Universidade Estadual de Campinas
Instituto de Computação

Jordão Okuma Barbosa Ferraz Bragantini

Interactive Image Segmentation: From Graph-based Algorithms to Feature-Space Annotation

Segmentação Interativa de Imagens: De Algoritmos Baseados em Grafos à Anotação de Características

CAMPINAS
2021

Jordão Okuma Barbosa Ferraz Bragantini

Interactive Image Segmentation: From Graph-based Algorithms to Feature-Space Annotation

Segmentação Interativa de Imagens: De Algoritmos Baseados em Grafos à Anotação de Características

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Alexandre Xavier Falcão

Este exemplar corresponde à versão final da Dissertação defendida por Jordão Okuma Barbosa Ferraz Bragantini e orientada pelo Prof. Dr. Alexandre Xavier Falcão.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

B73i

Bragantini, Jordão, 1996-
Interactive image segmentation : from graph-based to algorithms to feature-space annotation / Jordão Okuma Barbosa Ferraz Bragantini. – Campinas, SP : [s.n.], 2021.

Orientador: Alexandre Xavier Falcão.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Segmentação de imagens. 2. Processamento de imagens. I. Falcão, Alexandre Xavier, 1966-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Segmentação interativa de imagens : de algoritmos baseados em grafos à anotação de características
Palavras-chave em inglês:
Image segmentation
Image processing
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Alexandre Xavier Falcão [Orientador]
Silvio Jamil Ferzoli Guimarães
Moacir Antonelli Ponti
Data de defesa: 06-10-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-7652-2735
- Currículo Lattes do autor: http://lattes.cnpq.br/1633080202914632

Universidade Estadual de Campinas
Instituto de Computação

Jordão Okuma Barbosa Ferraz Bragantini

Interactive Image Segmentation: From Graph-based Algorithms to Feature-Space Annotation

Segmentação Interativa de Imagens: De Algoritmos Baseados em Grafos à Anotação de Características

Banca Examinadora:

- Prof. Dr. Alexandre Xavier Falcão
  Univerisdade Estadual de Campinas
- Prof. Dr. Silvio Jamil Ferzoli Guimarães
  Pontifícia Universidade Católica de Minas Gerais
- Prof. Dr. Moacir Antonelli Ponti
  Universidade de São Paulo

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 06 de outubro de 2021

we humans tend to overestimate AI advances and underestimate the complexity of our own intelligence

(Melanie Mitchell)

Acknowledgements

First and foremost, I would like to thank my advisor Alexandre Falcão, who guided me during my master's degree and accepted me as an undergraduate researcher despite having very little knowledge about research and computer science. I am also highly grateful to Laurent Najman for welcoming me as a research intern and providing invaluable input to my research. Both of these experiences would not have happened if it wasn't for the research grants offered by FAPESP during my undergraduate degree and CAPES during my masters, for which I am immensely grateful.

I am also thankful for the memorable time at the university's "bandejão" and at the Laboratory of Image Data Science shared with my colleagues, Alan Peixinho, Deangeli Neves, João Paulo Penalber, John Vargas, Azael Sousa, Bárbara Benato, César Fernández, Felipe Galvão, Daniel Osaku, Elvis Capia, Felipe Belém, Leonardo Melo, Italos Estilon, David Cardenas, Carolina Cuba, Guilherme Ruppert and especially, Samuel Botter Martins who wholeheartedly assisted me since the beginning of this journey.

I am immensely grateful for my colleagues from the Statistics department and our relaxing conversations during the prolonged hour of studying in the IMECC's basement, my friends with whom I shared a house through my degree, and my childhood friend, Matheus, who made all these years at the university much more enjoyable.

I am thankful for the faculty of the Statistics department, who significantly contributed to my academic development.

I want to thank the Institute of Computing Staff, especially Wilson Bagni who was always available for my endless questions.

Last but not least, I would not have accomplished this without the unconditional love and support from my parents, Karyn and Mauro, the caring discussions with my sister Isadora, and the companionship of my girlfriend, now wife, Barbara. I'm highly thankful for their patience and forgiveness when I neglected them to work or study.

This study was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

Nos últimos anos, algoritmos de aprendizado de máquina se tornaram a abordagem predominante para resolução de diversas tarefas de visão computacional e processamento de imagens, dado um conjunto de exemplos (*i.e.* dados rotulados). O desempenho desses algoritmos está altamente correlacionado com a diversidade de exemplos e qualidade das rótulos, especialmente nos métodos baseados em redes neurais, que requerem uma quantidade significativa de dados.

Notavelmente, a rotulação de segmentos nas imagens requerem um grande esforço para produzir anotações de alta qualidade, devido à unidade de anotação, *pixels*, recorrendo a metodologias interativas para o auxílio do usuário, as quais são o tema central desta tese.

O problema de segmentação de imagem interativa envolve duas tarefas fortemente acopladas: reconhecimento, e delineamento de objetos. Neste contexto, o reconhecimento de objetos se refere à sinalização de uma região de interesse e sua atribuição de um rótulo. O delineamento é responsável por delimitar com precisão o limite (*i.e.* contorno) do objeto. Na vasta literatura de métodos interativos de segmentação de imagem, um usuário humano é empregado inicialmente para reconhecer os objetos de interesse, uma vez que, esta tarefa requer um maior esforço cognitivo. Com esta informação inicial, a máquina realiza a tarefa mais trabalhosa de delinear milhares de pixels.

A partir desta estratégia colaborativa entre homem e máquina para segmentação interativa de imagens, este estudo contribui com os seguintes pontos:

- *Dynamic Trees*: Estendemos o operador Transformada Imagem-Floresta, propondo uma nova metodologia que atualiza os pesos do grafo da imagem conforme a segmentação é calculada, obtendo resultados de alta qualidade na segmentação da imagem a partir de marcadores inseridos pelo usuário.

- *Grabber*: Os métodos interativos existentes baseados em aprendizado profundo aproximam a forma do objeto de interesse com pouca interação do usuário; no entanto, esses não conseguem concluir a segmentação de forma eficaz devido à incapacidade de delinear os detalhes com precisão. O Grabber serve como ferramenta para finalizar o delineamento do contorno a partir de qualquer máscara de segmentação.

- *Anotação de segmentos no espaço das características*: Investigamos outro paradigma de anotação de segmentos, onde a etapa de delineamento ocorre antes do reconhecimento manual — a metodologia proposta apresenta resultados promissores para a segmentação de imagens em maior escala — e abre novos rumos de pesquisa.

Em resumo, investigamos a literatura existente de segmentação interativa de imagens, contribuindo com a introdução de novos algoritmos que realizam a segmentação a partir de marcadores e contornos, e propondo um novo paradigma para a anotação de imagens em larga escala.

Abstract

In recent years, machine learning algorithms that solve problems from a collection of examples (*i.e.* labeled data), have grown to be the predominant approach for solving computer vision and image processing tasks. These algorithms' performance is highly correlated with the abundance of examples and their quality, especially methods based on neural networks, which are significantly data-hungry.

Notably, image segmentation annotation requires extensive effort to produce high-quality labeling due to the fine-scale of the units (*pixels*) and resorts to interactive methodologies to provide user assistance, being the central theme of this thesis.

The interactive image segmentation problem involves two tightly coupled tasks, object recognition and delineation. In our context, object recognition concerns signaling that a region is of interest and assigning a label to it. Delineation is responsible for precisely delimitating the object's boundaries. In the vast literature of interactive image segmentation methods, a human user is employed to recognize the objects of interest first, which requires a more significant cognitive effort. With this initial input, the machine performs the cumbersome task of delineating thousands of pixels.

From this collaborative strategy between human and machine employed for interactive image segmentation, this study contributes the following:

- *Dynamic Trees*: We build upon the existing framework of the Image Foresting Transform, proposing a novel methodology that updates the weights of the image graph as the segmentation is computed, obtaining remarkable results on the image segmentation from user inserted markers.

- *Grabber*: Existing deep learning-based interactive methods approximate the object of interest shape with minor user input; however, they fail to effectively finish the segmentation due to the inability to delineate the boundaries precisely. Grabber, serve as a tool to resume the contour delineation from any segmentation mask.

- *Feature Space Annotation*: We investigate another paradigm of segment annotation, where the delineation step occurs before the manual recognition — the proposed methodology shows promising results for segmenting images at a larger scale — and opens up new research directions.

In summary, we investigate the existing literature on interactive image segmentation, contributing to it by introducing novel algorithms that performs the segmentation from markers and contours, and finally proposing a new paradigm for image annotation at scale.

List of Figures

List of Tables

Glossary

ANN Artificial Neural Network. 17, 27–29, 39

CNN Convolutional Neural Network. 14–17, 19, 27, 29–31, 33–35, 68, 73–76, 89

DT Dynamic Trees. 16, 17, 56–58, 60–62, 64–66

fBRS Feature Back-propagation Refinement Scheme. 9, 34, 55–57, 61, 62, 64–67

feature vector representation of a given sample, for example the pixel color or a value computed from an algorithm. 9, 10, 17, 18, 22, 27, 28, 30–32, 34–36, 45, 47, 48, 57, 60, 68–76, 78–80, 87, 89

GMM Gaussian Mixture Model. 58, 61, 62

gradient In the context of images, referenced as image's gradient or gradient image, it represents a mapping where the regions with transitions of colors (or features) of the original image are highlighted. It is used interchangeably with edge detection throughout the text. Not to be confused with a function's gradient, despite the related origin.. 73, 74

IFT Image Foresting Transform. 16, 17, 19, 23–25, 45–49, 57, 61

IoU Intersection over Union. 62, 64, 65, 79, 80

projection lower-dimensional mapping of a higher dimensional space. 9, 17, 39, 42, 68, 70, 71, 74–80, 87, 88

PW Power Watershed. 17, 19, 21–24, 44, 49

ROI Region of Interest. 33

saliency image, region, or object in highlight according to a criterion. 26, 27, 35, 36, 55, 61, 62, 65, 73, 89

segment Cluster of pixels with similar characteristics, when not indicated it is composed of a single connected component.. 10, 14–16, 18, 19, 26, 27, 31, 34, 35, 56–58, 61, 63, 68–80, 87–89

UMAP Uniform Manifold Approximation and Projection. 41, 42, 74, 79

Contents

Chapter 1

Introduction

Image segmentation concerns splitting an image into segments (*i.e.* regions) with similar characteristics. Its applications range from medical [62] and biological [155] imaging, photo-editing [143], remote-sensing [114] to autonomous vehicles [48, 172]. It is a significantly difficult problem due to the number of units being analyzed (*e.g.* a mobile image contains more than a million pixels), the dependency between neighboring regions, and the ambiguity of precisely defining a segment (*e.g.* a whole tree can be considered a segment or it can be partitioned into branches and leaves). Due to these obstacles, interactive methods are indispensable tools that allows achieving the most accurate results to provide examples for learning-based methodologies or when automatic techniques fail.

Moreover, with the advancement of automatic methods based on Convolutional Neural Network (CNN) [97, 94, 83], the necessity of high-quality annotated data increased drastically, especially in the context of segmentation, which requires significantly more effort than other image-related tasks. For example, the ImageNet [58] dataset for image classification in between 2010 and 2012 reached more than 10 million labeled images, while a much more recent dataset, LVIS [77] from 2019, contains the annotation of 2.2 million high-quality segmentation instances, being a magnitude larger than others segmentation datasets [61, 133], but at the same time a fraction of the older ImageNet dataset. Hence, interactive segmentation techniques are of utmost importance to assist the labeling of segments, providing not only user control, and allowing accurate annotation but also performing the labeling with a low effort so that users can annotate several images quickly.

1.1 Interactive Image Segmentation

Interactive image segmentation techniques are effective because they combine the complementary competencies of humans and machines [65]. Humans can quickly identify objects but are not effective at delineating an object's boundary or at labeling a group of pixels without assistance. In comparison, machines can process a large amount of data in well-defined tasks but lack the cognitive abilities of discerning objects, especially in the combinatorial context of objects in an image, where a pixel can belong to multiple labels depending on the given context. Thus, in this thesis and most of the literature [126, 65, 64, 143, 49, 39, 170, 111, 36], the interactive image segmentation

paradigm employs the user for detecting the object or region of interest and the machine for segmenting (*i.e.* partitioning, grouping, delineating) the images.

Furthermore, CNN methods also contributed to the development of novel techniques for interactive image segmentation [170, 100, 99, 111, 89, 150, 93, 173, 101], significantly reducing the annotation burden, but introducing new problems, as they can greatly approximate the desired object's shape but fail at responding to the user interactions, displaying significant bias towards results seen during training.

In contrast, classical graph-based approaches are very responsive to user input but require extensive user input to perform accurately. And it is an on going challenge the most effective way of combining these methodologies to fully exploit their complementary advantages.

Our contributions in this work are three-fold: first, we improve upon existing graph-based methodologies by introducing a new algorithm that estimates the arc-weights on-the-fly, improving the graphical model as the segmentation progresses; second, we develop a tool for delineation correction agnostic to the label generating procedure, assisting the segmentation of non-responsive algorithms; and finally, we propose a large-scale annotation procedure by questioning the necessity of labeling on the image domain, implementing a proof-of-concept that performs image annotation on an abstract space generated from a set of initial segments, allowing annotation of multiple images at once, and reducing user effort as additional labels are acquired.

1.2 Objectives

Since high-quality annotated data is a huge limiting factor to applying state-of-the-art techniques on problems where labels are scarce, this work's general goal is to improve the annotation of segments on image data by investigating and understanding existing interactive image segmentation methods. Them being, classical techniques that model the images as graphs, and more recent CNN-based methods that solve the segmentation as a pixel classification problem.

From this investigation, it was clear that CNN-based methods were significantly pushing the boundaries of existing interactive image segmentation methodologies and introducing new shortcomings. Hence, one of our goals was to provide a tool with greater user control to assist the correction of segments while being agnostic to the original segmentation procedure, benefiting CNN-based methods, which obtain an excellent approximation of the object while the boundaries might be inaccurate and unable to be fixed.

Moreover, existing techniques only allow annotation of a single object at once, demanding massive effort when considering the dataset creation process as a whole, since it requires annotating thousands of images containing multiple objects each — leading to our research question of "Can images be effectively annotated in a domain other than the image domain?" which strives to develop a novel procedure for large-scale image segmentation annotation.

1.3 Contributions

This thesis's contributions start from graph-based techniques, which were the most popular and successful techniques preceding CNN-based approaches. With the CNN's progress, our contribution shifted to identifying limitations of the annotation on the image domain and proposing a novel methodology for large-scale annotation.

In summary, our contributions are:

- We developed a novel methodology for graph-based interactive image segmentation that dynamically estimates the arc-weight between pixels during the region growing process of the Image Foresting Transform operator, thus providing a more robust estimate less sensitive to noise. This method, called Dynamic Trees, showed promising results, beating existing graph-based algorithms on standard image segmentation baselines.

- To complement CNN-based methods for interactive segmentation, where user control is sacrificed for the networks' predictive power, we developed a new technique, called Grabber, to accurately correct segments without ruining the correct regions, our results showed that state-of-the-art graph and CNN-based methods benefit from using it as a finishing procedure.

- We proposed a novel methodology for annotating image segments that allow labeling multiple images at once on an abstract space representation, speeding up the annotation process when redundant information is present. We implemented a proof-of-concept obtaining competitive results with state-of-the-art methods.

While these contributions were developed independently in a sequence as our study progressed, they complement each other. For example, the methodology for large-scale annotation results in coarse segments that can be corrected with the graph propagation methodology or the contour-based method, Grabber. Furthermore, they go along with the progress in the area, starting with graph-based methodologies with simple learning mechanisms to a more elaborate scheme that leverages the power of deep learning.

The implementation of all the methodologies described in this thesis are available at:

- `https://github.com/PyIFT/pyift` is the library with the algorithm proposed in Chapter 3 and the back-end of 4's methodology.

- `https://github.com/LIDS-UNICAMP/grabber` the napari plugin [152] of the tool presented in Chapter 4.

- `https://github.com/LIDS-UNICAMP/rethinking-interactive-image-segmentation` contains Chapter 5 tool/methodology.

1.4 Structure of the thesis

This thesis is organized such, in **Chapter 2**, we review the necessary background for understanding existing interactive segmentation methodologies while also reviewing the

state-of-the-art techniques. Initially presenting what image segmentation is, then introducing the Power Watershed (PW) framework that unifies several classical graph-based algorithms for image segmentation, next the Image Foresting Transform (IFT) is presented, it intersects with the PW family, and it is the building block for the proposed methodologies on later chapters. Then we briefly review Artificial Neural Networks (ANNs), leading to CNNs and their application to interactive image segmentation and edge detection and the current state-of-the-art for these tasks. Further, we introduce additional techniques required for our large-scale annotation methodology: metric-learning that is applied to improve the data representation as the user interacts, and dimensionality reduction that is used to represent the data from its abstract representation to a two-dimensional embedding for visualization.

From **Chapter 3** forward, the proposed methodologies and their respective results are presented, them being: **Chapter 3**, a methodology for graph-based image segmentation from markers with dynamic arc-weight, Dynamic Trees; **Chapter 4**, a technique for delineation correction agnostic to the initial label generating method, Grabber; **Chapter 5**, a novel scheme for interactive image segmentation annotation that allows multiple images to be annotated once using feature space projection. And finally the conclusion.

Chapter 2

Technical Background

Since we are concerned with the annotation of image segments, we will first describe what image segmentation is, then the classical graph-based methodologies for interactive image segmentation and an unsupervised approach, the hierarchical watershed, are presented. Further, we give a brief overview of neural networks and convolution networks to introduce interactive segmentation and boundary detection methods based on them. Next, we introduce metric learning and dimensionality reduction, which are fundamental to our work of interactive segmentation through feature space annotation. Additionally, works related to this thesis are distributed throughout the text, some being presented in this background chapter and others in the chapters of the methods most related to them.

2.1 Image Segmentation

Image segmentation concerns with the partition of an image into groups of pixels (*i.e.* segments, superpixels) with similar characteristics, in the best scenario, clustering pixels that belong to a single object into unique segment, Figure 2.1.

It is often an intermediate step of a larger task, such as the extraction of brain volumes to assist medical diagnosis [116, 118, 117] or the assignment of labels to segments for scene understanding of an autonomous vehicle driving system [48, 172].

Moreover, this cluster assignment is not straightforward due to the hierarchical structure of images, where a pixel can belong to multiple objects depending on the context (*e.g.* it can belong to a face or the whole person) complicating the annotation and assessment of the segmentation quality.

In the interactive context, a user provides cues to indicate objects of interest, and the algorithm delineates the remaining unlabeled regions. Thus, exploiting the human superior cognitive ability to visually detect objects efficiently (*i.e.* weak detection), while the machines assist the laborious task of assigning a label to each pixel (*i.e.* segmentation, delineation).

The cues to assist the segmentation algorithm may come in several forms, as shown in Figure 2.2. For instance, contour-based methods [65, 126] receive anchor points which secures a pixel as a boundary, and the algorithm tracks the object's contour between them, usually in an orderly fashion. The annotation can be scribbles (*i.e.* markers, seeds) [64, 34,

Original Segmentation

Figure 2.1: Example of image segmentation. Original input image on the left. Segments represented by their color average on the right.

75, 49], such that the segmentation is analogous to the semi-supervised (*i.e.* transductive) machine learning methods, where a set of labeled samples (*i.e.* pixels) propagates their labels to the remaining unlabeled samples. Other methods [143, 158, 111, 173] delineate the segment of interest from a bounding-box, by leveraging the rectangle as a rough estimate of the objects' location and modeling the inside and outside characteristics [143, 158]. More recently, click-based methods [170, 100, 99, 89, 150, 101] were proposed, where the segmentation is obtained from an initial click inside the object (*i.e.* positive click), and additional clicks are supplied until the annotation converges; negative clicks are also allowed to provide background cues.

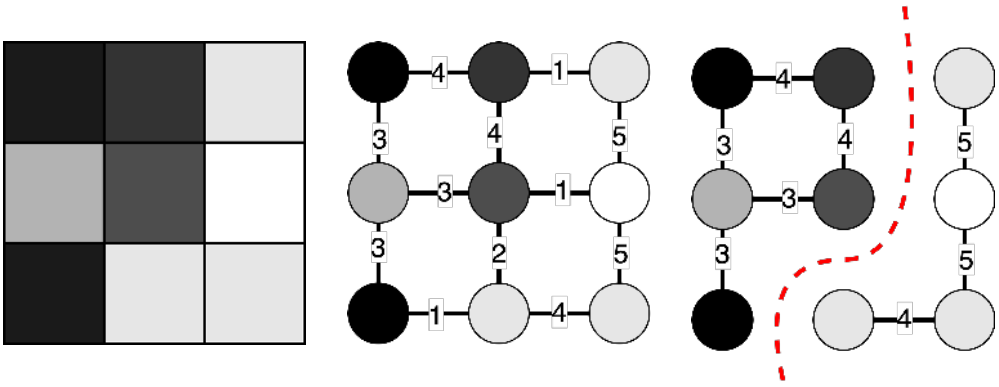
Algorithms that solve the segmentation from these cues can be roughly divided into two categories: Structured prediction, where the assignment of the pixels' labels consider the results of neighboring pixels. For this case, we will focus our discussion on the Power Watershed (PW) [49] family that generalizes several graph-based methodologies and the Image Foresting Transform (IFT) [64] framework that is central to the proposed methodologies of Chapters 3 and 4. And pixel classification, primarily CNN-based methods, where the segmentation problem is treated as a inductive classification task over the image pixels. Note that despite employing different methodologies, both approaches fall on the same paradigm of first obtaining a weak detection of the object through human interaction and later performing the delineation from these cues through their respective algorithm.

It must be noted that despite the two being competing approaches, they also have complementary characteristics, the CNN can efficiently learn patterns from data avoiding the burden of tunning hyper-parameters or pre-processing steps to obtain the optimal performance so that the graph-based techniques can serve as post-processing of the CNN's output, as with conditional random fields for semantic segmentation [40] and watershed for instance segmentation [19, 166, 70].

(a) (b)

(c) (d)

Figure 2.2: Different examples of user cues: (a) Contour annotation; (b) Scribbles; (c) Bounding-boxes; (d) Clicks.



(a) (b) (c)

Figure 2.3: Example of a graph from image; (a) Artificial 3×3 grayscale image; (b) A graph with similarity arc weights; (c) Segmentation into two connected components (*i.e.* graph partition) by removing the edges with minimum sum of weights.

2.2 Power Watershed for Interactive Image Segmentation

To solve segmentation efficiently in a structured manner, most methods model the image as a graph, where each pixel is seen as a node and connected to its adjacent neighbor; the adjacency can be extended to multiple radii on the image grid, but we will consider the 4-neighbor adjacency throughout the text (radius of one). On this graph, a binary segmentation can be interpreted as a classification of the nodes into two classes, 0 and 1, or a cut disconnecting nodes with distinct labels. Figure 2.3 presents an artificial example of a minimum cut on the graph, where the weights between pixels are inversely proportional to their color differences and segmentation is obtained by dividing the graph with the cut of minimum sum of edge weights.

Over the last two decades, multiple different methodologies (and criteria) were proposed to solve the segmentation from this graphical model as an optimization problem.

Classical approaches are: Shortest-paths [64], where the segmentation is obtained from a competition between multiple path sources, and each pixel label assignment is according to the source that provided the path with least cost. Watershed cuts (MST) [51] inspired by the morphological watershed operator [31], but it considers the annotated regions as the watershed basins' minima; Random-walks (RW) [75], where the results are according to the probability of a random process starting from the labeled regions reaching each unlabeled pixel; Graph-cuts (GC) [34] where the graph is partitioned given the minimum-cut criterion, where the sum of the removed edge weights is the smallest.

Notably, all of these methods are unified by the seminal work of Couprie *et al.* [49] called Power Watershed (PW), extending [149, 12]. Further, Ciesielski *et al.* [45, 43] explored their similarity with the Fuzzy Connectedness [159], another segmentation algorithm, significantly popular in the medical imaging community.

These methods represent an image I as a graph $\mathcal{G} = (V, E, w)$, where V are the vertices (*e.g.* image pixels), E represents the edges which connect two adjacent nodes from V (*e.g.* pixels adjacent on the image domain), and w is an arc-weight function,

which maps the edges into a non-negative real value, assigning larger values to nodes that should remain connected. The segmentation on the PW framework resumes to a partition of the graph into two different label assignments (*i.e.* two disconnected components, a cut) by minimizing the objective function

$$J(\mathbf{x} \mid \mathbf{y}) = \underbrace{\alpha \sum_{(i,j)\in E} w(i,j)^p |x_i - x_j|^q}_{\text{pairwise term}} + \underbrace{\sum_{v\in V} u(x_v \mid y_v)^p |x_v - y_v|^q}_{\text{unary term}} \tag{2.1}$$

where the array $\mathbf{x}$ is our variable of interest that assigns a value between 0 and 1 to each node in V, obtaining the segmentation given a decision criterium, for example, node i belongs to label 1 if $x_i > 0.5$ and 0 otherwise; $\mathbf{y}$ represents an binary array of the expected values for each node, it is obtained from user annotation or some previous knowledge about the task, providing additional information about the object location.

The function $u(x_i \mid y_i) : [0,1] \times \{0,1\} \mapsto \{\mathbb{R}_{\geq 0}, \infty\}$, assigns a non-negative value proportional to the likelihood of that given node i belonging to its target label y_i and α controls the trade-off between the pairwise (*i.e.* neighboring agreement) and the unary term (*i.e.* prior fidelity). Therefore, if α is 0, the pairwise term is ignored and the optimal $\mathbf{x}$ is according to a threshold over $u(.)$, p and q are parameters that generalizes the PW to the classical methods and they will be discussed ahead.

The similarity weight function is usually defined by

$$w(i,j)^p = \exp(-\|I(i) - I(j)\|)^p = \exp(-p\|I(i) - I(j)\|) \tag{2.2}$$

where $I(v)$ represents the color space or any other feature representation of a pixel v in V. Such that, as p increases the $w(i,j)^p$ tends to zero quicker, leading to a sharper bell curve resulting from the exponential function, increasing the influence of pixels with greater differences — less smooth segmentation boundaries.

For binary segmentation with hard constraints, as the usual case in the interactive settings where the annotated regions are fixed to its given label, the optimization can be simplified by setting the unary terms to $u(x_i \mid y_i = 1) = \infty$ for every pixel i annotated as foreground and $u(x_i \mid y_i = 0) = \infty$ when labeled as background and zero otherwise, including the remaining unlabeled pixels. Enforcing that labeled regions to never disconnect from theirs infinity linked source, leading a simplified optimization problem of

$$\arg\min_{\mathbf{x}} J(\mathbf{x} \mid \mathbf{y}) = \sum_{(i,j)\in E} w(i,j)^p |x_i - x_j|^q \tag{2.3}$$

$$\text{subject to } x_v = 1 \quad \forall\ v \in S_F \tag{2.4}$$

$$x_v = 0 \quad \forall\ v \in S_B \tag{2.5}$$

where the S_F is the subset of V containing the pixels assigned to the foreground by the user and S_B to the background.

From this simplified model the PW generalizes to the other methods by varying the p and q parameters (Table 2.1), providing a greater understanding of the nature of the classical graph-based segmentation algorithms. However, despite all of these methods be-

q \ p	0	finite	∞
1	Collapsed Seeds	Graph Cut	Power Watershed q = 1
2	l2-norm Voronoi	Random Walks	Power Watershed q = 2
∞	l1-norm Voronoi	l1-norm Voronoi	Shortest paths

Table 2.1: Equivalency between different p, q parametrizations and other algorithms.

longing to this unifying framework, their optimizations are most efficient when employing their respective algorithms.

For a binary solution of $\mathbf{x}$ with $q = 1$ and p finite, $|x_i - x_j|$ is equivalent to a indicator function, that is 1 when x_i is different from x_j and 0 otherwise. Hence, the objective function is the sum of the edge weights of pixels with distinct labels, being equivalent to the minimum cut problem, also known as Graph-Cut [35]) in the computer vision community, and can be efficiently computed on sparse image graphs using the Maxflow algorithm proposed by Boykov and Jolly [34] or the Incremental Breadth-First Search algorithm [71], we refer to [161] for a detailed comparison of different maxflow algorithms' computational performance for image segmentation.

As shown by [149], when $q = 2$ with p finite, the optimization is equivalent to the Random-walk (RW) [75] objective's function, where the objective function is a quadratic equation and is solved as a sparse linear system of equations. Further, it obtains a fuzzy label assignment for each unlabeled pixel. Recently, Sanmartín *et al.* [68] proved that the RW is the expected label assignment of sampling Watershed Cuts of the image graph from a Gibbs distribution of the spanning trees' cost, further extending the theory between the relationships of these algorithms.

As the parameters p and q tends to infinity and taking into account that $\sqrt[p]{\cdot}$ is a monotonic function, the simplified objective function (Equation 2.5) tends to the maximum norm [149]

$$\begin{aligned} \arg\min_{\mathbf{x}} J(\mathbf{x} \mid \mathbf{y}) &= \lim_{p,q\to\infty} \sum_{(i,j)\in E} w(i,j)^p |x_i - x_j|^q \\ &= \max_{(i,j)\in E} w(i,j)|x_i - x_j| \end{aligned} \tag{2.6}$$

Thus, this parametrization can be solved by computing the maximum spanning tree (MST) [50] and removing the edge with minimum weight subject to the constraint that the seed nodes with distinct labels belong to the same connected component in $\mathcal{G}$. In other words, splitting the seeds by the minimum edge from the maximum spanning tree.

Additionally, it can be computed using the shortest path algorithm by accumulating the costs with maximum operator [64]. While it does not obtain an MST [51], its labeling is the same.

The shortest paths methodology falls under IFT's [64] framework, which is fundamental to two of our contributions (Chapters 3, 4) and its presented on its own Section 2.3.

Unfortunately, the maximum-norm objective function's solution might not be unique. The PW solves this issue by setting q to a finite value and proposing a different algorithm

that executes a MST segmentation but solves the RW's linear system on regions with tied edge weights (non-unique solutions). Thus, being a intermediate solution between the shortest paths and the RW.

2.3 Image Foresting Transform

As mentioned earlier, when p and q tends to infinity the PW objective function (Equation 2.5) can be solved through the IFT framework, which supports several applications beyond image segmentation, such as distances transforms, multi-scale skeletons [63], boundary-tracking [65] and pattern recognition [131, 140]. Note that, just a subset of the IFT operators falls under the PW framework, several other connectivity functions have their own category [124, 56, 113, 55] regarding optimum path forests.

To be more convenient with the following chapters, from now on the graph's weights measure a dissimilarity between pixels, $w(i,j) = \|I(i) - I(j)\|$. Thus, solutions with larger weighted edges should be penalized, for example, the solution for the Watershed Cut would be based on a minimum spanning tree, or optimum path forest as it will be described here, as opposed to maximum spanning tree from Equation 2.6.

Additional notations extending the previously shown graph formulation are necessary for the IFT algorithm: $\pi_{p_n} = \langle \pi_{p_{n-1}}, p_n \rangle = \cdots = \langle p_0, p_1, \cdots, p_n \rangle$ is a sequence of nodes (*i.e.* path) with end in p_n, rooted in p_0, such that $(p_i, p_{i+1}) \in E$, it can also be a trivial path with a single node, $\pi_q = \langle q \rangle$, Π_q represents the set of all paths with terminus q in $\mathcal{G}$ and Π the set of all paths; a connectivity function $f : \Pi \mapsto \{R_{\geq 0}, \infty\}$ defines the accumulative weights along the paths; the path-cost mapping $C : V \mapsto \{R_{\geq 0}, \infty\}$ records the current optimal connectivity value for each node; the mapping $P : V \mapsto \{V, nil\}$ indicates a node's predecessor in the path sequence, or nil if it is starting node (*i.e.* root) — the optimum path forest can be recovered by following the predecessors path for every node in V and the resulting labeling map $L : V \mapsto \{0, 1\}$ for the binary case.

The connectivity functions f may be defined in different ways beyond the PW framework and even violating the optimum cost mapping conditions described in [44] while still producing effective object delineation [124, 56, 113, 55]. The most popular connectivity function is

$$f_{\max}(\langle q \rangle) = \begin{cases} 0 & \text{if } q \in S_F \cup S_B \\ \infty & \text{otherwise} \end{cases}$$
$$f_{\max}(\langle \pi_p, q \rangle) = \max\{f_{\max}(\pi_p), w(p,q)\} \tag{2.7}$$

Hence, as the paths extend their connectivity (*i.e.* path cost) value is the bottleneck (maximum) accumulative cost along its route.

Moreover, at the IFT initialization, every node belongs to a trivial path, and from $f_{\max}$ their path-costs are initialized with 0 if they belong to seed and otherwise infinity. Thus, their path will be replaced as the IFT algorithm progresses, minimizing the total path cost

Algorithm 1 IMAGE FORESTING TRANSFORM

INPUT: Graph $\mathcal{G} = (V, E, w)$, seeds set S_F and S_B
OUTPUT: Cost map C, Predecessor map P and label map L
AUXILIARY: Priority queue $Q = \emptyset$, temporary variable *pathcost*

```
1.  For each p ∈ V
2.      C(p) ← ∞
3.      P(p) ← nil
4.      If p ∈ S_F ∪ S_B
5.          C(p) ← 0
6.          If p ∈ S_F
7.              L(p) ← 1
8.          Else
9.              L(p) ← 0
10.     Insert p in Q
11. While Q ≠ ∅
12.     Remove p from Q, such that p = arg min_{∀q∈Q}{C(q)}
13.     For each q ∈ V | (p, q) ∈ E, q ∈ Q
14.         pathcost ← max{C(p), w(p, q)}
15.         If pathcost < C(q)
16.             C(q) ← pathcost
17.             P(q) ← p
18.             L(q) ← L(p)
```

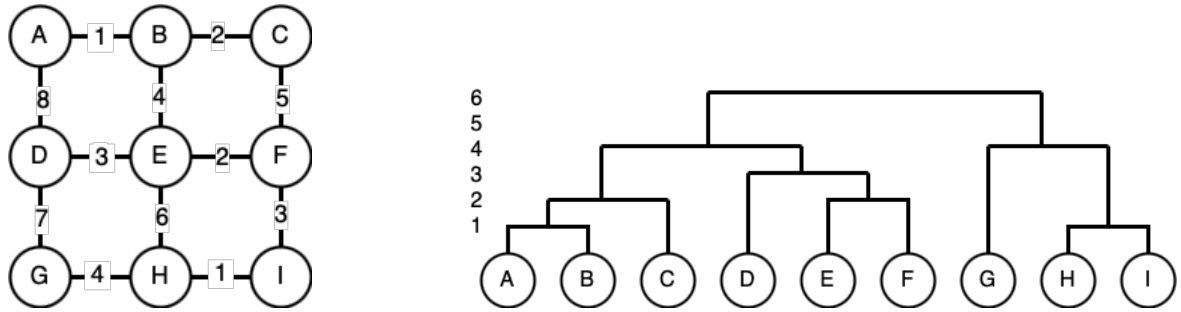
$$C_{total} = \sum_{q \in V} C(q) = \sum_{q \in V} \min_{\pi_q \in \Pi_q} f(\pi_q) \tag{2.8}$$

Algorithm 1 presents the IFT algorithm for image segmentation that minimizes the Equation 2.8; the lines from 1 to 10 are the algorithm initialization with the trivial path cost function, and the provided seed sets S_F and S_B; Until every node is processed (line 11), the node p with minimum path cost is selected (line 12) to extend its path to its neighbors q, if it offers a lower path cost than the current path cost of q (line 13-15) this path is extended, updating q's path cost, predecessor map, and labeling. Hence, at each iteration the optimum path forest propagates its labels to neighbouring nodes with sub-optimal path costs, returning the final labeling map L and the other additional mappings when finished.

2.4 Hierarchical Watershed with Attributes

In the unsupervised scenario, the hierarchical watershed [32, 129, 121] defines an operator that constructs a hierarchy from a non-empty set, in our case, the pixels of an image without any user input.

Using the previously introduced definition of an image I represented as a weighted graph $\mathcal{G} = (V, E, w)$ and defining the hierarchy as in [129] — a chain of partitions (*i.e.* clustering of V), $\mathcal{H} = \{\mathbf{P}_0, \mathbf{P}_1, \cdots, \mathbf{P}_l\}$, where each grouping in $\mathbf{P}_i$ is a superset of the groups in $\mathbf{P}_{i-1}$. Moreover, in $\mathbf{P}_0$ each element in V is its own group, and $\mathbf{P}_l$ contains



(a) (b)

Figure 2.4: (a) Planar graph; (b) dendrogram of hierarchy build by non-decreasing ordering of edges.

a single group containing all elements of V. This can be interpreted as a tree (*i.e.* dendrogram) where $\mathbf{P}_0$ are the leaves and $\mathbf{P}_l$ the root, Figure 2.4.

The original watershed transform operator [31], can be described following a topographic perspective, where a grayscale image (or its gradient) is viewed as a topographic map where the pixels' brightness represents different heights over the image grid, and segments are assigned through a flood filling procedure starting from each local minimum, assigning a single distinct label minimum (*i.e.* watershed basin), the contour of these segments are the watershed lines of the image topography. During the flood filling, at each iteration the pixels at the water level are merged into their local minimum, and this procedure stops after every pixel has been processed.

The watershed hierarchy extends this concept by assigning each pixel to multiple partitions according to the distinct sets obtained during the increase of the water level (*i.e.* flood filling). Therefore, the hierarchy is composed of the partitions resulted from merging neighboring regions when the water level rises above the watershed line separating them after the flood filling iterations.

The framework proposed in [53] formalizes this concept of region merging to any edge-weighted graph, and its dual representation as a saliency map, analogous to the ultra-metric contours (UCM) [129] of a hierarchy on an image grid. Where the saliency map assigns a value to each edge in $\mathcal{G}$ proportional to the level where the regions linked by it became connected in $\mathcal{H}$. Hence, regions connected at the lower level of the hierarchy will have a smaller saliency and regions becomes connected closer to the root have a greater saliency value, Figure 2.5 presents an example, the image graph edges saliency map are embedded into a Khalimsky [129, 18] for visualization. A hierarchy results in a segmentation by selecting saliency value to split the regions, it can be seen as an horizontal cut in the hierarchy dendrogram.

Moreover, while the watershed hierarchy was initially proposed to operate on an sequence of non-decreasing edge weights. It also allows the processing of others attributes and manipulation of the hierarchy tree structure with filtering operations.

For instance, hierarchy can be constructed with an area attribute, where the regions are the result of multiple area closing operations [162] over graph edges with increasing radius, as described in [53]. From a single-linkage clustering (*i.e.* minimum-spanning tree) of the graph $\mathcal{G}$, the procedure of *uprooting* sequentially accumulates (from the tree

(a) (b) (c)

Figure 2.5: (a) Original image; (b) its edge-estimation using [105]; (c) saliency map from its watershed hierarchy by dynamics, resized with the maximum over a neighborhood to improve visualization of thin lines. Greater brightness (*i.e.* saliency) means that the regions were merged closer to the root of the hierarchy.

leaves to its root) the area of the hierarchical partitions as they merge, resulting in a new hierarchy with a new ordering according to the nodes' area, Figure 2.6c. Note that, this is also a non-decreasing attribute, since the area will never decrease.

Notably, the standard watershed hierarchy from 2.6a is strikingly similar to the original image's edge-estimation, this is due to the duality of a fuzzy contour and the hierarchical representation of its segments.

Additionally, Perret *et al.* [135] proposed an efficient algorithm to filter nodes of an hierarchy. This tackles the problem of having a hierarchy where undesirable partitions are high up the tree and a cut would result in undesired segments. By filtering the hierarchy with a different attribute from the original construction, they can merge regions independently of their saliency, Figure 2.6. Barcelos *et al.* [21] also presents some additional studies of simplification (filtering) of non-relevant regions.

2.5 Convolutional Neural Networks

Convolutional Neural Network (CNN) extends Artificial Neural Networks (ANNs) [96, 73] to the domain of images [97, 94, 83, 107], videos [90], time-series [95], graphs [92], and other kinds of unstructured data, where using the standard perceptron is not effective or impractical. Hence, in this section, we review ANNs, their relation to CNNs, and how CNNs can be applied for image classification and image segmentation problems.

They are of utmost relevance to this work because CNNs have shown outstanding performance for interactive segmentation, and they can extract features from images exceptionally well, being a key technology in our features space annotation methodology (Chapter 5). Moreover, the state-of-the-art methods for interactive segmentation are based on them.

Despite its original bio-inspired origins with the perceptron model [142] inspired by

(a) (b) (c)

Figure 2.6: (a) Standard watershed hierarchy; (b) result from (a) filtered by area; (c) watershed hierarchy by area. The filtered hierarchy (b) had its irrelevant segments merged approaching the results from (c). Saliencies were resized with the maximum over a neighborhood to improve visualization of thin lines.

the brain's neural network, modeled by a single neural network's layer. The ANN's computational and empirical aspects have advanced much faster in the recent decade than its biological and theoretical counterparts. Hence, in this section, we follow a more geometrical interpretation of the network's behavior, where each neuron is a half-plane (subject to the activation function) dissecting the data's feature space given the inner product between the weights (*i.e.* plane normal vector) and its input.

An ANN comprises multiple densely connected layers, where each layer has weights and biases (affine transformation) and an activation function (non-linear operation) [148], the combination of the weights and activation function is also known as neurons. Thus, a single fully connected layer expecting input of size d_i and output of size d_{i+1} is given by

$$f_i(\mathbf{x}|W_i, \mathbf{b}_i) = \phi(\mathbf{x}W_i + \mathbf{b}_i) \tag{2.9}$$

where $\mathbf{x} \in \mathbb{R}^{d_i}$ is the input, $W_i \in \mathbb{R}^{d_i \times d_{i+1}}$ and $\mathbf{b}_i \in \mathbb{R}^{d_{i+1}}$ are the network weights, $\phi(.)$ is the activation function, that can be defined from a myriad of options, but lets consider the Rectified Linear Unit (ReLU), where

$$\phi(\mathbf{x}) = \max\{0, \mathbf{x}\} = [\mathbf{x}]_{\geq 0} \tag{2.10}$$

Hence, each array $w_j \in W_i$ represents a hyperplane's normal vector, the w_j and $\mathbf{x}$ dot products are computed by the matrix multiplication $\mathbf{x}W_i$, resulting in positive values if they point to the same direction as its respective w_j and negative values otherwise. When employing the ReLU activation function, negative values are zeroed out, forming polytopes (*i.e.* n-dimensional polyhedron) containing the data.

By stacking n fully connected layer with an input of size d_0, output of size d_n and

widths $d_1, \cdots, d_{n-1}$, an ANN f can be defined as

$$f(\mathbf{x}|\theta) = f_{n-1}(f_{n-2}(\cdots f_0(\mathbf{x}))) \tag{2.11}$$

where θ is the set of all parameters $W_0, b_0, W_1, b_1, \cdots W_{n-1}, b_{n-1}$ parameters (omitted on the right hand side for clarity). The parameters θ and the layers ordering is usually called the network architecture.

Applying this methodology without changes to images, treating pixels individually, is computationally impractical. For example, to process a single image with size of 400×400 (much smaller than a picture from an average mobile device) with 3 color channels represented as $\mathbf{x} \in \mathbb{R}^{d=400\times400\times3}$, and a single layer with a squared weighted matrix $W \in \mathbb{R}^{d\times d}$, would require more than a terabyte of memory.

The convolutional networks inspired by filtering operations from classical image processing reduce the amounts of weights by sharing their values at different locations of the image at each layer, in a process that is analogous to the cross-correlation operation. Such that we formalize a convolutional layer as

$$f_i(\mathbf{x}|W_i, \mathbf{b}_i) = \phi(\mathbf{x} \circ W + \mathbf{b}) \tag{2.12}$$

where $\mathbf{x} \in \mathbb{R}^{H\times W\times d_i}$ is the layer input, $W_i \in \mathbb{R}^{d_i\times k\times k\times d_{i+1}}$ and $\mathbf{b}_i \in \mathbb{R}^{d_{i+1}}$, are the kernel weights and bias. Such that, the cross-correlation is applied d_{i+1} times, each time with a different weights, on the d_i-multi-band image $\mathbf{x}$ with a $k \times k$ window.

The resulting data height (H) and width (W) is subject to several factors, the kernel size k, spacing between sparse weights (*i.e.* dilation), the time steps between operations (*i.e.* stride) and boundary conditions (*i.e.* padding). We refer to [3] to review the resulting data dimensions computation.

These CNN networks architecture can be implemented by naively stacking the previously mentioned convolutional layers, but employing a more elaborate organization of layers (*i.e.* convolutional blocks), can significantly improve their accuracy and computational performance [82, 83, 163, 138].

Notably, the Residual Block [83] is defined as

$$\begin{aligned} f_{res}(\mathbf{x}) &= \phi[\phi(\mathbf{x} \circ W_{2i} + \mathbf{b}_{2i}) \circ W_{2i+1} + \mathbf{b}_{2i+1} + \mathbf{x}] \\ &= \phi[f_{2i}(\mathbf{x}) \circ W_{2i+1} + \mathbf{b}_{2i+1} + \mathbf{x}) \end{aligned} \tag{2.13}$$

such that, f_{res} has two convolutional layers, the second layer sums the original input before its activation function, thus, never losing the input signal. This improves the computational performance by reducing the intermediate image depth with the W_{2i} weights and recovering its dimensionality with W_{2i+1}, saving computational resources. Moreover, by adding the original input it avoids the problem of vanishing gradient, benefiting the backpropagation optimization [97], and allowing training of much deeper networks [83].

For different image related tasks the CNN's architectures are adapted by swapping the final layer according to the expected result (*e.g.* categories of images [83, 157, 138], categories of pixels [141, 163], regression of bounding boxes coordinates [139]).

To classify images into $\mathcal{C}$ discrete categories (*e.g.*, $\{cats, dogs, birds\}$), the CNN archi-

tecture wishes to extract a representation of the data (*i.e.* feature space representation) where only the samples of the same classes are clustered together, we will call this feature extractor network f_{cnn}. From this representation, fully connected layers with softmax activation function (*i.e.* decision layer) assigns a label according to the strongest activated classes, this will be called f_{fully}. Thus, we defined this network as

$$f_{class}(\mathbf{x} \mid \theta) = f_{fully}(\psi(f_{cnn}(\mathbf{x} \mid \theta_{cnn}) \mid \theta_{fully})) \tag{2.14}$$

where $f_{cnn} : \mathbb{R}^{H \times W \times 3} \mapsto R^{h \times w \times d}$, such that $h \ll H$, $w \ll W$ and $d \gg 3$, a low-resolution high-dimensional representation; ψ is a flattening function, usually being the standard flattening which collapses the 3-dimensional image structure into a 1-dimensional array with the same number of elements, or an global average pooling which averages along the $h \times w$ dimensions into a d-dimensional array; and the fully connected network is defined by $f_{fully} : \mathbb{R}^d \mapsto [0, 1]^{|\mathcal{C}|}$.

With CNN's, the image segmentation is also treated as a classification problem [141, 107], but it predicts a value for each pixel, resulting in an $[0, 1]^{H \times W \times |\mathcal{C}|}$ output array. Moreover, the segmentation architectures avoid reducing the input dimensions into a flat array and employs a decoder-encoder regime, where the input image is mapped into a lower resolution image with a greater number of channels (*i.e.* encoder network) and then reconstructed back into the original resolution (*e.g.* decoder) for the pixel-wise prediction with a shared weights classifier (*i.e.* convolutional layer with 1×1 window). Thus, it does not have fully connected layers and it can process images from any size, this was originally proposed as Fully Convolutional Networks (FCN) [107], and can be defined as

$$f_{segm}(\mathbf{x} \mid \theta) = f_{decoder}(f_{encoder}(\mathbf{x} \mid \theta_{encoder}) \mid \theta_{decoder}) \tag{2.15}$$

where the encoder network, $f_{encoder}$, could be equal to f_{cnn} from Equation 2.14. The reduction of the input image resolution provides a larger receptive field; that is, pixels that were further away on the original image domain can be evaluated with a small filter as they get closer (*i.e.* smaller image), achieving greater semantic representation of the data without diminishing the computational performance with a large kernel. The downside is that the reconstruction results have greater artifacts as small details are lost [163].

The decoder network, $f_{decoder} : R^{h \times w \times d} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}|}$, reconstructs the original image resolution through transposed convolutions, or upsampling layers [107]. The last layer applies a 1×1 convolution followed by a softmax activation, as a linear classifier, in a similar fashion with f_{class}'s decision layer, assigning each pixel the label with the strongest activation.

2.5.1 CNNs with User Input

For interactive image segmentation, CNN's architectures for segmentation are adapted to include user input and use it as cues to assist the object's segmentation, originally presented by Xu *et al.* [170] in 2016 with Deep Object Selection (DOS), it proposed the use of an RGB image as input with two additional channels of a foreground and a background distance maps from the user-defined clicks; the network outputs a probability

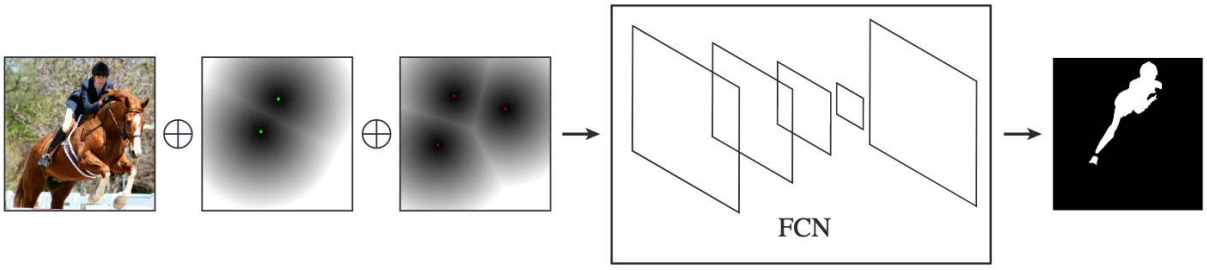


Figure 2.7: Example of RGB input with foreground and background distance maps from green (foreground) and red (background) clicks resulting segmentation. Image from [170].

map for estimating the unary terms to perform graph-based segmentation with the GraphCut algorithm, Figure 2.7.

The distance maps are the inverse of the distance transform from the user-inserted points. Such that, these points have the greatest values and it diminishes as it goes further away from its origins. Usually, the distances are normalized within a range and they might be transformed to have a gaussian shape [111], although not that common.

While the literature heavily employs distance maps [170, 100, 111, 89, 150, 93, 101, 173], its crucial role of adding spatial information to a spatially invariant method (*e.g.* CNNs) is never deeply discussed, to our knowledge.

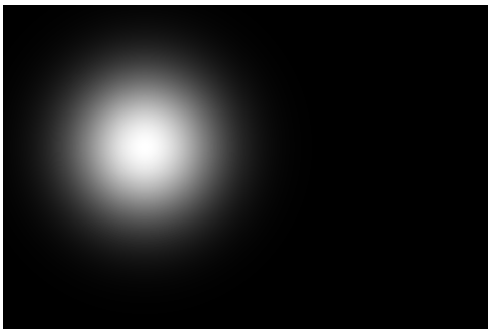
Figure 2.8 shows an example, suppose the user wants to segment only the flower on the left. However, both of the flowers have similar characteristics, and our feature extractor is spatially invariant, so the extracted features will maintain this behavior. Hence, to discriminate between them would be extremely difficult, if not impossible. By adding a click on the left flower center and appending an additional axis for the distance map from this point, their representation is no longer similar, assisting the classifier discrimination. Moreover, a network can learn upon these additional features, obtaining better results than just from an additional dimension on the input data.

To train the network to perform the segmentation from the inputs (*i.e.* clicks), multiple interactions are simulated during training, most of the time in a single shot by sampling a random number of clicks; where the positive clicks are positioned randomly inside the object with a minimum distance between each other and the object's boundary, and the negatives follow a similar strategy but on the objects' outside. Most of the training strategies [170, 100, 99, 89, 150, 101] are a variation of procedure. Mahadevan *et al.* [109] proposed an iterative procedure that inserts clicks on the largest connected component of the segmentation error, reproducing an user behaviour more closely, yielding improvements to the network performance but at the cost of slower training, since multiple iterations are required per image.

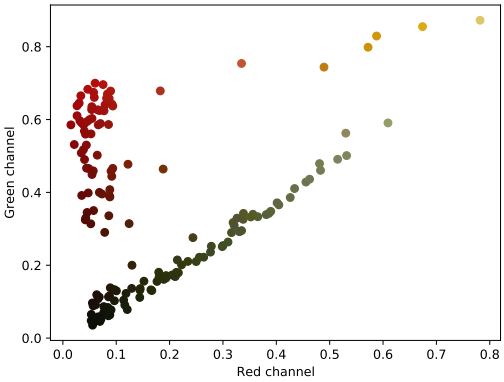
Seeking to improve upon DOS and exploiting the combinatorial nature of objects in an image, Li *et al.* [99] proposed to obtain the desired segment in a two-fold; first, a CNN estimates multiples candidates segments, learning ambiguity from the user input (*e.g.*, a click in a person's legs could signal the leg or the whole person) and a second network to select a single mask from the set of candidates.

Regional Interactive Segmentation Network (RIS-Net) [100] improves the model responsiveness to user input by performing a focused segmentation from candidate regions.

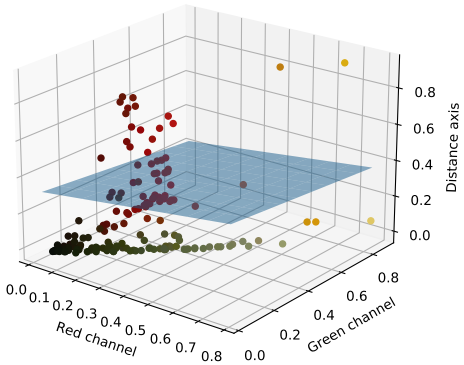
(a)



(b)



(c)



(d)

Figure 2.8: Effect of distance map when classifying a single flower: (a) Original image; (b) Distance map centered at left flower (object of interest); (c) Red and green color space of a sub sample of pixels; (d) Red, green and distance feature space of the same pixels with plane separating the flowers, this was impossible without the distance map axis.

As it will be presented further ahead, cropping and resizing plays a significant role in improving CNN's performance, since they are not resolution invariant. As DOS, RIS-Net obtains a foreground probability map from the 5 channels input. Additionally, it samples bounding boxes around regions of interest (ROI) containing negative and a positive click, these regions are resized and processed through the network independently, obtaining a finer segmentation where it requires the most — on regions between competing clicks, where the user made corrections. At the end of the network, these ROIs and the global segmentation are processed together with a pooling layer to obtain the final segmentation.

Furthermore, other forms of user input have been explored. Deep Extreme Cut (DEXTR) [111] starts the segmentation from four user inserted extreme points (left, right, top and bottom most of the object), from these points, they generate a single distance map, so the network processes a 4 channel input (*i.e.* RGB plus distance map). Since the object is constrained to be inside these points, they crop the image with a tight rectangle containing them and rescale it to a fixed size input; on their ablation study, this is shown to improve their score by 7.9 %. Overall, their performance is truly outstanding; however, applying annotation only on extreme points restricts the user interaction since negative input are not allowed.

Zhang *et al.* [173] improved upon this by using two or more negative clicks to extract a bounding box around the object and crop the image with it, and by allowing positive clicks to indicate the foreground with its respective distance map. Further, they perceived that the model errors are near the object's boundary and do not improve as the networks get deeper, and these errors were mostly results of artifacts from the low-resolution to original resolution reconstruction. To address this, the final prediction layer employs Pyramid Scene Parsing [174] module from multiple resolutions of the network's features along with the image reconstruction step from the decoding blocks.

The Poly-RNN [39], Poly-RNN++ [8], and Curve-GCN [102] obtain an initial segmentation from a bounding box around an object and employ user interaction on anchors points from the segmentation contour, the network regress the control-points coordinates as the user interacts with it, providing better user supervision to the object delineation.

Lin *et al.* [101] obtains great improvement over existing methodologies by applying the simple strategy of treating the first click separately from the rest, their motivation was that the first click is usually centered on the object and it is always a positive click. Thus, treating it differently from other clicks provides additional information. Their network has two parallel branches, one for the segmentation with only the first click input and another which receives the remaining input; the second branch combines the representation extracted from both and predicts a single segmentation output. The loss function penalizes the segmentation from the single click and the combined prediction separately. Interestingly, this loss tackles the problem of biasing the network's prediction to some amount of clicks, where if provided with a large number of simulated input during training, the network is not able to obtain a segmentation with just a few clicks, and if trained with few interactions, its performance is better in a scenario with reduced interaction, but the segmentation does not converge to the desired results when additional interactions are required.

More recently, the techniques have shifted from changing how the user interacts with

the network or the network's architecture to how it can update during inference given the user interaction to adapt to its needs. Backpropagating Refinement Scheme (BRS) [89], recognized that the input distance maps are far from optimal since the distance transform can degrade the feature representation when not centered or in small/thin regions. Thus, they formulated an optimization problem to minimize the disagreement between the annotated pixels (*i.e.*, clicks) and the prediction by adjusting the foreground and background distance maps during inference. This optimization is done by computing the distance maps derivative over the whole network and using a quasi-newton optimizer, L-BFGS [104].

Feature Back-propagation Refinement Scheme (fBRS) [150] reduces BRS's processing time by avoiding computing the derivative over the whole network, optimizing only an affine transformation of features from a pre-determined intermediate layer. Multiple layers' positions were evaluated, the computational performance is best when the layer is closer to the network output (fewer derivatives are necessary) and with a smaller number of features (fewer variables subject to optimization). Applying on the middle of the decoder network obtained the best trade-off between accuracy and speed.

Kontogianni *et al.* [93] went beyond optimizing the weights of the image being currently annotated, proposing an optimization strategy that carries on the information learned from the annotation of novel instances to subsequent objects. Thus, the system not only learn during user interaction but also retains the information and reduce the annotation burden of the following segments. During inference, they optimize the binary cross-entropy between annotated pixels and the network output plus a penalization term with the Memory Aware Synapses [11], that weighs the changes of the network's parameters according to their importance to the training data's output (*i.e.* greater gradients). Thus, updating the most only the least important parameters found during training, diminishing the degradation of the segmentation performance of seen objects.

Concurrent to the development of this thesis, Sofiiuk *et al.* [151] surpassed the state-of-the-art with a feed forward CNN architecture with a carefully engineered architecture without updates during inference. This new architecture uses HRNet as the backbone [163], binary disks with a radius of 5 pixels in place of the of distance maps and have an additional mask with the network's previous prediction (empty if it is the first click) as input. Moreover, these three additional inputs (two distance maps plus previous prediction) are processed separately by two different layers and summed together. Furthermore, they show that poorly labeled data diminishes the model performance, and when employing a new available dataset, LVIS [77], their performance are even better.

2.5.2 Edge Detection

In this section, we review the recent literature of edge detection with CNNs. Like segmentation, it is usually an intermediate step to a more general task. The classical methods [37] were mainly used to extract characteristics from the images. With the recent advancements employing CNNs [30, 168, 106, 164, 80] they can tackle more sophisticated tasks, almost obtaining the segmentation entirely on their own and are part of the our methodology described in Chapter 5.

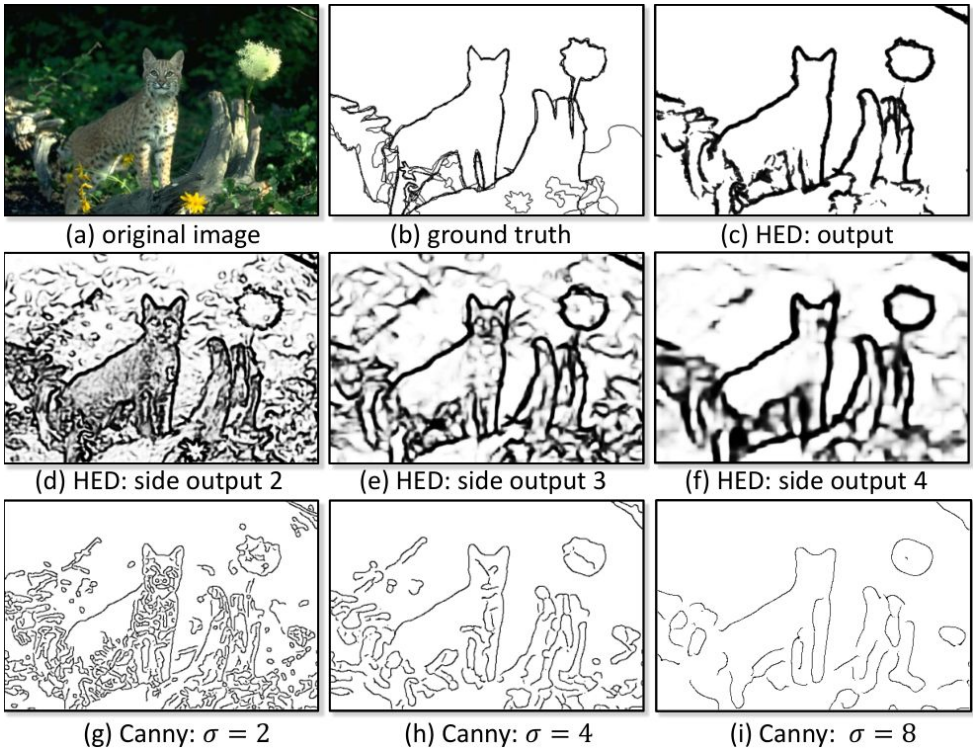


Figure 2.9: Comparison between CNN-based edge detection and the classical Canny algorithm. Image from [168]

Edge detection (*i.e.* boundary estimation) has a duality with obtaining segments [129, 130] — there is an image partition (*i.e.* disjoint segments) from binary contours and a hierarchical partition from fuzzy contours, as described in Section 2.4. Moreover, they are sometimes used as an additional loss term to improve the boundary adherence of deep learning-based object saliency [105] or segmentation [114].

Edge detection was typically considered a low-level vision problem [37], where it was an intermediate step to some higher-level task, for example, it was used for feature extraction for object detection [10]. In 2015, Bertasius *et al.* [30] shifted the paradigm showing that high-level information could assist edge detection by employing a CNN to predict the boundaries, it used a architecture two parallel branches, where one predicted if the central pixel from the patch contains a contour and the other regressed the intensity of that contour.

In parallel, the seminal work of Holistically-nested Edge Detection (HED) [168] was proposed (Figure 2.9). It also used CNN, exploiting higher-level semantic representation to predict boundaries. However, it treats the task as a pixel classification problem as networks for image segmentation, but obtaining the predictions at multiple stages (*i.e.* side predictions), resizing them to the original resolution during the forward pass, and finally combining them to obtain a single final fused prediction, an diagram is shown on Figure 2.10. Thus, it learns to predict edges at multiple scales. During training, the loss function penalizes the fused prediction and each side prediction individually. To our knowledge, every subsequent work on edge detection has taken inspiration or is somewhat based on HED.

Richer Convolutional Features [106] extends HED by learning to fuse and predict the

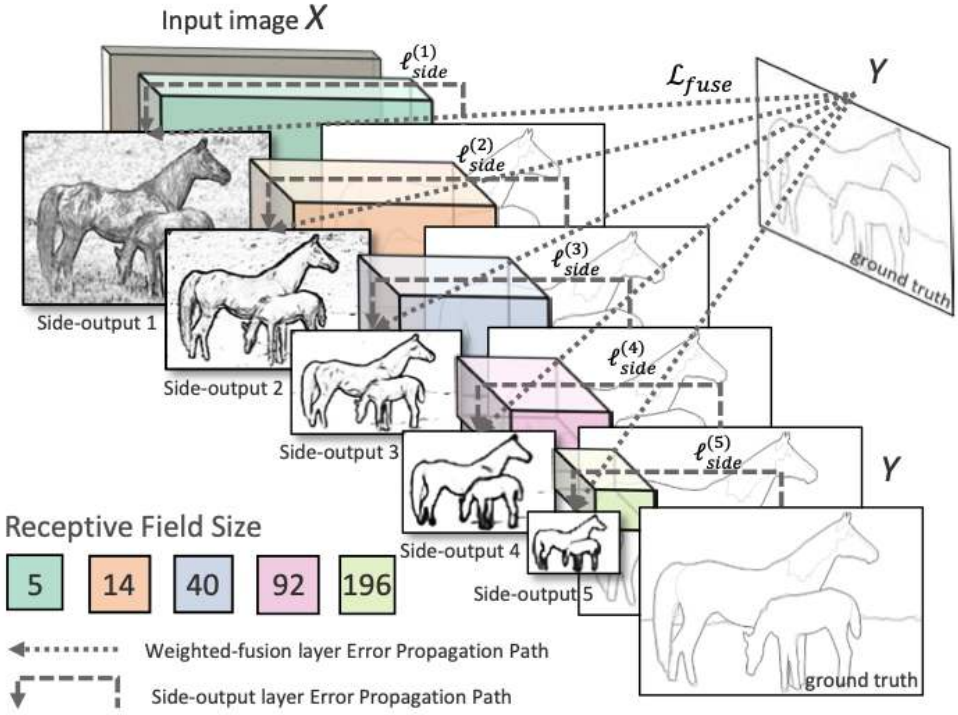


Figure 2.10: Diagram of Holistically-nested Edge Detection architecture and example of side predictions and fusing. Image from [168]

side outputs with convolutional blocks with 1×1 kernels.

Crisp Edge Detector [164] proposed a new architecture that avoids using resizing to obtain a sharper boundary (*i.e.* narrow), since thick edges are an artefact from the blurry up-sampled features. Their network consists of a forward pathway where the input resolution is reduced and a backward pathway where the resolution is recovered. The backward pathway process the lower resolution features back to the original size while combining the forward features with equivalent dimensions, resulting in a single scale prediction with the original input dimensions. Moreover, Deng [59] obtain even crispier boundaries by combining a Dice loss penalization to the standard cross-entropy objective function for the edge classification.

Bi-Directional Cascade Network [80] improves upon the forward and backward pathway architecture by combining the outputs and evaluating the loss function of every stage during the forward and backward pass; they also propose the use of dilated convolutions to fuse the inputs of different scales.

Other existing approaches are more problem-oriented and tune the edge prediction to a specific or surrogate task. For example, Convolutional Oriented Boundaries (COB) [112] estimates multi-scale contours with eight different edge orientations to apply the oriented watershed [18] transform to perform generic image segmentation. PoolNet [105] uses an architecture with a U-shape [141] (*i.e.* decoder-encoder architecture) to predict salient objects while also training to predict the objects' boundaries over different scale during the reconstruction step, with the goal of improving the saliency object's prediction with greater boundary adherence.

2.6 Metric Learning

In this section we review metric learning and its relationship with dimensionality reduction. Both of these techniques are employed at Chapter 5, metric learning is used to facilitate the user interaction as it provide more labeled data and dimensionality reduction to present the data in a 2-dimensional embedding to the user.

Metric learning is a subset of Machine Learning techniques that concerns with estimating a distance function between data points given a optimality criterion. It was initially applied to boost the performance of distance-based classification methods [72, 165], such as the k-nearest neighbors (KNN) [69]. Moreover, it has several relationships with other tasks, such as learning a new data embedding [85, 108], clustering [169], and improving classification methods beyond distance-based classifiers [42].

A canonical example of a change to the metric between data points, despite not always being presented as such, is applying the whitening transform, where data is centered around the origin and scaled by the inverse of its standard deviation. Formalized as, given some samples $\mathbf{x}_i$, with mean $\boldsymbol{\mu}$ and covariance matrix Σ, it can be standardized such that the distances are computed in space with unitary variance

$$\mathbf{y}_i = \Sigma^{-1/2}(\mathbf{x}_i - \boldsymbol{\mu}) \tag{2.16}$$

where the power of $1/2$ is the Cholesky decomposition. Measuring the squared euclidean distance between two standardized samples is equivalent to computing a Mahalanobis [110] metric $d_{\Sigma^{-1}}(\mathbf{x}_i, \mathbf{x}_j)$ over the original samples as follows

$$\begin{aligned}
\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 &= (\mathbf{y}_i - \mathbf{y}_j)'(\mathbf{y}_i - \mathbf{y}_j) && (2.17)\\
&= (\Sigma^{-1/2}(\mathbf{x}_i - \boldsymbol{\mu}) - \Sigma^{-1/2}(\mathbf{x}_j - \boldsymbol{\mu}))'(\Sigma^{-1/2}(\mathbf{x}_i - \boldsymbol{\mu}) - \Sigma^{-1/2}(\mathbf{x}_j - \boldsymbol{\mu}))\\
&= (\Sigma^{-1/2}\mathbf{x}_i - \Sigma^{-1/2}\mathbf{x}_j)'(\Sigma^{-1/2}\mathbf{x}_i - \Sigma^{-1/2}\mathbf{x}_j)\\
&= (\Sigma^{-1/2}(\mathbf{x}_i - \mathbf{x}_j))'(\Sigma^{-1/2}(\mathbf{x}_i - \mathbf{x}_j))\\
&= (\mathbf{x}_i - \mathbf{x}_j)'\Sigma^{-1/2'}\Sigma^{-1/2}(\mathbf{x}_i - \mathbf{x}_j)\\
&= (\mathbf{x}_i - \mathbf{x}_j)'\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)\\
&= d_{\Sigma^{-1}}(\mathbf{x}_i, \mathbf{x}_j) && (2.18)
\end{aligned}$$

Thus, computing distances on the standardized data (affine transformed) is the same as weighting each axis of the differences by some coefficient in Σ^{-1}. Furthermore, this is valid for any linear transformation, L, such that we have a new embedding new embedding $\mathbf{y}_i = L\mathbf{x}_i$, resulting in a quadratic form $d_M(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)'M(\mathbf{x}_i - \mathbf{x}_j) = (\mathbf{y}_i - \mathbf{y}_j)'(\mathbf{y}_i - \mathbf{y}_j) = \|y_i - y_j\|^2$, where M is a positive definite matrix that optimizes some objective criterion given a application of choice.

For distance-based classification, Weinberger and Saul [165] introduced the Large Margin Nearest Neighbor (LMNN), now known in the Deep Learning community as Triplet Loss, to learn a metric (equivalent to a linear transformation) that reduces the classification error of a KNN classifier. The proposed minimization objective function (Eq. 2.19), penalizes the distance between a set of target neighbors and samples of a different class that are within a margin inside the target neighborhood (*i.e.* impostors), this margin

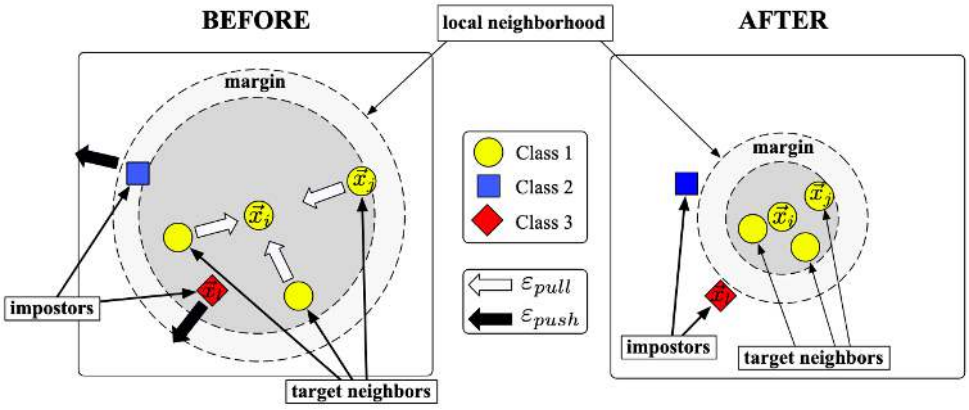


Figure 2.11: Illustration of LMNN learning: Before learning, adversarial samples (*i.e.*, impostors) breach within class 1 margins. After, class 1 samples are self-contained within a region. Arrows indicate the loss function gradient's direction. Figure from [165].

is usually fixed to 1 since any change to it equivalent to the updating the scale of M, Figure 2.11.

$$\mathcal{L}_{LMNN}(M) = \underbrace{\sum_{i\sim j} d_M(\mathbf{x}_i, \mathbf{x}_j)}_{\text{target term}} + \underbrace{\sum_{i\sim j}\sum_{k} I(y_i \neq y_j)[1 + d_M(\mathbf{x}_i, \mathbf{x}_j) - d_M(\mathbf{x}_i, \mathbf{x}_l)]_{\geq 0}}_{\text{margin violation term}} \quad (2.19)$$

where $i \sim j$ indicates the k-nearest neighbors of the same class, y_i is the label of a given sample $\mathbf{x}_i$, $I(.)$ is an indicator function, that is one if the condition is true and zero otherwise, $[\ .\ ]_{\geq 0}$ zeros the values when negative, penalizing only when there is a margin violation, similar to the hinge loss. This objective function, enforces tight clusters within neighbors of the same class (target term) and pushes adversarial samples away (margin violation term).

While this method only estimates a single linear transformation for the whole data, its objective function is not global, penalizing only where necessary — where there is margin violation. To be a valid metric M is restricted to be positive define, such that, this is a semidefinite programming problem and thus convex [33]. They show it can be more efficiently solved with a projected gradient method by restricting M to have positive eigenvalues during gradient descent than with a generic solver. Its computational performance can be further improved by avoiding recomputing the KNN at every iteration of gradient descent.

Goldberg *et al.* [72] proposed the Neighbor Component Analysis (NCA), where they relax the significantly discontinuous nearest neighbor classification function to a softmax function, where the neighbors with the smallest distance have the greatest probability of belonging to the same class. Learning is done through gradient ascent by maximizing the objective function

$$\mathcal{L}_{NCA}(M) = \sum_{i,j|y_i=y_j} \frac{\exp(-d_M(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{i\neq k}\exp(-d_M(\mathbf{x}_i, \mathbf{x}_k))} \quad (2.20)$$

In plain English, it is maximizing the softmax probabilistic assignment between the distances of same class neighbors. This objective function is non-convex and optimizing

using gradient descent leads to a local maximum.

With the rise of deep learning, metric learning methods abandoned working with linear transformations and with metrics (*i.e.* quadratic forms) directly, switching into ANNs for transforming the data from the original space into the learned space, and where the objectives are some function of a distance $d_{l2}(\mathbf{x}_i, \mathbf{x}_j) = \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|$ or a similarity $d_{cosine}(\mathbf{x}_i, \mathbf{x}_j) = 1 - f(\mathbf{x}_i)'f(\mathbf{x}_j)$ measurement between the samples from a network embedding, $f(.)$.

Notably, in computer vision, it gained traction in image retrieval tasks with improvements in triplet mining methods [84, 175], novel loss functions [167, 137] and improvements in video segmentation methods [42]. Musgrave *et al.* [127] showed that most of these novel objective functions are prone to overfitting, requiring more laborious parameter tuning, and simple objectives such as the original Triplet Loss (LMNN) goes a long way.

Concurrently to the development of this thesis, self-supervised methods (*e.g.*, SimCLR [41], SEER [74]) that employ some form of metric learning between perturbed (*e.g.*, flipped, crop, gaussian noise) versions of the same image started gaining attention, they are base on the assumption that these perturbations preserve the content of the image and the network should be robust to it. Thus, they can train a network without labeled data.

2.6.1 Dimensionality Reduction

Similar to metric learning, dimensionality reduction estimates a new transformation of the original data, but with the goal of obtaining a lower-dimensional representation that preserves the information from the original high dimensional space without any labeled data; that is, preserving the global structure of the data, local neighborhoods, or the overall shape of a set of points (*i.e.* manifold).

The classical approach for linear dimensionality reduction is the Principal Component Analysis (PCA), where the data is projected into a k-dimensional space belonging to the k leading eigenvectors of the data's covariance matrix — reducing the dimensionality while preserving orthogonal axis with the greatest variance. It is also related to our standardization example of Equation 2.18.

Due to the convoluted structures of high-dimensional data, non-linear dimensionality reduction methods are the most widely used to greatly reduce the data dimension. They can be roughly divided into two groups: kernel-based methods (*e.g.*, Kernel PCA [147]) where some prior knowledge of data points relationship is available, and it is used to construct the Gram matrix; and neighborhood-based methods [85, 108, 120], which are more flexible, usually requiring only a parameter to describe the neighborhood radius. Our discussion will be concentrated on the latter since they are more flexible and obtains a better 2-dimensional projection on most cases.

One of the classical approaches is the Stochastic Neighbor Embedding [85], that models the data distribution with multiple Gaussians centered around each sample on the original high dimensional space and on the artificial lower dimensional space. From this, the lower-dimensional embedding is updated to minimize the Kullback-Leiber divergence between these two distributions of the data, obtaining a low dimensional representation with similar

distribution.

For each sample i they define probabilities distribution to every other point j. Such that, the high dimensional probability is given by

$$p_{ij} = \frac{\exp(-d_\sigma(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{i \neq k} \exp(-d_\sigma(\mathbf{x}_i, \mathbf{x}_j))} \tag{2.21}$$

where $d_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2$, σ^2 is a hyper-parameter, which can be set manually or from function given the number of neighbors. The low-dimensional embedding, $\mathbf{y}_i$, probability distribution is defined by

$$q_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{i \neq k} \exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)} \tag{2.22}$$

Thus, the SNE minimization objective is

$$L_{KL}(\mathbf{p}, \mathbf{q}) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.23}$$

and its optimization is done through gradient descent.

By evaluating the objective function's derivative over $\mathbf{y}_i$ we obtain additional insights about the algorithm behavior

$$\frac{\partial L}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_i - \mathbf{y}_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji}) \tag{2.24}$$

where, $\mathbf{y}_i$ moves towards $\mathbf{y}_j$ if they are closer on the high-dimensional space ($p_{ij} > q_{ij}$ and $p_{ji} > q_{ji}$) and outwards otherwise.

Note that, the probability distribution between points (Eq. 2.21 and 2.22) are the same as the objective function (Eq. 2.20) of the NCA algorithm presented on the previous section. Hence, while the SNE algorithm minimizes the KL divergence between the low-dimensional and high-dimensional distributions, the NCA minimizes the divergence between the transformed data embedding and a binary encoding of same class samples distribution, where it is one if they belong to the same class and zero otherwise.

Van der Maaten and Hinton [108] proposed a slight change to the original SNE model motivated by the problem of crowding on high-dimensional spaces, where distances gets more very similar as the data dimension increases, by using a long-tailed distribution on the low dimensional embedding, thus, spreading the points further a part on the 2D space.

The distribution of choice was a t-student distribution with 1 degree of freedom. Therefore, q_{ij} (Eq. 2.22) becomes

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{i \neq k} (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}} \tag{2.25}$$

and the gradient of the KL-divergence loss function is given by

$$\frac{\partial L}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_i - \mathbf{y}_j)(\tilde{p}_{ij} - q_{ij}) \underbrace{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}_{\text{low-dim. smoothing}} \tag{2.26}$$

Its behaviour is similar to Equation 2.24, but the pushing and pulling effect diminishes faster as the points are further a part on the low-dimensional embedding. Additionally, they consider a symmetric high-dimensional probability, $\tilde{p}_{ij} = (p_{ij} + p_{ji})/2$.

One of the downsides of both of (t-)SNE algorithms is their high computational cost; at every iteration of gradient descent, the distance between every sample must be recomputed to update the low-dimensional distributions q_{ij} denominator. From this, McInnes *et al.* developed the Uniform Manifold Approximation and Projection (UMAP) [120], providing a new theoretical foundation for dimensionality reduction, proposing a much faster and, arguably, better algorithm. A recent work [54] have shown that UMAP's official and widely used implementation does not comply with the original theoretical motivation. Hence, we will mostly focus on the implemented version. Additionally, they underline UMAP's relationship with contrastive learning.

The algorithm steps are very similar to t-SNE, with small nuances in the equations. It consists of three main steps: Compute a fuzzy representation of the edges, analogous to t-SNE's high-dimensional probability distribution; Initialize a low-dimensional embedding; Optimize embedding with cross-entropy objective function, in contrast to the KL-divergence from t-SNE.

The fuzzy edge weights are given by

$$p_{ij} = \exp\left(\frac{-\max(0, \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \rho_i)}{\sigma_i}\right) \tag{2.27}$$

where, ρ_i is the distance from the nearest neighbor to i, enforcing that $p_{ij} = 1$ for the closest sample, and σ_i is found from the following equation

$$\sum_j p_{ij} = \sum_j \exp\left(\frac{-\max(0, \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \rho_i)}{\sigma_i}\right) = \log_2(k) \tag{2.28}$$

where k is the hyper-parameter of number of neighbors and binary search can be used to solve this equation. This represents the uniformity assumption of the UMAP.

The final graph is made symmetric with

$$\tilde{p}_{ij} = p_{ij} + p_{ji} - p_{ij}p_{ji} \tag{2.29}$$

And the low-dimensional membership strength is proportional to a t-distribution as in t-SNE but with two additional parameters to adjust its shape

$$q_{ij} = \Phi(\mathbf{y}_i, \mathbf{y}_j | a, b) = (1 + a\|\mathbf{y}_i - \mathbf{y}_j\|^{2b})^{-1} \tag{2.30}$$

The parameters a, b are found by non-linear least squares to produce a smooth curve that is similar to the high-dimensional distribution of the data, by fitting $\Phi(., .|a, b)$ to

the function

$$\Psi(\mathbf{y}_i, \mathbf{y}_j | min\_dist) = \exp\left(-\max(0, \|\mathbf{y}_i - \mathbf{y}_j\| - min\_dist)\right) \tag{2.31}$$

where, min_dist is a hyper-parameter defining the minimum distance between samples on the projection.

Finally, the objective function is given by the binary cross-entropy between the high and low dimensional membership strengths

$$\begin{aligned} L_{BCE}(\mathbf{p}, \mathbf{q}) &= \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right) && (2.32) \\ &= \sum_i \sum_j p_{ij}(\log(p_{ij}) - \log(q_{ij})) + (1 - p_{ij})(\log(1 - p_{ij}) - \log(1 - q_{ij})) \\ &\propto -\sum_i \sum_j p_{ij} \log(q_{ij}) + (1 - p_{ij}) \log(1 - q_{ij}) && (2.33) \end{aligned}$$

Equation 2.32 is simplified to Equation 2.33 because the $\mathbf{p}$ is fixed during the optimization. Moreover, p_{ij} is zero almost every where outside of the k neighborhood, thus, the equation objective function can be further reduced to

$$L_{BCE}(\mathbf{p}, \mathbf{q}) \propto \underbrace{-\sum_{i \sim j} p_{ij} \log(q_{ij})}_{\text{attractive term}} + \underbrace{\sum_i \sum_j (1 - p_{ij}) \log(1 - q_{ij})}_{\text{repulsive term}} \tag{2.34}$$

To further speedup the implementation, the repulsive term, which is quadratic over the number of data points, is not computed exactly, and random pairs are sampled at every iteration of gradient descent, matching the strategy used by Word2Vec [122].

In summary, UMAP computational performance is better because it avoids recomputing the whole dataset's distance matrix at every iteration by sampling negative pairs and using a loss function that does not require a sum over all sample on the denominator.

It is also optimized using gradient descent, such that at each UMAP's optimization iteration, an attractive force and a repulsive force is applied to the current embedding, adjusting the data low-dimensional embedding. Thus, the objective function gradients' over $\mathbf{y}_i$ for the *attractive term* is

$$-\underbrace{\frac{2ab\|\mathbf{y}_i - \mathbf{y}_j\|^{2(b-1)}}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2}}_{\text{attractive smoothing}} p_{ij}(\mathbf{y}_i - \mathbf{y}_j) \tag{2.35}$$

and the for the *repulsive term* is

$$\underbrace{\frac{2b}{(\epsilon + \|\mathbf{y}_i - \mathbf{y}_j\|^2)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^{2b})}}_{\text{repulsive smoothing}} (1 - p_{ij})(\mathbf{y}_i - \mathbf{y}_j) \tag{2.36}$$

Figure 2.12 gives additional insight into the behavior of these two forces. The attractive

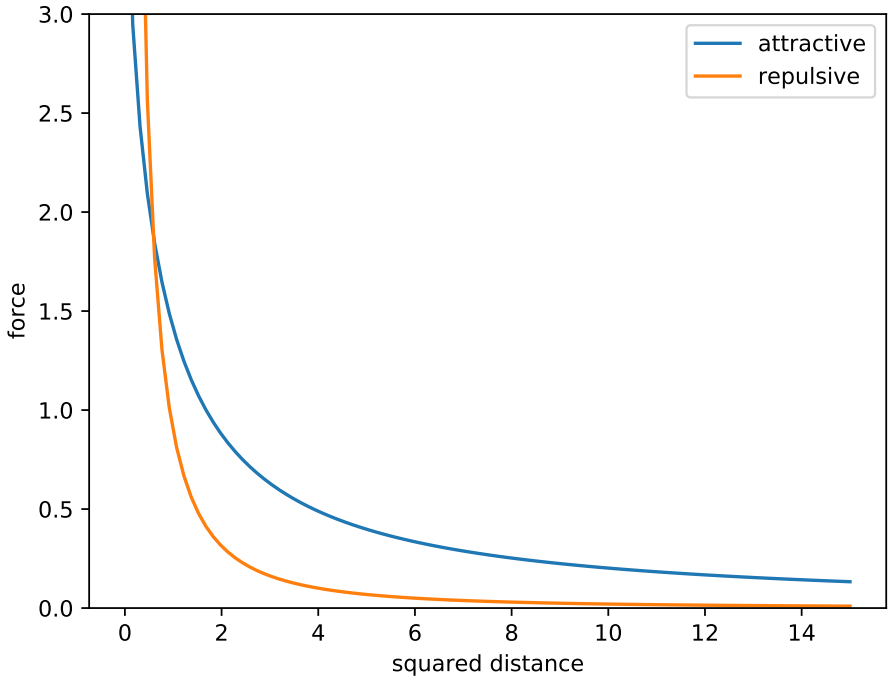


Figure 2.12: Shape of UMAP smoothing terms of attractive and repulsive forces according to their squared distance (*i.e.* $\|\mathbf{y}_i - \mathbf{y}_j\|^2$) with fixed $a = 1.58$ and $b = 0.895$, estimated from MNIST dataset. The repulsive term (orange line) goes abruptly to zero as the hinge loss from LMNN.

force has a longer tail, pulling neighbors with stronger high-dimensional membership closer despite their distances on low dimensional embedding. While the repulsive strength is very similar to the Triplet Loss (Eq. 2.19), going abruptly to zero as it pushes samples away, having a small high-dimensional membership strength (*i.e.* significant pushing force) when they are inside a close range and being null almost everywhere outside of this small radius.

Chapter 3

Interactive Segmentation by Dynamic Trees

In this Chapter we present the first algorithmic contribution of this thesis. As described in the previous pages, image segmentation is challenging and often requires users' assistance for correction.

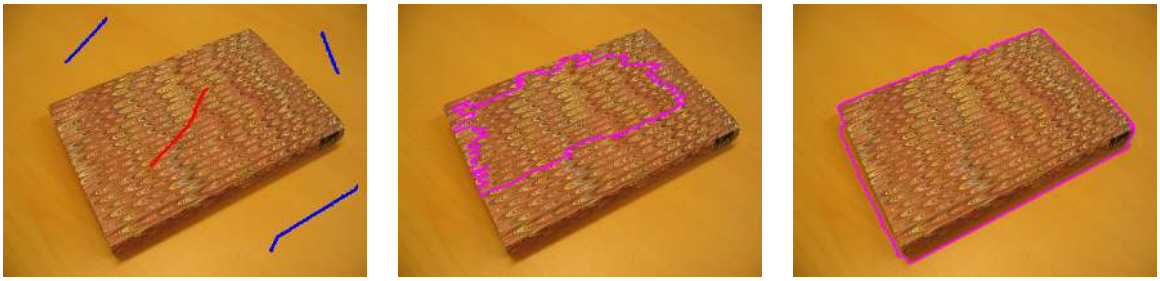
Among the many established approaches to solving the interactive segmentation problem, graph-based algorithms from user-defined markers have showed to be quite effective [49], enjoying a developed theoretical background [12, 43, 44], and being easily extendable (sometimes no change is necessary) to the semi-supervised classification domain (*i.e.* transductive learning) for non-image data [13].

However, most of these approaches resolve the segmentation on static graphical models [64, 35, 75, 51, 49], such that unlabeled data information is only partially used during the segmentation process, providing a pathway to propagate labels over the graph, but without updating the propagation's strength as labels are estimated.

Some techniques [143, 158] acquire additional information from the unlabeled samples, updating the likelihood of each pixel through multiple executions, exploiting the algorithm's intermediate label assignment of the originally unlabeled pixels to compute a likelihood model until convergence of the label assignments. More specifically, the unary term from the objective function (Equation 2.1) is re-estimated using the results from the previous iteration of the algorithm. Being the same as an Expectation Maximization [57] process, where at one iteration the label assignment is computed maximizing the objective function given a data distribution and in the following iteration its probability density function is updated given the estimated labels.

In this chapter, we present a novel approach that dynamically estimates the graph's arc-weight (*i.e.* pairwise term) as the segmentation is computed in a region growing fashion on a single algorithm execution. Thus, improving the model with unlabeled data information as the segmentation advances. Our experiments show that this approach outperforms all classical methods with static arc-weights of the Power Watershed family [64, 35, 75, 51, 49] and is competitive with approaches that re-estimate the unary terms [158] and the state-of-the-art [38].

Originally, our idea was to use a classifier to learn the pattern of the competing regions (labels) as they propagated. However, this was shown to be ineffective in two ways, it



(a) (b) (c)

Figure 3.1: (a) Original image with foreground (red) and background (blue) markers; Segmentation results delineated by magenta contour with: (b) static arc-weight [64]; (c) dynamic arc-weight [36].

was computationally intensive and made the algorithms less interactive, and it was not accurate, underperforming due to poor generalization. To our surprise, using the mean resulted in the best results.

The following sections presents our motivation, the proposed framework, the algorithm and examples of dynamic arc-weight estimation methods, the experimental results, and the conclusion.

3.1 Motivation

Existing methodologies solve interactive segmentation on an image graph with static arc-weights. These weights are prone to failing to accurately define the object's boundary due to noises, image-related artifacts or by having discrepant features from annotated pixels. This is confirmed when several different criteria cannot solve a segmentation without considerate user effort (*i.e.*, annotation) despite the object seeming somewhat separated from the background, Figure 3.1. Thus, we propose an algorithm that outperforms existing approaches by using a moving average to provide a more robust arc-weight estimate, some variations of this algorithm are provided, and its relationship with existing methods is discussed.

3.2 Methodology

Existing algorithms can solve interactive segmentation in real-time only on limited scenarios; for example, the maxflow-mincut algorithm [34] is NP-hard beyond the binary case, the number of linear systems required to solving the Random-Walk [75] increases with the number of distinct labels, and on both cases the final label of any pixel is only fixed upon the algorithm's convergence. Moreover, they can only be executed on graphs with fixed arc weights, restricting the objective functions that they can optimize.

In contrast, the IFT framework, presented in Section 2.3, performs the segmentation in a region growing manner, fixing nodes with immutable labels as the optimum-path propagates, and obtaining satisfactory results even when its original assumptions [44] are violated [153, 124, 56]. Moreover, the IFT path-cost map C and the predecessor P

save how strongly connected a node is to its source and the path required to reaching it. Thus, from C we have a measurement that is proportional to the confidence of each node labeling.

From this, we propose that additional information beyond the labeled nodes can be used to improve the optimum-path routing between nodes with weak connectivity. For that, we propose to explore the information of nodes with strong connectivity to provide this additional data. A simple, yet effective criterion to achieve this goal is to measure how much the weakly connected node differs from the average of each strongly connected component, considering a different arc weight to each connected component. Thus, measuring how much a weakly connected nodes belongs to the strongly connected clusters, a reasonable measurement is the pixel's color difference to the strongly connected cluster's average. To our knowledge, this is the first time dynamic arc weights have been explored for image segmentation.

To perform this strategy, a naive algorithm could execute the IFT multiple times. At every IFT execution, it would select a non-decreasing subset of strongly connected pixels; their average would be computed and used to update the weakly connected samples' arc weights; the subsequent executions would select a greater number of strongly connected pixels until no pixels would required to have their arc weights recomputed — all pixels would be permanently labeled.

However, this leads to undesirable properties: The IFT is executed multiple times, each time for a different threshold in C to select the subset of strongly connected nodes, significantly decreasing the algorithm speed; It requires multiple arc-weights computations for the weakly connected nodes, one for each strongly connected cluster (one for each average), further increasing the memory footprint and complicating the algorithm; And computing the average multiple times further decreases the algorithm's computational efficiency.

Hence, we propose to solve the segmentation is a single IFT execution using dynamic programming, which is already an essential requirement for an efficient implementation of the IFT algorithm. The strategy is as follows, at every iteration, only the unlabeled nodes connected to labeled regions are evaluated, so only these arc weights require recomputation. Thus, the burden of recomputing multiple arc weights between every unlabeled node is eliminated. Note that, by increasing the C's threshold at every interaction, the set of strongly connected nodes never decreases, only including additional nodes at every step. Hence, we save a moving average for each group and update incrementally as strongly connected nodes are removed from the IFT's queue. Therefore, unlabeled arc weights are only computed when necessary, and the groups' averages are updated on-the-fly.

Our method, Dynamic IFT, is presented on Algorithm 2, minor adjustments might be necessary on line 14 to extend to additional arc-weight functions presented on the next section.

Extending the IFT (Algorithm 1), an extra mapping $R : V \mapsto V$ is used to save the nodes' roots. Lines 1-11 initialize the algorithm, line 13 selects the strongest connected node that has not propagated its label yet, line 14 updates the root node group, line 16-17 computes the path-cost, which can be defined in different ways and will be explained next,

Algorithm 2 DYNAMIC ARC-WEIGHT IFT

INPUT: Graph $\mathcal{G} = (V, E, w)$, seeds' set $\mathcal{S}$
OUTPUT: Cost map C, Root map R, Predecessor map P, label map L and set of trees' averages $\{T_i | \ i \in R\}$
AUXILIARY: Priority queue $Q = \emptyset$, temporary variable *pathcost*

```
1.  For each p ∈ V
2.      C(p) ← ∞
3.      R(p) ← p
4.      P(p) ← nil
5.      T_R(p) ← ∅
6.  For each S_i ∈ S
7.      For each p ∈ S_i
8.          C(p) ← 0
9.          L(p) ← i
10. For each p ∈ V
11.     Insert p in Q
12. While Q ≠ ∅
13.     Remove p from Q, such that p = arg min_{∀q∈Q}{C(q)}
14.     T_R(p) ← {T_R(p) ∪ p}
15.     For each q ∈ V | (p, q) ∈ E, q ∈ Q
16.         Estimate w(p, q) as described in Section 3.2.1
17.         pathcost ← max{C(p), w(p, q)}
18.         If pathcost < C(q)
19.             C(q) ← pathcost
20.             R(q) ← R(p)
21.             P(q) ← p
22.             L(q) ← L(p)
```

and the lines 18-22 extends the path routing if a better path was found.

3.2.1 Arc-weight Estimation

This section discusses multiple strategies for arc-weight estimation (line 16 of Algorithm 2) that can be easily applied to the Dynamic IFT framework, resulting in variants of the moving average arc weight for different assumptions over the regions of interest properties (*i.e.* pixels' features).

For some applications, notably in the bioimaging area, it would be reasonable to assume that each object has a homogeneous characteristic. Hence, a single average per object can be computed with the arc-weight function

$$w_L(p, q) = \|\mu_{L(p)} - I(q)\| \tag{3.1}$$

where $\mu_{L(p)}$ is the average of $I(.)$ for every conquered node with the same label as p.

This assumption is violated in many cases, especially in natural imaging, because object features have distributions with multiple modes. From our experience, each optimum-path tree ends up acquiring a distinct characteristic. Thus, we can assign one average for

each optimum-path tree,

$$w_R(p,q) = \|\mu_{R(p)} - I(q)\| \tag{3.2}$$

where $\mu_{R(p)}$ is the features' average of nodes rooted in $R(p)$. Thus, indexing each average through its tree roots.

The arc weight can also take into account multiple sources from the same label, enjoying global information at the cost of computational performance; this arc weight is given by

$$w_G(p,q) = \min_{\forall r \in \mathcal{S}_{L(p)}} \|\mu_r - I(q)\| \tag{3.3}$$

where μ_r is the features' average of nodes rooted in r. At every iteration multiple roots are evaluated, selecting the one that offers the least cost. Note that, this increases the computational complexity since multiple tree are evaluated at each arc-weight estimation, in contrast to one in the other scenarios.

Further, it could be the case where optimum-path features' distribution shifts along the graph, so the standard average would not be a reasonable estimate. Thus, we propose using an average with exponential decay along the paths, so nodes closer to the tail along the predecessor path have more significant influence over the extending path arc weight, given by

$$w_{\exp}(p,q) = \|\mu_{P(p)} - I(q)\|$$

where

$$\mu_{P(p)} = (1-\alpha)I(p) + \alpha\mu_{P(p)}$$

By defining $P^2(p) = P(P(p))$ we can further expand the equation using recursion

$$\begin{aligned}
\mu_p &= (1-\alpha)I(p) + \alpha\mu_{P(p)} \\
&= (1-\alpha)I(p) + \alpha[(1-\alpha)I(P(p)) + \alpha\mu_{P(P(p))}] \\
&= (1-\alpha)\alpha^0 I(P^0(p)) + (1-\alpha)\alpha^1 I(P^1(p)) + \alpha^2\mu_{P^2(p)} \\
&\dots \\
&= (1-\alpha)\sum_{i=0} \alpha^i I(P^i(p))
\end{aligned} \tag{3.4}$$

The parameter α is a value between 0 and 1. When $\alpha = 0$ it is the standard IFT; when α the inverse of the number of predecessors, it is the predecessor path average without weighting.

Additionally, this arc-weight function allows the computation of the IFT differentially (DIFT) [62], a characteristic that is not enjoyed by the other functions.

For Equations 3.1- 3.3 it is not possible to guarantee the algorithm results would be the same once a a optimum-path tree is pruned. For example, given a optimum-path forest computed with one of the given arc-weight functions and the $f_{\max}$ connectivity function (Eq. 2.7), the path cost is non-decreasing as required by the DIFT. Now, lets suppose we have two optimum-path branches A and B belonging to same the average (*e.g.* every node has the same root in the case of w_R, Eq. 3.2) and A was conquered before B, any operation that extends or prunes A will make the path costs in B obsolete because it was computed with values that no longer belongs to the arc-weight average. Hence, the

Method	Intersection over Union		
	Grabcut Andrade [14]	Grabcut Gulshan [76]	Geodesic Star [76]
RW [75]	0.727 ± 0.159	0.525 ± 0.179	0.517 ± 0.185
GC [34]	0.746 ± 0.156	0.488 ± 0.226	0.443 ± 0.228
WS [50]	0.800 ± 0.138	0.583 ± 0.219	0.507 ± 0.217
OC [158]	0.728 ± 0.207	0.628 ± 0.267	0.405 ± 0.265
LP [38]	0.764 ± 0.158	0.627 ± 0.188	0.564 ± 0.204
PW [49]	0.800 ± 0.138	0.585 ± 0.219	0.507 ± 0.218
IFT [64]	0.798 ± 0.137	0.585 ± 0.217	0.506 ± 0.217
DT_L [36]	0.691 ± 0.192	0.607 ± 0.242	0.452 ± 0.247
DT_{exp} [36]	0.816 ± 0.132	0.603 ± 0.207	0.528 ± 0.218
DT_R [36]	$\mathbf{0.832 \pm 0.133}$	$\mathbf{0.691 \pm 0.185}$	$\mathbf{0.602 \pm 0.204}$

Table 3.1: Quantitative results of segmentation on Microsoft's datasets.

Equations 3.1- 3.3 cannot be updated efficiently because they required recomputing paths from different branches.

The dynamic arc weight with exponential decay (Eq. 3.2.1) does not display this behavior because each node store its own moving average, they are never updated after they leave the queue and each average only affects the optimum-path branch rooted in it self. Hence, the tree removal operation (*i.e.* pruning) would remaining the same as proposed by Falcão *et al.* [62].

3.3 Experimental Results

To evaluate the proposed methodology, we used the Microsoft's datasets, GrabCut [143] that contains 50 images and the markers provided by Andrade and Carrera [14], and Geodesic Star dataset [76] that contains 151 images and their own set of markers. Note that, Geodesic Star's dataset contains the images from GrabCut, but the markers set is different (very reduced), thus yielding very different results.

We also evaluated on the DAVIS [133] dataset of foreground segmentation on videos, following [89], 10% of the frames were sampled resulting on 345 images. Since, markers are not provided, we defined the foreground seeds as the erosion with radius 15 from the ground-truth masks and the backgrounds seeds as the complement of a dilation with the same radius — images where the foreground disappeared from the erosion were discarded, this only occurred 5 times.

We compared with the before mentioned methods (Section 2.2), GraphCut (GC) [34], Random Walks (RW) [75], Watershed Cuts (WS) [50], Image Foresting Transform (IFT) [64], Power Watershed with $q = 2$ (PW) [49], and more recent approaches, One Cut (OC) [158] and Laplacian Coordinates (LP) [38]. The images were evaluated on the RGB color space and the arc-weights are a function of the pixels' color Euclidean distance. The w_{exp} parameter α was fixed to 0.5.

The annotations from [76] are not very informative, since it only contains about four markers per image that cover a small area of both the background and foreground, Fig-

	Intersection over Union
Method	DAVIS [133]
RW [75]	0.784 ± 0.148
GC [34]	0.761 ± 0.148
WS [50]	0.787 ± 0.143
OC [158]	0.601 ± 0.218
LP [38]	0.809 ± 0.140
PW [49]	0.788 ± 0.143
IFT [64]	0.788 ± 0.143
DT_L [36]	0.676 ± 0.145
DT_{exp} [36]	0.798 ± 0.142
DT_R [36]	$\mathbf{0.822 \pm 0.136}$

Table 3.2: Quantitative results of segmentation on DAVIS dataset.

ure 3.2. The markers from [14] contains a more carefully selected set of scribbles. This can be seen when comparing the two first columns of Table 3.1 over the images of the Grabcut dataset.

Figure 3.2 presents the segmentation results of the multiple methods on the Microsoft's dataset with the Geodesic Star markers. The results of the image-foresting transform (IFT), watershed (WS), and power watershed (PW) are almost identical as expected given their close relationship described in Section 2.2. Since the One-Cut (OC) does not constrain connectivity, it miss-classifies some regions in the background that are similar in color to the object of interest, mushroom. The Dynamic Tree Label (DT_L) obtains satisfactory result due to the homogeneous characteristics of the object, which can be effectively summarized in a single mean, the same can be side to DT_R, which extends it to multiple means, the variant with exponential decay (exp), produces results similar to the watershed. Hence, an increase in the parameter α might yield results more similar to the other variants.

Figures 3.3, 3.4 displays the results of the DAVIS dataset. In Figure 3.4, the region of interest contains multiple colors, hence DT_L fails, segmenting only the darker blue regions correctly. The RW and LP do not segment the woman's limbs due to their tendency smoothing of boundaries. The remaining algorithms produce reasonable results. Notably, DT_R and DT_{exp} can segment the baby's foot due to their strong adaptation to local characteristics.

3.4 Conclusion

In this chapter, we proposed a novel approach for interactive image segmentation that updates the arc weights on the fly. It achieved superior performance to existing static arc-weighed methods on multiple datasets even when compared to methods with dynamic unary values (*e.g.* One Cut). In between the article publication and the thesis writing, several methods extended presented methodology [24, 20, 16], showing, without doubt, the advantage of using adaptive arc-weights.

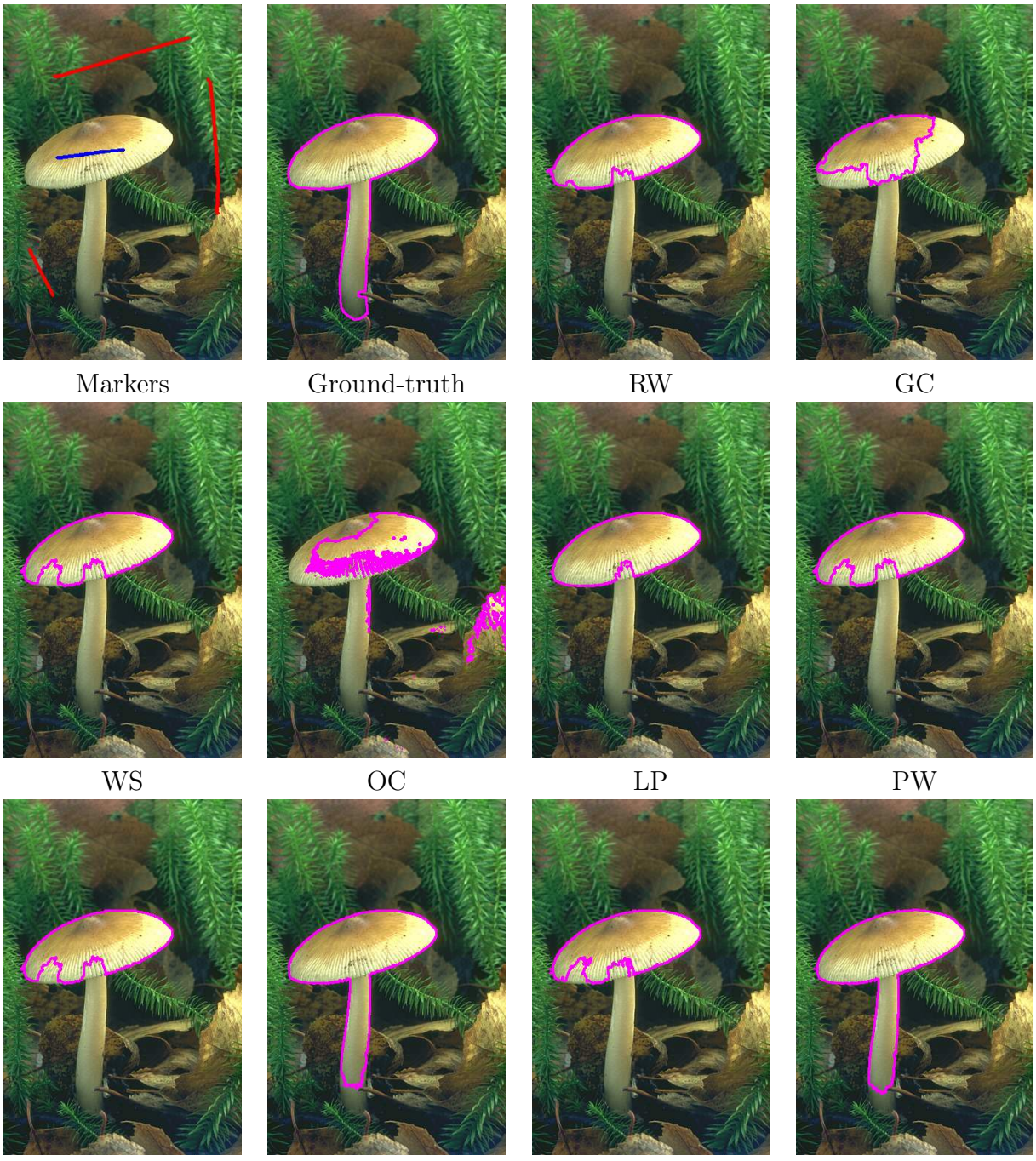


Figure 3.2: Qualitative results on Microsoft's dataset. Foreground markers are blue and background are red. Where the presented DTs variants are regarding the labels (L), exponential decay (exp) and root (R).

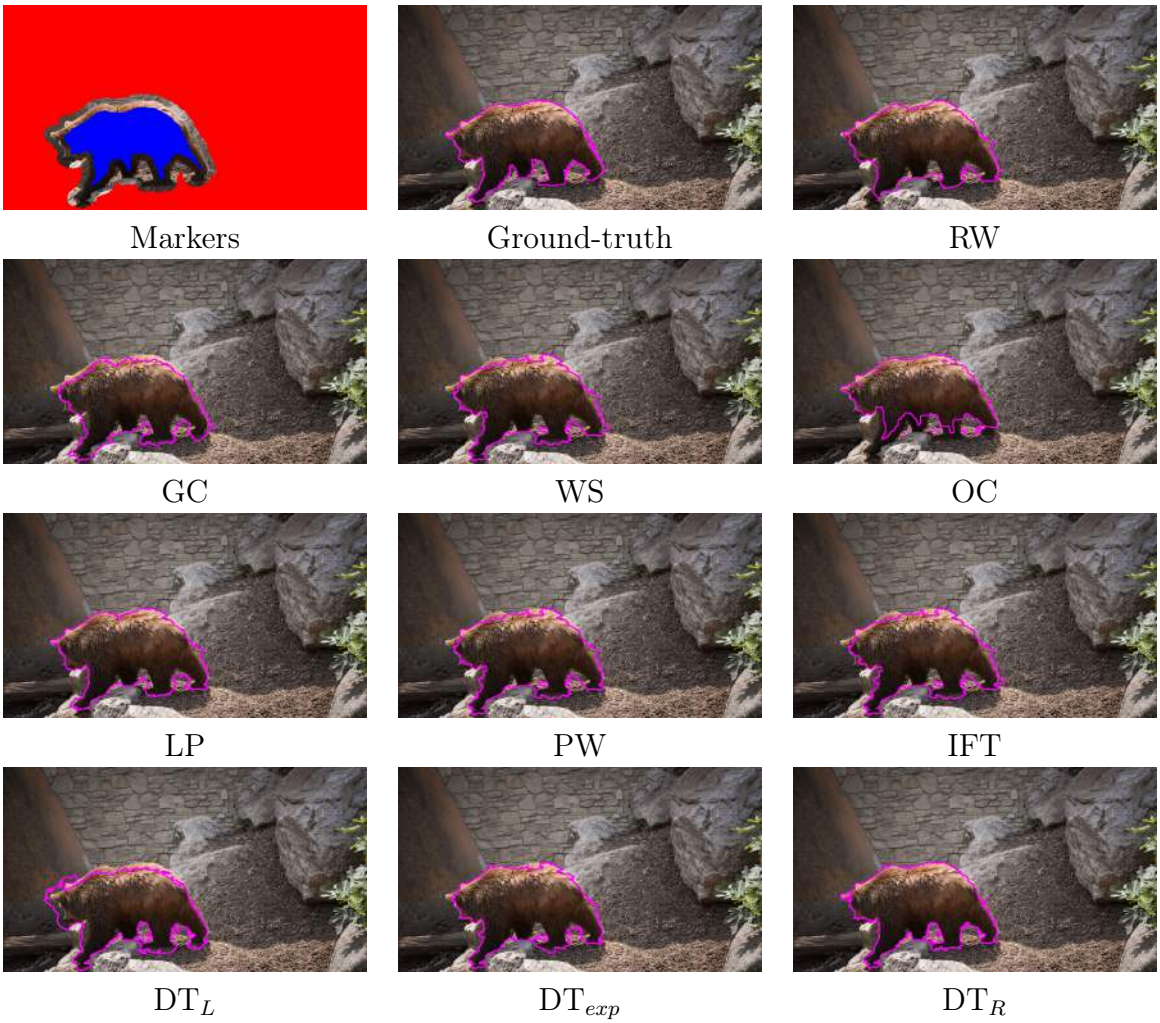


Figure 3.3: Qualitative results on DAVIS. Foreground markers are blue and background are red, generated from erosion and dilation of ground-truth. Where the presented DTs variants are regarding the labels (L), exponential decay (exp) and root (R).

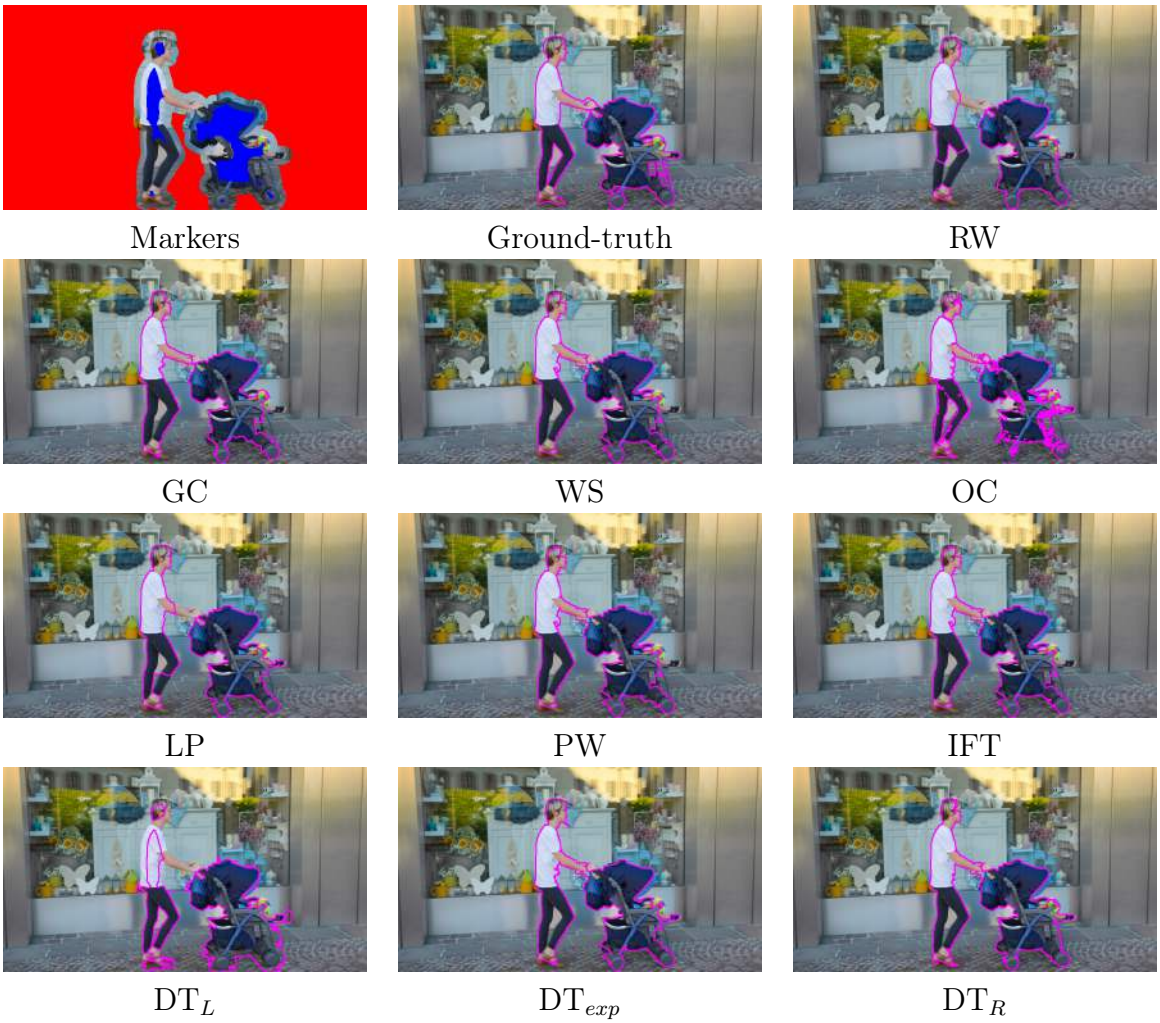


Figure 3.4: Qualitative results on DAVIS. Foreground markers are blue and background are red, generated from erosion and dilation of ground-truth. Where the presented DTs variants are regarding the labels (L), exponential decay (exp) and root (R).

On the next chapter its synergy is evaluated with a contour correction mechanism, further improving the image annotation process.

Chapter 4

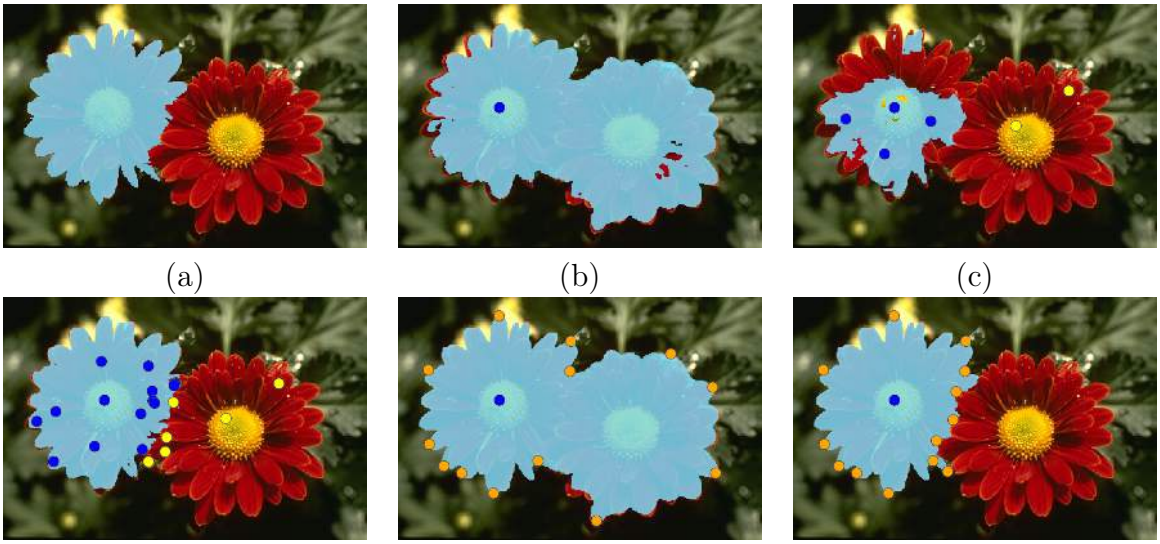
Interactive Correction from Contours by Grabber

As described previously, interactive image segmentation requires object localization and delineation, and it could be followed by an enhancement and verification steps. Object localization may involve the indication of interior and exterior markers, points along the object boundary, a bounding box around the object, the information about its pose, and its approximate position in the image domain, for instance. Object enhancement improves the contrast between object and background to facilitate their separation (*e.g.* arc weight estimation, an object saliency map, an edge detection image). Object delineation defines the precise spatial extension of the object in the image and it can usually improve when using a suitable object enhancement. Object verification may be done by the user or by optimizing some criterion function. As accurate segmentation rarely works from a single input action, the user usually verifies the result and takes actions for correcting errors, such that all previous steps can be improved along with multiple user interventions.

Object localization and verification are better performed by humans while machines usually outperform humans in object enhancement and delineation. An ideal method for interactive image segmentation should explore the complementary abilities of humans and machines to minimize user effort while maintaining complete user control over the segmentation process [65]. For that, the user's actions to correct segmentation should be effective, not causing errors in other parts of the object where the results are already correct.

In the recent years, interactive image segmentation methods have considerably evolved. Initially, the methods did not learn the parameters of the segmentation model [31, 91]. Many methods then started learning a model but fixing it during segmentation [65, 47, 125, 170, 111]. Other approaches learn and update a model from user inputs during segmentation [143, 171, 154, 36] and a few most recent ones start from a pre-trained model that is adapted from user inputs during segmentation [89, 150, 93].

However, while classical approaches, such as live-wire and live-lane [65], can provide considerable user control over the segmentation process at the cost of a higher user effort in more complex boundaries, the existing methods still lack user control during segmentation correction. See the comparison in Figure 4.1 for a recent method results, Feature Back-propagation Refinement Scheme (fBRS), which is based on deep neural networks [150].



(a) (b) (c)

(d) (e) (f)

Figure 4.1: (a) Original image with ground-truth segmentation. (b), (c), (d) Results of fBRS [150] after 1, 6, and 25 iterations, respectively, of an oracle user that inserts internal (blue) and external (yellow) points. One can see that corrections in (c) ruin correct parts of the result in (b). (e) Grabber can improve the result from iteration 1 of fBRS, delineating an optimum contour constrained to pass through estimated anchor points (orange) by letting the user move, add, and remove anchor points, (f); Grabber convergences faster with higher accuracy with 13 fewer user interventions.

We address the above problem with *Grabber* — a tool to improve convergence (user control) in interactive image segmentation. Grabber is inspired by the high user control offered in methods based on optimum boundary segments between user-selected anchor points [65, 126, 66, 123, 46, 98], but it provides a unique way to complete segmentation from any other initial method. It estimates anchor points from a segmentation mask and sort the points in one boundary orientation, rather than requiring the user to provide a sequence of anchor points in a given order along the boundary as in Live-Wire [65]. The object's delineation is obtained by an optimum contour constrained to pass through the anchor points. It can also leverage object's properties from the initial coarse segmentation (*e.g.* internal and external probability density maps) to improve boundary delineation. The user can control object delineation in any part of the boundary by adding, removing, and moving anchor points.

In order to demonstrate the impact of Grabber to increase convergence in interactive segmentation, we integrate it with two recent approaches, a pixel classification method based on deep neural networks, fBRS [150], and a graph-based method, named Dynamic Trees (DT) [36, 67] presented in the previous Chapter. Thus, Grabber is not meant to directly compete with other approaches but to improve them, as soon as the user experiences difficulty in convergence or realizes that Grabber can complete segmentation faster than the initial method. Figures 4.1(e)-(f) illustrate its potential to improve interactive segmentation when integrated with fBRS.

As main contributions, we present a tool, named Grabber, to improve convergence in interactive segmentation, when integrated with other approaches; and two hybrid meth-

ods, fBRS-Grabber and DT-Grabber, that result from that integration.

4.1 Methodology

For a given segmentation mask, we estimates anchor points along the boundary of the mask and let the user manipulate them: add, remove, and move points. The initial anchor points are obtained in a desired order (clockwise or anti-clockwise) using the Douglas-Peucker algorithm [60] with a curvature-approximation threshold ϵ. It transforms the contour of the mask into line segments and uses their extreme points as anchors. Then it estimates the object boundary as an optimum contour constrained to pass through those anchor points. For that, Grabber uses the Image Foresting Transform (IFT) algorithm from Section 2.3.

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\} \subset V$ be the set of anchor points in a given order, such that $s_i \prec s_{i+1}$, $i = 1, 2, \ldots, n-1$. The IFT algorithm computes one optimum contour $\mathcal{L}$ constrained to pass through anchor points by following their order in $\mathcal{S}$. This requires n executions of the IFT algorithm to find the minimum-cost segments from s_i to s_{i+1}, $i = 1, 2, \ldots, n-1$, and then from s_n to s_1. The of set contour segments, $\mathcal{L}$, is empty before the first execution. After each execution $1 \leq i < n$, $\mathcal{L}$ returns with the vertices of the optimum path $P^*(s_{i+1})$ with terminus s_{i+1} and root s_1. For the last execution n, s_1 is removed from $\mathcal{L}$ in order to close the contour as an optimum path $P^*(s_1)$ with root in a vertex s whose $P(s) = s_1$. The path-cost function f may be defined as

$$
\begin{aligned}
f(\langle u \rangle) &= \begin{cases} 0 & \text{if } u = s_1 \text{ and } \mathcal{L} = \emptyset, \\ +\infty & \text{if } u \in V \setminus \mathcal{L}, \end{cases} \\
f(\pi_u \cdot \langle u, v \rangle) &= f(\pi_u) + w_I(u, v), \qquad (4.1) \\
w_I(u, v) &= \exp\left(-\frac{\|I(l_{u,v}) - I(r_{u,v})\|}{\sigma_I}\right) \qquad (4.2)
\end{aligned}
$$

where $l_{u,v}$ and $r_{u,v}$ are the left and right vertices of an edge $(u, v) \in E$, respectively, and $\sigma_I > 0$ is a hyper parameter, its choice depends on the observed visual differences between the immediate interior and exterior of the boundary. It is proportional to the inverse of the p parameter from the PW framework, Section 2.1. When small it adheres more to the local features, as it increases it smooths out tending to a geodesic over the image grid.

Algorithm 3 presents the contour-based object delineation of Grabber. In Lines 1-2, it executes the trivial path-cost initialization of Equation 4.1, all paths initially trivial. The main loop (Lines 3-5) computes the minimum-cost segments from s_i to s_{i+1}, $i = 1, 2, \ldots, n-1$. It then removes s_1 from $\mathcal{L}$ in Line 6, updating its trivial path cost according to Equation 4.1, to close the contour in Line 7. Every time an optimum segment is computed, it is added to the contour in $\mathcal{L}$.

Algorithm 4 shows how to compute and concatenate optimum segments in $\mathcal{L}$. Line 1 inserts the source s in a priority queue Q and in an auxiliary set $\mathcal{U}$. Set $\mathcal{U}$ must contain all vertices inserted in Q during the algorithm in order to restore the costs of trivial paths, according to Equation 4.1, for the subsequent execution. Note that the source s is always

Algorithm 3 Contour-based segmentation algorithm

INPUT:	Graph $\mathcal{G} = (V, E, w_I)$ and anchor-points set $\mathcal{S}$.
OUTPUT:	Optimum object contour $\mathcal{L}$.
AUXILIARY:	Path-cost map C and predecessor map P.

1. **For each** $u \in V$
2. $C(u) \leftarrow \infty$
3. $\quad P(u) \leftarrow u.$
4. $\mathcal{L} \leftarrow \emptyset$
5. $i \leftarrow 1$
6. $C(s_1) \leftarrow 0.$
7. **While** $i < n$
8. $(C, P, \mathcal{L}) \leftarrow$ Optimum-Segment$(G, s_i, s_{i+1}, C, P, \mathcal{L})$.
9. $\quad i \leftarrow i + 1$
10. $\mathcal{L} \leftarrow \mathcal{L} \setminus \{s_1\}$
11. $C(s_1) \leftarrow \infty$
12. $(C, P, \mathcal{L}) \leftarrow$ Optimum-Segment$(G, s_n, s_1, C, P, \mathcal{L})$
13. Return $\mathcal{L}$.

inserted in Q with optimum cost $C(P^*(s))$, as obtained in the previous execution. The optimum segments already in $\mathcal{L}$ have lower costs than that and the remaining vertices are available to be conquered by a new segment from s. The main loop (Lines 2-17) removes the vertex u with minimum path cost from Q in Line 3. If u is the destination point t (Line 4), the algorithm concatenates the previous segments to the optimum segment from s to t (Lines 5-7), restores the trivial paths and their costs according to Equation 4.1 (Lines 8-9), and resets Q and $\mathcal{U}$ to end the loop. While u is not the destination point, the algorithm visits its adjacent vertices (Lines 12-17) to which the extended paths $\pi_u \cdot \langle u, v \rangle$ with cost $pathcost = f(\pi_u \cdot \langle u, v \rangle)$ (Line 13) may be lower than the cost $C(v) = f(\pi_v)$ of the current path π_v. If this is the case, then path π_v is replaced by $\pi_u \cdot \langle u, v \rangle$ when $P(v)$ is set to u in Line 16. The cost $C(v)$ and status of v in Q and $\mathcal{U}$ are updated accordingly (Lines 15-17).

4.1.1 Integration and Usage of Grabber

Segmentation starts with another interactive or automatic method and concludes with Grabber. When integrating it with an interactive approach, the user should change to Grabber as soon as corrections with the first method are not converging.

Figure 4.2 illustrates one example where the user draws internal and external markers (Figure 4.2(b)) to separate the object in Figure 4.2(a) from the background by using DT [36, 67].

We demonstrate in Section 4.2 that the integration of Grabber and DT can improve user interaction convergence. Figure 4.2(c) shows the result of Grabber when the algorithms of this chapter are used as proposed. Additionally, the segmentation mask, allows to extract object information — *e.g.*, Gaussian Mixture Models (GMMs) from the interior (Figure 4.2(d)) and exterior (Figure 4.2(e)) of the mask — and use that information to replace the edge-weight function $w_I(u, v)$ in Equation 4.2 by the one below

Algorithm 4 Optimum-segment finding algorithm

INPUT: Graph $\mathcal{G} = (V, E, w_I)$, source $s \in V$, destination $t \in V$, previous path-cost map C, predecessor map P, and contour set $\mathcal{L}$.
OUTPUT: Updated path-cost map C, predecessor map P, and contour set $\mathcal{L}$.
AUXILIARY: Priority queue $Q = \emptyset$, set $\mathcal{U} = \emptyset$, and temporary variable *pathcost*

```
1.  Q ← Q ∪ {s}, and U ← U ∪ {s}.
2.  While Q ≠ 0
3.      Remove u from Q, such that u = arg min_{v∈Q}{C(v)}.
4.      If u = t then
5.          While u ≠ s
6.              L ← L ∪ {u}
7.              u ← P(u).
8.          L ← L ∪ {s}.
9.          For each u ∈ U \ L
10.             C(u) ← ∞
11.             P(u) ← nil.
12.         Q ← ∅
13.         L ← ∅
14.     Else
15.         // IFT iteration
16.         For each v | (u, v) ∈ E and C(v) > C(u)
17.             pathcost ← C(u) + w_I(u, v)
18.             If pathcost < C(v) then
19.                 If v ∈ Q then Q ← Q \ {v}.
20.                 C(v) ← pathcost
21.                 P(v) ← u
22.                 Q ← Q ∪ {v}
23.                 U ← U ∪ {v}.
24. Return (C, P, L).
```

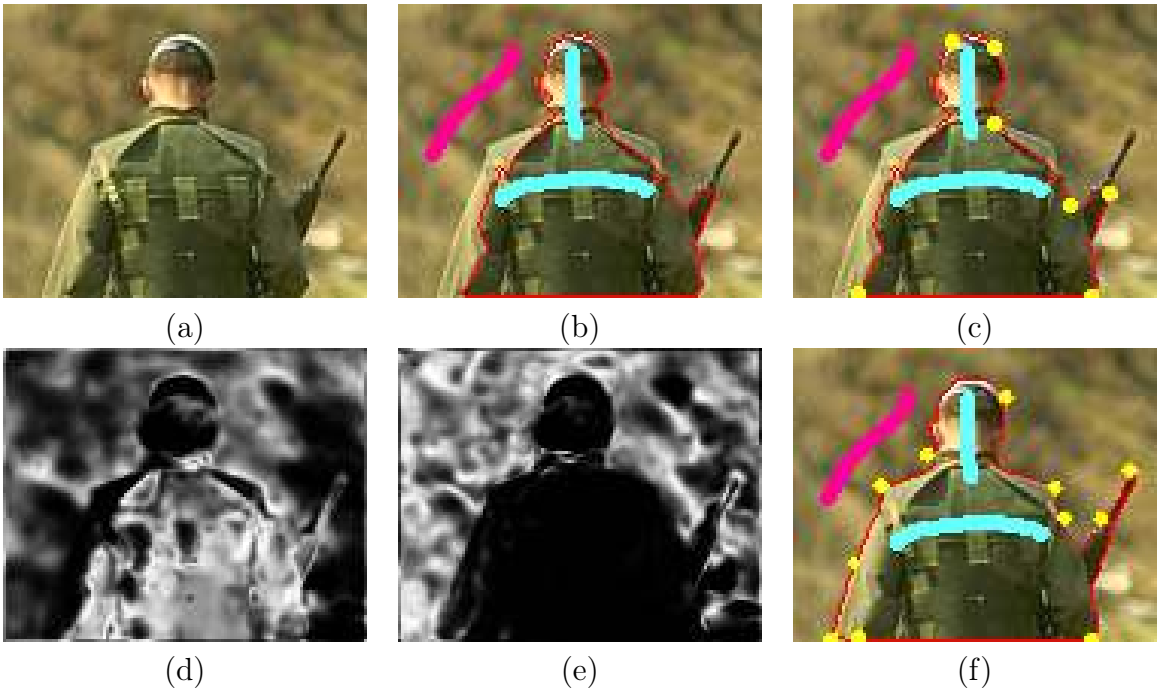


Figure 4.2: Grabber: (a) input image; (b) initial segmentation with DT [36] from object (blue) and background (pink) markers; (c) the resulting object contour from Grabber; Gaussian density maps for the (d) object and (e) background; (f) final segmentation with Grabber, after anchor-point manipulation.

$$w_G(u,v) = w_I(u,v)\exp\left(-\frac{\sum_{\lambda\in\{0,1\}}|D_\lambda(r_{u,v}) - D_\lambda(l_{u,v})|}{\sigma_G}\right) \tag{4.3}$$

where $D_\lambda(.)$ are the Gaussian density maps for each label $\lambda \in \{0,1\}$ (*i.e.* background and object), and $\sigma_G > 0$ is a hyper-parameter. Figure 4.2f shows the final result of Grabber using Equation 4.3 and after manipulating anchor points. The density maps can be updated after each iteration of the proposed technique. The choices of σ_I in w_I and σ_G are important to control the balance between local image features and the density maps.

The initial number of anchor points in Grabber is an empirical parameter, but one can explore gradient and salience information along the border of the mask to estimate more effective points. We chose points based on the simplification degree given the curvature-approximation of the original contour. When the user clicks on one anchor point v_k, the paths from the previous point v_{k-1} and to the next point v_{k+1} are redefined by running Algorithm 4 on the affected contour segment (Figure 4.3). Therefore, Grabber can limit user interaction to a single pair of contour segments of the object's border, providing the desired control over the correction process.

When a new anchor point is added by the user, the contour is not immediately redefined since the user might want to move the point to a better location. A good practice is shown in Figure 4.4. To improve a contour section and keep the rest unchanged, it is best to add points at the extremities of that section and then make further point manipulation inside the newly defined contour segment.

Object information may be obtained by different strategies. In [153], the authors

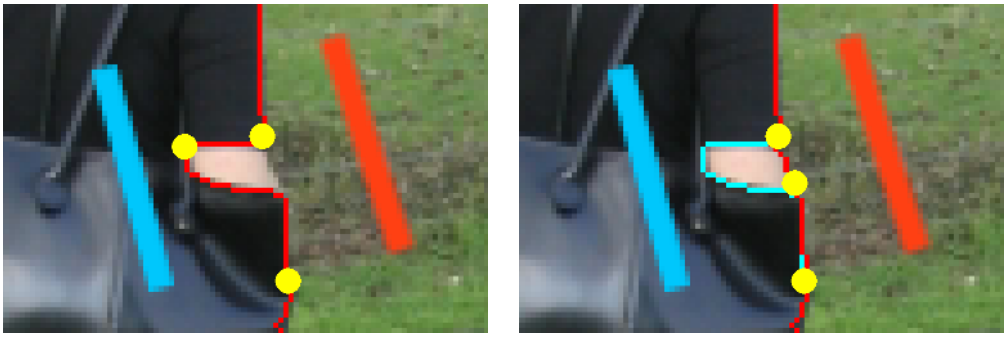
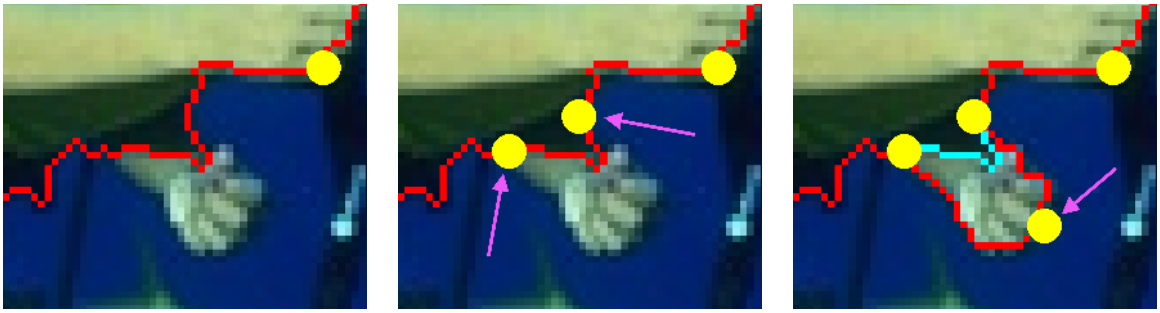


Figure 4.3: Contour-based correction of a border section. (a) Initial segmentation with DT [36] followed by Grabber. (b) The middle anchor point is moved to the object's border and Grabber fixes delineation in red. The previous segment (error) is shown in cyan.



(a) (b) (c)

Figure 4.4: A good practice in Grabber: (a) initial contour for correction; (b) points are added to both extremities of a contour section; (c) a new point is added on that section and moved to a better location; (d) the resulting correction is shown in red with the previous incorrect delineation in cyan. Note that the rest of the contour remains unchanged.

present an intelligent approach to select the pixels from internal and external markers that best discriminate the object and background for image enhancement as an object saliency map. Live Markers [154] uses this method for edge-weight estimation in the IFT algorithm. It also allows the user to draw live-wire segments on the border of the object, creating internal and external markers around them, but the segmentation is always a watershed transform from those markers. Another possibility is to use the object map predicted by the initial segmentation method from internal and external clicks during segmentation by fBRS [150]. Figure 4.5 illustrates the object maps obtained by the method in [153], when running Live Markers, by GMM from the segmentation mask of Live Markers (map D_1 in Equation 4.3), and by the network in fBRS. In general, Grabber can improve contour-based delineation when it incorporates object information in edge-weight estimation. Assuming that $O(u)$ is the object map value from some of these approaches, Equation 4.2 may be replaced by

$$w_C(u,v) = w_I(u,v) \exp\left(-\frac{|O(r_{u,v}) - O(l_{u,v})|}{\sigma_O}\right) \tag{4.4}$$

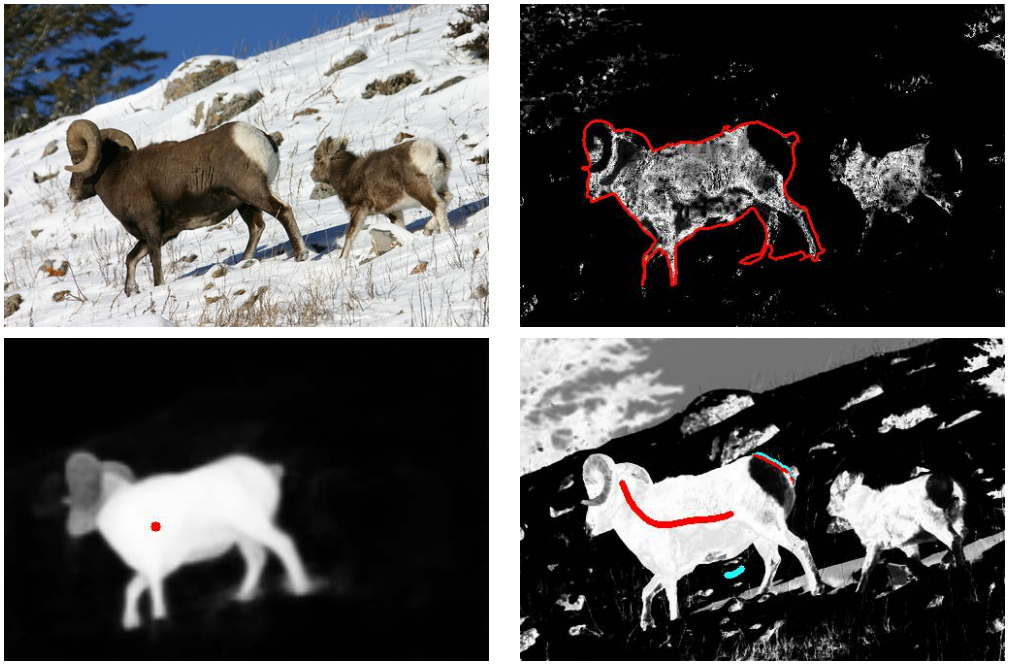


Figure 4.5: (a) Original image. Object saliency maps by (b) GMM probability from a large sheep mask (red boundary), (c) fBRS prediction from a single click (red dot), and (d) the method in [153] from background (cyan) and foreground (red) markers.

4.2 Experiments and Discussion

This section evaluates Grabber combined with two methods, fBRS [150] and DT [36]. The resulting approaches are called fBRS-Grabber and DT-Grabber. We indicate the edge weight function used in each case — *e.g.*, DT-Grabber-w_I when Grabber uses Equation 4.2.

The experiments used an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz CPU and a Titan X with 12 GB of memory. We adopted a stress experiment similar to the convergence analysis from [150]. It measures the number of interactions required to achieve a fixed threshold of Intersection over Union (IoU) [170, 111, 89, 150] — *i.e.* the number of clicks/anchor manipulations required to achieve 0.95 IoU (NoC@0.95) limited to 50 interactions. And the total of images which did not achieve the desired score given the threshold is also reported. The experiments used 100 images of the testing set of Berkeley [115] from [119], and a subset provided by [89], with 10 percent of the images (video frames) from DAVIS [133], 345 images. The ground-truths of the datasets were pre-processed, removing disconnected objects and holes to facilitate the experiments with a proposed robot for Grabber (Section 4.2.1).

ResNet101, the network used in fBRS, was pre-trained on SBD [78], and it is used as provided by [150]. The robot user for fBRS and DT is the same [170], it only adds internal and external clicks to largest error. fBRS crops a region around the object and use distance maps of the clicks to improve an object saliency map, solving segmentation by thresholding the network output. In DT, the clicks define the object as an optimum-path forest rooted at the internal ones.

We switched from fBRS (or DT to Grabber when the IoU changed less than 1%

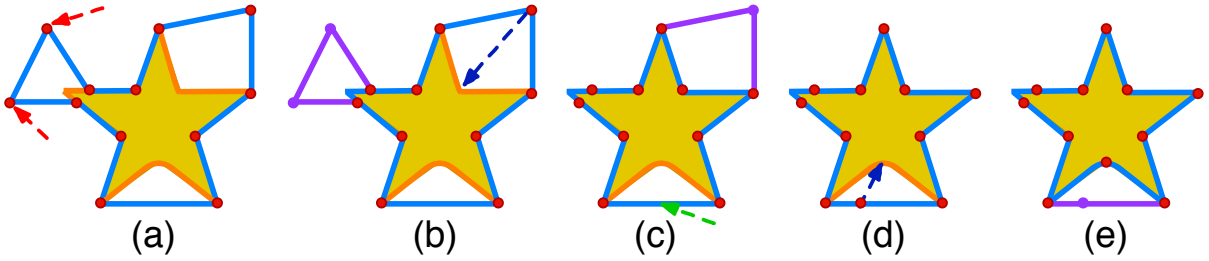


Figure 4.6: Example of robot iterations. The blue, orange and purple lines represent current, desired, and removed contour segments, respectively, and the red dots are anchors. Red, blue, and green arrows point to positions where anchors should be deleted, moved, or added. (a) Initial segmentation; (b) after deleting two anchors; (c) after moving an anchor; (d) after adding an anchor; (e) after moving the anchor added in the previous iteration.

for three consecutive interactions and Grabber started without resetting the number of interactions. Its parameters were optimized by grid-search on 150 random images from SBD [78], with range from 0.25 to 1.5 with step size of 0.25 for σ's, and 100 to 500 with a step size of 100 for ϵ, resulting in $\epsilon = 100$, $\sigma_I = 0.5$, and $\sigma_O = 0.25$. The combination with the lowest standard deviation was chosen when a tie occurred.

4.2.1 Proposed Robot User for Grabber

To simulate an user, Grabber's robot starts locating a reference pixel for each anchor. The reference pixel is the closest one to the anchor on the ground-truth border. It then locates the closest point to each reference pixel on the segmentation's border. If an anchor is not the closest point to its own reference pixel (*i.e.* competing anchors), the robot removes that anchor (Figure 4.6(a)-(b)). This behavior imitates the action of a user removing anchors from regions where Grabber should update the contour without adding new anchors. The distance used is the length of a geodesic path in the component between the GT and the segmentation mask, thus avoiding tangling the contours in non-convex shapes.

After that, the robot lists all pixels at which the GT and segmentation overlap. For every pair of consecutive intersection points between the overlapping GT and segmentation borders, it computes the area between the borders and tries to correct these regions in decreasing order of area.

Given the current largest and incorrect region, the robot adds a pair of anchors on its extremities to limit the changes to the local contour, just as the strategy shown in Figure 4.4. This is a key factor to maintain user control over the segmentation process. Then, the robot moves the furthest anchor from its reference pixel onto the GT border (Figure 4.6(b)-(c) and (d)-(e)). If there are no anchors on the segment, the robot adds one anchor and moves it (Figure 4.6(c)-(d)).

To better emulate a real user, the anchor target is around the middle of the GT contour segment on the pixel with the weakest gradient. Thus, imposing a constraint where the difference between foreground and background could be non-existent and more needed, Figure 4.6(d).

4.3 Results

Tables 4.1-4.4 show that the integration of fBRS and DT with Grabber can improve convergence and, at the same time, reduce the average response time of the interactions, and increase the mean IoU. Table 4.1 shows that Grabber can considerably reduce the number of images to which the main methods could not achieve 0.95 IoU in 50 interactions to almost half. It can also reduce the average number of interactions to achieve 0.95 IoU. Grabber was more required for DT than for fBRS, which was expected since DT is simpler. However, fBRS-Grabber-w_C performed worse than fBRS-Grabber-w_I in DAVIS and did not make much difference in Berkeley with respect to fBRS-Grabber-w_I.

Method	Dataset	# Images ≥ 50	NoC@0.95	Grabber (%)
fBRS	Berkeley	23	16.77	-
fBRS-Grabber-w_I	Berkeley	**12**	14.53	42.0
fBRS-Grabber-w_C	Berkeley	**12**	**14.02**	43.0
DT	Berkeley	31	27.71	-
DT-Grabber-w_I	Berkeley	**22**	**26.77**	71.0
fBRS	DAVIS	133	24.83	-
fBRS-Grabber-w_I	DAVIS	**93**	**24.49**	65.8
fBRS-Grabber-w_C	DAVIS	100	24.74	65.8
DT	DAVIS	163	39.07	-
DT-Grabber-w_I	DAVIS	**139**	**37.85**	91.6

Table 4.1: For each method and dataset, the number of images which it could not achieve 0.95 IoU in 50 interactions (bold indicates better), average NoC@0.95, and percentage of images that required Grabber.

Table 4.2 presents the average number of interactions until grabber initialization. Together with the average NoC@0.95 in Table 4.1, it shows that Grabber's executions are non-trivial because they stand for a large IoU gain.

Table 4.3 shows that Grabber can reduce the average response time for the interactions in both methods. Indeed, when fBRS stops, the response time of Grabber is negligible (real time). One may have an idea about that by observing the response times of DT-Grabber-w_I.

Table 4.4 shows that Grabber can increase the effectiveness of the main approach, especially in Berkeley. DAVIS is a more challenging dataset and the proposed robot for Grabber requires improvements to fully emulate a real user. It is also worth noting that

Method	Dataset	# Clicks	Dataset	# Clicks
fBRS	Berkeley	5.73 ± 4.32	DAVIS	7.48 ± 4.98
DT	Berkeley	13.21 ± 6.69	DAVIS	17.19 ± 8.27

Table 4.2: For each starting method and dataset, the average number of interactions until Grabber's initialization.

Method	Dataset	Time per Inter.	Dataset	Time per Inter.
fBRS-Grabber-w_I	Berkeley	0.218	DAVIS	0.457
fBRS-Grabber-w_C	Berkeley	0.225	DAVIS	0.458
fBRS	Berkeley	0.738	DAVIS	1.798
DT	Berkeley	0.111	DAVIS	0.287
DT-Grabber-w_I	Berkeley	0.078	DAVIS	0.200

Table 4.3: For each method and dataset, the average response time in seconds per interaction.

Method	Dataset	IoU	Dataset	IoU
fBRS	Berkeley	0.938 ± 0.023	DAVIS	0.915 ± 0.071
fBRS-Grabber-w_I	Berkeley	0.946 ± 0.014	DAVIS	0.916 ± 0.122
fBRS-Grabber-w_C	Berkeley	0.946 ± 0.012	DAVIS	0.922 ± 0.105
DT	Berkeley	0.925 ± 0.077	DAVIS	0.913 ± 0.098
DT-Grabber-w_I	Berkeley	0.941 ± 0.027	DAVIS	0.910 ± 0.125

Table 4.4: For each method and dataset, the mean and standard deviation of IoU considering only the images that required Grabber.

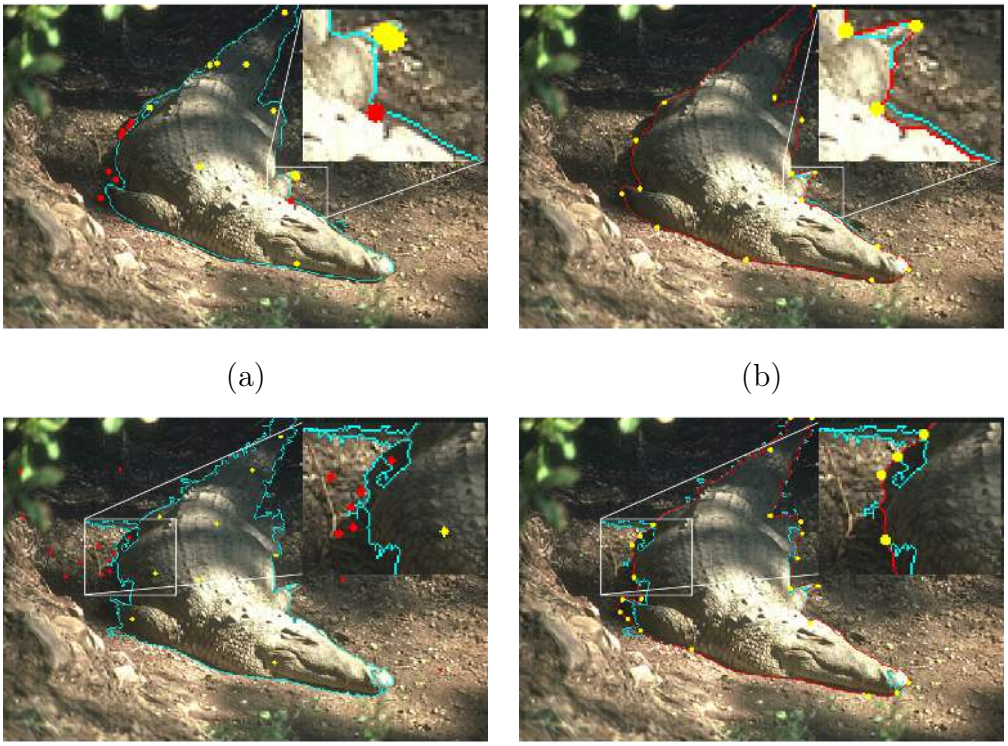
the IoU of fBRS may be 5% overestimated since the same robot is used for training and testing [27].

Finally, Figures 4.7(a)-(b) show that the Grabber's contour has higher adherence to the object's border than the segmentation of fBRS, and Figures 4.7(c)-(d) show the same for DT-Grabber.

4.4 Conclusion

The experiments demonstrate the advantages of integrating Grabber with fBRS and DT. Similar results should be observed with other methods. We expected better results with fBRS-Grabber-w_C than with fBRS-Grabber-w_I, because the use of object saliency maps can usually improve segmentation, as observed in [67].

Figures 4.8 (a)-(f) show the progress of the object saliency map along with a few interactions in fBRS. Figure 4.8(c) shows that the map of fBRS is biased to enhance certain classes (*i.e.* people). Due to crop and optimization based on the distance maps of the clicks, fBRS indeed improves the map for the object of interest — the shirt of a person. Nevertheless, even the initial map in Figure 4.8(c) affects Grabber positively since its results are better with than without the map for distinct values of ϵ (Figures 4.8(g)-(l)). This example also shows that a higher value of ϵ would improve convergence in fBRS-Grabber-w_C and fBRS-Grabber-w_I, when compared to the best value of ϵ estimated in SBD. Additionally, the user experience can be considerably improved when using Grabber to speed up convergence in interactive segmentation.



(a) (b)

(c) (d)

Figure 4.7: Crocodile image segmented with (a) fBRS (cyan contour) from internal (yellow) and external (red) clicks. Grabber provides (b) higher boundary adherence with less effort (red contour, previous contour in cyan). (c) Segmentation with DT from object (yellow) and background (red) markers. (d) Correction with Grabber (red contour, previous contour in cyan).

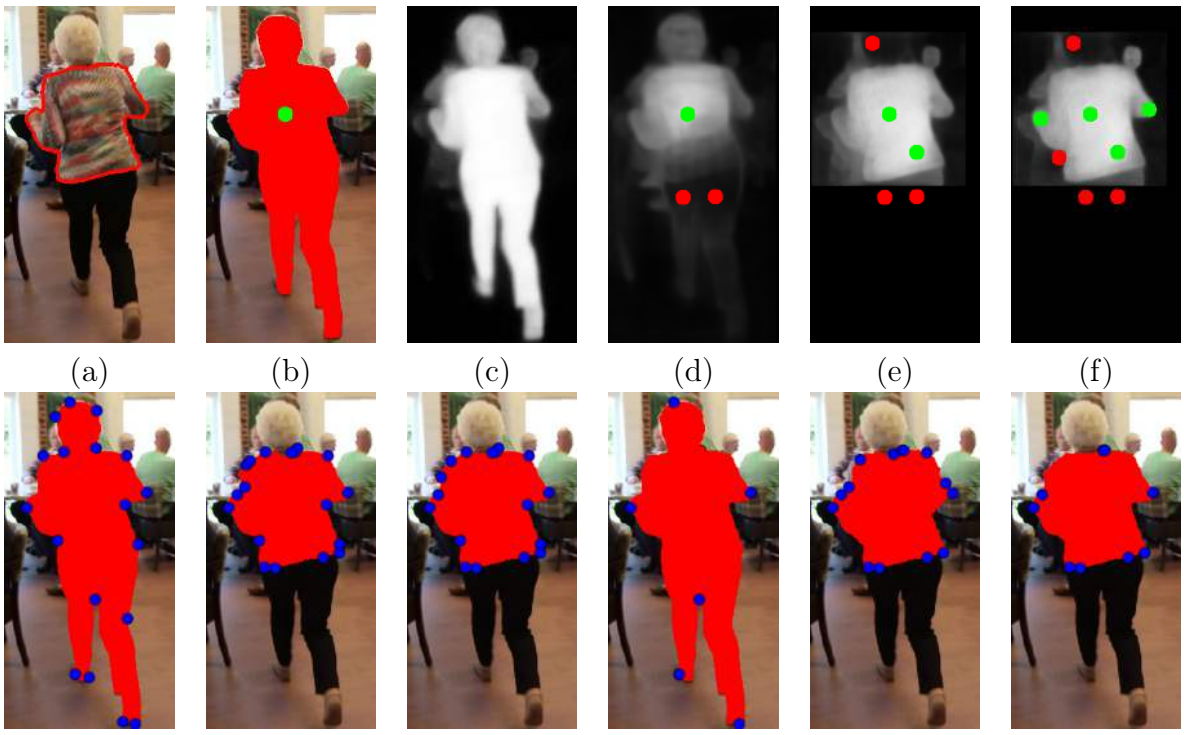


Figure 4.8: (a) Original image with ground-truth contour. (b) fBRS's segmentation with one click and (c) its saliency map biased to people. (d-f) Saliency map progress given additional clicks. (g) Grabber's anchors (blue), for $\epsilon = 100$, and its results (h) without and (i) with the saliency map from (c), requiring 1 less interaction than fBRS. (j) Grabber's anchors (blue), for $\epsilon = 1000$, and its results (k) without and (l) with the saliency map from (c), requiring 5 less interactions than fBRS.

Chapter 5

Large-Scale Segment Annotation in the Feature Space

As described in previous chapters, Convolutional Neural Networks (CNNs) can achieve excellent results on image classification, image segmentation, pose detection, and other images/video-related tasks [83, 81, 94, 40, 163], at the cost of an enormous amount of high-quality annotated data and processing power. Thus, interactive image segmentation with reduced user effort is of primary interest to create such datasets for CNNs' training. Concerning image segmentation tasks, the annotations are pixel-wise labels, usually created through interactive image segmentation methods [126, 65, 34, 143, 75, 158] or by specifying polygons in the object boundaries [144].

Recent interactive image segmentation methods based on deep learning can significantly reduce user effort by performing object delineation from a few clicks, sometimes in a single user interaction [99, 101, 173]. However, such deep neural networks do not take user input as hard constraints and so cannot provide enough user control. Novel methods can circumvent this issue by refining the neural network's weights while enforcing the correct results on the annotated pixels [89, 150, 93]. Their results are remarkable for foreground segmentation. Still, in complex cases or objects unseen during training, the segmentation may be unsatisfactory even by extensive user effort.

The big picture in today's image annotation tasks is that thousands of images with multiple objects require user interaction. While they might not share the same visual appearance, their semantics are most likely related. Hence, thousands of clicks to obtain thousands of segments with similar contexts do not sound as appealing as before.

This chapter presents a scheme for interactive large-scale image annotation that allows user to label many similar segments at once. It starts by defining segments from multiple images and computing their features with a neural network pre-trained in another domain. The user annotation is based on feature space projection, Figure 5.1, and as it progresses, the similarities between segments are updated with metric learning, increasing the discrimination among classes, and further reducing the labeling burden.

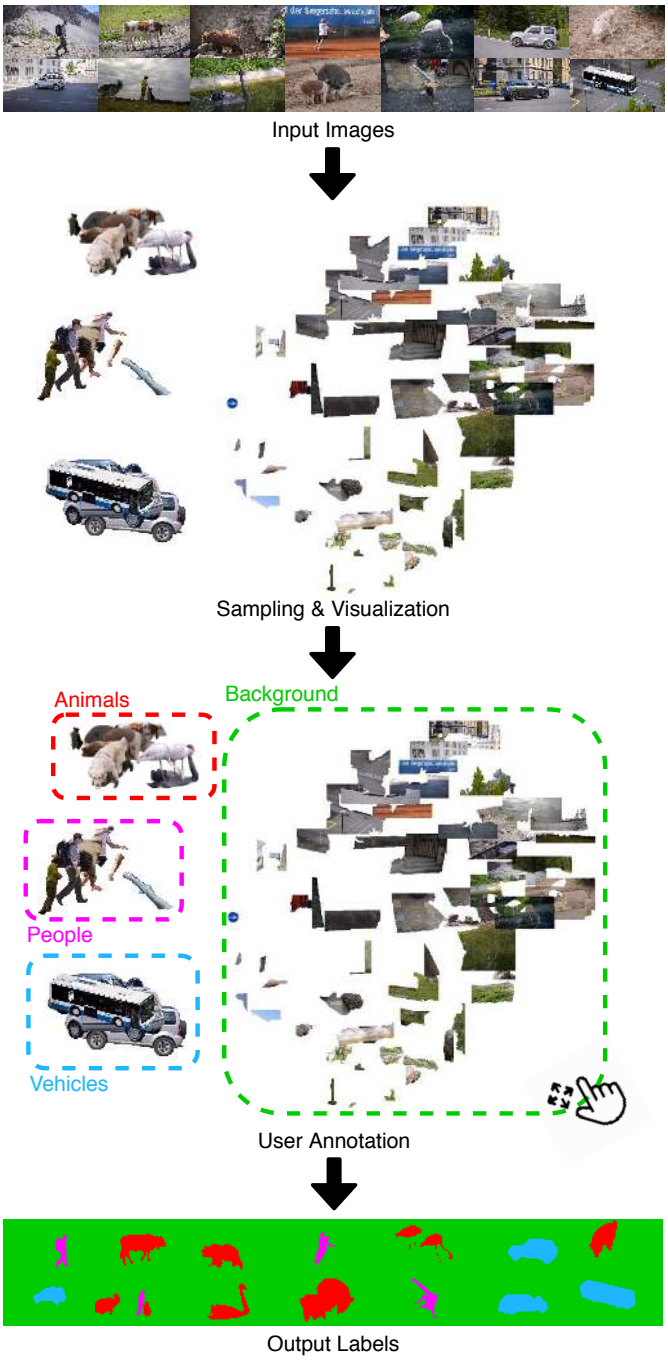


Figure 5.1: Our approach to interactive image segmentation: candidate segments are sampled from the dataset and presented in groups of similar examples to the user, who annotates multiple segments in a single interaction.

Contributions

To our knowledge, this is the first interactive image segmentation methodology that does not receive user input on the image domain. Hence, our goal is not to beat the state-of-art of interactive image segmentation but to demonstrate that other forms of human-machine interaction, notably feature space interaction, can benefit the interactive image segmentation paradigm and can be combined with existing methods to perform more

efficient annotation.

5.1 Related Works

In this chapter, we address the problem of assigning a label (*i.e.* class) to every pixel in a collection of images, denoted as image annotation in the remaining of the paper. This is related to the foreground (*i.e.* region) segmentation microtasks, where the region of interest has no specific class assigned to it, and the delineation of the object is of primary interest — in standard image annotation procedures, this is the step preceding label assignment [144].

Current deep interactive segmentation methods, from click [170, 99, 111, 93, 89, 150, 101], bounding-box [111, 27, 173] to polygon-based approaches [39, 8, 102] address this microtask of segmenting and then labeling each region individually to generate annotated data.

A minority of methods segment multiple objects jointly; to our knowledge, in deep learning, this has been employed only once [9]; And in classical methods, a hand-full could do this efficiently [49, 64, 75].

Fluid annotation [15] proposes a unified human-machine interface to perform the complete image annotation; the user annotation process starts from the predictions of an existing model, requiring user interaction only where the model lacks accuracy, further reducing the annotation effort. The user decides which action it will perform at any moment without employing active learning (AL). Hence, the assumption is that the user know and will take the actions that will decrease the annotation budget the most.

This approach falls in the Visual Interactive Labeling (VIAL) [29] framework, where the user interface should empower the users, allowing them to decide the optimal move to perform the task efficiently. Extensive experiments [28] have shown that this paradigm is as competitive as AL and obtains superior performance when starting with a small amount of annotated data.

Inside the VIAL paradigm, feature space projection has been employed for user guidance in semi-supervised label propagation [26, 25] and for object detection in remote-sensing [160]. However, its use for image segmentation has not been explored yet.

Other relevant works and their relationship with our methodology are further discussed in the next section along with the method presentation.

5.2 Proposed Method

Our methodology allows the user to annotate multiple classes jointly and requires lower effort when the data are redundant by allowing multiple images to be annotated at once. Multiple image annotation is done by extracting candidate regions (*i.e.* segments) and presenting them simultaneously to the user. The candidate segments are displayed closer together according to their visual similarity [145] for the label assignment of multiple segments at once. Hence, user interaction in the image domain is only employed when necessary — not to assign labels but to fix incorrect segments. Since, this action is the

one that requires the most user effort and has been the target of several weakly-supervised methodologies [17, 176, 177, 86] that try to avoid it altogether. Therefore, the proposed approach is based on the following pillars:

- Segment annotation problem should be evaluated as a single task [15]. While dividing the problem into microtasks is useful to facilitate the user and machine interaction, they should not be treated independently since the final goal is the complete image annotation.
- The human is the protagonist in the process, as described in the VIAL process [29], deciding which action minimizes user effort for image annotation while the machine assists in well-defined microtasks.
- Annotation in the image domain is burdensome; thus, it should be avoided, but not neglected, since perfect segmentation is still an unrealistic assumption.
- The machine should assist the user initially, even when no annotated examples are present [28], and as the annotation progresses, labeling should get easier because more information is provided.

5.2.1 Overview

The proposed methodology is summarized in Figure 5.2 and each component is described in the subsequent sections.

The user interface is composed of two primary components, the projection View and the Image View. Red contours in Figure 5.2 delineate which functionalities are present in these widgets. The projection View is concerned with displaying the segments arranged in a canvas (Figure 5.1), enabling the user to interact with it: assigning labels to clusters, focusing on cluttered regions, and selecting samples for segmentation inspection and correction in the image domain.

Image View displays the image containing a segment selected in the canvas. It is highlighted to allow fast component recognition among the other segments' contours. Samples already labeled are colored by class. This widget allows further user interaction to fix incorrect delineation, further discussed in Section 5.2.7. Segment selection also works backwardly, when a region is selected in the Image View, its feature space neighborhood is focused on the projection canvas, accelerating the search for relevant clusters by providing a mapping from the image domain to the projection canvas.

The colored rectangles in Figure 5.2 represent data processing stages: yellow represents fixed operations that are not updated during user interaction, red elements are updated as the user annotation progresses, and the greens are the user interaction modules. Arrows show how the data flows in the pipeline.

The pipeline works as follows, starting from a collection of images, their gradients are computed to partition each image into segments, which will be the units processed and annotated in the next stages.

Since we wish to cluster together similar segments, we must define a similarity criterion. Therefore, for each segment, we obtain their deep representation (*i.e.* features). Their

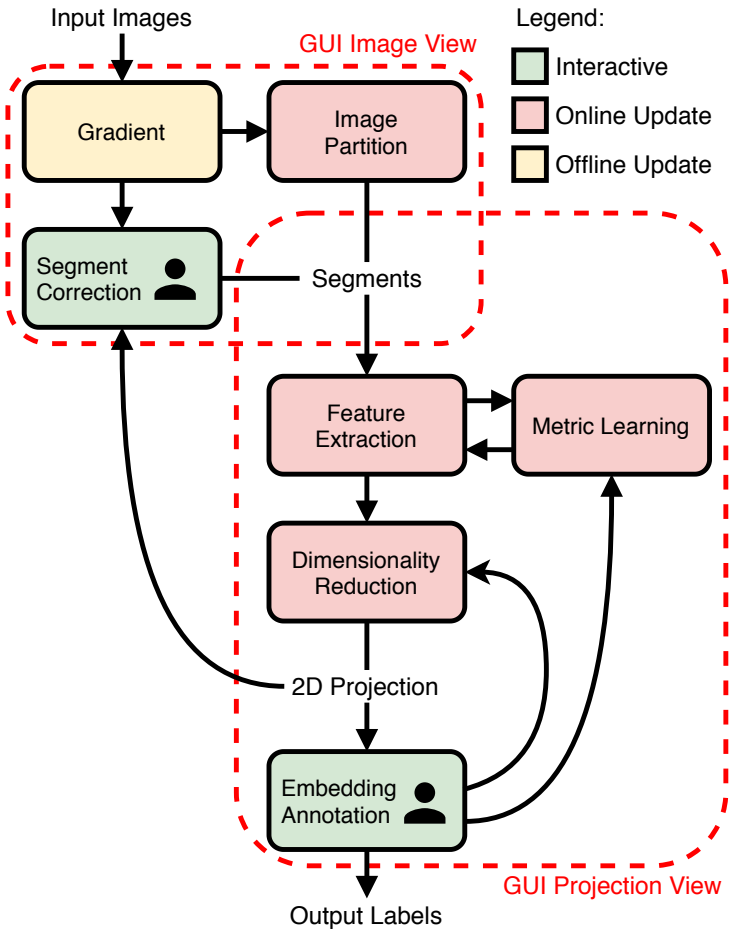


Figure 5.2: The proposed feature space annotation pipeline.

Euclidean distances are used to express this information — they are more dissimilar as they are further apart in the feature space.

The next step concerns the notion of similarity between segments as presented and perceived by the user. We propose communicating this information to the user by displaying samples with similar examples in the same neighborhood. Hence, the segments' features are used to project them into the 2D plane while preserving, as best as possible, their relative feature space distances.

The user labeling process is executed in the 2D canvas by defining a bounding-box and assigning the selected label to the segments inside it. As the labeling progresses, their deep representation is updated using metric learning, improving class separability, enhancing the 2D embedding, thus, reducing the annotation effort. We refer to [29] for a review in visual interactive labeling and [145] for interactive dimensionality reduction systems.

This pipeline relies only upon the assumption that it is possible to find meaningful candidate segments from a set of images and extract discriminant features from them to cluster together similar segments. Even though these problems are not solved yet, existing methods can satisfy these requirements, as they are validated in our study of parts (Section **??**).

5.2.2 Edge Detection and Image Partition

Edge detection and watershed-based image partition are operations of the first step of the pipeline, obtaining initial candidate segments. In the ideal scenario, the desired regions are represented by a single connected component, requiring no further user interaction besides labeling.

However, obtaining meaningful regions is a challenging and unsolved problem. Desired segments vary from application to application. On some occasions, users wish to segment humans and vehicles in a scene, while in the same image, other users may desire to segment the clothes and billboards. Thus, the proposed approach has to be class agnostic and enables the user to obtain different segment categories without effort.

The usual approach computes several solutions (*e.g.* Multiscale Combinatorial Grouping (MCG) [136]) and employs a selection policy to obtain the segments of interest. However, it does not guarantee disjoint regions, and it generates thousands of candidates, further complicating the annotation process.

Given that segments should be disjoint, and they are also task dependent, we chose to employ hierarchical segmentation techniques for this stage, described in Section 2.4 of Chapter 2. Most of the relevant literature for this problem aims at obtaining the best gradient (edge saliency) to compute segmentation.

In Convolutional Oriented Boundaries [112], a CNN predicts multiple boundaries in multiple scales and orientations and are combined into an ultrametric contour map to perform the hierarchical segmentation – Refer to [129, 130] to review the duality between contours and hierarchies.

Recently, various CNN architectures for edge estimation were developed [168, 106, 59, 80, 103] as reviewed in Section 2.5.2 of Chapter 2, where the multi-scale representation is fused into a single output image. In such methods, boundaries that appear in finer scales present lower values than the clearest ones.

While segmentation can be computed from the gradient information directly, performing it in learned features has shown to be competitive. Isaacs *et al.* [87] propose a novel approach to use deep features to improve class agnostic segmentation, learning a new mapping for the pixels values, without considering any class information and further separating their representation apart.

Other tasks also employ edge estimation to enhance their performance, notably PoolNet [105] switches between saliency object prediction and edge estimation in the training loop with the same architecture to obtain saliency with greater boundary adherence. We noticed that this approach produces less irrelevant boundaries for image segmentation; thus, we chose this method for our experimental setup.

We opted to employ the flexible hierarchical watershed framework [53, 52] for delineating candidate segments on the gradient image (*i.e.* edge detection) estimated by PoolNet [105] architecture. The hierarchical watershed allows manipulation of the region merging criterion, granting the ability to rapidly update the segments' delineation. Besides, hierarchical segmentation lets the user update segments without much effort (*e.g.* obtaining a more refined segmentation by reducing the threshold, but as a trade-off, the number of components increases). Further, a watershed algorithm can also interactively

correct the delineated segments (Section 5.2.7).

Starting from a group of N images without annotations, $\{I_1, I_2, \ldots, I_N\}$, their gradient images are computed. For each gradient G_i, $i \in [1, N]$, its watershed hierarchy is built and disjoint segments $\{S_{i,1}, \ldots, S_{i,n_i}\}$ are obtained by thresholding the hierarchy. The required parameters (threshold and hierarchy criterion) are robust and easy to be defined by visual inspection on a few images. More details about that are presented in Section 5.2.7.

5.2.3 Feature Extraction

Before presenting the regions arranged by similarity, a feature space representation where dissonant samples are separated must be computed. For that, we refer to CNN architectures for image classification tasks, without their fully connected layers [107] used for image classification.

Each segment is treated individually; we crop a rectangle around the segment in the original image, considering an additional border to not impair the network's receptive field. In this rectangle, pixels that do not belong to the segment are zeroed out. Otherwise, segments belonging to the same image would present similar representations. The segment images are then resized to 224×224 and forwarded through the network, which outputs a high-dimensional representation, $\phi_{i,j}$. In this instance $\phi_{i,j} \in \mathbb{R}^{2048}$, for each $S_{i,j}$. We noticed that processing the segment images without resizing them did not produce significant benefits and restricted the use of large batches' efficient inference.

Since our focus is on image annotation, where labeled data might not be readily available, feature extraction starts without fine-tuning and it is only optimized as the labeling progresses. Any CNN architecture can be employed, but computational performance is crucial for the user interactivity. We use the High-Resolution Network (HRNet) [163] architecture, pre-trained on ImageNet without the final fully connected layers. It is publicly available with multiple depths, and its performance is superior to other established works for image classification, such as ResNet [83]. During the development of this work, other architectures were proposed [157, 138], significantly improving the classification performance while using comparable computing resources. They were not employed in our experimental setup, but they might improve the presented results.

5.2.4 Dimensionality Reduction

Dimensionality reduction aims at reducing a feature space from a higher to a lower dimension with similar characteristics. Some methods enforce the global structure (*e.g.* PCA); other approaches, such as non-linear methods, focus on local consistency, penalizing neighborhood disagreement between the higher and lower dimensional spaces.

In some applications, the reduction aims to preserve the original features ' characteristics; in our case, we wish to facilitate the annotation as much as possible. Hence, a reduction that groups similar segments and segregates dissonant examples is more beneficial than preserving the original information.

For this, we employed a more recent approach, known as UMAP [120], for the following reasons: the projection is computed faster than t-SNE; samples can be added without

fitting the whole data; its parameters seems to provide more flexibility to control the projection scattering — enforcing local or global coherence — and most importantly, it allows to use labeled data to enforce consistency between samples of the same class while still allowing unlabeled data to be inserted.

Note that dimensionality reduction is critical to the whole pipeline because it arranges the data to be presented to the user, where most of the interaction will occur.

The 2D embedding can produce artifacts, displaying distinct segments clustered together due to the trade-off between global and local consistency even though they might be distant in the higher-dimensional feature space. Therefore, the user can select a subset of samples and interact with their local projection in a pop-up window, where the projection parameter is tuned to enforce local consistency. The locally preserving embedding (Figure 5.3) separates the selected cluttered segments (in pink) into groups of similar objects (tennis court, big households, small households, etc.), making it easier for label assignment.

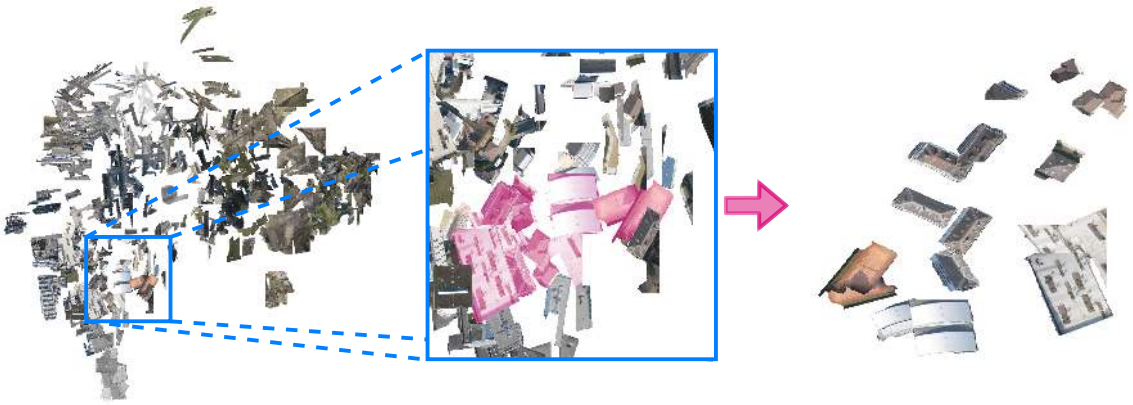


Figure 5.3: Local re-projection example: Global projection with a region highlighted in blue; A subset of segments is selected by the user, in pink; Their local embedding is computed for a simpler annotation.

On images, CNN's features obtain remarkable results consistent with the human notion of similarity between objects. Considering that an annotator evaluates images visually, data projection is our preferred approach to inform the user about possible clusters, as explained in the next section. Other visualizations must be explored for other kinds of data, such as sound or text, where the user would have difficulty to visually exploit the notion of similarity [29, 145].

5.2.5 Embedding Annotation

Each segment is displayed on their 2-dimensional coordinate, as described in the previous section. To annotate a set of segments, the user selects a bounding-box around them in the canvas, assigning the designed label. Hence, each $S_{i,j}$ inside the defined box is assigned to a label $L_{i,j}$. Finally, to obtain annotated masks in the image domain, the label $L_{i,j}$ of a segment $S_{i,j}$ is mapped to its pixels in I_i, resulting in an image segmentation (pixel annotation).

Due to several reasons, such as spurious segments or over-segmented objects, a region could be indistinguishable. Hence, when a single segment is selected in the projection, its image is displayed in the Image View as presented earlier. This action also works backwardly, the user can navigate over the images, visualize the current segments, and upon selection, the segments are focused in the projection view. Thus, avoiding the effort of searching individual samples in the segment scattering.

Additional care is necessary when presenting a large number of images, mainly if each one contains several objects, because the number of segments displayed on the canvas may impair the user's ability to distinguish their respective classes for annotation. Therefore, only a subset of the data is shown to the user initially. Additional batches are provided as requested while the labeling and the embedding progress, reducing the annotation burden.

5.2.6 Metric Learning

The proposed pipeline is not specific to any objects' class and does not require pre-training, but as the annotation progresses, the available labels can be employed to reduce user effort — less effort is necessary when the clusters are homogeneous and not spread apart.

For that, we employ a *metric learning* algorithm. Figure 5.4 shows an example of how metric learning can make clusters of a same class more compact and better separate clusters from distinct classes in our application.

In our pipeline we employed the classical large-margin loss [165], presented in Section 2.6 of Chapter 2, using our previously mentioned feature extractor network. We here followed Musgrave *et al.* [127], which showed that some novel methods are prone to overfitting and require more laborious parameter tuning.

5.2.7 Segment Correction

Since segment delineation is not guaranteed to be perfect, component correction is crucial, especially for producing ground-truth data, where pixel-level accuracy is of uttermost importance. Hence, segments containing multiple objects (under segmentation) are corrected by positive and negative clicks, splitting the segment into two new regions according to the user's cues.

Here we use a classical graph-based algorithm as we focus mainly on the feature space annotation; more modern CNN-based approaches can be employed on a real scenario.

Given an under-segmented region $S_{i,j}$, we define an undirected graph $\mathcal{G} = (V, E, w)$ where the vertices V are the pixels in $S_{i,j}$, the edges connect 4-neighbors, constrained to be inside $S_{i,j}$, and each edge $(p, q) \in E$ is weighted by $w(p, q) = (G_{i,j}(p) + G_{i,j}(q))/2$. A segment partition is obtained by the IFT [64] algorithm, from Section 2.3, for the labeled watershed operator given two sets of clicks C_{pos} and C_{neg} as defined by the user. This operation offers full control over segmentation, is fast, and improves segment delineation.

Further, the user can change the hierarchical criterion for watershed segmentation, preventing interactive segment correction in multiple images. For example, in an image overcrowded with irrelevant small objects, the user can bias the hierarchy to partition

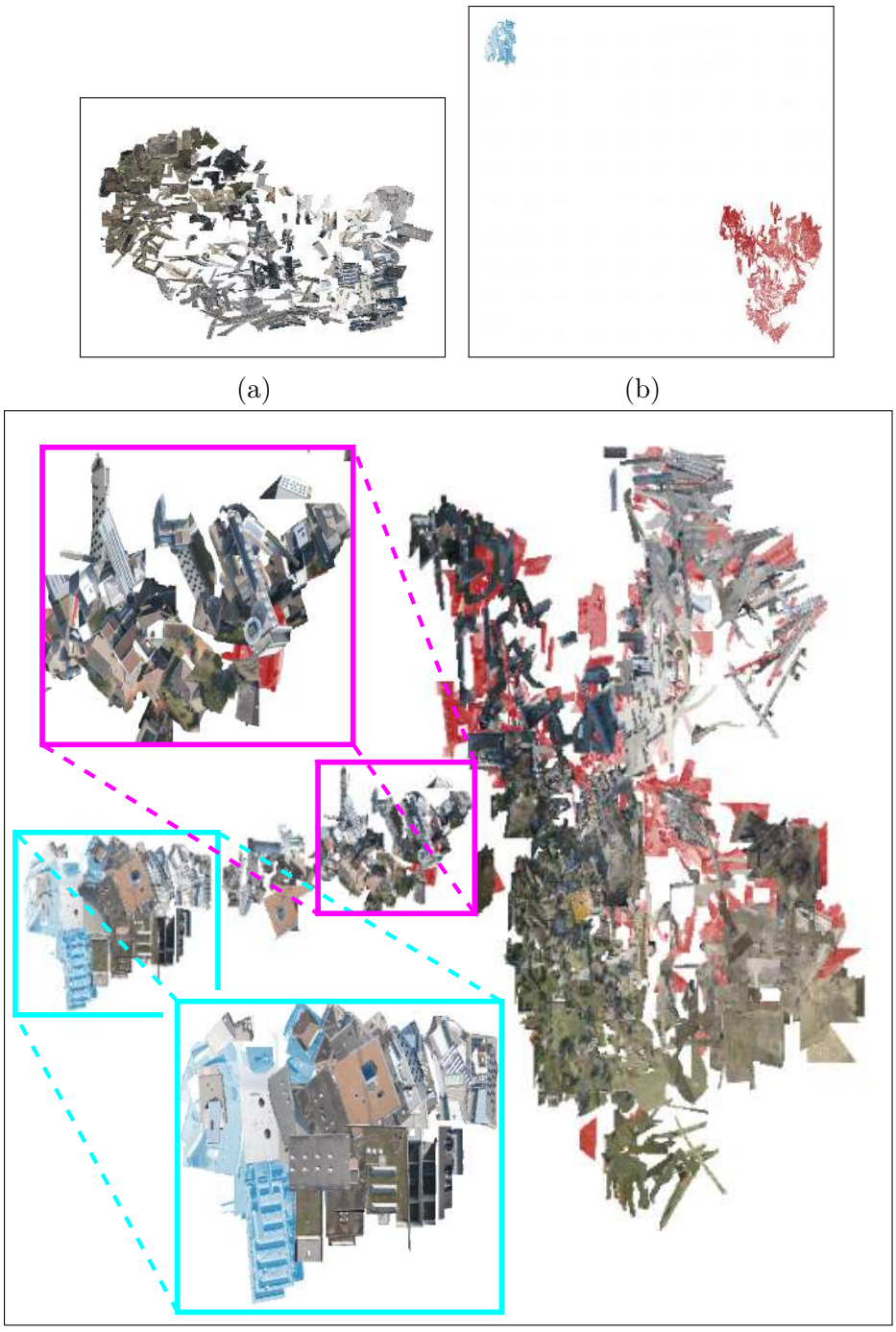


Figure 5.4: Example of metric learning in the Rooftop dataset: (a) Segment arrangement from an initial batch (10 images). (b) Displacement after labeling and employing metric learning, foreground (rooftops) in blue and background in red. (c) projection with additional unlabeled data (plus 20 images). Most rooftops' segments are clustered together (the cyan box). The magenta box indicates clusters from mixed classes and spurious segments. They suggest where labels are required the most for a next iteration of labeling and metric learning. The remaining clusters can be easily annotated.

larger objects by defining the hierarchy ordering according to the objects' area in the image domain, filtering out the spurious segments, as explained in Section 2.4.

Therefore, the Image View interface allows inspection of multiple hierarchical segmentation criteria and their result for given a threshold. The segments are recomputed upon user confirmation, maintaining the labels of unchanged segments. Novel segments go through the pipeline for feature extraction and projection into the canvas.

5.3 Experiments

This section starts by describing the datasets chosen for the experiments and the implementation details for our approach. Since our method partitions the images into segments and solves the simultaneous annotation of multiple objects by interactive labeling of similar segments' clusters, we present a study of two ideal scenarios to evaluate each main step individually. In the first experiment, the images are partitioned into perfect segments constrained to each object's mask, the rest of the pipeline is executed as proposed, evaluating the feature space annotation efficiency. Next, we assess the image partition by assuming optimal labeling, thus, only investigating the initial segmentation performance. Hence, the study of parts evaluates the limitations of the initial unsupervised hierarchical segmentation techniques, description with metric learning, and projection. Subsequently, we compare with state-of-the-art methods and present the quantitative and qualitative results.

Note that our goal goes beyond showing that the proposed method can outperform others. We point a research direction that exploits new ways of human-machine interaction for more effective data annotation.

Typically, a robot user executes the deep interactive segmentation experiments; in contrast, our study is conducted by a volunteer with experience in interactive image segmentation. Thus we are taking into account the effort required to locate and identify objects of interest.

5.3.1 Datasets

We selected image datasets from video segmentation, co-segmentation, semantic segmentation tasks, and a remote-sensing dataset from collaborator's laboratory that deviates from standard computer vision tasks.

1. *CMU-Cornell iCoSeg* [22]: It contains 643 natural images divided into 38 groups. Within a group, the images have the same foreground and background but are seen from different point-of-views.

2. *DAVIS* [133]: It is a video segmentation dataset containing 50 different sequences. Following the same procedure as in [89], multiple objects in each frame were treated as a single one, and the same subset of 345 frames (10 % of the total) was employed.

3. *Rooftop* [156]: It is a remote sensing dataset with 63 images, and in total containing 390 instances of disjoint rooftop polygons.

4. *Cityscapes* [48]: It is a semantic segmentation dataset for autonomous driving research. It contains video frames from 27 cities divided into 2975 images for training, 500 for validation, and 1525 for testing. The dataset contains 30 classes (*e.g.* roads, cars, trucks, poles), we evaluated using only the 19 default classes.

5. *VizFracSet* [1]: It is a preliminary geological dataset of rock's fractures, it consists of a single RGB image of 4044 x 2915 pixels acquired by an aerial drone from Jandaíra, Brazil. This large image was split into 40 equally sized tiles for analysis. There are two set of labels of the fractures (one was produced to fix disagreements with the existing official annotations).

5.3.2 Implementation details

We implemented a user interface in Qt for Python. To segment images into components, we used Higra [134, 2] and the image gradients generated with PoolNet [4]. We computed gradients over four scales, 0.5, 1, 1.5, and 2, and averaged their output to obtain a final gradient image. For the remaining operations, including the baselines, we used NumPy [79], the PyTorch Metric Learning package [128] and the available implementations in PyTorch [132].

For segment description with metric learning, we used the publicly available *HRNet-W18-C-Small-v1* [6] configuration pre-trained on the ImageNet dataset. In the metric-learning stage, the Triplet-Loss margin is fixed at 0.05. At each call, the embedding is optimized through Stochastic Gradient Descent (SGD) with momentum of 0.8 and weight decay of 0.0005 over three epochs with 1000 triplets each. The learning rate starts at 0.1 and, at each epoch, it is divided by 10.

We used UMAP [120] with 15 neighbors for feature projection and a minimum distance of 0.01 in the main canvas. The zoom-in canvas used UMAP with five neighbors and 0.1 minimum distance; when labels were available, the semi-supervised trade-off parameter was fixed at 0.5, penalizing intra-class and global consistencies equally.

5.3.3 Study of Parts

Our approach depends on two main independent steps: the image partition into segments and the interactive labeling of those regions. The inaccuracy of one of them would significantly deteriorate the performance of the feature space annotation for image segmentation labeling. Therefore, we present an study of parts that considers two ideal scenarios: (a) interactive projection labeling of perfect segments and (b) image partition into segments followed by optimal labeling.

In (a), the user annotates segments from a perfect image partition inside and outside the objects' masks. Hence, every segment will always belong to a single class. Table 5.1 reports the results. The Intersection over Union (IoU) distribution is heavily right-skewed, as noticed from the differences between average IoU and median IoU, indicating that most segments were labeled correctly. Visual inspection revealed that user annotation errors occurred only in small components. Table 5.1 also reports the total time (in seconds) spent annotating (user) and processing (machine), starting from the initial segment projection

Dataset	Avg. IoU	Median IoU	Time (s)
iCoSeg	95.07	99.96	5.32
DAVIS	98.54	99.97	7.82
Rooftop	95.10	99.99	3.96

Table 5.1: Average IoU, median IoU, average total processing time in seconds per image.

Dataset	Avg. IoU	Median IoU
iCoSeg	84.15	91.86
DAVIS	82.46	88.50
Rooftop	75.14	76.77

Table 5.2: Automatic segmentation results with their respective dataset.

presented to the user. It indicates that feature space projection annotation with metric learning is effective for image segmentation annotation.

In (b), we measure the IoU of the watershed hierarchical cut using a fixed parameter — the threshold of 1000 with the volume criterion. The segments were then labeled by majority vote among the true labels of their pixels. Table 5.2 shows the quality of segmentation, which imposes the upper bound to the quality of the overall projection labeling procedure if no segment correction was performed.

For reference, Click Carving [88] reports an average IoU of 84.31 in the iCoSeg dataset when selecting the optimal segment (*i.e.* highest IoU among proposals) from a pool of approximately 2000 segment proposals per image, produced with MCG [136]. In contrast, we obtain an equivalent performance of 84.15 with a fixed segmentation with disjoints candidates only.

Figure 5.5 presents example of the candidate segments on the three datasets.

5.3.4 Quantitative analysis using baselines

Since existing baselines report scores only in a very limited scenario, we executed our own experiments according to the code availability; Them being, f-BRS-B [150] (CVPR2020) and FCANet [101] (CVPR2020), both with *Resnet101* backbone, with their publicly available weights [5, 7] trained on the SBD [78] and SBD plus PASCAL VOC [61] datasets, respectively. We are not comparing with IOG [173] (CVPR2020) because we could not reproduce the results (subpar performance) with their available code and weights, and [93] (ECCV2020) is not publicly available. The results are reported over the final segmentation mask, given a sequence of 3 and 5 clicks.

Table 5.3 report the average IoU and the total time spent in annotation. For click-based methods, the interaction time was estimated as 2.4s for the initial click and 0.9s for additional clicks [23].

We achieve comparable accuracy results with state-of-the-art methods while employing less sophisticated segmentation procedures, qualitative results are presented in Figures 5.6- 5.8. Despite this, existing methods require less time to annotate these datasets;

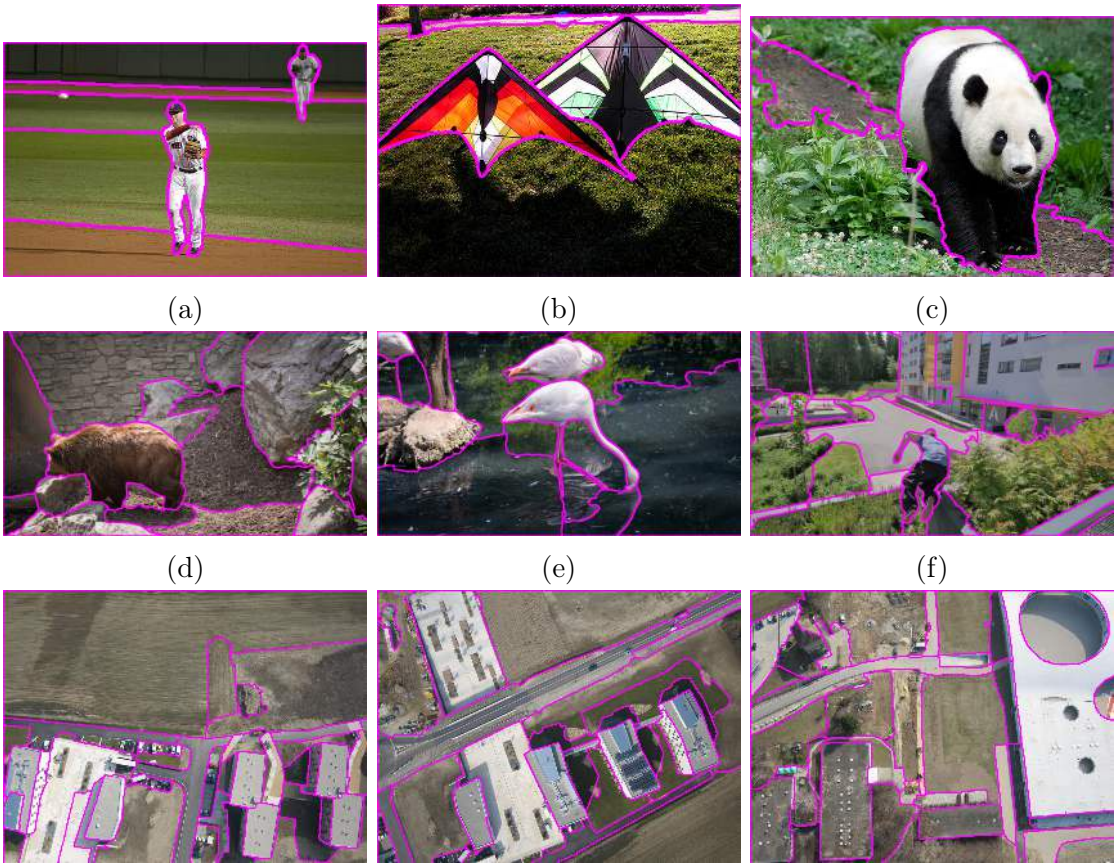


Figure 5.5: Candidate segments from study of parts. First row is from the iCoSeg dataset, second is from DAVIS and the last from Rooftop.

this is due to them being specialized in the foreground annotation microtask, while our approach wastes time annotating the background — this is exacerbated on the DAVIS dataset where a background object might be a category equal to the foreground.

The following experiment evaluates our performance on a semantic segmentation, where labeling the whole image is the final goal, not just the microtask of delineating a single object.

5.3.5 Qualitative analysis of semantic segmentation

To verify the proposed approach in a domain-specific scenario, we evaluate its performance on Cityscapes [48]. Since the true labels of the test set are not available, we took the same approach as [39], by testing on the validation set. Further, the annotation quality was evaluated on 98 randomly chosen images (about 20% of the validation set).

PoolNet was optimized based on the boundaries of the training set's true labels for five epochs using the Adam optimizer, a learning rate of $5e^{-5}$, weight decay of $5e^{-4}$, and a batch of size 8. Predictions were performed on a single scale. The fine-tuned gradient ignores irrelevant boundaries, reducing over segmentation.

The original article reports an agreement (*i.e.* accuracy) between annotators of 96%. We obtained an agreement of 91.5% with the true labels of the validation set (Figure 5.9),

Figure 5.6: The magenta boundaries delineate regions with foreground labels for the ground-truth data, our method, and the baselines using 5 clicks per image on the iCoSeg dataset. Figure (i) shows that FCANet has difficulties when segmenting multiples instances, as mentioned in their original article.

Dataset	iCoSeg		DAVIS		Rooftop	
Method	IoU	Time (s)	IoU	Time (s)	IoU	Time (s)
f-BRS (3 clicks)	79.82	4.2	79.87	4.2	62.57	4.2
f-BRS (5 clicks)	82.14	6	82.44	6	74.53	6
FCANet (3 clicks)	84.63	4.2	82.44	4.2	65.99	4.2
FCANet (5 clicks)	**88.00**	6	**86.63**	6	**81.38**	6
Ours	84.29	5.96	84.53	8.74	77.28	7.02

Table 5.3: Average IoU and time over images, except for Rooftop, where time is computed over instances. For robot user experiments, with multiple budgets (3 and 5 clicks), time was estimated according to this study [23]. Our method obtains comparable accuracy, but it spends additional time annotating foreground and background. The Cityscapes experiment shows our results on a more realistic scenario where every pixel is labeled (not a microtask).

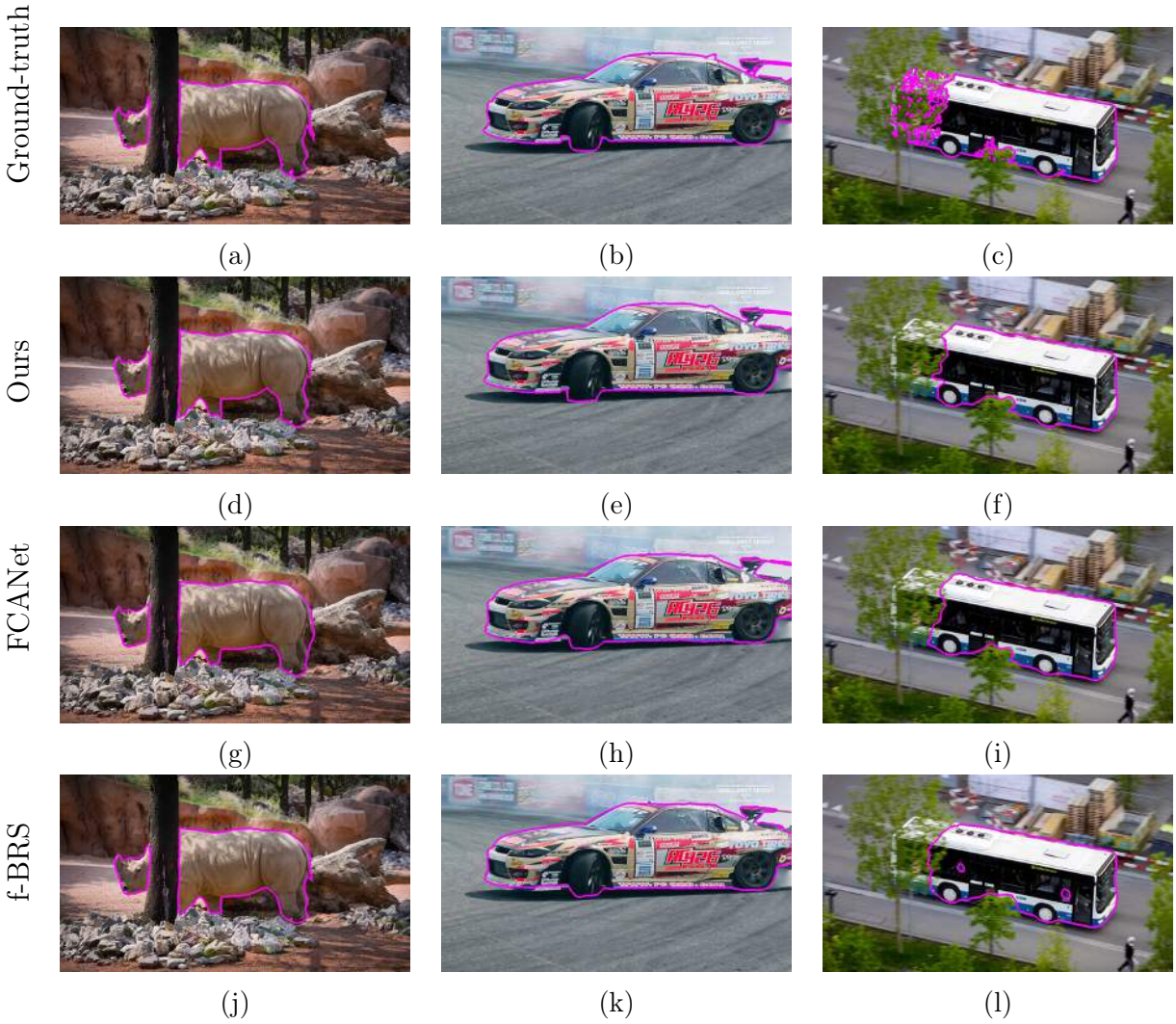


Figure 5.7: The magenta boundaries delineate regions with foreground labels for the ground-truth data, our method, and the baselines using 5 clicks per image on the DAVIS dataset.

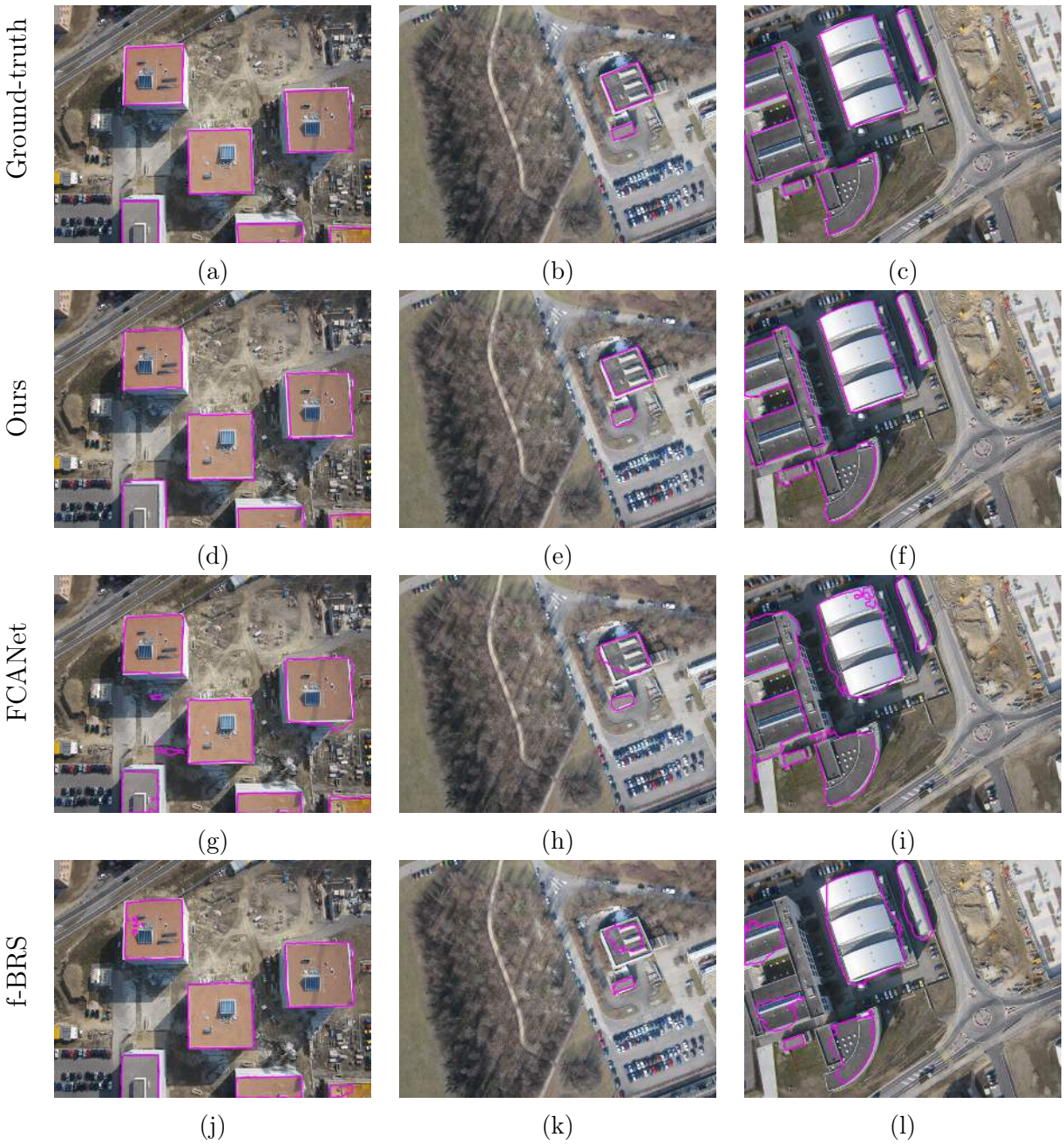


Figure 5.8: The magenta boundaries delineate regions with foreground labels for the ground-truth data, our method, and the baselines using 5 clicks per instance on the Rooftop dataset.

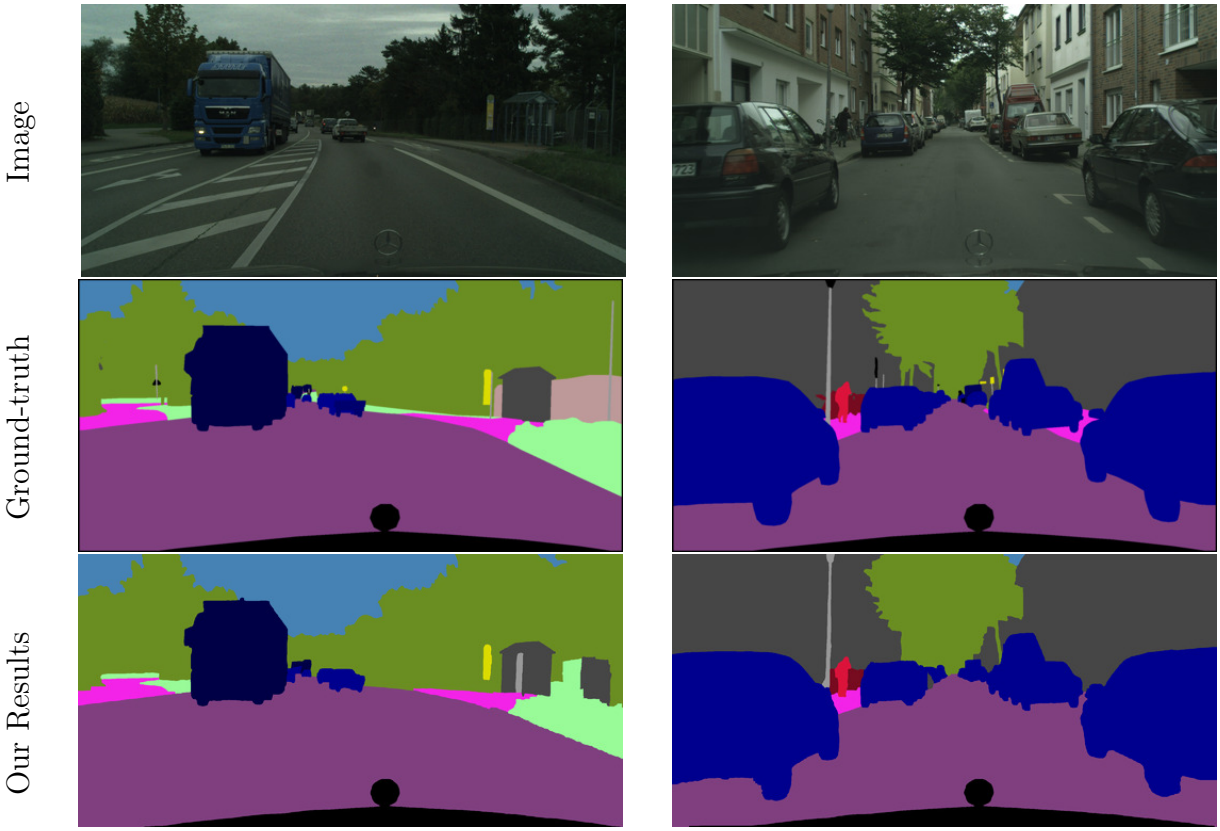


Figure 5.9: Cityscapes result, each column is a different image, row indicates which kind.

while spending less than 1.5% of their time — *i.e.* our experiment took 1 hour and 58 minutes to annotate the 98 images, while to produce the same amount of ground-truth data took approximately 6.1 days (average of 1.5 hour per image [48]) — about 74.75 times faster than the original procedure. These 98 images contain in total 6500 default classes' polygons (*i.e.*, instances). Thus, with the estimate of 6 secs per instance, FCANet would take 10 hours and 50 minutes to label them.

5.3.6 Qualitative analysis of fracture annotation

The fracture dataset [1] (Figure 5.10) deviates significantly from the standard computer vision datasets even when considering remote sensing tasks, because of the very specific landscape, and only a specialist is able to evaluate it with utmost confidence, while in common tasks most users are able to distinguish the different objects in the scenes with very little training. Moreover, the object of interest is not similar to any examples present in the training set of existing models. Thus, the performance of transfer learning is greatly impaired, this is noted in Figure 5.11, where the edge estimation, which worked in all of the previous datasets, fails to distinguish relevant edges resulting a poor watershed segmentation. Therefore, we resorted to a superpixel algorithm that operates directly in the image color space [24]. Note that the problem is not the fracture's edges are not detected by the network, is that edges that are not relevant to our problem are also detected at the same rate and intensity. Beyond the unsupervised segmentation technique the same methodology was employed.

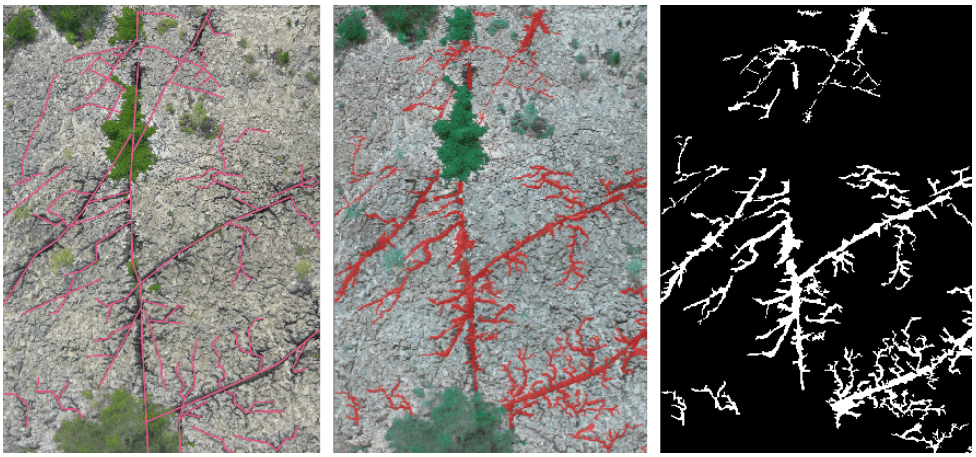
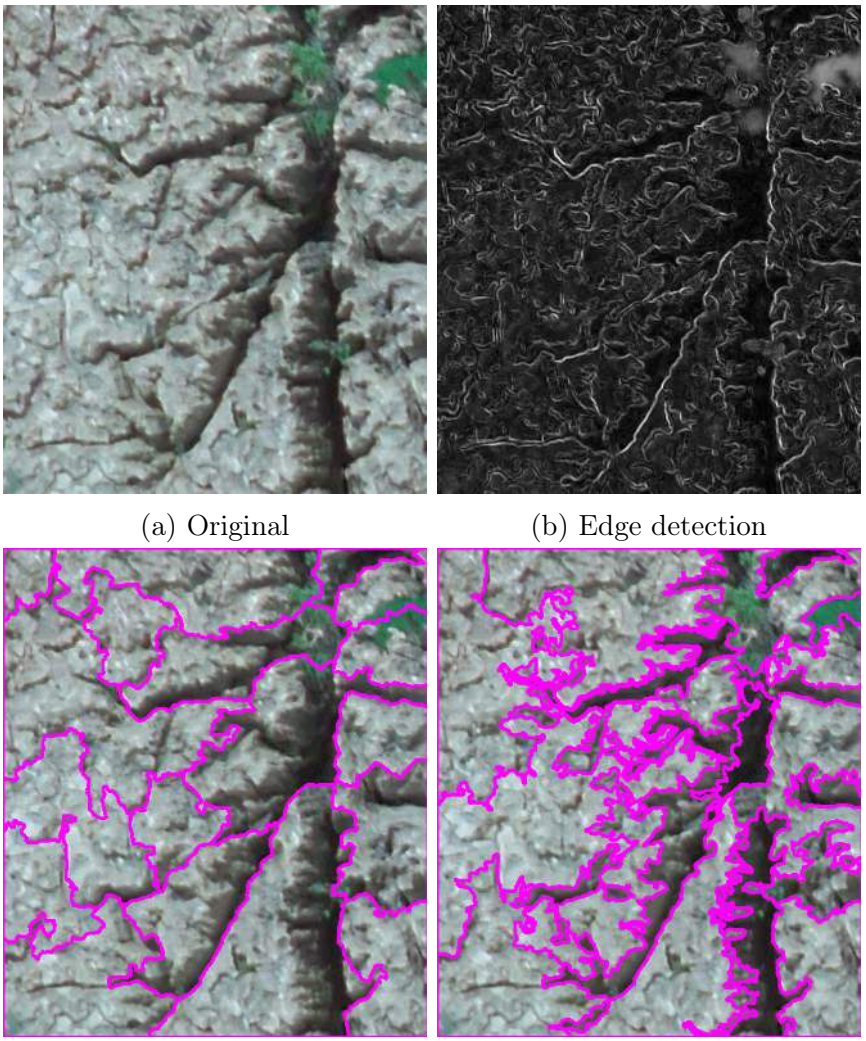


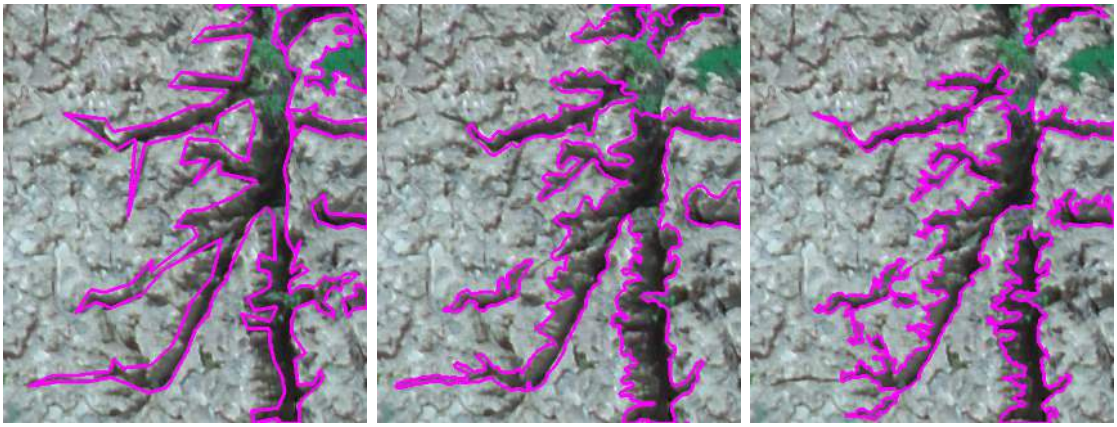
Figure 5.10: Fracture image example: (a) Original image with pink line segments over the fractures; (b) Red manual segmentation overlapping the fractures and its respective mask in (c). Image from [1].



(a) Original (b) Edge detection

(c) Hierarchical watershed (d) DISF

Figure 5.11: The edge detection (b) of the original image (a) contains several irrelevant boundaries, undermining the watershed (c) accuracy. The DISF (d) obtains a significantly better segmentation even with a lower number of superpixels.



(a) Label #1 (b) Label #2 (c) Our results

(d) Label #1 (e) Label #2 (f) Our results

Figure 5.12: Qualitative results of fracture annotations; The first row presents a tile where the methodology was successful, note that there is a disagreement between the labels and it is the same tile from the previous figure. The second row shows a case where the user missed annotating smaller regions.

We obtained an accuracy of $0.917 \pm +/- 0.048$ when comparing to the original labeling and 0.922 ± 0.046 to the other annotation. The user took 12 minutes and 6 seconds to annotate the 40 tiles, averaging 18.15 seconds per tile. Figure 5.12 presents some qualitative results.

5.4 Discussion

We presented a novel interactive image segmentation paradigm for simultaneous annotation of segments from multiple images in their deep features ' projection space. Despite employing less sophisticated segmentation methods, it achieves comparable performance with more modern approaches. Moreover, individual modules can be swapped to obtain optimal performance (*e.g.*, FCANet for segmentation correction, unifying the embedding and feature extraction with Parametric UMAP [146]).

We think that this work leads to several opportunities for combining the whole pipeline into a holistic segmentation procedure, where redundant samples are labeled at once, and annotation on the image domain occurs only when necessary. In a real scenario, we

suggest letting a leading user interact with the projection to evaluate existing annotated data, model performance (segments clustering), and, when necessary, delegating segment correction to workers, as it is currently done in existing annotation procedures, diminishing the total images annotated on the image domain.

Chapter 6

Conclusion and Future Work

In this thesis, we studied multiple approaches for interactive image segmentation, starting from graph-based methods to the recent deep learning-based techniques. We identified that additional information could be incorporated into the graphical model as the segmentation progresses by performing the segmentation through a region growing from the user cues and at each growth step updating the arc weight at the region's frontier; this approach can be implemented efficiently using dynamic programming and showed superior performance over existing graph-based techniques, Chapter 3. These results indicate that more sophisticated algorithms that reestimate its model as the segmentation progress could result in more significant improvements. For example, an extension of the power watershed where the random walk step uses the information from the currently segments regions.

We recognized that despite the incredible performance of deep interactive segmentation networks, they are not efficient for obtaining the most accurate results. Hence, we proposed Grabber (Chapter 4) to assist the finalization of the annotation when accuracy is of the utmost interest. Starting from any existing mask (*i.e.* prediction computed from a CNN), it provides anchor points along its boundaries for user editing; as the user moves the anchors, its boundaries segments adhere to its position, resulting in an accurate delineation while providing user control and leveraging prior information as a saliency map without degenerating the initial segmentation.

Last but not least, we implemented a proof of concept of a novel framework for annotating multiple segments at once, Chapter 5. Given an unlabeled set of images, it extracts multiple segments and their features; the segments are arranged in a 2D canvas and presented to the user, such that similar segments are nearby and can be labeled with a single action; as the annotation progresses, the arrangement of the segments is updated to cluster the existing labels better, further reducing the required annotation effort. Our implementation showed comparable results to existing deep learning techniques for foreground and background annotation and significantly reduced the annotation time of data for semantic segmentation tasks with greater redundancy between samples at a small cost over the final accuracy.

We think the latter method serves as a starting point for novel methodologies for annotating segments at scale. Notably, active learning could be inserted into the system to predict the label of segments where the labeling is trivial and recommending to the user

which samples would improve the predictions the most. Moreover, the approach could be explored in scenarios with simultaneous annotation from multiple users.

Bibliography

[1] A fracture dataset for computer vision and machine learning with geological curation. `https://github.com/lucaskup/VizFracSet`. Last Accessed: 2021-08-07.

[2] "Higra: Hierarchical graph analysis" package. `https://higra.readthedocs.io/en/stable/`. Last Accessed: 2020-11-06.

[3] PyTorch conv2d documentation. `https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html`. Last Accessed: 2021-02-25.

[4] Repository of "a simple pooling-based design for real-time salient object detection". `https://github.com/backseason/PoolNet`. Last Accessed: 2020-11-06.

[5] Repository of "f-BRS: Rethinking backpropagating refinement for interactive segmentation repository". `https://github.com/saic-vul/fbrs_interactive_segmentation`. Last Accessed: 2020-11-06.

[6] Repository of "high-resolution networks (HRNets for image classification". `https://github.com/HRNet/HRNet-Image-Classification`. Last Accessed: 2020-11-06.

[7] Repository of "interactive image segmentation with first click attention". `https://github.com/frazerlin/fcanet`. Last Accessed: 2021-03-10.

[8] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–868, 2018.

[9] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive full image segmentation by considering all regions jointly. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11622–11631, 2019.

[10] Mohamed Ali and David Clausi. Using the canny edge detector for feature extraction and enhancement of remote sensing images. In *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, volume 5, pages 2298–2300. IEEE, 2001.

[11] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *IEEE European Conference on Computer Vision*, pages 139–154, 2018.

[12] Cédric Allène, Jean-Yves Audibert, Michel Couprie, and Renaud Keriven. Some links between extremum spanning forests, watersheds and min-cuts. *Image and Vision Computing*, 28(10):1460–1471, 2010.

[13] W. P. Amorim, A. X. Falcão, J. P. Papa, and M. H. Carvalho. Improving semi-supervised learning through optimum connectivity. *Pattern Recognition*, 60:72–85, 2016.

[14] F. Andrade and E. V. Carrera. Supervised evaluation of seed-based interactive image segmentation algorithms. In *Sym. on Signal Processing, Images and Computer Vision*, pages 1–7, 2015.

[15] Mykhaylo Andriluka, Jasper RR Uijlings, and Vittorio Ferrari. Fluid annotation: a human-machine collaboration interface for full image annotation. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1957–1966, 2018.

[16] David Aparco-Cardenas, Pedro J. de Rezende, and Alexandre X. Falcão. Object delineation by iterative dynamic trees. In *Iberoamerican Congress on Pattern Recognition*, pages 1–10, 2021.

[17] Nikita Araslanov and Stefan Roth. Single-stage semantic segmentation from image labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4253–4262, 2020.

[18] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2010.

[19] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[20] Isabela Borlido Barcelos, Felipe Belém, Paulo Miranda, Alexandre Xavier Falcão, Zenilton KG do Patrocínio, and Silvio Jamil F Guimarães. Towards interactive image segmentation by dynamic and iterative spanning forest. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 351–364. Springer, 2021.

[21] Isabela Borlido Barcelos, Gabriel Barbosa da Fonseca, Laurent Najman, Yukiko Kenmochi, Benjamin Perret, Jean Cousty, Zenilton KG do Patrocínio, and Silvio Jamil F Guimaraes. Exploring hierarchy simplification for non-significant region removal. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 100–107. IEEE, 2019.

[22] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. Interactively co-segmentating topically related images with intelligent scribble guidance. *International Journal of Computer Vision*, 93(3):273–292, 2011.

[23] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *IEEE European Conference on Computer Vision*, pages 549–565. Springer, 2016.

[24] Felipe C Belém, Silvio Jamil F Guimaraes, and Alexandre X Falcao. Superpixel segmentation using dynamic and iterative spanning forest. *IEEE Signal Processing Letters*, 27:1440–1444, 2020.

[25] Bárbara C Benato, Jancarlo F Gomes, Alexandru C Telea, and Alexandre X Falcão. Semi-automatic data annotation guided by feature space projection. *Pattern Recognition*, 109:107612, 2019.

[26] Bárbara Caroline Benato, Alexandru Cristian Telea, and Alexandre Xavier Falcão. Semi-supervised learning with interactive label propagation guided by feature space projections. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 392–399. IEEE, 2018.

[27] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11700–11709, 2019.

[28] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):298–308, 2017.

[29] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. Vial: a unified process for visual interactive labeling. *The Visual Computer*, 34(9):1189–1207, 2018.

[30] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4380–4389, 2015.

[31] Serge Beucher. Use of watersheds in contour detection. In *Proceedings of the Int. Workshop on Image Processing*, pages 17–21, 1979.

[32] Serge Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 69–76. Springer, 1994.

[33] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[34] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[35] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[36] Jordão Bragantini, Samuel Botter Martins, Cesar Castelo-Fernandez, and Alexandre Xavier Falcão. Graph-based image segmentation using dynamic trees. In *Iberoamerican Congress on Pattern Recognition*, pages 470–478. Springer, 2018.

[37] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[38] Wallace Casaca, João Paulo Gois, Harlen C Batagelo, Gabriel Taubin, and Luis Gustavo Nonato. Laplacian coordinates: Theory and methods for seeded image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[39] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5230–5238, 2017.

[40] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[41] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

[42] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1189–1198, 2018.

[43] K. C. Ciesielski and et al. Joint graph cut and relative fuzzy connectedness image segmentation algorithm. *Medical Image Analysis*, 17(8):1046–1057, 2013.

[44] K. C. Ciesielski and et al. Path-value functions for which dijkstra's algorithm returns optimal mapping. *Journal of Mathematical Imaging and Vision*, 60(7):1–12, 2018.

[45] Krzysztof Chris Ciesielski, Jayaram K Udupa, Alexandre X Falcao, and Paulo AV Miranda. Fuzzy connectedness image segmentation in graph cut formulation: A linear-time algorithm and a comparative analysis. *Journal of Mathematical Imaging and Vision*, 44(3):375–398, 2012.

[46] Marcos AT Condori, Lucy AC Mansilla, and Paulo AV Miranda. Bandeirantes: A graph-based approach for curve tracing and boundary tracking. In *Int. Symp. on Math. Morph. and Its App. to Signal and Image Processing*, pages 95–106, 2017.

[47] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 681–685, 2001.

[48] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[49] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, 2011.

[50] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1362–1374, 2009.

[51] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Thinnings, shortest path forests, and topological watersheds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):925–939, 2010.

[52] Jean Cousty and Laurent Najman. Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 272–283. Springer, 2011.

[53] Jean Cousty, Laurent Najman, Yukiko Kenmochi, and Silvio Guimarães. Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. *Journal of Mathematical Imaging and Vision*, 60(4):479–502, 2018.

[54] Sebastian Damrich and Fred A Hamprecht. On umap's true loss function. *arXiv preprint arXiv:2103.14608*, 2021.

[55] Caio de Moraes Braz and Paulo AV Miranda. Image segmentation by image foresting transform with geodesic band constraints. In *IEEE International Conference on Image Processing*, pages 4333–4337. IEEE, 2014.

[56] Caio L Demario and Paulo AV Miranda. Relaxed oriented image foresting transform for seeded image segmentation. In *IEEE International Conference on Image Processing*, pages 1520–1524. IEEE, 2019.

[57] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[58] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[59] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *IEEE European Conference on Computer Vision*, pages 562–578, 2018.

[60] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the Int. Journal for Geographic Info. and Geovisualization*, 10:112–122, 1973.

[61] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[62] A. X. Falcão and F. P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Transactions on Medical Imaging*, 23(9):1100–1108, 2004.

[63] A. X. Falcão, L. F. Costa, and B. S. S. Cunha. Multiscale skeletons by image foresting transform and its application to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, 2002.

[64] A. X. Falcão, J. Stolfi, and R. A. de Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[65] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. A. Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing*, 60(4):233–260, 1998.

[66] Alexandre X. Falcão, Jayaram K. Udupa, and Flavio Keidi Miyazawa. An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE Transactions on Medical Imaging*, 19:55–62, 2000.

[67] Alexandre Xavier Falcão and Jordão Bragantini. The role of optimum connectivity in image segmentation: Can the algorithm learn object information during the process? In *Int. Conf. on Discrete Geometry for Computer Imagery*, pages 180–194, 2019.

[68] Enrique Fita Sanmartin, Sebastian Damrich, and Fred A Hamprecht. Probabilistic watershed: Sampling all spanning forests for seeded segmentation and semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[69] Evelyn Fix. *Discriminatory Analysis: Non-parametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[70] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1669–1680, 2018.

[71] Andrew V Goldberg, Sagi Hed, Haim Kaplan, Robert E Tarjan, and Renato F Werneck. Maximum flows by incremental breadth-first search. In *European Symposium on Algorithms*, pages 457–468. Springer, 2011.

[72] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. *Advances in Neural Information Processing Systems*, 17:513–520, 2004.

[73] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*, volume 1. MIT press Cambridge, 2016.

[74] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[75] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[76] V. Gulshan and et al. Geodesic star convexity for interactive image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3129–3136, 2010.

[77] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[78] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *IEEE International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

[79] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[80] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bidirectional cascade network for perceptual edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3837, 2019.

[81] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[83] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[84] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[85] Geoffrey Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840. Citeseer, 2002.

[86] Kuang-Jui Hsu, Yen-Yu Lin, and Yung-Yu Chuang. Deepco3: Deep instance co-segmentation by co-peak search and co-saliency detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8846–8855, 2019.

[87] Or Isaacs, Oran Shayer, and Michael Lindenbaum. Enhancing generic segmentation with learned region representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12946–12955, 2020.

[88] Suyog Dutt Jain and Kristen Grauman. Click carving: Interactive object segmentation in images and videos with point clicks. *International Journal of Computer Vision*, 127(9):1321–1344, 2019.

[89] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5306, 2019.

[90] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[91] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[92] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. OpenReview.net, 2017.

[93] Theodora Kontogianni, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. Continuous adaptation for interactive object segmentation by learning from corrections. In *IEEE European Conference on Computer Vision*, pages 579–596. Springer, 2020.

[94] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series.

[96] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[97] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[98] André V Leinio, Lucas Lellis, and Fábio AM Cappabianco. Interactive border contour with automatic tracking algorithm selection for medical images. In *Iberoamerican Congress on Pattern Recognition*, pages 748–756, 2018.

[99] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 577–585, 2018.

[100] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *IEEE International Conference on Computer Vision*, pages 2746–2754. IEEE, 2017.

[101] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. Interactive image segmentation with first click attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13339–13348, 2020.

[102] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5257–5266, 2019.

[103] Drew Linsley, Junkyung Kim, Alekh Ashok, and Thomas Serre. Recurrent neural circuits for contour detection. In *International Conference on Learning Representations*, 2019.

[104] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[105] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3917–3926, 2019.

[106] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3000–3009, 2017.

[107] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[108] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[109] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. 2018.

[110] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[111] K.-K. Maninis and et al. Deep extreme cut: From extreme points to object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[112] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):819–833, 2017.

[113] Lucy AC Mansilla and Paulo AV Miranda. Image segmentation by oriented image foresting transform with geodesic star convexity. In *International Conference on Computer Analysis of Images and Patterns*, pages 572–579. Springer, 2013.

[114] Dimitrios Marmanis, Konrad Schindler, Jan Dirk Wegner, Silvano Galliani, Mihai Datcu, and Uwe Stilla. Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:158–172, 2018.

[115] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, volume 2, pages 416–423. IEEE, 2001.

[116] S. B. Martins, T. V. Spina, C. Yasuda, and A. X. Falcão. A multi-object statistical atlas adaptive for deformable registration errors in anomalous medical image segmentation. In *SPIE on Medical Imaging: Image Processing*, pages 101332G–101332G, 2017.

[117] Samuel B Martins, Guilherme Ruppert, Fabiano Reis, Clarissa L Yasuda, and Alexandre X Falcão. A supervoxel-based approach for unsupervised abnormal asymmetry detection in mr images of the brain. In *Int. Symposium on Biomedical Imaging (ISBI)*, pages 882–885. IEEE, 2019.

[118] Samuel B Martins, Alexandru C Telea, and Alexandre X Falcao. Extending supervoxel-based abnormal brain asymmetry detection to the native image space. In *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 450–453. IEEE, 2019.

[119] Kevin McGuinness and Noel E O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.

[120] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[121] Fernand Meyer. The dynamics of minima and contours. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 329–336. Springer, 1996.

[122] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2013.

[123] P. A. V. Miranda, A. X. Falcão, and T. V Spina. Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking. *IEEE Transactions on Image Processing*, 21(6):3042–3052, 2012.

[124] P. A. V. Miranda and L. A. C. Mansilla. Oriented image foresting transform segmentation by seed competition. *IEEE Transactions on Image Processing*, 23(1):389–398, 2014.

[125] Paulo AV Miranda and Alexandre X Falcão. Links between image segmentation based on optimum-path forest and minimum cut in graph. *Journal of Mathematical Imaging and Vision*, 35(2):128–142, 2009.

[126] Eric N Mortensen and William A Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd Annual Conference On Computer Graphics and Interactive Techniques*, pages 191–198, 1995.

[127] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *IEEE European Conference on Computer Vision*. Springer, 2020.

[128] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. PyTorch metric learning, 2020.

[129] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1163–1173, 1996.

[130] Laurent Najman. On the equivalence between hierarchical segmentations and ultrametric watersheds. *Journal of Mathematical Imaging and Vision*, 40(3):231–247, 2011.

[131] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[132] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[133] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.

[134] B Perret, Giovanni Chierchia, Jean Cousty, SJF Guimarães, Yukiko Kenmochi, and Laurent Najman. Higra: Hierarchical graph analysis. *SoftwareX*, 10:100335, 2019.

[135] Benjamin Perret, Jean Cousty, Silvio Jamil Ferzoli Guimarães, Yukiko Kenmochi, and Laurent Najman. Removing non-significant regions in hierarchical clustering and segmentation. *Pattern Recognition Letters*, 128:433–439, 2019.

[136] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):128–140, 2016.

[137] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6458, 2019.

[138] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[139] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[140] L.M. Rocha, F.A.M. Cappabianco, and A.X. Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009.

[141] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[142] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[143] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Trans. on Graphics*, volume 23, pages 309–314, 2004.

[144] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

[145] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2016.

[146] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap: learning embeddings with deep neural networks for representation and semi-supervised learning. *arXiv preprint arXiv:2009.12981*, 2020.

[147] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[148] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[149] Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *IEEE International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[150] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020.

[151] Konstantin Sofiiuk, Ilia A Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. *arXiv preprint arXiv:2102.06583*, 2021.

[152] Nicholas Sofroniew, Talley Lambert, Kira Evans, Juan Nunez-Iglesias, Grzegorz Bokota, Gonzalo Peña-Castellanos, Philip Winston, Kevin Yamauchi, Matthias Bussonnier, Draga Doncila Pop, Ziyang Liu, ACS, Pam, alisterburt, Genevieve Buckley, Andy Sweet, Lorenzo Gaifas, Jaime Rodríguez-Guerra, Lukasz Migas, Volker Hilsenstein, Jordão Bragantini, Gregory R. Lee, Hector, Jeremy Freeman, Peter Boone, Alan R Lowe, Christoph Gohlke, Loic Royer, Andrea Pirré, and Hagai Har-Gil. napari/napari: 0.4.12rc2, October 2021.

[153] T. V. Spina, P. A. V. Miranda, and A. X. Falcão. Intelligent understanding of user interaction in image segmentation. *Intern. Journal of Pattern Recognition and Artificial Intelligence*, 26(02):1265001, 2012.

[154] T. V. Spina, P. A. V. Miranda, and A.X. Falcão. Hybrid approaches for interactive image segmentation using the live markers paradigm. *IEEE Transactions on Image Processing*, 23(12):5756–5769, 2014.

[155] T. V. Spina, J. Stegmaier, A. X. Falcão, E. Meyerowitz, and A. Cunha. Segment3d: A web-based application for collaborative segmentation of 3d images used in the shoot apical meristem. In *Int. Symposium on Biomedical Imaging (ISBI)*, pages 391–395, 2018.

[156] Xiaolu Sun, C Mario Christoudias, and Pascal Fua. Free-shape polygonal object localization. In *IEEE European Conference on Computer Vision*, pages 317–332. Springer, 2014.

[157] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[158] Meng Tang, Lena Gorelick, Olga Veksler, and Yuri Boykov. Grabcut in one cut. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1769–1776, 2013.

[159] J. K. Udupa and S. Samarasekera. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graphical models and image processing*, 58(3):246–261, 1996.

[160] John E Vargas-Muñoz, Ping Zhou, Alexandre X Falcão, and Devis Tuia. Interactive coconut tree annotation using feature space projections. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 5718–5721. IEEE, 2019.

[161] Tanmay Verma and Dhruv Batra. Maxflow revisited: An empirical comparison of maxflow algorithms for dense vision problems. In *BMVC*, pages 1–12, 2012.

[162] Luc Vincent. Morphological area openings and closings for grey-scale images. In *Shape in Picture*, pages 197–208. Springer, 1994.

[163] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[164] Yupei Wang, Xin Zhao, and Kaiqi Huang. Deep crisp boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3892–3900, 2017.

[165] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

[166] Steffen Wolf, Lukas Schott, Ullrich Kothe, and Fred Hamprecht. Learned watershed: End-to-end learning of seeded segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2019, 2017.

[167] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2840–2848, 2017.

[168] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1395–1403, 2015.

[169] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

[170] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 373–381, 2016.

[171] Wenxian Yang, Jianfei Cai, Jianmin Zheng, and Jiebo Luo. User-friendly interactive image segmentation through unified combinatorial user inputs. *IEEE Transactions on Image Processing*, 19(9):2470–2479, 2010.

[172] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020.

[173] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12234–12244, 2020.

[174] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.

[175] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 72–81, 2019.

[176] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised instance segmentation using class peak response. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3800, 2018.

[177] Yi Zhu, Yanzhao Zhou, Huijuan Xu, Qixiang Ye, David Doermann, and Jianbin Jiao. Learning instance activation maps for weakly supervised instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3116–3125, 2019.