MCV-M2:
Optimisation and Inference for Computer Vision

—

Team 7 - Joan F. Serracant, Martí Cobos

Introduction

In this project we will implement optimization algorithms used in computer vision. The goal is to solve the image inpainting and completion problem.

This project is developed for the course *M2: Optimization and Inference Techniques for Computer Vision* under the *Master in Computer Vision* by *Universitat Autònoma de Barcelona (UAB*).

These slides contain the experiments done as well as their conclusions.

Index

What is optimization?

Optimization is a mathematical formulation to solve a problem by selecting the best solution according to a specified criteria.

The following procedure can be applied to solve a problem by optimization:

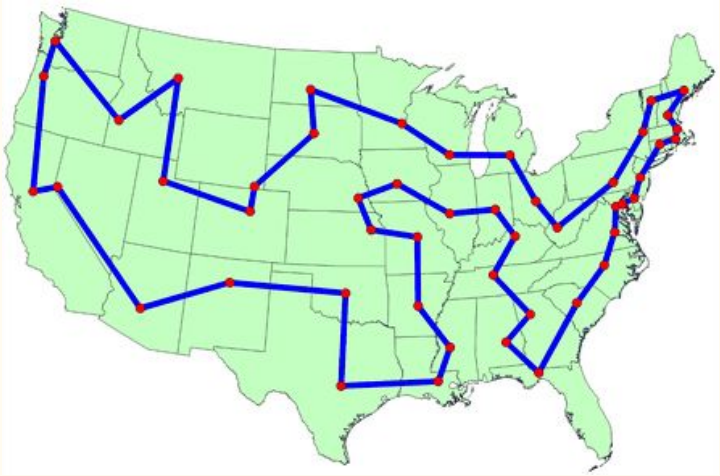
1. Define a criteria to solve a real world problem
2. Translate criteria into an objective (cost) function to minimize or maximize
3. Solve the problem by optimizing the cost function

Example: Travelling salesman problem

A salesman should visit a set of cities with no predefined order. A cost (based on distance, time, or expenses) is defined for travelling from one city to another.

Through **optimization**, an optimal tour can be determined. The solution tour allows the salesman to visit all the cities once, with same start and finish city at a minimum cost.

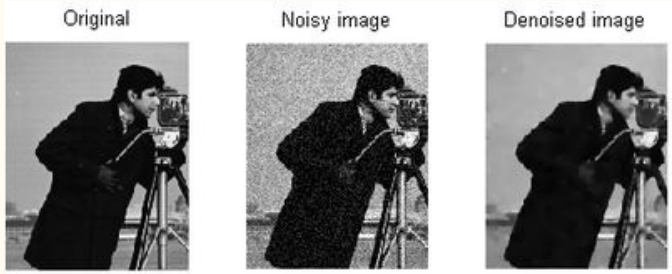
In addition this problem does not have an explicit solution.



Optimization in Computer Vision

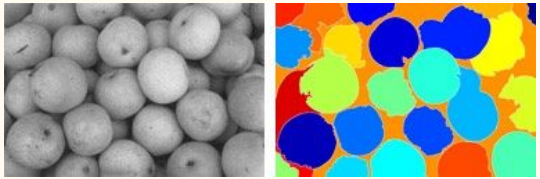
By treating an image as a function, some computer vision problems can be solved using optimization techniques, as an example the following problems could be optimized by defining an objective function to minimize:

- Image denoising:



- Image restoration (inpainting):

- Image segmentation:



Week 1: Image Inpainting

Computer Vision Problem: Reconstruct lost information on images by generating a new image with smooth transition on the area to inpaint

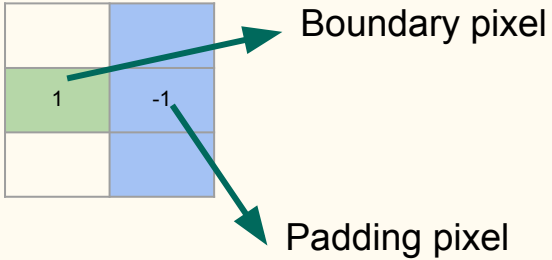
Objective function: integration of gradients on the domain (area to inpaint):

$$\begin{cases} \arg\min_{u \epsilon W^{1,2}(\Omega)} \int_D |\nabla u(x)|^2 \, dx \\ u|_{\delta D} = f \end{cases}$$

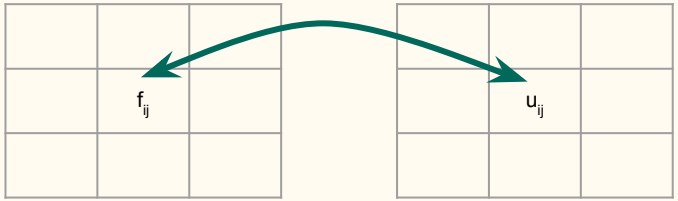
Week 1: Image Inpainting

Problem solution: Build the matrix A and vector b for the algebraic system of equations obtained after applying Gateaux derivative, and minimum criteria for cost function (derivative equal to Zero).

Boundary pixels: $u_{i,j} - u_{i,j+1} = 0$ for right boundary



Known pixels:

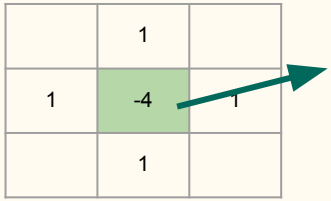


Equal values

Original image

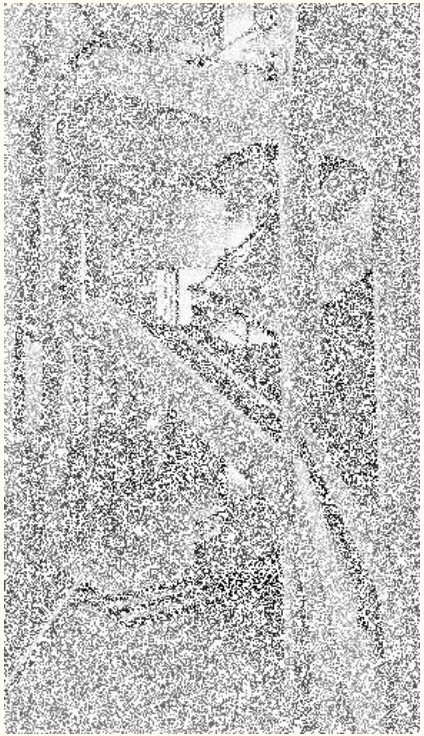
Unknown image

Pixels to Inpaint: $u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j} = 0$



Week 1: Image Inpainting

Solution Example:



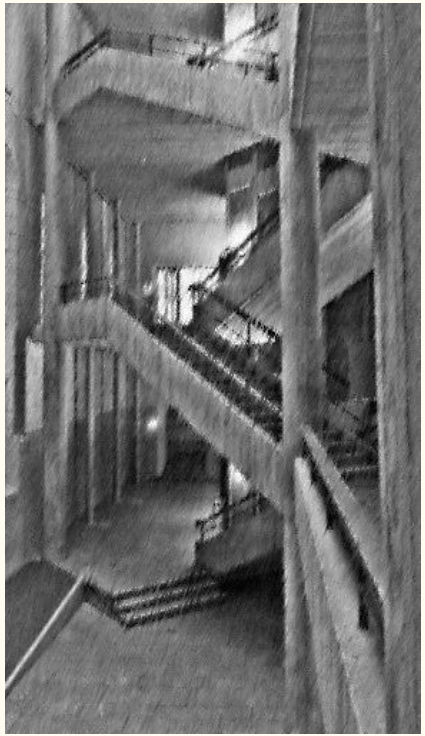


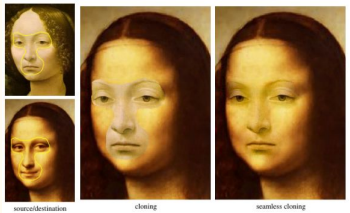
Image inpainted successfully with a solution image with high level of detail with respect to the original scene

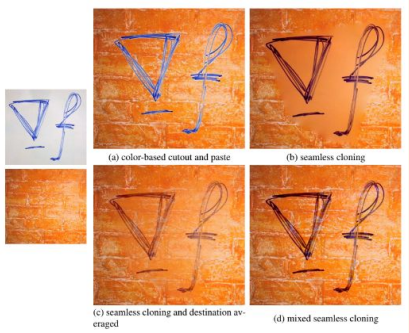
Week 2: Poisson Editing

Source: Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. ACM Trans. Graph. 22, 3 (July 2003), 313-318

Computer Vision Problem: Seamless importation of source image regions into a destination region.
Criteria to define the problem:

1. Transition from src to dst images must be smooth
2. Details (gradient) of dst image must be kept





Objective function: Two techniques are described to perform Seamless cloning:

Importing gradients: the guidance field **v** is taken from the src image **g**

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

$$\mathbf{v} = \nabla g,$$

$$\Delta f = \Delta g \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}.$$

Mixing gradients: texture from src image **g** is transferred but some texture of the dst image **f** is also kept

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

$$\text{for all } \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})|, \\ \nabla g(\mathbf{x}) & \text{otherwise.} \end{cases}$$

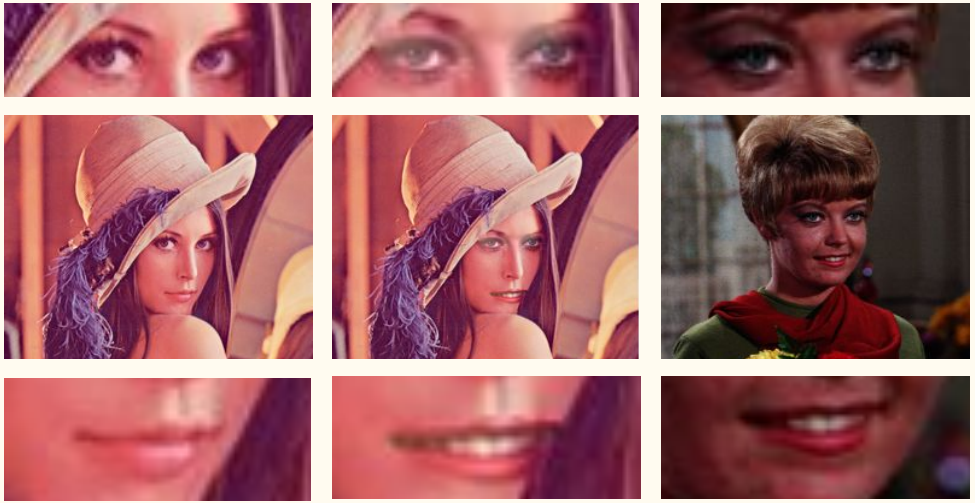
Week 2: Poisson Editing

Problem solution: similar to image inpainting problem, an algebraic system of equations is obtained after applying Gateaux derivative, and minimum criteria for cost function (derivative equal to Zero).

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

$$\Delta f = f_{i+1,j} - f_{i-1,j} - f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = div\,(v) = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} = \frac{\partial \nabla g}{\partial x} + \frac{\partial \nabla g}{\partial y}$$

Gradient (shape and texture) is kept from right image but colors are adapted to the ones of the left image



Importing gradients algorithm

Week 2: Poisson Editing

Solution Example:

Mixing gradients result:

Importing gradients result:

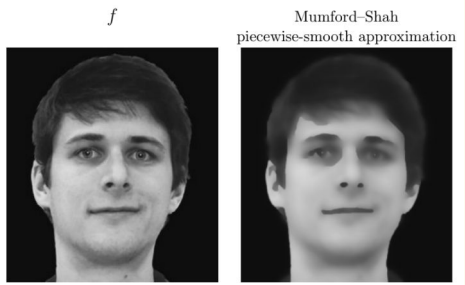
- Building tiles from destination image have been preserved.
- Result image seems realistic with better results than importing gradients algorithm.

Week 3: Image Segmentation

Source: Pascal Getreuer, Chan-Vese Segmentation, Image Processing On Line, 2 (2012), pp. 214–224.

Computer Vision Problem: Segment an image according to colour on the region. Criteria to segment an image:

1. Minimum length of the region border to avoid jagged edges
2. Similarity criteria between original image and segmented image



Objective function: minimize curve length, area inside curve and difference between original image and segmented image using following cost function:

$$\underset{c_1,c_2,C}{\arg\min}\ \mu\,\mathrm{Length}(C) + \nu\,\mathrm{Area}(\mathit{inside}(C)) + \lambda_1 \int_{\mathit{inside}(C)} |f(x) - c_1|^2\,dx + \lambda_2 \int_{\mathit{outside}(C)} |f(x) - c_2|^2\,dx.$$

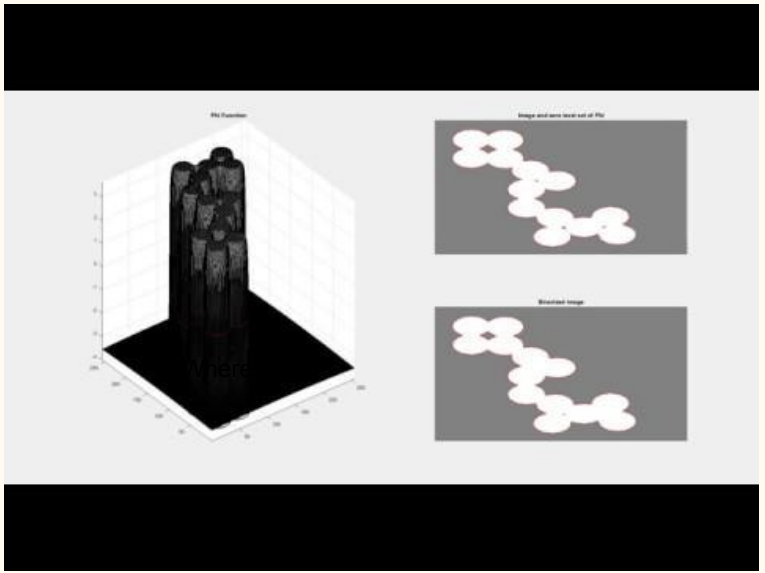
Week 3: Image Segmentation

Problem solution: To simplify the problem Level Set Functions are used. Contour C can be expressed in terms of a level set function φ so $C = \{x \in \Omega : \varphi(x) = 0\}$. Now function over x is computed instead of a set of curves.
New function to minimize using level sets:

$$\arg\min_{c_1,c_2,\varphi} \mu \int_\Omega \delta(\varphi(x))|\nabla\varphi(x)|\,dx + \nu \int_\Omega H(\varphi(x))\,dx + \lambda_1 \int_\Omega |f(x)-c_1|^2 H(\varphi(x))\,dx + \lambda_2 \int_\Omega |f(x)-c_2|^2 (1-H(\varphi(x)))\,dx$$

Cost function minimization implemented using gradient descent:

$$\varphi_{i,j}^{n+1} \leftarrow \left[\varphi_{i,j}^n + dt\,\delta_\epsilon(\varphi_{i,j}^n)\left(A_{i,j}\varphi_{i+1,j}^n + A_{i-1,j}\varphi_{i-1,j}^{n+1} + B_{i,j}\varphi_{i,j+1}^n + B_{i,j-1}\varphi_{i,j-1}^{n+1} - \nu - \lambda_1(f_{i,j}-c_1)^2 + \lambda_2(f_{i,j}-c_2)^2\right)\right] / \left[1 + dt\,\delta_\epsilon(\varphi_{i,j})(A_{i,j} + A_{i-1,j} + B_{i,j} + B_{i,j-1})\right].$$



Week 3: Image Segmentation

Solution Example:

μ = 0.1

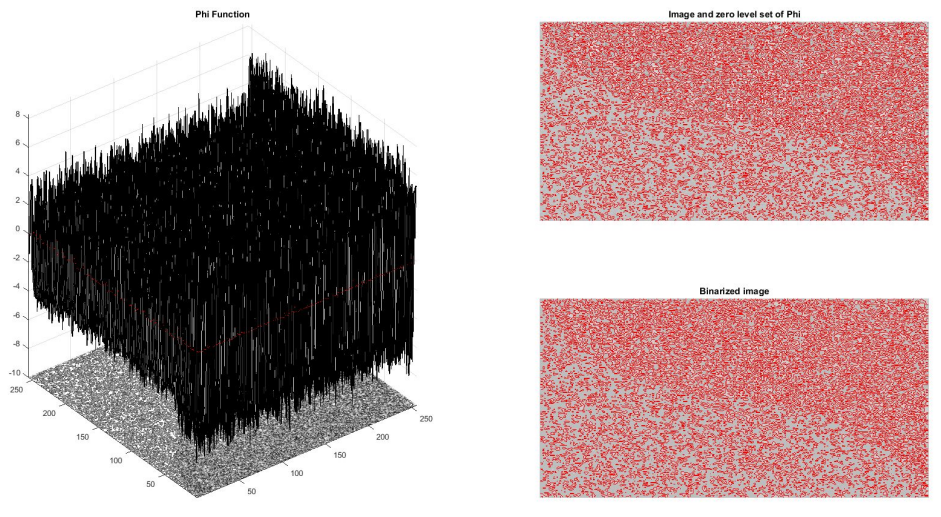
μ = 0.01

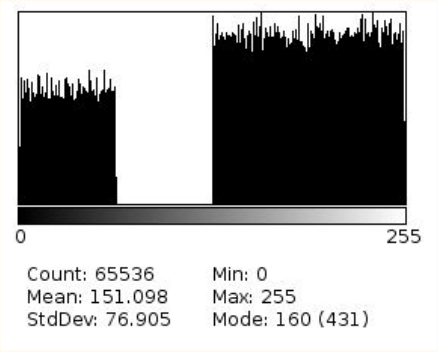
μ = 0.001

μ as a factor to control length of the contour

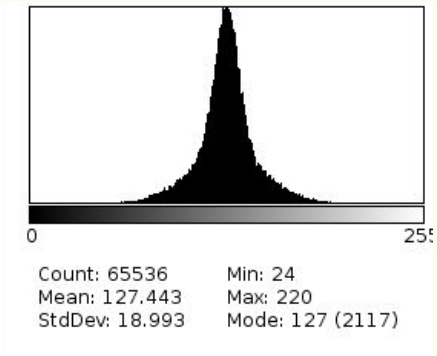
Week 3: Image Segmentation

- A weakness of Chan-Vese segmentation is that it depends on the mean values c_1, c_2 of the two segmented areas. Thus it can only work on the assumption that the mean value of areas to be segmented must be different.
- In our test images two images looked similar, phantom18 and 19, but the latter could not be segmented. The reason is that, although the amount of noise is similar, in “Phantom19” the mean grayscale value for both areas is the same (see the histograms below)





Phantom18 histogram



Phantom19 histogram (both areas with the same mean)

Week 4: Optimization vs. Machine Learning

- But all the cool kids are doing ML … why optimization?

Optimization	Machine Learning
Analytical approach: problem is framed in a mathematically well defined form.	Numerical approach: makes guesses at the solution and tests if it is good enough to stop.
Objective functions with several terms or complex graphical models require big computing resources.	Needs huge datasets for training, big models and big computing resources (Deep Learning)
Training is not needed: if the solution exists we can go straight to it no costly images. There is no need to obtain a image database which could be costly in some cases.	Very good at extracting features, trends and patterns
As the problem is analytically well defined, specific features are evaluated in order to make a decision on the processed image. This gives more confidence on the result which could be relevant for medical or security applications.	By training a machine learning algorithm(or deep learning) it is difficult to understand what the algorithm is learning. This could reduce the confidence on the result as under certain circumstances the feature which is being analyzed is very important

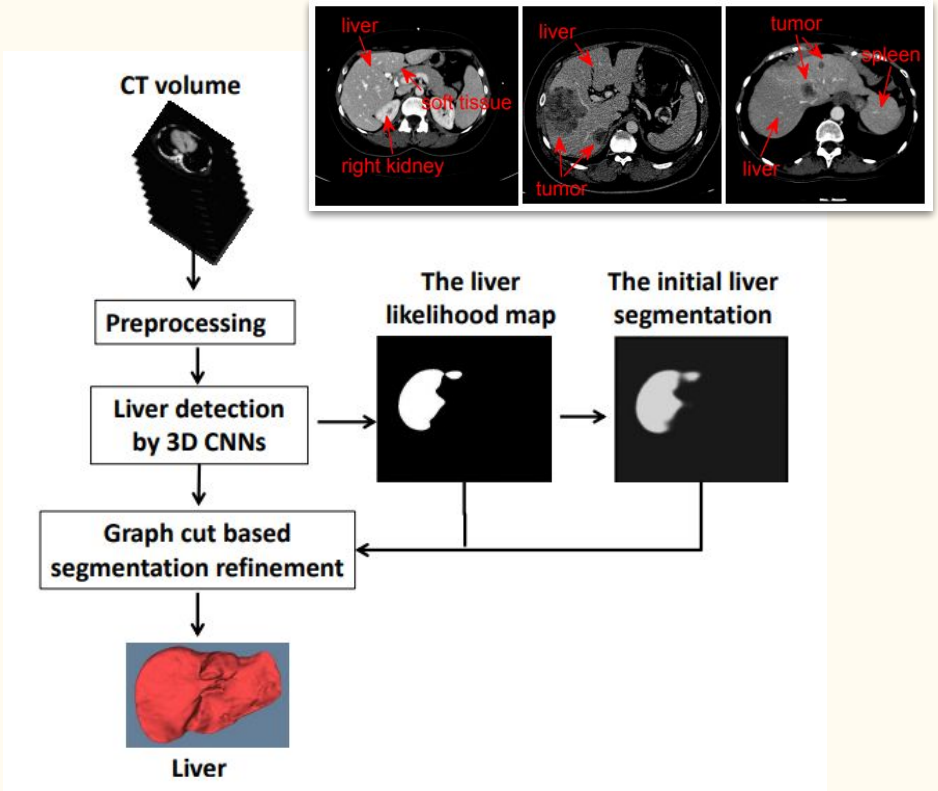
If we had to choose, we'd choose BOTH and take profit from the best of both worlds.

Week 4: Graphical models

Automatic 3D liver location and segmentation via convolutional neural networks and graph cut

Fang Lu · Fa Wu · Peijun Hu · Zhiyi Peng · Dexing Kong*

- The paper describes a method to segment the liver in a CT volume by using a 3D convolutional neural network and graph cut.
- 3D CNN is used to get a first roughly estimated segmentation of the liver
- The output of the 3D CNN is used as the input for a graph cut that refines the segmentation.
- 3D CNN is good at learning features to detect the liver, while graph cut is good at finding the exact border with other organs.



Week 4: Graphical models

The energy function to be minimized by the graph cut is the following:

$$E(l) = \lambda E_D(l) + E_B(l)$$
$$= \lambda \sum_{x \in V} D_x(l_x) + \sum_{x \in V} \sum_{y \in N_x} B_{xy}(x,y)\delta(l_x, l_y),$$

where

$$\delta_{xy}(l_x, l_y) = \begin{cases} 1, & if \quad l_x \neq l_y, \\ 0, & otherwise. \end{cases}$$

Boundary term: encodes the discontinuity between neighboring voxels x and y

$$B_{xy}(x,y) = \frac{1}{1 + \beta |I(x) - I(y)|^2},$$

Data fitting term: degree of similarity between a voxel x and the foreground of background. [$l_x \in \{0,1\}$ where 0 is non-liver and 1 is liver]

$$D_x(l_x) = max(-\mathcal{R}(x), 0)l_x + max(\mathcal{R}(x), 0)(1 - l_x),$$

where

$$\mathcal{R}(x) = \sum_{y \in N_x} B_{xy}(x,y)[f(x) + L(x) - 0.5 + \gamma \mathcal{P}(x)]$$

$$f(x) = \frac{(I(x) - \zeta)(I(x) - \eta)}{(\eta - \zeta)^2}.$$

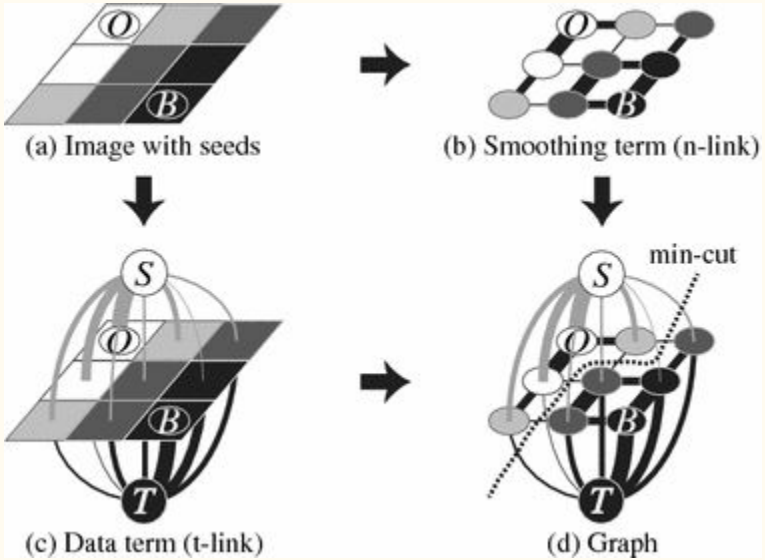
Thresholding map where [ζ, η] is the intensity range in the liver from 3D CNN

$L(x)$ Probability map

$\mathcal{P}(x)$ Local appearance map keeps segmentation similar to the one provided by 3D CNN

Week 4: Graphical models

- Graph cuts can be used in computer vision in order to solve problems formulated in terms of energy minimization.
- In the paper a graph is constructed so each voxel is connected both to
  - Its neighbors
  - S (source) and T (sink) the two terminal nodes for the segmentation



- Then weight of edges are defined: e_{sx} for edges from voxels to source, e_{xt} for edges from voxels to sink and e_{xy} for edges between neighboring voxels, with the following values

$$e_{sx} = \begin{cases} D_x(l_x = 0), & if \quad \mathcal{R}(x) > 0, \\ 0, & otherwise. \end{cases}$$

$$e_{xt} = \begin{cases} D_x(l_x = 1), & if \quad \mathcal{R}(x) < 0, \\ 0, & otherwise. \end{cases}$$

$$e_{xy} = B_{xy}(x, y),$$

- Finally the graph is cut through those edges with minimum weight

Week 4: Graphical models

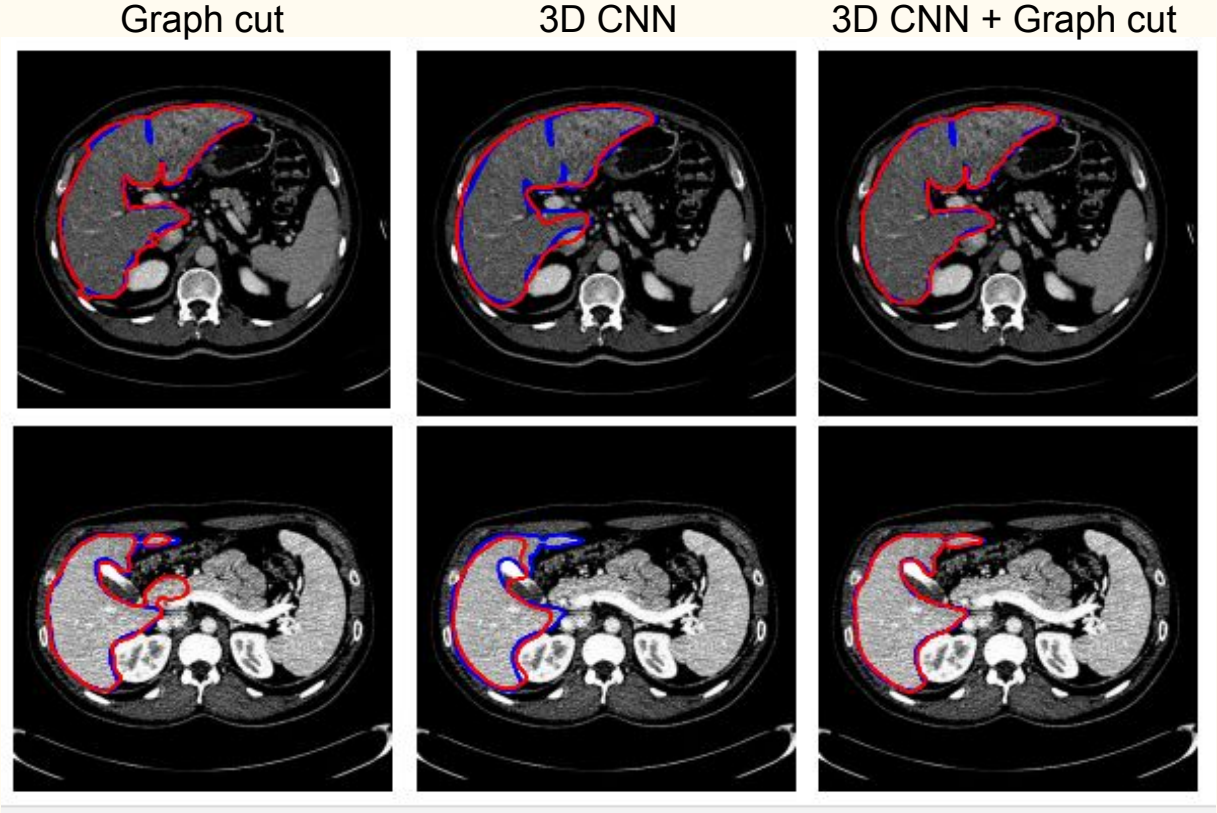
Our proposal of pseudo-code to implement paper's solution. Paper does not specify the min-cut/max-flow algorithm to perform graph cut, so we chose FordFulkerson method.

```
Input: I a CT volume, trained 3DCNN
Output: O a 3D image of segmented liver
Function LiverSegmentation(I, O)
      1: P ← preprocess(I)
      2: S_0 ← 3DCNN.predict(P)
      3: G, s, t ← construct_graph(I, P, S_0)
      4: max_flow ← FordFulkersonGraphCut(G, s, t)
      5: O ← segment(I, G, max_flow)
      6: return O
```

```
Input: G a graph, s a source node, t a sink node
Output: F the max-flow
Function FordFulkersonGraphCut(G, s, t)
      1: initialize flow to 0
      2: path = findAugmentingPath(G, s, t)
      3: while path exists:
      4:     augment flow along path
      5:     G_f = createResidualGraph()
      6:     path = findAugmentingPath(G_f, s, t)
      7: return flow
```

(from https://brilliant.org/wiki/ford-fulkerson-algorithm/)

Week 4: Graphical models



Results comparing segmentation (red) and ground truth (blue) using graph cut, 3D CNN and 3D CNN + graph cut.

Conclusions

- **Image inpainting** problem implemented through optimization with great results. Even images with low number of pixels available are restored and original image can be deduced.
- **Poisson editing** problem implemented through optimization with great results. Additional images were tested. For instance “Banksy at CVC” was edited successfully with **Mixing Gradients** algorithm as as it improves solution over the importing gradients algorithm as it preserves information from destination image. Result image appears more realistic than with **Importing Gradients** method.
- **Image segmentation** problem has been implemented through optimization techniques. **Chan Vese** algorithm (based on **level sets**) has been implemented and some limitations have been found for this procedure. For instance, this algorithm doesn’t work properly if image **mean** values of regions to be segmented are similar.
- **Graphical models** is a useful technique that can be applied to computer vision tasks. This technique tries to optimize the result of each pixel by taking into account the information of the neighbouring pixels. This procedure has been compared to **Machine Learning** showing advantages of using both techniques
- A paper has been analyzed describing a Computer Vision pipeline which combines the output of a **3D-CNN** with an optimization technique based on **Energy Minimization** and **Graph-Cut**. Experimental results of this combination outperforms based on either Deep Learning or Optimization techniques.