Deep View Morphing

Dinghuang Ji*
UNC at Chapel Hill
jdh@cs.unc.edu

Junghyun Kwon†
Ricoh Innovations
junghyunkwon@gmail.com

Max McFarland
Ricoh Innovations
max@ric.ricoh.com

Silvio Savarese
Stanford University
ssilvio@stanford.edu

Abstract

Recently, convolutional neural networks (CNN) have been successfully applied to view synthesis problems. However, such CNN-based methods can suffer from lack of texture details, shape distortions, or high computational complexity. In this paper, we propose a novel CNN architecture for view synthesis called "Deep View Morphing" that does not suffer from these issues. To synthesize a middle view of two input images, a rectification network first rectifies the two input images. An encoder-decoder network then generates dense correspondences between the rectified images and blending masks to predict the visibility of pixels of the rectified images in the middle view. A view morphing network finally synthesizes the middle view using the dense correspondences and blending masks. We experimentally show the proposed method significantly outperforms the state-of-the-art CNN-based view synthesis method.

1. Introduction

View synthesis is to create unseen novel views based on a set of available existing views. It has many appealing applications in computer vision and graphics such as virtual 3D tour from 2D images and photo editing with 3D object manipulation capabilities. Traditionally, view synthesis has been solved by image-based rendering [1, 19, 20, 28, 22, 9, 27, 30] and 3D model-based rendering [15, 25, 33, 14, 7, 31, 12] .

Recently, convolutional neural networks (CNN) have been successfully applied to various view synthesis problems, *e.g.*, multi-view synthesis from a single view [32, 29], view interpolation [6], or both [36]. While their results are impressive and promising, they still have limitations. Direct pixel generation methods such as [32] and [29] have a main advantage that the overall geometric shapes are well predicted but their synthesis results usually lack detailed textures. On the other hand, the pixel sampling methods such as [6] and [36] can synthesize novel views with detailed textures but they suffer from high computational complexity [6] or geometric shape distortions [36].

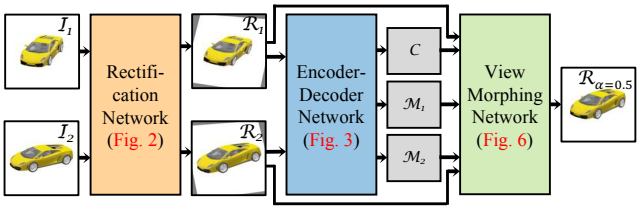

Figure 1. Overall pipeline of Deep View Morphing. A rectification network (orange, Section 3.1) takes in $\mathcal{I}_1$ and $\mathcal{I}_2$ and outputs a rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$. Then an encoder-decoder network (blue, Section 3.2) takes in $\mathcal{R}_1$ and $\mathcal{R}_2$ and outputs the dense correspondences $\mathcal{C}$ and blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$. Finally, a view morphing network(green, Section 3.3) synthesizes a middle view $\mathcal{R}_{\alpha=0.5}$ from $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{C}$.

In this paper, we propose a novel CNN architecture that can efficiently synthesize novel views with detailed textures as well as well-preserved geometric shapes under the view interpolation setting . We are mainly inspired by View Morphing, the classic work by Seitz and Dyer [27], which showed it is possible to synthesize shape-preserving novel views by simple linear interpolation of the corresponding pixels of a rectified image pair. Following the spirit of View Morphing, our approach introduces a novel deep CNN architecture to generalize the procedure in [27]—for that reason, we named it Deep View Morphing (DVM).

Figure 1 shows the overall pipeline of DVM. A rectification network (orange in Fig. 1) takes in a pair of input images and outputs a rectified pair. Then an encoder-decoder network (blue in Fig. 1) takes in the rectified pair and outputs dense correspondences between them and blending masks. A view morphing network (green in Fig. 1) finally synthesizes a middle view using the dense correspondences and blending masks. The novel aspects of DVM are:

- The idea of adding a rectification network before the view synthesis phase—this is critical in that rectification guarantees the correspondences should be 1D, which makes the correspondence search by the encoder-

*The majoriy of the work has been done during this author's 2016 summer internship at Ricoh Innovations.

†This author is currently at SAIC Innovation Center.

decoder network significantly easier. As a result, we can obtain highly accurate correspondences and consequently high quality view synthesis results. The rectification network is inspired by [16], which learns how to transform input images to maximize the classification accuracy. In DVM, the rectification network learns how to transform an input image pair for rectification.

- DVM does not require additional information other than the input image pair compared to [36] that needs viewpoint transformation information and [6] that needs camera parameters and higher-dimensional intermediate representation of input images.
- As all layers of DVM are differentiable, it can be efficiently trained end-to-end with a single loss at the end.

In Section 4, we experimentally show that: (i) DVM can produce high quality view synthesis results not only for synthesized images rendered with ShapeNet 3D models [2] but also for real images of Multi-PIE data [10]; (ii) DVM substantially outperforms [36], the state-of-the-art CNN-based view synthesis method under the view interpolation setting, via extensive qualitative and quantitative comparisons; (iii) DVM generalizes well to categories not used in training; and (iv) all intermediate views beyond the middle view can be synthesized utilizing the predicted correspondences.

1.1. Related works

View synthesis by traditional methods. Earlier view synthesis works based on image-based rendering include the well-known Beier and Neely's feature-based morphing [1] and learning-based methods to produce novel views of human faces [30] and human stick figures [19]. For shape-preserving view synthesis, geometric constraints have been added such as known depth values at each pixel [3], epipolar constraints between a pair of images [27], and trilinear tensors that link correspondences between triplets of images [28]. In this paper, DVM generalizes the procedure in [27] using a single CNN architecture.

Structure-from-motion can be used for view synthesis by rendering reconstructed 3D models onto virtual views. This typically involves the steps of camera pose estimation [12, 31, 35] and image based 3D reconstruction [7, 34]. However, as these methods reply on pixel correspondences across views, their results can be problematic for textureless regions. The intervention of users is often required to obtain accurate 3D geometries of objects or scenes [15, 25, 33, 14]. Compared to these 3D model-based methods, DVM can predict highly accurate correspondences even for textureless regions and does not need the intervention of users or domain experts.

View synthesis by CNN. Hinton et al. [13] proposed auto-encoder architectures to learn a group of auto-encoders that learn how to geometrically transform input images. Dosovitskiy et al. [5] proposed a generative CNN architecture to synthesize images given the object identity and pose. Yang et al. [32] proposed recurrent convolutional encoder-decoder networks to learn how to synthesize images of rotated objects from a single input image by decoupling pose and identity latent factors while Tatarchenko et al. [29] proposed a similar CNN architecture without explicit decoupling of such factors. A key limitation of [5, 32, 29] is output images are often blurry and lack detailed textures as they generate pixel values from scratch. In order to solve this issue, Zhou et al. [36] proposed to sample from input images by predicting the appearance flow between the input and output for both multi-view synthesis from a single view and view interpolation. To resolve disocclusion and geometric distortion, Park et al. [26] further proposed disocclusion aware flow prediction followed by image completion and refinement stage. Flynn et al. [6] also proposed to optimally sample and blend from plane sweep volumes created from input images for view interpolation. Recently, Liu et al. [23] adopted tri-linear interpolation to obtain better accuracy in synthesizing new views with two input images.

Among these CNN-based view synthesis methods, [6] and [36] are closely related to DVM as they can solve the view interpolation problem. Both demonstrated impressive view interpolation results, but they still have limitations. Those related to [6] include: (i) the need of creating plane sweep volumes, (ii) higher computational complexity, and (iii) assumption that camera parameters are known in testing. Although [36] is computationally more efficient than [6] and does not require known camera parameters in testing, it still has some limitations. For instance, [36] assumes that viewpoint transformation is given in testing. Moreover, lack of geometric constraints on the appearance flow can lead to shape or texture distortions. Contrarily, DVM can synthesize novel views efficiently without the need of any additional information other than two input images. Moreover, the rectification of two input images in DVM plays a key role in that it imposes geometric constraints that lead to shape-preserving view synthesis results.

2. View Morphing

We start with briefly summarizing View Morphing [27] for the case of unknown camera parameters.

2.1. Rectification

Given two input images $\mathcal{I}_1$ and $\mathcal{I}_2$, the first step of View Morphing is to rectify them by applying homographies to each of them to make the corresponding points appear on the same rows. Such homographies can be computed from the fundamental matrix [11]. The rectified image pair can be considered as captured from two parallel view cameras. In [27], it is shown that the linear interpolation of parallel views yields shape-preserving view synthesis results.

2.2. View synthesis by interpolation

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ denote the rectified versions of $\mathcal{I}_1$ and $\mathcal{I}_2$. Novel view images can be synthesized by linearly interpolating positions and colors of corresponding pixels of $\mathcal{R}_1$ and $\mathcal{R}_2$. As the image pair is already rectified, such synthesis can be done on a row by row basis .

Let $P_1 = \{p_1^1, \ldots, p_1^N\}$ and $P_2 = \{p_2^1, \ldots, p_2^N\}$ denote the point correspondence sets between $\mathcal{R}_1$ and $\mathcal{R}_2$ where $p_1^i, p_2^j \in \Re^2$ are corresponding points when $i = j$. With α between 0 and 1, a novel view $\mathcal{R}_\alpha$ can be synthesized as

$$\mathcal{R}_\alpha\left((1-\alpha)p_1^i + \alpha p_2^i\right) = (1-\alpha)\mathcal{R}_1(p_1^i) + \alpha\mathcal{R}_2(p_2^i), \quad (1)$$

where $i = 1, \ldots, N$. As point correspondences found by feature matching are usually sparse, more correspondences need to be determined by interpolating the existing ones. Extra steps are usually further applied to deal with folds or holes caused by the visibility changes between $\mathcal{R}_1$ and $\mathcal{R}_2$.

2.3. Post-warping

As $\mathcal{R}_\alpha$ is synthesized on the image plane determined by the image planes of the rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$, it might not represent desired views. Therefore, post-warping with homographies can be optionally applied to $\mathcal{R}_\alpha$ to obtain desired views. Such homographies can be determined by user-specified control points.

3. Deep View Morphing

DVM is an end-to-end generalization of View Morphing by a single CNN architecture shown in Fig. 1. The rectification network (orange in Fig. 1) first rectifies two input images $\mathcal{I}_1$ and $\mathcal{I}_2$ without the need of having point correspondences across views. The encoder-decoder network (blue in Fig. 1) then outputs the dense correspondences $\mathcal{C}$ between the rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$ and blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$. Finally, the view morphing network (green in Fig. 1) synthesizes a novel view $\mathcal{R}_{\alpha=0.5}$ from $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{C}$. All layers of DVM are differentiable and it allows efficient end-to-end training. Although DVM is specifically configured to synthesize the middle view of $\mathcal{R}_1$ and $\mathcal{R}_2$, we can still synthesize all intermediate views utilizing the predicted dense correspondences as shown in Appendix C of the arXiv version of the paper [17].

What is common between the rectification network and encoder-decoder network is they require a mechanism to encode correlations between two images as a form of CNN features. Similarly to [4], we can consider two possible ways of such mechanisms: (i) early fusion by channel-wise concatenation of raw input images and (ii) late fusion by channel-wise concatenation of CNN features of input images. We chose to use the early fusion for the rectification network and late fusion for the encoder-decoder network (see Appendix A of the arXiv version of the paper [17] for in-depth analysis). We now present the details of each sub-network.

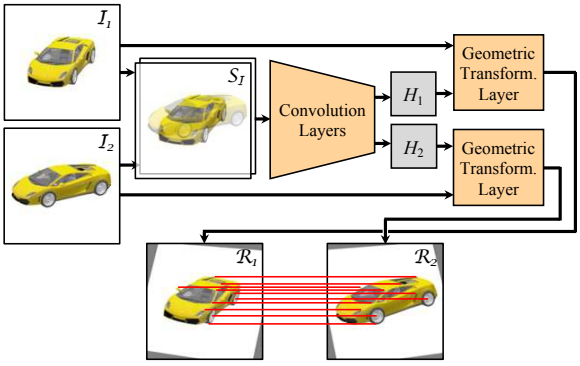

Figure 2. Rectification network of Deep View Morphing. $\mathcal{I}_1$ and $\mathcal{I}_2$ are stacked to be 6-channel input $\mathcal{S}_I$. The last convolution layer outputs two homographies H_1 and H_2 to be applied to $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively, via geometric transformation layers. The final output of the rectification network is a rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$. Red horizontal lines are shown to highlight several corresponding points between $\mathcal{R}_1$ and $\mathcal{R}_2$ that lie over horizontal epipolar lines.

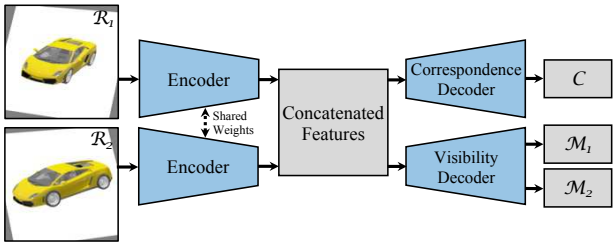

Figure 3. Encoder-decoder network of Deep View Morphing. Each of two encoders sharing weights processes each of the rectified pair. The correspondence decoder and visibility decoder take in the concatenated encoder features and output the dense correspondences $\mathcal{C}$ and blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

3.1. Rectification network

Figure 2 shows the CNN architecture of the rectification network. We first stack two input images $\mathcal{I}_1$ and $\mathcal{I}_2$ to obtain 6-channel input $\mathcal{S}_\mathcal{I}$. Then convolution layers together with ReLU and max pooling layers process the stacked input $\mathcal{S}_\mathcal{I}$ to generate two homographies H_1 and H_2 in the form of 9D vectors. Finally, geometric transformation layers generate a rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$ by applying H_1 and H_2 to $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. The differentiation of the geometric transformation by homographies is straightforward and can be found in Appendix B of the arXiv version of the paper [17].

3.2. Encoder-decoder network

Encoders. The main role of encoders shown in Fig. 3 is to encode correlations between two input images $\mathcal{R}_1$ and $\mathcal{R}_2$ into CNN features. There are two encoders sharing

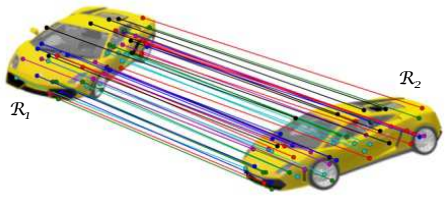

Figure 4. Example of dense correspondences between $\mathcal{R}_1$ and $\mathcal{R}_2$ predicted by the correspondence decoder. For better visualization, $\mathcal{R}_2$ is placed lower than $\mathcal{R}_1$ and only 50 correspondences that are randomly chosen on the foreground are shown.

weights, each of which processes each of the rectified pair with convolution layers followed by ReLU and max pooling. The CNN features from the two encoders are concatenated channel-wise by the late fusion and fed into the correspondence decoder and visibility decoder.

Correspondence decoder. The correspondence decoder shown in Fig. 3 processes the concatenated encoder features by successive deconvolution layers as done in [4, 5, 32, 29, 36]. The last layer of the correspondence decoder is a convolution layer and outputs the dense correspondences $\mathcal{C}$ between $\mathcal{R}_1$ and $\mathcal{R}_2$. As $\mathcal{R}_1$ and $\mathcal{R}_2$ are already rectified by the rectification network, the predicted correspondences are only 1D, *i.e.*, correspondences along the same rows.

Assume that $\mathcal{C}$ is defined with respect to the pixel coordinates p of $\mathcal{R}_1$. We can then represent the point correspondence sets $P_1 = \{p_1^1, \dots, p_1^M\}$ and $P_2 = \{p_2^1, \dots, p_2^M\}$ as

$$p_1^i = p^i, \;\; p_2^i = p^i + \mathcal{C}(p^i), \; i = 1, \dots, M, \qquad (2)$$

where M is the number of pixels in $\mathcal{R}_1$. With these P_1 and P_2, we can now synthesize a middle view $\mathcal{R}_{\alpha=0.5}$ by (1).

In (1), obtaining $\mathcal{R}_2(p_2^i)$ needs interpolation because $p_2^i = p^i + \mathcal{C}(p^i)$ are generally non-integer valued. Such interpolation can be done very efficiently as it is sampling from regular grids. We also need to sample $\mathcal{R}_{\alpha=0.5}(q)$ on regular grid coordinates q from $\mathcal{R}_{\alpha=0.5}(0.5p_1^i + 0.5p_2^i)$ as $0.5p_1^i + 0.5p_2^i$ are non-integer valued. Unlike $\mathcal{R}_2(p_2^i)$, sampling $\mathcal{R}_{\alpha=0.5}(q)$ from $\mathcal{R}_{\alpha=0.5}(0.5p_1^i + 0.5p_2^i)$ can be tricky because it is sampling from irregularly placed samples.

To overcome this issue of sampling from irregularly placed samples, we can define $\mathcal{C}$ differently: $\mathcal{C}$ is defined with respect to the pixel coordinates q of $\mathcal{R}_{\alpha=0.5}$. That is, the point correspondence sets P_1 and P_2 are obtained as

$$p_1^i = q^i + \mathcal{C}(q^i), \;\; p_2^i = q^i - \mathcal{C}(q^i), \; i = 1, \dots, M. \qquad (3)$$

Then the middle view $\mathcal{R}_{\alpha=0.5}$ can be easily synthesized as

$$\mathcal{R}_{\alpha=0.5}(q) = 0.5\mathcal{R}_1(P_1) + 0.5\mathcal{R}_2(P_2), \qquad (4)$$

where both $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ can be efficiently sampled.

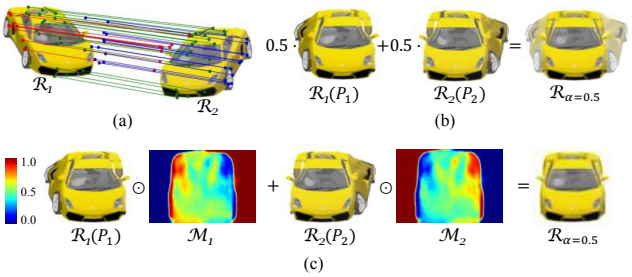

Figure 5. (a) The correspondences for commonly visible regions are predicted accurately (green), but those for regions only visible in $\mathcal{R}_1$ or $\mathcal{R}_2$ are ill-defined and cannot be predicted correctly (red and blue). (b) The middle view synthesized by (4) using all of the correspondences suffers from severe ghosting artifacts. (c) The blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$ generated by the visibility decoder correctly predict the visibility of pixels of $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ in the middle view, and thus we can obtain the ghosting-free middle view by (5). For example, the left side of the car in $\mathcal{R}_1(P_1)$ has very low value in $\mathcal{M}_1$ close to 0 (dark blue) as it should not appear in the middle view while the corresponding region in $\mathcal{R}_2(P_2)$ is the background that should appear in the middle view and hence very high value in $\mathcal{M}_2$ close to 1 (dark red).

Figure 4 shows an example of the dense correspondences between $\mathcal{R}_1$ and $\mathcal{R}_2$ predicted by the correspondence decoder. It is notable that the predicted correspondences are highly accurate even for textureless regions.

Visibility decoder. It is not unusual for $\mathcal{R}_1$ and $\mathcal{R}_2$ to have different visibility patterns as shown in Fig. 5(a). In such cases, the correspondences of pixels only visible in either one of views are ill-defined and thus cannot be predicted correctly. The undesirable consequence of using (4) with all of the correspondences for such cases is severe ghosting artifacts as shown in Fig. 5(b).

In order to solve this issue, we adopt the idea to use blending masks proposed in [36]. We use the visibility decoder shown in Fig. 3 to predict visibility of each pixel of $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ in the synthesized view $\mathcal{R}_{\alpha=0.5}$. The visibility decoder processes the concatenated encoder features by successive deconvolution layers. At the end of the visibility decoder, a convolution layer outputs 1-channel feature map $\mathcal{M}$ that is converted to a blending mask $\mathcal{M}_1$ for $\mathcal{R}_1(P_1)$ by a sigmoid function. A blending mask $\mathcal{M}_2$ for $\mathcal{R}_2(P_2)$ is determined by $\mathcal{M}_2 = 1 - \mathcal{M}_1$. $\mathcal{M}_1$ and $\mathcal{M}_2$ represent the probability of each pixel of $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ to appear in the synthesized view $\mathcal{R}_{\alpha=0.5}$.

Now we can synthesize the middle view $\mathcal{R}_{\alpha=0.5}$ using all of the correspondences and $\mathcal{M}_1$ and $\mathcal{M}_2$ as

$$\mathcal{R}_{\alpha=0.5}(q) = \mathcal{R}_1(P_1) \odot \mathcal{M}_1 + \mathcal{R}_2(P_2) \odot \mathcal{M}_2, \qquad (5)$$

where $\odot$ represents element-wise multiplication. As shown in Fig 5(c), regions that should not appear in the middle view have very low values close to 0 (dark blue) in $\mathcal{M}_1$

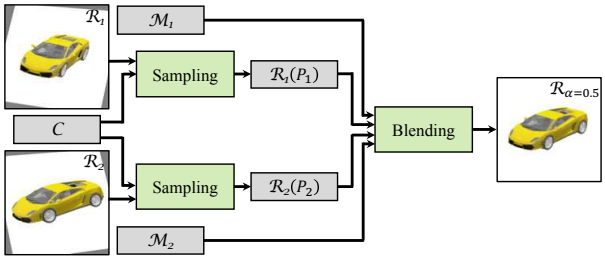

Figure 6. View morphing network of Deep View Morphing. Sampling layers output $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ by sampling from $\mathcal{R}_1$ and $\mathcal{R}_2$ based on the dense correspondences $\mathcal{C}$. Then the blending layer synthesizes a middle view $\mathcal{R}_{\alpha=0.5}$ via (5).

and $\mathcal{M}_2$ while commonly visible regions have similar values around 0.5 (green and yellow). As a result, we can obtain ghosting-free $\mathcal{R}_{\alpha=0.5}$ by (5) as shown in Fig. 5(c).

3.3. View morphing network

Figure 6 shows the view morphing network. Sampling layers take in the dense correspondences $\mathcal{C}$ and the rectified pair $\mathcal{R}_1$ and $\mathcal{R}_2$, and output $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ in (5) by sampling pixel values of $\mathcal{R}_1$ and $\mathcal{R}_2$ at P_1 and P_2 determined by (3). Here, we can use 1D interpolation for the sampling because $\mathcal{C}$ represents 1D correspondences on the same rows. Then the blending layer synthesizes the middle view $\mathcal{R}_{\alpha=0.5}$ from $\mathcal{R}_1(P_1)$ and $\mathcal{R}_2(P_2)$ and their corresponding blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$ by (5). The view morphing network does not have learnable weights as both sampling and blending are fixed operations.

3.4. Network training

All layers of DVM are differentiable and thus end-to-end-training with a single loss at the end comparing the synthesized middle view and ground truth middle view is possible. For training, we use the Euclidean loss defined as

$$L = \sum_{i=1}^{M} \frac{1}{2} ||\mathcal{R}_{\alpha=0.5}(q^i) - \mathcal{R}_{\text{GT}}(q^i)||_2^2, \qquad (6)$$

where $\mathcal{R}_{\text{GT}}$ is the desired ground truth middle view image and M is the number of pixels. Note that we do not need the post-warping step as in [27] (Section 2.3) because the rectification network is trained to rectify $\mathcal{I}_1$ and $\mathcal{I}_2$ so that the middle view of $\mathcal{R}_1$ and $\mathcal{R}_2$ can be directly matched against the desired ground truth middle view $\mathcal{R}_{\text{GT}}$.

3.5. Implementation details

The CNN architecture details of DVM such as number of layers and kernel sizes and other implementation details are shown in Appendix A of the arXiv version of the paper [17]. With Intel Xeon E5-2630 and a single Nvidia Titan X, DVM processes a batch of 20 input pairs of 224×224 in 0.269 secs using the modified version of Caffe [18].

4. Experiments

We now demonstrate the view synthesis performance of DVM via experiments using two datasets: (i) ShapeNet [2] and (ii) Multi-PIE [10]. We mainly compare the performance of DVM with that of "View Synthesis by Appearance Flow" (VSAF) [36]. We evaluated VSAF using the codes kindly provided by the authors. For training of both methods, we initialized all weights by the Xavier method [8] and all biases by constants of 0.01, and used the Adam solver [21] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with the mini-batch sizes of 160 and initial learning rates of 0.0001.

4.1. Experiment 1: ShapeNet

Training data. We used "Car", "Chair", "Airplane", and "Vessel" of ShapeNet to create training data. We randomly split all 3D models of each category into 80% training and 20% test instances. We rendered each model using Blender (https://www.blender.org) using cameras at azimuths of $0°$ to $355°$ with $5°$ steps and elevations of $0°$ to $30°$ with $10°$ steps with the fixed distance to objects. We finally cropped object regions using the same central squares for all viewpoints and resized them to 224×224.

We created training triplets $\{\mathcal{I}_1, \mathcal{R}_{\text{GT}}, \mathcal{I}_2\}$ where $\mathcal{I}_1$, $\mathcal{R}_{\text{GT}}$, and $\mathcal{I}_2$ have the same elevations. Let ϕ_1, ϕ_2, and ϕ_{GT} denote azimuths of $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{R}_{\text{GT}}$. We first sampled $\mathcal{I}_1$ with ϕ_1 multiples of $10°$, and sampled $\mathcal{I}_2$ to satisfy $\Delta\phi = \phi_2 - \phi_1 = \{20°, 30°, 40°, 50°\}$. $\mathcal{R}_{\text{GT}}$ is then selected to satisfy $\phi_{\text{GT}} - \phi_1 = \phi_2 - \phi_{\text{GT}} = \{10°, 15°, 20°, 25°\}$. We provided VSAF with 8D one-hot vectors [36] to represent viewpoint transformations from $\mathcal{I}_1$ to $\mathcal{R}_{\text{GT}}$ and from $\mathcal{I}_2$ to $\mathcal{R}_{\text{GT}}$ equivalent to azimuth differences of $\{\pm 10°, \pm 15°, \pm 20°, \pm 25°\}$. The number of training triplets for "Car", "Chair", "Airplane", and "Vessel" are about 3.4M, 3.1M, 1.9M, and 0.9M, respectively. More details of the ShapeNet training data are shown in Appendix C of the arXiv version of the paper [17].

Category-specific training. We first show view synthesis results of DVM and VSAF trained on each category separately. Both DVM and VSAF were trained using exactly the same training data. For evaluating the view synthesis results, we randomly sampled 200,000 test triplets for each category created with the same configuration as that of the training triplets. As an error metric, we use the mean squared error (MSE) between the synthesized output and ground truth summed over all pixels.

Figure 7 shows qualitative comparisons of view synthesis results by DVM and VSAF. It is clear that view synthesis results by DVM are visually more pleasing with much less ghosting artifacts and much closer to the ground truth views than those by VSAF. Table 1 shows the mean of MSE by DVM and VSAF for each category. The mean of MSE by DVM are considerably smaller than those by

Table 1. Mean of MSE by DVM and VSAF trained for "Car", "Chair", "Airplane", and "Vessel" in a category-specific way.

	Car	Chair	Airplane	Vessel
DVM	**44.70**	**61.00**	**22.30**	**42.74**
VSAF	70.11	140.35	46.80	95.99

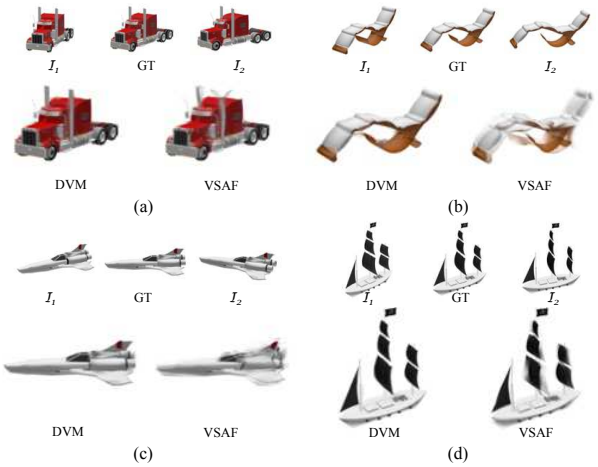

Figure 7. Comparisons of view synthesis results by DVM and VSAF on test samples of (a) "Car", (b) "Chair", (c) "Airplane", and (d) "Vessel" of ShapeNet. Two input images are shown on the left and right sides of the ground truth image ("GT"). More comparisons are shown in Appendix C of the arXiv version of the paper [17].

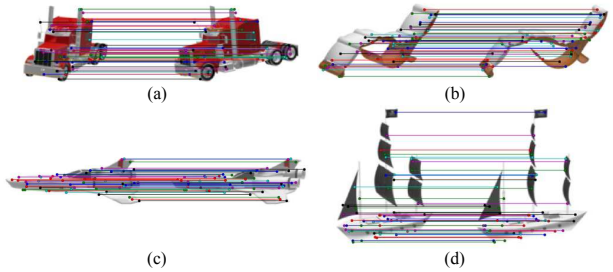

Figure 8. Examples of rectification results and dense correspondences obtained by DVM on the test input images shown in Fig. 7. More examples are shown in Appendix C of the arXiv version of the paper [17].

VSAF for all four categories, which matches well the qualitative comparisons in Fig. 7 . The mean of MSE by DVM for "Car", "Chair", "Airplane", and "Vessel" are 63.8%, 43.5%, 47.6%, and 44.5% of that by VSAF, respectively.

Figure 8 shows the rectification results and dense correspondences obtained by DVM for the test input images shown in Fig. 7. Note that DVM yields highly accurate rectification results and dense correspondence results. In fact, it is not possible to synthesize the middle view accurately if one of them is incorrect. The quantitative analysis of the rectification accuracy by DVM is shown in Appendix C of the arXiv version of the paper [17].

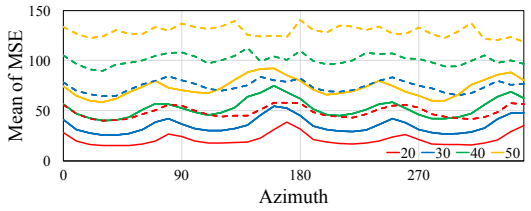

Figure 9. Plots of mean of MSE by DVM (solid) and VSAF (dashed) as a function of ϕ_1 (azimuth of $\mathcal{I}_1$) for all test triplets of "Car", "Chair", "Airplane", and "Vessel". Different line colors represent different azimuth differences $\Delta\phi$ between $\mathcal{I}_1$ and $\mathcal{I}_2$.

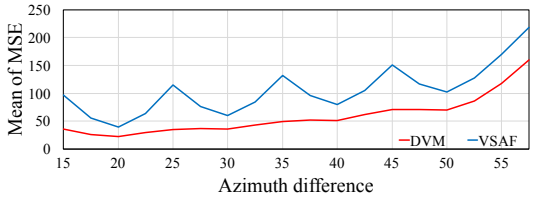

Figure 10. Plots of mean of MSE by DVM (red) and VSAF (blue) as a function of azimuth difference $\Delta\phi$ between $\mathcal{I}_1$ and $\mathcal{I}_2$ for "Car". Here, the azimuth differences are $15^\circ \leq \Delta\phi < 60^\circ$ with 2.5° steps.

Figure 9 shows plots of mean of MSE by DVM and VSAF as a function of ϕ_1 (azimuth of $\mathcal{I}_1$) where different line colors represent different azimuth differences $\Delta\phi$ between $\mathcal{I}_1$ and $\mathcal{I}_2$. As expected, the mean of MSE increases as $\Delta\phi$ increases. Note that the mean of MSE by DVM for $\Delta\phi = 50^\circ$ is similar to that by VSAF for $\Delta\phi = 30^\circ$. Also note that the mean of MSE by DVM for each $\Delta\phi$ has peaks near $\phi_1 = 90^\circ \cdot i - \Delta\phi/2, i = 0, 1, 2, 3$, where there is considerable visibility changes between $\mathcal{I}_1$ and $\mathcal{I}_2$, *e.g.*, from a right-front view $\mathcal{I}_1$ to a left-front view $\mathcal{I}_2$.

We also compare the performance of DVM and VSAF trained for the larger azimuth differences up to 90°. Due to the limited space, the results are shown in Appendix C of the arXiv version of the paper [17].

Robustness test. We now test the robustness of DVM and VSAF to inputs that have different azimuths and elevations from those of the training data. We newly created 200,000 test triplets of "Car" with azimuths and elevations that are 5° shifted from those of the training triplets but still with $\Delta\phi = \{20^\circ, 30^\circ, 40^\circ, 50^\circ\}$. The mean of MSE for the 5° shifted test triplets by DVM and VSAF are 71.75 and 107.64, respectively. Compared to the mean of MSE by DVM and VSAF on the original test triplets of "Car" in Tab. 1, both DVM and VSAF performed worse similarly: 61% MSE increase by DVM and 54% MSE increase by VSAF. However, note that the mean of MSE by DVM on the 5° shifted test triplets (71.75) is similar to that by VSAF on the original test triplets (70.11).

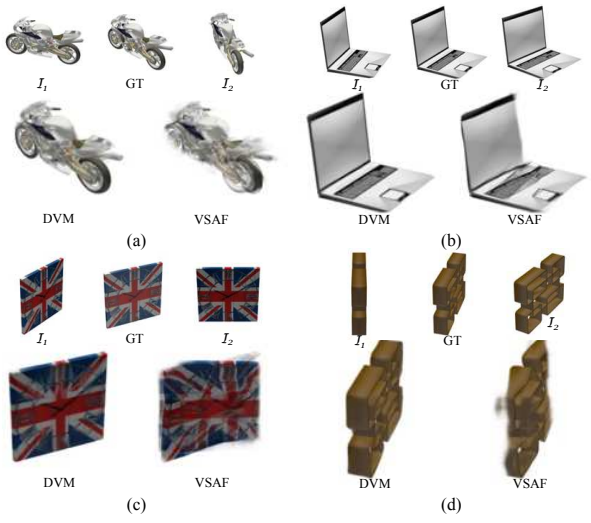

Figure 11. Comparisons of view synthesis results by DVM and VSAF on test samples of unseen (a) "Motorcycle", (b) "Laptop", (c) "Clock", and (d) "Bookshelf" of ShapeNet. More comparisons are shown in Appendix C of the arXiv version of the paper [17].

We also test the robustness of DVM and VSAF to inputs with azimuth differences $\Delta\phi$ that are different from those of the training data. We newly created 500,000 test triplets of "Car" with $\mathcal{I}_1$ that are the same as the training triplets and $\mathcal{I}_2$ and $\mathcal{R}_{\text{GT}}$ corresponding to $15^\circ \leq \Delta\phi < 60^\circ$ with 2.5° steps. We provided VSAF with 8D one-hot vectors by finding the elements from $\{\pm 10^\circ, \pm 15^\circ, \pm 20^\circ, \pm 25^\circ\}$ closest to $\phi_1 - \phi_{\text{GT}}$ and $\phi_2 - \phi_{\text{GT}}$.

Figure 10 shows plots of mean of MSE by DVM and VSAF for the new 500,000 test triplets of "Car". It is clear that DVM is much more robust to the unseen $\Delta\phi$ than VSAF. VSAF yielded much higher MSE for the unseen $\Delta\phi$ compared to that for $\Delta\phi$ multiples of 10. Contrarily, the MSE increase by DVM for such unseen $\Delta\phi$ is minimal except for $\Delta\phi > 50^\circ$. This result suggests that DVM which directly considers two input images together for synthesis without relying on the viewpoint transformation inputs has more generalizability than VSAF.

Category-agnostic training. We now show view synthesis results of DVM and VSAF trained in a category-agnostic way, *i.e.*, we trained DVM and VSAF using all training triplets of all four categories altogether. For this category-agnostic training, we limited the maximum number of training triplets of each category to 1M. For testing, we additionally selected four unseen categories from ShapeNet: "Motorcycle", "Laptop", "Clock", and "Bookshelf". The test triplets of the unseen categories were created with the same configuration as that of the training triplets.

Figure 11 shows qualitative comparisons of view synthesis results by DVM and VSAF on the unseen categories. We can see the view synthesis results by DVM are still highly accurate even for the unseen categories. Especially, DVM even can predict the blending masks correctly as shown in Fig. 11(d). Contrarily, VSAF yielded view synthesis results with lots of ghosting artifacts and severe shape distortions.

Table 2. Mean of MSE by DVM and VSAF trained for "Car", "Chair", "Airplane", and "Vessel" in a category-agnostic way.

	Car	Chair	Airplane	Vessel
DVM	**52.56**	**73.01**	**24.73**	**38.42**
VSAF	83.36	161.59	51.95	88.47
	Motorcycle	Laptop	Clock	Bookshelf
DVM	**154.45**	**102.27**	**214.02**	**171.81**
VSAF	469.01	262.33	491.82	520.22

Table 2 shows the mean of MSE by DVM and VSAF trained in a category-agnostic way. Compared to Tab. 1, we can see the mean of MSE by both DVM and VSAF for "Car", "Chair", and "Airplane" slightly increased due to less training samples of the corresponding categories. Contrarily, the mean of MSE by both DVM and VSAF for "Vessel" decreased mainly due to the training samples of the other categories. The performance difference between DVM and VSAF for the unseen categories is much greater than that for the seen categories. The mean of MSE by DVM for "Motorcycle", "Laptop", "Clock", and "Bookshelf" are 32.9%, 39.0%, 43.5%, and 33.0% of that by VSAF, respectively. These promising results by DVM on the unseen categories suggest that DVM can learn general features necessary for rectifying image pairs and establishing correspondences between them. The quantitative analysis of the rectification accuracy by DVM for the unseen categories is shown in Appendix C of the arXiv version of the paper [17].

4.2. Experiment 2: Multi-PIE

Training data. Multi-PIE dataset [10] contains face images of 337 subjects captured at 13 viewpoints from 0° to 180° azimuth angles. We split 337 subjects into 270 training and 67 test subjects. We used 11 viewpoints from 15° to 165° because images at 0° and 180° have drastically different color characteristics. We sampled $\mathcal{I}_1$ and $\mathcal{I}_2$ to have $\Delta\phi = \{30^\circ, 60^\circ\}$, and picked $\mathcal{R}_{\text{GT}}$ to satisfy $\phi_{\text{GT}} - \phi_1 = \phi_2 - \phi_{\text{GT}} = \{15^\circ, 30^\circ\}$. The number of training triplets constructed in this way is 643,760. We provided VSAF with 4D one-hot vectors accordingly.

Multi-PIE provides detailed facial landmarks annotations but only for subsets of whole images. Using those annotations, we created two sets of training data with (i) loose and (ii) tight facial region crops. For the loose crops, we used a single bounding box for all images of the same viewpoint that encloses all facial landmarks of those images. For the tight crops, we first performed face

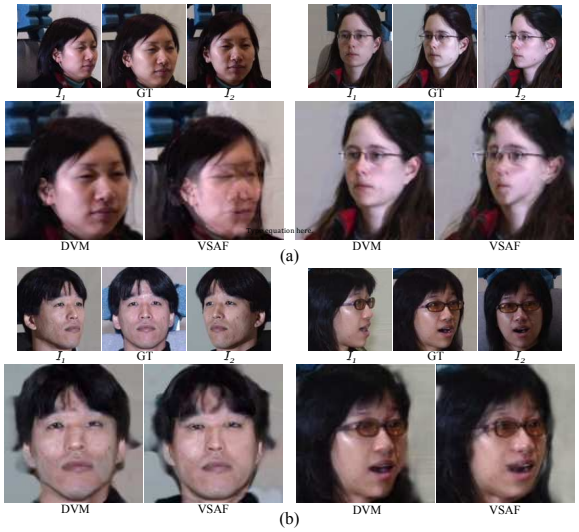

Figure 12. Comparisons of view synthesis results by DVM and VSAF on Multi-PIE test samples with (a) loose and (b) tight crops. More comparisons are shown in Appendix C of the arXiv version of the paper [17].

Table 3. Mean of MSE by DVM and VSAF for Multi-PIE test triplets with the loose and tight facial region crops.

	Loose facial region crops	Tight facial region crops
DVM	**162.62**	**164.77**
VSAF	267.83	194.30

segmentation using FCN [24] trained with convex hull masks of facial landmarks. We then used a bounding box of the segmented region for each image. For both cases, we extended bounding boxes to be square and include all facial regions and finally resized them to 224×224.

Results. We trained DVM and VSAF using each of the two training sets separately. For testing, we created two sets of 157,120 test triplets from 67 test subjects, one with the loose crops and the other with the tight crops, with the same configuration as that of the training sets.

Figure 12(a) shows qualitative comparisons of view synthesis results by DVM and VSAF on the test triplets with the loose facial region crops. The view synthesis results by VSAF suffer lots of ghosting artifacts and severe shape distortions mainly because (i) faces are not aligned well and (ii) their scales can be different. Contrarily, DVM yielded very satisfactory view synthesis results by successfully dealing with the unaligned faces and scale differences thanks to the presence of the rectification network. These successful view synthesis results by DVM have significance in that DVM can synthesize novel views quite well even with the the camera setup not as precise as that of the ShapeNet rendering and objects with different scales.

Figure 12(b) shows qualitative comparisons of view synthesis results by DVM and VSAF on the test triplets with the tight facial region crops. The view synthesis results by VSAF are much improved compared to the case of the loose facial region crops because the facial regions are aligned fairly well and their scale differences are negligible. However, the view synthesis results by DVM are still better than those by VSAF with less ghosting artifacts and less shape distortions. Table 3 shows the mean of MSE by DVM and VSAF for the Multi-PIE test triplets that match well the qualitative comparisons in Fig. 12.

4.3. Experiment 3: Intermediate view synthesis

We can synthesize all intermediate views by linearly interpolating the blending masks $\mathcal{M}_1$ and $\mathcal{M}_2$ as well as $\mathcal{R}_1$ and $\mathcal{R}_2$. As the dense correspondences predicted by DVM are highly accurate, we can synthesize highly realistic intermediate views. The detailed procedure to synthesize the intermediate views and results are shown in Appendix C of the arXiv version of the paper [17].

5. Conclusion and discussion

In this paper, we proposed DVM, a CNN-based view synthesis method inspired by View Morphing [27]. Two input images are first automatically rectified by the rectification network. The encoder-decoder network then outputs the dense correspondences between the rectified images and blending masks to predict the visibility of pixels of the rectified images in the middle view. Finally, the view morphing network synthesizes the middle view using the dense correspondences and blending masks. We experimentally showed that DVM can synthesize novel views with detailed textures and well-preserved geometric shapes clearly better than those by the CNN-based state-of-the-art.

Deep View Morphing still can be improved in some aspects. For example, it is generally difficult for Deep View Morphing to deal with very complex thin structures. Plus, the current blending masks cannot properly deal with the different illumination and color characteristics between input images, and thus blending seams can be visible in some cases. Examples of these challenging cases for DVM are shown in Appendix C. Future work will be focused on improving the performance for such cases.

Acknowledgment

We would like to thank Tinghui Zhou for kindly sharing codes of "View Synthesis by Appearance Flow" [36] and helpful comments with us.

References

[1] T. Beier and S. Neely. Feature-based image metamorphosis. In *Proc. SIGGRAPH*, 1992. 1, 2

[2] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xia, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. Technical report, arXiv:1512.03012, 2015. 2, 5

[3] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH*, 1993. 2

[4] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 3, 4

[5] A. Dosovitskiy, J. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proc. CVPR*, 2015. 2, 4

[6] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proc. ICCV*, 2015. 1, 2

[7] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010. 1, 2

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, 2010. 5

[9] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proc. SIGGRAPH*, 1996. 1

[10] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 2010. 2, 5, 7

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 2

[12] J. Heinly, J. Schonberger, E. Dunn, and J.-M. Frahm. Reconstructing the world* in six days *(as captured by the yahoo 100 million image dataset). In *Proc. CVPR*, 2015. 1, 2

[13] G. Hinton, A. Krizhevsky, and S. Wang. Transforming auto-encoders. In *Proc. ICANN*, 2011. 2

[14] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. *ACM transactions on graphics*, 2005. 1, 2

[15] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proc. 24th annual conference on Computer graphics and interactive techniques*, 1997. 1, 2

[16] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proc. NIPS*, 2015. 2

[17] D. Ji, J. Kwon, M. McFarland, and S. Savarese. Deep view morphing. Technical report, arXiv:1703.02168, 2017. 3, 5, 6, 7, 8

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. Technical report, arXiv:1408.5093, 2014. 5

[19] M. Jones and T. Poggio. Model-based matching of line drawings by linear combination of prototypes. In *Proc. ICCV*, 1995. 1, 2

[20] A. Katayama, K. Tanaka, T. Oshino, and H. Tamura. A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images. *Stereoscopic Displays and Virutal Reality Systems*, 1995. 1

[21] D. Kingma, J. Ba, and G. Gamow. Adam: A method for stochastic optimization. Technical report, arXiv:1412.6980, 2014. 5

[22] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH*, 1996. 1

[23] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proc. CVPR*, 2017. 2

[24] J. Long, E. Shelhamer, and T. Darrel. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015. 8

[25] B. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proc. 28th annual conference on Computer graphics and interactive techniques*, 2001. 1, 2

[26] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proc. CVPR*, 2017. 2

[27] S. Seitz and C. Dyer. View morphing. In *Proc. SIGGRAPH*, 1996. 1, 2, 5, 8

[28] A. Shashua. Algebraic functions for recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1995. 1, 2

[29] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *Proc. ECCV*, 2016. 1, 2, 4

[30] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1997. 1, 2

[31] C. Wu. Towards linear-time incremental structure from motion. In *Proc. 3DV*, 2013. 1, 2

[32] J. Yang, S. Reed, M. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Proc. NIPS*, 2015. 1, 2, 4

[33] L. Zhang, G. Dugas-Phocion, J. Samson, and S. Seitz. Single-view modelling of free-form scenes. *The Journal of Visualization and Computer Animation*, 2002. 1, 2

[34] E. Zheng, E. Dunn, V. Jojic, and J.-M. Frahm. Patchmatch based joint view selection and depthmap estimation. In *Proc. CVPR*, 2014. 2

[35] E. Zheng and C. Wu. Structure from motion using structure-less resection. In *Proc. ICCV*, 2015. 2

[36] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *Proc. ECCV*, 2016. 1, 2, 4, 5, 8