Transformable Bottleneck Networks - Supplemental Material

Kyle Olszewski[13*], Sergey Tulyakov[2], Oliver Woodford[2], Hao Li[134], and Linjie Luo[5*]
[1]University of Southern California, [2]Snap Inc., [3]USC ICT, [4]Pinscreen Inc., [5]ByteDance Inc.

1. Architecture

The overall architecture of our novel view synthesis network is depicted in Table 1. In this table and the corresponding diagrams, *conv* indicates a standard convolutional layer of the specified filter size and stride[1]. In our model, these layers are followed by a batch normalization operation. *upconv* indicates a nearest-neighbor upsampling operation that increases the output width and height by a factor of 2, followed by a convolution with filter size 3×3 and stride 1, which produces an output of the same size[2], and a batch normalization operation. The *reshape* operation is used before and after the 3D block to produce outputs that match the specified dimensions. *output* is a layer in which a 3×3 convolution with stride 1 is applied, followed by a sigmoid operation that produces output in the range of 0 to 1 in each channel. The final output is an RGB image with an additional channel for the segmentation mask.

The architecture of the *unet_block* segments is depicted in Fig. 1. This component uses a standard U-Net architecture [4] with skip connections connecting the encoder and decoder in each block. The encoder is made up of 3 residual blocks [1], as depicted in Fig. 2. These blocks each reduce the dimensions of the input by a factor of 2. The output of these layers is concatenated with the output of the corresponding *upconv* layers in the decoder, which increase the scale of the input by a factor of 2. As depicted, these concatenated feature maps are then passed through *conv* blocks. In this and subsequent diagrams, the number at the bottom of each cell indicates the number of feature maps output by this operation.

The architecture of the *3d_block* segment is depicted in Fig. 3. This block consists of 2 convolution layers (3×3, stride 1) applied before and after the spatial transformation.

Layer Name	Output Size	Filter Size, Stride	Notes
input_image	$160 \times 160 \times 3$		
conv	$80 \times 80 \times 32$	4×4, 2	
conv	$40 \times 40 \times 64$	7×7, 2	
unet_block	$40 \times 40 \times 800$		See Fig. 1
reshape	$40 \times 40 \times 40 \times 20$		Reshape 2D to 3D
3d_block	$40 \times 40 \times 40 \times 20$		See Fig. 3
reshape	$40 \times 40 \times 800$		Reshape 3D to 2D
unet_block	$40 \times 40 \times 20$		See Fig. 1
upconv	$80 \times 80 \times 32$		
conv	$80 \times 80 \times 32$	3×3, 1	
upconv	$160 \times 160 \times 32$		
output	$160 \times 160 \times 4$	3×3, 1	

Table 1: The architecture of our Transformable Bottleneck Network. Please consult the referenced figures for details on the individual segments of the network.

1.1. 3D Reconstruction

For the results provided for the 3D reconstruction task, we use the overall network structure described in Table 1, except that we do not apply the first *conv* and final *upconv* layers, which halve and double the overall output dimensions, respectively. This results in a $32 \times 32 \times 32$ feature volume (with 20 features per cell) when the network is applied to the 64×64 RGB images used as input to the network. This corresponds to the dimensions of the occupancy volume used in [6] and in our evaluations.

The network branch that serves as our occupancy decoder (see overview figure in the paper) has the same structure as the *3d_block* described above. However, in this case, the final 3D convolution layer produces only 1 feature per cell, and no further spatial transformation is applied in the middle of this block, as we are simply interested in obtaining the occupancy status for each cell in the feature volume. We apply a softmax operation in the depth dimension to the features produced by the occupancy decoder. In our experiments, we found that this softmax operation helped to normalize the input to a range that worked well for our reconstruction task, reducing the influence of extreme values in the occupancy volume.

To synthesize the 2D segmentation masks used for training, we reshape the occupancy volume into a 32×32 feature map with 32 features per cell, then apply a 1×1 convolution with stride 1 to these features to produce a single scalar feature per cell, followed by a sigmoid operation. This produces a 2D 32×32 segmentation mask with values between 0 and 1. This segmentation mask is then upsampled to the target resolution, 64×64. This mask is then used to compute the loss compared to the ground-truth segmentation masks from the dataset.

During training, this branch is applied to the feature vol-

*This work was performed while the author was at Snap Inc.

[1]In the text, table and following diagrams, *conv* blocks use a filter size of 3×3 and stride 1, except when otherwise noted.

[2]Padding is used as necessary to maintain the output dimensions specified at each layer.

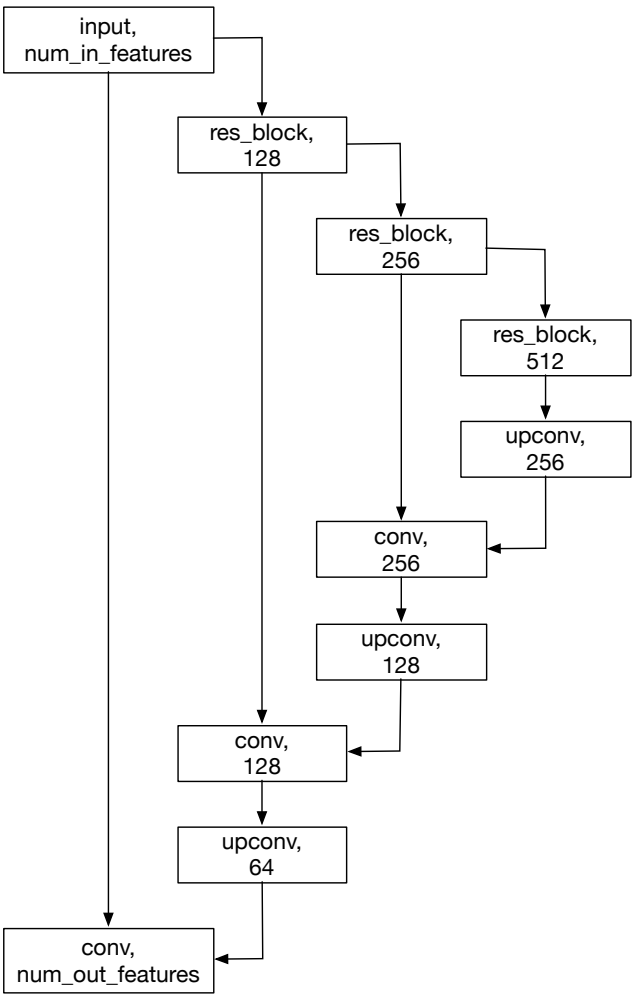

Figure 1: The architecture of the *unet_block* layers of our model. The number at the bottom of each block indicates the number of feature maps produced by the operation.

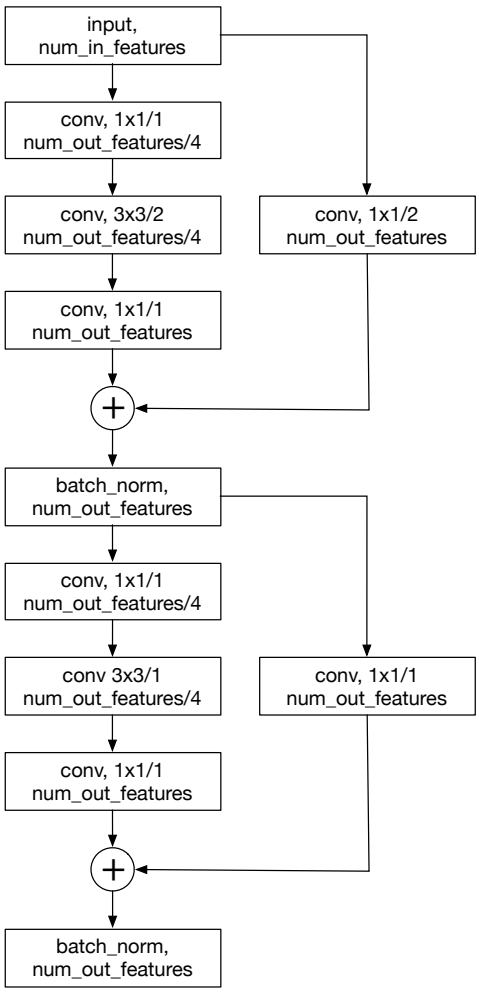

Figure 2: The architecture of the *res_block* layers used in our architecture, as seen in Fig. 1. The top row of the *conv* blocks indicates the filter size and stride, while the number at the bottom of each block indicates the number of feature maps produced by the operation.

ume immediately before the spatial transformation to obtain the occupancy volumes and segmentation masks corresponding to each source image, and after the feature volume aggregation and spatial transformation for the occupancy volume and segmentation mask corresponding to the target image.

For the 3D reconstruction evaluations, we generate target occupancy volumes aligned to the canonical view of the object used in the meshes that are voxelized to obtain the ground-truth occupancy volume for each object.

2. 3D Reconstruction Results

In Table 2 we provide details on the results of the 3D reconstruction experiments described in the paper (Sec. 4.3, Fig. 5) and the comparison with those obtained by Tulsiani *et al*. [6]. [3] We report the Intersection-over-Union (IoU, higher is better) between the reconstructed volume and the ground-truth results obtained by voxelizing the mesh rendered for the corresponding image. The top row provides the results obtained using our method and theirs for only one input image, from which we extract the corresponding occupancy volume. The subsequent rows present the results obtained using our method when using additional views and averaging the corresponding bottleneck layers (as is done when using multiple input images for novel view synthesis) before applying the occupancy decoder.

"real" indicates that additional views of the rendered ob-

[3] For a fair comparison, we report numbers obtained using the pretrained models, datasets, and evaluation framework made available online by the authors for this work, which were overall somewhat lower than those reported in their paper.

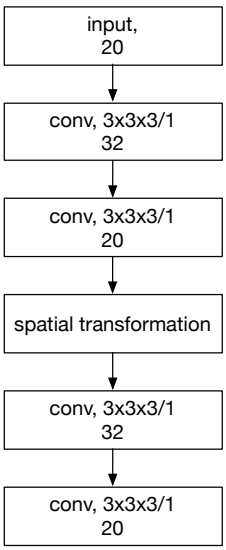

Figure 3: The 3D component of our network. The top row of the *conv* blocks indicates the filter size and stride, while the number at the bottom of each block indicates the number of feature maps produced by the operation.

ject (chosen from the 10 renderings per object in the dataset used for evaluation) were used to create the occupancy volume. These results thus show how our method improves its results when the additional information provided by these views. "synthetic" indicates that these additional views of the object under different poses were *synthesized* by our encoder-decoder framework, given the single original image as input, before being passed through the encoder again and aggregated in the bottleneck with those from the other views. As such, the "synthetic" results still rely on only a *single* "real" image as input. This allows for a fair comparison between our method and [6] in these cases.

"random poses" indicates that the azimuth and elevation for the synthesized viewpoints were selected at random from the same distributions as were used for rendering the training and evaluation sets. "regular poses" indicates that these additional images were synthesized at regular intervals around the vertical axis. This allows the synthesized images to complement one another by providing contextual information that may be missing when poses are chosen at random. Our results demonstrate that using synthesized images with regular poses outperforms not only [6] and our method when using a single image, but even the use of real images at random poses. The reconstruction quality generally improves somewhat as additional views are synthesized, but using as little as 4 additional synthesized views, we obtain results that are superior to those obtained using each alternative we evaluated. This indicates that the generative power of our encoder-decoder framework can be used to create images that improve the overall quality of the structural information stored in the bottleneck produced by the encoder, when the encoded bottlenecks for these synthesized images are aggregated with that from the original input image.

	Methods	IoU		
		Chair	Car	Aero
	TBN	.3042	.4664	.2699
	Tulsiani *et al*. [6]	.3913	.7113	.3332
+1 view	TBN, real, random poses	.3455	.5233	.3300
	TBN, synthetic, random poses	.3387	.5213	.3251
	TBN, synthetic, regular poses	.3628	.5727	.3752
+2 views	TBN, real, random poses	.3650	.5479	.3582
	TBN, synthetic, random poses	.3532	.5433	.3474
	TBN, synthetic, regular poses	.3738	.6025	.4060
+3 views	TBN, real, random poses	.3753	.5638	.3741
	TBN, synthetic, random poses	.3600	.5573	.3587
	TBN, synthetic, regular poses	.4312	.6785	.4490
+4 views	TBN, real, random poses	.3822	.5754	.3858
	TBN, synthetic, random poses	.3648	.5674	.3668
	TBN, synthetic, regular poses	.4507	.7128	**.4661**
+5 views	TBN, real, random poses	.3878	.5840	.3941
	TBN, synthetic, random poses	.3687	.5748	.3725
	TBN, synthetic, regular poses	.4455	.7020	.4498
+6 views	TBN, real, random poses	.3918	.5913	.4004
	TBN, synthetic, random poses	.3714	.5814	.3768
	TBN, synthetic, regular poses	.4486	.7075	.4522
+7 views	TBN, real, random poses	.3946	.5968	.4049
	TBN, synthetic, random poses	.3732	.5862	.3797
	TBN, synthetic, regular poses	.4546	.7070	.4530
+8 views	TBN, real, random poses	.3972	.5996	.4090
	TBN, synthetic, random poses	.3748	.5884	.3827
	TBN, synthetic, regular poses	**.4630**	**.7131**	.4594
+9 views	TBN, real, random poses	.3988	.6023	.4132
	TBN, synthetic, random poses	.3757	.5906	.3851
	TBN, synthetic, regular poses	.4561	.7088	.4565

Table 2: Quantitative results for 3D reconstruction using a single input image, and with up to 9 additional views (real or synthesized). We report the intersection-over-union (IoU, higher is better) for our method and Tulsiani *et al*. [6], which uses a single image as input.

We note that we obtain substantially better quantitative results on the chair and aero datasets, but obtain only slightly better results for the car dataset. We believe that this is due to the relatively simple and uniform structures of the objects in the car dataset, compared to the more varied shapes seen in the other datasets. The benefit obtained using our approach is more substantial for the latter datasets, in which simply producing a rough estimate of an average object's shape would result in larger errors than it would for the cars.

3. Training

The equation defining the total training loss is, as described in the paper,

$$\mathcal{L}_{\mathrm{T}}(\Theta) = \mathcal{L}_{\mathrm{R}} + \lambda_1 \mathcal{L}_{\mathrm{P}} + \lambda_2 \mathcal{L}_{\mathrm{S}} + \lambda_3 \mathcal{L}_{\mathrm{A}} + \lambda_4 \mathcal{L}_{\mathrm{M}}, \quad (1)$$

where $\mathcal{L}_{\mathrm{R}}$ is the L_1 reconstruction loss, $\mathcal{L}_{\mathrm{P}}$ is the L_2 loss in the feature space of the VGG-19 network[4], $\mathcal{L}_{\mathrm{S}}$ is the structural similarity (SSIM) index loss, $\mathcal{L}_{\mathrm{A}}$ is the adversarial loss using the discriminator architecture from [7]), and $\mathcal{L}_{\mathrm{M}}$ is the segmentation masking loss. Please see the paper for details on each of these loss terms. We empirically determined appropriate weights for the hyper-parameters controlling the contribution of the different loss components: $\lambda_1 = 5$, $\lambda_2 = 10$, $\lambda_3 = 0.05$, and $\lambda_4 = 10$.

We train the network using the Adam optimizer [2] with a learning rate set to 0.0002, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Convergence on the test set typically takes approximately 8 days for each dataset we used for our evaluations.

4. Datasets

4.1. Novel View Synthesis

4.1.1. ShapeNet Chairs and Cars

We evaluate our framework's novel view synthesis (NVS) capabilities using the dataset provided for the benchmark in [5].[5] While the images were rendered at 256×256, our NVS network architecture accepts and produces images at a resolution of 160×160 for the $40 \times 40 \times 40$ volumetric bottleneck that we use for these evaluations[6]. We thus apply bilinear resampling to downsample the input and upsample the output to the resolution used during training. As this operation is differentiable, losses during training are measured with respect to the target image at its original resolution. We also report these losses used for the benchmark at the original target image resolution to make for a fair comparison to the other methods that we evaluated.

The car dataset consists of 5,997 models used for training and 1,500 used for testing. Rendering 54 views per each model [7] results in 323,838 training images and 81,000 testing images. The chairs dataset consists of 558 training models and 140 testing models, resulting in 30,132 training images and 7,560 testing images.

Note that, while the training and testing images were rendered at 20-degree intervals around the vertical axis, in our supplementary video we provide examples of models rendered at 10-degree intervals. This demonstrates that our method is able to generalize to intermediate poses not seen during training. In contrast, for their ShapeNet evaluations, [5] uses one-hot vectors indicating the discrete azimuth and elevation intervals at which the source images were rendered, and the specified pose for the target image. It is thus unclear how or whether their method would be able to generalize to intermediate poses not used for training.

Our NVS results for cars in the supplementary video also demonstrate that our network is able to synthesize transparent features such as the glass in the car windows.

4.1.2. Human Action Dataset

Each subject is rendered while performing 48 animation sequences, using rigged human models (varying in gender, ethnicity, size, age, and clothing) and animation sequences obtained from Renderpeople [3]. For 4 frames selected at regular intervals in each animation sequence, the subjects are rendered at 12 viewpoints sampled at 30-degree intervals around the vertical axis. This results in 428,544 images. We use 128 subjects for training and the remaining 58 for evaluation, resulting in a total of 294,912 training images and 133,632 testing images.

While we use 30-degree increments for training on this dataset, in our supplementary video we provide synthesis results in which the subject is rendered at 15-degree intervals. This further demonstrates our method's generalization capabilities.

4.2. 3D Reconstruction

To measure our framework's 3D reconstruction capabilities and compare it to recent work, we use the dataset and evaluation framework provided by [6][8].

The dataset consists of rendered images of ShapeNet models from 3 object categories: chairs, cars and aeroplanes. We use 2831/810/404 models for training/testing/validation for the aeroplane dataset, 5247/1500/750 models for the car dataset and 4744/1356/678 models for the chair dataset. There are 10 images per each model, rendered with varying lighting conditions and the viewpoint azimuth and elevation uniformly sampled at random intervals in the ranges $[0, 360)$ and $[-20, 30]$, respectively.

While the images are rendered at a resolution of 224×224, we bilinearly downsample them to 64×64 for our network, which results in the $32 \times 32 \times 32$ occupancy volume that we use for evaluation. In contrast, we use images of size 160×160 and a $40 \times 40 \times 40$ feature volume for our novel view synthesis task. These 3D reconstruction results thus demonstrate that our network is able to extract meaningful structure from the input images even in the case of low input resolution and a smaller volumetric bottleneck resolution.

[4] We use the loss computed on the conv1_1, conv2_1, conv3_1, and relu3_3 layers of the VGG-19 network.

[5] The official code release, with pre-trained models and datasets, can be found at https://github.com/shaohua0116/Multiview2Novelview.

[6] Using a larger volumetric bottleneck results in substantially higher memory usage and much longer training times

[7] 18 azimuth angles sampled at 20-degree intervals and 3 elevations (0, 10 and 20 degrees).

[8] The official code release, with pre-trained models and tools for generating these datasets and evaluating the reconstruction results, can be found at https://github.com/shubhtuls/drc.

5. Segmentation Supervision Ablation Study

As discussed in the paper, we supervise our networks using a segmentation loss given the ground-truth foreground segmentation masks for each image. While this is useful for performing 3D reconstruction, to determine how crucial this supervision is for our approach to novel view synthesis we conducted an ablation study using a reduced version of our model. The architecture and training procedure is as described above, except that we use input images of a resolution of 128×128 and a bottleneck resolution of 32^3. Random noise was used as the background for each input image. We found that our approach worked comparably well in reconstructing the foreground of the target evaluation images with and without this supervision.

Using the evaluation framework described in Sec. 4.2 for the chair dataset (using 4 input images for each target image), with segmentation supervision we achieved an average SSIM of 0.921 and an L1 loss (computed only for the foreground pixels of the target evaluation images) of 0.189. Without segmentation supervision, we achieved an SSIM of 0.920 and an L1 loss of 0.182. This suggests that, while useful for 3D reconstruction, this loss is not strictly necessary for novel view synthesis, as when it is omitted the network still learns to extract the features necessary to transform the foreground image content to the target view.

6. Creative Manipulation Implementation.

To perform spatial transformations to the encoded feature volume, we assign a 3-dimensional coordinate $p_i = (x, y, z)$ to each cell i corresponding to its spatial position in the volume, such that coordinate $(0, 0, 0)$ corresponds to the center of the bottleneck volume. During training, given an input and output image pair $\{I_k, I_l\}$, we apply a rigid transformation corresponding to the relative pose of these images to these coordinates to determine the spatial position p'_i in the transformed feature volume corresponding to p_i in the original feature volume. This provides us with the flow field $F_{k \to l}$ used to sample the encoded feature volume to produce the transformed feature volume that is passed to the decoder (see Sec. 3.1).

During training, we only apply rigid transformations corresponding to changes in the azimuth (rotations around the vertical axis, corresponding to the y-axis in our representation) and elevation (rotations around the horizontal x-axis) of the viewpoint of the scene. However, in our results we demonstrate that this training process allows for performing *non-rigid* transformations that enable a large variety of plausible manipulations to the image content. Here we describe in more detail the method in which we perform these transformations to obtain the results seen in our paper (Figs. 1, 6-7) and the supplementary video.

Vertical and Horizontal Stretching. Rather than directly sampling from the region in the encoded volume based on the rigid pose between the input and output views as described, we can vary the sampling strategy to produce stretching effects such as those seen in our results (Fig. 1, row 4, Fig. 6, rows 1-2, and Fig. 7 in the main paper, and in the supplementary video, 2:50-3:40, 3:58-4:14, and 5:40-6:05).

For example, suppose that we have n regularly sampled values corresponding to the y-positions of the cells in a slice of the transformed feature volume. We can find the corresponding values $(y_0, ..., y_{n-1})$ to use when sampling the encoded feature volume as follows:

$$y_i = a + \frac{b - a}{n - 1} \times i \tag{2}$$

Thus we have $y_0 = a$ and $y_{n-1} = b$. By adjusting a and b we can change the position and size of the region in the input volume from which values are sampled, which will alter how the transformed region is compressed or stretched in the decoded image. We use this technique with multiple slices with input positions that vary over time to produce the vertically stretching chair animations in our results. (After sampling the input volume as described, we apply a rigid rotation to produce the novel viewpoints seen in these images.)

We can apply similar techniques to produce stretching effects in the x- and z-dimensions. By varying the slice parameters over time, we cause the cars seen in our supplementary video to vary in length and width over the course of the animation.

Vertical Twisting. We can also apply different rigid rotations to separate regions of the encoded feature volume, we can achieve the "twisting" effect seen in our results (Fig. 6, row 3 in the main paper and in the supplementary video, 2:50-3:40). Given a user-specified point on the vertical y-axis and a rotation value α, we apply a rigid rotation around the y-axis to the feature volume of α degrees for all cells above this point and $-\alpha$ degrees for all cells below this point. Varying the α parameter over time produces the twisting effect seen in the swivel chairs portrayed in our results.

Volume Merging and Reflection. By combining the content of different regions of multiple encoded feature volumes, we can decode images in which this content has been merged in a corresponding fashion as seen in Fig. 1 of the main paper (row 4, columns 4-6) and in the supplementary video (4:14-4:40). In these examples, the top and bottom halves of the feature volumes for 2 different individuals have been combined to produce new subjects with an appearance corresponding to the upper half of the first subject and the lower half of the second. Note that while this ap-

pears to produce an effect similar to that of merging the upper and lower regions of the rendered images, after performing this merging *once* for the encoded bottlenecks for each subject, we can rigidly transform the result to produce novel views of the subject, as seen in the supplementary video.

Similarly, we can alter and replicate regions of a single bottleneck to produce novel content, as seen in the slicing and stitching examples in the main paper (Fig. 6, rows 4-5) and in the supplementary video (3:40-3:58). For these examples, we discard the feature volume content contained within one half of the xy-plane, and reflect the content of the cells in the remaining half across this plane to fill the missing regions. As a result we can produce new shapes in which the front of the depicted car has been replaced by the back (Fig. 6, row 4), or vice-versa (Fig. 6, row 5). As before, we can also apply rigid transformations to the result to produce novel views of the image content. Interestingly, these rendered results still plausibly produce the specified manipulation of the encoded feature volume, though cars with such unusual shapes were never seen during training.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] Renderpeople, 2018. `http://renderpeople.com`.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[5] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J. Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision*, 2018.

[6] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[7] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.