Learning to Dodge A Bullet: Concyclic View Morphing via Deep Learning

Shi Jin[1, 3⋆] Ruiynag Liu[1, 3⋆] Yu Ji[2] Jinwei Ye[3] Jingyi Yu[1, 2]

[1] ShanghaiTech University, Shanghai, China
[2] Plex-VR, Baton Rouge, LA, USA
[3] Louisiana State University, Baton Rouge, LA, USA

Abstract. The bullet-time effect, presented in feature film "The Matrix", has been widely adopted in feature films and TV commercials to create an amazing stopping-time illusion. Producing such visual effects, however, typically requires using a large number of cameras/images surrounding the subject. In this paper, we present a learning-based solution that is capable of producing the bullet-time effect from only a small set of images. Specifically, we present a view morphing framework that can synthesize smooth and realistic transitions along *a circular view path* using as few as three reference images. We apply a novel cyclic rectification technique to align the reference images onto a common circle and then feed the rectified results into a deep network to predict its motion field and per-pixel visibility for new view interpolation. Comprehensive experiments on synthetic and real data show that our new framework outperforms the state-of-the-art and provides an inexpensive and practical solution for producing the bullet-time effects.

Keywords: bullet-time effect, image-based rendering, view morphing, convolutional neural network (CNN)

1 Introduction

Visual effects have now become an integral part of film and television productions as they provide unique viewing experiences. One of the most famous examples is the "bullet-time" effect presented in feature film The Matrix. It creates the stopping-time illusion with smooth transitions of viewpoints surrounding the actor. To produce this effect, over 160 cameras were synchronized and precisely arranged: they are aligned on a track through a laser targeting system, forming a complex curve through space. Such specialized acquisition systems, however, are expensive and require tremendous efforts to construct.

Creating the bullet-time effects has been made more flexible by using image-based rendering techniques. Classic methods rely on geometric information (*e.g.*, visual hulls [1], depth maps [2], and optical flow [3, 4]) to interpolate novel perspectives from sampled views. Latest approaches can handle fewer number of

⋆ This work was performed when Shi and Ruiyang were visiting students at LSU.

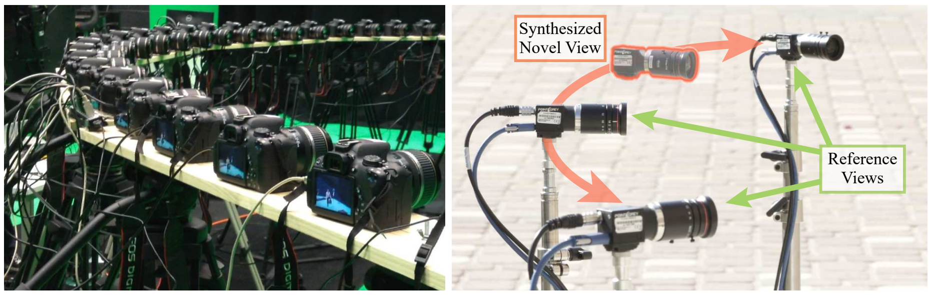

Fig. 1. Left: Specialized acquisition system with numerous cameras is often needed for producing the bullet-time effect; Right: We propose to morph transition images on a circular path from a sparse set of view samples for rendering such effect.

images but still generally require large overlap between the neighboring views to ensure reliable 3D reconstruction and then view interpolation. In image-based modeling, view morphing has been adopted for synthesizing smooth transitions under strong viewpoint variations. The seminal work of Seitz and Dyer [5] shows that shape-preserving morphing can be achieved by linearly interpolating corresponding pixels in two rectified images. Most recently, deep learning based techniques such as deep view morphing (DVM) [6] provides a more generic scheme by exploiting redundant patterns in the training data. By far, state-of-the-art methods unanimously assume linear camera paths and have not shown success in creating the 360° effects such as the bullet-time.

In this paper, we present a novel learning-based solution that is capable of producing the bullet-time effect from only a small set of images. Specifically, we design a view morphing framework that can synthesize smooth and realistic transitions along *a circular view path* using as few as three reference images (as shown in Fig. 1). We apply a novel cyclic rectification technique to align the reference images onto a common circle. Cyclic rectification allows us to rectify groups of three images with minimal projective distortions. We then feed the rectified results into a novel deep network for novel view synthesis. Our network consists of an encoder-decoder network for predicting the motion fields and visibility masks as well as a blending network for image interpolation. By using a third intermediate image, our network can reliably handle occlusions and large view angle changes (up to 120°).

We perform comprehensive experiments on synthetic and real data to validate our approach. We show that our framework outperforms the state-of-the-arts [6–8] in both visual quality and errors. For synthetic experiments, we test on the SURREAL [9] and ShapeNet datasets [10] and demonstrate the benefits of our technique on producing 360° rendering of dynamic human models and complex 3D objects. As shown in Fig. 1, we set up a three-camera system to capture real 3D human motions and demonstrate high quality novel view reconstruction. Our morphed view sequence can be used for generating the bullet-time effect.

2 Related Work

Image-based Rendering. Our work belongs to image-based rendering (IBR) that generates novel views directly from input images. The most notable techniques are light field rendering [11] and Lumigraph [12]. Light field rendering synthesizes novel views by filtering and interpolating view samples while lumigraph applies coarse geometry to compensate for non-uniform sampling. More recently, Penner *et al.* [13] utilizes a soft 3D reconstruction to improve the quality of view synthesis from a light field input. Rematas *et al.* [14] aligns the proxy model and the appearance with user interaction. IBR techniques have been widely for rendering various space-time visual effects [4, 15], such as the freeze-frame effect. Carranza *et al.* [1] uses a multi-view system to produce free-viewpoint videos. They recover 3D models from silhouettes for synthesizing novel views from arbitrary perspectives. Zitnick *et al.* [2] use depth maps estimated from multi-view stereo to guide viewpoint interpolation. Ballan *et al.* [16] synthesize novel views from images captured by a group of un-structured cameras and they use structure-from-motion for dense 3D reconstruction. All these methods rely on either explicit or implicit geometric proxy (*e.g.*, 3D models or depth maps) for novel view synthesis. Therefore, a large number of input images are needed to infer reliable geometry of the scene/object. Our approach aims at synthesizing high-quality novel views using only three images without estimating the geometry. This is enabled by using a deep convolutional network that encodes the geometric information from input images into feature tensors.

Image Morphing. The class of IBR technique that is most close to our work is image morphing, which reconstructs smooth transitions between two input images. The key idea is to establish dense correspondences for interpolating colors from the source images. Earlier works study morphing between arbitrary objects using feature correspondences [3, 17–19]. While our work focuses on generating realistic natural transitions between different views of the same object. The seminal work of Seitz and Dyer [5] shows that such shape-preserving morphing can be achieved by linear interpolation of corresponding pixels in two rectified images. The morphing follows the linear path between the two original optical centers. To obtain dense correspondences, either stereo matching [4, 20] or optical flow [15] can be used, depending on whether the cameras are pre-calibrated. Drastic viewpoint change and occlusions often downgrade the morphing quality by introducing ghosting artifacts. Some methods adopt auxiliary geometry such as silhouettes [21] and triangulated surfaces [22] to alleviate this problem. Mahajan *et al.* [23] propose a path-based image interpolation framework that operates in the gradient domain to reduce blurry and ghosting artifacts. Our approach morphs intermediate views along a circular path and by using a third intermediate image in the middle, we can handle occlusions well without using geometry.

CNN-based Image Synthesis. In recent years, convolutional neural networks (CNNs) have been successfully applied on various image synthesis tasks. Dosovitskiy *et al.* [24] propose a generative CNN to synthesize models given existing instances. Tatarchenko *et al.* [25] use CNN to generate arbitrary perspectives of an object from one image and recover the object's 3D model using the synthesized views. Niklause *et al.* [26, 27] apply CNN to interpolate video frames. These methods use CNN to directly predict pixel colors from scratch and often suffer from blurriness and distortions. Jaderberg *et al.* [28] propose to insert differentiable layers to CNN in order to explicitly perform geometric transformations on images. This design allows CNN to exploit geometric cues (*e.g.*, depths, optical flow, epipolar geometry, *etc.*) for view synthesis. Flynn *et al.* [29] blend CNN-predicted images at different depth layers to generate new views. Kalantari *et al.* [30] apply CNN on light field view synthesis. Zhou *et al.* [8] estimate appearance flow by CNN and use it to synthesize new perspectives of the input image. Park *et al.* [7] propose to estimate the flow only in visible areas and then complete the rest by an adversarial image completion network. Most recently, Ji *et al.* [6] propose the deep view morphing (DVM) network that generalizes the classic view morphing scheme [5] to a learning model. This work is closely related to ours since we apply CNN on similar morphing task. However, there are a few key differences: 1) instead of synthesizing one middle view, our approach generates a sequence of morphed images using the motion field; 2) by using a third intermediate image, we can better handle occlusions and large view angle changes (up to 120°); and 3) our morphed view sequence can be considered as taken along a circular camera path that is suitable for rendering freeze-frame effect.

3 Cyclic Rectification

Stereo rectification reduces the search space for correspondence matching to 1D horizontal scan lines and the rectified images can be viewed as taken by two parallel-viewing cameras. It is usually the first step in view morphing algorithms since establishing correspondences is important for interpolating intermediate views. However, such rectification scheme is not optimal for our three-view circular-path morphing: 1) the three images need to be rectified in pairs instead of as a whole group and 2) large projective distortion may appear in boundaries of the rectified images if the three cameras are configured on a circular path. We therefore propose a novel *cyclic rectification* scheme that warps three images to face towards the center of a common circle. Since three non-colinear points are cyclic, we can always fit a circumscribed circle given the center-of-projection (CoP) of the three images. By applying our cyclic rectification, correspondence matching is also constrained to 1D lines in the rectified images. Although the scan lines are not horizontal, they can be easily determined by pixel locations. In Sec. 4.3, we impose the scan line constraints onto the network training to improve matching accuracy.

Given three reference images $\{\mathcal{I}_l, \mathcal{I}_m, \mathcal{I}_r\}$ and their camera calibration parameters $\{K_i, R_i, t_i | i = l, m, r\}$ (where K_i is intrinsic matrix, R_i and t_i are extrinsic rotation and translation, subscripts l, m, and r stands for "left", "middle", and "right"), to perform cyclic rectification, we first fit the center of circumscribed

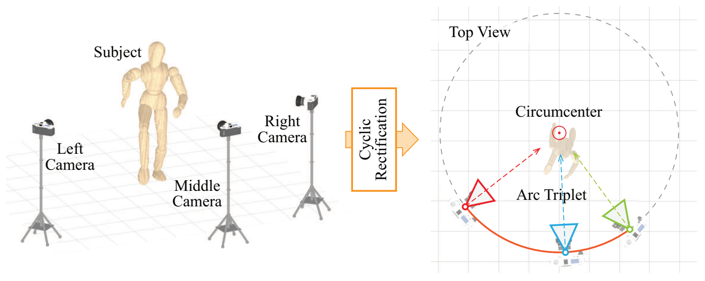

Fig. 2. Cyclic rectification. We configure three cameras along a circular path for capturing the reference images. After cyclic rectification, the reference images are aligned on a common circle (*i.e.*, their optical principal axes all pass through the circumcenter) and we call them the *arc triplet*.

circle (i.e., the circumcenter) using the cameras' CoPs and then construct homographies for warping the three images. Fig. 2 illustrates this scheme.

Circumcenter Fitting. Let's consider the triangle formed by the three CoPs. The circumcenter of the triangle can be constructed as the intersection point of the edges' perpendicular bisectors. Since the three cameras are calibrated in a common world coordinate, the extrinsic translation vectors $\{t_i|i = l, m, r\}$ are essentially the CoP coordinates. Thus $\{t_i - t_j|i, j = l, r, m; i \neq j\}$ are the edges of the triangle. We first solve the normal $\mathbf{n}$ of the circle plane from

$$\mathbf{n} \cdot (t_i - t_j) = 0 \tag{1}$$

Then the normalized perpendicular bisectors of the edges can be computed as

$$\mathbf{d}_{ij} = \frac{\mathbf{n} \times (t_i - t_j)}{\|t_i - t_j\|} \tag{2}$$

We determine the circumcenter O by triangulating the three perpendicular bisectors $\{\mathbf{d}_{ij}|i, j = l, r, m; i \neq j\}$

$$O = \frac{1}{2}(t_i + t_j) + \alpha_{ij}\mathbf{d}_{ij} \tag{3}$$

where $\{\alpha_{ij}|i, j = l, r, m; i \neq j\}$ are propagation factors along $\mathbf{d}_{ij}$. Since Eq. 3 is an over-determined linear system, O can be easily solved by SVD.

Homographic Warping. Next, we derive the homographies $\{H_i|i = l, r, m\}$ for warping the three reference images $\{\mathcal{I}_l, \mathcal{I}_m, \mathcal{I}_r\}$ such that the rectified images all face towards the circumcenter O. In particular, we transform the camera coordinate in a two-step rotation: we first rotate the y axis to align with the circle plane normal $\mathbf{n}$ and then rotate the z axis to point to the circumcenter O. Given the original camera coordinates $\{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i|i = l, r, m\}$ as calibrated in the

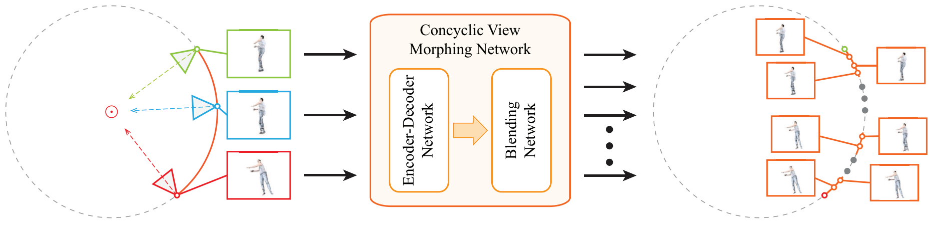

Fig. 3. The overall structure of our *Concyclic View Morphing Network* (CVMN). It takes the arc triplet as input and synthesize sequence of concyclic views.

extrinsic rotation matrix $R_i = [\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i]$, the camera coordinates after cyclic rectification can be calculated as

$$\begin{cases} \mathbf{x}'_i = \mathbf{y}'_i \times \mathbf{z}'_i \\ \mathbf{y}'_i = \mathrm{sgn}(\mathbf{n} \cdot \mathbf{y}_i) \cdot \mathbf{n} \\ \mathbf{z}'_i = \mathrm{sgn}(\mathbf{z}_i \cdot (O - t_i)) \cdot \pi(O - t_i) \end{cases} \tag{4}$$

where $i = r, m, l$; $\mathrm{sgn}(\cdot)$ is the sign function and $\pi(\cdot)$ is the normalization operator. We then formulate the new extrinsic rotation matrix as $R'_i = [\mathbf{x}'_i, \mathbf{y}'_i, \mathbf{z}'_i]$. As a result, the homographies for cyclic rectification can be constructed as $H_i = K_i R'^{\top}_i R_i K_i^{-1}$, $i = r, m, l$.

Finally, we use $\{H_i | i = l, r, m\}$ to warp $\{\mathcal{I}_l, \mathcal{I}_m, \mathcal{I}_r\}$ and the resulting cyclic rectified images $\{\mathcal{C}_l, \mathcal{C}_m, \mathcal{C}_r\}$ are called *arc triplet*.

4 Concyclic View Morphing Network

We design a novel convolutional network that takes the arc triplet as input to synthesize a sequence of evenly distributed concyclic morphing views. We call this network the *Concyclic View Morphing Network* (CVMN). The synthesized images can be viewed as taken along a circular camera path since their CoPs are concyclic. The overall structure of our CVMN is shown in Fig. 3. It consists of two sub-networks: an encoder-decoder network for estimating the motion fields $\{\mathcal{F}_i | i = 1, ..., N\}$ and visibility masks $\{\mathcal{M}_i | i = 1, ..., N\}$ of the morphing views given $\{\mathcal{C}_l, \mathcal{C}_m, \mathcal{C}_r\}$ and a blending network for synthesizing the concyclic view sequence $\{\mathcal{C}_i | i = 1, ..., N\}$ from $\{\mathcal{F}_i | i = 1, ..., N\}$ and $\{\mathcal{M}_i | i = 1, ..., N\}$. Here N represents the total number of images in the output morphing sequence.

4.1 Encoder-Decoder Network

The encoder-decoder network has proved to be effective in establishing pixel correspondences in various applications [31,32]. We therefore adopt this structure for predicting pixel-based motion vectors for morphing intermediate views. In our network, we first use an encoder to extract correlating features among the arc triplet. We then use a two-branch decoder to estimate 1) motion vectors

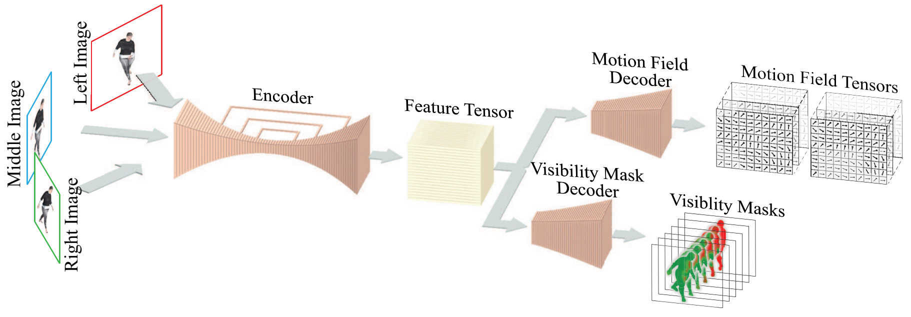

Fig. 4. The encoder-decoder network of CVMN.

and 2) visibility masks with respect to the left and right reference views. Our encoder-decoder network architecture is illustrated in Fig. 4.

Encoder. We adopt the hourglass structure [32] for our encoder in order to capture features from different scales. The balanced bottom-up (from high-res to low-res) and top-down (from low-res to high-res) structure enables pixel-based predictions in our decoders. Our hourglass layer setup is similar to [32]. The encoder outputs a full-resolution feature tensor.

Since our input has three images from the arc triplet, we apply the hourglass encoder in three separate passes (one per image) and then concatenate the output feature tensors. Although it is also possible to first concatenate the three input images and then run the encoder in one pass, such scheme results in high-dimensional input and is computationally impractical for the training process.

Motion Field Decoder. The motion field decoder takes the output feature tensor from the encoder and predicts motion fields for each image in the morphing sequence. Specifically, two motion fields are considered: one w.r.t the left reference image $\mathcal{C}_l$ and the other w.r.t. the right reference image $\mathcal{C}_r$. We use the displacement vector between corresponding pixels to represent the motion field and we use backward mapping (from source $\mathcal{C}_i$ to target $\mathcal{C}_l$ or $\mathcal{C}_r$) for computing the displacement vectors in order to reduce artifacts caused by irregular sampling.

Take $\mathcal{C}_l$ for example and let's consider an intermediate image $\mathcal{C}_i$. Given a pair of corresponding pixels $p_l = (x_l, y_l)$ in $\mathcal{C}_l$ and $p_i = (x_i, y_i)$ in $\mathcal{C}_i$, the displacement vector $\Delta_i^l(p) = (u_i^l(p), v_i^l(p))$ from p_i to p_l can be computed by

$$p_l = p_i + \Delta_i^l(p) \tag{5}$$

The right image based displacement vectors $\{\Delta_i^r(p) = (u_i^r(p), v_i^r(p)) | p = 1, ..., M\}$ (where M is the image resolution) can be computed similarly. By concatenating $\Delta_i^l(p)$ and $\Delta_i^r(p)$, we obtain a 4D motion vector $(u_i^l(p), v_i^l(p), u_i^r(p), v_i^r(p))$

for each pixel p. As a result, the motion field for the entire morphing sequence is composed of four scalar fields: $\mathcal{F} = (U^l, V^l, U^r, V^r)$, where $U^l = \{u_i^l | i = 1, ..., N\}$; V^l, U^r, and V^r follow similar construction.

Structure-wise, we arrange deconvolution and convolution layers alternately to extract motion vectors from the encoded correspondence features. The reason for this intervening layer design is because we found by experiments that appending proper convolution layer after deconvolution can reduce blocky artifacts in our output images. Since our motion field $\mathcal{F}$ has four components (U^l, V^l, U^r, and V^r), we run four instances of the decoder to predict each component in a separate pass. It is worth noting that by encoding features from the middle reference image $\mathcal{C}_m$, the accuracy of motion field estimation is greatly improved.

Visibility Mask Decoder. Large viewpoint change and occlusions cause the visibility issue in view morphing problems: pixels in an intermediate view are partially visible in both left and right reference images. Direct combining the resampled reference images results in severe ghosting artifacts. Similar to [6, 8], we use visibility masks to mitigate this problem.

Given an intermediate image $\mathcal{C}_i$, we define two visibility masks $\mathcal{M}_i^l$ and $\mathcal{M}_i^r$ to indicate the per-pixel visibility levels w.r.t. to $\mathcal{C}_l$ and $\mathcal{C}_r$. The larger the value in the mask, the higher the possibility for a pixel to be seen in the reference images. However, instead following a probability model to restrict the mask values within $[0, 1]$, we relax this constraint and allow the masks to take any real numbers greater than zero. We empirically find out that this relaxation help our our network converge faster in the training process.

Similar to the motion field decoder, our visibility mask decoder is composed of intervening deconvolution/convolution layers and takes the feature tensor from the encoder as input. At the end of the decoder, we use a ReLU layer to constraint the output values to be greater than zero. Since our visibility masks $\mathcal{M}$ has two components ($\mathcal{M}^l$ and $\mathcal{M}^r$), we run two instances of the decoder to estimate each component in a separate pass.

4.2 Blending Network

Finally, we use a blending network to synthesize a sequence of concyclic views $\{\mathcal{C}_i | i = 1, ..., N\}$ from the left and right reference images $\mathcal{C}_l, \mathcal{C}_r$ and the decoder outputs $\{\mathcal{F}_i | i = 1, ..., N\}, \{\mathcal{M}_i | i = 1, ..., N\}$, where N is the total number of morphed images. Our network architecture is shown in Fig. 5.

We first adopt two sampling layers to resample pixels in $\mathcal{C}_l$ and $\mathcal{C}_r$ using the motion field $\mathcal{F} = (U^l, V^l, U^r, V^r)$. The resampled images can be computed by $\mathcal{R}(\mathcal{C}_{\{l,r\}}; U^{\{l,r\}}, V^{\{l,r\}})$, where $\mathcal{R}(\cdot)$ is an operator that shifts corresponding pixels in the source images according to a motion vector (see Eq. (5)). Then we blend the resampled left and right images weighted by the visibility masks $\mathcal{M} = (\mathcal{M}^l, \mathcal{M}^r)$. Notice that our decode relaxes the range constraint of the output masks, we therefore need to normalize the visibility masks: $\bar{\mathcal{M}}_i^l = \frac{\mathcal{M}_i^l}{(\mathcal{M}_i^l + \mathcal{M}_i^r)}$,

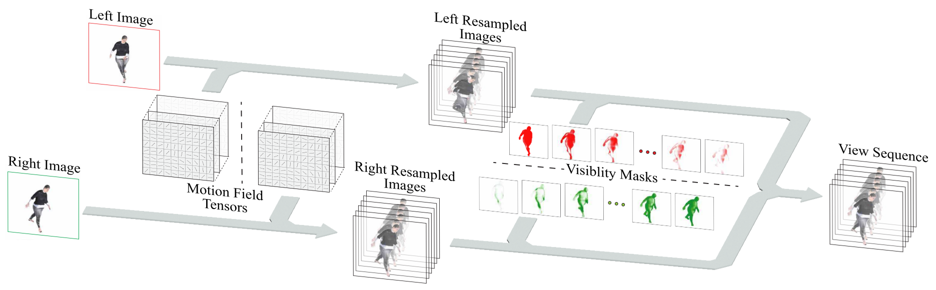

Fig. 5. The blending network of CVMN.

$\bar{\mathcal{M}}_i^r = \frac{\mathcal{M}_i^r}{(\mathcal{M}_i^l + \mathcal{M}_i^r)}$, where $i = 1, ...N$. The final output image sequence $\{\mathcal{C}_i | i = 1, ..., N\}$ can be computed by

$$\mathcal{C}_i = \mathcal{R}(\mathcal{C}_l; U_i^l, V_i^l) \otimes \bar{\mathcal{M}}_i^l + \mathcal{R}(\mathcal{C}_r; U_i^r, V_i^r) \otimes \bar{\mathcal{M}}_i^r \tag{6}$$

where $i = 1, ..., N$ and $\otimes$ is the pixel-wise multiplication operator.

Although all components in the blending network are fixed operations and do not have learnable weights, they are all differentiable layers [28] that can be chained into backpropagation.

4.3 Network Training

To guide the training process of our CVMN, we design a loss function that considers the following three metrics: 1) resemblance between the estimated novel views and the desired ground truth; 2) consistency between left-warped and right-warped images (since we consider motion fields in both directions); and 3) the epipolar line constraints in source images for motion field estimation. Assume Y is the underlying ground-truth view sequence and $\mathcal{R}^{\{l,r\}} = \mathcal{R}(\mathcal{C}_{\{l,r\}}; U^{\{l,r\}}, V^{\{l,r\}})$, our loss function can be written as

$$\mathcal{L} = \sum_{i=1}^{N} \|Y_i - \mathcal{C}_i\|_1 + \lambda \|(\mathcal{R}_i^l - \mathcal{R}_i^r) \otimes \bar{\mathcal{M}}_i^l \otimes \bar{\mathcal{M}}_i^r\|_2 + \gamma \Phi(\rho_i, p_i) \tag{7}$$

where λ, γ are hyper parameters for balancing the error terms; $\Phi(\cdot)$ is a function calculating the distance between a line and a point; p_i is a pixel in $\mathcal{C}_i$ warped by the motion field $\mathcal{F}_i$; and ρ is an epipolar line in source images. The detailed derivation of ρ from p_i can be found in the supplemental material.

5 Experiments

We perform comprehensive experiments on synthetic and real data to validate our approach. For synthetic experiments, we test on the SURREAL [9] and

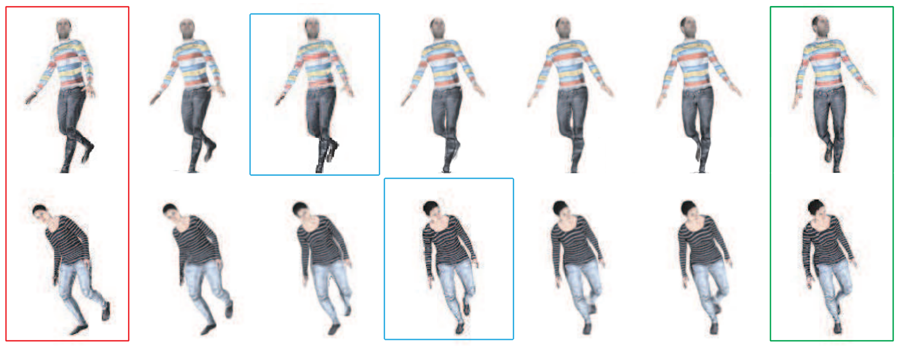

Fig. 6. Morphing sequences synthesized by CVMN. Due to space limit, we only pick seven samples from the whole sequence (24 images in total). The boxed images are the input reference views. More results can be found in the supplemental material.

ShapeNet datasets [10] and compare with the state-of-the-art methods DVM [6], TVSN [7] and VSAF [8]. Our approach outperforms these methods in both visual quality and quantitative errors. For real experiments, we set up a three-camera system to capture real 3D human motions and demonstrate high quality novel view reconstruction. Finally, we show a bullet-time rendering result using our morphed view sequence.

For training our CVMN, we use the Adam solver with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is 0.0001. We use the same settings for training the DVM. We run our network on a single Nvidia Titan X and choose a batch size of 8. We evaluate our approach on different images resolutions (up to 256). The architecture details of our CVMN, such as number of layers, kernel sizes, *etc.*, can be found in the supplemental material.

5.1 Experiments on SURREAL

Data Preparation. The SURREAL dataset [9] includes a large number of human motion sequences parametrized by SMPL [33]. Continuous motion frames are provided in each sequence. To generate the training and testing data for human motion, we first gather a set of 3D human models and textures. We export 30439 3D human models from 312 sequences. We select 929 texture images and randomly assign them to the 3D models. We then use the textured 3D models to render image sequences for training and testing. Specifically, we move our camera on a circular path and set it to look at the center of the circle for rendering concyclic views. For a motion sequence, we render images from 30 different elevation planes and on each plane we render a sequence of 24 images where the viewing angle change varies from 30° to 120° from the left-most image to the right-most image. In total, we generate around 1M motion sequences. We randomly pick one tenth of the data for testing and the rest are used for training.

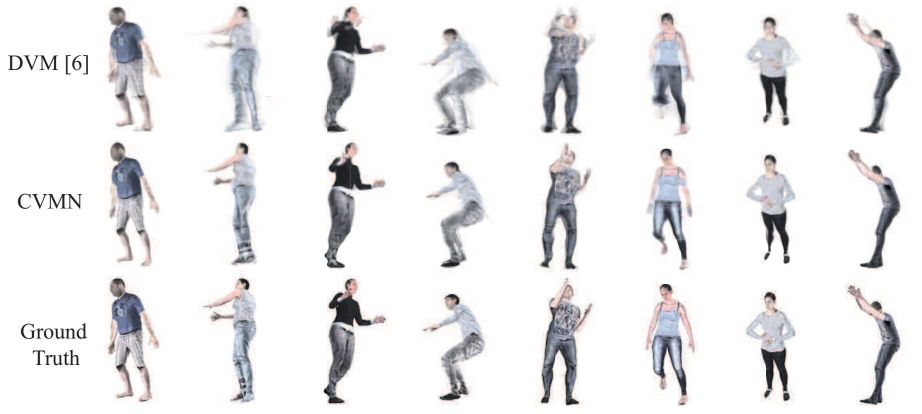

Fig. 7. Comparison with DVM. We pick the middle view in our synthesized sequence to compare with DVM. In these examples, we avoid using the middle view as our reference image.

In each training epoch, we shuffle and iterate over all the sequences and thus every sequence is labeled. We generate arc triplets from the motion sequences. Given a sequence $\mathcal{S} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{24}\}$, we always pick $\mathcal{C}_1$ as $\mathcal{C}_l$ and $\mathcal{C}_{24}$ as $\mathcal{C}_r$. The third intermediate reference image $\mathcal{C}_m$ is picked from $\mathcal{S}$ following a Gaussian distribution, since we expect our CVMN to tolerate variations in camera position.

Ablation Studies. In order to show our network design is optimal, we first compare our CVMN with its two variants: 1) CVMN-I2, which only uses two images ($\mathcal{C}_l$ and $\mathcal{C}_r$) as input to the encoder; and 2) CVMN-O3, which uses all three images from the arc triplet as input to our decoders for estimating $\mathcal{F}$ and $\mathcal{M}$ of the whole triplet including $\mathcal{C}_m$ (in this case, $\mathcal{F}$ and $\mathcal{M}$ have an extra dimension for $\mathcal{C}_m$), and the blending network also blends $\mathcal{C}_m$. All the other settings remain the same for the three network variations. The hyper-parameter λ, γ in Eq. (7) are set to 10 and 1 for all training sessions. We use the mean absolute error (MAE) and structural similarity index (SSIM) as error metric when comparing the predicted sequence with the ground-truth sequence.

Quantitative evaluations (as shown in Table 1) demonstrate that our proposed network outperforms its two variants. This is because the third intermediate view $\mathcal{C}_m$ help us better handle occlusion and the encode sufficiently extracts

Table 1. Quantitative evaluation on the SURREAL dataset.

Architecture	CVMN	CVMN-I2	CVMN-O3	DVM [6]
MAE	**1.453**	2.039	2.175	3.315
SSIM	**0.983**	0.966	0.967	0.945

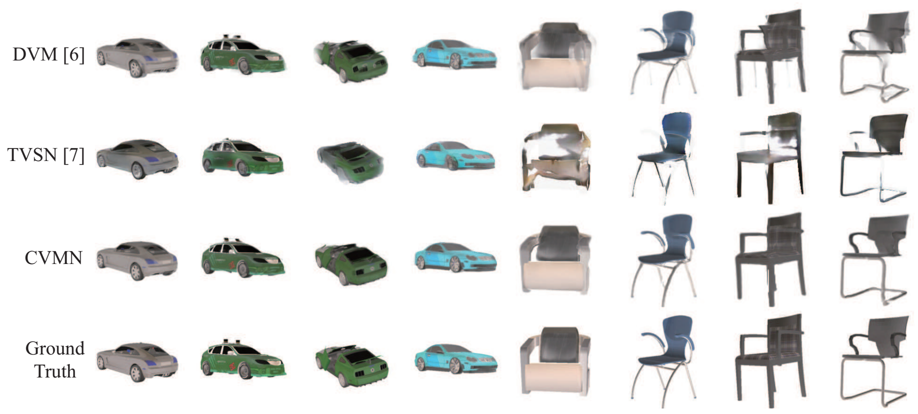

Fig. 8. Quanlitative comparisons with DVM [6] and TVSN [7] on ShapeNet.

the additional information. Fig.6 shows two motion sequences synthesized by our CVMN. The three reference views are marked in boxes. We can see that shapes and textures are well preserved in our synthesized images. Qualitative comparisons can be found in the supplemental material.

Comparison with Deep View Morphing (DVM). We also compare our approach with the state-of-the-art DVM [6]. We implement DVM following the description in the paper. To train the DVM, we randomly pick a pair of images from a sequence $\mathcal{S} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{24}\}$ and use $\mathcal{C}_{\lfloor (i+j)/2 \rfloor}$ as label. We perform quantitative and qualitative comparisons with DVM as shown in Table 1 and Fig.7. In both evaluations, we achieve better results. As shown in Fig.7, images synthesized by DVM suffer from ghosting artifacts this is because DVM cannot handle cases with complex occlusions (*e.g.*, moving arms in some sequences).

5.2 Experiments on ShapeNet

To demonstrate that our approach is generic and also works well on arbitrary 3D objects, we perform experiments on the ShapeNet dataset [10]. Specifically, we test on the car and chair models. The data preparation process is similar to the SURREAL dataset. Except that the viewing angle variation is between 30° to 90°. We use 20% of the models for testing and the rest for training. In total, the number of training sequences for "car" and "chair" are around 100K and 200K. The training process is also similar to SURREAL.

We perform both quantitative and qualitative comparisons with DVM [6], VSAF [8] and TVSN [7]. For VSAF and TVSN, we use the pre-trained model provided by the authors. When rendering their testing data, the viewing angle variations are picked from $\{40°, 60°, 80°\}$ in order to have fair comparisons. For quantitative comparisons, we use MAE as the error metric and the results are shown in Table 2. The visual quality comparison is shown in Fig.8. TVSN

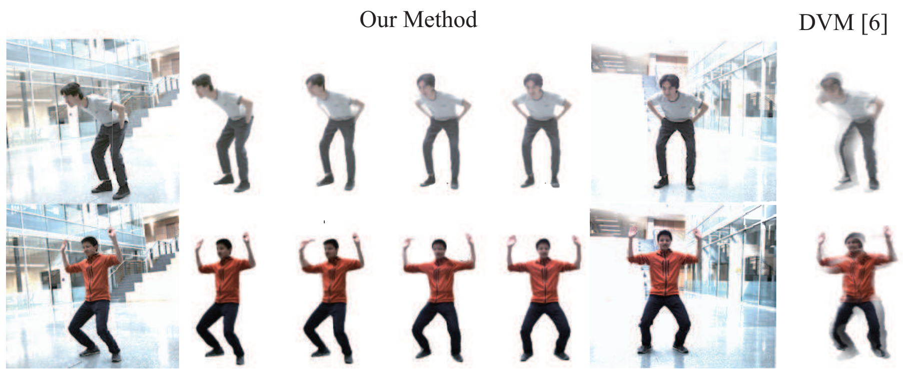

Fig. 9. Real scene results. We show four samples from our morphing sequence. We also show the middle view synthesized by DVM.

does not work well on chair models and again DVM suffers from the ghosting artifacts. Our approach works well on both categories and the synthesized images are highly close to the ground truth.

Table 2. Quantitative evaluation on the ShapeNet dataset.

Method	CVMN	DVM [6]	VSAF [8]	TVSN [7]
Car	**1.608**	3.441	7.828	5.380
Chair	**2.777**	5.579	20.54	10.02

5.3 Experiments on Real Scenes

We also test our approach on real captured motion sequences. We build a three-camera system to capture real 3D human motions for testing. This setup is shown in Fig.1. The three cameras are well synchronized and calibrated using structure-from-motion (SfM). We moved the camera positions when capturing different sequences in order to test on inputs with different viewing angle variations. Overall, the viewing angle variations between the left and right cameras are between $30°$ to $60°$.We first pre-process the captured images to correct the radial distortion and remove the background. Then we apply the cyclic rectification to obtain the arc triplets. Finally, we feed the arc triplets into our CVMN to synthesize the morphing sequences. Here we use the CVMN model trained on SURREAL dataset. Fig.9 shows samples from the resulting morphing sequences. Although the real data is more challenging due to noise, dynamic range, and lighting variations, our approach can still generate high quality results. This shows that our approach is both accurate and robust. We also compare with the

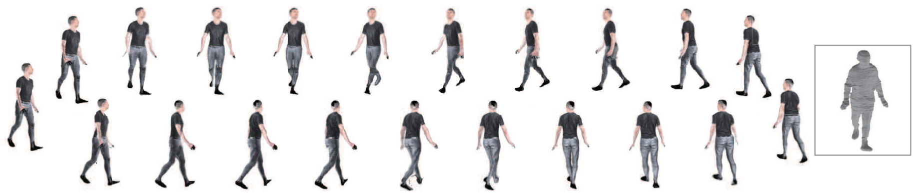

Fig. 10. Bullet-time effect rendering result. We show 21 samples out of the the 144 views in our bullet-time rendering sequence. We also show a visual hull reconstruction from the view sequence.

results produced by DVM. However, there exists severe ghosting due to large viewpoint variations.

5.4 Bullet-Time Effect Rendering

Finally, we demonstrate rendering the bullet-time effect using our synthesized view sequence. Since our synthesized views are aligned on a circular path, they are suitable for creating the bullet-time effect. To render the effect in 360°, we use 6 arc triplets composed to 12 images (neighboring triplets are sharing one image) to sample the full circle. We then generate morphing sequencing for each triplet using our approach. The motion sequences are picked from the SURREAL dataset. Fig.10 shows sample images in our bullet-time rendering sequence. Complete videos and more results are available in the supplemental material. We also perform visual hull reconstruction using the image sequence. The accurate reconstruction indicates that our synthesized views are not only visually pleasant but also geometrically correct.

6 Conclusion and Discussion

In this paper, we have presented a CNN-based view morphing framework for synthesizing intermediate views along a circular view path from three reference images. We proposed a novel cyclic rectification method for aligning the three images in one pass. Further, we developed a concyclic view morphing network for synthesizing smooth transitions from motion field and per-pixel visibility. Our approach has been validated on both synthetic and real data. We also demonstrated high quality bullet time effect rendering using our framework.

However, there are several limitations in our approach. First, our approach cannot properly handle objects with specular highlights since our network assumes Lambertian surfaces when establishing correspondences. A possible solution is to consider realistic reflectance models (*e.g.*, [34]) in our network. Second, backgrounds are not considered in our current network. Therefore, accurate background subtraction is required for our network to work well. In the future, we plan to apply semantic learning in our reference images to achieve accurate and consistent background segmentation.

References

1. Carranza, J., Theobalt, C., Magnor, M.A., Seidel, H.P.: Free-viewpoint video of human actors. ACM Trans. Graph. **22**(3) (2003) 569–577
2. Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. ACM Trans. Graph. **23**(3) (2004) 600–608
3. Liao, J., Lima, R.S., Nehab, D., Hoppe, H., Sander, P.V., Yu, J.: Automating image morphing using structural similarity on a halfway domain. ACM Trans. Graph. **33**(5) (2014) 168:1–168:12
4. Linz, C., Lipski, C., Rogge, L., Theobalt, C., Magnor, M.: Space-time visual effects as a post-production process. In: Proceedings of the 1st International Workshop on 3D Video Processing, ACM (2010)
5. Seitz, S.M., Dyer, C.R.: View morphing. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, ACM (1996) 21–30
6. Ji, D., Kwon, J., McFarland, M., Savarese, S.: Deep view morphing. IEEE Conference on Computer Vision and Pattern Recognition (2017)
7. Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: Transformation-grounded image generation network for novel 3d view synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)
8. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: European Conference on Computer Vision. (2016)
9. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from Synthetic Humans. In: The IEEE Conference on Computer Vision and Pattern Recognition. (2017)
10. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 (2015)
11. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, ACM (1996) 31–42
12. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, ACM (1996) 43–54
13. Penner, E., Zhang, L.: Soft 3d reconstruction for view synthesis. ACM Trans. Graph. **36**(6) (2017) 235:1–235:11
14. Rematas, K., Nguyen, C.H., Ritschel, T., Fritz, M., Tuytelaars, T.: Novel views of objects from a single image. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(8) (2017) 1576–1590
15. Lipski, C., Linz, C., Berger, K., Sellent, A., Magnor, M.: Virtual video camera: Image-based viewpoint navigation through space and time. Computer Graphics Forum (2010) 2555–2568
16. Ballan, L., Brostow, G.J., Puwein, J., Pollefeys, M.: Unstructured video-based rendering: Interactive exploration of casually captured videos. ACM Trans. Graph. **29**(4) (2010) 87:1–87:11
17. Zhang, Z., Wang, L., Guo, B., Shum, H.Y.: Feature-based light field morphing. ACM Trans. Graph. **21**(3) (2002) 457–464

18. Beier, T., Neely, S.: Feature-based image metamorphosis. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '92 (1992) 35–42
19. Lee, S., Wolberg, G., Shin, S.Y.: Polymorph: morphing among multiple images. IEEE Computer Graphics and Applications **18**(1) (1998) 58–71
20. Quenot, G.M.: Image matching using dynamic programming: Application to stereovision and image interpolation. In: Image Communication. (1996)
21. Chaurasia, G., Sorkine-Hornung, O., Drettakis, G.: Silhouette-aware warping for image-based rendering. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) **30**(4) (2011)
22. Germann, M., Popa, T., Keiser, R., Ziegler, R., Gross, M.: Novel-view synthesis of outdoor sport events using an adaptive view-dependent geometry. Comput. Graph. Forum **31** (2012) 325–333
23. Mahajan, D., Huang, F.C., Matusik, W., Ramamoorthi, R., Belhumeur, P.: Moving gradients: A path-based method for plausible image interpolation. ACM Trans. Graph. **28**(3) (2009) 42:1–42:11
24. Dosovitskiy, A., Springenberg, J.T., Brox, T.: Learning to generate chairs with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. (2015)
25. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3d models from single images with a convolutional network. In: European Conference on Computer Vision. (2016)
26. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)
27. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: IEEE International Conference on Computer Vision. (2017)
28. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. (2015) 2017–2025
29. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deep stereo: Learning to predict new views from the world's imagery. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)
30. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Trans. Graph. **35**(6) (2016) 193:1–193:10
31. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)
32. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In Leibe, B., Matas, J., Sebe, N., Welling, M., eds.: European Conference on Computer Vision. (2016) 483–499
33. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6) (2015) 248:1–248:16
34. Rematas, K., Ritschel, T., Fritz, M., Gavves, E., Tuytelaars, T.: Deep reflectance maps. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)