Deep View Morphing

Alex R. Kuefler
Department of Symbolic Systems
Stanford University
akuefler@stanford.edu

Abstract

View morphing takes two views of a 3D object and infers the appearance of the object at intermediary views. This problem is traditionally approached through the lens of epipolar geometry, but such techniques require computing pixel correspondences, and are brittle to changes in visibility. This work introduces a neural network based approach for learning to generate morphs of 3D objects. Three experiments are performed to assess the model's ability to reconstruct images and generalize to unseen objects.

Keywords
Deep learning, view morphing, deconvolutional neural networks, image reconstruction.

1. Introduction

A remarkable property of the visual system is its ability to infer 3D structure from a limited number of 2D projections. This process, called "stereopsis", has been the focus of intensive study in computer vision [1] as well as cognitive science [2, 3]. Due to the inherent ambiguity of mapping from lower to higher-dimensional spaces, depth perception is not merely a sensory experience, but it is also an act of perception that engages cognitive faculties such as world-knowledge and memory [4].

This work presents a convolutional neural network (CNN) for learning properties of objects' 3D geometry from 2D images. In particular, the Deep View Morphing Network (DVMN) receives two images of an object, each taken from a different camera angle, and generates an image of how the object would appear if viewed from an intermediary angle. This work extends previous, theory-based View Morphing paradigm using deep machine learning.

2. Background

Image Morphing has been well studied over the past decades and has traditionally relied on various interpolation procedures for smoothly transforming one image to another by generating intermediary frames. For example, [5] introduces two-pass mesh warping algorithms, which given images I_0 and I_1 computes mesh-grids M_0 and M_1 whose vertices lie on image landmarks and are constrained to be equivalent topologically. The warping problem then reduces to linear interpolation between M_0 and M_1. What Image Morphing techniques such as this share in common is that they consist of feature specification, warp generation, and transition control stages [6]. For example, in mesh warping, feature specification occurs with the initialization of M_0 and M_1 whose graphical organization can essentially be thought of as whole-image descriptors.

In contrast, the View Morphing paradigm leverages an understanding of projective geometry in order to produce more convincing warps. [7] originally describes a procedure where images I_0 and I_1 undergo pre-warping, such that $\hat{I}_0 = H_0^{-1} I_0$ and $\hat{I}_0 = H_1^{-1} I_1$ where H_0^{-1} and H_1^{-1} are projective transformations. Interpolating between the projected images $\hat{I}_0$ and $\hat{I}_1$ then produces a combined representation I_s that is then re-projected back into the reference frame to form an image. By performing linear interpolation in the projected space rather than the input-space, View Morphing accurately models changes in viewpoint and projective shape, where Image Morphing only preserves shape under parallel image translations and zooms.

Later work has improved the fidelity of Image Morphing [8] or attempted to generalize View Morphing to more than two cameras [9], but the 1996 algorithm introduced by [7] appears to be the state of the art for warps that preserve intermediate 3D geometry. Nevertheless, there are three notable shortcomings of this approach. First, view morphing relies on principles of epipolar geometry, requiring that the camera matrices be known and that pixel correspondences are found. Any misalignments in correspondences can potentially produce artifacts in the morph. Second, view morphing assumes that objects to be morphed have near-constant visibility, and is brittle when the difference between two views is characterized by an extreme angle. And finally, evaluating the performance of the morphing algorithm is largely subjective, and is not easily measured quantitatively. This point remains true for many image generation techniques, particularly those in the unsupervised learning literature [10].

2.1. Contribution to Past Work

The View Morphing method proposed in this project potentially overcomes these three shortcomings identified for past approaches. Because the DVMN performs all transformations using learnable weight-matrices, there is no need to estimate camera parameters or identify corresponding points. Similarly, the DVMN compresses image inputs into a low-dimensional feature vector, rather than mapping them onto projection planes. As such, there is no reason in principle to believe this "pre-warped" representation is subject to the constraints of projective transformations. Thus morphs may still be robust between extremely different

views, assuming the network is trained on sufficient data. Furthermore, because the model is trained with stochastic gradient descent (SGD) to minimize an objective, the quality of its morphs can be assessed quantitatively by measuring the reconstruction loss on a validation or testing set.

More generally, this approach is consistent with a larger trend in Artificial Intelligence wherein data-driven techniques replace traditional, theory-based approaches. Whereas pattern recognition tasks such as image classification have a much longer history of relying on machine learning, reconstruction and 3D-inferrence are strongly supported by a theory of projective geometry. Approaches like the DVMN present model-free alternatives to previous algorithms.

3. Approach

The DVMN is trained to minimize image reconstruction error using supervised backpropagation, similarly to the approach taken by [11]. Formally, let X be a set of training examples where $X = \{(x_1^1, x_2^1), \ldots, (x_k^n, x_l^n)\}$ and let $Y = \{y_{1,2}^1, \ldots, y_{k,l}^n\}$ be a set of target values, such that x_j^i is the image of the i^{th} 3D object viewed at angle j and $y_{j,j'}^i$ is the image of the i^{th} object viewed at angle $\frac{|\theta_j - \theta_{j'}|}{2}$. Then the network h_θ is trained to minimize the Euclidean error,

$$J(\theta) = \sum_{i=1}^{n} \sum_j \sum_{j' \neq j} \left\| h_\theta \left(x_j^i, x_{j'}^i \right) - y_{j,j'}^i \right\|$$

with respect to the network parameters θ. In practice, the i^{th} object contributes numerous views to the training set, and the viewing angles j and j' are chosen such that $|j - j'| \, mod \, 360 > 10$, ensuring the two constituent images of a training example are sufficiently different to have a meaningful intermediate angle.

3.1. Dataset

All examples were drawn from the Caltech "3D Objects on Turntable" database [18]. This dataset consists of images of approximately 95 3D objects placed on a rotating turntable and photographed every 5 degrees with a high-resolution camera. For the purposes of this project, a subset of 7 objects were chosen to form the training and testing sets, which consisted of anywhere between 20-4,000 examples, depending on the experiment (where an "example" is a tuple consisting of multiple images of the same object from various viewing angles).

Figure 1. K-means segmentation.

3.2. Preprocessing

Given that the network's major task was to learn to represent the rotation between two objects, and only secondarily concerned with generating high-quality images, foreground segmentation was used to preprocess the data. All images were clustered using k-means (where k is 7-18 centroids depending on the contrast of the 3D object against the background). Pixels whose nearest centroid consisted of RGB values whose channels all exceed 100 were masked black, as they predominantly belong to the background of this particular dataset. This segmentation procedure produced some imperfections, but largely succeeded in foregrounding the turntable and 3D object against a purely black background (See **Fig. 1**). Images were also downsampled to $2^s \times 2^s$ RGB pixels where the scale s was typically set to be 6. The resulting 64x64 color images contain a considerable amount of detail, while greatly reducing the computational expense of training and prediction.

3.2. Architectural Details

The DVMN consists of two major "sub-networks" trained end-to-end. An extractor sub-network E takes a $\left(x_j^i, x_{j'}^i\right)$ tuple as input and calculates a merged feature vector $\phi(x^i)$ for the entire example. A generator network G takes $\phi(x^i)$ as input, performs a number u of upsampling operations, and outputs an estimated image $\hat{y}_{j,j'}^i \in \mathbb{R}^{2^s \times 2^s \times 3}$.

The specific architecture of E differs little from most CNN formulations. It consists of two convolutional layers with stride 1 and padding 1, ensuring the input image representations retain their dimensionality. These layers are followed by a third convolution and 4x4 max pooling, ag-

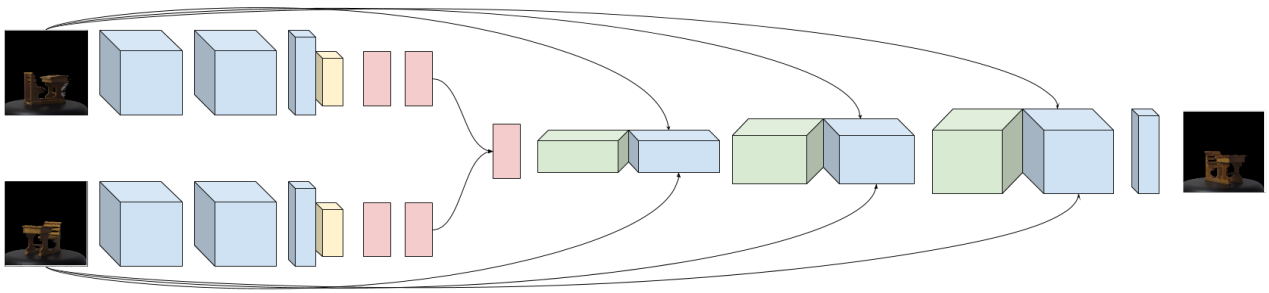

Figure 2. DVMN Architecture with skip connections. Convolutional layers (blue), Max pooling (yellow), FC layers (red), and upsampling (green).

gressively downsampling the feature maps. The downsampled feature maps are then processed with two fully-connected (FC) layers to produce a flattened $\phi(x^i) \in \mathbb{R}^{2^d \times 2^d \times 3}$, where $d = s - u$.

The architecture of G was designed to perform one of a few possible upsampling methods on its input:

Upconvolution (variously known as "convolution transpose", "deconvolution", or "fractionally strided convolution") is a deep learning operation that has seen use in image generating networks [12, 13]. By associating single units in an input layer to a volume of outputs through a weight matrix, they can learn to undo 2D-convolutions, producing output volumes from lower-dimensional representations.

Up-Down Convolution follows an upconvolution with a classic convolutional layer in which zero-padding and stride have been chosen in such a way to preserve the dimensions. [11] suggest that this works better in image reconstruction tasks empirically. Intuitively, the upconvolution can be thought of as being responsible for upsampling, where the proceeding convolution is responsible for fine-tuning and smoothing the blown-up image.

Simple Up-Down observes that image resizing through bilinear interpolation [14] is a differentiable operation that can be built into a neural network without adding more parameters. Some instantiations of DVMN essentially perform Up-Down Convolution where the "Up" stage is replaced with parameter-free image resizing.

The chosen upsampler is repeated u times and followed by a convolutional layer with 3 kernels, producing a $2^s \times 2^s \times 3$ output volume readily interpretable as an RGB image. Rectified linear (ReLU) or simply linear units are used in the output layer. Exponential linear units (ELU) are used everywhere else throughout the network [15].

A final detail of the structure of G is the use of skip-connections. [16] observes that it may be easier to learn a "residual" mapping than a function based on stacked layers. If $\mathcal{H}(x)$ is an underlying function to be approximated by a CNN whose predictions are defined as $\mathcal{F}(x) := \mathcal{H}(x) - x$, then the optimization problem reduces to learning $\mathcal{F}(x) + x$. Intuitively, a network that allows its input x to "skip" ahead into a hidden representation $\mathcal{F}(x)$ facilitates backward flow of gradients, bypassing intermediate transformations by the network's weight matrices.

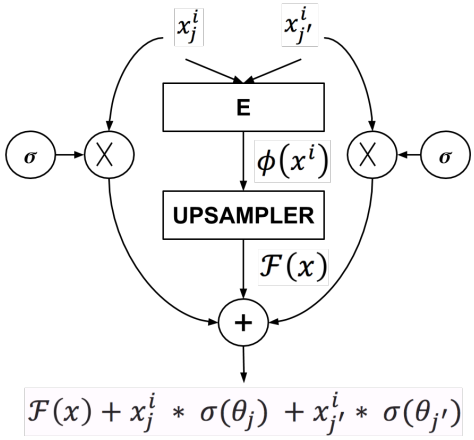

Figure 3. Gated Skip-Connection.

Because the output images of the DVMN are so similar to the inputs (generally retaining the same colors and many spatial properties), it is reasonable to suspect that skip connections will improve image generation. In other words, if the input images x_j^i and $x_{j'}^i$ are immediately present in G, then the DVMN may be able to focus more of its optimization on capturing the delta between the two images in $\phi(x^i)$, rather than capturing properties stable between the two images (e.g., color). A gating mechanism was also im-

Figure 4. Reconstructions from 18 conditions for experiment 1.

plemented, allowing the DVMN to learn elementwise, sigmoid weights controlling the extent to which x_j^i and x_j^i, are added into the a hidden representation.

4. Results

Three experiments were carried out in order to evaluate the DVMN's ability to generate convincing images, its performance on a larger sample of data, and its generalization to unseen objects.

4.1. Experiment 1

The first experiment was designed to assess which architectural settings were the most conducive for image generation. Between the filter sizes, the number of layers, the type and number of upsamplers, the use of (gated) skip connections and more, there were many choices for network structures.

Table 1. 18 conditions of experiment 1.

Upsampler	Skip Connect.	Output
Upconv.	False	Linear
Upconv.	False	ReLU
Upconv.	Gated	Linear
Upconv.	Gated	ReLU
Upconv.	True	Linear
Upconv.	True	ReLU
Up-Down	False	Linear
Up-Down	False	ReLU
Up-Down	Gated	Linear
Up-Down	Gated	ReLU
Up-Down	True	Linear
Up-Down	True	ReLU
Simple-Up	False	Linear
Simple-Up	False	ReLU
Simple-Up	Gated	Linear
Simple-Up	Gated	ReLU
Simple-Up	True	Linear
Simple-Up	True	ReLU

A single turntable object b (i.e., a bunch of bananas) was chosen to generate training examples by computing every combination of viewing angle (subject to the constraint that the two viewing angles forming an input tuple were at least 10 degrees apart). For concreteness, the sample then consisted of tuples of the form $S = \{(x_0^b, x_{10}^b, y_5^b), (x_0^b, x_{20}^b, y_{10}^b), \ldots (x_{345}^b, x_{355}^b, y_{350}^b)\}$. 20 training and 20 testing samples were chosen from S to form the datasets for experiment 1.

Eighteen experimental conditions were created by selectively changing the upsampling method, the presence of skip connections (and the presence of gates when skip connections were used), and the activation function of the output layer.

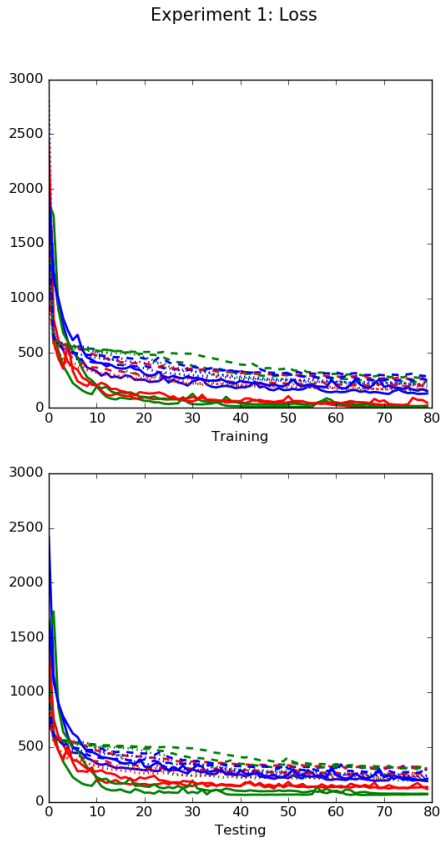

Figure 5. Training and testing loss in Exp. 1. Color indicates upsampling: simple (red), up-down (green), upconv (blue). Solid lines have no skip connections. Dashed lines are gated, dotted are not.

The network was trained for 400 epochs in each condition, using SGD with Adam updates and the hyperparameter settings recommended in [17]. **Fig. 5** displays the training and testing loss every 5 iterations. Two notable features emerge from the loss plots. First, a significant gap appears between those conditions in which skip-connections were used and those in which they were not, indicating that skip-connections actually impeded image reconstruction. Second, mere *upconvolution* based methods underperformed the other two upsamplers, achieving its lowest testing loss at 188.0 as opposed to 61.2 (*Up-Down Convolution*) and 115.5 (*Simple Up-Down)*. From these results, it seems fair to conclude that skip-connections (as they have been described here) should be avoided for image generation, and

that post-convolutions after upconvolution layers do appear to make a significant difference, as predicted by [11].

These findings are borne out by the reconstructed images shown in **Fig. 4**. The visualizations indicate that the use of linear activation functions cause shimmers to appear around the reconstruction. This effect is caused by pixels becoming "dead" as they fall out of the 0-255 range, and can be corrected by changing display settings (e.g., thresholding pixels before plotting). More seriously, the high loss caused by the *upconvolutions* and skip-connections is made visible in the blurriness of their respective reconstructions. The skip-connections in particular cause the blurred image to smear over a greater horizontal area. This effect is likely caused by a large amount of the input views remaining residually in the reconstruction: b, viewed in profile, is wider than it is tall and the additive effect of the skip-connections remembers this.

Interestingly, the parameter-free upsizing operation performed during *Simple Up-Down* appeared to perform on par with the significantly more complicated *Up-Down Convolution*.

4.2. Experiment 2

The setup of the second experiment is largely identical to that of Experiment 1, however, the size of the dataset under consideration was increased and a second 3D object, t, (a table) was added. Again, a sample of data was created by constructing combinations of the viewing angles, such that $S = \{(x_0^b, x_{10}^b, y_5^b), (x_0^t, x_{10}^t, y_5^t), \dots\}$. 4,960 examples were then chosen from S and split evenly into training and testing sets.

Table 2. 4 conditions of experiment 2.

Upsampler	u
Up-Down	4
Up-Down	5
Simple-Up	4
Simple-Up	5

Considering the findings of Experiment 1, Experiment 2 consists of fewer conditions. Output nonlinearities were fixed as ReLUs and skip connections were excluded from all conditions. Instead, the upsampling method and the number u of upsampling operations (which also controls the dimensionality of $\phi(x^i)$) were manipulated. Considering the increased amount of data, training also occurred over only 50 epochs.

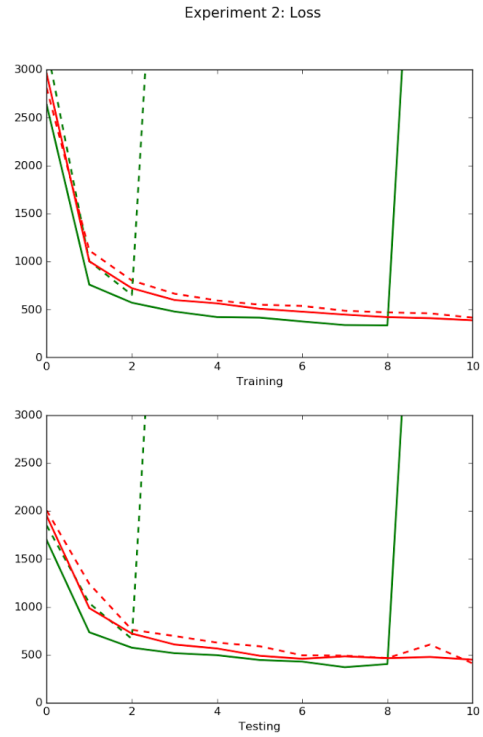

Figure 6. Training and testing loss in Exp. 2. Color indicates upsampling: simple (red), up-down (green). Dashed lines have 5 upsamples, solid 4.

As show in **Fig. 6**, the *Up-Down Convolution* networks achieved lower loss both during training and testing, but showed a tendency to diverge. The loss of the $u = 5$ condition spiked particularly early on. This instability may be related to the power of the model combined with the low dimensionality of $\phi(x^i)$, which must compress information from a 12,288-valued matrix (i.e., the 64x64x3 image) into only 12 values. Training of the *Simple Up-Down* networks was more stable.

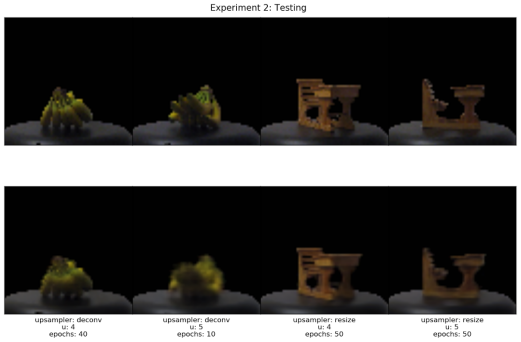

Figure 7. Exp. 2 Reconstructions. Top row is ground truth. Bottom is predicted.

The images reconstructed after 50 epochs again appeared fairly crisp. Even images generated by the *Up-Down Convolution* networks before their training diverged look mostly correct, if a little fuzzy.

4.3. Experiment 3

So far the reconstructions have looked promising, and the training and testing plots reveal little overfitting. However, the argument can be made that the networks' good performance may be caused by high correlation between the training and testing sets. Because the networks train on every single image appearing in the testing set (the difference between training and testing is not the images used, but the combinations of viewing angles), its task may be described as:

- Learn how to copy every image viewed during training.
- During inference, select the correct viewing angle from the set of memorized images.

As such, the networks may be behaving more like nearest neighbor classifiers, rather than truly generative models.

The third experiment re-designs the learning task to something more generalizable. Instead, we want the network to:

- Learn a low-dimensional representation capturing perspective information from both viewing angles.
- During inference, expand this representation into an appropriate, but perhaps completely novel image.

These goals are accomplished by changing the manner in which the training and testing sets are formed. A subset of 7 turn-table objects were chosen and each appeared only during training, or during testing. Training examples were created combinatorially as before, but drawn from the set of views of the bananas, table, clock, and flower objects. The testing set included views of the motorcycle, teddy bear, and car objects. In total, a random sample of 4000 training and 2500 testing examples were chosen.

Table 3. 4 conditions of experiment 3.

Upsampler	Weight-Sharing
Up-Down	True
Up-Down	False
Simple-Up	True
Simple-Up	False

Due to the goal of generalizing information about 3D structure to novel objects, greater care was taken to regularize the model and reduce overfitting. L2 regularization was introduced to the loss function and a 25% chance of dropout occurred in all trainable layers. Observe also that applying two different E networks to the two different input images may be redundant, as feature extraction should act symmetrically (i.e., the network should give the same output to the same two object views, irrespective of their order in the input tuple). As such, weight-sharing in E is applied for some experimental conditions, leveraging domain knowledge to reduce the number of parameters.

In order to combat divergence during training, the learning rate was decreased by an order of magnitude and u was held at 3 (ensuring $\phi(x^i) \in \mathbb{R}^{192}$). Otherwise, the hyperparameters (exponential decay rate, filter numbers, sizes, etc.) remained the same. Networks were set to train for 50 epochs.

Despite these adjustments, two of the models diverged while training. Interestingly, it was the *Simple Up-Down*, rather than the *Up-Down Convolutional* networks that diverged. An explosion in the L2 regularization term, whose value approached infinity, appears to be the cause of divergence. Later work should compute this in a more numerically stable way.

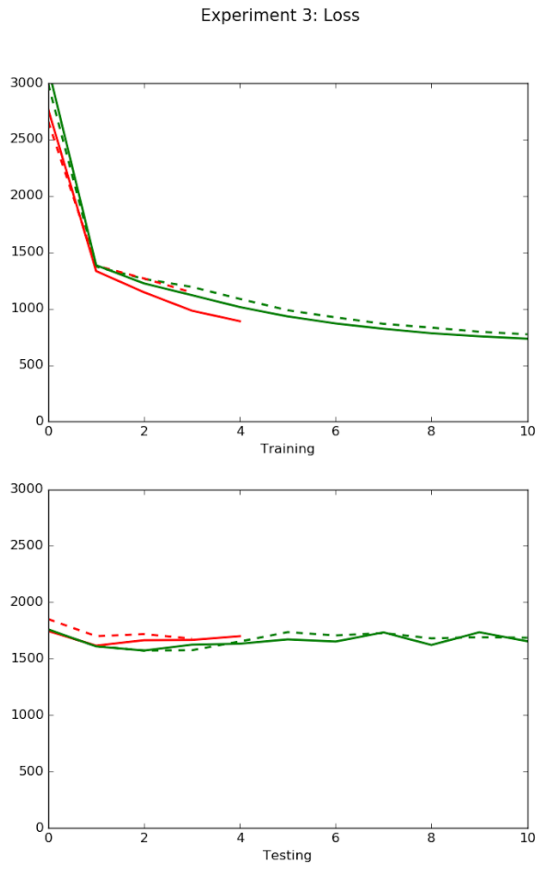

Figure 7. Training and testing loss in Exp. 3. Color indicates upsampling: simple (red), up-down (green). Dashed lines indicate weight-sharing.

As evidenced by the loss plot, efforts to regularize the network were unsuccessful. Those models that succeeded in training the entire 50 epochs showed almost no ability to generalize in the allotted time, as the testing curve remained nearly flat.

Figure 8. Exp. 3 Training Reconstructions. Top row is ground truth. Bottom is predicted.

Based on the reconstructions in **Fig. 9**, it appears that the network, when confronted with novel objects, attempts to reconstruct an amalgam of training set objects. For example, the *Simple Up-Down* network (third image) produced a reconstruction whose green base strongly resembles that of the flower, but is elongated like a view of the clock. The orange-yellow coloration of the testing reconstructions is likely caused by the presence of the bananas and table in the testing set. The training reconstructions (**Fig. 8**) appear to demonstrate that weight-sharing seems like a viable option for the DVMN, but the networks may not have trained for long enough to make a definitive judgment.

Figure 9. Exp. 3 Training Reconstructions. Top row is ground truth. Bottom is predicted.

5. Conclusions

This work attempts to address the problem of View Morphing from a deep learning perspective. A CNN based model is presented and a number of architectural details are explored. Arguments have been made for (and against) the use of skip-connections in image reconstruction. The usefulness of different approaches to learnable upsampling was also examined.

The results of Experiments 1 and 2 demonstrate that the DVMN is capable of generating faithful reconstructions and producing the appropriate image in response to different viewpoint combinations. However, experiment 3 showed that this approach generalizes poorly to novel objects. A major limiting factor is the amount of data used. It may be unreasonable to expect a model of this complexity to learn how to infer 3D structure from viewing only 4 objects, even if the number of views from which to learn is approximately 4000. Fortunately, this problem can likely be addressed with more training time and more computing power. The use of computer-generated images may also be useful for creating 3D examples viewable from many angles.

It is also worth noting that whereas the DVMN has clearly defined feature specification (extraction with E) and warp generation (inference with G), unlike other Image Morphing methods, there is no notion of transition control. The DVMN is constrained to producing views at the midpoint angle. Follow-up work may experiment with allowing the DVMN to produce smooth warps, perhaps by allowing it to accept a rotation value as input, in addition to the two viewing images.

Although skip-connections as described here were unbeneficial during reconstruction, there may be other formulations that improve generalization. Supplying the skips with their own transformation matrices (perhaps approximating the projective transforms performed in traditional View Morphing) may allow more information from the input to sneak past the dimensional bottleneck at $\phi(x^i)$. Such an approach would ensure color and shape information remains present during image generation, and would interpolate nicely between the deep learning and epipolar paradigms.

References

[1] Zhang, Z. (1998). Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, *27*(2), 161-195.

[2] Marr, D., Poggio, T., Hildreth, E. C., & Grimson, W. E. L. (1991). *A computational theory of human stereo vision* (pp. 263-295). Birkhäuser Boston.

[3] Mayhew, J. E., & Frisby, J. P. (1981). Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, *17*(1), 349-385.

[4] Churchland, P. M. (1996). The engine of reason, the seat of the soul: A philosophical journey into the brain. MIT Press.

[5] Smythe, D. B. (1990). A two-pass mesh warping algorithm for object transformation and image interpolation. *Rapport technique*, *1030*, 31.

[6] Wolberg, G. (1998). Image morphing: a survey. *The visual computer*, *14*(8), 360-372.

[7] Seitz, S. M., & Dyer, C. R. (1996, August). Toward image-based scene representation using view morphing. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on* (Vol. 1, pp. 84-89). IEEE.

[8] Shechtman, E., Rav-Acha, A., Irani, M., & Seitz, S. (2010, June). Regenerative morphing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 615-622). IEEE.

[9] Georgiev, T., & Wainer, M. (1997). Morphing between multiple images. S. Illinois Univ. at Cardbondale dept. of Comp. Science, Technical Report, 17.

[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672-2680).

[11] Dosovitskiy, A., Tobias Springenberg, J., & Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1538-1546).

[12] Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010, June). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 2528-2535). IEEE.

[13] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* preprint arXiv:1511.06434.

[14] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemaway, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, M., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, Z. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* preprint arXiv:1603.04467.

[15] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv* preprint arXiv:1511.07289.

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385.*

[17] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization.*arXiv preprint arXiv:1412.6980.*

[18] Moreels, P., & Perona, P. (2005, October). Evaluation of features detectors and descriptors based on 3D objects. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* (Vol. 1, pp. 800-807). IEEE.