Assignment 1

Moser Stefan
09-277-013

Superresolution

Problem

Through various effects, an image may end up with a low resolution. This can be due to hardware limitations (e. g. an image taken from a cheap security camera), bandwith limitations (e. g. an image was downscaled for transmitting over a connection with very low bandwith) or others.

Motivation

To retrieve further details, we might now want to increase image quality by increasing its resolution. While this is not possible in a manner as depicted in popular TV series, we can make certain assumptions for regularization combined with convex optimization to increase quality of upscaling compared to a naive implementation by a fair amount. In this project we use a regularization term that penalizes high gradients, assuming that the original image largely consists of smooth patches.

Derivation of Gradient

The energy term we want to optimize in this problem is

$$E(u) = \|\nabla u\| + \frac{\lambda}{2} \|Du - g\|_2^2 . \tag{1}$$

The left summand we will call regularisation term from here on, the right one fitting term.

The regularisation term consists only of the gradient of the superresolution image u, conserving the smoothness of the image. While any norm would be acceptable, I chose the 2-norm for its nice mathematical properties.

The fitting term consists of the channel selector D, the superresolution image u and the initial image g (with low resolution). This term ensures that the optimum

$$\tilde{u} = \min_{u \in X} E(u) = \min_{u \in X} \|\nabla u\| + \frac{\lambda}{2} \|Du - g\|_2^2 \tag{2}$$

stays close to the initial image when applying the channel selector.

The gradient of the fitting term on the right can be derived pretty easily using the chain rule

$$\begin{aligned} \frac{\partial E(u)}{\partial u} &= \|\nabla u\| \frac{\partial}{\partial u} + \frac{\lambda}{2}(Du - g)^T(Du - g)\frac{\partial}{\partial u} && (3) \\ &= \|\nabla u\| \frac{\partial}{\partial u} + \lambda(D^T Du - D^T g) && (4) \end{aligned}$$

by expressing the images u and g as vectors. For further use, we define this term as

$$f(u) = D^T Du - D^T g \tag{5}$$

The regularization term however needs some more work. For this purpose we first discretize the norm of the gradient as

$$\|\nabla u\| \approx \sum_{i,j} \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2} \tag{6}$$

One single term of this sum can then be expressed as

$$\tau[i,j] = \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2} \tag{7}$$

Using these tools, we can now easily define the gradient of the regularization term in a single component as

$$\frac{\partial \|\nabla u\|}{\partial u[i,j]} = \frac{\partial \tau[i,j]}{\partial u[i,j]} + \frac{\partial \tau[i-1,j]}{\partial u[i,j]} + \frac{\partial \tau[i,j-1]}{\partial u[i,j]} \tag{8}$$

With all other terms being independent of $u[i,j]$ and thus being zero. These three expressions can now be computed as

$$
\begin{aligned}
\frac{\partial \tau[i,j]}{\partial u[i,j]} &= \frac{\partial \sqrt{(u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2}}{\partial u[i,j]} && (9)\\
&= \frac{1}{2}((u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2)^{-\frac{1}{2}} && (10)\\
&\quad \frac{\partial}{\partial u[i,j]}(u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2 && (11)\\
&= \frac{1}{2}\tau[i,j]^{-1} && (12)\\
&\quad \frac{\partial}{\partial u[i,j]}(u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2 && (13)\\
&= \frac{1}{2}\tau[i,j]^{-1}(-2(u[i+1,j]-u[i,j]) - 2(u[i,j+1]-u[i,j])) && (14)\\
&= \tau[i,j]^{-1}(2u[i,j]-u[i+1,j]-u[i,j+1]) && (15)\\
&= \frac{2u[i,j]-u[i+1,j]-u[i,j+1]}{\tau[i,j]} && (16)
\end{aligned}
$$

In a similar fashion, we can compute the partial derivatives for $\tau[i-1,j]$ and $\tau[i,j-1]$

$$
\begin{aligned}
\frac{\partial \tau[i-1,j]}{\partial u[i,j]} &= \frac{\partial \sqrt{(u[i,j]-u[i-1,j])^2 + (u[i-1,j+1]-u[i-1,j])^2}}{\partial u[i,j]} && (17)\\
&= \frac{1}{2}\cdot\frac{1}{\tau[i-1,j]}\cdot 2\cdot(u[i,j]-u[i-1,j]) && (18)\\
&= \frac{u[i,j]-u[i-1,j]}{\tau[i-1,j]} && (19)
\end{aligned}
$$

respectively

$$
\begin{aligned}
\frac{\partial \tau[i,j-1]}{\partial u[i,j]} &= \frac{\partial \sqrt{(u[i+1,j]-u[i,j-1])^2 + (u[i,j]-u[i,j-1])^2}}{\partial u[i,j]} && (20)\\
&= \frac{1}{2}\cdot\frac{1}{\tau[i,j-1]}\cdot 2\cdot(u[i,j]-u[i,j-1]) && (21)\\
&= \frac{u[i,j]-u[i,j-1]}{\tau[i,j-1]} && (22)
\end{aligned}
$$

The τ terms need additional care, since they appear in the denominator. To prevent division by zero and increase numerical stability, we approximate them as

$$\tilde{\tau}[i,j] = \sqrt{(u[i+1,j]-u[i,j])^2 + (u[i,j+1]-u[i,j])^2 + \delta} \qquad (23)$$

with δ being a small positive number.

Combining our efforts from before, we can express the gradient of the objective function as

$$\nabla E[i] = \frac{2u[i,j] - u[i+1,j] - u[i,j+1]}{\tau[i,j]} + \tag{24}$$

$$\frac{u[i,j] - u[i-1,j]}{\tilde{\tau}[i-1,j]} + \tag{25}$$

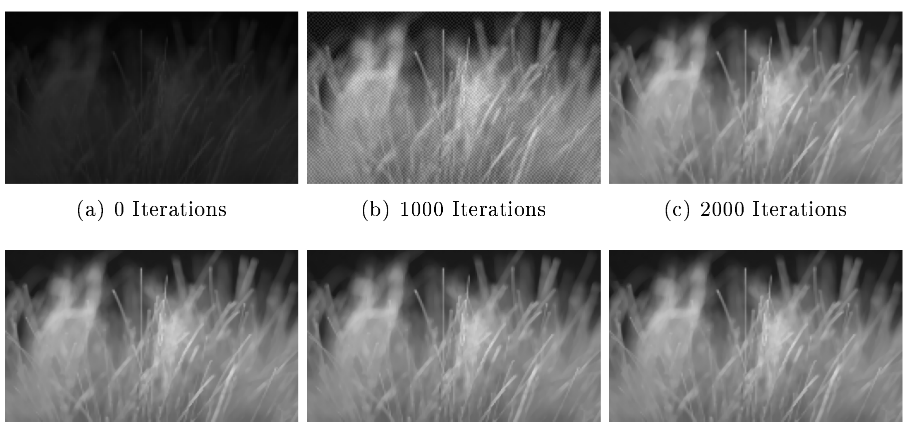
$$\frac{u[i,j] - u[i,j-1]}{\tilde{\tau}[i,j-1]} + \tag{26}$$

$$\lambda \cdot f(u)[i] \tag{27}$$

Implementation

Iteration count

In my implementation, I used Dg as initial guess. While around 5000 iterations seemed to be enough in most cases, I went up to 10'000 iterations to be sure there is no misleading effect due to low iteration count.



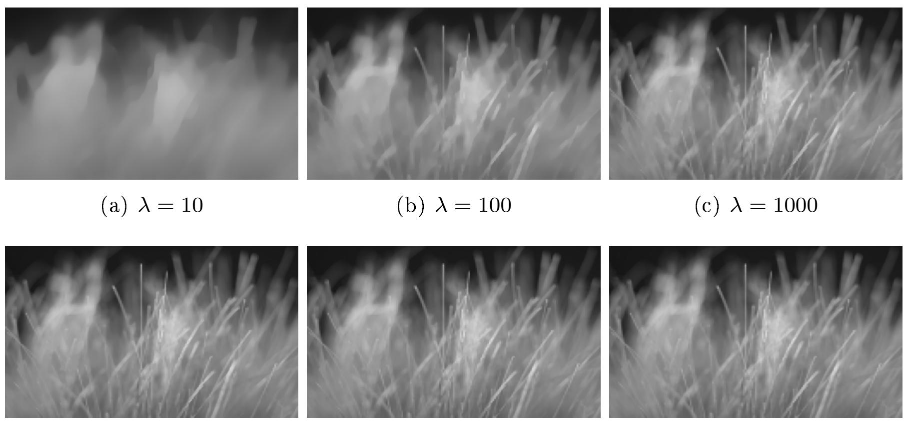
(a) 0 Iterations (b) 1000 Iterations (c) 2000 Iterations

(d) 4000 Iterations (e) 6000 Iterations (f) 8000 Iterations

Fig. 1: The superresolution image after various iteration counts. 0 iterations shows the initial guess. After iteration 4000 there are only minor changes. The lambda was set to 700 for these images.

Effect of Lambda

As can be seen in Figure 2, the lambda defines the smoothness of the resulting image. Surprisingly, the quality only deteriorates with extremely large lambdas. The choice of lambda also impacts the choice of the learning rate: With a large lambda, the learning rate has to be lowered in order for the optimization to not become unstable.



(a) $\lambda = 10$ (b) $\lambda = 100$ (c) $\lambda = 1000$

(d) $\lambda = 2000$ (e) $\lambda = 5000$ (f) $\lambda = 10000$

Fig. 2: Various resulting images with different lambdas after 8000 iterations.

Optimal Lambda

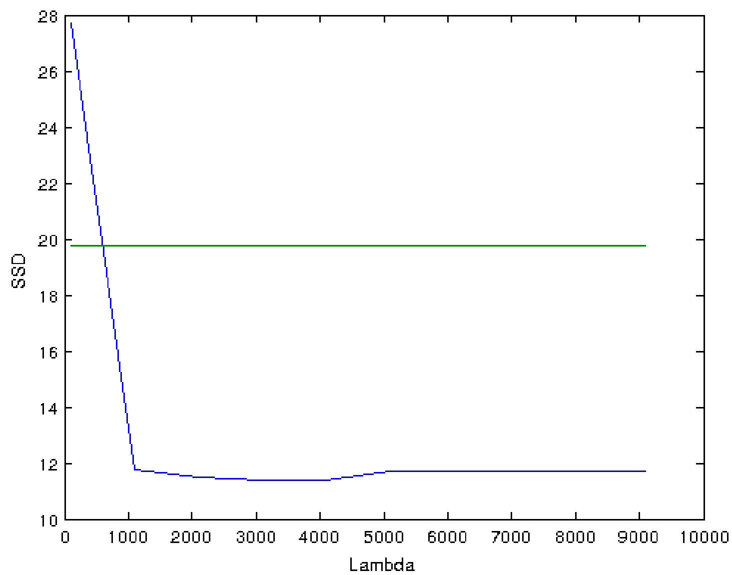


Fig. 3: SSD for various lambdas after 8000 iterations. For comparison, the SSD of the image obtained through nearest-neighbor scaling was added (green line).

I tested a wide variety of lambdas before concluding that the optimal lambda is somewhere around 4000 (see Figure 3). The improvements above a lambda of 1000 are only marginal however. As already stated above, the results don't become much worse when choosing even larger lambdas.