



Latent transformations neural network for object view synthesis

Sangpil Kim¹ · Nick Winovich¹ · Hyung-Gun Chi¹ · Guang Lin¹ · Karthik Ramani¹

© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

We propose a fully convolutional conditional generative neural network, the latent transformation neural network, capable of rigid and non-rigid object view synthesis using a lightweight architecture suited for real-time applications and embedded systems. In contrast to existing object view synthesis methods which incorporate conditioning information via concatenation, we introduce a dedicated network component, the conditional transformation unit. This unit is designed to learn the latent space transformations corresponding to specified target views. In addition, a consistency loss term is defined to guide the network toward learning the desired latent space mappings, a task-divided decoder is constructed to refine the quality of generated views of objects, and an adaptive discriminator is introduced to improve the adversarial training process. The generalizability of the proposed methodology is demonstrated on a collection of three diverse tasks: multi-view synthesis on real hand depth images, view synthesis of real and synthetic faces, and the rotation of rigid objects. The proposed model is shown to be comparable with the state-of-the-art methods in structural similarity index measure and L_1 metrics while simultaneously achieving a 24% reduction in the compute time for inference of novel images.

Keywords Object view synthesis · Latent transformation · Fully convolutional · Conditional generative model

1 Introduction

The task of synthesizing novel views of objects from a single reference frame/view is an important problem which has a variety of practical applications in computer vision, graphics, and robotics. In computer vision, view synthesis can be used to generate 3D point cloud representations of objects from a single input image [40]; in the context of hand pose estimation algorithms, generating additional synthetic views can also help reduce occlusion and improve the accuracy of the estimated poses [12,14]. In computer graphics, view synthesis has been used to apply changes in lighting and viewpoint to single-view images of faces [24]. In robotics, synthetic views can be used to help predict unobserved part locations and improve the performance of object grasping with manipulators [42]. However, synthesizing novel views from a single input image is a formidable task with serious complications arising from the complexity of the target object and the presence of heavily self-occluded parts.

Generative models have been shown to provide effective frameworks for representing complex, structured datasets and generating realistic samples from underlying data distributions [7,13]. This concept has also been extended to form conditional models capable of sampling from conditional distributions in order to allow certain properties of the generated data to be controlled or selected [26,43]. Generative models without encoders [5,46] are generally used to sample from broad classes of the data distributions; however, these models are not designed to incorporate input data and therefore cannot preserve characteristic features of specified input data. Models have also been proposed which incorporate encoding components to overcome this by learning to map input data onto an associated *latent space* representation within a generative framework [21,25]. The resulting inference models allow for the defining features of inputs to be preserved while specified target properties are adjusted through conditioning [45].

Conventional conditional models have largely relied on rather simple methods, such as concatenation, for implementing this conditioning process; however, cGANs [27] have shown that utilizing the conditioning information in a less trivial, more methodical manner has the potential to significantly improve the performance of conditional generative

✉ Karthik Ramani
ramani@purdue.edu

Sangpil Kim
kim2030@purdue.edu

¹ Purdue University, West Lafayette, IN 47906, USA

models. In this work, we provide a general framework for effectively performing inference with conditional generative models by strategically controlling the interaction between conditioning information and latent representations within a generative inference model. In this framework, a conditional transformation unit (CTU), Φ , is introduced to provide a means for navigating the underlying manifold structure of the latent space. The CTU is realized in the form of a collection of convolutional layers which are designed to approximate the latent space operators defined by mapping encoded inputs to the encoded representations of specified targets (see Fig. 1). This is enforced by introducing a *consistency loss* term to guide the CTU mappings during training. In addition, a conditional discriminator unit (CDU), Ψ , also realized as a collection of convolutional layers, is included in the network's discriminator. This CDU is designed to improve the network's ability to identify and eliminate transformation-specific artifacts in the network's predictions.

The network has also been equipped with RGB balance parameters consisting of three values $\{\theta_R, \theta_G, \theta_B\}$ designed to give the network the ability to quickly adjust the global color balance of the images it produces to better align with that of the true data distribution. In this way, the network is easily able to remove unnatural hues and focus on estimating local pixel values by adjusting the three RGB parameters rather than correcting each pixel individually. In addition, we introduce a novel estimation strategy for efficiently learning shape and color properties simultaneously; a *task-divided* decoder (TD) is designed to produce a coarse pixel value map along with a refinement map in order to split the network's overall task into distinct, dedicated network components.

Summary of contributions:

1. We introduce the conditional transformation unit, with a family of modular filter weights, to learn high-level mappings within a low-dimensional latent space. In addition, we present a consistency loss term which is used to guide the transformations learned during training.
2. We propose a novel framework for 3D object view synthesis which separates the generative process into distinct network components dedicated to learning (i) coarse pixel value estimates, (ii) pixel refinement map, and (iii) the global RGB color balance of the dataset.
3. We introduce the conditional discriminator unit designed to improve adversarial training by identifying and eliminating transformation-specific artifacts present in the generated images.

Each contribution proposed above has been shown to provide a significant improvement to the network's overall performance through a series of ablation studies. The resulting latent transformation neural network (LTNN) is placed

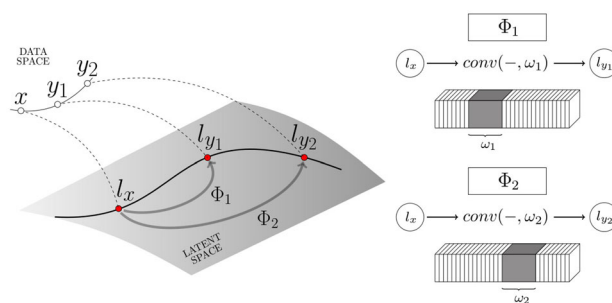


Fig. 1 The conditional transformation unit Φ constructs a collection of mappings $\{\Phi_k\}$ in the latent space which produce object view changes to the decoded outputs. Conditioning information is used to select the appropriate convolutional weights ω_k for the specified transformation; the encoding l_x of the original input image x is transformed to $\hat{l}_{y_k} = \Phi_k(l_x) = \text{conv}(l_x, \omega_k)$ and provides an approximation to the encoding l_{y_k} of the attribute-modified target image y_k

through a series of comparative studies on a diverse range of experiments where it is seen to be comparable with the existing state-of-the-art models for (i) simultaneous multi-view synthesis of real hand depth images in real-time, (ii) the synthesis of rotated views of rigid objects given a single image, and (iii) object view synthesis and attribute modification of real and synthetic faces.

Moreover, the CTU conditioning framework allows for additional conditioning information, or target views, to be added to the training procedure ad infinitum without any increase to the network's inference speed.

2 Related work

Conditional generative models have been widely used in computer vision areas such as geometric prediction [30,34,40,48] and non-rigid object modification such as human face deformation [1,11,33,47]. Dosovitskiy et al. [8] has proposed a supervised, conditional generative model trained to generate images of chairs, tables, and cars with specified attributes which are controlled by transformation and view parameters passed to the network. MV3D [40] is pioneering deep learning work for object view synthesis which uses an encoder–decoder network to directly generate pixels of a target view with depth information in the loss function along with view point information passed as a conditional term. The appearance flow network (AFN) [48] proposed a method for view synthesis of objects by predicting appearance flow fields, which are used to move pixels from an input to a target view. However, this method requires detailed camera pose information and is not capable of predicting pixels which are missing in the source views. The M2N from [38] proposed view prediction using a recurrent network and a self-learned confidence map iteratively synthesizes views with recurrent pixel generator with appearance flow. TVSN [30] uses a vis-

ibility map, which indicates visible parts in a target image to identify occlusion in different views. However, this method requires mesh models for each object in order to extract visibility maps for training the network. The DFN by Jia et al. [20] proposed using a dynamic filter which is conditioned on a sequence of previous frames; this is fundamentally different from our method since the filter is applied to the original inputs rather than the latent embeddings. Moreover, it relies on temporal information and is not applicable for predictions given a single image. The IterGAN model introduced by Galama and Mensink [10] is also designed to synthesize novel views from a single image, with a specific emphasis on the synthesis of rotated views of objects in small, iterative steps. The conditional variational autoencoder (CVAE) incorporates conditioning information into the standard variational autoencoder (VAE) framework [23] and is capable of synthesizing specified attribute changes in an identity preserving manner [37,45]. Other works have introduced a clamping strategy to enforce a specific organizational structure in the latent space [24,33]; these networks require extremely detailed labels for supervision, such as the graphics code parameters used to create each example, and are therefore very difficult to implement for more general tasks (e.g., training with real images). These models are all reliant on additional knowledge for training, such as depth information, camera poses, or mesh models, and are not applicable in embedded systems and real-time applications due to the high computational demand and the number of neural networks' parameters since these methods did not consider the efficiency of the model.

CVAE-GAN [2] further adds adversarial training to the CVAE framework in order to improve the quality of generated predictions. The work from Zhang et al. [47] has introduced the conditional adversarial autoencoder (CAAE) designed to model age progression/regression in human faces. This is achieved by concatenating conditioning information (i.e., age) with the input's latent representation before proceeding to the decoding process. The framework also includes an adaptive discriminator with conditional information passed using a resize/concatenate procedure. To the best of our knowledge, all existing conditional generative models are designed for inference use fixed hidden layers and concatenate conditioning information directly with latent representations. In contrast to these existing methods, the proposed model incorporates conditioning information by defining dedicated, transformation-specific convolutional layers at the latent level. This conditioning framework allows the network to synthesize multiple transformed views from a single input, while retaining a fully convolutional structure which avoids the dense connections used in existing inference-based conditional models. Most significantly, the proposed LTNN framework is shown to be comparable with the state-of-the-art models in a diverse range of object view

synthesis tasks, while requiring substantially less FLOPs and memory consumption for inference than other methods.

3 Latent transformation neural network

In this section, we introduce the methods used to define the proposed LTNN model. We first give a brief overview of the LTNN network structure. We then detail how conditional transformation unit mappings are defined and trained to operate on the latent space, followed by a description of the conditional discriminator unit implementation and the network loss function used to guide the training process. Lastly, we describe the task division framework used for the decoding process.

The basic workflow of the proposed model is as follows:

1. Encode the input image x to a latent representation $l_x = \text{Encode}(x)$.
2. Use conditioning information k to select conditional, convolutional filter weights ω_k .
3. Map the latent representation l_x to $\hat{l}_{y_k} = \Phi_k(l_x) = \text{conv}(l_x, \omega_k)$, an approximation of the encoded latent representation l_{y_k} of the specified target image y_k .
4. Decode \hat{l}_{y_k} to obtain a coarse pixel value map and a refinement map.
5. Scale the channels of the pixel value map by the RGB balance parameters and take the Hadamard product with the refinement map to obtain the final prediction \hat{y}_k .
6. Pass real images y_k as well as generated images \hat{y}_k to the discriminator, and use the conditioning information to select the discriminator's conditional filter weights $\bar{\omega}_k$.
7. Compute loss and update weights using ADAM optimization and backpropagation.

3.1 Conditional transformation unit

Generative models have frequently been designed to explicitly disentangle the latent space in order to enable high-level attribute modification through linear, latent space interpolation. This linear latent structure is imposed by design decisions, however, and may not be the most natural way for a network to internalize features of the data distribution. Several approaches have been proposed which include non-linear layers for processing conditioning information at the latent space level. In these conventional conditional generative frameworks, conditioning information is introduced by combining features extracted from the input with features extracted from the conditioning information (often using dense connection layers); these features are typically combined using standard vector concatenation, although some have opted to use channel concatenation.

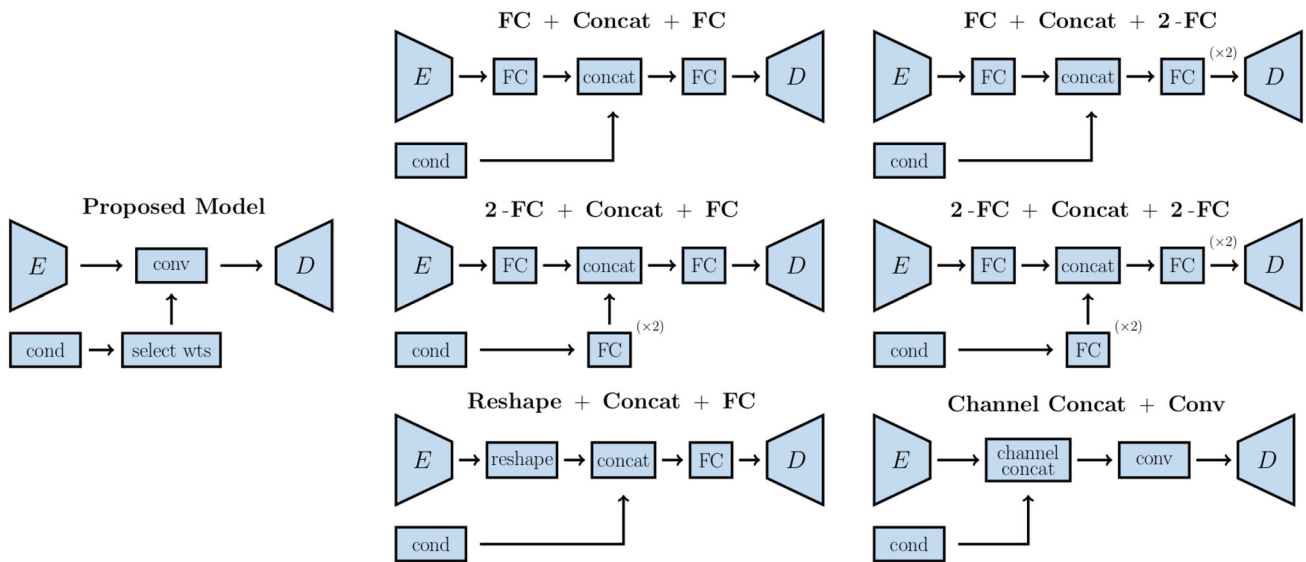


Fig. 2 Selected methods for incorporating conditioning information; the proposed LTNN method is illustrated on the left, and six conventional alternatives are shown to the right

In particular, conventional approaches for incorporating conditional information generally fall into three classes: (1) apply a fully connected layer before and after concatenating a vector storing conditional information [24,40,47,48], (2) flatten the network features and concatenate with a vector storing conditional information [30], (3) tile a conditional vector to create a two-dimensional array with the same shape as the network features and concatenate channel-wise [2,38]. Since the first class is more prevalent than the others in practice, we have subdivided this class into four cases: FC-Concat-FC [47], FC-Concat-2FC [24], 2FC-Concat-FC [48], and 2FC-Concat-2FC [40]. Six of these conventional conditional network designs are illustrated in Fig. 2 along with the proposed LTNN network design for incorporating conditioning information.

Rather than directly concatenating conditioning information with network features, we propose using a conditional transformation unit (CTU), consisting of a collection of distinct convolutional mappings in the network's latent space. More specifically, the CTU maintains independent convolution kernel weights for each target view in consideration. Conditioning information is used to select which collection of kernel weights, i.e., which CTU mapping, should be used in the CTU convolutional layer to perform a specified transformation. In addition to the convolutional kernel weights, each CTU mapping incorporates a Swish activation [32] with independent parameters for each specified target view. The kernel weights and Swish parameters of each CTU mapping are selectively updated by controlling the gradient flow based on the conditioning information provided.

The CTU mappings are trained to transform the encoded, latent space representation of the network's input in a man-

ner which produces high-level view or attribute changes upon decoding. In this way, different angles of view, light directions, and deformations, for example, can be generated from a single input image. In one embodiment, the training process for the conditional transformation units can be designed to form a semigroup $\{\Phi_t\}_{t \geq 0}$ of operators:

$$\text{i.e., } \begin{cases} \Phi_0 = id \\ \Phi_{t+s} = \Phi_t \circ \Phi_s \quad \forall t, s \geq 0 \end{cases} \quad (1)$$

defined on the latent space and trained to follow the geometric flow corresponding to a specified attribute. In the context of rotating three-dimensional objects, for example, the transformation units are trained on the input images paired with several target outputs corresponding to different angles of rotation; the network then uses conditioning information, which specifies the angle by which the object should be rotated, to select the appropriate transformation unit. In this context, the semigroup criteria correspond to the fact that rotating an object 10° twice should align with the result of rotating the object by 20° once.

Since the encoder and decoder are not influenced by the specified angle of rotation, the network's encoding/decoding structure learns to model objects at different angles simultaneously; the single, low-dimensional latent representation of the input contains all information required to produce rotated views of the original object. Other embodiments can depart with this semigroup formulation, however, training conditional transformation units to instead produce a more diverse collection of non-sequential viewpoints, for example, as is the case for multi-view hand synthesis.

LTNN Training Procedure

Provide: Labeled dataset $\{(x, \{y_k\}_{k \in \mathcal{T}})\}$ with target transformations indexed by a fixed set \mathcal{T} , encoder weights θ_E , decoder weights θ_D , RGB balance parameters $\{\theta_R, \theta_G, \theta_B\}$, conditional transformation unit weights $\{\omega_k\}_{k \in \mathcal{T}}$, discriminator \mathcal{D} with standard weights $\theta_{\mathcal{D}}$ and conditionally selected weights $\{\bar{\omega}_k\}_{k \in \mathcal{T}}$, and loss function hyperparameters $\gamma, \rho, \lambda, \kappa$ corresponding to the smoothness, reconstruction, adversarial, and consistency loss terms, respectively. The specific loss function components are defined in detail in Eqs. 3–2 in Sect. 3.2.

```

1: procedure TRAIN()
2:   // Sample input and targets from training set
3:    $x, \{y_k\}_{k \in \mathcal{T}} = \text{get\_train\_batch}()$ 
4:   // Compute encoding of original input image
5:    $l_x = \text{Encode}[x]$ 
6:   for  $k$  in  $\mathcal{T}$  do
7:     // Compute true encoding of specified target image
8:      $l_{y_k} = \text{Encode}[y_k]$ 
9:     // Compute approximate encoding of target with CTU
10:     $\hat{l}_{y_k} = \text{conv}(l_x, \omega_k)$ 
11:    // Compute RGB value and refinement maps
12:     $\hat{y}_k^{\text{value}}, \hat{y}_k^{\text{refine}} = \text{Decode}[\hat{l}_{y_k}]$ 
13:    // Assemble final network prediction for target
14:     $\hat{y}_k = [\theta_C \cdot \hat{y}_{k,C}^{\text{value}} \odot \hat{y}_{k,C}^{\text{refine}}]_{C \in \{R,G,B\}}$ 
15:
16:    // Update encoder, decoder, RGB, and CTU weights
17:     $\mathcal{L}_{adv} = -\log(\mathcal{D}(\hat{y}_k, \bar{\omega}_k))$ 
18:     $\mathcal{L}_{guide} = \gamma \cdot \mathcal{L}_{smooth}(\hat{y}_k) + \rho \cdot \mathcal{L}_{recon}(\hat{y}_k, y_k)$ 
19:     $\mathcal{L}_{consist} = \|\hat{l}_{y_k} - l_{y_k}\|_1$ 
20:     $\mathcal{L} = \lambda \cdot \mathcal{L}_{adv} + \mathcal{L}_{guide} + \kappa \cdot \mathcal{L}_{consist}$ 
21:    for  $\theta$  in  $\{\theta_E, \theta_D, \theta_R, \theta_G, \theta_B, \omega_k\}$  do
22:       $\theta = \theta - \nabla_{\theta} \mathcal{L}$ 
23:
24:    // Update discriminator and CDU weights
25:     $\mathcal{L}_{adv}^{\mathcal{D}} = -\log(\mathcal{D}(y_k, \bar{\omega}_k)) - \log(1 - \mathcal{D}(\hat{y}_k, \bar{\omega}_k))$ 
26:    for  $\theta$  in  $\{\theta_{\mathcal{D}}, \bar{\omega}_k\}$  do
27:       $\theta = \theta - \nabla_{\theta} \mathcal{L}_{adv}^{\mathcal{D}}$ 

```

To enforce this behavior on the latent space CTU mappings in practice, a consistency term is introduced into the loss function, as specified in Eq. 2. This loss term is minimized precisely when the CTU mappings behave as depicted in Fig. 1; in particular, the output of the CTU mapping associated with a particular transformation is designed to match the encoding of the associated ground truth target view. More precisely, given an input image x , the consistency loss associated with the k th transformation is defined in terms of the ground truth, transformed target view y_k by:

$$\mathcal{L}_{consist} = \|\Phi_k(\text{Encode}[x]) - \text{Encode}[y_k]\|_1 \tag{2}$$

3.2 Discriminator and loss function

The discriminator used in the adversarial training process is also passed conditioning information which specifies the transformation which the model has attempted to make. The conditional discriminator unit (CDU), which is implemented as a convolutional layer with modular weights similar to the

CTU, is trained to specifically identify unrealistic artifacts which are being produced by the corresponding conditional transformation unit mappings. This is accomplished by maintaining independent convolutional kernel weights for each specified target view and using the conditioning information passed to the discriminator to select the kernel weights for the CDU layer. The incorporation of this context-aware discriminator structure has significantly boosted the performance of the network (see Table 1). The discriminator, \mathcal{D} , is trained using the adversarial loss term $\mathcal{L}_{adv}^{\mathcal{D}}$ defined in Eq. 3. The proposed model uses the adversarial loss in Eq. 4 to effectively capture multimodal distributions [36], which helps to sharpen the generated views.

$$\mathcal{L}_{adv}^{\mathcal{D}} = -\log \mathcal{D}(y_k, \bar{\omega}_k) - \log(1 - \mathcal{D}(\hat{y}_k, \bar{\omega}_k)) \tag{3}$$

$$\mathcal{L}_{adv} = -\log \mathcal{D}(\hat{y}_k, \bar{\omega}_k). \tag{4}$$

Reducing the total variation is widely used in view synthesis methods [30,47]. In particular, the L_{smooth} term is used to reduce noise in the generated images by reducing the variation of pixels, which is inspired by total variation image denoising. Experimental evidence shows that the inclusion of the L_{smooth} loss term leads to an improvement in the overall quality of the synthesized images (see Table 1). We have experimented with various shift sizes and found that the shift size $\tau = 1$ yields the best performance.

Additional loss terms corresponding to accurate structural reconstruction and smoothness [19] in the generated views are defined in Eqs. 5 and 6:

$$\mathcal{L}_{recon} = \|\hat{y}_k - y_k\|_2^2 \tag{5}$$

$$\mathcal{L}_{smooth} = \sum_{i \in \{0, \pm 1\}} \sum_{j \in \{0, \pm 1\}} \|\hat{y}_k - \tau_{i,j} \hat{y}_k\|_1, \tag{6}$$

where y_k is the modified target image corresponding to an input x , $\bar{\omega}_k$ are the weights of the CDU mapping corresponding to the k th transformation, Φ_k is the CTU mapping for the k th transformation, $\hat{y}_k = \text{Decode}(\Phi_k(\text{Encode}[x]))$ is the network prediction, and $\tau_{i,j}$ is the two-dimensional, discrete shift operator. The final loss function for the encoder and decoder components is given by:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{adv} + \rho \cdot \mathcal{L}_{recon} + \gamma \cdot \mathcal{L}_{smooth} + \kappa \cdot \mathcal{L}_{consist} \tag{7}$$

with hyperparameters typically selected so that $\lambda, \rho \gg \gamma, \kappa$. The consistency loss is designed to guide the CTU mappings toward approximations of the latent space mappings which connect the latent representations of input images and target images as depicted in Fig. 1. In particular, the consistency term enforces the condition that the transformed encoding, $\hat{l}_{y_k} = \Phi_k(\text{Encode}[x])$, approximates the encoding of the k th target image, $l_{y_k} = \text{Encode}[y_k]$, during the training process.

Table 1 Ablation/comparison results of six different conventional alternatives for fusing condition information into the latent space and ablation study of conditional transformation unit (CTU), conditional discriminator unit (CDU), and task-divided decoder (TD)

Model	Elevation		Azimuth		Light Direction		Age	
	SSIM	L_1	SSIM	L_1	SSIM	L_1	SSIM	L_1
LTNN (CTU + CDU + TD)	.923	.107	.923	.108	.941	.093	.925	.102
LTNN w/o L_{smooth}	.918	.118	.921	.114	.935	.112	.911	.110
CTU + CDU	.901	.135	.908	.125	.921	.121	.868	.118
CTU	.889	.142	.878	.135	.901	.131	.831	.148
Channel Concat + Conv	.803	.179	.821	.173	.816	.182	.780	.188
2-FC + Concat + 2-FC	.674	.258	.499	.355	.779	.322	.686	.243
2-FC + Concat + FC	.691	.233	.506	.358	.787	.316	.687	.240
FC + Concat + 2-FC	.673	.261	.500	.360	.774	.346	.683	.249
FC + Concat + FC	.681	.271	.497	.355	.785	.315	.692	.246
Reshape + Concat + FC	.671	.276	.489	.357	.780	.318	.685	.251

For valid comparison, we used identical encoder, decoder, and training procedure with synthetic face dataset
 Bold values indicate the best performance

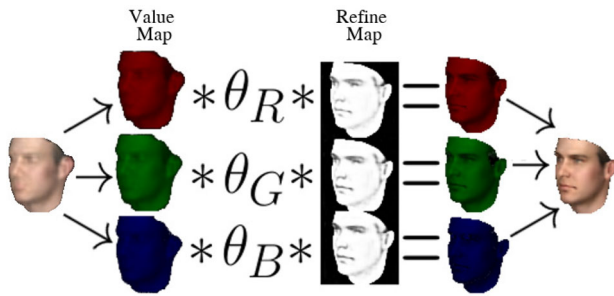


Fig. 3 Proposed task-divided design for the LTNN decoder. The coarse pixel value estimation map is split into RGB channels, rescaled by the RGB balance parameters, and multiplied element-wise by the refinement map values to produce the final network prediction

3.3 Task-divided decoder

The decoding process has been divided into three tasks: estimating the refinement map, pixel values, and RGB color balance of the dataset. We have found this decoupled framework for estimation helps the network converge to better minima to produce sharp, realistic outputs without additional loss terms. The decoding process begins with a series of convolutional layers followed by bilinear interpolation to upsample the low-resolution latent information. The last component of the decoder’s upsampling process consists of two distinct convolutional layers used for task divide; one layer is allocated for predicting the refinement map, while

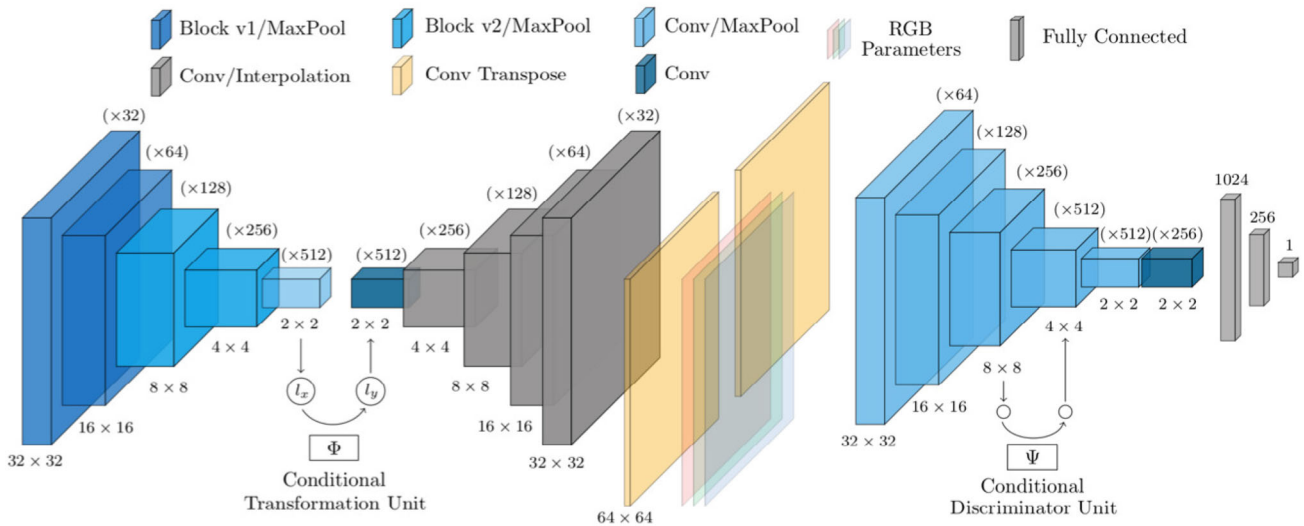


Fig. 4 The proposed network structure for the encoder/decoder (left) and discriminator (right) for 64×64 input images. Features have been color-coded according to the type of layer which has produced them. The CTU and CDU components both store and train separate collections of 3×3 filter weights for each conditional transformation; in particular,

the number of distinct 3×3 filters associated with the CTU and CDU corresponds to the number of distinct conditional transformations the network is designed to produce. For 256×256 input images, we have added two Block v1/MaxPool layers in the front of encoder and two Conv/Interpolation layers at the end of the decoder

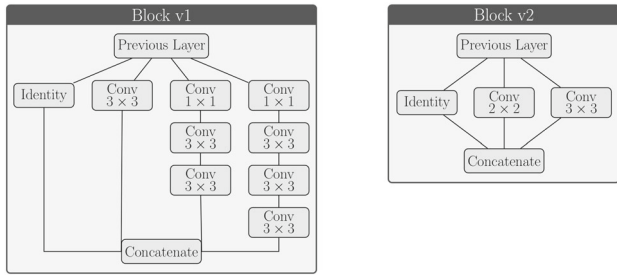


Fig. 5 Layer definitions for Block v1 and Block v2 collaborative filters. Once the total number of output channels, N_{out} , is specified, the remaining $N_{out} - N_{in}$ output channels are allocated to the non-identity filters (where N_{in} denotes the number of input channels). For the Block v1 layer at the start of the proposed LTNN model, for example, the input is a image with $N_{in} = 3$ channels and the specified number of output channels is $N_{out} = 32$. One of the 32 channels is accounted for by the identity component, and the remaining 29 channels are the three non-identity filters. When the remaining channel count is not divisible by 3, we allocate the remainder of the output channels to the single 3×3 convolutional layer. Swish activation functions are used for each filter; however, the filters with multiple convolutional layers do not use activation functions for the intermediate 3×3 convolutional layers

the other is trained to predict pixel values. The refinement map layer incorporates a sigmoidal activation function which outputs scaling factors intended to refine the coarse pixel value estimations; the pixel value estimation layer does not use an activation so that the output values are not restricted to the range of a specific activation function. RGB balance parameters, consisting of three trainable variables, are used as weights for balancing the color channels of the pixel value map. The Hadamard product, \odot , of the refinement map and the RGB-rescaled value map serves as the network’s final output:

$$\hat{y} = [\hat{y}_R, \hat{y}_G, \hat{y}_B] \quad \text{where}$$

$$\hat{y}_C = \theta_C \cdot \hat{y}_C^{value} \odot \hat{y}_C^{refine} \quad \text{for } C \in \{R, G, B\} \quad (8)$$

In this way, the network has the capacity to mask values which lie outside of the target object (i.e., by setting refinement map values to zero) which allows the value map to focus on the object itself during the training process. Experimental results show that the refinement maps learn to produce masks which closely resemble the target objects’ shapes and have sharp drop-offs along the boundaries. No additional information has been provided to the network for training the refinement map; the masking behavior illustrated in Figs. 3 and 6 is learned implicitly by the network during training and is made possible by the design of the network’s architecture. As shown in Fig. 3, the refinement map produces a shape mask and mask out errors in each pixels by masking values which lie outside of the target object (i.e., by setting refinement map values to zero).

4 Architecture details

The overview of the pipeline is shown in Fig. 4. Input images are passed through a Block v1 collaborative filter layer (see Fig. 5) along with a max pooling layer to produce the features at the far left end of the figure. At the bottleneck between the encoder and decoder, a conditional transformation unit (CTU) is applied to map the 2×2 latent features directly to the transformed 2×2 latent features on the right. This CTU is implemented as a convolutional layer with 3×3 filter weights selected based on the conditioning information provided to the network. The features near the end of the decoder component are processed by two independent convolution transpose layers for non-rigid object and bilinear interpolation for the rigid object: one corresponding to the value estimation map and the other corresponding to the refinement map. The channels of the value estimation map are rescaled by the RGB balance parameters, and the Hadamard product is taken with the refinement map to produce the final network output. For rigid object experiment, we added tangent hyperbolic activation function after the Hadamard product to bound the output values range in $[-1, 1]$. The CDU is also designed to have the same 3×3 kernel size as the CTU and is applied between the third and fourth layers of the discriminator. For the stereo face dataset [9] experiment, we have added an additional Block v1 layer in the encoder and additional convolutional layer followed by bilinear interpolation in decoder to utilize the full $128 \times 128 \times 3$ resolution images and two Block v1 layers and two convolutional layers followed by bilinear interpolation for the $256 \times 256 \times 3$ resolution image of rigid object views.

The encoder incorporates two main block layers, as defined in Fig. 5, which are designed to provide efficient feature extraction; these blocks follow a similar design to that proposed by Szegedy et al. [39], but include dense connections between blocks, as introduced by Huang et al. [16]. We normalize the output of each network layer using the batch normalization method as described in [18]. For the decoder, we have opted for a minimalist design, inspired by the work of [31]. Standard convolutional layers with 3×3 filters and same padding are used through the penultimate decoding layer and transpose convolutional layers with 1×1 filters for non-rigid objects and 5×5 for other experiments. We have used same padding to produce the value estimation and refinement maps. All parameters have been initialized using the variance scaling initialization method described in [15].

Our method has been implemented and developed using the TensorFlow framework. The models have been trained using stochastic gradient descent (SGD) and the ADAM optimizer [22] with initial parameters: learning_rate = 0.005, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ (as defined in the TensorFlow API r1.6 documentation for tf.train.AdamOptimizer), along with loss function hyperparameters: $\lambda = 0.8$, $\rho = 0.2$, $\gamma = 0.000025$, and $\kappa = 0.00005$ (as introduced in Eq.7). The

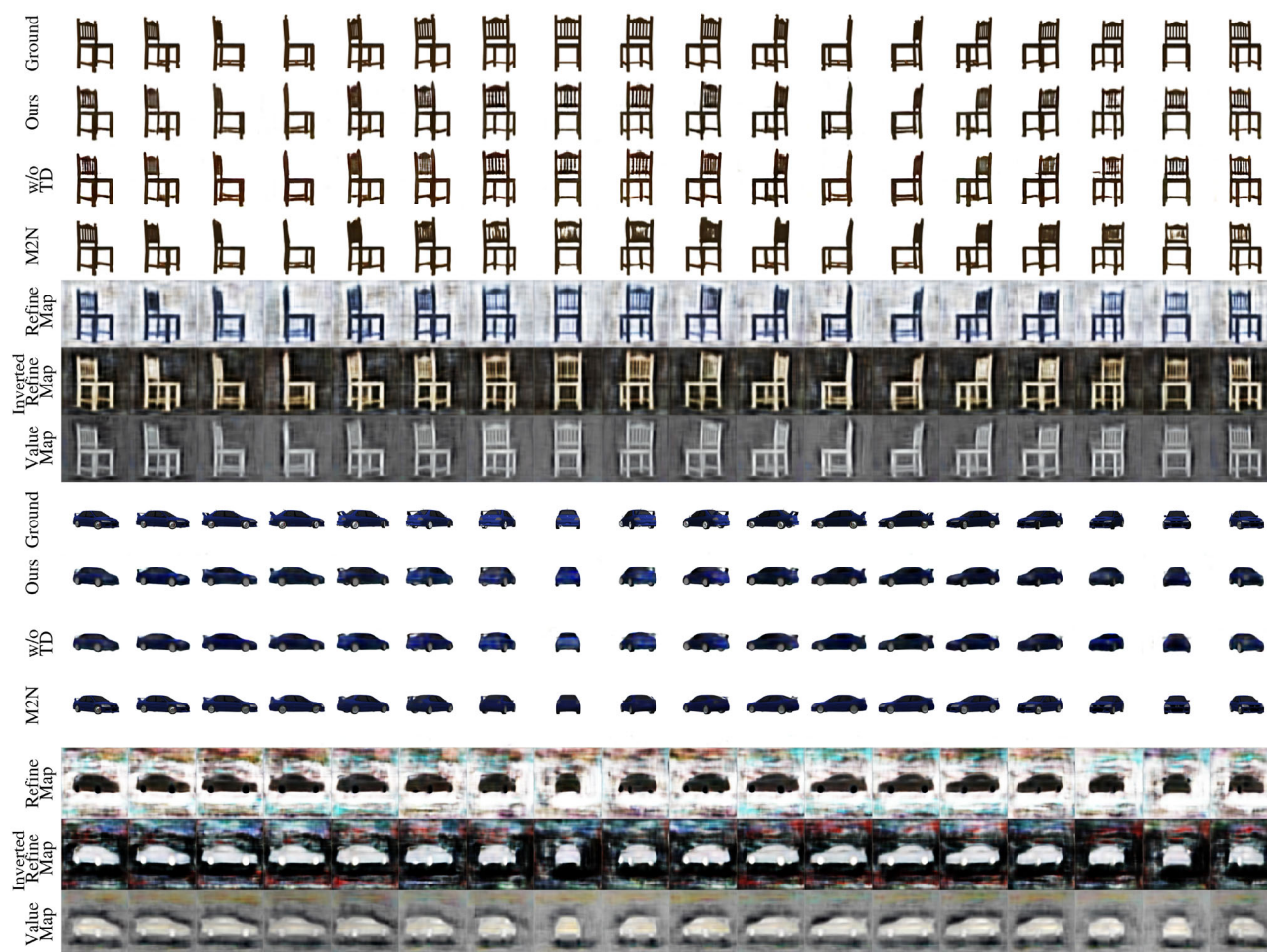


Fig. 6 Qualitative comparison of 360° view prediction of rigid objects. A single image, shown in the first column of the “Ground” row, is used as the input for the network. Results are shown for the proposed network with and without task division (“w/o TD”) as well as a comparison with

M2N. The pixel value map and refinement maps corresponding to the task division framework are also provided as well as an inverted view of the refinement map for better visibility

discriminator is updated once every two encoder/decoder updates, and one-sided label smoothing [36] has been used to improve the stability of the discriminator training procedure.

5 Experiments and results

We conduct experiments on a diverse collection of datasets including both rigid and non-rigid objects. To show the generalizability of our method, we have conducted a series of experiments: (i) hand pose estimation using a synthetic training set and real NYU hand depth image data [41] for testing, (ii) synthesis of rotated views of rigid objects using the 3D object dataset [4], (iii) synthesis of rotated views using a real face dataset [9], and (iv) the modification of a diverse range of attributes on a synthetic face dataset [17]. For each experiment, we have trained the models using

80% of the datasets. Since ground truth target depth images were not available for the real hand dataset, an indirect metric has been used to quantitatively evaluate the model as described in Sect. 5.2. Ground truth data were available for all other experiments, and models were evaluated directly using the L_1 mean pixel-wise error and the structural similarity index measure (SSIM) [44] used in [30,38]. To evaluate the proposed framework with existing works, two comparison groups have been formed: conditional inference methods, CVAE-GAN [2] and CAAE [47], with comparable hourglass structures for comparison on experiments with non-rigid objects, and view synthesis methods, MV3D [40], M2N [38], AFN [48], and TVSN [30], for comparison on experiments with rigid objects. Additional ablation experiments have been performed to compare the proposed CTU conditioning method with other conventional concatenation methods (see Fig. 2); results are shown in Fig. 9 and Table 1.

Table 2 FLOPs and parameter counts corresponding to inference for a single image with resolution $256 \times 256 \times 3$

Model	Parameters (Million)	GFLOPs/image
Ours	17.0	2.183
M2N	127.1	341.404
TVSN	57.3	2.860
AFN	70.3	2.671
MV3D	69.7	3.056

These calculations are based on code provided by the authors and the definitions prescribed in the associated papers. Smaller numbers are better for parameters and GFLOPs/Image
 Bold values indicate the best performance

Table 3 Quantitative comparison for 360° view synthesis of rigid objects

Model	Car		Chair	
	SSIM	L_1	SSIM	L_1
Ours	.902	.121	.897	.178
Ours (w/o TD)	.861	.187	.871	.261
M2N	.923	.098	.895	.181
TVSN	.913	.119	.894	.230
AFN	.877	.148	.891	.240
MV3D	.875	.139	.895	.248

Smaller numbers are better for L_1 and higher numbers are better for SSIM. We performed ablation experiment with and without task-divided decoder (TD) and compared with other methods
 Bold values indicate the best performance

5.1 Experiment on rigid objects

Rigid object experiment We have experimented with novel 3D view synthesis tasks given a single view of an object with an arbitrary pose. The goal of this experiment is to synthesize an image of the object after a specified transformation or change in viewpoint has been applied to the original view. To evaluate our method in the context of rigid objects, we have performed a collection of tests on the chair and car datasets. Given a single input view of an object, we leverage the LTNN model to produce 360° views of the object. We have tested our model's ability to perform 360° view estimation on 3D objects and compared the results with the other state-of-the-art methods. The models are trained on the same dataset used in M2N [38]. The car and chair categories from the ShapeNet [3] 3D model repository have been rotated horizontally 18 times by 20° along with elevation changes of 0° , 10° , and 20° . The M2N and TVSN results are slightly better for the car category; however, these works have incorporated skip connections between the encoder layers and decoder layers, proposed in U-net [35], which substantially increases the computational demand for these networks (see Table 2). As can be seen in Tables 2 and 3, the proposed model is compa-

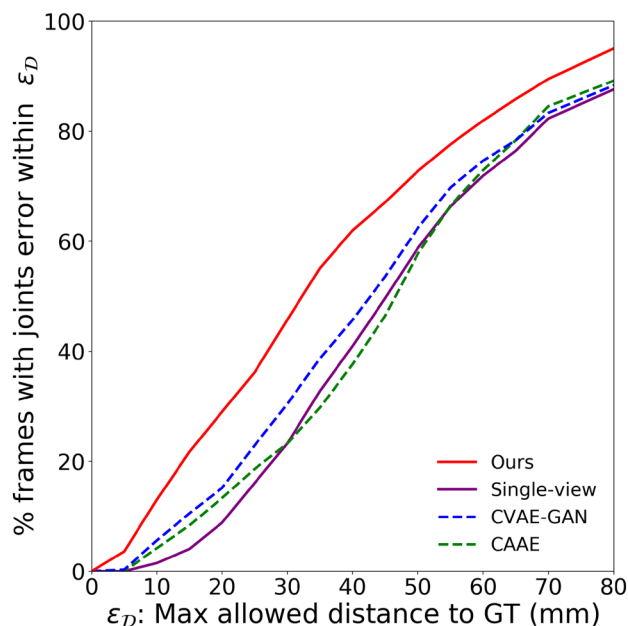
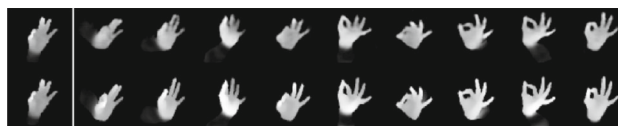
**Fig. 7** Quantitative evaluation for multi-view hand synthesis using the real NYU dataset**Fig. 8** Comparison of CVAE-GAN (top) with the proposed LTNN model (bottom) using the noisy NYU hand dataset [41]. The input depth-map hand pose image is shown to the far left, followed by the network predictions for nine synthesized view points. The views synthesized using LTNN are seen to be sharper and also yield higher accuracy for pose estimation (see Fig. 11)

table with existing models specifically designed for the task of multi-view prediction while requiring the least FLOPs for inference compared with all other methods. The low computational cost of the LTNN model highlights the efficiency of the CTU/CDU framework for incorporating conditional information into the network for view synthesis. Moreover, as seen in the qualitative results provided in Fig. 6, using a task-divided decoder helps to eliminate artifacts in the generated views; in particular, the spokes on the back of the chair and the spoiler on the back of the car are seen to be synthesized much more clearly when using a task-divided decoder.

5.2 Experiment on non-rigid objects

Hand pose experiment To assess the performance of the proposed network on non-rigid objects, we consider the problem of hand pose estimation. As the number of available view points of a given hand is increased, the task of estimating the associated hand pose becomes significantly easier [14]. Motivated by this fact, we synthesize multiple views of a

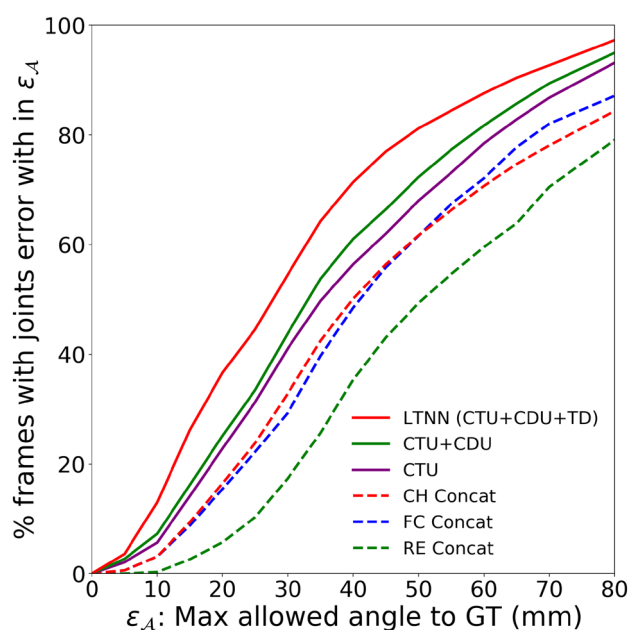


Fig. 9 LTNN ablation experiment results and comparison with alternative conditioning frameworks using synthetic hand dataset. Our models: conditional transformation unit (CTU), conditional discriminator unit (CDU), task-divide decoder (TD), and LTNN consisting of all previous components. Alternative concatenation methods: channel-wise concatenation (CH Concat), fully connected concatenation (FC Concat), and reshape fully connected feature vector concatenation (RE Concat)

hand given a single view and evaluate the accuracy of the estimated hand pose using the synthesized views. The underlying assumption of the assessment is that the accuracy of the hand pose estimation will be improved precisely when the synthesized views provide faithful representations of the true hand pose. Since ground truth predictions for the real NYU hand dataset were not available, the LTNN model has been trained using a synthetic dataset generated using 3D mesh hand models. The NYU dataset does, however, provide ground truth coordinates for the input hand pose; using this, we were able to indirectly evaluate the performance of the model by assessing the accuracy of a hand pose estimation method using the network's multi-view predictions as input. More specifically, the LTNN model was trained to generate nine different views which were then fed into the pose estimation network from Choi et al. [6] (also trained using the synthetic dataset). For an evaluation metric, the maximum error in the predicted joint locations has been computed for each frame (i.e., each hand pose in the dataset). The cumulative number of frames with maximum error below a threshold distance ϵ_D has then been computed, as is commonly used in hand pose estimation tasks [6,29]. A comparison of the pose estimation results using synthetic views generated by the proposed model, the CVAE-GAN model, and the CAAE model are presented in Fig. 7, along with the results obtained by performing pose estimation using the single-view input frame



Fig. 10 Qualitative evaluation for view synthesis of real faces using the image dataset [9]

alone. In particular, for a threshold distance $\epsilon_D = 40$ mm, the proposed model yields the highest accuracy with 61.98% of the frames having all predicted joint locations within a distance of 40 mm from the ground truth values. The second highest accuracy is achieved with the CVAE-GAN model with 45.70% of frames predicted within the 40 mm threshold.

A comparison of the quantitative hand pose estimation results is provided in Fig. 7 where the proposed LTNN framework is seen to provide a substantial improvement over existing methods; qualitative results are also available in Fig. 8. Ablation study results for assessing the impact of individual components of the LTNN model are also provided in Fig. 9; in particular, we note that the inclusion of the CTU, CDU, and task-divided decoder each provides significant improvements to the performance of the network. With regard to real-time applications, the proposed model runs at 114 fps without batching and at 1975 fps when applied to a mini-batch of size 128 (using a single TITAN Xp GPU and an Intel i7-6850K CPU).

Real face experiment We have also conducted an experiment using a real face dataset to show the applicability of LTNN for real images. The stereo face database [9], consisting of images of 100 individuals from 10 different viewpoints, was used for experiments with real faces. These faces were first segmented using the method of [28], and then we manually cleaned up the failure cases. The cleaned faces have been cropped and centered to form the final dataset. The LTNN model was trained to synthesize images of input faces corresponding to three consecutive horizontal rotations. Qualitative results for the real face experiment are provided in Fig. 10; in particular, we note that the quality of the views generated by the proposed LTNN model is consistent for each of the four views, while the quality of the views generated

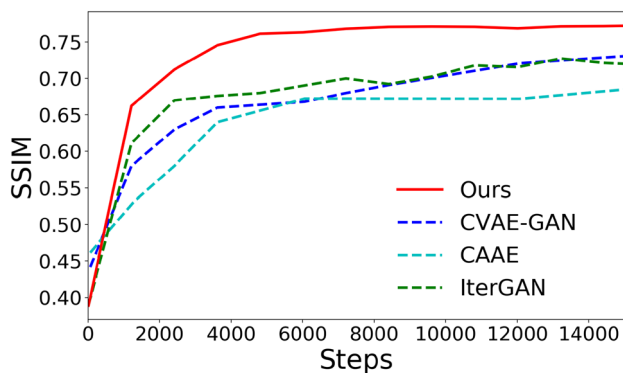


Fig. 11 Quantitative evaluation with SSIM of model performances for experiment on the real face dataset [9]. Higher values are better

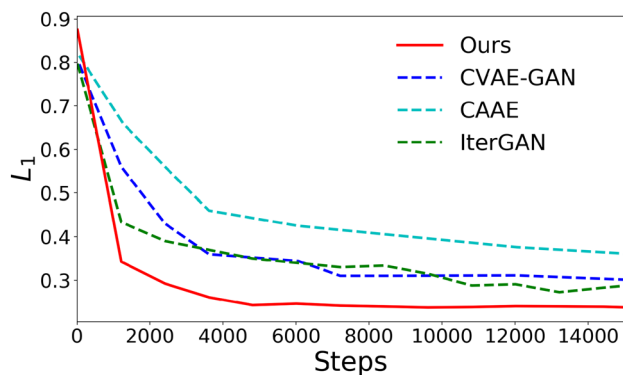


Fig. 12 Quantitative evaluation with L_1 of model performances for experiment on the real face dataset [9]. Lower values are better

using other methods decreases substantially as the change in angle is increased. This illustrates the advantage of using CTU mappings to navigate the latent space and avoid the accumulation of errors inherent to iterative methods. Moreover, as shown in Figs. 11 and 12, the LTNN model provides

substantial improvements to alternative methods with respect to the SSIM and L_1 metrics and converges much faster as well.

5.3 Diverse attribute exploration

To evaluate the proposed framework’s performance on a more diverse range of attribute modification tasks, a synthetic face dataset and other conditional generative models, CVAE-GAN and CAAE, with comparable hourglass structures to the LTNN model have been selected for comparison. The generated images from the LTNN model are available in Fig. 13. These models have been trained to synthesize discrete changes in elevation, azimuth, light direction, and age from a single image. As shown in Tables 4 and 5, the LTNN model outperforms the CVAE-GAN and CAAE models by a significant margin in both SSIM and L_1 metrics; additional quantitative results are provided in Table 1, along with a collection of ablation results for the LTNN model.

Multiple attributes can also be modified simultaneously using LTNN by composing CTU mappings. For example, one can train four CTU mappings $\{\Phi_k^{light}\}_{k=0}^3$ corresponding to incremental changes in lighting and four CTU mappings $\{\Phi_k^{azim}\}_{k=0}^3$ corresponding to incremental changes in azimuth. In this setting, the network predictions for lighting and azimuth changes correspond to the values of $\text{Decode}[\Phi_k^{light}(l_x)]$ and $\text{Decode}[\Phi_k^{azim}(l_x)]$, respectively (where l_x denotes the encoding of the original input image). To predict the effect of simultaneously changing both lighting and azimuth, we can compose the associated CTU mappings in the latent space; that is, we may take our network prediction for the lighting change associated with Φ_i^{light} combined with the azimuth change associated with Φ_j^{azim} to be:

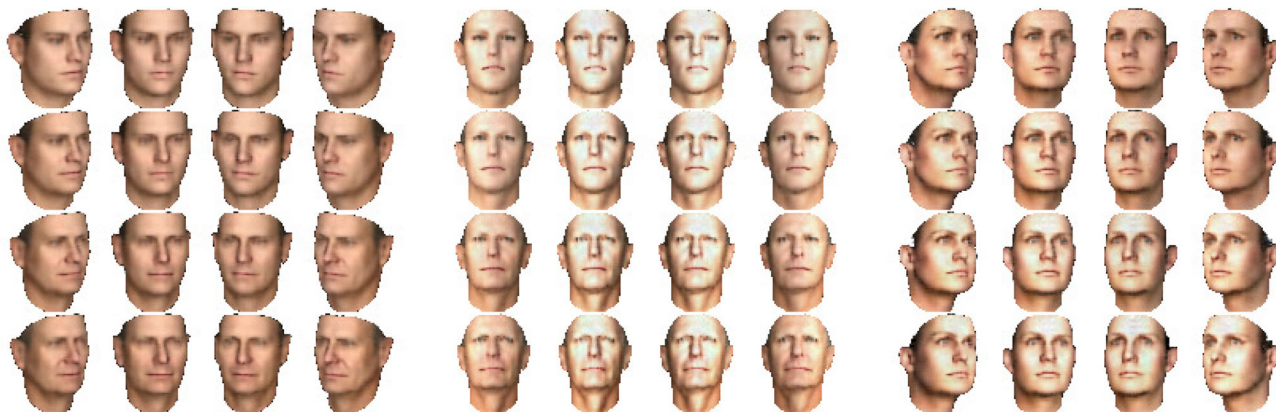


Fig. 13 Simultaneous learning of multiple attribute modifications. Azimuth and age (left), light and age (center), and light and azimuth (right) combined modifications are shown. The network has been trained

using four CTU mappings per attribute (e.g., four azimuth mappings and four age mappings); results shown have been generated by composing CTU mappings in the latent space and decoding

Table 4 Quantitative results for light direction and age modification on the synthetic face dataset

Model	Light direction		Age	
	SSIM	L_1	SSIM	L_1
Ours	.941	.093	.925	.102
CVAE-GAN	.824	.209	.848	.166
CAAE	.856	.270	.751	.207

Bold values indicate the best performance

Table 5 Quantitative results for azimuth and elevation modification on the synthetic face dataset

Model	Elevation		Azimuth	
	SSIM	L_1	SSIM	L_1
Ours	.923	.107	.923	.108
CVAE-GAN	.864	.158	.863	.180
CAAE	.777	.175	.521	.338

Bold values indicate the best performance

$$\hat{y} = \text{Decode}[\hat{l}_y] \quad \text{where} \\ \hat{l}_y = \Phi_i^{\text{light}} \circ \Phi_j^{\text{azim}}(l_x) = \Phi_i^{\text{light}}[\Phi_j^{\text{azim}}(l_x)]. \quad (9)$$

5.4 Near-continuous attribute modification

Near-continuous attribute modification is also possible within the proposed framework; this can be performed by a simple, piecewise-linear interpolation procedure in the latent space. For example, we can train nine CTU mappings $\{\Phi_k\}_{k=0}^8$ corresponding to incremental 7° changes in elevation $\{\theta_k\}_{k=0}^8$. The network predictions for an elevation change of $\theta_0 = 0^\circ$ and $\theta_1 = 7^\circ$ are then given by the values $\text{Decode}[\Phi_0(l_x)]$ and $\text{Decode}[\Phi_1(l_x)]$, respectively (where l_x denotes the encoding of the input image). To predict an elevation change of

3.5° , we can perform linear interpolation in the latent space between the representations $\Phi_0(l_x)$ and $\Phi_1(l_x)$; that is, we may take our network prediction for the intermediate change of 3.5° to be:

$$\hat{y} = \text{Decode}[\hat{l}_y] \quad \text{where} \quad \hat{l}_y = 0.5 \cdot \Phi_0(l_x) + 0.5 \cdot \Phi_1(l_x). \quad (10)$$

More generally, we can interpolate between the latent CTU map representations to predict a change θ via:

$$\hat{y} = \text{Decode}[\hat{l}_y] \quad \text{where} \quad \hat{l}_y = \lambda \cdot \Phi_k(l_x) + (1-\lambda) \cdot \Phi_{k+1}(l_x), \quad (11)$$

where $k \in \{0, \dots, 7\}$ and $\lambda \in [0, 1]$ are chosen so that $\theta = \lambda \cdot \theta_k + (1-\lambda) \cdot \theta_{k+1}$. In this way, the proposed framework naturally allows for continuous attribute changes to be approximated while only requiring training for a finite collection of discrete changes. Qualitative results for near-continuous attribute modification on the synthetic face dataset are provided in Fig. 14; in particular, we note that views generated by the network effectively model gradual changes in the attributes without any noticeable degradation in quality. This highlights the fact that the model has learned a smooth latent space structure which can be navigated effectively by the CTU mappings while maintaining the identities of the original input faces.

6 Conclusion

In this work, we have introduced an effective, general framework for incorporating conditioning information into inference-based generative models. We have proposed a modular approach to incorporating conditioning information

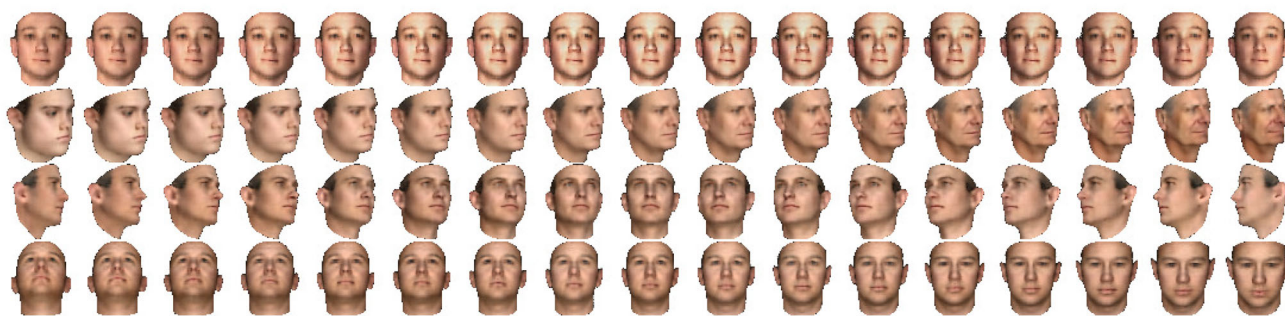


Fig. 14 Near-continuous attribute modification is attainable using piecewise-linear interpolation in the latent space. Provided a grayscale image (corresponding to the faces on the far left), modified images corresponding to changes in light direction (first), age (second), azimuth (third), and elevation (fourth) are produced with 17 degrees of varia-

tion. These attribute-modified images have been produced using nine CTU mappings, corresponding to varying degrees of modification, and linearly interpolating between the discrete transformation encodings in the latent space

using CTUs and a consistency loss term, defined an efficient task-divided decoder setup for deconstructing the data generation process into manageable subtasks, and shown that a context-aware discriminator can be used to improve the performance of the adversarial training process. The performance of this framework has been assessed on a diverse range of tasks and shown to perform comparably with the state-of-the-art methods while reducing computational operations and memory consumption.

Acknowledgements Karthik Ramani acknowledges the US National Science Foundation Awards NRI-1637961 and IIP-1632154. Guang Lin acknowledges the US National Science Foundation Awards DMS-1555072, DMS-1736364 and DMS-1821233. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency. We gratefully appreciate the support of NVIDIA Corporation with the donation of GPUs used for this research.

References

- Antipov, G., Baccouche, M., Dugelay, J.L.: Face aging with conditional generative adversarial networks (2017). arXiv preprint [arXiv:1702.01983](https://arxiv.org/abs/1702.01983)
- Bao, J., Chen, D., Wen, F., Li, H., Hua, G.: CVAE-GAN: fine-grained image generation through asymmetric training (2017). arXiv preprint [arXiv:1703.10155](https://arxiv.org/abs/1703.10155)
- Chang, A., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: An information-rich 3D model repository. *I(7)*, 8 (2015). arXiv preprint [arXiv:1512.03012](https://arxiv.org/abs/1512.03012)
- 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, Stanford University—Princeton University—Toyota Technological Institute at Chicago (2015). [arXiv:1512.03012](https://arxiv.org/abs/1512.03012) [cs.GR]
- Chen, X., Duan, Y., Houthoofd, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2172–2180 (2016)
- Choi, C., Kim, S., Ramani, K.: Learning hand articulations by hallucinating heat distribution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3104–3113 (2017)
- Dinerstein, J., Egbert, P.K., Cline, D.: Enhancing computer graphics through machine learning: a survey. *Vis. Comput.* **23**(1), 25–43 (2007)
- Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1538–1546 (2015)
- Fransens, R., Strehla, C., Van Gool, L.: Parametric stereo for multi-pose face recognition and 3D-face modeling. In: International Workshop on Analysis and Modeling of Faces and Gestures, pp. 109–124. Springer (2005)
- Galama, Y., Mensink, T.: Iterative GANs for rotating visual objects (2018)
- Gauthier, J.: Conditional generative adversarial nets for convolutional face generation. In: Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition. Winter Semester 2014(5), 2 (2014)
- Ge, L., Liang, H., Yuan, J., Thalmann, D.: Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3593–3601 (2016)
- Goodfellow, I.J.: NIPS 2016 Tutorial: Generative Adversarial Networks (2017). CoRR [arXiv:1701.00160](https://arxiv.org/abs/1701.00160)
- Guan, H., Chang, J.S., Chen, L., Feris, R.S., Turk, M.: Multi-view appearance-based 3D hand pose estimation. In: 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06), pp. 154–154. IEEE (2006)
- He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
- Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks (2016). arXiv preprint [arXiv:1608.06993](https://arxiv.org/abs/1608.06993)
- IEEE: A 3D Face Model for Pose and Illumination Invariant Face Recognition (2009)
- Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
- Jason, J.Y., Harley, A.W., Derpanis, K.G.: Back to basics: unsupervised learning of optical flow via brightness constancy and motion smoothness. In: Computer Vision—ECCV 2016 Workshops, pp. 3–10. Springer (2016)
- Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in Neural Information Processing Systems, pp. 667–675 (2016)
- Kim, S., Kim, D., Choi, S.: Citycraft: 3D virtual city creation from a single image. *Vis. Comput.* (2019). <https://doi.org/10.1007/s00371-019-01701-x>
- Kingma, D., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
- Kingma, D.P., Welling, M.: Auto-encoding variational Bayes (2013). arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114)
- Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: Advances in Neural Information Processing Systems, pp. 2539–2547 (2015)
- Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders (2015). arXiv preprint [arXiv:1511.05644](https://arxiv.org/abs/1511.05644)
- Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). arXiv preprint [arXiv:1411.1784](https://arxiv.org/abs/1411.1784)
- Miyato, T., Koyama, M.: cGANs with projection discriminator (2018). arXiv preprint [arXiv:1802.05637](https://arxiv.org/abs/1802.05637)
- Nirkin, Y., Masi, I., Tuan, A.T., Hassner, T., Medioni, G.: On face segmentation, face swapping, and face perception. In: 2018 13th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2018), pp. 98–105. IEEE (2018)
- Oberweger, M., Lepetit, V.: Deepprior++: improving fast and accurate 3D hand pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 585–594 (2017)
- Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: Transformation-grounded image generation network for novel 3D view synthesis. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 702–711. IEEE (2017)
- Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: a deep neural network architecture for real-time semantic segmentation (2016). arXiv preprint [arXiv:1606.02147](https://arxiv.org/abs/1606.02147)
- Ramachandran, P., Zoph, B., Le, Q.V.: Swish: a self-gated activation function (2017). arXiv preprint [arXiv:1710.05941](https://arxiv.org/abs/1710.05941)
- Reed, S., Sohn, K., Zhang, Y., Lee, H.: Learning to disentangle factors of variation with manifold interaction. In: International Conference on Machine Learning, pp. 1431–1439 (2014)
- Rezende, D.J., Eslami, S.A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N.: Unsupervised learning of 3D structure from images.

- In: *Advances in Neural Information Processing Systems*, pp. 4996–5004 (2016)
35. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer (2015)
 36. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: *Advances in Neural Information Processing Systems*, pp. 2234–2242 (2016)
 37. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: *Advances in Neural Information Processing Systems*, pp. 3483–3491 (2015)
 38. Sun, S.H., Huh, M., Liao, Y.H., Zhang, N., Lim, J.J.: Multi-view to novel view: Synthesizing novel views with self-learned confidence. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 155–171 (2018)
 39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (2015)
 40. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3D models from single images with a convolutional network. In: *European Conference on Computer Vision*, pp. 322–337. Springer (2016)
 41. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Gr.* **33**(5), 169 (2014)
 42. Varley, J., DeChant, C., Richardson, A., Ruales, J., Allen, P.: Shape completion enabled robotic grasping. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2442–2447. IEEE (2017)
 43. Wang, Q., Artières, T., Chen, M., Denoyer, L.: Adversarial learning for modeling human motion. *Vis. Comput.* (2018). <https://doi.org/10.1007/s00371-018-1594-7>
 44. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**(4), 600–612 (2004)
 45. Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: conditional image generation from visual attributes. In: *European Conference on Computer Vision*, pp. 776–791. Springer (2016)
 46. Zhang, S., Han, Z., Lai, Y.K., Zwicker, M., Zhang, H.: Stylistic scene enhancement GAN: mixed stylistic enhancement generation for 3D indoor scenes. *Vis. Comput.* **35**(6–8), 1157–1169 (2019)
 47. Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5810–5818 (2017)
 48. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: *European Conference on Computer Vision*, pp. 286–301. Springer (2016)



Sangpil Kim is a Ph.D. student in the Electrical and Computer Engineering at Purdue University. He received his B.S. degree from the Korea University, South Korea, in 2015. His current research interests are in computer vision and deep learning.



Nick Winovich is a Ph.D. candidate in the Department of Mathematics at Purdue University. He earned his B.A. in Mathematics and Spanish at the University of Notre Dame in 2012 and subsequently received an M.S. in Mathematics at the University of Oregon in 2015. His current research focuses on the intersection of probability theory and numerical computing, with an emphasis on applications of Gaussian processes and neural network models for partial differential equations.



Hyung-gun Chi is a Master's student in Mechanical Engineering at Purdue University. He received his B.S. degree from the school of Mechanical Engineering, Yonsei University, South Korea, in 2017. His current research interests lie at the intersection of computer vision and robotics.



Guang Lin is the Director of Purdue Data Science Consulting Service, Associate Professor of Mathematics and School of Mechanical Engineering at Purdue University, with courtesy appointment in Statistics. He earned his B.S. in mechanics from Zhejiang University in 1997, and an M.S. and a Ph.D. from Brown University in 2004 and 2007 in applied mathematics, respectively. He received many awards from the National Science Foundation and other organizations. He has

served as the Associate Editor of *SIAM Multiscale Modeling and Simulations* and in the editorial board of many International Journals. In 2019, he received University Faculty Scholars from Purdue University.



Karthik Ramani is the Donald W. Feddersen Professor of School of Mechanical Engineering at Purdue University, with courtesy appointments in Electrical and Computer Engineering and College of Education. He earned his B.Tech from the Indian Institute of Technology, Madras, in 1985, an M.S. from Ohio State University, in 1987, and a Ph.D. from Stanford University in 1991, all in Mechanical Engineering. His research interests are in collaborative intelligence, human-machine

interactions, spatial interfaces, deep shape learning, and manufacturing productivity. He has published recently in *ACM [CHI & UIST]*, *IEEE [CVPR, ECCV, ICCV]*, *ICLR*, *ICRA*, *Scientific Reports*, and *ASME JMD*.