Furthering FlowSSMs Models

Greg DePaul, Ryan A. Anderson, Shizhe Xu, Suresh Kumar Choudhary,

July 2023

Abstract

We survey the necessary architectural develops that lead to the development of FlowSSM [1]. We also engineer a sequence model to perform the integration of the identified flow trajectories surrounding a mean shape.

1 Occupancy Probability Function

Our work began by exploring the origins of IM-Net [2]. Formally the architecture is as follows. IM-Net is a function

$$f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \to [0, 1]$$

that takes a parameter θ. This parameter θ is formally the weights and biases of the following neural network:

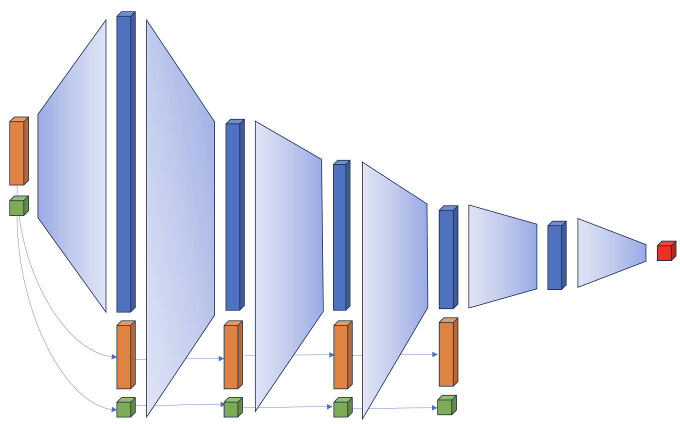


Figure 1: IM Net

The output of the function is mean to represent the following:

$$f_\theta(x, \text{latent space of shape } S) = \mathbb{P}(x \in S)$$

This model is thusly named the occupancy probability model, as stated in [1].
In order to train this network, we need to define the following quantities:

- $S :=$ set of points sampled from the target shape
- $\mathcal{F}(p) := \begin{cases} 0 & \text{if point } p \text{ is outside of the shape} \\ 1 & \text{otherwise} \end{cases}$
- To compensate for the density change, we assign a weight w_p to each sampled point p, representing the sampling density near p.

Now we can define the loss function to be :

$$\mathcal{L}(\theta) := \frac{\sum_{p \in S} |f_\theta(p) - \mathcal{F}(p)|^2 \cdot w_p}{\sum_{p \in S} w_p}$$

And therefore we optimize over this very loss function.

2 How to Train the Latent Space?

2.1 Procrustes Alignment

In order to develop a latent space, we need to consider the relative scope of this task. Do we want our shapes to be translationally invariant? Rotationally invariant? In order for a machine learning model to understand such a task, there would need to be the following changes:

- Develop an architecture that is invariant under rotations or translations (Very difficult theoretically)
- Augment the training set to account for all possibly achievable translations and rotations (Very expensive)

For the purposes of this project, we simply use a mean shapes (developed using traditional methods or more interesting methods [3]), and then train only on these aligned shapes.

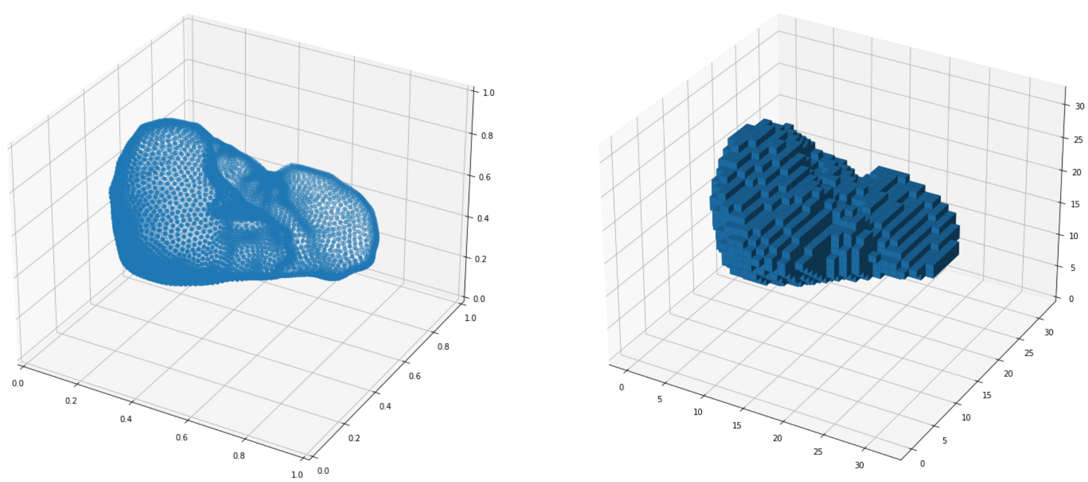
2.2 Convert to Voxel Representation

We then take every aligned shape and develop a voxel representation of that shape.

Definition 2.1. *A voxel representation is a binary representation of a shape S within $2^{d\times d\times d}$. There is a surjection from the bounding box surrounding the shape S to the voxel representation. We denote d as the dimension of the voxel representation.*

Note 2.2. *The code I have developed for creating a 64 dimensional voxel representation takes quite a long time. I am confidant my code could be improved. Software packages like Open3D and Trimesh "offer" voxel conversions, but these tend not to be aligned to any bounding box, which of course is a problem.*

Below, you will see a Voxel representation for the liver of dimension $d = 32$.



(a) Original Point Cloud

(b) Voxelized Mesh

Figure 2: Convert Meshes in $\mathbb{R}^3$ to Voxel Representations $2^{d\times d\times d}$

2.3 Use an Autoencoder

Once we have a voxel representation, we then take that representation through an autoencoder.

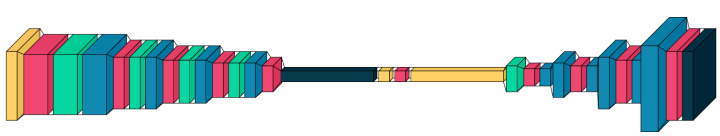


Figure 3: 32 Bit Voxel Encoding

3 Transition from Probability to Deformation Velocity

3.1 Differential Equation Formulation

Within [4], they restate this problem of identifying the boundary via integration over a flow field. We've taken the liberty of reworking this formulation as the following boundary value problem: We need to learn a flow function $v(x,t) := f_\theta(x, tz)$ such that:

$$\begin{cases} x' = v(x,t) \\ x(0) = \text{ template shape} \\ x(1) = \text{ target shape} \end{cases}$$

3.2 ShapeFlow Method

The template shape described as $x(0)$ is generated by the method described in [5]. Only one template is used to represent a given shape space.

3.3 Euler's Method Formulation

We recall Euler's first order method of integrating as

$$x_{n+1} = x_n + \Delta_t \cdot v(x_n, t)$$

Therefore, we can rewrite this as the following computational block:

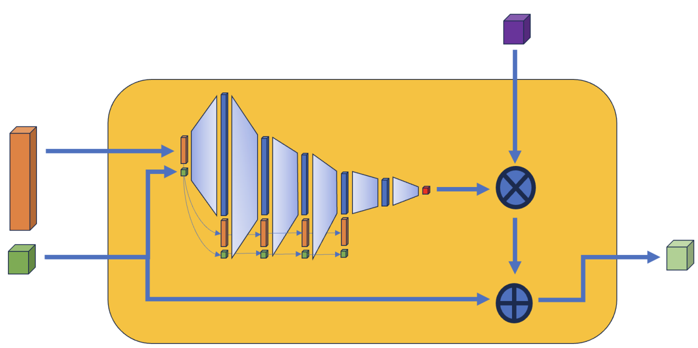


Figure 4: Euclid Block

Then we can allow a certain number of these blocks to be sequenced together, with an accompanying parameter of time:

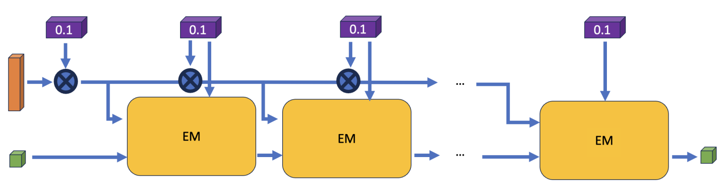


Figure 5: Sequence of Euclid Blocks

This sequence is our final model of consideration.
To measure the performance of this model, we use the Chamfer Distance:

Definition 3.1. *The correspondence-free, symmetric point-set to point-set Chamfer Distance $\mathcal{C}$ between randomly sampled surface points of the target $P_i \subset X_i$ and deformed, sampled surface points of the template $P_\Phi = \Phi^i(P_\mathcal{T} \subset \mathcal{T}, 1)$:*

$$\mathcal{C}(P_i, P_\phi) = \frac{1}{2|P_i|} \sum_{x_i \in P_i} \min_{x \in P_\Phi} \|x_i - x\|_2 + \frac{1}{2|P_\Phi|} \sum_{x_i \in P_\Phi} \min_{x \in P_i} \|x_i - x\|_2$$

4 Comparison to Tamaz / Lüdke Work

	Model	Average Symmetric Surface	Average Chamfer
Femur	FlowSSM		
	Our Work		$0.28572568 \pm 0.048069667$
Liver	FlowSSM		
	Our Work		$0.45721212 \pm 0.099601954$

References

[1] Tamaz Amiranashvili, David Lüdke, Hongwei Li, bjoern menze, and Stefan Zachow. Learning shape reconstruction from sparse measurements with neural implicit functions. In *Medical Imaging with Deep Learning*, 2022.

[2] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling, 2019.

[3] Chiyu "Max" Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. Shapeflow: Learnable deformations among 3d shapes, 2021.

[4] David Lüdke, Tamaz Amiranashvili, Felix Ambellan, Ivan Ezhov, Bjoern Menze, and Stefan Zachow. Landmark-free statistical shape modeling via neural flow deformations. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2022*, volume 13432, 2022.

[5] Chiyu "Max" Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. Shapeflow: Learnable deformations among 3d shapes, 2021.