Homework 3
Interpolation and Image Inpainting

Student: Yuanyuan Zhao, 2300013077, School of EECS, zhaoyuanyuan@stu.pku.edu.cn
Lecturer: Yisong Chen

Part 1: Image Inpainting Program Design

Given 4 original-damaged image pairs, we need to designed a method to remove the Red Text on the damaged ones, in an effort to get closed to the original at most. Specific implementation can be found in the file: image_inpainting.mlx.

(a) **Build the Mask**

(i) Based on our observation of the damaged images and their RGB values, the fact was found that all the damaging texts are pure red([255, 0, 0] in RGB).
So we have

```
target_red = [255,0,0];
```

(ii) Next, we need to find the damaged regions to be inpainted, to say, pixels whose RGB value == [255, 0, 0].
So we have

```
mask = img_damaged(:,:,1)==target_red(1) &
img_damaged(:,:,2)==target_red(2) &
img_damaged(:,:,3)==target_red(3);
```

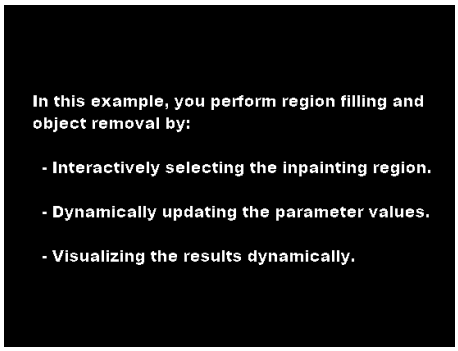
This strict correspondence might be short of adaptiveness since in a real scenario what we are going to remove is not a pure color in all likelihood. In that case a threshold is needed and pixels within the target color range will get masked. But after all, the basic idea is to match those pixels with damaging features (e.g., pure red in this homework).

(b) **Inpaint the Masked Region**

(i) After acquring the masked region, we call the function `regionfill(I, mask)` to replace masked pixel values using their neighbors. There are other inpainting methods such as `inpaintExemplar` in Matlab as well. But under comparison, `regionfill` seems to achieve the best performance despite some subtle inauthentic details which will be discussed later.

(ii) Algorithm of `regionfill`: `regionfill` smoothly interpolates inward from the pixel values on the outer boundary of the regions. `regionfill` calculates the discrete Laplacian over the regions and solves the Dirichlet boundary value problem.(1) Specific implementation details and principles will be discussed in the following section.

(a) Damaged Image

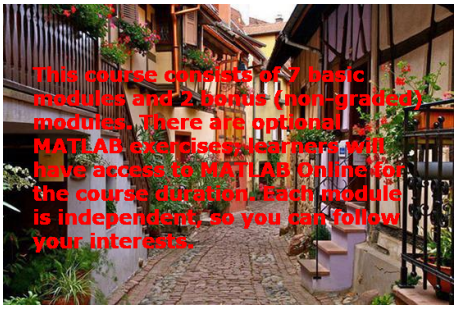


(b) Mask

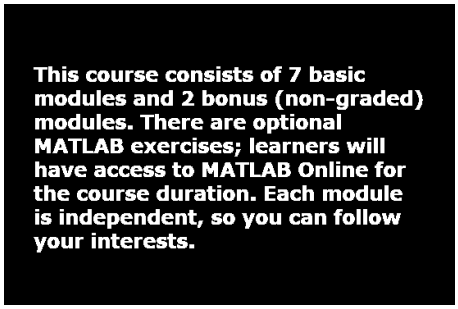
(c) Inpainted Image

(d) Original Image

Figure 1: Colmar: Before, during and after inpainting



(a) Damaged Image



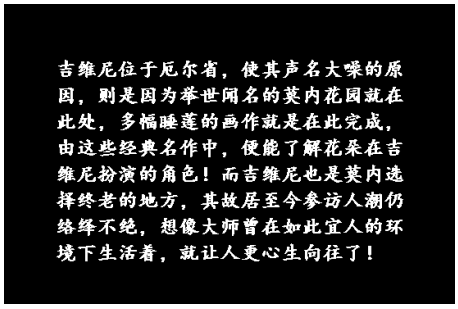
(b) Mask

(c) Inpainted Image

(d) Original Image

Figure 2: Eguisheim: Before, during and after inpainting

(a) Damaged Image



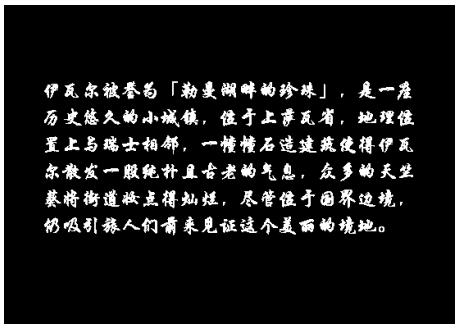
(b) Mask

(c) Inpainted Image

(d) Original Image

Figure 3: Giverny: Before, during and after inpainting

(a) Damaged Image



(b) Mask

(c) Inpainted Image

(d) Original Image

Figure 4: Yvoire: Before, during and after inpainting

Image Pairs	Colmar	Eguisheim	Giverny	Yvoire
PSNR(dB)	32.51	27.44	29.31	29.5
SSIM	0.99	0.96	0.96	0.97

Table 1: PSNR and SSIM evaluation for each image pair

Part 2: Result Analysis

(a) **Inpainted Images**
As shown in Figure 1, 2, 3, 4, the ultimate results are satisfying at first glance. Yet when zooming in, we can still find them inauthentic somewhere. To be specific, traces of characters are still there despite the color integration with their surroundings. This is inevitably rooted in our inpainting algorithm, which will be elaborated on later.

(b) **Evaluations**
As listed in Table 1, the PSNR values are close to 30 dB, and the SSIM values are close to 1, both of which indicate that the inpainted images are highly similar to the original ones.

Part 3: Dive into `regionfill`

(a) **Implementation and Principles**(2)
Image inpainting aims to reconstruct missing regions (e.g., scratches, occlusions) by propagating information from intact neighboring pixels. Laplace's equation naturally models this process.

(i) Laplace's equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

Analogy to Steady-State Heat Flow: Imagine heating the edges of a missing region; Laplace's equation ensures heat (or pixel intensity) diffuses evenly until equilibrium. This mimics how pixel values "diffuse" into gaps.
Smoothness Prior: The equation enforces smooth transitions, avoiding sharp discontinuities. This is ideal for reconstructing textures or gradients (e.g., skies, skin tones).
In a simple, discretized version of Laplace's equation, the value of every grid element in the interior of the region equals the average of its north, east, south, and west neighbors in the grid.

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{h}[\frac{1}{h}(f(x+h,y) - f(x,y)) - \frac{1}{h}(f(x,y) - f(x-h,y))]$$

$$\text{let } h = 1, \quad \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

$$\Rightarrow f(i,j) = \frac{1}{4}[f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1)]$$

(ii) Sparse Matrix Linear System
Now we're going to set up a linear system of equations, $Ax = b$, so that the

solution gives us every pixel of the output image. Let's think of the pixels as being numbered in columnwise order from 1 to the total number of pixels.

```
u = find(mask);
v = ones(size(u));
% for masked pixels, calculate the average
ijv_mask = [...
    u  u         1.00*v
    u  u_north  -0.25*v
    u  u_east   -0.25*v
    u  u_south  -0.25*v
    u  u_west   -0.25*v ];
% for non-masked pixels, the original value
ijv_nonmask = [w  w  1.00*ones(size(w))];
ijv = [ijv_mask; ijv_nonmask];
A = sparse(ijv(:,1),ijv(:,2),ijv(:,3));
% b contains the original pixel values
% for pixels outside the mask
% or 0 for pixels inside the mask
b = I(:);
b(mask(:)) = 0;
x = A\b;
J2 = reshape(x,size(I)); % J2 is what we need
```

(b) **Strengths**

(i) Computationally efficient.

(ii) Preserves edges and gradients outside the masked region.

(iii) Works well for small holes or smooth textures.

(c) **Limitations**

(i) Struggles with large or complex missing regions (e.g., repetitive patterns).

(ii) Fails to capture high-frequency details (e.g., fine textures).

(iii) Assumes smoothness, which may oversimplify real-world images.

Part 4: Thoughts

Interpolation is the underlying idea of a bunch of algorithms in digital image processing, and varying methods of interpolation design determine varying effects and corresponding applicable scenarios. For instance, Laplace's equation, as a kind of interpolation, achieves smoothness yet discard high-frequency details and that's the trade-off.

Simple as interpolation may seem, it implies the widely-used idea of *Knowing the unknown from what we've got*, paving the way for countless advanced algorithms we are familiar with, even the neural network.

Exploring those behind a handy Matlab function call (e.g., `regionfill` in this report), I've

> discovered something beyond the Matlab documents, to say, the blogs recording the whole procedure of a function implementation. And that's exactly where I got some invaluable insights.

References

[1] MathWorks, "regionfill," June 2022. [Online]. Available: https://www.mathworks.com/help/images/ref/regionfill.html.

[2] S. Eddins, "Region filling and laplace's equation," June 2015. [Online]. Available: https://blogs.mathworks.com/steve/2015/06/17/region-filling-and-laplaces-equation/.