Graphical Models Project:
Graph Cut for multi-label image segmentation

Pierrick Bournez, Antonine Batifol

March 2025

1 Introduction

1.1 Motivation

Image segmentation is a fundamental problem in computer vision, with applications ranging from medical imaging to object recognition. Traditional neural network models, such as CNNs, require large amount of labeled data and extensive computational resources to perform image segmentation. In contrast graph-cut based segmentation can perform one-shot segmentation and also handle multi-label segmentation. In this project, we explore graph cut algorithm to perform multi-label segmentation.The code is available on GitHub

1.2 Objective

In this project, we implemented the following:

- Reimplementation of the max-flow/min-cut algorithm and comparison of our implementation against networkx and pymaxflow python modules.
- Multi-label segmentation with α-expansion.
- Comparison of performance with different pairwise and unary costs
- Graph recycling to improve computational efficiency.
- Extensions: interactive binary (background/foreground) and multi-label segmentation

2 Problem Formulation

2.1 Graph Cut and Energy Minimization

Image segmentation problem can be expressed as an energy minimization problem, where image are represented as a directed graph $G = (V, E)$, with V the set of vertices corresponding to the image pixels, and E the edges. We consider a 4-connected neighborhood, where each pixel is connected with its right, left, above and below neighbours (except on the edges). Those connections are represented by the edges. If we denote by $\mathcal{L}$ the set of labels, n the number of pixels and $x \in \mathcal{L}^n$ a possible labeling we would like to minimize a energy $E : \mathcal{L}^n \to \mathbf{R}$ that maps any possible labelling to a global cost.

The energy function typically adopted in computer vision, and considered herein, is expressed as a sum of unary (ϕ_i) and pairwise (ϕ_{ij}) potential terms:

$$E(\mathbf{x}) = \sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j).$$

The **unary potential** $\phi_i(x_i)$ acts as a data fidelity term. It represents the cost of the assignement of a pixel to a label. Conversely, the **pairwise potential** $\phi_{ij}(x_i, x_j)$ enforces spatial coherence by penalizing label differences between neighboring pixel.
We detail the different unary potentials we considered in Section 3.1. For the pairwise potentials, we restrict ourselves to potential models like Potts model with $\gamma = 200$. The various formulations for unary potentials explored in this work are described in Section 3.1. Regarding the pairwise potentials, we adopt a simplified model based on the Potts model, with a fixed penalty parameter $\gamma = 200$. This model is defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \gamma & \text{if } x_i = x_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

2.2 α-expansion Algorithm

Since most multi-label energy functions are not submodular, proposed algorithms only focused on finding approximate or partially optimal solution. In this project, we reimplemented the α-expansion algorithm. At each iteration we choose a label, and apply graph cut to find the optimal expansion that decreases the energy [2]. At it each iteration we consider a binary sub-problem and separate the pixel between being labeled as α or keeping their old label. To do so we construct a new graph [1] at each iteration introducing two additional nodes, the **source** (α) and **sink** ($\overline{\alpha}$), as well as auxiliary nodes. The weights of the edges represent the unary and pairwise costs as defined below in Table 1. We then compute the maxflow and each node is assigned a label depending of the partition of the mincut he belongs to (label as α if it belong to the source partition and keep, old label

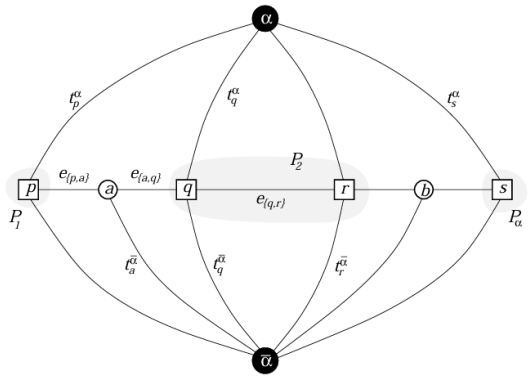


Figure 1: Alpha-expansion Graph

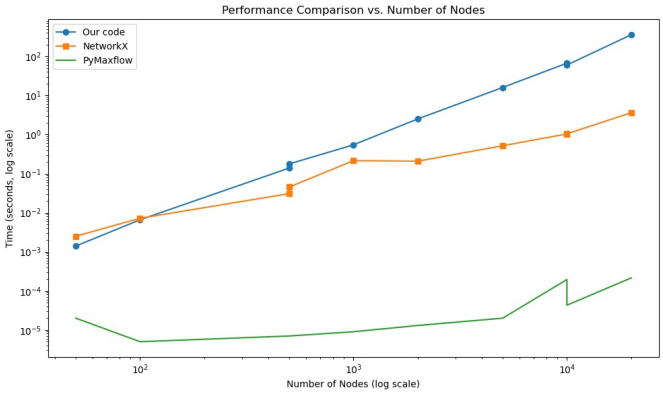


Figure 2: Maxflox Executation Time Comparison

otherwise). We perform these expansion moves until the energy stops decreasing.

Edge	Weight	For
$t_{\bar{\alpha}}^p$	∞	$p \in \mathcal{P}_\alpha$
$t_{\bar{\alpha}}^p$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_α^p	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	$\{p,q\} \in \mathcal{N}, f_p \neq f_q$
$t_{\bar{\alpha}}^a$	$V(f_p, f_q)$	$\{p,q\} \in \mathcal{N}, f_p \neq f_q$
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p = f_q$

Table 1: Edges weights and conditions in the α-expansion graph

2.3 Max-flow/Min-cut with Ford-Fulkerson Algorithm

We used Ford-Fulkerson to calculate the maxflow from the source s to the sink t and find the minimum S/T-cut.

Algorithm:

1. Initialize flow $f(u, v) = 0$ for all edges.
2. While there exists an **augmenting path** from source s to sink t in the residual graph:
   - We find an augmenting path using BFS
   - Compute the minimum residual capacity along the path).
   - Increase flow along this path by this capacity.
   - Update the residual capacities: $c_f(u, v) = c(u, v) - f(u, v)$ and $c_f(v, u) = f(u, v)$ (reverse flow)
3. Repeat until no more augmenting paths exist.
4. Find the edges that have no residual capacity and are thus part of the cut.

For the project we reimplemented the max-flow algorithm from scratch using only basic python libraries like numpy. We then compare in Figure 2 our performance against other libraries like pymaxflow which is a python wrapper for the C++ library and networkx. We compare both the execution time and the maxflow value to ensure correctness of our implementation for multiple nodes number, keeping the same type of 4-connected network.

While our implementation consistently produced correct max-flow values, it exhibited significantly slower execution times compared to the other libraries. For instance, processing a graph with 20,000 nodes required over six minutes, making it unsuitable for practical use, even for low-resolution images (e.g., $256 \times 256 \approx 65\,000$ nodes), which would entail prohibitively long runtimes.

Consequently, for the remainder of the project, we utilize `pymaxflow` due to its high efficiency stemming from its underlying C++ implementation. This performance limitation illustrates one of the challenges we encountered when developing graph algorithms in high-level interpreted languages such like Python.

3 Results

One of the other difficulty we encountered was finding the proper unary and pairwise costs, that yield good quality segmentation.

3.1 Unary Potential

We evaluated different formulations for the unary potential:

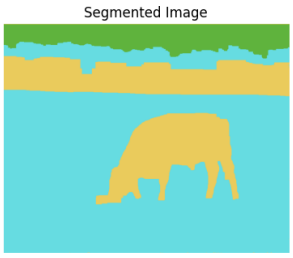
- **Semi-automatic approach:** Using the k-means clustering method, we computed histograms in the pixel space for each label. These histograms were then modeled as Gaussian distributions, which were used to define the corresponding unary terms.
- **L2-based potential:** Given the reference colors of each label, we computed the L2 norm in a color

metric space between the assigned label colors and the actual pixel colors. Initially, this was performed in the standard RGB space. However, a limitation of the RGB space is that perceptually similar colors may not be close in Euclidean distance [3]. To address this, we experimented with an alternative metric space that accounts for perceptual color differences: the CIELAB color space.

The results of the L2 norm in RGB and CIELAB spaces are presented in Figures 4 and 5, respectively.

Overall, the results are promising. Applying the L2 norm in both RGB and CIELAB spaces improved segmentation accuracy, successfully identifying structural elements such as the cow's legs. The CIELAB-based unary potential further enhanced segmentation, even correctly identifying the cow's horns.

However, increasing the number of labels introduced more noise. As a consequence, parts of the cow's back and some regions of the grass were misclassified. These findings highlight the sensitivity of color-based segmentation using graph cuts to input parameters, unary potential formulations, and hyperparameter tuning, emphasizing the importance of proper scaling and term weighting.



3 labels

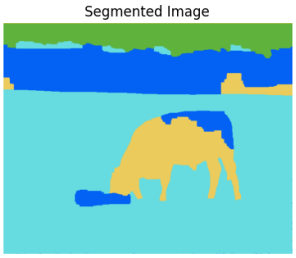
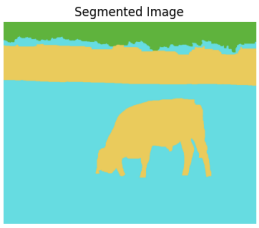
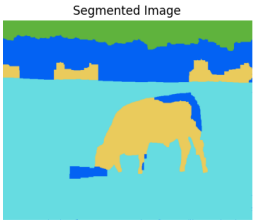


Figure 3: 4 labels

Figure 4: Results of the cow using 3 and 4 label with the L2 norm



3 labels



4 labels

Figure 5: Results of the cow using 3 and 4 label with the L2 norm in the CILEAB space

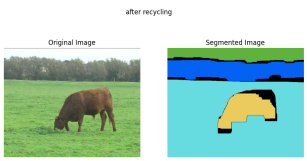
4 Energy Minimization with Graph Recycling

To enhance the efficiency of energy minimization, we implemented the graph recycling algorithm proposed in [1]. Following the approach introduced by Kovtun, the method aims to reduce the number of unknown variables by solving a set of auxiliary binary problems P_m, one for each label $l_m \in \mathcal{L}$. This strategy allows partial labeling of the image before applying more computationally intensive multi-label optimization. We used the same unary potential as defined in the original article

Graph Recycling Algorithm

1. Initialize all pixels as unlabeled (ϵ).
2. For each label $\alpha \in \mathcal{L}$, solve the binary subproblem:
   - Assign label α to pixel i if $x_i^\alpha = 1$.
3. Update unary potentials for remaining unlabeled pixels and apply α-expansion on this subset.

Figure 7 illustrates the results of this procedure. After the recycling step, the majority of pixels are already labeled, significantly reducing the number of variables requiring optimization during the subsequent α-expansion phase. This leads to a substantial reduction in computational complexity while maintaining segmentation quality.



Input image and labeling after the recycling step. Black pixels indicate unlabeled regions.

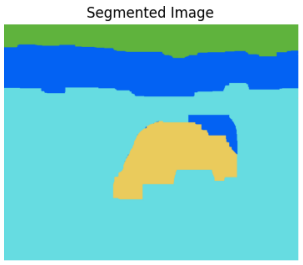


Figure 6: Segmentation with 4 labels

Figure 7: Segmentation results using graph recycling followed by α-expansion applied only to the unlabeled pixels.

5 Interactive image segmentation

We implemented an extension of our previous work where we allow user interaction and used pairwise costs based on pixel value instead of only pixel labelling. At the beginning the user must indicate with small brush some pixels that belong to the the background ($\mathcal{B}$) and others that belong to the foreground ($\mathcal{F}$). These pixels, named seeds will be linked to the terminal nodes, which

represent background (the sink) and foreground (the source).We implemented a simplified version where only seeds get strong unary cost. The pairwise cost between to pixel p and q was defined as $V_{\{p,q\}} = \frac{1}{1+\|I(p)-I(q)\|}$ where I map a pixel to its color value and $K = 10^9$. The graph is defined below.

type	edge	cost	for
n-link	$\{p, q\}$	$V_{\{p,q\}}$	$\{p, q\} \in N$
t-link	{p, S}	0	$p \notin \mathcal{F}$
	{p, S}	K	$p \in \mathcal{F}$
	{p, T}	0	$p \notin \mathcal{B}$
	{p, T}	K	$p \in \mathcal{B}$

Likewise we compute the maxflow and find the min-cut to assign the labels. Nodes that belong to the source partition will be labeled as foreground and background otherwise.

User brush Segmented image

Figure 8: Results of interactive binary segmentation

We show the results of our algorithm in Figure 8 We also observed the influence of position in Figure 9. We found that the algorithm is quite stable and the results do not seem to depend on seed positioning except when the object has complex borders and thin contour. In this case we benefit from user interaction, be redefining more precisely the brush in zones where the algorithm encountered more difficulties.

Figure 9: Influence of seeds position

We extend this method to multi-label segmentation as you can see in Figure 10.

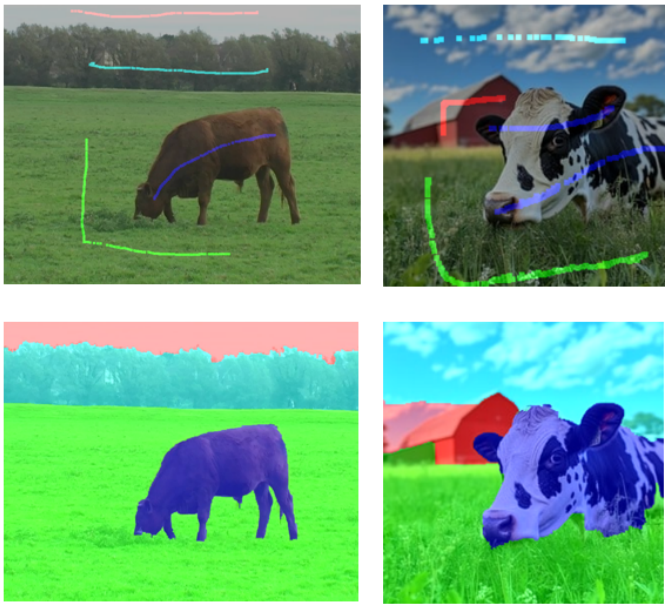


Figure 10: Results Multi-label interactive segmentation

As the interactive segmentation was only an extansion of our model, we implement a simple model which could benefit from more complex unary costs, that also take into account non-seeds pixels.

6 Conclusion

We successfully applied the α-expansion algorithm and graph recycling strategies to perform multi-label segmentation within a reasonable computational time.

In this work, we adhered to the same qualitative evaluation criteria as defined in the original study. Commonly used image segmentation datasets are primarily designed for object segmentation tasks, which rely on complex priors and high-level semantic cues, thereby requiring more than color-based energy terms alone. Consequently, when such datasets are used in the context of purely color-based segmentation, the resulting masks often encompass large, semantically inconsistent regions that do not correspond to any meaningful object boundaries. Finding a dataset that encompasses these color priors constitutes an interesting and valuable direction for future research.

Furthermore, we encountered significant challenges in the definition and tuning of the unary and pairwise cost functions. These terms are inherently image-dependent and lack intuitive parametrization, which often led to suboptimal visual segmentations. Systematic approaches for estimating these parameters merit further investigation.

References

[1] K Alahari, P Kohli, and P H S Torr. Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1846–1857, October 2010.

[2] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[3] Bernhard Hill, Th Roger, and Friedrich Wilhelm Vorhagen. Comparative analysis of the quantization of color spaces on the basis of the cielab color-difference formula. *ACM Transactions on Graphics (TOG)*, 16(2):109–154, 1997.