Graphical Models Project

Original article: Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs
Implementation of Graph-cut for image segmentation

Antonine Batifol, Pierrick Bournez

31st March 2025

Table of Contents

Introduction

Objectives

We implemented the following image color-based segmentation algorithm

- Multi-label segmentation with α-expansion
- Maxflow/min-cut reimplementation and benchmark
- Implementation of Graph recycling
- Extension: Image Segmentation leveraging graph cut
  - Interactive Binary Segmentation (background/foreground)
  - Multi-label segmentation with alpha expansion

Original Image

Segmented Image

Image Segmentation: Modeling as an Energy Minimization Problem

Energy minimization problem

Input: Image I with a set of colored pixels P, a set of label L
Goal: Find $f_p : P \to L$ the best label assignement, solution of the energy minimization problem :

$$E(f) = \sum_{p \in P} D_p(f_p) + \sum_{\{p,q\} \in N} V(f_p, f_q), \tag{1}$$

where:

- $D_p(f_p)$ is the **unary cost/potential** of pixel p , measuring how well label f_p fits pixel p.
- $V(f_p, f_q)$ is the **pairwise cost/potential**, enforcing smoothness by penalizing differences between neighboring pixels.
- N is the set of neighboring pixels (4-connected neighborhood)

Unary and pairwise cost

Unary Cost

For each label $\alpha \in L$, we define a principal color

- We compute the L_2-distance in the RGB and CIELAB space between the pixel and the reference color of label $\alpha \in L$
- Based on colors distribution and histograms

Pairwise cost

We adopt a pott model with $\gamma = 200$:

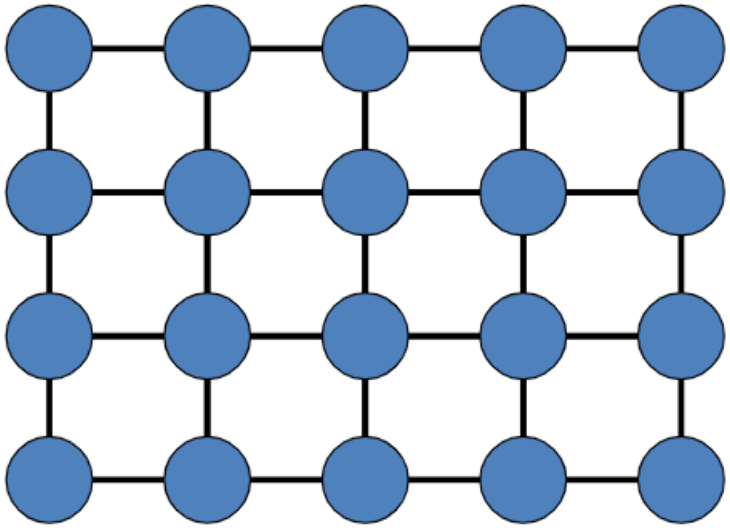
$$V(f_i, f_j) = \begin{cases} 0 & \text{if } f_i = f_j, \\ \gamma & \text{otherwise,} \end{cases} \tag{2}$$

Image segmentation as an energy minimization problem

Image representation

We represent a image with a weighted graph $G = (V, E)$, where:

- V is the set of nodes corresponding to each pixels.
- E the set of edges that connect neighboring pixels. Weighted edges representing the pairwise cost



α-expansion algorithm

α-expansion: iterative graph-cut for multi-label segmentation

Main idea:

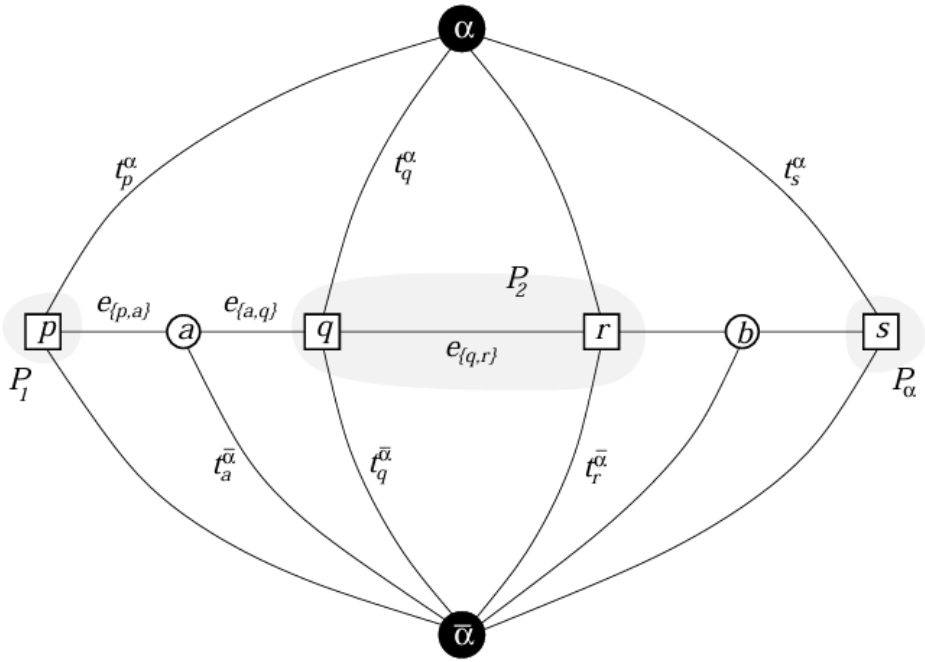
- Break the minimization problem into binary sub-problem
- Minimizing the binary sub-problem is roughly equivalent to finding the mincut in a graph.

1. **Iterate** over each label α in $\mathcal{L}$.
2. **Construct binary graph-cut:**
   - Each pixel keeps its label or switches to α.
   - Add **auxiliary nodes** between pixel with different label
3. **Graph structure:**
   - **Nodes:** Pixels
   - **Edges:**
     - **Terminal edges** $\rightarrow$ Unary cost $D_p(L_p)$.
     - **Neighbor edges** $\rightarrow$ Pairwise cost $V_{p,q}(L_p, L_q)$.
     - **Auxiliary nodes** $\rightarrow$ cost of binary labeling α or $\hat{\alpha}$
4. **Solve min-cut** $\rightarrow$ Determine optimal expansion move.
5. **Update labels** and repeat until convergence (energy decreases)

α-expansion Algorithm

We add **auxiliary nodes** between two neighbours pixel that have different labels and create two new edges

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



Alpha expansion graph

Max-Flow / Min-Cut with Ford-Fulkerson Algorithm

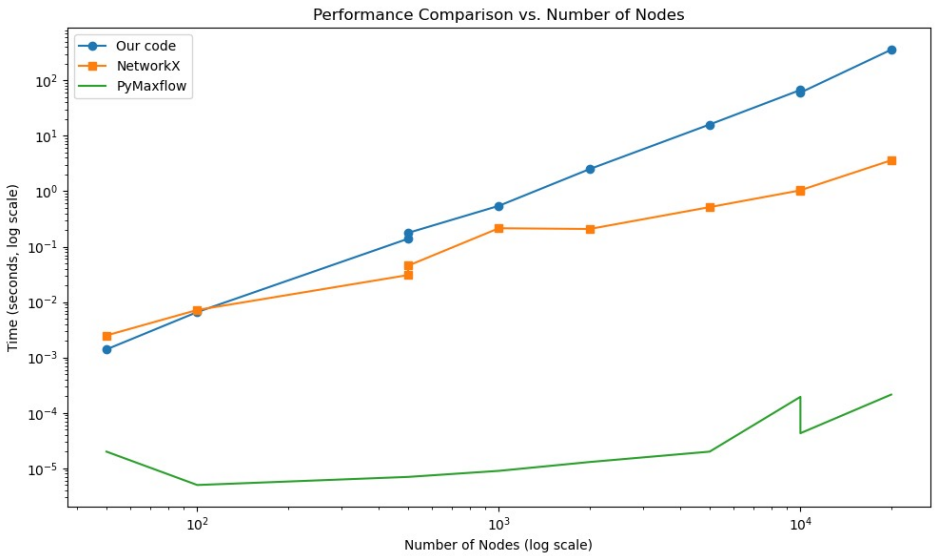
The maximum flow from s to t is equal to the minimum capacity of an S/T - cut. We use Ford-Fulkerson to calculate the maxflow.

Algorithm:

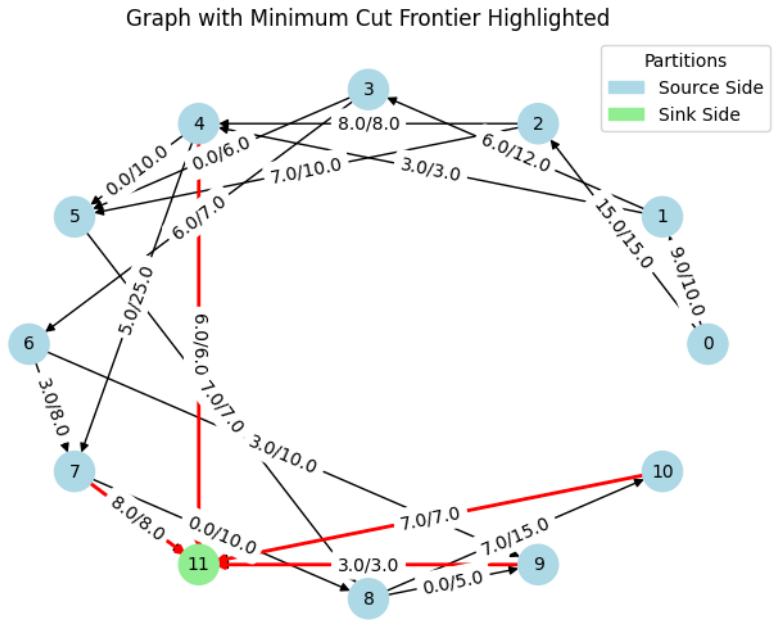
1. Initialize flow $f(u, v) = 0$ for all edges.
2. While there exists an **augmenting path** from source s to sink t in the residual graph:
   - Find an augmenting path using BFS or DFS.
   - Compute the minimum residual capacity along the path).
   - Increase flow along this path by this capacity.
   - 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.

Performance of our maxflow implementation

Since we use python with basic libraries like numpy to construct the graph and reimplement the maxflow algorithm, our code is very slow compared to other libraries like pymaxflow which is a python wrapper for the C++ library.



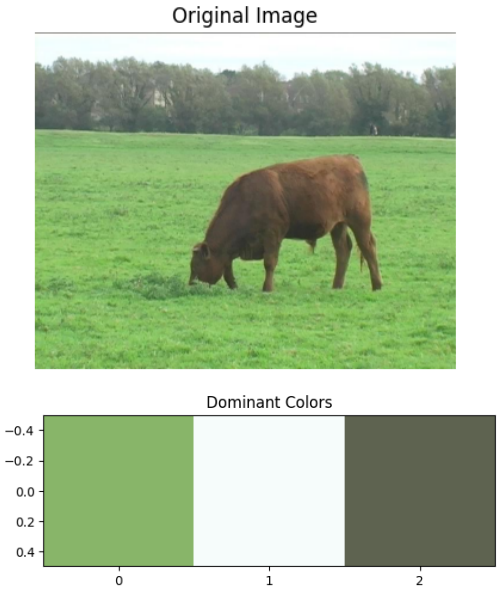
Execution Time Comparison for maxflow computation



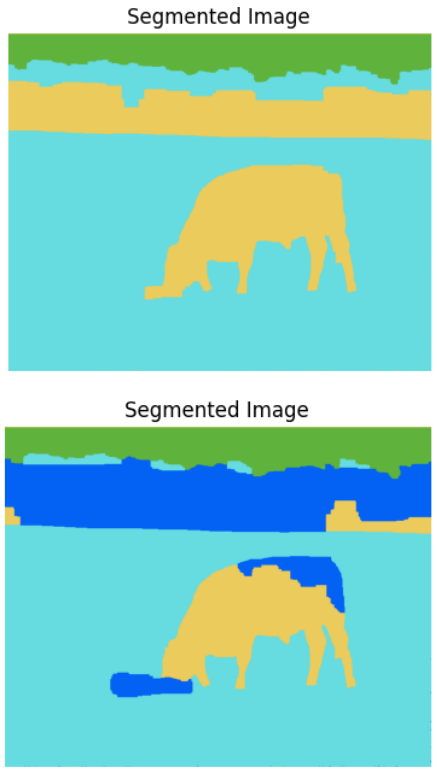
Example of graphcut with our code

α-expansion Results

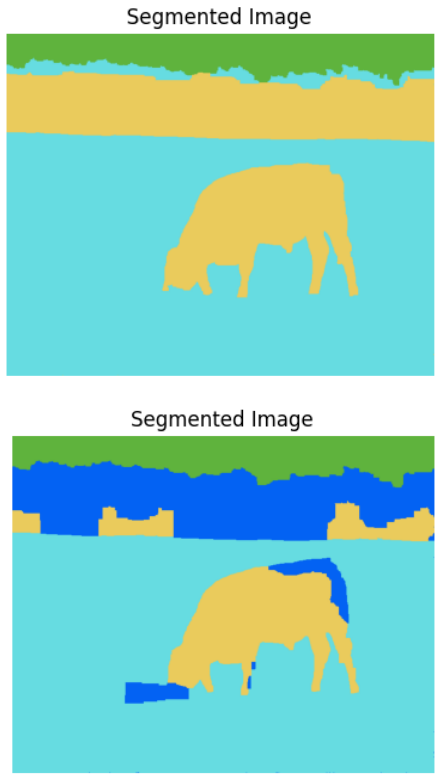
We present the results obtained for the cases with 3 and 4 labels. The dominant colors are determined using k-means. While the outcomes are generally satisfactory, there remains potential for further improvement.



Input image



Result for L2 unary cost



Result for CIELAB unary cost

Recycling Graph

Before $\alpha -$ expansion, we could identify partial solution and runs only the label on the "difficult" pixel.

Sub-Problem

We create an auxiliary α-binary problem for label $\alpha \in L$

- Unary cost:

$$D_p(f_p) = \begin{cases} D_p(\alpha) & \text{if } f_p = 1 \\ min_{p \in L \setminus \{\alpha\}} D_p(f_p) & \text{otherwise} \end{cases}$$

- Binary cost use the same potts value as the original energy problem
- If a pixel is assigned the value 1 after graphcut, then its label will be updated to α

Recycling Algorithm

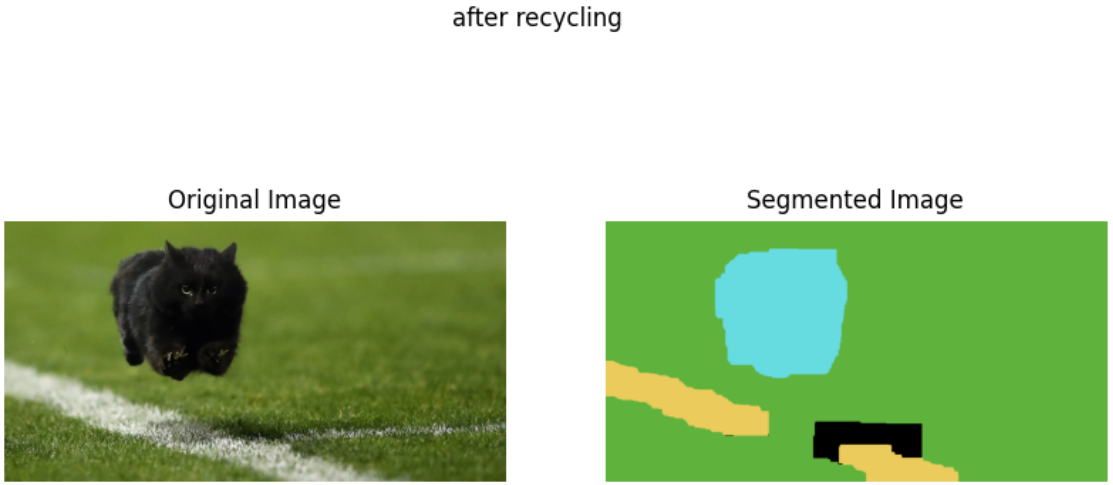
Recycling Algorithm

1. Initialize all pixels with the unlabeled state ϵ.
2. For each $\alpha \in L$:
   - Solve the corresponding binary subproblem for label α.
   - If $x_i^\alpha = 1$, assign label α to pixel i.
3. Update the unary costs for the remaining unlabeled nodes.
4. Apply the α-expansion algorithm to the unlabeled nodes.

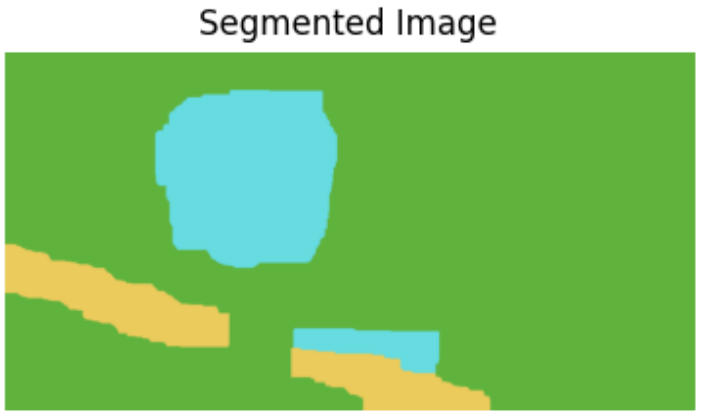
Idea: The computational bottleneck in the α-expansion algorithm lies in the computation of graph cuts. This recycling approach reduce both time and memory complexity by pre-processing part of the labeling.

Recycling Results

The algorithm is able to efficiently assign labels to a large portion of the image.
However, the resulting label configurations may be non-intuitive to interpret, as the pairwise terms are defined based on label differences rather than pixel color similarity.



Input to the algorithm and intermediate result after the recycling step.

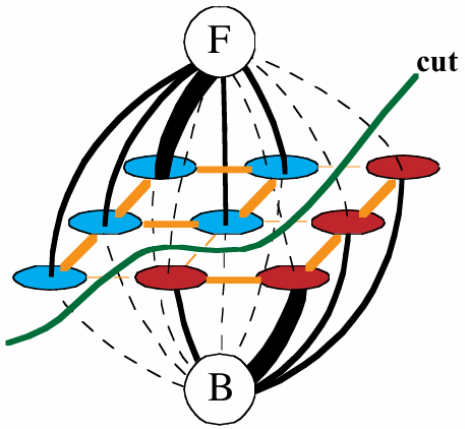


Final output after full label optimization.

Interactive binary segmentation

The seeds (pixel annotated/brushed by the user as background ($\mathcal{B}$) or foreground ($\mathcal{F}$)) are linked to the terminal node, the sink for background and source for foreground, with large capacity

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	$p \in \mathcal{B}$



Binary cut graph

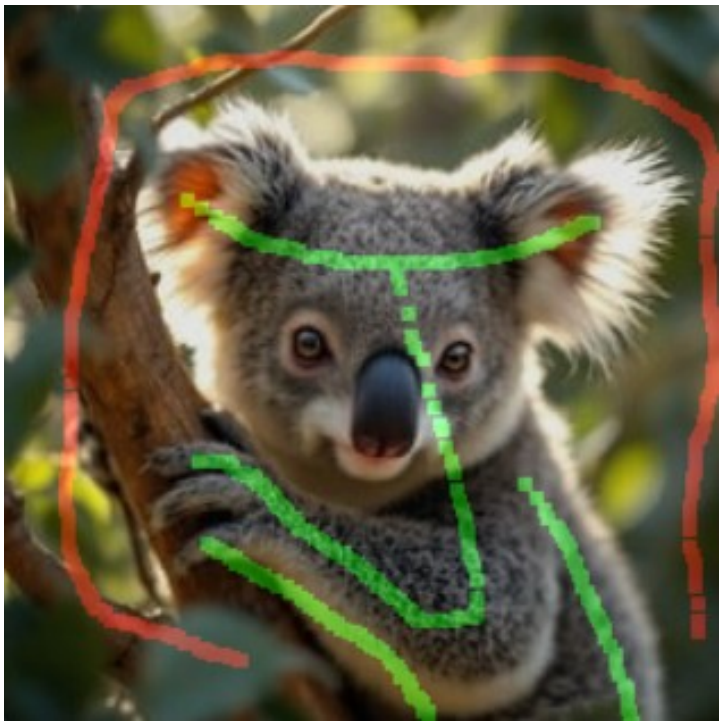
Labeling

- If $p \in S$ then p is labeled as foreground
- If $p \in T$ then p is labeled as background

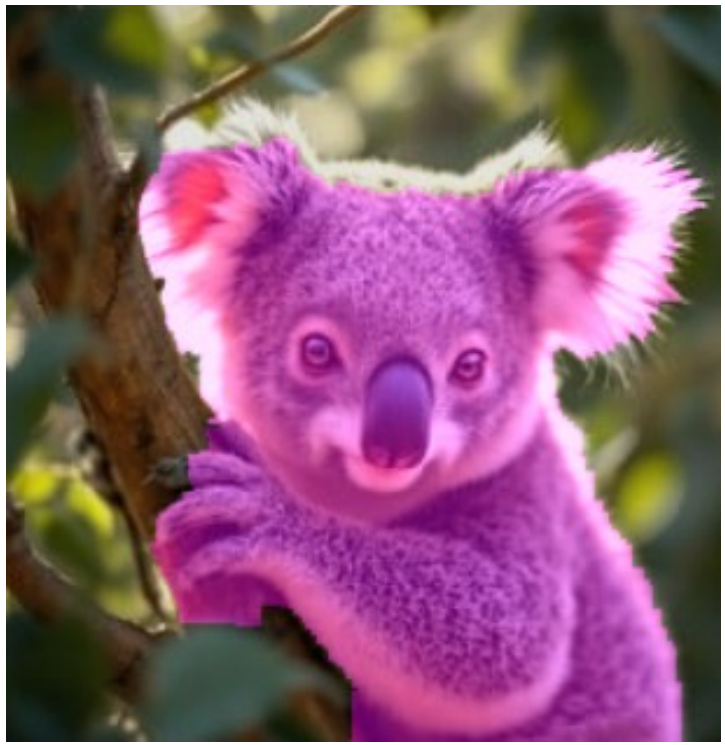
Interactive binary segmentation

Simple implementation:

- $V_{\{p,q\}} = \frac{1}{1+\|I(p)-I(q)\|}$
- $K = 10^9$ (large capacity to ensure seeds remains as user indicated)



Interactive user brush for initialization



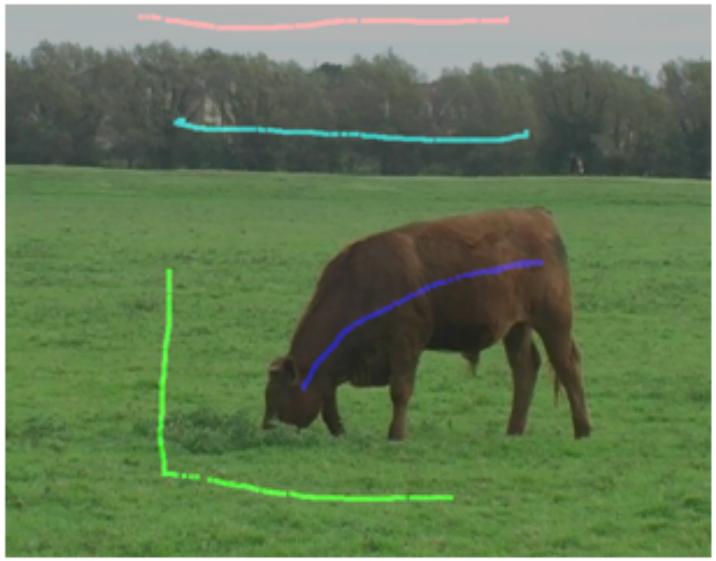
Segmented Image (foreground/background)

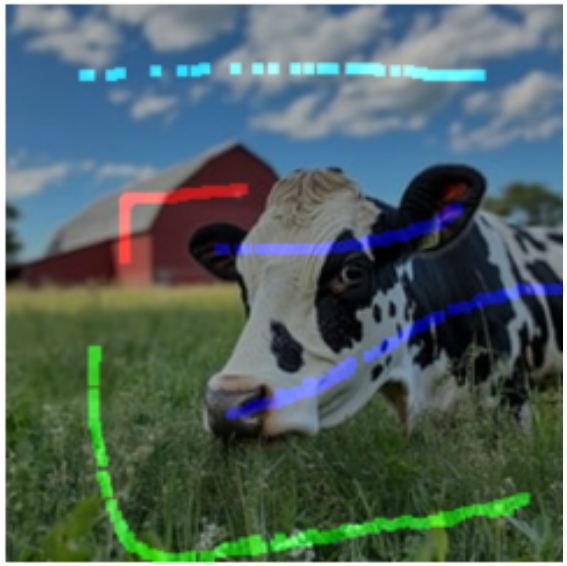
Influence of seeds

The algorithm is quite stable and the results do not seem to depend of the seeds positioning (except for complex/thin contour)

Binary segmentation for different seeds position

Interactive Multi-label segmentation





Conclusion

- We implemented the α-expansion algorithm and the recycle step from the article to perform multi-label segmentation
- We explored and implemted an extension with pixel-color based pairwise terms and interactive initialization.

Futur work:

- Implement other pairwise costs based on both with neighbor pixel values and labels
- Try to make to speed up the computation to also handle larger-sized images