M2 - Optimization and Inference Techniques for CV

Final Presentation

Graph Cuts

•••

Group 8

Alex Carrillo
Guillem Martínez

MSc in Computer Vision

Optimization

Criteria(s) examples
"Route length" → solution → Shortest route
"Travel time" → solution → Fastest route

Concept

In **optimization**, we have to find a solution for a given problem which is "best" in the sense of a **criteria**.

Goal and approach

1. Define a suitable criteria
2. Select best solution based on an optimization algorithm

   (different criteria might lead to different "optimal" solutions)

Ingredients

- A sense of the set of all possible solutions for our optimization problem
- Something that enables us to calculate a **quantity** which indicates the "goodness" of a particular solution
- (optional) Additional constraints which the solution has to satisfy

Optimization

Expressed mathematically:

$$f : S \to \mathbb{R}$$

$$x^* = \arg\min_{x \in S} f(x)$$

Mathematically, the ingredients are:

- a solution set ***S***
- an (objective) function ***f***

Find the solution ***x**** that minimizes (max.) the objective function ***f***.

Moreover,

- An optimization algorithm has to find a solution in reasonable time and accuracy
- Ideally, the problem task has to be a ***well-posed*** problem:
  1. A solution exists
  2. The solution is *unique*
  3. Small variations of initial conditions result in only small variations of the solution
  4. Sometimes, it can be solved analytically

  Else, we deal with complex ***ill-posed*** problems (typical in computer vision)

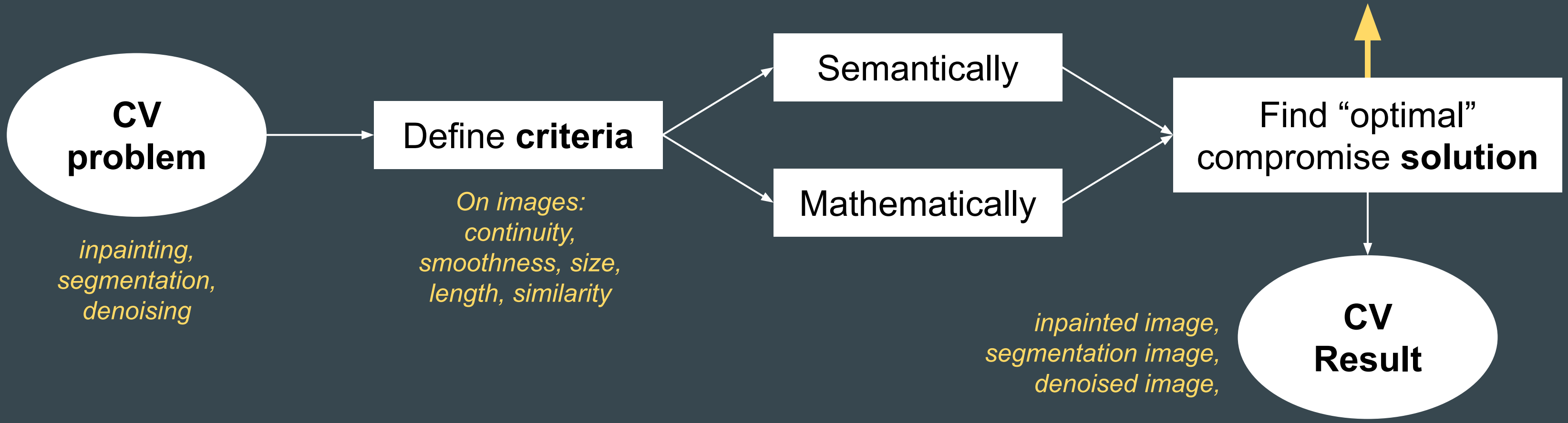
Optimization in Computer Vision

Many optimization problems arise from computer vision:

- Image restoration (denoising, inpainting), segmentation, compression, object detection, 3D (or multi-view) scene reconstruction, optical flow…

- Often, find acceptable solutions to an ***ill-posed*** problem
- *Spatial* and *visual* relationships play an important role in optimization

Choice of optimization method implementation depends on the problem!



Chosen Paper: *Interactive Graph Cuts* (Segmentation)

Problem definition

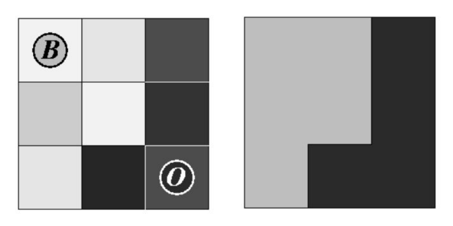
- **Segmentation problem**
  - Finding different regions or segments (set of pixels) that partitionate an image into **meaningful parts**
  - **Grouping objects** by some criteria, such that those within a group respond similarly and those in a different group respond differently

Goal

- Simplify the representation of an image
- Divide an image into two segments: "**object**" and "**background**"

Result

- An **image** that maps the different regions or segments (set of pixels)



original image *segmentation result*

Approaches

- Fully automatic segmentation
  - (which seems to) Never be perfect…
- **Interactive segmentation**
  - (is evaluated) More reliable…

Interactive segmentation (object and background)

Y. Y. Boykov and M. . -P. Jolly, "**Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images**", *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, pp. 105-112 vol.1, doi: 10.1109/ICCV.2001.937505.

Chosen Paper: Interactive Segmentation Criteria

1. User imposes certain **hard constraints** for segmentation

    1.1. Indicate certain pixels (*seeds*) that absolutely have to be part of the **object**

    1.2. Indicate certain pixels (*seeds*) that absolutely have to be part of the **background**

2. The rest of the image is **segmented automatically**

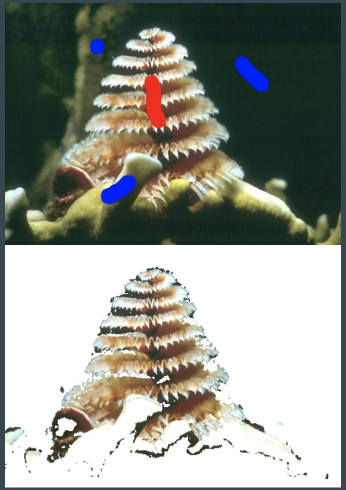
    (globally optimizing among all segmentations satisfying the hard constraints)

3. Get segmentation results **quickly** via very intuitive interactions

    (when user adds/removes any hard constraints *seeds*)

This (mathematically thinking) translates into:

- Defining a **cost** (function) in terms of boundary and regions properties of the segments
- The regions properties can be viewed as **soft constraints** for segmentation
- **Efficiently** (re)compute a global optimal segmentation



Interactive segmentation (object and background)

Y. Y. Boykov and M. . -P. Jolly, "**Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images**", *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, pp. 105-112 vol.1, doi: 10.1109/ICCV.2001.937505.

Chosen Paper: Graph Cuts Segmentation Mathematically

1. Represent image as a Markov Random Field represented by a **graph**:

$$\mathcal{V} = \mathcal{P} \cup \{S, T\} \qquad \mathcal{E} = \mathcal{N} \bigcup_{p \in \mathcal{P}} \{\{p, S\}, \{p, T\}\}$$

   P = **pixel node**
   S = terminal (source) node / **object**
   T = terminal (sink) node / **background**
   N = n-links (neighborhood links)
   {p, S or T} = t-links (terminal links)

2. Assign a non-negative weight w_e to each edge $\mathcal{E}$

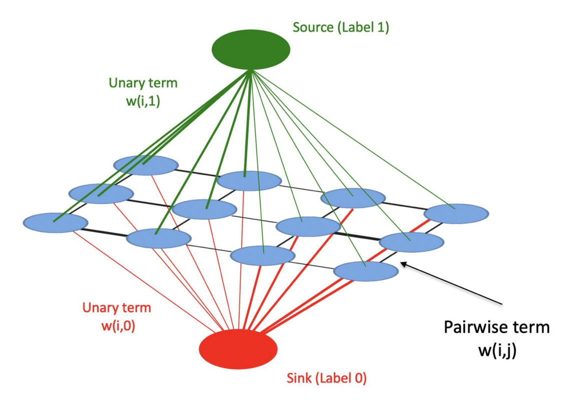
3. Define the **segmentation** as a "graph cut" ***C***

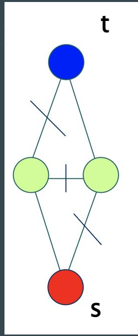
   (a set of edges that separate the terminals on the graph)

$$|C| = \sum_{e \in C} w_e$$

4. Conditions for a **feasible cut**

   4.1. ***C*** cuts exactly one t-link at each ***p***
   4.2. ***{p, q}*** in ***C*** iff ***p***, ***q*** are linked to different terminals
   4.3. If ***p*** in **object**, then ***{p, T}*** in ***C***
   4.4. If ***p*** in **background**, then ***{p, S}*** in ***C***





Y. Y. Boykov and M. . -P. Jolly, "**Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images**", *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, pp. 105-112 vol.1, doi: 10.1109/ICCV.2001.937505.

Chosen Paper: Energy function Mathematically

A = Segmentation binary vector

$$E(A) = \lambda \cdot R(A) + B(A)$$

Where **λ** is a weighting coefficient

Where **R(A)** is the regional term

$$R(A) = \sum_{p \in P} R_p(A_p)$$

Where **R_p** represents the penalty to assign to a label **A_p**

e.g. could be represented as:

$$R_p(Obj) = -\ln \Pr(Ip \mid O)$$
$$R_p(Bkg) = -\ln \Pr(Ip \mid B)$$

Where **Pr** is a histogram intensity distribution

Where **B(A)** is the boundary properties term

$$B(A) = \sum_{\{p,\, q\} \in N} B_{\{p,\, q\}} \cdot \delta(A_p,\, A_q)$$

Where **$B_{\{p,q\}}$** >= 0:

Represents the similarity between pixels p and q.

The closer to 0, the less it is similar.

e.g. local intensity gradient, zero-crossing …

And where **δ** function is:

$$\delta(A_p,\, A_q) = \begin{cases} 1 & A_p \neq A_q \\ 0 & \text{otherwise.} \end{cases}$$

Chosen Paper: Graph Initialization Mathematically

How is the graph initialized?

Graph Initialization

	edge	weight (cost)	for	
neighbor links	{p, q}	$B_{\{p,\, q\}}$	{p, q} ∈ N	
terminal links	{p, S}	λR_p(“bkg”)	p ∈ P, p ∉ O ∪ B	
		K	p ∈ O	selected by user
		0	p ∈ B	
terminal links	{p, T }	λR_p(“obj”)	p ∈ P, p ∉ O ∪ B	
		0	p ∈ O	selected by user
		K	p ∈ B	

And K is:

$$K = 1 + \max_{p \in P} \sum_{q:\, \{p,\, q\} \in N} B_{\{p,\, q\}}$$

What happens if we assign more labels?

Newly seeded pixels as object

t-link	Initial cost	add	New cost
{p, S}	λR_p(“bkg”)	K + λRp(“obj”)	K + C_p
{p, T }	λR_p(“obj”)	λRp(“bkg”)	C_p

Chosen Paper: Graph Initialization Mathematically

Once the graph is (re)initialized, to obtain the segmentation we can apply a low-order polynomial graph cut algorithm such as:

- Ford-Fulkerson style “augmenting paths”
    - Dinic algorithm → $O(mn^2)$
    - Boykov-Kolmogorov version of Dinic’s → $O(mn^2 |C|)$
- Goldberg-Tarjan style “push-relabel”

Side-note: In the paper, authors use Boykov-Kolmogorov algorithm because in practice is faster than other alternatives.

P1: Inpainting

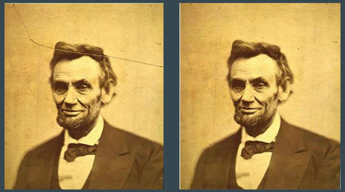
Inpainting:

- Technique in which damaged, deteriorated or missing regions of images are reconstructed by interpolation of surrounding areas

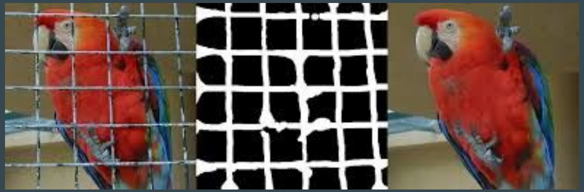
Conclusions and Results

- Notable results for inpainting small regions
- Bigger region, bigger blur in inpainted region, (may occur because):
  - The content available for inpainting is partial
  - The # of neighbours pixels to look at is not enough (4-connectivity)
- The smooth transition of the Laplacian operator is not enough to “hide” a visible region in an optimal way

before after



before after



original mask result

original mask result

P2: Poisson editing

Poisson editing:

- Seamless insert an image into a destination, by shifting part of its environment (color) and keeping intact (or not) the original details (without “any” visually unappealing seams)

Conclusions and Results

- Importing Gradients
  - Keeps details of the inserted image
  - Blurred background and visible borders
- Mixing Gradients
  - Keeps most relevant (striking) details of both images
  - Useful for objects with holes or transparent
- Choose the method based on the application

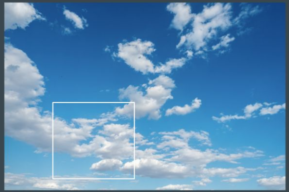
Source image B

Destination image A

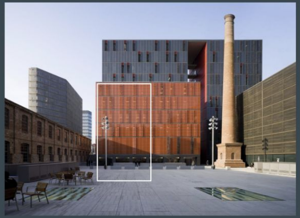
Importing Gradients

Mixing Gradients

Source | Target | Result



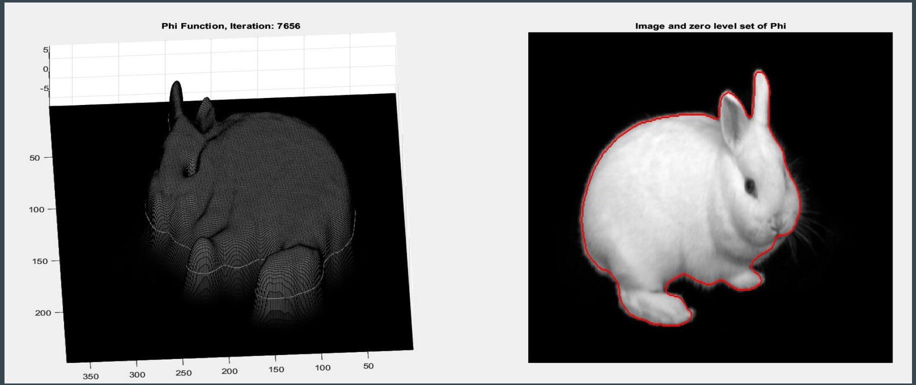
Source | Target | Result



P3: Segmentation:

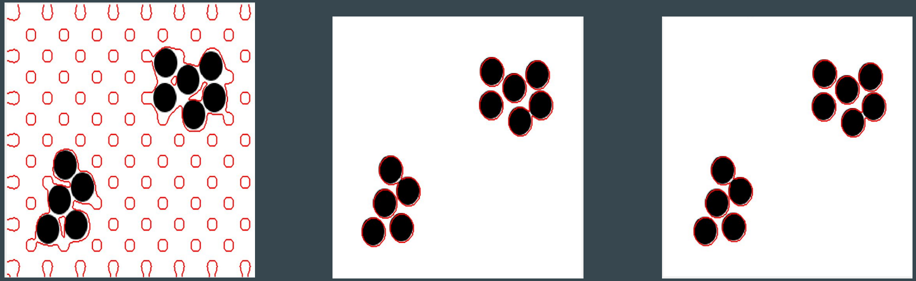
Segmentation:

- Segmentation is the process of finding different regions or segments (set of pixels) that partitionate an image into **meaningful parts**.



Conclusions and Results

- The Chan-Vese algorithm is a suitable algorithm for a basic binary segmentation problem.
- Varying hyperparameters such as length penalty, regularization term will result in smoother segmentation or sharper boundaries



Discussion

Optimization for computer vision:

- Optimization is the backbone of most algorithms that solve computer vision tasks, therefore, is a fundamental field
- By itself, it can already output acceptable solutions to some computer vision tasks
- No need (or few) for data needed to operate, in contrast of ML or DL solutions
- Easy to explain or understand, in contrast to not-so-easy explainable ML or DL solutions
- As the task difficulty increases, generally, more criteria(s) needs to be defined, which can be a challenging task

Conclusions

Through this Module:

- Learnt the optimization foundations in the context of computer vision
- Developed and/or explained five different optimization algorithms for three different computer vision tasks
- Grasped an understanding of the trade-off between explainability and performance