Assignment 1

Michèle Wyss
10-104-123

Inpainting

Problem

Inpainting is an image processing method that aims for reconstruction of lost or corrupted parts of an image. The goal is to get a plausible-looking image without undefined or invalid pieces.

Motivations

Sometimes it's a bad thing that some objects appear in a photograph, e.g. as a tourist I'd like to make a picture of a place but some people are standing around and occlude parts of the actual target image. A simple idea is to just remove the persons after the photograph has been taken. This is a typical example of where an image with holes and invalid regions appears. Image inpainting is one possible approach to generate sensible content for those regions that were occluded before.

Derivation of gradient

Gradient descent is a rather simple iterative approach to minimize a function. Starting at an (randomly chosen) initial point, the following update rule is performed:

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta), \tag{1}$$

where θ_j are the unknown parameters of the model, *alpha* is the *learning rate* and $J(\theta)$ is the gradient of a cost function that is aimed to be minimized.

For our inpainting problem, we formulate the following cost function:

$$J(u) = \frac{\lambda}{2} \|u - g\|_\Omega^2 + \|\nabla u\|_2.$$

Here, the value u that we optimize for is an image with sensible content, g is the input image containing undefined pieces and λ is a regularization parameter to control the tradeoff between exactness of fitting and smoothness of the result. The above update rule will then look as follows:

$$u[i,j] := u[i,j] - \alpha \cdot \frac{\partial}{\partial u[i,j]} J(u).$$

Clearly, we need to know the derivative $\frac{\partial}{\partial u} J(u)$ to apply gradient descent. It is given by

$$\begin{aligned}
\frac{\partial}{\partial u[i,j]} \left(\frac{\lambda}{2} \|u - g\|_\Omega^2 + \|\nabla u\|_2 \right) &= \frac{\partial}{\partial u[i,j]} \left(\frac{\lambda}{2} \left(\sum_{i,j} \Omega[i,j] \cdot (u[i,j] - g[i,j])^2 \right) + \|\nabla u\|_2 \right) \\
&= \frac{\lambda}{2} \cdot \frac{\partial}{\partial u[i,j]} \left(\sum_{i,j} \Omega[i,j] \cdot (u[i,j] - g[i,j])^2 \right) + \frac{\partial}{\partial u[i,j]} \|\nabla u\|_2 \\
&= \frac{\lambda}{2} \cdot \frac{\partial}{\partial u[i,j]} \Omega[i,j] \cdot (u[i,j] - g[i,j])^2 + \frac{\partial}{\partial u[i,j]} \|\nabla u\|_2 \\
&= \lambda \cdot \Omega[i,j] \cdot (u[i,j] - g[i,j]) + \frac{\partial}{\partial u[i,j]} \|\nabla u\|_2
\end{aligned}$$

The target image u is 2-dimensional and therefore $\|\nabla u\|_2$ is given by

$$\|\nabla u\|_2 = \sum_{i,j} \underbrace{\sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2}}_{\tau[i,j]}. \tag{2}$$

Because for fixed i and j most of the summation terms in Equation (2) are constant, the derivative $\frac{\partial}{\partial u[i,j]}\|\nabla u\|_2$ reduces to

$$\frac{\partial}{\partial u[i,j]}\|\nabla u\|_2 = \frac{\partial \tau[i,j]}{\partial u[i,j]} + \frac{\partial \tau[i-1,j]}{\partial u[i,j]} + \frac{\partial \tau[i,j-1]}{\partial u[i,j]}.$$

There are now derivative terms that are left to compute. This can be done straight forward:

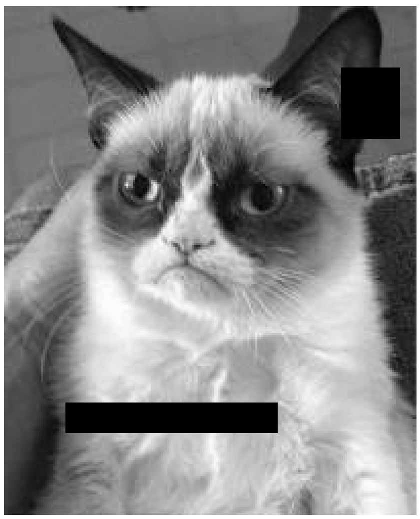
$$\begin{aligned}
\frac{\partial \tau[i,j]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]}\sqrt{(u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2} \\
&= \frac{1}{2}\cdot\frac{1}{\tau[i,j]}\cdot((-2)\cdot(u[i+1,j]-u[i,j]) - 2(u[i,j+1]-u[i,j])) \\
&= \frac{-((u[i+1,j]-u[i,j]) + (u[i,j+1]-u[i,j]))}{\tau[i,j]} \\
&= \frac{2u[i,j]-u[i+1,j]-u[i,j+1]}{\tau[i,j]}, \\
\frac{\partial \tau[i-1,j]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]}\sqrt{(u[i,j]-u[i-1,j])^2 + (u[i-1,j+1]-u[i-1,j])^2} \\
&= \frac{1}{2}\cdot\frac{1}{\tau[i-1,j]}\cdot(2\cdot(u[i,j]-u[i-1,j])) \\
&= \frac{u[i,j]-u[i-1,j]}{\tau[i-1,j]}, \\
\frac{\partial \tau[i,j-1]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]}\sqrt{(u[i+1,j-1]-u[i,j-1])^2 + (u[i,j]-u[i,j-1])^2} \\
&= \frac{1}{2}\cdot\frac{1}{\tau[i,j-1]}\cdot 2(u[i,j]-u[i,j-1]) \\
&= \frac{u[i,j]-u[i,j-1]}{\tau[i,j-1]}.
\end{aligned}$$

All in all we have therefore the following update rule:

$$\begin{aligned}
u[i,j] := u[i,j] - \alpha\cdot\Bigg(\lambda\Omega[i,j]\cdot(u[i,j]-g[i,j]) &+ \frac{2u[i,j]-u[i+1,j]-u[i,j+1]}{\tau[i,j]} \\
&+ \frac{u[i,j]-u[i-1,j]}{\tau[i-1,j]} \\
&+ \frac{u[i,j]-u[i,j-1]}{\tau[i,j-1]}\Bigg)
\end{aligned} \tag{3}$$

Implement gradient descent for inpainting

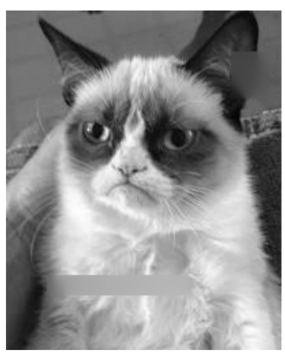
I implemented inpainting using the mentioned gradient descent method. The gradients were computed as derived above. A quite small learning rate of $\alpha = 0.0005$ ensures that the iterative approach converges – however, this also leads to the need of many iterations to get an acceptable result.



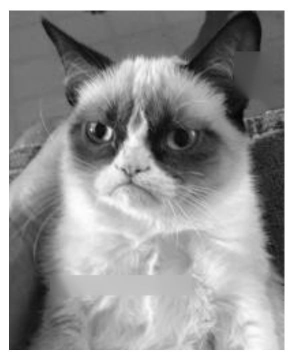
Input

Output ($100'000$ iterations)

1000 iterations

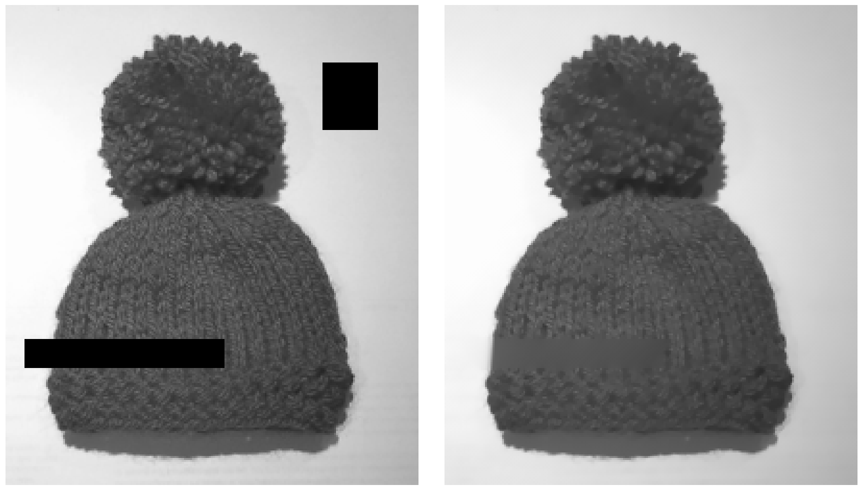


$10'000$ iterations



$50'000$ iterations

Another example:



Input | Output (50′000 iterations)

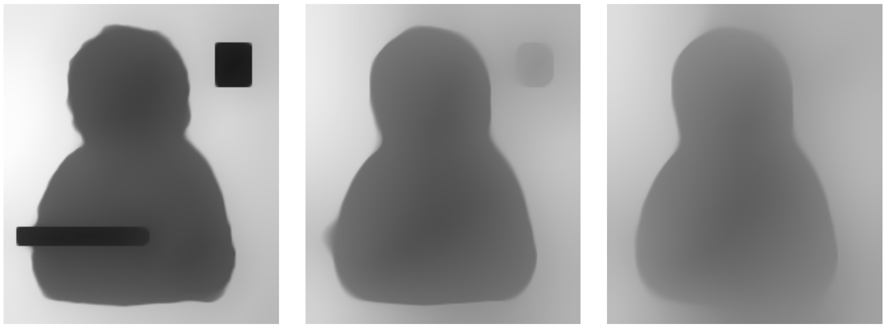
2′000 iterations | 5′000 iterations | 10′000 iterations

The effect of λ

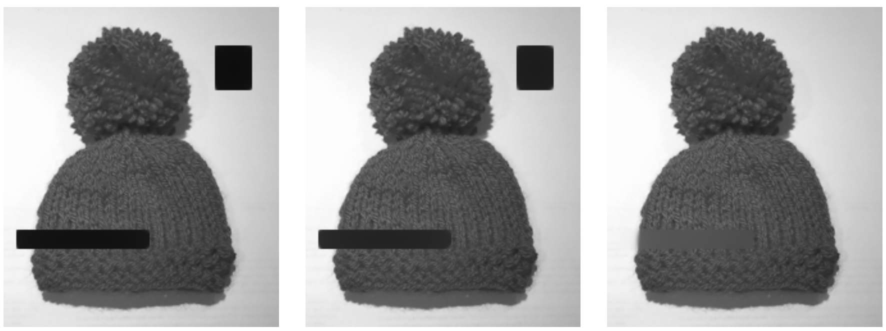
In the objective function

$$J(u) = \frac{\lambda}{2}\|u - g\|_{\Omega}^{2} + \|\nabla u\|_2$$

the parameter λ is a *regularization parameter*. It controls the tradeoff between the two constraints of fitting the image and keeping the result smooth. A very low λ, e.g. $\lambda = 0.1$ leads to an oversmoothed image whereas a very large value for λ may possibly fail to reach a nice-looking (smooth) image (see Figure 3). Moreover, a large lambda-value generally requires a smaller learning rate α. This can be seen directly by considering the final update rule (3) from the fact that the first term gets a high weight and may cause the gradient descent method to diverge.



1’000 iterations, $\lambda = 0.1$ | 5’000 iterations, $\lambda = 0.1$ | $10'000$ iterations, $\lambda = 0.1$



$5'000$ iterations | $10'000$ iterations | $100'000$ iterations

Fig. 3: Top row: results obtained with $\lambda = 0.1$. Bottom row: results obtained with $\lambda = 10'000$.

1. **Derivation of gradient.** In this section you should:
   - Write the finite difference approximation of the objective function E.
   - Compute the gradient of the objective function $\nabla_u E$.
2. **Implement gradient descent for inpainting.** In this section you should:
   - Show some images, as the the gradient method progresses iteration by iteration. Display the initial and the final image and 3 more images in between.
3. **Show images obtained by very high, very low and optimal λ.** In this section you should:
   - Display 3 images with different λ (very low, very high and optimal).
   - Describe the effect of λ on the solution.

4. **Find optimal λ.** In this section you should:

    - Display the SSD vs. λ graph.
    - Describe the effect of λ with respect to the SSD between the ground truth and the solution image.