POLITECNICO DI TORINO

Master's Degree Course in Mathematical Engineering

Master's Degree Thesis

Enhanced Normalized Cuts with Spectral Weight Adjustment for Image Segmentation

Supervisor

Prof. EDOARDO FADDA

Candidate

GIACOMO BASTIANI

July 2024

Abstract

This work investigates an enhancement to the normalized cuts algorithm, introducing a preliminary spectral segmentation analysis to make the process of binary image segmentation more effective. In particular, we propose two improvements: *(i)* use the results of the proposed preliminary spectral clustering algorithm as a prior for the final segmentation *(ii)* use Bayesian optimization coupled with Gaussian processes to tune the hyperparameters.
Then, using the results from this procedure, and by maximizing a confidence measure it is possible to obtain a set of pixels belonging to the foreground and the background. These pixels are passed to a `min-cut/max-flow` algorithm as the source and sink nodes for further refinement, therefore automating the process of foreground/background pixel selection for this algorithm.
Finally, the normalized cuts algorithm has been customized to perform video segmentation in real time. Experiments on a subset of the *MSRA10K* dataset and in different video conditions showed relevant improvements in segmentation quality and real-time usage. The proposed methodologies have the potential to be the starting point for more advanced techniques in fields requiring precise and efficient video segmentation, such as hand tracking.

Table of Contents

List of Figures

Acronyms

Ncut
: Normalized Cuts

MFSC
: Multiscale Fast Spectral Clustering

S/T
: Source/Sink

KNN
: K-Nearest Neighbors

GP
: Gaussian Process

PSC
: Preliminary Spectral Clustering

Chapter 1

Introduction

An important challenge in computer vision is the *perceptual grouping problem*, which consists in trying to aggregate the visual elements within a scene into coherent structures. For example, considering a simple image with a red ball in the foreground and a green grassy field in the background, the challenge consists in accurately separating the object (ball) from the background (grass) despite the visual impediments, like shadows, that might lead to grouping them partially together. This problem involves exploiting the relationships between pixels, edges, colors, and other visual features, and trying to organize them into meaningful and distinct objects.

Graph cuts, a computational technique that draws inspiration from the principles of graph theory, have widely been used to address the segmentation of objects inside images: This approach consists in representing the image as a graph, where the nodes are represented by the pixels, and the edges capture the relationships between them. The ultimate goal of this graph-based representation is to extract the *global impressions* of an image, i.e., the most prominent shapes and objects, the way elements are positioned within the frame, and the overall crowdedness or sparseness of the image. The graph cut algorithms seek an optimal cut that separates the graph into two disjoint sets: One set represents the foreground (object of interest), and the other set represents the background (or other objects). To obtain a meaningful segmentation, many of these algorithms include the manual selection of at least one pixel for both the foreground (source node, S) and the background (sink node, T). However, without any interaction with the user, it can be difficult to identify certain nodes in the graph as belonging to the foreground/background to achieve a reasonable partitioning of the image.

As the foundation for this work is the Normalized Cuts (`Ncut`) algorithm, introduced in [1], which has been one of the most important algorithms in the field

of image segmentation. The traditional `Ncut` approach can be computationally intensive, and is not always as accurate as some applications require. To address these limitations, this work explores an enhanced approach to the `Ncut` algorithm by integrating a preliminary spectral segmentation analysis as a prior step. This spectral analysis helps in adjusting the edge weights, leading to more accurate segmentation boundaries.

Spectral clustering is a technique that leverages the properties of eigenvalues and eigenvectors of *similarity matrices* derived from the data. Unlike traditional clustering methods that rely on direct measures of data similarity in the original feature space, spectral clustering operates by mapping data points to a lower-dimensional space where the clustering structure becomes more apparent. This is particularly advantageous in image segmentation, where the relationships between pixels are often complex and non-linear. The strength of spectral clustering lies in its ability to handle complex structures and its flexibility in defining similarity measures. It is particularly effective in settings where the clusters are not spherical or linearly separable, as it relies on the global structure of the data rather than local properties alone. This makes spectral clustering a versatile tool for image segmentation, capable of producing high-quality results in a variety of challenging conditions.

This combination of spectral clustering and normalized cuts improves the segmentation results compared to using each method individually. In fact, despite employing a similar graph construction and using spectral techniques to obtain the values for the bipartition of the graph, spectral clustering and the normalized cuts algorithm often yield different results. This difference arises from the distinct objectives and constraints that each method applies during the segmentation process. Spectral clustering aims to partition the graph based on the eigenvectors of the similarity matrix, focusing primarily on minimizing the cut size. This approach emphasizes separating the nodes (pixels) into internally cohesive groups, without explicitly considering the relative size of the segments. On the other hand, the `Ncut` algorithm seeks to minimize the normalized cut value, which balances the cut size with the association within each segment. `Ncut` ensures that the resulting segments are not only cohesive but also approximately balanced in terms of size. This additional constraint often leads to partitions that better capture the global structure of the image and can prevent the formation of disproportionately small or large segments.

After obtaining the initial segmentation result by combining spectral clustering as a prior to the normalized cuts algorithm, the highest confidence pixels are finally passed to a `min-cut/max-flow` algorithm. In this way, the selection procedure for the foreground/background pixels is completely automated.

A key challenge in implementing image segmentation algorithms is the selection and tuning of hyperparameters. In spectral clustering, the chosen parameters play a crucial role in determining the quality of the segmentation. Efficient hyperparameter optimization is essential to achieve a good performance. In this sense, *Bayesian optimization* is a robust technique, which offers an approach to explore the hyperparameter space and identify the best configuration. It is based on the use of probabilistic models, and it provides both good performances and computational efficiency.

Another important challenge is the adaptation of the normalized cuts algorithm for live image segmentation. This involves modifying the traditional `Ncut` approach to meet the real-time processing and dynamic requirements of live image streams, which is critical for applications that demand immediate and accurate segmentation. The main focus is on the bipartition of each frame of the video into the object of interest and the background. This can be crucial for applications such as hand tracking, where real-time and accurate image segmentation is essential. Furthermore, the advancements in live image segmentation could have significant potential for future applications in various fields, as virtual reality, where precise and dynamic segmentation can greatly enhance user interaction and immersion.

Chapter 2

Literature review

Many methods have been developed to tackle the difficult challenge of image segmentation. In particular, in [2], the authors provide an overview of some image segmentation methods, dividing them into three main categories: *Traditional methods*, such as thresholding and clustering, *graph theoretical methods*, like normalized cuts and efficient graph-based image segmentation, and combinations of both. They emphasize the role of graph-based methods: By modeling the images as weighted graphs it is possible to achieve an effective segmentation. The methods highlighted in the survey are particularly suitable for handling complex image data, given their flexibility and computational efficiency.

Furthermore, an interesting summary of graph-based algorithms is provided in [3]. Various graph-cut techniques are discussed in the paper, including `min-cut/max-flow` algorithms, which have proven to be effective for precise and efficient image segmentation tasks. The authors also address interactive-based graph cut algorithms, in which the user has to interact with the image, providing either a bounding box to confine the object of interest or one or more pixels which have the role of background/foreground seeds.

In fact, many algorithms require the addition of hard constraints (seeds) to be able to segment the image in an efficient way: The topological constraints lower the dimension of the search space of feasible segmentations. This is the case for the algorithm presented in [4], which requires the user to provide initial labels for some pixels in the image, marking them as either foreground or background. These user-defined seeds are used to create terminal nodes (source node, S, and sink node, T) in the graph, which represent the foreground and background, respectively. The graph cut algorithm then finds the minimum cut that separates the foreground and background nodes, effectively segmenting the image. The cut minimizes an energy function, resulting in an optimal segmentation based on the provided seeds and

the defined energy terms.

Another interesting interactive image segmentation algorithm is proposed in [5]: The user has to mark the image with two polygons, one for the *foreground boundary*, and another for the *background boundary*. The *foreground boundary* is marked along the inner side of the object's contour using red points, creating the foreground region, while the *background boundary* is marked along the outer side of the object's contour using blue points, creating the background region. These marked regions serve as input parameters for the graph cut.

More complex approaches that involve user interaction have been proposed in recent years, like the one presented in [6], which includes geodesic distance and appearance overlap constraints.

An algorithm that does not rely on user interaction is the normalized cuts algorithm, which has been proposed in [1]. This paper has been a cornerstone in the field of image segmentation since its publication in 2000. Based on the graph theoretic formulation of grouping, the authors propose the *Normalized Cut*, a criterion that measures the disassociation between two groups while taking into account the total association within the groups. This approach aims to minimize the cut across a graph representing the image, ensuring that the segmented regions are as distinct from each other as possible while maintaining strong internal cohesion. The algorithm operates by solving an eigenvalue problem to find the optimal partitioning of the graph. By looking at the segmentation as a global optimization problem, the `Ncut` algorithm provides a robust framework that can be applied to various types of images, like natural scenes, and in many applications, like person detection.

The normalized spectral clustering approach discussed in [1] served as a basis for other image segmentation methods. In [7], the authors introduced the Multiscale Fast Spectral Clustering (`MFSC`) algorithm, which aims at reducing the computational complexity of normalized cuts while maintaining high performances. By applying spectral clustering at multiple scales this method can adapt to different levels of detail within an image. Moreover this approach enhances the overall segmentation quality: The multiscale fusion ensures that the segmentation results are not only accurate but also robust to variations in image content and resolution.

Despite its high computational cost, `Ncut` can still be useful in many applications. An interesting approach for detecting early blight in potato leaves using spectral clustering enhanced by normalized cuts is presented in [8]. The spectral clustering process involves constructing spectral embeddings using a certian number of eigenvectors of the *Laplacian* and then performing `K-means` clustering.

Efficient hyperparameter optimization is important to enhance the performance of these complex image segmentation algorithms. *Bayesian optimization* is a robust technique for this purpose. In [9] the authors demonstrated the application of *Bayesian optimization* for tuning machine learning algorithms, highlighting its efficiency in exploring the hyperparameter space to improve the model performance. Moreover they provide a method that uses probabilistic models to guide the search for optimal parameters, therefore ensuring computational efficiency and good model performance.

Similarly, in [10] the authors introduce the application of *Bayesian optimization* in scenarios where function evaluations are costly. By constructing surrogate models to approximate the objective function, *Bayesian optimization* facilitates efficient parameter tuning, which is particularly convenient for computationally intensive tasks like image segmentation.

Chapter 3

Preliminary Spectral Clustering

3.1 Method

Spectral clustering is particularly effective in partitioning an image into distinct regions due to its ability to capture the global structure of the image by analyzing the spectrum (eigenvalues) of the graph Laplacian. In the proposed spectral clustering workflow, the focus is on RGB images, which are composed of three color channels: Red, green, and blue. Each pixel in an RGB image is represented by a triplet of values corresponding to the intensity of these three channels.

To ensure computational efficiency, particularly for high-resolution images, a preprocessing step is employed, where the image is resized if its dimensions exceed a specified target size. In particular, given an image I of size (h, w, c), where h, w, and c represent, respectively, the height, width and number of color channels of the image, it is resized based on a target dimension (H, W) if either h or w exceeds H or W. The scaling factor s can be computed as:

$$s = \left\lfloor \frac{H}{\max(h, w)} \cdot 100 \right\rfloor, \tag{3.1}$$

and the new dimensions will be $h' = \left\lfloor h \cdot \frac{s}{100} \right\rfloor$, $w' = \left\lfloor w \cdot \frac{s}{100} \right\rfloor$. The number of color channels will remain the same.

Each of the $M = h' \times w'$ pixels in the resized image is represented by a feature vector that combines its spatial coordinates and color values. The pixel at position

(i, j) is denoted by $\mathbf{p}_{ij}$, with color values $\mathbf{c}_{ij} = (r_{ij}, g_{ij}, b_{ij})$ for the red, green, and blue channels respectively. The feature vector $\mathbf{f}_{ij}$ is defined as:

$$\mathbf{f}_{ij} = \left(\frac{i}{\sigma_{space}}, \frac{j}{\sigma_{space}}, \frac{r_{ij}}{\sigma_{color}}, \frac{g_{ij}}{\sigma_{color}}, \frac{b_{ij}}{\sigma_{color}} \right), \tag{3.2}$$

where σ_{space} and σ_{color} are hyperparameters that control the influence of spatial and color information. If, for example, σ_{space} is high, the algorithm will be less sensitive to the spatial distance between pixels. Therefore the feature matrix will be:

$$F = \begin{bmatrix} \mathbf{f}_{00} \\ \mathbf{f}_{10} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{f}_{(H-1)(W-1)} \end{bmatrix}. \tag{3.3}$$

The next step is to create the K-Nearest Neighbors (`KNN`) graph $\mathcal{G} = (V, E)$, where each node (pixel) is connected to its K nearest neighbors based on the Euclidean distance in the feature space. These distances are converted to similarities using a Gaussian kernel to form the adjacency matrix W, the elements of which represent the weight of the edges between each pair of nodes. In particular, indicating by $\mathcal{N}_u$ the set of the K nearest neighbors of the pixel u, it holds:

$$\begin{cases} W_{uv} = \exp\left(-\frac{\|\mathbf{f}_u - \mathbf{f}_v\|^2}{\sigma_{knn}} \right), & \text{if } v \in \mathcal{N}_u \\ 0, & \text{if } v \notin \mathcal{N}_u \end{cases}, \tag{3.4}$$

with $\mathbf{f}_u$ and $\mathbf{f}_v$ being the feature vectors of pixels u and v, while σ_{knn} controls the width of the Gaussian kernel. A larger value of σ_{knn} results in a slower decay of similarity, meaning that even pixels with larger distances between their feature vectors will still have significant similarity values.

The *Laplacian matrix* L can be computed as:

$$L = D - W, \tag{3.5}$$

where D (which is known as *Degree matrix*) is a diagonal matrix with diagonal elements $d_u = \sum_{v=1}^{M} W_{uv}$. After obtaining the *Laplacian matrix*, it is possible to partition the graph into B sets by applying the following steps:

1. Compute the B smallest eigenvalues of L, $\lambda_1, \ldots, \lambda_B$, in ascending order and their corresponding eigenvectors $\mathbf{z}_b$, $b = 1, \ldots, B$;

2. Build the matrix $Z \in \mathbb{R}^{M \times B}$, in which the b-th column corresponds to the vector $\mathbf{z}_b$. Let $\mathbf{y}_m = Z(m, :)$, $m = 1, \ldots, M$, $\mathbf{y}_m \in \mathbb{R}^B$. The rows of the matrix Z represent the low-dimensional embeddings of the pixels;

3. Cluster the points $\mathbf{y}_m$, $m = 1, \ldots, M$ into B clusters $C_1, \ldots, C_B$ using the `K-Means` algorithm, which solves the following optimization problem:

$$\min_{\{C_b\}_{b=1}^{B}} \sum_{b=1}^{B} \sum_{\mathbf{y} \in C_b} ||\mathbf{y} - \mu_b||^2,$$

   where $\mathbf{y}$ is a row of the matrix Z and μ_b is the centroid of the b-th cluster C_b.

Since we focus on background and foreground, in the following B will be set to 2. The result of the `K-Means` clustering is a label for each pixel indicating its cluster membership (in this case foreground or background). Finally, the label vector is reshaped to the original resized image dimensions to obtain the final segmented image.

3.2 Hyperparameter exploration

Throughout all the experiments, the target dimensions have been fixed to $(H, W) =$ (50,50), the number of clusters is set to $B = 2$ (as already mentioned), and in the construction of the `KNN` graph the number of neighbors considered for each pixel is 128. Moreover the identified clusters depend on the seed used for the `K-Means` algorithm. In all the experiments the seed was fixed in order to be able to replicate the results.

Before addressing the hyperparameter tuning process, a series of preliminary experiments have been conducted to show the effect of different sigma values (σ_{color}, σ_{space}, σ_{knn}). By fixing these hyperparameters, it is possible to make different observations for every single hyperparameter by varying one of them at a time. The first one to be analyzed has been σ_{color}. In fact, in general, it is more likely that pixels with the same color belong to the same set. The two images used to evaluate the effect of σ_{color}, σ_{space}, and σ_{knn} are taken from the COCO dataset [11].

The starting point that has been considered is:

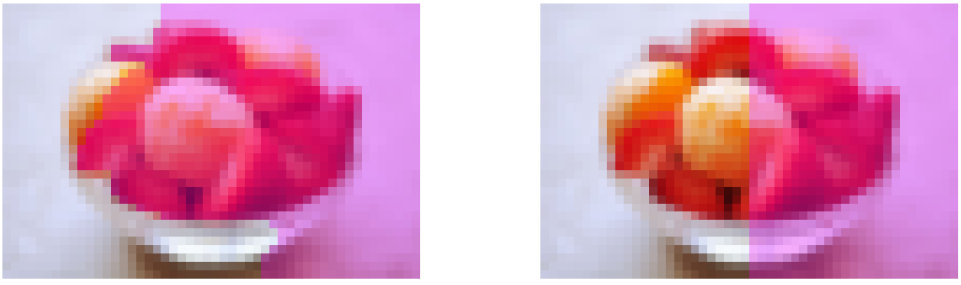
$$\sigma_{color} = \sigma_{space} = \sigma_{knn} = 1 \ .$$

3.2.1 Evaluation of σ_{color}

As previously discussed, σ_{color} is used as a normalization parameter to vary the influence of the color information of a pixel. For low values of this hyperparameter, the algorithm becomes very sensitive to changes in color between pixels, tending to group together pixels that have similar colors like in Figure 3.1(a) and 3.2(a). On the other hand, for high values of σ_{color} the color information about the pixels becomes irrelevant, and the image tends to be split in half at random, as it is possible to notice by comparing Figures 3.1(b) and 3.2(b).

(a) $\sigma_{color} = 0.05$ **(b)** $\sigma_{color} = 10$

Figure 3.1: `PSC`: Effect of different values of σ_{color} - bear image



(a) $\sigma_{color} = 0.2$ **(b)** $\sigma_{color} = 10$

Figure 3.2: `PSC`: Effect of different values of σ_{color} - fruit image

3.2.2 Evaluation of σ_{space}

The hyperparameter σ_{space} determines the sensitivity of the clustering algorithm to the spatial distance between pixels. A higher value of σ_{space} makes the algorithm less sensitive to spatial distances, while a lower value of σ_{space} increases the sensitivity to spatial differences, making the algorithm more responsive to the relative positions of the pixels. If the objective is to segment based on distinct regions that are spatially coherent, it could be better to use a lower σ_{space}. On the other hand, for segmenting regions based on color similarity across the image, a higher σ_{space} might be more appropriate.

(a) $\sigma_{space} = 3$ **(b)** $\sigma_{space} = 20$

Figure 3.3: PSC: Effect of different values of σ_{space} - bear image

(a) $\sigma_{space} = 3$ **(b)** $\sigma_{space} = 20$

Figure 3.4: PSC: Effect of different values of σ_{space} - fruit image

After the σ_{color} hyperparameter has been fixed ($\sigma_{color} = 0.08$ in these examples), it can be noticed in Figures 3.3(a) and 3.4(a) that using a lower value of σ_{space} (in this case $\sigma_{space} = 3$) different colors and textures are segmented separately. In Figure 3.3(a) and in Figure 3.4(a), the objects are separated from the background in an effective way. With higher values of σ_{space} (in this case $\sigma_{space} = 20$) the algorithm tends to merge larger regions together, as shown in Figures 3.3(b) and 3.4(b). This can lead to under-segmentation, where large regions of the image are grouped together, potentially merging some parts of the foreground and background that should be distinct. The segmentation boundary becomes less precise, failing to capture finer details.

3.2.3 Evaluation of σ_{knn}

The parameter σ_{knn} plays an important role in determining how pixel similarities are weighted. For low values of this hyperparameter, the similarities will have lower values for higher distances, and so only pixels with very similar feature vectors will have high similarity values. On the other hand, for high values of σ_{knn} even pixels with very different feature vectors can have significant similarity values. The effect of the choice $\sigma_{knn} = 1$ can be noticed in Figures 3.3(a) and 3.4(a), while the results of using of a large value for the considered hyperparameter are shown in Figure 3.5.

(a) $\sigma_{knn} = 10$ **(b)** $\sigma_{knn} = 10$

Figure 3.5: PSC: Effect of different values of σ_{knn}

3.3 Hypeprarameter tuning

The set of hyperparameters $(\sigma_{color}, \sigma_{space}, \sigma_{knn})$ has been obtained, drawing inspiration from [9], by applying *Bayesian optimization* using Gaussian Processes (GP).

The method operates by constructing a surrogate model of the objective function, which in this case is the *Intersection over Union* (IoU) score.

3.3.1 Metric

The IoU metric is used to evaluate the binary segmentation performance. It measures the overlap between two bounding boxes. The predicted bounding box (result from the model) and the ground truth bounding box (actual labeled data). It can be computed as:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}, \tag{3.6}$$

where $Area\ of\ overlap$ is the area where the predicted and ground truth bounding boxes intersect, while $Area\ of\ union$ is the total area covered by both bounding boxes combined, calculated as the sum of the individual areas minus the $Area\ of\ overlap$. Its interpretation is quite straightforward:

- $IoU = 0$: No overlap between the predicted and ground truth bounding boxes;
- $0 < IoU < 1$: Partial overlap between the predicted and ground truth bounding boxes;
- $IoU = 1$: Perfect overlap between the predicted and ground truth bounding boxes.

A common threshold for considering a detection as correct is $IoU > 0.5$, like proposed in [12], though this can vary depending on the specific application or dataset.

3.3.2 *Bayesian optimization* using Gaussian Processes

Returning to the *Bayesian optimization* framework, an objective function $f : \Theta \to \mathbb{R}$, defined over the bounded set Θ, is considered. This function represents the negative mean IoU score, and the goal is to find the set of hyperparameters $\boldsymbol{\theta} = (\sigma_{knn}, \sigma_{color}, \sigma_{space})$ that minimizes $f(\boldsymbol{\theta})$, thus maximizing the mean IoU score over a given set of labelled images. *Bayesian optimization* constructs a probabilistic model for $f(\boldsymbol{\theta})$ and uses this model to decide where to evaluate the function next within Θ, while accounting for uncertainty.

A space-filling design [13] is used to select an initial set of hyperparameters. In particular, Latin Hypercube Sampling (`LHS`), introduced in [14], is a statistical method used to create a range of possible parameter values from a multidimensional

distribution. Given a number k of parameters and a number n of samples, for each parameter x_i its range $[a_i, b_i]$ is divided into n intervals:

$$\left[a_i, a_i + \frac{b_i - a_i}{n}\right), \left[a_i + \frac{b_i - a_i}{n}, a_i + 2\frac{b_i - a_i}{n}\right), \ldots, \left[a_i + (n-1)\frac{b_i - a_i}{n}, b_i\right] .$$

Then one value v_{ij} is randomly sampled from each interval j for x_i. The sampled values for each parameter are finally permuted to create the sampling matrix. This method ensures that the initial points cover the hyperparameter space uniformly and reduces the likelihood of missing important regions of the parameter space. In this case, the hyperparameters were considered within the following ranges:

$$\sigma_{knn} \in [0.5, 10],\ \sigma_{color} \in [0.001, 0.1],\ \sigma_{space} \in [1, 10] .$$

For each set of initial hyperparameters $\boldsymbol{\theta}_i = (\sigma_{knn_i}, \sigma_{color_i}, \sigma_{space_i})$, the objective function is evaluated. This involves:

1. Running the spectral clustering algorithm with the given hyperparameters;
2. Computing the IoU score for each segmentation result;
3. Averaging the IoU scores across a subset of images to obtain the mean IoU;
4. Negating the mean IoU (since the optimization process seeks to minimize the objective function).

These evaluations provide a set of data points $\mathcal{D} = \{(\boldsymbol{\theta}_i, y_i)\}_{i=1}^n$, where y_i represents the negative mean IoU scores obtained using the set $\boldsymbol{\theta}_i$.

The Gaussian Process regression involves learning a distribution over functions that best explains the observed data $\mathcal{D}$. It is assumed that:

$$f(\boldsymbol{\theta}) \sim \mathcal{GP}(\mu(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta}')) ,$$

i.e., $f(\boldsymbol{\theta})$ is taken from a Gaussian process prior. The `GP` defines a distribution over functions, where $\mu(\boldsymbol{\theta})$ is the mean function, which represents the expected value of the function at $\boldsymbol{\theta}$, and $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$ is the covariance function. The Gaussian process provides probabilistic predictions, giving both a mean estimate and uncertainty for each set of hyperparameters $\boldsymbol{\theta} = (\sigma_{knn}, \sigma_{color}, \sigma_{spce})$.

Given the initial data $\mathcal{D} = \{(\boldsymbol{\theta}_i, y_i)\}_{i=1}^n$, where $y_i = f(\boldsymbol{\theta}_i) + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, the `GP` updates its prior belief to form the posterior distribution with new mean and variance given by Equations (3.7) and (3.8).

$$\mu_*(\boldsymbol{\theta}_*) = \mu(\boldsymbol{\theta}_*) + k(\boldsymbol{\theta}_*, \Theta)[K(\Theta, \Theta) + \sigma_n^2 I]^{-1}(y - \mu(\Theta)) \tag{3.7}$$

$$\sigma_*^2(\boldsymbol{\theta}_*) = k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*) - k(\boldsymbol{\theta}_*, \Theta)[K(\Theta, \Theta) + \sigma_n^2 I]^{-1}k(\Theta, \boldsymbol{\theta}_*) \tag{3.8}$$

In particular, Θ is the matrix of observed hyperparameters, $K(\Theta, \Theta)$ is the covariance matrix of the observed points, $k(\boldsymbol{\theta}_*, \Theta)$ is the covariance vector between the new point $\boldsymbol{\theta}_*$ and the observed points, and σ_n^2 is the noise variance.

An acquisition function $\alpha : \Theta \to \mathbb{R}^+$ is used to select the next set of hyperparameters to evaluate by optimizing $\boldsymbol{\theta}_{next} = \arg\max_{\boldsymbol{\theta}} \alpha(\boldsymbol{\theta})$. Please note that the acquisition function depends on the previous observations. The *Expected Improvement* (EI) function, which balances exploration and exploitation, can be written as:

$$\alpha_{EI}(\boldsymbol{\theta}) = \mathbb{E}[\max(f_{best} - f(\boldsymbol{\theta}),0)] \ . \tag{3.9}$$

Then following steps are repeated until the maximum number of iterations is reached:

1. Evaluate the objective function at $\boldsymbol{\theta}_{next}$;
2. Update the dataset $\mathcal{D} \leftarrow \mathcal{D}(\boldsymbol{\theta}_{next}, y_{next})$;
3. Refit the `GP` model with the updated dataset.

3.3.3 Image dataset

A subset of images from the *MSRA10K* dataset [15] [16] [17] [18] has been used to tune the hyperparameters. The dataset contains a total of 10,000 images. Each image in the *MSRA10K* dataset is paired with a binary mask that highlights the salient object in the image. This makes it particularly useful for training models designed for binary image segmentation. Additionally, the dataset contains images of various scenes and objects, making it ideal for obtaining hyperparameters that lead to good performances in a wide range of applications and cases.

3.3.4 Hyperparameter selection

Due to the computational complexity of the image segmentation task, processing the entire dataset would be computationally prohibitive. To ensure that the evaluation was feasible within the available computational resources, we select a random subset of 100 images. This subset provides a representative sample while making the optimization process manageable. Each image and its corresponding binary mask

in the selected subset were resized using the same approach proposed in Section 3.1. The resizing step for the binary masks involves interpolation, which takes into account the neighboring pixels to compute the new value: This can result in non-binary values. To convert these interpolated values back to binary, a threshold of 0.5 is applied. In particular, if the interpolated value is greater than or equal to 0.5, the pixel is assigned a value of 1; otherwise, it is assigned a value of 0. An example image is shown in Figure 3.6.

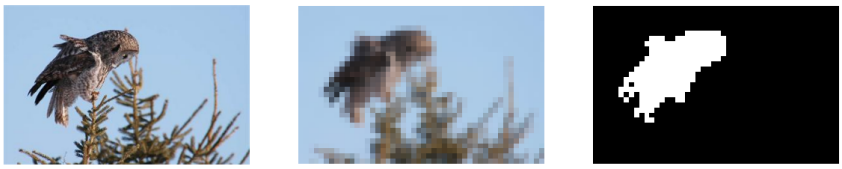


Figure 3.6: Original image from *MSRA10K*, resized image, resized mask

The triplet of hyperparameters that gives the best IoU value is:

$$(\sigma_{color} = 0.03,\ \sigma_{space} = 1,\ \sigma_{knn} = 6)\ ,$$

with a mean score, which can be computed as described in Equation (3.10), of $\overline{IoU} \approx 0.49$. This can be considered an acceptable value, given the resizing process applied to both images and masks.

$$\overline{IoU} = \frac{1}{N}\sum_{i=1}^{N} IoU_i \tag{3.10}$$

3.4 Results

The focus of the proposed spectral clustering algorithm is more on grouping similar pixels rather than ensuring balanced segment sizes. This method produces segments that are more distinct based on the feature vectors, leading to segments that might be less spatially coherent but more distinct in terms of color. The proposed approach offers great flexibility, resulting in segments that are more variable in size and coherence.

In some cases the proposed approach already produces high-quality segmentation results. An example is illustrated in Figure 3.7, which shows, from left to right, the original image, the resized image, and the segmented result using the optimized hyperparameters. In this case, the segmentation algorithm successfully isolates the flower from the background, demonstrating the capability of the method to accurately identify and segment distinct regions within the image. By leveraging the optimized hyperparameters found through *Bayesian optimization*, the proposed spectral clustering algorithm can effectively balance the trade-offs between different aspects of the image, such as color and spatial information, to produce precise segmentation results.

Figure 3.7: PSC: Image 72377

For many images, the algorithm only partially identifies the objects within the scene. Examples illustrating partial segmentations are shown in Figures 3.8, 3.9, and 3.10. In particular, in Figure 3.8 the segmentation algorithm correctly identifies a big part of the foreground. Even if it does not capture the entire object, the small segments identified in Figures 3.9 and 3.10 are correctly classified, indicating that the algorithm is able to recognize key features within the image even if it tends to group together a reduced portion of pixels.

In other cases, the algorithm fails to perfectly isolate the object. Figures 3.11, 3.12, and 3.13 show that while the algorithm is able to distinguish part of the object, it does not produce a perfect segmentation. This indicates areas where further refinement is necessary.

The algorithm has been tested on a subset of 50 images from the aforementioned dataset, resulting in a mean score $\overline{IoU} \approx 0.51$. Values above 0.5 typically indicate a good level of agreement between the predicted segmentation and the ground truth, reflecting the algorithm's effectiveness even after resizing.

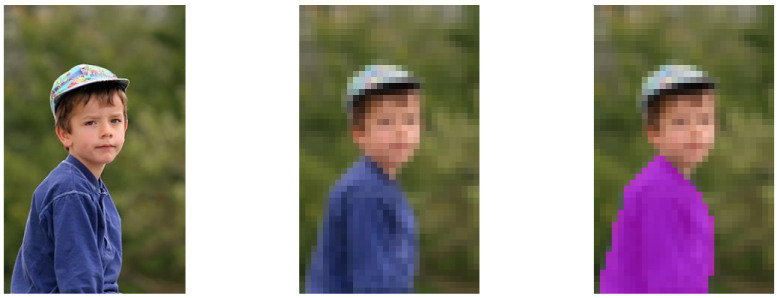


Figure 3.8: PSC: Image 198837

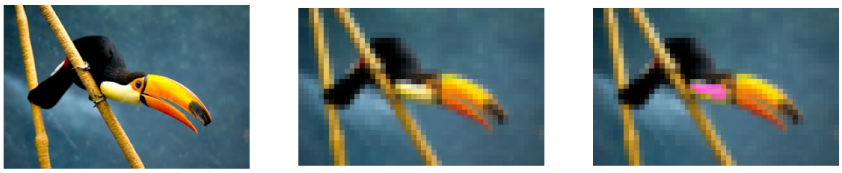


Figure 3.9: PSC: Image 178773

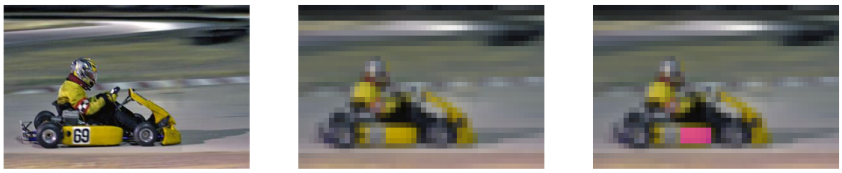


Figure 3.10: PSC: Image 178145

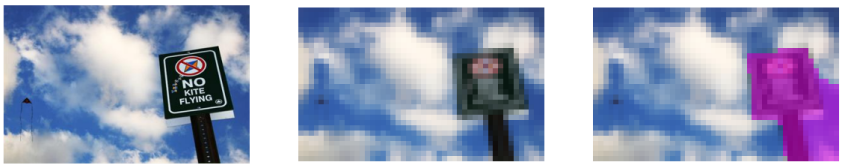


Figure 3.11: PSC: Image 198958

Figure 3.12: PSC: Image 54160

Figure 3.13: PSC: Image 124674

Chapter 4

Normalized Cuts

4.1 Method

In the approach proposed in [1], a weighted undirected graph $\mathcal{G} = (V, E)$ is constructed by taking each pixel as a node and forming an edge between every pair of nodes. The goal is to partition the set of nodes V into m disjoint sets $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_m$ in such a way to have a high intra-similarity and a low inter-similarity between the sets, where the similarity can be measured using the *Normalized Cut*, which is presented in Equation (4.1). Given the graph $\mathcal{G} = (V, E)$, two disjoint sets A, B can be obtained by eliminating the edges between these sets. Please note that it holds:

$$A \cup B = V,\ A \cap B = \emptyset\ .$$

Since the weights of the edges define how strong two connected nodes are, the optimal bipartition of the graph is the one corresponding to the minimum *cut* value (4.2), i.e., the sum of the removed weights. However in this way, for example considering weights that are inversely proportional to the distance between the nodes, partitions with one node or a very small number of nodes are favoured. The measure of disassociation proposed by the authors can be computed as:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{4.1}$$

where

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v), \tag{4.2}$$

and

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t). \tag{4.3}$$

In Equation (4.3), $assoc(A, V)$ represents the sum of the weights connecting the nodes belonging to A to all the other nodes in the graph. In this way, the disassociation between a small set of nodes and the rest of the graph is not necessarily small anymore.

As shown in the paper, the bipartition that minimizes the *Ncut* value can be obtained by taking the eigenvector corresponding to the second smallest eigenvalue for the following generalized eigensystem:

$$(D - W)\mathbf{y} = \lambda D\mathbf{y} \ . \tag{4.4}$$

Note that this system can be rewritten as a standard eigensystem:

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z}, \tag{4.5}$$

with $\mathbf{z} = D^{\frac{1}{2}}\mathbf{y}$. Moreover, since for an undirected graph the *Laplacian matrix* $(D - W)$ is symmetric positive semidefinite, the matrix $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ is still symmetric positive semidefinite.

In the proposed implementation, the weights of the edges, i.e., the elements of W, have been defined as:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{I}(\mathbf{p}_i)-\mathbf{I}(\mathbf{p}_j)\|^2}{\sigma_I}\right) \cdot \exp\left(-\frac{\|\mathbf{p}_i-\mathbf{p}_j\|^2}{\sigma_d}\right), & \text{if } \|\mathbf{p}_i - \mathbf{p}_j\| \leq r \\ 0, & \text{otherwise} \end{cases} , \tag{4.6}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ represent the pixels at positions (x_i, y_i) and (x_j, y_j), and $\mathbf{I}(\mathbf{p}_i)$ denotes the vector of color intensities of the i-th pixel. Even if in the original paper the primary focus is on brightness images, in the proposed implementation the algorithm has been adapted to work with RGB images. The parameters σ_I and σ_d control, similarly to what has already been discussed in Sections 3.2.1 and 3.2.2, the sensitivity of the algorithm to the pixel color intensities and to spacial distances. Note that the weight is 0 if two pixels are more distant than a certain fixed value r.

The normalized cuts method was employed for bipartitioning the image. The primary objective is to segment the image into the foreground and background regions. By minimizing the normalized cut value, the algorithm produces segments that are both internally coherent and balanced in size.

There are many ways in which the eigenvector corresponding to the second smallest eigenvalue for the eigensystem described in Equation (4.4) can be used to partition the graph into two sets. The choice has been to take the sign of the elements

of the eigenvector to label each pixel as belonging to the foreground/background. In particular, given the eigenvector of interest $\mathbf{y}_2$, for the i-th pixel it holds:

$$Label(i) = \begin{cases} \text{Foregorund}, & \text{if } \mathbf{y}_2(i) > 0 \\ \text{Backgorund}, & \text{otherwise} \end{cases} . \tag{4.7}$$

Another possibility could be to take the median value as a splitting point, but this approach leads to worse results in general. In their implementation, the authors use as the splitting point for the eigenvector the point that gives the best $Ncut(A, B)$ value. This is obtained by checking multiple evenly spaced splitting points. However in this work, after various test, it has been chosen to stick with the sign function, as it already gives good results and is much faster.

4.2 Hyperparameter tuning

Before applying the `Ncut` algorithm, the images have been preprocessed as shown in Section 3.1. The target dimensions considered are still $(H, W) = (50,50)$. There are three hyperparameters that need to be tuned: r, σ_I, and σ_d.

The hyperparameter r defines the radius of the neighborhood around each pixel within which the weight between the considered pixel and the other pixels is computed. When it is small, the algorithm considers only a limited local neighborhood for each pixel. This can be useful for capturing fine details and small structures within the image. However, it might miss larger, more global structures, leading to over-segmentation. For high values of r, the graph will emphasize global relationships over local details, potentially ignoring fine structures and boundaries.

σ_I controls the algorithm's sensitivity to differences in pixel intensity. If it is too high, the algorithm fails to distinguish between different regions with significant intensity differences. However if this hyperparameter is too low, the image might be segmented into small regions because the algorithm perceives the edge between two pixels with small color differences as having a smaller weight than it should have.

Finally, the hyperparameter σ_d controls the influence of the spatial distance between pixels. If it is too low, the algorithm becomes very sensitive to the distances between pixels. This means that even small distances between two pixels will result in a tiny weight associated to the edge that connects them, and so the algorithm might fail to capture meaningful structures within the image. On the other hand, if this hyperparameter is too high, even pixels that are far apart will be considered similar if, for example, their color is similar. Therefore it might happen

that distinct objects are merged into a single element.

These hyperparameters have been tuned using the same method proposed in Chapter 3, Section 3.3. In this case, drawing inspiration from the original paper [1], the ranges that have been chosen for the *Bayesian optimization* process are:

$$r \in [1,20],\ \sigma_I \in [0.001,0.5],\ \sigma_d \in [1,20]\ ,$$

with r being an integer value.

4.3 Results

The triplet of hyperparameters that has been obtained using the same subset of 100 images considered in Section 3.3.4 is:

$$(r = 10,\ \sigma_I = 0.0057,\ \sigma_d = 9.3)$$

which, in this case, gives a mean score $\overline{IoU} \approx 0.51$.

The `Ncut` algorithm is designed to segment images by minimizing the disassociation between segments while maximizing the association within segments. In particular, this approach works well for images where the regions to be segmented have distinct global properties, i.e, the features (like color, texture, overall intensity, etc.) of the image are consistent across large regions. An example of good performance of this algorithm in presented in Figure 4.1, in which both the original image and the segmented resized image are reported.

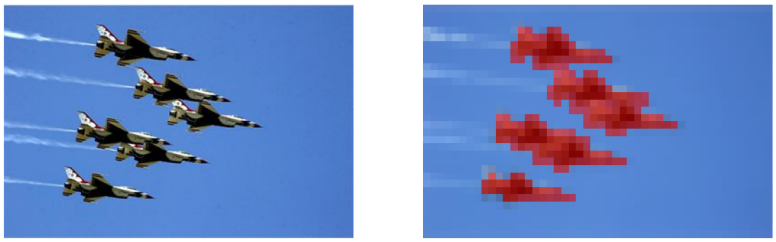


Figure 4.1: `Ncut`: Image 198406

This method is better, with respect to the PSC algorithm, at considering the overall structure of the image, as it can be seen in Figure 4.2.

However, with some images, the normalized cuts algorithm struggles in accurately isolating objects that have complex shapes or color patterns, as it is show in Figures 4.3 and 4.4.

Figure 4.2: Ncut: Image 178773

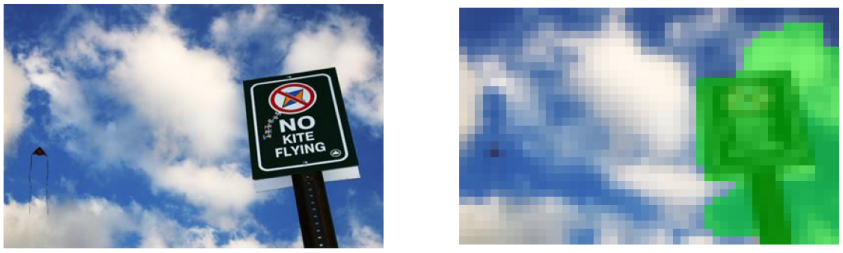


Figure 4.3: Ncut: Image 198958

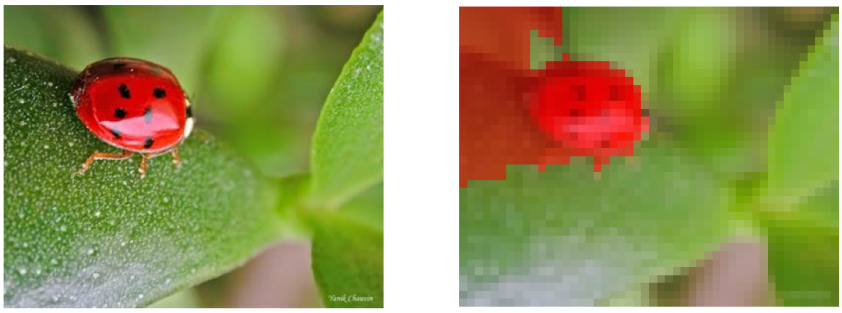


Figure 4.4: `Ncut`: Image 53923

In scenes with complex objects or where the foreground and background share similar properties, like the one presented in Figure 4.5, the normalized cuts method has difficulties in separating the segments in a meaningful way.

Figure 4.5: `Ncut`: Image 198394

The overall performance of the `Ncut` algorithm is evaluated on the same subset of 50 random images from the *MSRA10K* dataset used in Section 3.4. The mean *Intersection over Union* score achieved in this case is 0.54, indicating a moderate level of accuracy in the segmentation results.

Chapter 5

Enhanced Normalized Cuts

5.1 The algorithm

To improve the segmentation results obtained from the standard `Ncut` algorithm, this section proposes an enhanced method that incorporates prior information from the spectral clustering segmentation.

The choice to use spectral clustering as a prior for the normalized cuts algorithm is driven by the observation that the preliminary spectral clustering algorithm described in Chapter 3 often successfully identifies a small, but correct part of the object with respect to the background. This initial segmentation can be used as a reliable prior for further refinement. In particular, the weights in the `Ncut` algorithm can be adjusted to reflect this prior information.

The enhanced normalized cuts method involves, first of all, obtaining an initial segmentation of the image by applying the `PSC` algorithm. Using this segmentation, the weights between pixels in the normalized cuts algorithm are adjusted. It holds:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{I}(\mathbf{p}_i)-\mathbf{I}(\mathbf{p}_j)\|^2}{\sigma_I}\right) \cdot \exp\left(-\frac{\|\mathbf{p}_i-\mathbf{p}_j\|^2}{\sigma_d}\right) \cdot S_{ij}, & \text{if } \|\mathbf{p}_i - \mathbf{p}_j\| \leq r \\ 0, & \text{otherwise} \end{cases}, \tag{5.1}$$

where

$$S_{ij} = \begin{cases} \alpha, & \text{if } \text{label}_i = \text{label}_j \\ \frac{1}{\alpha}, & \text{if } \text{label}_i \neq \text{label}_j \end{cases}, \tag{5.2}$$

with $\alpha > 1$.

$\mathbf{p}_i$ and $\mathbf{I}(\mathbf{p}_i)$ represent, respectively, the spatial coordinates and the color intensities of the i-th pixel. The role of σ_I and σ_d has been widely discussed in Chapters 3 and 4. In this case the factor S_{ij}, called *similarity factor*, influences how strongly the initial `PSC` results affect the final segmentation. High values of S increase the weight between pixels that have the same label in the preliminary spectral clustering result, therefore encouraging these pixels to stay together in the final segmentation.

The rest of the algorithm is the same as the one presented in Section 4.1. The graph is constructed, where each node represents a pixel and each edge is represented by the computed weight 5.1 between pixels. After computing the *symmetric normalized Laplacian matrix* defined in Equation (5.3), the eigenvalue problem described in Equation (5.4) is solved for the smallest eigenvalues.

$$L_{sym} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \tag{5.3}$$

$$L_{sym}\mathbf{v}_k = \lambda_k \mathbf{v}_k \tag{5.4}$$

The eigenvector $\mathbf{v}_2$ corresponding to the second smallest eigenvalue λ_2 is used to bipartition the graph based on the sign of its components:

$$Label(i) = \begin{cases} 1, & \text{if } \mathbf{v}_2(i) > 0 \\ -1, & \text{if } \mathbf{v}_2(i) \leq 0 \end{cases} . \tag{5.5}$$

The normalized cuts algorithm benefits from the preliminary spectral clustering result by focusing on refining the segmentation rather than starting from scratch. This approach leads to a better handling of complex structures and boundaries within the image. The initial segmentation acts as a guide, helping the `Ncut` algorithm to maintain coherence within segments that have already been roughly identified.

The images have been resized using $(H, W) = (50,50)$ as target dimensions. However, after the segmentation, the image is resized using $(H', W') = (100,100)$ as new target dimensions. This process is necessary because the highest confidence (see Section 5.2) pixels will be passed to the `min-cut/max-flow` algorithm to refine the segmentation. For this purpose it is better to use higher quality images. The method that has been used to map the original segmentation result to the new dimensions is the *nearest neighbor interpolation*: It assigns the value of the nearest pixel in the original image to the corresponding pixel in the new resized image.

Specifically, denoting the source image as I_s with dimensions $H_s \times W_s$, and the target image as I_t with dimensions $H_t \times W_t$, the target coordinates are given by:

$$x_t = \left\lfloor \frac{x_s \cdot W_t}{W_s} \right\rfloor, \; y_t = \left\lfloor \frac{y_s \cdot H_t}{H_s} \right\rfloor .$$

5.2 Confidence measure

In order to identify the single pixels belonging to the foreground/background that should be passed to the successive `min-cut/max-flow` algorithm to automate the process of the S/T nodes selection, a confidence measure is introduced to understand how reliable the segmentation result is for each pixel.

The Gaussian Kernel presented in Equation (5.6) is used to weight the contribution of the neighboring pixels when computing the confidence for a given pixel.

$$G(x, y; \sigma) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{5.6}$$

The parameter σ controls the spread of the kernel, while (x, y) are the coordinates relative to the center of the kernel.

For each pixel (i, j) in the segmented image only a local neighborhood around it is considered, and the confidence is computed by following these steps:

1. Extract a region of radius r around the pixel (i, j);
2. Apply the Gaussian kernel to the local neighborhood. The kernel weights the contribution of each pixel based on its distance from the center pixel (i, j);
3. Sum the weighted contributions of the pixels in the local neighborhood that belong to the foreground and background segments separately;
4. Normalize these sums to obtain the final confidence values.

Equations (5.7) and (5.8) describe how the foreground and background confidences are computed, respectively.

$$C_{foreground}(i, j) = \sum_{k,l \in N(i,j)} G(k - i, l - j; \sigma) \cdot \mathbf{1}_{\{S(k,l)=1\}} \tag{5.7}$$

$$C_{background}(i, j) = \sum_{k,l \in N(i,j)} G(k - i, l - j; \sigma) \cdot \mathbf{1}_{\{S(k,l)=0\}} \tag{5.8}$$

$N(i,j)$ denotes the neighborhood around the pixel (i,j), $S(k,l)$ is the segment label of the pixel (k,l), and $\mathbf{1}_{\{S(k,l)=b\}}$ is an indicator function that is 1 if $S(k,l)=b$ ($b=1$ indicates the foreground, $b=0$ indicates the background), 0 otherwise.

The normalization step ensures that the confidence values are in the [0,1] range:

$$C_{foreground}(i,j) \leftarrow \frac{C_{foreground}(i,j)}{\max(C_{foreground})} ,$$

$$C_{background}(i,j) \leftarrow \frac{C_{background}(i,j)}{\max(C_{background})} .$$

An example of *confidence map* is presented in Figure 5.1. In the first row the original image and the corresponding segmentated image can be found, whereas in the second row the *confidence map* and the image representing the highest confidence pixels are represented.

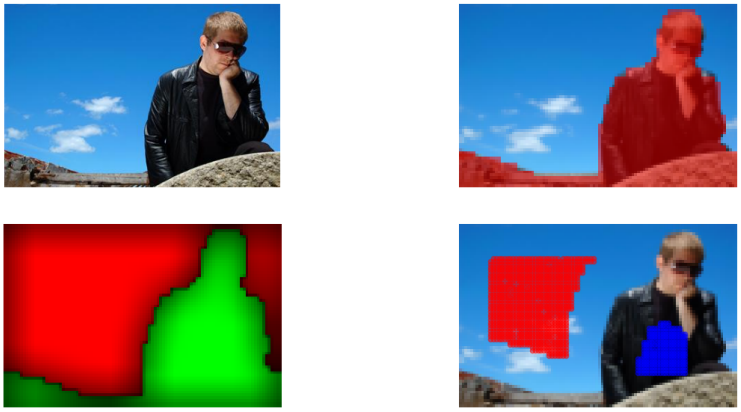


Figure 5.1: Enhanced Normalized Cuts: Image 178773

The red color in the *confidence map* indicates the pixels belonging to the background, while the green color indicates the pixels that belong to the foreground. The shadowed areas represent the pixels that have a lower confidence value. As it can be noticed, these areas are present near the boundary between the object and the background.

One pixel with the highest confidence for both the foreground and the background is then passed to the subsequent `min-cut/max-flow` algorithm.

5.3 Hyperparameter tuning

The hyperparameters have been tuned by exploiting the *Bayesian optimization* using Gaussian processes method presented in Section 3.3 too.

In this case the hyperparameter r, whose role has already been presented in Section 4.2, is fixed to $r = 10$. This choice originates from the observation that using a very high value or a very low value for this parameter leads to almost the same results. Moreover, in this way the hyperparameter space dimension is smaller, reducing the computational cost, and therefore making the the parameter tuning process faster.

Apart from σ_I and σ_d, already discussed in Section 4.2, the new hyperparameter that needs to be tuned is the *similarity factor* S. As it is shown in Section 5.1, this factor is used to adjust the weights between pixels. The *similarity factor* helps in reinforcing the boundaries identified through the `PSC` algorithm, possibly leading to more accurate boundaries in the final segmentation, particularly in complex images where the standard `Ncut` algorithm might struggle. High values of S ensure that the homogenous regions identified in the initial segmentation remain intact, providing a stable foundation for the subsequent process. On the other hand, a low value of the *similarity factor* allows the algorithm to adapt in a better way to the specific local features of the image, improving segmentation in areas where the initial clustering might be inaccurate or too coarse. Moreover this could provide the flexibility to refine and adjust segment boundaries based on the image's specific characteristics rather than strictly sticking to the `PSC` result.

The selected ranges for the hyperparameters in the *Bayesian optimization* process are:

$$\sigma_I \in [0.001,0.5],\ \sigma_d \in [1,20],\ S \in [1.01,10]\ .$$

5.4 Results

The triplet of hyperparameters obtained after the tuning process is:

$$(\sigma_I = 0.027,\ \sigma_d = 14.40,\ S = 5.58)\ ,$$

with a score $\overline{IoU} \approx 0.52$.

Various experiments have shown that the enhanced normalized cuts algorithm provides slightly improved segmentation results compared to the standard `Ncut` algorithm. In particular, the test on the same subset of 50 images presented in

Sections 3.4 and 4.3 gives a mean *Intersection over Union* value of $\overline{IoU} \approx 0.56$.

In general, the results obtained by the enhanced normalized cuts algorithm are similar to those achieved using the standard normalized cuts, as it can be noticed in Figures 5.2 and 5.3. However, in certain situations, the preliminary spectral clustering segmentation incorporated into the enhanced algorithm helps to achieve better segmentation results.

5.4.1 Comparison with `PSC` and `Ncut`

In order to be able to show the actual improvements that can be obtained using the proposed enhanced normalized cuts algorithm, in this Section the same optimal hyperparameters obtained for the standard normalized cuts are used, i.e., $\sigma_I = 0.0057,\ \sigma_d = 9.3$. The similarity factor that is here considered is $S = 2.5$.

The figures presented in this section show four images, arranged from top to bottom and left to right corresponding, respectively, to the original image, `PSC` segmentation, standard `Ncut` segmentation, and enhanced normalized cuts segmentation.

In images with complex textures or varying intensities due to the presence of, for example, shadows, the enhanced algorithm benefits from the preliminary spectral clustering result. By leveraging the preliminary segmentation, the proposed algorithm obtains a better balance between global consistency and local detail refinement, leading to more coherent segmentations in certain challenging images.

In Figure 5.4, even if a part of the background is still associated to the object, the corresponding area is smaller in the enhanced normalized cuts segmentation with respect to the standard `Ncut` segmentation.

It is possible to see in Figures 5.5, 5.6, 5.7, 5.8, and 5.9 that the inclusion of the `PSC` segmentation result in the `Ncut` algorithm leads to better results. The standard normalized cuts algorithm typically performs well in segmenting images. However, it can sometimes struggle with accurately delineating object boundaries and maintaining segment coherence. The improvement obtained using the enhanced normalized cuts algorithm is due to the increased weight given to pixels belonging to the same segment in the preliminary spectral clustering step, which helps in preserving segment coherence and improving boundary detection.

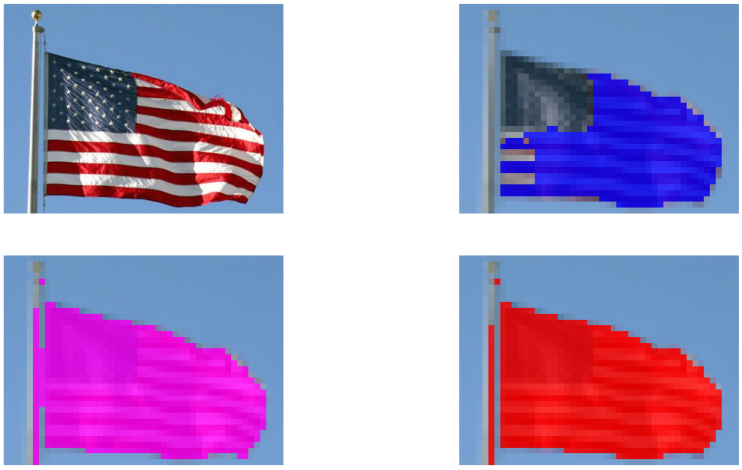


Figure 5.2: Comparison between different algorithms: Image 128465

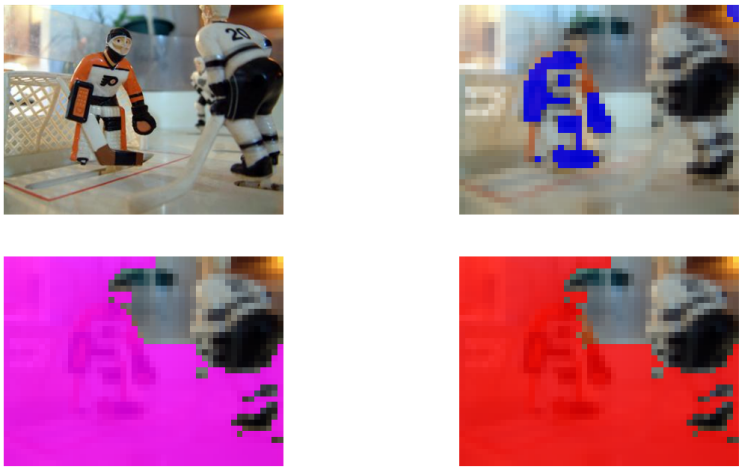


Figure 5.3: Comparison between different algorithms: Image 79799

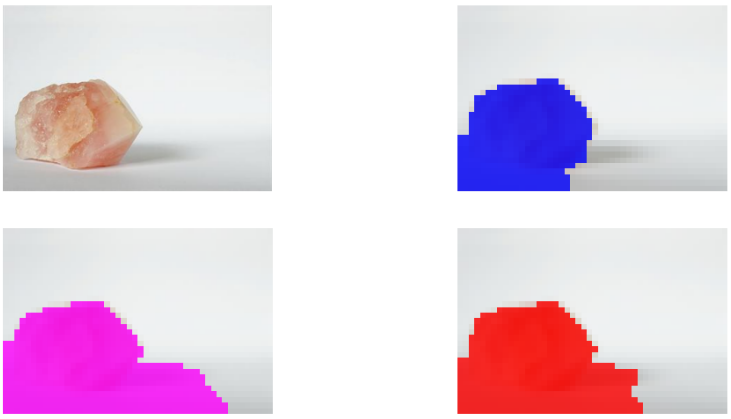


Figure 5.4: Comparison between different algorithms: Image 162706

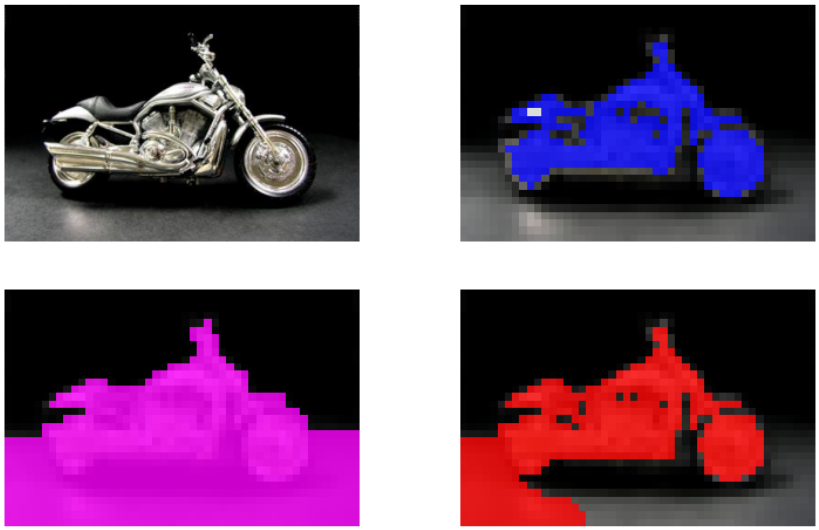


Figure 5.5: Comparison between different algorithms: Image 103452

Figure 5.6: Comparison between different algorithms: Image 137258

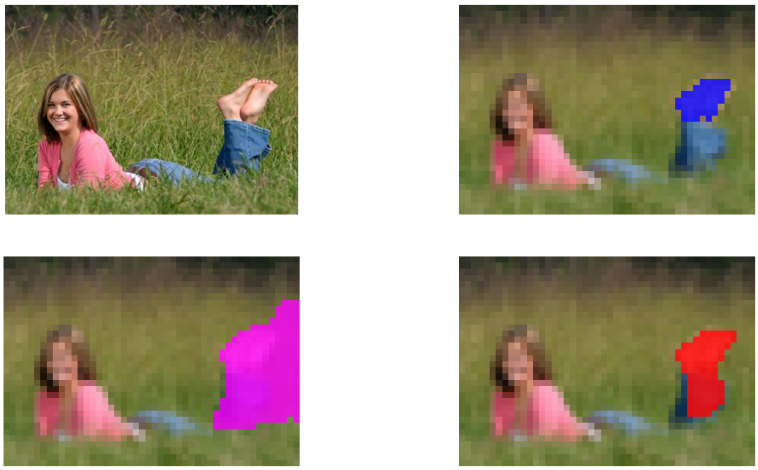


Figure 5.7: Comparison between different algorithms: Image 107659

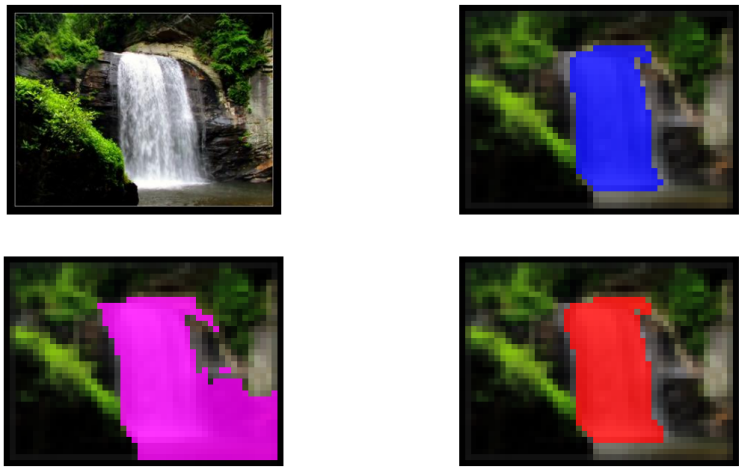


Figure 5.8: Comparison between different algorithms: Image 140145

Figure 5.9: Comparison between different algorithms: Image 152323

Despite these improvements, the results are not perfect and require further refinement. However, they represent a solid foundation for many applications, considering the diversity of the dataset used. The robustness of the enhanced normalized cuts algorithm across varied image conditions suggests its potential for broad applicability and its capability to serve as a baseline for future advancements in segmentation techniques.

5.5 Integration with `min-cut/max-flow`

The highest confidence pixels are used as seeds to a `min-cut/max-flow` algorithm, which takes inspiration from [4]. After numerous tests, it has been decided to consider as seeds the first high confidence pixel for both the foreground and the background (going from top to bottom and from left to right) encountered in the enhanced normalized cuts results.

Like the algorithms discussed in Chapters 3, 4, and 5, given a graph $\mathcal{G} = (V, E)$, where V is the set of nodes (corresponding to the pixels) and E is the set of edges (connections between pixels), each pixel p in the image corresponds to a node in V. There are two types of edges:

- **t-links**: They connect pixels to the source (foreground) or sink (background) nodes. The weight $w(p, \text{source})$ reflects the cost of assigning pixel p to the foreground, while the weight $w(p, \text{sink})$ is the cost of assigning pixel p to the background;
- **n-links**: These edges connect neighboring pixels. The weight $w(p, q)$ reflects the similarity between pixels p and q, and it is computed based on their intensity and spatial similarities.

In particular, $w(p, \text{source})$ is set to a high value if p is the highest confidence foreground pixel, while $w(p, \text{sink})$ is set to a high value if the pixel p is the highest confidence background pixel, as:

$$w(p_{foreground}, \text{source}) = \infty, \tag{5.9}$$

$$w(p_{background}, \text{sink}) = \infty. \tag{5.10}$$

Finally, the cut that minimizes the total weight of edges crossing the cut is found, partitioning the image into foreground and background.

Using the highest confidence pixels from the enhanced normalized cuts algorithm as seeds ensures that the `min-cut/max-flow` algorithm starts with reliable informations about the foreground and background regions. `min-cut/max-flow` adapts well to local image features, capturing complex boundaries and fine details of objects, which can be challenging for methods that rely only on global criteria like normalized cuts.

As already mentioned, the images in this case are resized using H' and W' as target dimensions:

$$(H', W') = (100{,}100) \ .$$

The higher quality of the images produces overall better results.

5.5.1 Refined results

After identifying the highest confidence pixels using the enhanced normalized cuts algorithm, the segmentation results are refined by applying the `min-cut/max-flow` algorithm.

This refinement process has shown to enhance in a significant way the segmentation results by ensuring that the final partitions are both globally consistent and locally accurate. In particular, the algorithm is particularly effective in dealing with complex textures and intensity variations within an image, as it globally optimizes the segmentation based on the flow network, therefore reducing the effects of local noise and texture variations. Even in challenging conditions, such as images with low contrast or intricate details, the refined results show a marked improvement over the initial segmentation.

In Figures 5.10 and 5.12 the biggest and brighter pixels represent the ones selected as seeds for both the foreground (red pixel) and background (blue pixel).

The integration of the `min-cut/max-flow` algorithm as a post-processing step in the enhanced normalized cuts framework improves the overall quality of segmentation. This combined approach not only refines the segmentation boundaries and increases segment coherence but also provides robust results across diverse and challenging images, like the ones presented in Figures 5.11 and 5.13.

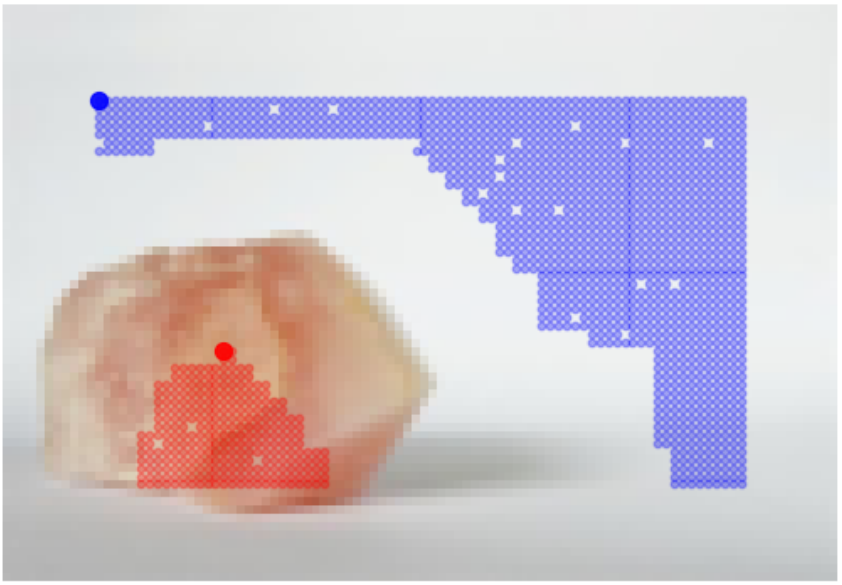


Figure 5.10: Highest confidence pixels: Image 162706

Figure 5.11: `min-cut/max-flow`: Image 162706

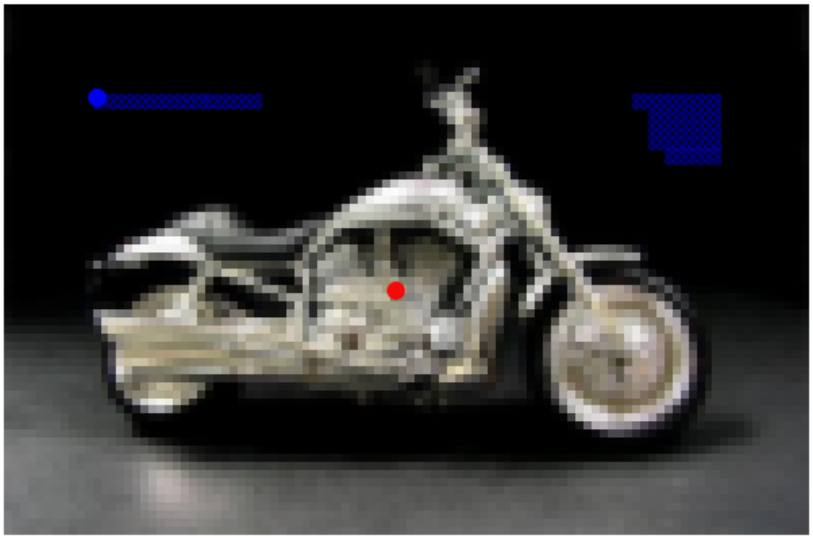


Figure 5.12: Highest confidence pixels: Image 103452

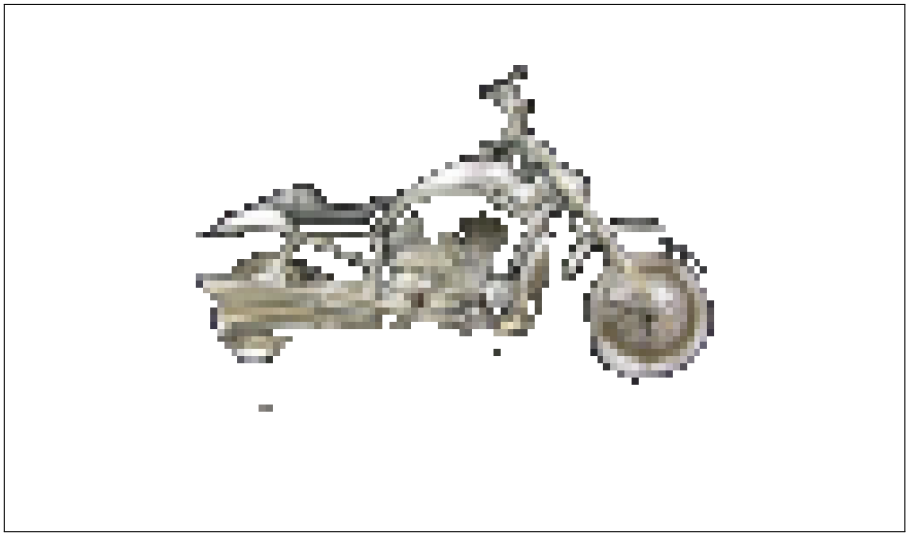


Figure 5.13: `min-cut/max-flow`: Image 103452

Chapter 6

Real-time video segmentation using Normalized Cuts

6.1 Normalized Cuts for video segmentation

The computational complexity of `Ncut`, presented in Chapter 4, makes it difficult to adapt this algorithm for video segmentation. Constructing a graph where each pixel is a node and each edge represents the similarity between pixels is computationally intensive. The number of edges increases quadratically with the number of pixels, making this process highly demanding for high-resolution videos.

Given the computational demands of normalized cuts, the following adaptations were implemented to make it feasible for real-time video segmentation:

- **Frame resizing**: Images are resized to a much smaller target size using the resizing method proposed in Section 3.1, using $(H'', W'') = (25, 25)$ as target dimensions;
- **Uniform background**: A plain, single-colored background reduces the complexity of the segmentation task by minimizing the number of distinct regions in the image;
- **Bright glove**: A glove with a bright color ensures that the hand stands out clearly against the background, facilitating easier detection and segmentation;
- **Real-time constraints**: To achieve real-time performance, additional optimizations are considered, like lowering the frame resolution to $(H''', W''') =$ (400,600) and processing only every n-th frame if necessary.

6.2 Issues and challenges

One important challenge encountered with this implementation is the inherent variability in the labels produced by the normalized cuts algorithm across consecutive frames. `Ncut` labels segments based on eigenvector values, which can vary slightly between frames due to many factors, like changes in lighting, object movement, and so on. As a result, there can be frequent switches between foreground and background labels. This issue occurs as a flickering effect in the real-time segmented video, where regions identified as foreground in one frame may be labeled as background in the next frame, and vice versa, as it is possible to notice in the sequence of frames shown in Figure 6.1. This instability is a critical challenge for real-time applications.

Figure 6.1: Video segmentation issue

6.3 Results

To test the effectiveness of the segmentation algorithm, various experiments have been conducted using both bare hands and bright-colored gloves against different backgrounds.

The segmentation results obtained testing bare hands on a white background are not perfect, as presented in Figure 6.2. Shadows and varying lighting conditions significantly affect the segmentation performance. The presence of shadows causes the algorithm to mislabel shadowed regions as part of the foreground, leading to inconsistent and inaccurate segmentation.

A similar observation can be made when using a red glove on the same background, as shown in Figure 6.3.

Figure 6.2: Video segmentation: Bare hands on white background

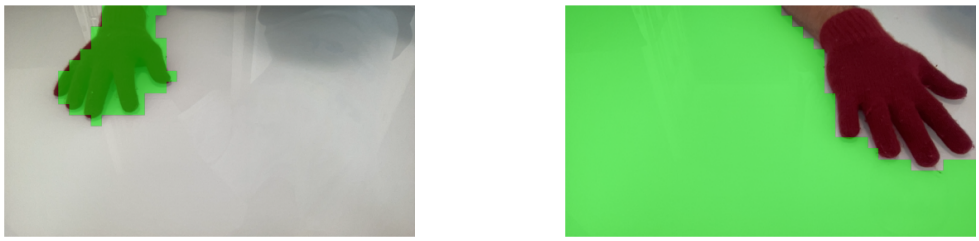


Figure 6.3: Video segmentation: Red glove on white background

Particularly good results are obtained using a bright, blue colored glove on the same white background, as it is possible to notice in Figure 6.1. Moreover, Figure 6.4 shows that the presence of other objects in the scene does not significantly affect the segmentation accuracy, indicating robust performance in this scenario.

Clearly, using a glove which is similar in color to the background leads to poor results due to the lack of contrast between the object and the background, as it is possible to see in Figure 6.5. However the results are not great even when using bare hands on the red background (Figure 6.6). The algorithm is unable to consistently differentiate between the skin tone and the red background.

Figure 6.4: Video segmentation: Blue glove on white crowded background

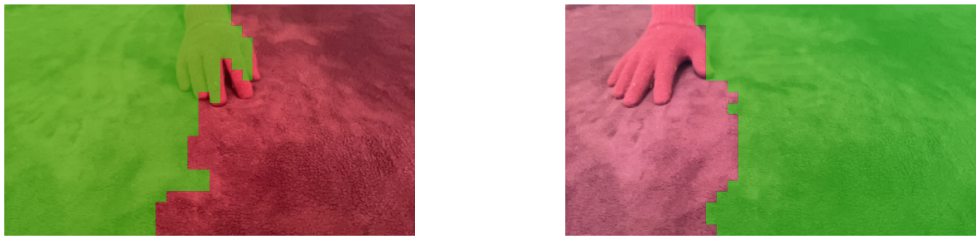


Figure 6.5: Video segmentation: Red glove on red background

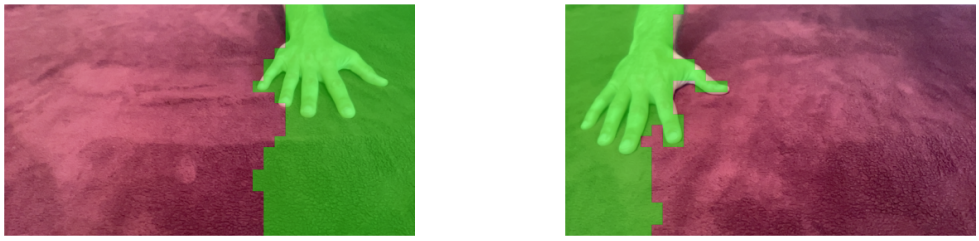


Figure 6.6: Video segmentation: Bare hands on red background

On the other hand, the `Ncut` algorithm performs well with both black and blue gloves on the red background, as shown in Figures 6.7 and 6.8. The high contrast between the object and the background allows normalized cuts to accurately identify and segment the hand.

The results demonstrate that the proposed algorithm performs best in scenarios where there is a high contrast between the object of interest and the background. For instance, using a blue glove on a red or a white background produces consistent and accurate segmentation results. The clear differentiation in color allows `Ncut` to easily identify and separate the hand from the background.

Figure 6.7: Video segmentation: Blue glove on red background

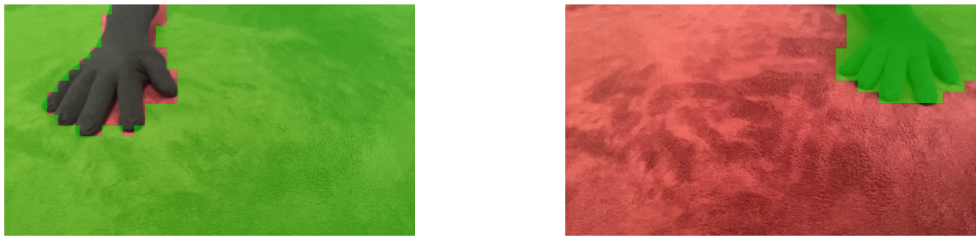


Figure 6.8: Video segmentation: Black glove on red background

Chapter 7

Conclusion

This work explores and evaluates different advanced image segmentation techniques, including preliminary spectral clustering, normalized cuts, enhanced normalized cuts, and the application of `Ncuts` for real-time video segmentation. The results demonstrate the potential and limitations of each method, giving useful insights for future research and uses in image processing

Preliminary spectral clustering, together with normalized cuts, serves as the foundation for the enhanced normalized cuts method. By leveraging the spectral properties of the *Laplacian matrix* corresponding to the image graph, `PSC` is able to identify the primary segments within the image. Even if the results from preliminary spectral clustering alone are often incomplete, capturing only small parts of the object of interest, they provide important initial segmentations that can be refined further.

Contrary to preliminary spectral clustering, which often identifies smaller segments, normalized cuts is able to consider larger portions of the object. This is because `Ncut` uses the *Normalized Cut* as measure of dissociation, which balances the cut cost with the total connection strength within each segment, rather than using a simple cut criterion, that tends to segments isolated points or small clusters of pixels. While effective in many scenarios, standard normalized cuts occasionally struggles with complex images, particularly those with intricate textures.

In order to try addressing the limitations of the standard `Ncut`, we introduced an enhanced normalized cuts algorithm that incorporates the preliminary spectral clustering segmentation results as a prior. The integration of this prior helps to guide the algorithm, resulting in better object detection and boundary delineation. The results that have been obtained indicate that while both methods perform similarly in many cases, the enhanced approach significantly improves the segmentation

accuracy in challenging images, i.e., in scenarios with complex textures and diverse image content. The final refinement is obtained through a `min-cut/max-flow` algorithm. This method uses the high-confidence pixels identified from the enhanced normalized cuts results as seeds. This step further optimizes the segmentation by ensuring globally optimal cuts, effectively refining object boundaries.

Finally standard normalized cuts has been adapted for real-time video segmentation. The experiments revealed that high contrast between the foreground object and the background is essential for an effective segmentation. The algorithm struggles in low-contrast scenarios, such as red gloves on a red background or bare hands on the same background, leading to poor results. Shadows and varying lighting conditions also pose important challenges. Despite these issues, the algorithm shows robustness in simpler settings with bright gloves and plain or moderately textured backgrounds, even in the presence of other objects, providing accurate and consistent results.

Chapter 8

Future work

While the proposed enhanced normalized cuts and real-time video segmentation methods have shown promising results, their computational complexity remains a challenge. Future work should focus on optimizing these algorithms for speed and efficiency. Techniques such as parallel processing or GPU acceleration could be explored to make the proposed methods faster, and therefore more suitable also for real-time applications without compromising the segmentation quality.

Once a better computational efficiency is reached, the hyperparameter tuning step for the various algorithms could be performed using the entire *MSRA10K* dataset, possibly using cross-validation to obtain more robust results. Moreover, using diverse datasets with different image content, such as medical imaging datasets, satellite imagery, and autonomous driving datasets, cold help generalize even more the final results. An ambitious final goal could be to implement an adaptive algorithm that can dynamically adjust all the parameters based on the image content without reducing the final segmentation quality.

For real-time segmentation, incorporating temporal smoothing techniques can help stabilize the segmentation across consecutive frames, possibly reducing the flickering effect observed, in particular, in low-contrast scenarios. Additionally, implementing shadow removal methods and improving the handling of varying lighting conditions is important to address the challenges posed by shadows and lighting variations, particularly in scenarios involving bare hands or similar foreground-background colors. Finally, incorporating a user feedback mechanisms to fine-tune the segmentation in real-time could enhance the overall algorithm's accuracy.

Appendix A

Code structure overview

Here a general overview of the structure of the Python code used in this work is provided. We use a functional programming approach, focusing on specific tasks through individual functions rather than object-oriented programming with classes. This design choice makes the code modular and easy to maintain, as each function is responsible for a distinct part of the workflow. There are three main scripts and several submodules contained within a subfolder named **sp_segm**.

The three main scripts are:

- **main_ncut.py**: In this script the single image name from the considered dataset has to be specified in order to be able to call the identified image inside the **sp_segm** subfolder. It uses the highest confidence pixels from the enhanced normalized cuts results to obtain the final refined segmentation through the `min-cut/max-flow` algorithm. The final result is stored in the **results** subfolder;
- **visualization_image.py**: It provides the visualization of the original image and the segmentation results obtained using the preliminary spectral clustering, `Ncut`, and enhanced normalized cuts algorithms;
- **video_segmentation.py**: This handles the real-time video segmentation task using normalized cuts, capturing the frames with the local device camera.

The scripts contained inside the **sp_segm** subfolder are:

- **Bayes_opt_param.py**: This script conducts *Bayesian optimization* to obtain the optimal hyperparameters for the various segmentation algorithms;
- **param_evaluation.py**: Given the algorithm and the chosen hyperparameters, this script provides the IoU score on a test set;

- **spectral_segmentation.py**: It performs the preliminary spectral segmentation;
- **Ncut_prior_rgb.py**: Implementation of the enhanced normalized cuts algorithm using the results from the '**spectral_segmentation.py**' script as a prior;
- **Normalized_cut.py**: It contains the implementation of the standard normalized cuts algorithm;
- **res_visualization.py**: This includes the function used to display the original image and the various segmentation results in the **visualization_image.py** script;
- **Ncut_video.py**: It provides the function used in **video_segmentation.py** for real-time video segmentation;
- **take_photo.py**: This script captures an image using the local device camera and stores it in the **hands** subfolder, as it has been used to capture photos of an hand on different backgrounds.

Bibliography

[1] Jianbo Shi and J. Malik. «Normalized cuts and image segmentation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905. DOI: 10.1109/34.868688. URL: https://doi.org/10.1109/34.868688 (cit. on pp. 1, 6, 23, 26).

[2] Ravindra Hegadi, Jonathan L. Gross, George C. Stockman, Linda G. Shapiro, and Xiaoping Cheng. «A Survey on Traditional and Graph Theoretical Techniques for Image Segmentation». In: 2014. URL: https://api.semanticscholar.org/CorpusID:8894823 (cit. on p. 5).

[3] Faliu Yi and Inkyu Moon. «Image segmentation: A survey of graph-cut methods». In: *2012 International Conference on Systems and Informatics (ICSAI2012)*. IEEE, May 2012. DOI: 10.1109/icsai.2012.6223428. URL: http://dx.doi.org/10.1109/ICSAI.2012.6223428 (cit. on p. 5).

[4] 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: http://dx.doi.org/10.1007/s11263-006-7934-5 (cit. on pp. 5, 39).

[5] Qiuhua Zheng, Wenqing Li, Weihua Hu, and Guohua Wu. «An Interactive Image Segmentation Algorithm Based on Graph Cut». In: *Procedia Engineering* 29 (2012), pp. 1420–1424. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2012.01.149. URL: http://dx.doi.org/10.1016/j.proeng.2012.01.149 (cit. on p. 6).

[6] Zili Peng, Shaojun Qu, and Qiaoliang Li. «Interactive image segmentation using geodesic appearance overlap graph cut». In: *Signal Processing: Image Communication* 78 (Oct. 2019), pp. 159–170. ISSN: 0923-5965. DOI: 10.1016/j.image.2019.06.012. URL: http://dx.doi.org/10.1016/j.image.2019.06.012 (cit. on p. 6).

[7] Chongyang Zhang, Guofeng Zhu, Minxin Chen, Hong Chen, and Chenjian Wu. *Image Segmentation Based on Multiscale Fast Spectral Clustering*. 2018. DOI: 10.48550/ARXIV.1812.04816. URL: https://arxiv.org/abs/1812.04816 (cit. on p. 6).

[8] Deepkiran, Mrinal Pandey, and Laxman Singh. «Spectral Segmentation Augmented with Normalized Cuts for Detection of Early Blight Disease in Potato». In: *Procedia Computer Science* 233 (2024), pp. 1034–1043. ISSN: 1877-0509. DOI: 10.1016/j.procs.2024.03.292. URL: http://dx.doi.org/10.1016/j.procs.2024.03.292 (cit. on p. 6).

[9] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. DOI: 10.48550/ARXIV.1206.2944. URL: https://arxiv.org/abs/1206.2944 (cit. on pp. 7, 14).

[10] Eric Brochu, Vlad M. Cora, and Nando de Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. 2010. DOI: 10.48550/ARXIV.1012.2599. URL: https://arxiv.org/abs/1012.2599 (cit. on p. 7).

[11] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312 (cit. on p. 11).

[12] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. «The Pascal Visual Object Classes (VOC) Challenge». In: *International Journal of Computer Vision* 88.2 (Sept. 2009), pp. 303–338. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4. URL: http://dx.doi.org/10.1007/s11263-009-0275-4 (cit. on p. 15).

[13] Thomas J. Santner, Brian J. Williams, and William I. Notz. «Space-Filling Designs for Computer Experiments». In: *The Design and Analysis of Computer Experiments*. Springer New York, 2018, pp. 145–200. ISBN: 9781493988471. DOI: 10.1007/978-1-4939-8847-1_5. URL: http://dx.doi.org/10.1007/978-1-4939-8847-1_5 (cit. on p. 15).

[14] M. D. McKay, R. J. Beckman, and W. J. Conover. «Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code». In: *Technometrics* 21.2 (May 1979), pp. 239–245. ISSN: 1537-2723. DOI: 10.1080/00401706.1979.10489755. URL: http://dx.doi.org/10.1080/00401706.1979.10489755 (cit. on p. 15).

[15] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. «Global Contrast based Salient Region Detection». In: *IEEE TPAMI* 37.3 (2015), pp. 569–582. DOI: 10.1109/TPAMI.2014.2345401 (cit. on p. 17).

[16] Ming-Ming Cheng, Jonathan Warrell, Wen-Yan Lin, Shuai Zheng, Vibhav Vineet, and Nigel Crook. «Efficient Salient Region Detection with Soft Image Abstraction». In: *IEEE ICCV*. 2013, pp. 1529–1536 (cit. on p. 17).

[17] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. «Salient Object Detection: A Survey». In: *ArXiv e-prints* (2014). arXiv: `arXiv:1411.5878` (cit. on p. 17).

[18] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. «Salient Object Detection: A Benchmark». In: *IEEE TIP* 24.12 (2015), pp. 5706–5722. DOI: `10.1109/TIP.2015.2487833` (cit. on p. 17).