Smoothing

Having learned different methods for acquiring surfaces, we now move towards manipulating and processing surfaces. This and the next few chapters deal with *smoothing*. We will first motivate the problem, then we will detour to learn a lot of necessary mathematical background, and then we will introduce a few standard smoothing algorithms.

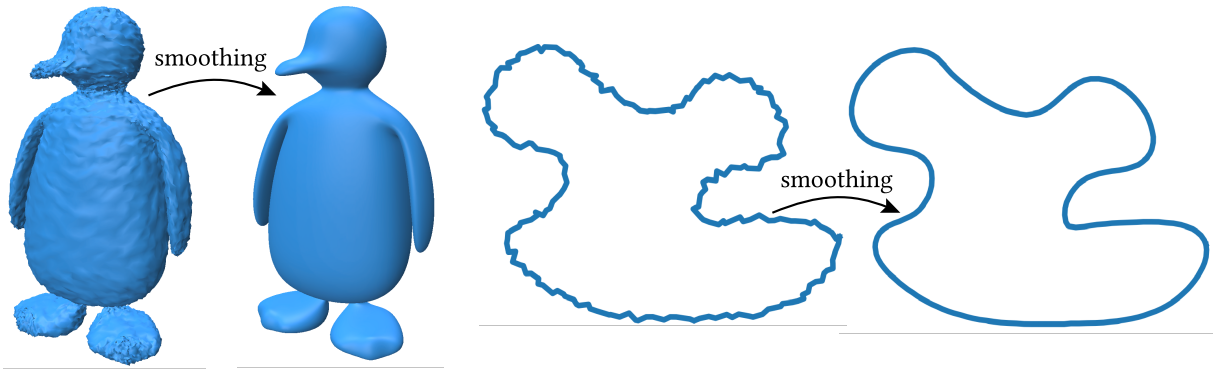


Figure 1: A smoothing operation can remove noise from surfaces that come, e.g., from a scanning process.

Smoothing as denoising

There are a myriad reason for why one would want to smooth a surface. One such reason is to remove noise from a surface, or *denoising*. This operation is sometimes also known as *surface fairing*. A surface scan, such as the one employed in the previous chapter, will naturally be noisy because of natural noise inherent in the capturing process. The resulting noisy surfaces have to be denoised via *smoothing* to obtain a smooth surface. Examples of this process are shown in Figure 1.

Denoising a function in 1D

You might be familiar with many ways of denoising for smoothing. Let us first look at denoising from a signal processing perspective in 1D. We have a noisy function $v : \mathbb{R} \to \mathbb{R}$, and we want to get a smooth function $u : \mathbb{R} \to \mathbb{R}$. One sensible approach could be subsampling v. This is actually often done in certain visualization methods. Figure 2 shows an example of the subsampling process in action:

one-day view one-month view

Figure 2: SMI market data as displayed by Google Finance after market close on Jan 29, 2024. While the daily view samples data every five minutes, the monthly view only samples once a day, leading to a smoother result.

Subsampling results in the removal of high-frequency (or small-scale) noise. In Figure 2, the entire one-day view on the left, which is full of noisy ups and downs, becomes a simple line segment in the one-month view. All these high-frequency variations are removed, one might say, because they are not relevant for somebody who tries to analyze one month's worth of market movements compared to somebody trying to analyze one day's worth.

"Something to think about" 1: Noise vs. high-frequency data

What is noise, and what is high-frequency data? What makes us want to remove one kind of high-frequency detail, but leave in another?

Subsampling can be a very simple and useful strategy for data that has natural points in the domain on which to sample. In stock market data, for example, one can naturally sample at market close to sample at the end of each day. But if our data has no obvious points to subsample at, then we would have to arbitrarily choose the subsampling point, which might bias our data.

An alternative to this is to take an average over each area instead of simply subsampling. Consider a function given by values at the points $v_0, \dots, v_n$. A simple averaging operation might look like this:

$$\begin{aligned} u_0 &= \frac{1}{2}(v_0 + v_1) \\ u_i &= \frac{1}{3}(v_{i-1} + v_i + v_{i+1}) \quad i \neq 0, n \\ u_n &= \frac{1}{2}(v_{n-1} + v_n) \end{aligned} \tag{1}$$

Figure 3 shows a sketch of such a simple averaging operation. A green node is assigned the average of itself and its neighbor nodes.

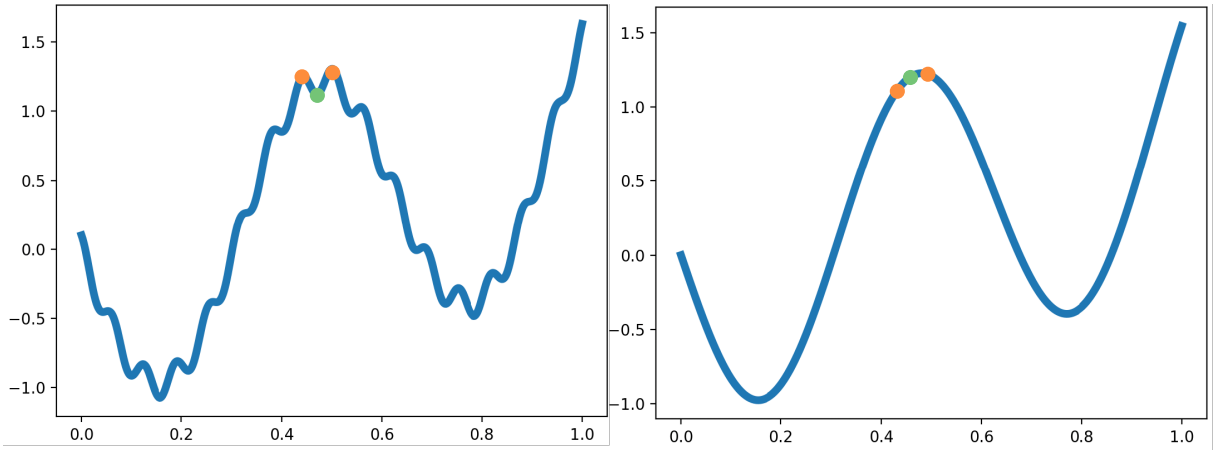


Figure 3: For an averaging operation, we set the value of each node to be the average of itself and all previous nodes.

"Something to think about" 2: What is smoothness?

We want to make functions smoother. But what does it even mean to be a smooth function? What is the smoothest function?

Now, an averaging operation does not let us control the amount of averaging. Ideally, we would like to specify the amount of averaging that is performed. A proper averaging parameter $\lambda \geq 0$ fulfills the following properties:

1. If $\lambda = 0$, then nothing changes at all.
2. The greater λ, the greater the smoothing.
3. If v is a constant function, nothing should happen.

The last property is particularly important, since it gives some substance to our notions of *less smooth* and *more smooth*. Under this interpretation, the smoothest function is a constant function, and the smoother something is, the more constant it is.[1]

Let us derive an averaging operation that fulfills the conditions from above. Ignoring the boundaries u_0, u_n for now, we vary the different coefficients:

$$u_i = \alpha_{i-1} v_{i-1} + \alpha_i v_i + \alpha_{i+1} v_{i+1}. \tag{2}$$

If $\lambda = 0$ then nothing should change, so we can try

$$u_i = \beta_{i-1} \lambda v_{i-1} + (1-\lambda) v_i + \beta_{i+1} \lambda v_{i+1}. \tag{3}$$

If we want constant functions to have no effect, let's plug in a constant into the equation and see what this means for the βs:

$$c = \beta_{i-1} \lambda c + (1-\lambda) c + \beta_{i+1} \lambda c = c + \lambda(\beta_{i-1} - 1 + \beta_{i+1}) c \tag{4}$$

We can thus conclude that $\beta_{i-1} + \beta_{i+1} = 1$. If we additionally want our averaging scheme to be symmetric, i.e., it should not matter in which direction we average in, then $\beta_{i-1} = \beta_{i+1} = \frac{1}{2}$. This makes our tunable averaging scheme:

$$u_i = \frac{1}{2}\lambda v_{i-1} + (1-\lambda) v_i + \frac{1}{2}\lambda v_{i+1} = v_i + \frac{\lambda}{2}(v_{i-1} - 2v_i + v_{i+1}). \tag{5}$$

On boundaries, we have:

$$\begin{aligned} u_0 &= v_0 + \frac{\lambda}{2}(v_1 - v_0) \\ u_n &= v_n + \frac{\lambda}{2}(v_{n-1} - v_n). \end{aligned} \tag{6}$$

With the parameter λ we can now control the intensity of the smoothing. But the intensity parameter is still dependent on the resolution of the grid. Imagine that our function is $v : [0,1] \to \mathbb{R}$, and we have two different discretizations of the line: the first with n segments, each point h from the other, $v_0, \dots, v_n$; the second with $\frac{n}{2}$ segments, each point $\hbar = 2h$ from the other, $\tilde{v}_0, \dots, \tilde{v}_{2n}$. Let us try to smooth the function $f(x) = x^2$. We know that $v_4 = f(4h) = (4h)^2$, and $\tilde{v}_2 = (2\hbar)^2 = (4h)^2$.

Both v_4 and $\tilde{v}_2$ correspond to the same value, so they should be smoothed somewhat similarly. So, what happens?

$$\begin{aligned} u_4 &= (4h)^2 + \frac{\lambda}{2}\Big((3h)^2 - 2(4h)^2 + (5h)^2\Big) = (4h)^2 + \lambda h^2 \\ \tilde{u}_2 &= (4h)^2 + \frac{\lambda}{2}\Big((2h)^2\Big) - 2(4h)^2 + (6h)^2) = (4h)^2 + 4\lambda h^2. \end{aligned} \tag{7}$$

The result changed – the smoothing became more intense. In order to account for that, we incorporate the grid spacing h into our average smoothing scheme:

[1] This is not always true, even for some of the examples mentioned in this chapter – sometimes the smoothest function is a linear function, or even something else.

$$
\begin{aligned}
u_0 &= v_0 + \frac{\lambda}{2h^2}(v_1 - v_0) \\
u_i &= v_i + \frac{\lambda}{2h^2}(v_{i-1} - 2v_i + v_{i+1}) \quad i \neq 0, n. \\
u_n &= v_n + \frac{\lambda}{2h^2}(v_{n-1} - v_n)
\end{aligned} \tag{8}
$$

This assures that, no matter what the grid spacing is, the parameter λ controls smoothing intensity. For the above example with $f(x) = x^2$, smoothing becomes a resolution-independent $16h^2 + \lambda$ in both the tilde- and non-tilde cases.

We now have a smoothing via averaging scheme as an alternative to mere subsampling that is symmetric, where we can control the intensity of smoothing, and that intensity can be controlled independent of the resolution of the grid.

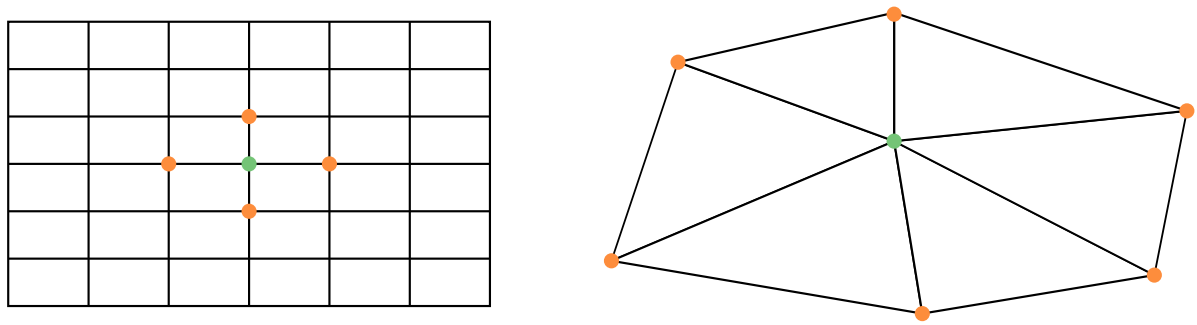
This smoothing scheme is an explicit finite difference scheme in one dimension. This is not what we will end up using eventually for actual smoothing, neither in 1D, not in 2D, but it is a useful way to start thinking about smoothing: Smoothing is averaging.

"Something to think about" 3: Limits for λ

When $\lambda = 0$, there is no smoothing. But how big can we make λ? What happens for $\lambda = \frac{1}{2}$, or $\lambda = 1$?

Denoising a function in 2D

The averaging schemes we learned in 1D can be generalized to 2D. Consider now smoothing a function $v : \Omega \to \mathbb{R}$ on a surface. We can average similarly to 1D, but the neighbors are now all neighboring grid vertices or all neighboring vertices on a triangle mesh (Figure 4).



grid neighbors 1-ring neighborhood

Figure 4: Averaging operations on a two-dimensional grid and on a triangle mesh.

Let $\mathcal{N}(i)$ be the set of all neighbors of vertex i in a grid or triangle mesh, and let h be an average of edge lengths. Then a basic 2D averaging operation is:

$$
u_i = v_i + \frac{\lambda}{|\mathcal{N}(i)|\ h^2} \sum_{j \in \mathcal{N}(i)} (v_j - v_i) \tag{9}
$$

This is a controllable, scale-independent averaging operation similar to the one we developed in 1D.

Denoising a surface

Instead of merely smoothing a function $v : \Omega \to \mathbb{R}$, we can also smooth multi-dimensional functions $\Omega \to \mathbb{R}^3$. One such function is the *coordinate function* (x, y, z). If we smooth the coordinate function

itself, then we denoise *the surface itself*, and not merely a function defined on the surface. Using this, one can denoise surfaces themselves, like in Figure 1.

Averaging means derivatives

The averaging we have discussed in the last section has strong similarities to derivatives. This is because the finite difference scheme is merely a discretization of the action of differentiation.

Remember how derivatives are defined in calculus:

$$v'(x) = \lim_{h \to 0} \frac{v(x+h) - v(x)}{h} \tag{10}$$

On a one-dimensional grid, we can obtain a discrete approximation of the derivative by simply forgoing the limit in (10), the *finite difference approximations*:

$$\begin{aligned} v_i' &= \frac{v_{i+1} - v_i}{h} \\ v_i' &= \frac{v_i - v_{i-1}}{h} \\ v_i' &= \frac{v_{i+1} - v_{i-1}}{2h}. \end{aligned} \tag{11}$$

These three approximations are the forward, backward, and centered finite differences.

If we do repeated centered finite differences on a staggered grid, then we have:

$$\begin{aligned} v_{i+\frac{1}{2}}' &= \frac{v_{i+1} - v_i}{h} \\ v_{i-\frac{1}{2}}' &= \frac{v_i - v_{i-1}}{h} \\ v_i'' &= \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2}. \end{aligned} \tag{12}$$

So, it turns out, our averaging process is actually the finite difference discretization of the following operation:

$$u(x) = v(x) + \frac{\lambda}{2} v''(x) \tag{13}$$

"Something to think about" 4: Higher dimensions

What does the second derivative in (13) generalize to in higher dimensions?

Smoothing as a physical process

So far we have viewed smoothing as a signal processing operation: We take a signal, and we average it to get a smoother signal. An alternative interpretation of the smoothing operation is to view is as a physical process. Which physical processes naturally produce smoothing?

Diffusion

One natural process we can exploit for smoothing is *diffusion*. Due to random statistical processes, if a substance starts with unequal concentration, the concentration wants to equalize, which acts a little bit like a signal filter – the noisiest, high-frequency disparities are removed first. This process can be seen in Figure 5. Additionally, with diffusion, over time substances tend towards complete

equal concentration in the entire domain. This means that the steady state of diffusion is a constant function, just like with our averaging scheme.

before diffusion
harsh gradient in domain

after diffusion
more equalized distribution

Figure 5: When a substance has non-uniform concentrations in a domain, over time the concentrations equalize, which performs a smoothing operation [1].

The diffusion process is governed by the diffusion equation:

$$\frac{\partial u(x,t)}{\partial t} = \rho \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{14}$$

where $u(x,t)$ is the concentration of a material in the domain in space (x) and time (t). ρ is a material parameter that governs the speed of diffusion. (14) is a *Partial Differential Equation* (PDE), which adds a time dimension in addition to the spatial dimension we have had so far. This is because diffusion is a physical process that happens over time. You can also understand smoothing / denoising as a process that occurs over time: the averaging operations from (13) can be performed multiple times, mimicking an evolution in time.

Heat

Very similar to diffusion, heat differences in nature equalize over time. This process is governed by the *heat equation*, which is (in its simple form) exactly the same as the diffusion equation:

$$\frac{\partial u(x,t)}{\partial t} = \rho \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{15}$$

where $u(x,t)$ is the heat of a material, x is the spatial coordinate, and t is the time component. If we discretize the time component of the PDE using a superscript, $u^{(j)}$, then (15) becomes:

$$u^{(j+1)} = u^{(j)} + (\delta t)\rho \frac{\partial^2 u^{(j)}}{\partial x^2} \tag{16}$$

which is, again, the same as our averaging operation from (13).

"Something to think about" 5: Other finite differences

What if we use other finite differences in discretizing the time portion of the diffusion and heat equations?

These physical processes are what we will actually use for smoothing in practice. In order to understand them, we will need to brush up on our calculus knowledge. This will happen in the next chapter.

Bibliography

[1] Sbyrnes321. 2010. File:DiffusionMicroMacro.gif via Wikimedia, https://commons.wikimedia.org/wiki/File:DiffusionMicroMacro.gif.