# Learning Semantic Deformation Flows with 3D Convolutional Networks SUPPLEMENTAL MATERIAL

M. Ersin Yumer, Niloy J. Mitra

Adobe Research, University College London
yumer@adobe.com, n.mitra@cs.ucl.ac.uk

## 1 Deformed Shape Sampling for Training

We utilize Yumer *et al.* 's [1] method to score and subsequently sample deformed shapes for training. Given a shape, the system first assigns a score per semantic attribute (between 0 and 1). An example from this sampling process is illustrated in Figure 1.

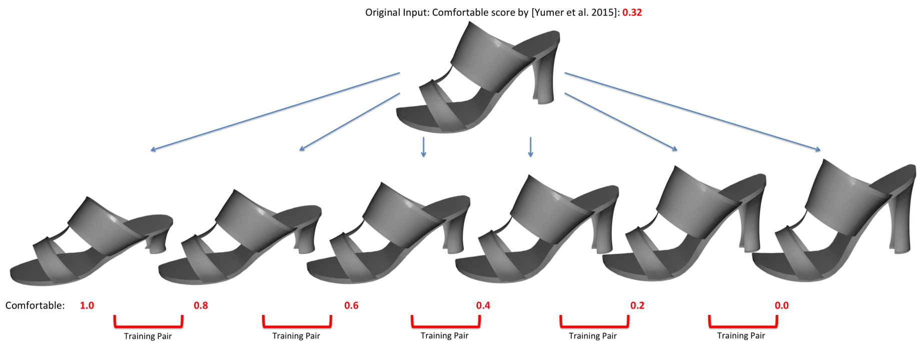

**Fig. 1.** We sample the original input at 5 steps using Yumer *et al.* 's system [1] and use the pairs for training.

## 2 Train on ShapeNet[2] – Test on SemEd[1]

Here, we present average deformation error test results where training for all methods except for Yumer *et al.* [1] is done on the ShapeNet dataset [2] (using all ShapeNet models in Table 1 of the paper) and testing on the SemEd dataset [1] (using all SemEd models in Table 1 of the paper). Note that these results are aggregate errors over the five different attributes for each shape category. This is a strain experiment because the previous work we compare with, and use to generate ground truth data for meshes [1], is trained on SemEd and utilizes a mixture of experts approach. Tables 1, 2, and 3 show that the results of this

**Table 1.** Train on ShapeNet - Test on SemEd. Mesh deformation error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error - worst performance**.

| | [1] | *k*NN | S1-32 | S2-32 | F1-16 | F2-16 | F1-32 | F2-32 | F2-32-fm |
|---|---|---|---|---|---|---|---|---|---|
| Cars | **0.0** | 7.56 | **9.46** | 8.41 | 4.87 | 3.81 | 2.05 | 1.16 | 1.04 |
| Shoes | **0.0** | 9.94 | **15.86** | 14.53 | 7.49 | 7.22 | 4.76 | 3.50 | 3.41 |
| Chairs | **0.0** | 8.78 | **9.48** | 8.29 | 7.41 | 6.55 | 4.16 | 3.24 | 2.42 |
| Airplanes | **0.0** | 14.41 | **14.44** | 13.98 | 10.55 | 9.79 | 6.56 | 6.30 | 5.10 |

**Table 2.** Train on ShapeNet - Test on SemEd. Point set deformation test error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error - worst performance**.

| | [1] | *k*NN | S1-32 | S2-32 | F1-16 | F2-16 | F1-32 | F2-32 | F2-32-fp |
|---|---|---|---|---|---|---|---|---|---|
| Cars | 1.46 | 6.48 | **11.18** | 9.24 | 6.07 | 4.34 | 2.51 | 1.58 | **0.81** |
| Shoes | 2.98 | 9.78 | 14.41 | **14.81** | 8.05 | 8.38 | 4.99 | 3.35 | **2.68** |
| Chairs | 3.02 | 8.58 | **9.99** | 8.79 | 7.62 | 6.51 | 4.63 | 3.54 | **2.43** |
| Airplanes | 5.08 | **14.51** | 13.88 | 13.65 | 11.75 | 10.00 | 6.45 | 5.96 | **4.37** |

**Table 3.** Train on ShapeNet - Test on SemEd. Depth scan deformation test error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error - worst performance**.

| | [1] | *k*NN | S1-32 | S2-32 | F1-16 | F2-16 | F1-32 | F2-32 | F2-32-fs |
|---|---|---|---|---|---|---|---|---|---|
| Cars | **12.55** | 11.45 | 10.47 | 10.09 | 5.44 | 4.65 | 2.57 | 1.65 | **1.58** |
| Shoes | 12.92 | 11.24 | **15.16** | 13.54 | 8.38 | 8.91 | 4.89 | 3.93 | **3.11** |
| Chairs | 10.02 | 10.10 | **10.91** | 9.64 | 9.46 | 6.98 | 4.94 | 4.38 | **3.26** |
| Airplanes | **16.99** | 13.90 | 14.34 | 12.86 | 12.84 | 10.59 | 6.95 | 5.73 | **5.49** |

experiment is in line with the experiments presented in the paper where training and testing is performed on a mixed case with models both from ShapeNet and SemEd datasets.

## 3 Additional Visual Results

Figure 2 shows progressive activation of different semantic attributes for deformation, using the F2-32-fm presented in the paper. Additional representative results corresponding to the tests with finetuned F2-32 networks for each data type in the paper are given in Figure 3 and Figure 4.

## 4 Baselines

### 4.1 Voxel Synthesis Baselines

Our direct synthesis baselines use the same network architectures presented in the paper by replacing the deformation flow at the output with voxelized representation of the deformed shape. We have two synthesis baselines corresponding to the two architectures in the paper: S1-32 and S2-32 (Figure 2 and Section 3.1 in the paper). Note that the output of the synthesis networks are in voxel format. We use marching cubes [3] for reconstructing a surface mesh representation for the deformed shape, in order to get a surface representation in 3D for appropriate comparison with the other methods (Figure 5).

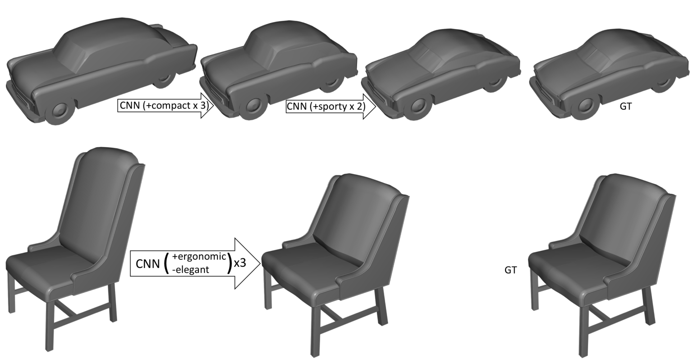

**Fig. 2.** Top: Activation of different semantic deformation indicators in successive order, Bottom: Activation of different semantic deformation indicators together.

## 4.2 *k*NN Baseline

Our *k*NN baseline is a data-driven nearest neighbor method where we leverage the *input shape-output ground truth deformation flow* pairs synthesized for training as a database. At test time query the nearest neighbor shapes to the input shape and use the corresponding deformation flows to deform the input shape. We use an Euclidean metric for nearest neighbor matching: the point set and range scan data types use point data directly, whereas for the mesh representation we use their corresponding point sets for nearest neighbor matching (we build three different nearest neighbor databases for the three data types). The nearest neighbor Euclidean metric, is computed as follows:

$$r_{ij} = \frac{1}{|\mathcal{V}_i|} \sum_{v \in \mathcal{V}_i} ||v - v_{nn-j}|| \tag{1}$$

where $r_{ij}$ is the average distance of the points in the input shape $i$, to the $j^{\text{th}}$ shape in the database. $\mathcal{V}_i$ is the set of all points in the input shape, $|\mathcal{V}_i|$ is the number of all points in the input shape, and $v_{nn-j}$ is the nearest point in the $j^{\text{th}}$ database shape to the point $v$ in the input shape.

We blend the corresponding deformation flows of the $k$ neighbors proportionally weighted by their inverse distance to the input. We experimented with various $k$ between 3 and 25, however we report our results using $k = 15$ which performed best. Specifically, we use the following blending function to compute the deformation flow:

$$\mathbf{F}_i = \sum_{k \in \mathcal{K}_{15}} \omega_k \mathbf{F}_k \tag{2}$$

where $\mathbf{F}_i$ is the deformation flow to be applied to the input, $\mathcal{K}_{15}$ is the set of 15 nearest neighbors to the input, $\omega_k$ is the weight of $k^{\text{th}}$ neighbor's weight. We compute the weights as follows with a Gaussian decay:

$$\omega_k = \frac{1}{Z} e^{-\frac{r_{ij}^2}{2\sigma}} \tag{3}$$

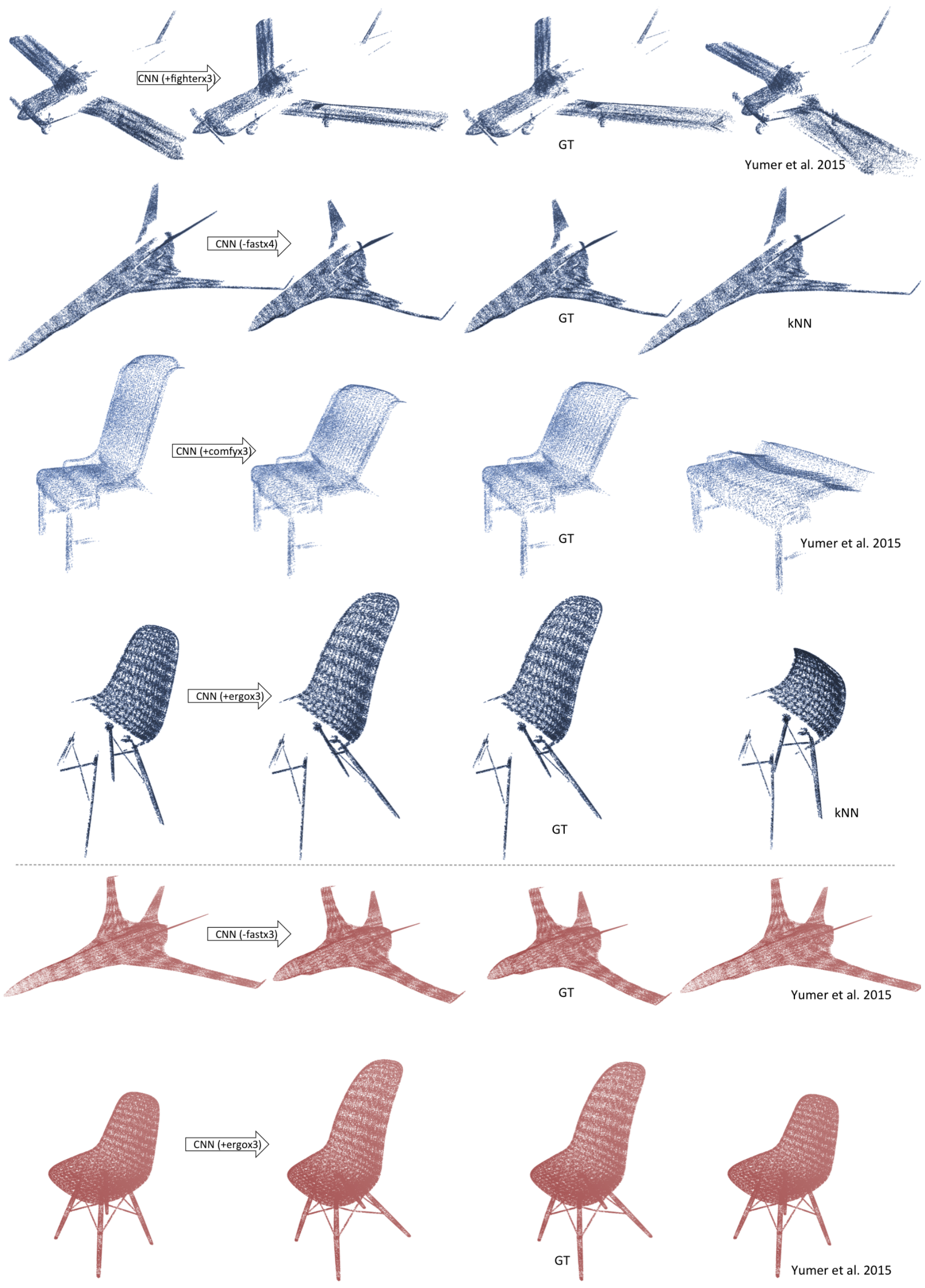

**Fig. 3.** Results from single frame depth scan (blue) and point set (red) data.

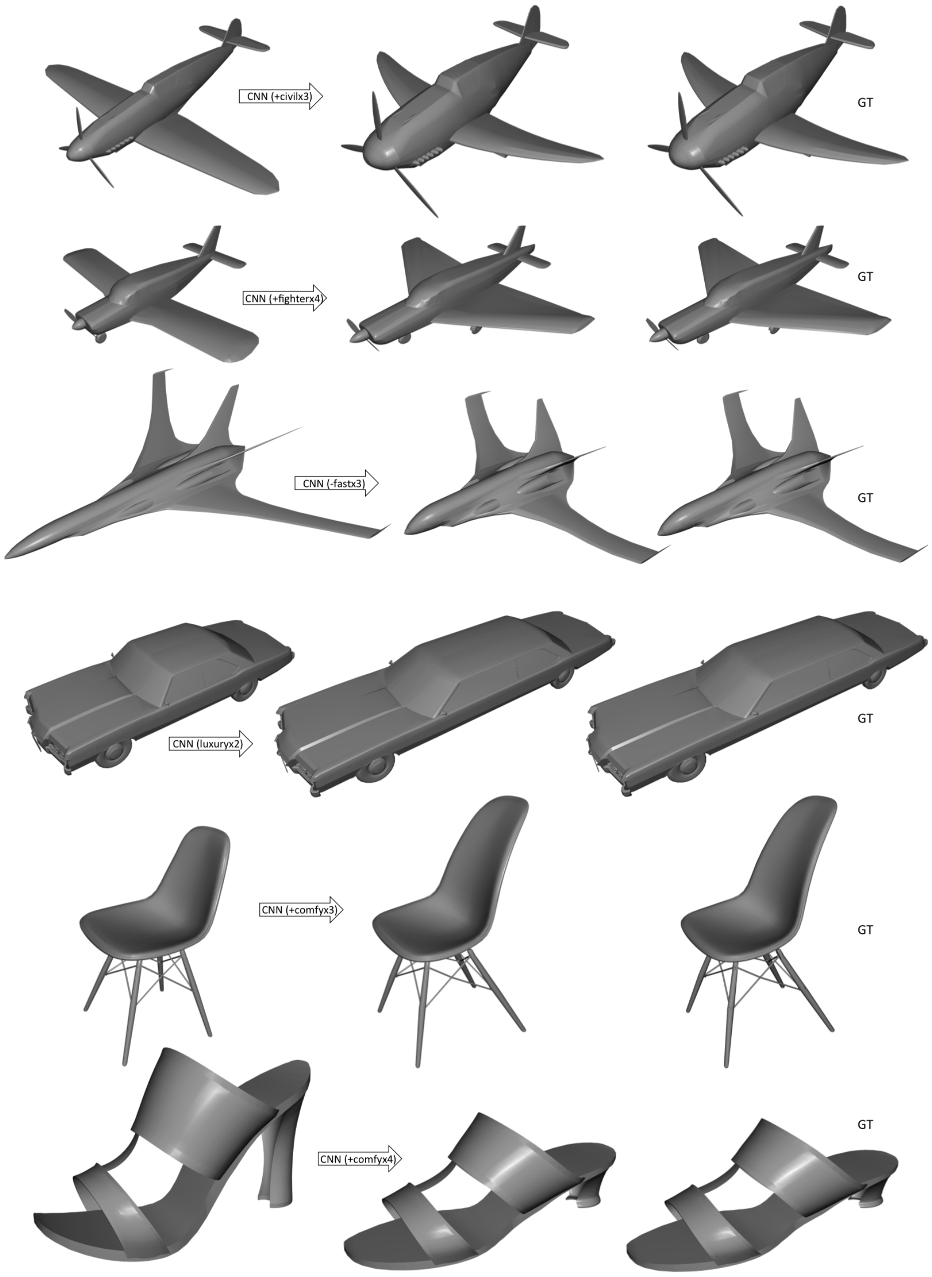

**Fig. 4.** Results from shape mesh representation data.

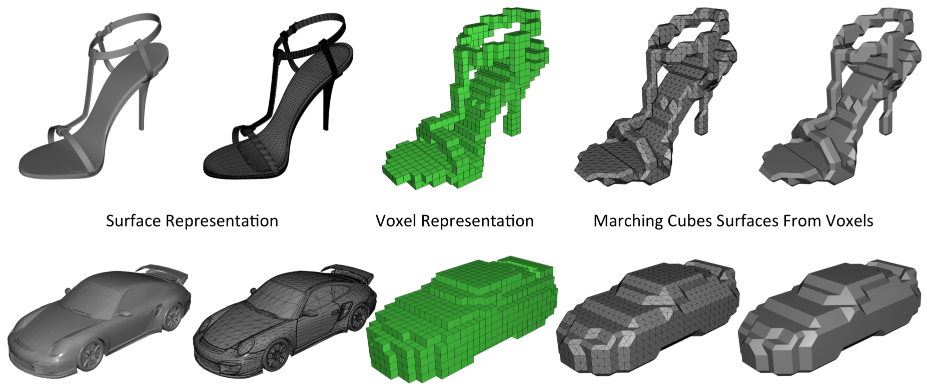

**Fig. 5.** Voxel and marching cubes surface representation examples. Note the low quality of the marching cubes reconstruction due to the low voxel resolution.

where $r_{ij}$ is computed via Equation 1, $\sigma$ is the standard deviation of the distances to the shapes in $\mathcal{K}_{15}$, and $Z$ is a normalization factor such that $\sum_{k \in \mathcal{K}_{15}} \omega_k = 1$.

## 5 Bernstein Polynomials

Here, we present a additional details regarding the Bernstein polynomials used in our deformation method where voxel vertices are used as control points of an FFD lattice.

An $n^{th}$ degree Bernstein polynomial is a linear combination of Bernstein basis polynomials [4]. Bernstein polynomials are widely used in smooth interpolation and have became practically relevant with Bézier curves [5].

A Bernstein polynomial $B_n(x)$ is given by:

$$B_n(x) = \sum_{\theta=0}^{n} \beta_{\theta,n} b_{\theta,n}(x) \tag{4}$$

where the $\beta_{\theta,n}$ is the Bernstein coefficient, which is unit in our case. $b_{\theta,n}(x)$ is the Bernstein basis function defined as:

$$b_{\theta,n}(x) = \binom{n}{\theta} x^{\theta} (1-x)^{n-\theta}. \tag{5}$$

where $\binom{n}{\theta}$ is a binomial coefficient.

## References

1. Yumer, M.E., Chaudhuri, S., Hodgins, J.K., Kara, L.B.: Semantic shape editing using deformation handles. ACM Transactions on Graphics (TOG) **34**(4) (2015) 86
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)

3. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: ACM siggraph computer graphics. Volume 21., ACM (1987) 163–169
4. Korovkin, P.: Bernstein polynomials. Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer, ISBN (2001) 979–1
5. Agoston, M.K.: Computer graphics and geometric modeling. Volume 1. Springer (2005)