Unified Shape Analysis and Synthesis via Deformable Voxel Grids

Raphaël Groscot[1] and Laurent D. Cohen[1]

University Paris-Dauphine, PSL Research University, CEREMADE
CNRS UMR 7534, 75016 Paris, France
{groscot, cohen}@ceremade.dauphine.fr

Abstract. This paper extends the investigation and application of Deformable Voxel Grids (DVGs) into a unified framework for 3D shape analysis and synthesis. DVGs consist in a grid that approximates a shape's silhouette via energy-minimization. This provides an improved embedding space over a regular voxel grid as it aligns with the geometry of the shape, and subsequently allows for the deformation of the shape by manipulating the DVG's control points. We demonstrate how DVGs directly and naturally serve for an array of applications: correspondences, style transfer, shape retrieval, and PCA deformations. We further address the challenge of morphing non-parametric shapes, an ill-posed problem because of the trade-off between plausibility and smoothness, particularly under large topology changes. Thanks to DVGs, we extract a shape content descriptor, and propose a similarity metric adapted to the extracted content and a formulation of morphings as minimal paths in a graph. Our approach leverages the strengths and interpretability of DVGs while achieving morphing capabilities comparable to those provided by neural networks. Throughout the course of our study, we conducted qualitative and quantitative analyses on the robustness and quality of our proposed methods, and we provide valuable insights into the effective manual intervention that can enhance quality, given the interpretability of each component in our method. In conclusion, this work elucidates the wide-ranging implications and potential of DVGs in 3D shape comparison, processing, and morphing, paving the way for future research and applications in the field.

Keywords: Shape space · Deformable models · Generative 3D modeling.

1 Introduction

Three-dimensional shapes present a unique challenge when it comes to comparison due to their lack of a standardized basis of representation.

On the one hand are surfacic representations: pointclouds and meshes. Pointclouds are unordered sets, so comparing two requires a registration step which essentially determines point-to-point correspondences. This matching can operate under different assumptions: for instance, ICP [4] for rigid affine transformations, or Earth Mover's Distance [8] for minimal transport-cost one-to-one matching. The same holds for meshes, but the explicit connectivity also allows for spectral methods, whose local descriptors derive from eigenvalues of operators such as the Laplacian [39,32].

On the other hand are volumetric representations: voxel grids and signed distance fields (SDF). Being scalar signals in the unit cube of $\mathbb{R}^3$, they indeed have a canonical embedding space. However, a big part of voxel space is non discriminative: the center (*resp.* the border) of the cube is mostly always occupied (*resp.* empty). To capture rich geometric details, they also require a sufficient grid resolution, leading to high dimensional spaces where distances are not meaningful. As for SDFs, they generally serve as a proxy to either render the shape via ray marching [21] or generate a mesh via marching cubes [29]. This is why, in the context of shape editing, both these representations are mainly useful for boolean operations – intersection and union.

Instead of relying solely on their geometry, shapes can be compared using parameters which translate to meaningful properties, such as length, height, etc. In the modeling phase, this can be achieved by parametric surface or volume elements, such as splines and nurbs. But in the analysis phase, these parameters are generally not accessible. This is where shape priors come to play, either in the form of statistical distribution estimation [9], or with methods relying on deformations from a template [27] or between pairs of shapes [19]. These are controlled by the Free Form Deformation (FFD, [37]) which, although having volume-preserving properties, offers unintuitive controls.

Another approach comes with neural networks, and more especially generative models such as GANs [12] and VAEs [25] They offer a so-called *latent space* which serves as a canonical, Euclidean parameter space, with a typical dimensionality lower than that of the geometric shape space, amenable to meaningful distances between points. Several works demonstrated how latent arithmetics allows for similarity clustering [35,36], shape analogies [1], and recombinations [13,7]. However, they rely on heavy computations on large datasets, which not only requires powerful hardware and a long training, but also depends on the reconstruction capacity of the chosen generator architecture.

This paper shows how the Deformable Voxel Grid [16,14,15] (DVG) geometric representation serves as a unified analysis and synthesis framework, with various applications, including: dataset exploration, similarity search, deformations, approximation with quadrilaterals, correspondences, and morphing. The fact that it works surprisingly well, given the simplicity of the methods, is a strong clue that DVGs encode relevant information about shapes, and demonstrates the versatility of this representation. It also experimentally highlights the strengths and weaknesses of a generic DVG optimization procedures.

The optimization of a DVG to a shape, and its subsequent registration, allows to write $S = p^V(C)$ where S is a given shape, V the positions of its control points, C the cubified version of S, and p^V a cube-to-grid projector (see Section 3 for more details). This corresponds to a separation in a canonical space of S into a low-level description of its appearance (V), and a high-level description of its surface details (C). This separation is what enables the array of applications we present.

What is furthermore remarkable is that the DVG model was designed in a very generic way, without focus on any of these applications. This is why we do not resort to more complex methods, except for the very last one, morphing. In the other cases, improving the performances of each application to reach state-of-the-art is out of the scope of the present work.

2 Related Work

Deep learning and shape latent spaces In this area, use cases typically include one of the following tasks: shape processing and data augmentation [33,9,23,18,28], shape prediction from 2D data [38,18,43,42,8], shape completion [2,17,34], and latent space exploration [41,43,28,1,13,7].

Over the past years, generative models, using architectures inspired by Variational Auto Encoders [25] and GANs [12], have been used for shape morphing, via linear interpolations in the latent space. Works can represent shapes in various formats, such as pointclouds [35,36,8,1,13], voxels [43,28,7], octrees [40], or 4D particle dynamics [31].

Other works, focusing on the prediction of a 3D shape from a given image, used more specific representations such as surface patches [17] or deformed ellipsoids [42], but they do not appear to offer a direct way of producing shape morphings. More recently, networks predicting implicit functions [34,20] appeared to allow smooth, arbitrary-topology meshes, while being compatible with latent space interpolations.

Deep learning approaches are based on the assumption that learned descriptors, as opposed to handcrafted ones, are better suited to capture the variability of natural signals. Moreover, generative models offer a latent space amenable to the generation of new shapes. However, neural networks come with known limitations: their lack of interpretability and their constraints to converge to visually-pleasing results. As a matter of fact, they typically require rich databases, powerful GPUs, and suffer from long training times and difficult parameters tuning. All this makes reproducing the results of deep learning based methods hard, even when a portion of the code is public.

Our work takes quite an opposite view, exploring the possibility of achieving similar results without any neural network. We show in fact that we can obtain the same capabilities offered by generative networks' latent spaces, by carefully handcrafting and designing every component. As a result, we already generate satisfying results even with modest datasets (around 500 shapes), and because every component has a clear meaning, one can easily improve the desired outcome by manual intervention.

In a way, this work ultimately consists in investigating what remains once those applications are stripped from neural networks, in order to better understand the specificities they bring.

Morphing based on deformations The problem of realistic shape morphing was tackled by [10] for human and animal bodies, interpreting a collection of shapes as a deformation space. By establishing shape correspondences, they obtain a shape distance allowing them to express morphings as a minimal path among clusters of similar body poses. We adopt a framework similar to theirs, while focusing on shapes which have varied topologies and no natural parameterization, such as chairs and sofas, such that a morphing cannot be interpreted as a mere deformation. Our work can be seen as a derivation of the same ideas, but adapted to different modalities, typically addressed by deep learning methods.

As far as they are concerned, instead of relying on geometric generative models, shape deformation is another popular choice to generate realistic shapes at a small cost,

leveraging the similarity between objects belonging to the same class. To parameterize these deformations, most approaches [19,27] use the Free Form Deformation [37], which arranges control points on a regular volumetric grid, and then uses cubic interpolation to distort the object as the points move. The key differences with our model are the following. First, they tackle different problems, such as partial shape alignment [19] or shape reconstruction by deforming a template [27]. Second, and most importantly, their shape deformations are pair-specific, trained to predict deformations between pairs $((A,B)$ where A is deformed into $B)$. On the contrary, our model provides a consistent shape cubification, without any learning, allowing to compare all shapes (in terms of similarity measure); and we use this representation to estimate minimal-energy morphings.

More recently, [44] showed how to reconstruct shapes by deforming an implicit template, predicted by a neural network, giving shape correspondences and deformations. In our method, we can see the cubification step as a template deformation, where the template is the unit cube, and where the deformation is not learned but computed. We try to achieve similar results, in surface quality and interpolation smoothness, but without the constraints and limitations of deep learning as explained above.

Parametric and statistical shapes descriptors Describing a shape class by a given set of parameters (also referred to as a *dictionary*) is a fundamental operation for applications such as classification, model retrieval, or similarity search. Some approaches [23,18] learn probabilistic distributions of shapes from the properties of their semantical parts, or even from the relations between parts simplified into simpler geometric primitives [41]. Others learn explicit parameterizations, typically possible on shapes representing body poses [2].

Our method relies on shape cubification, serving as a shape descriptor which, while being class-independent, is more adapted to shapes having strong reflection symmetries. The key difference with these other methods is twofold. First, we show how traversing the space of plausible shapes does not require statistical inference but can be expressed as a minimal-path problem in a graph, whose structure captures the geometric relations between existing shapes. Second, our descriptor is invertible, which allows us to generate new shapes (for the intermediate states of morphings), without any neural network.

Deformable Voxel Grids Our cubification relies on Deformable Voxel Grids [16,14,15], a model inspired by the Topological Active Volume (TAV) from [3], which is a volumetric extension of active contours [24,6]. The unfamiliar reader can think of active contours as parametric curves which minimize a given energy, typically used for segmenting objects in images [22]. In the case of DVGs, the energy ensures that after an initialization as a regular voxel grid around the input shape, the grid is optimized until it tightly and smoothly embraces the shape (see Figure 1).

3 Shape cubification via Deformable Voxel Grids

Fitting DVGs around shapes offers a consistent representation basis within the same class (e.g. chair or car): the same semantical parts of different objects tend to occupy

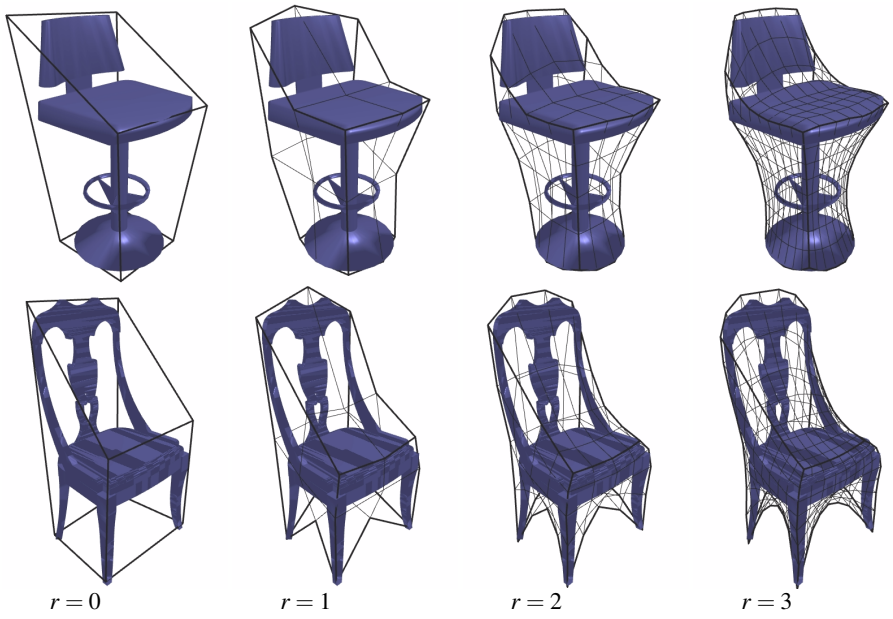


Fig. 1. Optimization of DVGs at progressive resolutions r (constant number of steps per r).

the same region of the cube, allowing for easier shape comparisons. As a matter of fact, an optimized DVG can be interpreted as a smooth deformation of the unit cube $\mathbb{R}^3$ adapted to a given shape S, and we can apply the inverse deformation to S. We call this "*shape registration*", and it provides us with an invertible shape cubification.

Forward DVG projection For a given DVG cell c, a point q inside can be expressed by its *local coordinates*, a triplet in $[0,1]^3$:

$$\tilde{u},\tilde{v},\tilde{w} \in [0,1]^3 \text{ s.t. } f_c(\tilde{u},\tilde{v},\tilde{w}) = q \tag{1}$$

where $p_1, p_2, \ldots, p_8$ are the positions of the eight vertices of c. The interpolator f_c can be linear or smooth (we use, respectively, a trilinear p^V_{tri} and a Thin Plate Spline p^V_{tps} interpolators). Both can be defined by matching the control points V^0 of a regular cube to V, those of a given DVG.

Then, an affine transformation maps the cell to its correct location within the whole DVG grid system (see Figure 2b).

Backward DVG projection We suppose the signed distance field (SDF) of shape S is given. Each cell of V is subdivided into smaller subcells, and the value of the SDF is queried at the locations of each subcell centroid, which naturally have (u,v,w) coordinates, coming from the indexing of V: vertex index (i,j,k) has local coordinates $(u,v,w) = (\frac{i}{r-1}, \frac{j}{r-1}, \frac{k}{r-1})$. The cubified SDF can be used in two ways.

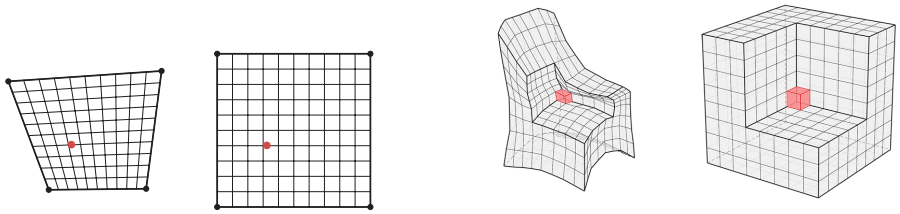


Fig. 2. A natural way of setting grid coordinates on a quadrilateral is via bilinear interpolation, which maps regular subdivisons of $[0,1]^2$ onto the quad cell (*left to right*). Determining the local coordinates of a given point within the cell corresponds to inverting this interpolation (*right to left*). The same is done for registering a point inside a DVG, but in 3D, with the inversion of a trilinear interpolation.

Content descriptor A downsampling of the cubified SDF provides a useful descriptor to group models by similarity. From this, we build the shape graph used for morphings. See more details in Section 5.

Shape cubifiction and reconstruction From the cubified SDF, we obtain C, the mesh of the cubified shape, using marching cubes [29]. The precision of the geometry is limited by the grid resolution of the DVG and the number of cell subdivisions. The original shape can be recovered by projecting C into V using the spline projector p^V_{tps}. This operation is important because it gives the baseline shape representation capacity for the morphings we generate: as a matter of fact, our morphings are done in V-space and C-space separately, and use the p^V_{tps} projector to effectively create the intermediate shapes. The intuition is that the DVG separates style and configuration, respectively into C and V. This way, we find morphings that minimize the amount of displaced mass to transfer style (C), while the deformation abilities of the grid allow to interpolate the configurations V.

Figure 3 shows examples of such cubifictions and reconstructions using both p^V_{tri} and p^V_{tps}.

4 Analysis and Synthesis on cubified shapes

4.1 Overview of applications

All applications were tested on shapes from various categories of ShapeNet [5]; the models distribution is summarized in Table 1. Let us briefly provide intuitions for why, and how, V and/or C are used for each application.

Analysis

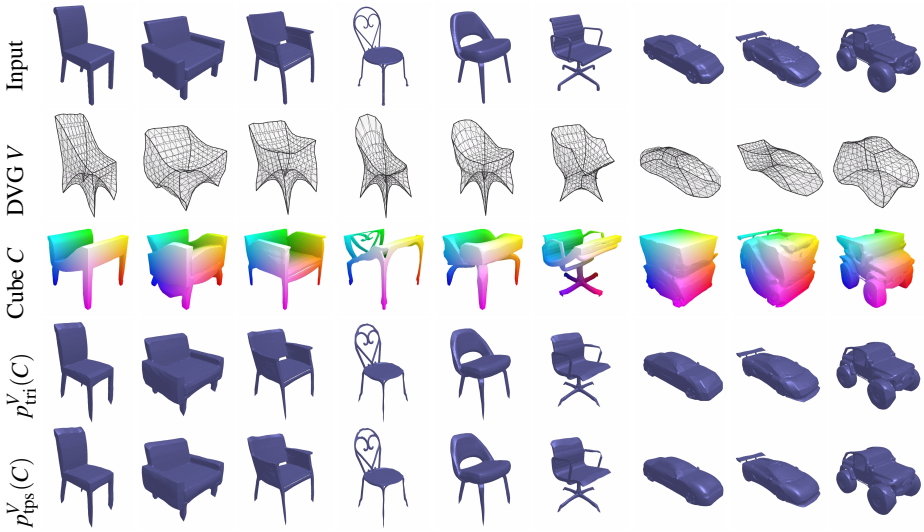


Fig. 3. (Zoom in to see details) Qualitative results for shape reconstruction via DVGs: estimating V for each shape allows to cubify it (C), and this can be reprojected into V by a trilinear p^V_{tri} or a spline p^V_{tps} projector (see 3), the latter yielding smoother surfaces. The cubified shapes are color-coded by assigning a value in $[0,1]^3$ to an RGB color.

Table 1. Distribution of shapes used in this work, taken from the ShapeNet [5] dataset.

Category	Car	Chair	Airplane	Sofa	Rifle	Lamp	Bench	Speaker	*Total*
Size	1500	1500	1100	800	1000	850	1500	750	9000

♠ **Correspondences:** Cubified shapes tend to have similar parts in similar locations, which suggests a potential for estimating shape correspondences, using a naive closest-point matching.

♠ **Similarity search:** Relying solely on V, which approximates the outer surface of S, gives a topological-invariant shape descriptor which can be used to retrieve similar models.

Synthesis

♢ **Approximation with quads:** Surfaces given as triangle meshes can be approximated by quadrilateral meshes, thanks to the grid system of V.

♢ **Style transfer:** Projecting C_1 into V_2, which conforms a shape S_1 into the same outer shape as another shape S_2. This is reminiscent of style transfer.

♢ **Semantical editing:** A generalization of the previous application, learning a parameterization of V across a full dataset, in order to deform a given shape.

♢ **Morphing:** Interpolating continuously between values of V and C.

4.2 Applications relying solely on V

V as a shape descriptor: mode exploration and similarity search We test the intuition that the low-level representation encoded by V contains enough information about a shape. In order to do so, a shape descriptor can be computed by flattening V, a $9 \times 9 \times 9 \times 3$ array, into a vector of $\mathbb{R}^{2187}$. First, because this high dimensional space is not displayable, we visualize its t-SNE [30] embedding. An embedding according to just three dimensions already reveals that categories are regrouped into distinct clusters (Figure 4). We also did the same operation, but only considering the points of V located at the surface of the grid (total of 386 points, hence dimensionality $d = 1158$). This yielded a virtually indistinguishable t-SNE embedding. These two observations confirm two points:

1. V is an acceptable low-resolution shape descriptor;
2. Most of the signature information of the shape is located at the surface of V.

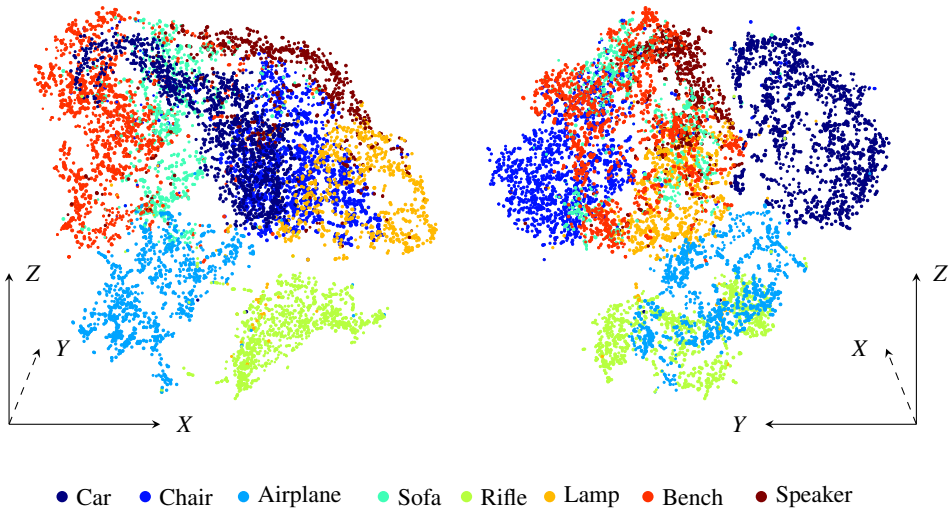


Fig. 4. Three-dimensional t-SNE embedding of values of V across several shape categories (viewed from different angles for better appreciation of the 3D). Notice how each category is largely localized in its own cluster.

Just like with the latent space of generative models, clusters are not entirely impervious to models from other categories. For instance, some airplanes look like cars, etc. Then, we also propose to use DVGs in order to search a dataset for models similar to a given query shape. To do so, we use as a shape descriptor the full $9 \times 9 \times 9 \times 3$ array. By construction, its parameterization is shared across all models, such that a simple Euclidean distance translates to a point-to-point distance (no point matching required, they

are already matched). In Figure 5 we show the first three nearest-neighbors of several models, randomly selected from our dataset. We have seen that the different categories belong to different clusters, and here again, when taken from the whole dataset (without any supervision), the nearest-neighbors are in most cases consistent with the category of the query model. Moreover, the found models appear to be structurally similar objects, that is to say, to belong to the same semantical sub-cluster. In Figure 5 we can indeed find examples of armchairs, fighter jets, airline planes, etc.

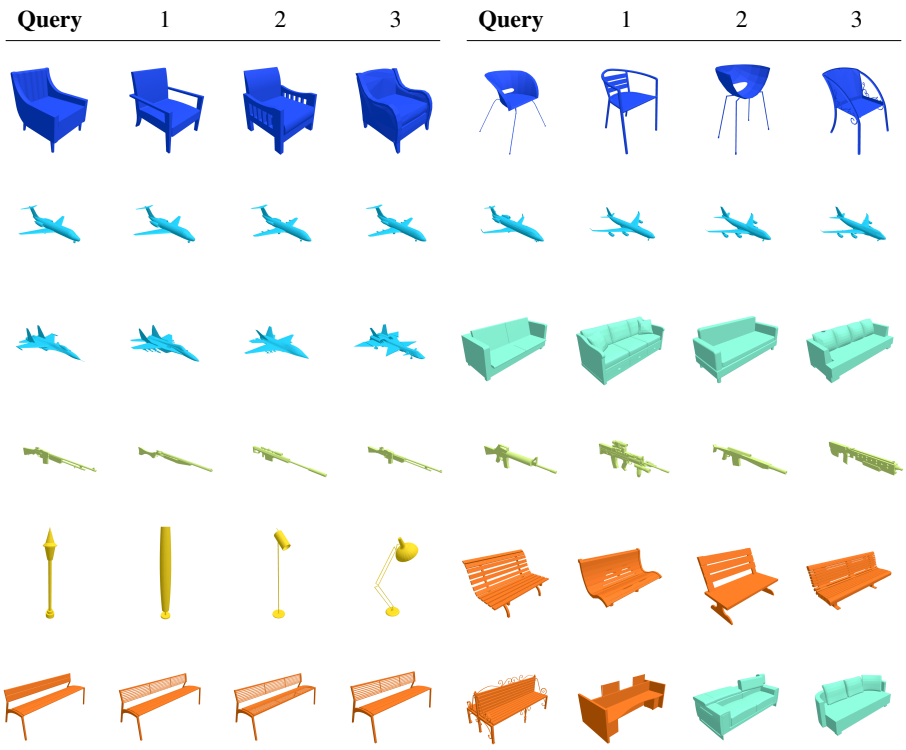


Fig. 5. Examples of similarity search using V as a descriptor (Euclidean distance): for each Query model, we find the three nearest-neighbors in the whole dataset. Models are colored according to their respective categories (same colors as in Figure 4). In the majority of cases, they not only belong to the same category, but are also functionally similar. The last example, on the bottom-right corner, shows a case of confusion, where an uncommon bench is similar to sofas.

Deformation-based applications: style transfer and semantical editing These two applications consist in the manipulation of V in order to deform its underlying shape S. Given two shapes $S_1 = p^{V_1}(C_1)$ and $S_2 = p^{V_2}(C_2)$, we can transfer the style of S_1 into the grid system of S_2 by switching V_1 for V_2:

$$S_{1\text{ in }2} = p^{V_2}(C_1) \text{ and } S_{2\text{ in }1} = p^{V_1}(C_2) \tag{2}$$

This formulation emphasizes the separability of V and C, which we argue can be interpreted as a notion of "container" (V) and "content" (C), in a paradigm similar to *style* and *content* in style transfer[1]. We show in Figure 6a an array of different V and C values combined together.

From our experiments, we confirmed that this kind of style transfer works well for shape types satisfying our underlying assumptions, which we summarize here: 1. volumic shapes, with no near-degenerate dimension; 2. shapes which remain realistic under mild volume distortion; 3. (V_i, C_i) pairs with matching semantics[2]. Note that the surface quality of the generated shapes, when using shapes cubified via our implicit registration depends on the accuracy of the SDF estimation. We used the Mesh to SDF tool from [34], which tends to generate a lot of artifacts when used in conjunction with Marching Cubes [29] to reconstruct the 0 level-set. To keep satisfying shapes, we reconstruct an ε level-set ($\varepsilon \approx 0.01$), which explains the apparent "thickness" of the generated shapes.

As for statistical deformations, we show, as a minimal working example, that they can be performed by a simple PCA in V-space. More specifically, for a given shape's V, we explore in its vicinity the subspace generated by the first two principal components. The shape is reprojected into the corresponding V values (see Figure 6c). The first principal components of V translate into high-level furniture properties, such as length, height, slant, etc. Built into an editor with sliders, this provides an interactive tool to quickly deform a shape and explore its possible variations. Notice that the plausibility of the shape decreases as the magnitude of the displacement increases – that is to say, as we get further away from the center of the image.

4.3 Applications requiring C

Approximation with quads Instead of relying on Marching Cubes to implicitly register the shape, it can actually be directly voxelized within the DVG grid system: instead of reconstructing a mesh, we simply "light on" the DVG cells which intersect the object, and extract the outer surface of the resulting object. This can be done at any arbitrary subdivision level of the DVG[3]. While this does not serve a reconstruction accuracy purpose, it offers a quad-mesh surface approximation with the following properties:

1. Consistency across a dataset, which can be useful for building coherent scenes out of multiple objects;
2. Control of the approximation resolution.

[1] Note however that our application is different from the typical style transfer found in the literature.

[2] As for the consistent alignment of shapes, this could be performed as a pre-processing step.

[3] Say we optimized V up to its fourth resolution level, $r = 3$, into an $8 \times 8 \times 8$ grid. All subsequent subdivisions will not be optimized, but can still be used to refine the resolution of the voxelization.

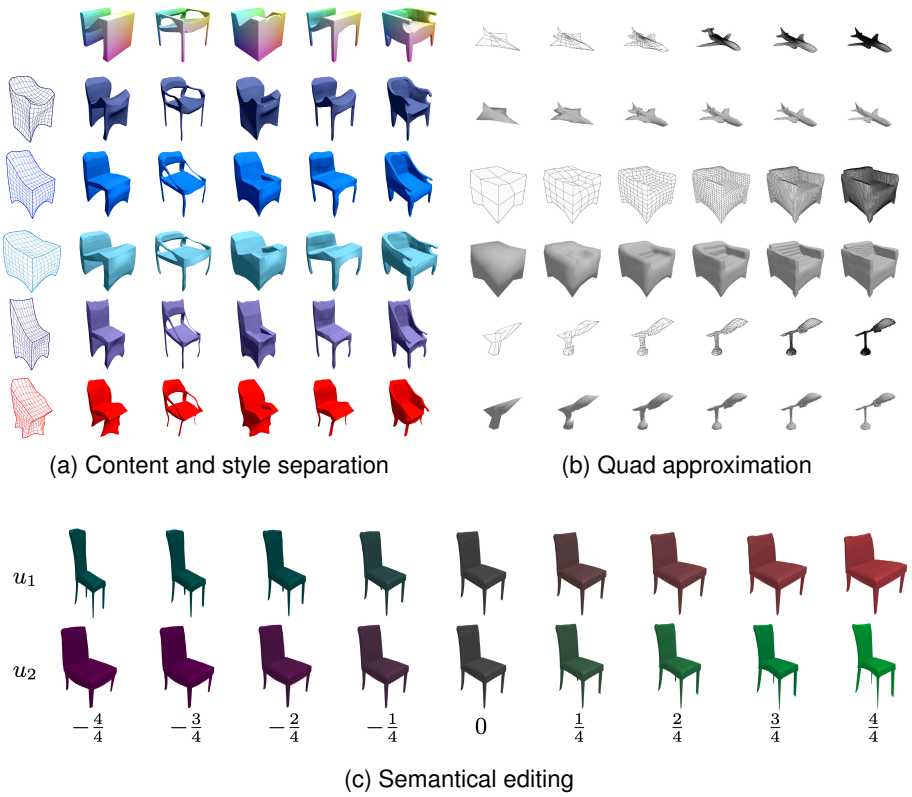


Fig. 6. (a) Container (V, rows) and content (C, columns) separation. This generates interesting and controllable shape variations, mostly useful for furniture.
(b) Extracting a quad mesh from optimized DVGs, with increasing resolutions. The odd rows correspond to a rendering of the corresponding mesh: first, each quad is split into two triangles; then, normals are computed for each face via edges cross-product; finally, normals are averaged for each vertex and then interpolated across faces (method known as *smooth normals*).
(c) Having encoded $k = 1500$ chairs with DVGs, a PCA on $\{V_k\}_k$ yields principal modes of deformations, applied to deform any shape. For example in this chair dataset, they correspond to vertical and horizontal elongations (original shape on central column; u_i is the i-th principal component, the color gradient corresponds to the deviation from the shape's DVG).

Some examples are shown in Figure 6b. While producing far from ideal meshes, it works well for a surprising variety of shapes, across all categories of our dataset. Notably, rendering the resulting meshes produces much more satisfying surfaces than regular voxel grids of similar resolutions. We propose two ways of improving the results. First, one could re-fit the obtained quad mesh to the surface (similarly to Mesh-RCNN [11]). Or, after the convergence of V, the elastic energy could be dropped on useless internal nodes, similarly to the original Topological Active Volumes [3]. Second, the staircase effect, typical of voxel grids and still present here, to a lesser extent, could be mitigated with the development of a "diagonal tracing" strategy, which would adaptively add a "ramp" (a diagonal quad between cells) where needed. This could be done with an adaptation of the Marching Cube algorithm, but requires a careful study of its ambiguity lookup table.

Correspondences One last way of investigating the usefulness of cubifiction is shape correspondences. This intuition emerged from the style transfer application (Section 4.2): if a pair of shapes is compatible with style transfer, their cubified versions should display similar parts in similar locations of the cube. Thus, we tried to make correspondences between cubified shapes by identifying points by their position – more precisely, to their nearest neighbor in the other shape. Reprojecting the cubes C_1 and C_2 to their respective grids V_1 and V_2 transfers these correspondences back to the original shapes.

This very naive approach, while obviously limited, produces satisfying results for varied shapes. We show such examples in Figure 7, where identified points have the same color (we also draw colored lines to help visualize the correctness of correspondences). One particularly interesting benefit is the explainability of the correspondences, provided by the grid. A grid misalignment between two shapes, resulting from a bad convergence of the first DVG resolution levels, can lead to wrong correspondences.

5 Graph-based shape space

5.1 Motivation

In the previous Section, the analogy with latent spaces was apparent in several places. This led us to question whether DVGs, and their ability to separate shapes into $S = p^V(C)$, could be used as a replacement for generative models in the application they seem particularly powerful, namely, morphing. With style transfer, we have already seen how we can transform (V_1, C_1, V_2, C_2) into (V_2, C_1), by replacing V_1 by V_2 in $S_1 = p^{V_1}(C_2)$. This can be done continuously, with a linear interpolation on V from V_1 to V_2. Moreover, the correspondences experiment confirmed that similar shapes have similar C content, which suggests that there is a possibility to easily interpolate between C_1 and C_2. In such a case, we can then build a morphing from S_1 to S_2, by a simultaneous linear interpolation from V_1 to V_2 and from C_1 to C_2.

What about shapes that are too dissimilar? We precisely developed a method around this idea, leveraging a whole dataset to find a good sequence of intermediate states.

Once all shapes of a dataset are consistently cubified, we propose to discretize the global shape space in the form of a weighted graph, whose edges derive from a similar-

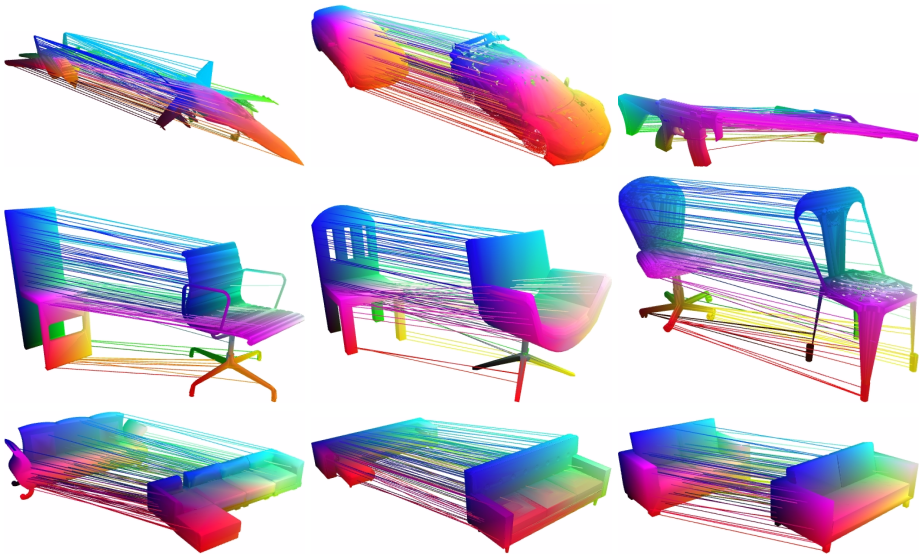


Fig. 7. Correspondences computed by matching closest points in cubified shapes, and transferred back to the original shapes.

ity measure between cubified shapes. This graph formalizes the notion of shape morphing, as a morphing from shape A to B will correspond to a minimal path from node $\mathcal{N}_A$ to $\mathcal{N}_B$. This choice is motivated by the fact that for a large enough shape dataset, most intermediate steps of a morphing are close to existing shapes.

This is why we explore the possibility of discrete morphings, restricted to known shapes, effectively bypassing the necessity to learn how to sample new shapes; while imposing them a minimal energy criteria.

We first present a general framework for shape morphings as minimal paths in a shape graph, for any arbitrary shape embedding. Then, we show how it can be used with cubified shapes and how our invertible cubification actually allows to easily extrapolate the discrete morphings to continuous ones.

5.2 Morphings as minimal paths

In this part, we consider the problem of morphings with shape priors, that is to say, morphings such that intermediate states are plausible. We operate under the minimal assumption that the shape prior is given by a finite set of exemplars, $\mathcal{S} = \{S_1, \cdots, S_N\}$, where the shapes are given in an arbitrary embedding. A morphing corresponds to a sequence of shapes from $\mathcal{S}$, but we want a metric to evaluate the quality of a morphing. In order to do so, we impose a cost (or an energy) to a morphing:

$$E(M = (S_1, \cdots, S_k)) \triangleq \sum_{i=1}^{k-1} E(S_i, S_{i+1}) \tag{3}$$

Where $E(S_i, S_j)$ is the energy of the transition $S_i \to S_j$. Such energies can be evaluated as paths length in a weighted graph $\mathcal{G}$ defined by:

- nodes $\{\mathcal{N}_1, \cdots, \mathcal{N}_N\}$, corresponding to the shapes in $\mathcal{S}$;
- positive weighted edges $\{w_{i,j}\}$ where w_{ij} can be interpreted as a similarity between shapes S_i and S_j. By convention, an absent edge (i, j) is equivalent to $w_{ij} = \infty$.

We call a morphing $A \to B$ *minimal* if it is achieved by a shortest path in $\mathcal{G}$ from $\mathcal{N}_A$ to $\mathcal{N}_B$. In order to consider symmetric morphings (i.e. equal to the time-reversed morphing), we assume $\mathcal{G}$ in an undirected graph, i.e, $\forall i, j, w_{i,j} = w_{j,i}$.

5.3 Graph of cubified shapes

We can apply the previous formalism to the space of cubified shapes. We propose a metric between shapes cubified via a DVG, which interprets as an approximate transport cost.

We extract the volume indicator function $\mathbb{1}_S$ by thresholding the SDF: its average value is binned within each DVG cell, in order to obtain an r^3 voxel image which serves as a shape content descriptor. Each cell value, between 0 and 1, represents the proportion of the cell which intersects the shape.

This provides a voxel image of a cubified shape. With a DVG resolution $r = 8$, this leads to a representation space with $8^3 = 512$ dimensions, enough for the curse of dimensionality to prevent Euclidian distances from being meaningful. This naive approach does not leverage the proximity of the cells, which is why we propose a method based on the morphological dilation operator.

Atypical models detection and removal A preliminary step is to exclude models for which the aforementioned volumetric descriptor is inadequate, that is to say, when the density of presence inside the cells is not homogeneous. To detect such models, we simply compute and sum all the inner-cell standard deviations of the discrete $\mathbb{1}_S$ obtained in Section 3. Figure 8 shows the most adequate and inadequate models for the chair dataset: unsurprisingly, sofas and armchairs, which admit blocky cubifictions, are the most adequate models; while chairs with many intricate details are the least. Because our descriptor, and the subsequent similarity metric, are blind to these errors, removing these inadequate shapes from the graph shape space prevents them from appearing in shape morphings.

Similarity metric Using the cross structuring element, real-valued dilation allows to add a one-voxel margin to a shape. Thus, the added voxels all correspond to cells whose L_1 distance to the shape is $1/r$. We can define a forward similarity metric D^{AB} from descriptor A to B, which penalizes the mass of B located outside the dilation of A:

$$\begin{aligned} D^{AB} &= \min(\text{dilation}(A) - B, 0) \\ D^{BA} &= \min(\text{dilation}(B) - A, 0) \\ d(A,B) &\triangleq \|D^{AB}\|_1 + \|D^{BA}\|_1 \end{aligned} \tag{4}$$

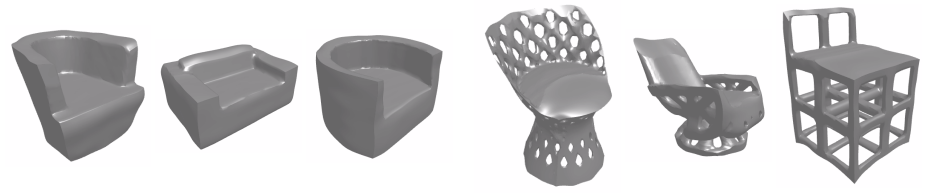


Fig. 8. Examples of adequate and inadequate models, with respect to our volumetric descriptor which averages density of presence within $8 \times 8 \times 8$ DVG cells.

Note that for all locations outside of dilation(A), the penalty imposed by $\|D^{AB}\|_1$ is the same as the L_1 voxel distance. Thus, for dissimilar models (e.g., if B is mostly located outside of dilation(A)), this metric d becomes less interpretable, because of the curse of dimensionality. We use this metric to build a shape graph $\mathcal{G}$, after all pairwise distances are evaluated. In general, each shape is connected to its k-nearest neighbors. However, the linking rules can vary for several reasons:

- Certain shapes may be particularly different from the rest of the dataset. In order to prevent them from being considered in morphings, we trim off links whose weights are above a threshold τ_w.
- To ensure that the final graph $\mathcal{G}$ is made of only one connected component, we can also decide to keep at least $k_{\min}$ connections for every node, even if their weights are above τ_w.

The impact of such trade-offs is discussed in Section 7.3.

5.4 Continuous morphings

To find a minimal path in $\mathcal{G}$, we use Dijkstra's algorithm. The returned length corresponds to the energy of the minimal morphing, while the sequence of nodes provides a discrete morphing. Thanks to our invertible cubification, this shape sequence can be prolonged to a continuous morphing, by interpolating separately the style and content (V,C) of each shape. For the control points positions V, the interpolation is trivial and can just be linear; as for the interpolation of content values, we also propose linear interpolation. More precisely, we interpolate the cubified SDFs, and generate the geometry with marching cubes.

Because each edge in the path has a known length, the continuous path can be parameterized by arc length (see Discussion 7.6). For an arbitrary number of frames, whose positions are equally spaced along the path, this grants more interpolation frames in between less similar shapes, which are the most likely to have topology changes.

For a given sequence of style-content separated shapes $((V_1,C_1)$, ..., $(V_k,C_k))$, and their corresponding edge lengths $L = (l_1^2, ..., l_{k-1}^k)$ in $\mathcal{G}$, we can formalize the continuous morphing using a time parameter $t \in [0,1]$:

$$\begin{aligned} i, i+1, \tilde{t} &= s_L(t) \\ \tilde{V} &= (1-t)\cdot V_1 + t \cdot V_k \\ y(t) &= p_{\text{tps}}^{\tilde{V}}((1-\tilde{t})\cdot C_i + \tilde{t}\cdot C_{i+1}) \end{aligned} \tag{5}$$

Where $s_L(t)$ is the discrete arc lenth parameterization function, returning the indices $(i, i+1)$ of the edge nodes and the local time parameter $\tilde{t}$. Note that the interpolation on V is straight from V_1 to V_k: the graph $\mathcal{G}$ is only used for interpolating the content C.

The same framework can be used to morph between new, unknown shapes, by embedding them into graph $\mathcal{G}$, following all the steps: DVG optimization, shape cubification, links creation to connect these new shapes to the already-existing graph.

6 Shape morphing results

We conducted our experiments using shapes from the ShapeNet [5] dataset; more specifically using 500 from the chair category and 200 from the car category. Because the continuous morphings require all shapes to be closed manifolds, and for fair comparisons against [26] which preprocesses shapes the same way, we first converted them into manifolds using the same method as [34]. We then sampled $l = 4096$ points to be used as the DVG input pointclouds.

6.1 Shape morphings

To produce shape morphings, we randomly picked pairs of nodes in $\mathcal{G}$, and applied the method explained in Section 5.4. Following Equation (5), each morphing consists in a sequence of triplets $(\tilde{V}, \tilde{C}, y)$. While we are typically only interested in the final geometry y, observing $\tilde{V}$ and $\tilde{C}$ provides, along with the found minimal path in $\mathcal{G}$, an explanation for the generated geometries. We show such triplets in Figure 9.

6.2 Robustness to misalignment

We assume all our shapes are consistently aligned, as standard among generative models. However, being a deformable model, we tested the ability of the DVG to converge to the correct configuration when the input shape has been rotated. Given the hierarchical subdivisions and the centrality of the first levels (see discussion in Section 7), we compared the determined first level for two conditions: ground truth alignment, and noisy alignment (rotation with random Euler angles, according to $\mathcal{N}(0, \sigma = 0.2\text{rad} \approx 12^\circ)$). For fairness, we kept a constant number of gradient descent steps. The error is then measured as the Euclidean distance between the control points of the two grids, $\|\text{GT} - \text{pred}\|_2$. Figure 10b shows an example of a misaligned grid

We also estimated, for any given τ_θ, $\mathbb{E}_{\theta_{\max} \leq \tau_\theta}(\|\text{GT} - \text{pred}\|_2)$, the empirical expectancy on datasets where the maximum angular error is smaller than a threshold τ_θ (see plot in Figure 10a). We observe that it increases at a reasonably low pace, confirming the advantages of a deformable model. As for high angular errors (more than 30°),

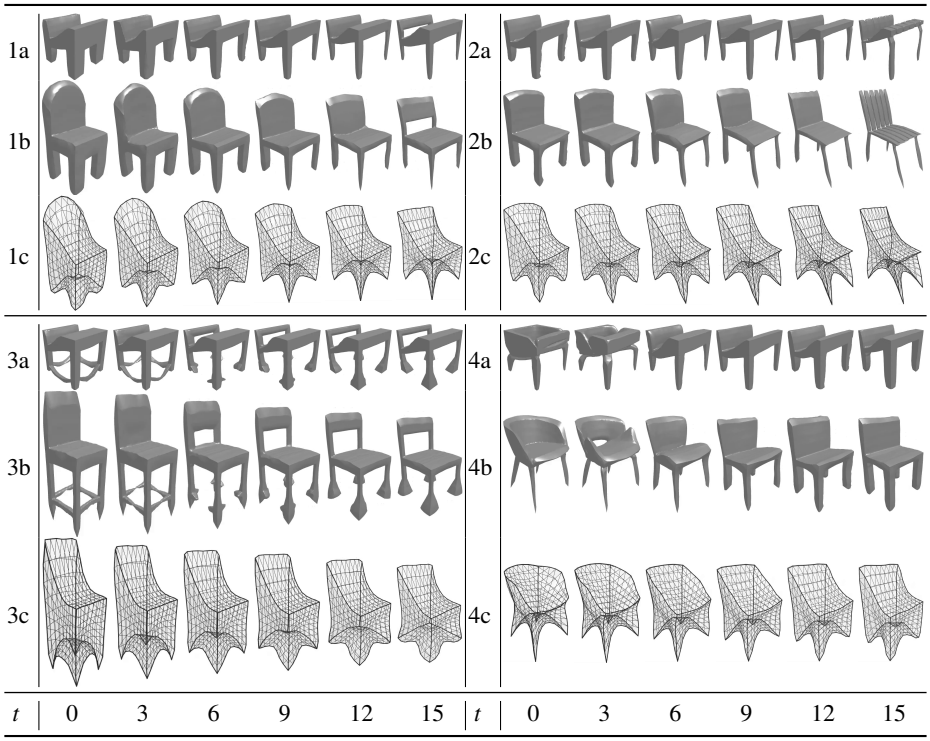


Fig. 9. (Zoom in to see details) Examples of morphings generated with our method. Rows (a), (b), and (c) respectively correspond to the interpolation of cubes C, final shapes, and DVG grids V.

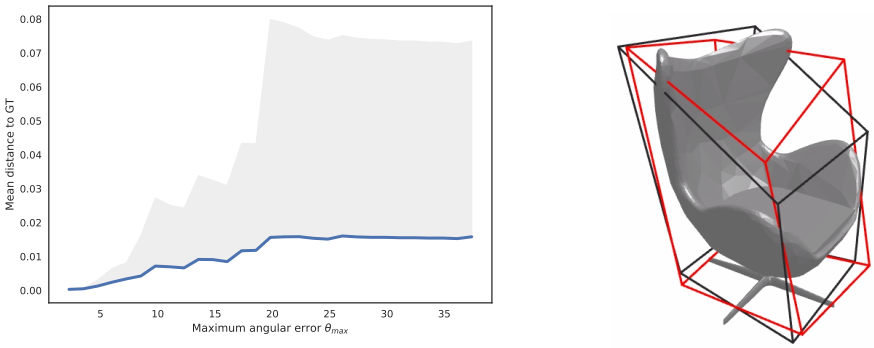
the predicted grid can be flipped: a control point which should be at the top is now located on the side. For a single shape, this is not a problem. However, on a whole dataset, this would break the consistent cubification we require to build our similarity measure.

6.3 Qualitative analysis: comparison with deep learning

In our work, one of the main objectives was to produce results comparable in quality to those obtained via deep learning. We chose to compare our results to the adversarial neural network developed by [26], as it also relies on an SDF representation, and has published the weights of a pre-trained network, allowing us to produce new morphings.

For fair comparisons, we adopted the following methodology:

- We kept our graph $\mathcal{G}$ untouched, built from the same 500 chair examples as in the previous experiments;
- We first generated baseline morphings as latent space interpolations between random codes corresponding to chairs (about 4k examples);
- For each of these morphings, we extracted the first and last states: these provided query shapes that we embedded in $\mathcal{G}$ (as explained in Section 5.4) in order to generate our morphings;



(a) Prediction error against shape misalignment (b) Misaligned v_0

Fig. 10. (a) Mean error against maximum rotation angle $\theta_{\max}$, grey area is the standard deviation; (b) Ground truth in black, misaligned prediction in red, where $\theta_{\max} \approx 20°$, error $= 0.078$. Refer to Section 6.2 for a discussion on these figures.

– To match the surface quality of [26], we decreased the resolution of our SDFs before the mesh reconstruction via marching cubes.

Figure 11 presents some comparative results, selected for their representativity. Here are our observations for each of the five shown examples:

1. These shapes happened to have a direct link when embedded in $\mathcal{G}$: the morphing entirely comes from the DVG.
2. While the morphing of the seat is visually pleasing, the SDF interpolation is responsible for a hole in the leg (frames 4–10).
3. The armrests removal looks less pleasing, but on the flip side, the progressive rounding of the back is more natural.
4. Most of the artifacts (frames 7–12) come from the short-circuit effect, discussed in 7.5.
5. Apart from similar surface artifacts, it exemplifies the impact of our minimal paths for chair-to-armchair morphings. Indeed, they appear to favor transitions which add thin armrests halfway through.

When watched as videos, our morphings also appear generally less smooth. This is due to our SDF interpolations: when new mass appears within the 1-voxel margin, it can look sudden, and then less pleasant to the eye. Despite these imperfections, we found it pretty remarkable to achieve such results while only relying on a much smaller dataset. It would indeed seem hard, if not impossible, to train a neural network with such limited data.

7 Discussion on shape morphings

Our overall method comprises many components, each requiring design choices which influence the quality of the results. Because the systematic analysis of general shape morphing (for non-parametric shapes) is still uncharted territory, we presented quantitative and qualitative results where each component is designed in the minimally-viable way. However, our system admits many local improvements. Hence the following observations and suggestions, noted from our experiments.

7.1 Importance of the first hierarchical levels

While [14] establishes the importance of the progressive refinement of DVGs, our experiments further emphasize the greater importance of the first levels. If the second level is unfrozen before the first level has correctly converged, the misalignment of the cube edge with the dominant features of the shape will remain. This problem can arise when optimizing a batch of DVGs on many shapes, with a constant number of epochs per level. To prevent this from happening, one has to make sure the first level has enough time to converge on all the training shapes, or resort to an adaptive gradient descent scheme.

7.2 Manual edition of a DVG

In a real use case scenario, a determined DVG can be manually corrected. For instance, it can easily be symmetrized – by averaging with its symmetric. This could be useful for shape reconstructions and morphings, to ensure that the generated geometries are indeed symmetric. All the results we show did not resort to any manual correction, in order to exhibit the bare abilities of our model. Yet it would be interesting, for future work, to investigate the usefulness of ad-hoc post-processing.

7.3 Graph connectivity

The quality of the morphings generated by our method depends on the graph building procedure, and more specifically, the node linking rule. Ordinarily, these graphs can be built obeying either a k-nearest neighbor condition, or a distance threshold condition. By design, our metric becomes less interpretable as the estimated distance increases. This is why we need to impose a distance threshold criteria. Doing so, the graph can however have several connected components, limiting the ability to interpolate between shapes of distinct modes (say, between an office chair and a sofa).

In practice, finding threshold values can be hard, given the non-uniform distribution of pairwise distances (for instance, we observed that the distances between many sofas are disproportionately small). To mitigate the potential mistakes, we build the k-nearest neighbor graph with three constraints, in decreasing priority: for node i,

1. $k_i \geq k_{\min}$
2. $k_i \leq k_{\max}$
3. $\forall j \leftrightarrow i, d(S_i, S_j) \leq \tau_w$

7.4 Volumetric descriptor sensitivity and specificity

In order to preserve fine information in the volumetric descriptor, we decided to average the indicator function $\mathbb{1}_S$ within each DVG cell, instead of keeping all cells where it is non null (which would correspond to a classical voxelization). However, the values can be small, and have a negligible influence on the similarity measure, even where there is non-negligible mass. Take the example of a half cube within a DVG cell, its average presence density is $1/2^3 = 1/8 = 0.125$. This is why we propose to apply the cubic root as a contrast function to increase sensitivity to low values – before feeding the descriptors to the similarity metric $d(A,B)$.

We also performed experiments where the resolution of the volumetric descriptor is $r = 16$, effectively halving the one-voxel margin tolerance. With this increased specificity, neighbors are more similar than before; but dissimilar models are further away than before. This led to discrete morphings which all contain many intermediate steps. Overall, the generated morphings were unpleasantly convoluted. We then settled for $r = 8$ as it appeared to be the best compromise, on our chair dataset.

7.5 Misleadingly low similarity and short circuits

Models whose topology is not adequately represented by our metric are, as explained in Section 5.3, not included in the graph. More precisely, we exclude the 20% most inadequate models.

But some models, not excluded from previous considerations, can badly influence the quality of morphings: those which display sharp surface features, not captured by our descriptor. They are typically not amongst the most adequate models, but still passed the aforementioned 20% threshold. Such a situation is depicted in Figure 12, rows (a).

Another interesting phenomenon appears when a pair of unwanted models hijacks many morphings. If they are each connected to distinct regions of $\mathcal{G}$, they provide a short circuit to many minimal paths.

This is the case for models at locations (1a,11) and (1a,12). They indeed appeared in many of our randomly generated morphings, creating unwanted surface artifacts. We show, in rows (b), that manually discarding these undesired models and short circuits can enhance the quality of the outputs. However, we kept all the other morphings we show in this paper untouched, in order to exhibit the results without any manual intervention.

7.6 Metric and arc-length parameterization

Because our similarity measure $d(A,B)$ only penalizes difference in shapes beyond a one-voxel margin, many pairs of shapes have a low distance, sometimes even null. This is due to the fact that the dataset contains many redundant shapes, with similar content C (for instance, many chairs resembling the model on Figure 11 at (1a,0). Contrary to the previous discussion 7.5 (on misleadingly low similarity), this is the case where similarities ought to be low. Conversely, for dissimilar shapes, the edge length can be disproportionately large, accounting for most of the total path length.

Since the discrete paths make continuous morphings by arc-length parameterization, this issue can lead to unpleasing morphings. It could have ben addressed in two ways:

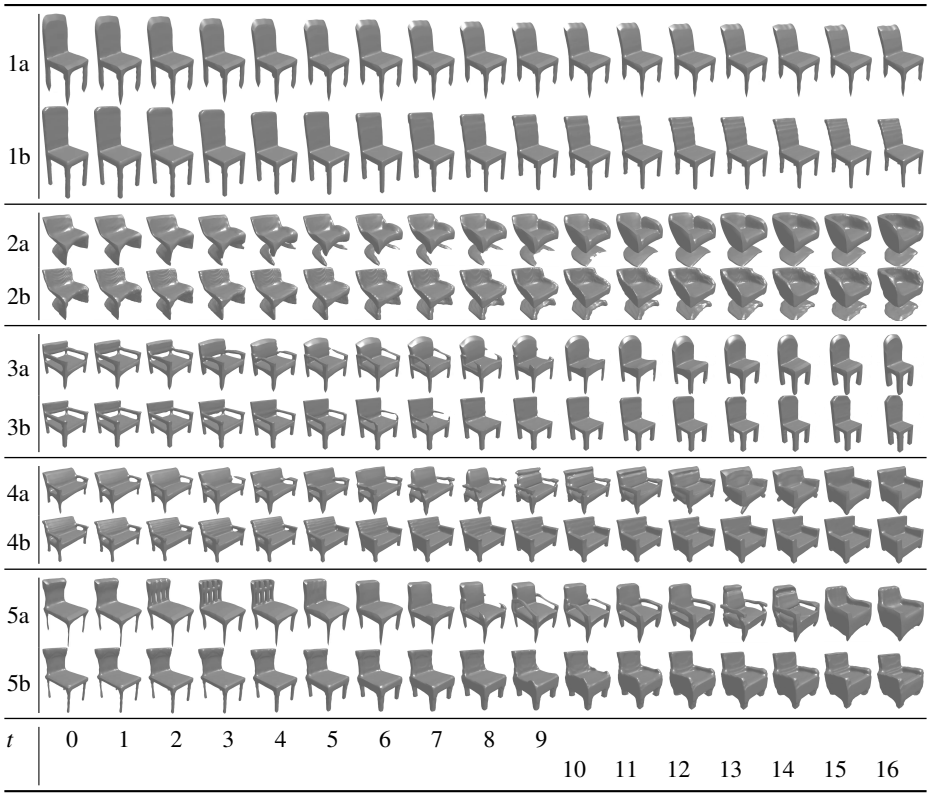


Fig. 11. (Zoom in to see details) Qualitative comparison of morphings obtained with our method ((a) rows) and with the Adversarial neural network of [26], based on implicit functions ((b) rows), time $t \in [0, 16]$. No manual correction was involved, and the grid resolution of the SDF was adapted to match surface precision.

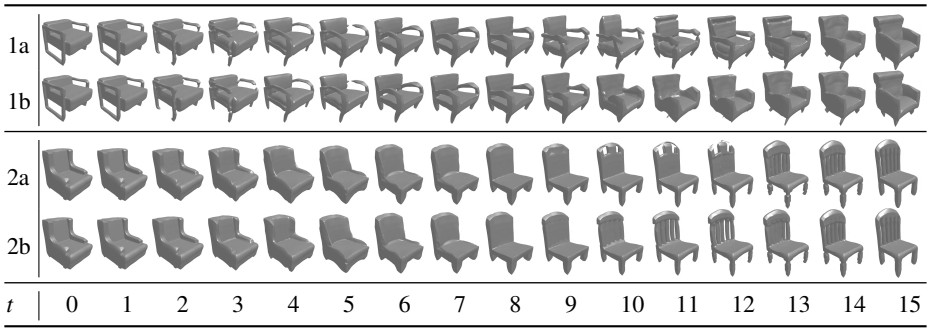


Fig. 12. (Zoom in to see details) Manual intervention on morphings: some generated morphings can display unsightly intermediate shapes because of the found minimal path (see (a) rows, at times 10 through 12). Manually discarding the unwanted models from the graph allows to redetermine an alternative, more pleasing, morphing ((b) rows).

1. Regrouping, like [10], cliques of interconnected shapes; and allowing at most one representative of a clique within a morphing;
2. Applying a non-linear transformation on the path length, before the generation of the continuous morphing.

We opted for the second option as it is the simplest and provides the baseline we are aiming for. We apply the function $x \mapsto 1 + \sqrt{x}$, where the square root rebalances low and high values, and the constant 1 corrects for the almost-null edge lengths.

8 Conclusion

In all the applications we presented using DVGs, the biggest limitation lies in the consistency of the grid alignment across models. In the present work, we only relied on the generic optimization procedure from [14]. One way to improve the performance of all applications is to improve this grid alignment, and we suggest two directions for future investigation. First, including a term based on semantical parts segmentations to the DVG energy. Second, performing adaptative grid subdivisions during the optimization of V, in order to handle degenerate dimensions better (like with flat or elongated shapes) and to bring finer details where they are needed.

As for shape morphings, our basic intuition is to connect shapes similar in content, so that morphing between them is "simple": the DVG cubification trick makes such a simple formulation of morphings possible. As we have shown, performing a simple linear interpolation on cubified SDFs already generates qualitatively pleasing morphings, therefore establishing a strong baseline. More complex approaches, based for instance on optimal-transport, could probably yield better results. Yet, we produced results qualitatively similar to the state-of-the-art deep learning methods, while relying on limited data.

Even if our DVG shape parameterization is not specific to any class, we restricted our shape morphings to chairs, because of the challenges posed by their varied topologies, and their strong reflection symmetries are compatible with cubification.

For future work, we would like to investigate the use of this model on shape categories displaying less symmetries. It would also be interesting to reproduce these results at a larger scale, say with the complete chair subset of ShapeNet, or even when adding other categories to the same graph: would we find different shape types separated in different clusters?

Finally, we could bridge the gap between our method and neural networks. Indeed, the SDFs interpolations are inherently limited and may not be able to fully capture shape priors, even in a large scale application. In this case, a generative model, trained only on cubified shapes for instance, could provide an interesting solution.

Acknowledgements This work was funded in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements davenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J.: Learning representations and generative models for 3D point clouds (2017)
2. Allen, B., Curless, B., Curless, B., Popović, Z.: The space of human body shapes: Reconstruction and parameterization from range scans. ACM Trans. Graph. **22**(3), 587–594 (Jul 2003)
3. Barreira, N., Penedo, M.G., Mariño, C., Ansia, F.M.: Topological active volumes. In: Petkov, N., Westenberg, M.A. (eds.) Computer Analysis of Images and Patterns. pp. 337–344. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
4. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(2), 239–256 (1992)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
6. Cohen, L.D.: On active contour models and balloons. CVGIP: Image Understanding **53**(2), 211 – 218 (1991)
7. Dubrovina, A., Xia, F., Achlioptas, P., Shalah, M., Groscot, R., Guibas, L.J.: Composite shape modeling via latent space factorization. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
8. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3D object reconstruction from a single image. CoRR **abs/1612.00603** (2016)
9. Fish, N., Averkiou, M., van Kaick, O., Sorkine-Hornung, O., Cohen-Or, D., Mitra, N.J.: Meta-representation of shape families. ACM Trans. Graph. **33**(4), 34:1–34:11 (Jul 2014)
10. Gao, L., Lai, Y.K., Huang, Q., Hu, S.: A data-driven approach to realistic shape morphing. Computer Graphics Forum **32** (05 2013). https://doi.org/10.1111/cgf.12065
11. Gkioxari, G., Malik, J., Johnson, J.: Mesh R-CNN (06 2019)
12. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014)
13. Groscot, R., Cohen, L., Guibas, L.: Shape part transfer via semantic latent space factorization. In: Nielsen, F., Barbaresco, F. (eds.) Geometric Science of Information. pp. 511–519. Springer International Publishing, Cham (2019)
14. Groscot, R., Cohen, L.D.: Deformable voxel grids for shape comparisons. In: Jiang, X., Tao, W., Zeng, D., Xie, Y. (eds.) Fourteenth International Conference on Digital Image Processing (ICDIP 2022). vol. 12342, p. 123423G. International Society for Optics and Photonics, SPIE (2022). https://doi.org/10.1117/12.2645961, https://doi.org/10.1117/12.2645961
15. Groscot, R., Cohen, L.D.: Shape morphing as a minimal path in the graph of cubified shapes. In: de Sousa, A.A., Bashford-Rogers, T., Bouatouch, K. (eds.) Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2023, Volume 1: GRAPP, Lisbon, Portugal, February 19-21, 2023. pp. 98–109. SCITEPRESS (2023). https://doi.org/10.5220/0011680200003417, https://doi.org/10.5220/0011680200003417
16. Groscot, R.: Separable 3D Shape Representations for Shape Processing. Ph.D. thesis (2021), http://www.theses.fr/2021UPSLD018, thèse de doctorat dirigée par Cohen, Laurent David Mathématiques appliquées Université Paris sciences et lettres 2021
17. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In: CVPR 2018. Salt Lake City, United States (Jun 2018)

18. Haibin, H., Kalogerakis, E., Marlin, B.: Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. Computer Graphics Forum **34** (08 2015)
19. Hanocka, R., Fish, N., Wang, Z., Giryes, R., Fleishman, S., Cohen-Or, D.: Alignet: Partial-shape agnostic alignment via unsupervised learning (2018)
20. Hao, Z., Averbuch-Elor, H., Snavely, N., Belongie, S.: Dualsdf: Semantic shape manipulation using a two-level representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
21. Hart, J.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer **12** (06 1995)
22. Hemalatha, R., Thamizhvani, T., Dhivya, A.J.A., Joseph, J.E., Babu, B., Chandrasekaran, R.: Active contour based segmentation techniques for medical image analysis. In: Koprowski, R. (ed.) Medical and Biological Image Analysis, chap. 2. IntechOpen, Rijeka (2018)
23. Kalogerakis, E., Chaudhuri, S., Koller, D., Koltun, V.: A probabilistic model for component-based shape synthesis. ACM Trans. Graph. **31**(4), 55:1–55:11 (Jul 2012)
24. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1**(4), 321–331 (1988). https://doi.org/10.1007/BF00133570, https://doi.org/10.1007/BF00133570
25. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. arXiv e-prints arXiv:1312.6114 (Dec 2013)
26. Kleineberg, M., Fey, M., Weichert, F.: Adversarial generation of continuous implicit shape representations (2020)
27. Kurenkov, A., Ji, J., Garg, A., Mehta, V., Gwak, J., Choy, C., Savarese, S.: Deformnet: Free-form deformation network for 3d shape reconstruction from a single image (2017)
28. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.J.: GRASS: Generative recursive autoencoders for shape structures. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017) (2017)
29. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH '87, New York, NY, USA (1987)
30. van der Maaten, L., Hinton, G.E.: Visualizing high-dimensional data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (2008)
31. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4d reconstruction by learning particle dynamics. In: International Conference on Computer Vision (Oct 2019)
32. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: A flexible representation of maps between shapes. ACM Trans. Graph. **31**(4) (Jul 2012)
33. Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: Transformation-grounded image generation network for novel 3d view synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
34. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
35. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3D classification and segmentation (2016)
36. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. CoRR **abs/1706.02413** (2017)
37. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. SIGGRAPH Comput. Graph. **20**(4), 151–160 (Aug 1986). https://doi.org/10.1145/15886.15903, https://doi.org/10.1145/15886.15903
38. Shin, D., Fowlkes, C.C., Hoiem, D.: Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3061–3069 (2018). https://doi.org/10.1109/CVPR.2018.00323

39. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. Comput. Graph. Forum **28**, 1383–1392 (07 2009)
40. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. CoRR **abs/1703.09438** (2017)
41. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: Computer Vision and Pattern Regognition (CVPR) (2017)
42. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: ECCV (2018)
43. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 82–90. NIPS'16, Curran Associates Inc., USA (2016)
44. Zheng, Z., Yu, T., Dai, Q., Liu, Y.: Deep implicit templates for 3d shape representation (2020)