Auto-encoding meshes of any topology with the current-splatting and exponentiation layers

Alexandre Bône, Olivier Colliot, Stanley Durrleman

Auto-encoding meshes of any topology with the current-splatting and exponentiation layers

Alexandre Bône Olivier Colliot Stanley Durrleman
The Alzheimer's Disease Neuroimaging Initiative
ARAMIS Lab, ICM, Inserm U1127, CNRS UMR 7225, Sorbonne University, Inria, Paris, France
{alexandre.bone, olivier.colliot, stanley.durrleman}@icm-institute.org

Abstract

Deep learning has met key applications in image computing, but still lacks processing paradigms for meshes, i.e. collections of elementary geometrical parts such as points, segments or triangles. Meshes are both a powerful representation for geometrical objects, and a challenge for network architectures because of their inherent irregular structure. This work contributes to adapt classical deep learning paradigms to this particular type of data in three ways. First, we introduce the current-splatting layer which embeds meshes in a metric space, allowing the downstream network to process them without any assumption on their topology: they may be composed of varied numbers of elements or connected components, contain holes, or bear high levels of geometrical noise. Second, we adapt to meshes the exponentiation layer which, from an upstream image array, generates shapes with a diffeomorphic control over their topology. Third, we take advantage of those layers to devise a variational auto-encoding architecture, which we interpret as a generative statistical model that learns adapted low-dimensional representations for mesh data sets. An explicit norm-control layer ensures the correspondence between the latent-space Euclidean metric and the shape-space log-Euclidean one. We illustrate this method on simulated and real data sets, and show the practical relevance of the learned representation for visualization, classification and mesh synthesis.

1. Introduction

Deep learning has met key applications in image computing, but still lacks processing paradigms for mesh data. Understood as collections of elementary geometrical parts such as lines in 2D or triangles in 3D, meshes are a compact and natural representation for geometrical data. The inherent difficulty with meshes is that they do not have regular structure: two meshes might be similar in their 3D geometry yet very different in their parametrization – e.g. composed of varying numbers of elementary triangles or connected components of such triangles, contain holes breaking their topology, or bear high levels of geometrical noise. Practical tasks (such as regression or classification) remain however ultimately the same regardless of the data type, which leads to the question: is it possible to simply adapt image deep learning paradigms to work with meshes?

A first challenge lies in building an "embedding" layer, able to represent input meshes with irregular structure into vectors of fixed dimension, that can then be processed by any classical network architecture. Focusing on the case of point clouds, [15, 24] introduce specific architecture for object classification, part segmentation, and semantic segmentation. In the case of connected meshes, intrinsic operators are defined in [21], as well as in the geometric deep learning papers [2, 17, 18] surveyed by [3]. Considering mesh points as graph nodes, they introduce convolution-like operators able to compute feature vectors from sub-graphs extracted at a fixed number of seed vertices. Those techniques are well-suited to process model-based graphs such as molecular structures or computer-aided designs because local topologies carry information, but large receptive fields (and high computational power) would be required to process noisy data-driven graphs such as connectomes or segmented organs. Global graph representations are extracted in [26], but only after convergence of a costly iterative procedure. An opposed trend in the literature advocates for transforming input 3-dimensional shapes into either binary volumetric images [25, 32] or series of textured 2D images obtained by selecting a set of viewpoints [28]. If those approaches might be computationally intensive, the opportunity to seamlessly use well-understood deep network architectures is a particularly appealing asset, which helped them achieving top performance for object recognition. To the best of our knowledge, those image-transform methods however either ignore the information offered by the normals of the meshes, or take them indirectly into account through some arbitrary and fixed illumination. If this is not

the case of signed distance functions used in [4, 5, 27] to represent the environment from depth sensor signals [20], the computational complexity to estimate such maps from fully-determined shapes seems too elevated to be used as a feature extraction layer – which should typically rather consist in a forward operation.

A second challenge, reciprocal to the first, is to build an output layer able to generate meshes. Going beyond the generation of "shapes" as image-like structures, [7] synthesizes point clouds. For higher-level shape primitives like surfaces, a shared paradigm in [9, 11, 12, 13, 23, 29, 31] consist in linearly deforming the vertices of a template mesh, while keeping its connectivity unchanged. In [13, 31], the template is fixed to a generic ellipsoid. In [9], it is assembled from a bank of parts, when in [23] several templates from a large bank are linearly combined. A specific face template and its allowed linear deformation modes are fixed a priori in [29], when those are learned in [11, 12]. A shared assumption is that the deformation should not modify the template mesh topology: those papers either assume all shapes isomorphic to the sphere [13, 31], rely on an upstream classifier [9, 23], or restrict to a single class [11, 12, 29]. This central topological hypothesis is systematically enforced in an extrinsic manner through dedicated regularity terms, and is only verified at convergence.

Based on the theory of currents [30], we introduce in Section 2 the current-splatting layer, which embeds meshes in a metric space without any assumption on their topology. The normals are directly and compactly taken into account in this transformation, capturing maximal information for the downstream network to process. The introduced metric space naturally offers a loss function that measures the similarity between any two meshes. In [6, 16], the authors define the so-called exponentiation layer for imaging data, which smoothly deforms a template image to generate a new one with an intrinsic diffeomorphic control on its topology. We adapt this layer to meshes in Section 3, enabling the synthesis of smooth and regular objects from the upstream network. In Section 4, we take advantage of those input and output layers to devise a variational auto-encoding architecture, which we interpret as a generative statistical model for meshes. An explicit norm-control layer ensures the correspondence between the latent-space Euclidean metric and the shape-space log-Euclidean one. Experimental results on varied data sets are reported in Section 5.

2. Meshes seen as splatted currents

A mesh y is understood in this paper as a homogeneous collection of elementary geometrical parts such as points, segments, or triangles. A standard description is therefore to write y as a list of points, along with a list of segments or triangles that we call connectivity, point clouds being seen as degenerated meshes with no connectivity. We focus in the rest of the article on non-degenerated meshes, and more specifically on surfaces of $\mathbb{R}^3$ in the next sub-section to introduce the current theory. All notations and concepts can however be adapted to collections of segments or points.

2.1. Continuous theory

The pragmatic description of a surface mesh as a finite collection of triangles can be understood as the dicretization of an infinite set of points $x \in \mathbb{R}^3$ with infinitesimal normal vectors $n \in \mathbb{R}^3$ attached to them. The geometric measure theory [19] studies those objects called rectifiable sets under loose piece-wise smoothness hypotheses: the strategy is to embed them in a functional Hilbert space, where desirable basic operators such as addition, subtraction, averaging or scalar product will be naturally defined. Given a space Ω of square-integrable vector fields $\omega : \mathbb{R}^3 \to \mathbb{R}^3$, we associate to the rectifiable surface y the mapping defined by:

$$\mathcal{C}^*(y)(\omega) = \int_y \omega(x)^\top \cdot n(x) \cdot d\sigma(x) \tag{1}$$

where x denotes a parametrization of y, $d\sigma(x)$ an infinitesimal surface element and $(.)^\top$ the transposition operator. Equation (1) is invariant under parametrization change, hence the mapping $\mathcal{C}^*(y)$ is a linear form on Ω. We call currents such linear forms, which are elements of Ω^*, the dual space of test fields. Following [30], we further assume that Ω is a reproducing kernel Hilbert space with kernel $K_\Omega : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ and scalar product denoted $\langle .|. \rangle_\Omega$. The Riesz representation theorem gives the existence of some $\mathcal{C}(y) \in \Omega$ such that $\mathcal{C}^*(y)(.) = \big\langle ., \mathcal{C}(y) \big\rangle_\Omega$. Combining equation (1) and the reproducing property $\omega(x) = \big\langle \omega, K_\Omega(x, .) \big\rangle_\Omega$, this representant can be identified:

$$\mathcal{C}(y)(.) = \int_y K_\Omega(x, .) \cdot n(x) \cdot d\sigma(x). \tag{2}$$

Since $\big\langle \mathcal{C}^*(y), \mathcal{C}^*(y') \big\rangle_{\Omega^*} = \big\langle \mathcal{C}(y), \mathcal{C}(y') \big\rangle_\Omega$, this inner-product on currents finally writes:

$$\int_y \int_{y'} n'(x')^\top \cdot K_\Omega(x, x') \cdot n(x) \cdot d\sigma(x) \cdot d\sigma'(x') \tag{3}$$

which induces the distance metric:

$$d_\Omega(\mathcal{C}, \mathcal{C}')^2 = \langle \mathcal{C}, \mathcal{C} \rangle_\Omega + \langle \mathcal{C}', \mathcal{C}' \rangle_\Omega - 2 \cdot \langle \mathcal{C}, \mathcal{C}' \rangle_\Omega. \tag{4}$$

2.2. Practical discrete case

In practice, we propose to choose the simple radial Gaussian kernel of radius $\sigma_\Omega > 0$ for K_Ω. Moreover, meshes are represented as finite collections of T elements. Under those hypotheses, equation (2) becomes:

$$\mathcal{C}(y)(x) = \sum_{k=1}^{T} \exp \frac{-\big\|x - c_k\big\|^2_{\ell^2}}{\sigma_\Omega^2} \cdot n_k \tag{5}$$

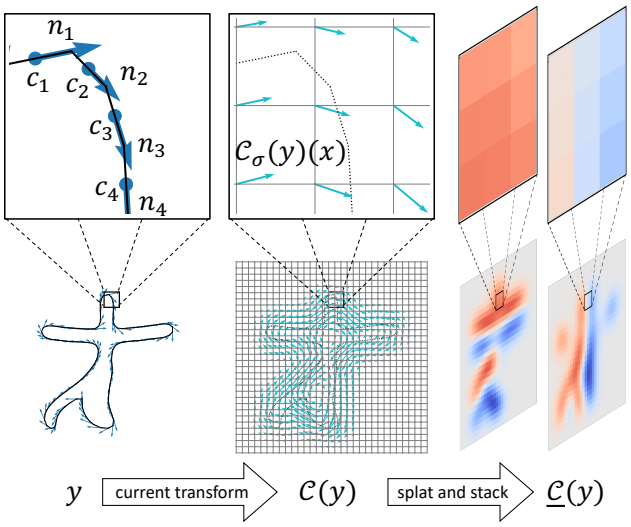

Figure 1. Current-splatting mechanics. The input mesh is first transformed in a vector field, which is then discretized on a fixed grid to form a d-dimensional image. If a topologically simple object was selected for the sake of clarity, note that any other topology could be similarly treated.

where x is a point of $\mathbb{R}^d$, the c_k and n_k respectively are the centers and normals of the triangles composing y. In the case of a collection of segments, the $(n_k)_k$ are the tangent vectors. In the same manner, equation (3) becomes:

$$\Big\langle \mathcal{C}(y), \mathcal{C}(y') \Big\rangle_\Omega = \sum_{k=1}^{T} \sum_{l=1}^{T'} \exp \frac{-\left\| c'_l - c_k \right\|_{\ell^2}^2}{\sigma_\Omega^2} \cdot n_k^\top n'_l. \quad (6)$$

Given a discrete grid g_Ω of $\mathbb{R}^d$, we finally define the splatted current $\underline{\mathcal{C}}(y)$ as the d-channel image resulting from the discretization of $\mathcal{C}(y)$ on the grid g_Ω.

Note the following properties of the current-splatting transform: (i) it does not assume any particular topology of the meshes, (ii) it is invariant under parametrization change, (iii) it captures the proximity relationships between elements, (iv) it captures the orientation information encoded by the normals of the triangles (or tangents of the segments). All those properties are achieved at the cost of smoothing out all geometrical features of characteristic radius inferior to σ_Ω, which can on the other hand be useful to filter out geometrical noise.

Architecture. The architecture of the current-splatting layer is presented by Figure 1. The input mesh y is first transformed in the function $\mathcal{C}(y)$, before being splatted into a d-channel image $\underline{\mathcal{C}}(y)$.

Hyper-parameters (σ_Ω, g_Ω). The characteristic length σ_Ω should ideally be larger than the noise to eliminate, and smaller than the signal to capture. In practice, the grid g_Ω is obtained by uniformly dividing a bounding box containing all the considered meshes. A good heuristic is to choose a spacing between each node approximately equal to $\sigma_\Omega/2$.

3. Meshes seen as deformations of a template

A mesh-generating layer is by essence an output layer, and is therefore strongly linked to the loss function used to train the network. In this work, we take advantage of the current framework and use the distance metric defined by equation (4), which is advantageously free of any topological assumption. However, because of the low-pass filtering behavior of the current transform, a naive output layer synthesizing a mesh by directly predicting the position of its points would be free to generate very noisy geometries.

3.1. Continuous theory

To control the regularity of the generated meshes, we impose that they are diffeomorphic to a reference mesh y_0. This constraint suggests a method: instead of predicting a mesh y directly, we want the network to generate a diffeomorphism ϕ, before computing the deformed template $y = \phi \star y_0$. As in [1], we construct diffeomorphisms by following the streamlines of static smooth vector fields $v \in V \subset C_0^\infty(\mathbb{R}^d, \mathbb{R}^d)$ during the time segment $[0, 1]$:

$$\Phi(v) = \phi_1 \quad \text{where} \quad \partial_t \phi_t = v \circ \phi_t, \quad \phi_0 = \mathrm{Id}_{\mathbb{R}^d}. \quad (7)$$

The mapping $\Phi : V \to \mathcal{G}_V = \{\Phi(v) \,|\, v \in V\}$ is locally invertible around the identity: similarly to [34], we define on this neighbourhood of $\mathcal{G}_V$ the "log-Euclidean" distance:

$$d_V(\phi, \phi') = \left\| \Phi^{-1}(\phi') - \Phi^{-1}(\phi) \right\|_V \quad (8)$$

which induces a distance on the corresponding neighbourhood of the orbit shape space $\mathcal{G}_V \star y_0$. We further assume that V is a reproducing kernel Hilbert space with kernel K_V. The Riesz representation theorem gives the existence of the "momenta" dual vector field $m \in V^*$:

$$v(.) = \int_{\mathbb{R}^d} K_V(x, .) \cdot m(x) \cdot d\nu(x) \quad (9)$$

where $d\nu(x)$ is an infinitesimal element of $\mathbb{R}^d$. The inner product $\langle v, v' \rangle_V$ on V can now be derived:

$$\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} m'(x')^\top \cdot K_V(x, x') \cdot m(x) \cdot d\nu(x) \cdot d\nu'(x') \quad (10)$$

which defines the norm operator $\|v\|_V = \langle v | v \rangle^{1/2}$.

3.2. Practical discrete case

In practice, we propose to choose the simple radial Gaussian kernel of radius $\sigma_V > 0$ for K_V. Moreover, the ambient space $\mathbb{R}^d$ is discretized into a grid g_V. Under those hypothesis, equations (9) and (10) write, in matrix forms:

$$\underline{v} = \underline{K}_V \cdot \underline{m} \qquad \text{and} \qquad \langle \underline{v} | \underline{v'} \rangle_V = \underline{m}^\top \cdot \underline{K}_V \cdot \underline{m'} \quad (11)$$

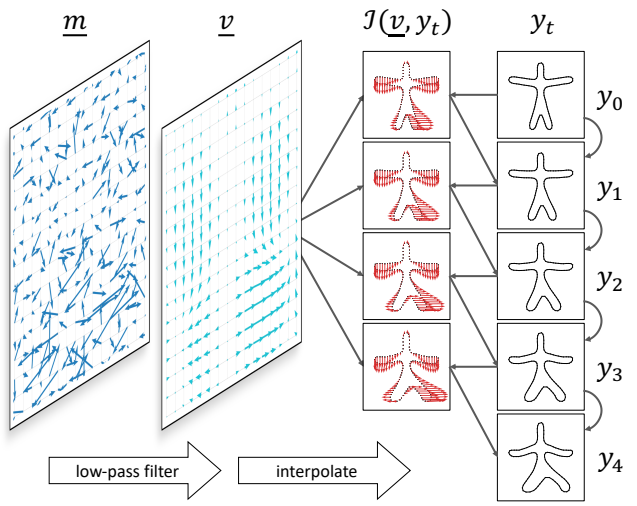

Figure 2. Exponentiation mechanics. The input d-dimensional array $\underline{m}$ is first filtered by a Gaussian convolution layer. Interpreted as a discretized velocity field, the resulting $\underline{v}$ is interpolated on the successive positions of the moving mesh y_t, which is finally updated accordingly.

where the notations $\underline{v}$ and $\underline{m}$ refer to the g_V-discretized vector fields v and m respectively. The notation $\underline{K}_V$ denotes the kernel tensor defined by, for any triplet of indices (i_0, j_0, k_0) of the grid g_V:

$$\left[\underline{K}_V\right]_{(i_0,j_0,k_0)} = \sum_{i,j,k} \exp \frac{-\left\|g_{i,j,k} - g_{i_0,j_0,k_0}\right\|_{\ell^2}^2}{\sigma_V^2} \quad (12)$$

in the case $d = 3$, easily adaptable to lower dimensions. The time segment $[0, 1]$ is uniformly discretized into T subsegments of length $dt = 1/T$. The differential equation (7) becomes, for any time index $0 \leq t \leq T-1$ and point $x_0 \in \mathbb{R}^d$:

$$x_{t+1} \;=\; x_t + dt \cdot v(x_t) \;\approx\; x_t + dt \cdot \mathcal{I}(\underline{v}, x_t) \quad (13)$$

where $\mathcal{I}(\underline{v}, x_t)$ simply denotes the bi- or tri-linear interpolation of the discretized velocity field $\underline{v}$ at location x_t.

Architecture. The architecture of the exponentiation layer is depicted by Figure 2. It takes as input a d-channel image, interpreted as a d-dimensional momentum vector field m discretized over a spatial grid g_V. This upstream stimulus $\underline{m}$ is filtered into the discrete velocity field $\underline{v}$ by a Gaussian convolution layer with kernel width σ_V, according to equations (11, 12). A recurrent residual network of length T then implements equation (13) for the template mesh $(y_t)_t$: the interpolated velocity field $\mathcal{I}(\underline{v}, y_t)$ is computed, scaled by dt, and added to the current mesh positions. The final mesh y_T forms the output of the exponentiation layer.

Hyper-parameters (σ_V, g_V, T). The notation y_0 encompasses both: (i) the positions of the points forming the mesh, which are parameters of the exponentiation layer (i.e. estimated), (ii) the mesh connectivity, which is fixed a priori. All synthesized meshes will therefore have this same topology. The characteristic length hyper-parameter σ_V should ideally be of the order of the smallest geometrical features to generate. In practice, the grid g_V hyper-parameter is obtained by uniformly dividing a bounding box containing the initial y_0. A good heuristic is to choose a spacing between each node approximately equal to $\sigma_V/2$. The number of integration steps T forms a last hyper-parameter. We chose $T = 5$ in all our experiments.

4. Meshes seen as low-dimensional codes

We take advantage of the current-splatting and exponentiation layers to devise an auto-encoding architecture, which aims at learning a low-dimensional representation of a data set of meshes $(y_i)_{i=1}^n$. Given some user-defined latent-space dimension $q \in \mathbb{N}^*$, any shape y_i will be represented in the network by a corresponding code $z_i \in \mathbb{R}^q$. Note that meshes represented by a varying number of points, segments or triangles are then homogeneously represented by simple low-dimensional vectors of the Euclidean space $\mathbb{R}^q$, where simple operations such as computing averages are naturally defined. We choose to work with a variational auto-encoder: the latent codes $(z_i)_i$ are seen as probability distributions. This allows to capture the uncertainty associated with such low-dimensional representations, and offers a statistical interpretation of the resulting architecture.

4.1. Continuous theory

Statistical model. We note D_δ a δ-parametric mapping that associates a velocity vector field $v \in V$ to any code vector $z \in \mathbb{R}^q$. We further require D_δ to be isometric, i.e. $\|z\|_{\ell^2} = \|v\|_V$. Given a data set of meshes $(y_i)_i^n$, we model the observations as random deformations of a template y_0:

$$y_i \overset{\text{iid}}{\sim} \mathcal{N}_\Omega\big(\Phi\big[D_\delta(z_i)\big] \star y_0,\, \epsilon^2\big) \;\text{ with }\; z_i \overset{\text{iid}}{\sim} \mathcal{N}\big(0,\, \lambda^2\big) \quad (14)$$

where $\epsilon > 0$ and $\lambda > 0$. The normal distribution $\mathcal{N}_\Omega$ is defined in the space of the Ω-currents, equipped with the distance metric d_Ω defined by equation (4). Equation (14) defines a mixed-effects model with fixed effects $\theta = (y_0, \delta, \epsilon, \lambda)$ and random effects $(z_i)_i$. We note respectively $p_\theta(y_i|z_i)$ and $p_\theta(z_i)$ the density functions of the two normal distributions involved in equation (14).

Variational inference. We estimate the parameters θ with a variational Bayes approach [10], which consist in minimizing the loss $\sum_{i=1}^n \mathcal{L}_{\theta,\eta}(y_i)$ given by:

$$\mathcal{L}_{\theta,\eta}(y_i) = -\int \log p_\theta(y_i|z_i) \cdot q_\eta(z_i|y_i) \cdot dz_i \\ + \mathrm{KL}\big[q_\eta(z_i|y_i) \,\big\|\, p_\theta(z_i)\big] \quad (15)$$

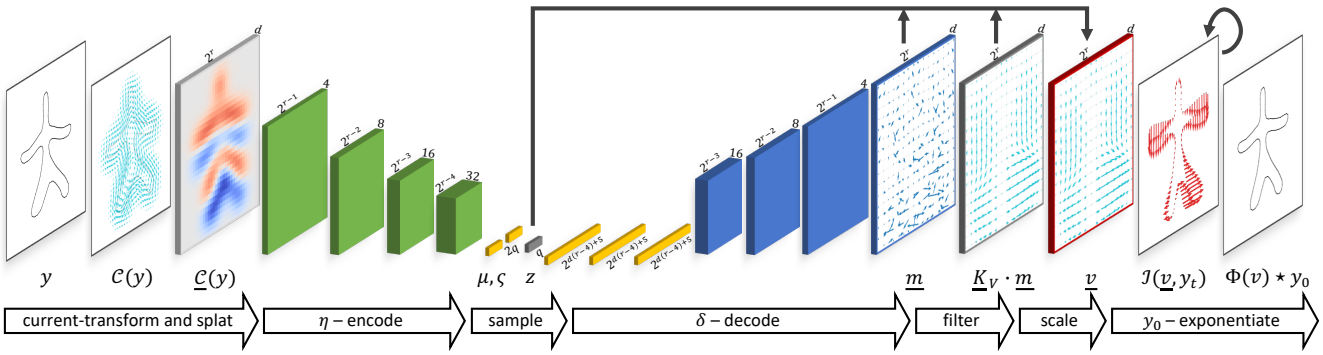

Figure 3. Architecture of the mesh auto-encoder. The current-splatting layer transforms the input mesh into a d-dimensional array, which is encoded by four convolution layers (in green) followed by a fully-connected layer (in yellow). Sampled codes are then decoded by three fully connected layers followed by three deconvolution layers (in blue). After filtering by Gaussian convolution, a scaling layer (in red) explicitly enforces the isometry of the decoding mapping. The exponentiation layer finally synthesize the output mesh.

where $\mathrm{KL}(.\|.)$ denotes the Kullback-Leibler divergence operator, and where $z_i \to q_\eta(z_i|y_i)$ is a η-parametric recognition model that approximates the true posterior $z_i \to p_\theta(z_i|y_i)$. Minimizing the loss function defined by equation (15) actually consists in maximizing a lower bound of the model likelihood, with equivalence in the perfect approximation case. We choose the recognition distribution $q_\eta(.|y_i)$ to be an uncorrelated Gaussian $\mathcal{N}(\mu_i, \varsigma_i)$, whose parameters are predicted from the current transform $\mathcal{C}(y_i)$ by the parametric mapping $E_\eta : \mathcal{C}(y_i) \to \mu_i, \varsigma_i$. The Kullback-Leibler term in equation (15) can be seen as a regularizing loss $R_{\theta,\eta}(y_i)$, easily derivable for the chosen recognition model:

$$R_{\theta,\eta}(y_i) = \frac{1}{2}\sum_{k=1}^{q}\left[\frac{\mu_{i,k}^2 + \varsigma_{i,k}^2}{\lambda^2} - \log\frac{\varsigma_{i,k}^2}{\lambda^2}\right] + \text{cst}. \quad (16)$$

The remaining term, called attachment, is approximated by drawing L samples $z_{i,l} \overset{\text{iid}}{\sim} q_\eta(.|y_i)$:

$$A_{\theta,\eta}(y_i) = \frac{1}{2}\sum_{l=1}^{L}\left[\frac{\epsilon_{i,l}^2}{\epsilon^2} + |\Omega|\cdot\log\epsilon^2\right] + \text{cst} \quad (17)$$

$$\text{with} \quad \epsilon_{i,l}^2 = d_\Omega\big(\mathcal{C}(y_i),\, \mathcal{C}(\Phi\big[D_\delta(z_{i,l})\big]\star y_0)\big)^2 \quad (18)$$

and where $|\Omega|$ is the normalization parameter for $\mathcal{N}_\Omega$. These losses are given modulo an additive constant with respect to θ and η which are jointly estimated. The high-dimensional parameters y_0, δ and η are optimized by mini-batch stochastic gradient descent. After each mini-batch, the remaining scalar parameters λ, ϵ are updated according to the closed-form solutions:

$$\lambda^2 \leftarrow \sum_{i=1}^{n}\sum_{k=1}^{q}\frac{\mu_{i,k}^2 + \varsigma_{i,k}^2}{n\cdot q}, \quad \epsilon^2 \leftarrow \sum_{i=1}^{n}\sum_{l=1}^{L}\frac{\epsilon_{i,l}^2}{n\cdot L\cdot|\Omega|}. \quad (19)$$

Remark. Assuming D_δ isometric is sufficient to achieve equality between the log-Euclidean metric defined in Section 3 and the natural ℓ^2 metric of $\mathbb{R}^q$. In other words, the Euclidean distance between the latent-space representations $(z_i)_i$ can be seen as a convenient proxy to visualize and measure the relative similarity between the corresponding data points $(y_i)_i$. In addition, the estimated template y_0 can be seen as a Fréchet average of those data points [1, 22].

4.2. Practical discrete case

As suggested in [14], in practice the encoding E_η and decoding D_δ mappings are neural networks, noted $\underline{E}_\eta$ and $\underline{D}_\delta$. The "discrete" encoder $\underline{E}_\eta$ takes g_Ω-splatted currents $\underline{\mathcal{C}}$ as inputs. In those discrete settings, the normalizer $|\Omega|$ equals the number of nodes of the grid g_Ω [8]. The decoding counterpart $\underline{D}_\delta$ outputs g_V-discretized momentum vector fields $\underline{m}$. A Gaussian convolution layer then computes the associated discrete velocity field, which is finally explicitly scaled into $\underline{v}$, enforcing the isometric assumption.

Architecture. The proposed architecture is illustrated by Figure 3. The current-splatting layer first transforms the input meshes into d-channel square or cube images of length 2^r along each axis. Four convolution layers with kernel size and stride of 2 then reduce the spatial dimension, when the number of channels successively increases from d to 4, 8, 16 and 32. A fully-connected layer of output size $2q$ completes the encoder architecture. Its output is interpreted as mean and variance parameters of the probabilistic code distribution: during training, a single realization z is sampled with the so-called reparametrization trick to ensure the backpropagation of gradients [14]. The decoder involves four deconvolution layers symmetric to their encoding counterparts, preceded by three fully connected layers. All decoding layers are chosen without bias. The filtering and scaling layers finally generate the velocity field $\underline{v}$, which is then exponentiated. All convolution and linear layers are equipped with $\tanh$ activation functions, at the exception of the last layer of the encoder.

Hyper-parameters (σ_Ω, σ_V, g, T, q)**.** In our auto-encoding context, a good heuristic is to choose σ_Ω of the order of the geometrical noise to eliminate (or slightly larger), and σ_V of the order of the geometrical signal to synthesize (or slightly smaller), while preserving $\sigma_\Omega \leq \sigma_V$. For the sake of simplicity, we choose the two grids g_Ω and g_V equal (which can always be achieved by union). We further assume this grid g to be composed of $|g| = 2^{rd}$ nodes, with $r \geq 4$. The number of time-steps is fixed to $T = 5$ in our experiments, when the latent-space dimension q is task-dependent.

5. Experiments

5.1. Simulated rectangles

We simulate two-dimensional rectangle meshes by independently scaling the two axes of a baseline unit square. A train data set of $n = 21^2 = 441$ rectangles is created with scaling factors uniformly distributed between 50% and 150%. A test data set composed of $n = 32^2 = 1024$ elements is also generated with factors ranging from 22.5% to 177.5%. The splatting and deformation kernel widths σ_Ω, σ_V are respectively fixed to a fifth and a half of the baseline square width. Finally, the latent-space dimension is $q = 2$ and we train the networks with ten thousands epochs.

We learn two models: the original mesh auto-encoder presented in this article, and an unconstrained variation where the isometry between the latent codes and the velocity fields is not enforced (see Figure 3). For identifiability reasons, the parameter λ is not estimated in the latter case, and is fixed to 1. Table 1 gives the mean ℓ^2 distances between the original and the reconstructed rectangles, both for the train and test rectangles. The fit is almost perfect on training data points, and satisfying for test ones. The isometric architecture outperforms the unconstrained one. Figure 4 plots some examples of such reconstructions for the test data set (outer part), as well as the learned latent codes

	Train (n=441)	Test (n=1024)
Isometric	0.0022 ± 0.0004	0.014 ± 0.018
Unconstrained	0.0024 ± 0.0009	0.020 ± 0.024

Table 1. Averages and associated standard deviations of the mean point-to-point ℓ^2 distance between the original and reconstructed rectangles in several configurations. Values are percentages of the average rectangle width.

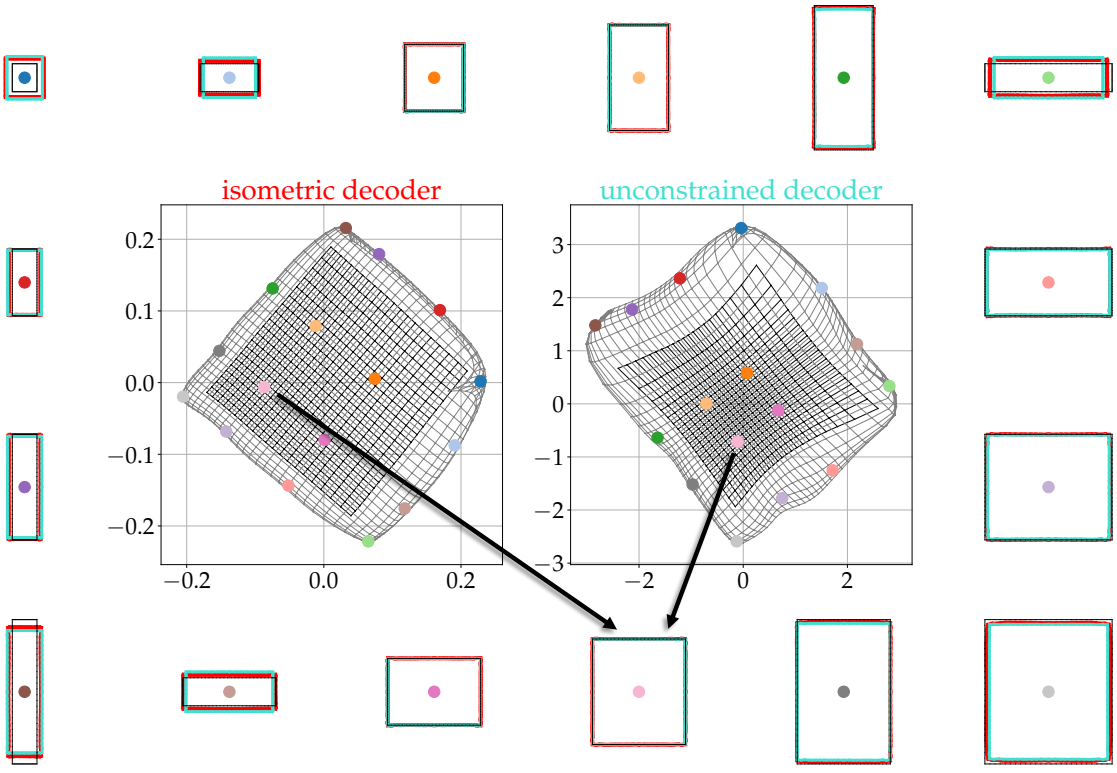

Figure 4. *[Inner].* Latent spaces learned by the proposed mesh auto-encoder (left), and a modified architecture without norm scaling (right). For both, the black grid marks the encoded train rectangles, when the lighter grey grid marks the encoded test rectangles. *[Outer].* Examples of 16 test rectangles (black lines), superimposed with the reconstructions obtained either with the isometric (red lines) or unconstrained (blue lines) decoders. A color code allows to identify the corresponding latent representations.

(inner part). The fit is visually perfect for the test rectangles which belong to the training distribution, i.e. of width and length between 0.5 and 1.5. The fit quality slightly deteriorates otherwise. The latent-space learned with the isometric decoder is more regular than its unconstrained counterpart: the training samples are evenly spaced along two orthogonal components, which is in line with the way data has been generated. In other words, the learned representation better preserves the true distances between meshes.

5.2. Emotive faces

The Birmingham University 3D dynamic facial expression database [33] provides short video sequences from 101 subjects (58 females, 43 males) mimicking emotions, such as surprise. Focusing on this case, we uniformly collect 8 frames for each subject from the first 1.4 seconds of video. Each frame comes with 73 segments delimiting facial features, on which we base the following analysis. For each subject, we consider every other frame as train (respectively test), which defines two data sets composed $n = 404$ meshes each. In a preprocessing step, meshes are aligned together with the Procrustes method, and projected on a 2-dimensional plane. We choose $\sigma_\Omega = 15\%$ and $\sigma_V = 30\%$ in percentage of the average distance between the eyes. The auto-encoder is learned with a latent space of dimension $q = 5$, and ten thousands epochs.

Evaluated with the mean point-to-point ℓ^2 distance, the average residuals amount to $9.2\% \pm 1.0$ and $9.4\% \pm 1.2$ on the train and test data sets respectively (values in percentage of the average distance between eyes). Figure 5 gives two representations of the learned latent-space, along with the reconstruction of two sequences. Reminding that every other face in each sequence was considered as a train or test data point, we see that meshes are tightly reconstructed in all cases, showing a good generalization of the auto-encoder. Those sequences correspond to smooth and progressive trajectories in the latent space, as it can be seen for 10 randomly selected sequences. A principal component analysis (PCA) was learned on train codes and used to visualize the 5-dimensional latent space. Those trajectories are globally following the first axis, which nicely corresponds to the dynamics of a surprised face: the mouth enlarges and the eyebrows move up. The second axis seems to encode more static information, such as the distance between eyes and eyebrows, or more subtly the total width of the faces. This axis visually seems to correlate with the gender, as it can be

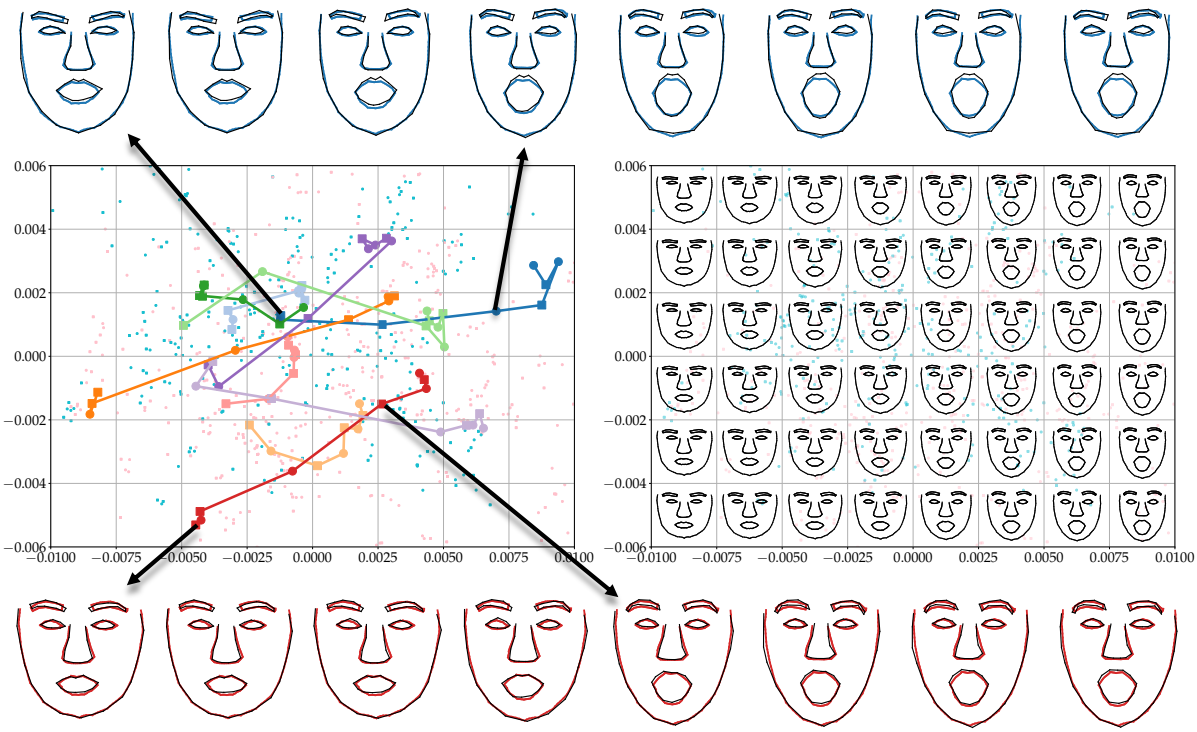

Figure 5. *[Outer].* Two particular sequences of faces mimicking being surprised (times goes from left to right). In black are plotted the original meshes, and in blue or red color the reconstructions. The first, third, fifth and seventh frames are train data points.
[Center left]. Principal component projection of the latent-space codes. Ten randomly-chosen sequences are represented with colored lines. Square markers (respectively circle) indicate train data points (respectively test). All remaining codes are plotted in the background.
[Center right]. Principal component projection of the latent-space codes. The pink color (respectively blue) indicates female subjects (respectively male). Are superimposed in bold black the decoded meshes corresponding to the code at the center of the grid cell.

noticed from the background pink or blue points.

5.3. Hippocampus sub-cortical structures

The Alzheimer's disease neuroimaging initiative gathers longitudinal clinical measurements about normal controls (CN), subjects with mild cognitive impairments (MCI) or patients diagnosed with Alzheimer's disease (AD). We select a balanced set of $n = 225$ T1-weighted magnetic resonance images providing from as many subjects. After standard alignment preprocessing, the right hippocampus cortical sub-structures are segmented into triangular meshes of varied topology. We choose $\sigma_\Omega = 5\,mm$ and $\sigma_V = 10\,mm$. For reference, the right hippocampus of an healthy subject typically measures around $50\,mm$ in length. We learn the autoencoder for ten thousands epochs in three different configurations $q \in \{5, 10, 20\}$.

In the absence of point-to-point correspondence, we measure the fitting residuals with the current distance (see equations 4 and 6): they amount to 47.0 $\text{mm}^2{}_{(\pm 9.6)}$, 53.5 $\text{mm}^2{}_{(\pm 10.3)}$ and 43.4 $\text{mm}^2{}_{(\pm 10.0)}$ for configurations $q = 5$, 10 and 20 respectively, which is satisfyingly small. For reference, the rightmost image of Figure 6 plots a reconstruction with a current residual of $60.1\,\text{mm}^2$. This figure provides as well an illustration of the exponentiation mechanism, which deforms the estimated template y_0 in a natural way as close as possible to the target. Note the satisfying refinement of the initial prototype template into a realistic real hippocampus mesh. We finally learn linear discriminant classifiers in each latent space. Table 2 gives the results obtained with a stratified 10-fold approach, for two tasks of increasing difficulty: (i) discriminate CN from AD, (ii) jointly discriminate CN, MCI and AD. Good performance is obtained for the first task, especially since only the geometry of a single hippocampus is considered (and not a full T1 image). The second task proves harder, but classification scores remain well above the chance threshold of 33.3%. Figure 7 gives some intuition about those scores: in the proposed PCA projection of the learned latent space ($q = 10$), the three classes are nicely organized from left to right into CN, MCI and AD subjects. We show that this axis also correlates with the volume of the hippocampus, which is in line with current medical knowledge.

	$q = 5$	$q = 10$	$q = 20$
2-class	86.0 ± 8.4 %	92.0 ± 7.6 %	92.8 ± 6.5 %
3-class	64.9 ± 4.8 %	62.2 ± 9.4 %	68.1 ± 8.5 %

Table 2. Average linear discriminant analysis classification scores, obtained with a stratified 10-fold method. The 2-class task consist in discriminating CN subjects from AD ones (chance level 50%), when the 3-class task adds the MCI subjects (chance level 33%).

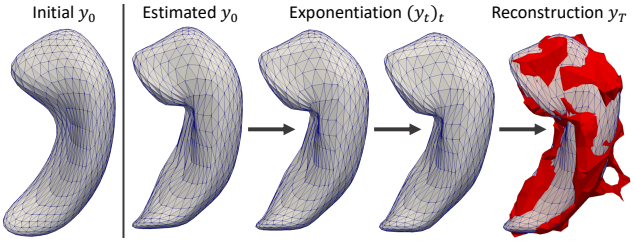

Figure 6. The leftmost image plots the initial template mesh y_0. The remaining of the figure displays the exponentiation mechanism that warps the estimated template into some target, displayed in red. The current residual distance amounts to $60.1\,mm^2$.

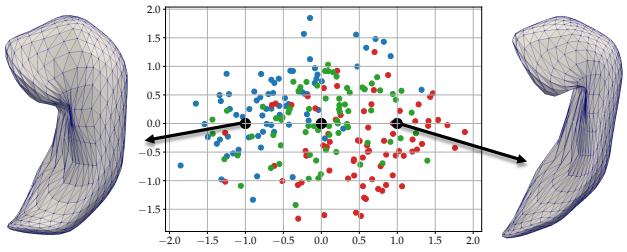

Figure 7. PCA projection of the learned latent space in the configuration $q = 10$. The codes are represented with blue points if they correspond to CN subjects, green points to MCI and red to AD. Two particular latent points are decoded and displayed. The third central code correspond to the template y_0 (see Figure 6).

6. Conclusion

We introduced the current-splatting layer which allows neural networks to process meshes without any assumption on their topology. Conversely, we adapted to meshes the exponentiation layer in order to synthesize shapes with a diffeomorphic control on their topology. Taking advantage of those input and output layers, we proposed an auto-encoding architecture that learns a generative statistical model from a distribution of meshes. A norm-control layer explicitly enforces the correspondence between the Euclidean latent-space metric and the shape-space log-Euclidean one. An experiment with a simulated data set showed that this layer fosters the learning of latent spaces more representative of the data distribution. Experiments with real data sets demonstrated the ability of our auto-encoder to handle varied types of geometrical data, and to learn relevant low-dimensional representations. The proposed method requires the manual choice of two important characteristic length hyper-parameters. As a perspective however, both the current-splatting and the exponentiation layers could easily be generalized to handle multiple characteristic scales, at the cost of linearly increasing number of channels and therefore of computational pressure.

Acknowledgments. This work has been partly funded by grants ERC 678304, H2020 EU project 666992, ANR-10-IAIHU-06.

References

[1] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache. A log-euclidean framework for statistics on diffeomorphisms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 924–931. Springer, 2006. 3, 5

[2] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016. 1

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1

[4] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems*, volume 2, 2013. 2

[5] A. Dai, C. Ruizhongtai Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017. 2

[6] A. V. Dalca, G. Balakrishnan, J. Guttag, and M. R. Sabuncu. Unsupervised learning for fast probabilistic diffeomorphic registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 729–738. Springer, 2018. 2

[7] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[8] P. Gori, O. Colliot, L. Marrakchi-Kacem, Y. Worbe, C. Poupon, A. Hartmann, N. Ayache, and S. Durrleman. A Bayesian Framework for Joint Morphometry of Surface and Curve meshes in Multi-Object Complexes. *Medical Image Analysis*, 35:458–474, Jan. 2017. 5

[9] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (TOG)*, 34(4):87, 2015. 2

[10] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. 4

[11] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 2

[12] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1966–1974, 2015. 2

[13] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 2

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *stat*, 1050:10, 2014. 5

[15] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 1

[16] J. Krebs, H. e Delingette, B. Mailhé, N. Ayache, and T. Mansi. Learning a probabilistic model for diffeomorphic registration. *IEEE transactions on medical imaging*, 2019. 2

[17] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 1

[18] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017. 1

[19] F. Morgan. *Geometric measure theory: a beginner's guide*. Academic press, 2016. 2

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, volume 11, pages 127–136, 2011. 2

[21] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016. 1

[22] X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006. 5

[23] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes. Image2mesh: A learning framework for single image 3d reconstruction. In *Asian Conference on Computer Vision*, pages 365–381. Springer, 2018. 2

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1

[25] G. Riegler, A. Osman Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017. 1

[26] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008. 1

[27] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017. 2

[28] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1

[29] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1274–1283, 2017. 2

[30] M. Vaillant and J. Glaunès. Surface matching via currents. In *Information processing in medical imaging*, pages 1–5. Springer, 2005. 2

[31] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 2, 10

[32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1

[33] L. Yin, X. Chen, Y. Sun, T. Worm, and M. Reale. A high-resolution 3d dynamic facial expression database. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE, 2008. 7

[34] L. Younes. *Shapes and Diffeomorphisms*. Applied Mathematical Sciences. Springer Berlin Heidelberg, 2010. 3

A. Appendix: reconstruction of a 3D car mesh

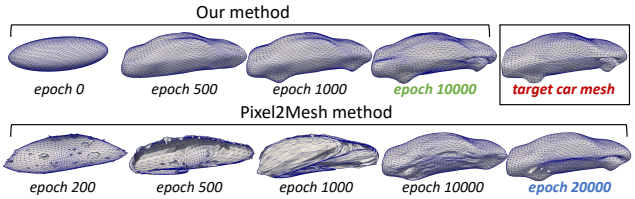

Figure 8. Intermediate and final reconstructions of the target car mesh (top-right figure) achieved by our method (top row, stopped after $10k$ epochs) or P2M (bottom row, stopped after $20k$ epochs).

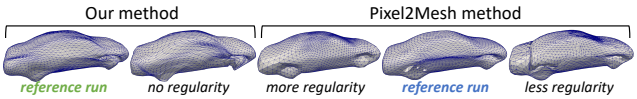

Figure 9. Comparison of the final reconstructions achieved by our method or P2M, when the attachment vs. regularity trade-offs are multiplied by either 0, 0.1, or 10.

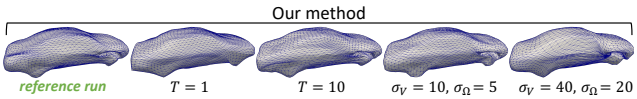

Figure 10. Comparison of the final reconstructions achieved by our method, when the number of integration steps T or the kernel sizes σ_V, σ_Ω (in % of the target car width) are modified from the reference configuration $T=5$, $\sigma_V=20$, $\sigma_\Omega=10$.

We compare the mesh synthesis mechanics of our method with the Pixel2Mesh (P2M) approach [31], where shapes are obtained by successively deforming and oversampling a reference ellipsoid mesh. Displayed in the top-right corner of Figure 8, a 3D car mesh is generated from their already-trained model[1], that we then re-train from scratch on this unique data-point. Our method is estimated from this same observation, with σ_V and σ_Ω respectively fixed to 20 and 10 percents of the target car width, and $q=1$ (optimal choice when $n=1$). As for all other experiments presented in the paper, the grids are determined according to the proposed heuristics (see Methods). For fair comparison, the template shape y_0, normally learned, is fixed to the 2562-vertices ellipsoid mesh on which is based P2M. Figure 8 shows that P2M starts from very irregular meshes before slowly converging towards car shapes. Our method always produces meshes diffeomorphic to the ellipsoid template, and only explore a relevant space of solutions during the whole optimization process. Figure 9 compares the output of our method when the regularity term is ignored, and of P2M where the original trade-off between attachment and regularity loss terms is divided or multiplied by 10. We recall that our method normally learns its own trade-off (see Eq. 19). Qualitatively, our method performs equally well. The P2M results are changed: the reconstructed car is either too smooth or of broken topology e.g. with self-intersecting edges. Finally, our method is learned in four perturbed configurations of hyper-parameters: $T=1$ or 10 (instead of $T=5$), or kernel sizes $(\sigma_V, \sigma_\Omega)$ divided or multiplied by 2. Figure 10 shows that our method is globally robust to such modifications, though the fine details of the target car kink above the front-right wheel is not really captured in the case $T=1$.

[1] Available at: https://github.com/nywang16/Pixel2Mesh