Geometry Processing (**Project A**)

Learning Function Spaces: Implicit Generative Models with Dense Correspondences

Implicit representations for 3D geometry have grown increasingly popular in recent years due to their independence to discrete topological structures. In principle, this suggests that they might be a valuable tool in developing new solutions for several notoriously difficult tasks in geometry processing, for example, partial correspondence, shape reconstruction, interpolation and correspondence under topology changes.

However, combining deep learning with implicit representations poses new challenges, since implicit representations for 3D geometry are continuous functions with few inputs (usually ranging from 3 to 6). Using neural networks to learn conditional, high-frequency information in low dimensional function spaces has proved non-trivial by itself.

We propose to apply a recently published method which allows to learn latent spaces of implicit functions with smooth transitions between data points, and to use its output to evaluate a novel gradient-based correspondence algorithm.

This project's scope is to:
a) Gain insight about low-dimensional learning in function spaces
b) Work with complex, non-euclidean data such as 3D objects
c) Address a relevant task in geometric deep learning, shape correspondence

Bibliography:
[1] Tancik et al., "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains"
[2] Sitzmann et al., "Implicit Neural Representations with Periodic Activation Functions"
[3] Liu et al., "Learning Smooth Neural Functions via Lipschitz Regularization"
[4] Park et al., "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation"
[5] Mescheder et al., "Occupancy Networks: Learning 3D Reconstruction in Function Space"
[6] Mildenhall et al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis"

Language modeling for audio tasks (**Projects B**)

1. **Multi-Latent Autoregressive Source Separation (MLASS)**

   Problem Statement: The Latent Autoregressive Source Separation (LASS) method has shown competitive results in source separation in the weakly supervised setting and offers significant speedups in terms of inference time and scalability to higher dimensional data. However, it currently lacks the capability to separate more than two sources. The aim of this project is to extend LASS to enable source separation in a discrete latent domain of a VQ-VAE with more than two sources.

   Task: Students will be required to develop a method to sample each source z_i from the conditional distribution $P(z_i \mid m)$, marginalizing out the remaining random variables z_j with $j \neq i$, by leveraging classical algorithms over directed graphical models such as belief propagation tailored for inference with a finite number of latent variables. This should enable MLASS to separate more than two sources without the memory complexity issues associated with the naive generalization of the likelihood function $P(m \mid z_1, ..., z_m)$.

   References:

   1. Vector-Quantized Variational Autoencoders (VQ-VAE). https://arxiv.org/abs/1711.00937
   2. Latent Autoregressive Source Separation (LASS). https://arxiv.org/abs/2301.08562
   3. Belief Propagation in Graphical Models. https://cvg.cit.tum.de/_media/teaching/ss2017/pgmcv/in2329-08_bp.pdf

   Additional Notes: Students should have a good understanding of autoregressive models, VQ-VAE autoencoders, and the basics of directed graphical models to successfully complete this project.

---

2. **LQVAE + LASS hybrid**

   Problem Statement: Current state-of-the-art generative audio source separation models are either resource-intensive or impractical for use at inference time. The LQ-VAE and LASS models both offer alternative solutions to these limitations, but it is unclear which approach provides better separation performance. The goal of this project is to combine the LQ-VAE and LASS methodologies, potentially trading the flexibility of LASS for improved separation performance.

Task: Students will be required to compare the source separation music performance of the LQ-VAE model and the LASS model by re-training both models using publicly available datasets. They will then train a model with a loss function as in LQ-VAE, but using the technique of counting occurrences in the model codebook at inference time, as is done in LASS. The project aims to assess whether this hybrid approach can lead to better separation performance while maintaining efficiency at inference time.

References:

1. Linearly Quantized Variational Autoencoder (LQ-VAE).
   https://arxiv.org/abs/2110.05313
   https://github.com/michelemancusi/LQVAE-separation
2. Latent Autoregressive Source Separation (LASS).
   https://arxiv.org/abs/2110.05313

Additional Notes: Students should have a good understanding of autoregressive models and VQ-VAE autoencoders to successfully complete this project.

---

3. **Source Separation in Residual Quantized Latent Domains**

Problem Statement: State-of-the-art generative models for music and audio, such as EnCodec, AudioLM and MusicLM, are based on Residual Quantized Autoencoders (RQ-VAE), which map time-domain segments of waveform data to discrete latent representations comprising sequences of causally dependent tokens (residuals). These residuals enable higher perceptual quality in reconstructed audio. The goal of this project is to adapt generative source separation, as demonstrated in the Latent Autoregressive Source Separation (LASS) paper, to the residual setting, or alternatively, to use a discriminative approach to train a model for performing this separation procedure.

Task: Students have two options for this project:

1. Use a discriminative approach by training a model to perform the source separation procedure in residual quantized latent domain.
2. (Advanced) Adapt generative source separation as in the LASS paper to the residual setting by approximating the inference of the local posterior, taking into account the stack of residual codes.

Both approaches should be evaluated for their effectiveness in source separation with respect to standard VQ-VAE approaches (in the first case by training the model on a standard VQ-VAE domain and in second case with respect to LASS)

References:

1. Residual Quantized Autoencoders (RQ-VAE).
   https://arxiv.org/abs/2203.01941
2. Latent Autoregressive Source Separation (LASS).
   https://arxiv.org/abs/2110.05313
3. EnCodec: High Fidelity Neural Audio Compression.
   https://arxiv.org/abs/2210.13438
   https://github.com/facebookresearch/encodec
   https://ai.honu.io/papers/encodec/samples.html

Additional Notes: Students should have a good understanding of autoregressive models, VQ-VAE autoencoders and residual quantization to successfully complete this project.

---

4. **Multi-source Masked Language Models for Audio**

Problem Statement: Masked Language Models (MLM) like BERT have been successful in Natural Language Processing and have been extended to multi-modal settings, such as text-and-vision and audio-and-vision. Multi-source diffusion models (MSDMs) have also been proposed to tackle multiple inference tasks in the music domain. The goal of this project is to develop a multi-source masked language model for audio, where masking is performed both intra-source and inter-source. This model should be based on common quantized representations of audio, such as VQ-VAE or RQ-VAE (residual quantization).

Task: Students will be required to:

1. Train a VQ-VAE on a music domain.
2. Adapt a transformer architecture on the quantized domain, training with masked modeling on multi-source audio data, using the Slakh dataset or the in-house pre-separated MTG-Jamendo dataset.
3. Benchmark the performance of the model with respect to MSDM qualitatively on the generative tasks and quantitatively on the separation task.
4. (Advanced) Generalize the masked language model to the RQ-VAE quantization.

References:

1. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
   https://arxiv.org/abs/1810.04805

2. VQ-VAE and RQ-VAE.
   https://arxiv.org/abs/1711.00937
   https://arxiv.org/abs/2203.01941
3. Multi-Source Diffusion Models (MSDM).
   https://arxiv.org/abs/2302.02257

Additional Notes: Students should have a good understanding of Masked Language Models and VQ-VAE methodologies to successfully complete this project.

Enhancing Perceptual Quality in Audio Models (**Projects C**)

1. **Open-source implementation of Harmonic Convolutional Networks**

   Problem Statement: Convolutional neural networks (CNNs) have shown great success in image recognition and generation. However, their effectiveness in audio processing is not as strong, and current network architectures for audio do not show strong evidence of capturing natural audio priors. The paper "Deep Audio Priors Emerge From Harmonic Convolutional Networks" (ICLR 2020) introduces Harmonic Convolution, which utilizes the harmonic structure in audio signals to help deep networks capture audio priors. The goal of this project is to implement, replicate, and open-source the code for performing harmonic convolutions as described in the paper, and compare its performance to traditional 1D convolutions in known audio architectures.

   Task: Students will be required to:

   1. Reproduce the Harmonic Convolution as described in the "Deep Audio Priors Emerge From Harmonic Convolutional Networks" paper, since the original code is unavailable.
   2. Develop an open-source implementation of Harmonic Convolution.
   3. Integrate the Harmonic Convolution module into state-of-the-art architectures for audio tasks such as source separation or music generation (especially diffusion models such as MSDM).
   4. Conduct comparison studies with 1D convolutions, assessing the performance of the modified architectures compared to their regular operation.

   References:

   1. Deep Audio Priors Emerge From Harmonic Convolutional Networks. https://iclr.cc/virtual_2020/poster_rygjHxrYDB.html
   2. Multi-Source Diffusion Models (MSDM). https://arxiv.org/abs/2302.02257

   Additional Notes: Students should have a good understanding of convolutional neural networks and audio processing to successfully complete this project. Experience in implementing novel deep learning architectures, conducting performance comparison studies and state-of-the-art diffusion models for audio will be helpful.

---

2. **DDSP for source separation and generative modeling**

   Problem Statement: Traditional neural models of audio operate in either the time or frequency domain. The Differentiable Digital Signal Processing (DDSP) library introduces a differentiable parametric synthesizer that can be trained end-to-end with a neural network. This project aims to explore the use of DDSP for music source separation and state-of-the-art generative modeling tasks.

   Task: Students will be required to:

   1. Study the DDSP paper and understand the concept of a differentiable parametric synthesizer and its integration with neural networks.
   2. Implement music source separation using a regressor-based architecture with DDSP, taking advantage of its ability to generate clean sounds without typical artifacts found in music source separation models.
   3. (Advanced) Build a standard audio or a multi-source audio diffusion model whose gradient is defined over the parameter space of the DDSP, with the aim of improving perceptual quality.

   References:

   1. DDSP: Differentiable Digital Signal Processing. https://arxiv.org/abs/2001.04643
   2. DDSP GitHub Repository. https://github.com/magenta/ddsp
   3. Multi-Source Diffusion Models for Simultaneous Music Generation and Separation (MSDM). https://arxiv.org/abs/2302.02257

   Additional Notes: Students should have a strong background in audio processing and deep learning. Experience in implementing novel deep learning architectures and working with audio synthesis or generative models is beneficial.

Source Separation (**Project D**)

Universal Sound Separation via Diffusion Models

Problem Statement: Universal sound separation aims to separate audio mixtures containing arbitrary sounds without labels for the sources present in the mix. State-of-the-art techniques, such as permutation invariant adversarial training, have shown improvements in this task. This project will explore whether generative modeling, specifically multi-source diffusion models, can further improve the performance of universal sound separation.

Task: Students will be required to:

1. Familiarize themselves with the field of universal sound separation, permutation invariant training, and relevant state-of-the-art techniques, such as the method presented in "Adversarial Permutation Invariant Training for Universal Sound Separation."
2. Train a multi-source diffusion model on the FUSS dataset, the standard dataset for supervised universal sound separation.
3. Perform inference using the Dirac separator introduced in "Multi-Source Diffusion Models for Simultaneous Music Generation and Separation," while making the diffusion model agnostic to source orderings.
4. Investigate permutation augmentations at training time and explore more advanced permutation invariant neural architectures.

References:

1. Adversarial Permutation Invariant Training for Universal Sound Separation. https://arxiv.org/abs/2210.12108
2. FUSS Dataset. https://zenodo.org/record/3694384
3. Multi-Source Diffusion Models for Simultaneous Music Generation and Separation. https://arxiv.org/abs/2302.02257

Additional Notes: Students should have a strong background in audio processing, deep learning, and experience with generative models. Familiarity with diffusion models and permutation invariant training will be beneficial for successfully completing this project.

Choose your own (**Project E**)

If none of the above whets your appetite, you can propose your own idea for a project. However, your proposal must be approved by us.

In order to propose a new project, you are required to write a well motivated 1-page description (references can be put on a second page). This should include references, describe its feasibility and data availability, describe the expected outcome, the possible risks, what is your intended line of attack, and state in what way it tackles an interesting and new problem in the field of deep learning.

The choose-your-own project must contain aspects of originality. Simply reproducing the results of existing work is not sufficient and will not be approved. Submitting projects from previous exams is also not allowed.

If you choose this path, please send your proposal to: rodola@di.uniroma1.it

Please use the subject “[DLAI23 projects] Project name”.

Projects for the Deep Learning Course
Computer Graphics Oriented Applications

Filippo Maggioli

May 2023

Contents

Chapter 1

Background Knowledge

The projects proposed in this report require at least some knowledge about implicit representations and/or procedural texturing. A brief recap of the required background knowledge is presented in this section.

1.1 Implicit Representations

In computer graphics, there are various possibilities of representing 3D models, depending on the type of application. The most used is the triangular mesh representation, where the surface is discretized by mean of many flat triangles. While this representation is perfect for high-performance applications such as videogames and generalizes well for representing virtually any shape, it suffers from finite resolution and high memory consumption when it comes to representing smooth surfaces.

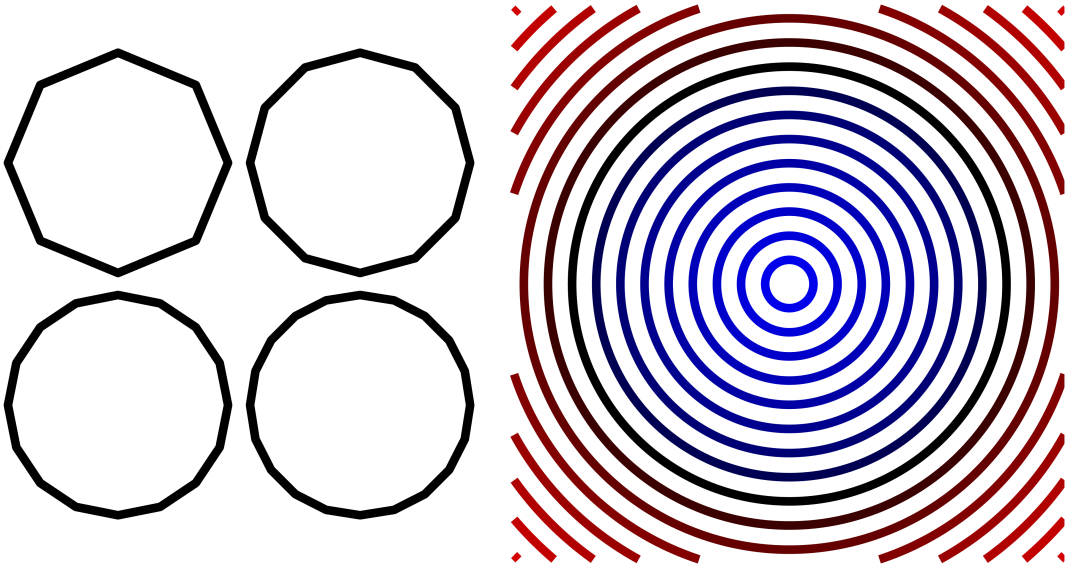

Figure 1.1: Comparing circles plotted at increasing discrete resolution with the signed distance field describing the same circle. The red zone is outside the circle, the blue zone is inside and the black zone approaches the exact outline of the circle.

Implicit representations are a way to encode surfaces within mathematical equations. Instead of asking where the surface is located in 3D space, we ask, for every point in $\mathbb{R}^3$, how far we are from the

surface. This can seem quite expensive in terms of memory, but just think of the simple example of a sphere. A (reasonably) smooth sphere is represented with 10k points and 30k triangles, and still when we move close we can see the flat surface of the triangles. The distance from a sphere with radius r and centered at $\boldsymbol{c}$ is defined at any point $\boldsymbol{p}$ as $\|\boldsymbol{p} - \boldsymbol{c}\|_2 - r$.

Implicit representations are getting popular not only because of their elegance and simplicity, but also because they are very useful in representing surfaces for a variety of applications, ranging from fluid dynamics to geometric deep learning. Moreover, they can be easily rendered using a technique called sphere tracing, or ray marching (https://iquilezles.org/articles/raymarchingdf/), which has the great advantage of being way faster than usual path/ray tracing, but achieves much better quality than the triangle rasterization generally used in high-performance applications.

Formally speaking, an implicit surface is represented by a function $\varphi : \mathbb{R}^3 \to \mathbb{R}$, called *Signed Distance Field* (or *SDF*). This function can be interrogated in any point of $\mathbb{R}^3$ and returns the signed distance from the surface. The distance (which in general should only be positive) is signed because the sign indicates whether or not the point is inside the surface. A positive sign indicates the point is outside the surface, where a negative sign indicates the point is inside. The surface is completely determined by the set of points $\boldsymbol{x}$ such that $\varphi(\boldsymbol{x}) = 0$.

Particularly remarkable properties of an SDF are:

- the gradient is unitary everywhere, that is $\|\nabla\varphi(\boldsymbol{x})\|_2 = 1$ for all $\boldsymbol{x} \in \mathbb{R}^3$;
- the normal of the surface is defined by $\nabla\varphi$ at the surface points;
- given a point $\boldsymbol{p} \in \mathbb{R}^3$, the nearest point $\boldsymbol{x_p}$ that lies on the surface is given by $\boldsymbol{x_p} = \boldsymbol{p} - \varphi(\boldsymbol{p})\nabla\varphi(\boldsymbol{p})$.

1.2 Procedural Texturing

Textures are a smart solution to artificially add detail to a surface. In general, the idea is that each triangle of a triangular mesh is mapped to a certain portion of a square image, and the render engine draws the pixels of that region onto the triangles, essentially adding extra resolution and detail to the flat surface. This technique can be also used to deflect light in different ways, to create the illusion that the surface has a more detailed depth and resolution.

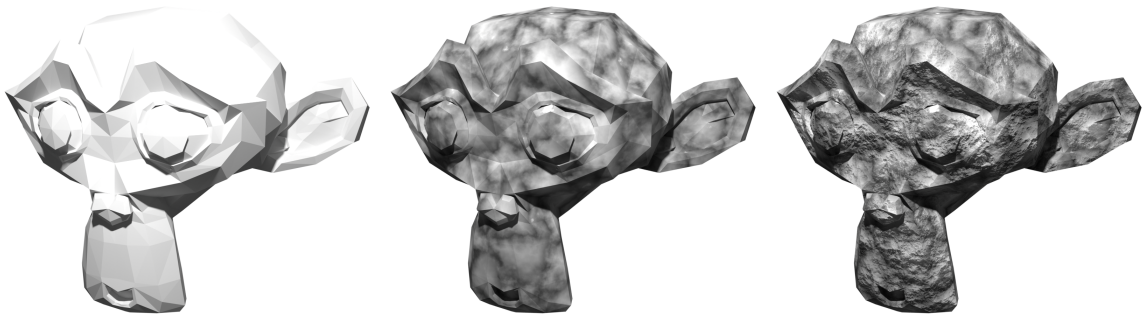

Figure 1.2: Comparing a simple surface (left) with the same surface, but colored with a texture (middle), and with the added bump map for artificial adding details. On the middle frame, the edges of the triangles are still all visible, and the flat surfaces are easily identifiable, whereas the last frame shows a noticeable amount of additional detail, but no triangles have been added.

This way of adding detail to a surface introduces a variety of problems. First of all, the resolution of the detail is still finite, as it depends on the number of pixels composing the texture. Moreover, not all triangles are the same size and shape, hence the difference must be taken into account when computing the mapping, or we would introduce distortions and non-homogeneous pixel densities. Another kind of problem that could arise is that adjacent triangles in 3D could (will) be non-adjacent in 2D, which

would generate visual artifacts at the edges. Finally, texture design is a really time-consuming process for artists, which is made even harder when considering the previous issues.

Procedural texturing is a way to overcome all these problems at once. Instead of defining the additional details on a square image and map each triangle to that image, we provide an analytic (or algorithmic) formulation of the detail directly in $\mathbb{R}^3$. Formally, we define our procedural texture as a function $\boldsymbol{\xi} : \mathbb{R}^3 \to \mathbb{R}^3$, which maps each point in 3D to a color. More in general, our function will be $\boldsymbol{\xi} : \mathbb{R}^3 \to \mathbb{R}^n$, as we can define many other additional features, such as transparency, reflectivity, emission, light scattering, etc.

In this way, when we draw a pixel, we use the 3D position of that pixel to compute the extra detail for our final rendering. The example in Figure 1.2 is obtained in exactly this way, designing the function in 5 minutes without any artistic skills, with the extra benefits that the same formula can be reused with virtually any shape for which we want similar pattern, and that tuning a bunch of parameters would introduce a lot of variety.

Besides all these advantages, there is another point for procedural textures: they can be used with SDFs. Implicit surfaces can be arbitrarily complicated, and there is no guarantee that a mapping to a plane can be easily defined (or defined at all). Furthermore, they are not made up of triangles that can be mapped one by one onto an image. But having a function that defines the color at each point in 3D space, we can add as much detail as we desire.

Chapter 2

Texture Synthesis

Affordable

In many applications, textures does not require to be applied to general surfaces, but they are just used to add detail to a planar surface (think of a wall, a plain, a body of water, etc.). When the surface is too large with respect to the texture, we can either stretch the texture to fit the plane or tiling it. In the first case, the result is a very low-resolution low-quality pattern, whereas in the latter we produce visualization artifacts like the one presented in Figure 2.1.

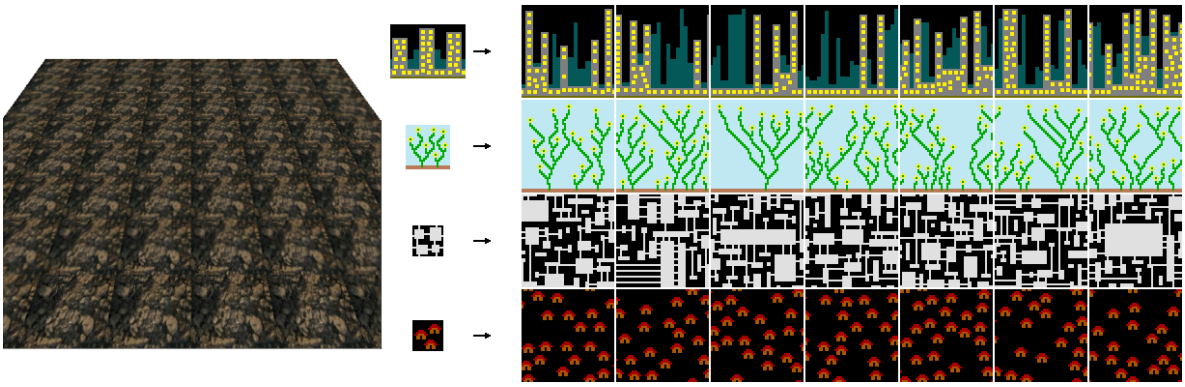

Figure 2.1: On the left, standard texture tiling: the repeated pattern is really noticeable and produces a visually non appealing result. On the right, sample based pattern extension: the resulting image is larger of the input and presents the same pattern, but in a non repetitive fashion.

However, there are some clever solutions that allow for taking a small sample image and producing a larger texture with a recognizable pattern, but without repetitions or stretches. Nevertheless, the problem is difficult enough for these methods to be very slow and/or error prone.

The goal of this project is to develop a method that accomplish this task taking advantage of deep learning techniques. There are two possible directions:

- a data-driven approach that uses a variety of examples to learn how to enlarge textures without repeating patterns;
- a data-free approach that given a small sample generates a larger pattern.

Chapter 3

Dimensionally Extending Textures

Challenging

Procedural textures are very attractive, and they can produce a very large variety of astonishing quality results. However, they are not always the definitive answer, as some patterns can be hard (or even impossible) to capture with analytic or algorithmic formulation. For some applications, the parametrization of triangles is just better.

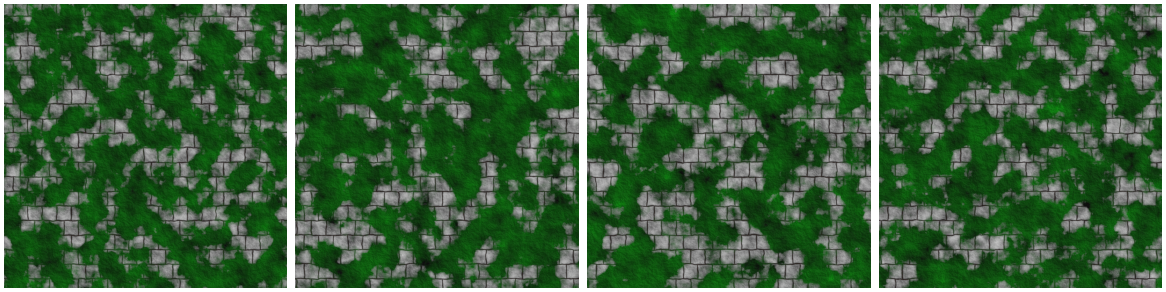

Figure 3.1: 2D slices of a 3D texture. The texture is defined in a 3-dimensional space, but each frame is obtained by taking a single 2-dimensional slice along the Z axis. Each texture is very different from the others, but the pattern is highly recognizable.

The idea behind this project is to learn a 3-dimensional pattern from a 2-dimensional one. We know that procedural textures are often described as a function assigning a color to each point of $\mathbb{R}^3$, and the resulting coloration of a shape would be given by the color at the 3D positions that lie on the surface. For instance, in Figure 3.1 we can see four 2-dimensional slices of the cube $[-1, 1]^3$, taken along the Z axis. Each texture is wildly different, but they all represents the same thing and the pattern can be immediately recognized.

What we want to obtain with this project is, given a 2-dimensional pattern (or a small collection), to obtain a function $\boldsymbol{\xi} : \mathbb{R}^3 \to \mathbb{R}^3$ that generalizes that pattern to the 3D space, effectively realizing a 3-dimensional texture that, as we take any slice, produces an image that is different from the original one, but that at the same time has a similar pattern.

Chapter 4

Surface Parametrization

Challenging

Implicit representations are really game changing in many applications, but they are far from being a one-fits-all solution. One of their most significant downsides is the lack of a parametrization. Informally speaking, the parametrization of a shape is an invertible function $\boldsymbol{\varphi} : [0,1]^2 \to \mathbb{R}^3$ that deforms the plane until it fits the surface.

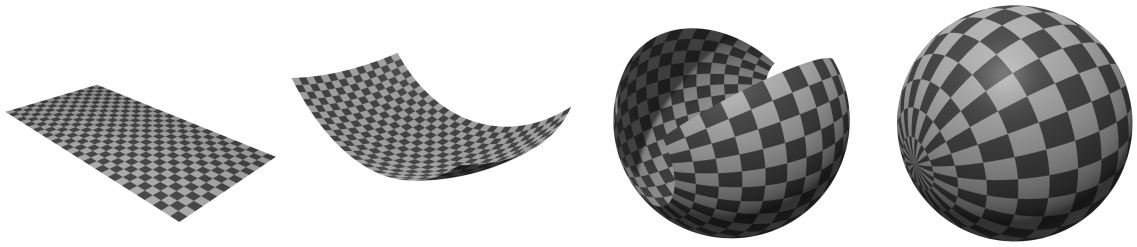

Figure 4.1: Folding a plane into a sphere. The plane is bent and deformed until it becomes a sphere. Such a deformation is impossible to achieve with real-world materials, but it is mathematically well-defined.

For some basic shapes, the parametrization is well-defined and known. For example, the folding of the sphere shown in Figure 4.1 can be achieved with

$$\begin{aligned} (u,v) &\mapsto (\sin(u)\cos(v),\ \sin(u)\sin(v),\ \cos(u)) \\ (x,y,z) &\mapsto \left(\arccos(z),\ \arctan\left(\frac{y}{x}\right)\right) \end{aligned} \tag{4.1}$$

However, for general surfaces finding a parametrization is not trivial, and it is often even not possible. For this project, we aim to find a network that, given some signed distance field $\psi : \mathbb{R}^3 \to \mathbb{R}$ identifying a surface, optimizes for an invertible function $\boldsymbol{\varphi} : [0,1]^2 \to \mathbb{R}^3$ that parametrizes the same surface.

Chapter 5

Composing Primitives

Very challenging

There are a lot of basic shapes that can be defined implicitly via analytic functions. However, not all the shapes have an analytic formulation, and thus crafting or editing a surface represented with a signed distance field is still far from being an easy task.

Figure 5.1: Sphere tracing rendering of a scene crafted by composing different implicit surfaces. The descriptive power of the composition is high enough to represent even some simple animations.

Nonetheless, talented artists can still achieve good results in representing complex scenes by using only simple primitive shapes such as spheres, cubes and cylinders. In general, composition is achieved by taking unions, intersections or differences of two or multiple shapes, which is done by using operators such as min and max. For better results artists tend to use the smoothed versions of these operators (https://en.wikipedia.org/wiki/Smooth_maximum). By introducing a time variable, it is also possible to deform, move and modify the objects in the scene, resulting in smooth and nice animations (see Figure 5.1 for reference).

Anyway, composing primitives is very challenging and time-consuming, and when the scene becomes complex even a small change could affect the entire composition in ways that are difficult to predict. Within this project, there are two possible directions:

- given a surface represented as a triangular mesh, finding the composition of primitive shapes that results in an implicit representation of the same surface;
- supposing to have a composition of primitives that already fits a surface, finding the transformations that animate the surface to the target pose.