Computers & Graphics 109 (2022) 65–74

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical section

Deep generation of 3D articulated models and animations from 2D stick figures☆

Alican Akman [a,*], Yusuf Sahillioğlu [b], T. Metin Sezgin [c]

[a] Imperial College London, Department of Computing, London, SW7 2BX, United Kingdom
[b] Middle East Technical University, Computer Engineering Department, Ankara, 06800, Turkey
[c] Koç University, Computer Engineering Department, Istanbul, 34450, Turkey

ARTICLE INFO

Article history:
Received 21 March 2022
Received in revised form 9 August 2022
Accepted 11 October 2022
Available online 21 October 2022

Keywords:
Computer graphics
3D model generation
Deep learning
Sketch-based shape modeling

ABSTRACT

Generating 3D models from 2D images or sketches is a widely studied important problem in computer graphics. We describe the first method to generate a 3D human model from a single sketched stick figure. In contrast to the existing human modeling techniques, our method does not require a statistical body shape model. We exploit Variational Autoencoders to develop a novel framework capable of transitioning from a simple 2D stick figure sketch, to a corresponding 3D human model. Our network learns the mapping between the input sketch and the output 3D model. Furthermore, our model learns the embedding space around these models. We demonstrate that our network can generate not only 3D models, but also 3D animations through interpolation and extrapolation in the learned embedding space. In addition to 3D human models, we produce 3D horse models in order to show the generalization ability of our framework. Extensive experiments show that our model learns to generate compatible 3D models and animations with 2D sketches.

1. Introduction

3D content is still not as big as image and video data. One of the main reasons of this lack of abundance is the labor going into the creation process. Despite the increasing number of talented artists and automated tools, it is obviously not as simple and quick as hitting a record button on the phone.

3D content is, on the other hand, as important as the image and video data since it is used in many useful pipelines ranging from 3D printing to 3D gaming and filming.

With these considerations in mind, we aim to make the important 3D content creation task simpler and faster. To this end, we train a neural network over 2D stick figure and corresponding 3D model pairs. Utilization of easy-to-sketch and sufficiently-expressive 2D stick figures is a unique feature of our system that makes our system work properly even with a moderate amount of training, e.g., 72 distinct poses of a human model and 8 distinct poses of a horse model are used. We focus on human models as they are prominent in 3D applications and extend our generation ability on horse models.

Given 2D stick figure sketches, our algorithm is able to produce visually appealing 3D point cloud models without requiring any other input such as a statistical body shape model. After an easy and seamless tweaking in the network, the system is also capable of producing dynamic 3D models, i.e., animations, between source and target stick figures as shown in Fig. 1.

2. Related work

Thanks to their natural expressive power, sketches are common modes for interaction for various graphics applications [1,2].

The majority of sketch-based 3D human modeling methods deal with re-posing a rigged input model under the guidance of user sketches. [3] performs this action by transforming imaginary lines running down a character's major bone chains, whereas [4,5] propose incremental schemes that pose characters one limb at a time. [6] proposes a skeleton-based as-rigid-as-possible deformation energy that reposes the template model using a stick figure. 2D stick figures to pose characters benefit from user annotations [7], specific priors [8], and database queries [9,10]. Bessmeltsev et al. [11] claim that ambiguity problems of all these methods can be alleviated by contour-based gesture drawing. The deep regression network of [12] utilizes contour drawing to allow face creation in minutes. Another system which takes one or more contour drawings as its input uses deep convolutional neural networks to create a variety of 3D shapes [13]. We avoid forcing the user to supply a statistical body shape model as input, hence save a significant amount of effort and time that would otherwise

☆ This article was recommended for publication by Dr. T Popa.

* Corresponding author.
E-mail address: a.akman21@imperial.ac.uk (A. Akman).

https://doi.org/10.1016/j.cag.2022.10.004

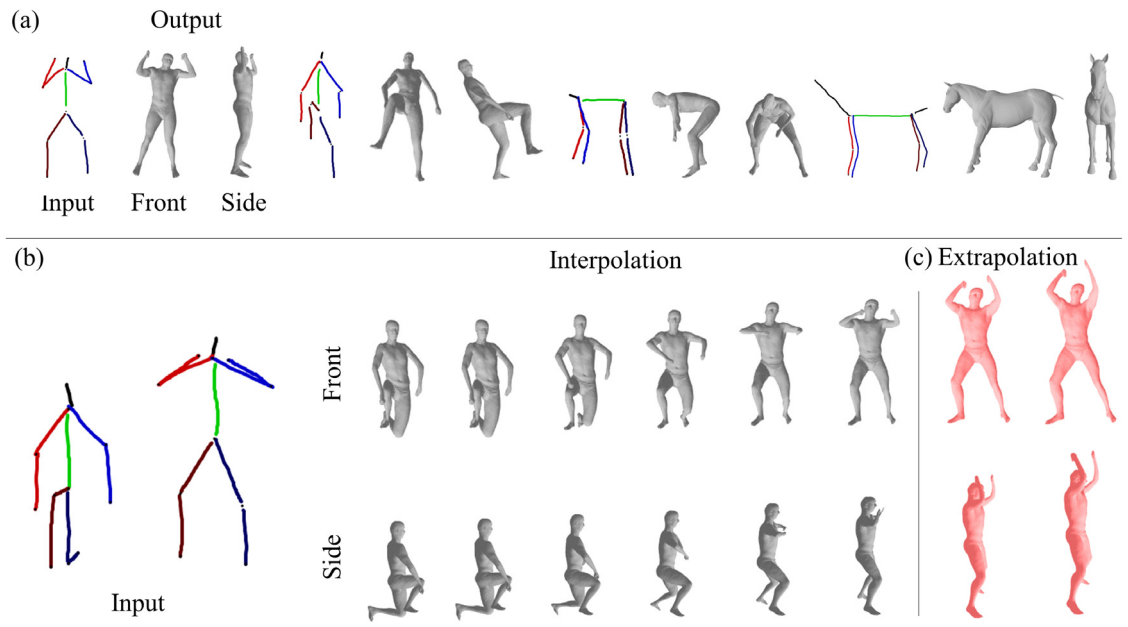

Fig. 1. Our framework is capable of performing three tasks. (a) It can generate 3D models from given 2D stick figure sketches. (b) It can generate dynamic 3D models, i.e., animations, between given source and target stick figures. (c) It can further extrapolate the produced 3D model sequence by using the learned interpolation vector.

be spent on rig creation. We merely require a user sketch, which, when fed into our network, produces the 3D model quickly in the specified pose. For example, as the network is trained with the SCAPE models [14], our resulting 3D shape looks like the SCAPE actor, i.e., a 30 year-old fit man.

There also exist sketch-based modeling methods for other specific objects such as hairs [15,16] and plants [17], as well as general-purpose methods that are not restricted to a particular object class. These generic methods, consequently, may not perform as accurately as their object-specific counterparts for those objects but still demonstrate impressive results. 3D free-form design by the pioneer Teddy model [18] is improved in FiberMesh [19] and SmoothSketch [20] by removing the potential cusps and T-junctions with the addition of features such as topological shape reconstruction and hidden contour completion. The recent SymmSketch system [21] exploits symmetry assumption to generate symmetric 3D free-form models from 2D sketches. In order to increase quality in generating 3D models, [22] focuses on piecewise-smooth man-made shapes. Their deep neural network-based system infers a set of parametric surfaces that realize the drawing in 3D. Other solutions to the sketch-based generic 3D model creation problem depend on image guidance [23,24], snapping one of the predefined primitives to the sketch by fitting its projection to the sketch lines [25], and controlled curvature variation patterns [26].

3D model generation and editing have been extended to 3D scenes as well. Dating back to 1996 [27], this line of works generally index 3D model repositories by their 2D projections from multiple views and retrieve the elements that best match the 2D sketch query [28,29]. Xu et al. [30] extend this idea further by jointly processing the sketched objects, e.g., while a single computer mouse sketch is not easy to recognize, other input sketches such as computer keyboard may provide useful cues. Sketch-based user interfaces arise in 2D image generation as well [31,32].

Sketches also arise frequently in shape retrieval applications due to their simplicity and expressive. Our focus, the human stick figure sketch, has been used successfully in [33] to retrieve 3D human models from large databases. The prominent example in this domain [34] as well as the convolutional neural network based method [35] report good performance with gesture drawings when it comes to retrieving humans. These three methods, as well as many other sketch-based retrieval methods [36], are in general successful on retrieving non-human models as well.

Although human body types under the same pose can be learned easily with moderate amounts of data through statistical shape modeling [37,38], this approach requires much greater amounts of input data to learn plausible shape poses under various deformations [39,40]. In addition to the data issue, this family of methods that are based on statistical analysis of human body shape operate directly on vertex positions, which brings the disadvantage that rotated body parts have a completely different representation. This issue is addressed with various heuristics, most successful of which leads to the SMPL model [41] that enables 3D human extraction from 2D images [42,43]. Our learning-based solution requires moderate amount of training data, and also alleviates the rotated part issue by simply populating the input data with 17 other rotated versions of each model. Our method needs to be trained on a dataset of 3D models with the same identity. In the face of auto-rigging methods that make using a rigged 3D model a more flexible approach for different identities, there are shape correspondence methods that enable our framework to be used for different identities. A survey of recent works in shape correspondence is provided in [44].

3. Overview

We have two main objectives: (i) Generating 3D models from a single sketched stick figure, (ii) creating 3D animations between two 3D models, generated from 2D source and target sketches. In addition, we present an application that allows interactive modeling using our algorithm.

Our approach is powered by a Variational Autoencoder network (VAE). We train this network with pairs of 3D and 2D points. The 3D points come from the SCAPE 3D human figure database and TOSCA 3D horse figure database, while the 2D points are obtained by projecting joint positions of these models on a 2D surface. Hence the correspondence information is preserved. Our

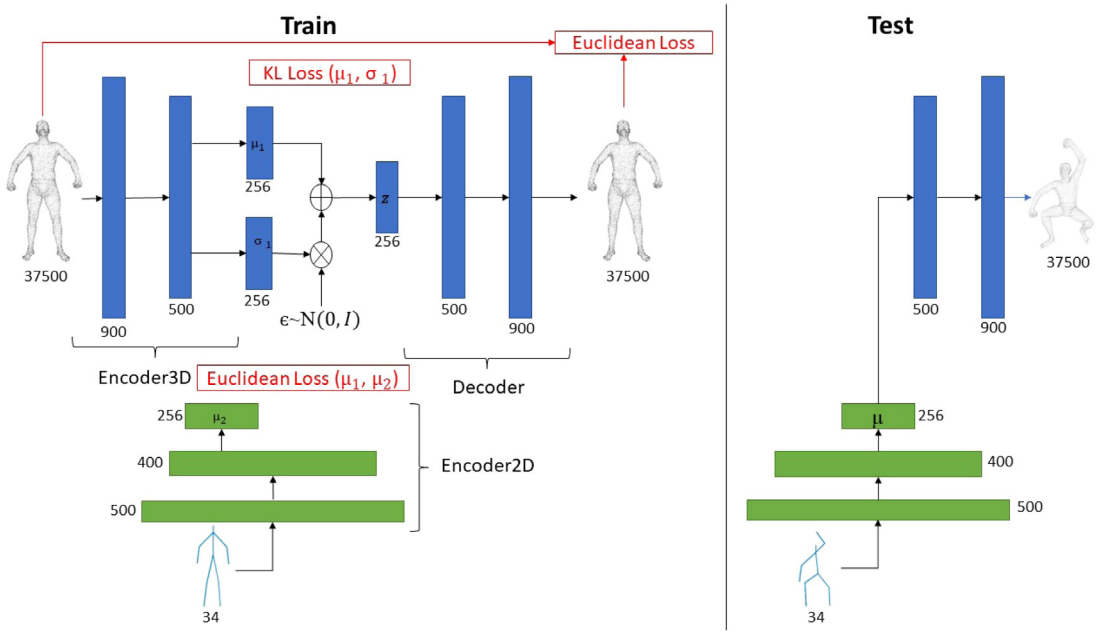

Fig. 2. Our neural network architecture. **(a) Train Network:** We train this network with (3D point cloud, 2D points of stick figure) pairs during training time. It consists of a VAE: Encoder3D and Decoder consecutively, and another external encoder: Encoder2D. We use regression loss from the output of Encoder2D to the mean vector of the VAE in addition to standard losses of VAE. While $\oplus$ represents vector addition, $\otimes$ represents multiplication for the reparameterization trick to sample the latent vector z. **(b) Test Network:** We remove Encoder3D and reparameterization layer from our VAE and use Encoder2D-Decoder as our network in our experiments.

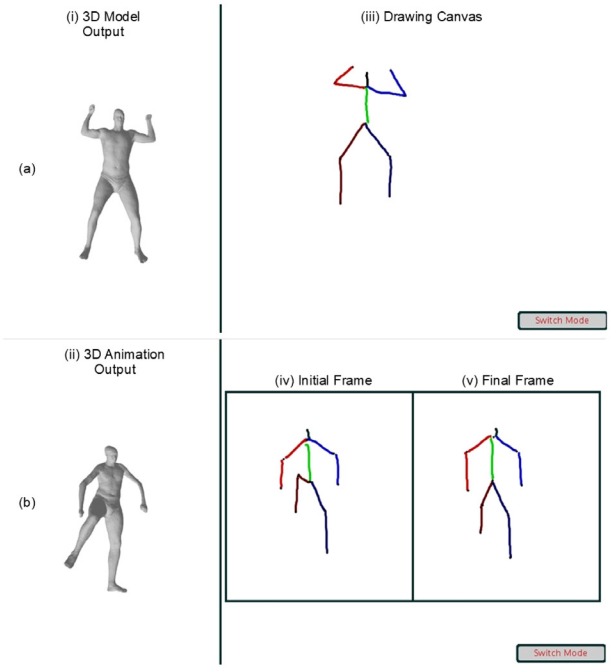

Fig. 3. Screenshots from our user interface. (a) 3D model generation mode. (b) 3D animation generation mode. The users sketch either (iii) a 2D stick figure to generate the corresponding 3D model, or (iv, v) sketch the initial and final frames to generate corresponding 3D animation.

neural network model ties the 2D and 3D representations through a latent space, which allows us to generate 3D point clouds from 2D stick figures.

The latent space that ties the 2D and 3D representations also acts as a convenient lower dimensional embedding for interpolation and extrapolation. Given a set of target key frames in the form of 2D drawings, we can map them into the lower dimensional embedding space, and then interpolate between them to obtain a number of intermediate points in the embedding space. These intermediate points can then be mapped to 3D through the network to obtain a smooth 3D animation sequence. Furthermore, extrapolation allows extending the animation sequence beyond the target key frames.

4. Methodology

Our method aims to generate static and dynamic 3D models from scratch, that is, we require only the 2D input sketch and no other data such as a statistical body shape model waiting to be reposed. To make this possible, we learn a model that maps 2D stick figures to 3D models.

4.1. Training data generation

The original SCAPE [14] and TOSCA [45] datasets consist of 72 key 3D meshes of a human actor and 8 key 3D meshes of a horse respectively. They also contain point-to-point correspondence information between these distinct model poses. We use a simple algorithm to extend these datasets by rotating the existing figures with different angles. First, we determine the axes and the angles of the rotation with respect to the original coordinate system shown in the wrapped figure. We ignore the rotation with respect to the x-axis, since stick figures are less likely to be drawn from this view. Next, we rotate the models with respect to the y-axis and z-axis, in a range of -90 degree to 90 at intervals of 30 degrees for the y-axis, and -45 to 45 degrees at intervals of 45 degrees for the z-axis. In the end of this process, we output 21 models per key model in the SCAPE and TOSCA datasets.

Since our network is trained with (2D joints, 3D model) pairs, we also extract 2D joints from a 3D model in a particular perspective. For SCAPE dataset, we designate 11 essential points that alone can describe a 3D human pose. These are the following: forehead(1), elbows(2), hands(2), neck(1), abdomen(1),

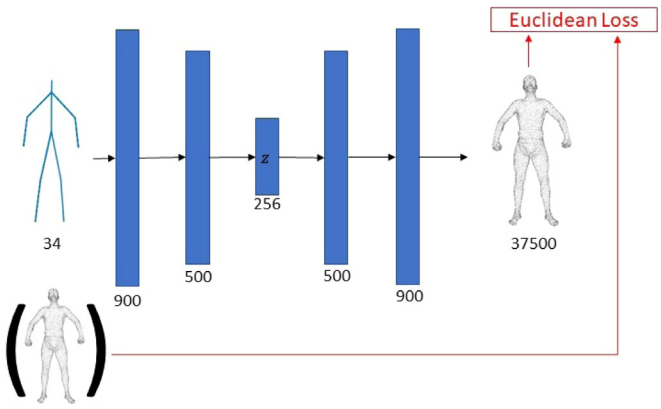

Fig. 4. Neural network architecture for standard autoencoder baseline.

knees(2) and feet(2). These essential points extends to 12 for the TOSCA dataset: forehead(1), shoulder joint(1), hip joint(1) knees(4), feet(4) and tail(1). Since the datasets have the point-to-point correspondence information in itself, we select these 3D points in a pilot mesh from each dataset. We project these joints onto a 2D camera plane ($x - y$ in our case) across the entire datasets to create 2D joint projections. In order to be independent from the coordinate system, we represent these points with relative positions (Δx, Δy). We determine two separate specific orders in 2D points forming a sketching path with 17 points for our human model and 20 points for our horse model (some joints are visited twice but in the reverse direction). These sketching paths represent the order of visited body joints while sketching. For example, users should follow this path for the human stick figure: Forehead to neck, neck to shoulders (right parts first), shoulders to elbows, elbows to hands, neck to abdomen, abdomen to knees, knees to feet. Each sketching path determines the order of 2D points that form the input vector of our neural network. The input vector format also handles front/back ambiguity while generating 3D models. We set the first point in the sketching path as the origin, $(0, 0)$ and then we set the remaining points with respect to their relative position to the preceding point.

4.2. Neural network architecture

We build upon the work of Girdhar et al. [46] while designing our neural network architecture. Girdhar et al. aim to learn a vector representation that is *generative*, i.e., it can generate 3D models in the form of voxels; and *predictable*, i.e., it can be predicted by a 2D image. We utilize variational autoencoders rather than standard autoencoders to build our neural network as shown in Fig. 2. Unlike standard autoencoders, VAEs are generative networks whose latent distribution is regularized during the training in order to be close to a standard Gaussian distribution. This property of VAEs ensures that its latent distribution has a meaningful organization which allows us to generate novel 3D models by sampling in this distribution. In addition to generating novel 3D models, since our framework is capable of learning the vector space around these 3D models, it enables meaningful transitions between them and extrapolations beyond them.

For our training network, we have two encoders and one decoder for each dataset: Encoder3D, Encoder2D, and Decoder. Encoder3D and Decoder together serve as a VAE. Our VAE takes in a 3D point cloud as input, and reconstructs the same model as output. While our VAE learns to reconstruct 3D models, it forces latent distribution of the dataset to approximate normal distribution which makes the latent space interpolatable. Meanwhile, we use our Encoder2D to predict latent vectors of corresponding 3D models from 2D points. In order to provide our latent distribution with similarity information between 3D models, we design this partial architecture for our neural network instead of using a VAE which directly generates 3D models from 2D sketches. Thus, our Encoder3D is capable of learning relations between 3D models rather than 2D sketches while creating a regularized latent distribution. With this method, we aim to explore latent space better and generate more meaningful transitions between 3D models. It also prevents the model from the training difficulties of a direct VAE where low-dimensional 2D space (34) is directly mapped to high-dimensional 3D space (37 500).

Encoder3D-Decoder VAE Network Architecture:

Our VAE takes $12\,500 \times 3$ points of human 3D model or $19\,248 \times 3$ points of horse 3D model as input. Encoder3D contains two fully connected layers and its outputs are a mean vector and a deviation vector. We use ReLU as an activation layer in all the internal layers. There is no activation layer in the output layers. Our Decoder takes the latent vector z as input. It also consists of two fully connected layers with a ReLU activation layer and one

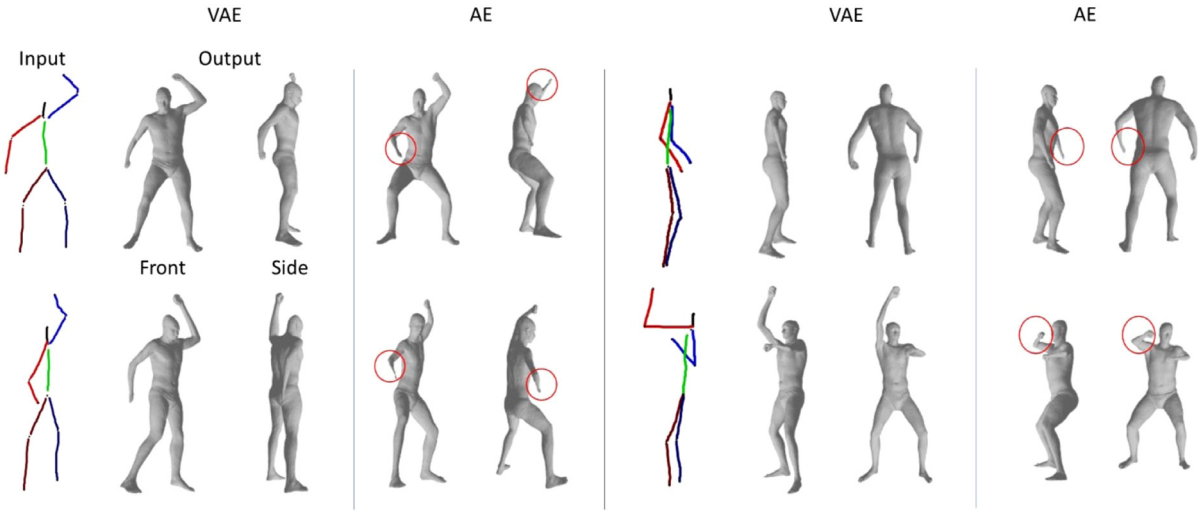

Fig. 5. Generated 3D human models in front and side views for given sketches using our VAE network and standard AE network. Flaws are highlighted with red circles.

fully connected output layer with a *tanh* activation layer. It gives a reconstructed point cloud of the input 3D model as output.

We train our VAE with the standard KL divergence and reconstruction losses. The total loss for our VAE is given in Eq. (1).

$$L_{\text{VAE}} = \alpha \frac{1}{BN} \sum_{i=1}^{B} \sum_{j=1}^{N} (x_{ij} - \hat{x}_{ij})^2 + D_{\text{KL}}(q(z|x) \parallel p(z)) \tag{1}$$

In Eq. (1), α is a parameter to balance between the reconstruction loss and KL divergence loss, B is the training batch size, N is the 1D dimension of vectorized point cloud (37 500 in our case for human models), x_{ij} is the jth dimension of ith model in the training data batch, $\hat{x}_{ij}$ is the jth dimension of model i's output from our VAE, z is the reparameterized latent vector, $p(z)$ is the prior probability, $q(z|f)$ is the posterior probability, and D_{KL} is KL divergence.

Mapping 2D Sketch Points to Latent Vector Space:

Our Encoder2D learns to predict the latent vectors of 3D models that correspond to 2D sketches as discussed. It takes 17×2 points of human stick figure or 20×2 points of horse stick figure as input to map it into mean vector. It has the same structure with Encoder3D except its input and internal fully connected layers' dimensions. In the test case we use Encoder2D and Decoder as a standard autoencoder. Decoder takes the mean vector output of Encoder2D as its input and generates 3D point cloud as the output.

We train our Encoder2D with mean square loss to regress 256D representation of mean vector given by pre-trained Encoder3D. The loss for our Encoder2D is given in Eq. (2).

$$L_{\text{2D}} = \frac{1}{BZ} \sum_{i=1}^{B} \sum_{j=1}^{Z} (\mu_{ij}^1 - \mu_{ij}^2)^2 \tag{2}$$

In Eq. (2), B is the training batch size, Z is the dimension of latent space (256 in our case), μ_{ij}^1 is the jth dimension of mean vector produced by Encoder3D to ith model in training batch, and μ_{ij}^2 is the jth dimension of mean vector produced by Encoder2D to ith model in the training batch.

4.3. Training details

We follow a three-stage process to train our network for each dataset. (i) We train our variational auto encoder independently with the loss function in Eq. (1). We run this stage for 300 epochs. (ii) We train our Encoder2D with the loss function in Eq. (2) using Encoder3D trained from (i). Specifically, we train our Encoder2D to regress the latent vector produced from the pre-trained Encoder3D for the input 3D model. We run this stage for 300 epochs. (iii) We use both losses jointly to fine-tune our framework. We run this stage for 200 epochs. It takes about two days to complete the whole training session.

For the experiments throughout this paper, we set $\alpha = 10^5$. We set the prior probability over latent variables as a standard normal distribution, $p(z) = \mathcal{N}(z; 0, I)$. We set the learning rate as 10^{-3} and 10^4 for our human model generation and horse model generation networks respectively. We use the Adam Optimizer as our optimizer in training.

4.4. User interface

As we have explained in prior sections, our method can be used in a variety of applications in different fields such as character generation and making quick animations. In order to better utilize these applications we propose a user interface with facilitative properties that enables users to perform our method in a better manner (Fig. 3). Our user interface acts as an agent that ensures the input–output communication between the user and our neural network. The user first chooses whether to generate human 3D models or horse 3D models. Then, the user can choose whether to generate 3D models from 2D stick figure sketches or create animations between the source and target stick figure sketches. While it takes about one second to process a sketch input for generation of the corresponding 3D model,

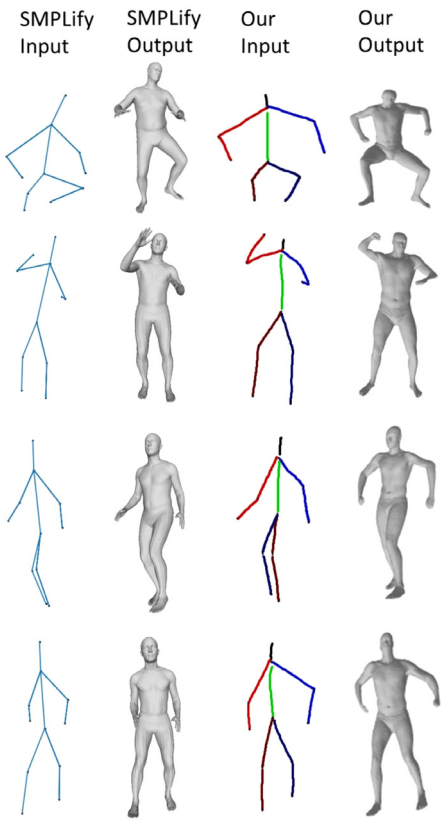

Fig. 6. Qualitative comparison of our method (columns 3 and 4) with [47] (columns 1 and 2).

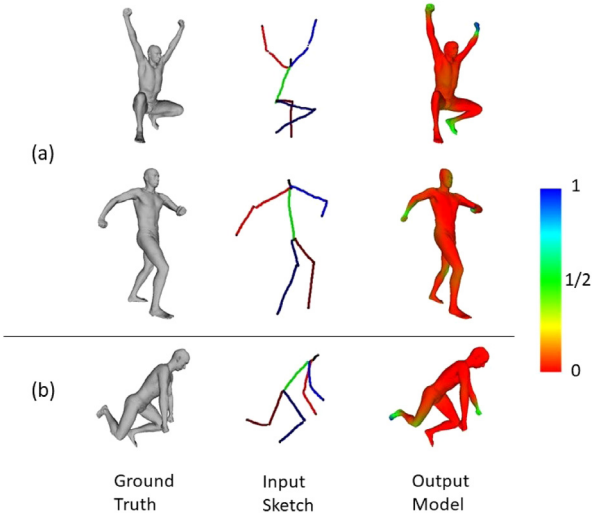

Fig. 7. Generation results for (a) two human stick figure sketches observed in training and (b) an unobserved human stick figure sketch. The generated models (last column) are colored with respect to the distance map to the ground truth (first column). Distance values are normalized between 0 and 1.

Table 1
Per-vertex reconstruction error on validation data with different network architectures. VAE represents our method.

Method	Z (Latent Dim.)	Mean ($\times 10^{-3}$)	Std ($\times 10^{-3}$)
AE	256	2.996	1.568
VAE	256	2.118	2.598

Table 2
Per-vertex reconstruction error on validation data with different latent dimensions. 256 is our promoted latent space dimension.

Method	Z (Latent Dim.)	Mean ($\times 10^{-3}$)	Std ($\times 10^{-3}$)
VAE	128	3.887	3.802
VAE	256	2.118	2.598
VAE	512	10.830	6.888

this time extends approximately to five seconds for producing animation between (interpolation) and beyond (extrapolation) sketch inputs.

Our user interface takes a sketch input from the user via an embedded 2D canvas (Fig. 3(iii)). The users are required to sketch in a predetermined path that allows us to know the order of sketched vectors of body parts. The collected sketch is transformed into a map of the joint locations as an input to our neural network using this information. The user interface then shows the 3D model output produced by the neural network on the embedded 3D canvas (Fig. 3(i)). For 3D animation generation, the users are required to sketch the initial and final frames (Fig. 3(iv, v)). Although our output is a 3D point cloud, for better visualization, our user interface utilizes the mesh information that already exists in the SCAPE and TOSCA datasets. Polygon mesh information is provided in these datasets in order to build 3D model shape on top of 3D cloud. Since this information is shared across each dataset due to point-to-point correspondence and our framework does not break this correspondence during generation process, we could embed this information in our generated 3D models. Generated point clouds are consequently combined with this information to display 3D models as surfaces with appropriate rendering mechanisms such as shading and lighting.

Since we trained an abundance of neural networks until achieving the best one with the optimal parameters, our user interface showed two different 3D model outputs coming from two different neural networks for comparisons during the development phase. The interface in Fig. 3 belongs to our release mode where only the promoted 3D output is displayed.

We construct our user interface such that it is purified from unnatural interaction tools such as buttons and menus. Generation process starts as soon as the last stroke is drawn without forcing the user to hit the start button. We provide brief information in text that describes the canvas organization. To make the interaction more fluent, we add a simple "scribble" detector to understand the undo operation.

5. Experiments and results

In this section, we first evaluate our framework qualitatively and quantitatively. We evaluate three tasks performed by our framework. (i) Generating 3D models from 2D stick figure sketches. (ii) Generating 3D animations between source and target stick figure sketches. (iii) Performing simple extrapolation beyond stick figures. All the input stick figures in the figures were provided by 6 novice users composed of undergraduate and graduate students.

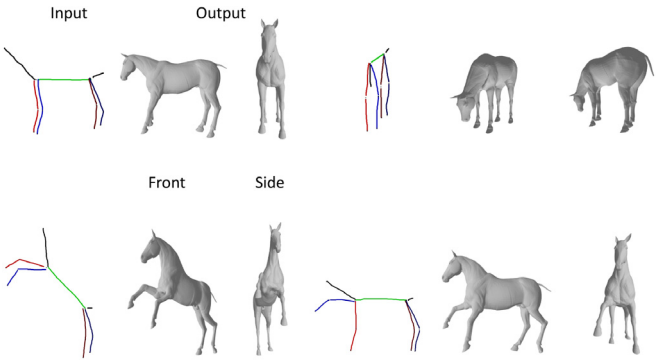

Fig. 8. Generated 3D horse models in front and side views for given sketches using our VAE network.

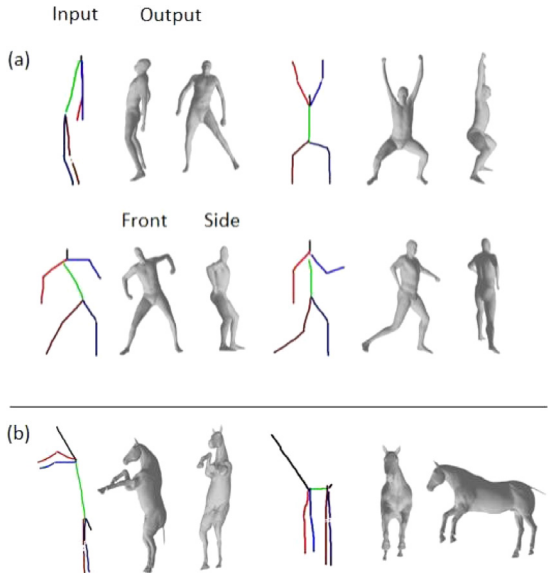

Fig. 9. More generation results for (a) human models and (b) horse models.

5.1. Framework evaluation

Standard Autoencoder Baseline. To quantitatively justify our preference on variational autoencoders, we design a standard autoencoder (AE) baseline with similar dimensions and activation functions (Fig. 4). We train this network on SCAPE dataset for 300 epochs with Euclidean loss between generated 3D models and ground truth ones. We compare the per-vertex reconstruction loss on a validation set consisting of held-out 3D models with our VAE network and standard AE network trained with human models. The results in Table 1 shows that our VAE network outperforms AE network, exploiting latent space more efficiently to enhance the generation quality of novel 3D models.

Latent Dimensions. We evaluate different latent dimensions for training our VAE framework with SCAPE dataset using per-vertex reconstruction loss on a validation set. The results in Table 2 show that using 256 dimensions improves generation quality compared to lower dimensions. Higher dimensions lead to overfitting. We use 256 dimensions for the following experiments.

5.2. Generating 3D models from 2D stick figure sketches

To evaluate the generation ability, we feed 2D human stick figure sketches as input to our framework. Our user interface

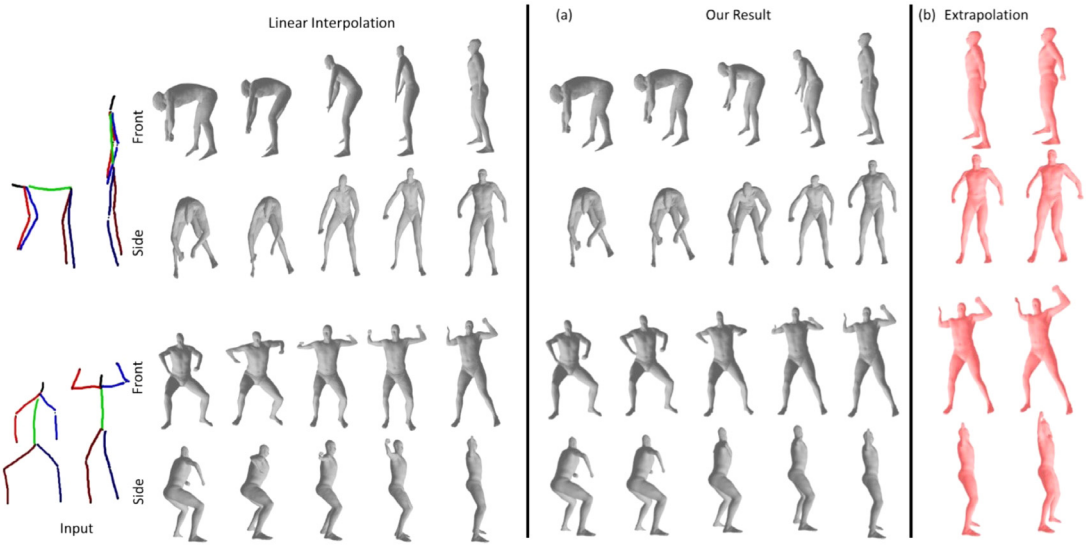

Fig. 10. Qualitative comparison with linear interpolation. (a) Produced 3D model sequences for given sketches using our network are better than linear interpolation results. (b) Extrapolation results for 3D model sequences are given as red models.

is used for the sketching activity. In Fig. 5, we compare generation results for sample sketches with our VAE network and standard AE baseline. These results show that standard AE network generates models with anatomical flaws (collapsed arm in Fig. 5) and deficient body orientations. Our VAE network produces compatible models of high quality.

We compare the generation ability of our framework with a recent study [47] that predicts 3D shape from 2D joints of human. While our framework outputs 3D point clouds, the method described in [47], tries to fit a statistical body shape model, SMPL, on 2D joints. Their learned statistical model is a male who is in a different shape than our male model as shown in the second column of Fig. 6. We take the visuals reported in their paper and draw the corresponding human stick figures for a fair comparison. Despite being restricted to the reported poses in [47], our method compares favorably. The sitting pose, for instance, which is not quite captured by their statistical model shows in an inaccurate 3D sitting while our 3D model sits successfully (Fig. 6 - top row).

We also compare our 3D human model generation ability to the ground-truth by feeding sketches that resemble 3 of the 72 SCAPE poses used during training. Two of these poses were observed during training at the same orientations as we draw them in the first two rows of Fig. 7, yet the last one is drawn with a new orientation that was not observed during training (Fig. 7-last row). Consequently, we had to align the generated model with the ground-truth model via ICP alignment [48] for the last row prior to the comparison. We observe almost perfect replications of the SCAPE poses in all cases as depicted by the colored difference maps.

In order to show generalization ability of our method with different 3D model types, we evaluate our VAE framework which is trained on the TOSCA dataset. To evaluate the generation ability with 3D horse models, we feed 2D horse stick figure sketches as input to our framework. In Fig. 8, we show generation results for sample sketches with our VAE network. We also show more generation results for both human models and horse models in Fig. 9.

5.3. Generation of dynamic 3D models - Interpolation

We test the capability of our network to generate 3D human animations between source and target stick figures. To accomplish this, our framework takes two stick figure sketches via our user interface. The framework then encodes the stick figures into the embedding space to extract their latent variables. We create a list of latent variables by linearly interpolating between the source and the target latent variables. We feed this list as an input to our decoder, with each element of the list being fed one by one to produce the desired 3D model sequence. In Fig. 10(a), we compare our results with direct linear interpolation between source and target 3D model output. Our results show that the interpolation in embedding space can avoid anatomical errors which usually occur in methods using direct linear interpolation. Further interpolation results can be found in the teaser image and the accompanying video.

We also compare our 3D human model generation ability with interpolation at the sketch level. In order to implement this, we design an algorithm that interpolates between source and target stick figure sketches. Our algorithm first finds the best rotation vector between each component of source and target stick figure sketches using the proposed method in [49]. Then it generates interpolated rotations in the form of quaternions using quaternion spherical interpolation. We obtain the list of interpolated sketches with these interpolated rotations. We feed this sketch list as an input to our decoder, with each element of the list being fed one by one to produce the desired 3D model sequence. In Fig. 11, we compare our embedding space interpolation results with sketch level interpolation. The 3D model generation results on interpolated sketches shows the high adaptation capacity of our framework to novel sketches. However, interpolation in embedding space results in more meaningful transition between source and target poses. For example, in the sketch interpolation results in Fig. 11, the leg of the middle 3D model unnecessarily turns right due to similar pose information coming from the input sketch as pointed out in the red circle.

5.4. Generation of dynamic 3D models - Extrapolation

Our results show that the interpolation vector in the embedding space is capable of reasonably representing the motion in real world actions. To improve upon this idea, we exploit the learned interpolation vector in order to predict future movements. We show our results for extrapolation in Fig. 10(b). This figure shows that the learned interpolation vector between two 3D shapes contains meaningful information of movement in 3D. Further extrapolation results can be found in the teaser image and the accompanying video.

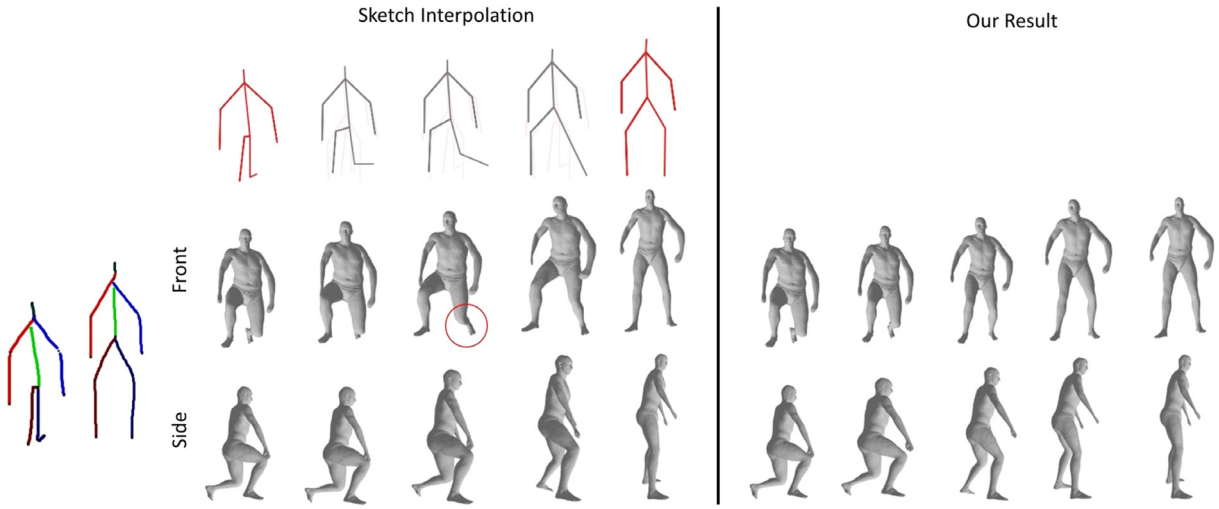

Fig. 11. Qualitative comparison with interpolation at the sketch level. The transition between produced 3D model sequences using interpolation in embedding space are more meaningful than interpolation at sketch level.

5.5. Timing

We finish our evaluations with our timing information. The closest work to our framework reports 10 min of production time for an amateur user to create 3D faces from 2D sketches [12]. In our system, on the other hand, it takes an amateur user 10 s of stick figure drawing and 1 s of algorithm processing to create 3D bodies from 2D sketches. There are two main reasons for this remarkable performance advantage of our framework: (i) Human bodies can be naturally represented with easy-to-draw stick figures whereas faces cannot. The simplicity and expressive make the learning easier and more efficient. (ii) Our deep regression network is significantly less complex than the one employed in [12]. In another work, the authors propose a framework which takes as input a vector-format rough gesture drawing, and rigged 3D character model, then poses the character to conform to the depicted pose [50]. This work requires an existing 3D model, which takes several minutes to create. In contrast, we can generate 3D models from scratch in a matter of seconds.

Our application runs on a PC with 8 GB ram and i7 2.80 GHz CPU. Training of our model is done on Tesla K40 m GPU and took about 2 days (800 epochs total).

6. Conclusions

In this paper, we presented a deep learning based framework that is capable of generating 3D models from 2D stick figure sketches and producing dynamic 3D models between (interpolation) and beyond (extrapolation) two given stick figure sketches. Unlike existing methods, our method does not require a statistical body shape model. We demonstrated that our framework not only gives compatible results on generation, but also compares favorably with existing approaches. We further supported our framework with a well-designed user interface to make it practical for a variety of applications.

7. Limitations

The proposed system has several limitations that are listed as follows:

- Training of our network is dependent on the existing 3D shapes in the dataset. Our network cannot learn vastly different shapes than existing ones: it produces incompatible 3D models with sketch inputs that are not closely represented in the dataset. For example, our network does not correctly capture the arm orientation for the right model in Fig. 12.
- The system can only generate human shapes because of the content of the dataset.
- The system can produce articulated shapes. Although it can twist and bend 3D model body, it cannot, for instance, stretch or resize a body part.
- The system benefits from the one-to-one correspondence information of the dataset. Thus, the quality of results depends on this information.
- Since our network takes its input in a specific order, our user interface constrains users to sketch in that order. Users cannot sketch stick figures in an arbitrary order.

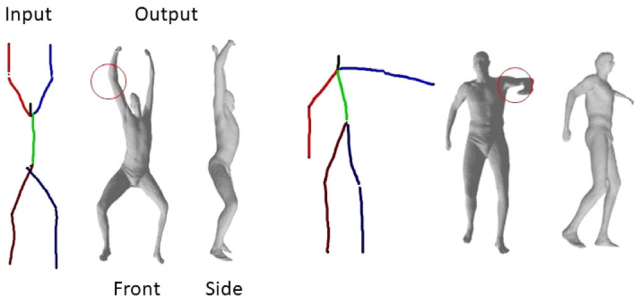

Fig. 12. Failure cases of our framework. Our framework generates anatomically unnatural results if the input sketch has disproportionate body parts (left), or it is significantly different than the ones used during training (right).

8. Future work

Potential future work directions that align with our proposed system are described as follows. Human stick figures used in this paper can be generalized to any other shape using their skeletons. Consequently, an automated skeleton extraction algorithm would enable further training of our network, which in turn extends our solution to non-human objects. Voxelization of our input data would spare us from the one-to-one correspondence information requirement, which in turn would enable our interpolation scheme to morph from different object classes that do not often have this type of information, e.g., from cat to giraffe. Automatic one-to-one correspondence computation [51–53] can also be considered to avoid voxelization. Latent space can be exploited in a

better manner in order to obtain a more sophisticated extrapolation algorithm than the basic one we introduced in this paper. New sketching cues can be designed and incorporated into our network to be able to produce body types different than the one used during training, e.g., training with the fit SCAPE actor and production with an obese actress.

CRediT authorship contribution statement

Alican Akman: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Visualization, Project administration, Funding acquisition. **Yusuf Sahillioğlu:** Writing – review & editing, Supervision, Project administration. **T. Metin Sezgin:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have used publicly available 3D model datasets.

Acknowledgements

This work was supported by TUBITAK under the project EEEAG-115E471 and EEEAG-119E572, and by European Commission ERA-NET Program, The IMOTION project.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2022.10.004.

References

[1] Sezgin TM, Stahovich T, Davis R. Sketch based interfaces: early processing for sketch understanding. In: Siggraph 2006 courses. ACM; 2006, p. 22.
[2] Olsen L, Samavati FF, Sousa MC, Jorge JA. Sketch-based modeling: A survey. Comput Graph 2009;33(1):85–103.
[3] Guay M, Cani M-P, Ronfard R. The line of action: an intuitive interface for expressive character posing. ACM Trans Graph 2013;32(6):205.
[4] Öztireli AC, Baran I, Popa T, Dalstein B, Sumner RW, Gross M. Differential blending for expressive sketch-based posing. In: Proceedings of the 12th ACM SIGGRAPH/eurographics symposium on computer animation. ACM; 2013, p. 155–64.
[5] Hahn F, Mutzel F, Coros S, Thomaszewski B, Nitti M, Gross M, Sumner RW. Sketch abstractions for character posing. In: Proceedings of the 14th ACM SIGGRAPH/eurographics symposium on computer animation. ACM; 2015, p. 185–91.
[6] alar Seylan, Sahilliolu Y. 3D shape deformation using stick figures. Comput Aided Des 2022;151:103352. http://dx.doi.org/10.1016/j.cad.2022.103352, URL: https://www.sciencedirect.com/science/article/pii/S0010448522001075.
[7] Davis J, Agrawala M, Chuang E, Popović Z, Salesin D. A sketching interface for articulated figure animation. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association; 2003, p. 320–8.
[8] Lin J, Igarashi T, Mitani J, Liao M, He Y. A sketching interface for sitting pose design in the virtual environment. IEEE Trans Vis Comput Graphics 2012;18(11):1979–91.
[9] Wei XK, Chai J, et al. Intuitive interactive human-character posing with millions of example poses. IEEE Comput Graph Appl 2011;31(4):78–88.
[10] Choi MG, Yang K, Igarashi T, Mitani J, Lee J. Retrieval and visualization of human motion data via stick figures. In: Computer graphics forum, Vol. 31. Wiley Online Library; 2012, p. 2057–65.
[11] Bessmeltsev M, Vining N, Sheffer A. Gesture3D: posing 3D characters via gesture drawings. ACM Trans Graph 2016;35(6):165.
[12] Han X, Gao C, Yu Y. DeepSketch2Face: A deep learning based sketching system for 3D face and caricature modeling. ACM Trans Graph 2017;36(4):126.
[13] Delanoy J, Aubry M, Isola P, Efros A, Bousseau A. 3D sketching using multi-view deep volumetric prediction. Proc ACM Comput Graph Interact Tech 2018;1(21). URL: http://www-sop.inria.fr/reves/Basilic/2018/DAIEB18.
[14] Anguelov D, Srinivasan P, Koller D, Thrun S, Rodgers J, Davis J. SCAPE: Shape completion and animation of people. ACM Trans Graph 2005;24(3):408–16. http://dx.doi.org/10.1145/1073204.1073207, URL: http://doi.acm.org/10.1145/1073204.1073207.
[15] Fu H, Wei Y, Tai C-L, Quan L. Sketching hairstyles. In: Proceedings of the 4th eurographics workshop on sketch-based interfaces and modeling. ACM; 2007, p. 31–6.
[16] Seki S, Igarashi T. Sketch-based 3D hair posing by contour drawings. In: Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation. ACM; 2017, p. 29.
[17] Anastacio F, Sousa MC, Samavati F, Jorge JA. Modeling plant structures using concept sketches. In: Proceedings of the 4th international symposium on non-photorealistic animation and rendering. ACM; 2006, p. 105–13.
[18] Igarashi T, Matsuoka S, Tanaka H. Teddy: a sketching interface for 3D freeform design. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co.; 1999, p. 409–16.
[19] Nealen A, Igarashi T, Sorkine O, Alexa M. FiberMesh: designing freeform surfaces with 3D curves. ACM Trans Graph 2007;26(3):41.
[20] Karpenko OA, Hughes JF. SmoothSketch: 3D free-form shapes from complex sketches. ACM Trans Graph 2006;25(3):589–98.
[21] Miao Y, Hu F, Zhang X, Chen J, Pajarola R. SymmSketch: Creating symmetric 3D free-form shapes from 2D sketches. Comput Vis Media 2015;1(1):3–16.
[22] Smirnov D, Bessmeltsev M, Solomon J. Deep sketch-based modeling of man-made shapes. 2019, ArXiv abs/1906.12337.
[23] Tsang S, Balakrishnan R, Singh K, Ranjan A. A suggestive interface for image guided 3D sketching. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM; 2004, p. 591–8.
[24] Yang C, Sharon D, van de Panne M. Sketch-based modeling of parameterized objects.. In: SIGGRAPH sketches. Citeseer; 2005, p. 89.
[25] Shtof A, Agathos A, Gingold Y, Shamir A, Cohen-Or D. Geosemantic snapping for sketch-based modeling. Comput Graph Forum 2013;32(2pt2):245–53.
[26] Li C, Pan H, Liu Y, Tong X, Sheffer A, Wang W. BendSketch: modeling freeform surfaces through 2D sketching. ACM Trans Graph 2017;36(4):125.
[27] Zeleznik R, Herndon K, Hughes J. SKETCH: an interface for sketching 3d scenes. In: Proceedings of the SIGGRAPH. 1996.
[28] Shin H, Igarashi T. Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. In: Proceedings of graphics interface 2007. ACM; 2007, p. 63–70.
[29] Lee J, Funkhouser TA. Sketch-based search and composition of 3D models. In: SBM. 2008, p. 97–104.
[30] Xu K, Chen K, Fu H, Sun W-L, Hu S-M. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. ACM Trans Graph 2013;32(4):123.
[31] Chen T, Cheng M-M, Tan P, Shamir A, Hu S-M. Sketch2photo: Internet image montage. ACM Trans Graph 2009;28(5):124.
[32] Eitz M, Richter R, Hildebrand K, Boubekeur T, Alexa M. Photosketcher: interactive sketch-based image synthesis. IEEE Comput Graph Appl 2011;31(6):56–66.
[33] Sahillioğlu Y, Sezgin M. Sketch-based articulated 3D shape retrieval. IEEE Comput Graph Appl 2017;37(6):88–101.
[34] Eitz M, Richter R, Boubekeur T, Hildebrand K, Alexa M. Sketch-based shape retrieval. ACM Trans Graph 2012;31(4):31.
[35] Wang F, Kang L, Li Y. Sketch-based 3d shape retrieval using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, p. 1875–83.
[36] Li B, Lu Y, Li C, Godil A, Schreck T, Aono M, Burtscher M, Fu H, Furuya T, Johan H, et al. Shrec'14 track: Extended large scale sketch-based 3D shape retrieval. In: Eurographics workshop on 3D object retrieval, Vol. 2014. 2014.
[37] Allen B, Curless B, Popović Z. The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans Graph 2003;22(3):587–94.
[38] Seo H, Magnenat-Thalmann N. An example-based approach to human body manipulation. Graph Models 2004;66(1):1–23.
[39] Hasler N, Stoll C, Sunkel M, Rosenhahn B, Seidel H-P. A statistical model of human pose and body shape. Comput Graph Forum 2009;28(2):337–46.
[40] Pishchulin L, Wuhrer S, Helten T, Theobalt C, Schiele B. Building statistical shape spaces for 3d human modeling. Pattern Recognit 2017;67:276–86.
[41] Loper M, Mahmood N, Romero J, Pons-Moll G, Black MJ. SMPL: A skinned multi-person linear model. ACM Trans Graph 2015;34(6):248.
[42] Kanazawa A, Black MJ, Jacobs DW, Malik J. End-to-end recovery of human shape and pose. In: The IEEE conference on computer vision and pattern recognition (CVPR). 2018.

[43] Omran M, Lassner C, Pons-Moll G, Gehler PV, Schiele B. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In: 2018 international conference on 3D vision, 3DV 2018, Verona, Italy, September 5-8, 2018. 2018, p. 484–94.
[44] Sahillioğlu Y. Recent advances in shape correspondence. Vis Comput 2020;36(8):1705–21.
[45] Bronstein A, Bronstein M, Kimmel R. Numerical geometry of non-rigid shapes. 1st ed.. Springer Publishing Company, Incorporated; 2008.
[46] Girdhar R, Fouhey DF, Rodriguez M, Gupta A. Learning a predictable and generative vector representation for objects. 2016, CoRR, abs/1603.08637, URL: http://arxiv.org/abs/1603.08637, arXiv:1603.08637.
[47] Bogo F, Kanazawa A, Lassner C, Gehler P, Romero J, Black MJ. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In: Computer vision – ECCV 2016. Lecture Notes in Computer Science, Springer International Publishing; 2016.
[48] Besl P, McKay N. Method for registration of 3-D shapes. In: Sensor fusion IV: control paradigms and data structures, Vol. 1611. 1992.
[49] Arun KS, Huang TS, Blostein SD. Least-squares fitting of two 3-D point sets. IEEE Trans Pattern Anal Mach Intell 1987;PAMI-9(5):698–700.
[50] Bessmeltsev M, Vining N, Sheffer A. Gesture3D: Posing 3D characters via gesture drawings. ACM Trans Graph (Proc ACM SIGGRAPH Asia 2016) 2016;35(6). doi:??.
[51] Kim VG, Lipman Y, Funkhouser T. Blended intrinsic maps. ACM Trans Graph 2011;30(4):79.
[52] Sahillioğlu Y, Yemez Y. Coarse-to-fine isometric shape correspondence by tracking symmetric flips. Comput Graph Forum 2013;32(1):177–89.
[53] Sahillioğlu Y. A genetic isometric shape correspondence algorithm with adaptive sampling. ACM Trans Graph 2018;37(5):175.