MCV-M2:
Image Inpainting and Completion

Team 7 - Joan F. Serracant, Martí Cobos

Week 2: Poisson editing

Index

- Task 1: Seamless cloning techniques from Patrick Perez paper
- Task 2: Implementation of importing gradients method
- Task 3: Importing gradients method results
- Task 4: Results on our own images
- Task 5: Implementation of mixing gradients method
- Task 6: Mixing gradients results
- Conclusions

Task 1: Seamless cloning techniques from Patrick Perez paper

Patrick’s paper describes a set of tools to achieve seamless importation of source image regions into a destination region.

Image interpolation is used by means of a guidance vector field **v**

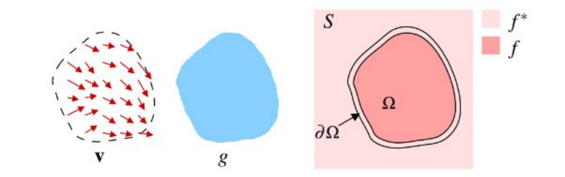


Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

Criteria to define the problem:

1. Transition from src to dst images must be smooth
2. Details (gradient) of dst image must be kept

Mathematically, criteria can be expressed as the minimization problem

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

Criteria 2   Criteria 1

which can be solved by the Poisson equation with Dirichlet boundary conditions

$$\Delta f = \operatorname{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

Source: Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. ACM Trans. Graph. 22, 3 (July 2003), 313-318

Task 1: Seamless cloning techniques from Patrick Perez paper

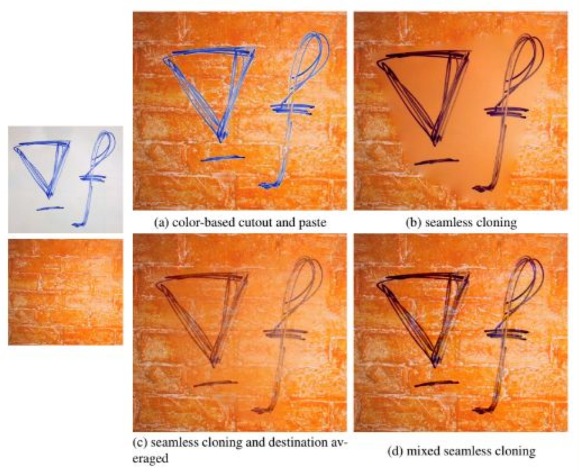
Two techniques are described to perform Seamless cloning:

Importing gradients: where the guidance field **v** is the gradient field taken from the src image **g**

$$\mathbf{v} = \nabla g,$$ With the new problem solution $$\Delta f = \Delta g \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}.$$

Mixing gradients: where we want to take texture from src image **g** but we also want to keep some texture of the dst image **f** and the guidance field **v** is defined as

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



(a) color-based cutout and paste

(b) seamless cloning

(c) seamless cloning and destination averaged

(d) mixed seamless cloning

Source: Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. ACM Trans. Graph. 22, 3 (July 2003), 313-318

Task 2: Implementation of importing gradients method

As we said, the solution to $\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2$ with $f|_{\partial\Omega} = f^*|_{\partial\Omega}$, is $\Delta f = \operatorname{div}\mathbf{v}$ over Ω, with $f|_{\partial\Omega} = f^*|_{\partial\Omega}$,

Remember last week's inpainting problem was $\begin{cases} \underset{u \epsilon W^{1,2}(\Omega)}{\arg\min} \int_D |\nabla u(x)|^2 \, dx \\ u|_{\delta D} = f \end{cases}$

Last week's u = f in Patrick's paper

with solution $\begin{cases} \Delta u = 0 \text{ in } D \, (1) \\ u = f \text{ in } \partial D \, (2) \end{cases}$ and we proved that $\Delta u = u_{i+1,j} - u_{i-1,j} - u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0$

If we do the same for the importing gradients problem, $\Delta f = \operatorname{div}\mathbf{v}$ over Ω, with $f|_{\partial\Omega} = f^*|_{\partial\Omega}$, becomes

$$\Delta f = f_{i+1,j} - f_{i-1,j} - f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = div\,(v) = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} = \frac{\partial \nabla g}{\partial x} + \frac{\partial \nabla g}{\partial y}$$

The solution is then quite similar to what we implemented last week.

Task 2: Implementation of importing gradients method (Matlab code)

Our implementation in Matlab code consists in:

- In start.m file we compute $\frac{\partial \nabla g}{\partial x} + \frac{\partial \nabla g}{\partial y}$ of the masked src image and place it on the masked region of dst image

Computing $\frac{\partial \nabla g}{\partial x} + \frac{\partial \nabla g}{\partial y}$

```
drivingGrad_i = sol_DiFwd(src(:,:,nC),param.hi) - sol_DiBwd(src(:,:,nC), param.hi);
drivingGrad_j = sol_DjFwd(src(:,:,nC),param.hj) - sol_DjBwd(src(:,:,nC), param.hj);
driving_on_src = drivingGrad_i + drivingGrad_j;
```

Positioning values in the region of interest on dst image

```
driving_on_dst = zeros(size(src(:,:,1)));
driving_on_dst(mask_dst(:)) = driving_on_src(mask_src(:));
param.driving = driving_on_dst;
```

Task 2: Implementation of importing gradients method (Matlab code)

- Changed G7_Laplace_Equation_Axb.m to G7_Poisson_Equation_Axb.m by changing

G7_Laplace_Equation_Axb.m

```
idx_Ai(idx) = p; idx_Aj(idx) = p;        a_ij(idx) = -4; 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;
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;

b(p) = 0;
```

G7_Poisson_Equation_Axb.m

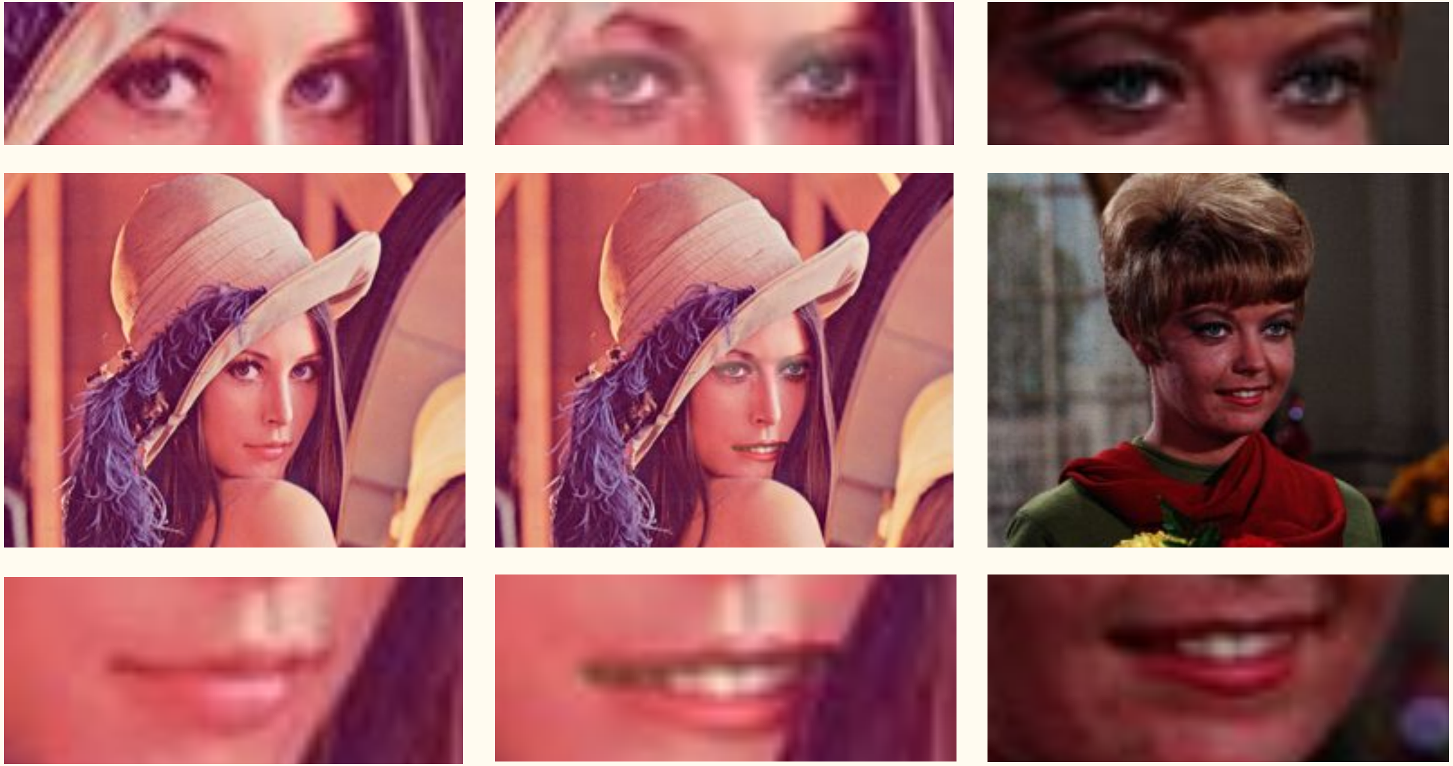
```
idx_Ai(idx) = p; idx_Aj(idx) = p;        a_ij(idx) = -4; idx=idx+1;
idx_Ai(idx) = p; idx_Aj(idx) = p+1;      a_ij(idx) = 1;  idx=idx+1;
idx_Ai(idx) = p; idx_Aj(idx) = p-1;      a_ij(idx) = 1;  idx=idx+1;
idx_Ai(idx) = p; idx_Aj(idx) = p+ni+2;   a_ij(idx) = 1;  idx=idx+1;
idx_Ai(idx) = p; idx_Aj(idx) = p-ni-2;   a_ij(idx) = 1;  idx=idx+1;

if (isfield(param, 'driving'))
     b(p) = driving_ext(i-1,j-1);
else
     b(p) = 0;
end
```

i-1 and j-1 taking into account that i and j variables iterate through extended images with ghost boundaries

Task 3: Importing gradients method results

Gradient (shape and texture) is kept from right image but colors are adapted to the ones of the left image

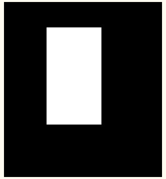


Task 4: Results on our own images

Test Image: Snowboarding on Mt.Everest

Destination image:
Mt.Everest

Source image: Team Member Snowboarding

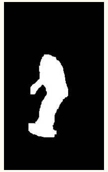


Rectangular Mask

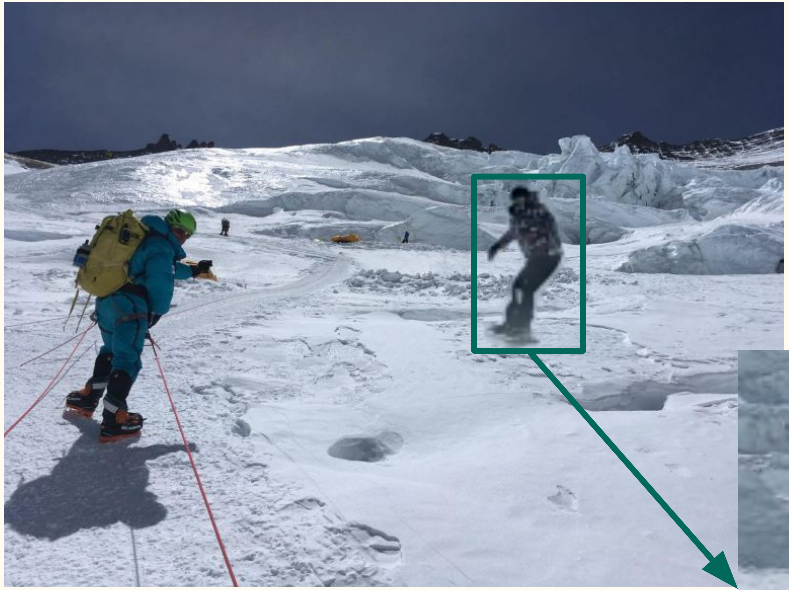
- Color is transferred, keeping gradient from source image.
- High frequency information on the background of destination image is lost.

Task 4: Results on our own images

Improving results by refining the mask:



Refined Mask



Rectangular mask result:

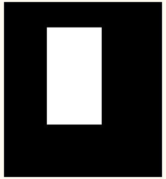
- Background information is saved with refined mask.
- Mask difficult to generate, not optimal solution

Task 4: Results on our own images

Test Image: Banksy at CVC

Destination image:
Banksy painting

Source image:
CVC building



Rectangular Mask

- Color is transferred correctly.
- Building tiles from destination image are lost, image doesn't appear realistic. This issue will be solved with mixing gradients implementation

Task 5: Implementation of mixing gradients method

In order to implement the mixing technique for seamless cloning, the guidance field must be changed as follows:

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

Where f* is the destination image.

Once guidance field has been computed,the following equation must be solved as with importing gradients technique:

$$\Delta f = \operatorname{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

$$\Delta f = f_{i+1,j} - f_{i-1,j} - f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = div\,(v) = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}$$

Task 5: Implementation of mixing gradients method

Our implementation in Matlab code consists in:

- In start_mixed_gradients.m file guidance field is firstly computed as:

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

Source Gradient:

```
sourceGrad_i = sol_DiFwd( src(:,:,nC), param.hi );
sourceGrad_j = sol_DjFwd( src(:,:,nC), param.hj );
```

Destination Gradient:

```
dstGrad_i = sol_DiFwd( dst(:,:,nC), param.hi );
dstGrad_j = sol_DjFwd( dst(:,:,nC), param.hj );
```

Initialize driving vector as source gradient:

```
drivingGrad_i = zeros(size(dst(:,:,1)));
drivingGrad_i(mask_dst(:)) = sourceGrad_i(mask_src(:));
drivingGrad_j = zeros(size(dst(:,:,1)));
drivingGrad_j(mask_dst(:)) = sourceGrad_j(mask_src(:));
```

```
drivingDestGrad_i = zeros(size(dst(:,:,1)));
drivingDestGrad_j = zeros(size(dst(:,:,1)));
drivingDestGrad_i(mask_dst(:)) = dstGrad_i(mask_dst(:));
drivingDestGrad_j(mask_dst(:)) = dstGrad_j(mask_dst(:));
```

Compute guidance field applying above condition:

```
drivingGrad_i (abs(drivingDestGrad_i)>abs(drivingGrad_i)) = drivingDestGrad_i(abs(drivingDestGrad_i)>abs(drivingGrad_i));
drivingGrad_j (abs(drivingDestGrad_j)>abs(drivingGrad_j)) = drivingDestGrad_j(abs(drivingDestGrad_j)>abs(drivingGrad_j));
```

- Afterwards the divergence of the guidance field is computed:

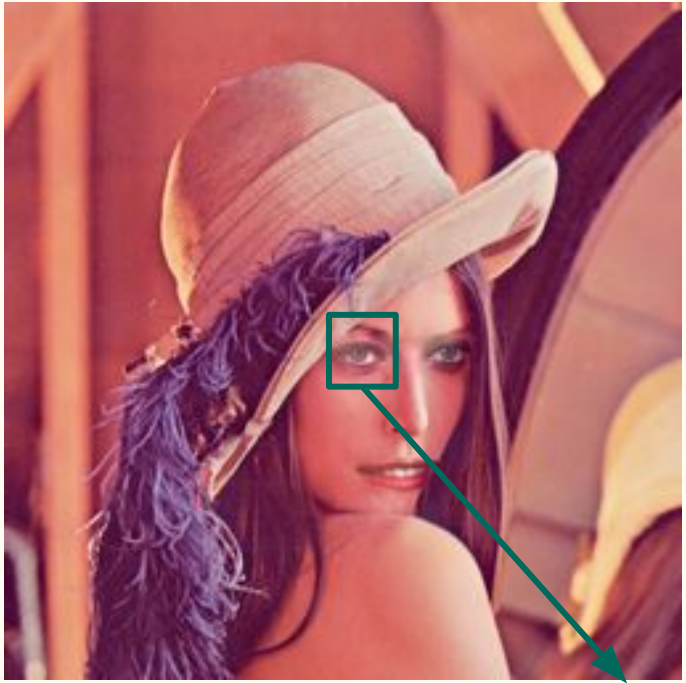
$$div\,(v) = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}$$

Compute divergence of guidance field:

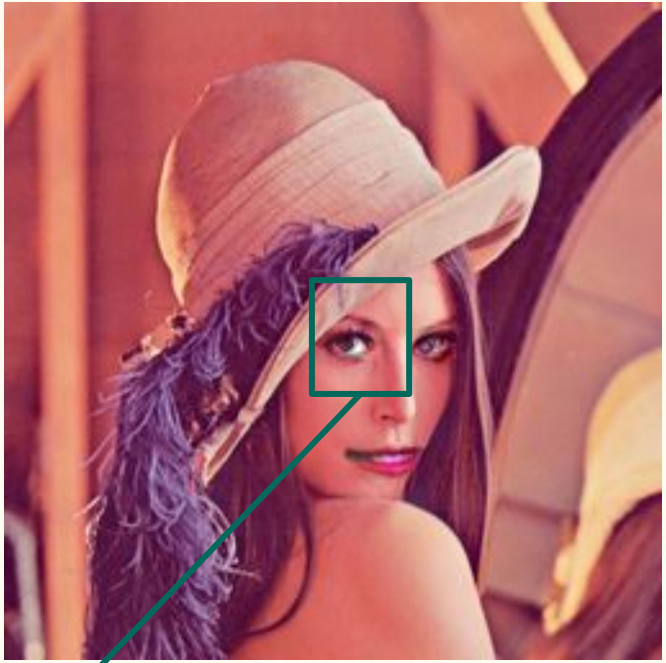
```
driving_on_dst = +sol_DiBwd( drivingGrad_i(:,:), param.hi ) + sol_DjBwd( drivingGrad_j(:,:), param.hj );
param.driving = driving_on_dst;
```

Task 6: Mixing gradients results

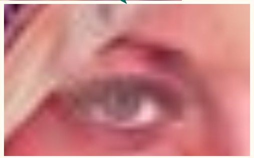
Importing gradients:

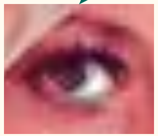


Mixing gradients:



Part of the hat is blurred with **Importing gradients** algorithm





Mixing gradient result:

- Image appear more realistic
- The hat region is preserved

Task 6: Mixing gradients results

Rectangular masks with mixing gradients algorithms:

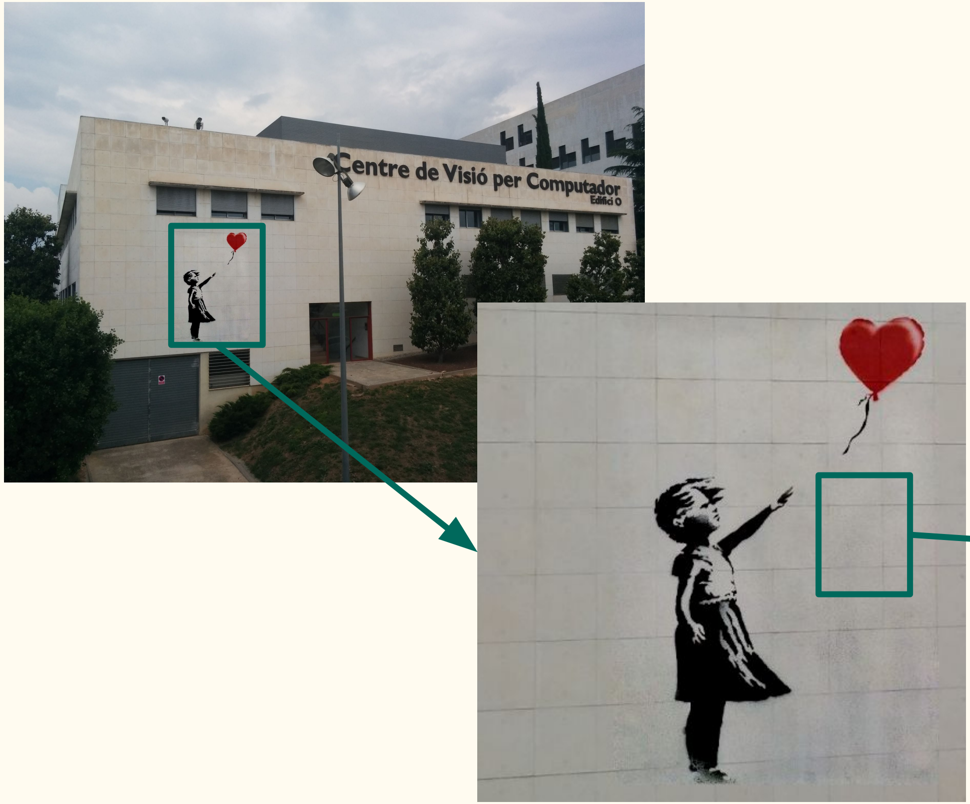
Refined mask with mixing gradients algorithms:

- Both masks produce similar results. The background is preserved with the rectangular mask
- Snowboarder from source image is mixed with the background (producing a ghostly appearance) due to the fact the gradient of the destination image is high compared to the source image
- This image set is not solved properly by the Poisson editing algorithm.

Task 6: Mixing gradients results

Importing gradients result:

Mixing gradients result:



- Building tiles from destination image have been preserved.
- Result image seems realistic with better results than importing gradients algorithm.

Conclusions

- Poisson editing problem implemented through optimization with great results. Both importing gradients and mixing gradients algorithms have been implemented. Test image has been edited successfully.
- Additional images have been tested. With the first one (Snowboarding on Mt.Everest) we did not obtain good results, the destination image has strong gradients which were either removed or produced a ghostly appearance on the snowboarder. Different mask shapes were tested with poor results.
- The second image (Banksy at CVC) was edited successfully with mixing gradients algorithm.
- Mixing gradients improves solution over the importing gradients algorithm as it preserves information from destination image. Result image appears more realistic