object or background, respectively. In this thesis, $R_p(\cdot)$ may reflect on how the color of pixel p fits into given color space of the object and background. The weight of the t-links is defined as following equations:

$$R_p(\text{“obj”}) = -\ln \Pr(C_p|\text{“obj”}) \tag{4.6}$$

$$R_p(\text{“bkg”}) = -\ln \Pr(C_p|\text{“bk”}) \tag{4.7}$$

4.4 Manual vs. Automatic Seed Selection

Numerous methods have been developed to model region information, including histogram-based and clustering-based methods. Histogram-based methods operate by counting the bright and dark pixels in the image, but such methods often have difficulty in constructing sufficient histograms of the color space and thus face challenges in color image segmentation. Meanwhile, clustering-based methods analyze image features by applying clustering algorithms and use the clustering results to represent a priori labeling information. Commonly used clustering algorithms include the K-means algorithm and the Gaussian Mixture Model (GMM). In addition, some methods may take advantage of the statistical properties of the image to initialize the segmentation process, for example, by assuming that the brightest or darkest regions of the image belong to a particular group. Image

segmentation can also be implemented using a combination of algorithms: without seeds, a simpler segmentation method (e.g., threshold segmentation, region growing) can be used to obtain an initial result, which can then be used as input to Graph Cut to refine the segmentation.

This section will explore two approaches to acquiring regional information: one relies on user-provided seed points, and the other employs automatic clustering techniques, such as K-means clustering, to obtain prior knowledge.

4.4.1 Interactive Graph Cuts: User Enter Seeds

A common and effective strategy is to use seeds input by the user to get the region information. The user selects pixels that belong to the foreground (objects) and the background as seeds (sources and sinks), and this approach uses a priori knowledge provided by the user to initialize and guide the segmentation process. This provides a set of topological (hard) constraints for image segmentation, which avoids blindly exploring the entire image and greatly improves the efficiency and accuracy of segmentation.

Smart Scissors [59] and Real-Time Lines [60] utilize boundary-based hard constraints, which allow the user to specify the particular pixels through the segmentation boundaries that should be passed through. The "shortest path" between the labeled pixels is then computed as the segmentation boundary according to an energy function based on the gradient of the image. The difficulty with this hard constraint is that the user's input must be very precisely localized to the desired boundary. In comparison, Boykov first proposed a new approach to interactive segmentation of N-dimensional images, which allows the user to participate directly in the segmentation process by selecting regions of interest in the image or by specifying seed points to achieve more accurate object extraction without requiring the user to locate them precisely. Moving the seeds appropriately around the object (within a certain range) usually does not change the segmentation results. [23]

During the image segmentation process, the user designates special source point s (representing the foreground) and sink point t (representing the background) by labeling specific pixels as "O" (foreground seed) and "B" (background seed), respectively, where $O \subset V$ and $B \subset V$, ensuring that $O \cap B = \emptyset$.

The following table gives weights of edges in $\mathcal{E}$:

where

$$K = 1 + \max_{p \in P} \sum_{q:\{p,q\} \in \mathcal{N}} B_{p,q}.$$

$$\text{weight} = \begin{cases} B_{p,q} & \{p,q\} \in \text{Neighboring pixel} \\ \alpha \cdot R_p(0) & \text{for edge } \{p,S\} \\ \alpha \cdot R_p(1) & \text{for edge } \{p,T\} \end{cases} \tag{4.8}$$

edge	weight (cost)	for
$\{p, q\}$	$B_{p,q}$	$\{p, q\} \in \mathcal{N}$
$\{p, S\}$	$\lambda \cdot R_p(\text{“bkg”})$	$p \in P, p \notin O \cup B$
	K	$p \in O$
	0	$p \in B$
$\{p, T\}$	$\lambda \cdot R_p(\text{“obj”})$	$p \in P, p \notin O \cup B$
	0	$p \in O$
	K	$p \in B$

Table 4.1: Weights of edges

As shown in Table 4.1, when a pixel is labeled as an object (i.e., foreground), the t-link weight between that pixel and the node s is set to be large, while the t-link weight between that pixel and the node t is set to 0. In contrast, when a pixel is labeled as the background, that pixel has a high weight between it and the node t, and the weight of the t-link between that pixel and the node s is set to 0. This configuration ensures that the approach can accurately distinguish between foreground objects and background based on the seeds provided by the user.

4.4.2 Fully Automated Image Segmentation Relying on K-means

Based on sections 4.4.1, it is found that the selection of seed points is crucial to the effectiveness of the Graph Cut method. However, relying on users to manually select seed points each time significantly increases operational burden, affects the efficiency of image segmentation, and reduces the repeatability of the process. Moreover, non-expert users may have difficulty accurately selecting the optimal seed points, impacting the quality of the segmentation results. Therefore, the automated selection of seed points for foreground and background is an important research direction in image segmentation.

The following are several methods and techniques that are currently widely used for automated seed selection.

1. **Based on Color Histogram Analysis** By analyzing the color histogram of an image, regions with concentrated or significantly different color distributions can be automatically selected as seed points. This method first calculates the color histogram of an image in a selected color space (e.g., RGB, HSV) and counts the pixel distribution of each color channel. High-frequency color regions usually represent the main objects in the image. They can be used as foreground seed points, whereas low-frequency color regions or regions near the edges of the image are used as background seed points.

This approach is more common in images with significant color contrast or large foreground-background color differences, such as medical image segmentation [61]. In tumor or lesion detection, lesion areas and healthy tissues often have significant differences in gray-level distribution. By analyzing the histogram of medical images (e.g., MRI or CT), regions with high-frequency gray levels can be identified as foreground seed points, which often correspond to lesions. Regions with low-frequency gray levels are used as background seed points, representing normal tissues or backgrounds. Medical images usually have less lighting variation, reducing the impact of illumination changes on histogram analysis.

This approach is easy to use, doesn't require much computation, and is well-suited for real-time applications [62]. However, it is less effective for images with complex colors or mixed foreground and background colors and is sensitive to illumination changes [63]. Therefore, combining this method with other techniques may be necessary to improve accuracy and robustness in complex image processing [61].

2. **Clustering-Based Algorithms**
   Image segmentation methods based on clustering algorithms, especially K-means clustering, are highly efficient at estimating an image region's predominant color by calculating the clusters' center. A property that makes K-means a tool that can automatically generate accurate seed points, serving as a starting point for image segmentation or further processing. Despite limitations such as sensitivity to noise and outliers, neglect of spatial continuity between pixels, and difficulties when dealing with complex images, these carefully selected seed points are used in the Graph Cut method as initial markers for automated seed selection. Automated selection of seed points can significantly improve the efficiency and accuracy of image segmentation and reduce the need for manual intervention.

   This method shows good adaptability and efficiency in practical applications, especially for images with significant differences in color or location. For example, studies have shown that automated segmentation of medical images can be effectively realized by combining K-means clustering and Graph Cut methods to improve the accuracy and efficiency of segmentation [64] [65]. These studies support the promise of K-means clustering as a powerful tool for image segmentation tasks.

(a) **Color Selection of Cluster Centers**

In image processing using the K-means clustering algorithm, selecting the colors of the cluster center is a critical step, as it directly affects the accuracy and effectiveness of segmentation. Typically, the cluster centers' colors represent their regions' main color features. In practical applications, there are two common methods for determining the colors of the cluster centers: selecting the color of representative pixels near each cluster center or calculating the average color of all pixels within the cluster.

Choosing the color of pixels near the cluster center as the representative color is intuitive and easy to implement. However, this method might occasionally select anomalous pixels, especially in images with noise or uneven color distribution. Such color anomalies can mislead the clustering results, affecting the quality of the final image segmentation.

Calculating the average color of all pixels within the cluster provides a more stable and reliable option. This method comprehensively considers the color information of the entire region, effectively reducing the risk of erroneous segmentation. The average color reflects the overall color trend within the cluster, enhancing the robustness of the clustering and making the results more stable and reliable. Therefore, in applications requiring high accuracy and robustness in image segmentation, using the average color of the region as the cluster center color is generally a better choice. This method avoids mis-segmentation due to color anomalies and provides more consistent and accurate image processing results.

(b) **Pre-defining the K Value**

Pre-defining the K value is a critical step when using the K-means clustering algorithm for image segmentation. Directly setting the K value to 2 to divide the image into foreground and background, while straightforward and easy to understand, may be insufficient for processing complex or detail-rich images. When the K value is set to 2, the algorithm generates only two cluster centers, capturing only the two most prominent colors or features in the image, and may overlook other important visual information, leading to sub-optimal segmentation quality. Additionally, since each cluster center's color is the average of all pixel colors in that region, this could result in a deviation from the actual colors of some pixels. Especially when the color distribution between the foreground and background is similar or the color transition is smooth, setting the K value to 2 and using the average color as a representative may fail to capture the actual color variations accurately, thus impacting segmentation accuracy.

As the K value increases, more cluster centers are created, allowing a

finer division of colors and features within the image. This results in the cluster center colors more closely reflecting the actual color diversity in the image. This approach not only better handles images with complex color distributions but also captures more details, thereby enhancing the accuracy and effectiveness of the segmentation.

However, the K value should not be set too high. When the K value is too high, it can lead to over-segmentation of the image, breaking up even continuous regions into multiple small segments. This can make it difficult to distinguish between foreground and background seed points, increasing the complexity of subsequent processing. Additionally, as the K value increases, the computational burden also rises, and the clustering's stability against noise may decrease, leading to unstable results and a loss of algorithm interpretability. Therefore, by appropriately adjusting the K value, it is possible to maintain the interpretability of the algorithm while enhancing its adaptability and accuracy in handling different types of images. Figures 4.11, 4.12, and 4.13 illustrate the logic behind using the K-means clustering for automated seed selection and compare the results of the segmentation with different values of k.

An in-depth comparative analysis of manual and automatic seed placement methods and their impact on image segmentation results will be presented in Chapter 5. The experiment aims to discuss the impact of each seed setting method on the segmentation process and delineate their advantages and disadvantages in different scenarios.

Once we have the energy equation and the edge weights in the graph (both n-links and t-links), we can apply the mathematical formulas of Chapter 3 to solve the minimum-cut problem of the energy equation. This allows us to accurately segment the target object and the background in the image.

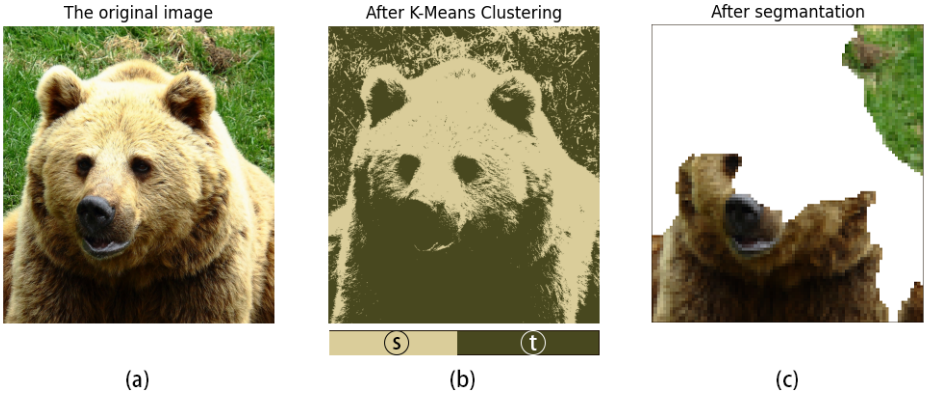


Figure 4.11: The automated segmentation results using k-means with $k = 2$

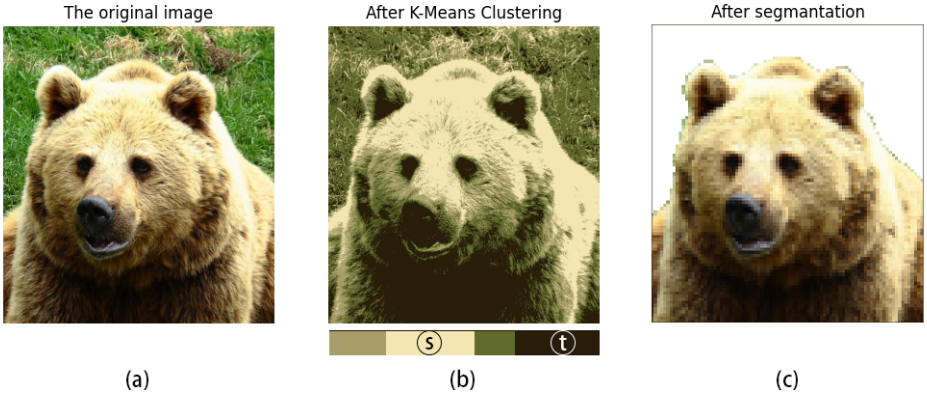


Figure 4.12: The automated segmentation results using k-means with $k = 4$

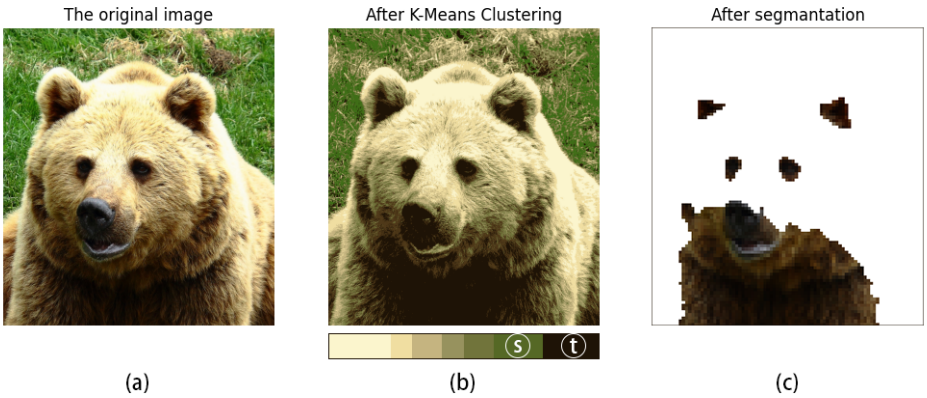


Figure 4.13: The automated segmentation results using k-means with $k = 8$

Chapter 5

Experiments and Results

5.1 Pre-experimentation Preparation

5.1.1 Dataset Descriptions

In this thesis, we use images from the widely recognized PASCAL Visual Object Classes (VOC) dataset to thoroughly evaluate the effectiveness of the Min-Cut image segmentation method [66]. The PASCAL VOC dataset is a standard benchmark in computer vision, used for various tasks such as image classification, object detection, and image segmentation. This dataset is highly regarded and has been the basis for many influential papers and neural network models.

The VOC2007 dataset includes twenty diverse object categories, such as vehicles, household items, animals, and other objects. The variety in these categories offers a rich set of visual content, allowing our segmentation method to demonstrate its accuracy and effectiveness across different complex environments. Figure 5.1 shows the 20 object categories in the VOC2007 dataset.

For the segmented image set, each image is accompanied by two types of precise annotations: class segmentation and object segmentation. In class segmentation, each pixel is labeled as a category or background. In object segmentation, each pixel is labeled with an object number (from which the category can be determined) or background. Figure 5.2 shows examples of these two segmentation types for an image in the training set.

Since this thesis focuses on image segmentation, primarily targeting the separation of object and background, we selected images with a single class object from the 20 categories as the dataset and used class segmentation annotations for evaluation.

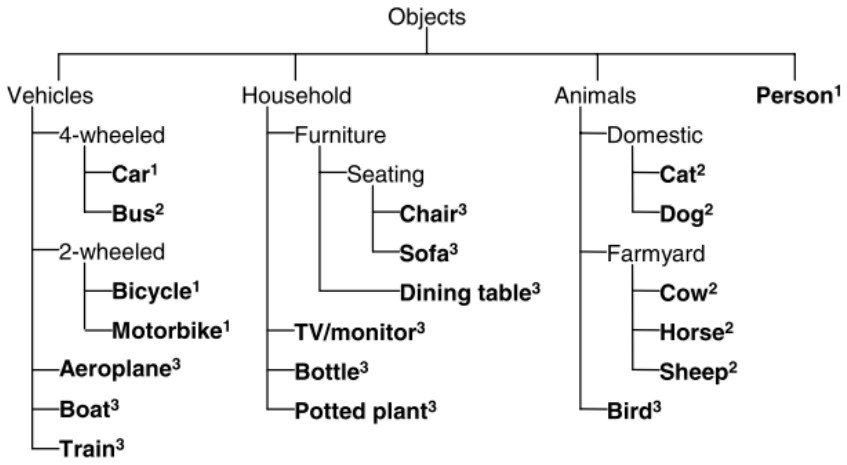


Figure 5.1: VOC2007 classes. Leaf nodes correspond to the 20 classes. [66]

Figure 5.2: Example of segmentation ground truth. a. Training image b. Class segmentation showing background, car, horse and person labels. The cream- colored 'void' label is also used in border regions and to mask difficult objects. c. Object segmentation where individual object instances are separately labelled. [66]

5.1.2 Gurobi Optimizer

In operations research, linear programming (LP), integer programming (IP), and mixed integer programming (MILP) typically involve numerous variables and constraints. Although optimization problems with continuous variables and dozens of constraints can be solved using existing algorithms, real-world optimization

problems often involve thousands of variables and constraints, making manual solutions or simple algorithms impractical, particularly for discrete models, where computing the optimal solution is highly complex.

As discussed in Section 3.3, while specialized algorithms such as Ford-Fulkerson are often applied to the standard Min-Cut problem for image segmentation, real-world applications frequently require additional constraints and objectives. For example, considerations such as region consistency, edge smoothing, and region-specific shape constraints might be necessary. These complex constraints and objective functions can be challenging to implement with specialized graph-cut algorithms. However, modeling the Min-Cut problem as an integer linear programming (ILP) model allows for a more flexible expression of these constraints and objectives.

Moreover, as image resolution and application requirements increase, the size of the image segmentation task can become substantial, involving thousands of pixels and the corresponding variables and constraints. Specialized graph cut algorithms perform well on small-scale problems but may face performance bottlenecks on larger problems. Such large-scale problems require specialized, high-quality solvers such as Gurobi Optimization.

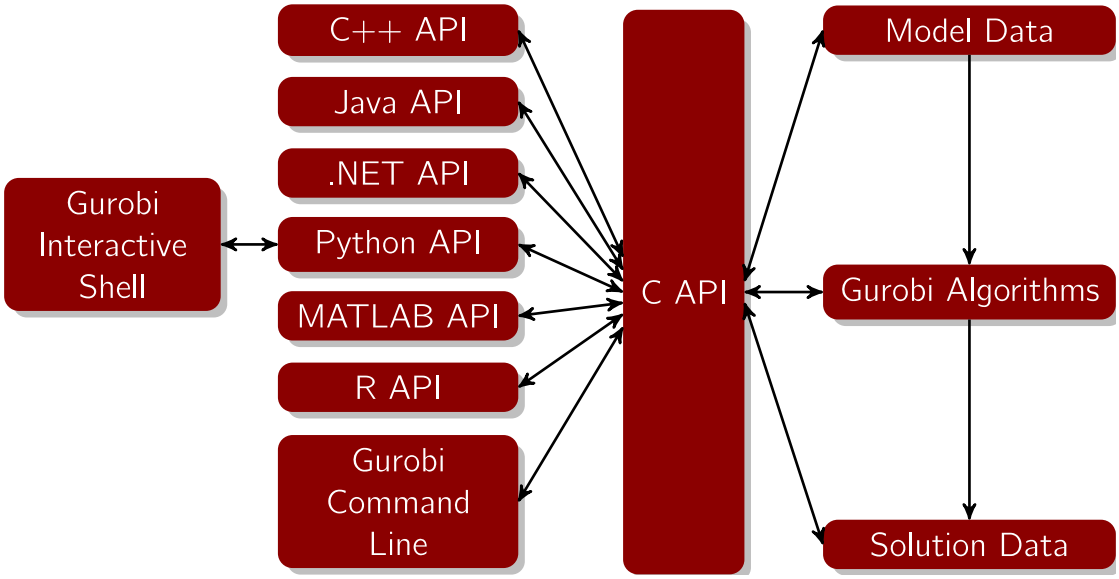


Figure 5.3: The general API architecture of Gurobi Optimization

Gurobi Optimization is a high-performance, general-purpose optimization solver capable of handling large-scale ILP problems and effectively scaling to larger problem sizes [67]. It is particularly adept at solving complex integer programming problems using advanced algorithms such as Simplex, Interior Point, and Branch-and-Bound. The Gurobi Optimization is optimized to find high-quality solutions

within a reasonable time frame, employing multi-threaded parallel computation, heuristics, and preprocessing techniques to enhance solution speed and quality. It excels in managing complex optimization problems of varying sizes and complexities [68].

Gurobi Optimization supports various programming languages and modeling tools, As shown in Figure 5.3, including Python, C++, Java, and MATLAB, providing flexible interfaces and extensive functionality. These features allow for easy integration with existing systems and workflows.

In summary, Gurobi Optimization is widely regarded as the best solver for large-scale and complex optimization problems due to its efficient solving capabilities, user-friendly programming interface, broad application range, and reliability and accuracy.

In this chapter, the performance of the two mathematical models proposed in chapter 3 will be experimentally compared in terms of image segmentation results.

5.1.3 Image Pre-processing

Despite Gurobi's exceptional performance in solving complex integer programming problems, its runtime and memory requirements can rapidly increase with the input size. In the graph cut image segmentation task, it is necessary to pre-process larger images to improve computational efficiency and reduce resource consumption significantly. In this thesis, we resize larger images to dimensions below 100×100 pixels. This approach effectively reduces the number of nodes in the network, while preserving as much of the essential information of the images as possible. Consequently, the structure of the graph is simplified, making the computation of the Min-Cut algorithm more efficient. By adopting this method, we can save computational resources and handle more data within the constraints of limited hardware conditions.

5.1.4 Evaluation Indicator

- **Intersection over Union (IoU))**

  Intersection over Union (IoU), also known as the Jaccard Index, is a metric used to evaluate the accuracy of image segmentation.

  IoU is defined as the area of intersection between the predicted segmentation map A and the ground truth map B, divided by the area of the union between the two maps:

$$\text{IoU} = J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

A larger IoU value signifies that the predicted segmentation closely matches the ground truth, leading to more accurate and reliable segmentation results.

- **Association of node**

  Association is a metric used to measure the compactness within segmented regions in image segmentation. Assoc(A, V) is the sum of all edges with one end in A. Essentially, it is the sum of all weights for every element in the partition A connected to any element in V.

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t) \tag{5.2}$$

  A larger Assoc value indicates greater similarity within each segmented region, stronger connections between nodes within the region, and thus higher internal compactness. Consequently, a larger Assoc value typically leads to better segmentation results, as it reflects more cohesive and well-defined regions.

5.2 Experimental and Results Analysis

In this thesis, we used a system running Windows 11 (version 10.0.22631) on an AMD64 architecture. The hardware was powered by an Intel(R) Core(TM) i9-14900HX processor with a base frequency of 2.20 GHz (Intel64 Family 6 Model 183 Stepping 1, GenuineIntel) and an NVIDIA GeForce RTX 4060 laptop GPU.

The software environment included Python version 3.12.2 and Gurobi Optimizer version 11.0.0 for solving optimization problems.

We conducted three sets of experiments to progressively analyze the independent impact of each parameter on image segmentation performance. The main variables considered in the experiments are as follows, with detailed descriptions provided in Section 4:

- **Color Spaces (2 types)**: RGB and LAB
- **Edge Connection Types (3 types)**: Lattice, Triangulated, Quadrilateral
- **Edge Directionality (2 types)**: Unidirectional and Bidirectional

Experiments 1, 2, and 3 utilized Mathematical Model 1 to solve the minimum cut of the energy equation for each model, employing the Gurobi Optimization solver for the computations. The Python script used to perform Min-Cut Optimization with Gurobi is shown in Listing A.1. To avoid the influence of external factors on the experimental results and to ensure precision and consistency, manually selected

source and sink seed points were used in all three experiments to achieve the most optimal segmentation results.

Besides these variables, we investigated the impact of various experimental setups. Experiment 4 explored the influence of user-interactive seed point selection in contrast to automated seed point selection with k-means clustering on the efficacy of image segmentation, whereas Experiment 5 evaluated the outcomes of two different mathematical models (detailed in Chapter 3).

5.2.1 Experiment 1

We designed the experiment with two models for comparison to examine the effect of image segmentation in different color spaces. Model 1 utilizes the RGB color space, using the RGB color distance between pixels as the weight of the edges between nodes. The edges are connected in a quadrilateral manner and are bidirectional. Model 2 employs the same type of edge connection and bidirectional edges as Model 1 but is constructed in the LAB color space.

Figure 5.4: Sample segmentation results from the VOC 2007 segmentation dataset. The first two lines are the original images and ground truth. The third and fourth models use RGB color distance and LAB color distance as edge weight, respectively.

Figure 5.4 presents sample segmentation results from the VOC 2007 segmentation dataset, including original images, ground truth, and segmentation results from two different models. The ground truth provides precise segmentation contours

for each object, serving as a reference standard for evaluating model performance. From Figure **??**, we observe that when using the RGB color distance as the edge weight, the model performs well in segmentation but may show some errors in detailed areas. The LAB color-distance model typically delivers smoother and more accurate segmentation edges, particularly excelling in complex scenes and detailed processing. Overall, the LAB color-distance model outperforms the RGB model in segmentation accuracy and detail preservation.

From Table 5.1, it can also be seen that while achieving better segmentation results, the LAB color-distance model processes faster than the RGB model. Model 2 also consistently outperforms Model 1 in terms of $Assoc(S)$ and $Assoc(T)$ values, indicating Model 2's ability to maintain strong links within segmented areas. Higher association values indicate that Model 2 effectively maintains the connections between pixel sets in each segmented area, enabling it to accurately identify and preserve the boundaries between different regions during image segmentation tasks.

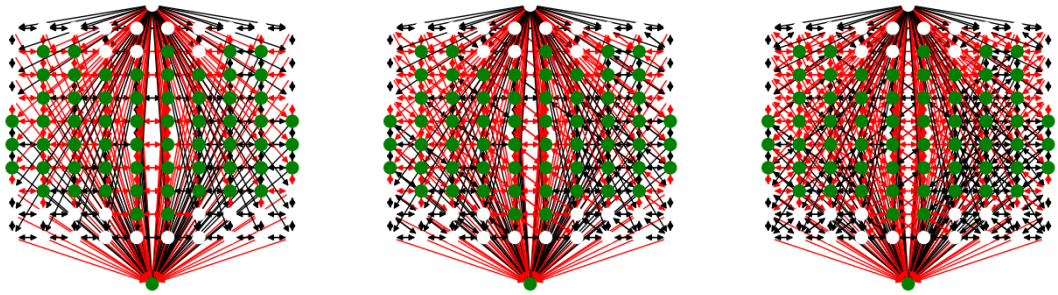
Table 5.1: Quantitative comparison of each model in different color spaces. We display runtime, Intersection over Union (IoU), and association of S and T nodes.

Instance	Model 1				Model 2			
	Time (s)	IoU (%)	Assoc(S)	Assoc(T)	Time (s)	IoU (%)	Assoc(S)	Assoc(T)
Aeroplane	35.34	65	69255	737	8.36	71	95369	2900
Dining table	9.22	88	53097	7037	8.36	90	65931	11490
Person	29.60	89	26930	18558	18.55	92	38741	27841
Sofa	24.55	85	14808	34815	22.09	86	20399	52216
Sheep	22.99	38	9757	6716	19.75	45	33866	12201

5.2.2 Experiment 2

We developed the experiment using three different models to evaluate how the type of edge connection influences the performance of image segmentation. Each of the three models employs the LAB color space and bidirectional edges but differs in their edge connection types. In detail, Model 3 implements a mesh edge connection, Model 4 utilizes triangular connections, and Model 5 adopts quadrilateral connections.

Figure 5.5 illustrates various edge connections in the network. This figure compares three network structures: (a) has fewer connections between nodes, while (b) and (c) show a marked increase in connections, indicating a more complex network relationship. The red and black lines represent the edges between the nodes.



(a) (b) (c)

Figure 5.5: Illustration of a network layout displaying connections among nodes. Object nodes are depicted in green, while background nodes are shown in white. Red lines represent the edges that have been cut.

Figure 5.6 shows that the lattice edge model in the third row may lack precision to capture more complex shapes. The triangular edge model in the fourth row excels at capturing details and complex shapes, although it might result in over-segmentation in certain areas. The quadrilateral edge model in the fifth row shows stable performance in capturing overall contours, effectively maintaining the integrity of the objects.

From Table 5.2, although the lattice edge model's segmentation performance is not as good as the quadrilateral edge model, it connects fewer edges between nodes. This reduction in connections speeds up the calculation of the minimum energy equation, thereby increasing the model's image segmentation speed.

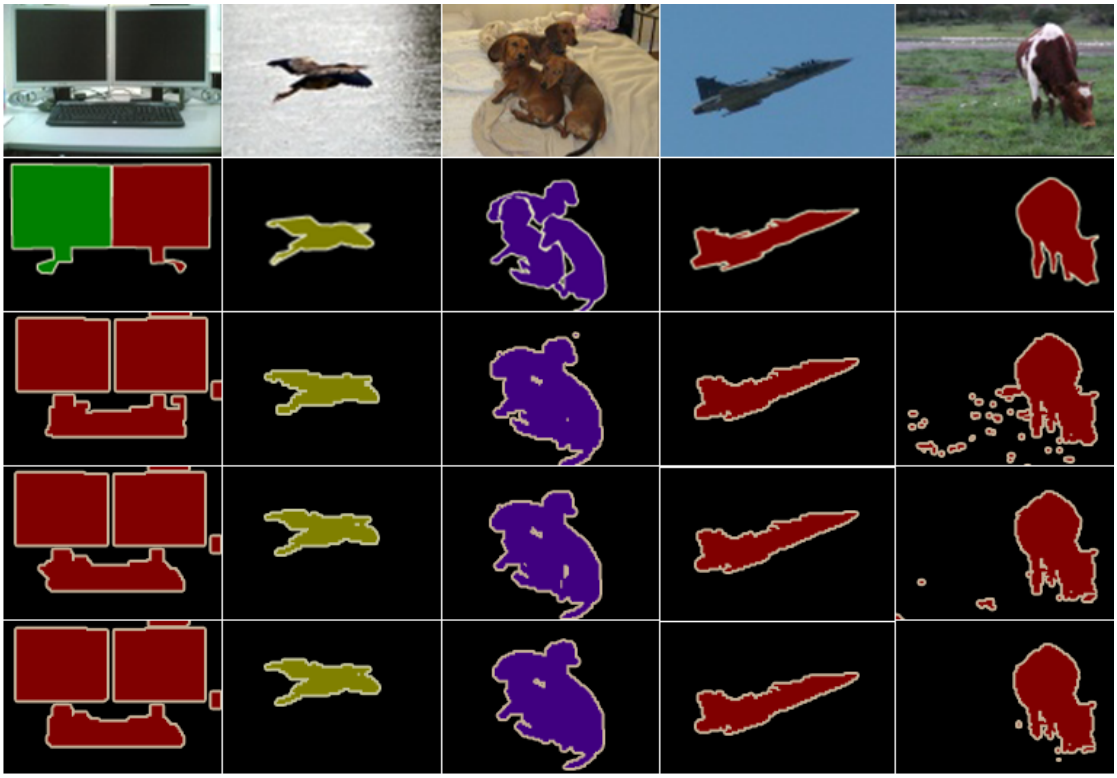


Figure 5.6: The figure illustrates segmentation outcomes from the VOC 2007 dataset. The top two rows contain the original images and the corresponding ground truth. The third, fourth, and fifth rows respectively demonstrate the model's application of lattice, triangular, and quadrilateral configurations for edge connections.

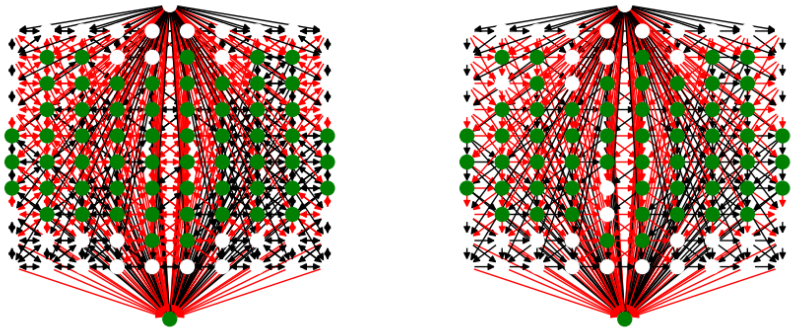
Table 5.2: Quantitative comparison of each model in different edge connection types. We display runtime, Intersection over Union (IoU), and association of S and T nodes.

Instance	Model 3		Model 4		Model 5	
	Time (s)	IoU (%)	Time (s)	IoU (%)	Time (s)	IoU (%)
TV/monitor	10.07	58	15.85	58	27.06	57
Bird	7.16	36	10.97	48	13.33	75
Dog	10.24	69	15.11	71	18.76	70
Aeroplane	11.37	88	15.69	87	24.81	87
Cow	8.64	71	25.76	86	36.43	86

5.2.3 Experiment 3

In the image segmentation process, the direction of edges determines how information is transmitted within the network, directly impacting the segmentation results. Specifically, when using color differences between pixels as the characteristic for segmentation, n-links are considered as two anti-parallel edges with equal weights. To examine the impact of edge direction on segmentation results, we conducted control experiments treating n-links as unidirectional edges directed from s to t. Model 6 uses the LAB color space, a quadrilateral edge connection type, and bidirectional edges. Model 7 employs the same color space and edge connection type but with unidirectional edges.

Figure 5.7 illustrates different types of network structure. Figure (a) shows a network structure with bidirectional n-links, while Figure (b) presents a structure with unidirectional n-links. Figure (b) shows that some object pixels are inaccurately segmented, whereas Figure (a) demonstrates better segmentation results with more precisely cut edges.



(a) (b)

Figure 5.7: Illustration of network structure with bidirectional n-links (a) and network structure with bidirectional n-links (b).

Figure 5.8 presents sample segmentation results from the VOC 2007 dataset. The first and third images in the second row show the effects of using bidirectional n-link for segmentation, while the second and fourth images show the results with unidirectional n-link. The bidirectional n-link model clearly captures the boundaries of the object with more accuracy, producing segmented regions that closely match the ground truth. In contrast, the unidirectional n-link results exhibit more segmentation errors and discontinuities. This may be because the unidirectional n-link restricts the flow of information, leading to insufficient handling of complex boundaries and, consequently, lower segmentation accuracy.

Figure 5.8: Sample segmentation results from the VOC 2007 dataset. The first row shows the original images and ground truth. The second row displays the model's segmentation results using bidirectional n-link and unidirectional n-link, respectively.

5.2.4 Experiment 4

Experiment 4 explored the impact of user-interactive seed point selection on image segmentation performance compared to k-means clustering-based automatic seed point selection. Apart from the seed point selection method, all other parameters were kept constant. We used quadrilateral edge connection in the LAB color space and bidirectional n-link segmentation.

Figure 5.9 shows sample segmentation results from the VOC 2007 dataset. The results demonstrate that the performance of seed point selection based on k-means clustering varies with different k values. A smaller k leads to coarse segmentation, failing to accurately capture image details, whereas a larger k can cause oversegmentation and a higher risk of misclassification. In contrast, user-interactive seed point selection typically yields precise segmentations, effectively capturing the features and details of the images. Table 5.3 also shows that under the user-interactive method, all instances exhibit high IoU values, with varying characteristics in Assoc(S) and Assoc(T), notably stable performance for the Cat

instance.

Although user-interactive seed point selection yields more accurate results, automatic segmentation methods have unique advantages. Automatic segmentation can efficiently handle large volumes of images. Therefore, user-interactive seed point selection is preferable for high-precision single-image segmentation. For processing large datasets with relatively lower precision requirements, k-means clustering-based automatic seed point selection is more suitable.

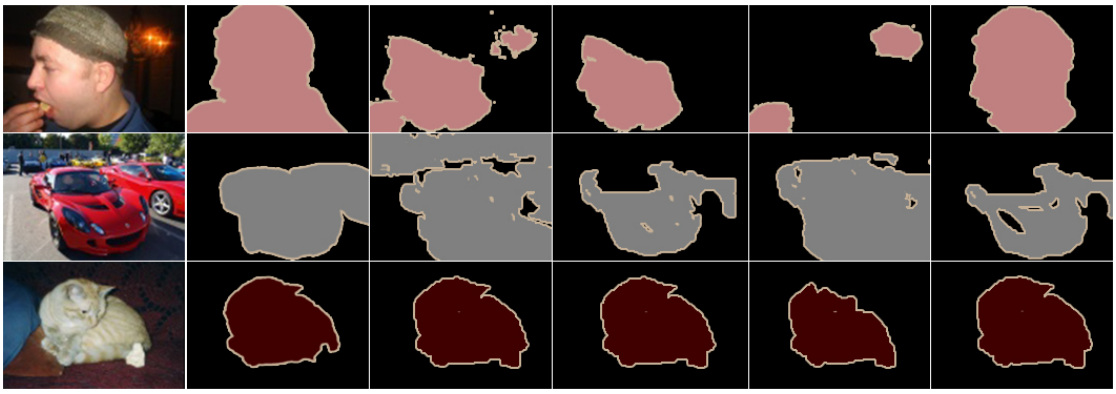


Figure 5.9: Sample segmentation results from the VOC 2007 dataset. The first column is the original image, and the second is the ground truth. Columns three to five show results from k-means clustering with k values of 2, 4, and 8. The last column displays user-interactive seed point selection results.

Table 5.3: Comparison of IoU (%), Assoc(S), and Assoc(T) for different k values and user-interactive methods across different instances.

Instance	k = 2			k = 4		
	IoU (%)	Assoc(S)	Assoc(T)	IoU (%)	Assoc(S)	Assoc(T)
Person	54	21700	3737	44.8	32742	8651
Car	46	17945	41161	62	41916	15844
Cat	90.6	66310	24263	91	58253	24295
Instance	k = 8			user-interactive		
	IoU (%)	Assoc(S)	Assoc(T)	IoU (%)	Assoc(S)	Assoc(T)
Person	7.9	83606	9191	74	59172	24220
Car	64	28380	28878	56	43635	14811
Cat	75	19886	62231	90.8	59172	24220

5.2.5 Experiment 5

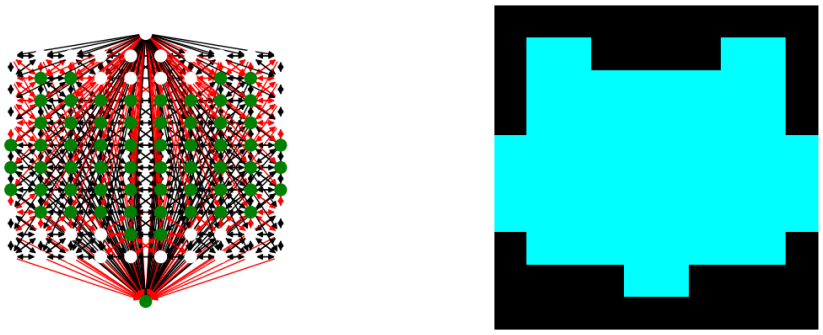
In this experiment, we implemented image segmentation using two different mathematical models. Model 1 is a classic approach based on the minimum cut of an energy function. Model 2 introduces additional variables and constraints.

We used the Gurobi Optimizer to solve the constructed minimum cut problem, finding the optimal segmentation by minimizing the energy function. The Python scripts to execute the minimum cut optimization are shown in Listings A.1 and A.2.

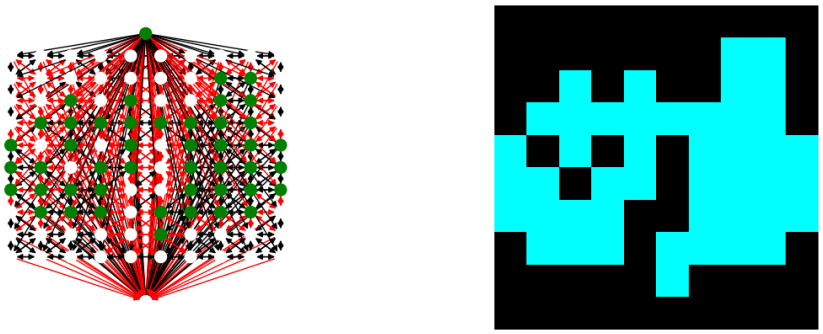
From Figures 5.10(a) and (b), Model 1 performs well in simple image segmentation. Figure (a) displays the graph structure of Model 1. Red edges indicate cut edges, black edges indicate uncut edges, and green and white nodes represent object and background pixels, respectively. Figure (b) shows the segmentation result, with the blue area representing the segmented target region and the black area representing the background. Model 1 effectively segments the target area with clear boundaries. In contrast, Figures 5.10(c) and (d) show that Model 2 performs worse on this simple image. Figure (c) shows the graph structure of Model 2, which introduces more variables and constraints, increasing the model's complexity. However, the segmentation result in Figure (d) is less accurate than that of Model 1, with the blue target area showing noticeable segmentation errors.

Figure 5.11 shows differences in the performance of the two mathematical models in practical image segmentation. Under the same conditions and experimental settings, Figure (a) shows the segmentation result using Model 1, while Figure (b) shows the result using Model 2. In Figure (a), Model 1 performs excellently, successfully segmenting the target (a bear) from the background. The bear's boundaries are clear and complete, with a distinct separation between the background and the target. In contrast, Model 2's result in Figure (b) is poorer, with the segmented bear image having incomplete boundaries and noticeable noise and error regions.

Overall, Model 1 demonstrates superior performance in image segmentation tasks. Although Model 2 incorporates more variables and constraints, it fails to deliver the anticipated results in practical applications. Therefore, Model 1 is recommended for practical image segmentation purposes.



(a) (b)



(c) (d)

Figure 5.10: Segmentation results obtained using two different mathematical models. The first row shows the results using Mathematical Model 1, while the second row shows the results using Mathematical Model 2.

(a) (b)

Figure 5.11: Segmentation results using two different mathematical models. (a) shows the results using Mathematical Model 1. (b) shows the results using Mathematical Model 2.

Chapter 6

Conclusion

This thesis examined and compared various Min-Cut models for image segmentation and analyzed the performance of different mathematical models in optimizing segmentation results. It also considered the advantages and disadvantages of automatic seed selection and user interaction methods.

Among all models evaluated, those using the LAB color space and quadrilateral edge connections exhibited superior performance, achieving higher segmentation accuracy and faster processing speeds. The analysis of edge directionality showed that bidirectional n-links more accurately captured object boundaries than unidirectional n-links. Among the two different Min-Cut mathematical models, the classical Min-Cut Model 1 generally outperformed Model 2, which introduced more variables and constraints. Therefore, it is recommended that the classical Min-Cut Model 1 be prioritized in realistic image segmentation tasks.

User-interactive methods significantly enhanced segmentation precision through interactive seed selection. These methods effectively captured image features and details often overlooked by automated approaches. User input helped guide the segmentation process, especially in complex images where automated methods might struggle. While less precise than user-interactive techniques, automated methods such as k-means clustering performed well in handling large datasets, providing a scalable solution for initial segmentation.

The Gurobi optimizer proved to be an effective tool for solving the large-scale integer linear programming problems involved in image segmentation. It significantly simplified the process of handling complex constraints and markedly accelerated efficiency.

From the analysis in this thesis, the minimum cut model for image segmentation also revealed several limitations.

First, the computational demands of the models are substantial. Even with the powerful Gurobi optimizer, processing times are generally lengthy, particularly when dealing with high-resolution images and large datasets, posing a significant

limitation for practical applications.

Secondly, these models revealed the difficulty of generalizing to unfamiliar datasets, underscoring the need for more resilient methods to handle diverse data effectively.

Finally, selecting seeds is a critical step in using Min-Cut for image segmentation, but both user-interactive and fully automated methods have drawbacks. While user-interactive methods achieved high accuracy, they are not scalable to large datasets. Fully automated methods, although scalable, do not always reach the precision of user-interactive techniques.

Appendix A

Gurobi Optimization Script

Listing A.1: Python script for performing min-cut optimization using Gurobi for the first mathematical model

```
class Gurobi_minCut():
    def __init__(self, G):
        self.G = G
        self.affinity = nx.get_edge_attributes(self.G,'affinity')
    def run(self,s,t):
        # Decision model and variables
        m = gp.Model("min_cut")
        W = m.addVars(self.G.edges, vtype=GRB.CONTINUOUS,
        lb=0.0, ub=1.0, name="W")
        U = m.addVars(self.G.nodes, vtype=GRB.CONTINUOUS,
        lb=0.0, ub=1.0, name="U")
        # Add constraints
        m.addConstr(U[s] == 1)
        m.addConstr(U[t] == 0)
        m.addConstrs( W[i,j] >= U[i]- U[j] for i,j in self.G.edges)

        # Solve the model
        m.optimize()
        optimal_value = m.objVal
        edges_cut = []
        for i,j in self.G.edges:
            if W[i,j].x > 0:
                edges_cut.append((i,j))
        S = [ i for i in self.G.nodes if U[i].x > 0 ] # s-side of cut

        return edges_cut, S
```

Listing A.2: Python script for performing min-cut optimization using Gurobi for the second mathematical model

```
class Gurobi_minCut2():
    def __init__(self, G):
        self.G = G
        self.affinity = nx.get_edge_attributes(self.G,'affinity')
    def run(self):
        # Decision model and variables
        m = gp.Model("min_cut")
        Y = m.addVars(self.G.edges, vtype=GRB.CONTINUOUS,
            lb=0.0, ub=1.0, name="Y")
        z = m.addVars(self.G.nodes, vtype=GRB.CONTINUOUS,
            lb=0.0, ub=1.0, name="z")
        V_s = m.addVars(self.G.nodes, vtype=GRB.CONTINUOUS,
            lb=0.0, ub=1.0, name="V_s")
        V_t = m.addVars(self.G.nodes, vtype=GRB.CONTINUOUS,
            lb=0.0, ub=1.0, name="V_t")
        # Define the objective function
        m.setObjective(gp.quicksum(self.affinity[i,j] * Y[i,j] for i,
    j in self.G.edges),
        sense=GRB.MINIMIZE)
        # Add constraints
        m.addConstrs( Y[i,j] >= z[i]- z[j] for i,j in self.G.edges)
        m.addConstrs( z[i] >= V_s[i] for i in self.G.nodes)
        m.addConstrs( z[i] <= 1 - V_t[i] for i in self.G.nodes)
        m.addConstr( gp.quicksum(V_s[i] for i in self.G.nodes) == 1)
    m.addConstr( gp.quicksum(V_s[i] for i in self.G.nodes) == 1) m.
    addConstr( gp.quicksum(V_t[i] for i in self.G.nodes) == 1)
        m.addConstrs( V_s[i] + V_t[i] <= 1 for i in self.G.nodes)

        # Solve the model
        m.optimize()
        edges_cut = []
        for i,j in self.G.edges:
            if Y[i,j].x > 0:
                edges_cut.append((i,j))
        S = [ i for i in self.G.nodes if z[i].x > 0 ] # s-side of cut
        nodes_V_s = []
        nodes_V_t = []
        for i in self.G.nodes:
            if V_s[i].x > 0.99:
                nodes_V_s.append(i)
            if V_t[i].x > 0.99:
                nodes_V_t.append(i)

        return edges_cut, S
```

Bibliography

[1] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. «Visual reinforcement learning with imagined goals». In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 1).

[2] Vincent Taschereau-Dumouchel, Matthias Michel, Hakwan Lau, Stefan G Hofmann, and Joseph E LeDoux. «Putting the "mental" back in "mental disorders": a perspective from research on fear and anxiety». In: *Molecular Psychiatry* 27.3 (2022), pp. 1322–1330 (cit. on p. 1).

[3] Ghazanfar Latif, Kévin Bouchard, Julien Maitre, Arnaud Back, and Léo Paul Bédard. «Deep-Learning-Based Automatic Mineral Grain Segmentation and Recognition». In: *Minerals* 12.4 (2022). ISSN: 2075-163X. DOI: 10.3390/min12040455. URL: https://www.mdpi.com/2075-163X/12/4/455 (cit. on p. 1).

[4] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. «Recent progress in semantic image segmentation». In: *Artificial Intelligence Review* 52 (2019), pp. 1089–1106 (cit. on p. 1).

[5] Çağrı Kaymak and Ayşegül Uçar. «A brief survey and an application of semantic image segmentation for autonomous driving». In: *Handbook of Deep Learning Applications* (2019), pp. 161–200 (cit. on p. 1).

[6] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. «Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images». In: *Remote Sensing* 9.4 (2017), p. 368 (cit. on p. 1).

[7] Xinchen Wang, Weiwei Zhang, Xuncheng Wu, Lingyun Xiao, Yubin Qian, and Zhi Fang. «Real-time vehicle type classification with deep convolutional neural networks». In: *Journal of Real-Time Image Processing* 16 (2019), pp. 5–14 (cit. on p. 1).

[8] Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. «Editgan: High-precision semantic image editing». In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 16331–16345 (cit. on p. 1).

[9] Salem Saleh Al-Amri, Namdeo V Kalyankar, et al. «Image segmentation by using threshold techniques». In: *arXiv preprint arXiv:1005.4020* (2010) (cit. on pp. 2, 11).

[10] John Canny. «A computational approach to edge detection». In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on p. 2).

[11] Jun Tang. «A color image segmentation algorithm based on region growing». In: *2010 2nd international conference on computer engineering and technology*. Vol. 6. IEEE. 2010, pp. V6–634 (cit. on p. 2).

[12] Yuri Y Boykov and M-P Jolly. «Interactive graph cuts for optimal boundary & region segmentation of objects in ND images». In: *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 105–112 (cit. on pp. 2, 33).

[13] Jianbo Shi and Jitendra Malik. «Normalized cuts and image segmentation». In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905 (cit. on pp. 2, 6).

[14] James MacQueen et al. «Some methods for classification and analysis of multivariate observations». In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297 (cit. on pp. 2, 8).

[15] Ulrike Von Luxburg. «A tutorial on spectral clustering». In: *Statistics and computing* 17 (2007), pp. 395–416 (cit. on p. 2).

[16] Dorin Comaniciu and Peter Meer. «Mean shift: A robust approach toward feature space analysis». In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619 (cit. on pp. 2, 8).

[17] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. «SLIC superpixels compared to state-of-the-art superpixel methods». In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282 (cit. on pp. 2, 8).

[18] Keiron O'Shea and Ryan Nash. «An introduction to convolutional neural networks». In: *arXiv preprint arXiv:1511.08458* (2015) (cit. on pp. 2, 10).

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. «Fully convolutional networks for semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440 (cit. on pp. 2, 10).

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative adversarial nets». In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 2).

[21] Lester Randolph Ford and Delbert R Fulkerson. «A simple algorithm for finding maximal network flows and an application to the Hitchcock problem». In: *Canadian journal of Mathematics* 9 (1957), pp. 210–218 (cit. on pp. 3, 22).

[22] Boris V Cherkassky and Andrew V Goldberg. «On implementing the push—relabel method for the maximum flow problem». In: *Algorithmica* 19 (1997), pp. 390–410 (cit. on p. 3).

[23] Yuri Boykov and Vladimir Kolmogorov. «An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision». In: *IEEE transactions on pattern analysis and machine intelligence* 26.9 (2004), pp. 1124–1137 (cit. on pp. 6, 30, 34).

[24] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. «" GrabCut" interactive foreground extraction using iterated graph cuts». In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 309–314 (cit. on p. 6).

[25] Pedro F Felzenszwalb and Daniel P Huttenlocher. «Efficient graph-based image segmentation». In: *International journal of computer vision* 59 (2004), pp. 167–181 (cit. on p. 6).

[26] Jos M Bioucas-Dias and Gonalo Valadao. «Phase unwrapping via graph cuts». In: *IEEE Transactions on Image processing* 16.3 (2007), pp. 698–709 (cit. on p. 6).

[27] Akira Suga, Keita Fukuda, Tetsuya Takiguchi, and Yasuo Ariki. «Object recognition and segmentation using SIFT and Graph Cuts». In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4 (cit. on p. 6).

[28] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. «Learning to detect a salient object». In: *IEEE Transactions on Pattern analysis and machine intelligence* 33.2 (2010), pp. 353–367 (cit. on p. 6).

[29] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013 (cit. on p. 8).

[30] Andrew Ng, Michael Jordan, and Yair Weiss. «On spectral clustering: Analysis and an algorithm». In: *Advances in neural information processing systems* 14 (2001) (cit. on p. 8).

[31] Roderick Urquhart. «Graph theoretical clustering based on limited neighbourhood sets». In: *Pattern recognition* 15.3 (1982), pp. 173–187 (cit. on p. 8).

[32] Author Name. «K-Means Clustering: 7 Pros and Cons Uncovered». In: *Data Rundown* (). `https://datarundown.com` (cit. on p. 9).

[33] Ting Su and Jennifer G Dy. «In search of deterministic methods for initializing K-means and Gaussian mixture clustering». In: *Intelligent Data Analysis* 11.4 (2007), pp. 319–338 (cit. on p. 9).

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-net: Convolutional networks for biomedical image segmentation». In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241 (cit. on p. 10).

[35] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. «Segnet: A deep convolutional encoder-decoder architecture for image segmentation». In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495 (cit. on p. 10).

[36] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. «Image segmentation using deep learning: A survey». In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3523–3542 (cit. on p. 11).

[37] K Bhargavi and S Jyothi. «A survey on threshold based segmentation technique in image processing». In: *International Journal of Innovative Research and Development* 3.12 (2014), pp. 234–239 (cit. on p. 11).

[38] Shiuh-Yung Chen, Wei-Chung Lin, and Chin-Tu Chen. «Split-and-merge image segmentation based on localized feature analysis and statistical tests». In: *CVGIP: Graphical Models and Image Processing* 53.5 (1991), pp. 457–475. ISSN: 1049-9652. DOI: `https://doi.org/10.1016/1049-9652(91)90030-N`. URL: `https://www.sciencedirect.com/science/article/pii/104996529190030N` (cit. on p. 11).

[39] Francis HY Chan, Francis K Lam, and Hui Zhu. «Adaptive thresholding by variational method». In: *IEEE Transactions on Image Processing* 7.3 (1998), pp. 468–473 (cit. on p. 11).

[40] Vincent Torre and Tomaso A Poggio. «On edge detection». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1986), pp. 147–163 (cit. on p. 11).

[41] Raman Maini and Himanshu Aggarwal. «Study and comparison of various image edge detection techniques». In: *International journal of image processing (IJIP)* 3.1 (2009), pp. 1–11 (cit. on p. 11).

[42] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. «Design of an image edge detection filter using the Sobel operator». In: *IEEE Journal of solid-state circuits* 23.2 (1988), pp. 358–367 (cit. on p. 11).

[43] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. «Active Shape Models-Their Training and Application». In: *Computer Vision and Image Understanding* 61.1 (1995), pp. 38–59. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.1995.1004. URL: https://www.sciencedirect.com/science/article/pii/S1077314285710041 (cit. on pp. 11, 12).

[44] András Kelemen, Gábor Székely, and Guido Gerig. «Elastic model-based segmentation of 3-D neuroradiological data sets». In: *IEEE Transactions on medical imaging* 18.10 (1999), pp. 828–839 (cit. on p. 12).

[45] Olivier Ecabert et al. «Automatic model-based segmentation of the heart in CT images». In: *IEEE transactions on medical imaging* 27.9 (2008), pp. 1189–1201 (cit. on p. 12).

[46] Xuefeng Song and Ramakant Nevatia. «A model-based vehicle segmentation method for tracking». In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1124–1131 (cit. on p. 12).

[47] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. «Three-dimensional model-based object recognition and segmentation in cluttered scenes». In: *IEEE transactions on pattern analysis and machine intelligence* 28.10 (2006), pp. 1584–1601 (cit. on p. 12).

[48] Theo Gevers and Arnold WM Smeulders. «Color-based object recognition». In: *Pattern recognition* 32.3 (1999), pp. 453–464 (cit. on p. 13).

[49] Rolf G Kuehni. «Color space and its divisions». In: *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 26.3 (2001), pp. 209–222 (cit. on p. 13).

[50] Noor A Ibraheem, Mokhtar M Hasan, Rafiqul Z Khan, and Pramod K Mishra. «Understanding color models: a review». In: *ARPN Journal of science and technology* 2.3 (2012), pp. 265–275 (cit. on pp. 13, 14).

[51] Sabine Süsstrunk, Robert Buckley, and Steve Swen. «Standard RGB color spaces». In: *Color and imaging conference*. Vol. 7. Society of Imaging Science and Technology. 1999, pp. 127–134 (cit. on p. 14).

[52] Commission Internationale de l'Éclairage. *CIE Colorimetry*. Joint ISO/CIE Standard. ISO/CIE. 2007 (cit. on p. 14).

[53] *ISO 11664-4:2008(E)/CIE S 014-4/E:2007: Colorimetry - Part 4: 1976 L*a*b* Colour Space*. ISO/CIE. International Organization for Standardization and Commission Internationale de l'Éclairage, 2008 (cit. on p. 15).

[54] CIE CCUP. *Commission internationale de l'Eclairage proceedings*. 1931 (cit. on p. 15).

[55] Douglas Brent West et al. *Introduction to graph theory*. Vol. 2. Prentice hall Upper Saddle River, 2001 (cit. on p. 18).

[56] Lester Randolph Ford and Delbert R Fulkerson. «Maximal flow through a network». In: *Canadian journal of Mathematics* 8 (1956), pp. 399–404 (cit. on p. 21).

[57] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. «Network flows». In: (1988) (cit. on p. 21).

[58] Chandra Chekuri, Andrew V Goldberg, David R Karger, Matthew S Levine, and Clifford Stein. «Experimental Study of Minimum Cut Algorithms.» In: *SODA*. Vol. 97. 1997, pp. 324–333 (cit. on p. 23).

[59] Eric N Mortensen and William A Barrett. «Interactive segmentation with intelligent scissors». In: *Graphical models and image processing* 60.5 (1998), pp. 349–384 (cit. on p. 34).

[60] Alexandre X Falcao, Jayaram K Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A Lotufo. «User-steered image segmentation paradigms: Live wire and live lane». In: *Graphical models and image processing* 60.4 (1998), pp. 233–260 (cit. on p. 34).

[61] Yuri Boykov and Gareth Funka-Lea. «Graph cuts and efficient ND image segmentation». In: *International journal of computer vision* 70.2 (2006), pp. 109–131 (cit. on p. 36).

[62] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. «Towards internet-scale multi-view stereo». In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 1434–1441 (cit. on p. 36).

[63] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009 (cit. on p. 36).

[64] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. «An efficient k-means clustering algorithm: Analysis and implementation». In: *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 881–892 (cit. on p. 36).

[65] David Arthur, Sergei Vassilvitskii, et al. «k-means++: The advantages of careful seeding». In: *Soda*. Vol. 7. 2007, pp. 1027–1035 (cit. on p. 36).

[66] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html (cit. on pp. 40, 41).

[67] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024. URL: `https://www.gurobi.com/` (cit. on p. 42).

[68] Josef Jablonskỳ et al. «Benchmarks for current linear and mixed integer optimization solvers». In: *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis* 63.6 (2015), pp. 1923–1928 (cit. on p. 43).