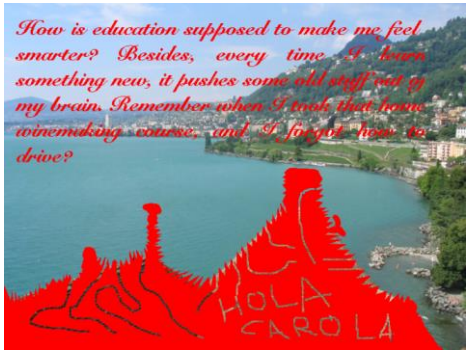
Optimisation in computer vision: Seamless Cloning

Group members: José Manuel López Camuñas, Marcos V. Conde, Alex Martin Martinez

INDEX

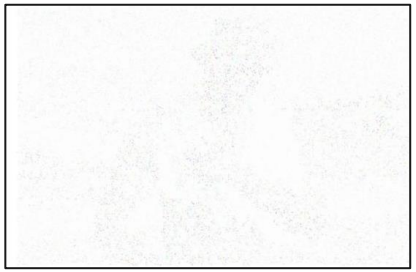
Recap: Week 1

Last week we worked on the Image Inpainting problem, due to the limitations of our algorithm (time-wise), it was not possible for us to generate many result samples. Here we provide high-resolution results:



Before

After



Before

After

Recap: Week 1

A

B

High-resolution RGB images. We compare our method (**A**) with the SOTA work (**B**) by Yu *et.al.* on Generative Image Inpainting: "*Free-Form Image Inpainting with Gated Convolution*" (ICCV 2019 Oral).
Source: https://github.com/JiahuiYu/generative_inpainting

Recap: Week 1

A

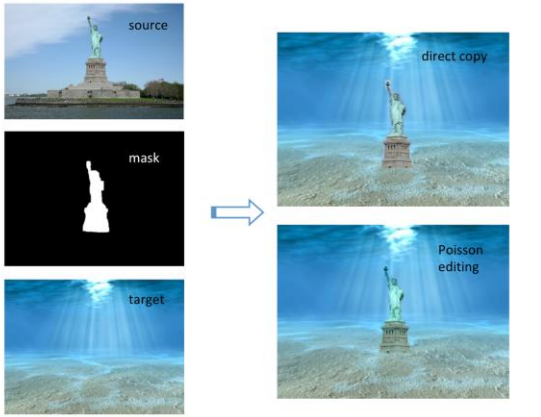
B

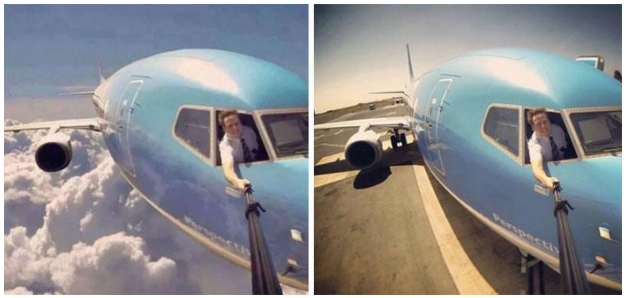
Recap: Week 1

The method struggles with high-resolution images and complex non-uniform textures. However, in comparison with the *state-of-the-art* methods, results are not too bad.

Problem 1: Seamless Cloning

Adding new objects, persons or animals into other images is a usual method to generate scenarios that never occured, a classical Image Manipulation technique. Many fake images are generated on a daily basis, but the methods to make such images have to take into account details to make them look real.

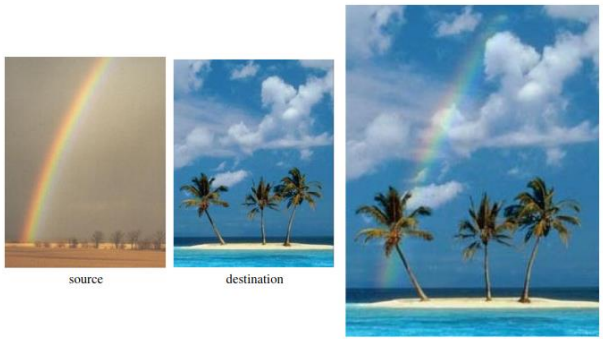




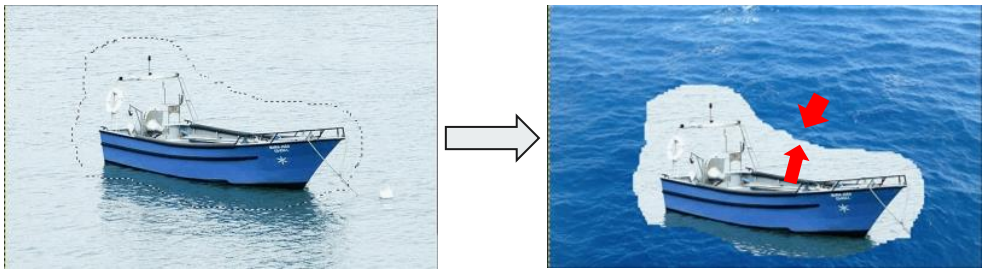
Sample source: https://github.com/PPPW/poisson-image-editing

Problem 1: Seamless Cloning

To generate a realistic credible images, using parts of other images, the main problem is the boundaries. As the **source image** from which the object comes from, and the **destination image**, can have different background colors and textures, the generation of the fake image can not be copy-paste the object into the desired image. We need to correct the contour and boundaries of the object in the destination.



Sample source: Pérez *et.al.* “Poisson Image Editing”



Source image (object “boat”)

Destination image (“sea”)

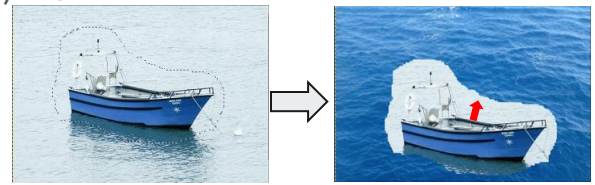
Solution: Import gradients

To generate **target** images that have a smooth transition so they can be realistic, we want to import only the details and structure of the **source** image (boat) into the particular region in the **destination** image (sea) where the object has been placed. To do so, we have to use the gradients of the source image to extract the details. We impose them as the solution of the equation so they appear in the target image.

In the previous week, we concluded the following condition for inpainting areas:

$$4 \cdot H(i,j) - (H(i-1,j)+H(i+1,j)+H(i,j-1)+H(i,j+1)) = 0$$

where H(i,j) was the new image.



Source image Destination

Solution: Import gradients

This week, exploring Poisson Image Editing [1], the new condition in the areas where we want the bring another object in the image will be:

$\nabla H = \nabla B$

$$\Delta f = \Delta g \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}. \quad (10)$$

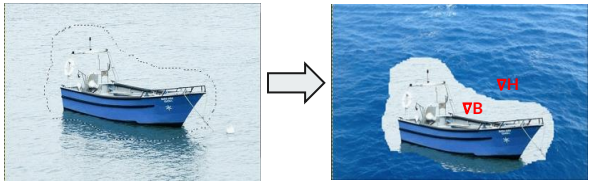
or expressed with finite difference:

$$4 \cdot H(i,j) - (H(i-1,j)+H(i+1,j)+H(i,j-1)+H(i,j+1)) = 4 \cdot B(i,j) - (B(i-1,j)+B(i+1,j)+B(i,j-1)+B(i,j+1))$$

where B(i,j) is the source image from which the object has been extracted, and H is the new resultant image.

Summing up all the conditions to achieve a smooth transition and bring the details from the source image B to the destination image A:

1. H(i,j)= A(i,j) in the boundaries of B
2. ∇H = ∇B in region B
3. The ghost boundary conditions form the previous week.



Source image Destination

[1] Patrick Perez *et.al.* ¨Poisson Image Editing“, ACM 2003.

Implementation Details of Import gradients

To move the details from the source image into the destination image, first the gradient of the source have to be calculated. With the functions “*sol_DiBwd*”, “*sol_DiFwd*”, “*sol Dj Bwd*” *and* “*sol_DjFwd*”, we calculate the backward (Bwd) and forward (Fwd) finite difference for each direction (Di, Dj). Below we show the code for calculating the finite differences in the i-direction:

```
function [ result ] = DiBwd( I, hi )
% Compute the backward finite differences with respect to the
% i coordinate only for the 2:end rows. The first row is not replaced
    if (~exist('hi', 'var'))
        hi=1;
    end;

    result=I;
    %Begin To Complete 10
    result(2:end, :) = (I(1:end-1, :)-I(2:end, :))./hi; %result(2:end, :) =??
    %End To Complete 10
end
```

sol_DiBwd function

```
function [ result ] = DiFwd( I, hi )
% Compute the Forward finite differences with respect to the
% i coordinate only for the 1:end-1 rows. The last row is not replaced

    if (~exist('hi', 'var'))
        hi=1;
    end;

    result = I;
    %Begin To Complete 8
    result(1:end-1, :) = (I(2:end, :)-I(1:end-1, :))./hi; %result(1:end-1, :)=??
    %End To Complete 8
end
```

sol_DiFwd function

Implementation Details of Import gradients

After the finite difference of the gradients is calculated for all the directions (green) as explained before, those gradients are then added to have the gradient of the source image (red):

```
for nC = 1: nChannels

    %TO DO: COMPLETE the ??
    drivingGrad_i = sol_DiBwd(src,param.hi) + sol_DiFwd(src,param.hi);
    drivingGrad_j = sol_DjBwd(src,param.hj) + sol_DjFwd(src,param.hj);

    driving_on_src = drivingGrad_j + drivingGrad_i;

    driving_on_dst = zeros(size(src(:,:,1)));
    driving_on_dst(mask_dst(:)) = driving_on_src(mask_src(:));

    param.driving = driving_on_dst;

    dst1(:,:,nC) = sol_Poisson_Equation_Axb(dst1(:,:,nC), mask_dst,  param);
end
```

⇨ solve the system of linear equations

Implementation Details of Import gradients

The variable *param.driving* is passed into the "*sol_Poisson_Equation_AxB*" function so that it uses the gradients of the source image instead of 0 when solving the system of linear equations:

```
if (isfield(param,'driving'))
    b(p) = -param.driving(i,j);
else
    b(p) = 0;
end
```

The minus sign in the second line is due to the fact that summing the finite differences results in this:

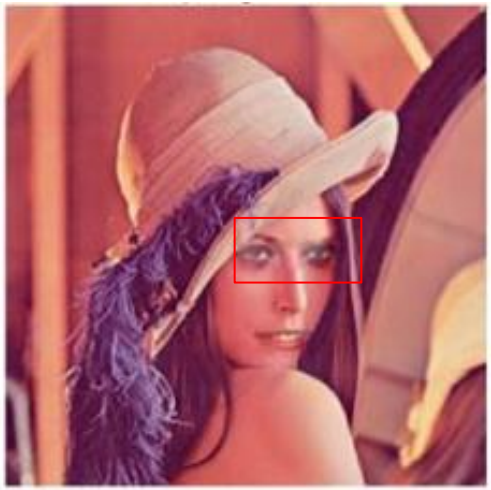
$$B(i-1,j) - B(i,j) + B(i+1,j) - B(i,j) + B(i,j-1) - B(i,j) + B(i,j+1) - B(i,j)$$

but the gradient it's exactly the negative of this expression : $4 \cdot B(i,j) - (B(i-1,j)+B(i+1,j)+B(i,j-1)+B(i,j+1))$

Import gradients Results

Source Image (B)

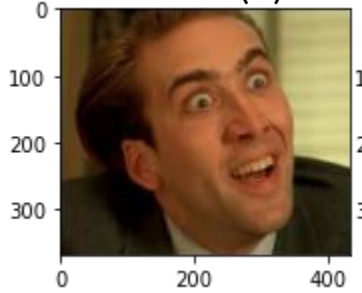
Destination Image (A)



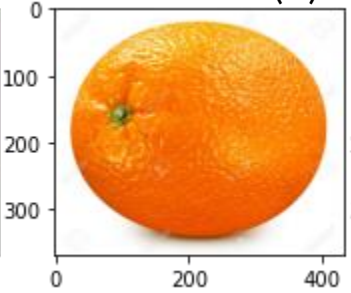
Target Image (H). We moved the eyes and the mouth from B to A.

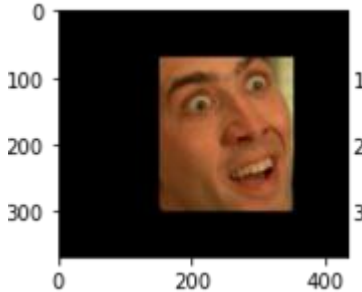
Import gradients Results

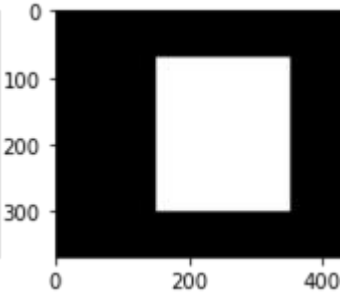
Source (B)



Destination (A)







Target Image (H). We moved the face from B to A.

Import gradients Results

Source (B)

Destination (A)

Target Image (H). We moved the crocodile from B to A.

Problem 2 and Mixed gradients

As we can see in the generated images from the “import gradient” method, the **details from the destination image are lost** where the source image has been imported, this is because we are imposing the gradient of the source image in that region. Although the object is represented, it creates a rare effect on the borders.

There are situations where it is desirable to **combine properties** from the source and destination images. “**Mixed gradients**” [1], instead of using only the gradient of the source image, selects the bigger gradient between the source and the destination, to use it as the solution to the equation we want to solve. This method accomplishes then to have the details from both images the source and the destination.

These condition is expressed mathematically by:

1. $\nabla H = \nabla A$ if $|\nabla A| > |\nabla B|$
2. $\nabla H = \nabla B$ otherwise

$$\text{for all } \mathbf{x} \in \Omega,\ \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})|, \\ \nabla g(\mathbf{x}) & \text{otherwise.} \end{cases} \quad (12)$$

The other conditions remains the same.

[1] Patrick Perez *et.al.* ¨Poisson Image Editing“, ACM 2003.

Implementation details of Mixed gradients

In the matlab code for the i**mport gradient method** the following lines were added:

```
if (isfield(param,'driving_dst'))
    if (strcmp(param.method,'import'))
        b(p) = -param.driving_src(i,j);
    else

        if abs(param.fwdi_src(i,j)) < abs(param.fwdi_dst(i,j))
            fwdi = param.fwdi_dst(i,j);
        else
            fwdi = param.fwdi_src(i,j);
        end

        if abs(param.fwdj_src(i,j)) < abs(param.fwdj_dst(i,j))
            fwdj = param.fwdj_dst(i,j);
        else
            fwdj = param.fwdj_src(i,j);
        end

        if abs(param.bwdi_src(i,j)) < abs(param.bwdi_dst(i,j))
            bwdi = param.bwdi_dst(i,j);
        else
            bwdi = param.bwdi_src(i,j);
        end

        if abs(param.bwdj_src(i,j)) < abs(param.bwdj_dst(i,j))
            bwdj = param.bwdj_dst(i,j);
        else
            bwdj = param.bwdj_src(i,j);
        end

        b(p) = -(fwdi+fwdj+bwdj+bwdi);
    end
```

If the *param.method* variable was equal to import the method used was the “import gradient”, in any other case the “mixed gradients” method was used. The finite differences of **each direction (**4 in total considering forward and backwards in i,j directions**)** were compared and the greater one’s **absolute** was used as the value of the finite difference in that direction to then calculate the new value of b(p) (green).

In the paper [1], ***f**** is the destination (“dst” in our code) and ***g*** the source image (“src” in our code).

$$\text{for all } \mathbf{x} \in \Omega,\ \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})|, \\ \nabla g(\mathbf{x}) & \text{otherwise.} \end{cases} \quad (12)$$

[1] Patrick Perez *et.al.* ¨Poisson Image Editing“, ACM 2003.

Implementation details of Mixed gradients

For the gradients to be compared the *start.m* file was also modified in the following way to calculate the gradient of both images to then compare them. Note that before we only calculated gradients for the source (red), now also for the destination (green).

This variables were then used as an input into the ***sol_Poisson_Equation_AxB.m***

```
fwdi_src = sol_DiFwd(src(:,:,nC),param.hi);
fwdj_src = sol_DjFwd(src(:,:,nC),param.hj);
bwdi_src = sol_DiBwd(src(:,:,nC),param.hi);
bwdj_src = sol_DjBwd(src(:,:,nC),param.hj);
```

```
fwdi_dst = sol_DiFwd(dst(:,:,nC),param.hi);
fwdj_dst = sol_DjFwd(dst(:,:,nC),param.hj);
bwdi_dst = sol_DiBwd(dst(:,:,nC),param.hi);
bwdj_dst = sol_DjBwd(dst(:,:,nC),param.hj);
```

```
param.fwdi_src = fwdi_src;
param.fwdj_src = fwdj_src;
param.bwdi_src = bwdi_src;
param.bwdj_src = bwdj_src;

param.fwdi_dst = fwdi_dst;
param.fwdj_dst = fwdj_dst;
param.bwdi_dst = bwdi_dst;
param.bwdj_dst = bwdj_dst;
```

Mixed gradients Results

The same masks and images were used to test the mixed gradients method:

Source Image (B)

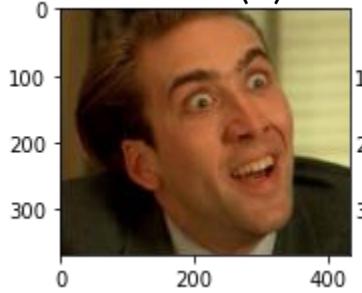
Destination Image (A)

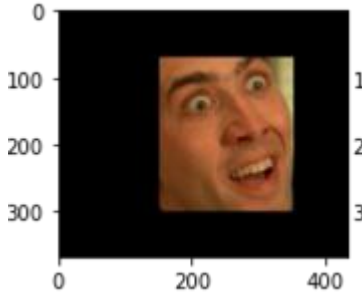
Mixed Gradients

Import Gradients

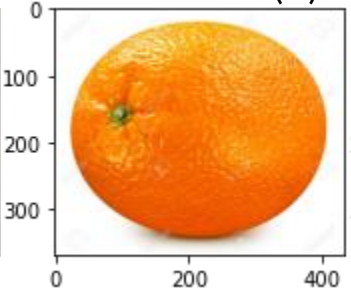
Mixed gradients Results

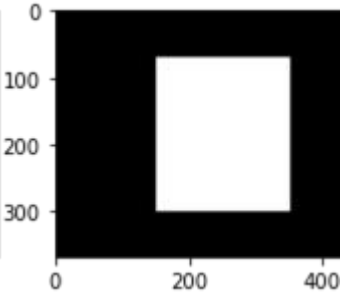
Source (B)

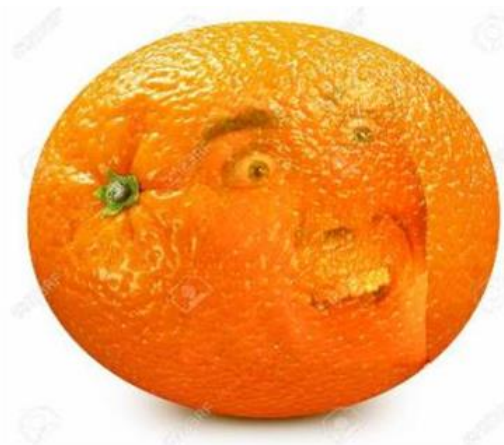




Destination (A)







Mixed Gradients

Import Gradients

Mixed gradients Results

Source (B)

Destination (A)

Mixed Gradients

Import Gradients

In this example, because of the complex textures, shadows and contours in the water, the method tends to choose the biggest gradient (water), and ends up propagating/representing only water.

Discussion and conclusions

- Using the **import gradient** method we can move objects (textures, shapes, details, etc.) from a source image to a destination.
- The method's inference is "fast" using high resolution images, and qualitative results are good.
- The **main drawback** of this method is that the boundaries of the mask used to determine the object to be imported from the source to the destination can be easily detected as you can observe a strange effect at that location. Moreover the colors from the source are replaced partially with the ones from the destination.
- The **Mixed gradients** method tries to solve these problems in situations like: add objects with holes, or partially transparent ones, on top of a textured or cluttered background.
- Mixed gradients solves the problem in the boundaries of the imported object.
- However, it **does not work in all scenarios**, as this method uses the maximum gradient from either the destination or source, if the details in the destination where you want to bring the object are greater than the details from the object the gradients from the destination will be selected and the resulting image will not have the desired outcome of cloning (e.g. our crocodile sample).