Shape Part Transfer via Semantic Latent Space Factorization

Raphaël Groscot[1](✉), Laurent Cohen[1], and Leonidas Guibas[2]

[1] University Paris Dauphine, PSL Research University CEREMADE, CNRS, UMR 7534, 75016 Paris, France
groscot@ceremade.dauphine.fr
[2] Stanford University, Stanford, USA

Abstract. We present a latent space factorization that controls a generative neural network for shapes in a semantic way. Our method uses the segmentation data present in a collection of shapes to explicitly factorize the encoder of a pointcloud autoencoder network, replacing it by several sub-encoders. This allows to learn a semantically-structured latent space in which we can uncover statistical modes corresponding to semantically similar shapes, as well as mixing parts from several objects to create hybrids and quickly explore design ideas through varying shape combinations. Our work differs from existing methods in two ways: first, it proves the usefulness of neural networks to achieve shape combinations and second, adapts the whole geometry of the object to accommodate for its different parts.

Keywords: Autoencoder · Pointcloud · Latent space

1 Introduction

Design ideas exploration is a necessary step for creative modeling. Building tools that help quickly prototyping ideas can significantly improve designers' workflow. Given the tremendous size of 3D shape repositories, scanning all previously existing models can be cumbersome. This is why we propose, in this work, a first step to building such a tool: a shape composer that allows to combine parts coming from different objects into a single and coherent new object. Unlike other works that extract and snap different parts into new positions, we explore the possibility of holistic composition with the use of generative neural networks.

This paper presents a semantically-rich way of controlling generative networks for 3D shapes, without limiting the user to predefined labels. On the contrary, our approach is essentially data-driven in two ways. First, because we rely on a large collection of shapes to train our generative model; second, because the dataset itself is used by the user to tweak the output. More specifically, the

Raphaël Groscot—This work was initiated during a long visit in the Geometric Computation group at Stanford University.

dataset contains various shapes along with their segmentations into meaningful object parts. Our generative network is then trained to produce shapes in a way that is compatible with the segmentation. This is achieved by factorizing the latent space of the generative model according to the different possible shape parts. Thanks to this, a user can edit any given shape and decide to only change part of it, by picking the desired geometry within the dataset. Moreover, the network automatically adapts the final shape in a holistic way to make sure the new part fits naturally. This leads to an inherent ambiguity in our task. On the one hand, if we want to change the wings of a plane (for instance), we do not want *too much* change in the other parts of the plane. On the other hand, we still want the rest of the plane to adapt for the change. This is why we propose an asymmetry in the design of our model.

As a matter of fact, our method relies on the Variational Auto Encoder (VAE) framework (Sect. 3.1), but where the encoder is subdivided in several partial encoders, one for each semantical part, that are mixed into a global code which is then given to the decoder (Sect. 3.3). The decoder is structure-agnostic: it only knows to transform a general code into a plausible shape, so that when codes are manipulated and changed, the reconstruction should still look like a plausible shape.

2 Related Work

Our method is related to different research efforts in 3D shapes analysis and generation. We separate our review in three categories: generative modeling, shapes neural networks, and data-driven shapes editing.

Generative Neural Networks. Generative models suchs as Generative Adversarial Networks (GANs) [6] and Variational Auto Encoders (VAEs) [10] both offer ways to sample from a distribution that matches a given dataset. VAEs rely on an *autoencoder* scheme, where a network is asked to project data samples to a subspace of much lower dimensionality (*encode*), while being able to reconstruct the original data (*decode*). Adding a variational constraint that imposes a prior (e.g. gaussian) on the latent distribution makes sure that the model generalizes well. Their compression-like behavior can then be used for several tasks among which unsupervised learning, sampling, interpolation and denoising [4]. One drawback is that the output is typically blurred, because their loss does not account for a perceptual term. On the contrary, GANs aim at *mimicking* a given distribution by generating samples that are indistinguishable from the original dataset; they can hence generate much sharper results, to the cost of harder training and difficulty to control for mode collapse [12]. Conditioning on the likelihood [3,16] allows to have a finer control on their outputs. Our work aims at the same property by means of imposing a specific factorization on an autoencoder latent space.

Shape Neural Networks. As opposed to images, 3D shapes do not naturally fit in a neural network framework. The main issue is to represent them in a fixed-size Euclidian domain. The most direct way to do so is to use voxel grids and directly transpose Convolutional Neural Networks in 3D [7]. However, even if this approach can yield good results, generated shapes quality is limited by the grid discretization and the $O(n^3)$ complexity. To overcome these limits, Pointnet [18] introduced a neural network architecture based on pointclouds and permutation-invariant operators, which characterizes well an *unordered* set such as a pointcloud. It has successfully shown its usefulness for tasks such as classification and segmentation, and even has an extension that exploits hierarchical analysis [19]. This architecture can also be used to generate pointclouds from photographs [5]. Lastly, [2] has replaced the permutation invariance constraint by imposing a lexicographic order on the pointset, leading to pointcloud GANs with high reconstruction accuracy. Using shapes segmented into semantical parts, [17] learns the joint probability for structure and geometry – for instance, the presence or absence of engines on a plane will constraint the profile of the wings. While producing good quality results, their method does not allow to exchange parts between shapes. Our method relies on a variation of such a shape neural network, tailored at being used for shape combinations.

Data-Driven Shape Editing. Many existing methods give automated tools for shapes editing and design exploration. Existing works range from shape correspondences [7] to style similarity and transfer [13,14]. Others focus on generating diversity, by extracting and snapping parts together [8], or by hierchical shape analysis and synthesis [11]. While [8] creates a combinatorial diversity, our method focuses on the geometric prior for the whole shape, as contained in the computed latent space. We also share a common usage as [14], but while they use an example to guide the overall style of the reconstruction, we use multiple examples, each one guiding a specific part of the shape.

3 Method

3.1 Autoencoder Foundation

Our goal is to create new object shapes, by generating variations within their different parts, in a data-driven process. The first step is to be able to recreate objects from the dataset. A natural choice is to use a generative model, we chose autoencoders. Formally, the goal is to learn the two functions E (encoder) and D (decoder) such that, for all X in the dataset:

$$X = D(E(X)) \tag{1}$$

These two functions are implemented as neural networks that operate on pointclouds, either taken as a source (for E) or as a target (for D). The key specificity of our method is our factorization of E based on the available segmentation data.

The architecture of our foundational autoencoder is the following ($N = 1024$ points):

Input: a minibatch of 32 pointclouds, each represented as a $N \times 3$ matrix, accompanied by their segmentation data (see part 3.2);
Encoder: based on Pointnet [18] but in a much simpler version, with successive layers of per-point filters followed by ReLU layers;
Code mixer: the latent space factorization step, as explained in part 3.3;
Decoder: three fully connected layers with biases, except on the last layer;
Output: the last layer is ultimately reshaped to a $N \times 3$ matrix.

3.2 Consistent Segmentation data

To demonstrate our method, we use the *airplane* category from ShapenetCore and its segmentation obtained from [20], comprising of the four following parts: body, wings, engine, tail. We restricted our analysis only to models containing the four parts, but these parts need not have the same number of points. Since all models are aligned in a consistent manner (the plane body is aligned with the Z axis), our neural networks does not need any rotational invariance, and can leverage from the strong spatial relations of the models' parts for both the encoder and decoder. Note that the value of K depends on the given dataset: for the *airplane* category, $K = 4$.

3.3 Semantic Latent Space factorization

We use pointclouds to represent surfaces, a choice that leads to the following remark: any subset of a pointcloud is a pointcloud. Although this may seem trivial, note that this is not a property that usually holds in a machine learning setting: for instance, a segmented region in an image is not typically rectangular. This allows us to replace the encoder by K encoders, each for a part, which yields the following factorization:

$$E = E_1 * E_2 * ... * E_K \tag{2}$$

$$E(X) = C = [c_1, c_2, ..., c_n], c_i = E_i(X) \tag{3}$$

where each E_i represents a *partial encoder* for part i, and evaluates what we call a *subcode*. The above product corresponds to vector concatenation. In this form, the factorization of the latent space simply corresponds to assigning parts to dedicated coordinates. Figure 1 shows a diagram of the corresponding pipeline. Note that for a given part i, the corresponding partial encoder E_i will take as input pointclouds of different sizes, since one should not assume equal parts sizes across the dataset. This limitation can be lifted thanks to the Pointnet [18] max-pooling operation. Since part i is included in the whole shape of size $N = 1024$, we know its size has to be smaller than 1024, so we can add zeros until we fill a 1024×3 matrix. Then, one just has to make sure these padded zeros remain through all the layers of the network, until the final max-pooling discards them.

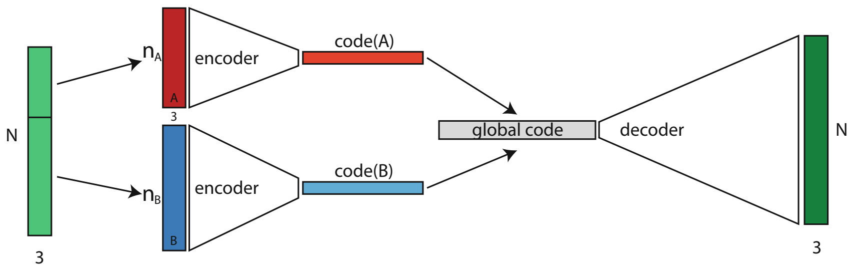


Fig. 1. Structure of our *hybrid encoder* model, illustrated for simplicity with $K = 2$ parts

3.4 Loss and Training

When it comes to pointclouds, two reconstruction losses can be considered: Earth Mover's Distance (EMD) and Chamfer Distance (CD). The former solves the optimal transport problem of transporting S_1 (where each point is seen as a Dirac delta function) onto S_2, and computing the optimal bijection ϕ:

$$d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{4}$$

Relaxing the global optimality of the assignment, Chamfer Distance computes the squared distance between each point of one pointcloud to its nearest neighbor in the other pointcloud:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{5}$$

Chamfer distance is easier to implementat, has a shorter computation time, and produces acceptable results for our usecase, so we chose to use it over EMD. The interested reader will find a comparison of generation results for both losses in [5].

This reconstruction loss becomes the objective function that is to be minimized. The reconstruction itself depends on the partial encoders and decoder networks, which are simply non-linear parametric functions. So, the learning task ultimately consists of finding the values for these parameters that minimize the objective function. As typically in machine learning, this is done by a stochastic gradient descent.

4 Experiments

We implemented our architecture using Tensorflow [1] and ran it on an Nvidia Gti1080 GPU. We trained over 40 epochs using Adam optimizer [9] with learning rate of 0.9 and a batchsize of 32.

4.1 Basic Autoencoder Mode

Since our network is based on an autoencoder, we first demonstrate its ability to reconstruct objects from the training set. Figure 2 shows examples of reconstructions, chosen to be representative of the type of objects present in our dataset. We can notice that the reconstruction quality highly depends on the sub category (not available) of the object: the typical plane present in the dataset is similar to the second column, so this is where the autoencoder concentrated most of its capacity.

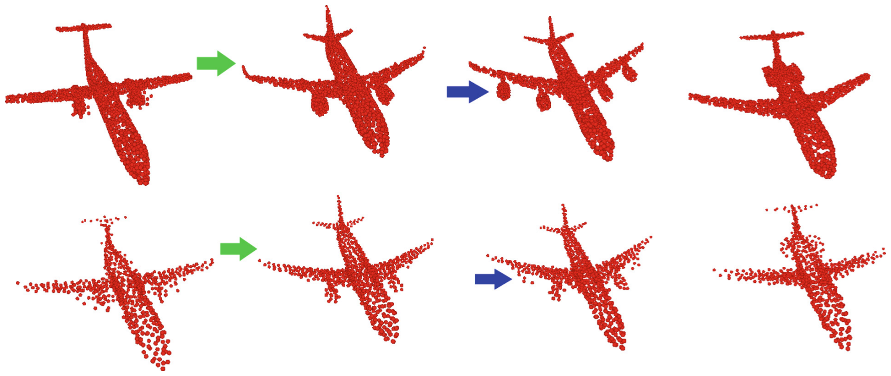
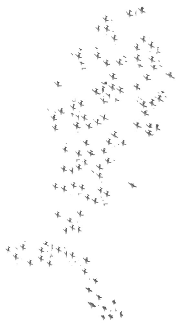
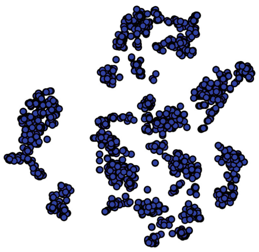


Fig. 2. Example of some reconstructions. Top row: original. Bottom row: reconstructed. The pairs of colored arrows point at errors in the predictions (Color figure online)

Clustering. The latent codes computed by E can be explored using standard dimensionality reduction techniques, such as PCA and tSNE [15]. Figure 3 shows the tSNE projection of our latent space over 2 dimensions, and snapshots of certain blobs with their corresponding shapes. Note how similar shapes live in the same blob. As with any tSNE projection, we remind the reader that distances between blobs are not significant.

Continuous Part Transfer. Thanks to the factorization of E, by simply interpolating on a given E_i, we can easily transfer a part of an object to another one while keeping the rest of the object unchanged. Let S be a source object, T the target and i the index of the part we wish to transfer from S to T. This is done with $E(T) = E_1(T) * E_2(T) * ... * E_K(T)$, replacing $E_i(T)$ by $E_i(S)$. A linear interpolation between $E_i(T)$ and $E_i(S)$ effectively realizes the continuous morphing of the part. Figure 4 shows the results of selectively transfering parts of a plane onto another one.


(a) Blob from the left


(b) Latent space

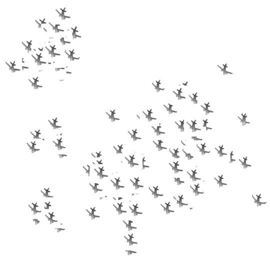
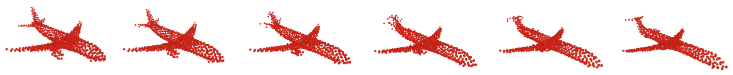
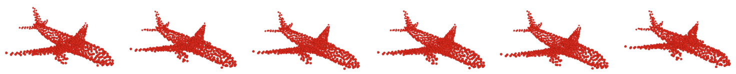
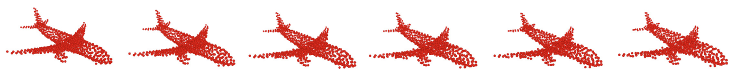
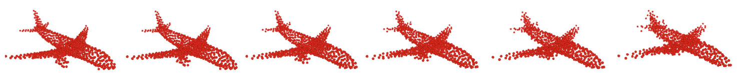

(c) Blob from the middle right

Fig. 3. tSNE projection of the encoder latent space, with close-ups of two blobs


(a) Interpolation of the whole plane


(b) Interpolation only on: *wing*


(c) Interpolation only on: *body*


(d) Interpolation only on: *engine*

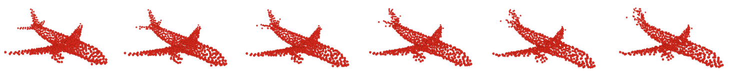

(e) Interpolation only on: *tail*

Fig. 4. Selective part transfer, compared to the global interpolation from a source to a target plane

5 Limitations and Discussion

An inherent limitation of our model is that of the autoencoder it is based on. Indeed, it suffers from a problem slightly similar to mode collapse, as shown in Fig. 2: it focuses all its reconstruction capacity towards the most frequent shapes from the dataset. In these reconstructions – running in simple autoencoder mode – see how the secondary engines (blue arrows) are just partially recovered: points that should be dedicated to them stayed on the wings. This is because such planes belong to a rare class. During training, the decoder converged to a state that favors the majority of wings, to the detriment of a minority of engines. It also means that our model cannot be suited for part transfer when one part belongs to an atypical object. Another consequence of lost small details is when they belong to a discriminative part. Let us consider once again the example of the plane with four engines: overall, the engines only have a mild contribution to the reconstruction loss. Adopting a part-specific loss could be a way of circumventing this problem.

As for part transfer, the ambiguity of our holistic design choice yields results which are sometimes hard to predict. Since we want the whole model to adapt for the new shape, we do not want to limit the geometry changes to the region of the transferred part. We are still able to swap parts but this process does not work for all parts. For several shapes, only one part – say the body of the plane – determines the geometry of the other parts, at least in the learnt latent space. So, swapping the body of such a plane with another one might lead to undesired changes in the other parts of the plane.

Moreover, our autoencoder foundation suffers the same unbalanced latent activation as reported in [17]: only a fraction of the latent dimensions have a significant contribution to the reconstruction. All these aforementioned effects, combined together, limit the current predictability of our part transfers. Further investigations are required to improve this point.

Acknowledgement. This work was made possible by an exchange program between PSL–ITI and Stanford University. The authors gratefully acknowledge the financial support coming from PSL, FJF and Fondation Armand et Janet Sibony.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). http://tensorflow.org/, software available from tensorflow.org
2. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J.: Learning Representations and Generative Models for 3D Point Clouds (2017)
3. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets (2016)
4. Doersch, C.: Tutorial on Variational Autoencoders. arXiv e-prints arXiv:1606.05908 (2016)

5. 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). http://arxiv.org/abs/1612.00603
6. Goodfellow, I.J., et al.: Generative adversarial networks (2014)
7. van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. Comput. Graph. Forum **30**(6), 1681–1707 (2011). https://doi.org/10.1111/j.1467-8659.2011.01884.x. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.01884.x
8. 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 (2012). https://doi.org/10.1145/2185520.2185551. http://doi.acm.org/10.1145/2185520.2185551
9. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv e-prints arXiv:1412.6980 (2014)
10. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. arXiv e-prints arXiv:1312.6114 (2013)
11. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.J.: GRASS: generative recursive autoencoders for shape structures. In: ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017) (2017)
12. Locatello, F., Vincent, D., Tolstikhin, I.O., Rätsch, G., Gelly, S., Schölkopf, B.: Clustering meets implicit generative models. CoRR abs/1804.11130 (2018). http://arxiv.org/abs/1804.11130
13. Lun, Z., Kalogerakis, E., Sheffer, A.: Elements of style: learning perceptual shape style similarity. ACM Trans. Graph. **34**(4), 84 (2015)
14. Lun, Z., Kalogerakis, E., Wang, R., Sheffer, A.: Functionality preserving shape style transfer. ACM Trans. Graph. **35**(6), 209:1–209:14 (2016). https://doi.org/10.1145/2980179.2980237. http://doi.acm.org/10.1145/2980179.2980237
15. van der Maaten, L., Hinton, G.E.: Visualizing high-dimensional data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
16. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. arXiv e-prints arXiv:1411.1784 (2014)
17. Nash, C., Williams, C.K.I.: The shape variational autoencoder: a deep generative model of part-segmented 3d objects. Comput. Graph. Forum **36**(5), 1–12 (2017). https://doi.org/10.1111/cgf.13240. https://doi.org/10.1111/cgf.13240
18. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3D classification and segmentation (2016)
19. 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). http://arxiv.org/abs/1706.02413
20. Yi, L., et al.: A scalable active framework for region annotation in 3D shape collections. ACM Trans. Graph. **35**(6), 2101–21012 (2016). https://doi.org/10.1145/2980179.2980238. http://doi.acm.org/10.1145/2980179.2980238