Università
Ca'Foscari
Venezia

Master's Degree

in Computer Science

Data Management and Analytics
Curriculum

Final Thesis

Interactive Image Segmentation

Using Graph Transduction Games

Supervisor
Ch. Prof. Sebastiano Vascon

Assistant supervisor
Ch. Prof. Marcello Pelillo

Graduand
Riccardo Bernardi
864018

Academic Year
2020 / 2021

"Remembering that you are going to die is the best way I know to avoid the trap of thinking you have something to lose. You are already naked. "

Steve Jobs

(Stanford commencement speech, 2005)

CA'FOSCARI UNIVERSITY VENICE

Abstract

Computer Science
Department of Environmental Science, Informatics and Statistics

Master's Degree

Interactive Image Segmentation with Graph Transduction Games

by Riccardo BERNARDI

In interactive image segmentation, a target object is annotated roughly by a user and the expected outcome is the same image in which some parts of the image are labelled as foreground and others as background. Since the task is interactive we expect to constrain the result based on the a-priori informations provided by the user. The same task can be extended into a multi-class problem laying in the field of semantic instance segmentation. The solution to these kind of problems turns out to be useful in particular in medical imaging analysis where the doctor wants to extract parts of the image that are relevant(like a pack of cancer cells) leaving out others. The problem is semi-supervised given the fact that the user provides just loose bounds or scribbles on the objects of interest to be extracted. Common pipelines consist of a preliminary subdivision of the image into superpixel and then into an optimization step that performs the actual labeling. Here we are going to explain the current state of the art of the superpixels that we used to divide roughly the image and the game theory mechanics we run to let the pixels polarizing autonomously towards one of the available classes(background-foreground). The decision is taken by every superpixel based on the photometric similarity with the others and we enhance it through Leung-Malik's filters. We tested our model against many other approaches considering both standard datasets and metrics. The results shown a consistent improvement over the state of the art when loose bounding boxes are considered.

Acknowledgements

This work would have never been possible without the huge help, advices and support of my amazing supervisor Sebastiano. Others took part to this work in a different way like my grandmother Franca who supported me with love and funded my studies, believing in me and finding always good words to sustain my morale. Thanks also to my whole family composed by Giovanna, Nicola, Camilla, Antonio, Francesca, Roberta for the support showed during the hard times. Thanks to my adorable colleagues & friends, some of them(but not all, I'm sorry if I will miss someone) are: Alessandro B, Matilde S, Giada L, Lorenzo C, Filippo Z, Giorgia T, Giorgia T L, Beatrice U, Steven B, Sara V, Tommaso T, Stiven, Barbara B, Daniele B., Francesco B, Sofia B, Thomas F, Davor D, Sara F, Emilio D T, Nala A, Andrea Z, Lorenzo G, Pietro R, Benedetta D, Claudio M, Diletta O, Giorgia P, Camilla G, Leone S, Paola P, Chiara A, Chiara T, Andrea M, Riccardo G, Maddalena B, Sofia G, Manpreet K, Marco F, Andrea C.

Contents

List of Figures

List of Tables

List of Abbreviations

F	**F**oreground
BB	**B**ounding **B**oxes
B	**B**ackground
SP	**S**uper **P**ixel(s)
SLIC	**S**uperPixel **L**inear **I**terative **C**lustering
SNIC	**S**uperPixel **N**on **I**terative **C**lustering
IIS	**I**nteractive **I**mage **S**segmentation

List of Symbols

A	matrix of similarity	$\in \mathbb{R}^{n \times n}$
L	Laplacian matrix	$\in \mathbb{R}^{n} x \mathbb{R}^{n}$
σ	variance	$\in \mathbb{R}^{+}$

Dedicated to my whole family, in particular my mother and my grandmother for the constant support.

Preface

I came to know about this topic through my supervisor and it became clear that it was of interest. The development of this thesis took months and months of experiments, many were fruitful but others like the convex hull were a failure but overall this was an amazing journey. I personally discovered a new field correlated with my previous interests that in general are the AI field. During this period my supervisor was fundamental for the research to happen since he encouraged me to try new approaches and to validate every idea. Every time I mentioned him a good result, he, instead of an applaud asked me to validate the idea through a visualization of the result or the process. I think this approach was the best way to teach me how to overcome moments in which you are stuck in a problem, when you don't know if the result is valid or not. Through these passages we arrived to an improvement of the state of the art.

Chapter 1

The Problem

1.1 Introduction

In interactive image segmentation, a target object is annotated roughly by a user and then is extracted a binary mask divided into foreground and background. The same task can be extended into a multi-class problem lying in the field of semantic instance segmentation. Interactive segmentation algorithms can be categorized into box-interfaced, click-interfaced or scribble-interfaced ones. A box-interfaced one receives the mask of a target object within a given bounding box by the user, instead a scribble-interfaced receives scribbles that can be given into one time, with a prepared mask, or incrementally asking for inputs iteratively until the segmentation is enough accurate. Click-interfaced will be done using the clicks in the regions added incrementally by the user. An example of the click-interfaced 1.1.

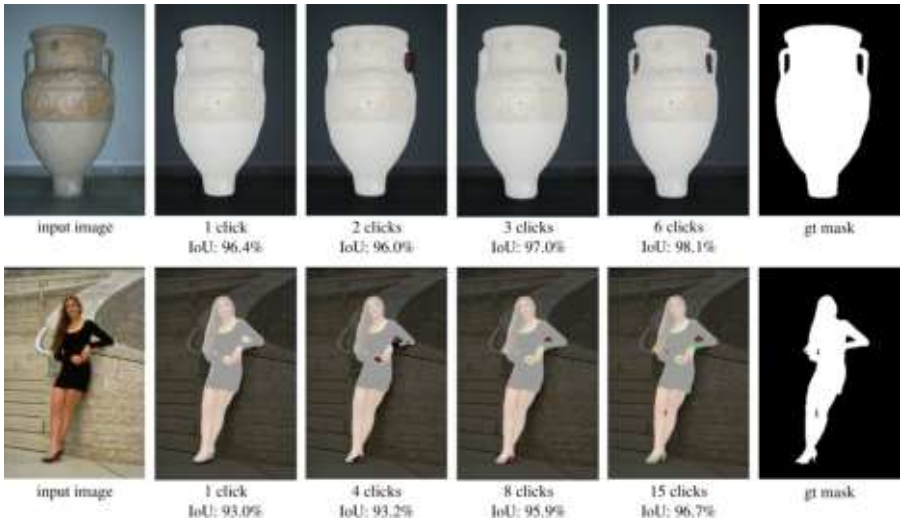


FIGURE 1.1: Example from fbrs. Image from [1]

In this image above we can see that was conducted a study on how many clicks were needed to have a quasi-perfect segmentation(>95%). But let's not be joked because in some cases the examples shown by some papers are only the best of the best, in practice the ones that are so easy to score very well. In these cases we can see that backgrounds and foregrounds were pretty different.

1.2 Statement of the problem

The Problem is about labelling correctly the foreground and the background pixels, having a good accuracy. The problem lies in the field of semi-supervised learning

since some pixels will be seeded with a label(clicked, scribbled, boxed) ones and the others will be label-less. The task will be to spread the information from seeded to unseeded in a smart way to achieve a good performance. One of the difficulties is spreading the labels in a way to achieve the maximum score non only on the core parts but also on the borders.

An example of borders those are difficult to be segmented here:

FIGURE 1.2: Example of bad vs good backgrounds

As the reader can note, in the first image its me and the background is unfavourable to segment my hand saying 3 and the algorithm can be joked by my face instead in the other case you only see the hand and the background is favourable since the gradient of the colour is high enough for a good segmentation. The second image is kindly offered by microsoft(www.lobe.com).

1.3 Significance of the study

Some of the practical applications of image segmentation lies in the fields of: Content-based image retrieval, Machine vision, Medical imaging, Object detection, Recognition Tasks, Traffic control systems, Video surveillance [2]. So the use of these kind of techniques is already broad and the significance of IIS is not in doubt. Though the meaning of this thesis is producing an advancement and improvement in the field through the use of the well known game theory. In particular the algorithm is so important in our life that without knowing it we use it almost everyday through photo editors, one of the most known is Adobe MagicWand that auto segments through our clicks, here an example:

FIGURE 1.3: Example of Adobe Magic Wand

We can recognise the Magic Wand from the characteristical dashed line.

1.4 Scope and Delimitations

The scope of the project was to able to interactively segment images for which the scribble was given as a mask a priori. After we also enlarged the scope since with the first approach the accuracy of the other models was so difficult to be beaten and we moved to the implementation of the trimaps based interactive segmentation. In this second case what is given a priori is a map that divides the image in 3 areas: background, undecided, foreground. It's pretty easy to understand that the one that is missing is the undecided part and its very difficult because the undecided part is on the borders. The implementation of this feature was easy because of the nature of the system we created, in fact whole of the functions are like plugins and can be swapped between them but also we have cockpit that registers all of these changements and runs the code as desired. What remains out of scope is he implementation of the clicks version of the interface that requires some interaction from the user and it would make a complete run not smooth because of blocking events. Another alternative to implement the click is through simulation based on the ground truth. this former approach would be better but still is not of interest since the SOTA would be difficult to be beaten. Another things that at least for now is out of scope is the use of a smarter graph neural network to make the extraction of the features not only easier but also adaptable. The current problem in fact is that the filters that are used to extract one feature vectore for each SP are handcrafted and they are approsimately 60, this is not a good idea since they are not adaptable to a different set of images and they are not weighted. A GNN would give the whole framework a boost giving the ability to learn the filters cycle after cycle and at the same time giving a weight to the filters. One of the problem at now is that since filters are equally weighted one bad filter can move drive others out. It works similarly to the Gresham's Law.

Chapter 2

Background Knowledge

2.1 Interactive Image Segmentation Models

The papers listed in this chapter are the ones we took as references and from which we took state of the art measures, metrics, datasets and from which we borrowed some techniques. Due to our background the main ideas are taken from the paper [3][4].

2.1.1 Dominant Sets for "Constrained" Image Segmentation

Short Description

The dominant set model, is used to find maximal cliques (clusters) in weighted graphs. Here the idea is to constrain a certain number of superpixels to be present in the output cluster based on the user's hints finding the most similar ones to the constrained ones. This boils down in finding the set of maximal cliques in a graph, containing the seed nodes.This model has been presented in [3][4].

Main idea

In the Constrained Dominant sets the first thing to be done is the construction of the Adjacency matrix A. This derives from the graph of superpixels usually called G and composed by a nodes set V and edges set called E. The graph is also provided with a weight function w since the graph is weighted. The set of constrained nodes that should be in the resulting cluster is called S and the method also carries an alpha parameter that should be properly tuned to find the right cluster. The alpha in particular we are going to see that should be a value larger than the largest of the eigenvalues of Adjacency matrix in which are present the nodes V except the ones constrained.

Constrained Domainat sets: In the case of constraints applied to certain pixels in the shape of scribbles the general equation of DS is enriched from 2.1:

$$maximize f(\mathbf{x}) = \mathbf{x}' A\mathbf{x} \text{ subject to } \mathbf{x} \in \Delta \tag{2.1}$$

to 2.2:

$$maximize f_S^{\alpha}(\mathbf{x}) = \mathbf{x}' \left(A - \alpha \hat{I}_S\right) \mathbf{x} \text{ subject to } \mathbf{x} \in \Delta \tag{2.2}$$

The f_S^{α} can be rewritten as 2.3:

$$f_S^{\alpha}(\mathbf{x}) = \mathbf{x}' A\mathbf{x} - \alpha \mathbf{x}_S' \mathbf{x}_S \tag{2.3}$$

Given the optimization problem we derive the KKT conditions. In the formulas here above the I_S is set as an NxN matrix in which the diagonal elements are set to 1

if they are in $V - S$. x_S is the (n k)-dimensional vector which is composed by all the components not in S.

Contributions

The paper introduced the notion of constrained dominant set and showed usefulness and applicability to the problems of co-segmentation and interactive image segmentation.

Experiments

Experiments are made on a set of labels/scribbles that is not available online but the images are the ones contained in GRABCUT [5]. The experiments are carried with scribbles, points, BB. The BB area were controlled by a Looseness parameter to identify how robust the method is.

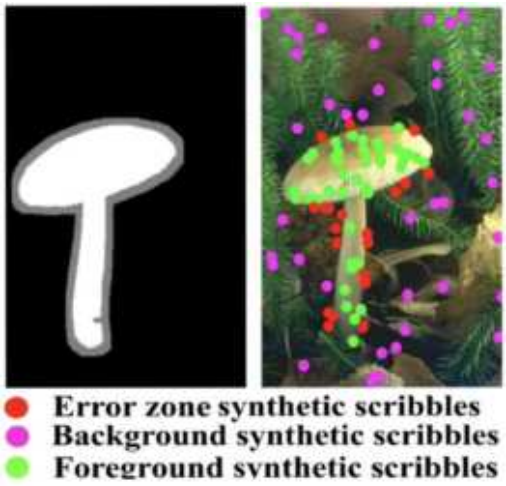


FIGURE 2.1: Mushroom segmented through syntetic points

Datasets

The images and the ground truths are taken from GRABCUT instead the labels are custom, generated through simulation of clicks.

Implementation details

Euclidean distance to calculate the distances and gaussian kernel to move to a similarity matrix, the pixels are then labelled based on the clusters extracted. It is used the Dominant Set Clustering method that given a similarity matrix as input exploits the game theory approach to give each superpixel a strategy to be played. The pixels that are labelled in blue are the ones that are playing the background move and the others are not playing. The idea is to find a cluster of one or more superpixels in an unsupervised way that can be considered a coherent cluster. The nodes are constrained with the scribbles over them.

2.1.2 "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Short Description

This task is implemented for the sake of image cutting in many programs like Adobe photoshop, paint and others under commercial names like Magic Wand or Intelligent Scissors. The idea is that given just few inputs from the user you should be able to cut the foreground out of the image. The approach involves the iterative use of graph cut algorithm to extract a part of the foreground at each iteration. The paper is [6].

Main idea

The idea is to create a model of intensities over some pixels of interest that have been bounding-boxed. The way the generative model is found is through the learning of parameters of a GMM-Gaussian Mixture Model. The idea is polarizing the pixels into only two categories, background and foreground, to do this the pixels are put in relation one with the others through a graph in which the weight is calculated bases on the colour intensity. Some of the points of the graph will be part of the foreground and other of the background, the idea is to find the minimum cut that divides the graph in the point of minimum energy. The approach is iterative and after that the bounding box is delimited then the user can refine the segmentation through scribbling.

Motivations

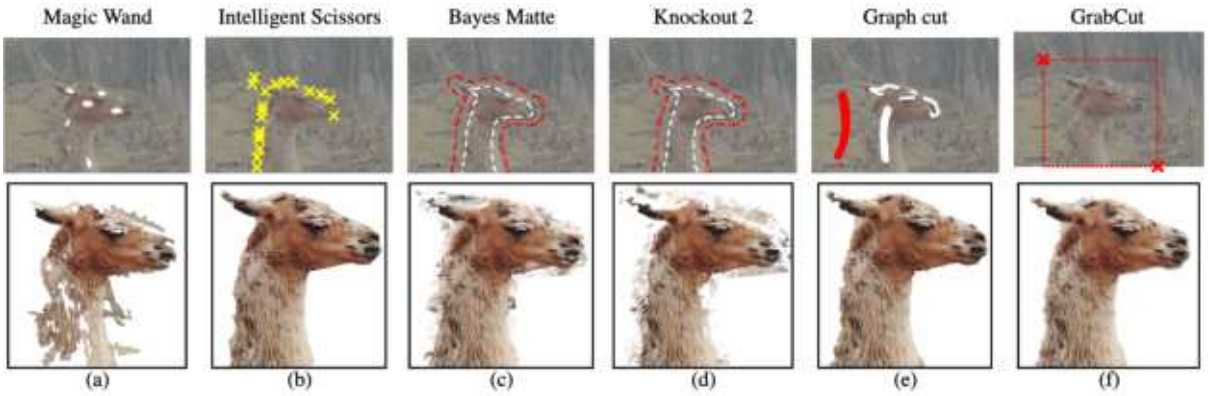
This paper differently from the other put the point on the segmentation for the background extraction for the image modification and editing, so the focus is the state of the art of the current software that are already segmenting images for the sake of advertising, photoshopping etc. In the other papers you can see that the focus is the extraction of organs or tumours from fMRI images.

Contributions

The main contribution is the idea of iterating an already well-known algorithm that performs well and the addition of a gaussian mixture model that encircles the intensities of a bunch of pixels to help the algorithm discriminate between background and foreground.

Experiments

Experiments are carried on various images, none from known datasets, but at the end of this paper this dataset become known as the GC50 and GC150 with the addition of images from VOC and Berkeley. The images are of various difficulty levels and the performances are tested against the Knockout 2 and Bayes Matte. The results highlight the fact that the grabcut algorithm outperforms the others and this can be seen from the borders precision of cutting and the smoothness of the cut. In particular the cut of grabcut is smoother and also takes smaller area that are not correlated with the foreground of interest.



Comparison of some matting and segmentation tools. The top row shows the user interaction required to complete the segmentation or matting process: white brush/lasso (foreground), red brush/lasso (background), yellow crosses (boundary). The bottom row illustrates the resulting segmentation. GrabCut appears to outperform the other approaches both in terms of the simplicity of user input and the quality of results. Original images on the top row are displayed with reduced intensity to facilitate overlay; see fig. 1. for original. Note that our implementation of Graph Cut [Boykov and Jolly 2001] uses colour mixture models instead of grey value histograms.

FIGURE 2.2: Comaparison of Grabcut vs others. Image taken from the paper.

Datasets

Unknown initially but after some time it became a benchmark dataset known as grabcut(50, because 50 images) and grabcut(150) after some additions of other authors. The GC150 is built upon VOC and Berkeley datasets.

2.1.3 Interactive Image Segmentation using Label Propagation through Complex Networks

Short Description

The proposed algorithm is divided into two stages: 1) input image is reduced to one ninth of its original size using bicubic interpolation, then a network is built with each node representing a pixel in the downsized image 2) the full input image is used. Again, each node represents a single pixel, however the connections are made only from the pixels not confidently labeled in the first stage to the nodes representing the adjacent pixels in the image, in a grid arrangement, which considers only pixel location. Label information propagates iteratively again, only to the unlabeled nodes. The paper we are talking about is [6].

Main idea

In the first part the labels are extracted from an image with the user input, the input can be a scribble, clicks, bounding boxes in which a different color represents each class, and another color is used for the unlabeled pixels. In the first stage, this image is also resized to one ninth of its original size, but using the nearest-neighbor interpolation. This is because otherwise new colors would be introduced and mistakenly interpreted as new classes. Nodes corresponding to labeled pixels are fully dominated by their corresponding class, and their domination vectors never change. On the other hand, nodes corresponding to unlabeled pixels have variable domination vectors. They are initially set in balance among all classes. Then, the iterative label propagation process takes place. At each iteration t, each unlabeled node gets contributions from all its neighbors to calculate its new domination levels. The iterative

process stops when the domination vectors converge. When the first stage finishes, most pixels are completely dominated by a single class. The exceptions are usually the pixels in classes that are borders since they are more difficult. In the second stage, nodes that were not labeled in the first stage continue to receive contributions from their neighbors. In the second stage a new graph is built, in which every pixel in the input image becomes a node without making any resizing operation. Each node v_i corresponding to an unlabeled pixel ($x_i \in$ XU) is connected to the 8 nodes v_j representing the adjacent pixels in the original image. For pixels in the image borders, which have only 3 or 5 adjacent pixels. In the second phase, neighbors are defined only by location.

Contributions

The key contributions of this paper are: sharing annotations across regions, focusing annotator effort on the biggest errors across the whole image, and a pixel-wise loss for Mask-RCNN that lets regions compete on the common image canvas. The paper have shown that on the COCO panoptic challenge dataset, the proposed technique improves the trade-off between annotation cost and quality.

Experiments

For quantitative results, the proposed method is applied to the 50 images of the Microsoft GrabCut dataset. The dataset come with scribbles or trimaps. Black color represents the background, ignored by the algorithm; dark gray is the labeled background; light gray is the unlabeled region, which labels are estimated by the proposed method; and white is the labeled foreground, which generates the foreground class particles. The error rates are computed as the ratio of the number of incorrectly classified pixels to the total amount of unlabeled pixels.

Datasets

The dataset used is the GrabCut for the experients and the COCO for some tests that didn't ended up in the Experiments section

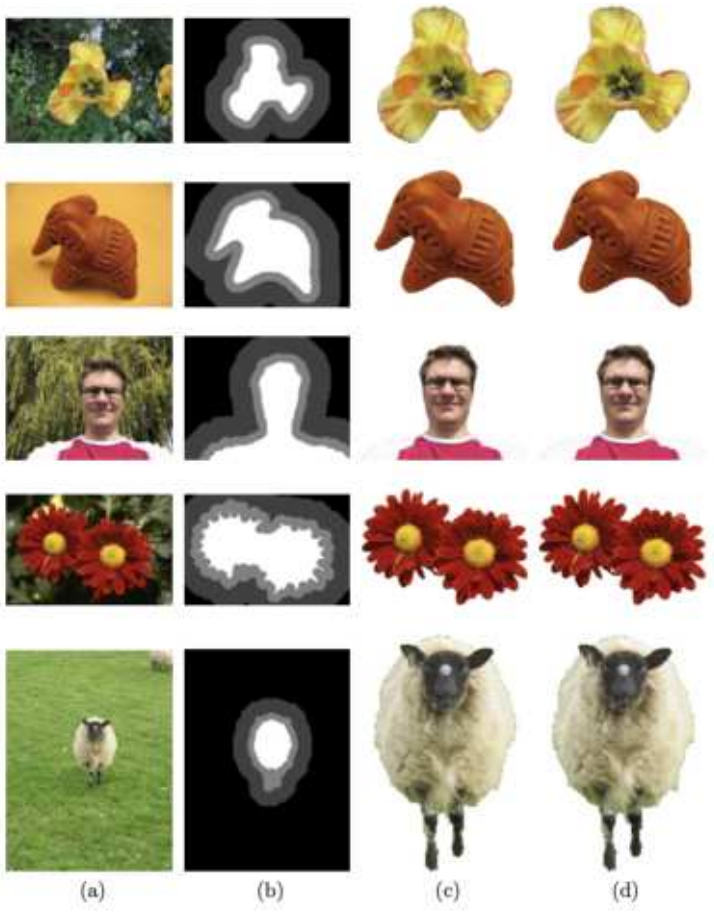
2.1.4 f-BRS: Rethinking Backpropagating Refinement for Interactive Segmentation

Short Description

f-BRS come out from the previous approach that was based on both the solution of an optimization problem and the use of a neural network. The new approach improves both the accuracy and the computational cost due to the fact that the forward-backward propagation are no more over the input but over synthesized features. The paper can be found here [1].

Main idea

The main idea lies on the fact that the re-parametrization of the network enables to work on a subset of the network(the last k layers) and to apply the optimization algorithm on the reduced subset of parameters. This method permits to balance the memory power of a neural network with the generality provided by an optimization



The proposed method applied to the Microsoft GrabCut dataset: (a) input images, (b) "trimaps" providing seed regions, (c) close-up foreground segmentation results with default parameters, (d) close-up foreground segmentation results with optimized k.

FIGURE 2.3: IIS using Label Propagation through Complex Networks

algorithm. The whole framework copes well against the computational cost of the network and the memory-less issue of an optimization algorithm.

Motivations

The field is of high interest and improving the current state of art is useful in many applied and real problems.

Contributions

Improving the state of the art in IIS with the idea of moving the computation from an entire net to a smaller part using synthetic features in behalf of the excluded part of the network.

Experiments

They tested the number of misclassified images after some fixed number of clicks that are provided by the user. They showed that their results were the smallest(the lower the better).

Results of interactive segmentation on an image from DAVIS dataset. First row: using proposed f-BRS-B (Section 3), second row: without BRS. Green dots denote positive clicks, red dots denote negative clicks.

FIGURE 2.4: fbrs method with respect to the number of clicks

Datasets

GrabCut, Berkeley, DAVIS, SBD

Implementation details

2.1.5 Segmentation by transduction

Short Description

The paper deals with the segmentation of areas inside the image that should be consistent. The regions are delimited by seeds or labels provided by the user. The method used here exploits the transductive approach in which we can see some pixels as decided and others to be decided and if one is decided it can pass its label to the nearest ones. In particular this paper uses the Laplacian graph regularizer that is tightly related to diffusion processes. The paper can be found here [7].

Main idea

The underlying assumption in graph Laplacians is that the input points are generated by a probability distribution with support on a submanifold of the Euclidean space. In transductive inference, we are going to search for a smooth fun tion such that the function is allowed to vary only on low-density regions of the input space. These methods are based on a neighborhood graph in which the nodes are the input points coming from both the training and test sets. The typical kernel is the Gaussian.

Motivations

The field is of high interest and improving the current state of art is useful in many applied and real problems.

Contributions

Firstly the introduction of the s-weighted graph Laplacian regularizer to solve the transduction problem and secondly the illustration of the link be- tween the continuous formulation of transductive inference and its discrete counterpart, introducing

a "free" parameter $\lambda = 1 - \frac{s}{2}$ as a measure of the output variation on low-density input regions.

Experiments

The experiments were conducted on the Grabcut dataset given the seeds/scribbles and on the Grabcut with the trimaps.

Top row: Initial images. Middle row: Masks provided for the segmentation. Bottom row: Results of our segmentation with $s = 1.5$. Our method performs essentially as well as the case $s = 2$ ([ ]; "random walker") with a score of 3.3%. The last column corresponds to the worst segmentation with 9.15%.

FIGURE 2.5: Results of grabcut algorithm using trimaps

Datasets

GrabCut

2.2 Graph Transduction as a Non-Cooperative Game

The theoretical formulation of the Graph Transduction Game (GTG for brevity) has been recently introduced in [8]. The idea starts from the basis of the transductive learning on undirected graphs, the solution of the label estimation is built upon game-theoretic notions, this is in contrast with more common solutions based on the laplacian regularization.

Short Description

The graph transduction is formulated in terms of a multi-player non-cooperative game where the players are the data points that take part in the game to decide their class memberships. In this setting, while the strategies played by the labeled points are already decided, as each of them knows which class it belongs to, the possible strategies available to unlabeled points are the whole set of hypotheses of being a member of one of the available classes. In this formulation the nash equilibrium comes into play being a reasonable answer to set the strategy of the unlabelled superpixels.

Main idea

The graph transduction is formulated as a non-cooperative multiplayer game and the labelling correspond to the Nash equilibria. The main idea is that a game is modelled as an interaction between players where the goal of each player is to maximize its own payoff by choosing the best action/pure strategy to play. The easiest example of this approach is the one in which we have an unweighted graph that has a connection or not with other nodes. The weight is introduced at this point in a KNN fashion so the weight of two adjacent edges is 1 instead its zero. This expands rapidly and easily to a the weighted case as can be seen. The expansion is done just by asking the user to provide a similarity measure between nodes and then a decided node can ask to an undecided one to join the playing of a certain strategy based on the similarity measure. In case a node is asked to join different strategies by different adjacent nodes the one with the higher similarity will decide the node strategy. In the game we have I_l that are labeled players and I_u that are unlabeled players. The transduction game is played just between undecided datapoints since the others are already decided and the strategy, if decided, is chosen a-priori. The payoff for each player is calculated in a polymatrix game fashion as in the formula 2.4 below:

$$u_i(x) = \sum_{j=1}^{n} x_i^T A_{ij} x_j \tag{2.4}$$

The formula can also take care in case of fixed choices and here below is the formulation. In the formula the A is the matrix of similarities, the x_i is the current playing strategy and the x_j is another playing strategy from another player. The u_i is the utility for the current player. e is the extreme mixed strategy vector in which every strategy is 0 except for the current one that is one. The formula of the utilization is here below 2.5.

$$u_i(x) = \sum_{j \in \mathcal{I}_\mathcal{U}} x_i^T A_{ij} x_j + \sum_{k=1}^{c} \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} x_i^T \left(A_{ij}\right)_k \tag{2.5}$$

The solution for the transduction game is a dynamic of replication in which we tries to play for some time certain strategies until we reach the nash equilibria, the point in which no player has any kind of advantage to move to another strategy and the game is stuck. The solution is reported here below in its discrete time formula 2.6 since it is the one we used but it also exists the one in continuos time. In the equation below x_{ih} denotes the probability that the player chooses to play its h_{th} pure strategy among all the available strategies.

$$x_{ih}(t+1) = x_{ih}(t) \frac{u_i\left(e_i^h\right)}{u_i(x(t))} \tag{2.6}$$

Motivations

Within this formulation, the Nash equilibrium concept for non-cooperative games turns out to offer a principled solution to the problem of finding a "consistent" labeling assignment[9][10].

Contributions

The greatest contribution is the linkage of two already known fields. The first field is the game theory set-up for the first time by John Nash that changed the economics studies giving new tools and the famous nash-equilibrium. The other field is the graph theory. The linkage is provided through the replicator dynamic that evolves the strategies of the players on a graph toward the nash equilibrium.

Experiments

In the experiments, the proposed approach (GTG) is compared against four well-known graph-based semi supervised learning algorithms like the Spectral Graph Transducer and Laplacian Regularized Least Squares. As it can be seen, LapRLS[11] method gives the best results for the relatively small data sets, YaleB[12] and Scene[13] instead for the other two, its performance is poor. In general, the proposed GTG[8] algorithm is either the best or the second best algorithm. Its score tends to be almost identical to LGC method in USPS, Yale-B and Scene instead with 20-news its accuracy is higher.

Datasets

YaleB, Scene, USPS, 20-news. Datasets composed by graphs.

2.3 SuperPixels

Superpixel techniques like [14][15] want to segment an image into regions by considering similarity measures defined using perceptual features. The motivation is to obtain regions that represent meaningfully the image but with far less data. The tool though can be used for compression of an image regardful of the different regions of the image. Another case of use is when we are going to use pixels to do learning on images and working at pixel level can be so expensive to make the problem untractable. In this case the superpixels are useful to reduce the dimensionality of the problem maintaning the region coherent. Superpixel techniques can be divided into two main approaches: (1) graph-based methods in which the problem is formulated as a graph cut problem and (2) region growing or clustering methods. The first approach is more difficult to be conducted and in general is preferred the second one. In the following we introduce three common superpixel models [16][15][17].

2.3.1 SLICs

The algorithm we are going to introduce here is the SLIC(simple linear iterative clustering), which works like a k-means clustering algorithm to efficiently generate superpixels. Refer here [14] to have a deeper dive.

Short Description

The SLIC algorithm computes uniform and compact superpixels, this is performed thanks to an approach based on local clustering of pixels based on the position of the pixels and its color in the CIELAB convention. It is created a new measure to improve the accuracy of the superpixel and it is controlled the compactness through a parameter to enforce a high similarity internally to each SP.

Main idea

SLIC uses clustering to create uniform size and compact superpixels that describes well the structure of objects in an image. The algorithm creates regions by three main steps: 1) creates initial regions according to a parameter that defines the desired number of superpixels 2) performs region clustering to aggregate pixels to the regions according to the similarity criteria and 3) reinforces connectivity. The algorithm starts pointing N centroids over a grid that is large enough to contain all the N SPs, after that every pixel nearby the centroid is assigned to one SP or another. The measure to assign the points is an improved euclidean in which we also take care of the color of the pixel to increase or dicrease the distance if the gradient is too high.

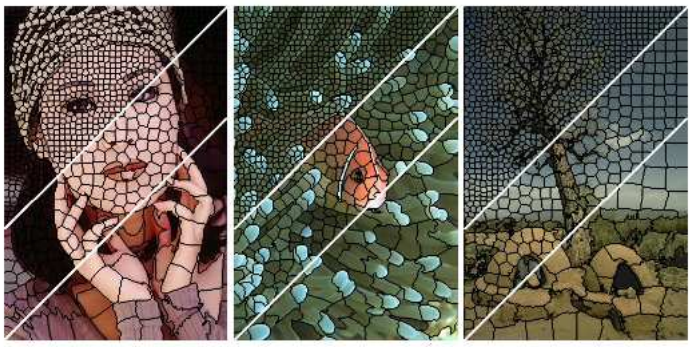


Image segmented using our algorithm into superpixels of (approximate) size 64, 256, and 1024 pixels. The superpixels are compact, uniform in size, and adhere well to region boundaries.

FIGURE 2.6: SLIC example result [16]

Motivations

The motivation for the invention of the superpixel is pretty clear: manipulating pixel by pixel an image can be impossible if the image is in high resolution. The SP approach leaves you the freedom of creating a much much smaller matrix. Remember also that an image has as many pixels as the camera is powerful instead the number of SP is a choice of the scientist. In this way you can move from an image 2000x2000 that end up in a 4.000.000x4.000.000 matrix into a much cheaper 100x100 matrix of SPs.

Contributions

An important contribution is bringing a simple and effective method with a cheap cost since it is linear. The method, in fact, is much much faster than many others like watershed[18] for example. The accuracy of the SPs is not so relevant since most of them, from the experience I gain through this thesis, are very similar in the output. Here a table 2.7.

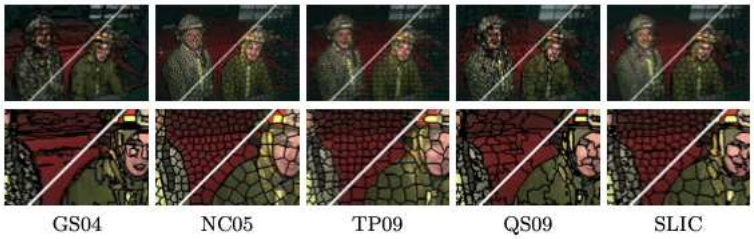
Experiments

A part from the previously stated good theoretical conditions we have also some experiments conducted to show some qualitative results. As we can see the borders of the parts segmented by the SPs are divided better with respect to the other methods.

Comparison of state of the art superpixel segmentation algorithms. N is the number of pixels in the image. GS04 and QS09 do not offer explicit control of the number of superpixels. SL08 complexity given in this table does not take into account the complexity of the boundary map computation. GS04 is $O(NlogN)$ complex but is comparable in speed to SLIC for images less than 0.5 million pixels while TP09 is also $O(N)$ complex but is 10 times slower than SLIC for 481×321 pixel images. In the case of QS09, d is a small constant (refer to [10] for details). The number of parameters listed in the table is the minimum required for typical usage.

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

FIGURE 2.7: SLIC complexity comparison result [14]



Visual comparison of the superpixels. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each. Each pair of rows show the whole segmented image and its central part blown-up respectively.

FIGURE 2.8: SLIC experiments vs others [16]

2.3.2 SNICs

[15]

SNIC is an improved version of the SLICs created by the same authors and its claimed to be faster and requiring less memory.

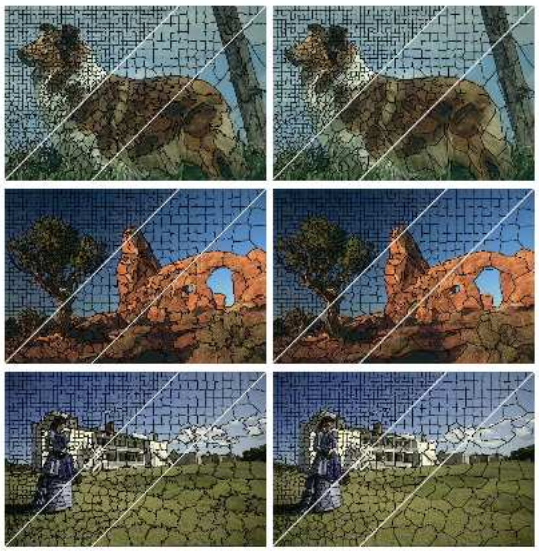
Short Description

Like SLIC, it is also utilized as initialization a method based on centroids with pixels chosen on a regular grid in the image plane. The affinity of a pixel to a centroid is measured using a distance in the five-dimensional space of color and spatial coordinates. The algorithm uses the same distance measure as the SLICs. This distance combines normalized spatial and color distances. The improvement though is elsewhere, it is based on how the pixels are attached to a centroid.

Main idea

The innovation of the SNIC algorithm is that it uses a priority queue to choose the next pixel to add to the cluster differently from its ancestor SLIC. The queue receives the nearest pixels and when it is asked to pop one pixel it is extracted the nearest one to the centroid so that the compactness is guaranteed from the principle. This allows the method also the be more efficient with respect to the ancestor that instead asks

many time to the nearest neighbors which one should be added. Each pixel, when added to the queue moves the center of the centroid and this operation happens right in that moment so that in the next addition of a pixel the movement of the centroid already happened.



Images on the left show SNIC segmentation for three different superpixel sizes. Images on the right show the corresponding SNICPOLY polygonal partitioning.

FIGURE 2.9: SNIC example result [15]

Motivations

This kind of algorithms are of large use not only for the sake of studies we are presenting in this thesis but also, for example, for remote imagery from satellites. The importance so for having good algorithms that quickly provide a compression of an image based on similarity of some pixels can be of great use for the sake of visualization of high resolution images without losing too much detail.

Contributions

The main contribution is the adding of the priority queue that speeds up the entire algorithm permitting a faster and better choice of the destination of a pixel with respect to a certain centroid.

Experiments

One of the experiments is here in image **??** and can be seen that the segments tend to be better on segmenting coherent regions in a single area. This approach is able to release more accurate superpixels that are going to require less effort of postprocessing.

Datasets

The benchmarking is done on the Berkeley 300 dataset taking into consideration both the color and grayscale groundtruth images for the range of 50 to 2000 superpixels.

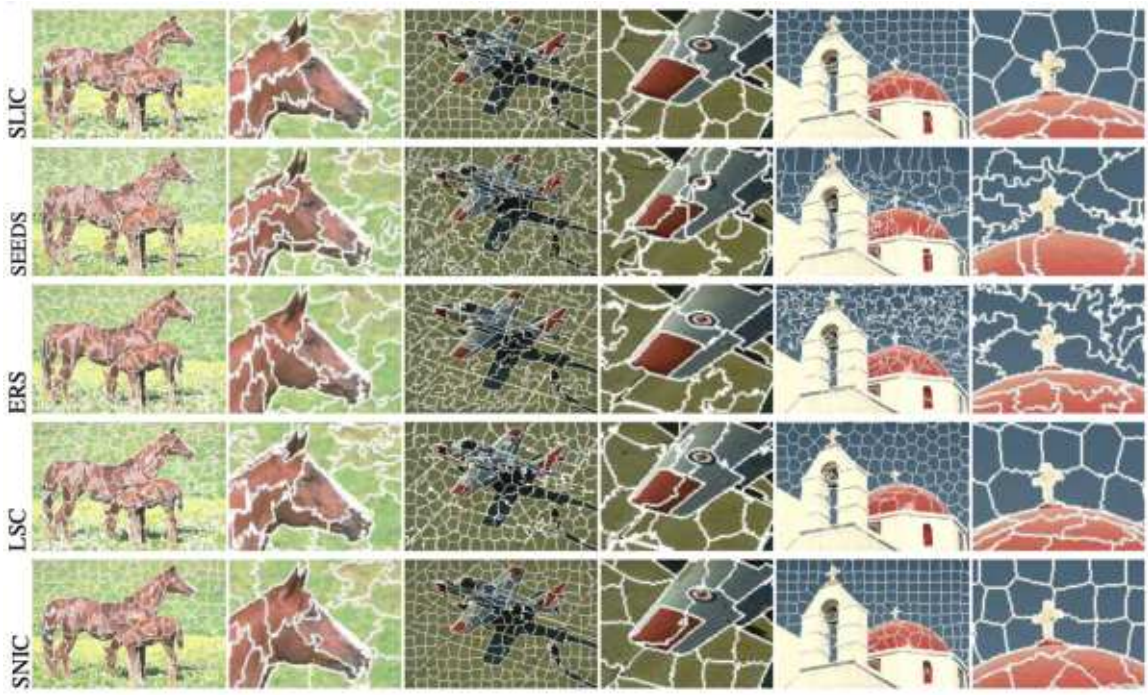


FIGURE 2.10: Experiment SNIC vs SOTA

2.3.3 SSN

[17]

SSN is a different kind of superpixels and they are added to this collection for the sake of completeness. The SSN comes out directly from the knn principle, moving the problem to a deep net making it differentiable.

Short Description

The SSN is introduced to make the usual superpixels differentiable and therefore integrable into deep learning frameworks to perform end to end pipelines. From the experiments we can note that the algorithm is not only as accurate as the other competitors but also it can pretty easily learn custom segmentations on custom datasets due to its neural network fashion.

Main idea

The thing that makes the competitors like SLIC and SNIC not differentiable is the fact that the pixel vs superpixel association, the operation in which it is matched a certain pixel to its SP centroid, is a non differentiable nearest neighbor. The key take away is that the NN operation is core and cannot be deleted instead it should be moved to a differentiable fashion. Instead of computing hard pixel-superpixel associations finding the pixel to be associated with the lower distance the authors proposed to compute soft-associations between pixels and superpixels. In particular we can dive deep and discover that instead of computing the pixel superpixel association we calculate the distance and we try to search for the minimum distance pixel to the suoerpixel optimizing a function that has the distance as exponent to the neper's numeber. Obviously this calculation could be very difficult to be be performed over

the entire image and for this reason the operation is performed only on the nearest 9 SPs of the pixels, the 8 that surround the pixel and the 9th is the one that is spatially on the same place of the pixel but can be that the gradient with the colors is so high to not fit aso well on that SP to add the pixel.

The formula before 2.7 and 2.8:

$$H_p^t = i \in \{0, \dots, m-1\} \arg\min D\left(I_p, S_i^{t-1}\right) \tag{2.7}$$

$$S_i^t = \frac{1}{Z_i^t} \sum_{p|H_p^t=i} I_p \tag{2.8}$$

And after the improvement derivability improvement 2.9 and 2.10:

$$Q_{pi}^t = e^{-D\left(I_p, S_i^{t-1}\right)} = e^{-\left\|I_p - S_i^{t-1}\right\|^2} \tag{2.9}$$

$$S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^{n} Q_{pi}^t I_p \tag{2.10}$$

As can be seen we pass from a non differentiable to a differentiable operation once the argmin is substituted by a neper's number to the minus norm of the distance between a certain pixels and the elected superpixel.

Motivations

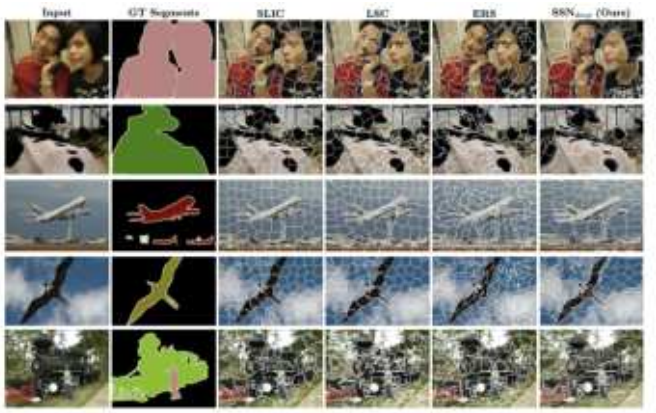
It's clear that this approach brings many new features and more elasticity. If for example a researcher wants to overfit to some images with his SPs now he is free of doing it instead before it was not possible since the features were not learnable and though the researcher was not able to customize superpixels over a set of images. Apart from this now the net is end2end trainable[19] and this means that can be easily integrated with other networks. In conclusion the paper brings a new method that is aligned with the sota and has also a appreciable runtime.

Contributions

The main contribution for sure is the simple idea of making something that was previously not differentiable so. It also solved the problem of the computational expensiveness explaining that just computing the 9 SPs near to the pixel was sufficient to receive the result expected and also find a good performance.

Experiments

An experiment that was done is about how the SSN deep compares with other state of the art methotds. It can be seen a comparative study here below. We can note that it tends, differently to the others, to make smoother areas that are also larger/longer or more diluted with respect to the other SOTA methods. In some cases many SPs are also overlapped or it seems so but the overall result is pretty good because it seems that the overlapping is concentrated more on the difficult borders and this means that the NN got the fact that the more difficult part to be segmented is the one on the edge and not the core that is mainly coherent.



Additional visual results on PascalVOC validation images. SSN_{deep} tends to produce smoother object contours and more superipxels near object boundaries in comparison to other superpixel techniques.

FIGURE 2.11: Experiment SSN deep vs SOTA

Chapter 3

Interactive Image Segmentation with Graph Transduction Games

3.1 Introduction

In this chapter we are going to describe the method we are proposing, the method is divided into sequential steps: 1) ingestion of one image 2) transformation of the image 3) creation of the superpixels 4) creation of the similarity graph 5) initialization of the probability assignment between nodes and labels (background/foreground) 6) running of GTG. These are explained better below since many passages are more complex than stated here. The key difference in our pipeline and other papers like [7][4] is that we use a game theoretic approach to move the labels from decided to undecided pixels.

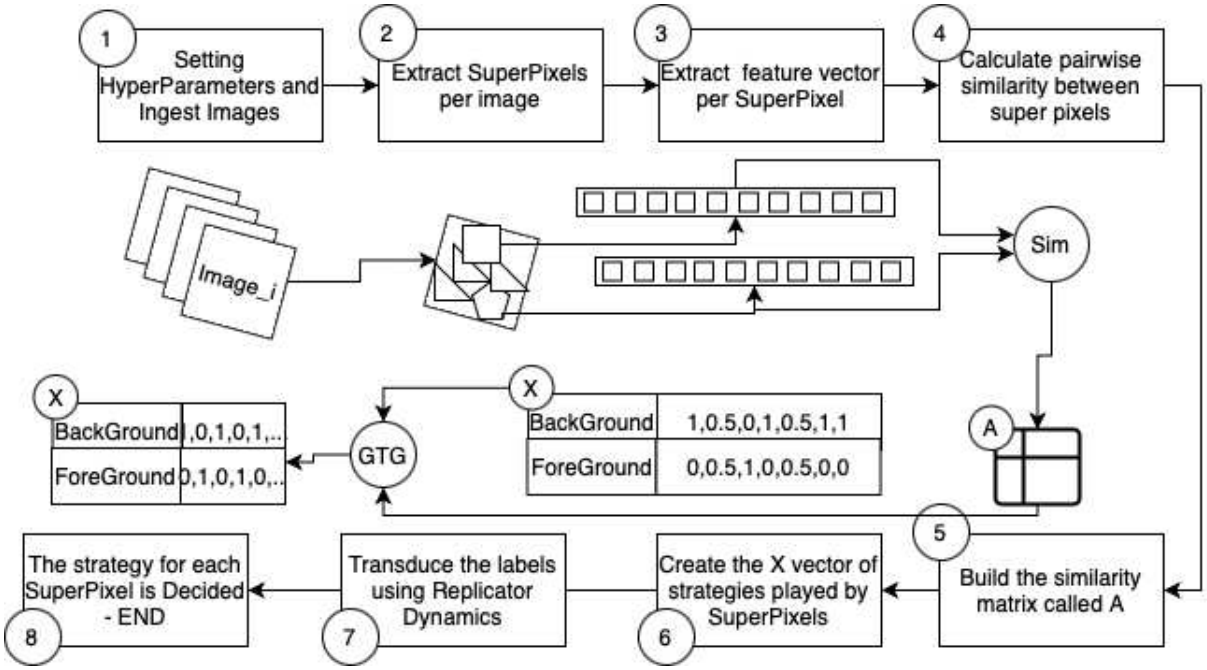


FIGURE 3.1: Diagram of the Whole method applied

3.2 Extraction of SPs

The process of extracting the SPs encapsulates also the preprocessing part. The whole process starts with the loading one-by-one of some tuples, each of them contains (image, label or seed, ground truth, name). All of these are needed because the image doesn't contain the seeds that are separated and the ground truth is necessary to calculate the accuracy at the end of the segmentation. The name is needed to name the output files. Every image is then transformed with the LAB transformation, Lab is a conversion of the same information to a lightness component L*, and two color components: a* and b*. Lightness is kept separate from color, so that you can adjust one without affecting the other. "Lightness" is designed to approximate human vision, which is very sensitive to green but less to blue. If you brighten in Lab space, the result will often look more correct to the eye, color-wise.

This computation is prepared to work in parallel or serially on the images and from the empirical results it seems that the parallel one is better performing compared to the serial one. An experiment here below3.1.

/	time	num images	num threads
single-thread	45secs	3	3
multi-thread	23secs	3	3
single-thread	208secs	15	4
multi-thread	75secs	15	4

TABLE 3.1: Comparative of single thread vs multi thread

The label or seed is modified because in the original version the 2 classes that are present are red seed and yellow seed instead are more meaningful in the case of binary classification to have a red seed and a green one. This because green means "pixels to be taken" and the inverse for the red. At the very end the superpixel function is launched and it turns back the image divided into segments. Here an example with 150 segments 3.2

Here an example with 3000 segments 3.3.

The superpixel function can be Slic, snic or many others and it is selected through an "environment" module that regulates whole of the global variables of the program. The superpixel functions in general has other parameters apart from the image to be processed such as the sigma and the cohesiveness of the SPs. Some of these parameters are going to be tuned but we already know some general rules such as the fact that increasing the number of prospected segments is going to improve the accuracy(fine grain). But instead increasing the number of superpixels can also mislead hugely if the image gradients are grown slowly and is slower whilst having few segments is going to give a much raw segmentation but, for sure, faster. The result of the extraction given an n by m by 3 image is a n by m mask in which every cell has a number [1...K] that is the segment's number.

3.3 SuperPixel Descriptors

For each SP we extract a feature vector composed of different pieces, each one describes in a different way the single superpixel and helps to give a fair feature vector that is going to be used for the similarity score between the superpixels.

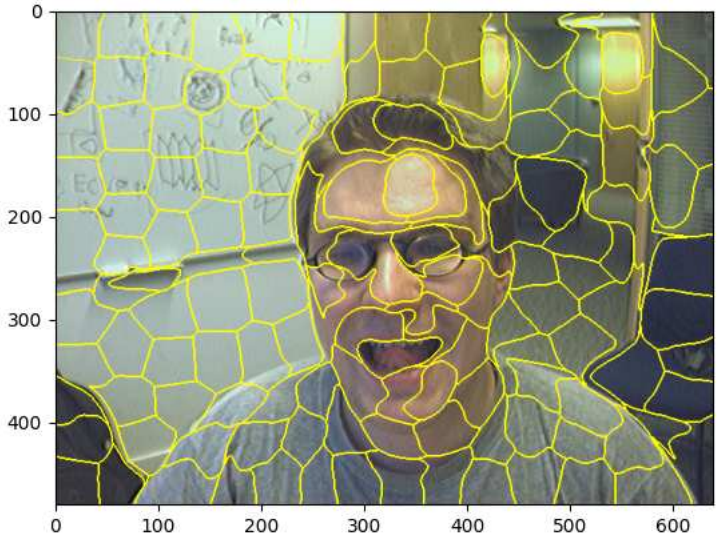


FIGURE 3.2: Segmentation with 150 superpixels of an image superimposing the superpixel segments

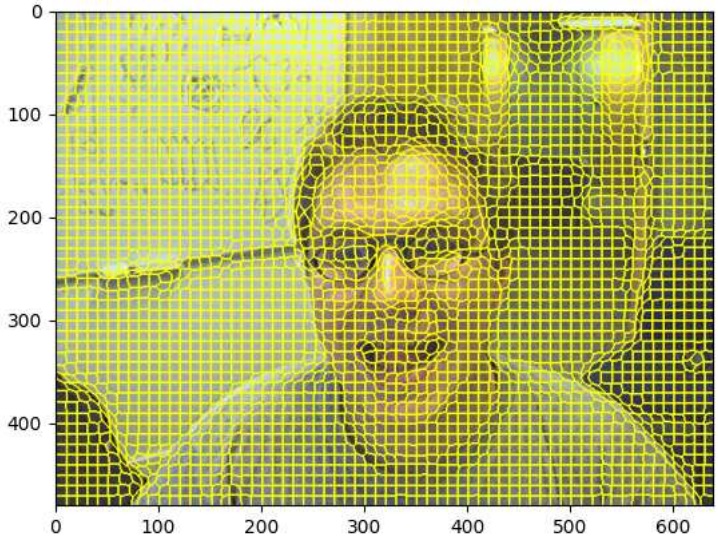


FIGURE 3.3: Segmentation with 3000 superpixels of a LAB image

3.3.1 RGB

The RGB component is clearly the easiest one but for sure it is helpful since an high gradient on this can correct other filters to a more rational measure of similarity.

3.3.2 LM-Filters

After the construction of the SPs we have in our hands some segments(cluster of pixels) that are named with a progressive number [0...K]. In the successive step the

construction of a map between segment number and pixels is of particular interest to simplify the access to a bunch of pixels at the same time even if they are in distributed on stretched segments. The map will be implemented as a dictionary in python and will look like this: 0: [(0,0),...,(x,y)].

At this point we created the Leung-Malik filters[20]. These are black-white filters made out of exponential or normal distributions with different parameters. They can be seen here below 3.4:

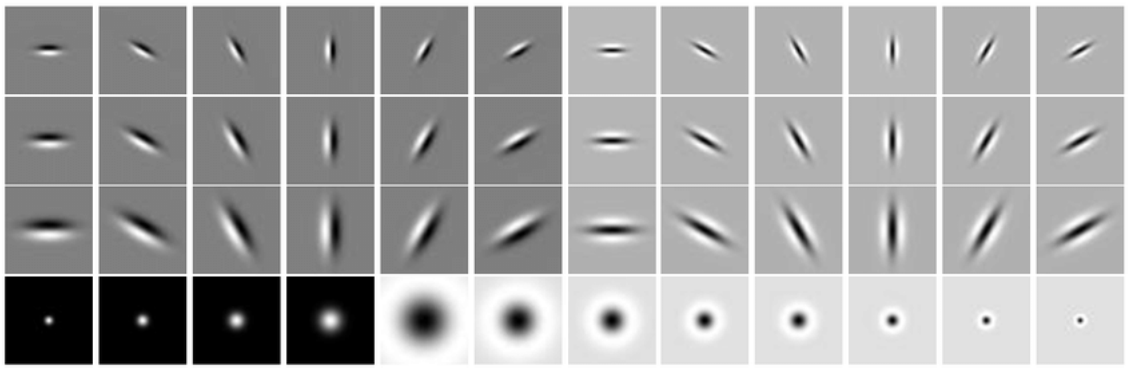


FIGURE 3.4: The LM-Filters

3.3.3 Sobel Operator

Apart from the LM-filters we added also some basic sobel/roberts and other filters. Here below an example of the use of sobel filter 3.5:

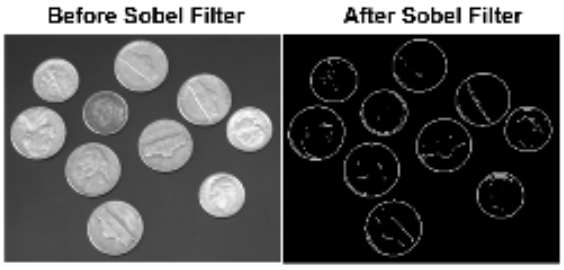


FIGURE 3.5: An example of the Sobel Filter

And how a Sobel filter is made 3.6:

All of them have the same size and are basically matrices. They are called kernels since the principle is the same even if the field is different. The ratio of calling them kernels is the fact that we are also going to introduce the use of convolutions to obtain from a certain starting image a new one that is filtered the way we apply the filter.

In particular given two matrices, in which one is the matrix of one of our images and the other is the kernel matrix, situation like in the image here below 3.7

To calculate for example 3.1:

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23} \tag{3.1}$$

We are going to exploit the formula here below 3.2:

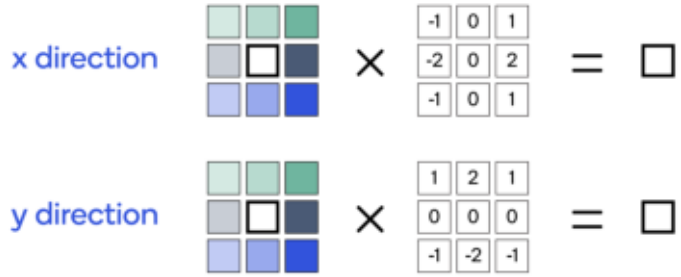


FIGURE 3.6: Sobel Filter components

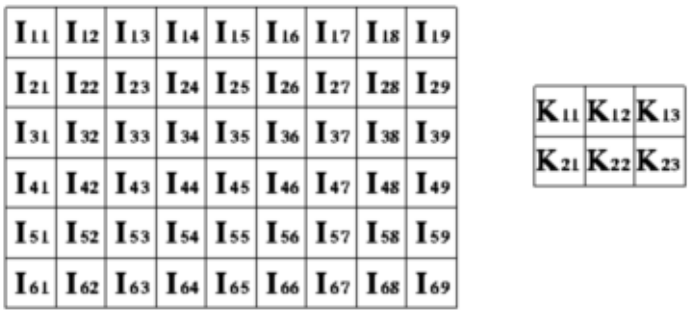


FIGURE 3.7: Matrices for the convolution

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l) \tag{3.2}$$

The example has been taken from:
https://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm

3.3.4 Feature Composition

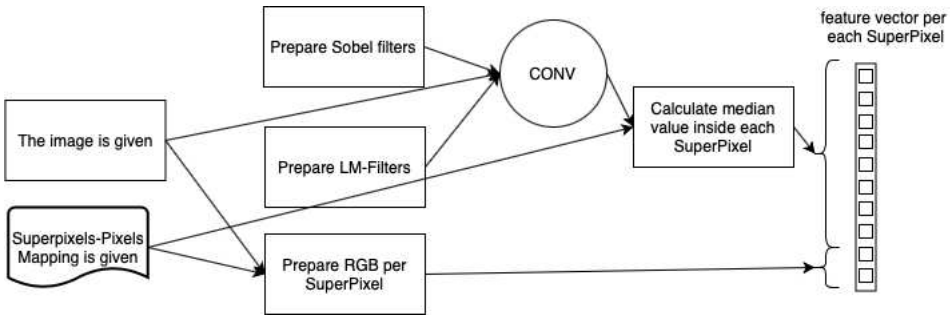


FIGURE 3.8: Diagram of the feature composition

The final descriptor is made of a composition of the one proposed in the previous sections (RGB, LM, Sobel). The composition is performed by simple concatenation of vectors. An idea of the whole process can be seen in 3.8. Each SP is associated with a feature vector $f \in \mathbb{R}^{1\times D}$. Operatively we take for every SP the vector of pixels and we take them out of the filtered image. Then the result comes taking

the median of the vector of pixels and it is a vector of RGB colors. Applying this technique to all of the SPs using all of the filters gives rise to our method. In terms of computational cost it is for sure not negligible the fact of having two nested for-loops and a convolution but actually the time per each image to be computed is among 10-12 seconds with my pc(MacbookPro, 13inches, i7@2.4ghz, 4+4cores, 16GB ram) and the whole program stops in 20mins with 151 images contained in GrabCut. The output of this process is that for each image you end up with 60 features per SP.

3.4 Graph Construction

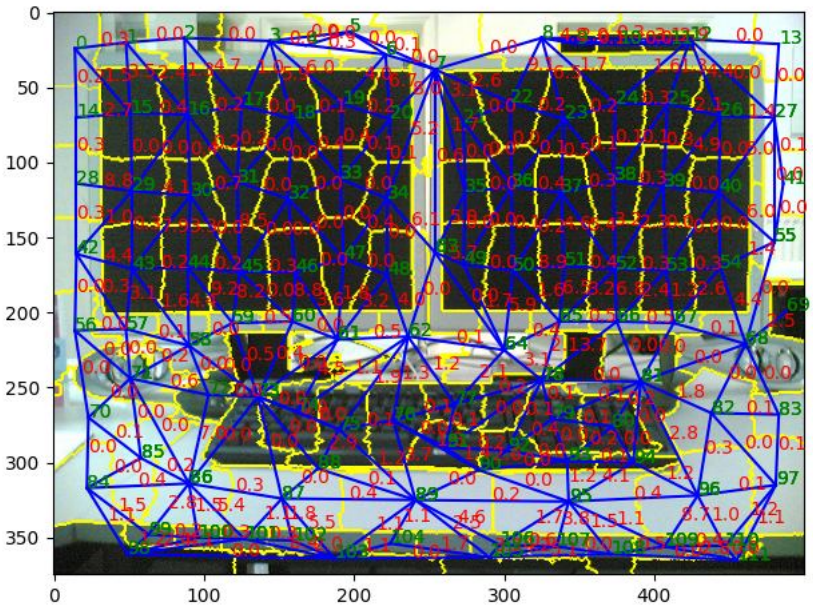


FIGURE 3.9: SuperPixels and their weights

In the image above 3.9 we can see a visualization of the SuperPixels(in green) and their pairwise weight with the neighbors(in red). The weight in the image is not normalized. The adjacency matrix A explained in the GTG method is built here in this way and the similarity is calculated between features vectors using the similarity/distance measures listed in the Experiments section. The Graph construction starts with the gathering of the information about neighbours between SPs and SPs. This information is needed because in our approach we think that the more two SPs are nearby the more they are able to be influenced and if one has a certain label it will be transmitted to the nearest. Albeit this is not enough because the existence of a point of contact between two SPs is not enough to have a measure of the strength of their relationship. For this reason the existence of a point of contact or not is encoded as a matrix called B(standing for borders) and it will be multiplied with the matrix W (standing for weights) that is a n by n matrix with a pairwise similarity coefficient between SPs. The weight matrix can be calculated in two ways: the first is with the pairwise cosine similarity(implemented in sklearn) and the second via euclidean distance 3.3.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \tag{3.3}$$

At the end the euclidean distance is transformed into a similarity measure by the mean of a gaussian kernel 3.4.

$$kernel(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \tag{3.4}$$

In the graph the nodes are the superpixels and the similarity matrix is populated with the weights calcuated through the formulas displayed here above onto the feature vector of each superpixel. The similarities are calculated over a feature vector of 60 features(still growing in number) that is created as explained in the previous section of LM-Filters. The multiplication of these two matrices (B W) gives rise to a new matrix that has zero as principal diagonal and has a weight different from zero off-diagonal when there is a point of contact between two SPs. The adjacency matrix is the one that is used in the replication dynamic 2.6 inside the GTG algorithm and that determines the strategies to be used by undecided superpixels.

3.5 Scribble propagation

The scribble propagation is done via the Graph Transduction Game ([8]), a popular class of semi-supervised learning techniques which aims to estimate a classification function defined over a graph of labeled and unlabeled data points. The basic idea is that every SP is a player that plays with a certain probability a move of one of the available classes. The unlabeled players, in the easiest setup, play every move(or class) with the same probability. In an advanced setup a prior is provided and the unlabeled players play with that prior. The yet labeled players instead play with 100% probability only the move/class they are already assigned. The whole setup can be represented by a vector of sizes #classes on the rows and #SPs on the columns. In the paper[8] the X vector is transposed with respect to the description provided here, this is a slight modification that does not affect how the algorithm works because the code runs respecting the paper description. This is a probability matrix therefore the summation by columns adds up to 1. The probability matrix will be called X. The players who already took a decision (have 100% on 1 class) will everytime remain the same during this process instead the ones who bet indifferently on all the available move/classes will change their strategy based on the similarity of other players. During this process the similarity will lead to a reinforcement of a certain strategy and at the end all of the players will be decided.

Example of propagation with scribbles 3.10:

FIGURE 3.10: GTG propagation with scribbles

Example of propagation with trimaps 3.11:

The reference image and its ground truth can be seen here below 3.12

FIGURE 3.11: GTG propagation with trimaps

FIGURE 3.12: Image and its GT of person8.png

Chapter 4

The Experiments

In this section we are going to list the experiments we did through the time we worked on this project, extensions to these experiments are in the section of the ablation studies where the reader will find optimization of the parameters that gave better results. The tests are run over two datasets but we plan to expand in a future version.

4.1 Datasets

We used the so-called GC50 that comes out from the paper [5] and its improved version that we called in this paper GC151 but actually is named Grabcut as its predecessor, the difference is in the number of images. The first grabcut dataset was composed of 50images [5] instead the new one has also images from VOC and Berkeley datasets.

4.1.1 GC50

It contains 49 images plus the last one, the sunflower that is present only as a trimap but the scribbles are not provided so we decided to exclude this image from the set to keep the dataset coherent with the different experiments. In Figure 4.1 are reported an image and its annotations from the G50 dataset.

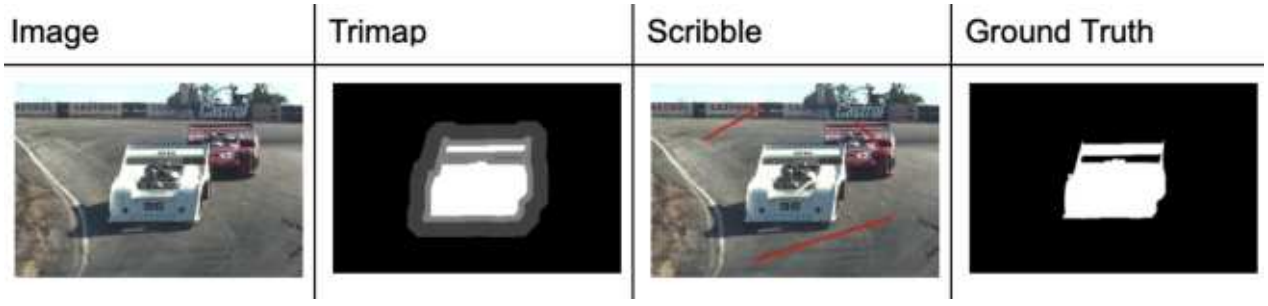


FIGURE 4.1: Example of images format of the BC51 dataset

In the example 4.1 the image called "21077" in which the task is segmenting only the car that is more in the foreground. The trimap has 4 bands of black that is background, white is foreground, and the last two types pf grays that are up to our modelling "undecided". In the third figure the reader can see the scribbles superimposed over the original image, the yellow scribble over the car (difficult to be seen but believe me, it exists) enlights the object to be extracted and the red scribbles are the things to be excluded. In the dataset the subject to be segmented, in most of the cases, appears in the center or nearby.

4.1.2 GC151

This dataset was created after the first one, in the 2007 and it was enlarged with other 100 images to make more difficult the challenge and to put the focus on the robustness of the algorithms that should make the new state-of-the-art. The dataset consists of 151 images. 49 images are taken from the GrabCut dataset, 99 from the PASCAL VOC datasetand 3 images from the Alpha matting dataset . The dataset is available through the oxford VGG lab website. This dataset was used in some preliminary tests and it is listed here because it was used for qualitative tests in some papers but it has not been used to evaluate our quantitative results.

4.2 Metrics

Metrics are designed, in this project, with a plug-in architecture, in practice they are in a separate dedicated file and are used as plugins in the sense that they can be pushed into some code and easily swapped with another metric that fits the same kind of data. This design pattern allows to have a cockpit (called ENV) in which you can simply state which kind of measure you want to explore in words like "cosine" and that one is used. At the time I'm writing only two measures are implemented, the one listed before and the euclidean, these two measures are used to calculate the distance of two superpixels given the feature vectors in a pairwise fashion.

4.2.1 J-Index

The Jaccard Index as guessable gives a score about the magnitude of cells correctly labelled by the algorithm over the total number of cells. The advantage of this measure is that it is the simplest one, instead the drawback is the fact that uncorrectly labelled pixels are submerged by the greatest number of the other pixels. The problem thus is that 99% of correctly labelled pixels can hide the fact that the more difficult pixels were mislabelled. For this reason this measure is not the best one and it preferred accuracy or error rate. The measure here below 4.1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

4.2.2 F1-Index

The F-score is often used in the field of information retrieval and it is appreciated because differently from the JIndex it is the harmonic mean of precision and recall. It deals better with not only the huge number of correctably guessed TP but it also enlights if there is a great number, in proportion, of FPs or FNs 4.2.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})} \tag{4.2}$$

4.2.3 Error Rate

The error rate is the inverse of the accuracy that is written just below this one 4.3.

$$err\_rate = 1 - accuracy \tag{4.3}$$

4.2.4 Accuracy

The accuracy is calculated as the number of correctly labelled points. In the example below we can see that the code performs an incremental verification of how much the accuracy increases at each iteration of the algorithm.

4.3 Competitors

We compare our model with the following methods (CDS [21], Segmented By Transuction[7], ecc). These models have been chosen because they are graph-based as our.

4.4 Annotations

We used two types of user annotations: scribbles and trimaps. These are used to indentify the pixels that have already been labeled by the user and cannot change, we are going to call these pixels Decided. The pixels though are not of interest so after the segmentation done via superpixels we extend the label to a pixel(if present) to the whole superimposed superpixel. The superpixel so will enjoy the fact of being Decided also if it has just one pixel Decided. We can see that this could be a problem but instead if the number of superpixel is high this is going to be a fair approximation since for example the Decided pixel is going to be one Decided over 10 pixels undecided in the same segment. The ratio between Decided and undecided inside the same superpixel should not be high obviously. After this operation the superpixels classified as Decided are going to have 1 in the probability vector of playing their color (RED = background, GREEN = foreground) instead the ones that are undecided have the same opportunity to be both red or green for this reason they are 50% GREEN and 50% RED in the probability vector. The algorithm will converge for sure also if the probability is odd because the similarity between the pixels will convince undecided superpixels to play the role that his closest superpixel advices.

4.5 Experimental setup

We are using python PyCharm IDE, no particular testing suite. We are testing on a macbook pro 13in. 2019 with i7@2.8ghz and 16GB of ram.

4.5.1 Experiment with GrabCut50 and trimaps

The experiment was run using the GC50 dataset that differently from its name has just 49 images, this happens because the dataset was archived from microsoft that hosted it. In this archival process all the images came to us except for one containing two flowers that came without the scribbles making impossible to use it because it wouldn't be comparable with the rest of the dataset. It wouldn't be fair. Apart from this we used the trimaps as mask over the images to make the segmentation. Trimaps are matrices with the same size of the image they are going to mask but have just 3 colors: black/white/gray. In case it is black it represents the background and should be excluded, in case it is white it is foreground and should be included. The most important part is the gray because it is on the edge between foreground and background image and though it has to be guessed which is the best separation

line between them. The gray part is the one over which we operate. We put the mask over the image and we try to guess the perfect border between fore and back.

In Figure 4.2 and 4.3 we reported some qualitative result.

FIGURE 4.2: Person1 image using 3000 segments and SNIC

Here below we can visualize also the mask relative to the image:

FIGURE 4.3: Person1 trimap

As we can see the image is segmented very well but we can get that the task is not so difficult since there is a very good background knowledge offered by the trimap itself that guides a lot the algorithm. Nonetheless there is a branch of research around the interactive segmentation using the trimaps and we are going to see that other papers, like the ones in the Background section, bring results based on trimaps.

4.5.2 Experiment with GrabCut50 and scribbles

This experiment is very similar to the one above in which we used trimaps, also the infrastructure that permits to run the masks is the same despite the evident difference between the two masks. This comes from the rule "never reinvent the rule" so we just adapted the code to run agnostically on any image just taking care of the colors and at most checking the tyoe of colors. This is also part of the plugin architecture landscape since the rapid prototipation of these results came directly out of a good infrastrure. For a qualitative result we can check here below 4.4.

FIGURE 4.4: Person1 image using 3000 segments and SNIC

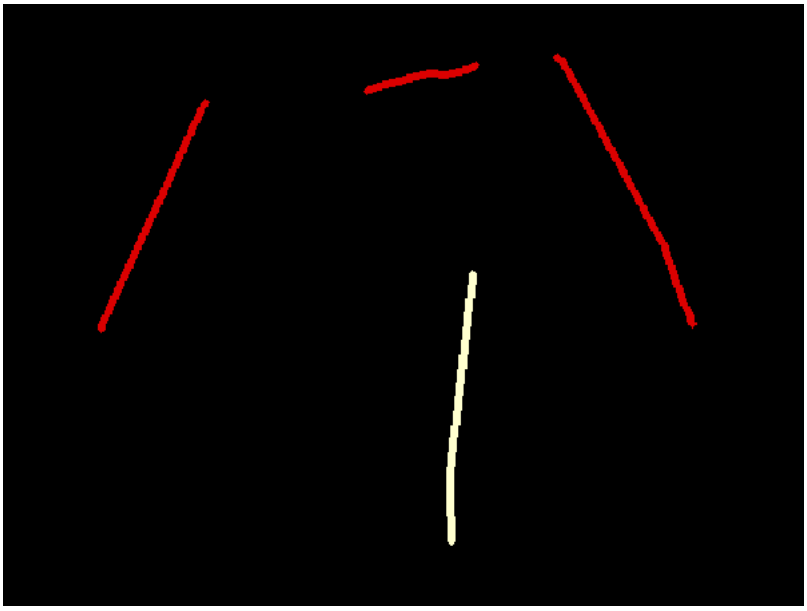


FIGURE 4.5: Person1 scribble

Here 4.5 we can visualize also the mask relative to the image.

As we can see the image is segmented in a poor way and for sure it is not going to score 100% of accuracy but as we can see while the ground truth is still the same as the trimap the quantity of informations we have through the scribbles is very very low.This inevitably brought to a bad quantitative and qualitative result. In this case the problem is so heavy that also the fact of increasing the number of segments used cannot improve the overall score and instead can bring a collateral effect that is substantial. Increasing, in fact, the number of segments brings a higher time to run: more iterations of the GTG algorithm and also more time for the computation of the SPs.

Chapter 5

The Results

The results shown here below are for the GC50 dataset, we run the algorithm alsdo on the GC151 but it takes quite a lot of time and for this reason we decided to drop on that side moving that task in the future works.

5.1 Trimaps

For what concerns the trimaps results we made the state of the art with this thesis. The task is intrinsinsically easy since the information provided is a lot and in fact other scientists made really good scores but we achieved a result that is better than the competitors, 0.23% of error against the lowest between the competitors that is 3.10%, here below a table 5.1.

The error rate we have on this task is 0.13, less than 1 pixel of error instead the reader can note from this table are between 3 and 5 pixels of error.

5.2 Scribbles

Here below 5.2 the comparison of some methods that use the scribbles, in this case we are not making the state of the art. Our method doesn't work well with the scribbles and makes too many errors, in fact our score is about 9 pixels of error while the others stay between a 8 - 9%.

Method	Error Rate
sDPMNL (boundary) [22]	11.43 %
GMMVL (location + color + boundary)	10.45 %
SVM (location + color + boundary) [23]	9.21 %
GM-MRF [24]	7.90 %
sDPMNL (color) [22]	7.65 %
Superpixels Hypergraph	7.30 %
Lazy Snapping [25]	6.65 %
Graph Cuts [26]	6.60 %
Cost volume filtering	6.20 %
Directed Image Neighborhood Hypergraph [27]	6.15 %
RobustPn	6.08 %
Grabcut [28]	5.46 %
Regularized Laplacian [7]	5.40 %
Grady's random walker [29]	5.40 %
Probabilistic Hypergraph	5.33 %
DPMVL (color + boundary)	5.19 %
Laplacian Coordinates	5.04 %
sDPMVL (color + boundary)	4.78 %
Sub-Markov Random Walk	4.61 %
Normalized Lazy Random Walker	4.37 %
Normalized Random Walker	4.35 %
Nonparametric Higher-Order	4.25 %
IISLPCN (default parameters)	4.15 %
Constrained Random Walks [30]	4.08 %
Lazy Randow Walks	3.89 %
Robust Multilayer Graph Constraints [31]	3.79 %
Texture Aware Model [32]	3.64 %
Pairwise Likelihood Learning	3.49 %
Multi-layer Graph Constraints	3.44 %
IISLPCN (optimized k)	3.21 %
Random Walks with Restart	3.11 %
Normalized Sub-Markov Random Walk	3.10 %
Difusive Likelihood	3.08 %
Our Method	**0.23 %**

TABLE 5.1: Comparative of our trimap model vs the others

Method	Error Rate
Segmentation model	Error rate
GMMRF [24]	7.9 %
Random Walker[29] (s=2)	5.4 %
Segmentation by Transduction [7] without AT	5.4 %
Square Laplacian regularizer	4.6 %
Random Walker (s=2)	3.3 %
Segmentation by Transduction [7] with AT	3.3 %
Our method	8.0 %

TABLE 5.2: Comparative of our scribble model vs the others

Chapter 6

The Ablation Studies

6.1 Finding the Performances upper bound

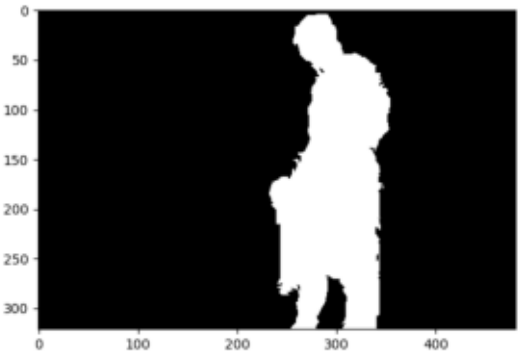


FIGURE 6.1: Maximum precision 99%

Finding the best possible performance ever is important to get a general idea of which is the best performance ever we can achieve through the methods we build up. To calculate it we segmented every image with 10'000 segments using the SLIC method and a compactness value of 10(default and prescribed as a good value). Given this we coloured automatically segments based on the ground truth. In practice we put the mask of segments over the GT and we coloured every segment based on the majority of the GT's colors underneath the segment. The result was that the best result ever is over the 99% of accuracy **??**. This is important because it means that we can really make a lot, in case the best ever result would have been 80%, by example, we should have defeated at that point taking everything over as an error.

6.2 SNIC vs SLIC performances

This can be seen with a plot here below and it is intereserting because it is not predictable which method is the best one.

Exactly as it can be seen the two are not stable and can be interleaved in a state of the art method to achieve the best performance since they tend to work better in different situations. Overall the best scoring when the number of segments is fixed is the SLIC method but on the long run the SNIC beats the SLIC. The main take away is that if it is mandatory to have a lot of segments for a certain image, for example in case the images are in hi-res, then use SNIC. In the other cases use SLIC.

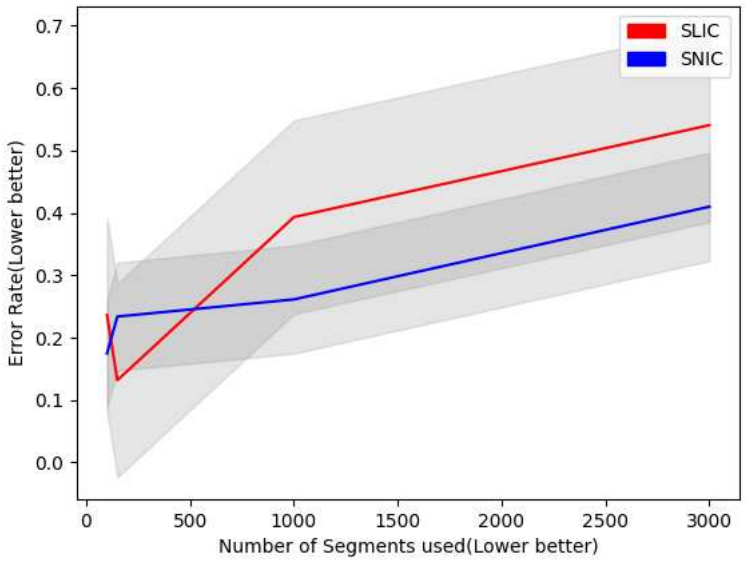


FIGURE 6.2: Comparison SLIC vs SNIC

6.3 Stability of GTG iterations

We have already seen the images below but now we should take care of the number of iterations of the GTG method. The linkage between the images below and the algorithm is that they are part of a GIF image that has 1 frame for every GTG iteration. In this case the GIF had 250 frames representing 250 GTG iterations. The fact is that in the image below I brought the first frame, the tenth and the last one. You can clearly get the point, from 10 to 250 we have 240 frames that basically are identical. In some cases watching carefully the GIF the reader can note that the partitions of background and foreground can move with small movements, this is because of the inability of GTG to decide difficult regions like borders but overall the algorithm is really stable. We have discovered though that 100 GTG iterations as upper bound is a pretty fair limit that doesn't put a road block to the capability of the algorithm to perform well, in practice it will converge after few tens of iterations and after those it will just start floating around noise.

FIGURE 6.3: GTG propagation with trimaps

6.4 Number of SPs

Here we tested the performances of XXX when the number of SP changes. Moreover, we stressed the performances considering different values of compatcness (see Sec YYY), a very important hyperparameter of SP. The tests conducted gave rise to

the two plots (see Figure 6.4 and 6.5). here below that are very useful for future benchmarks and optimizations since they perfectly give a simple interpretation of the situation: at the very start with a lower number of segments in the case of Scribbles the result is pretty similar but contrarily from what can be tought enforcing a lower compactness gives rise to a better result and on the long run having the default parameter(10) of compactness gives the best result. In practice it seems that enforcing too much the compactness of the clusters instead of finding very homogeneus clusters it ends up making wrong clusters.

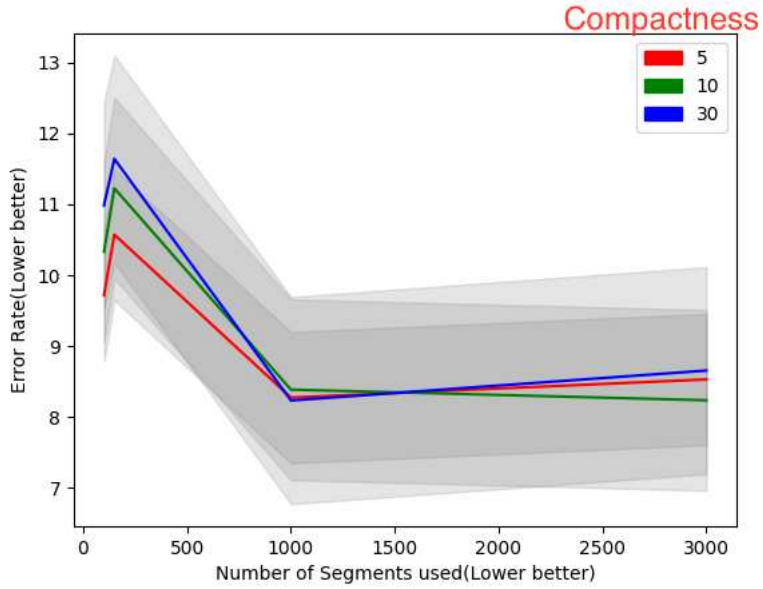


FIGURE 6.4: Comparison of segments vs compactness x err_rate for the scribbles

In the second plot here below about the trimaps we can see that overall the result is fine and under the 1 except in principle with a high number of compactness. It is interesting to note that inversely from the previous case in this a high level of compactness enforcement gives rise to a better performance on the long run.

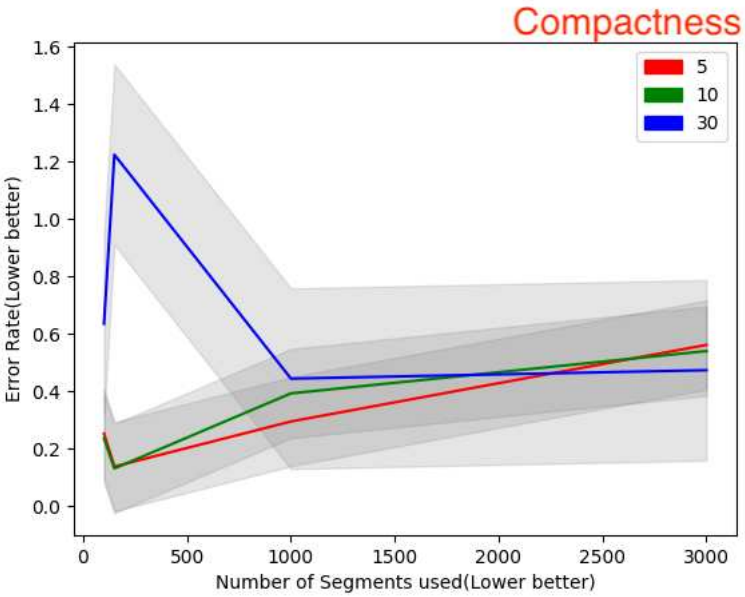


FIGURE 6.5: Comparison of segments vs compactness x error rate for the trimaps

6.5 Distance Measures

From what we studied during the experiments there is no difference between the two measures from the point of view of the accuracy but instead there is a difference from the point of view of the timing. In fact while the cosine distance is hand-crafted by me the euclidean is made by the sklearn developers that for sure are better than me and develop using criteria of high efficiency and caching paradigms. For this reason I think that if it is not mandatory to choose the cosine distance, the euclidean distance should be the default choice.

6.5.1 Euclidean

The euclidean measure is described here below in the formula and then it is implemented as is in the system. The formula 6.1 as is obviously doesn't work and so it is embeddded in other higher order functions that use it.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \tag{6.1}$$

6.5.2 Cosine

The cosine measure is described in the formula 6.2 and then it is implemented as is in the system.

$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6.2}$$

Chapter 7

Future Work

1. GC151 implementation, its easy but requires more time or more computational power, this was the obstacle.

2. Experiment with GrabCut50 and clicks

   The part of the experiment that involves the clicks were not developed at all since it is still part of the future advancements of the project but still merits to stay inside this thesis since we have a clear and detailed plan of how to implement it. In fact dealing with the blocking interactive input of a real human can be a mess and can require to build a user interface. For this reason this solution is to be avoided, instead a better fit solution for us is to simulate a series of points starting from the ground truth. The solution is as simple aas presented here and can be read more about it in the paper [4]. The Pros of this approach is to have results very rapidly instead retrieving real touches on a screen from a real user can require not only the time for the labelling of each image but also some time to create a proper UI, the contra is that the simulation algorithm can end up with behaviours that are not so human to be valid. An example is the fact that probably the simulation can end up with a lot of points that in the reality no one would have made because it should require too much time, the clicks can be too nearby one with the others but we know that our fingertip is not so small.

3. Improving the performance with the scribbles could be interesting since we achieved an encouraging result over the trimaps.

4. GNN

   In the node classification problem setup, each node v is characterized by its feature x_v and associated with a ground-truth label t_v. Given a partially labeled graph G, the goal is to leverage these labeled nodes to predict the labels of the unlabeled. It learns to represent each node with a d dimensional vector (state) h_v which contains the information of its neighborhood. The GNN setup is not yet implemented in this thesis but is a future improvement that should be out soon connecting the SSN that leverages the differentiability of SPs with the power of the GNN to effectively classify nodes.

Chapter 8

Conclusions

In the end we presented in the first chapter some of the most inspiring previous papers that helped us during the writing of this thesis. We started in particular from the paper of prof. Pelillo Constrained dominant sets that we studied in deep, both from the point of view of the general idea but also inthe implementation details. For example the sigma that is needed to be calculated while using the gaussian kernel as described here is taken from that paper. After this we read a lot of other papers. Another focal point was the selection of the grabcut that takes its name from the method proposed by the author of the paper grabcut. The selection of this dataset was absolutely easy because it is widely used but finding it was a bit difficult because the page that provided the dataset is no more maintained and the dataset itself came in two different fashions. The grabcut dataset was in the first approach composed by 50 images but only 49 came nowadays because one image does not have its relative scribble and therefore is unusable. An idea can be to make the scribble but this wouldn't be fair and would make the comparison with other papers impossible. The paper was also enlarged in 2007 and 101 images were added but the dataset kept its original name so we can find nowadays the same name for different flavours of datasets. The core part of the whole transduction method is obviously the GTG method that is quite easy and straightforward to be implemented since it is just some matrix multiplications and theoretically strong. What it took quite a lot of time was the entire pipeline to be developed. Finding the right transformations over the images, calculating the filters and how calulating the formers, these were just few of the decision we had to take and each one required extensive experiments but this helped making the whole pipeline more robust. A key role was played during the testing of the pipeline by the decision of visualizing every step. This was completely game changing since we were able, at the end of the development, to be able to visualize almost all the blocks of the process. One clear example is that even if GTG is an iterative process we found a way to visualize the result as a GIF image. This permitted to have better results and as a byproduct we got some fancy images to be put here in the thesis. One of the most difficult parts of the thesis, for sure, was the development of the feature vectors coupled to each superpixel. This involved a large amount of creativity since the filters we are using are different and can still be enlarged the number of them, they can be tuned to have bigger kernels and other hyperparameters are available. Other little decisions we made around the filters topic were about how to aggregate the value of a superpixel that was filtered. This is necessary because each superpixel is composed by pixels and though you have to summarize to just one value that represents the filter applied to the superpixel in the feature vector. We came up with the median that seemed the best one since does not introduces new colors, instead chooses one that is already present. The last important consideration is about the superpixel algorithms, we adopted and tested them(SLIC SNIC) because of their speed and accuracy. We can see that on my pc the

time required to elaborate each image is about 20 seconds and is a long time if you think having thousands of images. The superpixel algorithm takes 80% of this time and so it is important to rely on something as fast as possible and seemed that SLIC and SNIC were the best ones, others didn't even arrive to an end. In conclusion the result is really positive since we achieved a result that is state of the art in the trimap image segmentation over the grabcut dataset.

Bibliography

[1] Konstantin Sofiiuk et al. “f-BRS: Rethinking Backpropagating Refinement for Interactive Segmentation”. en. In: *arXiv:2001.10331 [cs]* (Aug. 2020). arXiv: 2001.10331. URL: http://arxiv.org/abs/2001.10331 (visited on 01/12/2021).

[2] Xuan Liao et al. “Iteratively-Refined Interactive 3D Medical Image Segmentation With Multi-Agent Reinforcement Learning”. en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 9391–9399. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.00941. URL: https://ieeexplore.ieee.org/document/9157356/ (visited on 04/08/2021).

[3] Eyasu Zemene, Leulseged Tesfaye Alemu, and Marcello Pelillo. “Dominant Sets for "Constrained" Image Segmentation”. en. In: *arXiv:1707.05309 [cs]* (July 2017). arXiv: 1707.05309. URL: http://arxiv.org/abs/1707.05309 (visited on 01/12/2021).

[4] Eyasu Zemene and Marcello Pelillo. “Interactive Image Segmentation Using Constrained Dominant Sets”. en. In: *arXiv:1608.00641 [cs]* (Aug. 2016). arXiv: 1608.00641. URL: http://arxiv.org/abs/1608.00641 (visited on 01/12/2021).

[5] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ““GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts”. en. In: (), p. 6.

[6] Fabricio Aparecido Breve. “Interactive Image Segmentation using Label Propagation through Complex Networks”. en. In: *arXiv:1901.02573 [cs]* (Jan. 2019). arXiv: 1901.02573. URL: http://arxiv.org/abs/1901.02573 (visited on 01/12/2021).

[7] Olivier Duchenne et al. “Segmentation by transduction”. en. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, AK: IEEE, June 2008, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587419. URL: https://ieeexplore.ieee.org/document/4587419/ (visited on 01/12/2021).

[8] Aykut Erdem and Marcello Pelillo. “Graph Transduction as a Non-Cooperative Game”. en. In: (), p. 29.

[9] Marcello Pelillo. “The Dynamics of Nonlinear Relaxation Labeling Processes”. en. In: (1997), p. 15.

[10] R. A. Hummel and S. W. Zucker. “On the Foundations of Relaxation Labeling Processes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5.3 (May 1983), pp. 267–287. ISSN: 1939-3539. DOI: 10.1109/TPAMI.1983.4767390.

[11] S. Wang, D. Wu, and K. Liu. “Semi-supervised Machine Learning Algorithm in Near Infrared Spectral Calibration: A Case Study to Determine Cetane Number and Total Aromatics of Diesel Fuels”. In: *2012 Fifth International Conference on Intelligent Computation Technology and Automation*. Jan. 2012, pp. 308–311. DOI: 10.1109/ICICTA.2012.84.

[12] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Trans. Pattern Anal. Mach. Intelligence* 23.6 (2001), pp. 643–660.

[13] Bolei Zhou et al. "Scene Parsing through ADE20K Dataset". en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 5122–5130. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.544. URL: http://ieeexplore.ieee.org/document/8100027/ (visited on 04/11/2021).

[14] R. Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (Nov. 2012), pp. 2274–2282. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2012.120. URL: http://ieeexplore.ieee.org/document/6205760/ (visited on 01/12/2021).

[15] Radhakrishna Achanta and Sabine Susstrunk. "Superpixels and Polygons Using Simple Non-iterative Clustering". en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 4895–4904. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.520. URL: http://ieeexplore.ieee.org/document/8100003/ (visited on 01/12/2021).

[16] Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-art Superpixel Methods". en. In: 6.1 (2011), p. 8.

[17] Varun Jampani et al. "Superpixel Sampling Networks". en. In: *arXiv:1807.10174 [cs]* (July 2018). arXiv: 1807.10174. URL: http://arxiv.org/abs/1807.10174 (visited on 01/12/2021).

[18] Zhongwen Hu, Qin Zou, and Qingquan Li. "Watershed Superpixel". In: Sept. 2015. DOI: 10.1109/ICIP.2015.7350818.

[19] Kai Wang, Liang Li, and Jiawan Zhang. "End-to-end trainable network for superpixel and image segmentation". en. In: *Pattern Recognition Letters* 140 (Dec. 2020), pp. 135–142. ISSN: 01678655. DOI: 10.1016/j.patrec.2020.09.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167865520303421 (visited on 04/05/2021).

[20] Thomas Leung and Jitendra Malik. "Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons". en. In: (), p. 16.

[21] Eyasu Zemene, Leulseged Tesfaye Alemu, and Marcello Pelillo. "Dominant Sets for "Constrained" Image Segmentation". en. In: *arXiv:1707.05309 [cs]* (July 2017). arXiv: 1707.05309. URL: http://arxiv.org/abs/1707.05309 (visited on 04/05/2021).

[22] Lei Ding and Alper Yilmaz. "Interactive image segmentation using probabilistic hypergraphs". In: *Pattern Recognition* 43.5 (2010), pp. 1863–1873. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2009.11.025. URL: https://www.sciencedirect.com/science/article/pii/S0031320309004440.

[23] Zheng Lin et al. "Interactive Image Segmentation With First Click Attention". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 13336–13345. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.01335. URL: https://ieeexplore.ieee.org/document/9157109/ (visited on 04/05/2021).

[24] A Blake et al. "Interactive Image Segmentation using an adaptive GMMRF model". en. In: (), p. 14.

[25] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. "Interactive Image Segmentation with Latent Diversity". en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 577–585. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00067. URL: https://ieeexplore.ieee.org/document/8578165/ (visited on 04/05/2021).

[26] Yuri Boykov and Gareth Funka-Lea. "Graph Cuts and Efficient N-D Image Segmentation". In: *International Journal of Computer Vision* 70.2 (Nov. 2006), pp. 109–131. ISSN: 1573-1405. DOI: 10.1007/s11263-006-7934-5. URL: https://doi.org/10.1007/s11263-006-7934-5.

[27] Aurélien Ducournau and Alain Bretto. "Random walks in directed hypergraphs and application to semi-supervised image segmentation". In: *Computer Vision and Image Understanding* 120 (2014), pp. 91–102. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2013.10.012. URL: https://www.sciencedirect.com/science/article/pii/S1077314213002038.

[28] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ""GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts". en. In: (), p. 6.

[29] L. Grady. "Random Walks for Image Segmentation". en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.11 (Nov. 2006), pp. 1768–1783. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2006.233. URL: http://ieeexplore.ieee.org/document/1704833/ (visited on 04/08/2021).

[30] Fengting Yang et al. "Superpixel Segmentation With Fully Convolutional Networks". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 13961–13970. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.01398. URL: https://ieeexplore.ieee.org/document/9156320/ (visited on 04/05/2021).

[31] Tao Wang et al. "Interactive Image Segmentation via Pairwise Likelihood Learning". en. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 2957–2963. ISBN: 978-0-9992411-0-3. DOI: 10.24963/ijcai.2017/412. URL: https://www.ijcai.org/proceedings/2017/412 (visited on 04/05/2021).

[32] Hailing Zhou, Jianmin Zheng, and Lei Wei. "Texture aware image segmentation using graph cuts and active contours". In: *Pattern Recognition* 46.6 (2013), pp. 1719–1733. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320312005316.