Introduction to Algebraic Methods
for
Computer Vision and Image Processing

Dr. Juan Fco. Garamendi

Universitat Pompeu Fabra

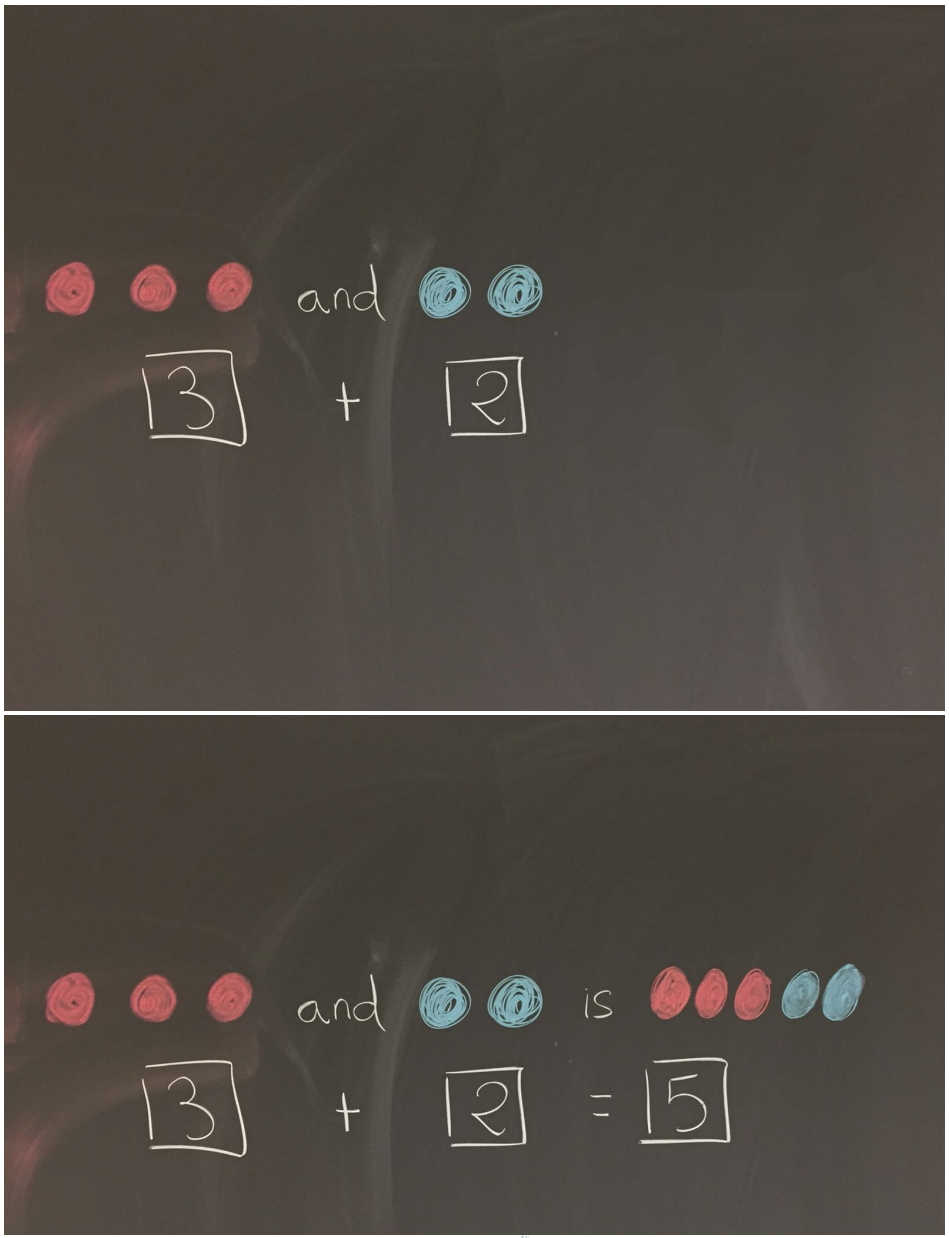
jfgaramendi@gmail.com

Optimization and Inference Techniques for Computer Vision

Master in
Computer Vision
Barcelona

October 4. 2021

Mathematical Modelling



- A mathematical model is a description of a system using mathematical concepts and language .
- A Model may help to explain a system and to study the effects of different components, and to make

A (mathematical) concept we will need

Vector

- An element of a vector space.
  - Collection of objects which may be added and scaled by numbers.
  - A collection of axioms.
    - Associativity of addition
    - Commutativity of addition
    - Blah, blah, blah.....

(Go to your first year degree course)

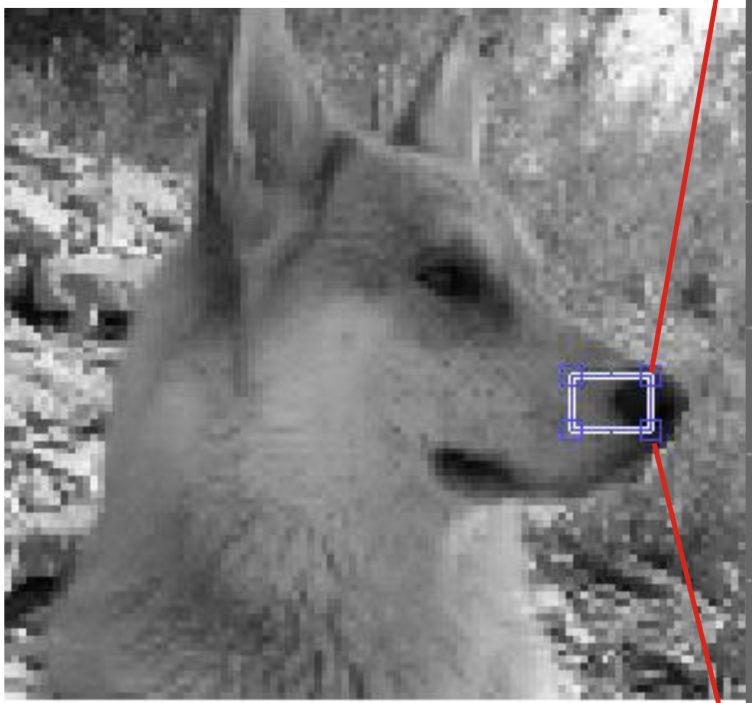
A basic example: Ordered tupplas of real numbers x, y and z:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ y_1 + y_2 \\ z_1 + z_2 \end{pmatrix}$$

A (mathematical) concept we will need

Vector

Images can be modelled as a vector pixel values. Each point in a multi-dimensional space represent an image



0.37	0.35	0.40	0.36	0.36	0.28	0.24	0.16	0.14
0.35	0.38	0.36	0.38	0.33	0.25	0.16	0.11	0.10
0.40	0.35	0.35	0.29	0.27	0.09	0.10	0.09	0.10
0.39	0.37	0.33	0.30	0.20	0.07	0.09	0.09	0.09
0.41	0.40	0.32	0.32	0.23	0.11	0.08	0.08	0.08
0.40	0.39	0.37	0.33	0.25	0.20	0.09	0.07	0.07
0.41	0.38	0.37	0.34	0.33	0.26	0.16	0.07	0.04
0.40	0.38	0.35	0.35	0.34	0.28	0.19	0.13	0.13

A (mathematical) concept we will need

Vector

The image is a 2D arrange of pixels. In RGB color space, each pixel is a **vector (of 3 components)**

A (mathematical) concept we will need

Vector

A different example: 3x3 pixel image u

$$\mathbf{u} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}$$

is modelled as a vector $\bar{u} \in \mathbb{R}^9 := \chi$

$$\bar{u} = \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{pmatrix}$$

We can define a scalar product and a norm over χ (the vector space of mxn images)

$$\langle \bar{u}, \bar{v} \rangle_\chi = \sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij} v_{ij}$$

and

$$||u||_\chi = \sqrt{\langle \bar{u}, \bar{u} \rangle_\chi} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij}^2}$$

Image Processing

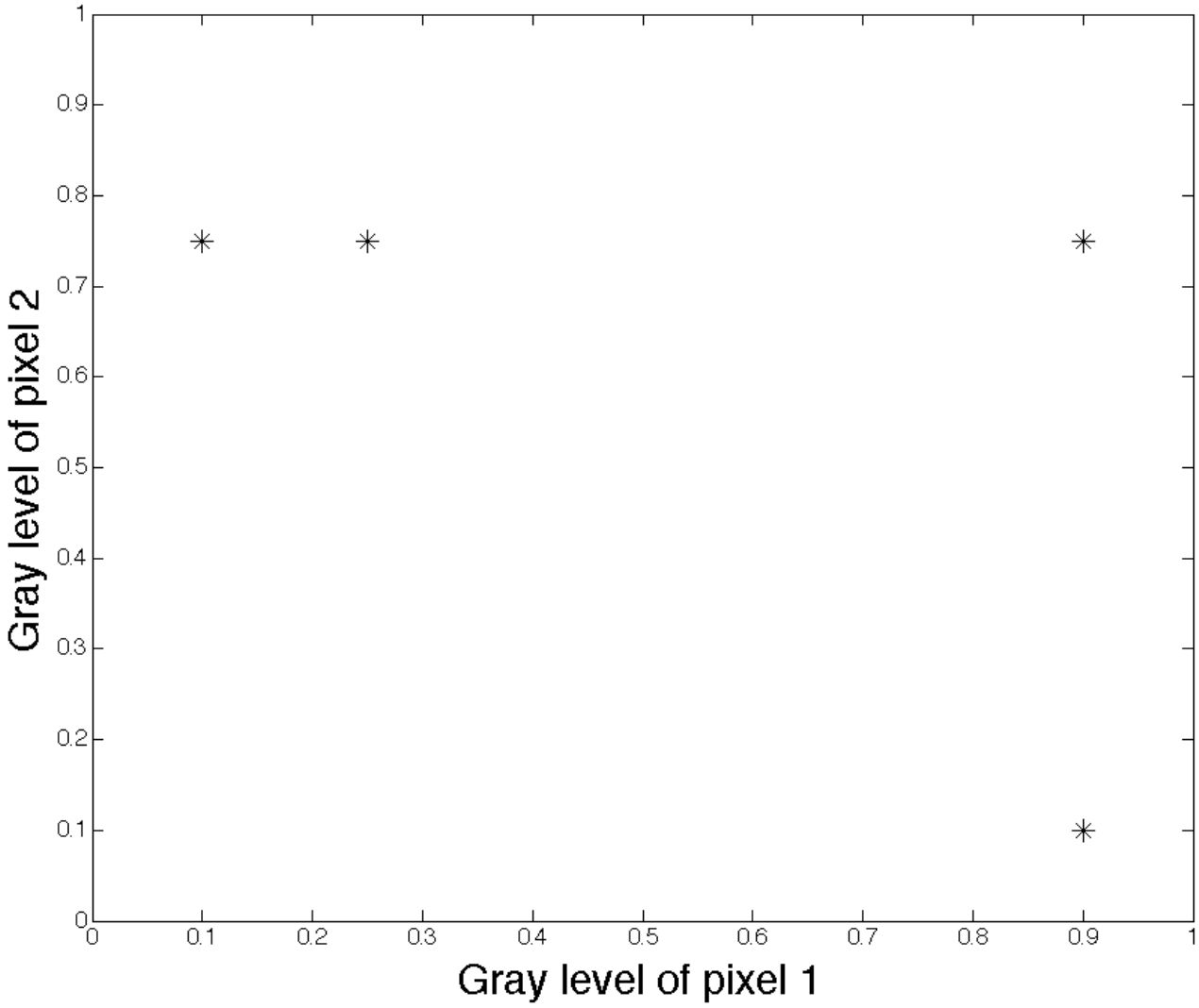
'Algebraic' World

Image 1 with only two pixels

$$I_1 = \begin{pmatrix} 0.1 \\ 0.75 \end{pmatrix}$$

Image 2 with only two pixels

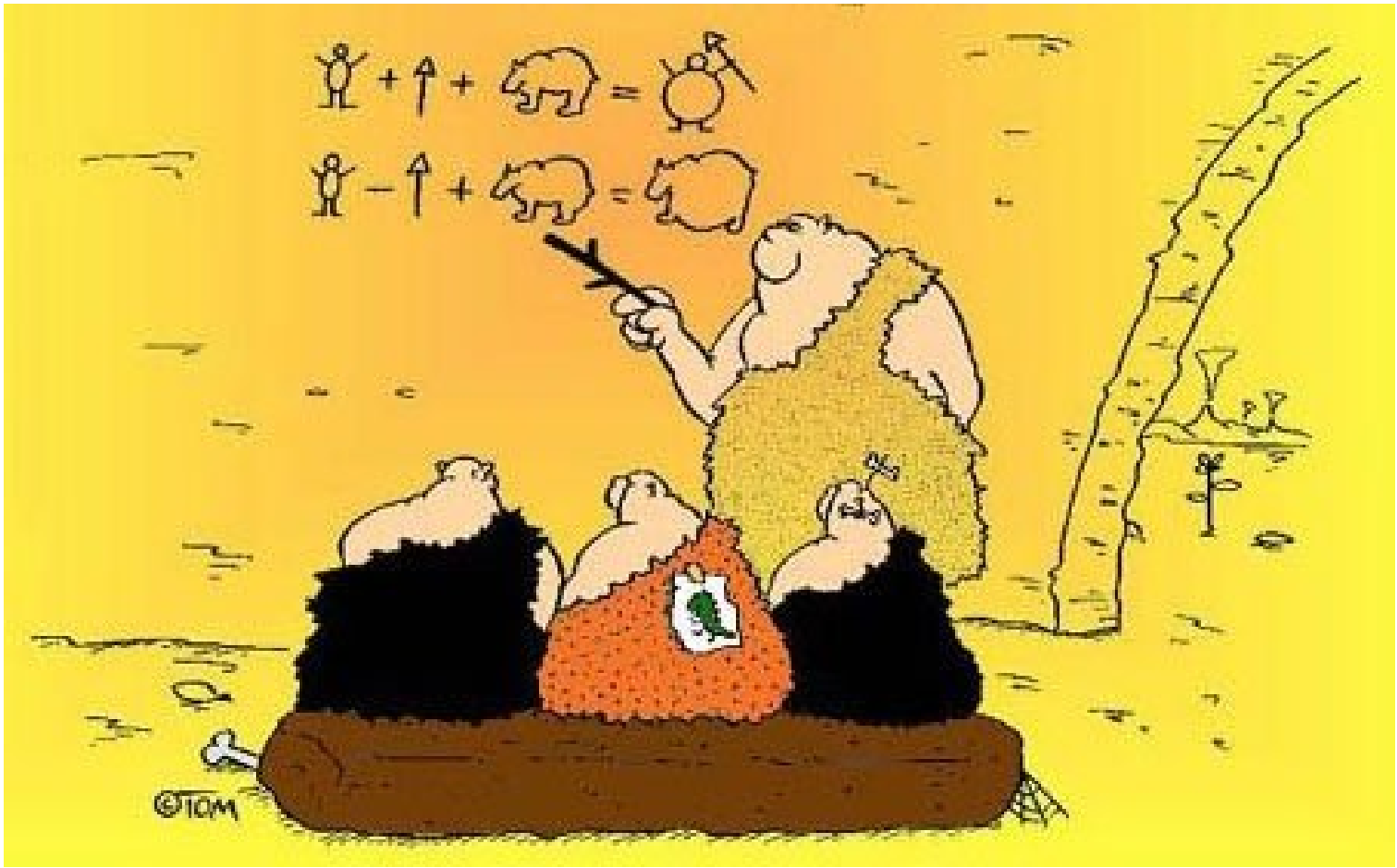
Image Processing



Today Message

- Our images are vectors (points) of a vectorial space. i.e. a n-pixel image is modelled as a point of $\mathbb{R}^n$.
- Express your problem as a function minimization over a vectorial space.

Modelling Real World Problems as Minimization Problems



AA: Modelling as Minization

Advantage

Once the problem in the real world (denoising, restoration, segmentation, optical flow.....) is modelled as a minimization problem, we can apply all the well established mathematical theory and tools to solve the problem.

Examples

Image restoration

Let's try first with our 2-pixel image

Given $\bar{f} = (f_1, f_2)^T$, the (damaged 2-pixel) image to restore, we describe the characteristics of the (unknown) restored image $\bar{u} = (u_1, u_2)^T$, the solution.

- Similar to the original image $\bar{f}$: Overall difference between $\bar{u}$ and $\bar{f}$ is small.

  $$(u_1 - f_1)^2 + (u_2 - f_2)^2$$

  or what is the same

  $$||\bar{f} - \bar{u}||^2$$

  where $||\cdot||$ is the euclidean norm
- Without noise: The noise changes very quickly from one pixel to the next one. So the solution should be "smooth".

  $$(u_1 - u_2)^2$$

Image restoration

Let's try first with our 2-pixel image

The 2-variable (as many variables as pixels) function to minimize is

$$F(u_1, u_2) = (u_1 - u_2)^2 + (u_1 - f_1)^2 + (u_2 - f_2)^2$$

Image restoration

Let's go with a MxN pixels image

Given $\bar{f}$, the (damaged) image to restore, we describe the characteristics of the (unknown) restored image $\bar{u}$, the solution.

- Similar to the original image $\bar{f}$: Overall difference between $\bar{u}$ and $\bar{f}$ is small.

  $$||\bar{f} - \bar{u}||^2$$

  where $||\cdot||$ is the euclidean norm
- Without noise: The noise changes very quickly from one pixel to the next one. So the solution should be "smooth".

  $$||\nabla^+ \bar{u}||^2$$

  where matrix ∇^+ is an operator to compute the difference between adjacent pixels

The function to minimize (with many variables as pixels) is

$$F(\bar{u}) = ||\nabla^+ \bar{u}||^2 + ||\bar{f} - \bar{u}||^2$$

Image restoration

Let's take a look to matrix ∇^+

As example we will use this 4x5 image

$$u = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \end{bmatrix}$$

Image restoration

Let's take a look to matrix ∇^+

Firstly, We define the matrix ∇_i^+. This matrix computes pixel differences along *i* direction (i.e vertical direction)

$$\nabla_i^+ u = \begin{bmatrix}
-1 & 1 & & & & & & & & & & & & & & \\
 & -1 & 1 & & & & & & & & & & & & & \\
 & & -1 & 1 & & & & & & & & & & & & \\
 & & & 0 & & & & & & & & & & & & \\
 & & & & -1 & 1 & & & & & & & & & & \\
 & & & & & -1 & 1 & & & & & & & & & \\
 & & & & & & -1 & 1 & & & & & & & & \\
 & & & & & & & 0 & & & & & & & & \\
 & & & & & & & & -1 & 1 & & & & & & \\
 & & & & & & & & & -1 & 1 & & & & & \\
 & & & & & & & & & & -1 & 1 & & & & \\
 & & & & & & & & & & & 0 & & & & \\
 & & & & & & & & & & & & -1 & 1 & & \\
 & & & & & & & & & & & & & -1 & 1 & \\
 & & & & & & & & & & & & & & -1 & 1 \\
 & & & & & & & & & & & & & & & 0
\end{bmatrix} \cdot \begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{41} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{42} \\ u_{13} \\ u_{23} \\ u_{33} \\ u_{43} \\ u_{14} \\ u_{24} \\ u_{34} \\ u_{44} \\ u_{15} \\ u_{25} \\ u_{35} \\ u_{45} \end{bmatrix} = \begin{bmatrix} u_{21} - u_{11} \\ u_{31} - u_{21} \\ u_{41} - u_{31} \\ 0 \\ u_{22} - u_{12} \\ u_{32} - u_{22} \\ u_{42} - u_{32} \\ 0 \\ u_{23} - u_{13} \\ u_{33} - u_{23} \\ u_{43} - u_{33} \\ 0 \\ u_{24} - u_{14} \\ u_{34} - u_{24} \\ u_{44} - u_{34} \\ 0 \\ u_{25} - u_{15} \\ u_{35} - u_{25} \\ u_{45} - u_{35} \\ 0 \end{bmatrix}$$

Image restoration

Let's take a look to matrix ∇^+

Secondly, We define the matrix ∇_j^+. This matrix computes pixel differences along *j* direction (i.e horizontall direction)

$$\nabla_j^+ u = \begin{bmatrix} -1 & 1 & & & & & & & & & & & & & & & \\ & -1 & 1 & & & & & & & & & & & & & & \\ & & -1 & 1 & & & & & & & & & & & & & \\ & & & -1 & 1 & & & & & & & & & & & & \\ \hdashline & & & & -1 & 1 & & & & & & & & & & & \\ & & & & & -1 & 1 & & & & & & & & & & \\ & & & & & & -1 & 1 & & & & & & & & & \\ & & & & & & & -1 & 1 & & & & & & & & \\ \hdashline & & & & & & & & -1 & 1 & & & & & & & \\ & & & & & & & & & -1 & 1 & & & & & & \\ & & & & & & & & & & -1 & 1 & & & & & \\ & & & & & & & & & & & -1 & 1 & & & & \\ \hdashline & & & & & & & & & & & & 0 & & & & \\ & & & & & & & & & & & & & 0 & & & \\ & & & & & & & & & & & & & & 0 & & \\ & & & & & & & & & & & & & & & 0 & \end{bmatrix} \cdot \begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{41} \\ \hdashline u_{12} \\ u_{22} \\ u_{32} \\ u_{42} \\ \hdashline u_{13} \\ u_{23} \\ u_{33} \\ u_{43} \\ \hdashline u_{14} \\ u_{24} \\ u_{34} \\ u_{44} \\ \hdashline u_{15} \\ u_{25} \\ u_{35} \\ u_{45} \end{bmatrix} = \begin{bmatrix} u_{12} - u_{11} \\ u_{22} - u_{21} \\ u_{32} - u_{31} \\ u_{42} - u_{41} \\ \hdashline u_{13} - u_{12} \\ u_{23} - u_{22} \\ u_{33} - u_{32} \\ u_{43} - u_{42} \\ \hdashline u_{14} - u_{13} \\ u_{24} - u_{23} \\ u_{34} - u_{33} \\ u_{44} - u_{43} \\ \hdashline u_{15} - u_{14} \\ u_{25} - u_{24} \\ u_{35} - u_{34} \\ u_{45} - u_{44} \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Image restoration

Let's take a look to matrix ∇^+

Thirdly, We define the matrix ∇^+ stacking the two previous matrices ∇_i^+ and ∇_j^+. This matrix computes pixel differences along both *i* and *j* direction

$$\nabla^+ u = \begin{bmatrix} \nabla_i^+ \\ - - - - - \\ \nabla_j^+ \end{bmatrix} \cdot u = \left.\begin{bmatrix} \nabla_i^+ u \\ - - - - \\ \nabla_j^+ u \end{bmatrix}\right\} (\text{vector with } 2MN \text{ components}).$$

Image restoration

Let's go with a MxN pixels image

Given $\bar{f}$, the (damaged) image to restore, we describe the characteristics of the (unknown) restored image $\bar{u}$, the solution.

- Similar to the original image $\bar{f}$: Overall difference between $\bar{u}$ and $\bar{f}$ is small.

  $$||\bar{f} - \bar{u}||_2^2$$

  where $||\cdot||$ is the euclidean norm
- Without noise: The noise changes very quickly from one pixel to the next one. So the solution should be "smooth".

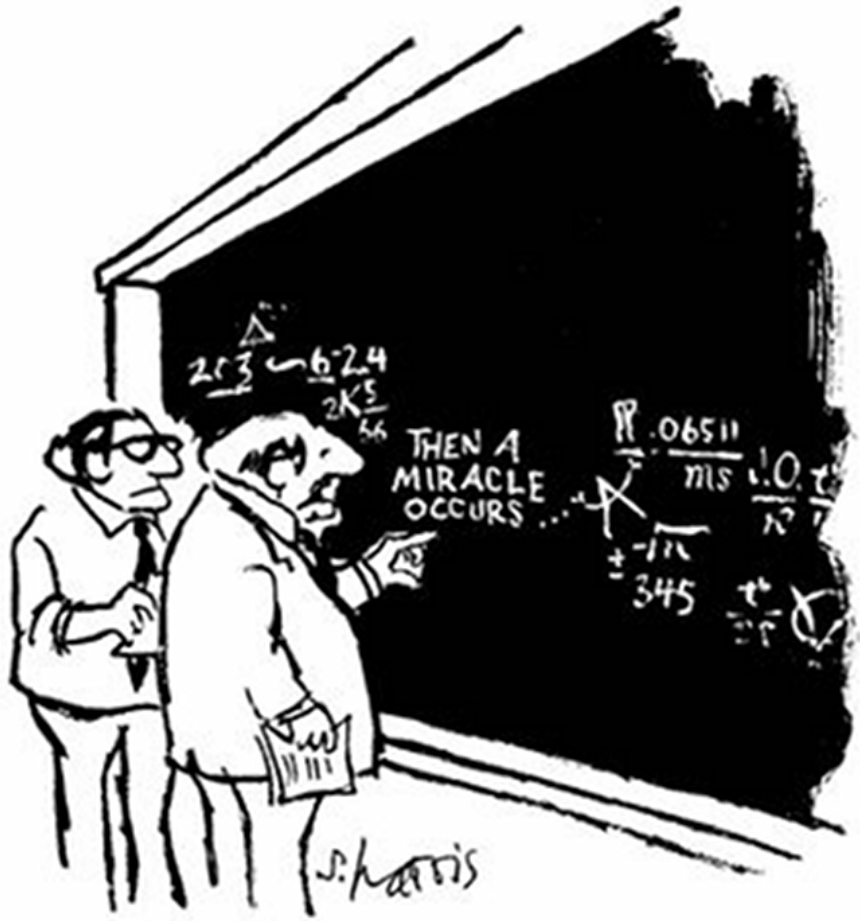
  $$||\nabla^+ \bar{u}||_2^2$$

  where matrix ∇^+ is an operator to compute the difference between adjacent pixels

The function to minimize (with many variables as pixels) is

$$F(\bar{u}) = ||\nabla^+ \bar{u}||_2^2 + ||\bar{f} - \bar{u}||_2^2$$

Understanding the Miracle: From the minimization problem to image solution



"I think you should be more explicit here in step two."

The Mathematical Problem to Solve

For a given cost function

$$F(\bar{u}) = F(x_1, ..., x_{NxM})$$

We look for the variable values $\bar{u} = (x_1, .., x_{NxM})^T$ that makes F be minimum (or maximum).

Solving the Mathematical Problem

Among others, there are two main strategies

- First strategy: Based on Gradient Descent:
  - Start from some initial solution $\bar{u}_0$.

$$\bar{u}^{[0]} = \bar{u}_0$$

  - Iteratively go to another solution $u^{[k+1]}$, following a direction $\bar{dir}$ that is far from the previous solution $u^{[k]}$ by a distance τ and has lower cost.

$$\bar{u}^{[k+1]} = \bar{u}^{[k]} + \tau \bar{dir}$$

  - Remember, gradient points to the direction of maximum change. Do previous step following the gradient descent direction of your function F.

$$\begin{cases} \bar{u}^{[0]} & = & \bar{u}_0 \\ \bar{u}^{[k+1]} & = & \bar{u}^{[k]} - \tau \nabla F \end{cases}$$

Solving the Mathematical Problem

- Second strategy: Finding the necessary condition for the extremum
  - The change of the function with respect to the unkown/s has/have to be zero. i.e, it's gradient has to be zero.
  - So, under some assumptions, if $\bar{u} = (u_1, ..., u_2)^T$ is a extremum, $\bar{u}$ have to satisfy the equation.

$$\nabla F(\bar{u}) = \bar{0}$$

  - More difficult to implement than Gradient Descent, but usually faster.

On both strategies

We have to compute the gradient of $F(\bar{u})$. Have you noticed that $\bar{u}$ is a vector? so, GOOD NEWS!!!! Computing $\nabla F(\bar{u})$ is something you learnt in High School!

IMPORTANT!!!!! Do not confuse matrix $\boldsymbol{\nabla}^+$ and vector $\bar{u}$ multiplication we saw before:

$$\boldsymbol{\nabla}^+ \bar{u}$$

with the gradient of function $F(\bar{u})$

$$\nabla F$$

Example: Inpainting

- Solving the Necessary condition for the extremum (on a row 4-pixel image): Slides
- Gradient descent (on a complete image): Whiteboard

Inpainting

$$
\begin{aligned}
F: \quad \mathbb{R}^n &\rightarrow \mathbb{R}, \\
\vec{x} &\mapsto F(\vec{x}) = F(x_1, .., x_n)
\end{aligned}
$$

Inpainting

For a given vectorized image $\bar{x} = (x_1, .., x_n)^T$ with unknown values $\bar{x_D} = (x_i, .., x_j)^T$, $\bar{x_D} \subset \bar{x}$

$$\bar{u} = \begin{cases} \arg\min_{\bar{u}} ||\nabla^+ \bar{u}||_2^2 \\ \bar{u}|_{\partial D} = \bar{x} \end{cases}$$

Inpainting

A 4-pixel $\bar{x} = (x_1, x_2, x_3, x_4)^T$ image with two unknown pixels x_2 x_3.

$$\arg\min_{u_2,u_3} (x_1 - u_2)^2 + (u_2 - u_3)^2 + (u_3 - x_4)^2$$

Let's give specific values to our known pixels $x_1 = 0.25$ and $x_4 = 0.75$, then our 4-pixel image is $\bar{u} = (0.25, u_2, u_3, 0.75)^T$ and our cost function to minimize is

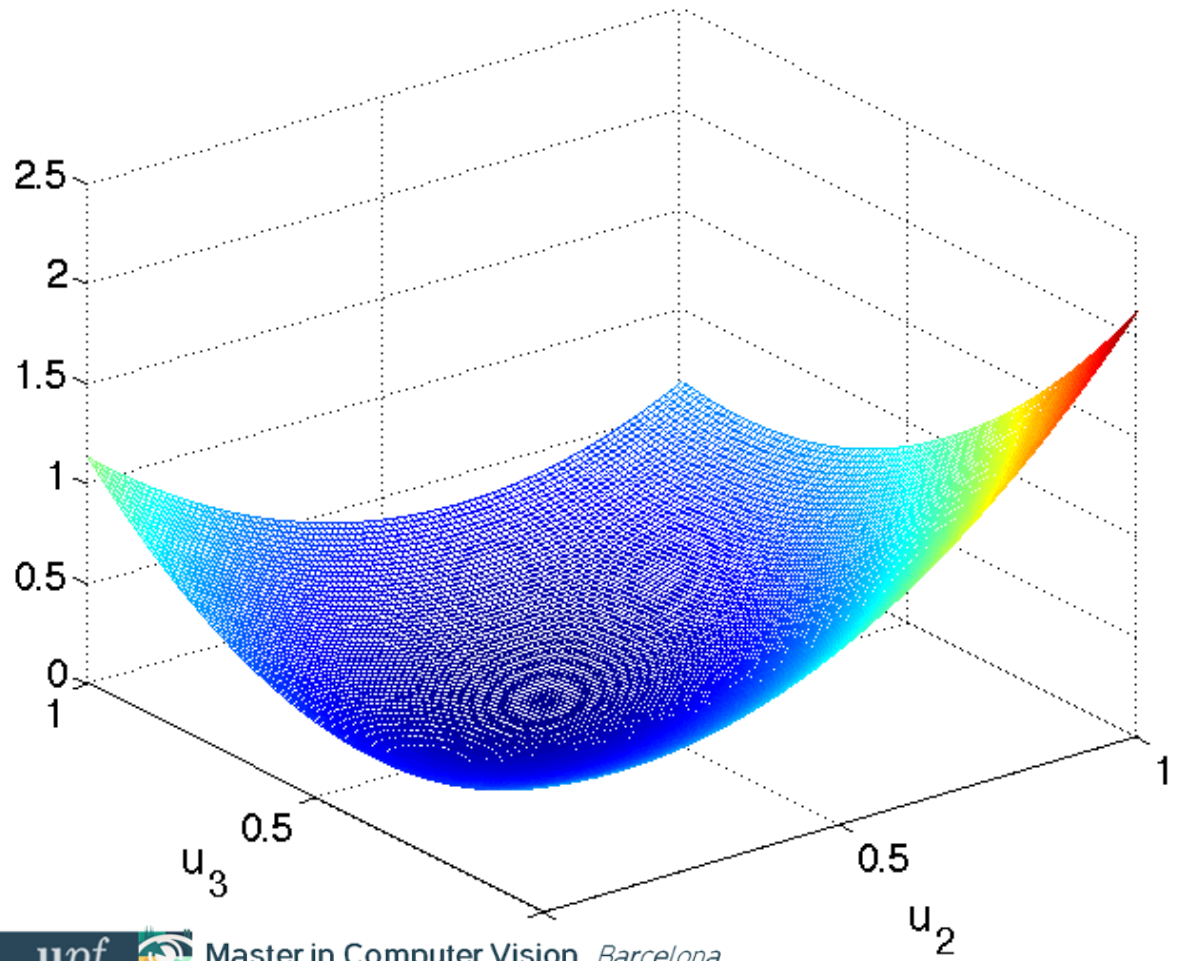
$$\arg\min_{u_2,u_3} (0.25 - u_2)^2 + (u_2 - u_3)^2 + (u_3 - 0.75)^2$$

AA: Inpainting

Obviously can be re-written as

$$\arg\min_{u_2,u_3}(0.25-u_2)^2+(u_2-u_3)^2+(u_3-0.75)^2$$

and it has the form of



Inpainting

Necessary condition for extremum: The change ratio of the cost function with respect to variables u_2 and u_3 has to be zero, i.e, it's partial derivatives have to be zero.
LET'S DO IT!!!!!!
Derivative w.r.t. u_2:

$$-2(0.25 - u_2) + 2(u_2 - u_3) = 0$$

Derivative w.r.t u_3

$$-2(u_2 - u_3) + 2(u_3 - 0.75) = 0$$

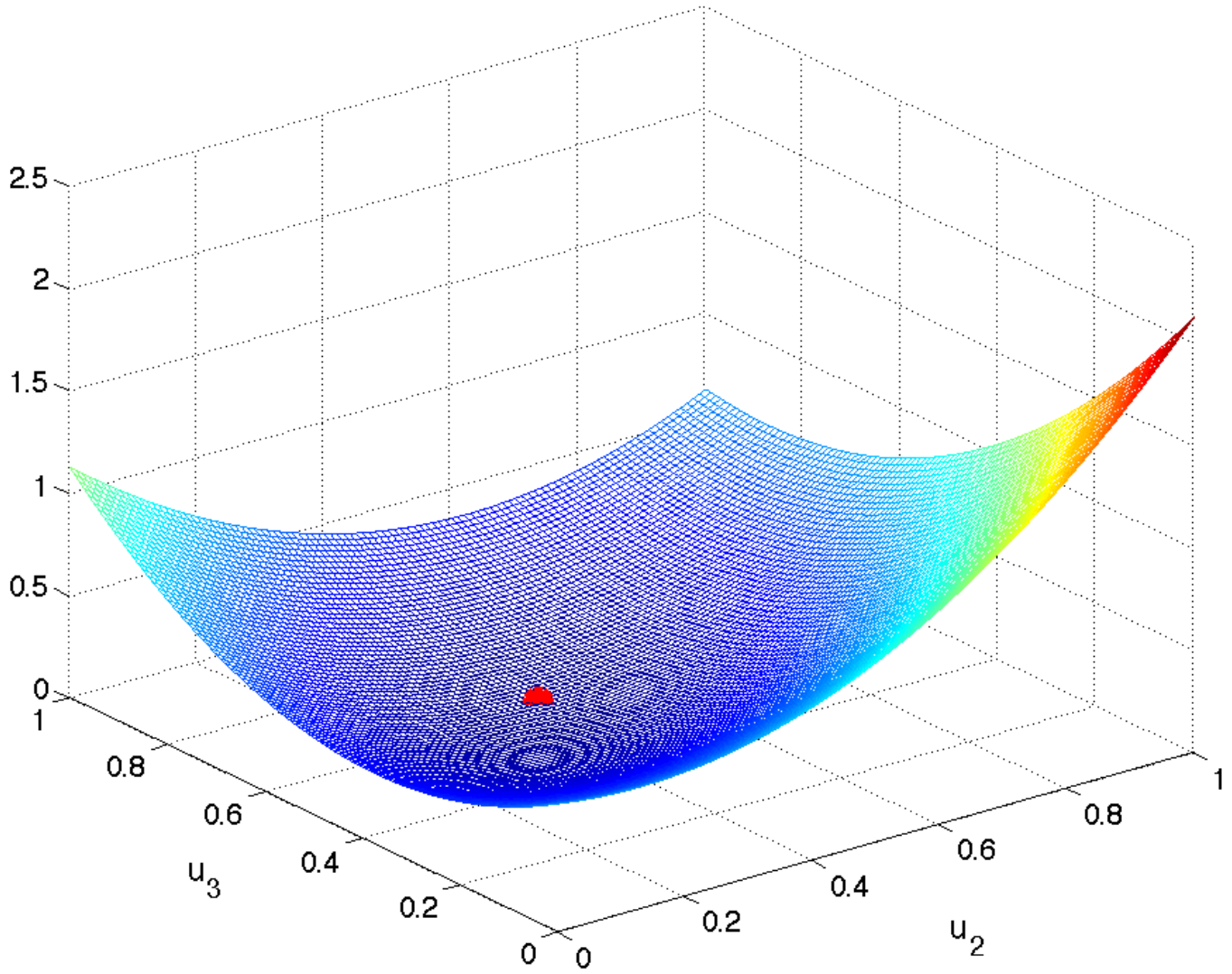
This gives the following system of equations

$$\begin{cases} 4u_2 - 2u_3 &= 0.5 \\ -2u_2 + 4u_3 &= 1.5 \end{cases}$$

with solution $(u_2, u_3) = (0.4167, 0.5833)$, so our solution images is $(0.25, 0.4167, 0.5833, 0.75)^T$

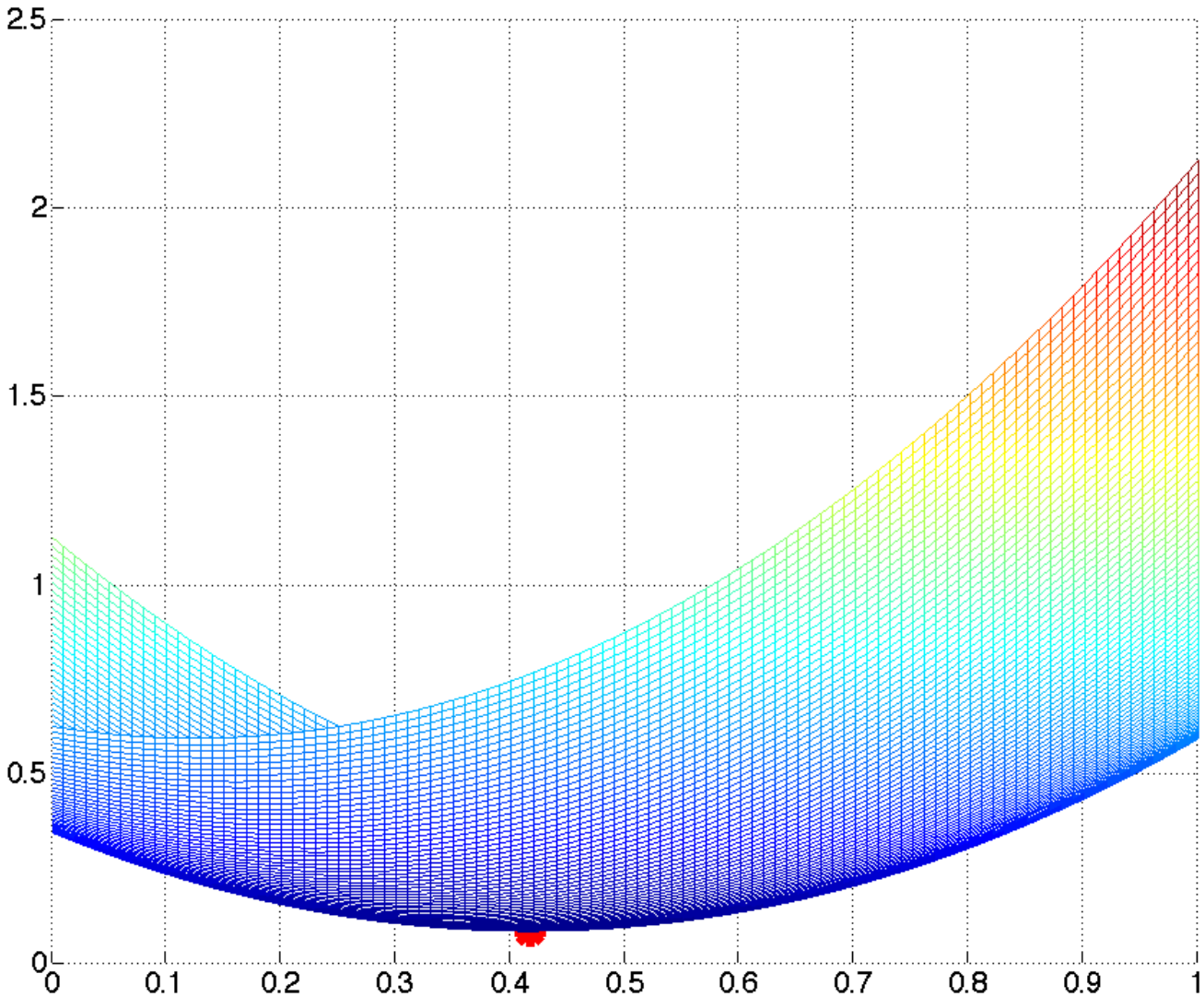
Inpainting

This cost function has the form of



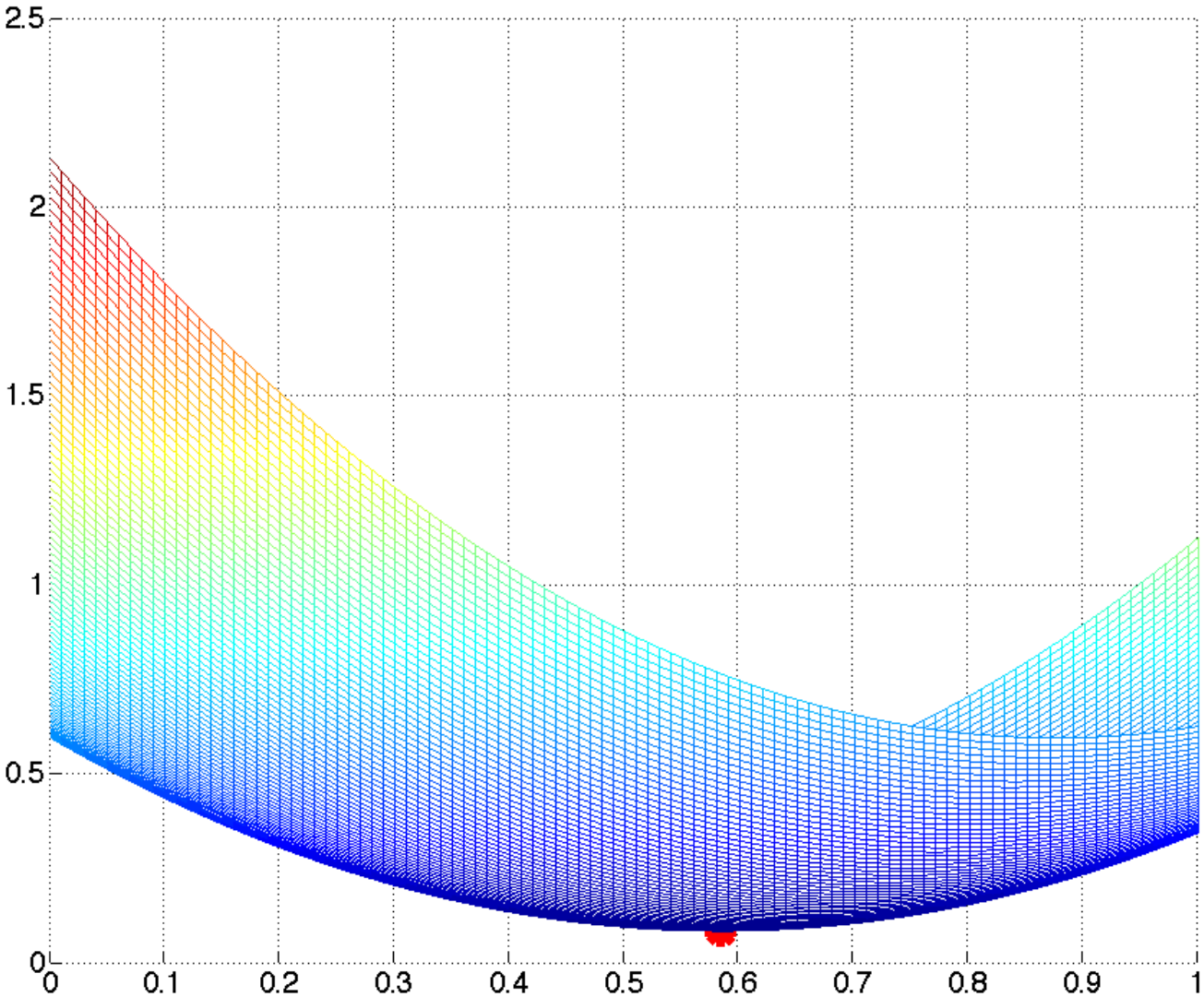
Inpainting

This cost function has the form of

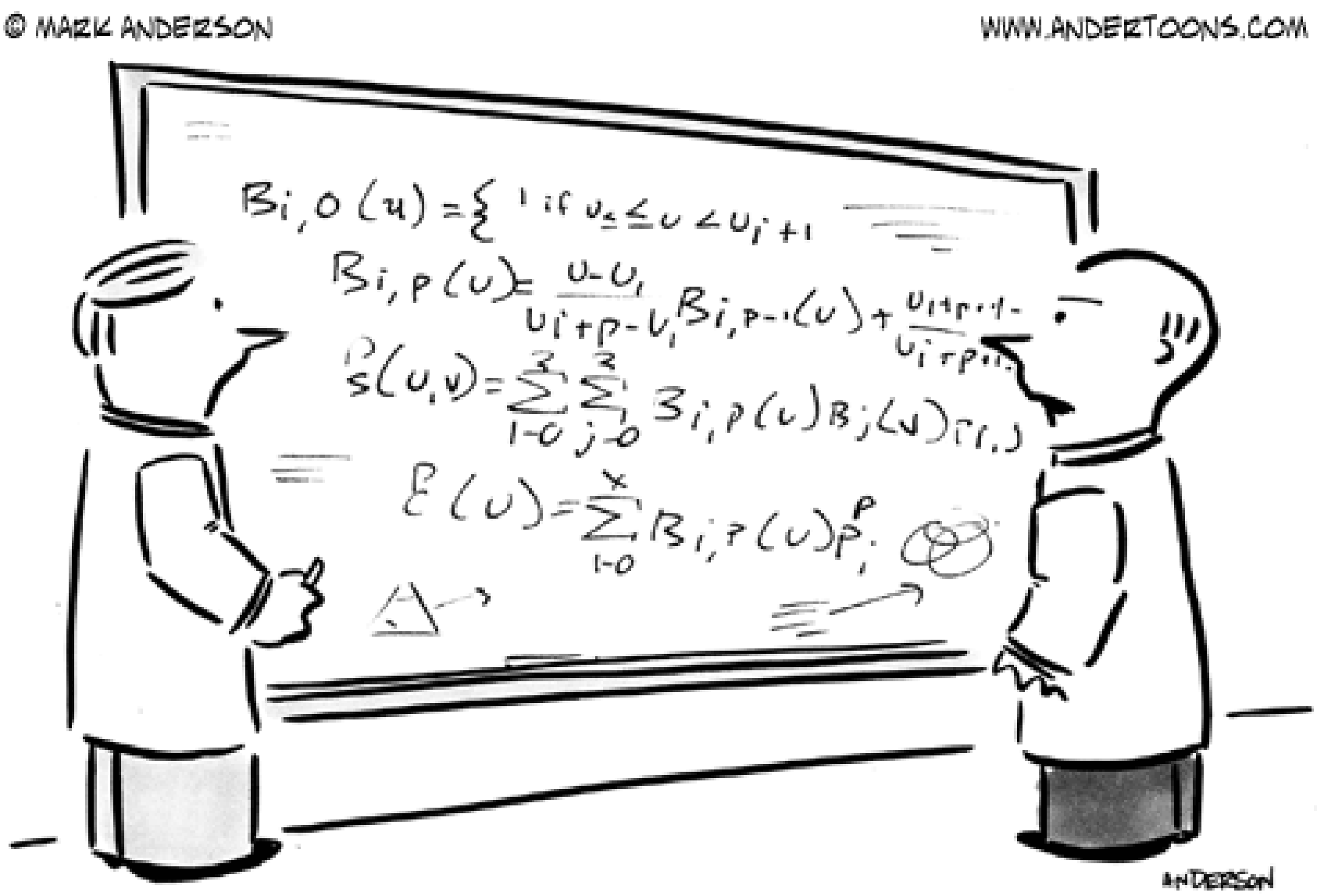


AA: Inpainting

This cost function has the form of



Questions



"What the hell is *that* supposed to mean?!"