Cross-View Image Synthesis using Conditional GANs

Krishna Regmi and Ali Borji
Center for Research in Computer Vision, University of Central Florida
krishna.regmi7@gmail.com, aborji@crcv.ucf.edu

Abstract

Learning to generate natural scenes has always been a challenging task in computer vision. It is even more painstaking when the generation is conditioned on images with drastically different views. This is mainly because understanding, corresponding, and transforming appearance and semantic information across the views is not trivial. In this paper, we attempt to solve the novel problem of cross-view image synthesis, aerial to street-view and vice versa, using conditional generative adversarial networks (cGAN). Two new architectures called Crossview Fork (X-Fork) and Crossview Sequential (X-Seq) are proposed to generate scenes with resolutions of 64×64 and 256×256 pixels. X-Fork architecture has a single discriminator and a single generator. The generator hallucinates both the image and its semantic segmentation in the target view. X-Seq architecture utilizes two cGANs. The first one generates the target image which is subsequently fed to the second cGAN for generating its corresponding semantic segmentation map. The feedback from the second cGAN helps the first cGAN generate sharper images. Both of our proposed architectures learn to generate natural images as well as their semantic segmentation maps. The proposed methods show that they are able to capture and maintain the true semantics of objects in source and target views better than the traditional image-to-image translation method which considers only the visual appearance of the scene. Extensive qualitative and quantitative evaluations support the effectiveness of our frameworks, compared to two state of the art methods, for natural scene generation across drastically different views.

1. Introduction

In this work, we address the problem of synthesizing ground-level images from overhead imagery and vice versa using conditional Generative Adversarial Networks [20]. Primarily, such models try to generate new images from conditioning variables as input. Preliminary works in GANs utilize unsupervised learning to generate samples from latent representations or from a random noise vector [9].

View synthesis is a long-standing problem in computer vision. This task is more challenging when views are drastically different, fields of views have little or no overlap, and objects are occluded. Furthermore, two objects that are similar in one view may look quite different in another (i.e., the view-invariance problem). For example, the aerial view of a building (i.e., the roof) tells very little about the color and design of the building seen from the street-view. The generation process is generally easier when the image contains a single object in a uniform background. In contrast, when the scene contains multiple objects, generating other view becomes much more challenging. This is due to the increase in underlying parameters that contribute to the variations (e.g., occlusions, shadows, etc). An example scenario, addressed here, is generating street-view (a.k.a ground level) image of a location from its aerial (a.k.a overhead) imagery. Figure 1 illustrates some corresponding images in two views.

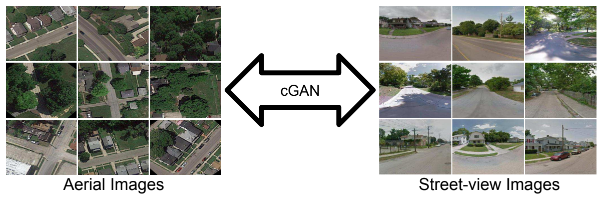


Figure 1: Example images in overhead/aerial view (left) and ground-level/street-view (right). The images reflect the great diversity and richness of features in two views implying that the network needs to learn a lot for meaningful cross-view generation. We propose to use cGANs to solve this problem.

Isola *et al.* [12] put forward a general-purpose framework to solve multiple image translation tasks. Their work translates images of objects or scenes which are represented by RGB images, gradient fields, edge maps, aerial images, sketches, etcetera between these representations. Thus, their method operates on representations in a single view. Formulating our problem as an image translation task between the views, we use their method as a baseline and extend it for cross-view image generation.

Inspired by recent works of [12, 35], we formulate the cross-view image synthesis problem as an image-to-image translation problem and solve it using the conditional generative adversarial network. Previous works in view synthesis [8, 38, 31] have generated images with single objects in them or natural scenes with very little variation in viewing angles between the input and target images. The network learns to copy large parts of image content from the input. Images in each view of our work contain high degree of details and clutter (e.g., trees, cars, roads, buildings, etcetra) along with variations between the corresponding image pairs. Thus, the network needs to learn that the corresponding images in each view need to contain all details and place them in correct positions with proper orientations and inclinations. Perhaps the closest work to ours is the one by Zhai *et al.* [35]. They generate ground-level panorama from aerial imagery by predicting ground image features from aerial image features and use them to synthesize images.

In general, some of the challenges pertaining to cross-view synthesis task are as follows. First, aerial images cover wider regions of the ground than the street-view images whereas street-view images contain more details about objects (e.g., house, road, trees) than aerial images. So, not only the information in aerial images is too noisy, but also less informative for street-view image synthesis. Similarly, a network needs to estimate a lot regions to synthesize aerial images. Second, transient objects like cars (also people) are not present at the corresponding locations in image pairs since they are taken at different times. Third, houses that are different in street-view look similar from aerial view. This causes synthesized street-view images to contain buildings with similar color or texture, prohibiting diversity in generated buildings. Fourth challenge regards variation among roads in two views due to perspective and occlusions. While the road edges are nearly linear and visible in street-view, they are often occluded by dense vegetations and contorted in aerial view. Fifth, when using model generated segmentation maps as ground truth to improve the quality of generated images, as done here, label noise and model errors introduce some artifacts in the results.

To address the above challenges, we propose the following methods. We start with a simple image-to-image translation network of [12] as a baseline. We then propose two new cGAN architectures that generate images as well as segmentation maps in target view. Addition of semantic segmentation generation to the architectures helps improve the generation of images. The first architecture, called X-Fork, is a slight modification of the baseline, forking at the penultimate block to generate two outputs, target view image and segmentation map. The second architecture, called X-Seq, has a sequence of two baseline networks connected. The target view image generated by the first network is fed to the second network to generate its corresponding segmentation map. Once trained, both architectures are able to generate better images than the baseline that learns to generate the images only. This implies that learning to generate segmentation map along with the image indeed improves the quality of generated image.

2. Related Works

2.1. Relating Aerial and Ground-level Images

Zhai *et al.* [35] explored to predict the semantic layout of ground image from its corresponding aerial image. They used the predicted layout to synthesize ground-level panorama. Prior works relating the aerial and ground imageries have addressed problems such as cross-view co-localization [18, 33], ground-to-aerial geo-localization [17] and geo-tagging the cross-view images [34].

Cross-view relations have also been studied between egocentric (first person) and exocentric (surveillance or third-person) domains for different purposes. Human re-identification by matching viewers in top-view and egocentric cameras have been tackled by establishing the correspondences between the views in [1]. Soran *et al.* [29] utilize the information from one egocentric camera and multiple exocentric cameras to solve the action recognition task. Ardeshir *et al.* [2] learn motion features of actions performed in ego- and exocentric domains to transfer motion information across the two domains.

2.2. Learning View Transformations

Existing works on viewpoint transformation have been conducted to synthesize novel views of the same objects [8, 31, 38]. Zhou *et al.* [38] proposed models that learn to copy the pixel information from input view and utilize them to preserve the identity and structure of the objects to generate new views. Tatarchenko *et al.* [31] trained an encode-decoder network to obtain 3D representation models of cars and chairs which they later used to generate different views of an unseen car or chair image. Dosovitskiy *et al.* [8] learned generative models by training on 3D renderings of cars, chairs and tables and synthesized intermediate views and objects by interpolating between views and models.

2.3. GAN and cGAN

Goodfellow *et al.* [9] are the pioneers of Generative Adversarial Networks that is very successful at generating sharp and unblurred images, much better compared to existing methods such as Restricted Boltzmann Machines [10, 28] or deep Boltzmann Machines [25].

Conditional GANs are used to synthesize images conditioned on different parameters during both training and testing. Examples include conditioning on labels of MNIST to generate digits by Mirza *et al.* [20], conditioning on image representations to translate an image between different representations [12], and generating panoramic ground-level scenes from aerial images of the same location[35]. Pathak *et al.* [23] generated missing parts in images (i.e., inpainting) using networks trained jointly with adversarial and reconstruction losses and produced sharp and coherent images. Reed *et al.* [24] synthesized images conditioned on detailed textual descriptions of the objects in the scene, and Zhang *et al.* [36] improved on that by using a two-stage Stacked GAN.

2.4. Cross-Domain Transformations using GANs

Kim *et al.* [13] utilized the GAN networks to learn the relation between images in two different domains such that these learned relations can be transferred between the domains. Similar work by Zhu *et al.* [39] learned mappings between unpaired images using cycle-consistency loss. They assume that a mapping from one domain to the other and back to the first should generate the original image. Both works exploited large unpaired datasets to learn the relation between domains and formulated the mapping task between images in different domains as a generation problem. Zhu *et al.* [39] compare their generation task with previous works on paired datasets by Isola *et al.* [12]. They conclude that the results with paired images is the upper-bound for their unpaired examples.

3. Background on GANs

Generative Adversarial Network architecture [9] consists of two adversarial networks: a generator and a discriminator that are trained simultaneously based on the min-max game theory. The generator G is optimized to map a d-dimensional noise vector (usually d=100) to an image (i.e., synthesizing) that is close to the true data distribution. The discriminator D, on the other hand, is optimized to accurately distinguish between the synthesized images coming from the generator and the real images from the true data distribution. The objective function of such a network is

$$\min_{G}\max_{D} L_{GAN}(G,D) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))], \quad (1)$$

where, x is real data sampled from data distribution p_{data} and z is a d-dimensional noise vector sampled from a Gaussian distribution p_z.

Conditional GANs synthesize images looking into some auxiliary variable which may be labels [20], text embeddings [36, 24] or images [12, 39, 13]. In conditional GANs, both the discriminator and the generator networks receive the conditioning variable represented by c in Eqn. (2). The generator uses this additional information during image synthesis while the discriminator makes its decision by looking at the pair of conditioning variable and the image it receives. Real pair input to the discriminator consists of true image from distribution and its corresponding label while the fake pair consists of synthesized image and the label. For conditional GAN, the objective function is

$$\min_{G}\max_{D} L_{cGAN}(G,D) = E_{x,c\sim p_{data}(x,c)}[logD(x,c)] + E_{x',c\sim p_{data}(x',c)}[log(1-D(x',c))], \quad (2)$$

where $x' = G(z,c)$ is the generated image.

In addition to the GAN loss, previous works (e.g., [12, 39, 23]) have tried to minimize the $L1$ or $L2$ distances between real and generated image pairs. This step aids the generator to synthesize images very similar to the ground truth. Minimizing $L1$ distance generates less blurred images than minimizing the $L2$ distance. That is, using the $L1$ distance increases image sharpness in generation tasks. Therefore, we use the $L1$ distance in our method. The expression to minimize the $L1$ distance is

$$\min_{G} L_{L1}(G) = E_{x,x'\sim p_{data}(x,x')}[||\, x - x'\, ||_1], \quad (3)$$

The objective function for such conditional GAN network is the sum of Eqns. (2) and (3).

Considering the synthesis of the ground level imagery (I_g) from aerial image (I_a), the conditional GAN loss and $L1$ loss are represented as in Eqns. (4) and (5), respectively. For ground to aerial synthesis, the roles of I_a and I_g are reversed.

$$\min_{G}\max_{D} L_{cGAN}(G,D) = E_{I_g,I_a\sim p_{data}(I_g,I_a)}[logD(I_g,I_a)] + E_{I_a,I'_g\sim p_{data}(I_a,I'_g)}[log(1-D(I'_g,I_a))], \quad (4)$$

$$\min_{G} L_{L1}(G) = E_{I_g,I'_g\sim p_{data}(I_g,I'_g)}[||\, I_g - I'_g\, ||_1], \quad (5)$$

where, $I'_g = G(I_a)$. We employ the network of [12] as our baseline architecture. The objective function for the baseline is the sum of conditional GAN loss in Eqn. (4) and $L1$ loss in Eqn. (5), as represented in Eqn. (6):

$$L_{network} = L_{cGAN}(G,D) + \lambda L_{L1}(G), \quad (6)$$

where, λ is the balancing factor between the losses.

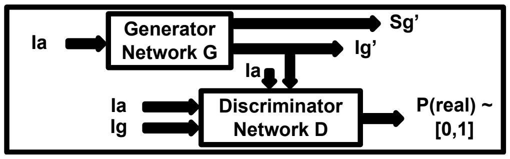
4. Proposed cGAN-based Approaches

In this section, we propose two architectures for the task of cross-view image synthesis.

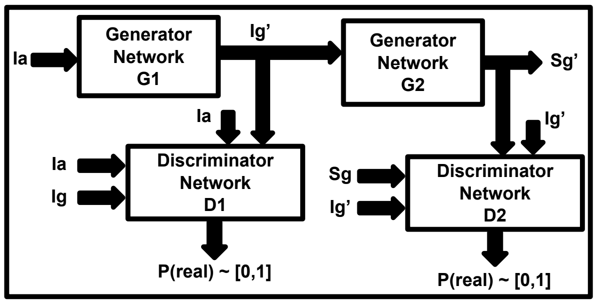
4.1. Crossview Fork (X-Fork)

Our first architecture, known as Crossview Fork, is shown in Figure 2a. The discriminator architecture is taken from the baseline [12] but the generator network is forked to synthesize images as well as segmentation maps. The fork-generator architecture is shown in Figure 3. The first six blocks of decoder share the weights. This is because the image and segmentation map contain a lot of shared features. The number of kernels used in each layer (block) of the generator are shown below the blocks.

Even though the X-Fork architecture generates the cross-view image and its segmentation map, the discriminator receives only the real/fake image pairs but not the segmentation pairs during the training. In other words, the generated segmentation map serves as an auxiliary output. Notice that here we are primarily interested in generating higher quality images rather than the segmentation maps. Thus, the conditional GAN loss for this network is still the same as in Eqn. (4). To use the segmentation information, in addition to the $L1$ distance between the generated image and the real image, we also include the $L1$ distance between the ground-truth segmentation and the generated segmentation map into the loss.



(a) X-Fork architecture.



(b) X-Seq architecture.

Figure 2: Our proposed network architectures. a) X-Fork: Similar to baseline architecture except that G forks to synthesize image and segmentation map in target view, and b) X-Seq: a sequence of two cGANs, G1 synthesizes target view image that is used by G2 for segmentation map synthesis in corresponding view. In both architectures, I_a and I_g are real images in aerial and ground views, respectively. S_g is the ground-truth segmentation map in street-view obtained using pre-trained RefineNet [16]. I'_g and S'_g are synthesized image and segmentation map in ground view.

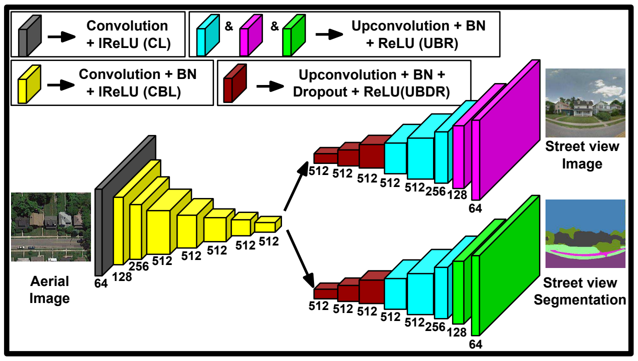


Figure 3: Generator of X-Fork architecture in Figure 2a. BN means batch-normalization layer. The first six blocks of decoder share weights, forking at the penultimate block. The number of channels in each convolution layer are shown below each blocks.

4.2. Crossview Sequential (X-Seq)

Our second architecture uses a sequence of two cGAN networks as shown in Figure 2b. The first network generates cross-view images similar to the baseline. The second network receives images from the first generator as conditioning input to synthesize the segmentation map in the same view. Thus, the first network is a cross-view cGAN while the second one is an image-to-segmentation cGAN. The whole architecture is trained end-to-end so that both cGANs learn simultaneously. Intuitively, the input-output dependency between the cGANs constrains the generated images and the segmentation maps, and in effect improves the quality of the generated outputs. Training the first network to generate better cross-view images enhances generation of better segmentation maps by the second generator. At the same time, the feedback from the better trained second network forces the first network to improve its generation. Thus, when both networks are trained in tandem, better quality images are generated compared to the baseline.

Replacing G and D in Eqns. (4) and (5) by G_1 and D_1, respectively, we obtain the equivalent expressions for losses of cross-view cGAN network in this architecture. For the image-to-segmentation cGAN network, the images generated by G_1 are considered as conditioning inputs. We now express the cGAN loss for this network as

$$\min_{G_2} \max_{D_2} L_{cGAN}(G_2, D_2) = E_{I'_g, S_g \sim p_{data}(I'_g, S_g)}[log D_2(S_g, I'_g)] + E_{S'_g, I'_g \sim p_{data}(S'_g, I'_g)}[log(1 - D_2(S'_g, I'_g))], \tag{7}$$

where, $I'_g = G_1(I_a)$ and $S'_g = G_2(I'_g)$. The $L1$ loss for the image-to-segmentation network is

$$\min_{G_2} L_{L1}(G_2) = E_{S_g, S'_g \sim p_{data}(S_g, S'_g)}[|| \, S_g - S'_g) \, ||_1], \tag{8}$$

The overall objective function for the X-Seq network is

$$L_{X-Seq} = L_{cGAN}(G_1, D_1) + \lambda L_{L1}(G_1) + L_{cGAN}(G_2, D_2) + \lambda L_{L1}(G_2). \tag{9}$$

Eqn. (9) is optimized during the training to learn the parameters G_1, D_1, G_2 and D_2.

5. Experimental Setting

5.1. Dataset

For the experiments in this work, we use the cross-view image dataset provided by Vo *et al.* [33]. This dataset consists of more than one million pairs of street-view and overhead view images collected from 11 different cities in the US. We select 76,048 image pairs from Dayton and create a train/test split of 55,000/21,048 pairs. We call it Dayton Dataset. The images in the original dataset have resolution of 354×354. We resize them to 256×256. Some example images are shown in Figure 1.

We also recruit the CVUSA dataset [34] for direct comparison of our work with Zhai *et al.* [35]. This dataset consists of 35,532/8,884 train/test split of image pairs. Following Zhai *et al.*, the aerial images are center-cropped to 224 × 224 and then resized to 256 × 256. We only generate a single camera-angle image rather than the panorama. To do so, we take the first quarter of the ground level images and segmentations from the dataset and resize them to 256 × 256 in our experiments. Please see Figure 7 for some images from the CVUSA dataset.

The two networks, X-Fork and X-Seq, learn to generate the target view images and segmentation maps conditioned on source view image. Training procedure requires the images as well as their semantic segmentation maps.

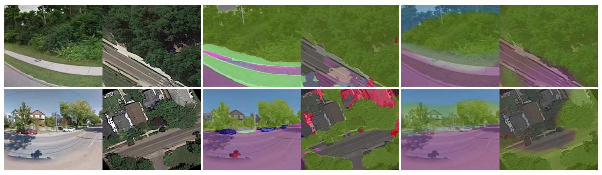


Figure 4: Original image pairs from training set (left), images with segmentation masks from pre-trained RefineNet [16] overlaid on original images (middle) and images with segmentation masks generated by X-Fork network overlaid on original images (right).

The CVUSA dataset has annotated segmentation maps for ground view images, but for Dayton dataset such information is not available. To compensate, we use one of the leading semantic segmentation methods, known as the RefineNet [16]. This network is pre-trained on outdoor scenes of the Cityscapes dataset [6] and is used to generate the segmentation maps that are utilized as ground truth maps. These semantic maps have pixel labels from 20 classes (e.g., road, sidewalk, building, vegetation, sky, void, etc). Figure 4 shows image pairs from the dataset and their segmentation masks overlaid in both views. As can be seen, the segmentation mask (label) generation process is far from perfect since it is unable to segment parts of buildings, roads, cars, etcetera in images.

5.2. Implementation Details

We use the conditional GAN architecture of [12] as the baseline and call it Pix2pix. The generator is an encoder-decoder network with blocks of Convolution, Batch Normalization [11] and activation layers. Leaky ReLU with a slope of 0.2 is used as the activation function in the encoder, whereas the decoder has ReLU activation except for its final layer where Tanh is used. The first three blocks of the decoder have a Dropout layer in between Batch normalization and activation layer, with dropout rate of 50%. The discriminator is similar to the encoder of the generator. The only difference is that the final layer uses sigmoid non-linearity that gives the probability of its input being real.

The used convolutional kernels are 4×4 with a stride of 2. The upconvolution in the decoder is Torch[5] implementation of $SpatialFullConvolution$, and upsamples the input by 2. For the encoder and the discriminator, convolutional operation downsamples the images by 2. No pooling operation is used in the networks. The λ used in Eqns. (6) and (9) is the balancing factor between the GAN loss and $L1$ loss. Its value is fixed at 100. Following the idea to smooth the labels by [30] and demonstration of its effectiveness by Salimans *et al.* [26], we use one-sided label smoothing to stabilize the training process, replacing 1 by 0.9 for real labels. During the training, we utilized different data augmentation methods like random jitter and horizontal flipping of images. The network is trained end-to-end with weights initialized with a random Gaussian distribution with zero mean and 0.02 standard deviation. It is implemented in Torch [5].

6. Results

Our experiments are conducted in **a2g** (aerial-to-ground) and **g2a** (ground-to-aerial) directions on Dayton dataset and **a2g** direction only on CVUSA dataset. We consider image resolutions of 64×64 and 256×256 on Dayton dataset while for experiments on CVUSA dataset, 256×256 resolution images are used.

First, we run experiments on lower resolution images (64×64) for proof of concept. Encouraging qualitative and quantitative results in this resolution motivated us to apply our methods to higher resolution (256×256) images. The lower resolution experiments are carried out for 100 epochs with batch size of 16, whereas the higher resolution experiments are conducted for 35 epochs with batch size of 4.

We conduct experiments on CVUSA dataset for comparison with Zhai *et al.*'s work [35]. Following their setup, we train our architectures for 30 epochs, using the Adam optimizer and moment parameters $\beta1$ = 0.5 and $\beta2$ = 0.999.

It is not straightforward to evaluate the quality of synthesized images [3]. In fact, evaluation of GAN methods continues to be an open problem [32]. A common evaluation method is to show the generated images to human observers and ask their opinion about the images. Human judgment is based on the response to the question: Is this image (image-pair) real or fake? Alternatively, the images generated by different generative models can be pitted against each other and the observer is asked to select the image that looks more real. But in experiments involving natural scenes, such evaluation methods are more challenging as multiple factors often affect the quality of the generated images. For example, the observer may not be sure whether to base his judgment on better visual quality, higher sharpness at object boundaries, or more semantic information present in the image (e.g., multiple objects in the images, more details on objects, etc). Therefore, instead of behavioral experiments, we illustrate qualitative results in Figures 5, 6 and 7 and conduct an in-depth quantitative evaluation on test images of two datasets.

6.1. Qualitative Evaluation

For 64×64 resolution experiments, the networks are modified by removing the last two blocks of CBL from discriminator and encoder, and the first two blocks of UBDR from decoder of the generator. We run experiments on all three methods. Qualitative results are depicted in Figure 5. The results affirm that the networks have learned to transfer the image representations across the views. Generated ground level images clearly show details about road, trees, sky, clouds, and pedestrian lanes. Trees, grass, road, house roofs are well rendered in the synthesized aerial images.

For 256×256 resolution synthesis, we conduct experiments on all three architectures and illustrate the qualitative results on Dayton and CVUSA datasets in Figures 6 and 7

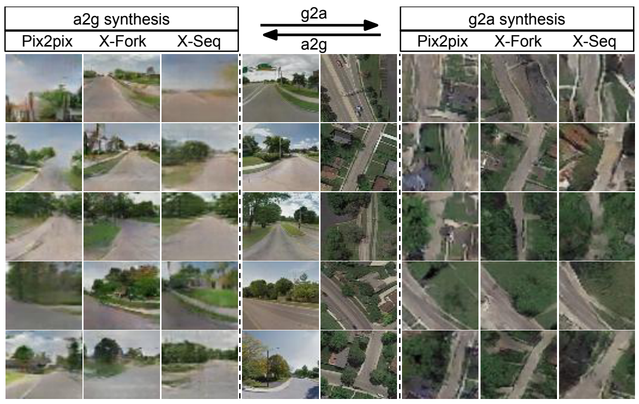


Figure 5: Example images generated by different methods in lower (64 × 64) resolution in **a2g** and **g2a** directions.

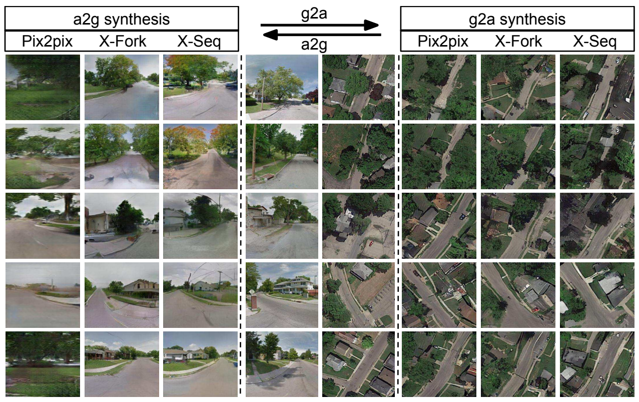


Figure 6: Example images generated by different methods in higher (256 × 256) resolution in **a2g** and **g2a** directions.

respectively. For Dayton dataset, we observe that the images generated in higher resolution contain more details of objects in both views and are less granulated than those in lower resolution. Houses, trees, pedestrian lanes, and roads look more natural. Test results on CVUSA dataset show that images generated by proposed methods are visually better compared to Zhai *et al.* [35] and Pix2pix [12] methods.

6.2. Quantitative Evaluation

The quantitative results of our experiments on both datasets are presented in Tables 1-4. **64×64** and **256×256** in column headers of the tables refer to results obtained for two resolutions of Dayton dataset. Next, we discuss the quantitative measures used to evaluate our methods.

6.2.1 Inception Score

A common quantitative GAN evaluation measure is the *Inception Score* [26]. The core idea behind the inception score is to assess how diverse the generated samples are within a class while being meaningfully representative of the class at the same time.

$$InceptionScore, I = exp(E_x D_{KL}(p(y|x)||p(y))), \quad (10)$$

where, x is a generated sample and y is its predicted label.

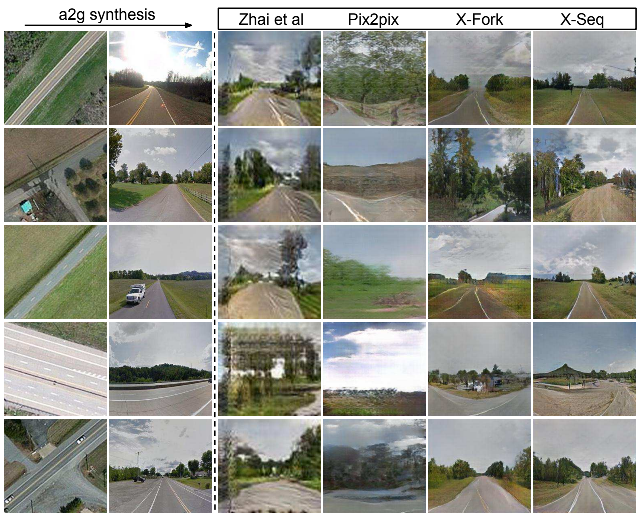


Figure 7: Qualitative results of our methods and baselines on CVUSA dataset in **a2g** direction. First two columns show true image pairs, next four columns show images generated by Zhai *et al.* [35], Pix2pix[12], X-Fork and X-Seq methods, respectively.

We can not use the Inception model because the datasets that we use include natural outdoor images that do not fit into ImageNet classes [7]. To solve this, we use the AlexNet model [14] trained on Places dataset [37] with 365 categories to compute the inception score. The Places dataset has images similar to those in our datasets. The scores are reported in Table 1. The scores for X-Fork generated images are closest to that of real data distribution for Dayton dataset in lower resolution in both directions and also in higher resolution in **a2g** direction. The X-Seq method works best for CVUSA dataset and for **g2a** synthesis in higher resolution over Dayton dataset.

We observe that the confidence scores predicted by the pre-trained model on our dataset are dispersed between classes for many samples and not all the categories are represented by the images. Therefore, we compute inception scores on Top-1 and Top-5 classes, where ”Top-k” means that top k predictions for each image are unchanged while the remaining predictions are smoothed by an epsilon equal to (1 - $\sum$(top-k predictions))/(n-k classes). Results on top-k classes follow a similar pattern as in all classes (except for Top-1 class on lower resolution in **g2a** over Dayton dataset).

In addition to inception score, we compute the top-k prediction accuracy between real and generated images. We use the same pre-trained Alexnet model to obtain annotations for real images and class predictions for generated images. We compute top-1 and top-5 accuracies. Results are shown in Table 2. For each setting, accuracies are computed in two ways: 1) considering all images, and 2) considering real images whose top-1 (highest) prediction is greater than 0.5. Below each accuracy heading, the first column considers all images whereas the second column computes accu-

Table 1: KL divergence scores between conditional and marginal probabilities (Inception Score).

Dir.	Methods	64×64			256×256			CVUSA		
⇄		all classes	Top-1 class	Top-5 classes	all classes	Top-1 class	Top-5 classes	all classes	Top-1 class	Top-5 classes
a2g	Zhai *et al.* [35]	–	–	–	–	–	–	1.8434	1.5171	1.8666
	Pix2pix [12]	1.8029	1.5014	1.9300	2.8515	1.9342	2.9083	3.2771	2.2219	3.4312
	X-Fork	**1.9600**	**1.5908**	**2.0348**	**3.0720**	**2.2402**	**3.0932**	3.4432	2.5447	3.5567
	X-Seq	1.8503	1.4850	1.9623	2.7384	2.1304	2.7674	**3.8151**	**2.6738**	**4.0077**
	Real Data	2.2096	1.6961	2.3008	3.7090	2.5590	3.7900	4.9971	3.4122	5.1150
g2a	Pix2pix [12]	1.7970	1.3029	1.6101	3.5676	2.0325	2.8141	–	–	–
	X-Fork	**1.8557**	1.3162	**1.6521**	3.1342	1.8656	2.5599	–	–	–
	X-Seq	1.7854	**1.3189**	1.6219	**3.5849**	**2.0489**	**2.8414**	–	–	–
	Real Data	2.1408	1.4302	1.8606	3.8979	2.3146	3.1682	–	–	–

Table 2: Accuracies: Top-1 and Top-5.

Dir.	Methods	64×64				256×256				CVUSA			
⇄		Top-1 Accuracy (%)		Top-5 Accuracy (%)		Top-1 Accuracy (%)		Top-5 Accuracy (%)		Top-1 Accuracy (%)		Top-5 Accuracy (%)	
a2g	Zhai *et al.* [35]	–	–	–	–	–	–	–	–	13.97	14.03	42.09	52.29
	Pix2pix [12]	7.90	15.33	27.61	39.07	6.8	9.15	23.55	27.00	7.33	9.25	25.81	32.67
	X-Fork	**16.63**	**34.73**	**46.35**	**70.01**	30.00	48.68	61.57	78.84	**20.58**	**31.24**	**50.51**	**63.66**
	X-Seq	4.83	5.56	19.55	24.96	**30.16**	**49.85**	**62.59**	**80.70**	15.98	24.14	42.91	54.41
g2a	Pix2pix [12]	1.65	2.24	7.49	12.68	10.23	16.02	30.90	40.49	–	–	–	–
	X-Fork	**4.00**	**16.41**	**15.42**	**35.82**	10.54	15.29	30.76	37.32	–	–	–	–
	X-Seq	1.55	2.99	6.27	8.96	**12.30**	**19.62**	**35.95**	**45.94**	–	–	–	–

Table 3: KL Divergence between model and data distributions.

Dir.	Method	64×64	256×256	CVUSA
a2g	Zhai *et al.* [35]	–	–	27.43 ± 1.63
	Pix2pix [12]	6.29 ± 0.8	38.26 ± 1.88	59.81 ± 2.12
	X-Fork	**3.42 ± 0.72**	6.00 ± 1.28	**11.71 ± 1.55**
	X-Seq	6.22 ± 0.87	**5.93 ± 1.32**	15.52 ± 1.73
g2a	Pix2pix [12]	6.39 ± 0.90	7.88 ± 1.24	–
	X-Fork	**4.45 ± 0.84**	**6.92 ± 1.15**	–
	X-Seq	7.20 ± 0.92	7.07 ± 1.19	–

racies the second way. For lower resolution images on Dayton dataset and for experiments on CVUSA dataset, X-Fork method outperforms the remaining methods. For higher resolution images, our methods show dramatic improvements over Pix2pix in the **a2g** direction, whereas X-Seq works best in the **g2a** direction.

6.2.2 KL(model || data)

We next compute the KL divergence between the model generated images and the real data distribution for quantitative analysis of our work, similar to some generative works [4, 21]. We again use the same pre-trained Alexnet as in the previous subsection. The lower KL score implies that the generated samples are closer to the real data distribution. The scores are provided in Table 3. As it can be seen, our proposed methods generate much better results than existing generative methods on both datasets. X-Fork generates images very similar to real distribution in all experiments except on the higher resolution **a2g** experiment where X-Seq is slightly better than X-Fork.

6.2.3 SSIM, PSNR and Sharpness Difference

As in some generative works [19, 15, 27, 22], we also employ Structural-Similarity (SSIM), Peak Signal-to-Noise Ratio (PSNR) and Sharpness Difference measures to evaluate our methods.

SSIM measures the similarity between the images based on their luminance, contrast and structural aspects. SSIM value ranges between -1 and +1. A higher value means greater similarity between the images being compared. It is computed as

$$SSIM(I_g, I'_g) = \frac{(2\mu_{I_g}\mu_{I'_g} + c_1)(2\sigma_{I_g I'_g} + c_2)}{(\mu^2_{I_g} + \mu^2_{I'_g} + c_1)(\sigma^2_{I_g} + \sigma^2_{I'_g} + c_2)} \tag{11}$$

PSNR measures the peak signal-to-noise ratio between two images to assess the quality of a transformed (generated) image compared to its original version. The higher the PSNR, the better is the quality of generated image. It is computed as

$$PSNR(I_g, I'_g) = 10log_{10}(\frac{max^2_{I'_g}}{mse}) \tag{12}$$

where, $mse(I_g, I'_g) = \frac{1}{n}\sum_{i=1}^{n}(I_g[i] - I'_g[i])^2$, and $max_{I_{g'}}$ = 255 (maximum pixel intensity value).

Table 4: SSIM, PSNR and Sharpness Difference between real data and samples generated using different methods.

Dir.	**Methods**	**64×64**			**256×256**			**CVUSA**		
⇄		SSIM	PSNR	Sharp Diff	SSIM	PSNR	Sharp Diff	SSIM	PSNR	Sharp Diff
a2g	Zhai *et al.* [35]	–	–	–	–	–	–	0.4147	17.4886	16.6184
	Pix2pix [12]	0.4808	19.4919	16.4489	0.4180	17.6291	19.2821	0.3923	17.6578	18.5239
	X-Fork	0.4921	19.6273	16.4928	0.4963	19.8928	19.4533	**0.4356**	**19.0509**	**18.6706**
	X-Seq	**0.5171**	**20.1049**	**16.6836**	**0.5031**	**20.2803**	**19.5258**	0.4231	18.8067	18.4378
g2a	Pix2pix [12]	0.3675	20.5135	14.7813	0.2693	20.2177	16.9477	–	–	–
	X-Fork	**0.3682**	**20.6933**	**14.7984**	**0.2763**	**20.5978**	**16.9962**	–	–	–
	X-Seq	0.3663	20.4239	14.7657	0.2725	20.2925	16.9285	–	–	–

Sharpness difference measures the loss of sharpness during image generation. To compute the sharpness difference between the generated image and the true image, we follow [19] and compute the difference of gradients between the images as

$$SharpDiff.(I_g, I'_g) = 10log_{10}(\frac{max^2_{I'_g}}{grads}), \quad (13)$$

where, $grads = \frac{1}{N}\sum_i \sum_j (|(\nabla_i I_g + \nabla_j I_g)\text{-}(\nabla_i I'_g + \nabla_j I'_g)|)$ and, $\nabla_i I = |I_{i,j} - I_{i-1,j}|$, $\nabla_j I = |I_{i,j} - I_{i,j-1}|$.

Sharpness difference in Eqn. (13) is inverse of $grads$. We would like the $grads$ to be small, so the higher the overall score the better.

The scores are reported in Table 4. Over Dayton dataset, X-Seq model works the best in **a2g** direction while X-Fork outperforms the rest in the **g2a** direction. On CVUSA, X-Fork improves over Zhai *et al.* by 5.03% in SSIM, 8.93% in PSNR, and 12.35% in Sharpness difference.

Because there is no consensus in evaluation of GANs, we had to use several scores. Theis *et al.* [32] show that these scores often do not agree with each other and this was observed in our evaluations as well. So, it is difficult to infer whether X-Fork or X-Seq is better. We find that the proposed methods are consistently superior to the baselines.

6.3. Generated Segmentation Maps

Our methods generate semantic segmentation maps along with the real images in cross-view. The overlay of segmentation maps generated by X-Fork network (pretrained RefineNet) on Dayton images are presented in the last (second) column of Figure 4. Please see supplementary materials for more qualitative results on semantic segmentation. The overlaid images show that the network is able to learn the semantic representations of object classes. For quantitative analysis, segmentation maps generated by our methods are compared against the segmentation maps obtained by applying RefineNet [16] to the target images. We compute per-class accuracies and mean IOU for the most common classes in our datasets: 'vegetation', 'road' and 'building' in aerial segmentation maps plus the 'sky' in ground segmentations. The scores are reported in Table 5. Even though X-Fork does better than X-Seq, we find that both methods achieve good scores for segmentation.

Table 5: Evaluation Scores for segmentation maps.

Methods	**a2g**		**g2a**	
	Per-class acc.	**mIOU**	**Per-class acc.**	**mIOU**
X-Fork	**0.6262**	**0.4163**	**0.5473**	**0.2157**
X-Seq	0.4783	0.3187	0.4990	0.2139

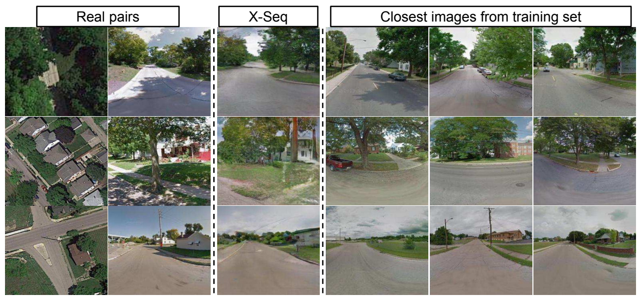


Figure 8: Along the columns, we show real image pairs, corresponding street-view image synthesized by X-Seq method and three nearest images in the training set retrieved by computing $L1$ distance between generated image and training set images.

6.4. kNN

Here, we test whether our proposed architectures have actually learned the representations between images in two views rather than just memorizing the blocks from training images to generate new ones. For this, we pick three images from the training set that are closest to the generated images in terms of $L1$ distance. As shown in Figure 8, the generated images have subtle differences with the training set images implying that our network has indeed learned important semantic representations in input view needed to transform the source image to target view.

7. Discussion and Conclusion

We explored image generation using conditional GANs between two drastically different views. Generating semantic segmentations together with images in target view helps the networks learn better images compared to the baselines. Extensive qualitative and quantitative evaluations testify the effectiveness of our methods. Using higher resolution images provided significant improvements in visual quality and added more details to synthesized images. The challenging nature of the problem leaves room for further improvements. Code and data will be shared.

Acknowledgement. We would like to thank NVIDIA for the donation of the Titan-X GPUs used in this work.

References

[1] S. Ardeshir and A. Borji. Ego2top: Matching viewers in egocentric and top-view videos. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, pages 253–268, 2016. 2

[2] S. Ardeshir, K. Regmi, and A. Borji. Egotransfer: Transferring motion across egocentric and exocentric domains using deep neural networks. *CoRR*, abs/1612.05836, 2016. 2

[3] A. Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018. 5

[4] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. *arXiv e-prints*, abs/1612.02136, 2016. 7

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 5

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 6

[8] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):692–705, Apr 2017. 1, 2

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1, 2, 3

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. 2

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. 5

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3, 5, 6, 7, 8

[13] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1857–1865, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. 3

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017. 6

[15] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 105–114, 2017. 7

[16] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, July 2017. 4, 5, 8

[17] T. Lin, Y. Cui, S. J. Belongie, and J. Hays. Learning deep representations for ground-to-aerial geolocalization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5007–5015, 2015. 2

[18] T.-Y. Lin, S. Belongie, and J. Hays. Cross-view image geolocalization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2

[19] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015. 7, 8

[20] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 1, 2, 3

[21] T. D. Nguyen, T. Le, H. Vu, and D. Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NIPS)*, page accepted, 2017. 7

[22] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. *CoRR*, abs/1703.02921, 2017. 7

[23] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 2, 3

[24] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR. 2, 3

[25] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 1, page 3, 2009. 2

[26] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016. 5, 6

[27] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1874–1883, 2016. 7

[28] P. Smolensky. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, pages 194–281. MIT Press, Cambridge, MA, USA, 1986. 2

[29] B. Soran, A. Farhadi, and L. G. Shapiro. Action recognition in the presence of one egocentric and multiple static cameras. In D. Cremers, I. D. Reid, H. Saito, and M. Yang, editors, *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part V*, volume 9007 of *Lecture Notes in Computer Science*, pages 178–193. Springer, 2014. 2

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826, 2016. 5

[31] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 322–337, Cham, 2016. Springer International Publishing. 1, 2

[32] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. 5, 8

[33] N. N. Vo and J. Hays. *Localizing and Orienting Street Views Using Overhead Imagery*, pages 494–509. Springer International Publishing, Cham, 2016. 2, 4

[34] S. Workman, R. Souvenir, and N. Jacobs. Wide-area image geolocalization with aerial reference imagery. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 4

[35] M. Zhai, Z. Bessinger, S. Workman, and N. Jacobs. Predicting ground-level scene layout from aerial imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4, 5, 6, 7, 8

[36] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017. 2, 3

[37] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 6

[38] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 286–301, Cham, 2016. Springer International Publishing. 1, 2

[39] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 3