M2 - Optimization and Inference Techniques for CV

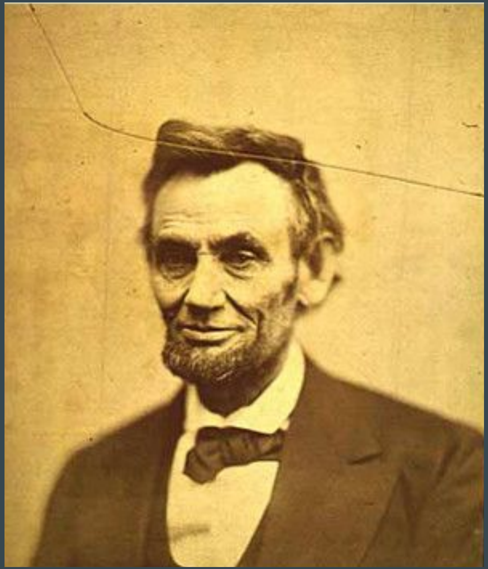
1. Inpainting

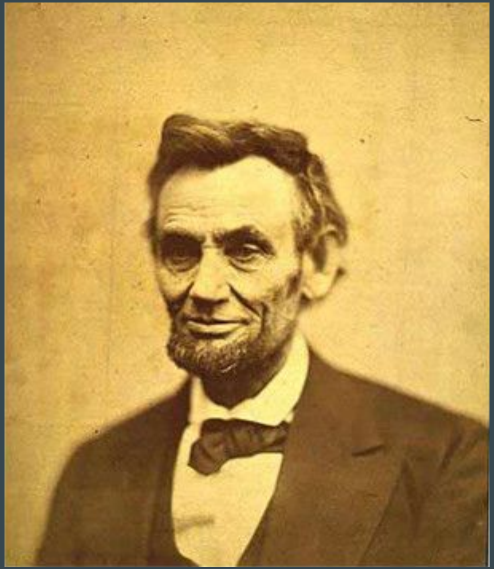
Alex Carrillo
Johnny Nuñez
Guillem Martínez

MSc in Computer Vision

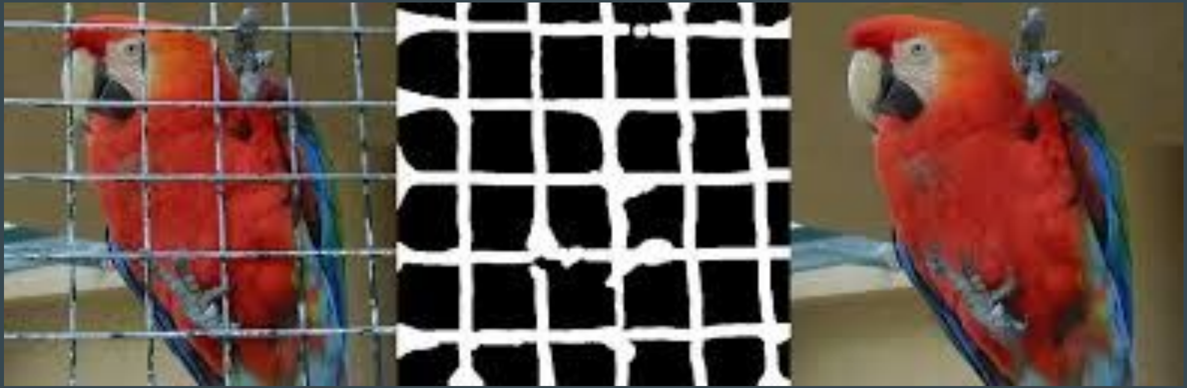
1. Inpainting

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





1. Inpainting



Other applications

- Object Removal
- Image compression

CNN Inpainting

OURS

Classic CV Inpainting

2. Criteria

Define the original images as **V** and the desired inpainted image as **U**.

Define the desired inpaining areas as **B**, and the areas that shouldn't change as **A**.

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

Mathematically:

(1) $V(x, y) = U(x, y)$ at each (x, y) in A
(2) $4V(x, y) - (V(x + 1, y) + V(x - 1, y) + V(x, y - 1) + V(x, y + 1) = 0$ at each (x, y) in B

3. Matematically

Based on two criterias, the output image:

- Should look smoothness (natural).
- Should be similar to the original (difference).

So, we have Energy Functional to optimize:

$$J(u) = \int_{\Omega} \frac{1}{2} |\nabla u|^2 + \frac{1}{2\lambda} |u - f|^2 dx$$

Find the necessary condition for the extremum

$$\longrightarrow \quad \nabla F(\mathbf{u}) = 0$$

3. Mathematically

We can see that the implementation of this algorithm is a laplacian operator that is used for smoothing operations in images. The Laplacian of a two-dimensional function f is an isotropic derivative operator (independent of the direction of the discontinuity in the image) defined by:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f(x,y) = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)$$

So, finally with this laplacian operator we obtain this filter:

0	1	0
1	−4	1
0	1	0

IMPORTANT: We use Laplacian to approximate J(u) with 4 values (4-connectivity)

3. Mathematically

IMPORTANT: We use Laplacian to approximate J(u) with 4 values (4-connectivity)

Alternatively, we could use a similar approach using 8-connectivity, namely Mean filtering, Median filtering or additionally Gaussian filters to tackle different inpainting problems.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

3x3 Average filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

3x3 Mean filter

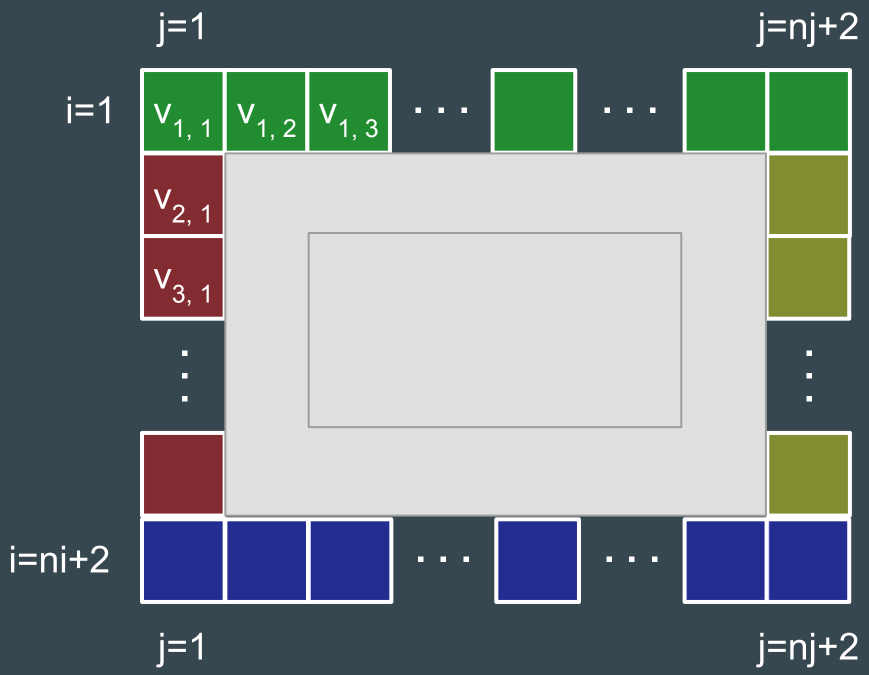
Method

The system consist in a matrix equation to resolve Av=u, which we have the v as original image and u the output image. The dimension we deal with is (m+2) * (n+2) since we employ ghost borders (padding). Image V will have same dimension than U. The matrices of both pictures are flattened into the corresponding vectors b and v.

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

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

Intuition of method implementation (Ghost boundaries)



Copy original boundaries of image:

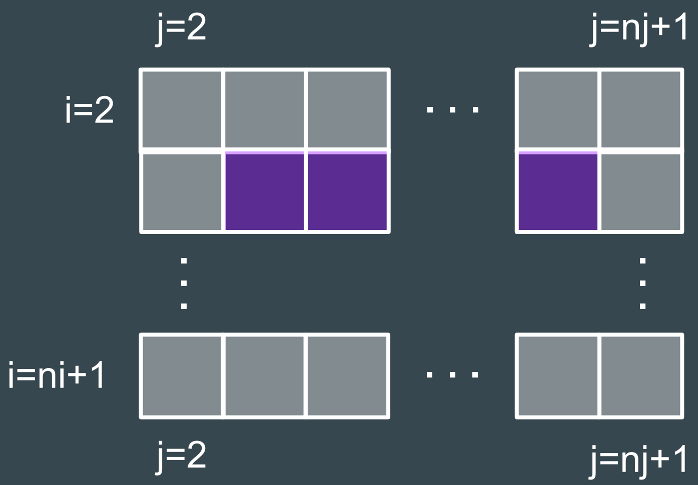
North side	$\rightarrow v_{1,j} - v_{2,j} = 0$	p p + 1
South side	$\rightarrow v_{ni+2,j} - v_{ni+1,j} = 0$	p p - 1
West side	$\rightarrow v_{i,1} - v_{i,2} = 0$	p p + (ni+2)
East side	$\rightarrow v_{i,nj+2} - v_{i,nj+1} = 0$	p p - (ni+2)

$A_{i,j}$ = a_ij(idx) = $\boxed{1}$ $\boxed{-1}$

Vectorial coordinate

This set of equations must be added to the system to be solved.

Intuition of method implementation (Inner points)



Pixels we do not have to inpaint:

Region “A” $\rightarrow$ $v_{i,j} = u_{i,j}$

Vectorial coordinate $\longrightarrow$ $\begin{array}{c|c} & p \\ p & \boxed{1} \end{array}$

If we have to inpaint these pixels:

Region “B” $\rightarrow$ $4v_{i,j} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1} = 0$

$\boxed{4}$ $\boxed{-1}$ $\boxed{-1}$ $\boxed{-1}$ $\boxed{-1}$

Vectorial coordinate $\longrightarrow$	p	p	p	p	p
	p	p - 1	p + 1	p + (ni+2)	p - (ni+2)

Laplacian operator converted to a set of equations and solved as a matrix equation.

Code: Boundaries Case

So following the code, for sparse function, we have the coefficients defined with their positions. We implement to North, South, West and East boundaries code. To do this we need loops to form coefficient matrix.

North

```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p + 1;
a_ij(idx) = -1;
idx=idx+1;

b(p) = 0;
```

South

```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p - 1;
a_ij(idx) = -1;
idx=idx+1;

b(p) = 0;
```

West

```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p + (ni+2);
a_ij(idx) = -1;
idx=idx+1;

b(p) = 0;
```

East

```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p - (ni+2);
a_ij(idx) = -1;
idx=idx+1;

b(p) = 0;
```

Code: InnerPoints

Case Inpainting:

```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 4;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p - (ni+2);
a_ij(idx) = -1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p + (ni+2);
a_ij(idx) = -1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p - 1;
a_ij(idx) = -1;
idx=idx+1;

idx_Ai(idx) = p;
idx_Aj(idx) = p + 1;
a_ij(idx) = -1;
idx=idx+1;
```

Laplacian to approximate J(u) with 5 values (4-connectivity)

Same Value:

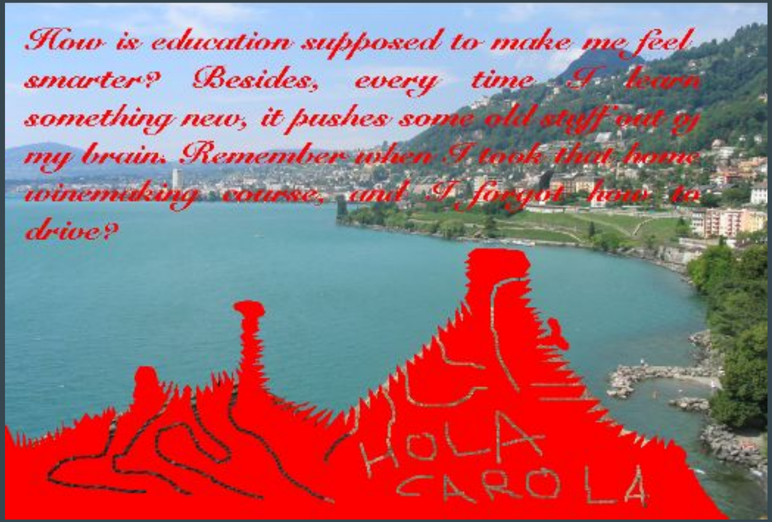
```
idx_Ai(idx) = p;
idx_Aj(idx) = p;
a_ij(idx) = 1;
idx=idx+1;

b(p) = f_ext(i, j);
```

A is a sparse matrix, so for memory requirements we create a sparse matrix

```
A = sparse(idx_Ai, idx_Aj, a_ij, nPixels, nPixels);
```

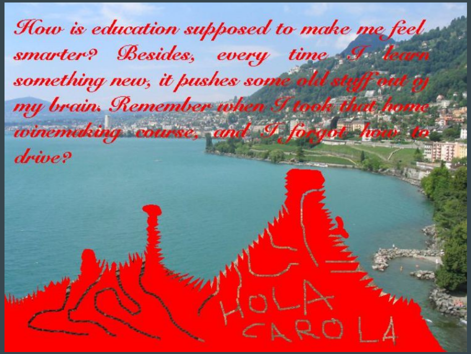
Challenge Code

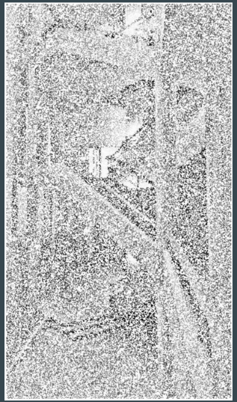
A mask is extracted by examining the intensities at the image's red area, and it is discovered that these intensities are just red (255, 0, 0).

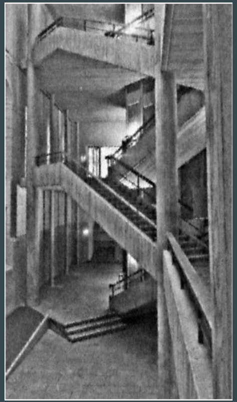



```
mask =  (I_ch1 == 1) & (I_ch2 == 0) & (I_ch3 == 0);
```

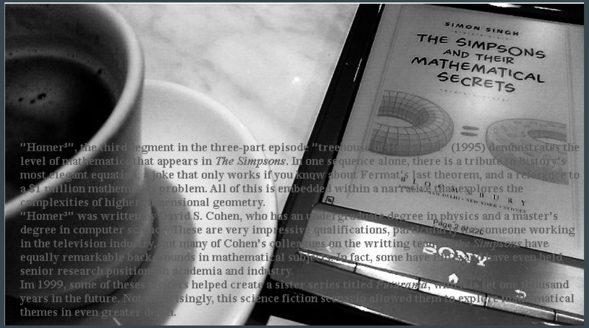
5. Results

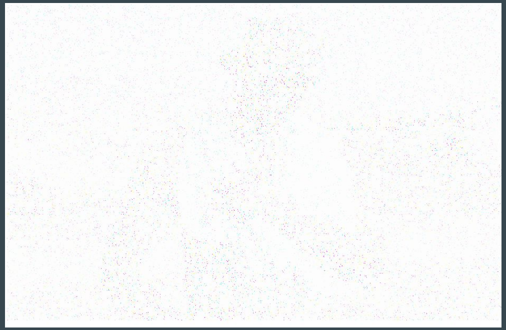

How is education supposed to make me feel
smarter? Besides, every time I learn
something new, it pushes some old stuff out of
my brain. Remember when I took that home
winemaking course, and I forgot how to
drive?
HOLA
CAROLA

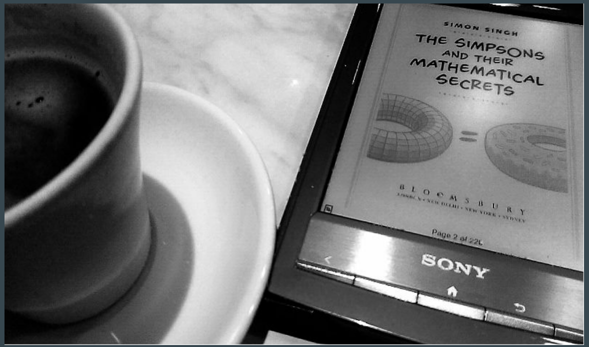




5. Results

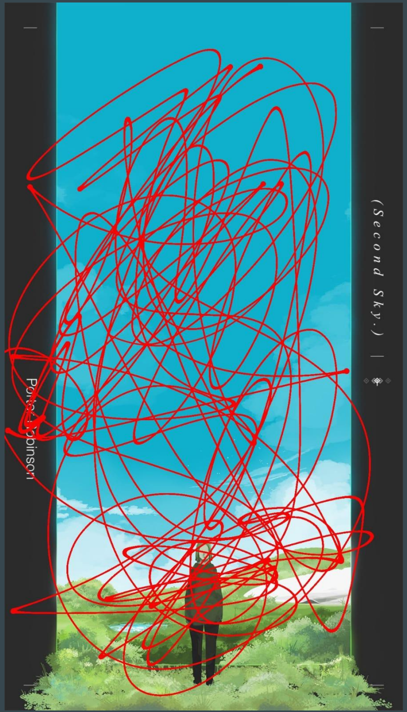

SIMON SINGH
THE SIMPSONS AND THEIR MATHEMATICAL SECRETS
SONY




SIMON SINGH
THE SIMPSONS AND THEIR MATHEMATICAL SECRETS
BLOOMSBURY
Page 2 of 22L
SONY

6. Custom Results

Original Image



Red Threshold Mask

Inpainted Image

6. Custom Results

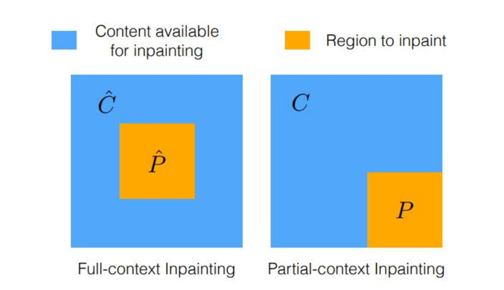
Original Image

Manual Segmentation Mask

Inpainted Image

7. Discussion

- This method has notable results for inpainting small regions
- Not necessary training and Ground Truth data.
- The bigger the region, the bigger the blur in the inpainting region of result. This may occur because:
    - The content available of inpainting is partial, as in C (rather than full-context as in Ĉ)
    - The number of pixels used in the method (4-connect.) is relatively small compared with the area to inpaint
- In some cases, the smooth transition of the laplacian operator is not enough to "hide" a visible region of the image in a optimal way



Thanks

That's it for today :)