Supplementary material : Geodesic Deep Shape Optimization

Pierre Baque [*1] **Edoardo Remelli** [*1] **François Fleuret** [2 1] **Pascal Fua** [1]

In this supplementary material, we first provide additional detail on the handling of the irregular vertices of the Cube-Mesh CNNs of Section 3.1. We also report a quantitative comparison between the forward-pass time for convolution operations on unstructured meshes and similar operations running on structured meshes such as the ones produced by our re-meshing algorithm. We then give analytical definitions of the 2D and 3D deformation parameterizations of Sections 5.1 and 5.2. Finally we specify the flow conditions for all reported experiments.

1. Handling Singular Points for Semi-Regular Quad-Meshes

As discussed in Section 3.1 of the paper, when mapping a surface onto a cube-mesh, we have to deal with irregular vertices, which correspond to the corners of the cube and have three neighbors instead of four. To perform convolutions efficiently we first unfold the cube surface onto a plane. As illustrated by Fig. 1, we can then simply pad irregular corners with the feature values associated to cube edges. This enables us to use standard convolutional kernels even in the neighborhood of irregular vertices. Furthermore, since we use Geodesic Convolutions, the irregularity is naturally handled by the interpolation operation.

2. Computational Efficiency

We provide a quantitative comparison between the forward-pass time of 5 convolution consecutive operations running on unstructured meshes and similar operations running on structured meshes such as the ones produced by our re-meshing algorithm. We use a Nvidia Titan X GPU and a Tensorflow implementation.

- Standard Convolution: Exec. Time 0.00098 s. Such a speed is possible thanks to the regular grid organization.

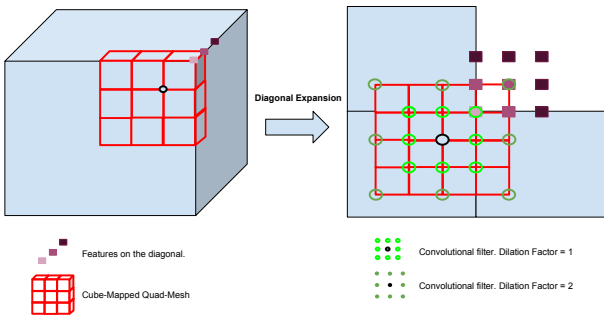

Figure 1. Handling the singularities of the Quad-Mesh for convolution purposes.

- Graph Laplacian Convolution Sparse of (Kipf & Welling, 2016): Exec. Time 0.00889 s. The model of (Kipf & Welling, 2016) has limited expressivity because the convolutions are performed by averaging neighbouring features using Adjacency matrices.

- Geometric Convolution Dense of (Monti et al., 2016): Exec. Time 0.27200 s. It corresponds to the dense version as implemented in the public code of (Monti et al., 2016). TensorFlow uses GPU computation with dense adjacency matrices. The main drawback is the memory requirements, which limit the graph size and the number of features.

3. Airfoil Parameterization in 2D

In this section we will first briefly describe the standard NACA airfoil 4 digit parameterization (Jacobs et al., 1948), which, confusingly involves 3 degrees of freedom. We then discussed our extension to 19 degrees of freedom.

NACA 4 digit. Without loss of generality, we can assume that the airfoil is of unitary cord length and let $0 \leq x \leq 1$ the coordinate that defines the position along that length. Let us further consider the airfoil thickness t, maximum camber m , along with its location p. To compute the airfoil

[*]Equal contribution [1]CVLab, EPFL, Lausanne, Switzerland [2]Machine Learning Group, Idiap, Martigny, Switzerland. Correspondence to: Pierre Baque <pierre.baque@epfl.ch>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

shape, we first define the mean camber line

$$y_c = \begin{cases} \frac{m}{p^2}\left(2px - x^2\right), & 0 \le x \le p \\ \frac{m}{(1-p)^2}\left((1-2p) + 2px - x^2\right), & p \le x \le 1 \end{cases}$$

and the airfoil thickness to camber y_t as

$$5t\left[0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right].$$

Since the thickness needs to be applied perpendicular to the camber line, the coordinates (x_U, y_U) and (x_L, y_L), of the upper and lower airfoil surface, respectively, become

$$x_U = x - y_t \sin\theta, \qquad y_U = y_c + y_t \cos\theta, \tag{1}$$
$$x_L = x + y_t \sin\theta, \qquad y_L = y_c - y_t \cos\theta, \tag{2}$$

where

$$\theta = \arctan\left(\frac{dy_c}{dx}\right), \tag{3}$$

$$\frac{dy_c}{dx} = \begin{cases} \frac{2m}{p^2}\left(p - x\right), & 0 \le x \le p \\ \frac{2m}{(1-p)^2}\left(p - x\right), & p \le x \le 1 \end{cases} \tag{4}$$

Thus, the wing shape is entirely defined by the choice of t, m , and p.

18-parameter foils. We increase the number of degrees of freedom by that writing the 3 parameters t, m, p as quadratic functions of x, that is,

$$t(x) = t_0 + t_1 x + t_2 x^2$$
$$m(x) = m_0 + m_1 x + m_2 x^2$$
$$p(x) = p_0 + p_1 x + p_2 x^2$$

where the the p_i, m_i, and q_i control the new degrees of freedom. Moreover we allow the lower and upper airfoil surfaces to be associated two two different camber lines, hence doubling the total number of degrees of freedom to $2 \times (3 + 3 + 3)$.

4. Surface Parameterization in 3D

As discussed in Section 5.2, we parametrize 3D shape deformations using a transformation function $f_{\mathbf{C}} : \mathbb{R}^3 \to \mathbb{R}^3$ that applies to the vertices of an initial shape $\mathbf{X}^0$, where $\mathbf{C}$ is a 21D vector. For clarity, let us split the 21 components of $\mathbf{C}$ into three groups, one for each axis $\mathbf{C} = \{C_i^x\}_{i=0\ldots6} \cup \{C_i^y\}_{i=0\ldots6} \cup \{C_i^z\}_{i=0\ldots6}$. As show in Fig. 2, L_x, L_y, L_z, denote the maximal size over each dimension and let (x, y, z) be the coordinates of a specific vertex X. We write

$$\begin{aligned} f_{\mathbf{C}}(X)_x &= C_0^x + x[C_1^x + C_2^x x \\ &\quad + C_3^x cos(\frac{y}{L_y}2\pi) + C_4^x cos(\frac{z}{L_z}2\pi) \\ &\quad + C_5^x sin(\frac{y}{L_y}2\pi) + C_6^x sin(\frac{z}{L_z}2\pi)] \,, \\ f_{\mathbf{C}}(X)_y &= C_0^y + y[C_1^x + C_2^y y \\ &\quad + C_3^y cos(\frac{x}{L_x}\pi) + C_4^y cos(\frac{x}{L_x}2\pi) \\ &\quad + C_5^y sin(\frac{x}{L_x}\pi) + C_6^y sin(\frac{x}{L_x}2\pi)] \,, \\ f_{\mathbf{C}}(X)_z &= C_0^z + z[C_1^x + C_2^z z \\ &\quad + C_3^z cos(\frac{x}{L_x}\pi) + C_4^z cos(\frac{x}{L_x}2\pi) \\ &\quad + C_5^z sin(\frac{x}{L_x}\pi) + C_6^z sin(\frac{x}{L_x}2\pi)] \,. \end{aligned}$$

This simple parametric transformation provides enough freedom to generate sophisticated shapes. Furthermore, the initial shape corresponds to setting all the parameters to 0, except from C_1^x, C_1^y, C_1^z, which are set to 1.

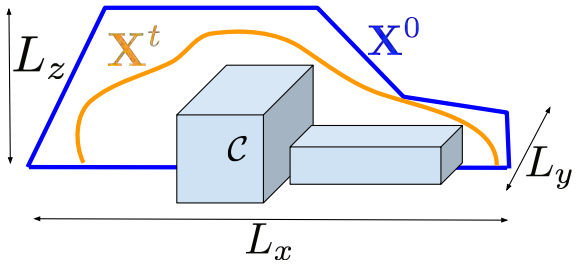

Figure 2. Schematic representation of car optimization.

5. Flow conditions in 2D.

The 2D foil simulator we used was XFoil (Drela, 1989). We considered viscous subsonic flows, with physical parameters Reynolds=9e5, Angle of Attack=0.0. The parameters were set for a concrete application: optimizing the shape of an hydrofoil for a student boat competition.

6. Flow conditions in 3D.

The 3D CFD simulator used was ANSYS Fluent (Inc., 2011). We considered a viscous k-epsilon turbulence model with Reynolds=4e6. This values refer to the optimization of car bodies, that is length of about 2m, and flow speed of 100km/h.

References

Drela, M. XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. In *Conference on Low Reynolds Number Aerodynamics*, pp. 1–12, 1989.

Inc., Ansys. *ANSYS FLUENT Theory Guide*. November 2011.

Jacobs, Eastmann N., Ward, Kenneth E., and Pinkerton, Robert M. The characteristics of 78 related airfoil sections from tests in the variable density wind tunnel. Technical Report 460, 1948.

Kipf, T.N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M.M. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. *arXiv preprint arXiv:1611.08402*, 2016.