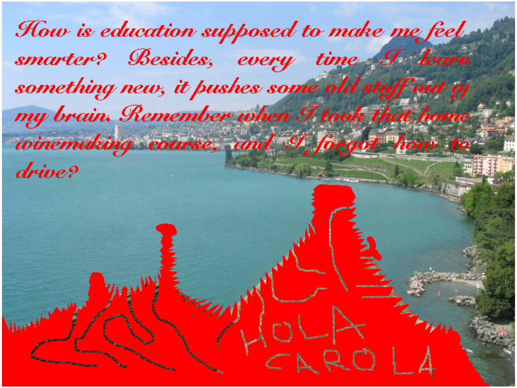
Image inpainting and completion

Onofre Martorell, Lidia Talavera

November 15, 2016

Goal of the project

Restore the given image using variational and graphical models.



Introduction

Steps

1. Image inpainting
2. Image segmentation
3. Poisson editing
4. Image completion

Image inpainting

Image inpainting

Replace lost or corrupted data on images or videos.

Example

The problem of inpainting can be modelled as

$$\begin{cases} \underset{u \in W^{1,2}(\Omega)}{\arg\min} \displaystyle\int_D |\nabla u(x)|^2 dx, \\ u|_{\partial D} = f \end{cases}$$

where f is the image to inpaint.
The associated Euler-Lagrange equation of this functional is

$$\begin{cases} \Delta u = 0 & \text{in } D \\ u = f & \text{in } \partial D \end{cases}$$

The equation is completed with homogeneous Neumann boundary conditions at the boundary of the image.

Discretization of Laplacian

The Laplacian is discreticised using finite differences:

$$\frac{1}{h_j^2}u_{i,j-1}+\frac{1}{h_i^2}u_{i-1,j}-\left(\frac{2}{h_i^2}+\frac{2}{h_j^2}\right)u_{i,j}+\frac{1}{h_i^2}u_{i+1,j}+\frac{1}{h_j^2}u_{i,j+1}=0$$

This is the equation that have to satisfy the pixels that we want to inpaint.

Boundary conditions

We have added one row or column to each side of the image and we have computed the Neumann boundary conditions. For example, at the east side, we get

$$\frac{u_{i,1} - u_{i,2}}{h_j} = 0 \Longrightarrow u_{i,1} = u_{i,2}$$

And for the pixels belonging to west, north or south boundary we have

$$u_{i,n_j} = u_{i,n_j-1}, \;\; u_{1,j} = u_{2,j} \text{ or } u_{n_i,j} = u_{n_i-1,j}.$$

Other pixels

$$u_{i,j} = f_{i,j}$$

Ordering the pixels of the image as

$$x = (u_{1,1}, u_{2,1}, \cdots, u_{i,j}, u_{i+1,j}, \cdots, u_{n_i+2,n_j+2})^T$$

the previous equations can be written as a linear system of equations

$$Ax = b,$$

which can be solved using Matlab.

Image inpainting

Results

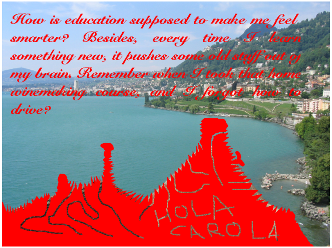


Image inpainting

Results

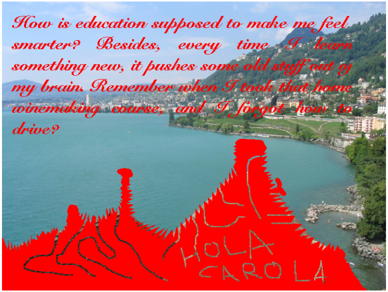
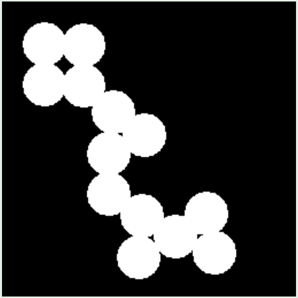


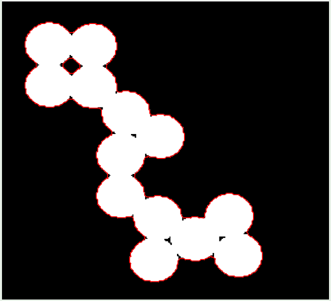
Image segmentation

Image segmentationn

Divide the image into different parts.

Example





We will use Chan-Vese segmentation which consist on finding a curve that is the boundary of the segmentation. The way to find that curve is minimizing the following functional:

$$\arg\min_{c_1,\ c_2,\ C}\ \mu \text{Lenght}(C) + \nu \text{Area}(\mathit{inside}(C)) +$$

$$\lambda_1 \int_{\mathit{inside}(C)} |f(x) - c_1|^2 dx + \lambda_2 \int_{\mathit{outside}(C)} |f(x) - c_2|^2 dx,$$

where C is the boundary of a closed set and $c1$, $c2$ are the values of u respectively inside and outside of C.

As it is difficult to manipulate C Chan-Vese segmentation uses a function ϕ and the C will be the zero crossing of φ that is

$$C = \{x \in \Omega : \varphi(x) = 0\}.$$

With this the functional is rewritten as:

$$\text{arg min}_{c_1,c_2,\varphi}\ \mu \int_\Omega \delta(\varphi(x))|\nabla\varphi(x)|dx + \nu \int_\Omega H(\varphi(x))dx +$$

$$\lambda_1 \int_\Omega |f(x) - c_1|^2 H(\varphi(x))dx + \lambda_2 \int_\Omega |f(x) - c_2|^2 (1 - H(\varphi(x)))dx,$$

In the previous formula H denotes the Heaviside function and δ the Dirac mass, its distributional derivative:

$$H = \begin{cases} 1 & t \geq 0, \\ 0 & t < 0 \end{cases}, \quad \delta(t) = \frac{d}{dt} H(t).$$

Note that we cannot derive $H(t)$. Because of that, in the implementation we take the Heaviside function as

$$H_\epsilon(t) = \frac{1}{2}\left(1 + \frac{2}{\pi} \arctan\left(\frac{t}{\epsilon}\right)\right).$$

Now we have to minimize the functional respect to c_1, c_2 and φ. The way to do it is the following: at each iteration we do this steps:

1. Update c_1 and c_2 as the average gray scale value where ϕ is positive or negative, respectively.
2. Evolve φ using the semi-implicit gradient descent

$$\varphi^n_{i,j} = [\varphi^n_{i,j} + dt \cdot \delta_\varepsilon(\varphi^n_{i,j})(A_{i,j}\varphi^n_{i+1,j}) + A_{i-1,j}\varphi^{n+1}_{i-1,j} + B_{i,j}\varphi^n_{i,j+1} +$$

$$B_{i,j-1}\varphi^{n+1}_{i,j-1} - \nu - \lambda_1(f_{i,j} - c_1)^2 + \lambda_2({}_{i,j} - c_2)^2)]/[1 + dt \cdot \delta_\varepsilon(\varphi^n_{i,j})(A_{i,j} +$$

$$A_{i-1,j} + B_{i,j} + B_{i,j-1})]$$

.

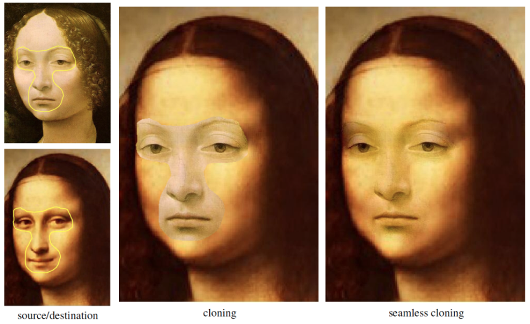
3. If we have reached the maximum number of iterations or the difference $\max(|\varphi^{n+1} - \varphi|)$ is lower than a given tolerance, we stop the algorithm.
4. Every certain number of iterations, reinitialize φ.

Poisson Image editing

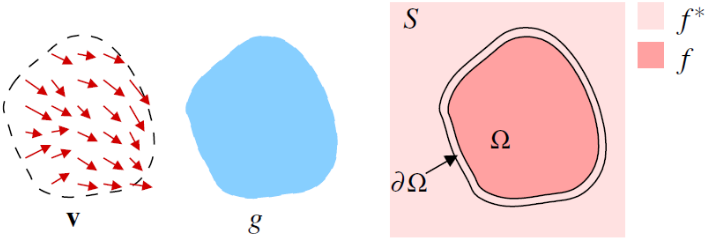
Poisson Image editing

Copy a region of an image to another one and adapt it in order that it does not seem fake.

Example



Seamless cloning will be applied through the implementation of the importing gradients method. Let f^* be the destination image, f the source image that contains the region we want to clone, Ω which will be cloned from f to f^* and $\overrightarrow{v}$ the guidance field of vectors.



To solve the problem we have used the next functional over each channel of the image:

$$\min_f \int_\Omega \left|\nabla f - \overrightarrow{v}\right|^2,$$

with $f_{|\delta\Omega} = f^*_{|\delta\Omega}$,

The associated Euler-Lagrange of this functional is

$$\begin{cases} \Delta f = div\,\overrightarrow{v} & \text{on} \quad \Omega \\ f = f^* & \text{in } \partial\Omega \end{cases}$$

We are using importing gradients, which means that the guidance field $\overrightarrow{v}$ is a gradient field taken directly from a source image. Denoting by g the source image, this is

$$\overrightarrow{v} = \nabla g.$$

Replacing this, we get

$$\begin{cases} \Delta f = \Delta g & \text{on} \quad \Omega \\ f = f^* & \text{in } \partial\Omega \end{cases}$$

What we have done is the following: we have taken the code of the previous week and we have changed it in order to detect if there is a variable containing the discretization of Δg for each point of the image. If this variable exists, it computes $\Delta f = \Delta g$; otherwise it computes $\Delta f = 0$.

Poisson Image editing

Results

Image completion

Image completion

Select a similar image to the given one and copy a selected region.

Image completion

Solution and implementation

1. From a huge database of image, find the most similar images using GIST descriptors.
2. From every found image, find the most similar region to the hole of our image.
3. Authomatically fine tune of the editing mask. This step consists on taking the segmentation obtained and tuning it using a graphical model.

The cost function used is

$$C(L) = \sum_p C_d(p, L(p)) + \sum_{p,q} C_i(p, q, L(p), L(q))$$

where

- $L(p)$ is the label of the pixel p: *exists* or *patch*.
- C_d are the unary potentials:
  - If p belongs to the mask, $C_d(p, exists) = \infty$ and $C_d(p, patch) = 0$
  - If p does not belong to the mask, $C_d(p, exists) = 0$ and $C_d(p, patch) = (k \cdot (\text{dist}(p, \text{mask}))^3$
- C_i are the binary potentials: if the pixels are adjacent and have different label, its value is the difference of the SSD at each pixel.

4. Copy the region to our image and perform Poisson Editing correction.
5. As we will have many results, choose by visual inspection the best one.

Image completion

Results

Conclusions

- We have solved our problem in different techniques that at the beginning didn't seem to be related among them.
- During the project, we have learned how to solve some variational and graphical models.

Any questions?