

A APPENDIX

A.1 IMPLEMENTATION DETAILS

The main part of the generator consists of four blocks containing two convolution layers and a residual connection. The feature count of the convolution layers per block is as follows: $[[8, 32], [64, 64], [32, 8], [1, 1]]$. The kernel size is $5 \times 5 \times 5$ for all layers. For the discriminators we use only convolution layers and no residual blocks. Both discriminators, spatial and temporal, have the same number of features per layer: $[8, 16, 32, 32]$. The fully-connected layer at the end of the layer consists of 64 and 1 neurons. The kernel size is $4 \times 4 \times 4$ for all layers. It is important to mention that for the first convolution of every network we do not use zero padding as usual but mirror padding. This is because we work with tiles from the training data and not with the complete frame and zero padding falsifies the values at the transitions between the tiles.

A.2 TRAINING DETAILS

For the training we have implemented our network with the Tensorflow Framework. We use an Adam optimizer with a learning rate of 0.00001 and a batch size of 16 for the 2D tests and 4 for the 3D tests for 50k iterations. All other weights are initialized with the respective standard initializers of Tensorflow version 2.1. The weighting factors α and β of Equation 4 are set to 1.0 and 10.0 correspondingly.

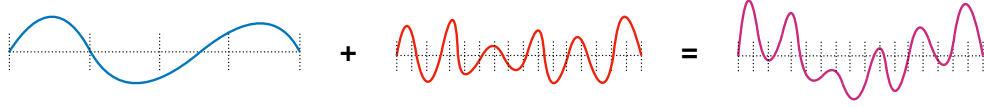


Figure 8: In blue a wave with a randomly varying low frequency f_l , which can still be represented by the low-resolution sampling and serves as input data set for our synthetic data set. In red, on the other hand, we see a wave with a high frequency f_h that can only be correctly represented with much higher resolution. Combined with the low-frequency version, we get the ground-truth data (violet) for the synthetic data set.

A.3 SYNTHETIC DATA SET

We use a synthetic data set to test and evaluate different aspects of our approach. The data set is designed to have a simple, clearly defined behavior, and such that the frequency spectrum of the surface can be evaluated reliably. Therefore, we use a horizontal wavy surface based on a 1D sine function $s_t(x)$ with randomly varying frequency $f_l \in (0, \frac{M}{2})$:

$$s_0(x) = \sin\left(\frac{\pi f_l(x)x}{M}\right), x \in [0, M), \quad (7)$$

where M is the resolution of the low-resolution data.

To make a time sequence out of this, we use a simple wave equation:

$$\frac{\delta^2 s}{\delta t^2} = \frac{\delta^2 s}{\delta x^2}, \quad (8)$$

Discretizing this we can calculate the vertical velocity of our surface as follows:

$$v_t(x) = v_{t-1}(x) + \frac{2 * s_t(x) - s_t(x - \Delta x) + s_t(x + \Delta x)}{\Delta x}, \quad (9)$$

where $v_0(x) = 0$. Given the velocity we can now calculate the next frame as follows:

$$s_t(x) = s_{t-\Delta t}(x) + \Delta t v_t(x), \quad (10)$$

For the high resolution data set, i.e., the targets to be learned, we use the same low resolution wave as base, and modulate it with a high frequency component (Figure 8):

$$t_0(x) = \sin\left(\frac{\pi f_l(x)x}{kM}\right) + \sin\left(\frac{\pi f_h(x)x}{kM}\right), x \in [0, kM), \quad (11)$$

where k is the chosen up-sampling factor and the frequency $f_h(x)$ is chosen so that it cannot be represented by the low-resolution version. According to the Nyquist-Shannon sampling theorem, the frequency should be higher than $\frac{M}{2}$ and below $\frac{kM}{2}$ (Figure 3(a)). The generation of a sequence is done in the same way like for the low-resolution data.

Based on this setup, we generate two different data sets of high resolution data: one where we modulate with a fixed high frequency $f_h(x) = \text{const}$, whereas in the second version we vary the this high frequency component (Figure 3(b)). Thus the first version represents a deterministic up-sampling, which a generator should be able to reconstruct perfectly, whereas the second version is ill-posed, i.e. several solutions are possible, and hence poses a much more difficult learning target. Both data sets consist of 10000 sequences with 30 frames each at the end. While the deterministic version only serves as a sanity check which we only discuss here in the appendix (A.5), the second, randomized version shows how well the method can approximate the ground-truth distribution for ill-posed tasks like the actual super-resolution problem for physical simulation data.

A.4 SIMULATION DATA

For the generation of simulation, data we use an SPH solver of the SPLisHSPlasH framework (Bender, 2017). There are different materials to choose from. With materials that exhibit high-frequency physical behavior, chaotic behavior occurs in some cases, such as splashes in water, which are typically very difficult to reconstruct. With more viscous materials, such as gel, details are mainly distinguished by folds and fine waves on the surface. The chaotic behaviour is very difficult to reconstruct and it is sometimes very difficult to understand how correct the behaviour is. For these reasons we focus more on materials like gel. With gel, fine wrinkles can form on the surface which allows a good evaluation of the method. Another special feature is that details are persistent over time. Methods such as Xie et al. (2018) cannot represent such details because they do not provide feedback in the network. This means there is no memory. Therefore, we use an plastic material with high viscosity, simulated with an advanced viscosity solver (Weiler et al., 2018).

For the training we generate 60 different scenes with 300 frames per simulation. The time step corresponds to 100 frames per second. The scenes consist of randomly generated shapes that fall into a pool from different heights at random times. This creates interesting waves and folds on the surface. The ground-truth resolution is 160^3 . For the generation of the low-resolution data we distinguish between the way the training is done. For the supervised setup the ground-truth data is scaled down by the desired up-sampling factor k and then smoothed with a Gaussian blur. This results in synchronous data pairs that can be used in a supervised setup. In the unsupervised setup, however, the low-resolution data is generated in the same way as the high-resolution data, with the correspondingly lower resolution. From the position and velocity data of the particles, we then generate our SDF and velocity grid which we use for training. Before the training, the data is normalized over the whole data set, so that the data is in the value range between -1 and 1. For a sharper edge in the SDF grid, which prevents unwanted noise in the generated data, we additionally modify the normalized values with a tangent hyperbolic function:

$$f(x) = \tanh(cx) \quad (12)$$

where c can be freely selected, depending on the strength of the transition. We found 5 to be a good value. Finally, we take advantage of the locality of our problem and only use excerpts from the training data frames in training. On the one hand, this saves memory, because the data is sometimes very large and can cause problems with the GPU memory, on the other hand it allows us to extract only relevant parts of the data. So in our example we can only consider the data in places where there is a surface. Finally we have the advantage to augment the data by overlapping the tiles we extract, so we can get a lot of information from only a few frames.

A.5 ADDITIONAL RESULTS

As a sanity check we used a deterministic setup as described in A.3. With this setup we tested if the network is able to modulate a static high frequency with a simple MSE loss. In Figure 9a this is shown to be the case. This serves as a baseline for what the network can achieve. In Table 9b we compare the frequency deviation with the MSE network trained with the non-deterministic setup. As expected, the error is much smaller with the deterministic setup.

In Figure 10 and Figure 11 we compare the frequency evaluation of two more data samples, the same as in Figure 4.

Finally there are some more 3D results in Figure 12 and Figure 13.

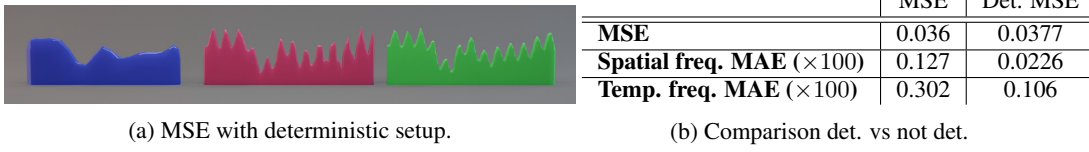


Figure 9: Evaluation of deterministic setup.

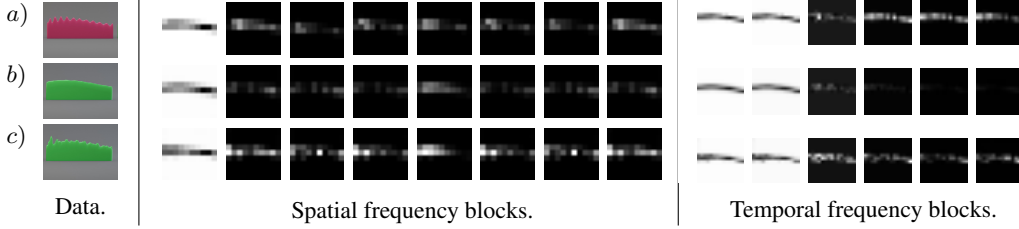


Figure 10: F.l.t.r the image data and the corresponding spatial and temporal frequency images are shown. We consider the ground-truth (1), a result based on MSE (2) and the surfGAN result (3).

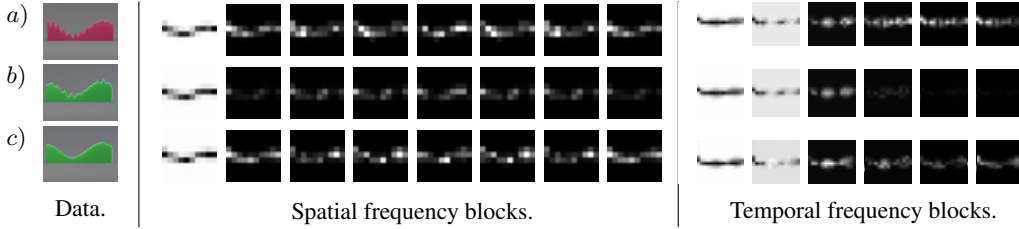


Figure 11: F.l.t.r the image data and the corresponding spatial and temporal frequency images are shown. We consider the ground-truth (a), a result based on MSE (b) and the surfGAN result (c).

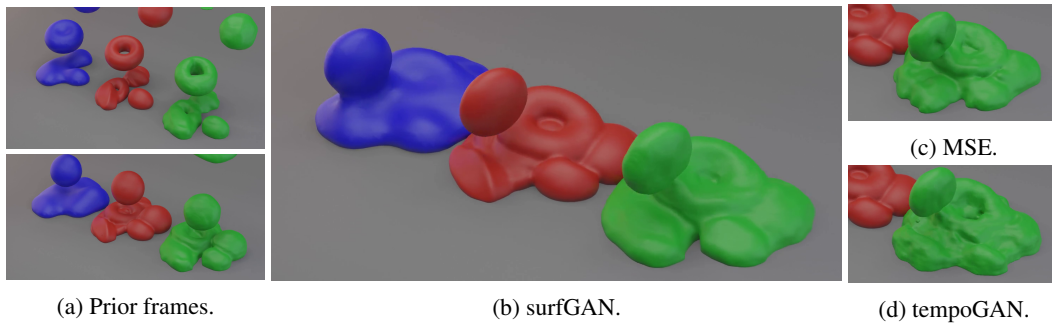


Figure 12: Comparison of a 3D test run with our surfGAN (b), with a MSE-based generator (c) and tempoGAN (d). The input is shown in blue, ground-truth in red, and predictions in green.

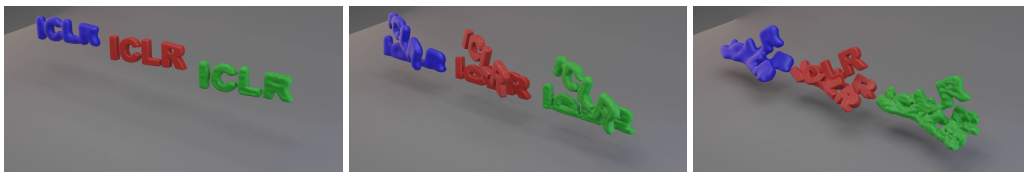


Figure 13: An additional output generated with the surfGAN network. The input is shown in blue, ground-truth in red, and the prediction in green.