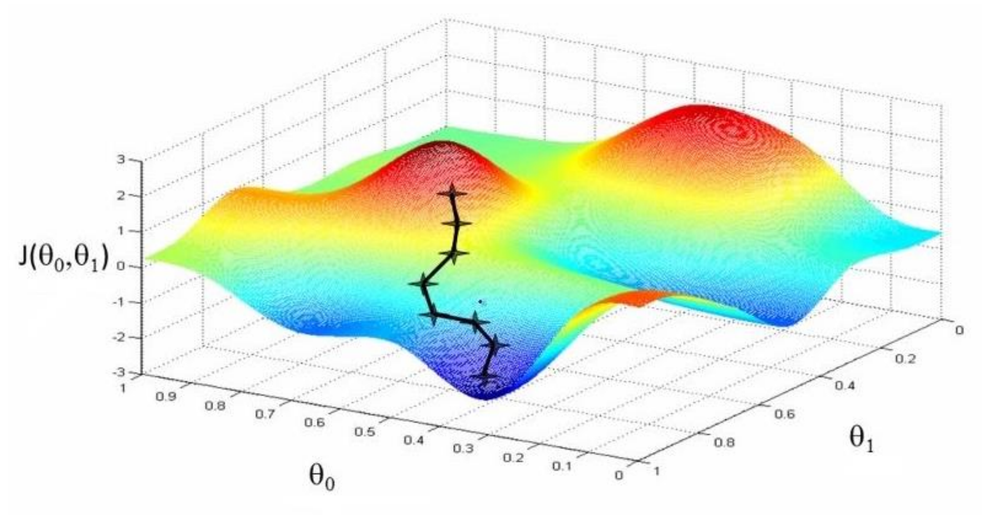
Master in Computer Vision Barcelona

M2 – Optimisation in Computer Vision
Part 2 – Poisson Editing

Karim Lekadir
karim.lekadir@ub.edu

Optimisation



- The act of making something as good as possible (dictionary)
- Making the best out of situation
- Finding maximum or minimum of a function (mathematics)
- Finding the best possible solution, given some criteria

Optimisation

- **Well-posed problem**:
  1. A solution exists
  2. The solution is unique
  3. The solution's behaviour changes continuously with the initial conditions

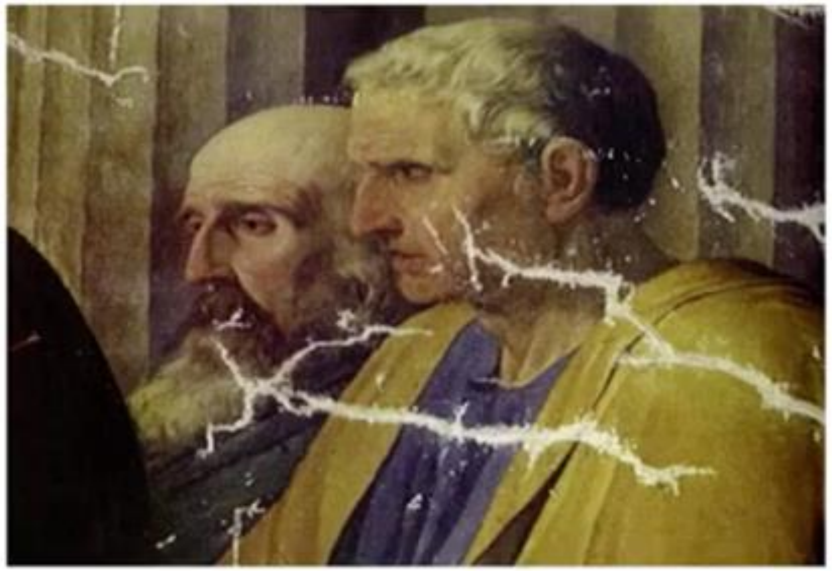
- In computer vision, we often deal with complex **ill-posed problems**.
- Analytical solutions are often not available.
- Optimisation can help find an acceptable solution to an ill-posed problem

Optimisation in Computer Vision

1. Define a set of **criteria** to solve the computer vision problem
2. Define each criterion **semantically**
3. Then, define each criterion **mathematically**
4. Find a solution that is an “optimal” compromise of the different criteria
5. For example, find a method to minimise or maximise an energy function that is the sum of multiple terms corresponding to the different criteria

Image Inpainting

- Inpainting is the process of producing a complete image from an image with damaged, deteriorating, or missing parts



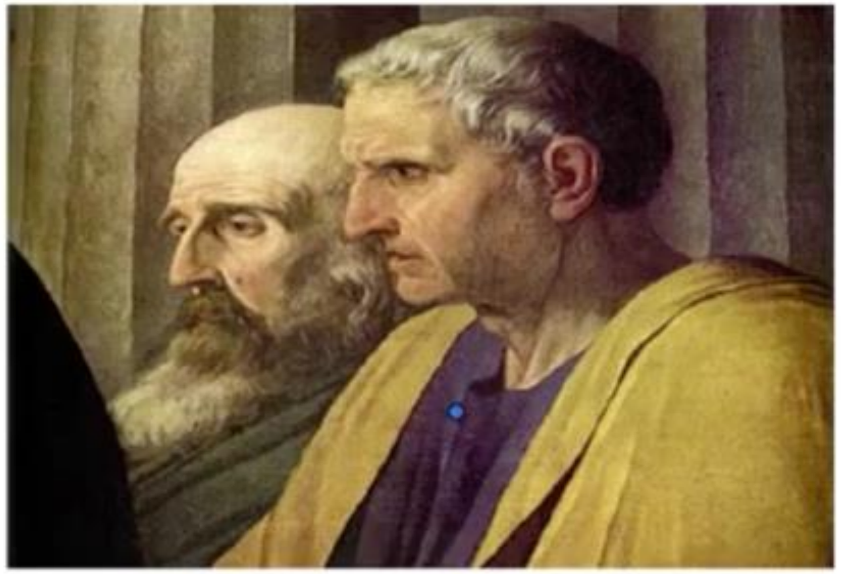
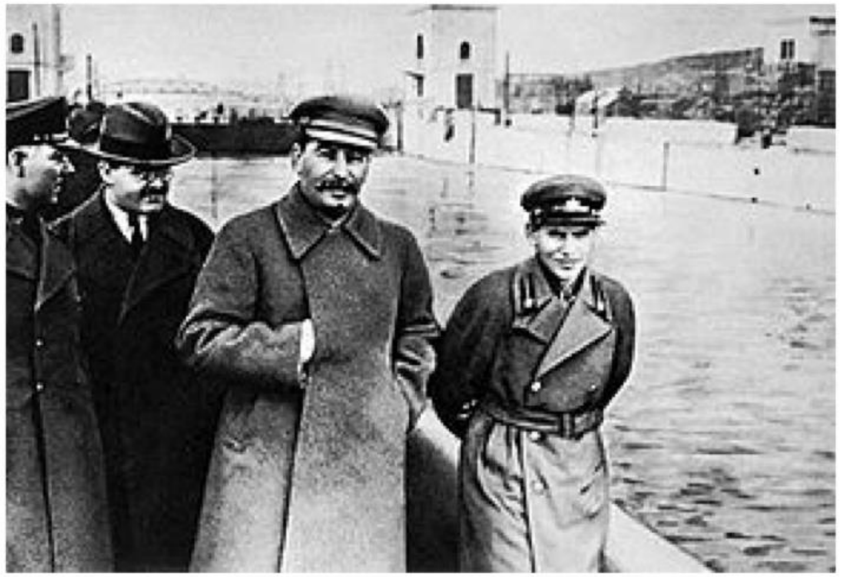


Image Inpainting

- Inpainting is the process of producing a complete image from an image with damaged, deteriorating, or missing parts



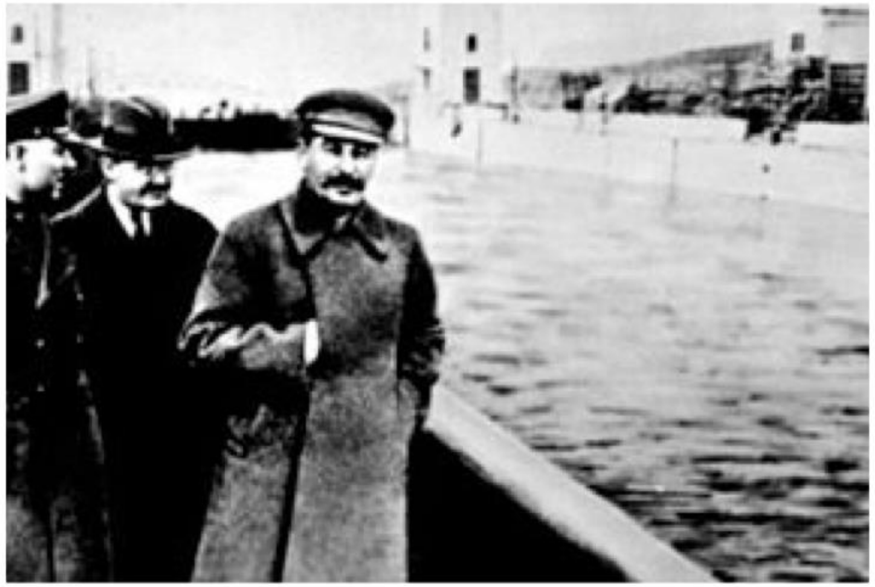
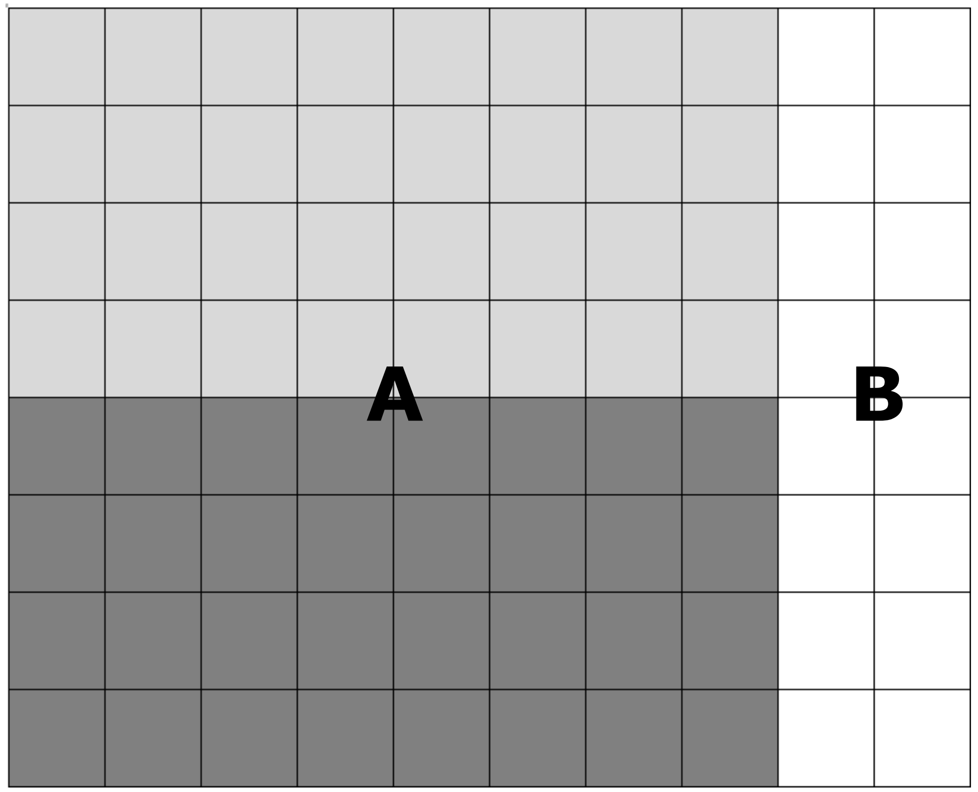


Image Inpainting

Well-posed problem



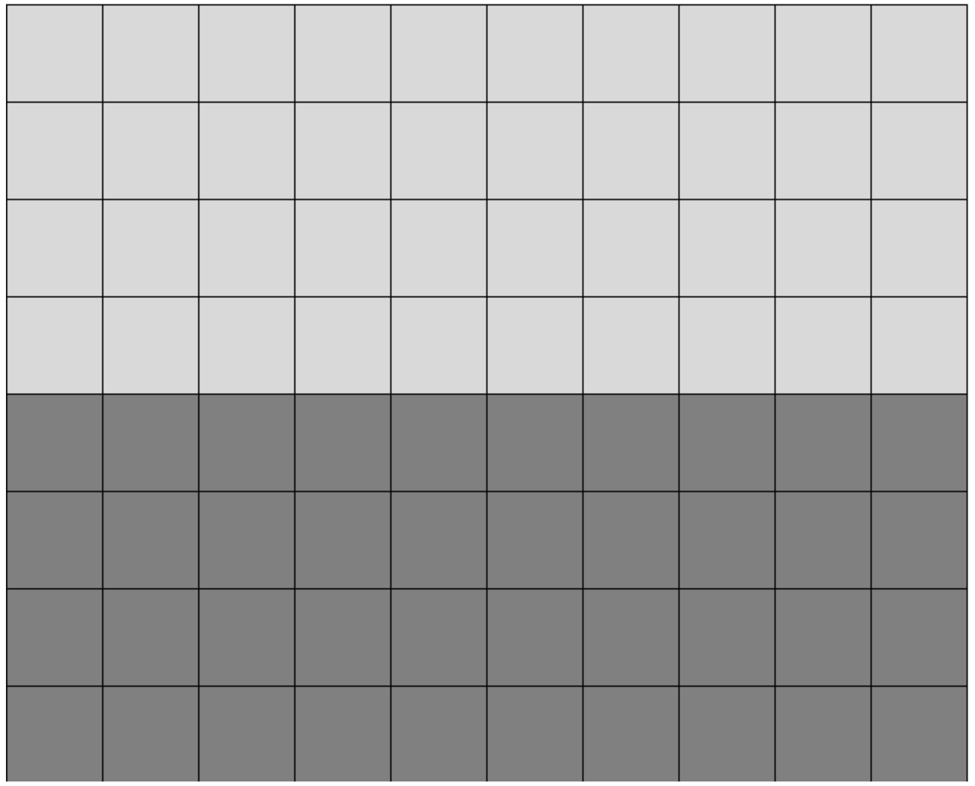
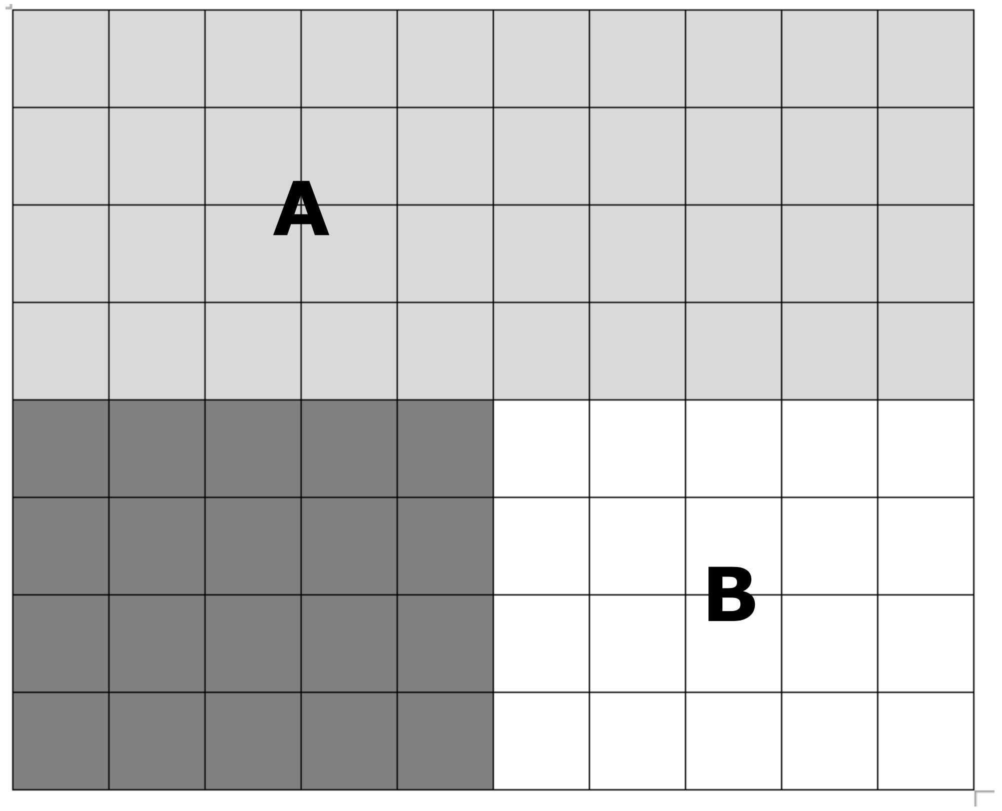


Image Inpainting

Ill-posed problem



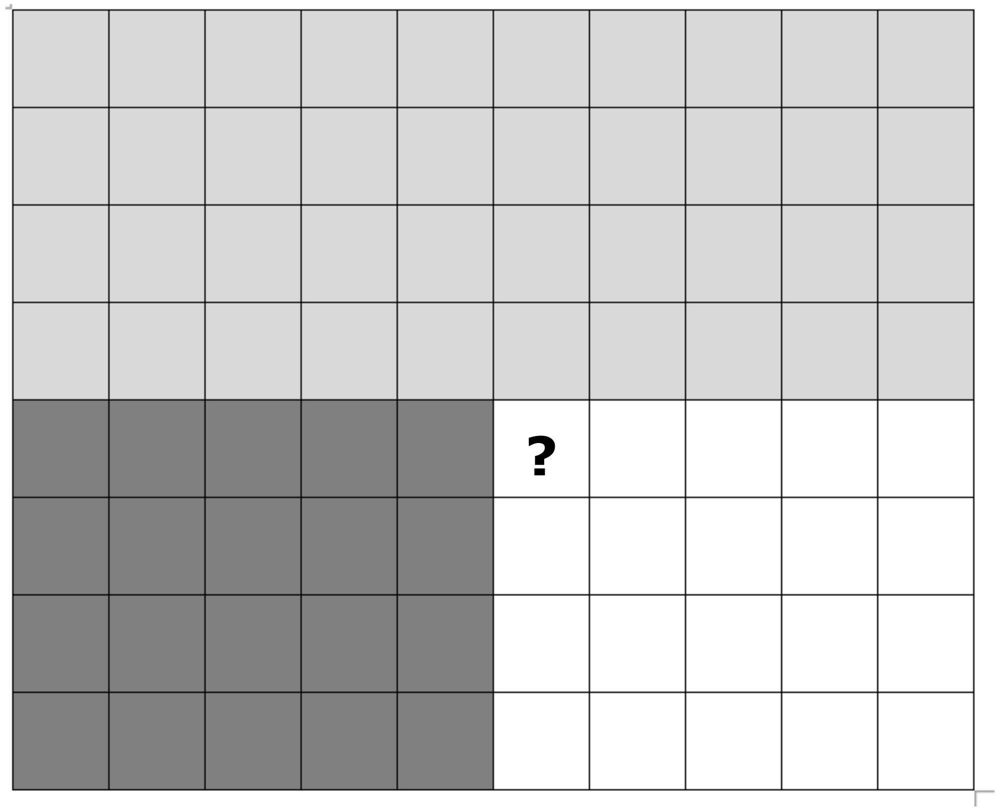


Image Inpainting

- We have an image U = (A , B), we produce a new image V
- Criteria:
  1. In A, the inpainted image should be the same in V as in U
  2. In B, the inpainted image should be smooth

$$V(x,y) = U(x,y) \text{ at each } (x,y) \text{ in A}$$

$$4V(x,y) - (V(x-1,y) + V(x-1,y) + V(x,y-1) + V(x,y+1)) = 0 \text{ at each } (x,y) \text{ in B}$$

Image Inpainting

- We must solve this for m by n pixels (e.g. 128 * 128 = 16384)
- We will have m by n equations and unknowns (system of linear equations)
- But we will work with (m+2) by (n+2) images (ghost boundaries)

$$\begin{pmatrix} 2 & -1 & 0 & \dots & 0 & -1 & 0 & & & \dots & & & 0 \\ & & & & & & \dots & & & & & & \\ 0 & & \dots & & & 0 & 1 & 0 & & & \dots & & 0 \\ & & & & & & \dots & & & & & & \\ 0 & \dots & 0 & -1 & 0 & \dots & 0 & -1 & 4 & -1 & 0 & \dots & 0 \;\; -1 \;\; 0 \;\; \dots \;\; 0 \\ & & & & & & \dots & & & & & & \\ 0 & & & & \dots & & & 0 & -1 & 0 & \dots & 0 \;\; -1 & 2 \end{pmatrix} * \begin{pmatrix} v_1 \\ \dots \\ v_{i,j} \\ \dots \\ \dots \\ \dots \\ v_{(m+2)*(n+2),(m+2)*(n+2)} \end{pmatrix} = \begin{array}{c} \begin{pmatrix} 0 \\ \dots \\ u_{i,j} \\ \dots \\ 0 \\ \dots \\ 0 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \text{North boundary} \\ \\ \leftarrow \text{Region A} \\ \\ \leftarrow \text{Region B} \\ \\ \leftarrow \text{South boundary} \end{array}$$

$\uparrow$ Size = ((m+2)*(n+2) , (m+2)*(n+2))

$\uparrow$ Size = ((m+2)*(n+2) , 1)

$\uparrow$ Size = ((m+2)*(n+2) , 1)

Code

```
A=sparse(idx_Ai, idx_Aj, a_ij, ???, ???); %??? and ???? is the size of matrix A

x=mldivide(A,b);      u_ext=reshape(x, ni+2, nj+2);


%Inner points
for j=2:nj+1
    for i=2:ni+1

        %from image matrix (i,j) coordinates to vectorial (p) coordinate
        p = (j-1)*(ni+2)+i;

        if (dom2Inp_ext(i,j)==1) %If we have to inpaint this pixel

            %Fill Idx_Ai, idx_Aj and a_ij with the corresponding values and
            %vector b
            %TO COMPLETE
```

Goal

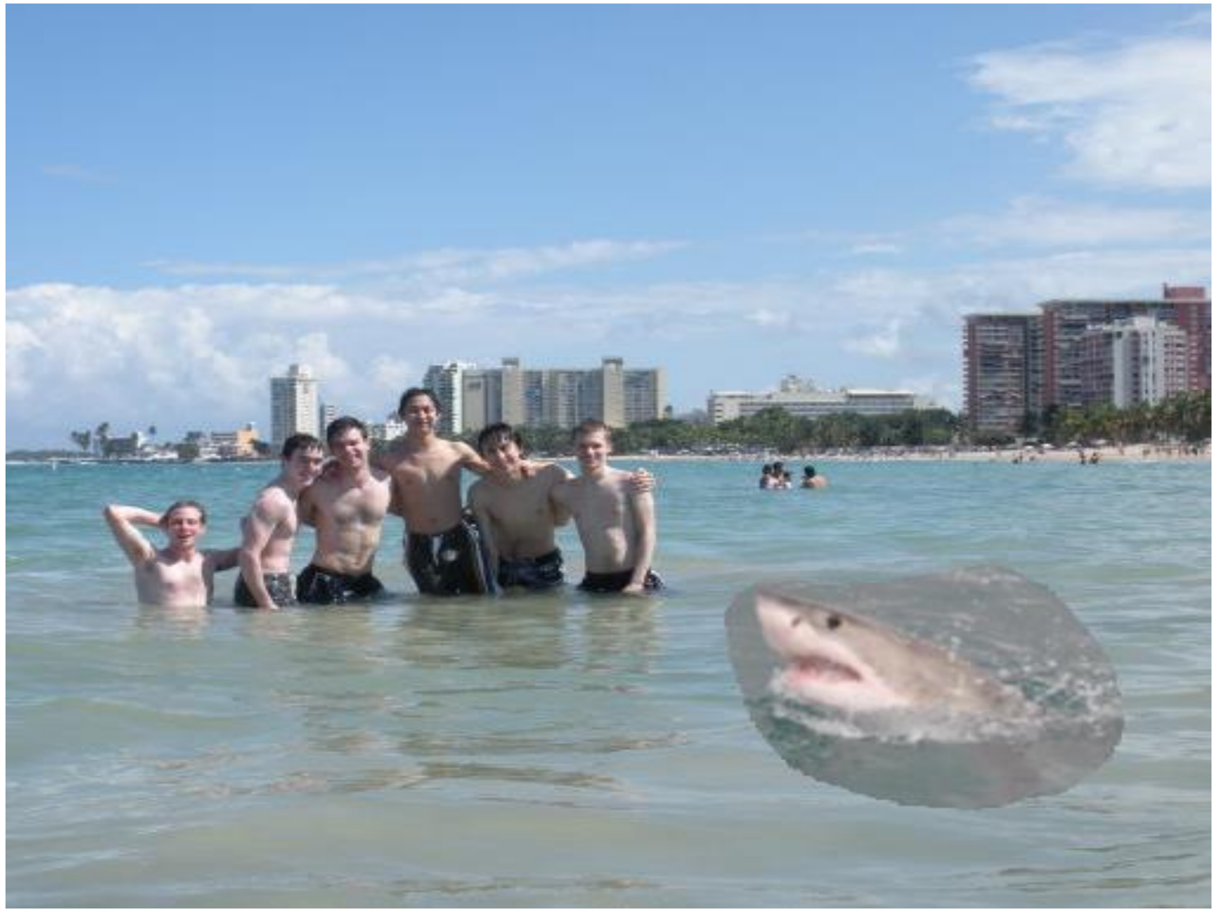
Image A

Goal

Image B

Goal

Image H
(Incorrect)



Goal

Image H
(Correct)

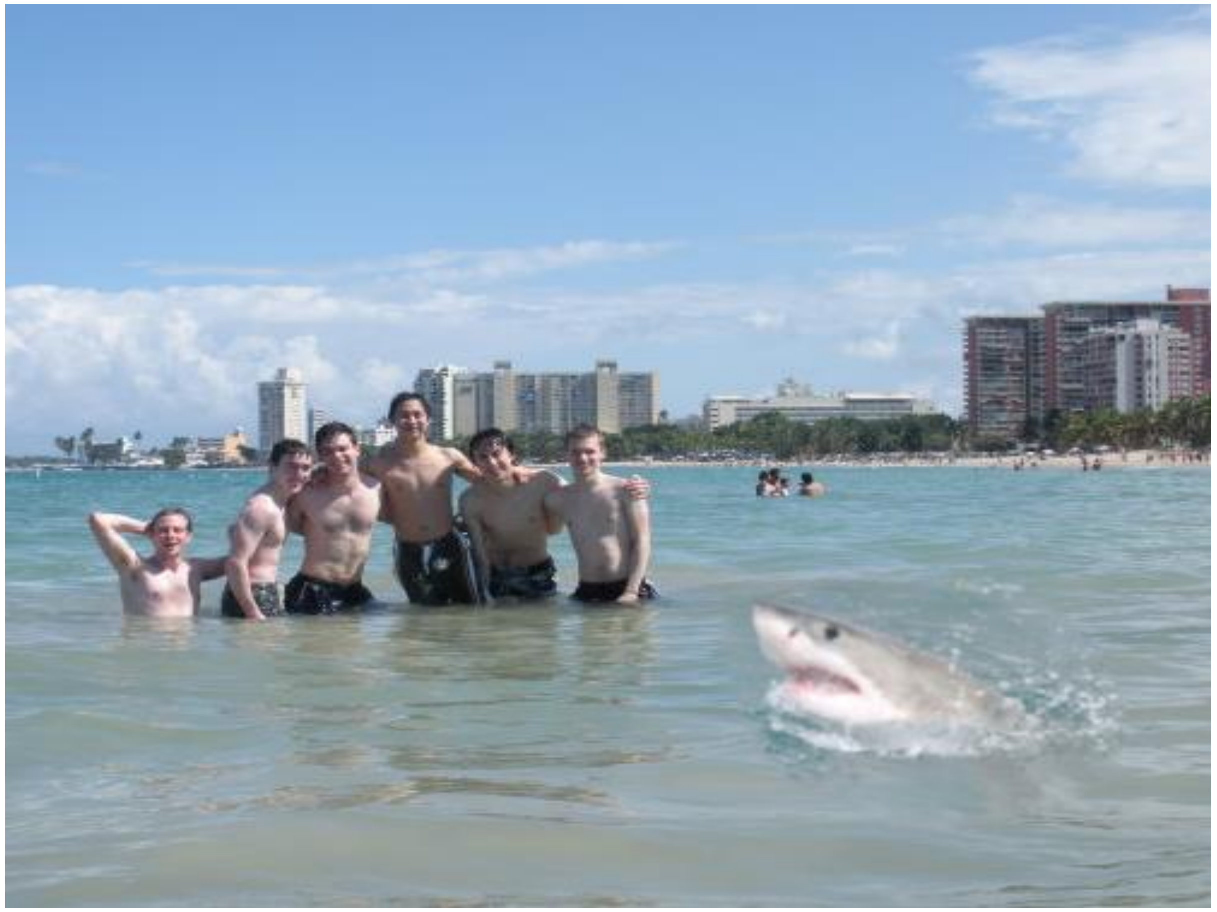


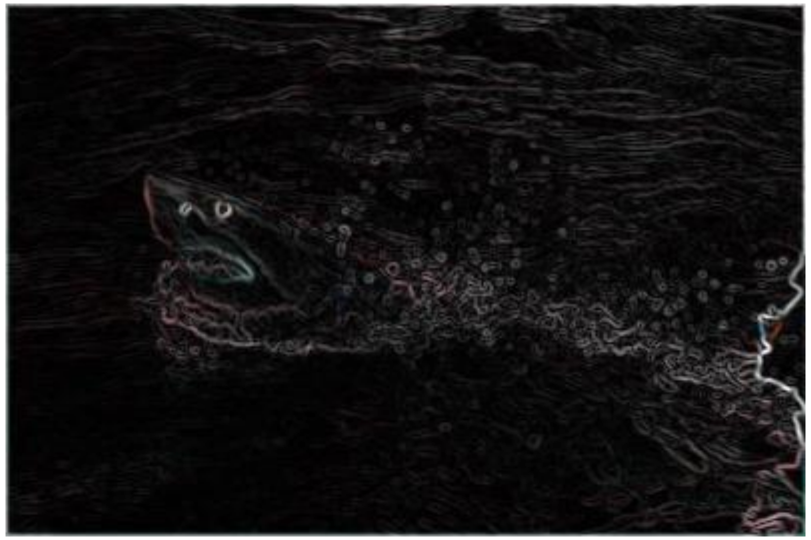
Image H (new)

Method

Image B

Method

∇B (gradient)



Method

- Criterion 1: We want the transition between A and B to be smooth

$H(x, y) = A(x, y)$ at each (x, y) of the boundary of B

- Criterion 2: We want to keep the details of image B, i.e. the gradients at each point

$\nabla H(x, y) = \nabla B(x, y)$ at each (x, y) inside B

Method

- Criterion 1: We want the transition between A and B to be smooth

$H(x, y) = A(x, y)$ at each (x, y) of the boundary of B

- Criterion 2: We want to keep the details of image B, i.e. the gradients at each point

$$4H(x, y) - \sum H(x + dx, y + dy) = 4B(x, y) - \sum B(x + dx, y + dy)$$

Image Inpainting

- We have an image U = (A , B), we produce a new image V
- Criteria:
  1. In A, the inpainted image should be the same in V as in U
  2. In B, the inpainted image should be smooth

$V(x,y) = U(x,y)$ at each (x,y) in A

$4V(x,y) - (V(x-1,y) + V(x-1,y) + V(x,y-1) + V(x,y+1)) = 0$ at each (x,y) in B

Image Inpainting

- We must solve this for m by n pixels (e.g. 128 * 128 = 16384)
- We will have m by n equations and unknowns (system of linear equations)
- But we will work with (m+2) by (n+2) images (ghost boundaries)

$$\begin{pmatrix} 2 & -1 & 0 & \dots & 0 & -1 & 0 & & & & \dots & & & & 0 \\ & & & & & & & \dots & & & & & & & \\ 0 & & & \dots & & & 0 & 1 & 0 & & & \dots & & & 0 \\ & & & & & & & \dots & & & & & & & \\ 0 & \dots & 0 & -1 & 0 & \dots & 0 \;\; -1 & 4 & -1 & 0 & \dots & 0 & -1 & 0 \;\; \dots & 0 \\ & & & & & & & \dots & & & & & & & \\ 0 & & & & \dots & & & & 0 & -1 & 0 & \dots & 0 & -1 & 2 \end{pmatrix} * \begin{pmatrix} v_1 \\ \dots \\ v_{i,j} \\ \dots \\ \dots \\ \dots \\ v_{(m+2)*(n+2),(m+2)*(n+2)} \end{pmatrix} = \begin{array}{l} \begin{pmatrix} 0 \\ \dots \\ u_{i,j} \\ \dots \\ 0 \\ \dots \\ 0 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \text{North boundary} \\ \\ \leftarrow \text{Region A} \\ \\ \leftarrow \text{Region B} \\ \\ \leftarrow \text{South boundary} \end{array}$$

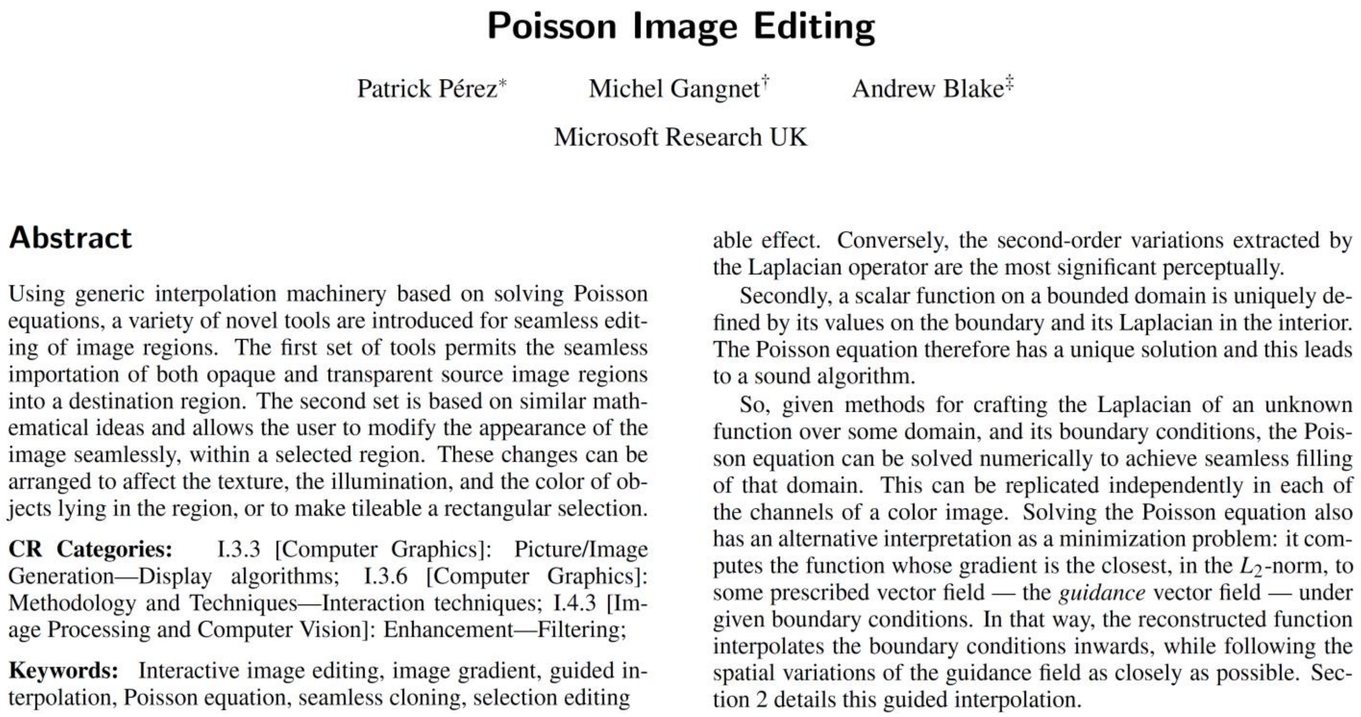
Size = ((m+2)*(n+2) , (m+2)*(n+2))

Size = ((m+2)*(n+2) , 1)

Size = ((m+2)*(n+2) , 1)

Other Methods

- Seamless cloning with mixing gradients



Poisson Image Editing

Patrick Pérez* Michel Gangnet† Andrew Blake‡

Microsoft Research UK

Abstract

Using generic interpolation machinery based on solving Poisson equations, a variety of novel tools are introduced for seamless editing of image regions. The first set of tools permits the seamless importation of both opaque and transparent source image regions into a destination region. The second set is based on similar mathematical ideas and allows the user to modify the appearance of the image seamlessly, within a selected region. These changes can be arranged to affect the texture, the illumination, and the color of objects lying in the region, or to make tileable a rectangular selection.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.4.3 [Image Processing and Computer Vision]: Enhancement—Filtering;

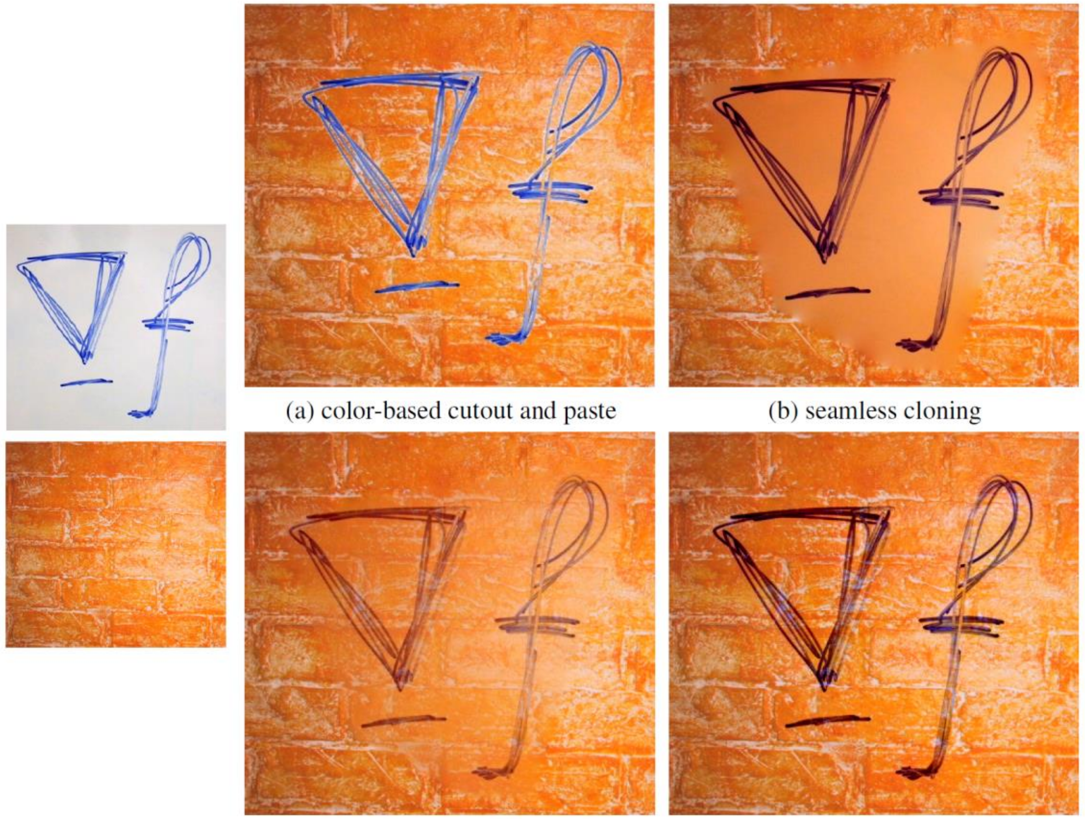
Keywords: Interactive image editing, image gradient, guided interpolation, Poisson equation, seamless cloning, selection editing

able effect. Conversely, the second-order variations extracted by the Laplacian operator are the most significant perceptually.

Secondly, a scalar function on a bounded domain is uniquely defined by its values on the boundary and its Laplacian in the interior. The Poisson equation therefore has a unique solution and this leads to a sound algorithm.

So, given methods for crafting the Laplacian of an unknown function over some domain, and its boundary conditions, the Poisson equation can be solved numerically to achieve seamless filling of that domain. This can be replicated independently in each of the channels of a color image. Solving the Poisson equation also has an alternative interpretation as a minimization problem: it computes the function whose gradient is the closest, in the L_2-norm, to some prescribed vector field — the *guidance* vector field — under given boundary conditions. In that way, the reconstructed function interpolates the boundary conditions inwards, while following the spatial variations of the guidance field as closely as possible. Section 2 details this guided interpolation.

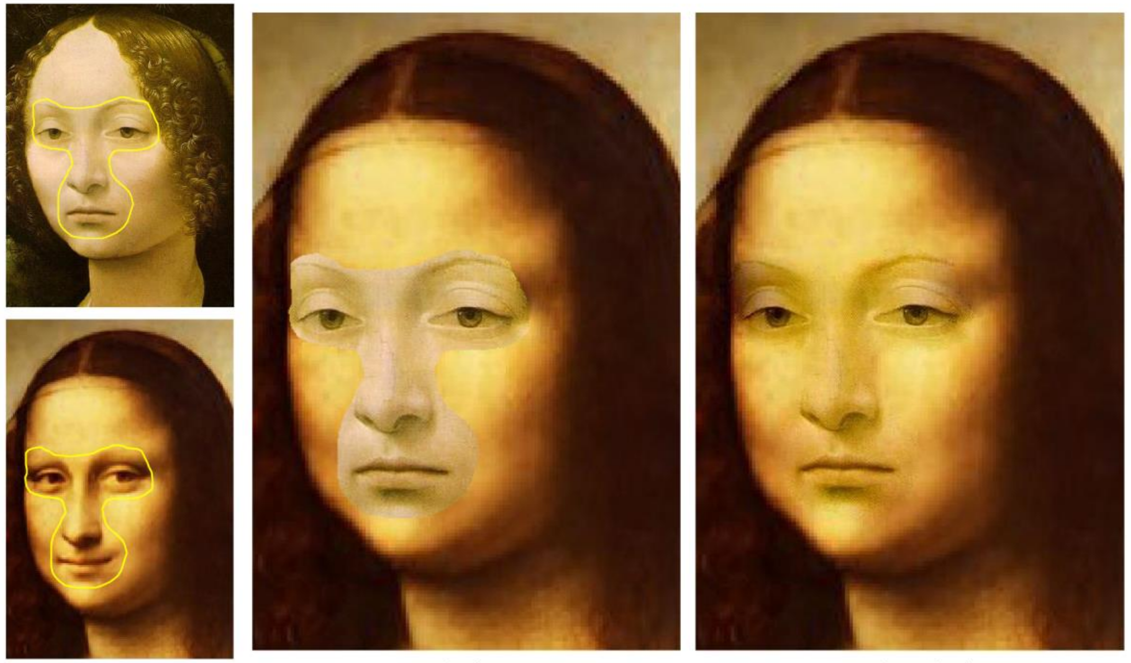
Other Examples


(a) color-based cutout and paste

(b) seamless cloning

(c) seamless cloning and destination av- eraged

(d) mixed seamless cloning


source/destination

cloning

seamless cloning