Cerberus: A Multi-headed Derenderer

Boyang Deng,* Simon Kornblith, and Geoffrey Hinton
Google Research, Brain Team
{bydeng, skornblith, geoffhinton}@google.com

Abstract

To generalize to novel visual scenes with new viewpoints and new object poses, a visual system needs representations of the shapes of the parts of an object that are invariant to changes in viewpoint or pose.

3D graphics representations disentangle visual factors such as viewpoints and lighting from object structure in a natural way. It is possible to learn to invert the process that converts 3D graphics representations into 2D images, provided the 3D graphics representations are available as labels. When only the unlabeled images are available, however, learning to derender is much harder.

We consider a simple model which is just a set of free floating parts. Each part has its own relation to the camera and its own triangular mesh which can be deformed to model the shape of the part. At test time, a neural network looks at a single image and extracts the shapes of the parts and their relations to the camera. Each part can be viewed as one head of a multi-headed derenderer. During training, the extracted parts are used as input to a differentiable 3D renderer and the reconstruction error is backpropagated to train the neural net. We make the learning task easier by encouraging the deformations of the part meshes to be invariant to changes in viewpoint and invariant to the changes in the relative positions of the parts that occur when the pose of an articulated body changes.

Cerberus, our multi-headed derenderer, outperforms previous methods for extracting 3D parts from single images without part annotations, and it does quite well at extracting natural parts of human figures.

1. Introduction

Over the years, many efforts have been made to learn visual representations by reconstructing inputs in pixel space [40, 13, 21]. Empirically, learned models have smooth latent manifolds and are able to generate outputs that resemble natural images. However, each pixel is af-

(a) Input

(b) 3D Output

(c) Parts

(d) New Lighting

(e) New Viewpoint

(f) New Pose

Figure 1: Given an input image (a), Cerberus can output a 3D model of the object (b). This 3D model has multiple parts (shown in different colors in c). With this 3D model, we can render images with new lighting (d) or from a new viewpoint (e). We can also manipulate the parts and generate a new pose (f).

fected by multiple factors, *e.g.* lighting, viewpoint, surface reflectance, and surface shape. Modeling pixel responses directly is challenging when the generative model

---

*Work done as part of the Google AI Residency Program

that combines these factors must be learned from data. By contrast, in the computer graphics community, techniques that use disentangled representations are well-established. The process of generating pixels from these representations, *i.e.* rendering, has also been exhaustively explored. Given a representation similar to the one used in graphics, it is straightforward to generalize to new viewpoints, new poses of the objects, or new lighting conditions (as shown in Figure 1). Moreover, with the advent of differentiable renderers [26, 20, 23, 10], we can avoid the expensive acquisition of 3D labels and learn 3D graphics representations simply by reconstructing 2D images and backpropagating the reconstruction error.

In the graphics world, complex objects are modeled by dividing them into simple parts. Decomposition into parts is especially important for applications such as gesture recognition and augmented reality which must deal with articulated bodies that can adopt a wide range of poses. Although there are clear benefits to part-based models, learning natural parts without the benefit of part annotations is difficult. Previous work has demonstrated that it is possible to learn sparse part information such as keypoints [33, 16] without requiring any supervision by making use of the way images transform. Here, we seek to discover dense part information without requiring part annotations.

To this end, we present Cerberus, a neural network that extracts a part-based 3D graphics representation from a single image, along with a training strategy that avoids the need for part annotations by using natural consistencies, *i.e.* the invariance of part shapes under changes in viewpoint or pose.[1] This training strategy ensures that Cerberus can learn to reconstruct geometrically correct 3D graphics models consisting of semantic parts without part supervision. The arrangements of the 3D parts extracted by Cerberus change with pose, and we can manipulate the 3D model to form a novel pose (Figure 1f).

We examine Cerberus on two datasets of articulated bodies. On the human dataset, which has substantial variability in pose, Cerberus not only outperforms previous work by a large margin, but also learns semantic parts such as head and legs without part annotations. These parts are consistent across poses: Cerberus produces better results than baselines even when it is restricted to applying parts extracted from an image of an individual to all other images.

In this work, we introduce the problem of unsupervised 3D perception of articulated bodies with only 2D supervision. Our key contributions are as follows:

- We propose a new architecture, Cerberus, for single image 3D perception. This architecture is more suit-

able for modeling articulated bodies than previous architectures.

- We tackle the problem of learning semantic parts without part supervision by using natural but powerful consistency constraints.

- Our architecture, trained with the proposed constraints, outperforms baselines, even when we restrict it to extract one set of parts and apply to use the same parts for all configurations of the same subject.

2. Related Work

The idea of computer vision as inverse graphics has a long history [31, 1, 8, 27]. Recent approaches claiming to perform inverse graphics typically consist of an encoder and decoder, with a latent space that has some meaning relative to underlying generative factors. Transforming autoencoders [12] model images by factorizing them as a set of *capsules* with corresponding 3D pose vectors, such that applying a 3D rotation to the 3D pose vector produces a rotated output. Other work has clamped latent variables to align to generative factors [22] or imposed information constraints on the latent space [11]. We instead prespecify the form of the latent representation by using a fixed differentiable renderer as the decoder. This strategy is common in recent work that learns 3D representations [26, 20, 23, 10, 18]. In 2D, Tieleman [36] also used a fixed decoder, reconstructing images based on affinely transformed learned templates.

We recover a 3D graphics representation from a single image, using only 2D supervision during training. Many previous approaches to inferring 3D representations have employed 3D supervision [4, 44, 34, 6, 43, 30] or fit low-dimensional parameterized models [3, 47, 7]. Nonetheless, there is a significant body of previous work that has used only 2D supervision. Non-neural network-based approaches have reconstructed 3D models from segmentation masks by combining SfM viewpoint estimation with voxel-based visual hull [42] and deformable point cloud approaches [19]. Neural network-based approaches have inferred 3D models using perspective projection [45] or ray-tracing [39, 37] of volumetric representations, differentiable point clouds [14], prediction of multiple 2.5D surfaces [32], REINFORCE gradients through off-the-shelf renderers [29], or fully differentiable mesh renderers [20, 10, 18].

Part-based 3D perception and modeling has a rich past, although it has recently fallen somewhat out of favor. Early computer vision projects attempted to develop programs capable of recognizing compound objects as combinations of parts [31, 9]. Biederman [2] influentially suggested that human object perception operates by decomposing objects into 36 primitive generalized-cones (geons). More recently,

---

[1] In work dealing only with rigid bodies, the word *pose* is often used to refer to the position and orientation of the object relative to the camera. In this work, we use pose to refer to the relative positions and orientations of the parts of an articulated body, as in human pose estimation.

Figure 2: Cerberus architecture. Here we visualize 3 out of N parts used in the pipeline. Up-sampling is performed by de-convolution. The object latent is obtained by global average pooling of the lowest-resolution feature maps. Predicted quaternions are used to construct rotation matrices.

van den Hengel *et al.* [41] proposed a method to estimate the constituent parts and arrangement of Lego models based on multiple silhouettes. AIR [5] infers arrangements of pre-specified meshes using a 3D renderer using finite-difference gradients. Other work has attempted to model 3D volumes with fixed primitives, obtaining plausible object parsings without explicit part-level supervision [38, 46, 35]. In contrast to these approaches, we learn rich part shapes in addition to positions, which allows us to extract the complex surface shapes of articulated bodies.

3. Cerberus Architecture

3.1. 3D Parameterization

Polygonal meshes, which are widely used in computer graphics, are an effective way to define shapes. A polygonal mesh is a collection of vertices, edges and faces that together describe the surface of an object. Each vertex can also be associated with auxiliary attributes like texture and albedo. Compared with voxel representations used by some previous works [45, 44], polygonal meshes are a more compact 3D representation and are easier to render with complex shading. In this work, we use triangular mesh (referred as "mesh" for the rest of this paper) as our 3D representation. One challenge of using meshes for learning 3D shapes is predicting the correct connectivities between vertices. Without constraints, neural networks are prone to erroneous connectivity predictions. To overcome this issue, we construct meshes by deforming a spherical mesh [20].

The edges, faces, and initial positions of all the vertices of this sphere are predefined. To model the shape of a part, the neural network predicts only the displacement of each vertex.

Since Cerberus models articulated objects, we seek to develop a representation that can be manipulated to allow the modeled object to take on different poses. Using a one-piece mesh for an articulated object makes this challenging because of the difficulty of finding vertex-wise correspondence between poses. In contrast, a part-based model can easily be made to pose in various ways.

Here we use an independent mesh for each part. We parameterize a part's local pose by its rotation and translation relative to the camera. The neural network predicts parameters of these transformations based on the pose of the object in the input image. After applying transformations to each part and putting them together in the same 3D space, we obtain a render-ready 3D model of the whole object in a specific pose.

3.2. 3D Reconstruction Pipeline

Our pipeline is illustrated in Figure 2. Given an input image, our pipeline outputs the 3D parameters defined in Section 3.1 for all the parts. The number of parts, N, is a predefined hyper-parameter. As shown in Figure 2, we use a base network similar to the hourglass block [28] to extract deformation, rotation, and translation parameters from a single image. We describe the process to get each of these parameters below.

Deformation: We predict vertex deformations of all the parts simultaneously. The input image goes through a down-sampling neural network to produce lower-resolution feature maps. We use global average pooling to transform these feature maps into a feature vector, referred to as the *object latent* in Figure 2. We linearly transform this object latent to get a shape latent, and then again linearly transform this shape latent to yield the vertex deformations. The shape latent is used to disentangle deformation (shape) and rotation (pose) as well as to implement pose consistency (described in Section 4.1).

Rotation: We use quaternions for rotations. A linear transformation is performed on the object latent to produce quaternions. We construct rotation matrices based on quaternions and multiply them with the corresponding deformed parts.

Translation: Instead of predicting translation parameters directly, we retrieve 3D translations from 2D coordinates and depth using an approach similar to KeypointNet [33]. After the down-sampling network, we use an up-sampling network with skip connections to enlarge feature maps to the same resolution as the input. For each part, we linearly transform the feature maps and apply a spatial softmax, yielding a "probability map" $\{p_{x,y}^k\}$. We compute the 2D coordinates for the part by taking the expectation over this map. We also calculate a depth map $\{d_{x,y}^k\}$ with elements represent the depth of pixel (x, y) for the k-th part. The resulting translation T_k for the k-th part is:

$$T_k = \pi^{-1}\left(\sum_{(x,y)\in G} [x \cdot p_{x,y}^k, y \cdot p_{x,y}^k, d_{x,y}^k \cdot p_{x,y}^k]\right) \quad (1)$$

During testing, the produced 3D model is our output. During training, rather than employing 3D supervision, we use a differentiable renderer to transform 3D representations into images. We render our 3D representation, compare the rendered result, R, with the input image, I, and then backpropagate through the renderer. The objective we use here is mean squared error pixel reconstruction loss:

$$\mathcal{L}_r = \frac{1}{|G|} \sum_{(x,y)\in G} (I_{x,y} - R_{x,y})^2 \quad (2)$$

4. Consistency Constraints

4.1. Pose Consistency

Although we learn part-based models to reconstruct 3D objects, we do not use any part supervision or keypoint annotations during training. Instead, we reflect on the way humans split an articulated body into multiple parts. In Figure 3a, we show 2 different ways to split a portion of the

(a) Pose Consistency

(b) Viewpoint Consistency

Figure 3: Consistency constraints enforced in our architecture. a: Two ways of segmenting a portion of the human body into parts. The segmentation in the bottom row is preferable because the shapes of parts remain unchanged when the pose of the person changes. b: Images of a single individual from 2 viewpoints. Although the rendered images are clearly different, the retrieved 3D model should be the same.

human body into parts. The strategy shown in the bottom row produces semantic parts (abdomen and thigh) whereas the strategy shown in the top row does not. A critical characteristic of the strategy shown in the bottom row is that, when the person's pose changes, the shape of each part remains almost the same. This gives us an essential hint on how to split semantic parts without supervision: For a pair of images of the same person in 2 poses, the 2 predicted sets of parts should have the same shape. In practice, we use a pair of images from the same viewpoint containing the same object in 2 different poses for training. We argue that collecting this kind of supervision is trivial, since we can simply use 2 frames from a video of a moving object filmed by a static camera.

4.2. Viewpoint Consistency

Learning 3D shape from a single image is an ill-posed problem. There exist an infinite number of possible 3D models that yield the same 2D projection. For the sake of learning correct shapes, we need our predicted 3D models

to be consistent across viewpoints during training. Specifically, we use a pair of images from 2 different viewpoints for the same object (in the same pose) during training. The goal is to predict the same 3D model from these 2 viewpoints (as shown in Figure 3b). Previous works have investigated this consistency and used cycle construction [20, 45] or a loss term [33] to implement the constraint.

4.3. Constraint Implementation

Combining the above constraints, each training example consists of a quadruplet of images, comprising the same object in 2 poses seen from 2 viewpoints. We index the two poses by a and b and the two viewpoints by 0 and 1. Given a quadruplet during training, Cerberus will output 4 shape latents (referred to as S^{a0}, S^{a1}, S^{b0}, and S^{b1})

To enforce the pose consistency constraint, we randomly select elements from the 4 shape latents corresponding to the 4 images in the training quadruplet to form the shape latent $\tilde{S}$ used for rendering. We can formulate $\tilde{S}$ as:

$$\tilde{S} = \sum_{x \in Q} \mathbb{1}_Z(x) \cdot S^x \tag{3}$$

where $Q = (a0, a1, b0, b1)$ is the quadruplet and Z is a vector whose elements are sampled from a uniform categorical distribution with Q as categories.

The viewpoint constraint implementation consists of 2 components. The first component follows the common design of rendering the same 3D model from 2 viewpoints and comparing them with ground truth images from these viewpoints. Because our model predicts translation parameters, we make a slight change to this design. We render the same rotated mesh with different translations for different viewpoints. Thus, the reconstruction loss for the input in pose a from viewpoint 0 is:

$$\mathcal{L}_r^{a0} = \frac{1}{2}(\mathcal{L}_r^{a0 \to a0} + \mathcal{L}_r^{a0 \to a1}) \tag{4}$$

where $\mathcal{L}_r^{a0 \to a1}$ stands for the reconstruction loss of rendering the rotated mesh produced from image $a0$ from viewpoint 1 with translation predicted from input $a1$. We use different translations for different viewpoints because the model may predict incorrect translations at the early stages of training. Using the translation predicted from a single viewpoint only guarantees that the object is visible from this viewpoint. From a different viewpoint, the renderer may not see the object. In this case, we would be unable to get a gradient from the renderer, so training would collapse.

The second component encourages the translations to be consistent across viewpoint, using a mean squared error loss to penalize inconsistent translations. For each pair of viewpoints, the translation loss term is:

$$\mathcal{L}_t = \frac{1}{N} \sum_i^N (T_i^0 - T_i^1)^2 \tag{5}$$

where T_i^0 and T_i^1 are predicted translations of the i-th part from 2 viewpoints. Since we have 2 poses, the full translation loss term for each quadruplet is:

$$\mathcal{L}_t' = \frac{1}{2}(\mathcal{L}_t^a + \mathcal{L}_t^b) \tag{6}$$

where $\mathcal{L}_t^a$ represents $\mathcal{L}_t$ for pose a and $\mathcal{L}_t^b$ represents $\mathcal{L}_t$ for pose b. The total reconstruction loss is:

$$\mathcal{L}_r' = \frac{1}{4}(\mathcal{L}_r^{a0} + \mathcal{L}_r^{a1} + \mathcal{L}_r^{b0} + \mathcal{L}_r^{b1}) \tag{7}$$

5. Experiments

Human Dataset. This dataset contains 3D human models in diverse body poses. We use SMPL [25], a parameterized deformable human model, to generate all the example meshes. In particular, the parameters of SMPL are fit to frames of human action video clips from Human3.6M [15] using Mosh [24]. In our experiments, we use the fitted results from [17]. We split the data into train and test splits. The train split comprises 19,500 pairs of body poses with 5 subjects (S1, S5, S6, S7, S8 in Human3.6M). Each pose is rendered from 2 different viewpoints. We fix the elevation angle and distance to the origin of all viewpoints but vary the azimuth angles. On scene set-ups, we use a directional light following the direction of the camera and an ambient light, all in white. In the test split, we use 2 unseen subjects (S9, S11 in Human3.6M). We render 810 different poses in total, each from 4 viewpoints.

Animal Dataset. This dataset consists of 3D models of quadrupeds. Compared with the Human dataset, it has more variance in shape but less variance in pose. Each example is generated by a deformable model, SMAL [48], for quadrupeds. We use 41 different animals released by [48]. Each of them has 47 poses in the train split and 3 poses in the test split. We render each training pose from 4 viewpoints and each test pose from 8 viewpoints. In total, the dataset contains 38,540 training quadruplets (as used in Section 4.3) and 984 test examples. The scene set-ups are the same as the human dataset.

Implementation Details. Our down-sampling network is ResNet-10-v1. The number of channels are 64, 128, 256, 512 for different feature map resolutions. The up-sampling network has 3 transposed convolution layers with skip connections from down-sampling layers of the same resolution. We use a spherical mesh with 162 vertices and 320 triangles as the starting point of the deformation for each part. We set the number of parts to 9 for all experiments. During training, we use 2 additional regularization loss terms. One is a background loss $\mathcal{L}_b = \frac{1}{N} \sum_k \sum_{x,y} p_{x,y}^k b_{x,y}$, where $b_{x,y}$ indicates whether pixel (x, y) is in the background. This loss

Input	NMR	NMRs	NMRr	Ours	Parts	Turn

Table 1: 3D Human Model Reconstructions. On the left are input images. In the middle are baseline results. NMR is the architecture and implementation from [20] without a smoothness loss. NMRs adds smoothness loss to NMR. NMRr is our re-implementation of NMR using the same renderer and images with shading as used by Cerberus. We also visualize our parts in different colors in the middle column of the right group. The Turn column shows Cerberus's outputs rendered from a different viewpoint.

helps avoid situations where a part is out of the renderer's scope and no gradient will pass through. The other is a smoothness loss $\mathcal{L}_s = \sum_{\theta_i \in \epsilon}(\cos\theta_i + 1)^2$ following [20]. ϵ is the set of dihedral angles of the output mesh. This loss helps encourage neighboring vertices to have similar displacement. We use Adam with learning rate 0.0005 and batch size 16 to optimize the weighted sum of all the loss terms, with weights $(\lambda_r, \lambda_t, \lambda_b, \lambda_s)$ set to $(1, 1, 1, 0.0001)$ for all experiments. We train our models for 100,000 steps. Unlike [20], we use a differentiable renderer based on gradients of barycentric coordinates [7].

5.1. 3D Human Reconstruction

We first test our model on single image 3D human reconstruction. This task is a standard benchmark for 3D visual perception methods. The goal of this task is to extract 3D models of the object in the given image. We

measure the quality of the predicted 3D models by voxel IoU (intersection-over-union), following [45]. Since our 3D models are meshes, we transform the predicted meshes and the ground-truth meshes into $32 \times 32 \times 32$ voxel grids. Because Cerberus predicts multiple parts, we take the union of all the part voxels as the output voxel.

We provide baseline results with the Neural Mesh 3D Renderer (NMR) [20]. Because we found that the smoothness loss proposed in [20] performs poorly for curved surfaces, we train NMR models both with and without this smoothness loss. We also recognize that NMR uses only silhouette supervision, whereas the differentiable renderer we use for Cerberus provides gradients for shaded surfaces. To exclude confounds related to the choice of renderer, we also re-implement NMR using our renderer on images rendered with shading. We visualize some example 3D outputs of all the baselines and Cerberus on the test split in Table 1.

Model	Human	Hard Human	Animal
NMR	0.2596	-	0.3000
NMRs	0.2233	-	0.2574
NMRr	0.3084	-	0.3201
Cerberus	0.4970	**0.4728**	**0.4255**
Free Cerberus	**0.5099**	0.4365	0.4196

Table 2: Single image 3D reconstruction test results on 2 datasets. We use voxel IoU (intersection-over-union) as our metric. Higher is better. NMR refers the model proposed by [20]. NMRs is NMR with smoothness loss. NMRr is our re-implementation using the same renderer as Cerberus. Free Cerberus is Cerberus trained without pose consistency. Hard Human reflects accuracy when reconstructing all poses in the test set using the same set of parts. Hard Human results are shown only for Cerberus, since NMR does not model individual parts.

As shown in Table 1, Cerberus predicts smooth 3D meshes that are visually more similar to the human in the input. Compared with NMR, which mainly reconstructs the outline shape of the torso, Cerberus can produces more details of the body, *e.g.* the legs. We hypothesize that this improvement is related to the flexibility of part-based modeling. More importantly, we find that, although our model is trained without any part annotations, it can predict semantic parts of the human body. For instance, the beige part shown in Table 1 is clearly recognizable as the head. Similarly, we find parts representing legs in Cerberus's outputs. Cerberus can even separate lower legs and thighs into different parts. We believe that this segmentation arises from our pose consistency constraint. We have both examples with bent knees and examples with straight legs in our training split. In order to model the body accurately, Cerberus must learn to separate these two parts. We also notice that Cerberus produces plausible results for invisible parts (as shown in the third row of Table 1), suggesting that the neural network implicitly incorporates the prior of human's body shape so that it outputs two legs even when only one leg is visible.

Quantitatively, Cerberus predicts 3D meshes with greater similarity to the target meshes than previous approaches (Table 2). Compared with the original NMR, Cerberus achieves double the test IoU. Our re-implementation of NMR has a higher accuracy than the original NMR, suggesting that reconstructing shaded images helps learn better shapes. Nonetheless, Cerberus outperforms this re-implemented NMR by a substantial margin.

5.2. Transferable Parts Among Poses

We also devise a benchmark to quantitatively evaluate the accuracy with which Cerberus segments parts. Instead

(a) Canonical Inputs and Reconstruction

(b) Reconstruction by transformations

Figure 4: Hard human reconstruction test. We extract a single set of deformed shape meshes from the canonical inputs (a) and apply new transformations (rotation and translation) predicted from other images to these meshes to reconstruct new poses (b). **Left**: Input images. **Middle**: Reconstructed 3D outputs. **Right**: Parts rendered in different colors.

of computing IoU when reconstructing each test case independently, we perform identity-conditional reconstruction. We first extract the deformed part meshes from 2 images of the 2 subjects in the test set. These images contain the canonical pose of the the subjects (shown in Figure 4a). Then, we reconstruct other examples in the test split by applying predicted rotation and translation to the deformed parts from the canonical pose of the same subject.

The voxel IoU of this more challenging evaluation ("Hard Human") is shown in Table 2. Even in this extreme test set-up, Cerberus's accuracy remains high, and our predictions remain better than all baseline methods. This result quantitatively confirms that the pose consistency constraint enforces Cerberus to learn meaningful parts that are transferable among diverse poses. We illustrate the output meshes produced by this evaluation set-up in Figure 4.

To validate the effectiveness of pose consistency constraint for learning transferable and semantic parts, we perform an ablation study by training Cerberus without pose

Figure 5: Comparison between Free Cerberus (w/o pose consistency) and Cerberus (w/ pose consistency). **Left:** Input. **Middle:** Output of Free Cerberus. **Right:** Output of Cerberus. Free Cerberus uses a single green part for 2 legs while Cerberus correctly models 2 legs.

consistency constraint. The test IoU of this model ("Free Cerberus") is shown in Table 2. Free Cerberus achieves higher accuracy in the standard evaluation than Cerberus with pose consistency. This is unsurprising, given that Free Cerberus has additional freedom in modeling objects. However, in the Hard Human evaluation, Free Cerberus's accuracy is significantly lower than Cerberus, with a large drop relative to the accuracy on standard evaluation. Thus, the pose consistency constraint is important to properly segment parts. We visualize the outputs of Cerberus and Free Cerberus in Figure 5. We see that Free Cerberus uses a single green part to represent two legs, whereas Cerberus models the legs with separate parts.

5.3. 3D Animal Reconstruction

In addition to reconstructing 3D human models from a single image, we also examine Cerberus's ability to reconstruct objects with greater variability in shape. To this end, we evaluate our method on the animal dataset. This dataset comprises of 41 different animals, ranging from deer with small heads and long thin limbs to hippopotamuses with large heads and short thick limbs.

We show test IoU on the animal dataset in Table 2. Baseline methods perform better on the animal dataset as compared to the human dataset, likely because there is less variability in pose. Nonetheless, Cerberus remains superior. Thus, Cerberus consistently outperforms baselines on objects with either high variability in pose (humans) or high variability in shape (animals).

We demonstrate the predicted 3D animals and parts from Cerberus in Figure 6. Based on the visualized test outputs, we see that Cerberus predicts high quality meshes for different animals in various poses from diverse viewpoints. Additionally, part segmentation is reasonable and consistent across different animals, but parts vary appropriately in shape. For example, the light green part always corresponds to the head, but this part has a pointed snout for the deer in Figure 6b and a round snout for the hippo in Figure 6d.

(a) Cougar

(b) Deer

(c) Tiger

(d) Hippo

Figure 6: 3D Animal Reconstruction. We visualize 4 examples from the test split and the 3D outputs of Cerberus. **Left:** Input images. **Middle:** 3D outputs. **Right:** Parts rendered in different colors.

6. Conclusion

We have proposed a new architecture and training paradigm for single-image 3D reconstruction with only 2D supervision. Our approach not only reconstructs 3D models more accurately than approaches that use a single monolithic mesh, but also infers semantic parts without part-level supervision.

Although we focus on the problem of 3D reconstruction, in the spirit of inverse graphics, our approach can potentially be adapted to tasks such as classification or pose estimation. Current state-of-the-art approaches to these tasks rely on extensive amounts of labeled training data. A 3D representation that explicitly disentangles shape, pose, and viewpoint has the potential to significantly improve sample efficiency, because the basic invariance properties of objects are reflected directly in the representation and need not be learned from labels.

References

[1] B. G. Baumgart. Geometric modeling for computer vision. Technical report, Stanford University, 1974. 2

[2] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 2

[3] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016. 2

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2

[5] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016. 3

[6] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[7] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 6

[8] U. Grenander. *Pattern Analysis. Lectures in Pattern theory*. Springer, 1978. 2

[9] A. Guzmán. Decomposition of a visual scene into three-dimensional bodies. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 291–304. ACM, 1968. 2

[10] P. Henderson and V. Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. *arXiv preprint arXiv:1807.09259*, 2018. 2

[11] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. 2

[12] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011. 2

[13] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 1

[14] E. Insafutdinov and A. Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2807–2817, 2018. 2

[15] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2014. 5

[16] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in Neural Information Processing Systems*, pages 4020–4031, 2018. 2

[17] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)*, 2018. 5

[18] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 2

[19] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1966–1974, 2015. 2

[20] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 2, 3, 5, 6, 7

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2013. 1

[22] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015. 2

[23] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. In *SIGGRAPH Asia 2018 Technical Papers*, page 222. ACM, 2018. 2

[24] M. Loper, N. Mahmood, and M. J. Black. Mosh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (TOG)*, 33(6):220, 2014. 5

[25] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015. 5

[26] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014. 2

[27] D. Mumford. Pattern theory: a unifying perspective. In *First European congress of mathematics*, pages 187–224. Springer, 1994. 2

[28] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016. 3

[29] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems*, pages 4996–5004, 2016. 2

[30] S. R. Richter and S. Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1936–1944, 2018. 2

[31] L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963. 2

[32] D. Shin, C. C. Fowlkes, and D. Hoiem. Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3069, 2018. 2

[33] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems*, pages 2063–2074, 2018. 2, 4, 5

[34] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017. 2

[35] Y. Tian, A. Luo, X. Sun, K. Ellis, W. T. Freeman, J. B. Tenenbaum, and J. Wu. Learning to infer and execute 3d shape programs. In *International Conference on Learning Representations*, 2019. 3

[36] T. Tieleman. Autoencoders with domain-specific decoders. In *Optimizing neural networks that generate images (Ph.D. thesis)*. University of Toronto (Canada), 2014. 2

[37] S. Tulsiani, A. A. Efros, and J. Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2897–2905, 2018. 2

[38] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 3

[39] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 2

[40] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE, 1991. 1

[41] A. van den Hengel, C. Russell, A. Dick, J. Bastian, D. Pooley, L. Fleming, and L. Agapito. Part-based modelling of compound scenes from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 878–886, 2015. 2

[42] S. Vicente, J. Carreira, L. Agapito, and J. Batista. Reconstructing pascal voc. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 41–48, 2014. 2

[43] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 2

[44] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016. 2, 3

[45] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016. 2, 3, 5, 6

[46] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3D-PRNN: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017. 3

[47] S. Zuffi, A. Kanazawa, and M. J. Black. Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3955–3963, 2018. 2

[48] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6365–6373, 2017. 5