Image Processing Assignment 3 - Poisson Image Editing

Julien Siems (16119007)

Attempted the following exercises

- Task 1 100
- Task 2 100
- Task 3 100
- Task 4 100
- Task 5 100
- Task 6 100

Task 1

Part a.)

Problem statement

The paper proposes a solution to seamless editing of image regions. It solves the following problems:

a. Seamless cloning of regions from a source image to a target image.

b. Seamless change of properties of an image inside a region.

Key ideas

One of the key ideas of this paper is the statement of the minimization problem. (Equation 3)

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{1}$$

whose solution should satisfy the following Poisson equation.

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2}$$

The left part of 2 means that the second order variations of the domain Ω must agree with the divergence of the guidance vector field $\mathbf{v}$. As described in the introduction of the paper, second order variations are more significant perceptually, than first order variations. Thus it is important for them to agree to make it look realistic.

The right part means that the function f in domain Ω should agree on their boundary with the surrounding function f^*. This condition ensures a smooth transition between Ω and the surrounding domain.

Depending on the effect that is supposed to be achieved one can choose different vector fields. This makes this technique very powerful.

Applications

- Concealment: Can be achieved by picking as source image a region within the image that is similar to the target region. As guidance vector field we could choose either the laplacian of the source region or we could mix the gradients depending on which of them has a higher value. (Figure 2 in the paper)
- Insertion: One picks a source region in one image and then inserts into a target image. This can be solved very similarly to the Concealment application. When objects with holes are inserted than using the mixed gradients technique which favours the higher gradent of either the source or target image can result in a better output.
- Selection editing:

  - Texture flattening: Achieved using a sieve that leaves only the most prominent features(For example most important edges etc.). A possible guidance vector field is given in equation 15 in the paper.
  - Local illumination changes: Used for smooth dynamic range changes. Defined on the log domain to avoid arithmetic underflow. A possible guidance vector field is given in equation 16. To increase the dynamic range these changes are made in the log domain.
  - Local color changes: One possible application is turning everything but a region in an image monochrome. This is solved by setting f^* to be the gray channel of itself. Then a region is picked in the original color f^*, which is inserted into the monochrome version using the approach described above for each color channel.
  - Seamless tiling: Acchieved by ensuring that the north south and east west borders agree.

Part b.)

What I like in the paper

- The changes look very complex and would also be time consuming to do manually. Still, the technique is fast.
- I like the fact that the domains can have an arbitrary shape.
- The selections when cloning or when using selective editing don't have to be extremely accurate. They can be rather loose and still result in very good results.
- Only the guidance vector field needs to be adjusted for different applications. It can be easy to illustrate and therefore to choose an appropriate vector field for the problem at hand.
- The technique is very neat. It makes sense that the border values have to agree and that the domain is guided by the most prominent features of the source.
- Technique can be applied to each color component independently.

What I don't like about the paper

- Because the approach only takes into the account the laplacian of each color channel some information about the actual color of the source image is lost. This artefact can be seen in Figure 3. While the insertion of the bear is very seamless the bear suddenly has more of a bright blue color which doesn't match it's normal color. An improved version of the algorithm found was developed that takes the color into account.[1]
- If the gradient around the border of the domain is very strong then the transition will not look seamless.

Task 2

Explanation and code

Equation 2 from the paper assumes no prior knowledge of any guidance vector field inside the domain. The laplacian inside the region is supposed to be zero. However, the boundary values of the unknown function f and the target function f^* have to agree.

A doesn't change in the LSE. b becomes:

$$b_{p_i} = \sum_{q \in N_{p_i} \cap \partial\Omega} f_q^* \tag{3}$$

Code listing:

[1] Dizdarolu and kiba, An improved method for color image editing, EURASIP Journal on Advances in Signal Processing 2011, 2011:98

```matlab
close all

I = double(rgb2gray(imread('tom_hanks.jpg')))/255;
I_orig = I;

% region specification
fig = figure;
[bw, xi, yi] = roipoly(I);
bwi = 1 - bw;

[bw_row, bw_col, ~] = find(bw);
domain_index = sub2ind(size(I), bw_row, bw_col);

%% building b
% boundary pixels
fstar = I - bw;
fstar(fstar < 0) = 0;
filter = [0 1 0; 1 0 1; 0 1 0];
sum_fstar_boundary = imfilter(fstar, filter, 'replicate');
fstar_val_boundary = sum_fstar_boundary(domain_index);

b = fstar_val_boundary;

%% building A
% Ugly for loop for adjacency
dim = size(bw_col, 1);
A = zeros(dim);
[w, l, c] = size(I);

coor = [bw_row, bw_col];

for x = 1:w
    for y = 1:l
        if (bw(x, y) == 1)
            i_index = find(ismember(coor,[x,y],'rows'));
            if(bw(x-1, y) == 1)
                j_index = find(ismember(coor,[x-1, y],'rows'));
                A(i_index, j_index) = -1;
            end
            if(bw(x, y-1) == 1)
                j_index = find(ismember(coor,[x, y-1],'rows'));
                A(i_index, j_index) = -1;
            end
            if(bw(x+1, y) == 1)
                j_index = find(ismember(coor,[x+1, y],'rows'));
                A(i_index, j_index) = -1;
            end
            if(bw(x, y+1) == 1)
                j_index = find(ismember(coor,[x, y+1],'rows'));
                A(i_index, j_index) = -1;
            end
        end
    end
end

%% Solving the LSE
A = A + diag(ones(1,dim)*4);
x = sparse(A)\b;


%% RESULT
I(domain_index) = x;
figure
subplot(1,2,1)
imagesc(I_orig)
colormap gray;
axis image
title('Original image')
subplot(1,2,2)
imagesc(I)
colormap gray;
axis image
```

```
hold on;
plot(xi, yi)
title('Edited image')
```

Results

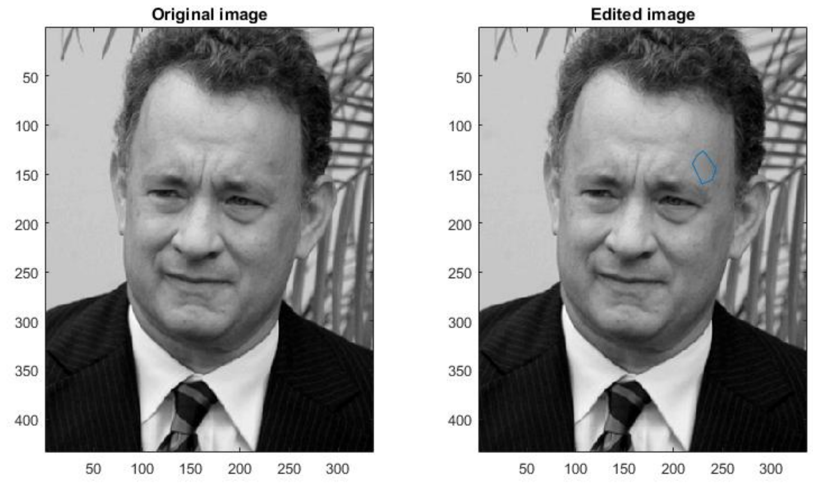


Figure 1: Interpolation of relatively small, smooth region. Tom Hanks image from https://upload.wikimedia.org/wikipedia/commons/9/98/Tom_Hanks_face.jpg

As can be seen in figure 1 small, smooth regions can work very well without a guidance vector field. When zooming out however, it becomes apparent that the region is noticeably too smooth compared to the surrounding skin, since a zero laplacian is assumed inside the domain.

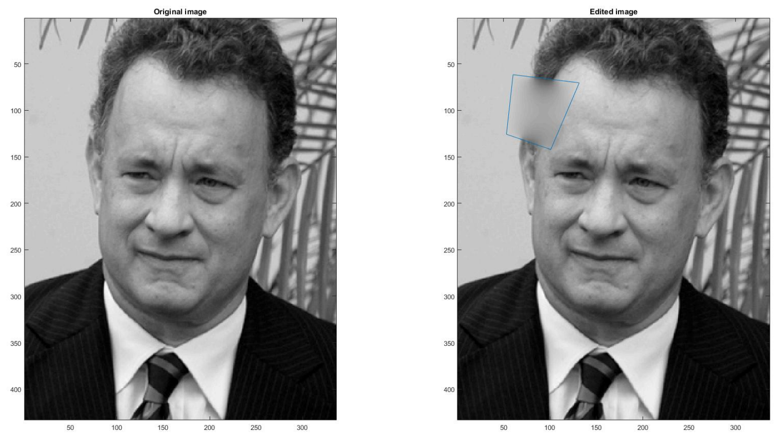


Figure 2: Interpolation of bigger region with a lot of edges.

In figure 2 the true downside of not using a guidance vector field can be seen. Close to the boundary of the region the interpolation is still realistic but in the middle of the region it seems very blurred. This artifact was mentioned in the paper.

What can also be seen is that the bigger the region the less constrained the the interpolation becomes towards the middle of the region.

Task 3 and 4

The only change we need to make to enable seamless cloning is to add the guidance vector field. Therefore b becomes

$$b_{p_i} = \sum_{q \in N_{p_i} \cap \partial\Omega} f_q^* + \sum_{q \in N_{p_i}} v_{p_i q} \tag{4}$$

Intuitively, the guidance vector field gives the domain the contours of the image to be pasted into the image.

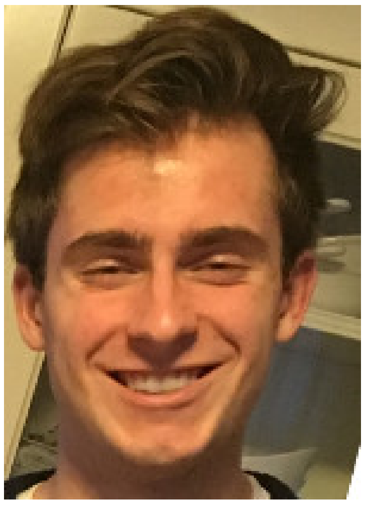
In the case of importing gradients this means that only the source gradients are considered. Accordingly:

$$\text{for all } \langle p, q \rangle, v_{pq} = g_p - g_q \tag{5}$$

In the mixed gradients approach always the stronger one of the gradients of either the source gradient or the target gradient is chosen. That way the structure of the target image in the domain is not completely lost. Accordingly:

$$\text{for all } x \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})| \\ \nabla g(\mathbf{x} & \text{otherwise} \end{cases} \tag{6}$$

The implementation can be found in poissonSolver.m To activate mixed gradients put the constant to 1 otherwise to 0. To start the image editing please refer to ex4.m.


(a)

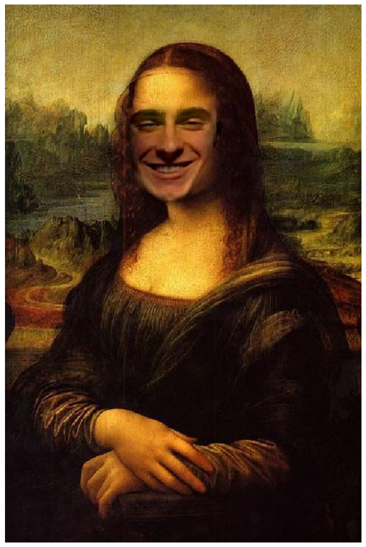
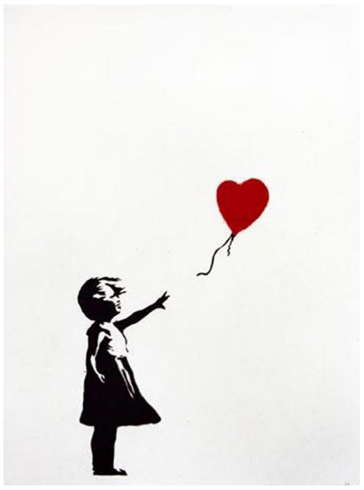
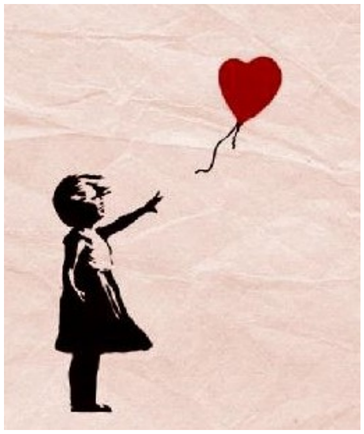

(b)

Figure 3: Example of importing gradients. Mixed Gradients wouldn't be a good idea in this case, because then both faces would be visible. However, the area around the face looks a lot smoother since it lacks the general noise of painting. This could be overcome by allowing mixed gradients where the gradients of the target image are thresholded to only allow the small gradients.



(a) Banksy: Balloon girl https://www.flickr.com/photos/beglen/151659424

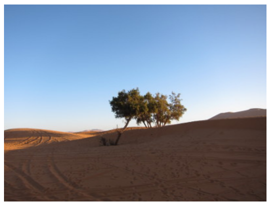


(b) Paper reference https://cdn.pixabay.com/photo/2015/09/22/12/19/paper-951491_960_720.jpg

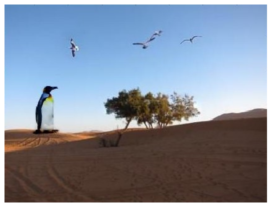
Figure 4: Example of mixed gradients. The girl with the balloon is pasted into the paper texture. If importing gradients was used it would be very obvious where the domain starts since it would look very smooth.

Task 5

Collage



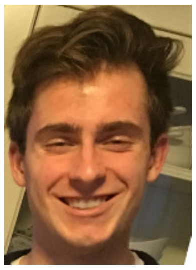
(a)



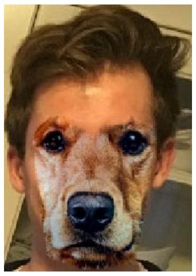
(b)

Figure 5: Birds and penguin were added to the image and the tree was increased in size. A mixture of importing and mixed gradients was used.

Dog face



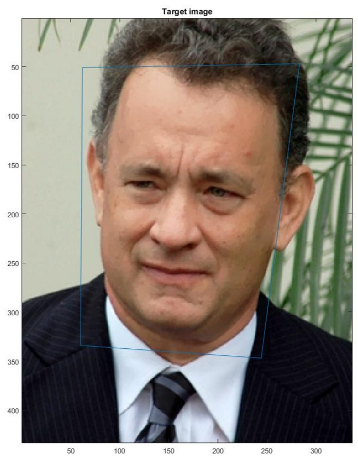
(a)



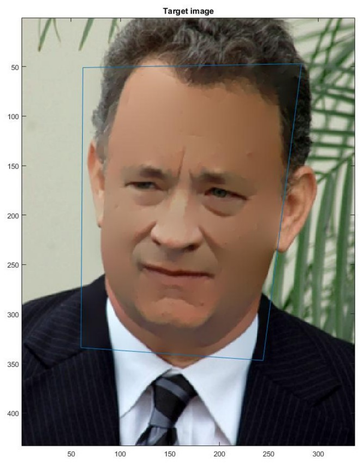
(b) Dog face image from https://www.behance.net/gallery/3711806/DOG-FACE-BOOK-Available-Now

Figure 6: Importing gradients was used to create this image.

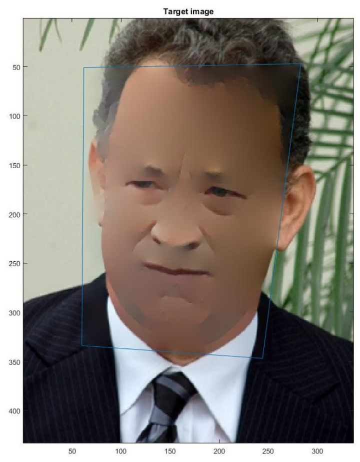
Task 6



(a)



(b)



(c)

Figure 7: Texture flattening applied to the same region with edge maps at different thresholds. (a) T=0.01 (b) T=0.06 (c) T=0.1

The domain is multiplied pixel wise with an edge map. I use equation (15) from the paper. Then to compute $\sum_{q \in N_p} v_{pq}$ I first check whether p is on an edge and then proceed as before.

To run the example please run *ex6.m*. Then you can select a region to flatten the texture off.

This technique can be useful for image segmentation.

The implementation is in ex6.m