Cross-View Action Synthesis

Kara Schatz
Xavier University
schatzk@xavier.edu

Dr. Yogesh Rawat
University of Central Florida
yogesh@crcv.ucf.edu

Dr. Mubarak Shah
University of Central Florida
shah@crcv.ucf.edu

Abstract

This paper addresses a relatively new task in Computer Vision: Cross-View Video Synthesis, in which a video is synthesized from an unseen viewpoint. We propose a deep learning framework for Cross-View Action Synthesis, which simply restricts the task to videos that contain some action. Given an input video that comtains the desired action and an input frame that contains the desired appearance from a different viewpoint, our framework generates a video that depicts both this action and this appearance by extracting and transforming motion and appearance features from these inputs, respectively. Our framework consists of 5 major components: a Representation Transformer to transform the motion features according to the viewpoint change, an Appearance Transformer to transform the appearance features according to the action occurring, a Keypoint Predictor to predict action keypoints, a Viewpoint Predictor to estimate the viewpoint change and facilitate motion feature transformation, and a Generator to synthesize a video from the desired viewpoint of the desired action and appearance. We demonstrate the effectiveness of our model through various experiments conducted on the NTU-RGB+D Dataset. We show that our framework can be used not only to synthesize the same video from an unseen viewpoint, but also that an entirely different background scene or actor can be synthesized performing the given action. This task has little to no prior research, so our framework and results set the standard for this problem.

1. Introduction

In recent years, much progress has been made in video synthesis. Mostly in regards to future frame prediction and conditioned image generation. Additionally, cross-view image synthesis is a task that has been explored in over the past few years. The method and architecture proposed here seeks to accomplish cross-view video synthesis, which lies at the intersection of these two tasks. While both tasks have been explored separately, very little has been done in regards to cross-view video synthesis, which involves reconstructing an input video from an unseen viewpoint. It is fairly simple for a human to imagine how an action would look from a different viewpoint, so we seek to apply computer vision to techniques that will allow a machine to do the same.

Our task is to synthesize an input action video from a different, unseen viewpoint. To do so we will also use appearance conditioning in the form of an image of the unseen view, and viewpoint conditioning in the form of an angular viewpoint change. Once trained, the network can predict the viewpoint change; it can thus generate the output video of the desired view given only the input video an image for appearance conditioning. We use a L2 reconstruction loss with the synthesized video from each view. We train the model on 2 views at once, which allows for a consistency loss between features from the two viewpoints to train our network in a self-supervised manner. Aside from this consistency loss, our model makes use of Gaussian Keypoints to pinpoint the areas of action, and it utilizes two transformation networks to transform the action according to the angluar viewpoint shift and the appearance according to the action representation.

2. Related Work

2.1. Cross-View Image Synthesis

Cross-view image synthesis, a task in which a single image is reconstructed from an unseen viewpoint, has been adressed in three main papers recently. The first of these is [3], which uses a conditional generative adversarial network structure to achieve their results. Our proposed method aims to solve a similar cross-view task, but we do so without adversarial loss. In [1], they address the taks while focusing on scene representation. They learn a representation of a given scene from multiple viewpoints, and use that to generate a single unseen viewpoint. Our process is similar in that we learn a representation from another viewpoint and use it to construct the desired viewpoint, but we rely on only a single known view to do so. Finally, in [4], a similar probelm is addressed, but they focus on using two more distinct viewpoints than the others. Namely, they use one ground

view and one aerial view. A conditional GAN approach is use here as well, but there is a focus on utilizing the scene geometry to facilitate the reconstruction. Our process uses neither a GAN nor a geometry-focused approach. However, we do perform keypoint extraction to improve the quality of our generated videos. While our method also adresses cross-view synthesis, we do so with videos and not single images as in each of these works.

2.2. Keypoint Extraction

Our propose method uses keypoint extraction from metion features in order to create good quality reconstructed videos. Many prior works have use similar extraction of keypoints to localize important features as well. There are two main works that perform keypoint extraction in a similar manner. The first is [2], in which their task is to learn important object landmarks, or keypoints. They achieve this through conditional image generation in which they impose the geometry of a given image onto the appearance of another image to generate an image with the appearance of the second image but with the shape or view of the first image. To do this, they extract appearance features from the target image and then extract Gaussian heatmaps of the most highly activate points, which estimate the location of the keypoints. While our method uses the same extraction process, we do so for extracted motion features of video sequences. The other work that addresses utilizes a similar keypoint extraction is [6], which aims to impose the action from a driving video onto the appearance of a source image. This work also perform keypoint extraction on video features and performs a similar task to ours, but the key difference is that our driving videos and source images are restricted in that they must be taken from different camera angles. In [6], they do not have this restriction, and thus viewpoint shift is not necessary as it is for our method.

3. Method

The architecture of the proposed method is shown in Figure 1. Each component will be broken down and described in detail in this section. The overall approach is to use the motion features extracted from the input video and the appearance features extracted from the image of the desired view to generate the desired video. The motion features provide all the information needed to reconstruct the action, while the appearance features provide appearance conditioning so the generator is capable of generating a view it has not seen. We transform both feature sets; the motion features are transformed according to the viewpoint change, which the network predicts, while the appearance features are transformed according to the motion that has occurred. We also extract keypoints of the motion features to localize areas of the frame where motion is occurring. Each of these pieces allows the network to learn a better representation of the appearance and the action so that it can generate a video of higher quality.

For training, the network requires 3 inputs: the input video, v_g, which is from the given view, the conditioning image, i_d, which is from the desired view, and the angular viewpoint change, θ. It outputs, g_d, which is the generated video that displays the action from the given view and the appearance from the deisred view. The video, v_d, from the desired view is the ground truth for this reconstruction. It also predicts the angular viewpoint change, $\hat{\theta}$. Thus, the network is trained based on the reconstruction loss, L_r, between g_d and v_d as well as the prediction loss, L_p, between θ and $\hat{\theta}$. For testing and other use, $\hat{\theta}$ is used for all computations, so that the input θ is not necessary. Therefore, once the network is trained, the desired video can be generated with just an input video v_g and appearance conditioning i_d.

It is also important to note that our implementation of the complete model includes skip connections from both I3D and VGG-16. Motion features are used at temporal and spatial sizes of 8x56x56, 8x28x28, and 4x14x14; appearance features are used at spatial sizes of 56x56, 28x28. and 14x14. Each of these feature sets are then transformed according to their respective transformer networks. The transformed motion features of various sizes are concatenated in during the upsampling process in the Keypoint Predictor; the transformed appearance features of various sizes are concatenated in during the upsampling process of the Generator network.

3.1. Feature Extraction

The first step of the network is to extract features from the input video and image so that comprehensive representations can be learned.

To extract the motion features or action representation, r, we use a modified version of the state of the art Inception I3D network, I. We first modifiy I3D by removing the final classificaiton layers as our approach does not require any action classification. Instead, we want to output a feature map from the video, so we also modify the max pooling layers to get the desired feature size of 4x14x14x256 as the final output of this network. These features are later used to compute the viewpoint-transformed action representation, r', the predicted viewpoint, $\hat{\theta}$, and the video keypoints, k.

To extract the appearance features, a, we use a modified version of the state of the art VGG-16, V. Again, as our method does not require any classification, we remove the final classification layers of VGG-16 as well. We also remove some of the convolutional layers that have larger feature sizes since our method does not require as many features. Finally, we remove some of the max pooling layers to achieve our desired feature size of 14x14x256 as the final output of the network. These features are later used to compute the action-transformed appeareance, a', and the

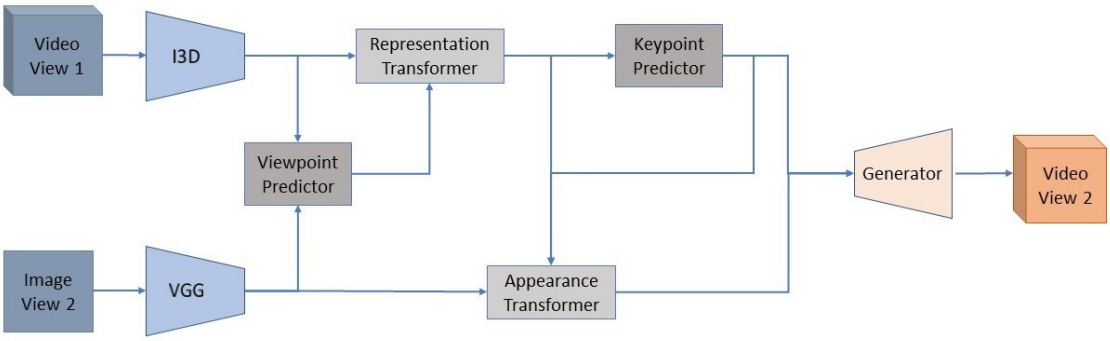

Figure 1. Diagram of the full architecture proposed. This includes all network components, but does not demonstrate the use of skip connections. Note that the angluar viewpoint change is not shown as an input here because it is predicted during testing and not needed as an input.

predicted viewpoint, $\hat{\theta}$.

3.2. Viewpoint Predictor

The Viewpoint Predictor component is a network, P, that predicts the angular viewpoint change. This component allows the network to be tested without requiring the ground truth θ as input. It takes as input the action representation, r, and the appearance features, a. It uses these to predict a single value, $\hat{\theta}$ representing the angular viewpoint change between the two.

This network can be formally defined by:

$$\hat{\theta} = P(r, a) \tag{1}$$

First, average pooling is performed on r to shrink it down to a single frame; it originally has four frames of features. This shrinked r is then stacked with a along the channel dimension for the remaining computations. These concatenated features are passed through two blocks that each consist of a 2D convolutional layer followed by ReLU activation and average pooling; the number of features decreases with each block. A 3x3 kernel is used for the convolutional layers, and the spatial feature size does not change. For the average pooling layers, a 2x2 kernel is used with a stride of 2 to halve the spatial feature size. Finally, the features are flattened and passed through a single fully connected layert that has only one output channel for the predicted angular viewpoint change, $\hat{\theta}$, is later used to produce the viewpoint-transformed action representation, r'.

This introduces our prediction loss, L_p, which is computed as the mean squared error between the ground truth viewpoint change, θ, and the predicted viewpoint change, $\hat{\theta}$.

$$L_p = \frac{1}{N}\sum_{n}^{N}(\hat{\theta} - \theta)^2 \tag{2}$$

Here, N represents the number of samples.

3.3. Representation Transformer

The Representation Transformer is a network, T_r, that computes motion features for the desired view, r', by transforming the action representation, r, of the given view based on the angular viewpoint change, θ or $\hat{\theta}$. Note that the ground truth value θ is used during training whereas the predicted value $\hat{\theta}$ is used otherwise.

This network can be formally defined by:

$$r' = T_r(r, \hat{\theta}) \tag{3}$$

First, $\hat{\theta}$ is expanded to the same temporal and spatial size as r by repeating $\hat{\theta}$ as each entry in the tensor. Then, it is passed through two layers of 3D convolutions before being stacked with r along the channel dimension. These concatenated features are then passed through three 3D convolutional layers each followed by ReLU activation. A 3x3 kernel is used for each of the convolutional layers and the temporal and spatial dimensions remain constant throughtout the entire network. The final convolutional layer has the same number of output channels as the input representation r. Thus, producing the transformed action representation, r', which is the same size as the original. r' is later used to compute the action-transformed appearance features, a'.

3.4. Keypoint Predictor

The Keypoint Predictor, K, extracts the locations of the action keypoints in v_g. It uses the transformed motion features, r', to produce a set of Gaussian heatmaps, k, that predict these keypoint locations.

This network can be formally defined by:

$$k = K(r') \tag{4}$$

First, r' is passed through a series of three blocks that each consist of a 3D convolutional layer, a ReLU activation, and an upsampling layer. Then, there is a fourth block consisting of a single 3D convolutional layer and Sigmoid activation. The three blocks each output the same number of channels, while the fourth outputs n channels, where n is the number of keypoints to predict. Each of the convolutions use a 3x3 kernel and maintain the same input and output size in the temporal and spatial dimensions. The upsampling layers all use trilinear interpolation and expand the temporal or spatial dimensions (or both) of the feature map by a factor of 2. The final feature size after these four blocks is 16x56x56xn, where there are n channels for the n keypoints, 16 frames, and a height and width of 56.

Then, the keypoints are extracted from these features as Gaussian heatmaps. The x and y coordinates for the keypoints are deteremined separately by first computing the mean along all the rows or columns and eliminating the flattened dimension. Then, we compute a softmax along the remaining spatial dimension, which allows the network to determine the most highly activated point for each channel and frame, and uses these coordinates as the x or y coordinates of the Gaussian mean. Then, this computation is repeated to compute the other coordinate. Thus, the Gaussian means correspond to the most highly activated points in the feature maps, so they are used as the centers for the n heatmaps that are produced for each frame. This method for keypoint extraction has been previously used by [2] and [6]. The predicted keypoints that are extracted, k, are later used in the appearance transformer as well as the video generator for reconstruction.

3.5. Appearance Transformer

The Appearance Transformer network, T_a, transforms the appearance features, a, according to the action occurring in the video. It uses the transformed action representation, r', to inform the network about how the appearance should be transformed and the keypoints, k to better localize the action and know where appearance modification is necessary. This produces the action-transformed appearance features that are the output of this network, a'.

This network can be formally defined by:

$$a' = T_a(a, r', k) \tag{5}$$

The Appearance Transfomer is a standard convolutional Gated Recurrent Unit that uses a as the initalizer for the hidden state and the concatenation of r' and k in the channel dimension as the cell input. Note that k is first average pooled to match the temporal and spatial sizes of r' in order to allow for concatenation. A 7x7 kernel is used for the convolutions, which do not change the feature size at all. Thus, the output of each cell is a single frame of the transformed appearance features with the same number of channels and spatial size. The final network output is the concatenation of all these single frames along the frame dimenstion to produce a multi-frame action-transformed version of the appearance features, a'; it has the same number of frames as r'. These features are then used in the video generator for reconstruction.

3.6. Generator

The final component of the proposed method is the Generator Network, G, that reconstructs a video, g_d, that has the appearance of i_d and the action of v_g. It uses the transformed appearance features, a', along with the extracted keypoints, k, in order to perform this reconstruction.

This network can be formally defined by:

$$g_d = G(a', k) \tag{6}$$

First, k is average pooled down to the same temporal and spatial size as a', and then these two are stacked along the channel dimension. They are then passed through as series of three blocks that each consisting of two convolutional layers followed by ReLU activation and an upsampling layer. Then, they are passed through a fourth block consisting of two convolutional layers, the first followed by ReLU activation and the other followed by Sigmoid activation. The convolutional layers all use 3x3 kernels, do not alter the temporal or spatial sizes at all, and decrease the number of channels. The upsampling layers all use trilinear interpolation and expand the temporal or spatial dimensions (or both) of the feature map by a factor of 2. The final output, g_d, after the Sigmoid activation has the same dimenstions as the ground truth video, v_d.

This introduces our reconstruction loss, L_r, which is computed as the mean squared error between the ground truth video, v_d, and the reconstructed video, g_d.

$$L_r = \frac{1}{N}\sum_{n}^{N}\sum_{f}^{F}\sum_{h}^{H}\sum_{w}^{W}\sum_{c}^{C}(g_{d_{fhwc}} - v_{d_{fhwc}})^2 \tag{7}$$

Here, N represents the number of samples, F is the number of frames in the video clip, H is the height of the video frame, W is the width of the video frames, and $C = 3$ for the three color channels (RGB).

4. Experiements

In this section, we provide details of the experiments we conducted using the proposed method and the NTU-RGB+D Dataset [5]. We include qualitative results from several variations of our model.

4.1. Dataset

We conducted our experiments on the NTU-RGB+D Dataset, which is a large scale dataset containing over

56,000 videos spanning more than 4 million frames depicting either one or two humans performing actions. We extract all the frames, whcih are of size 240x135. There are a total of 60 different actions depicted in the dataset using 40 different actors. 3 different cameras are used at various height settings to capture videos from 80 different viewpoints. The cameras are always placed 45°apart, so they are at -45°, 0°, and +45°. Each actor or actor pair performs each action twice: once facing the left camera and once facing the right camera. This allows the videos to span viewpoints over a total of 90°. [25]

4.2. Training

For training, we use 2 randomly chosen views out of the 3 available (-45°, 0°, +45°) for each sample video provided in the dataset. We then crop 50 pixels off the left and right side of every frame to get a more central view of the actor or actors, which usually appear in the center of the frame. Then, we take a random 112x112 cropping of these frames to use for training in out model. Each sample is 16 frames long, extracted with a frame skip rate of 2 to capture more action occurring in the sample. We compute the groud truth angular viewpoint change based on confiuration parameters provided with the dataset. For our experiments, we use a cross-view split of the dataset by randomly selecting 2 of the 3 available views for each sample video. We also use a cross-actor split by choosing random views in the same way, but also randomly using different actors or scenes/backgrounds for one of the views so that the appearance of the desired video does not match that of the input video. This allows us to prove that our method is robust with respect to appearance changes, and the action and appearance are actually learned separately as proposed. All our experiments use the Adam optimizer, a learning rate of 1e-4, a standard deviation of 0.1 for computing Gaussians, and a batch size of 14. We implemented all our code in Pytorch.

4.3. Results

We conducted experiments on several variations of our proposed model. Here we will show qualitative results of the most interesting experiments that provided important information regarding which parts of our network had the biggest impact or were abolsutely necessary to achieve our highest level of video quality.

4.3.1 Reconstruction with Same Actor and Same Scene

Below is a list of the model variations with which we experimented. We refer to each by their additions onto the Basic Model, which uses the extracted motion and appearance features only; there is no feature transformation, no keypoint extraction, no viewpoint prediction, and no skip connections used at all. This model is shown in Figure 2. Note that each of these experiments was run with no randomization of the sample actor or scene; the input video actor and scene matches that of the output video, so the only change is the viewpoint.

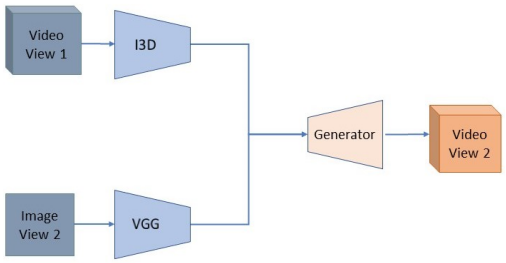

Figure 2. Diagram of the basic architecture, which includes none of the additional components added. It prodvides a baseline wtih which to compare our results as it is the most basic architecture following our approach of using an input video and appearance conditioning.

1. Basic Model
2. Basic Model with transformation of motion features only and skip connections of both appearance and motion features
3. Full Model

Results from each of these experiments can be seen in Figure 3, and we will give our own analysis here.

Version 1. This version was able to accurately perform the viewpoint transformation of the appearance, but no action appears in the output frames. In fact, all 16 output frames look virtually the same, and they match the input appearance conditioning very closely. This indicates that this model was only able to learn how to reconstruct this input frame. Additionally, it does a relatively poor job of this as the output frames are of lesser quality than the ground truth frames. Even the background details are not very clear despite the fact that these remain relatively constant throughout the full 16 frames of the sample. While this model does solve the viewpoint change part of the task, it has two major flaws in that it fails to maintain the action of the input video and it produces low quality images. We address these issues with model version 2.

Version 2. The most concerning issue with the first model was its inability to capture any significant motion information causing the reconstruction of still frames. By transforming the motion features according to the angular viewpoint change between in the input view and the desired view, this version is still able to accurately perform the viewpoint transformation, and maintain the action of the

input video, althought it is in low quality as the person becomes blurry as he/she moves. The Representation Transformer network allows the motion features to be learned properly for the desired viewpoint so that they can be reconstructed by the Generator network. This model version also improves the background quality significantly. By using skip connections from the VGG-16 feature extractor to the Generator network, the background features of the scene are rendered more clearly; having appearance features at a larger spatial size means that fewer details are lost during the downsampling and upsampling because they can be captured this way. In fact, the quality of the background improved so much that it is almost indistinguishable from the ground truth in these areas. The only area of the frame that suffers from blur in this version is the area in which the motion is occurring.

Version 3. Finally, we add transformation of the appearance features in this version as well as keypoint extraction and viewpoint prediction. To better improve the action quality, we transform the appearance features according to the action that is occurring. The goal is to utilize the Appearance Transformer network to essentially move the body of the person so that it follows the desired action. This will allow the generator to have information about the appearance of multiple different frames instead of solely the first frame. Adding keypoint extraction, allows the network to localize where the action is occurring. This is important as these are the only areas that really need to be modified and refined when generating new frames, since the background detail is already being generated well. These modifications did cause a significant decrease in the amount of motion blur occurring, but there is still some present. Thus, further modifications are necessary in order to minimize the motion blur.

4.3.2 Reconstruction with Different Actors and Scenes

Since the proposed method relies on an input image for appearance conditioning, it works even when the appearance of the input video and the appearance of the input image do not match. In other words, this appearance conditioning enables us to potentially impose the action of the input video onto another person in another place.

The results shown in Figure 4 are from testing with random actor and scene changes. We loaded weights from training with the same actor and scene, and then we ran the model on samples where the actor and/or scene could be different between the given view and the desired view. Note that we could not train on this configuration because we would not have ground truth videos with which to compute the loss since there are not two videos that are exactly the same action with different people. Thus, we load pretrained weights and only perform testing.

It turns out that the propsed method works very well on this configuration. It is able to reconstruct the video with the correct appearance from the input frame only and the correct action from the input video only. Thus, we know that each branch of our model (the motion branch and the appearance branch) is learning what it is supposed to learn; each only contributes information about motion or appearance appropriately. This reconstruction only works significantly well when the actors in the input video and input image are in a similar starting position, i.e. both standing or both sitting. In samples where one is sitting or one is standing the reconstruction is not as successful.

5. Conclusion and Future Work

In this work, we address the largely unexplored cross-view video synthesis problem, which aims to generate a video from an unseen camera viewpoint. We proposed a deep learning, convolutional based framework to solve this problem. Our proposed approach utilizes an input video that provides the action for the generated video and an input image that provides the appearance for the generated video. The model learns representations for both inputs and then transforms these representations to better suit the action and appearance of the desired output video. Finally, the model extracts action keypoints, which allows it to localize the action and better refine the output in these areas. We demonstrate the effectiveness of the proposed method in transforming the viewpoint and generating a logical video. We also recognize its shortcomings as the generated videos are low quality in the areas where the action occurs; there is significant motion blur that requires futher refining. The ability to render an unseen view allows the model to learn comprehensive feature transformation functions that can be useful for other tasks and expanded to more robust applications.

The proposed method lacks in certain areas as it experiences motion blur, as metioned above. Thus, we plan to modify the framework to minimize this issue. By enhancing the Appearance Transformation component of the framework, we should be able to create better quality frames that reduce the motion blur significantly. Using a modified mean squared error loss function that assigns a higher weight to pixels where motion is occurring will allow to network to focus on these areas and refine the appearance. Additionally, adding adversarial loss will allow the network to learn to produce better quality, realistic videos with significantly less motion blur.

While there are several improvements that can be made to the framework, it does a decent job of generating videos from an unseen view. This work is also one of very few to address this task, so it produces novel results that introduce a baseline upon which to improve.

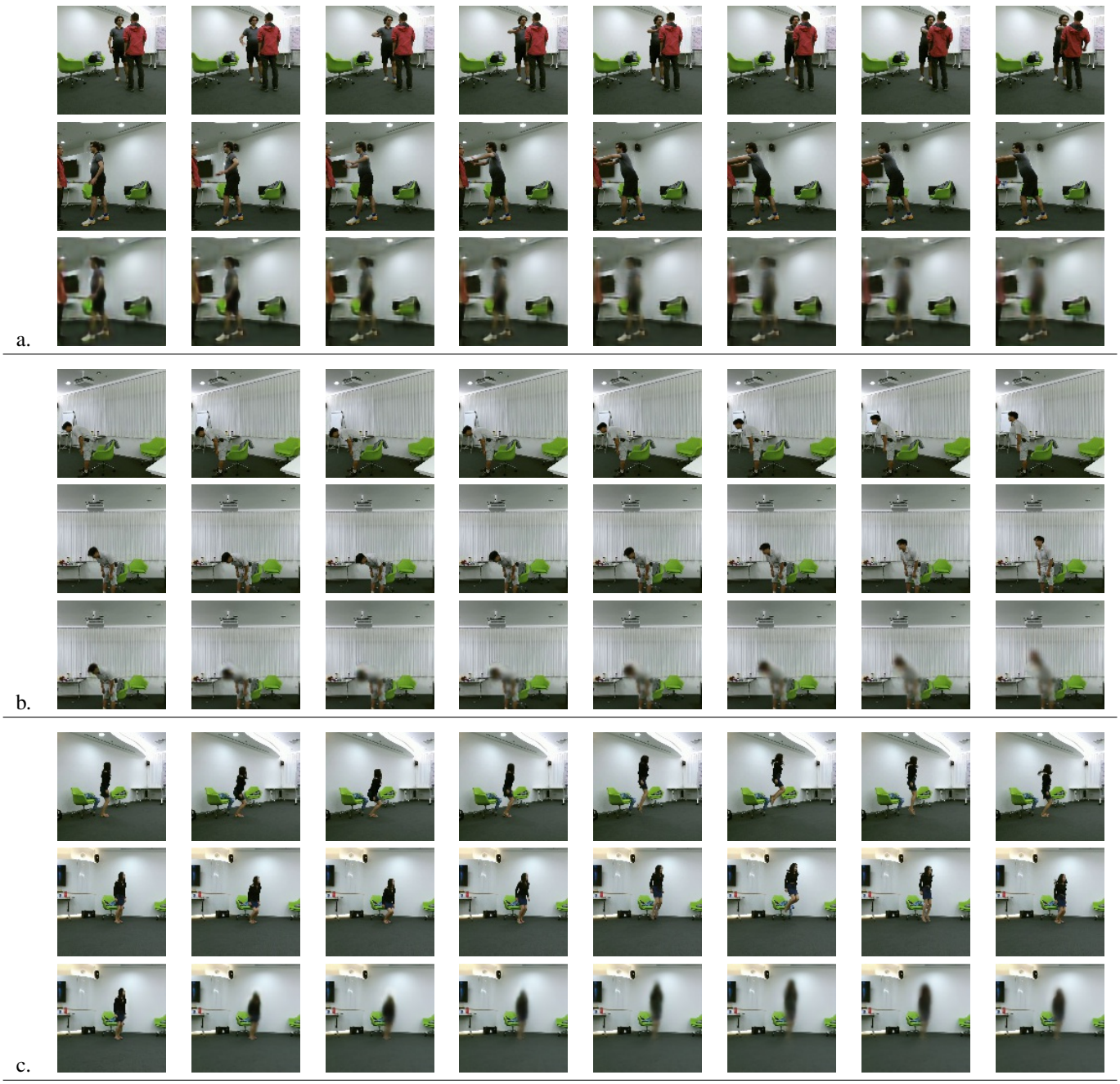

Figure 3. This displays the results for experiements run with the same actors and same scenes throughout. There are three different blocks for the three different model versions dicussed. In each, the top row contains 8 frames of the input video, the middle row contains the same 8 frames of the ground truth video, and the bottom row contains the same 8 frames of the recontructed video. For each of these, frames 1, 3, 5, 7, 9, 11, 13, and 15 are used. **a.** Model version 1 **b.** Model version 2 **c.** Model version 3

References

[1] S. M. A. E. et al. Neural scene representation and rendering. *Science Magazine*, 360, 2018.

[2] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Conditional image generation for learning the structure of visual objects. *CoRR*, abs/1806.07823, 2018.

[3] K. Regmi and A. Borji. Cross-view image synthesis using conditional gans. *CoRR*, abs/1803.03396, 2018.

[4] K. Regmi and A. Borji. Cross-view image synthesis using geometry-guided conditional gans. *CoRR*, abs/1808.05469, 2018.

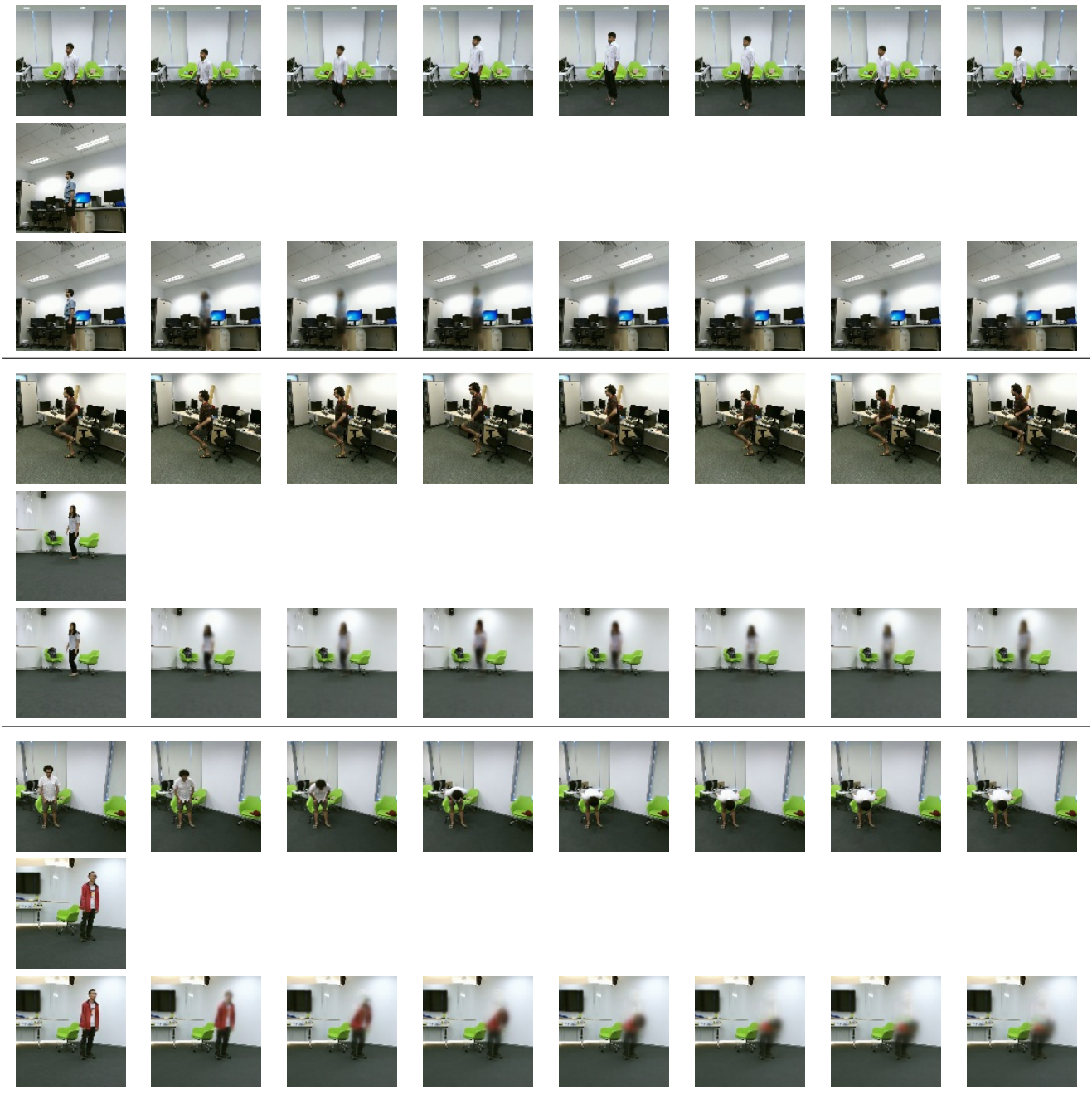

Figure 4. This displays the results for experiements run with different actors and different scenes throughout. There are three different blocks for three different samples on which the model ran. In each, the top row contains 8 frames of the input video, the middle row contains the single input image of apppearance conditioning from the desired view, and the bottom row contains 8 frames of the recontructed video. For each of these, frames 1, 3, 5, 7, 9, 11, 13, and 15 are used.

[5] A. Shahroudy, J. Liu, T. Ng, and G. Wang. NTU RGB+D: A large scale dataset for 3d human activity analysis. *CoRR*, abs/1604.02808, 2016.

[6] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe. Animating arbitrary objects via deep motion transfer. *CoRR*, abs/1812.08861, 2018.